

Dee Hwa Liong Academy Grade Management System

Nico C. Mendoza

Department of Physical Sciences and Mathematics

University of the Philippines Manila

Adviser: Gregorio B. Baes, Ph.D.

Abstract

Dee Hwa Liong Academy currently uses spreadsheet files to keep student records. In result, the academy faces several difficulties, which include tedious process for storing student records, long preparation time, and lack of security. Dee Hwa Liong Grade Management System will lessen the time needed for grade submission and deliberation, and it will also provide security features for the class records. The system will also eradicate the need of using flash drives and facebook messenger for submission of grades. Tracking late submissions of grades will also be easier since there will be deadline feature which will be set for each teacher in the system. The application will also provide security for students' records since grades will not be easily modified and an edit log will also be available. In addition, this application will also keep parents up to date with their child's performance.

Keywords: information system, web-based application, k to 12, grade management system

Contents

Abstract	2
I. Introduction	1
A. Background of the Study	1
B. Statement of the Problem	2
C. Objectives of the Study	3
D. Significance of the Project	6
E. Scope and Limitations	7
F. Assumptions	7
II. Review of Related Literature	8
III. Theoretical Framework	12
A. Assessment of Students	12
1. Nursery and Kinder1	12
2. Kinder2, Grades 1 to 10	12
3. Grades 11 to 12	13
B. Teacher-centric Grade Sheet	13
C. Condensed Grade	13
D. Section-centric Grade Sheet	13
E. Student-centric Grade Sheet	14
F. Grade Submission	14
G. Individual Deliberation of Grades	14
H. Group Deliberation of Grades	14
I. Printing of Report Cards	15
J. Node.js	15
K. Express.js	15
L. React.js	15
M. MySQL	16
IV. Design and Implementation	17
A. Context Diagram	17

B.	Data Flow Diagram	18
C.	Activity Diagrams	21
D.	Database Design	24
1.	Entity Relationship Diagram (ERD)	24
E.	Data Dictionary	32
1.	User Account	32
2.	Student	35
3.	Parent/Guardian	36
4.	Cashier	37
5.	Teacher	38
6.	Nonacademic	38
7.	Section	39
8.	Subject	39
9.	Grade Level	40
10.	Category	41
11.	Subcategory	42
12.	Class record	42
13.	Grade	43
14.	Attendance Log	45
15.	Advisory Table	46
16.	Class	47
17.	Submission Deadline	48
18.	Subcomponent Weight	48
F.	System Architecture	50
G.	Technical Architecture	50
V.	Bibliography	643

I. Introduction

A. Background of the Study

The Philippines recently implemented a comprehensive reform on its basic education known as the K to 12 program [1]. The K to 12 program encompasses kindergarten and 12 years of basic education - six years primary education, four years Junior High School and two years Senior High School [2]. With this program, the Philippines is slowly matching the global standards in secondary education. The main objectives of the program are to better prepare students for higher education, to gain eligibility for domestic and overseas educational institutions, and to provide immediate employability upon graduating [1].

With the new educational program, a new curriculum was introduced together with its subjects. The kindergarten curriculum framework applies the goals of the k to 12 program and implements the general principles of the National Early Learning Framework. Students in grades 1 to 10 will encounter an improved, context-based, and spiral progression learning curriculum with several subjects. On the other hand, Senior High School (SHS) is two years of secondary education with specialization. A student will choose a career track - Academic, Technical-Vocational-Livelihood, and Sports and Arts. The chosen career track will define a student's subjects [1].

The new program with new curriculum will also include changes in the grading system. The Department of Education (DepEd) provided teachers a free copy of Electronics Class Record (ECR) Templates [3]. The templates provide grade computation consistent with DepEd Order No. 8, s. 2015, referred to as the Policy Guidelines on Classroom Assessment for the K to 12 Basic Education Program.

ECR templates are MS Excel based. It has three (3) components for grades: Written Work, Performance Tasks, and Quarterly Assessment. By downloading and comparing the different ECR templates, the templates differ at least each grade level. The ECR also changes when Senior High students are being handled. Some of the changes include following a different set of weights for each component compared to grades 1 to 10.

Class records are one of the most important things kept by teachers. This holds all the class performance of the students a teacher handles [4]. At present creating an online or web-based class record system is possible. Teachers and school administrators

are also updated with technology [4]. They can use laptops, computers and even phones easily. Keeping class records or records, in general, is commonly done with the use of spreadsheet applications [4]. Even though spreadsheets have proven to be a useful tool for keeping records, creating an application with better management capabilities is possible.

B. Statement of the Problem

Dee Hwa Liong Academy is an educational institution that implements the K to 12 program. This academy uses spreadsheet to keep student records. In result, the academy faces several difficulties, which include tedious process for storing student records, long preparation time, and lack of security.

The academy suffers long preparation time for creating spreadsheet files for each teacher. These excel files are called Electronic Class Record (ECR), an excel file provided by DepEd which includes lists of student's grades in each component (Written Work, Performance Task, and Quarterly Assessment) per subject. Before the classes start, the Information Technology (IT) head gets all enrolled students, their corresponding section, and the subject load of each teacher from the registrar's office. The IT head manually generates a teacher-centric spreadsheet file from these data for each teacher, adding the teacher's current load with corresponding students enrolled in that subject. The weight of each component is also manually changed depending on the guidelines that was given by DepEd. The load of each teacher can vary, thus spreadsheet files contain multiple class records. The said preparation will be repeated twice a year for Senior Highschool teachers and once a year for non-SHS subject teachers.

After a five-day examination period, the teachers are given a two-week time window to fill up the scores of their handling students for each subject. After making sure that a teacher has already submitted all their completed class records, an individual deliberation is taken place. This deliberation process includes a face-to-face meeting with the IT head and registrar, running down all the student scores, and checking for possible incorrect or missing fields. Late submission of grades can cause a delay of the group deliberation. One of the difficulties that the IT head and registrar also encounter is the lack of security in the spreadsheet files. The submission of grades is usually done through flash drive or via facebook messenger. In effect, these excel files can easily be duplicated (losing track

of the most updated file) and a possible modification can be done without the permission of certain authorities. Moreover, these records are also prone to data corruption.

After checking every class records of all teachers, the IT head then manually generates section-centric spreadsheet files which contain lists of students under a given subject. Each file also includes the condensed grade of each student as well as their grade for each corresponding subjects. This process involves linking multiple teacher-centric class records to a single spreadsheet file.

Generating student-centric report cards can be a tedious process most especially for SHS students. The problem arose when the academy's population started to increase, thus also increases the number of students with backlog subjects. The IT head has to personalize the report cards for irregular students, since they have a different set of currently taking subjects.

C. Objectives of the Study

The main goal of the study is to develop a web-based application for record keeping of students' grades with the following functionalities:

1. The application will have a **System Administrator** who will be able to:
 - (a) Login and logout
 - (b) Change administrator password
 - (c) View and update profile
 - (d) Create user accounts
 - (e) Delete an account
 - i. Account deletion will only happen if the owner of the account is no longer affiliated with the school or is suspended from work and other reasons for deactivation.
 - (f) View the edit/update log of the class records
 - i. This is a read-only log. Its sole purpose is to keep track of all the changes happening in the class records submitted.
2. The application will have a **Director**'s account. The director will be able to:

- (a) Login and logout
- (b) View and update profile
- (c) Change his/her password
- (d) View grades from each subjects
 - i. These class records will also provide student information, name, student number, and section. The name of the teacher will also be included. The administrator has also the option to view the records in teacher-centric, section-centric, or student-centric
- (e) View condensed grades of each section
 - i. Condensed grades will not only show grades. It will also give information about the teachers assigned to each subject and the adviser of the section being viewed.
- (f) View list and number of students who passed/failed
- (g) View list and number of honor students

3. A **Registrar**'s account will be able to do the following functionalities:

- (a) Login and logout
- (b) View and update profile
- (c) Change his/her password
- (d) Create a school year and update the current quarter
- (e) Add/Update the names of sections
- (f) Add students to a section
- (g) Add subject load to teachers
- (h) Set deadline for submission of grades
 - i. Deadline for submission of grades will be set to be able to see who submitted late since a fine is to be collected for every day, after the due date, without submission. Deadline alerts will be automatically sent by the system, during 5 days and 3 days before the deadline and during the deadline day. Different deadline for teachers is possible especially if one teacher has more load compared to others.

- (i) View student records
 - i. The student record view can be teacher-centric, section-centric, and student-centric
- (j) View and produce report cards
- (k) View school information, Elementary Learners Data, Elementary Learners Age Profile, Junior High School Learners Data, JHS Learners Age Profile, Senior High School Repeaters Age Profile, SHS Learners Data by Track, SHS Learners Data in Technical-Vocational-Livelihood Track Specializations, Total Number of Enrollees, Number of Enrollees by Sex, Age, Grade Level, and etc., needed by DepEd and Private Educational Assistance Committee (PEAC).
- (l) View grade submission logs of teachers

4. A **Cashier**'s account will be able to do the following functionalities:

- (a) Login and logout
- (b) View and update profile
- (c) Change his/her password
- (d) View students currently enrolled
- (e) Disable a student account
 - i. Holding a student account account will only happen if the student has unpaid balance.

5. **Teacher**'s account will be able to do the following functionalities:

- (a) Login and logout
- (b) View and update profile
- (c) Change his/her password
- (d) Input and update class records
- (e) View class records
- (f) Submit grades

If the teacher is also an adviser, additional functionalities will be available:

(a) View condensed grades of the section he/she is handling

(b) View his/her advisee's report cards in pdf format

A pdf file will be available once the condensed grades have been finalized.

6. Student account will be able to do the following functionalities:

(a) Login and logout

(b) View and update profile

(c) Change his/her password

(d) View report cards from past grade levels to present

7. Parent/Guardian account will be able to do the following functionalities:

(a) Login and logout

(b) View and update profile

(c) Change his/her password

(d) View student's report cards from past grade levels to present

D. Significance of the Project

Using spreadsheets are proven to be a useful tool for keeping student records. But it also has disadvantages which includes long preparation time and integrity of files. Difficulties in generating subject-centric and student-centric class records from multiple teacher-centric class records also arise. It also has security problems; the file can be manipulated and modified by anyone without any permission.

Dee Hwa Liong Academy Grade Management System will lessen the time needed for post-enrollment processes, grade submission, and deliberation. The system will also eradicate the need of using flash drives and facebook messenger for submission of grades. Tracking late submissions of grades will also be easier since there will be deadline feature which will be set for each teacher in the system. The application will also provide security for students' records since grades will not be easily modified and an update log will also be available.

In addition, this application will also keep parents up to date with their child's performance and will provide a more efficient way of keeping track of students with backlog subjects and with honors.

E. Scope and Limitations

1. The system will be created based on the process followed by Dee Hwa Liong Academy.
2. The system will only cover the Kinder2 to Grade 12 students.
3. The deliberation of grades will still be done personally by the registrar, subject teachers, advisers, and possibly, by the director and principal.
4. Character traits of students will also be included in the system.
5. The system will be created to solve bottle neck problems of the grade management system of Dee Hwa Liong Academy. These bottle neck problems include the preparation and distribution of class records per subject teacher, produce a condensed class record per section, and formatting of grades for printing.
6. The system will be fully online, no offline counterpart, and accounts cannot be requested through the system. Accounts will be created by the system administrator.

F. Assumptions

Listed below are the assumptions for the Dee Hwa Liong Academy Grade Management System

1. Active student users are assumed to be officially enrolled by the school registrar. Enrollment is done outside the system.
2. React.js and Express.js will be used for developing the web application therefore the server to be used by the client must support a node.js server environment.
3. The grading system is provided by the academy and follows the K to 12 program grading system.

II. Review of Related Literature

Web-based information system plays a vital role in the educational institutes or colleges in order to maintain a record of the students easily. As far as the matter of students' academic career is concerned, it is very important to manage the accurate record and up-to-date information so that they can easily access it. Information system deals with all types of details of the students like course registration, notifications, semester calendar, academic record, exam components, exam slip, timetable, attendance details, students' feedback and many more as per the needs of any institution. It is very convenient not only for students but also for the director, registrar, teachers, and parents to access the academic record of students instantly by just one click. It does not only save time but also the problems faced by the staff. They do not need to use any ink and paper in order to do any sort of work related to that institution.

According to Gehlawat (2017), the school administrative processes and the official procedures of school such as grade management and enrollment period can be done in a more efficient way by the use of management information systems. The development of information technologies has provided a huge contribution to educational organizations. The implementation of information systems in universities can result in a significant decreased use of paper and turning most of the school office documents in an electronic format. Thus the schools are encouraged to employ information system to improve the performance of administrative services. Information technology provides several potential uses in educational organizations. It can range from simple interactive software for classroom teaching up to automation of system administrator processes such as admission and grade management [5].

Information and Communication Technologies (ICT) continues to innovate and improve the efficiency of several systems all around the world. In the education system, it is said that the exposure of students to application of ICT has a major role in learning process for university students both outside and inside the campus setting. Majority of universities that have implemented a system which fully adopted ICT have recorded significant advancement in the improvement of teaching, research, learning methods, and development. [6]. Hu discusses the importance of Student Information System for students. He also added that the technical modernization and campus network construc-

tion is an important way to achieve the current student data management information construction. The role of this system is associated mainly with the instructor, administrator, and student [7].

Mostly the institutions are rated on the availability of services as well as satisfaction provided to the users. In order to fulfill the needs and expectations of the students, the Web-based student information system plays a vital role. Therefore, in many universities students are assisted by in the form of registration, issuance of the transcripts of certificates, and financial recording. Student Information Management System (SIMS) software proves beneficial not only for students but also for administrators. Moreover, SIMS keeps students aware of any important event, activities etc. [8]. However, Asogwa et. al. (2015) observed that the revolution of technology, there are many universities in which the staff are still using the same old method for administrative purposes. Student information system has helped students to view their grades by just entering their roll number in student login panel of university. They do not need to waste their time by standing for hours in a queue [9].

93.5% of the processes which includes grade management and admission into exclusive private schools are performed manually using paper and ink. [10]. Fujo et. al. (2018) discussed the limitations and problems of having a manual system which include loss of forms which can caused by misplaced documents, long preparation time, finding an appropriate school and subjects an applicant can get admission, and disfigurement of forms handled by the student throughout the process for admission. Consequently, the published paper was submitted on an ongoing research work to design and implement a Tanzania Central Processing Admission System (TCPAS) that provides major changes towards the maintenance of admission costs, possible forging of documents for admission qualifications, encourage the use of a paperless system for admission, ability to reach more students which are geographically distant to the school, and centralization of the school data. Researchers have observed that one of the greatest achievements of Tanzania is the web based admissions system. They have done a great job in order to monitor and control the quality for admission into technical and tertiary education.

The electronic class record which was developed for the faculty members of the Lyceum of the Philippines University-Laguna International School is proven to follow the grading standards of the institution. By using the system, class record can easily be

managed and processes including computation of grades can be done in an efficient and convenient way. It also provides accuracy, reliability, security, use-friendliness, flexibility, and validity. But one of the system's limitation is the class record, which is made using Microsoft Excel, lacks security features including tracking down personnel who edited the file. The end users and administrator suggest locking the file once it is submitted. [4]

A key component of e-UP Project, one of the flagship project of the University of the Philippines Administration, is the establishing of multiple web-based information systems which include our own Student Academic Information System (SAIS), Financial Management Information System (FMIS), Supplies Procurement and Campus Management Information System (SPCMIS), Human Resource Information System (HRIS), and Executive Information System (EIS). The system also aims to provide a platform that solves the existing problems which involve human errors due to manual operations. It also aims to harmonize, integrate, and interoperate the ICT systems and across all Constituent Universities (CUs). The implementation of such system will also allow for the improvement on the efficiency of its core functions. [11]

The University of Calabar had faced challenges which were identified in the students' data processing in the Department. They include the long preparation time and the release of final grades to students to check their performance. There were also excessive paper works, poor management of student records, and poor data and security features of student records and files. It was also stated that there is also the problem of lack of information to properly guide students during the admission process of students which could possibly result to presumptions in offering and dropping courses. Uzede et. al. stated that developing an Information System (IS) such as Students' Record Management Information System (SRMIS) provides data in the form of prebuilt documents that helps the decision-making processes of the users. It can also lessen the processing time which includes result generation, that is, Cumulative Grade Point Average (CGPA) computation in the University of Calabar. This allows efficient and convenient access to students' information such as grades and student records. It also can enable the implementation of layers of security by allocating access privileges and monitoring of mischievous acts of altering scores in the result sheet.[12]

The emergence of online student information system is an evolution that has im-

proved the process of managing student data and records. It also becomes a symbol of modernity in universities. Web based student information systems also has a major role in providing the requirements of effective management of students' catalog and records in educational organizations. The implementation of Mzuzu University Student Online Management System (SOMS) resulted to the increase of performance in the school's registration process and it also provide access to examination results amongst the student community. Lubanga et. al. also discussed the seamless transfer of information between the university management and the students with the help of internet technologies [13].

III. Theoretical Framework

A. Assessment of Students

Dee Hwa Liong Academy handles students from Nursery, Kinder1, Kinder2, and Grades 1 to 12.

1. Nursery and Kinder1

Nursery and Kinder1 use a progressive curriculum, and the basis for assessing a student is a checklist. Checklists are called Developmental Assessment Scale. This scale is divided into four development skills - Physical Development, Self-help Skills, Socio-emotional Development, and Pre-academic Development. Physical development is further divided into gross motor development and fine motor development. On the other hand, pre-academic development is divided into reading readiness, language development, computer literacy, number readiness, music, art and P.E. (MAPE), and Chinese. Each skill under these development skills are graded according to a scale. All traits are observed by the students' adviser. The legends for this checklist are excellent (E) - excellent knowledge, very good (VG) - notable knowledge, good (G) - satisfactory knowledge, average (A) - fair knowledge, present (P) - observed, and not observed (OB) - not yet observed. Grades of nursery and kinder1 pupils are combined and deliberated quarterly.

2. Kinder2, Grades 1 to 10

Kinder2 and Grades 1 to 10 use the K to 12 curriculum. Teachers are given ECRs to input students' scores for each component. The basis for assessments are written work, performance tasks, and exams. The spreadsheet file has columns for formative tasks. Formative tasks are used to check if students are ready for graded seatwork and activities. Usually, formative tasks are not part of a student's grade but are recorded in the class record. Written work is divided into two, quizzes and others, while performance tasks comprise of oral participation, individual work, group work, individual project, group project and other output by students. In addition to these classroom activities, a quarterly exam is also used and recorded to check a student's performance. Similar to nursery and kinder1 students, grades are combined and deliberated quarterly.

3. Grades 11 to 12

Class records for grades 11 to 12 are similar to kinder2 and grades 1 to 10. The only difference is instead of combining the grades quarterly, it is done per semester. Each semester, two exams are taken by students: midterm and final exam.

Character traits are also graded by teachers. 50% of the character trait grades comes from the adviser while the remaining 50% will come from the subject teachers. The character traits to be observed are *makadiyos*, *makatao*, *makakalikasan*, and *maka-bansa*. Character traits are graded with a scale: always observed (AO) - 100, sometimes observed (SO) - 90, rarely observed (RO) - 80, and not observed (NO) - 70.

Sometimes, attendance is included in a student's grade. Student's attendance and tardiness are recorded daily by teachers.

B. Teacher-centric Grade Sheet

A teacher-centric class record is a spreadsheet file which contains lists of students currently handled by the teacher. The spreadsheet contains multiple tabs. Each tab represents the subject currently handled by the teacher. It contains a list of students taking that subject and their corresponding scores for each component. The final grade is automatically computed for every addition or adjustment of scores.

C. Condensed Grade

Condensed grade is a term used to describe the average grade of the student. It is obtained from the section-centric class record.

D. Section-centric Grade Sheet

A section-centric class record is a spreadsheet file which contains a list of students under a given subject. The first column has the student names. The remaining columns are populated by the grades of each student from a given subject. The last column contains the condensed grade.

E. Student-centric Grade Sheet

A student-centric class record is used for the printing of report cards. It is a spreadsheet file which contains a list of subjects the student is currently taking. The list of subjects is then followed by their corresponding grades.

F. Grade Submission

Submission of grades are done per semester for SHS students, and quarterly for kinder2 to grade 10. Submission of grades is done through messenger or passing a flash drive to the registrar.

The registrar also keeps track of the submission date of the teachers. This is done manually by writing down the date the teacher submitted his/her class record(s).

Character traits and attendance sheets will also be checked by the registrar.

G. Individual Deliberation of Grades

Deliberation of grades is done after the teacher has submitted his/her class records. It is always done personally with the registrar and the IT head. This is to ensure that all grades are correctly collected and final adjustments have been made before the creation of the condensed grades.

The registrar will talk and review the students' grades with the subject teacher. If no issues were raised during the talk, the registrar will accept the class record to be added to the condensed grades according to the sections, generating section-centric records. Adding students to the class record will also be done during the discussion with the subject teacher.

H. Group Deliberation of Grades

Group deliberation is done for the section-centric class records. In this deliberation, the subject teachers, adviser, and registrar will sit down together to talk about a section's class record. Grade clarifications and student's behaviors are also discussed during the deliberation

Grade adjustments are still possible in this process. The teacher can readjust the student scores, component weights, and the transmuted grade from the teacher-centric class record.

I. Printing of Report Cards

Report card is a teacher-centric record which is reviewed by each adviser to check if there are any errors in the input. It includes the grades for both academic and character traits. It will also include the student's attendance and tardiness record.

Report cards are created based on the condensed grade file. These cards are printed on a normal sheet of paper for the first three quarters. For the final quarter, it is printed on a card. These cards are distributed to parents quarterly.

J. Node.js

Node.js is an open source run-time environment. This was built in Chrome's V8 JavaScript engine [14]. It provides a long term efficiency through event-driven and non-blocking I/O model and server-size JavaScript [15]. Unlike apache web servers which uses PHP as a default language, this allows the creation of Web Applications and server connections using Javascript and a collection of external "modules" that manages various core functionality.

K. Express.js

Express.js covers the core Node.js *http* module (<http://nodejs.org/api/http.html>) to provide extensive functionalities and features [16]. This framework consists of many plugin modules called *middleware* [16]. Express acts as a foundation for a custom-built framework which fits the web application project.

L. React.js

React.js is a javascript library for building modern user interfaces [17]. It was created by Facebook and independent contributors and organizations. One of the key features of this library is the use of a "Virtual Document Object Model" or "Virtual DOM". It enables developers to build a whole web application as if the entire webpage is rendered on each individual page but only web components that actually change. It also uses Javascript XML (JSX), which is an extension to the Javascript syntax. Its syntax is similar to the Hypertext Markup Tool (HTML), which makes it similar to existing web developers.

M. MySQL

My Structured Query Language (MySQL) is an open-source relational database management system (RDBMS), and has around 6 million installations worldwide. [18]. This is available as free software and is under GNU General Public License (GPL) [19].

Some of the advantages of MySQL includes portability, good security features, flexible table structure, can be integrated with various programming languages, and small RAM usage [19].

IV. Design and Implementation

A. Context Diagram

The web application will have seven access levels such as the system administrator, director, registrar, cashier's office, teachers, parents/guardians, and students. A context diagram is shown below in Figure 1.

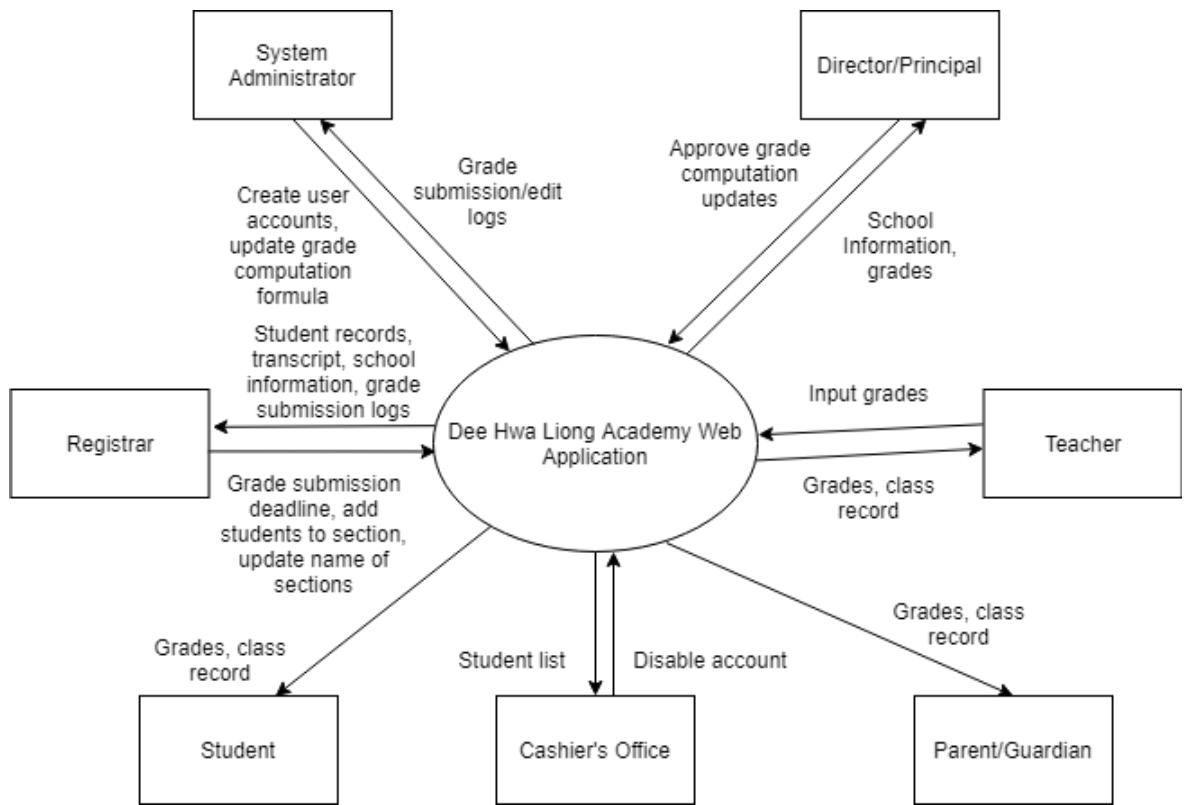


Figure 1 - Context Diagram, DHLA Grade Management System

B. Data Flow Diagram

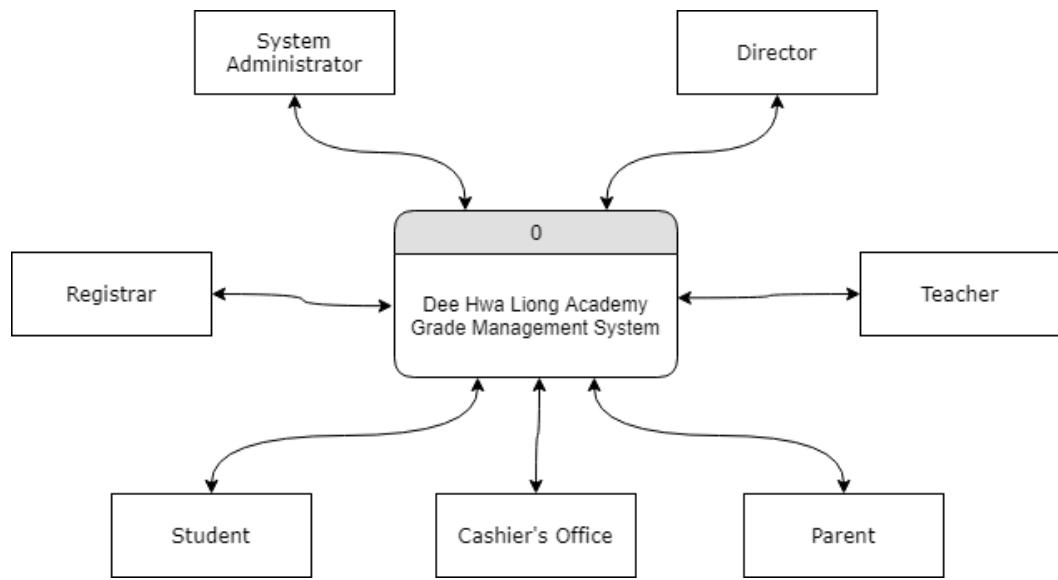


Figure 2 - Top Level Data Flow Diagram, DHLA Grade Management System

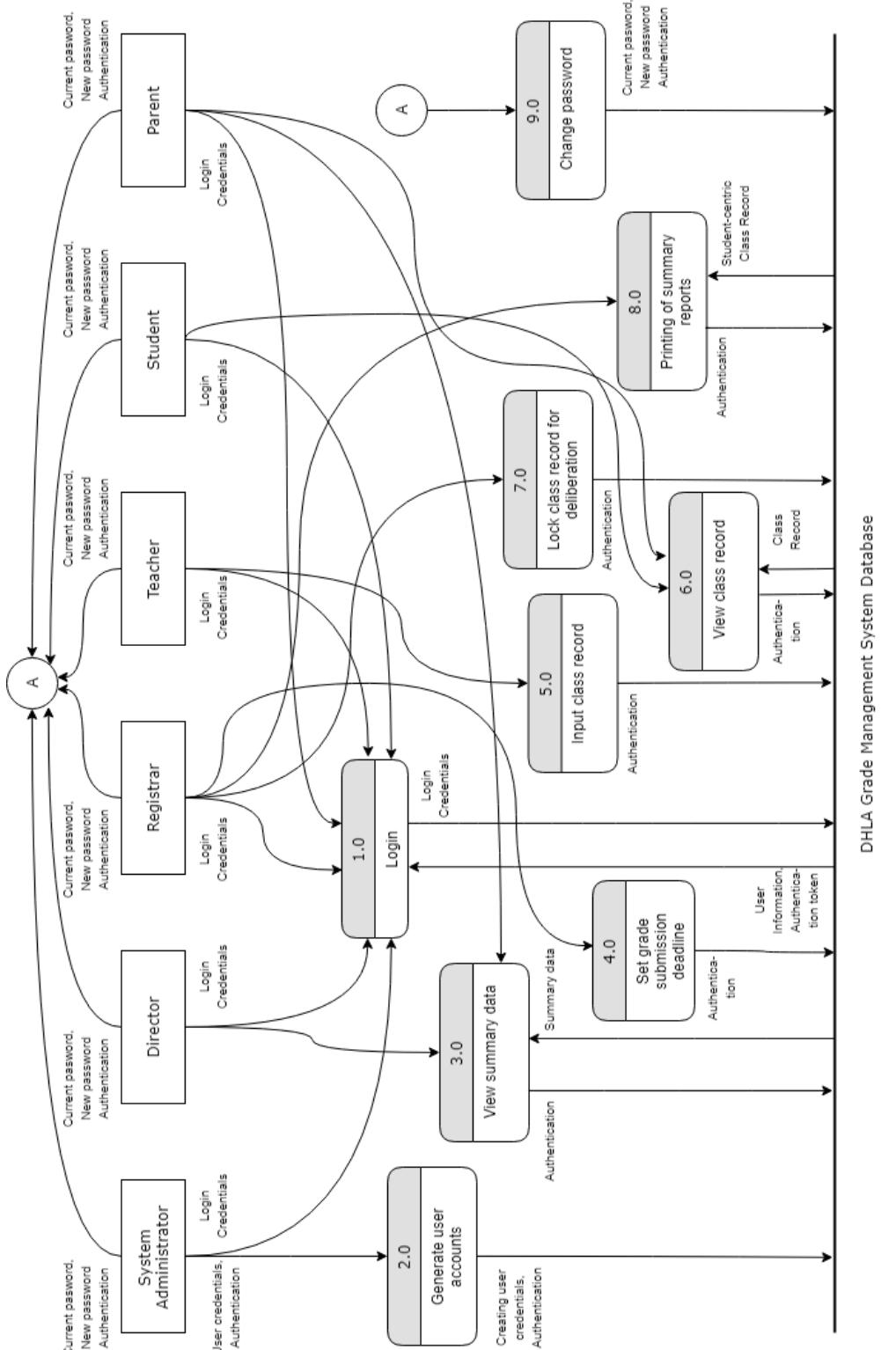


Figure 3 - Level 1 Data Flow Diagram, DHLA Grade Management System

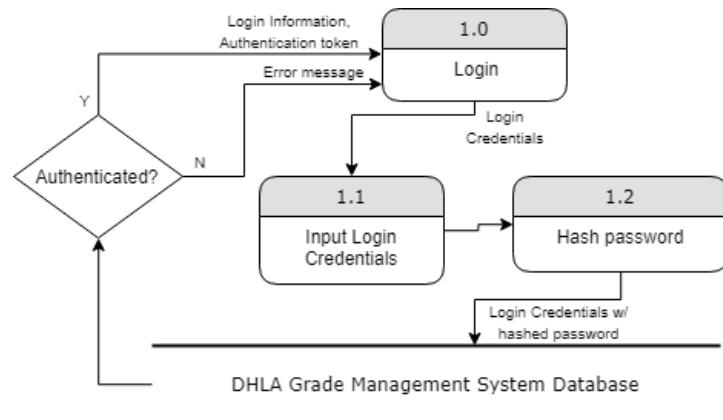


Figure 4 - Level 2 Data Flow Diagram for Process 1.0, DHLA Grade Management System

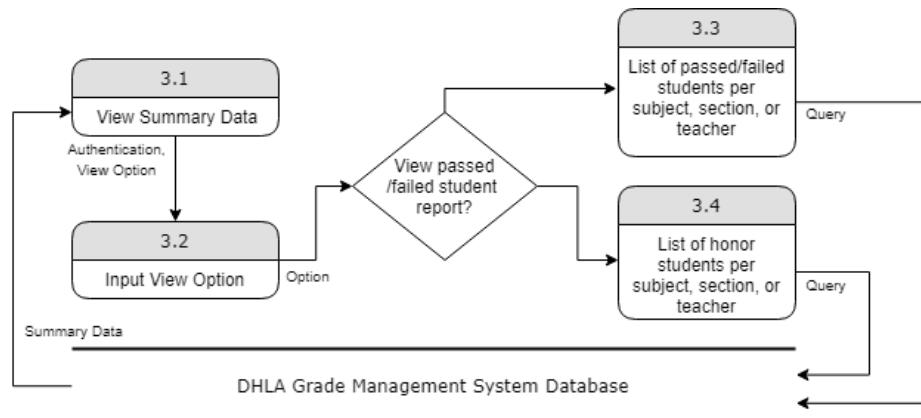


Figure 5 - Level 2 Data Flow Diagram for Process 3.0, DHLA Grade Management System

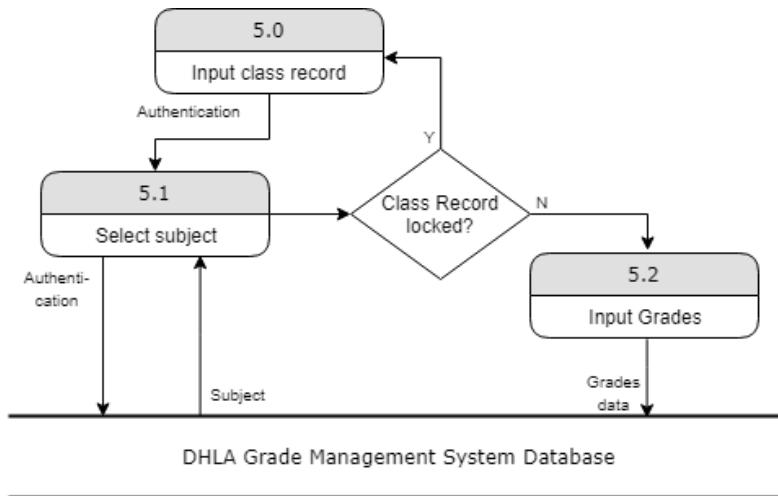


Figure 6 - Level 2 Data Flow Diagram for Process 5.0, DHLA Grade Management System

C. Activity Diagrams

All accounts will have a login and logout function. Each user account can also change the account's password. The password change is done within the system. No external links will be provided for password change. Profiles for each account will be available and can be updated.

The system administrator is the one managing the system.

The system administrator can create accounts for the other users such as director, registrar, teachers, and students. All account creation will not have any creation requests.

The administrator and registrar have an access to the activity logs of the class record. The activity log records every information changes that occur to class records in the system. It includes adding, updating, and deleting records. This activity log cannot be edited nor deleted, it is only readable. Both administrator and registrar can also search the log using date, time, teacher's name, section, school year, and grade level.

Activation and deactivation of accounts is also a functionality of the administrator. An account can be deactivated/deleted if the user is no longer affiliated with the school.

The director can view grades of each subject and the condensed grades. Viewing of summary reports is also possible.

Summary reports will get all the information needed from the database of the system.

The school registrar is in charge of keeping student records.

The school registrar will be able to set the deadline for submission of grades. Different deadlines may be set for different teachers due to their workload differences. Deadline alerts will be automatically sent, by the system, during 5 days and 3 days before the deadline and during the deadline day.

Since a school registrar keeps student records, they can view record from past school years and produce unofficial TORs. Viewing of summary reports will also be possible. Summary reports may be requested by the principal, director, and/or the DepEd and PEAC.

School registrars can update the student list of each section by adding or removing students from the class list. They can also update the section name and view the submitted grade of teachers.

Another activity log will be present under this user, which is the submission logs of teachers. This will be used to check the teachers who failed to comply with the deadline set by the registrar.

A school will never be a school without its teachers. In this system, there are two types of teachers: teacher-only and teacher/advisers.

Both types can input and update the grades of students. Their class records will be automatically be available after they input the class and subjects they are handling in their profiles.

Both teacher-only and advisers can also submit the grades through the application to create the condensed grades sheet. Before submitting the grades, the teacher must be sure that there are no errors with the record. A save function will be available to save changes made in the class record.

The difference between teacher-only and teacher/adviser, teacher/adviser can view the condensed grades of their advisees from the profiles. In addition, teacher/adviser can view his/her advisees' report cards after the condensed grades have been finalized.

Finally, the student, this user can only do simple things such as view grades from his/her past grade levels to present.



Figure 4 - Use Case Diagram, DHLA Grade Management System

D. Database Design

1. Entity Relationship Diagram (ERD)

Figure 1 shows the relationship among the user account, student, parent guardian, teacher, and nonacademic tables. The user account table contains the login credentials and basic information such as first name, last name, middle name, etc. of the user. System administrator, registrar, and principal/director fall under the nonacademic table.

Figure 2 shows the relationship among the class record, category, grade, teacher, section, grade level, and subject. The class record table contains the collection of grades of students in a section. The category table shows the type of activity being recorded in the class record. The grade table contains an individual grade of the student, and other important information such as date, category, and entry number. The section table shows the section name of the class record. The subject table contains the subject code and the subject name of the class record.

Figure 3 shows the relationship among the grade, student, category, and class record. The grade table contains an individual grade of the student, and other important information such as date, category, and entry number. The category table shows the type of activity being recorded in the class record. The class record table contains the collection of grades of students in a section.

Figure 4 shows the relationship among the attendance log, teacher, and the student. The attendance log table contains the atendance information of a student in a specific day. The table also contains the studentID of the student and the teacher recording the attendance.

Figure 5 shows the relationship among the advisory table, grade level, teacher and section. The advisory table contains the section being handled by the adviser. It also contains other information such as the grade level of the students being handled, the section, school year, and academic term. The teacher can only have an exact one advisory section.

Figure 6 shows the relationship among the component weight table, category, and subject. The component weight table shows the weight of each component from each subject. The category table shows the type of activity being recorded in the class record.

Figure 7 shows the relationship among the subcomponent weight table, section,

teacher, category, sub category, and subject. The subcomponent weight table shows the weight of each subcomponent of a component from each subject currently handled by the teacher. The category table shows the type of activity being recorded in the class record.

Figure 8 shows the relationship among the class, subject, teacher, and section. The class table shows the subject being handled by the teachers. Teachers can have multiple load since they can teach multiple sections throughout the academic year. They can also teach multiple subjects.

Figure 9 shows the relationship among submission deadline and teacher. The submission deadline table contains the deadline date for the submission of grades of the teacher set by the registrar. Teacher can only have exactly one submission deadline.

Figure 10 shows the relationship among student load and student. The student load table contains the load of the students in a given quarter. The students can have multiple student loads.

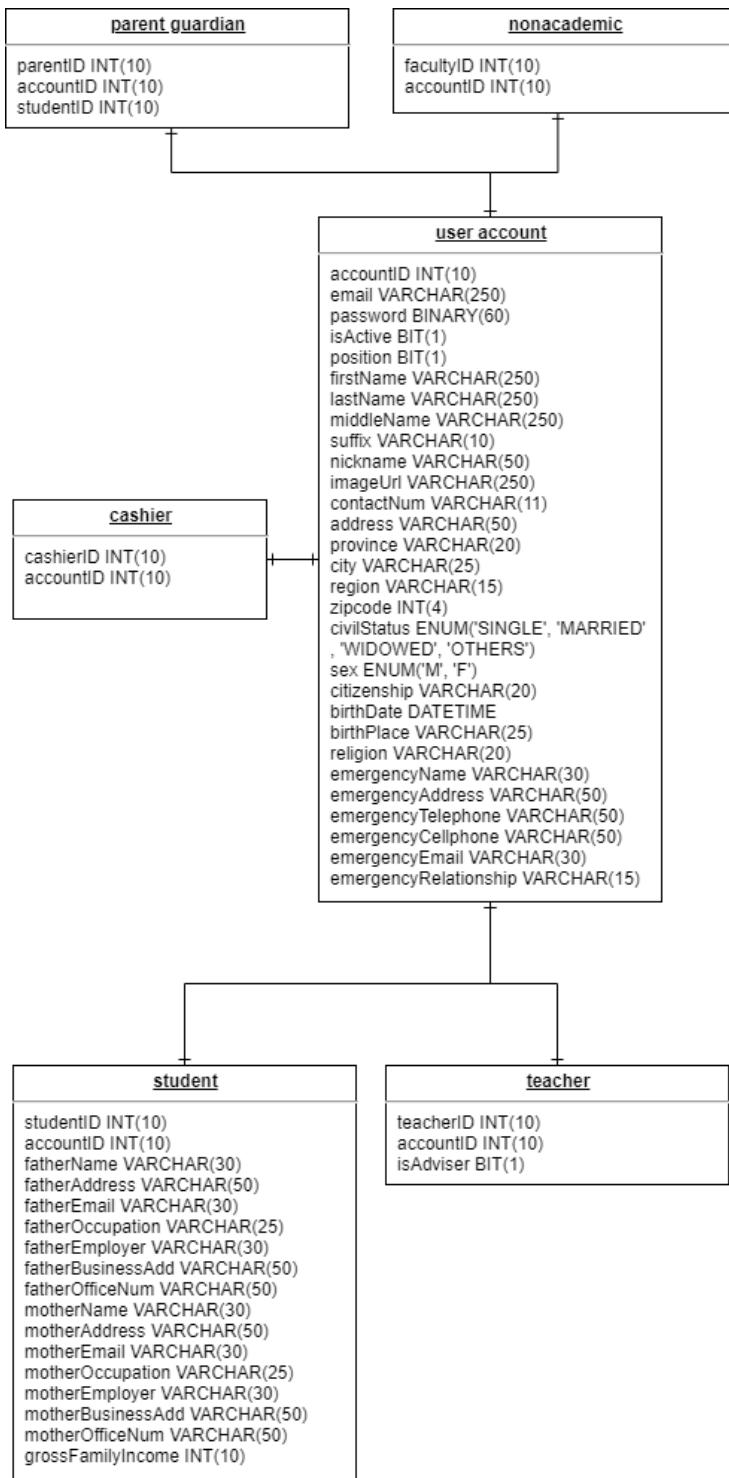


Figure 1 - Entity Relationship Diagram (ERD) (user account)

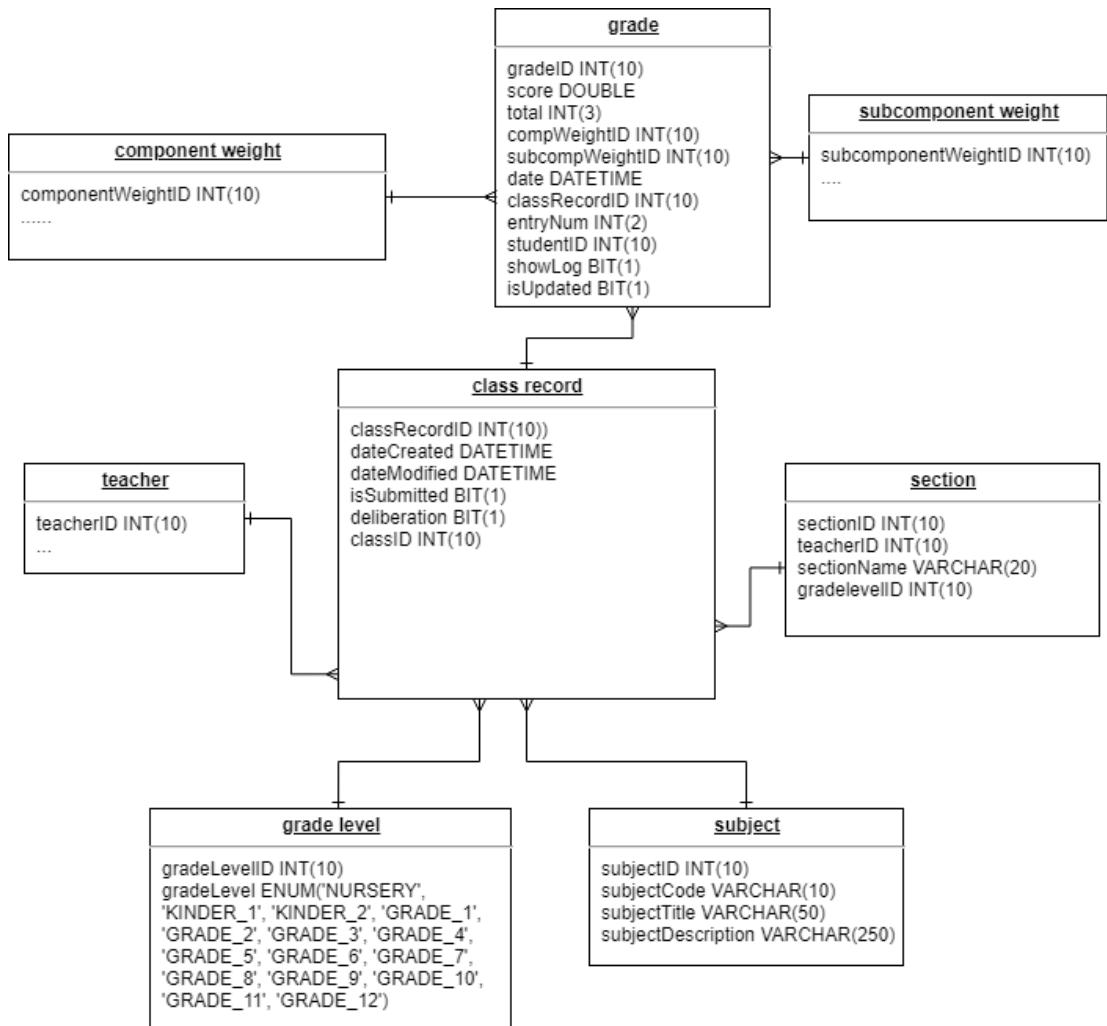


Figure 2 - Entity Relationship Diagram (ERD) (class record)

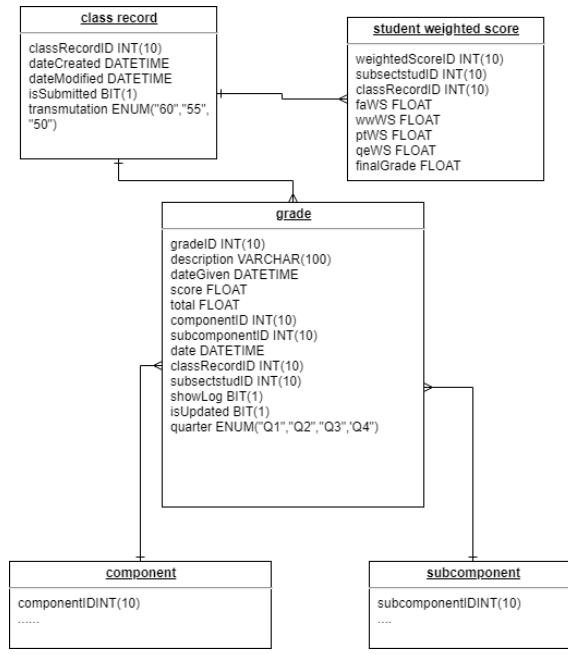


Figure 3 - Entity Relationship Diagram (ERD) (grade)

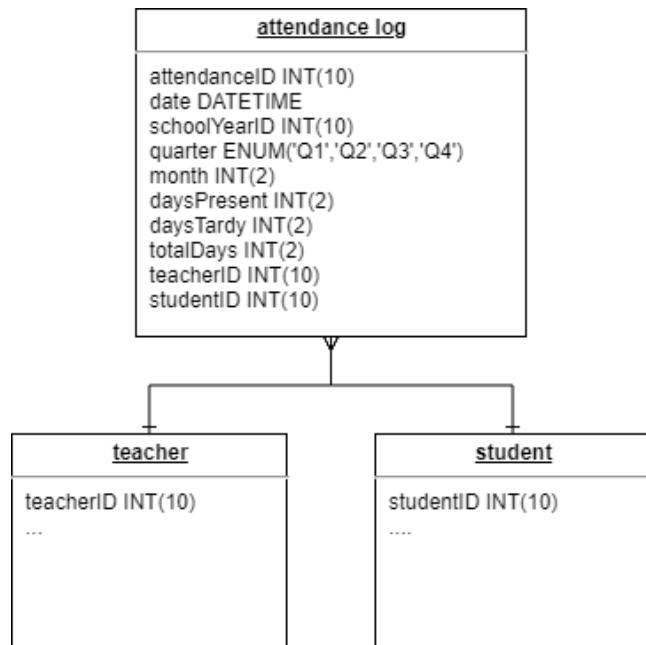


Figure 4 - Entity Relationship Diagram (ERD) (attendance log)

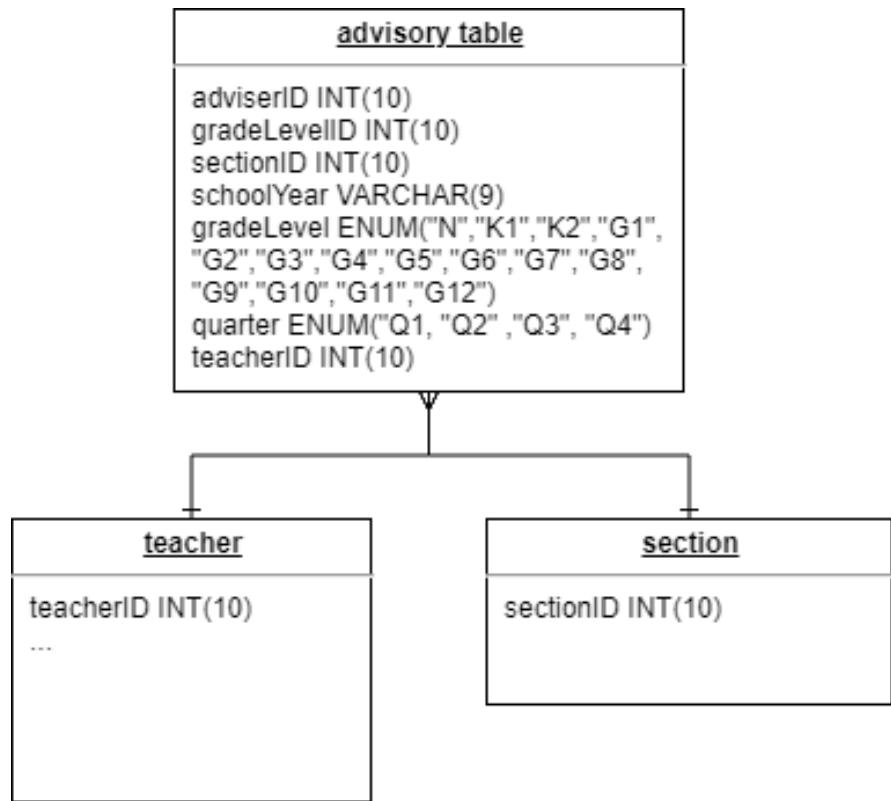


Figure 5 - Entity Relationship Diagram (ERD) (advisory table)

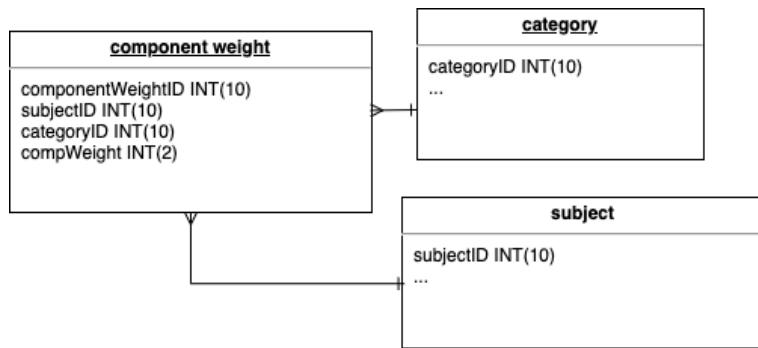


Figure 6 - Entity Relationship Diagram (ERD) (component weight)

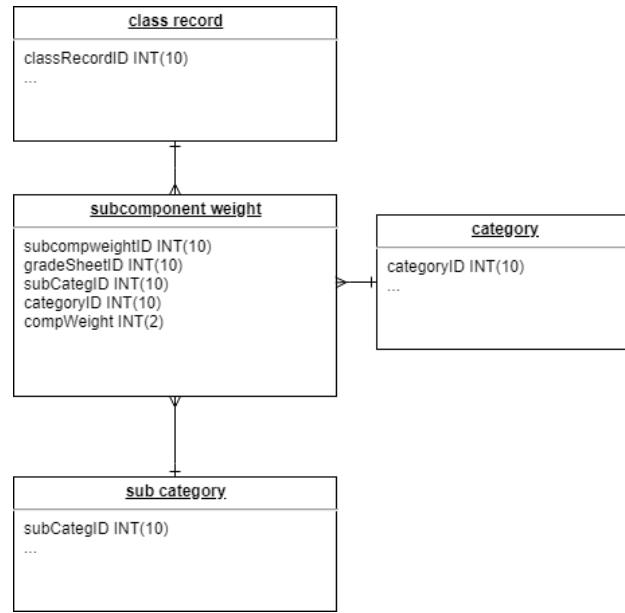


Figure 7 - Entity Relationship Diagram (ERD) (subcomponent weight)

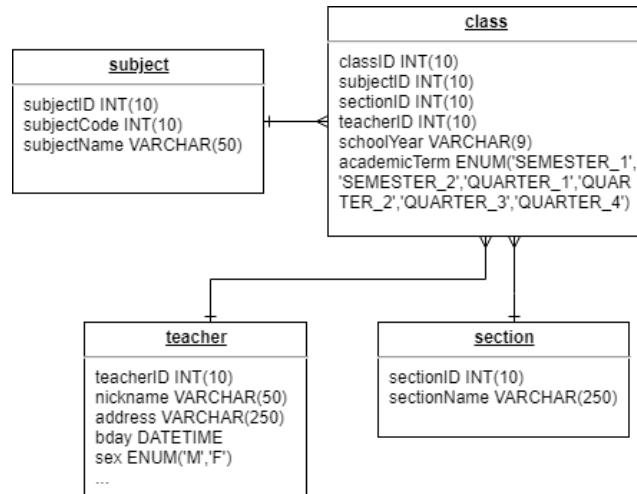


Figure 8 - Entity Relationship Diagram (ERD) (class)

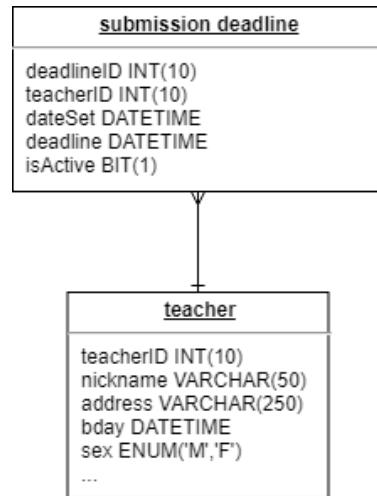


Figure 9 - Entity Relationship Diagram (ERD) (submission deadline)

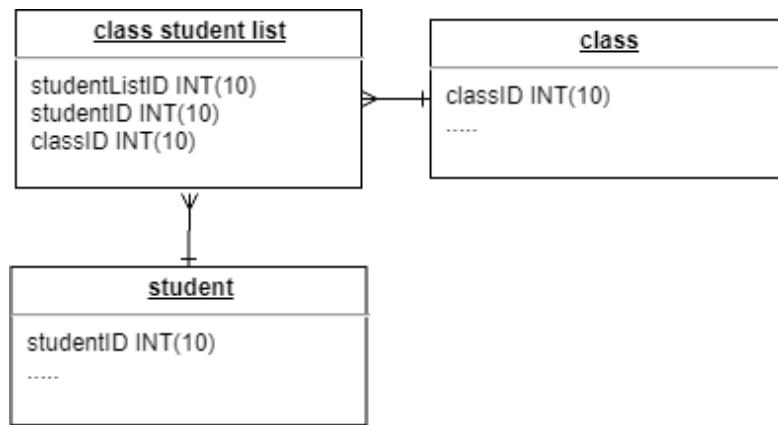


Figure 10 - Entity Relationship Diagram (ERD) (class student list)

E. Data Dictionary

Shown below are the database tables. Primary keys are in bold format.

1. User Account

This table will provide the information needed to login.

user account			
FIELD	TYPE	KEY TYPE	DESCRIPTION
accountID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of user
email	VARCHAR(250)		Unique email address of user
password	BINARY(60)		Password of user in the system
isActive	BIT(1)		Denotes if the account is activated (1) or not (0)
position	BIT(1)		Denotes the account type: administrator (0), director (1), registrar (2), teacher (3), student (4), parent (5)
firstName	VARCHAR(250)		First name of the user

lastName	VARCHAR(250)		Last name of the user
middleName	VARCHAR(250)		Middle name of the user
suffix	VARCHAR(10)		Suffix of the user
nickname	VARCHAR(50)		Nickname of the user
imageUrl	VARCHAR(250)		Path to the photo of the user
contactNum	VARCHAR(11)		User's contact number
address	VARCHAR(50)		User's address
province	VARCHAR(20)		User's province
city	VARCHAR(25)		User's city
region	VARCHAR(15)		User's region
zipcode	VARCHAR(4)		User's zip code
civilStatus	ENUM('SINGLE', 'MARRIED', 'WIDOWED', 'OTHERS')		User's civil status
sex	ENUM('M', 'F')		User's sex
citizenship	VARCHAR(20)		User's citizenship
birthDate	DATETIME		User's birth date

birthPlace	VARCHAR(25)		User's birth place
religion	VARCHAR(20)		User's religion
emergency- Name	VARCHAR(30)		Contact name of the student in case of emergency
emergency- Address	VARCHAR(50)		Contact address of the student in case of emergency
emergency- Telephone	VARCHAR(50)		Telephone num- ber of the student in case of emer- gency
emergency- Cellphone	VARCHAR(50)		Cellphone num- ber of the student in case of emer- gency
emergency- Email	VARCHAR(30)		Contact email of the student in case of emergency
emergency- Relationship	VARCHAR(15)		Relationship of the student in the emergency contact

Table 1: Data dictionary for **user account** table

2. Student

The table will contain information about the student. The gathered information fields came from Dee Hwa Liang Academy's Student Application form.

student			
FIELD	TYPE	KEY TYPE	DESCRIPTION
studentID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of student
accountID	INT(10)	FOREIGN	Unique identifier of user
fatherName	VARCHAR(30)		Student's father's name
fatherAddress	VARCHAR(30)		Student's father's address
fatherEmail	VARCHAR(30)		Student's father's email
fatherOccupation	VARCHAR(30)		Student's father's occupation
fatherEmployer	VARCHAR(30)		Student's father's employer
fatherBusinessAddress	VARCHAR(30)		Student's father's business address
fatherOfficeNum	VARCHAR(30)		Student's father's office

motherName	VARCHAR(30)		Student's mother's name
motherAdd -ress	VARCHAR(30)		Student's mother's address
motherEmail	VARCHAR(30)		Student's mother's email
motherOccupation	VARCHAR(30)		Student's mother's occupation
motherEmployer	VARCHAR(30)		Student's mother's employer
motherBusinessAddress	VARCHAR(30)		Student's mother's business address
motherOffice - Num	VARCHAR(30)		Student's mother's office

Table 2: Data dictionary for **student** table

3. Parent/Guardian

The table will contain information about the parent/guardian.

parent guardian

FIELD	TYPE	KEY TYPE	DESCRIPTION
parentID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of parent/-guardian
accountID	INT(10)	FOREIGN	Unique identifier of user
studentID	INT(10)	FOREIGN	Unique identifier of the parent/-guardian

Table 3: Data dictionary for **parent guardian** table

4. Cashier

The table will contain information about the cashier.

cashier			
FIELD	TYPE	KEY TYPE	DESCRIPTION
cashierID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of cashier
accountID	INT(10)	FOREIGN	Unique identifier of user

Table 4: Data dictionary for **parent guardian** table

5. Teacher

The table will contain basic information about the teacher.

teacher			
FIELD	TYPE	KEY TYPE	DESCRIPTION
teacherID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of teacher
accountID	INT(10)	FOREIGN	Unique identifier of user
isAdviser	BIT(1)		Denotes if the teacher has an advisory class (1) or not (0)

Table 5: Data dictionary for **teacher** table

6. Nonacademic

The table will contain basic information about the system administrator, director/principal, and registrar.

nonacademic			
FIELD	TYPE	KEY TYPE	DESCRIPTION
facultyID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of nonacademic

accountID	INT(10)	FOREIGN	Unique identifier of user
-----------	---------	---------	---------------------------

Table 6: Data dictionary for **nonacademic** table

7. Section

This table will contain all the names of sections.

section			
FIELD	TYPE	KEY TYPE	DESCRIPTION
sectionID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of section
teacherID	INT(10)	FOREIGN	Unique identifier and denotes the adviser of the section
sectionName	VARCHAR(20)		Denotes the name of the section
gradeLevelID	INT(10)	FOREIGN	Unique identifier of the grade level

Table 7: Data dictionary for **section** table

8. Subject

This table will contain all the subjects offered by Dee Hwa Liong Academy.

subject			
FIELD	TYPE	KEY TYPE	DESCRIPTION
subjectID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a subject
subjectCode	VARCHAR(10)		Subject code of the subject
subjectTitle	VARCHAR(50)		Title of the subject
subjectDescription	VARCHAR(250)		Description of the subject

Table 8: Data dictionary for **subject** table

9. Grade Level

Grade Level table will contain the grade levels under Dee Hwa Liong Academy.

grade level			
FIELD	TYPE	KEY TYPE	DESCRIPTION
gradeLevelID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a grade level

gradeLevel	ENUM('NURSERY', 'KINDER_1', 'KINDER_2', 'GRADE_1', 'GRADE_2', 'GRADE_3', 'GRADE_4', 'GRADE_5', 'GRADE_6', 'GRADE_7', 'GRADE_8', 'GRADE_9', 'GRADE_10', 'GRADE_11', 'GRADE_12')		Grade level of a student
------------	---	--	--------------------------

Table 9: Data dictionary for **grade level** table

10. Category

Category table will tell us what type of activity (formative assessment-seatwork, formative assessment-assignments, written work-quizzes, written work-others, performance task-oral participation, etc.) is being recorded in the class record

category			
FIELD	TYPE	KEY TYPE	DESCRIPTION
categoryID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a category
categoryName	ENUM('FORMATIVE', 'WRITTEN', 'PT', 'QUARTERLY_EXAM')		Name of category

Table 10: Data dictionary for **category** table

11. Subcategory

Subcategory table is a subcomponent of a specific component (formative assessment-seatwork, formative assessment-assignments, written work-quizzes, written work-others, performance task-oral participation, etc.) being recorded in the class record.

subcategory			
FIELD	TYPE	KEY TYPE	DESCRIPTION
subCategID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a subcategory
categoryName	VARCHAR(20)		Name of subcategory

Table 11: Data dictionary for **subcategory** table

12. Class record

This will consist of the grades of the students from every activity to condensed grades.

class record			
FIELD	TYPE	KEY TYPE	DESCRIPTION
classRecordID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a class record

dateCreated	DATETIME		Date when the class record was created
dateModified	DATETIME		Date when the class record was modified
isSubmitted	BIT(1)		Denotes if the grade has been submitted for deliberation (1) or not (0)
classID	INT(10)	FOREIGN	Unique identifier of a class

Table 12: Data dictionary for **class record** table

13. Grade

This represents a grade entry in a class record.

grade			
FIELD	TYPE	KEY TYPE	DESCRIPTION
gradeID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of a grade
score	DOUBLE		Student's score

total	INT(3)		The total number of items
categoryID	INT(10)	FOREIGN	Unique identifier of a category
subCategID	INT(10)	FOREIGN	Unique identifier of a subcategory
date	DATETIME		Date when the grade was created
classRecordID	INT(10)		Unique identifier of a class record
entryNum	INT(1)		Denotes the entry number of the grade in the class record
studentID	INT(10)	FOREIGN	Unique identifier of a student
showLog	BIT(1)		Denotes if the grade update will be show in the update log
isUpdated	BIT(1)		Denotes if the grade is the initial input (0) or not (1)

Table 13: Data dictionary for **grade** table

14. Attendance Log

This will hold the record of attendance of each student.

attendance log			
FIELD	TYPE	KEY TYPE	DESCRIPTION
attendanceID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of an attendance
date	DATETIME		School date
schoolYear	VARCHAR(9)		School year of the attendance log
academicTerm	ENUM('QUARTER_1', 'QUARTER_2', 'QUARTER_3', 'QUARTER_4')		Academic term of the attendance log
month	INT(2)		Month of the attendance record
daysPresent	INT(2)		Number of days the student is present
daysTardy	INT(2)		Number of days the student is tardy
totalDays	INT(2)		Total number of school days in that month

teacherID	INT(10)	FOREIGN	Unique identifier of a teacher
studentID	INT(10)	FOREIGN	Unique identifier of a student

Table 14: Data dictionary for **attendance log** table

15. Advisory Table

This table will show the section being handled by the adviser.

advisory table			
FIELD	TYPE	KEY TYPE	DESCRIPTION
adviserID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of an advisory table
gradeLevelID	INT(10)	FOREIGN	Unique identifier of a grade
sectionID	INT(10)	FOREIGN	Gives the unique identifier of a section if the teacher is an adviser
schoolYear	VARCHAR(9)		School year of the advisory table
academicTerm	ENUM('QUARTER_1', 'QUARTER_2', 'QUARTER_3', 'QUARTER_4')		Academic term of the advisory table

teacherID	INT(10)	FOREIGN	Unique identifier of a teacher
-----------	---------	---------	--------------------------------

Table 15: Data dictionary for **advisory table** table

16. Class

This table shows the subjects handled by different teachers.

class			
FIELD	TYPE	KEY TYPE	DESCRIPTION
classID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of the class
subjectID	INT(10)	FOREIGN	Unique identifier of a subject
sectionID	INT(10)	FOREIGN	Unique identifier of a section
teacherID	INT(10)	FOREIGN	Unique identifier of a teacher
schoolYear	VARCHAR(9)		School year of the teacher's load
academicTerm	ENUM('QUARTER_1', 'QUARTER_2', 'QUARTER_3', 'QUARTER_4')		Academic term of the teacher's load

Table 16: Data dictionary for **class** table

17. Submission Deadline

This table will contain the deadlines set by the registrar.

submission deadline			
FIELD	TYPE	KEY TYPE	DESCRIPTION
deadlineID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of the submission deadline
teacherID	INT(10)	FOREIGN	Unique identifier of a teacher
dateSet	DATETIME		When the deadline was set
deadline	DATETIME		Deadline of submission of grades
show	BIT(1)		Denotes if the Registrar will display the deadline to teachers (0) or not (1)

Table 17: Data dictionary for **submission deadline** table

18. Subcomponent Weight

This table will contain the subcomponent weights for each component.

formula			
FIELD	TYPE	KEY TYPE	DESCRIPTION
subcompID	INT(10), AUTO_INCREMENT	PRIMARY	Unique identifier of the formula
sectionID	INT(10)	FOREIGN	Unique identifier of a section
subjectID	INT(10)	FOREIGN	Unique identifier of a subject
teacherID	INT(10)	FOREIGN	Unique identifier of a teacher
subCategID	INT(10)	FOREIGN	Unique identifier of a subcategory
categoryID	INT(10)	FOREIGN	Unique identifier of a category
compWeight	INT(2)		The weight of the component per subject category

Table 18: Data dictionary for **subcomponent weight** table

F. System Architecture

Dee Hwa Liong Academy Web Application will be built under the Node.js server environment. Express.js will be used as a library for fetching data through API calls (using routers). The system will be using React.js library as the main front-end backbone. For the user interface, Ant Design, a design language made for React environment, alongside with Tabler React, an open-source UI framework for building dashboard applications will be used to make the website more responsive. MySQL will be used as the database of the system.

G. Technical Architecture

DHLA Grade Management System will be accessed online. It follows a client-server-database architecture. The main server should have the following specifications (minimum requirements):

System Technical Components

1. Node.js Run-time Environment for the Server (12.1.0 or higher)
2. MySQL (7.2.10 or higher)
3. Google Chrome (70.0.3538.102 or higher)
4. Opera (56.0.3051.104 or higher)
5. Microsoft Edge (42.17134.1.0 or higher)
6. Internet Connection

```

      35      section: {
      36        type: Sequelize.STRING
      37      },
      38      subject: {
      39        type: Sequelize.STRING
      40      },
      41      timestamp: {
      42        type: Sequelize.DATE
      43      },
      44      quarter: {
      45        type: Sequelize.STRING
      46      }
      47    },
      48    {
      49      freezeTableName: true
      50    }
      51  );
      52
      53 module.exports = ActivityLog;
  1  const Sequelize = require( "
  2    sequelize" );
  3  const sequelize = require( ".../config/db/database" );
  4  const AccountNotice = sequelize
  5    .define(
  6      "account notice", {
  7        accountnoticeID: {
  8          type: Sequelize.INTEGER,
  9          primaryKey: true,
 10         autoIncrement: true
 11       },
 12       accountID: {
 13         type: Sequelize.INTEGER
 14       },
 15       message: {
 16         type: Sequelize.STRING(
 17           100 )
 18       },
 19     },
 20   ),
 21   {
 22     freezeTableName: true
 23   }
 24 );
 25 module.exports = AccountNotice;

```

Listing 2: ActivityLog.js

```

  1  const Sequelize = require("sequelize");
  2  const sequelize = require("../config/db/database");
  3
  4  const ActivityLog = sequelize.
  5    define(
  6      "activity log",
  7      {
  8        logID: {
  9          type: Sequelize.INTEGER,
 10         primaryKey: true,
 11         autoIncrement: true
 12       },
 13       type: {
 14         type: Sequelize.ENUM(
 15           "ADD",
 16           "UPDATE",
 17           "DELETE",
 18           "CHANGE_STATUS",
 19           "SUBCOMP_UPDATE",
 20           "TRANSMU_UPDATE",
 21           "SUBCOMP_ADD",
 22           "SUBCOMP_DELETE",
 23           "DESC_UPDATE",
 24           "TOTAL_UPDATE"
 25         )
 26       },
 27       classRecordID: {
 28         type: Sequelize.INTEGER
 29       },
 30       position: {
 31         type: Sequelize.STRING
 32       },
 33       name: {
 34         type: Sequelize.STRING
 35       },

```

Listing 3: AttendanceLog.js

```

  1  const Sequelize = require("sequelize");
  2  const sequelize = require("../config/db/database");
  3
  4  const AttendanceLog = sequelize.
  5    define(
  6      "attendance log",
  7      {
  8        attendanceID: {
  9          type: Sequelize.INTEGER,
 10         primaryKey: true,
 11         autoIncrement: true
 12       },
 13       date: { type: Sequelize.DATE },
 14       schoolYearID: { type: Sequelize.INTEGER },
 15       quarter: {
 16         type: Sequelize.ENUM("Q1", "Q2", "Q3", "Q4")
 17       },
 18       month: { type: Sequelize.INTEGER(2) },
 19       daysPresent: { type: Sequelize.INTEGER(2) },
 20       daysTardy: { type: Sequelize.INTEGER(2) },
 21       totalDays: { type: Sequelize.INTEGER(2) },
 22       teacherID: { type: Sequelize.INTEGER },
 23       studentID: { type: Sequelize.INTEGER }
 24     },
 25   {
 26     freezeTableName: true
 27   }
 28
 29 module.exports = AttendanceLog;

```

Listing 4: ClassRecord.js

```

  1  const Sequelize = require("sequelize");

```

```

2 const sequelize = require("../
3   config/db/database");
4 const ClassRecord = sequelize.
5   define(
6     "class record",
7     {
8       classRecordID: {
9         type: Sequelize.INTEGER,
10        primaryKey: true,
11        autoIncrement: true
12      },
13      dateCreated: {
14        type: Sequelize.DATE
15      },
16      dateModified: {
17        type: Sequelize.DATE
18      },
19      q1Transmu: {
20        type: Sequelize.ENUM("60", "55", "50")
21      },
22      q2Transmu: {
23        type: Sequelize.ENUM("60", "55", "50")
24      },
25      q3Transmu: {
26        type: Sequelize.ENUM("60", "55", "50")
27      },
28      q4Transmu: {
29        type: Sequelize.ENUM("60", "55", "50")
30      },
31    },
32    {
33      freezeTableName: true
34    }
35  );
36 module.exports = ClassRecord;

```

Listing 5: ClassRecordStatus.js

```

1 const Sequelize = require(
2   "sequelize");
3 const sequelize = require("../
4   config/db/database");
5 const ClassRecordStatus = sequelize.
6   define(
7     "class record status",
8     {
9       classrecstatusID: {
10        type: Sequelize.INTEGER,
11        primaryKey: true,
12        autoIncrement: true
13      },
14      classRecordID: {
15        type: Sequelize.INTEGER
16      },
17      q1: {
18        type: Sequelize.ENUM("L", "E", "D", "F")
19      },
20      q2: {
21        type: Sequelize.ENUM("L", "E", "D", "F")
22      },
23      q3: {
24        type: Sequelize.ENUM("L", "E", "D", "F")
25      }
26    }
27  );
28 module.exports = ClassRecordStatus;

```

```

22       type: Sequelize.ENUM("L", "E"
23         , "D", "F")
24     },
25     q4: {
26       type: Sequelize.ENUM("L", "E"
27         , "D", "F")
28     },
29     q1DateSubmitted: {
30       type: Sequelize.DATE
31     },
32     q2DateSubmitted: {
33       type: Sequelize.DATE
34     },
35     q3DateSubmitted: {
36       type: Sequelize.DATE
37     },
38     q4DateSubmitted: {
39       type: Sequelize.DATE
40     }
41   },
42   {
43     freezeTableName: true
44   }
45 );
46 module.exports = ClassRecordStatus;

```

Listing 6: Component.js

```

1 const Sequelize = require(
2   "sequelize");
3 const sequelize = require("../
4   config/db/database");
5 const Component = sequelize.define(
6   "component",
7   {
8     componentID: {
9       type: Sequelize.INTEGER,
10      primaryKey: true,
11      autoIncrement: true
12    },
13    subjectID: {
14      type: Sequelize.INTEGER
15    },
16    component: {
17      type: Sequelize.ENUM("FA", "WW", "PT", "QE")
18    },
19    compWeight: {
20      type: Sequelize.FLOAT
21    }
22  },
23  {
24    freezeTableName: true
25  }
26 );
27 module.exports = Component;

```

Listing 7: Grade.js

```

1 const Sequelize = require(
2   "sequelize");
3 const sequelize = require("../
4   config/db/database");
5 // const LogDetails = require("../
6 //   models/LogDetails");
7 // const utils = require("../utils
8 // ");

```

```

5      const Grade = sequelize.define(
6        "grade",
7        {
8          gradeID: {
9            type: Sequelize.INTEGER,
10           primaryKey: true,
11           autoIncrement: true
12         },
13         description: {
14           type: Sequelize.STRING(100)
15         },
16         dateGiven: {
17           type: Sequelize.DATE
18         },
19         score: {
20           type: Sequelize.FLOAT
21         },
22         total: {
23           type: Sequelize.FLOAT
24         },
25         componentID: {
26           type: Sequelize.INTEGER
27         },
28         subcomponentID: {
29           type: Sequelize.INTEGER
30         },
31         date: {
32           type: Sequelize.DATE
33         },
34         classRecordID: {
35           type: Sequelize.INTEGER
36         },
37         subsectstudID: {
38           type: Sequelize.INTEGER
39         },
40         showLog: {
41           type: Sequelize.BOOLEAN
42         },
43         isUpdated: {
44           type: Sequelize.BOOLEAN
45         },
46         quarter: {
47           type: Sequelize.ENUM("Q1", "Q2", "Q3", "Q4")
48         },
49         attendance: {
50           type: Sequelize.ENUM("A", "E", "P")
51         }
52       },
53     },
54   {
55     freezeTableName: true
56   }
57 );
58 module.exports = Grade;
59 
```

8 type: Sequelize.INTEGER ,
 9 primaryKey: true ,
 10 autoIncrement: true
 11 },
 12 logID: {
 13 type: Sequelize.INTEGER
 14 },
 15 student: {
 16 type: Sequelize.STRING
 17 },
 18 component: {
 19 type: Sequelize.STRING
 20 },
 21 subcomponent: {
 22 type: Sequelize.STRING
 23 },
 24 description: {
 25 type: Sequelize.STRING
 26 },
 27 oldValue: {
 28 type: Sequelize.FLOAT
 29 },
 30 newValue: {
 31 type: Sequelize.FLOAT
 32 }
 33 },
 34 { freezeTableName: true }
 35);
 36
 37 module.exports = LogDetails;

Listing 9: Nonacademic.js

```

1  const Sequelize = require(
2    "sequelize");
3  const sequelize = require("../
4    config/db/database");
5  const Nonacademic = sequelize.
6    define(
7      "nonacademic",
8      {
9        facultyID: {
10          type: Sequelize.INTEGER ,
11          primaryKey: true ,
12          autoIncrement: true
13        },
14        accountID: {
15          type: Sequelize.INTEGER
16        }
17      },
18    { freezeTableName: true }
19  );
20 module.exports = Nonacademic;
```

Listing 8: LogDetails.js

```

1  const Sequelize = require(
2    "sequelize");
3  const sequelize = require("../
4    config/db/database");
5  const LogDetails = sequelize.define(
6    "log details",
7    {
      logdetailsID: {
```

Listing 10: ParentGuardian.js

```

1  const Sequelize = require(
2    "sequelize");
3  const sequelize = require("../
4    config/db/database");
5  const ParentGuardian = sequelize.define(
6    "parent guardian", {
```

```

7   type: Sequelize.INTEGER, 11
8   primaryKey: true, 12
9   autoIncrement: true 13
10  },
11  accountID: { 14
12    type: Sequelize.INTEGER 15
13  }, 16
14  studentIDs: { 17
15    type: Sequelize.STRING 18
16      (1000) 19
17    } 20
18  }, { 21
19    freezeTableName: true 22
20  } 23
21 ); 24
22 module.exports = ParentGuardian; 25
23 ; 26
24 ; 27
25 ; 28
26 ; 29
27 ; 30
28 ; 31
29 ; 32
30 ; 33
31 ; 34
32 ; 35
33 ; 36
34 ; 37
35 ; 38
36 ; 39
37 ; 40
38 ; 41
39 ; 42
40 ; 43
41 ; 44
42 ; 45
43 ; 46
44 ; 47
45 ; 48
46 ; 49
47 ; 50
48 ; 51
49 ; 52
50 ; 53
51 ; 54
52 ; 55
53 ; 56
54 ; 57
55 ; 58
56 ; 59
57 ; 60
58 ; 61
59 ; 62
60 ; 63
61 ; 64
62 ; 65
63 ; 66
64 ; 67
65 ; 68
66 ; 69
67 ; 70
68 ; 71
69 ; 72
70 ; 73
71 ; 74
72 ; 75
73 ; 76
74 ; 77
75 ; 78
76 ; 79
77 ; 80
78 ; 81
79 ; 82
80 ; 83
81 ; 84
82 ; 85
83 ; 86
84 ; 87
85 ; 88
86 ; 89
87 ; 90
88 ; 91
89 ; 92
90 ; 93
91 ; 94
92 ; 95
93 ; 96
94 ; 97
95 ; 98
96 ; 99
97 ; 100
98 ; 101
99 ; 102
100 ; 103
101 ; 104
102 ; 105
103 ; 106
104 ; 107
105 ; 108
106 ; 109
107 ; 110
108 ; 111
109 ; 112
110 ; 113
111 ; 114
112 ; 115
113 ; 116
114 ; 117
115 ; 118
116 ; 119
117 ; 120
118 ; 121
119 ; 122
120 ; 123
121 ; 124
122 ; 125
123 ; 126
124 ; 127
125 ; 128
126 ; 129
127 ; 130
128 ; 131
129 ; 132
130 ; 133
131 ; 134
132 ; 135
133 ; 136
134 ; 137
135 ; 138
136 ; 139
137 ; 140
138 ; 141
139 ; 142
140 ; 143
141 ; 144
142 ; 145
143 ; 146
144 ; 147
145 ; 148
146 ; 149
147 ; 150
148 ; 151
149 ; 152
150 ; 153
151 ; 154
152 ; 155
153 ; 156
154 ; 157
155 ; 158
156 ; 159
157 ; 160
158 ; 161
159 ; 162
160 ; 163
161 ; 164
162 ; 165
163 ; 166
164 ; 167
165 ; 168
166 ; 169
167 ; 170
168 ; 171
169 ; 172
170 ; 173
171 ; 174
172 ; 175
173 ; 176
174 ; 177
175 ; 178
176 ; 179
177 ; 180
178 ; 181
179 ; 182
180 ; 183
181 ; 184
182 ; 185
183 ; 186
184 ; 187
185 ; 188
186 ; 189
187 ; 190
188 ; 191
189 ; 192
190 ; 193
191 ; 194
192 ; 195
193 ; 196
194 ; 197
195 ; 198
196 ; 199
197 ; 200
198 ; 201
199 ; 202
200 ; 203
201 ; 204
202 ; 205
203 ; 206
204 ; 207
205 ; 208
206 ; 209
207 ; 210
208 ; 211
209 ; 212
210 ; 213
211 ; 214
212 ; 215
213 ; 216
214 ; 217
215 ; 218
216 ; 219
217 ; 220
218 ; 221
219 ; 222
220 ; 223
221 ; 224
222 ; 225
223 ; 226
224 ; 227
225 ; 228
226 ; 229
227 ; 230
228 ; 231
229 ; 232
230 ; 233
231 ; 234
232 ; 235
233 ; 236
234 ; 237
235 ; 238
236 ; 239
237 ; 240
238 ; 241
239 ; 242
240 ; 243
241 ; 244
242 ; 245
243 ; 246
244 ; 247
245 ; 248
246 ; 249
247 ; 250
248 ; 251
249 ; 252
250 ; 253
251 ; 254
252 ; 255
253 ; 256
254 ; 257
255 ; 258
256 ; 259
257 ; 260
258 ; 261
259 ; 262
260 ; 263
261 ; 264
262 ; 265
263 ; 266
264 ; 267
265 ; 268
266 ; 269
267 ; 270
268 ; 271
269 ; 272
270 ; 273
271 ; 274
272 ; 275
273 ; 276
274 ; 277
275 ; 278
276 ; 279
277 ; 280
278 ; 281
279 ; 282
280 ; 283
281 ; 284
282 ; 285
283 ; 286
284 ; 287
285 ; 288
286 ; 289
287 ; 290
288 ; 291
289 ; 292
290 ; 293
291 ; 294
292 ; 295
293 ; 296
294 ; 297
295 ; 298
296 ; 299
297 ; 300
298 ; 301
299 ; 302
300 ; 303
301 ; 304
302 ; 305
303 ; 306
304 ; 307
305 ; 308
306 ; 309
307 ; 310
308 ; 311
309 ; 312
310 ; 313
311 ; 314
312 ; 315
313 ; 316
314 ; 317
315 ; 318
316 ; 319
317 ; 320
318 ; 321
319 ; 322
320 ; 323
321 ; 324
322 ; 325
323 ; 326
324 ; 327
325 ; 328
326 ; 329
327 ; 330
328 ; 331
329 ; 332
330 ; 333
331 ; 334
332 ; 335
333 ; 336
334 ; 337
335 ; 338
336 ; 339
337 ; 340
338 ; 341
339 ; 342
340 ; 343
341 ; 344
342 ; 345
343 ; 346
344 ; 347
345 ; 348
346 ; 349
347 ; 350
348 ; 351
349 ; 352
350 ; 353
351 ; 354
352 ; 355
353 ; 356
354 ; 357
355 ; 358
356 ; 359
357 ; 360
358 ; 361
359 ; 362
360 ; 363
361 ; 364
362 ; 365
363 ; 366
364 ; 367
365 ; 368
366 ; 369
367 ; 370
368 ; 371
369 ; 372
370 ; 373
371 ; 374
372 ; 375
373 ; 376
374 ; 377
375 ; 378
376 ; 379
377 ; 380
378 ; 381
379 ; 382
380 ; 383
381 ; 384
382 ; 385
383 ; 386
384 ; 387
385 ; 388
386 ; 389
387 ; 390
388 ; 391
389 ; 392
390 ; 393
391 ; 394
392 ; 395
393 ; 396
394 ; 397
395 ; 398
396 ; 399
397 ; 400
398 ; 401
399 ; 402
400 ; 403
401 ; 404
402 ; 405
403 ; 406
404 ; 407
405 ; 408
406 ; 409
407 ; 410
408 ; 411
409 ; 412
410 ; 413
411 ; 414
412 ; 415
413 ; 416
414 ; 417
415 ; 418
416 ; 419
417 ; 420
418 ; 421
419 ; 422
420 ; 423
421 ; 424
422 ; 425
423 ; 426
424 ; 427
425 ; 428
426 ; 429
427 ; 430
428 ; 431
429 ; 432
430 ; 433
431 ; 434
432 ; 435
433 ; 436
434 ; 437
435 ; 438
436 ; 439
437 ; 440
438 ; 441
439 ; 442
440 ; 443
441 ; 444
442 ; 445
443 ; 446
444 ; 447
445 ; 448
446 ; 449
447 ; 450
448 ; 451
449 ; 452
450 ; 453
451 ; 454
452 ; 455
453 ; 456
454 ; 457
455 ; 458
456 ; 459
457 ; 460
458 ; 461
459 ; 462
460 ; 463
461 ; 464
462 ; 465
463 ; 466
464 ; 467
465 ; 468
466 ; 469
467 ; 470
468 ; 471
469 ; 472
470 ; 473
471 ; 474
472 ; 475
473 ; 476
474 ; 477
475 ; 478
476 ; 479
477 ; 480
478 ; 481
479 ; 482
480 ; 483
481 ; 484
482 ; 485
483 ; 486
484 ; 487
485 ; 488
486 ; 489
487 ; 490
488 ; 491
489 ; 492
490 ; 493
491 ; 494
492 ; 495
493 ; 496
494 ; 497
495 ; 498
496 ; 499
497 ; 500
498 ; 501
499 ; 502
500 ; 503
501 ; 504
502 ; 505
503 ; 506
504 ; 507
505 ; 508
506 ; 509
507 ; 510
508 ; 511
509 ; 512
510 ; 513
511 ; 514
512 ; 515
513 ; 516
514 ; 517
515 ; 518
516 ; 519
517 ; 520
518 ; 521
519 ; 522
520 ; 523
521 ; 524
522 ; 525
523 ; 526
524 ; 527
525 ; 528
526 ; 529
527 ; 530
528 ; 531
529 ; 532
530 ; 533
531 ; 534
532 ; 535
533 ; 536
534 ; 537
535 ; 538
536 ; 539
537 ; 540
538 ; 541
539 ; 542
540 ; 543
541 ; 544
542 ; 545
543 ; 546
544 ; 547
545 ; 548
546 ; 549
547 ; 550
548 ; 551
549 ; 552
550 ; 553
551 ; 554
552 ; 555
553 ; 556
554 ; 557
555 ; 558
556 ; 559
557 ; 560
558 ; 561
559 ; 562
560 ; 563
561 ; 564
562 ; 565
563 ; 566
564 ; 567
565 ; 568
566 ; 569
567 ; 570
568 ; 571
569 ; 572
570 ; 573
571 ; 574
572 ; 575
573 ; 576
574 ; 577
575 ; 578
576 ; 579
577 ; 580
578 ; 581
579 ; 582
580 ; 583
581 ; 584
582 ; 585
583 ; 586
584 ; 587
585 ; 588
586 ; 589
587 ; 590
588 ; 591
589 ; 592
590 ; 593
591 ; 594
592 ; 595
593 ; 596
594 ; 597
595 ; 598
596 ; 599
597 ; 600
598 ; 601
599 ; 602
600 ; 603
601 ; 604
602 ; 605
603 ; 606
604 ; 607
605 ; 608
606 ; 609
607 ; 610
608 ; 611
609 ; 612
610 ; 613
611 ; 614
612 ; 615
613 ; 616
614 ; 617
615 ; 618
616 ; 619
617 ; 620
618 ; 621
619 ; 622
620 ; 623
621 ; 624
622 ; 625
623 ; 626
624 ; 627
625 ; 628
626 ; 629
627 ; 630
628 ; 631
629 ; 632
630 ; 633
631 ; 634
632 ; 635
633 ; 636
634 ; 637
635 ; 638
636 ; 639
637 ; 640
638 ; 641
639 ; 642
640 ; 643
641 ; 644
642 ; 645
643 ; 646
644 ; 647
645 ; 648
646 ; 649
647 ; 650
648 ; 651
649 ; 652
650 ; 653
651 ; 654
652 ; 655
653 ; 656
654 ; 657
655 ; 658
656 ; 659
657 ; 660
658 ; 661
659 ; 662
660 ; 663
661 ; 664
662 ; 665
663 ; 666
664 ; 667
665 ; 668
666 ; 669
667 ; 670
668 ; 671
669 ; 672
670 ; 673
671 ; 674
672 ; 675
673 ; 676
674 ; 677
675 ; 678
676 ; 679
677 ; 680
678 ; 681
679 ; 682
680 ; 683
681 ; 684
682 ; 685
683 ; 686
684 ; 687
685 ; 688
686 ; 689
687 ; 690
688 ; 691
689 ; 692
690 ; 693
691 ; 694
692 ; 695
693 ; 696
694 ; 697
695 ; 698
696 ; 699
697 ; 700
698 ; 701
699 ; 702
700 ; 703
701 ; 704
702 ; 705
703 ; 706
704 ; 707
705 ; 708
706 ; 709
707 ; 710
708 ; 711
709 ; 712
710 ; 713
711 ; 714
712 ; 715
713 ; 716
714 ; 717
715 ; 718
716 ; 719
717 ; 720
718 ; 721
719 ; 722
720 ; 723
721 ; 724
722 ; 725
723 ; 726
724 ; 727
725 ; 728
726 ; 729
727 ; 730
728 ; 731
729 ; 732
730 ; 733
731 ; 734
732 ; 735
733 ; 736
734 ; 737
735 ; 738
736 ; 739
737 ; 740
738 ; 741
739 ; 742
740 ; 743
741 ; 744
742 ; 745
743 ; 746
744 ; 747
745 ; 748
746 ; 749
747 ; 750
748 ; 751
749 ; 752
750 ; 753
751 ; 754
752 ; 755
753 ; 756
754 ; 757
755 ; 758
756 ; 759
757 ; 760
758 ; 761
759 ; 762
760 ; 763
761 ; 764
762 ; 765
763 ; 766
764 ; 767
765 ; 768
766 ; 769
767 ; 770
768 ; 771
769 ; 772
770 ; 773
771 ; 774
772 ; 775
773 ; 776
774 ; 777
775 ; 778
776 ; 779
777 ; 780
778 ; 781
779 ; 782
780 ; 783
781 ; 784
782 ; 785
783 ; 786
784 ; 787
785 ; 788
786 ; 789
787 ; 790
788 ; 791
789 ; 792
790 ; 793
791 ; 794
792 ; 795
793 ; 796
794 ; 797
795 ; 798
796 ; 799
797 ; 800
798 ; 801
799 ; 802
800 ; 803
801 ; 804
802 ; 805
803 ; 806
804 ; 807
805 ; 808
806 ; 809
807 ; 810
808 ; 811
809 ; 812
810 ; 813
811 ; 814
812 ; 815
813 ; 816
814 ; 817
815 ; 818
816 ; 819
817 ; 820
818 ; 821
819 ; 822
820 ; 823
821 ; 824
822 ; 825
823 ; 826
824 ; 827
825 ; 828
826 ; 829
827 ; 830
828 ; 831
829 ; 832
830 ; 833
831 ; 834
832 ; 835
833 ; 836
834 ; 837
835 ; 838
836 ; 839
837 ; 840
838 ; 841
839 ; 842
840 ; 843
841 ; 844
842 ; 845
843 ; 846
844 ; 847
845 ; 848
846 ; 849
847 ; 850
848 ; 851
849 ; 852
850 ; 853
851 ; 854
852 ; 855
853 ; 856
854 ; 857
855 ; 858
856 ; 859
857 ; 860
858 ; 861
859 ; 862
860 ; 863
861 ; 864
862 ; 865
863 ; 866
864 ; 867
865 ; 868
866 ; 869
867 ; 870
868 ; 871
869 ; 872
870 ; 873
871 ; 874
872 ; 875
873 ; 876
874 ; 877
875 ; 878
876 ; 879
877 ; 880
878 ; 881
879 ; 882
880 ; 883
881 ; 884
882 ; 885
883 ; 886
884 ; 887
885 ; 888
886 ; 889
887 ; 890
888 ; 891
889 ; 892
890 ; 893
891 ; 894
892 ; 895
893 ; 896
894 ; 897
895 ; 898
896 ; 899
897 ; 900
898 ; 901
899 ; 902
900 ; 903
901 ; 904
902 ; 905
903 ; 906
904 ; 907
905 ; 908
906 ; 909
907 ; 910
908 ; 911
909 ; 912
910 ; 913
911 ; 914
912 ; 915
913 ; 916
914 ; 917
915 ; 918
916 ; 919
917 ; 920
918 ; 921
919 ; 922
920 ; 923
921 ; 924
922 ; 925
923 ; 926
924 ; 927
925 ; 928
926 ; 929
927 ; 930
928 ; 931
929 ; 932
930 ; 933
931 ; 934
932 ; 935
933 ; 936
934 ; 937
935 ; 938
936 ; 939
937 ; 940
938 ; 941
939 ; 942
940 ; 943
941 ; 944
942 ; 945
943 ; 946
944 ; 947
945 ; 948
946 ; 949
947 ; 950
948 ; 951
949 ; 952
950 ; 953
951 ; 954
952 ; 955
953 ; 956
954 ; 957
955 ; 958
956 ; 959
957 ; 960
958 ; 961
959 ; 962
960 ; 963
961 ; 964
962 ; 965
963 ; 966
964 ; 967
965 ; 968
966 ; 969
967 ; 970
968 ; 971
969 ; 972
970 ; 973
971 ; 974
972 ; 975
973 ; 976
974 ; 977
975 ; 978
976 ; 979
977 ; 980
978 ; 981
979 ; 982
980 ; 983
981 ; 984
982 ; 985
983 ; 986
984 ; 987
985 ; 988
986 ; 989
987 ; 990
988 ; 991
989 ; 992
990 ; 993
991 ; 994
992 ; 995
993 ; 996
994 ; 997
995 ; 998
996 ; 999
997 ; 1000

```

```

28     type: Sequelize.STRING      21           }
29         (30)                      22       },
30     },                           23       {
31     fatherBusinessAdd: {          24       freezeTableName: true
32         type: Sequelize.STRING   25     }
33         (30)                      26   );
34     },                           27
35     fatherOfficeNum: {           28 module.exports =
36         type: Sequelize.STRING    29     StudentFinalGrade;
37         (30)
38     },
39     motherName: {                1
40         type: Sequelize.STRING   2
41         (30)
42     },
43     motherAddress: {             3
44         type: Sequelize.STRING   4
45         (30)
46     },
47     motherEmail: {               5
48         type: Sequelize.STRING   6
49         (30)
50     },
51     motherOccupation: {          7
52         type: Sequelize.STRING   8
53         (30)
54     },
55     motherEmployer: {            9
56         type: Sequelize.STRING   10
57         (30)
58     },
59     motherBusinessAdd: {          11
60         type: Sequelize.STRING   12
61         (30)
62     },
63     motherOfficeNum: {           13
64         type: Sequelize.STRING   14
65         (30)
66     },
67     { freezeTableName: true }      15
68 );
69
70 module.exports = Student;

```

Listing 15: StudentGrades

```

const Sequelize = require("sequelize");
const sequelize = require("config/db/database");

const StudentGrades = sequelize.define(
  "student grades",
  {
    studentgradesID: {
      type: Sequelize.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    studsectID: {
      type: Sequelize.INTEGER
    },
    schoolYearID: {
      type: Sequelize.INTEGER
    },
    quarter: {
      type: Sequelize.ENUM,
      values: ["Q2", "Q3"],
      allowNull: false
    },
    grade: {
      type: Sequelize.FLOAT
    }
  },
  { freezeTableName: true }
);

```

Listing 14: StudentFinalGrade.js

```
1     const Sequelize = require("sequelize");
2     const sequelize = require("../config/db/database"); 1
3
4     const StudentFinalGrade = 2
5         sequelize.define( 3
6             "student final grade", 3
7             { 4
8                 finalGradeID: { 5
9                     type: Sequelize.INTEGER, 6
10                    primaryKey: true, 7
11                    autoIncrement: true 8
12                }, 9
13                schoolYearID: { 10
14                    type: Sequelize.INTEGER 11
15                }, 12
16                studsectID: { 13
17                    type: Sequelize.INTEGER 14
18                }, 15
19                grade: { 16
20                    type: Sequelize.FLOAT 17
```

Listing 15: StudentGrades.js

```
const Sequelize = require("sequelize");
const sequelize = require("../config/db/database");

const StudentGrades = sequelize
  .define(
    "student_grades",
    {
      studentgradesID: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true
      },
      studsectID: {
        type: Sequelize.INTEGER
      },
      schoolYearID: {
        type: Sequelize.INTEGER
      },
      quarter: {
        type: Sequelize.ENUM("Q1",
          "Q2", "Q3", "Q4")
      },
      grade: {
        type: Sequelize.FLOAT
      }
    },
    { freezeTableName: true }
  );

module.exports = StudentGrades;
```

Listing 16: StudentSection.js

```
const Sequelize = require("sequelize");
const sequelize = require("../config/db/database");

const StudentSection =
  sequelize.define(
    "student section",
    {
      studsectID: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true
      },
      studentID: {
        type: Sequelize.INTEGER
      },
      sectionID: {
        type: Sequelize.INTEGER
      },
    }
  );

```



```

14      },
15      name: {
16        type: Sequelize.STRING
17        (50)
18      },
19      componentID: {
20        type: Sequelize.INTEGER
21      },
22      compWeight: {
23        type: Sequelize.FLOAT
24      },
25      quarter: {
26        type: Sequelize.ENUM("Q1" 1
27          , "Q2", "Q3", "Q4")
28      },
29      { freezeTableName: true }
30    );
31 module.exports = Subcomponent;

```

Listing 20: Subject.js

```

1 const Sequelize = require(
2   "sequelize");
3 const sequelize = require("../
4   config/db/database");
5 const Subject = sequelize.
6   define(
7     "subject",
8     {
9       subjectID: {
10         type: Sequelize.INTEGER,
11         primaryKey: true,
12         autoIncrement: true
13     },
14       subjectCode: {
15         type: Sequelize.STRING
16     },
17       subjectName: {
18         type: Sequelize.STRING
19     },
20       archived: {
21         type: Sequelize.BOOLEAN,
22         defaultValue: 0
23     },
24       subjectType: {
25         type: Sequelize.ENUM(
26           "N",
27           "K1",
28           "K2",
29           "G1",
30           "G2",
31           "G3",
32           "G4",
33           "G5",
34           "G6",
35           "G7",
36           "G8",
37           "G9",
38           "G10",
39           "CORE",
40           "ACAD-APPLIED",
41           "ACAD-SPECIALIZED",
42           "TVL-APPLIED",
43           "TVL-SPECIALIZED",
44           "SPORTS-APPLIED",
45           "SPORTS-SPECIALIZED",
46           "AAD-APPLIED",
47           "AAD-SPECIALIZED"
48         )
49     },
50     { freezeTableName: true }
51   );
52 module.exports = Subject;

```

Listing 21: SubjectSection.js

```

const Sequelize = require(
  "sequelize");
const sequelize = require("../
  config/db/database");

const SubjectSection =
  sequelize.define(
  "subject section",
  {
    subsectID: {
      type: Sequelize.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    subjectID: {
      type: Sequelize.INTEGER
    },
    sectionID: {
      type: Sequelize.INTEGER
    },
    teacherID: {
      type: Sequelize.INTEGER
    },
    schoolYearID: {
      type: Sequelize.INTEGER
    },
    subjectType: {
      type: Sequelize.ENUM(
        "NON_SHS", "1ST_SEM", "
        2ND_SEM")
    },
    classRecordID: {
      type: Sequelize.INTEGER
    },
    { freezeTableName: true }
  );
module.exports = SubjectSection;

```

Listing 22: SubjectSectionStudent.js

```

const Sequelize = require(
  "sequelize");
const sequelize = require("../
  config/db/database");

const SubjectSectionStudent =
  sequelize.define(
  "subject section student",
  {
    subsectstudID: {
      type: Sequelize.INTEGER,
      primaryKey: true,
      autoIncrement: true
    },
    studsectID: {

```

```

13         type: Sequelize.INTEGER 14
14     },
15     subsectID: {
16         type: Sequelize.INTEGER 17
17     }
18 },
19 {
20     freezeTableName: true 21
21 }
22 );
23
24 module.exports =
25     SubjectSectionStudent;

```

Listing 23: SubmissionDeadline.js

```

1 const Sequelize = require(" 3
2     sequelize");
3 const sequelize = require("../ 4
4     config/db/database");
5
6 const SubmissionDeadline = 7
7     sequelize.define( 8
8         "submission deadline", 9
9             {
10                 deadlineID: { 11
11                     type: Sequelize.INTEGER, 12
12                     primaryKey: true, 13
13                     autoIncrement: true
14                 },
15                 teacherID: { 14
16                     type: Sequelize.INTEGER
17                 },
18                 dateSet: { 15
19                     type: Sequelize.DATE
20                 },
21                 deadline: { 16
22                     type: Sequelize.DATE
23                 },
24                 isActive: { 17
25                     type: Sequelize.BOOLEAN
26                 }
27             },
28         {
29             freezeTableName: true
30         }
31     );
32
33 module.exports =
34     SubmissionDeadline;

```

Listing 24: Teacher.js

```

1 const Sequelize = require(" 7
2     sequelize");
3 const sequelize = require("../ 8
4     config/db/database"); 10
5
6 const Teacher = sequelize. 11
7     define( 12
8         "teacher", 13
9             {
10                 teacherID: { 14
11                     type: Sequelize.INTEGER, 12
12                     primaryKey: true, 13
13                     autoIncrement: true
14                 },
15                 accountID: { 16
16                     type: Sequelize.INTEGER, 17
17                     primaryKey: true, 18
18                     autoIncrement: true
19                 }
20             },
21             accountID: { 21
22                 type: Sequelize.INTEGER
23             }
24         }
25     );
26
27 module.exports =
28     Teacher;

```

```

    },
    isAdviser: {
        type: Sequelize.BOOLEAN
    }
},
{
    freezeTableName: true
};

module.exports = Teacher;

```

Listing 25: TeacherSection.js

```

1 const Sequelize = require(" 2
2     sequelize");
3 const sequelize = require("../ 4
4     config/db/database");
5
6 const TeacherSection = 7
7     sequelize.define( 8
8         "teacher section", 9
9             {
10                 teachersectionID: { 11
11                     type: Sequelize.INTEGER,
12                     primaryKey: true,
13                     autoIncrement: true
14                 },
15                 sectionID: { type:
16                     Sequelize.INTEGER }, 17
17                 schoolYearID: { type:
18                     Sequelize.INTEGER }, 19
19                 teacherID: { type:
20                     Sequelize.INTEGER }
21             },
22             {
23                 freezeTableName: true
24             }
25         );
26
27 module.exports = TeacherSection
28     ;

```

Listing 26: UserAccount.js

```

1 const Sequelize = require(" 2
2     sequelize");
3 const sequelize = require("../ 4
4     config/db/database");
5
6 const UserAccount = sequelize. 7
7     define(
8         "user account", {
9             accountID: { 10
10                 type: Sequelize.INTEGER,
11                 primaryKey: true,
12                 autoIncrement: true
13             },
14             email: { 15
15                 type: Sequelize.STRING
16             },
17             password: { 18
18                 type: Sequelize.STRING,
19                 BINARY
20             },
21             isActive: { 22
22                 type: Sequelize.BOOLEAN
23             },
24             position: { 25
25                 type: Sequelize.BOOLEAN
26             }
27         }
28     );
29
30 module.exports =
31     UserAccount;

```

```

23     firstName: { 81           type: Sequelize.STRING
24       type: Sequelize.STRING 82           (50)
25   }, 83           },
26     lastName: { 84           type: Sequelize.STRING
27       type: Sequelize.STRING 85           (50)
28   }, 86           },
29     middleName: { 87           type: Sequelize.STRING
30       type: Sequelize.STRING 88           (50)
31   }, 89           },
32     suffix: { 90           type: Sequelize.STRING
33       type: Sequelize.STRING 91           (10)
34   }, 92           },
35     nickname: { 93           type: Sequelize.STRING
36       type: Sequelize.STRING 94           (50)
37   }, 95           },
38     imageUrl: { 96           type: Sequelize.STRING
39       type: Sequelize.STRING 97           (15)
40   }, 98           },
41     contactNum: { 99           type: Sequelize.STRING
42       type: Sequelize.STRING 100          (11)
43   }, 100          },
44     address: { 101          type: Sequelize.STRING
45       type: Sequelize.STRING 101          (50)
46   },
47     province: { 1           type: Sequelize.STRING
48       type: Sequelize.STRING 2           (20)
49   },
50     city: { 1           type: Sequelize.STRING
51       type: Sequelize.STRING 2           (25)
52   },
53     region: { 3           type: Sequelize.STRING
54       type: Sequelize.STRING 4           (15)
55   },
56     zipcode: { 5           type: Sequelize.STRING(4)
57   },
58     civilStatus: { 6           type: Sequelize.ENUM(
59       "SINGLE", "MARRIED", "7
60       "WIDOWED", "OTHERS") 8
61   },
62     sex: { 9           type: Sequelize.ENUM("M",
63       "F") 10
64   },
65     citizenship: { 11          type: Sequelize.STRING
66       type: Sequelize.STRING 12          (20)
67   },
68     birthDate: { 13          type: Sequelize.DATE
69   },
70     birthPlace: { 14          type: Sequelize.STRING
71       type: Sequelize.STRING 15          (25)
72   },
73     religion: { 16          type: Sequelize.STRING
74       type: Sequelize.STRING 17          (20)
75   },
76     emergencyName: { 18          type: Sequelize.STRING
77       type: Sequelize.STRING 18          (30)
78   },
79     emergencyAddress: {

```

Listing 27: erd.js

```

// Import Models (Database
  table names)
const UserAccount = require(
  "../models/UserAccount");
const Nonacademic = require(
  "../models/Nonacademic");
const Student = require("../
  models/Student");
const ParentGuardian = require(
  "../models/ParentGuardian");
const Teacher = require("../
  models/Teacher");
const Grade = require("../
  models/Grade");
const Section = require("../
  models/Section");
const Subject = require("../
  models/Subject");
const TeacherSection = require(
  "../models/TeacherSection");
const AttendanceLog = require(
  "../models/AttendanceLog");
const SubmissionDeadline =
  require("../models/
  SubmissionDeadline");
const SubjectSection = require(
  "../models/SubjectSection");
const SubjectSectionStudent =
  require("../models/
  SubjectSectionStudent");
const Component = require("../
  models/Component");
const StudentWeightedScore =
  require("../models/
  StudentWeightedScore");
const ClassRecord = require(
  "../models/ClassRecord");
const Subcomponent = require(
  "../models/Subcomponent");

```

```

19   const AccountNotice = require('66
20     './models/AccountNotice'); 67
21   const SchoolYear = require("./
22     models/SchoolYear");
23   const StudentSection = require("68
24     ./models/StudentSection")69
25   const StudentSubjectGrades = 70
26     require("./models/
27     StudentSubjectGrades"); 72
28   const ActivityLog = require("73
29     ./models/ActivityLog");
30   const LogDetails = require("..74
31     ./models/LogDetails");
32   const ClassRecordStatus = 75
33     require("./models/
34     ClassRecordStatus");
35   const StudentGrades = require("76
36     ./models/StudentGrades");
37   const StudentFinalGrade = 77
38     require("./models/
39     StudentFinalGrade");
40   const utils = require("../utils82
41     ");
42
43   const useraccountERD = async 84
44     function() {
45       // user account ERD
46
47       await UserAccounthasOne( 85
48         Nonacademic, {
49           foreignKey: "accountID"
50         });
51       await UserAccounthasOne( 86
52         Student, {
53           foreignKey: "accountID"
54         });
55       await UserAccounthasOne( 87
56         ParentGuardian, {
57           foreignKey: "accountID"
58         });
59       await UserAccounthasOne( 88
60         Teacher, {
61           foreignKey: "accountID"
62         });
63       await UserAccount.sync({ 89
64         force: false
65       }).then(() => { 90
66         console.log("user account
67           table created/synced");
68         Nonacademic.sync({ 91
69           force: false
70       }).then(() => { 92
71         console.log("nonacademic
72           table created/synced");
73       });
74       Student.sync({ 93
75         force: false
76     }).then(() => { 94
77       console.log("student
78         table created/synced");
79     });
80     ParentGuardian.sync({ 95
81       force: false
82   }).then(() => { 96
83     console.log("parent
84       guardian table created
85       /synced");
86   });
87   Teacher.sync({ 97
88     force: false
89   }).then(() => { 98
90     console.log("teacher
91       table created/synced");
92   });
93 });
94
95 const subjectsectionERD = async
96   function() {
97   //subject section erd
98   await SchoolYear.sync({
99     force: false
100 });
101
102 await Section.hasMany(
103   SubjectSection, {
104     foreignKey: "sectionID"
105   });
106
107 await Teacher.hasMany(
108   SubjectSection, {
109     foreignKey: "teacherID"
110   });
111
112 await Subject.hasMany(
113   SubjectSection, {
114     foreignKey: "subjectID"
115   });
116
117 await StudentSection.hasMany(
118   SubjectSectionStudent, {
119     foreignKey: "studsectID"
120   });
121
122 await SubjectSection.hasMany(
123   SubjectSectionStudent, {
124     foreignKey: "subsectID"
125   });
126
127 await SchoolYear.hasMany(
128   SubjectSection, {
129     foreignKey: "schoolYearID"
130   });
131
132 await Subject.sync({
133   force: false
134 }).then(async function() {
135   await Teacher.sync({
136     force: false
137   }).then(async function() {
138     await Section.sync({
139       force: false
140     }).then(async function() {
141       await SubjectSection.
142         sync({
143           force: false
144         }).then(async function()
145           () {
146             await
147               SubjectSectionStudent
148                 .sync({
149                   force: false
150                 }).then(async
151                   function() {
152                     console.log(""
153                       subject section
154                         student created/
155                           synced");
156                   });
157                 });
158               });
159             });
160           });
161         });
162       });
163     });
164   });
165
166 const studentsectionERD = async
167   function() {
168   await Student.hasMany(
169

```

```

120     StudentSection, { 170
121       foreignKey: "studentID" 171
122   });
123   await SchoolYear.hasMany( 172
124     StudentSection, { 173
125       foreignKey: "schoolYearID" 176
126   });
127   await Section.hasMany( 174
128     StudentSection, { 175
129       foreignKey: "sectionID" 177
130   });
131   await StudentSection.sync({ 178
132     force: false 179
133 }).then(() => { 180
134   console.log("student 181
135     section table created/ 180
136     synced"); 181
137   });
138   const componentERD = async 183
139     function() { 184
140       // component weight ERD 184
141       await Subject.hasMany( 185
142         Component, { 185
143           foreignKey: "subjectID" 186
144         });
145       await Component.sync({ 187
146         force: false 188
147     }).then(() => { 189
148       console.log("component 189
149         table created/synced"); 190
150     });
151   );
152   const gradeERD = async function 192
153     () { 192
154       // grade ERD 193
155       await ClassRecord.sync({ 193
156         force: false 194
157     }).then(async function() { 195
158       await ClassRecord.hasMany(196
159         StudentWeightedScore, { 196
160           foreignKey: " 197
161             classRecordID" 197
162         });
163       await SubjectSectionStudent198
164         .hasMany( 199
165           StudentWeightedScore, { 199
166             foreignKey: " 200
167               subsectstudID" 200
168         });
169       await ClassRecord.hasMany(202
170         StudentSubjectGrades, { 202
171           foreignKey: " 203
172             classRecordID" 203
173         });
174       await ClassRecord.hasOne(204
175         ClassRecordStatus, { 204
176           foreignKey: " 205
177             classRecordID" 205
178         });
179       await ClassRecord.hasMany(210
180         ClassRecordStatus, { 210
181           foreignKey: " 211
182             classRecordID" 211
183         });
184       await ClassRecord.hasMany(213
185         Grade, { 213
186           foreignKey: " 214
187             classRecordID" 214
188         });
189     });
190   );
191   const adviserERD = async
192     function() {
193       // adviser ERD
194       await SchoolYear.hasMany(
195         TeacherSection, {
196           foreignKey: "schoolYearID"
197         });
198       await Section.hasMany(
199         TeacherSection, {
200           foreignKey: "sectionID"
201         });
202     );
203   );
204   await ClassRecord.hasMany(
205     Subcomponent, {
206       foreignKey: "
207         classRecordID"
208     });
209   await Component.hasMany(
210     Subcomponent, {
211       foreignKey: "componentID"
212     });
213   await Subcomponent.sync({
214     force: false
215   }).then(async function() {
216     await
217       StudentWeightedScore.
218         sync({
219           force: false
220         });
221   ).then(async function() {
222     await
223       StudentSubjectGrades
224         .sync({
225           force: false
226         }).then(async function() {
227           await Subcomponent.
228             hasMany(Grade, {
229               foreignKey: "
230                 subcomponentID"
231             });
232           await Component.
233             hasMany(Grade, {
234               foreignKey: "
235                 componentID"
236             });
237           await
238             SubjectSectionStudent
239               .hasMany(Grade, {
240                 foreignKey: "
241                   subsectstudID"
242             });
243           await Grade.sync({
244             force: false
245           }).then(async
246             function() {
247               console.log("grade
248                 table created/
249                   synced");
250             });
251           await
252             ClassRecordStatus.
253               sync({
254                 force: false
255               }).then(async () => {
256                 console.log("class
257                   record status
258                     created/synced")
259               );
260             );
261           );
262         );
263       );
264     );
265   );
266   );
267   );
268   );
269   );

```

```

217     });
218     await Teacher.hasMany(
219       TeacherSection, {
220         foreignKey: "teacherID"
221     });
222     await TeacherSection.sync({
223       force: false
224     }).then(() => {
225       console.log("adviser table created/synced");
226     });
227   };
228   const attendanceLogERD = async
229     function() {
230       // attendance log ERD
231       await SchoolYear.hasMany(
232         AttendanceLog, {
233           foreignKey: "schoolYearID"
234         });
235       await Student.hasMany(
236         AttendanceLog, {
237           foreignKey: "studentID"
238         });
239       await Teacher.hasMany(
240         AttendanceLog, {
241           foreignKey: "teacherID"
242         });
243       await AttendanceLog.sync({
244         force: false
245     }).then(() => {
246       console.log("attendance log table created/synced");
247     });
248   };
249   const submissionDeadlineERD =
250     async function() {
251       // submission deadline ERD
252       await Teacher.hasMany(
253         SubmissionDeadline, {
254           foreignKey: "teacherID"
255         });
256       await SubmissionDeadline.sync({
257         force: false
258     }).then(() => {
259       console.log("submission deadline table created/
260         synced");
261     });
262   };
263   const studentnoticeERD = async
264     function() {
265       // student notice ERD
266       await Student.hasOne(
267         AccountNotice, {
268           foreignKey: "studentID"
269         });
270       await AccountNotice.sync({
271         force: false
272     }).then(() => {
273       console.log("Account notice table created/synced");
274     });
275   };
276   const activitylogERD = async
277     function() {
278       await ClassRecord.hasMany(
279         ActivityLog, {
280           foreignKey: "classRecordID"
281         });
282     };
283   const studentgradesERD = async
284     function() {
285       await SchoolYear.hasMany(
286         StudentGrades, {
287           foreignKey: "schoolYearID"
288         });
289       await StudentSection.hasMany(
290         StudentGrades, {
291           foreignKey: "studsectID"
292         });
293       await StudentSection.hasMany(
294         StudentFinalGrade, {
295           foreignKey: "schoolYearID"
296         });
297       await StudentFinalGrade.sync({
298         force: false
299     }).then(() => {
300       console.log("Student grades created/synced");
301     });
302   };
303   const hooks = async function()
304     {
305       ClassRecordStatus.addHook(
306         "afterUpdate",
307         async (classrecordstatus,
308           options) => {
309           const {
310             position,
311             classRecordID,
312             type,
313             accountID,
314             sectionID,
315             subjectID,
316             oldVal,
317             newVal,
318             quarter
319           } = options;
320           const name = await utils.
321             getAccountName(
322               accountID);
323           const section = await
324             utils.getSectionName(
325               sectionID);
326           const subject = await
327             utils.getSubjectCode(
328               subjectID);

```

```

317     const oldValText =      379
318       oldVal == "L"        380
319         ? "Locked"
320           : oldVal == "E"
321             ? "Encoding"
322               : oldVal == "D"
323                 ? "For deliberation"
324                   : "Posted";
325     const newValText =    382
326       newVal == "L"      383
327         ? "Locked"
328           : newVal == "E"
329             ? "Encoding"
330               : newVal == "D"
331                 ? "For deliberation"
332                   : "Posted";
333     ActivityLog.create({
334       type,
335         classRecordID,
336       position,
337         name,
338         section,
339         subject,
340         quarter,
341         timestamp: new Date()
342   }).then(async al => {
343     LogDetails.create({
344       logID: al.logID,
345         description: 'Changed
346           class record
347             status from \'${oldValText}\'
348               to \'${newValText}\'
349             ');
350   );
351   Subcomponent.addHook("        404
352     afterDestroy", async (
353       subcomponent, options) =405
354     {
355       const {
356         showLog,
357         position,
358         type,
359         accountID,
360         sectionID,
361         subjectID,
362         quarter,
363         subcompName,
364         componentName,
365         classRecordID
366     } = options;
367     if (showLog) {
368       const name = await utils.
369         getAccountName(
370           accountID);
371       const section = await
372         utils.getSectionName(
373           sectionID);
374       const subject = await
375         utils.getSubjectCode(
376           subjectID);
377       ActivityLog.create({        424
378         type,
379         classRecordID,
380         position,
381         name,
382         section,
383         subject,
384         quarter,
385         timestamp: new Date()
386   }).then(async al => {
387     LogDetails.create({        431
388       logID: al.logID,
389         description: 'Deleted
390           subcomponent '${subcompName}'
391             under '${componentName}'
392             });
393   });
394 } else {
395   // Do nothing
396 }
397 );
398 Subcomponent.addHook("        381
399     afterCreate", async (
400       subcomponent, options) =>
401     {
402       const {
403         showLog,
404         position,
405         type,
406         accountID,
407         sectionID,
408         subjectID,
409         quarter,
410         subcompName,
411         componentName,
412         classRecordID
413     } = options;
414     if (showLog) {
415       const name = await utils.
416         getAccountName(
417           accountID);
418       const section = await
419         utils.getSectionName(
420           sectionID);
421       const subject = await
422         utils.getSubjectCode(
423           subjectID);
424       ActivityLog.create({        425
425         type,
426         classRecordID,
427         position,
428         name,
429         section,
430         subject,
431         quarter,
432         timestamp: new Date()
433   }).then(async al => {
434     LogDetails.create({        432
435       logID: al.logID,
436         description: 'Added
437           subcomponent '${subcompName}'
438             under '${componentName}'
439             });
440   });
441 } else {
442   // Do nothing
443 }
444 );
445 Subcomponent.addHook("        382
446     afterUpdate", async (
447       subcomponent, options) =>
448     {
449       const {
450         showLog,
451         position,
452         type,
453         accountID,
454         sectionID,
455         subjectID,

```

```

433     quarter , utils.getSectionName(         488
434     subcompName , sectionID);
435     componentName , const subject = await
436     classRecordID , utils.getSubjectCode(
437     oldVal , subjectID);
438     newVal , ActivityLog.create({
439     oldName , type ,
440     newName , classRecordID ,
441 } = options; position ,
442 if (showLog) { name ,
443     const name = await utils section ,
444     getAccountName( timestamp: new Date()
445     accountID); }
446     const section = await utils.getSectionName()
447     getSectionName( )
448     const subject = await utils.getSubjectCode()
449     subjectID); subject ,
450     ActivityLog.create({ timestamp: new Date()
451     type , 500
452     classRecordID , 501
453     position , 501
454     name , 502
455     section , 503
456     subject , 504
457     quarter , 505
458     timestamp: new Date() 506
459 ).then(async al => { 507
460     LogDetails.create({ 509
461     logID: al.logID, 510
462     description: 'Updated
463     subcomponent '$510
464     subcompName}', ${511
465     oldName != newName? 512
466     ? '(changed name
467     from ${oldName} 513
468     ) to ${newName} 514
469     )' 515
470     : '' 516
471 } ${517
472     oldVal != newVal? 519
473     ? '(from ${oldVal}
474     )% to ${newVal}) 520
475     %' : '' 520
476     } under ${521
477     componentName}' 521
478     ); 521
479 } else { 522
480     // Do nothing 522
481 } ); 523
482
483 ClassRecord.addHook(" 525
484     afterUpdate", async ( 526
485     classrecord, options) =>527
486     const { 528
487     showLog, 529
488     position, 530
489     type, 531
490     accountID, 532
491     sectionID, 533
492     subjectID, 534
493     quarter, 535
494     classRecordID, 535
495     oldVal, 536
496     newVal 537
497 } = options;
498 if (showLog) { 538
499     const name = await utils
500     getAccountName( 539
501     accountID); 540
502     const section = await 541
503     utils.getSectionName( 542
504     sectionID);
505     const subject = await
506     utils.getSubjectCode(
507     subjectID);
508     ActivityLog.create({
509     type ,
510     classRecordID ,
511     position ,
512     name ,
513     section ,
514     subject ,
515     quarter ,
516     timestamp: new Date()
517 }).then(async al => {
518     LogDetails.create({
519     logID: al.logID,
520     description: 'Updated
521     subcomponent '$522
522     subcompName}', ${523
523     oldName != newName? 524
524     ? '(changed name
525     from ${oldName} 525
526     ) to ${newName} 526
527     )' 527
528 } ${529
529     oldValue = -1;
530     newValue = value;
531     LogDetails.create({
532     logID,
533     description: "Grade
534     added",
535     student,
536     component,
537     subcomponent,
538     oldValue,
539     newValue
540     });
541 } else {
542 });
543
544 Grade.addHook("afterDestroy", 545
545     async (grade, options) => 546
546     {
547     const {
548     showLog,
549     logID,
550     subsectstudID,
551     componentID,
552     value
553     } = options;
554     if (showLog) {
555     const student = await
556     utils.getStudentNameBySubsectstudID
557     (subsectstudID);
558     const component = await
559     utils.getComponentName
560     (componentID);
561     const subcomponent =
562     await utils.getSubcomponentName(
563     subcompID);
564     const oldValue = -1;
565     const newValue = value;
566     LogDetails.create({
567     logID,
568     description: "Grade
569     added",
570     student,
571     component,
572     subcomponent,
573     oldValue,
574     newValue
575     });
576 } else {
577 });
578 });
579
580 Grade.addHook("afterDestroy", 581
581     async (grade, options) => 582
582     {
583     const {
584     showLog,
585     logID,
586     subsectstudID,
587     componentID,
588     value
589     } = options;
590     if (showLog) {
591     const student = await
592     utils.getStudentNameBySubsectstudID
593     (subsectstudID);
594     const component = await
595     utils.getComponentName
596     (componentID);
597     const subcomponent =
598     await utils.getSubcomponentName(
599     subcompID);
600     const oldValue = -1;
601     const newValue = value;
602     LogDetails.create({
603     logID,
604     description: "Grade
605     added",
606     student,
607     component,
608     subcomponent,
609     oldValue,
610     newValue
611     });
612 } else {
613 });
614 });
615
616 Grade.addHook("afterDestroy", 617
617     async (grade, options) => 618
618     {
619     const {
620     showLog,
621     logID,
622     subsectstudID,
623     componentID,
624     value
625     } = options;
626     if (showLog) {
627     const student = await
628     utils.getStudentNameBySubsectstudID
629     (subsectstudID);
630     const component = await
631     utils.getComponentName
632     (componentID);
633     const subcomponent =
634     await utils.getSubcomponentName(
635     subcompID);
636     const oldValue = -1;
637     const newValue = value;
638     LogDetails.create({
639     logID,
640     description: "Grade
641     added",
642     student,
643     component,
644     subcomponent,
645     oldValue,
646     newValue
647     });
648 } else {
649 });
650 });
651
652 Grade.addHook("afterDestroy", 653
653     async (grade, options) => 654
654     {
655     const {
656     showLog,
657     logID,
658     subsectstudID,
659     componentID,
660     value
661     } = options;
662     if (showLog) {
663     const student = await
664     utils.getStudentNameBySubsectstudID
665     (subsectstudID);
666     const component = await
667     utils.getComponentName
668     (componentID);
669     const subcomponent =
670     await utils.getSubcomponentName(
671     subcompID);
672     const oldValue = -1;
673     const newValue = value;
674     LogDetails.create({
675     logID,
676     description: "Grade
677     added",
678     student,
679     component,
680     subcomponent,
681     oldValue,
682     newValue
683     });
684 } else {
685 });
686 });
687

```

Listing 28: passport.js

```
const passportJWT = require("passport-jwt");
const key = require("../config/key");
const UserAccount = require("../models/UserAccount");

let ExtractJWT = passportJWT.ExtractJwt;
let JwtStrategy = passportJWT.Strategy;
let jwtOptions = {};

jwtOptions.jwtFromRequest = ExtractJWT.fromAuthHeaderAsBearerToken();
jwtOptions.secretOrKey = key.secretKey;

module.exports = passport => {
  passport.use(
    "jwt",
    new JwtStrategy(jwtOptions,
      function(jwt_payload, next) {
        UserAccount.findOne({
          where: { email: jwt_payload.email } })
          .then(user => {
            if (user) {
              return next(null, user);
            }
            return next(null, false);
          })
          .catch(err => console.log(err));
    })
  );
  passport.use(
    "admin",
    new JwtStrategy(jwtOptions,
      (jwt_payload, next) =>
      {
        UserAccount.findOne({
```

```

32          where: { email:         90
33              jwt_payload.email,
34              position: "0" }      91
35      ).then((user, err) => { 92
36          if (user) {
37              next(null, user);
38          } else {
39              next(null, false); 93
40          }
41      );
42          97
43      passport.use(          98
44          "director",        100
45          new JwtStrategy(jwtOptions101
46              (jwt_payload, next) =>102
47                  {
48                      103
49                      UserAccount.findOne({ 104
50                          where: { email: 105
51                              jwt_payload.email,
52                              position: "1" } 106
53                      ).then((user, err) => {107
54                          if (user) {
55                              next(null, user);
56                          } else {
57                              next(null, false); 108
58                          }
59                      );
60                      111
61                  );
62          113
63      passport.use(          114
64          "registrar",        115
65          new JwtStrategy(jwtOptions116
66              (jwt_payload, next) =>117
67                  {
68                      118
69                      UserAccount.findOne({ 119
70                          where: { email: 120
71                              jwt_payload.email,
72                              position: "2" } 121
73                      ).then((user, err) => {122
74                          if (user) {
75                              next(null, user);
76                          } else {
77                              next(null, false); 123
78                          }
79                      );
80          127
81      passport.use(          129
82          "teacher",          130
83          new JwtStrategy(jwtOptions131
84              (jwt_payload, next) =>132
85                  {
86                      133
87                      UserAccount.findOne({ 134
88                          where: { email: 135
89                              jwt_payload.email,
90                              position: "3" } 136
91                      ).then((user, err) => {137
92                          if (user) {
93                              next(null, user); 1
94                          } else {
95                              next(null, false); 2
96                          }
97                      );
98                  );
99          132
100      passport.use(
101          "student",
102          new JwtStrategy(jwtOptions103
103              (jwt_payload, next) =>
104                  {
105                      UserAccount.findOne({
106                          where: { email:
107                              jwt_payload.email,
108                              position: "4" } 109
109                      ).then((user, err) => {
110                          if (user) {
111                              next(null, user);
112                          } else {
113                              next(null, false); 114
114                          }
115                      );
116          119
117      passport.use(
118          "guardian",
119          new JwtStrategy(jwtOptions120
120              (jwt_payload, next) =>
121                  {
122                      UserAccount.findOne({
123                          where: { email:
124                              jwt_payload.email,
125                              position: "5" } 126
126                      ).then((user, err) => {
127                          if (user) {
128                              next(null, user);
129                          } else {
130                              next(null, false); 131
131                          }
132                      );
133          137
134      passport.use(
135          "cashier",
136          new JwtStrategy(jwtOptions137
137              (jwt_payload, next) =>
138                  {
139                      UserAccount.findOne({
140                          where: { email:
141                              jwt_payload.email,
142                              position: "6" } 143
143                      ).then((user, err) => {
144                          if (user) {
145                              next(null, user);
146                          } else {
147                              next(null, false); 148
148                          }
149                      );
150                  );
151          155
152      );
153  );
154  );
155  );
156  );
157  );
158  );
159  );
160  );
161  );
162  );
163  );
164  );
165  );
166  );
167  );
168  );
169  );
170  );
171  );
172  );
173  );
174  );
175  );
176  );
177  );
178  );
179  );
180  );
181  );
182  );
183  );
184  );
185  );
186  );
187  );
188  );
189  );

```

Listing 29: admin.js

```

const express = require("express");
const router = express.Router();
const UserAccount = require("../models/UserAccount");
const Nonacademic = require("../models/Nonacademic");
const Teacher = require("../models/Teacher");

```

```

6   const Student = require("../models/Student");
7   const ParentGuardian = require("../models/ParentGuardian");
8   const Grade = require("../models/Grade");
9   const Subject = require("../models/Subject");
10  const Section = require("../models/Section");
11  const bcrypt = require("bcryptjs");
12  const passport = require("passport");
13  const Sequelize = require("sequelize");
14  const Op = Sequelize.Op;
15
16 // Input Validation
17 const validateCreateAccount = require("../validation/createaccount");
18 const validateEditProfileNonacademic = require("../validation/editprofilenonacademic");
19 // Import Utility Functions
20 const utils = require("../utils");
21
22 // @route POST api/admin/updateprofile
23 // @desc Update profile
24 // @access Private
25
26 router.post(
27   "/updateprofile",
28   passport.authenticate("admin",
29     {
30       session: false
31     },
32     (req, res) => {
33       const {
34         firstName,
35         lastName,
36         middleName,
37         suffix,
38         nickname,
39         contactNum,
40         address,
41         province,
42         city,
43         region,
44         zipcode,
45         civilStatus,
46         sex,
47         citizenship,
48         birthDate,
49         birthPlace,
50         religion,
51         emergencyName,
52         emergencyAddress,
53         emergencyTelephone,
54         emergencyCellphone,
55         emergencyEmail,
56         emergencyRelationship
57       } = req.body;
58
59       const {
60         errors,
61         isValid
62       } =

```

```

validateEditProfileNonacademic(req.body);

if (!isValid) {
  return res.status(400).json(errors);
}

UserAccount.findOne({
  where: {
    accountID: req.user.accountID
  }
}).then(user => {
  if (user) {
    user
      .update({
        firstName,
        lastName,
        middleName,
        suffix,
        nickname,
        contactNum,
        address,
        province,
        city,
        region,
        zipcode,
        civilStatus,
        sex,
        citizenship,
        birthDate,
        birthPlace,
        religion,
        emergencyName,
        emergencyAddress,
        emergencyTelephone,
        emergencyCellphone,
        emergencyEmail,
        emergencyRelationship
      })
      .then(user2 => {
        res.status(200).json({
          msg: "Profile updated successfully!"
        });
      });
  }
});

// @route POST api/admin/createaccount
// @desc Create an account for Director, Registrar, Teacher, Student, and Parent/Guardian
// @access Private

router.post(
  "/createaccount",
  passport.authenticate("admin",
    {
      session: false
    }),
  (req, res) => {
    const {
      email,
      firstName,
      middleName,
      lastName,

```

```

125      position          178      emergencyRelationship
126    } = req.body;           179      : na
127  const {                  180  }) .then(user2 =>
128    errors,                181      {
129    isValid           180      if (position ==
130  } = validateCreateAccount( 181      true ||
131    req.body);           181      position ==
132      if (!isValid) {     182      2 ||
133        res.status(400).json(
134          errors);         182      position ==
135      } else {            183      6) {
136        UserAccount.findOne({
137          where: {
138            email           182      // Create
139          }                 183      Director
140        }).then(user => {   183      or
141          if (user) {       184      Registrar
142            errors.email = "Email
143              already exists";
144            res.status(400).json(184      Table
145              errors);       185      Nonacademic.
146          } else {           186      create({ 
147            // Default password:186      accountID:
148            firstname+lastname
149              +123           187      user2.
150            // Still subject to
151            change          188      accountID
152            bcrypt.genSalt(10, ( 189      created
153              err, salt) => { 190      successfully
154              const password = 191      !
155                firstName + "" +
156                lastName + "1237
157                ";           192      });
158              bcrypt.hash( 189      });
159                password.       193      });
160                toLowerCase(), 193      );
161                salt, (err, hash) 194      );
162              ) => {           195      });
163              let na = "NA";    195      );
164              UserAccount.     196      );
165                create({       196      );
166                  email,        197      );
167                  password: hash, 197      );
168                  isActive: 1,   198      );
169                  position,      198      );
170                  firstName,    199      );
171                  lastName,     199      );
172                  middleName,   199      );
173                  suffix: na,    199      );
174                  nickname: na, 199      );
175                  imageUrl: na, 199      );
176                  contactNum: na, 199      );
177                  address: na,   199      );
178                  province: na, 199      );
179                  city: na,       199      );
180                  region: na,    199      );
181                  zipcode: na,   199      );
182                  civilStatus: " 199      );
183                  SINGLE",      199      );
184                  sex: "M",       199      );
185                  citizenship: na, 199      );
186                  ,             200      );
187                  birthDate: 0,   200      );
188                  birthPlace: na, 200      );
189                  religion: na, 200      );
190                  emergencyName: 200      );
191                  na,             200      );
192                  emergencyAddress 201      );
193                  : na,           201      );
194                  emergencyTelephone 202      );
195                  : na,           202      );
196                  emergencyCellphone 202      );
197                  : na,           202      );
198                  emergencyEmail: 203      );
199                  na,             203      );

```

```

204                               na, 239      );
205                               fatherAddress240s
206                               : na, 241      // @route POST api/admin/
207                               : na, 242      deactivate
208                               fatherOccupation243 // @desc Delete an account
209                               : na, 244      // @access Private
210                               fatherEmployer245
211                               : na, 246      router.post(
212                               : na, 247      "/deactivate",
213                               fatherOfficeName248
214                               : na, 249      passport.authenticate("admin"
215                               : na, 250      , {
216                               motherAddress251
217                               : na, 252      session: false
218                               motherEmail253
219                               : na, 254      });
220                               motherOccupation255
221                               : na, 256      (req, res) => {
222                               motherEmployer257
223                               : na, 258      const {
224                               motherBusiness259
225                               : na, 260      accountID
226                               motherOfficeName261
227                               : na, 262      } = req.body;
228                               msg: "User account
229                               Student
230                               account263      UserAccount.findOne({
231                               created264      where: {
232                               successf265      accountID
233                               !"      }
234                               });
235                               });
236                               });
237                               });
238

```

299


```

        , {
          session: false
        }) ,
      async(req, res) => {
        const {
          accountID,
          parentID
        } = req.body
        ParentGuardian.findOne({
          where: {
            accountID: parentID
          }
        }).then(async pg => {
          if (pg) {
            if (pg.studentIDs ===
              null || pg.
              studentIDs ===
              undefined) {
              let newData = [
                accountID]
              pg.update({
                studentIDs: JSON.
                  stringify(
                    newData)
              }).then(() => res.
                status(200).json({
                  msg: "Parent
                        assigned
                        successfully!"
                }))
            } else {
              let tempData = [...]
                JSON.parse(pg.
                  studentIDs)]
              if (!tempData.
                includes(accountID
                  )) {
                tempData.push(
                  accountID)
                pg.update({
                  studentIDs: JSON.
                    stringify(
                      tempData)
                }).then(() => res.
                  status(200).js458
                  ({
                    msg: "Parent
                          assigned
                          successfully!"
                  }))
              }
            }
          }
        })
      }
    }
  }
}

// @route POST api/admin/
// @desc Get all parent account
// @access Private
router.post(
  "/getallparents",
  passport.authenticate("admin"
    ,
    {
      session: false
    }),
  async(req, res) => {
    let {
      page,
      pageSize,
      keyword,
      accountID
    } = req.body;
    page = page - 1;
    let offset = page *
      pageSize;
    let limit = offset +
      pageSize;
    let listParents = await
      ParentGuardian.findAll()

```

```
482     .then(async pg => {      525
483       if (pg) {
484         let data = []          526
485         for (const [index,
486               value] of pg.entries 527
487               ()) {
488             if (value.studentIDs 529
489                 !== null && value 530
490                 studentIDs !== 531
491                 undefined) {
492               let studentIDs = 532
493                 JSON.parse(value 533
494                 .studentIDs)
495               if (studentIDs.      534
496                 includes(      535
497                   accountID)) {
498                 data.push({      536
499                   accountID:      537
500                     value.        538
501                     accountID 539
502                   })
503                 })
504               }
505             }
506           }
507         return data
508       }
509     })
510   UserAccount.findAll({
511     limit,                  542
512     offset,                 543
513     where: {                544
514       position: 5,           545
515       [Op.or]: [{            546
516         firstName: {         547
517           [Op.like]: '%${ 548
518             keyword}%'
519         }
520       }, {                  549
521         lastName: {          550
522           [Op.like]: '%${ 551
523             keyword}%'
524         }
525       }]
526     })
527   .then(users => {
528     let accountData = [];
529     if (users.length != 0) {
530       users.slice(0,
531         pageSize).forEach( 532
532           async(user, key, 533
533             arr) => {
534             const keyID = user. 535
535               accountID;
536             const email = user. 536
537               email;
538             const name = '${ 537
539               utils.capitalize( 538
540                 (
541                   user.lastName 540
542                 ), ${utils. 543
543                   capitalize(user 544
544                     firstName)} ${ 545
545                     user.middleName
546                     .charAt(0) 546
547                     .toUpperCase() 547
548                     }.';
549             const position =
550               utils. 549
551                 capitalize(user 550
552                   firstName) ${ 551
553                     user.middleName
554                     .charAt(0) 553
555                     .toUpperCase() 554
556                     }.';
557             res.status
558               (200).json
559               ({
560                 numOfPages:
561                   Math.
562                     ceil(
563                       count.
564                         count /
565                           pageSize
566                         ),
567                         accountList
568                           :
569                             accountData
570                           });
571           })
572         .then(count =>
573           {
574             accountData.
575               sort((a, b
576                 ) => {
577               if (a.name
578                 < b.name
579                 ) {
580                 return
581                   -1;
582               }
583               if (a.name
584                 > b.name
585                 ) {
586                 return 1;
587               }
588             })
589           .then(() =>
590             res.json
591               ({
592                 numOfPages:
593                   Math.
594                     ceil(
595                       count.
596                         count /
597                           pageSize
598                         ),
599                         accountList
600                           :
601                             accountData
602                           }));
603           })
604         })
605       })
606     })
607   .then(accountData => {
608     const imageUrl =
609       user.imageUrl;
610     const isActive =
611       user.isActive;
612     accountData.push({
613       key: keyID,
614       email,
615       name,
616       position,
617       imageUrl,
618       isActive
619     });
620   })
621   .catch(error => {
622     console.error(error);
623   })
624 
```



```

98         successfully 153
99     });
100    });
101   });
102 });
103 );
104 );
105 // @route POST api/cashier/
106 // getallstudents
107 // @desc Get all students from
108 // a keyword
109 router.post(
110   "/getallstudents",
111   passport.authenticate("cashier", {
112     session: false
113   }),
114   async(req, res) => {
115     let {
116       page,
117       pageSize,
118       keyword
119     } = req.body;
120     page = page - 1;
121     let offset = page *
122       pageSize;
123     let limit = offset +
124       pageSize;
125     UserAccount.findAll({
126       limit,
127       offset,
128       where: {
129         position: 4,
130         [Op.or]: [
131           {
132             firstName: {
133               [Op.like]: `%"${134
135               keyword}%`136
137             }
138           }
139         ]
140       }
141     })
142     .then(users => {
143       let accountData = [];
144       if (users.length != 0)
145       {
146         users.slice(0,
147           pageSize).forEach(
148             user, key, arr) => {
149           const keyID = user.accountID;
150           const email = user.email;
151           const name = user.firstName +
152             " " +
153             user.lastName;
154           const position = user.utils.
155             displayPosition();
156           const imageUrl = user.imageUrl;
157           const isActive = user.isActive;
158           accountData.push({
159             key: keyID,
160             email,
161             name,
162             position,
163             imageUrl,
164             isActive
165           });
166           if (key == arr.length - 1) {
167             UserAccount.
168               findAndCountAll(
169                 {
170                   where: {
171                     position: 4,
172                     [Op.or]: [
173                       {
174                         firstName: {
175                           [Op.like]: `%"${176
177               keyword}%`178
179             }
180           ]
181         }
182       })
183       .then(count =>
184         {
185           accountData.
186             sort((a, b) => (a.key > b.key ? 1 : -1));
187           res.status(200).json({
188             count: Math.ceil(count / pageSize),
189             accountList: accountData
190           });
191         })
192       .catch(err => {
193         res.status(404);
194       });
195     }
196   }
197 });
198 });
199 
```

```

192         }
193     })
194     .catch(err => {
195       res.status(404);
196     });
197   );
198 );
199
200 // @route POST api/cashier/
201 // @desc Restrict an account
202 // @access Private
203
204 router.post(
205   "/restrict",
206   passport.authenticate("cashier", {
207     session: false
208   }),
209   (req, res) => {
210     const {
211       accountID,
212       message
213     } = req.body;
214     UserAccount.findOne({
215       where: {
216         accountID
217       }
218     }).then(user => {
219       if (!user) {
220         errors.msg = "User account not found";
221         res.status(404).json(
222           errors);
223       } else {
224         user.update({
225           isActive: false
226         }).then(() => {
227           AccountNotice.create({
228             accountID,
229             message
230           }).then(accountnotice => {
231             res.status(200).json({
232               msg: "Account restricted successfully!"
233             });
234           });
235         });
236       });
237     );
238   );
239
240 // @route POST api/cashier/
241 // @desc Unrestrict an account
242 // @access Private
243
244 router.post(
245   "/unrestrict",
246   passport.authenticate("cashier", {
247     session: false
248   }),
249   (req, res) => {
250     const {
251       accountID
252     } = req.body;
253     UserAccount.findOne({
254       where: {
255         accountID
256       }
257     }).then(user => {
258       if (!user) {
259         errors.msg = "User account not found";
260         res.status(404).json(
261           errors);
262       } else {
263         user.update({
264           isActive: true
265         }).then(() => {
266           AccountNotice.findOne({
267             where: {
268               accountID
269             }
270           }).then(
271             accountnotice => {
272               if (accountnotice)
273                 accountnotice.destroy();
274               then(() => {
275                 res
276                   .status
277                     (200)
278                   .json({
279                     msg: "Account
280                       unrestricted
281                         successfully!
282                       !");
283                 });
284               });
285             });
286           });
287         });
288       });
289     });
290   );
291
292 module.exports = router;

```

Listing 31: director.js

```

const express = require("express");
const router = express.Router();
const UserAccount = require("../models/UserAccount");
const passport = require("passport");
const Sequelize = require("sequelize");
const Op = Sequelize.Op;

// Input Validation
const validateEditProfileNonacademic
  = require("../validation")

```

```

10     /editprofilenonacademic"); 76
11 //Import Utility Functions    77
12 const utils = require("../../" 78
13     utils");                 79
14 // @route POST api/director/ 80
15     updateprofile            81
16 // @desc Update profile      82
17 // @access Private           83
18 router.post(                  84
19     "/updateprofile",         85
20     passport.authenticate(" 86
21         director", { session: 87
22             false }),          88
23     (req, res) => {           89
24         const {                90
25             firstName,          91
26             lastName,           92
27             middleName,          93
28             suffix,              94
29             nickname,           95
30             contactNum,          96
31             address,             97
32             province,            98
33             city,                 99
34             region,               100
35             zipcode,              101
36             civilStatus,          102
37             sex,                  103
38             citizenship,          104
39             birthDate,             105
40             birthPlace,            106
41             religion,              107
42             emergencyName,          108
43             emergencyAddress,        109
44             emergencyTelephone,       110
45             emergencyCellphone,        111
46             emergencyEmail,           112
47             emergencyRelationship } 113
48 } = req.body;                  114
49
50 const { errors, isValid } = 115
51     validateEditProfileNonac 116
52     (req.body);                 117
53
54 if (!isValid) {               118
55     return res.status(400). 119
56         json(errors);        120
57 }
58 UserAccount.findOne({          121
59     where: {                  122
60         accountID: req.user. 123
61         accountID           124
62     }                         125
63 }).then(user => {           126
64     if (user) {               127
65         user                 128
66             .update({          129
67                 firstName,        130
68                 lastName,         131
69                 middleName,        132
70                 suffix,           133
71                 nickname,         134
72                 contactNum,        135
73                 address,          136
74                 province,         137
75                 city,              138
76                 region,            139
77                 zipcode,           140
78                 civilStatus,        141
79                 sex,                142
80                 citizenship,        143
81             })                  144
82         .then(user2 => {        145
83             res.status(200).      146
84                 json({ msg: " 147
85                     Profile updated 148
86                     successfully!" 149
87                 });            150
88         });
89     });
90     });
91     });
92     });
93     });
94     module.exports = router;
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
75310
75311
75312
75313
75314
75315
75316
75317
75318
75319
75320
75321
75322
75323
75324
75325
75326
75327
75328
75329
75330
75331
75332
75333
75334
75335
75336
75337
75338
75339
75340
75341
75342
75343
75344
75345
75346
75347
75348
75349
75350
75351
75352
75353
75354
75355
75356
75357
75358
75359
75360
75361
75362
75363
75364
75365
75366
75367
75368
75369
75370
75371
75372
75373
75374
75375
75376
75377
75378
75379
75380
75381
75382
75383
75384
75385
75386
75387
75388
75389
75390
75391
75392
75393
75394
75395
75396
75397
75398
75399
753100
753101
753102
753103
753104
753105
753106
753107
753108
753109
753110
753111
753112
753113
753114
753115
753116
753117
753118
753119
7531100
7531101
7531102
7531103
7531104
7531105
7531106
7531107
7531108
7531109
75311010
75311011
75311012
75311013
75311014
75311015
75311016
75311017
75311018
75311019
753110100
753110101
753110102
753110103
753110104
753110105
753110106
753110107
753110108
753110109
753110110
753110111
753110112
753110113
753110114
753110115
753110116
753110117
753110118
753110119
7531101100
7531101101
7531101102
7531101103
7531101104
7531101105
7531101106
7531101107
7531101108
7531101109
7531101110
7531101111
7531101112
7531101113
7531101114
7531101115
7531101116
7531101117
7531101118
7531101119
75311011100
75311011101
75311011102
75311011103
75311011104
75311011105
75311011106
75311011107
75311011108
75311011109
75311011110
75311011111
75311011112
75311011113
75311011114
75311011115
75311011116
75311011117
75311011118
75311011119
753110111100
753110111101
753110111102
753110111103
753110111104
753110111105
753110111106
753110111107
753110111108
753110111109
753110111110
753110111111
753110111112
753110111113
753110111114
753110111115
753110111116
753110111117
753110111118
753110111119
7531101111100
7531101111101
7531101111102
7531101111103
7531101111104
7531101111105
7531101111106
7531101111107
7531101111108
7531101111109
7531101111110
7531101111111
7531101111112
7531101111113
7531101111114
7531101111115
7531101111116
7531101111117
7531101111118
7531101111119
75311011111100
75311011111101
75311011111102
75311011111103
75311011111104
75311011111105
75311011111106
75311011111107
75311011111108
75311011111109
75311011111110
75311011111111
75311011111112
75311011111113
75311011111114
75311011111115
75311011111116
75311011111117
75311011111118
75311011111119
753110111111100
753110111111101
753110111111102
753110111111103
753110111111104
753110111111105
753110111111106
753110111111107
753110111111108
753110111111109
753110111111110
753110111111111
753110111111112
753110111111113
753110111111114
753110111111115
753110111111116
753110111111117
753110111111118
753110111111119
7531101111111100
7531101111111101
7531101111111102
7531101111111103
7531101111111104
7531101111111105
7531101111111106
7531101111111107
7531101111111108
7531101111111109
7531101111111110
7531101111111111
7531101111111112
7531101111111113
7531101111111114
7531101111111115
7531101111111116
7531101111111117
7531101111111118
7531101111111119
75311011111111100
75311011111111101
75311011111111102
75311011111111103
75311011111111104
75311011111111105
75311011111111106
75311011111111107
75311011111111108
75311011111111109
75311011111111110
75311011111111111
75311011111111112
75311011111111113
75311011111111114
75311011111111115
75311011111111116
75311011111111117
75311011111111118
75311011111111119
753110111111111100
753110111111111101
753110111111111102
753110111111111103
753110111111111104
753110111111111105
753110111111111106
753110111111111107
753110111111111108
753110111111111109
753110111111111110
753110111111111111
753110111111111112
753110111111111113
753110111111111114
753110111111111115
753110111111111116
753110111111111117
753110111111111118
753110111111111119
7531101111111111100
7531101111111111101
7531101111111111102
7531101111111111103
7531101111111111104
7531101111111111105
7531101111111111106
7531101111111111107
7531101111111111108
7531101111111111109
7531101111111111110
7531101111111111111
7531101111111111112
7531101111111111113
7531101111111111114
7531101111111111115
7531101111111111116
7531101111111111117
7531101111111111118
7531101111111111119
75311011111111111100
75311011111111111101
75311011111111111102
75311011111111111103
75311011111111111104
75311011111111111105
75311011111111111106
75311011111111111107
75311011111111111108
75311011111111111109
75311011111111111110
75311011111111111111
75311011111111111112
75311011111111111113
75311011111111111114
75311011111111111115
75311011111111111116
75311011111111111117
75311011111111111118
75311011111111111119
753110111111111111100
753110111111111111101
753110111111111111102
753110111111111111103
753110111111111111104
753110111111111111105
753110111111111111106
753110111111111111107
753110111111111111108
753110111111111111109
753110111111111111110
753110111111111111111
753110111111111111112
753110111111111111113
753110111111111111114
753110111111111111115
753110111111111111116
753110111111111111117
753110111111111111118
753110111111111111119
7531101111111111111100
7531101111111111111101
7531101111111111111102
7531101111111111111103
7531101111111111111104
7531101111111111111105
7531101111111111111106
7531101111111111111107
7531101111111111111108
7531101111111111111109
7531101111111111111110
7531101111111111111111
7531101111111111111112
7531101111111111111113
7531101111111111111114
7531101111111111111115
7531101111111111111116
7531101111111111111117
7531101111111111111118
7531101111111111111119
75311011111111111111100
75311011111111111111101
75311011111111111111102
75311011111111111111103
75311011111111111111104
75311011111111111111105
75311011111111111111106
75311011111111111111107
75311011111111111111108
75311011111111111111109
75311011111111111111110
75311011111111111111111
75311011111111111111112
75311011111111111111113
75311011111111111111114
75311011111111111111115
75311011111111111111116
75311011111111111111117
75311011111111111111118
75311011111111111111119
753110111111111111111100
753110111111111111111101
753110111111111111111102
753110111111111111111103
753110111111111111111104
753110111111111111111105
753110111111111111111106
753110111111111111111107
753110111111111111111108
753110111111111111111109
753110111111111111111110
753110111111111111111111
753110111111111111111112
753110111111111111111113
753110111111111111111114
753110111111111111111115
753110111111111111111116
753110111111111111111117
753110111111111111111118
753110111111111111111119
7531101111111111111111100
7531101111111111111111101
7531101111111111111111102
7531101111111111111111103
7531101111111111111111104
7531101111111111111111105
7531101111111111111111106
7531101111111111111111107
7531101111111111111111108
7531101111111111111111109
7531101111111111111111110
7531101111111111111111111
7531101111111111111111112
7531101111111111111111113
7531101111111111111111114
7531101111111111111111115
7531101111111111111111116
7531101111111111111111117
7531101111111111111111118
7531101111111111111111119
75311011111111111111111100
75311011111111111111111101
75311011111111111111111102
75311011111111111111111103
75311011111111111111111104
75311011111111111111111105
75311011111111111111111106
75311011111111111111111107
75311011111111111111111108
75311011111111111111111109
75311011111111111111111110
75311011111111111111111111
75311011111111111111111112
75311011111111111111111113
75311011111111111111111114
75311011111111111111111115
75311011111111111111111116
75311011111111111111111117
75311011111111111111111118
75311011111111111111111119
753110111111111111111111100
753110111111111111111111101
753110111111111111111111102
753110111111111111111111103
753110111111111111111111104
753110111111111111111111105
753110111111111111111111106
753110111111111111111111107
753110111111111111111111108
753110111111111111111111109
753110111111111111111111110
753110111111111111111111111
753110111111111111111111112
753110111111111111111111113
753110111111111111111111114
753110111111111111111111115
753110111111111111111111116
753110111111111111111111117
753110111111111111111111118
753110111111111111111111119
7531101111111111111111111100
7531101111111111111111111101
7531101111111111111111111102
7531101111111111111111111103
7531101111111111111111111104
7531101111111111111111111105
7531101111111111111111111106
7531101111111111111111111107
7531101111111111111111111108
7531101111111111111111111109
7531101111111111111111111110
7531101111111111111111111111
7531101111111111111111111112
7531101111111111111111111113
7531101111111111111111111114
7531101111111111111111111115
7531101111111111111111111116
7531101111111111111111111117
7531101111111111111111111118
753110111111
```

Listing 32: parent.js

```
const express = require("express");
const router = express.Router();
const passport = require("passport");
const UserAccount = require("../models/UserAccount");

// Input Validation
const validateEditProfileParent = require("../validation/editprofileparent");

// Import Utility Functions
const utils = require("../utils");

// @router POST api/parent/updateprofile
// @desc Update profile
// @access Private

router.post(
  "/updateprofile",
  passport.authenticate("guardian", { session: false }),
  (req, res) => {
    const {
      firstName,
      lastName,
      middleName,
      suffix,
      nickname,
      contactNum,
      address,
      province,
      city,
      region,
      zipcode,
      civilStatus,
      sex,
      citizenship,
    } = req.body;
    UserAccount.findByIdAndUpdate(req.params.parentId, {
      $set: {
        firstName,
        lastName,
        middleName,
        suffix,
        nickname,
        contactNum,
        address,
        province,
        city,
        region,
        zipcode,
        civilStatus,
        sex,
        citizenship,
      },
    }, { new: true })
      .then((parent) => res.json(parent))
      .catch((err) => res.status(400).json({ msg: err }));
  }
);
```

```

35         birthDate ,
36         birthPlace ,
37         religion ,
38         emergencyName ,
39         emergencyAddress ,
40         emergencyTelephone ,
41         emergencyCellphone ,
42         emergencyEmail ,
43         emergencyRelationship
44     } = req.body;
45
46     const { errors, isValid } =
47
48         validateEditProfileParent
49         (req.body);
50
51     if (!isValid) {
52         return res.status(400).
53             json(errors);
54     }
55
56     UserAccount.findOne({
57         where: {
58             accountID: req.user.
59                 accountID
60         }
61     }).then(user => {
62         if (user) {
63             user
64                 .update({
65                     firstName ,
66                     lastName ,
67                     middleName ,
68                     suffix ,
69                     nickname ,
70                     contactNum ,
71                     address ,
72                     province ,
73                     city ,
74                     region ,
75                     zipcode ,
76                     civilStatus ,
77                     sex ,
78                     citizenship ,
79                     birthDate ,
80                     birthPlace ,
81                     religion ,
82                     emergencyName ,
83                     emergencyAddress ,
84                     emergencyTelephone ,
85                     emergencyCellphone ,
86                     emergencyEmail ,
87                     emergencyRelationship
88                 })
89             .then(user2 => {
90                 res.status(200).
91                     json({ msg: "
92                         Profile updated
93                         successfully!"
94                     });
95             });
96         });
97     });
98
99     module.exports = router;

```

Listing 33: registrar.js

```

1  const express = require("express");
2  const PDFDocument = require("pdfkit");
3  const router = express.Router();
4  const UserAccount = require("../models/UserAccount");
5  const passport = require("passport");
6  const Sequelize = require("sequelize");
7  const Teacher = require("../models/Teacher");
8  const SubmissionDeadline = require("../models/SubmissionDeadline")
9    ;
10 const Section = require("../models/Section");
11 const StudentSection = require("../models/StudentSection");
12 const SubjectSection = require("../models/SubjectSection");
13 const ClassRecord = require("../models/ClassRecord");
14 const SubjectSectionStudent = require("../models/
   SubjectSectionStudent");
15 const StudentWeightedScore = require("../models/
   StudentWeightedScore");
16 const Student = require("../models/Student");
17 const TeacherSection = require("../models/TeacherSection");
18 const Subject = require("../models/Subject");
19 const Component = require("../models/Component");
20 const Subcomponent = require("../models/Subcomponent");
21 const StudentSubjectGrades = require("../models/
   StudentSubjectGrades");
22 const SchoolYear = require("../models/SchoolYear");
23 const Grade = require("../models/Grade");
24 const ClassRecordStatus = require("../models/ClassRecordStatus");
25 const ActivityLog = require("../models/ActivityLog");
26 const LogDetails = require("../models/LogDetails");
27 const StudentGrades = require("../models/StudentGrades");
28 const StudentFinalGrade = require("../models/StudentFinalGrade");
29 const Op = Sequelize.Op;
30
31 // Director and Registrar Shared API
32 const directorRegistrar = ["director", "registrar"]
33
34 // Import Utility Functions
35 const utils = require("../utils");
36
37 // Input Validation
38 const validateEditProfileNonacademic = require("../validation/
   editprofilenonacademic");
39 const validateAddSection = require("../validation/addsection");
40 const validateSubcomponent = require("../validation/subcomponents"
   );
41
42 // @route POST api/registar/updateprofile
43 // @desc Update profile
44 // @access Private
45
46 router.post(
47   "/updateprofile",
48   passport.authenticate("registrar", {
49     session: false
50   }),
51   (req, res) => {
52     const {
53       firstName,
54       lastName,
55       middleName,
56       suffix,
57       nickname,
58       contactNum,
59       address,
60       province,
61       city,
62       region,
63       zipcode,
64     }
65   )
66 )
67
68 module.exports = router;

```

```

63         civilStatus ,
64         sex ,
65         citizenship ,
66         birthDate ,
67         birthPlace ,
68         religion ,
69         emergencyName ,
70         emergencyAddress ,
71         emergencyTelephone ,
72         emergencyCellphone ,
73         emergencyEmail ,
74         emergencyRelationship
75     } = req.body;
76
77
78     const {
79         errors ,
80         isValid
81     } = validateEditProfileNonacademic(req.body);
82
83     if (!isValid) {
84         return res.status(400).json(errors);
85     }
86
87     UserAccount.findOne({
88         where: {
89             accountID: req.user.accountID
90         }
91     }).then(user => {
92         if (user) {
93             user
94                 .update({
95                     firstName ,
96                     lastName ,
97                     middleName ,
98                     suffix ,
99                     nickname ,
100                    contactNum ,
101                    address ,
102                    province ,
103                    city ,
104                    region ,
105                    zipcode ,
106                    civilStatus ,
107                    sex ,
108                    citizenship ,
109                    birthDate ,
110                    birthPlace ,
111                    religion ,
112                    emergencyName ,
113                    emergencyAddress ,
114                    emergencyTelephone ,
115                    emergencyCellphone ,
116                    emergencyEmail ,
117                    emergencyRelationship
118                })
119                .then(user2 => {
120                    res.status(200).json({
121                        msg: "Profile updated successfully!"
122                    });
123                });
124            }
125        });
126    );
127
128 // @route POST api/registrar/setdeadlineall
129 // @desc Set the submission deadline for all teachers
130 // @access Private
131
132 router.post(
133     "/setdeadlineall",
134     passport.authenticate("registrar", {
135         session: false
136     }),
137

```

```

138     (req, res) => {
139         let {
140             deadline
141         } = req.body;
142         const deadlineDate = utils.getPHTime(new Date(deadline));
143         const dateSet = utils.getPHTime();
144         if (deadlineDate.getTime() <= dateSet.getTime()) {
145             res.status(400).json({
146                 msg: "Invalid date. Date must be greater than current date."
147             });
148         } else {
149             Teacher.findAll().then(teachers => {
150                 if (teachers.length != 0) {
151                     for (const [index, value] of teachers.entries()) {
152                         SubmissionDeadline.findOne({
153                             where: {
154                                 teacherID: value.teacherID,
155                                 isActive: 1
156                             }
157                         }).then(sd => {
158                             if (sd) {
159                                 sd.update({
160                                     deadline: deadlineDate,
161                                     dateSet
162                                 });
163                             } else {
164                                 SubmissionDeadline.create({
165                                     teacherID: value.teacherID,
166                                     isActive: 1,
167                                     deadline: deadlineDate,
168                                     dateSet
169                                 });
170                             }
171                         });
172                     }
173                     res.status(200).json({
174                         msg: "Successfully updated the submission deadline!"
175                     });
176                 } else {
177                     res.status(404).json({
178                         msg: "No active teachers found."
179                     });
180                 }
181             });
182         }
183     );
184 };
185
186 // @route GET api/registrar/disableddeadline
187 // @desc Disable the submission deadline for all teachers
188 // @access Private
189
190 router.get(
191     "/disableddeadline",
192     passport.authenticate("registrar", {
193         session: false
194     }),
195     (req, res) => {
196         SubmissionDeadline.findAll({
197             where: {
198                 isActive: true
199             }
200         }).then(submissiondeadline => {
201             if (submissiondeadline.length == 0) {
202                 res.status(404).json({
203                     msg: "There is no active submission deadline."
204                 });
205             } else {
206                 submissiondeadline.forEach(async(entry, key, arr) => {
207                     await entry.update({
208                         isActive: 0
209                     });

```

```

210     });
211     res.status(200).json({
212       msg: "Disabled successfully!"
213     });
214   }
215 }
216 );
217 );
218
219 // @route POST api/registrar/removedeadline
220 // @desc Remove submission deadline by deadlineID
221 // @access Private
222
223 router.post(
224   "/removedeadline",
225   passport.authenticate("registrar", {
226     session: false
227   }),
228   async(req, res) => {
229     const {
230       deadlineID
231     } = req.body;
232     if (deadlineID != 0) {
233       SubmissionDeadline.findOne({
234         where: {
235           deadlineID
236         }
237       }).then(sd => {
238         if (sd) {
239           sd.update({
240             isActive: 0
241           }).then(() => {
242             res.status(200).json({
243               msg: "Submission deadline removed successfully."
244             });
245           });
246         } else {
247           res.status(404).json({
248             msg: "Submission deadline not found!"
249           });
250         }
251       });
252     } else {
253       SubmissionDeadline.findAll().then(sds => {
254         if (sds.length != 0) {
255           for (const [index, value] of sds.entries()) {
256             value.update({
257               isActive: 0
258             });
259           }
260           res.status(200).json({
261             msg: "Submission deadline removed successfully."
262           });
263         }
264       });
265     }
266   );
267 )
268
269 // @route POST api/registrar/setdeadline
270 // @desc Set the submission deadline for given teachers
271 // @access Private
272
273 router.post(
274   "/setdeadline",
275   passport.authenticate("registrar", {
276     session: false
277   }),
278   async(req, res) => {
279     const {
280       teacherID,
281       deadline
282     }

```

```

282     } = req.body;
283     const deadlineDate = utils.getPHTime(new Date(deadline));
284     const dateSet = utils.getPHTime();
285     if (deadlineDate.getTime() <= dateSet.getTime()) {
286         res.status(400).json({
287             msg: "Invalid date. Date must be greater than current date."
288         });
289     } else {
290         SubmissionDeadline.findOne({
291             where: {
292                 teacherID,
293                 isActive: 1
294             }
295         }).then(sd => {
296             if (sd) {
297                 sd.update({
298                     deadline: deadlineDate,
299                     dateSet
300                 });
301             } else {
302                 SubmissionDeadline.create({
303                     deadline: deadlineDate,
304                     dateSet,
305                     teacherID,
306                     isActive: 1
307                 });
308             }
309             res.status(200).json({
310                 msg: "Successfully updated the submission deadline!"
311             });
312         });
313     }
314 );
315
316
317 // @route POST api/registrar/getsections
318 // @desc Get sections based on keyword
319 // @access Private
320
321 router.post(
322     "/getsections",
323     passport.authenticate(["registrar", "director"], {
324         session: false
325     }),
326     async(req, res) => {
327         let {
328             page,
329             pageSize,
330             keyword
331         } = req.body;
332         page = page - 1;
333         let offset = page * pageSize;
334         let limit = offset + pageSize;
335         Section.findAll({
336             where: {
337                 sectionName: {
338                     [Op.like]: `%"${keyword}"%`'``,
339                 },
340                 archived: 0
341             }
342         }).then(sections => {
343             let sectionData = [];
344             if (sections.length != 0) {
345                 sections.forEach(async(section, key, arr) => {
346                     const keyID = section.sectionID;
347                     const name = section.sectionName;
348                     const gradeLevel = section.gradeLevel;
349                     sectionData.push({
350                         key: keyID,
351                         name,
352                         gradeLevel
353                     });
354                     if (key == arr.length - 1) {

```

```

355         Section.findAndCountAll({
356             where: {
357                 sectionName: {
358                     [Op.like]: `%${keyword}%`
359                 },
360                 archived: 0
361             }
362         })
363         .then(count => {
364             sectionData.sort((a, b) =>
365                 a.gradeLevel > b.gradeLevel ? 1 : -1
366             );
367             res.status(200).json({
368                 numOfPages: Math.ceil(count.count / pageSize),
369                 sectionList: sectionData.slice(
370                     pageSize * page,
371                     pageSize * (page + 1)
372                 )
373             });
374         })
375         .catch(err => {
376             res.status(404);
377         });
378     }
379   });
380 } else {
381   res.status(404).json({
382     msg: "Not found"
383   });
384 }
385 );
386 }
387 );
388
389 // @route POST api/registrar/sectiongradelevel
390 // @desc Get grade level by studentID
391 // @access Private
392
393 router.post(
394   "/sectiongradelevel",
395   passport.authenticate("registrar", {
396     session: false
397   }),
398   async(req, res) => {
399     const {
400       sectionID
401     } = req.body;
402     let gradeLevel = await utils.getGradeLevelBySectionID(sectionID);
403     if (gradeLevel === "") {
404       res.status(404).json({
405         msg: "Not found!"
406       });
407     } else {
408       res.status(200).json({
409         gradeLevel
410       });
411     }
412   }
413 );
414
415 // @route POST api/registrar/getpastgradelevel
416 // @desc Get past grade level
417 // @access Private
418
419 router.post(
420   "/getpastgradelevel",
421   passport.authenticate("registrar", {
422     session: false
423   }),
424   (req, res) => {
425     const {
426       gradeLevel

```

```

427     } = req.body;
428     const gradeLevels = [
429         "N",
430         "K1",
431         "K2",
432         "G1",
433         "G2",
434         "G3",
435         "G4",
436         "G5",
437         "G6",
438         "G7",
439         "G8",
440         "G9",
441         "G10",
442         "G11",
443         "G12"
444     ];
445     const gradeLevelIndex = gradeLevels.indexOf(gradeLevel);
446     if (gradeLevelIndex != 0) {
447         res.status(200).json({
448             gradeLevel: gradeLevels[gradeLevelIndex - 1]
449         });
450     } else {
451         res.status(200).json({
452             gradeLevel: gradeLevels[gradeLevelIndex]
453         });
454     }
455 }
456 );
457
458 // @route POST api/registrar/addsection
459 // @desc Add a new section
460 // @access Private
461
462 router.post(
463     "/addsection",
464     passport.authenticate("registrar", {
465         session: false
466     }),
467     async(req, res) => {
468         const {
469             sectionName,
470             gradeLevel
471         } = req.body;
472         const {
473             errors,
474             isValid
475         } = validateAddSection({
476             sectionName
477         );
478
479         if (!isValid) {
480             return res.status(400).json(errors);
481         }
482
483         Section.findOne({
484             where: {
485                 sectionName,
486                 archived: 0
487             }
488         }).then(section => {
489             if (section) {
490                 res.status(400).json({
491                     sectionName: "Section name is already taken"
492                 });
493             } else {
494                 Section.create({
495                     sectionName,
496                     gradeLevel
497                 }).then(section2 => {
498                     res.status(200).json({
499                         msg: "Section created successfully!"
500                     });

```

```

501         });
502     });
503   });
504 }
505 );
506
507 // @route POST api/registrar/editsection
508 // @desc Rename a section
509 // @access Private
510
511 router.post(
512   "/editsection",
513   passport.authenticate("registrar", {
514     session: false
515   }),
516   async(req, res) => {
517     const {
518       sectionID,
519       sectionName,
520       gradeLevel
521     } = req.body;
522     const {
523       errors,
524       isValid
525     } = validateAddSection({
526       sectionName
527     });
528     if (!isValid) {
529       return res.status(400).json(errors);
530     }
531     Section.findOne({
532       where: {
533         sectionID
534       }
535     }).then(section => {
536       if (section) {
537         section
538           .update({
539             sectionName,
540             gradeLevel
541           })
542           .then(section2 => {
543             res.status(200).json({
544               msg: "Section updated successfully!"
545             });
546           });
547       } else {
548         res.status(404).json({
549           msg: "Section not found"
550         });
551       }
552     });
553   }
554 );
555
556 // @route POST api/registrar/deletesection
557 // @desc Delete a section
558 // @access Private
559
560 router.post(
561   "/deletesection",
562   passport.authenticate("registrar", {
563     session: false
564   }),
565   async(req, res) => {
566     const {
567       sectionID
568     } = req.body;
569     Section.findOne({
570       where: {
571         sectionID
572       }

```

```

573         })
574     .then(async section => {
575       let {
576         schoolYearID,
577         quarter
578       } = await utils.getActiveSY();
579       await StudentSection.findAll({
580         where: {
581           sectionID,
582           schoolYearID
583         }
584       }).then(async studentsections => {
585         let hasData = [];
586         studentsections.forEach(async(studentsection, key, arr) =>
587           {
588             hasData.push(studentsection.studsectID);
589           });
590         if (hasData.length != 0) {
591           res.status(400).json({
592             msg: "Operation could not be completed. Remove all
593               students under this section first."
594           });
595         } else {
596           await SubjectSection.findAll({
597             where: {
598               sectionID,
599               schoolYearID
600             }
601           }).then(async ubjectsections => {
602             let hasData2 = [];
603             subjectsections.forEach(async(subjectsection, key, arr)
604               => {
605               hasData.push(subjectsection.subsectID);
606             });
607             if (hasData2.length != 0) {
608               res.status(400).json({
609                 msg: "Operation could not be completed. There are
610                   active subjects in this section."
611               });
612             } else {
613               await TeacherSection.findOne({
614                 where: {
615                   sectionID,
616                   schoolYearID
617                 }
618               }).then(adviser => {
619                 if (adviser) {
620                   adviser.destroy({});
```

```

641 // @access Private
642
643 router.post(
644   "/getallstudents",
645   passport.authenticate(["registrar", 'director'], {
646     session: false
647   }),
648   async(req, res) => {
649     let {
650       page,
651       pageSize,
652       keyword
653     } = req.body;
654     page = page - 1;
655     let offset = page * pageSize;
656     let limit = offset + pageSize;
657
658     UserAccount.findAll({
659       limit,
660       offset,
661       where: [
662         {
663           position: 4,
664           [Op.or]: [
665             {
666               firstName: {
667                 [Op.like]: `%"${keyword}"%` }
668             },
669             {
670               lastName: {
671                 [Op.like]: `%"${keyword}"%` }
672             }
673           ]
674         }
675       ].then(users => {
676         let accountData = [];
677         if (users.length != 0) {
678           users.slice(0, pageSize).forEach(async(user, key, arr) => {
679             const keyID = user.accountID;
680             const studentID = await utils.getStudentID(user.accountID);
681             const email = user.email;
682             const name = `${utils.capitalize(
683               user.lastName
684             )}, ${utils.capitalize(user.firstName)} ${user.middleName
685             .charAt(0)
686             .toUpperCase()}.`;
687             const position = utils.displayPosition(user.position);
688             const imageUrl = user.imageUrl;
689             const isActive = user.isActive;
690             accountData.push({
691               key: keyID,
692               studentID,
693               email,
694               name,
695               position,
696               imageUrl,
697               isActive
698             });
699             if (key == arr.length - 1) {
700               UserAccount.findAndCountAll({
701                 where: [
702                   {
703                     position: 4,
704                     [Op.or]: [
705                       {
706                         firstName: {
707                           [Op.like]: `%"${keyword}"%` }
708                         },
709                         {
710                           lastName: {
711                             [Op.like]: `%"${keyword}"%` }
712                           }
713                         ]
714                   }
715                 ]
716               })
717             }
718           })
719         }
720       })
721     }
722   }
723 )

```

```

713         .then(count => {
714             accountData.sort((a, b) =>
715                 a.accountID > b.accountID ? 1 : -1
716             );
717             res.status(200).json({
718                 numOfPages: Math.ceil(count.count / pageSize),
719                 accountList: accountData
720             });
721         })
722         .catch(err => {
723             res.status(200).json({
724                 numOfPages: 1,
725                 accountList: []
726             });
727         });
728     }
729   );
730 } else {
731     res.status(200).json({
732         numOfPages: 1,
733         accountList: []
734     });
735 }
736 )
737 .catch(err => {
738     res.status(200).json({
739         numOfPages: 1,
740         accountList: []
741     });
742 });
743 }
744 );
745
746 // @route POST api/registrar/getallteachers
747 // @desc Get all teachers from a keyword
748 // @access Private
749
750 router.post(
751   "/getallteachers",
752   passport.authenticate(["registrar", "director"], {
753     session: false
754   }),
755   async(req, res) => {
756     let {
757       page,
758       pageSize,
759       keyword
760     } = req.body;
761     page = page - 1;
762     let offset = page * pageSize;
763     let limit = offset + pageSize;
764     UserAccount.findAll({
765       limit,
766       offset,
767       where: {
768         position: 3,
769         [Op.or]: [
770           {
771             firstName: {
772               [Op.like]: `%"${keyword}"%`
773             }
774           },
775           {
776             lastName: {
777               [Op.like]: `%"${keyword}"%`
778             }
779           }
780         ]
781       }
782     })
783     .then(async users => {
784       let accountData = [];
785       if (users.length != 0) {
786         users.slice(0, pageSize).forEach(async(user, key, arr) => {
787           const keyID = user.accountID;

```

```

785     const teacherID = await utils.getTeacherID(user.accountID
786         );
787     const email = user.email;
788     const name = `${utils.capitalize(
789         user.lastName
790     )}, ${utils.capitalize(user.firstName)} ${user.middleName
791         .charAt(0)
792         .toUpperCase()}.`;
793     const position = utils.displayPosition(user.position);
794     const imageUrl = user.imageUrl;
795     const isActive = user.isActive;
796     const {
797         deadline,
798         deadlineID
799     } = await SubmissionDeadline.findOne({
800         where: {
801             teacherID,
802             isActive: 1
803         }
804     }).then(sd => {
805         if (sd) {
806             const {
807                 deadline,
808                 deadlineID
809             } = sd;
810             return {
811                 deadline,
812                 deadlineID
813             };
814         } else {
815             return {
816                 deadline: "NOT SET",
817                 deadlineID: -1
818             };
819         }
820     });
821     accountData.push({
822         deadline,
823         teacherID,
824         key: keyID,
825         email,
826         name,
827         position,
828         imageUrl,
829         isActive,
830         deadlineID
831     });
832     if (key == arr.length - 1) {
833         UserAccount.findAndCountAll({
834             where: {
835                 position: 3,
836                 [Op.or]: [
837                     {
838                         firstName: {
839                             [Op.like]: `%"${keyword}"%
840                         }
841                     },
842                     {
843                         lastName: {
844                             [Op.like]: `%"${keyword}"%
845                         }
846                     }
847                 ]
848             }
849         })
850         .then(count => {
851             accountData.sort((a, b) =>
852                 a.accountID > b.accountID ? 1 : -1
853             );
854             res.status(200).json({
855                 numOfPages: Math.ceil(count.count / pageSize),
856                 accountList: accountData
857             });
858         })
859         .catch(err => {
860             res.status(200).json({

```

```

857             numOfPages: 1,
858             accountList: []
859         });
860     });
861     }
862   });
863 } else {
864   res.status(200).json({
865   numOfPages: 1,
866   accountList: []
867 });
868 }
869 })
870 .catch(err => {
871   res.status(200).json({
872   numOfPages: 1,
873   accountList: []
874 });
875 });
876 }
877 );
878 // @route POST api/registrar/getfinalsubsect
879 // @desc Get subject section where class record status = 'F'
880 // @access Private
881
882 router.post(
883   "/getfinalsubsect",
884   passport.authenticate(directorRegistrar, {
885     session: false
886   }),
887   async(req, res) => {
888     let {
889       teacherID,
890       page,
891       pageSize,
892       quarter
893     } = req.body;
894     page = page - 1;
895     let offset = page * pageSize;
896     let limit = offset + pageSize;
897     const {
898       schoolYearID
899     } = await utils.getActiveSY();
900     const deadline = await SubmissionDeadline.findOne({
901       where: {
902         teacherID,
903         isActive: 1
904       }
905     }).then(sd => {
906       if (sd) {
907         return sd.deadline;
908       } else {
909         return "NOT SET";
910       }
911     });
912     const {
913       classRecordIDs,
914       crStatus
915     } = await SubjectSection.findAll({
916       where: {
917         teacherID,
918         schoolYearID,
919         subjectType: {
920           [Op.in]: quarter == "Q1" || quarter == "Q2" ?
921             ["NON_SHS", "1ST_SEM"] : ["NON_SHS", "2ND_SEM"]
922         }
923       }
924     }).then(async ss => {
925       if (ss) {
926         let data = [];
927         let crStatus = []
928         for (const [i, v] of ss.entries()) {

```

```

929     const crs = await ClassRecordStatus.findOne({
930         where: [
931             { classRecordID: v.classRecordID }
932         ]
933     });
934     if (quarter == "Q1" || quarter == "Q2") {
935         if (crs[quarter.toLowerCase()] == "F" || crs[quarter.toLowerCase()] == "D") {
936             data.push(v.classRecordID);
937             crStatus.push({
938                 classRecordID: v.classRecordID,
939                 status: crs[quarter.toLowerCase()]
940             })
941         }
942     } else {
943         if (quarter == "Q3") {
944             if (v.subjectType == "NON_SHS") {
945                 if (crs[quarter.toLowerCase()] == "F" || crs[quarter.toLowerCase()] == "D") {
946                     data.push(v.classRecordID);
947                     crStatus.push({
948                         classRecordID: v.classRecordID,
949                         status: crs[quarter.toLowerCase()]
950                     })
951                 }
952             } else {
953                 if (crs["q1"] == "F" || crs["q1"] == "D") {
954                     data.push(v.classRecordID);
955                     crStatus.push({
956                         classRecordID: v.classRecordID,
957                         status: crs['q1']
958                     })
959                 }
960             }
961         } else {
962             if (v.subjectType == "NON_SHS") {
963                 if (crs[quarter.toLowerCase()] == "F" || crs[quarter.toLowerCase()] == "D") {
964                     data.push(v.classRecordID);
965                     crStatus.push({
966                         classRecordID: v.classRecordID,
967                         status: crs[quarter.toLowerCase()]
968                     })
969                 }
970             } else {
971                 if (crs["q2"] == "F" || crs["q2"] == "D") {
972                     data.push(v.classRecordID);
973                     crStatus.push({
974                         classRecordID: v.classRecordID,
975                         status: crs['q2']
976                     })
977                 }
978             }
979         }
980     }
981     return {
982         classRecordIDs: data,
983         crStatus
984     };
985 }
986 }
987 });
988 if (classRecordIDs.length == 0) {
989     res.status(404).json({
990         msg: "Class Record not found!"
991     });
992 } else {
993     let condition = {};
994     condition["classRecordID"] = {
995         [Op.in]: classRecordIDs
996     };

```

```

997     await ClassRecordStatus.findAll({
998       limit,
999       offset,
1000      where: condition
1001    }).then(async crs => {
1002      if (crs) {
1003        if (crs.length == 0) {
1004          res.status(404).json({
1005            msg: "No class record for deliberation found"
1006          });
1007        } else {
1008          let data2 = [];
1009          let numOfPages = 1;
1010          crs.slice(0, pageSize).forEach(async(v, k, r) => {
1011            const {
1012              classRecordID
1013            } = v;
1014            const subjectType = await utils.
1015              getSubjectTypeByClassRecordID(
1016                classRecordID
1017              );
1018            const status = crStatus.find(a => a.classRecordID ==
1019              classRecordID).status
1020            let dateSubmitted = v['${quarter.toLowerCase()}'
1021              DateSubmitted'];
1022            if (subjectType == "2ND_SEM") {
1023              if (quarter == "Q3") {
1024                dateSubmitted = v.q1DateSubmitted;
1025              } else if (quarter == "Q4") {
1026                dateSubmitted = v.q2DateSubmitted;
1027              }
1028            }
1029            const {
1030              subjectCode,
1031              subjectName,
1032              section,
1033              subsectID
1034            } = await SubjectSection.findOne({
1035              where: {
1036                classRecordID
1037              }
1038            }).then(async cr => {
1039              if (cr) {
1040                const subjectCode = await utils.getSubjectCode(cr.
1041                  subjectID);
1042                const subsectID
1043                  = cr;
1044                const subjectName = await utils.getSubjectName(cr.
1045                  subjectID);
1046                const section = await utils.getSectionName(cr.
1047                  sectionID);
1048                return {
1049                  subjectCode,
1050                  subsectID,
1051                  subjectName,
1052                  section
1053                };
1054              }
1055            });
1056            data2.push({
1057              status,
1058              classRecordID,
1059              dateSubmitted,
1060              subjectCode,
1061              subjectName,
1062              section,
1063              subsectID
1064            });
1065            if (k == r.length - 1) {
1066              numOfPages = await ClassRecordStatus.findAndCountAll
1067                ({
1068                  where: condition

```

```

1063             }).then(count => {
1064                 return Math.ceil(count.count / pageSize);
1065             });
1066             res.status(200).json({
1067                 classRecordList: data2,
1068                 numOfPages,
1069                 deadline
1070             });
1071         }
1072     );
1073 }
1074 );
1075 );
1076 );
1077 );
1078 );
1079
1080 // @route POST api/registrar/getpastrecords
1081 // @desc Get past records of students by grade level
1082 // @access Private
1083
1084 router.post(
1085     "/getpastrecords",
1086     passport.authenticate("registrar", {
1087         session: false
1088     }),
1089     async(req, res) => {
1090         let {
1091             page,
1092             pageSize,
1093             keyword
1094         } = req.body;
1095         const {
1096             gradeLevel
1097         } = req.body;
1098         let pastSY = await utils.getPastSY();
1099         let pastSYID = await utils.getSYID(pastSY);
1100         const gradeLevels = [
1101             "N",
1102             "K1",
1103             "K2",
1104             "G1",
1105             "G2",
1106             "G3",
1107             "G4",
1108             "G5",
1109             "G6",
1110             "G7",
1111             "G8",
1112             "G9",
1113             "G10",
1114             "G11",
1115             "G12"
1116         ];
1117         const gradeLevelIndex = gradeLevels.indexOf(gradeLevel);
1118         let pastGradeLevel =
1119             gradeLevelIndex == 0 ?
1120                 gradeLevels[gradeLevelIndex] :
1121                 gradeLevels[gradeLevelIndex - 1];
1122         let sectionsID = await utils.getSectionsIDByGradeLevel(
1123             pastGradeLevel);
1124         let recordsData = {
1125             numOfPages: 1,
1126             studentData: []
1127         };
1128         if (sectionsID.length != 0) {
1129             recordsData = await utils.getStudentSectionBySYAndSectionID({
1130                 sectionID: sectionsID,
1131                 schoolYearID: pastSYID,
1132                 page,
1133                 pageSize,
1134                 keyword
1135             });
1136             res.status(200).json(recordsData);

```

```

1136         } else {
1137             res.status(404).json({
1138                 msg: "Not found!"
1139             });
1140         }
1141     );
1142 );
1143
1144 // @route POST api/registrar/searchstudent
1145 // @desc Suggestion search for students (5 entries)
1146 // @access Private
1147
1148 router.post(
1149     "/searchstudent",
1150     passport.authenticate("registrar", {
1151         session: false
1152     }),
1153     async(req, res) => {
1154         const {
1155             keyword
1156         } = req.body;
1157         let page = 1;
1158         let pageSize = 5;
1159         page = page - 1;
1160         let offset = page * pageSize;
1161         let limit = offset + pageSize;
1162         UserAccount.findAll({
1163             limit,
1164             offset,
1165             where: [
1166                 {
1167                     position: 4,
1168                     [Op.or]: [
1169                         {
1170                             firstName: [
1171                                 [Op.like]: `%"${keyword}"%`
1172                             ]
1173                         },
1174                         {
1175                             lastName: [
1176                                 [Op.like]: `%"${keyword}"%`
1177                             ]
1178                         }
1179                     ]
1180                 }
1181             ]
1182         })
1183         .then(users => {
1184             let studentData = [];
1185             if (users.length != 0) {
1186                 users.slice(0, pageSize).forEach(async(user, key, arr) => {
1187                     const keyID = await utils.getStudentID(user.accountID);
1188                     const name = `${utils.capitalize(
1189                         user.lastName
1190                     )}, ${utils.capitalize(user.firstName)} ${user.middleName
1191                     .charAt(0)
1192                     .toUpperCase()}`;
1193                     const imageUrl = user.imageUrl;
1194                     studentData.push({
1195                         key: keyID,
1196                         name,
1197                         imageUrl
1198                     });
1199                     if (key == arr.length - 1) {
1200                         studentData.sort((a, b) => (a.name > b.name ? 1 : -1));
1201                         res.status(200).json({
1202                             accountList: studentData
1203                         });
1204                     }
1205                 });
1206             } else {
1207                 res.status(404).json({
1208                     msg: "Not found"
1209                 });
1210             }
1211         })
1212         .catch(err => {

```

```
1208     res.status(404).json({
1209         msg: "Not found"
1210     });
1211 });
1212 );
1213 );
1214 );
1215 // @route POST api/registrar/searchteacher
1216 // @desc Suggestion search for teacher (5 entries)
1217 // @access Private
1218
1219 router.post(
1220     "/searchteacher",
1221     passport.authenticate("registrar", {
1222         session: false
1223     }),
1224     async(req, res) => {
1225         const {
1226             keyword
1227         } = req.body;
1228         let page = 1;
1229         let pageSize = 5;
1230         page = page - 1;
1231         let offset = page * pageSize;
1232         let limit = offset + pageSize;
1233         UserAccount.findAll({
1234             limit,
1235             offset,
1236             where: [
1237                 {
1238                     position: 3,
1239                     [Op.or]: [
1240                         {
1241                             firstName: {
1242                                 [Op.like]: `%${keyword}%`
1243                             }
1244                         },
1245                         {
1246                             lastName: {
1247                                 [Op.like]: `%${keyword}%`
1248                             }
1249                         }
1250                     ]
1251                 }
1252             .then(users => {
1253                 let teacherData = [];
1254                 if (users.length != 0) {
1255                     users.slice(0, pageSize).forEach(async(user, key, arr) => {
1256                         const keyID = await utils.getTeacherID(user.accountID);
1257                         const name = `${utils.capitalize(
1258                             user.lastName
1259                         )}, ${utils.capitalize(user.firstName)} ${user.middleName
1260                         .charAt(0)
1261                         .toUpperCase()}`;
1262                         const imageUrl = user.imageUrl;
1263                         teacherData.push({
1264                             key: keyID,
1265                             name,
1266                             imageUrl
1267                         });
1268                         if (key == arr.length - 1) {
1269                             teacherData.sort((a, b) => (a.name > b.name ? 1 : -1));
1270                             res.status(200).json({
1271                                 accountList: teacherData
1272                             });
1273                         }
1274                     });
1275                 } else {
1276                     res.status(404).json({
1277                         msg: "Not found"
1278                     });
1279                 }
1280             })
1281             .catch(err => {
1282                 res.status(404).json({
1283                     msg: "Not found"
1284                 });
1285             });
1286         );
1287     });
1288 
```

```

1281         });
1282     });
1283   );
1284 );
1285
1286 // @route POST api/registrar/searchenrolled
1287 // @desc Suggestion search for enrolled students (5 entries)
1288 // @access Private
1289
1290 router.post(
1291   "/searchenrolled",
1292   passport.authenticate("registrar", {
1293     session: false
1294   }),
1295   async(req, res) => {
1296     const {
1297       keyword
1298     } = req.body;
1299     let page = 1;
1300     let pageSize = 5;
1301     page = page - 1;
1302     let offset = page * pageSize;
1303     let limit = offset + pageSize;
1304     let studentsID = await utils.getStudentsIDByKeyword(keyword);
1305     let {
1306       schoolYearID,
1307       quarter
1308     } = await utils.getActiveSY();
1309     if (studentsID.length == 0) {
1310       res.status(404).json({
1311         msg: "Not found!"
1312       });
1313     } else {
1314       StudentSection.findAll({
1315         limit,
1316         offset,
1317         where: {
1318           studentID: studentsID,
1319           schoolYearID
1320         }
1321       }).then(async studentsections => {
1322         if (studentsections.length != 0) {
1323           let i = 0;
1324           let studsectarr = [];
1325           for (i; i < studentsections.slice(0, pageSize).length; i++) {
1326             let studsectID = studentsections[i].studsectID;
1327             let key = studentsections[i].studentID;
1328             let name = await utils.getStudentName(studentsections[i].
1329               studentID);
1330             let email = await utils.getStudentEmail(
1331               studentsections[i].studentID
1332             );
1333             let imageUrl = await utils.getStudentImageUrl(
1334               studentsections[i].studentID
1335             );
1336             let gradeLevel = await utils.getSectionGradeLevel(
1337               studentsections[i].sectionID
1338             );
1339             let sectionName = await utils.getSectionName(
1340               studentsections[i].sectionID
1341             );
1342             studsectarr.push({
1343               key,
1344               name,
1345               email,
1346               imageUrl,
1347               gradeLevel,
1348               sectionName,
1349               studsectID
1350             });
1351           res.status(200).json({

```

```

1352         studentList: studsectarr
1353     });
1354   } else {
1355     res.status(404).json({
1356       msg: "Not found!"
1357     });
1358   }
1359 });
1360 }
1361 );
1362 );
1363
1364 // @route POST api/registrar/getenrolled
1365 // @desc Get enrolled students by school year and section
1366 // @access Private
1367
1368 router.post(
1369   "/getenrolled",
1370   passport.authenticate("registrar", {
1371     session: false
1372   }),
1373   (req, res) => {}
1374 );
1375
1376 // @route GET api/registrar/createsy
1377 // @desc Create a school year
1378 // @access Private
1379
1380 router.post(
1381   "/createsy",
1382   passport.authenticate("registrar", {
1383     session: false
1384   }),
1385   async(req, res) => {
1386     const {
1387       schoolYear
1388     } = req.body;
1389     let checker = schoolYear.split("-");
1390     if (checker.length != 2) {
1391       res.status(400).json({
1392         msg: "Invalid school year format"
1393       });
1394     } else {
1395       if (parseInt(checker[1]) - parseInt(checker[0]) != 1) {
1396         res.status(400).json({
1397           msg: "Invalid school year format"
1398         });
1399       } else {
1400         SchoolYear.findOne({
1401           where: {
1402             isActive: 1
1403           }
1404         }).then(sy => {
1405           if (sy) {
1406             res.status(400).json({
1407               msg: "Invalid operation. There is an active school year
1408             "
1409           });
1410         } else {
1411           SchoolYear.findOne({
1412             where: {
1413               schoolYear
1414             }
1415           }).then(sy2 => {
1416             if (sy2) {
1417               res.status(404).json({
1418                 msg: "Invalid operation. School year already exists
1419               "
1420             });
1421           } else {
1422             SchoolYear.create({
1423               schoolYear,

```

```

1422             isActive: 1,
1423             quarter: "Q1"
1424         }).then(() => {
1425             res.status(200).json({
1426                 msg: "School year created successfully!"
1427             });
1428         });
1429     });
1430 });
1431 });
1432 );
1433 });
1434 );
1435 );
1436 );
1437
1438 // @route POST api/registrar/endschoolyear
1439 // @desc Set school year isActive value to 0
1440 // @access Private
1441
1442 router.post(
1443     "/endschoolyear",
1444     passport.authenticate("registrar", {
1445         session: false
1446     }),
1447     async(req, res) => {
1448         const {
1449             schoolYearID
1450         } = req.body;
1451         SchoolYear.findOne({
1452             where: {
1453                 schoolYearID
1454             }
1455         }).then(sy => {
1456             if (sy) {
1457                 sy.update({
1458                     isActive: 0
1459                 }).then(() => {
1460                     res.status(200).json({
1461                         msg: "School year has ended successfully!"
1462                     });
1463                 });
1464             } else {
1465                 res.status(404).json({
1466                     msg: "School year not found"
1467                 });
1468             }
1469         });
1470     }
1471 );
1472
1473 // @route POST api/registrar/listssubjectsection
1474 // @desc List subject section by schoolyearID and quarter and section
1475 // @access Private
1476
1477 router.post(
1478     "/listssubjectsection",
1479     passport.authenticate("registrar", {
1480         session: false
1481     }),
1482     async(req, res) => {
1483         const {
1484             sectionID
1485         } = req.body;
1486         const {
1487             schoolYearID,
1488             quarter
1489         } = await utils.getActiveSY();
1490         const data = await SubjectSection.findAll({
1491             where: {
1492                 sectionID,
1493                 schoolYearID

```

```

1494         }
1495     }).then(async ss => {
1496         if (ss.length != 0) {
1497             let data = [];
1498             for (const [index, value] of ss.entries()) {
1499                 const {
1500                     subsectID,
1501                     subjectType,
1502                     subjectID,
1503                     teacherID
1504                 } = value;
1505                 const subjectName = await utils.getSubjectName(subjectID);
1506                 const teacher = await utils.getTeacherName(teacherID);
1507                 if (subjectType == "NON_SHS") {
1508                     data.push({
1509                         subsectID,
1510                         subjectName,
1511                         teacher
1512                     });
1513                 } else {
1514                     if (subjectType == "1ST_SEM") {
1515                         if (quarter == "Q1" || quarter == "Q2") {
1516                             data.push({
1517                                 subsectID,
1518                                 subjectName,
1519                                 teacher
1520                             });
1521                         }
1522                     } else if (subjectType == "2ND_SEM") {
1523                         if (quarter == "Q3" || quarter == "Q4") {
1524                             data.push({
1525                                 subsectID,
1526                                 subjectName,
1527                                 teacher
1528                             });
1529                         }
1530                     }
1531                 }
1532             }
1533             return data;
1534         } else {
1535             return [];
1536         }
1537     });
1538     res.status(200).json({
1539         subjectList: data
1540     });
1541 }
1542 );
1543
1544 // @route POST api/registrar/changequartersy
1545 // @desc Change quarter of school year
1546 // @access Private
1547
1548 router.post(
1549     "/changequartersy",
1550     passport.authenticate("registrar", {
1551         session: false
1552     }),
1553     async(req, res) => {
1554         const {
1555             schoolYearID,
1556             quarter
1557         } = req.body;
1558         SchoolYear.findOne({
1559             where: {
1560                 schoolYearID
1561             }
1562         }).then(sy => {
1563             if (sy) {
1564                 sy.update({
1565                     quarter
1566                 }).then(async() => {

```

```

1567     const classRecordIDs = await SubjectSection.findAll({
1568       where: {
1569         schoolYearID
1570       }
1571     }).then(async ss => {
1572       if (ss) {
1573         let data = [];
1574         for (const [index, value] of ss.entries())
1575           data.push(value.classRecordID);
1576         return data;
1577       }
1578     });
1579     ClassRecordStatus.findAll({
1580       where: {
1581         classRecordID: {
1582           [Op.in]: classRecordIDs
1583         }
1584       }
1585     }).then(async crs => {
1586       if (crs) {
1587         for (const [index, value] of crs.entries()) {
1588           const subjectType = await utils.
1589             getSubjectTypeByClassRecordID(
1590               value.classRecordID
1591             );
1592             const {
1593               sectionID,
1594               subjectID
1595             } = await SubjectSection.findOne({
1596               where: {
1597                 classRecordID: value.classRecordID
1598               }
1599             }).then(ss => {
1600               if (ss) {
1601                 const {
1602                   sectionID,
1603                   subjectID
1604                 } = ss;
1605                 return {
1606                   sectionID,
1607                   subjectID
1608                 };
1609               }
1610             });
1611             if (quarter == "Q1") {
1612               if (
1613                 value.q1 == "L" &&
1614                 (subjectType == "NON_SHS" || subjectType == "1
1615                   ST_SEM")
1616               ) {
1617                 await value.update({
1618                   q1: "E"
1619                 },
1620                   {
1621                     position: "Registrar",
1622                     classRecordID: value.classRecordID,
1623                     type: "CHANGE_STATUS",
1624                     accountID: req.user.accountID,
1625                     sectionID,
1626                     subjectID,
1627                     oldVal: "L",
1628                     newVal: "E",
1629                     quarter: "Q1"
1630                   });
1631               }
1632             if (
1633               value.q2 == "E" &&
1634               (subjectType == "NON_SHS" || subjectType == "1
1635                   ST_SEM")
1636             ) {
1637               await value.update({
1638                 q2: "L"
1639               },
1640                 {
1641                   position: "Registrar",

```

```

1637         classRecordID: value.classRecordID,
1638         type: "CHANGE_STATUS",
1639         accountId: req.user.accountID,
1640         sectionID,
1641         subjectID,
1642         oldVal: "E",
1643         newVal: "L",
1644         quarter: "Q2"
1645     });
1646 }
1647 if (value.q3 == "E" && subjectType == "NON_SHS") {
1648     await value.update({
1649         q3: "L"
1650     },
1651     position: "Registrar",
1652     classRecordID: value.classRecordID,
1653     type: "CHANGE_STATUS",
1654     accountId: req.user.accountID,
1655     sectionID,
1656     subjectID,
1657     oldVal: "E",
1658     newVal: "L",
1659     quarter: "Q3"
1660 });
1661 }
1662 if (value.q1 == "E" && subjectType == "2ND_SEM") {
1663     await value.update({
1664         q1: "L"
1665     },
1666     position: "Registrar",
1667     classRecordID: value.classRecordID,
1668     type: "CHANGE_STATUS",
1669     accountId: req.user.accountID,
1670     sectionID,
1671     subjectID,
1672     oldVal: "E",
1673     newVal: "L",
1674     quarter: "Q1"
1675 });
1676 }
1677 if (value.q4 == "E" && subjectType == "NON_SHS") {
1678     await value.update({
1679         q4: "L"
1680     },
1681     position: "Registrar",
1682     classRecordID: value.classRecordID,
1683     type: "CHANGE_STATUS",
1684     accountId: req.user.accountID,
1685     sectionID,
1686     subjectID,
1687     oldVal: "E",
1688     newVal: "L",
1689     quarter: "Q4"
1690 });
1691 }
1692 if (value.q2 == "E" && subjectType == "2ND_SEM") {
1693     await value.update({
1694         q2: "L"
1695     },
1696     position: "Registrar",
1697     classRecordID: value.classRecordID,
1698     type: "CHANGE_STATUS",
1699     accountId: req.user.accountID,
1700     sectionID,
1701     subjectID,
1702     oldVal: "E",
1703     newVal: "L",
1704     quarter: "Q2"
1705 });
1706 }
1707 }
1708 }
1709 if (quarter == "Q2") {
1710     if (
1711         value.q2 == "L" &&

```

```

1713             (subjectType == "NON_SHS" || subjectType == "1
1714                 ST_SEM")
1715         ) {
1716             await value.update({
1717                 q2: "E"
1718             }, {
1719                 position: "Registrar",
1720                 classRecordID: value.classRecordID,
1721                 type: "CHANGE_STATUS",
1722                 accountID: req.user.accountID,
1723                 sectionID,
1724                 subjectID,
1725                 oldVal: "L",
1726                 newVal: "E",
1727                 quarter: "Q2"
1728             });
1729         }
1730         if (
1731             value.q1 == "E" &&
1732             (subjectType == "NON_SHS" || subjectType == "1
1733                 ST_SEM")
1734         ) {
1735             await value.update({
1736                 q1: "L"
1737             }, {
1738                 position: "Registrar",
1739                 classRecordID: value.classRecordID,
1740                 type: "CHANGE_STATUS",
1741                 accountID: req.user.accountID,
1742                 sectionID,
1743                 subjectID,
1744                 oldVal: "E",
1745                 newVal: "L",
1746                 quarter: "Q1"
1747             });
1748         if (value.q3 == "E" && subjectType == "NON_SHS") {
1749             await value.update({
1750                 q3: "L"
1751             }, {
1752                 position: "Registrar",
1753                 classRecordID: value.classRecordID,
1754                 type: "CHANGE_STATUS",
1755                 accountID: req.user.accountID,
1756                 sectionID,
1757                 subjectID,
1758                 oldVal: "E",
1759                 newVal: "L",
1760                 quarter: "Q3"
1761             });
1762         if (value.q1 == "E" && subjectType == "2ND_SEM") {
1763             await value.update({
1764                 q1: "L"
1765             }, {
1766                 position: "Registrar",
1767                 classRecordID: value.classRecordID,
1768                 type: "CHANGE_STATUS",
1769                 accountID: req.user.accountID,
1770                 sectionID,
1771                 subjectID,
1772                 oldVal: "E",
1773                 newVal: "L",
1774                 quarter: "Q1"
1775             });
1776         }
1777         if (value.q4 == "E" && subjectType == "NON_SHS") {
1778             await value.update({
1779                 q4: "L"
1780             }, {
1781                 position: "Registrar",
1782                 classRecordID: value.classRecordID,
1783                 type: "CHANGE_STATUS",
1784                 accountID: req.user.accountID,
1785                 sectionID,
1786                 subjectID,

```

```

1787             oldVal: "E",
1788             newVal: "L",
1789             quarter: "Q4"
1790         });
1791     }
1792     if (value.q2 == "E" && subjectType == "2ND_SEM") {
1793         await value.update({
1794             q2: "L"
1795         }, {
1796             position: "Registrar",
1797             classRecordID: value.classRecordID,
1798             type: "CHANGE_STATUS",
1799             accountID: req.user.accountID,
1800             sectionID,
1801             subjectID,
1802             oldVal: "E",
1803             newVal: "L",
1804             quarter: "Q2"
1805         });
1806     }
1807 }
1808
1809 if (quarter == "Q3") {
1810     if (value.q3 == "L" && subjectType == "NON_SHS") {
1811         await value.update({
1812             q3: "E"
1813         }, {
1814             position: "Registrar",
1815             classRecordID: value.classRecordID,
1816             type: "CHANGE_STATUS",
1817             accountID: req.user.accountID,
1818             sectionID,
1819             subjectID,
1820             oldVal: "L",
1821             newVal: "E",
1822             quarter: "Q3"
1823         });
1824     }
1825     if (value.q1 == "L" && subjectType == "2ND_SEM") {
1826         await value.update({
1827             q1: "E"
1828         }, {
1829             position: "Registrar",
1830             classRecordID: value.classRecordID,
1831             type: "CHANGE_STATUS",
1832             accountID: req.user.accountID,
1833             sectionID,
1834             subjectID,
1835             oldVal: "L",
1836             newVal: "E",
1837             quarter: "Q1"
1838         });
1839     }
1840     if (
1841         value.q2 == "E" &&
1842         (subjectType == "NON_SHS" || subjectType == "1
1843 ST_SEM")
1844     ) {
1845         await value.update({
1846             q2: "L"
1847         }, {
1848             position: "Registrar",
1849             classRecordID: value.classRecordID,
1850             type: "CHANGE_STATUS",
1851             accountID: req.user.accountID,
1852             sectionID,
1853             subjectID,
1854             oldVal: "E",
1855             newVal: "L",
1856             quarter: "Q2"
1857         });
1858     if (
1859         value.q1 == "E" &&
1860         (subjectType == "NON_SHS" || subjectType == "1

```

```

        ST_SEM")
1861 ) {
1862     await value.update({
1863         q1: "L"
1864     }, {
1865         position: "Registrar",
1866         classRecordID: value.classRecordID,
1867         type: "CHANGE_STATUS",
1868         accountID: req.user.accountID,
1869         sectionID,
1870         subjectID,
1871         oldVal: "E",
1872         newVal: "L",
1873         quarter: "Q1"
1874     });
1875 }
1876 if (value.q4 == "E" && subjectType == "NON_SHS") {
1877     await value.update({
1878         q4: "L"
1879     }, {
1880         position: "Registrar",
1881         classRecordID: value.classRecordID,
1882         type: "CHANGE_STATUS",
1883         accountID: req.user.accountID,
1884         sectionID,
1885         subjectID,
1886         oldVal: "E",
1887         newVal: "L",
1888         quarter: "Q4"
1889     });
1890 }
1891 if (value.q2 == "E" && subjectType == "2ND_SEM") {
1892     await value.update({
1893         q2: "L"
1894     }, {
1895         position: "Registrar",
1896         classRecordID: value.classRecordID,
1897         type: "CHANGE_STATUS",
1898         accountID: req.user.accountID,
1899         sectionID,
1900         subjectID,
1901         oldVal: "E",
1902         newVal: "L",
1903         quarter: "Q2"
1904     });
1905 }
1906 }
1907 if (quarter == "Q4") {
1908     if (value.q4 == "L" && subjectType == "NON_SHS") {
1909         await value.update({
1910             q4: "E"
1911         }, {
1912             position: "Registrar",
1913             classRecordID: value.classRecordID,
1914             type: "CHANGE_STATUS",
1915             accountID: req.user.accountID,
1916             sectionID,
1917             subjectID,
1918             oldVal: "L",
1919             newVal: "E",
1920             quarter: "Q4"
1921         });
1922     }
1923     if (value.q2 == "L" && subjectType == "2ND_SEM") {
1924         await value.update({
1925             q2: "E"
1926         }, {
1927             position: "Registrar",
1928             classRecordID: value.classRecordID,
1929             type: "CHANGE_STATUS",
1930             accountID: req.user.accountID,
1931             sectionID,
1932             subjectID,
1933             oldVal: "L",
1934             newVal: "E",
1935         })
}

```

```

1936         quarter: "Q2"
1937     });
1938 }
1939 if (
1940     value.q2 == "E" &&
1941     (subjectType == "NON_SHS" || subjectType == "1
1942     ST_SEM")
1943 ) {
1944     await value.update({
1945         q2: "L"
1946     }, {
1947         position: "Registrar",
1948         classRecordID: value.classRecordID,
1949         type: "CHANGE_STATUS",
1950         accountID: req.user.accountID,
1951         sectionID,
1952         subjectID,
1953         oldVal: "E",
1954         newVal: "L",
1955         quarter: "Q2"
1956     });
1957 if (value.q3 == "E" && subjectType == "NON_SHS") {
1958     await value.update({
1959         q3: "L"
1960     }, {
1961         position: "Registrar",
1962         classRecordID: value.classRecordID,
1963         type: "CHANGE_STATUS",
1964         accountID: req.user.accountID,
1965         sectionID,
1966         subjectID,
1967         oldVal: "E",
1968         newVal: "L",
1969         quarter: "Q3"
1970     });
1971 if (value.q1 == "E" && subjectType == "2ND_SEM") {
1972     await value.update({
1973         q1: "L"
1974     }, {
1975         position: "Registrar",
1976         classRecordID: value.classRecordID,
1977         type: "CHANGE_STATUS",
1978         accountID: req.user.accountID,
1979         sectionID,
1980         subjectID,
1981         oldVal: "E",
1982         newVal: "L",
1983         quarter: "Q1"
1984     });
1985 }
1986 if (
1987     value.q1 == "E" &&
1988     (subjectType == "NON_SHS" || subjectType == "1
1989     ST_SEM")
1990 ) {
1991     await value.update({
1992         q1: "L"
1993     }, {
1994         position: "Registrar",
1995         classRecordID: value.classRecordID,
1996         type: "CHANGE_STATUS",
1997         accountID: req.user.accountID,
1998         sectionID,
1999         subjectID,
2000         oldVal: "E",
2001         newVal: "L",
2002         quarter: "Q1"
2003     });
2004 }
2005 }
2006 }
2007 }
2008 res.status(200).json({

```

```

2009             msg: "Quarter successfully updated"
2010         });
2011     });
2012   });
2013 });
2014 } else {
2015   res.status(404).json({
2016     msg: "School year not found!"
2017   });
2018 }
2019 );
2020 );
2021 );
2022 );
2023 // @route POST api/registrar/getsy
2024 // @desc Get school year info by schoolYearID
2025 // @access Private
2026
2027 router.post(
2028   "/getsy",
2029   passport.authenticate(["registrar", "director"], {
2030     session: false
2031   }),
2032   async(req, res) => {
2033     let {
2034       schoolYearID
2035     } = req.body;
2036     let schoolYear = await utils.getSYname(schoolYearID);
2037     res.status(200).json({
2038       schoolYear,
2039       schoolYearID
2040     });
2041   }
2042 );
2043
2044 // @route GET api/registrar/getsy
2045 // @desc Get current school year
2046 // @access Private
2047 router.get(
2048   "/getsy",
2049   passport.authenticate(["registrar", "director"], {
2050     session: false
2051   }),
2052   async(req, res) => {
2053     let {
2054       schoolYearID,
2055       quarter
2056     } = await utils.getActiveSY();
2057     if (schoolYearID != 0) {
2058       let schoolYear = await utils.getSYname(schoolYearID);
2059       res.status(200).json({
2060         schoolYear,
2061         schoolYearID,
2062         quarter
2063       });
2064     } else {
2065       res.status(404).json({
2066         msg: "There is no active school year"
2067       });
2068     }
2069   }
2070 );
2071
2072 // @route GET api/registrar/getallsy
2073 // @desc Get all school year
2074 // @access Private
2075
2076 router.get(
2077   "/getallsy",
2078   passport.authenticate(["registrar", 'director'], {
2079     session: false
2080   }),

```

```

2081     async(req, res) => {
2082       SchoolYear.findAll().then(async sys => {
2083         if (sys) {
2084           let data = [];
2085           for (const [index, value] of sys.entries()) {
2086             data.push({
2087               schoolYearID: value.schoolYearID,
2088               schoolYear: value.schoolYear
2089             });
2090           }
2091           data.sort((a, b) => (a.schoolYear < b.schoolYear ? 1 : -1));
2092           res.status(200).json({
2093             schoolYearList: data
2094           });
2095         }
2096       });
2097     );
2098   );
2099 
2100 // @route GET api/registrar/getpastsy
2101 // @desc Get past school year
2102 // @access Private
2103 
2104 router.get(
2105   "/getpastsy",
2106   passport.authenticate("registrar", {
2107     session: false
2108   }),
2109   async(req, res) => {
2110     let pastSY = await utils.getPastSY();
2111     let pastSYID = await utils.getSYID(pastSY);
2112     res.status(200).json({
2113       schoolYearID: pastSYID,
2114       schoolYear: pastSY
2115     });
2116   }
2117 );
2118 
2119 // @route POST api/registrar/sectionname
2120 // @desc Get section name
2121 // @access Private
2122 
2123 router.post(
2124   "/sectionname",
2125   passport.authenticate("registrar", {
2126     session: false
2127   }),
2128   async(req, res) => {
2129     let {
2130       sectionID
2131     } = req.body;
2132     let sectionName = await utils.getSectionName(sectionID);
2133     res.status(200).json({
2134       sectionName
2135     });
2136   }
2137 );
2138 
2139 // @route POST api/registrar/getcurrentenrolled
2140 // @desc Get currently enrolled students by section
2141 // @access Private
2142 
2143 router.post(
2144   "/getcurrentenrolled",
2145   passport.authenticate("registrar", {
2146     session: false
2147   }),
2148   async(req, res) => {
2149     let {
2150       page,
2151       pageSize
2152     } = req.body;

```

```

2153     page = page - 1;
2154     let offset = page * pageSize;
2155     let limit = offset + pageSize;
2156     const {
2157       sectionID
2158     } = req.body;
2159     const {
2160       schoolYearID,
2161       quarter
2162     } = await utils.getActiveSY();
2163     const gradeLevel = await utils.getGradeLevelBySectionID(sectionID
2164     );
2165     StudentSection.findAll({
2166       limit,
2167       offset,
2168       where: {
2169         schoolYearID,
2170         sectionID
2171       }
2172     })
2173     .then(studentsections => {
2174       let studentData = [];
2175       if (studentsections.length == 0) {
2176         res.status(200).json({
2177           numOfPages: 1,
2178           studentList: [],
2179           gradeLevel
2180         });
2181       } else {
2182         studentsections
2183           .slice(0, pageSize)
2184           .forEach(async(studentsection, key, arr) => {
2185             const studsectID = studentsection.studsectID;
2186             const keyID = studentsection.studentID;
2187             const name = await utils.getStudentName(studentsection.
2188               studentID);
2189             const email = await utils.getStudentEmail(
2190               studentsection.studentID
2191             );
2192             const imageUrl = await utils.getStudentImageUrl(
2193               studentsection.studentID
2194             );
2195             const sectionName = await utils.getSectionName(
2196               sectionID);
2197             studentData.push({
2198               key: keyID,
2199               name,
2200               email,
2201               imageUrl,
2202               gradeLevel,
2203               sectionName,
2204               studsectID
2205             });
2206           if (key == arr.length - 1) {
2207             StudentSection.findAndCountAll({
2208               where: {
2209                 schoolYearID,
2210                 sectionID
2211               }
2212             })
2213             .then(count => {
2214               studentData.sort((a, b) => {
2215                 a.name > b.name ? 1 : -1;
2216               });
2217               res.status(200).json({
2218                 numOfPages: Math.ceil(count.count / pageSize),
2219                 studentList: studentData,
2220                 gradeLevel
2221               });
2222             })
2223             .catch(err => {
2224               res.status(404);
2225             });
2226           }
2227         });
2228       }
2229     });

```

```

2223         }
2224     });
2225   });
2226 }
2227 .catch(err => {
2228   res.status(404);
2229 });
2230 }
2231 );
2232
2233 // @route POST api/registrar/createstudentsection
2234 // @desc Add student to section
2235 // @access Private
2236
2237 router.post(
2238   "/createstudentsection",
2239   passport.authenticate("registrar", {
2240     session: false
2241   }),
2242   (req, res) => {
2243     const {
2244       studentID,
2245       sectionID,
2246       schoolYearID
2247     } = req.body;
2248     if (studentID == -1) {
2249       res.status(400).json({
2250         studentName: "You must select a student"
2251       });
2252     }
2253     StudentSection.findOne({
2254       where: {
2255         studentID,
2256         schoolYearID
2257       }
2258     }).then(studentsection => {
2259       if (studentsection) {
2260         res.status(400).json({
2261           studentName: "Student is already enrolled in a section"
2262         });
2263       } else {
2264         StudentSection.create({
2265           studentID,
2266           sectionID,
2267           schoolYearID
2268         }).then(async studentsection2 => {
2269           await StudentFinalGrade.create({
2270             studsectID: studentsection2.studsectID,
2271             grade: -1,
2272             schoolYearID
2273           });
2274           for (const [index, value] of ["Q1", "Q2", "Q3", "Q4"].entries()) {
2275             await StudentGrades.create({
2276               schoolYearID,
2277               quarter: value,
2278               studsectID: studentsection2.studsectID,
2279               grade: -1
2280             });
2281           }
2282           res.status(200).json({
2283             studentName: "Student enrolled successfully!",
2284             studsectID: studentsection2.studsectID
2285           });
2286         });
2287       }
2288     });
2289   });
2290 );
2291
2292 // @route POST api/registrar/deletestudentsection
2293 // @desc Unenroll student from a student section
2294 // @access Private

```

```

2295
2296   router.post(
2297     "/deletestudentsection",
2298     passport.authenticate("registrar", {
2299       session: false
2300     }),
2301     async(req, res) => {
2302       const {
2303         studentID,
2304         sectionID,
2305         schoolYearID
2306       } = req.body;
2307       StudentSection.findOne({
2308         where: {
2309           studentID,
2310           sectionID,
2311           schoolYearID
2312         }
2313       }).then(studentsection => {
2314         if (!studentsection) {
2315           res.status(400).json({
2316             msg: "Student section not found"
2317           });
2318         } else {
2319           SubjectSectionStudent.findAll({
2320             where: {
2321               studsectID: studentsection.studsectID
2322             }
2323           }).then(async ss => {
2324             if (ss.length != 0) {
2325               res.status(400).json({
2326                 msg: "Operation could not be completed. This student is
2327                   actively enrolled in a subject."
2328               });
2329             } else {
2330               studentsection.destroy().then(() => {
2331                 res.status(200).json({
2332                   msg: "Student unenrolled successfully!"
2333                 });
2334               });
2335             });
2336           }
2337         });
2338       });
2339     );
2340
2341 // @route POST api/registrar/createsubjectsection
2342 // @desc Create a subject load of teacher
2343 // @access Private
2344
2345   router.post(
2346     "/createsubjectsection",
2347     passport.authenticate("registrar", {
2348       session: false
2349     }),
2350     async(req, res) => {
2351       const {
2352         schoolYearID,
2353         quarter
2354       } = await utils.getActiveSY();
2355       const {
2356         payload,
2357         sectionID,
2358         subjectID,
2359         accountID
2360       } = req.body;
2361       const teacherID = await utils.getTeacherID(accountID);
2362       const subjectType = await Subject.findOne({
2363         where: {
2364           subjectID
2365         }
2366       }).then(subj => {

```

```

2367         if (subj) {
2368             return subj.subjectType;
2369         }
2370     });
2371     let isSHS = ![
2372         "N",
2373         "K1",
2374         "K2",
2375         "G1",
2376         "G2",
2377         "G3",
2378         "G4",
2379         "G5",
2380         "G6",
2381         "G7",
2382         "G8",
2383         "G9",
2384         "G10"
2385     ].includes(subjectType);
2386     let compareIn =
2387         quarter == "Q1" || quarter == "Q2" ?
2388         ["NON_SHS", "1ST_SEM"] :
2389         ["NON_SHS", "2ND_SEM"];
2390
2391     SubjectSection.findOne({
2392         where: {
2393             subjectID,
2394             sectionID,
2395             teacherID,
2396             schoolYearID,
2397             subjectType: {
2398                 [Op.in]: compareIn
2399             }
2400         }
2401     }).then(ss => {
2402         if (ss) {
2403             res.status(400).json({
2404                 msg: "Subject load already exists!"
2405             });
2406         } else {
2407             ClassRecord.create({
2408                 dateCreated: utils.getPHTime(),
2409                 dateModified: utils.getPHTime(),
2410                 isSubmitted: 0,
2411                 q1Transmu: "50",
2412                 q2Transmu: "50",
2413                 q3Transmu: "50",
2414                 q4Transmu: "50"
2415             })
2416             .then(classrecord => {
2417                 SubjectSection.create({
2418                     subjectID,
2419                     sectionID,
2420                     teacherID,
2421                     schoolYearID,
2422                     classRecordID: classrecord.classRecordID,
2423                     subjectType: isSHS ?
2424                         quarter == "Q1" || quarter == "Q2" ?
2425                         "1ST_SEM" :
2426                         "2ND_SEM" : "NON_SHS"
2427                 })
2428                 .then(async subjectsection => {
2429                     let objectSet = {
2430                         classRecordID: classrecord.classRecordID,
2431                         q1: "L",
2432                         q2: "L",
2433                         q3: "L",
2434                         q4: "L"
2435                     };
2436                     let subjectType = isSHS ?
2437                         quarter == "Q1" || quarter == "Q2" ?
2438                         "1ST_SEM" :
2439                         "2ND_SEM" :
2440                         "NON_SHS";
2441                     if (subjectType == "NON_SHS" || subjectType == "1

```

```

        ST_SEM") {
2442     objectSet[quarter.toLowerCase()] = "E";
2443 } else {
2444     objectSet["q1"] = quarter == "Q3" ? "E" : "L";
2445     objectSet["q2"] = quarter == "Q4" ? "E" : "L";
2446 }
2447 await ClassRecordStatus.create(objectSet);
2448 let i = 0;
2449 for (i; i < payload.length; i++) {
2450     SubjectSectionStudent.create({
2451         subsectID: subjectsection.subsectID,
2452         studsectID: payload[i].studsectID
2453     }).then(subsectstud => {
2454         let q = isSHS ? ["Q1", "Q2"] : ["Q1", "Q2", "Q3",
2455             "Q4"];
2456         let i = 0;
2457         for (i = 0; i < q.length; i++) {
2458             StudentWeightedScore.create({
2459                 subsectstudID: subsectstud.subsectstudID,
2460                 classRecordID: classrecord.classRecordID,
2461                 faWS: -1,
2462                 wwWS: -1,
2463                 ptWS: -1,
2464                 qeWS: -1,
2465                 finalGrade: -1,
2466                 quarter: q[i]
2467             });
2468         }
2469         StudentSubjectGrades.create({
2470             subsectstudID: subsectstud.subsectstudID,
2471             classRecordID: classrecord.classRecordID
2472         });
2473     });
2474 }
2475 Component.findOne({
2476     where: {
2477         subjectID,
2478         component: "FA"
2479     }
2480 }).then(comp1 => {
2481     if (comp1) {
2482         Component.findOne({
2483             where: {
2484                 subjectID,
2485                 component: "WW"
2486             }
2487         }).then(comp2 => {
2488             if (comp2) {
2489                 Component.findOne({
2490                     where: {
2491                         subjectID,
2492                         component: "PT"
2493                     }
2494                 }).then(comp3 => {
2495                     if (comp3) {
2496                         Component.findOne({
2497                             where: {
2498                                 subjectID,
2499                                 component: "QE"
2500                             }
2501                         }).then(comp4 => {
2502                             if (comp4) {
2503                                 let q = isSHS ?
2504                                     ["Q1", "Q2"] :
2505                                     ["Q1", "Q2", "Q3", "Q4"];
2506                                 let j = 0;
2507                                 for (j; j < q.length; j++) {
2508                                     Subcomponent.create({
2509                                         classRecordID: classrecord.
2510                                         classRecordID,
2511                                         name: "Subcomponent 1",
2512                                         componentID: comp1.componentID,
2513                                         compWeight: 100,

```

```

2512         quarter: q[j]
2513     });
2514     Subcomponent.create({
2515       classRecordID: classrecord.
2516         classRecordID,
2517         name: "Subcomponent 1",
2518         componentID: comp2.componentID,
2519         compWeight: 100,
2520         quarter: q[j]
2521     });
2522     Subcomponent.create({
2523       classRecordID: classrecord.
2524         classRecordID,
2525         name: "Subcomponent 1",
2526         componentID: comp3.componentID,
2527         compWeight: 100,
2528         quarter: q[j]
2529     });
2530     Subcomponent.create({
2531       classRecordID: classrecord.
2532         classRecordID,
2533         name: "Quarterly Exam",
2534         componentID: comp4.componentID,
2535         compWeight: 100,
2536         quarter: q[j]
2537     });
2538   }
2539   res.status(200).json({
2540     msg: "Added successfully!"
2541   });
2542 } else {
2543   res.status(404).json({
2544     msg: "QE component not found!"
2545   });
2546 } else {
2547   res.status(404).json({
2548     msg: "PT component not found!"
2549   });
2550 };
2551 } else {
2552   res.status(404).json({
2553     msg: "WW component not found!"
2554   });
2555 };
2556 } else {
2557   res.status(404).json({
2558     msg: "FA component not found!"
2559   });
2560 };
2561 };
2562 });
2563 });
2564 .catch(err => {
2565   res.status(400).json({
2566     msg: "Error adding subject section"
2567   });
2568 });
2569 });
2570 .catch(err => {
2571   res.status(400).json({
2572     msg: "Error adding class record"
2573   });
2574 });
2575 );
2576 });
2577 );
2578 );
2579 // @route POST api/registrar/deletesubjectsection
2580 // @desc Delete a subject load of teacher

```

```

2582 // @access Private
2583
2584 router.post(
2585   "/deletesubjectsection",
2586   passport.authenticate("registrar", {
2587     session: false
2588   }),
2589   async(req, res) => {
2590     const {
2591       subsectID
2592     } = req.body;
2593     SubjectSection.findOne({
2594       where: {
2595         subsectID
2596       }
2597     }).then(async subjectsection => {
2598       if (subjectsection) {
2599         SubjectSectionStudent.findAll({
2600           where: {
2601             subsectID
2602           }
2603         }).then(async sss => {
2604           if (sss.length != 0) {
2605             res.status(400).json({
2606               msg: "Operation could not be completed. Remove all
2607                 students under this subject first."
2608             });
2609           } else {
2610             await subjectsection.destroy().then(() => {
2611               res.status(200).json({
2612                 msg: "Subject load deleted successfully!"
2613               });
2614             });
2615           }
2616         } else {
2617           res.status(400).json({
2618             msg: "Error deleting subject load! 2"
2619           });
2620         }
2621       });
2622     });
2623   );
2624
2625 // @route POST api/registrar/studentprofile
2626 // @desc Get all personal information based on studentID
2627 // @access Private
2628
2629 router.post(
2630   "/studentprofile",
2631   passport.authenticate("registrar", {
2632     session: false
2633   }),
2634   async(req, res) => {
2635     const {
2636       studentID
2637     } = req.body;
2638     Student.findOne({
2639       where: {
2640         studentID
2641       }
2642     })
2643     .then(student => {
2644       UserAccount.findOne({
2645         where: {
2646           accountID: student.accountID
2647         }
2648       })
2649     .then(user => {
2650       const payload = {
2651         firstName: user.firstName,
2652         lastName: user.lastName,
2653         middleName: user.middleName,

```

```

2653         suffix: user.suffix,
2654         nickname: user.nickname,
2655         contactNum: user.contactNum,
2656         address: user.address,
2657         province: user.province,
2658         city: user.city,
2659         region: user.region,
2660         zipcode: user.zipcode,
2661         civilStatus: user.civilStatus,
2662         sex: user.sex,
2663         citizenship: user.citizenship
2664     };
2665     res.status(200).json(payload);
2666   });
2667 }
2668 .catch(err => {
2669   res.status(404).json({
2670     msg: "Student not found!"
2671   });
2672 });
2673 }
2674 );
2675
2676 // @route POST api/registrar/advisorytable
2677 // @desc Get all advisers from current school year
2678 // @access Private
2679
2680 router.post(
2681   "/advisorytable",
2682   passport.authenticate("registrar", {
2683     session: false
2684   }),
2685   async(req, res) => {
2686     let {
2687       page,
2688       pageSize,
2689       keyword
2690     } = req.body;
2691     page = page - 1;
2692     const offset = page * pageSize;
2693     const limit = offset + pageSize;
2694     const {
2695       schoolYearID
2696     } = await utils.getActiveSY();
2697     if (schoolYearID == 0) {
2698       res.status(404).json({
2699         msg: "There is no active school year"
2700       });
2701     } else {
2702       let data = [];
2703       let sectionsID = await utils.getSectionsID({
2704         limit,
2705         offset,
2706         keyword
2707       });
2708       let advisersData = await utils.getTeacherSectionBySchoolYearID(
2709         schoolYearID
2710       );
2711       let i = 0;
2712       for (i; i < sectionsID.slice(0, pageSize).length; i++) {
2713         let sectionID = sectionsID[i];
2714         let sectionName = await utils.getSectionName(sectionsID[i]);
2715         let adviser = "NO ADVISER";
2716         let teacherID = 0;
2717         let gradeLevel = await utils.getGradeLevelBySectionID(
2718           sectionsID[i]);
2719         data.push({
2720           sectionID,
2721           sectionName,
2722           adviser,
2723           teacherID,
2724           gradeLevel
2725         });
2726     }
2727   }

```

```

2726     i = 0;
2727     for (i; i < advisersData.length; i++) {
2728       let index = data.findIndex(
2729         val => val.sectionID == advisersData[i].sectionID
2730       );
2731       if (index != -1) {
2732         data[index].adviser = advisersData[i].teacherName;
2733         data[index].teacherID = advisersData[i].teacherID;
2734       }
2735     }
2736     Section.findAndCountAll({
2737       where: {
2738         archived: 0,
2739         sectionName: {
2740           [Op.like]: `%${keyword}%
2741         }
2742       }
2743     }).then(count => {
2744       res.status(200).json({
2745         numOfPages: Math.ceil(count.count / pageSize),
2746         advisoryData: data
2747       });
2748     });
2749   }
2750 }
2751 );
2752
2753 // @route POST api/registrar/unassignadviser
2754 // @desc Unassign adviser
2755 // @access Private
2756
2757 router.post(
2758   "/unassignadviser",
2759   passport.authenticate("registrar", {
2760     session: false
2761   }),
2762   async(req, res) => {
2763     const {
2764       schoolYearID,
2765       sectionID,
2766       teacherID
2767     } = req.body;
2768     TeacherSection.findOne({
2769       where: {
2770         schoolYearID,
2771         sectionID,
2772         teacherID
2773       }
2774     }).then(adviser => {
2775       if (adviser) {
2776         adviser.destroy().then(() => {
2777           res.status(200).json({
2778             msg: "You have successfully unassigned an adviser!"
2779           });
2780         });
2781       } else {
2782         res.status(404).json({
2783           teacherName: "Unassigning failed"
2784         });
2785       }
2786     });
2787   }
2788 );
2789
2790 // @route POST api/registrar/assignadviser
2791 // @desc Assign adivser
2792 // @access Private
2793
2794 router.post(
2795   "/assignadviser",
2796   passport.authenticate("registrar", {
2797     session: false

```

```

2798     }),
2799     async(req, res) => {
2800       const {
2801         schoolYearID,
2802         sectionID,
2803         teacherID
2804       } = req.body;
2805       TeacherSection.findOne({
2806         where: {
2807           schoolYearID,
2808           teacherID
2809         }
2810       }).then(advisor => {
2811         if (advisor) {
2812           res.status(404).json({
2813             teacherName: "There's already assigned adviser!"
2814           });
2815         } else {
2816           TeacherSection.create({
2817             schoolYearID,
2818             sectionID,
2819             teacherID
2820           })
2821           .then(advisor2 => {
2822             res.status(200).json({
2823               msg: "You have successfully assigned an adviser"
2824             });
2825           })
2826           .catch(err =>
2827             res.status(404).json({
2828               teacherName: "Assigning failed"
2829             })
2830           );
2831         }
2832       });
2833     );
2834   );
2835
2836 // @route POST api/registrar/getsubjects
2837 // @desc Get all subjects
2838 // @access Private
2839
2840 router.post(
2841   "/getsubjects",
2842   passport.authenticate("registrar", {
2843     session: false
2844   }),
2845   async(req, res) => {
2846     let {
2847       page,
2848       pageSize,
2849       keyword
2850     } = req.body;
2851     page = page - 1;
2852     let offset = page * pageSize;
2853     let limit = offset + pageSize;
2854     Subject.findAll({
2855       limit,
2856       offset,
2857       where: {
2858         subjectName: {
2859           [Op.like]: `%"${keyword}"%`
2860         }
2861       }
2862     })
2863     .then(subjects => {
2864       let subjectData = [];
2865       if (subjects.length != 0) {
2866         subjects.slice(0, pageSize).forEach(async(subject, key, arr
2867           ) => {
2868           const keyID = subject.subjectID;
2869           const {
2870             subjectCode,

```

```

2870         subjectName ,
2871         subjectType
2872     } = subject;
2873     subjectData.push({
2874         key: keyID,
2875         subjectCode,
2876         subjectName,
2877         subjectType
2878     });
2879     if (key == arr.length - 1) {
2880         Subject.findAndCountAll({
2881             where: {
2882                 subjectName: {
2883                     [Op.like]: `%${keyword}%
2884                 }
2885             }
2886         })
2887         .then(count => {
2888             subjectData.sort((a, b) =>
2889                 a.subjectName > b.subjectName ? 1 : -1
2890             );
2891             res.status(200).json({
2892                 numOfPages: Math.ceil(count.count / pageSize),
2893                 subjectList: subjectData
2894             });
2895         })
2896         .catch(err => {
2897             res.status(404).json({
2898                 msg: "Not found"
2899             });
2900         });
2901     }
2902 });
2903 } else {
2904     res.status(404).json({
2905         msg: "Not found"
2906     });
2907 }
2908 }
2909 .catch(err => {
2910     res.status(404).json({
2911         msg: "Not found"
2912     });
2913 });
2914 }
2915 );
2916
2917 // @route POST api/registrar/teacherinfo
2918 // @desc Get teacher information
2919 // @access Private
2920
2921 router.post(
2922     "/teacherinfo",
2923     passport.authenticate(directorRegistrar, {
2924         session: false
2925     }),
2926     async(req, res) => {
2927         const {
2928             teacherID
2929         } = req.body;
2930         const accountID = await Teacher.findOne({
2931             where: {
2932                 teacherID
2933             }
2934         }).then(
2935             t => {
2936                 if (t) {
2937                     return t.accountID;
2938                 }
2939             }
2940         );
2941         UserAccount.findOne({

```

```

2942         where: {
2943             accountID
2944         }
2945     }).then(async user => {
2946         if (user) {
2947             const {
2948                 accountID,
2949                 email,
2950                 imageUrl
2951             } = user;
2952             const name = `${utils.capitalize(user.lastName)}, ${utils.
2953                 capitalize(
2954                     user.firstName
2955                 )}${user.middleName.charAt(0).toUpperCase()}.`;
2956             res.status(200).json({
2957                 accountID,
2958                 email,
2959                 imageUrl,
2960                 name
2961             });
2962         } else {
2963             res.status(404).json({
2964                 msg: "User not found"
2965             });
2966         }
2967     });
2968 );
2969
2970 // @route POST api/registrar/studentinfo
2971 // @desc Get student info
2972 // @access Private
2973
2974 router.post(
2975     "/studentinfo",
2976     passport.authenticate(directorRegistrar, {
2977         session: false
2978     }),
2979     async(req, res) => {
2980         const {
2981             studentID
2982         } = req.body;
2983         const accountID = await Student.findOne({
2984             where: {
2985                 studentID
2986             }
2987         }).then(
2988             s => {
2989                 if (s) {
2990                     return s.accountID;
2991                 }
2992             }
2993         );
2994         UserAccount.findOne({
2995             where: {
2996                 accountID
2997             }
2998         }).then(async user => {
2999             if (user) {
3000                 const {
3001                     accountID,
3002                     email,
3003                     imageUrl
3004                 } = user;
3005                 const name = `${utils.capitalize(user.lastName)}, ${utils.
3006                     capitalize(
3007                         user.firstName
3008                     )}${user.middleName.charAt(0).toUpperCase()}.`;
3009                 res.status(200).json({
3010                     accountID,
3011                     email,
3012                     imageUrl,
3013                     name

```

```

3013     });
3014   } else {
3015     res.status(404).json({
3016       msg: "User not found"
3017     });
3018   }
3019 }
3020 );
3021 );
3022 );
3023 // @route POST api/registrar/userinfo
3024 // @desc Get user information
3025 // @access Private
3026
3027 router.post(
3028   "/userinfo",
3029   passport.authenticate("registrar", {
3030     session: false
3031   }),
3032   async(req, res) => {
3033     const {
3034       accountID
3035     } = req.body;
3036     UserAccount.findOne({
3037       where: {
3038         accountID
3039       }
3040     }).then(async user => {
3041       if (user) {
3042         const {
3043           accountID,
3044           email,
3045           imageUrl
3046         } = user;
3047         const name = `${utils.capitalize(user.lastName)}, ${utils.
3048           capitalize(
3049             user.firstName
3050           )}${user.middleName.charAt(0).toUpperCase()}.`;
3051         res.status(200).json({
3052           accountID,
3053           email,
3054           imageUrl,
3055           name
3056         });
3057       } else {
3058         res.status(404).json({
3059           msg: "User not found"
3060         });
3061       }
3062     });
3063   );
3064 );
3065 // @route POST api/registrar/getsubjectsection
3066 // @desc Get subject section by teacherID
3067 // @access Private
3068
3069 router.post(
3070   "/getsubjectsection",
3071   passport.authenticate("registrar", {
3072     session: false
3073   }),
3074   async(req, res) => {
3075     let {
3076       accountID,
3077       page,
3078       pageSize
3079     } = req.body;
3080     let {
3081       schoolYearID,
3082       quarter
3083     } = await utils.getActiveSY();
3084     page = page - 1;

```

```

3085     let offset = page * pageSize;
3086     let limit = offset + pageSize;
3087     let teacherID = await utils.getTeacherID(accountID);
3088     let compareIn =
3089         quarter == "Q1" || quarter == "Q2" ?
3090         ["NON_SHS", "1ST_SEM"] :
3091         ["NON_SHS", "2ND_SEM"];
3092     SubjectSection.findAll({
3093         limit,
3094         offset,
3095         where: {
3096             teacherID,
3097             schoolYearID,
3098             subjectType: {
3099                 [Op.in]: compareIn
3100             }
3101         }
3102     }).then(async subjectsections => {
3103         if (subjectsections.length == 0) {
3104             res.status(404).json({
3105                 msg: "No record"
3106             });
3107         } else {
3108             let subjectsectionData = [];
3109             let i = 0;
3110             for (i; i < subjectsections.slice(0, pageSize).length; i++) {
3111                 const key = subjectsections[i].subsectID;
3112                 const subjectCode = await utils.getSubjectCode(
3113                     subjectsections[i].subjectID
3114                 );
3115                 const subjectName = await utils.getSubjectName(
3116                     subjectsections[i].subjectID
3117                 );
3118                 const gradeLevel = await utils.getSectionGradeLevel(
3119                     subjectsections[i].sectionID
3120                 );
3121                 const sectionName = await utils.getSectionName(
3122                     subjectsections[i].sectionID
3123                 );
3124                 subjectsectionData.push({
3125                     key,
3126                     subjectCode,
3127                     subjectName,
3128                     gradeLevel,
3129                     sectionName
3130                 });
3131             }
3132             SubjectSection.findAndCountAll({
3133                 where: {
3134                     teacherID,
3135                     schoolYearID,
3136                     subjectType: {
3137                         [Op.in]: compareIn
3138                     }
3139                 }
3140             }).then(count => {
3141                 res.status(200).json({
3142                     numOfPages: Math.ceil(count.count / pageSize),
3143                     subjectSectionData: subjectsectionData
3144                 });
3145             });
3146         }
3147     });
3148 );
3149 );
3150
3151 // @route POST api/registrar/getsectionbygradelevel
3152 // @desc Get sections by grade level
3153 // @access Private
3154
3155 router.post(
3156     "/getsectionbygradelevel",

```

```

3157     passport.authenticate(directorRegistrar, {
3158       session: false
3159     }),
3160     async(req, res) => {
3161       const {
3162         gradeLevel
3163       } = req.body;
3164       Section.findAll({
3165         where: {
3166           gradeLevel,
3167           archived: 0
3168         }
3169       }).then(sections => {
3170         if (sections.length != 0) {
3171           let sectionsData = [];
3172           let i = 0;
3173           for (i; i < sections.length; i++) {
3174             const {
3175               sectionID,
3176               sectionName
3177             } = sections[i];
3178             sectionsData.push({
3179               sectionID,
3180               sectionName
3181             });
3182           }
3183           res.status(200).json({
3184             sectionsList: sectionsData
3185           });
3186         } else {
3187           res.status(404).json({
3188             msg: "Not found!"
3189           });
3190         }
3191       });
3192     }
3193   );
3194
3195 // @route POST api/registrar/getsubjectbygradelevel
3196 // @desc Get subjects by grade level
3197 // @access Private
3198
3199 router.post(
3200   "/getsubjectbygradelevel",
3201   passport.authenticate("registrar", {
3202     session: false
3203   }),
3204   async(req, res) => {
3205     let {
3206       gradeLevel
3207     } = req.body;
3208     const nonSHSGradeLevel = [
3209       "G1",
3210       "G2",
3211       "G3",
3212       "G4",
3213       "G5",
3214       "G6",
3215       "G7",
3216       "G8",
3217       "G9",
3218       "G10"
3219     ];
3220     if (["G11", "G12"].includes(gradeLevel)) {
3221       Subject.findAll({
3222         where: {
3223           subjectType: {
3224             [Op.notIn]: nonSHSGradeLevel
3225           },
3226           archived: 0
3227         }
3228       }).then(subjects => {
3229         if (subjects.length == 0) {

```

```

3230         res.status(404).json({
3231             msg: "Not found"
3232         });
3233     } else {
3234         let subjectsData = [];
3235         let i = 0;
3236         for (i; i < subjects.length; i++) {
3237             const {
3238                 subjectID,
3239                 subjectCode,
3240                 subjectName
3241             } = subjects[i];
3242             subjectsData.push({
3243                 subjectID,
3244                 subjectCode,
3245                 subjectName
3246             });
3247         }
3248         res.status(200).json({
3249             subjectsList: subjectsData
3250         });
3251     }
3252 });
3253 } else {
3254     Subject.findAll({
3255         where: {
3256             subjectType: gradeLevel,
3257             archived: 0
3258         }
3259     }).then(subjects => {
3260         if (subjects.length == 0) {
3261             res.status(404).json({
3262                 msg: "Not found"
3263             });
3264         } else {
3265             let subjectsData = [];
3266             let i = 0;
3267             for (i; i < subjects.length; i++) {
3268                 const {
3269                     subjectID,
3270                     subjectCode,
3271                     subjectName
3272                 } = subjects[i];
3273                 subjectsData.push({
3274                     subjectID,
3275                     subjectCode,
3276                     subjectName
3277                 });
3278             }
3279             res.status(200).json({
3280                 subjectsList: subjectsData
3281             });
3282         }
3283     });
3284   }
3285 }
3286 );
3287
3288 // @route POST api/registrar/getsubsectinfo
3289 // @desc Get subject section info
3290 // @access Private
3291
3292 router.post(
3293     "/getsubsectinfo",
3294     passport.authenticate("registrar", {
3295         session: false
3296     }),
3297     async(req, res) => {
3298         const {
3299             subsectID
3300         } = req.body;
3301         SubjectSection.findOne({
3302             where: {

```

```

3303     subsectID
3304   }
3305 }).then(subjectsection => {
3306   if (subjectsection) {
3307     SubjectSectionStudent.findAll({
3308       where: {
3309         subsectID
3310       }
3311     }).then(async subjectsectionstudents => {
3312       if (subjectsectionstudents.length == 0) {
3313         const sectionName = await utils.getSectionName(
3314           subjectsection.sectionID
3315         );
3316         const gradeLevel = await utils.getSectionGradeLevel(
3317           subjectsection.sectionID
3318         );
3319         const subjectName = await utils.getSubjectName(
3320           subjectsection.subjectID
3321         );
3322         res.status(200).json({
3323           sectionName,
3324           gradeLevel,
3325           subjectName,
3326           studentList: []
3327         });
3328     } else {
3329       let i = 0;
3330       let studarr = [];
3331       const sectionName = await utils.getSectionName(
3332         subjectsection.sectionID
3333       );
3334       const gradeLevel = await utils.getSectionGradeLevel(
3335         subjectsection.sectionID
3336       );
3337       const subjectName = await utils.getSubjectName(
3338         subjectsection.subjectID
3339       );
3340       for (i; i < subjectsectionstudents.length; i++) {
3341         let key = subjectsectionstudents[i].subsectstudID;
3342         let name = await utils.getStudentNameByStudsectID(
3343           subjectsectionstudents[i].studsectID
3344         );
3345         let email = await utils.getStudentEmailByStudsectID(
3346           subjectsectionstudents[i].studsectID
3347         );
3348         let sectionName = await utils.
3349           getSectionNameByStudsectID(
3350             subjectsectionstudents[i].studsectID
3351           );
3352         let imageUrl = await utils.
3353           getStudentImageUrlByStudsectID(
3354             subjectsectionstudents[i].studsectID
3355           );
3356         let gradeLevel = await utils.
3357           getSectionGradeLevelByStudsectID(
3358             subjectsectionstudents[i].studsectID
3359           );
3360         studarr.push({
3361           key,
3362           name,
3363           email,
3364           sectionName,
3365           imageUrl,
3366           gradeLevel
3367         });
3368       }
3369     }
3370     res.status(200).json({
3371       sectionName,
3372       gradeLevel,
3373       subjectName,
3374       studentList: studarr
3375     });
3376   }
3377 }
```

```

3373         });
3374     } else {
3375       res.status(404).json({
3376         msg: "Subject section not found!"
3377       });
3378     }
3379   });
3380 }
3381 );
3382
3383 // @route POST api/registrar/addsubsectstud
3384 // @desc Add a stud sect to subject section
3385 // @access Private
3386
3387 router.post(
3388   "/addsubsectstud",
3389   passport.authenticate("registrar", {
3390     session: false
3391   }),
3392   async(req, res) => {
3393     const {
3394       studsectID,
3395       subsectID
3396     } = req.body;
3397     const {
3398       schoolYearID,
3399       quarter
3400     } = await utils.getActiveSY();
3401     SubjectSectionStudent.findOne({
3402       where: {
3403         studsectID,
3404         subsectID
3405       }
3406     }).then(async subjectsectionstudent => {
3407       if (subjectsectionstudent) {
3408         res.status(400).json({
3409           msg: "Student already enrolled in the subject!"
3410         });
3411       } else {
3412         SubjectSectionStudent.create({
3413           studsectID,
3414           subsectID
3415         }).then(async subsectstud => {
3416           SubjectSection.findOne({
3417             where: {
3418               subsectID
3419             }
3420           }).then(async ss => {
3421             if (ss) {
3422               const subjectType = await Subject.findOne({
3423                 where: {
3424                   subjectID: ss.subjectID
3425                 }
3426               }).then(subj => {
3427                 if (subj) {
3428                   return subj.subjectType;
3429                 }
3430               });
3431             let isSHS = ![
3432               "N",
3433               "K1",
3434               "K2",
3435               "G1",
3436               "G2",
3437               "G3",
3438               "G4",
3439               "G5",
3440               "G6",
3441               "G7",
3442               "G8",
3443               "G9",
3444               "G10"
3445             ].includes(subjectType);

```

```

3446     let q = isSHS ? ["Q1", "Q2"] : ["Q1", "Q2", "Q3", "Q4"
3447         ];
3448     let i = 0;
3449     for (i = 0; i < q.length; i++) {
3450         StudentWeightedScore.create({
3451             subsectstudID: subsectstud.subsectstudID,
3452             classRecordID: ss.classRecordID,
3453             faWS: -1,
3454             wwWS: -1,
3455             ptWS: -1,
3456             qeWS: -1,
3457             finalGrade: -1,
3458             quarter: q[i]
3459         });
3460     }
3461     StudentSubjectGrades.create({
3462         subsectstudID: subsectstud.subsectstudID,
3463         classRecordID: ss.classRecordID
3464     });
3465     Grade.findAll({
3466         attributes: [
3467             [
3468                 Sequelize.fn("DISTINCT", Sequelize.col(
3469                     "description")),
3470                     "description"
3471                 ],
3472                 [Sequelize.col("total"), "total"],
3473                 [Sequelize.col("dateGiven"), "dateGiven"],
3474                 [Sequelize.col("componentID"), "componentID"],
3475                 [Sequelize.col("subcomponentID"), "subcomponentID"]
3476                     ],
3477                 [Sequelize.col("classRecordID"), "classRecordID"],
3478                 [Sequelize.col("quarter"), "quarter"]
3479             ],
3480             where: {
3481                 classRecordID: ss.classRecordID
3482             }
3483         }).then(async res => {
3484             if (res.length != 0) {
3485                 for (const [index, value] of res.entries()) {
3486                     await Grade.create({
3487                         description: value.description,
3488                         dateGiven: value.dateGiven,
3489                         score: 0,
3490                         total: value.total,
3491                         componentID: value.componentID,
3492                         subcomponentID: value.subcomponentID,
3493                         date: new Date(),
3494                         classRecordID: ss.classRecordID,
3495                         subsectstudID: subsectstud.subsectstudID,
3496                         quarter: value.quarter,
3497                         attendance: "P",
3498                         showLog: 0,
3499                         isUpdated: 0
3500                     });
3501                     await utils.refreshStudentWeightedScoreBySubsectID(
3502                         ss.subsectID
3503                     );
3504                 }
3505             }
3506             res.status(200).json({
3507                 msg: "Student added to subject successfully!"
3508             });
3509         } else {
3510             res.status(404).json({
3511                 msg: "Subject section does not exist!"
3512             });
3513         });
3514     });
3515 }

```

```

3516         });
3517     }
3518   );
3519
3520   // @route POST api/registrar/deletesubsectstud
3521   // @desc Delete a stud sect to subject section
3522   // @access Private
3523
3524   router.post(
3525     "/deletesubsectstud",
3526     passport.authenticate("registrar", {
3527       session: false
3528     }),
3529     async(req, res) => {
3530       const {
3531         subsectstudID
3532       } = req.body;
3533       SubjectSectionStudent.findOne({
3534         where: {
3535           subsectstudID
3536         }
3537       }).then(data => {
3538         if (data) {
3539           data.destroy().then(() => {
3540             res.status(200).json({
3541               msg: "Student deleted to subject successfully!"
3542             });
3543           });
3544         } else {
3545           res.status(404).json({
3546             msg: "Not found!"
3547           });
3548         }
3549       );
3550     }
3551   );
3552
3553   // @route POST api/registrar/getsubmittedsubsect
3554   // @desc Get subject section where class record status = 'D'
3555   // @access Private
3556
3557   router.post(
3558     "/getsubmittedsubsect",
3559     passport.authenticate("registrar", {
3560       session: false
3561     }),
3562     async(req, res) => {
3563       let {
3564         accountID,
3565         page,
3566         pageSize,
3567         quarter
3568       } = req.body;
3569       page = page - 1;
3570       let offset = page * pageSize;
3571       let limit = offset + pageSize;
3572       const {
3573         schoolYearID
3574       } = await utils.getActiveSY();
3575       const teacherID = await utils.getTeacherID(accountID);
3576       const teacher = await utils.getTeacherName(teacherID);
3577       const deadline = await SubmissionDeadline.findOne({
3578         where: {
3579           teacherID,
3580           isActive: 1
3581         }
3582       }).then(sd => {
3583         if (sd) {
3584           return sd.deadline;
3585         } else {
3586           return "NOT SET";
3587         }
3588       });

```

```

3589     const classRecordIDs = await SubjectSection.findAll({
3590       where: {
3591         teacherID,
3592         schoolYearID,
3593         subjectType: {
3594           [Op.in]: quarter == "Q1" || quarter == "Q2" ?
3595             ["NON_SHS", "1ST_SEM"] : ["NON_SHS", "2ND_SEM"]
3596         }
3597       }
3598     }).then(async ss => {
3599       if (ss) {
3600         let data = [];
3601         for (const [i, v] of ss.entries()) {
3602           const crs = await ClassRecordStatus.findOne({
3603             where: {
3604               classRecordID: v.classRecordID
3605             }
3606           });
3607           if (quarter == "Q1" || quarter == "Q2") {
3608             if (crs[quarter.toLowerCase()] == "D") {
3609               data.push(v.classRecordID);
3610             }
3611           } else {
3612             if (quarter == "Q3") {
3613               if (v.subjectType == "NON_SHS") {
3614                 if (crs[quarter.toLowerCase()] == "D") {
3615                   data.push(v.classRecordID);
3616                 }
3617               } else {
3618                 if (crs["q1"] == "D") {
3619                   data.push(v.classRecordID);
3620                 }
3621               }
3622             } else {
3623               if (v.subjectType == "NON_SHS") {
3624                 if (crs[quarter.toLowerCase()] == "D") {
3625                   data.push(v.classRecordID);
3626                 }
3627               } else {
3628                 if (crs["q2"] == "D") {
3629                   data.push(v.classRecordID);
3630                 }
3631               }
3632             }
3633           }
3634         return data;
3635       }
3636     });
3637   if (classRecordIDs.length == 0) {
3638     res.status(200).json({
3639       classRecordList: [],
3640       numOfPages: 1,
3641       deadline
3642     });
3643   } else {
3644     let condition = {};
3645     condition["classRecordID"] = {
3646       [Op.in]: classRecordIDs
3647     };
3648     await ClassRecordStatus.findAll({
3649       limit,
3650       offset,
3651       where: condition
3652     }).then(async crs => {
3653       if (crs) {
3654         if (crs.length == 0) {
3655           res.status(404).json({
3656             msg: "No class record for deliberation found"
3657           });
3658         } else {
3659

```

```

3660     let data2 = [];
3661     let num0fPages = 1;
3662     crs.slice(0, pageSize).forEach(async(v, k, r) => {
3663       const {
3664         classRecordID
3665       } = v;
3666       const subjectType = await utils.
3667         getSubjectTypeByClassRecordID(
3668           classRecordID
3669         );
3670       let dateSubmitted = v['${quarter.toLowerCase()}'
3671         DateSubmitted'];
3672       if (subjectType == "2ND_SEM") {
3673         if (quarter == "Q3") {
3674           dateSubmitted = v.q1DateSubmitted;
3675         } else if (quarter == "Q4") {
3676           dateSubmitted = v.q2DateSubmitted;
3677         }
3678       }
3679       const {
3680         subjectCode,
3681         subjectName,
3682         section,
3683         subsectID
3684       } = await SubjectSection.findOne({
3685         where: {
3686           classRecordID
3687         }
3688       }).then(async cr => {
3689         if (cr) {
3690           const subjectCode = await utils.getSubjectCode(cr.
3691             subjectID);
3692           const {
3693             subsectID
3694           } = cr;
3695           const subjectName = await utils.getSubjectName(cr.
3696             subjectID);
3697           const section = await utils.getSectionName(cr.
3698             sectionID);
3699           return {
3700             subjectCode,
3701             subsectID,
3702             subjectName,
3703             section
3704           };
3705         }
3706       });
3707       data2.push({
3708         teacher,
3709         classRecordID,
3710         dateSubmitted,
3711         subjectCode,
3712         subjectName,
3713         section,
3714         subsectID
3715       });
3716       if (k == r.length - 1) {
3717         num0fPages = await ClassRecordStatus.findAndCountAll
3718         ({
3719           where: condition
3720         }).then(count => {
3721           return Math.ceil(count.count / pageSize);
3722         });
3723         res.status(200).json({
3724           classRecordList: data2,
3725           num0fPages,
3726           deadline
3727         });
3728       }
3729     });
3730   }
3731 });
3732 });
3733 });
3734 });
3735 });
3736 });

```

```

3727         }
3728     }
3729   );
3730
3731   // @router POST api/registrar/condenseddeliberationgrade
3732   // @desc Get condensed grade of students by sectionID and quarter
3733   // @access Private
3734
3735   router.post(
3736     "/condenseddeliberationgrade",
3737     passport.authenticate("registrar", {
3738       session: false
3739     }),
3740     async(req, res) => {
3741       const {
3742         sectionID,
3743         quarter
3744       } = req.body;
3745       const {
3746         schoolYearID
3747       } = await utils.getActiveSY();
3748       const studsectIDs = await StudentSection.findAll({
3749         where: {
3750           sectionID,
3751           schoolYearID
3752         }
3753       }).then(ss => {
3754         if (ss) {
3755           let data = [];
3756           for (const [i, v] of ss.entries()) {
3757             data.push(v.studsectID);
3758           }
3759           return data;
3760         }
3761       });
3762       SubjectSectionStudent.findAll({
3763         where: {
3764           studsectID: {
3765             [Op.in]: studsectIDs
3766           }
3767         }
3768       }).then(async sss => {
3769         if (sss) {
3770           let data3 = [];
3771           for (const [i, x] of sss.entries()) {
3772             const grades = await SubjectSectionStudent.findAll({
3773               where: {
3774                 studsectID: x.studsectID
3775               }
3776             });
3777             .then(async sss2 => {
3778               if (sss2) {
3779                 let data2 = [];
3780                 for (const [i, v] of sss2.entries()) {
3781                   let name = await utils.getStudentNameBySubsectstudID(
3782                     v.subsectstudID
3783                   );
3784                   let sex = await utils.getStudentSexBySubsectstudID(
3785                     v.subsectstudID
3786                   );
3787                   let imageUrl = await utils.
3788                     getStudentImageUrlBySubsectstudID(
3789                       v.subsectstudID
3790                     );
3791                   let studsectID = v.studsectID;
3792                   let {
3793                     subjectType,
3794                     classRecordID
3795                   } = await SubjectSection.findOne({
3796                     where: {
3797                       subsectID: v.subsectID
3798                     }
3799                   });
3800                   .then(async ss => {

```

```

3798     if (ss) {
3799       return {
3800         subjectType: ss.subjectType,
3801         classRecordID: ss.classRecordID
3802       };
3803     }
3804   });
3805   let status = await ClassRecordStatus.findOne({
3806     where: {
3807       classRecordID
3808     }
3809   }).then(async crs => {
3810     if (crs) {
3811       if (quarter == "Q1") {
3812         if (
3813           subjectType == "NON_SHS" ||
3814           subjectType == "1ST_SEM"
3815         ) {
3816           return crs.q1;
3817         }
3818       } else if (quarter == "Q2") {
3819         if (
3820           subjectType == "NON_SHS" ||
3821           subjectType == "1ST_SEM"
3822         ) {
3823           return crs.q2;
3824         }
3825       } else if (quarter == "Q3") {
3826         if (subjectType == "NON_SHS") {
3827           return crs.q3;
3828         } else if (subjectType == "2ND_SEM") {
3829           return crs.q1;
3830         }
3831       } else {
3832         if (subjectType == "NON_SHS") {
3833           return crs.q4;
3834         } else if (subjectType == "2ND_SEM") {
3835           return crs.q2;
3836         }
3837       }
3838     }
3839   });
3840   if (typeof status !== "undefined") {
3841     let {
3842       score,
3843       subjectName
3844     } = await StudentSubjectGrades.findOne({
3845       where: {
3846         classRecordID,
3847         subsectstudID: v.subsectstudID
3848       }
3849     }).then(async ssg => {
3850       if (ssg) {
3851         let score;
3852         let subjectID = await utils.
3853           getSubjectIDBySubsectstudID(
3854             ssg.subsectstudID
3855           );
3856         let subjectName = await utils.getSubjectCode(
3857           subjectID);
3858         if (status == "F" || status == "D") {
3859           if (quarter == "Q1") {
3860             if (
3861               subjectType == "NON_SHS" ||
3862               subjectType == "1ST_SEM"
3863             ) {
3864               score = ssg.q1FinalGrade;
3865             }
3866           } else if (quarter == "Q2") {
3867             if (
3868               subjectType == "NON_SHS" ||
3869               subjectType == "1ST_SEM"
3870             ) {

```

```

3869                     score = ssg.q2FinalGrade;
3870                 }
3871             } else if (quarter == "Q3") {
3872                 if (subjectType == "NON_SHS") {
3873                     score = ssg.q3FinalGrade;
3874                 } else if (subjectType == "2ND_SEM") {
3875                     score = ssg.q1FinalGrade;
3876                 }
3877             } else {
3878                 if (subjectType == "NON_SHS") {
3879                     score = ssg.q4FinalGrade;
3880                 } else if (subjectType == "2ND_SEM") {
3881                     score = ssg.q2FinalGrade;
3882                 }
3883             }
3884         }
3885     return {
3886         score,
3887         subjectName
3888     };
3889 }
3890 });
3891 if (typeof score !== "undefined" && typeof score
3892 !== "null") {
3893     data2.push({
3894         status,
3895         imageUrl,
3896         sex,
3897         score,
3898         subjectName,
3899         name,
3900         studsectID
3901     });
3902 }
3903 }
3904 return data2;
3905 }
3906 });
3907 data3.push(grades);
3908 }
3909 const resData = utils.formatCondensedDeliberationGrade(data3)
3910 ;
3911 res.status(200).json({
3912     data: [
3913         ...resData.data.filter(a => a.sex == "M"),
3914         ...resData.data.filter(a => a.sex == "F")
3915     ],
3916     columns: resData.columns
3917 });
3918 );
3919 }
3920 );
3921
3922 // @router POST api/registrar/condensedfinalgrade
3923 // @desc Get condensed grade of students by sectionID and quarter
3924 // @access Private
3925
3926 router.post(
3927     "/condensedfinalgrade",
3928     passport.authenticate(directorRegistrar, {
3929         session: false
3930     }),
3931     async(req, res) => {
3932         const {
3933             sectionID,
3934             quarter
3935         } = req.body;
3936         const {
3937             schoolYearID
3938         } = await utils.getActiveSY();
3939         const studsectIDs = await StudentSection.findAll({

```

```

3940         where: {
3941             sectionID,
3942             schoolYearID
3943         }
3944     }).then(ss => {
3945         if (ss) {
3946             let data = [];
3947             for (const [i, v] of ss.entries()) {
3948                 data.push(v.studsectID);
3949             }
3950             return data;
3951         }
3952     });
3953     SubjectSectionStudent.findAll({
3954         where: {
3955             studsectID: {
3956                 [Op.in]: studsectIDs
3957             }
3958         }
3959     }).then(async sss => {
3960         if (sss) {
3961             let data3 = [];
3962             for (const [i, x] of sss.entries()) {
3963                 const grades = await SubjectSectionStudent.findAll({
3964                     where: {
3965                         studsectID: x.studsectID
3966                     }
3967                 }).then(async sss2 => {
3968                     if (sss2) {
3969                         let data2 = [];
3970                         for (const [i, v] of sss2.entries()) {
3971                             let name = await utils.getStudentNameBySubsectstudID(
3972                                 v.subsectstudID
3973                             );
3974                             let studentID = await StudentSection.findOne({
3975                                 where: {
3976                                     studsectID: v.studsectID
3977                                 }
3978                             }).then(ss => ss.studentID);
3979                             let sex = await utils.getStudentSexBySubsectstudID(
3980                                 v.subsectstudID
3981                             );
3982                             let imageUrl = await utils.
3983                                 getStudentImageUrlBySubsectstudID(
3984                                     v.subsectstudID
3985                             );
3986                             let studsectID = v.studsectID;
3987                             let {
3988                                 subjectType,
3989                                 classRecordID
3990                             } = await SubjectSection.findOne({
3991                                 where: {
3992                                     subsectID: v.subsectID
3993                                 }
3994                             }).then(async ss => {
3995                                 if (ss) {
3996                                     return {
3997                                         subjectType: ss.subjectType,
3998                                         classRecordID: ss.classRecordID
3999                                     };
4000                                 }
4001                             let teacher = await SubjectSection.findOne({
4002                                 where: {
4003                                     classRecordID
4004                                 }
4005                             }).then(async ss => await utils.getTeacherName(ss.
4006                                         teacherID))
4007                             let status = await ClassRecordStatus.findOne({
4008                                 where: {
4009                                     classRecordID
4010                                 }
4011                             })
4012                         data2.push({
4013                             studentID: studentID,
4014                             sex: sex,
4015                             imageUrl: imageUrl,
4016                             studsectID: studsectID,
4017                             subjectType: subjectType,
4018                             classRecordID: classRecordID,
4019                             teacher: teacher,
4020                             status: status
4021                         });
4022                     }
4023                 }
4024             }
4025             data3.push(data2);
4026         }
4027     })
4028     return data3;
4029 }

```

```

4010 }).then(async crs => {
4011   if (crs) {
4012     if (quarter == "Q1") {
4013       if (
4014         subjectType == "NON_SHS" ||
4015         subjectType == "1ST_SEM"
4016       ) {
4017         return crs.q1;
4018       }
4019     } else if (quarter == "Q2") {
4020       if (
4021         subjectType == "NON_SHS" ||
4022         subjectType == "1ST_SEM"
4023       ) {
4024         return crs.q2;
4025       }
4026     } else if (quarter == "Q3") {
4027       if (subjectType == "NON_SHS") {
4028         return crs.q3;
4029       } else if (subjectType == "2ND_SEM") {
4030         return crs.q1;
4031       }
4032     } else {
4033       if (subjectType == "NON_SHS") {
4034         return crs.q4;
4035       } else if (subjectType == "2ND_SEM") {
4036         return crs.q2;
4037       }
4038     }
4039   }
4040 });
4041 if (typeof status !== "undefined") {
4042   let {
4043     score,
4044     subjectName
4045   } = await StudentSubjectGrades.findOne({
4046     where: {
4047       classRecordID,
4048       subsectstudID: v.subsectstudID
4049     }
4050   }).then(async ssg => {
4051     if (ssg) {
4052       let score;
4053       let subjectID = await utils.
4054         getSubjectIDBySubsectstudID(
4055           ssg.subsectstudID
4056         );
4057       let subjectName = await utils.getSubjectCode(
4058         subjectID);
4059       if (status == "F" || status == "D") {
4060         if (quarter == "Q1") {
4061           if (
4062             subjectType == "NON_SHS" ||
4063             subjectType == "1ST_SEM"
4064           ) {
4065             score = ssg.q1FinalGrade;
4066           }
4067         } else if (quarter == "Q2") {
4068           if (
4069             subjectType == "NON_SHS" ||
4070             subjectType == "1ST_SEM"
4071           ) {
4072             score = ssg.q2FinalGrade;
4073           }
4074         } else if (quarter == "Q3") {
4075           if (subjectType == "NON_SHS") {
4076             score = ssg.q3FinalGrade;
4077           } else if (subjectType == "2ND_SEM") {
4078             score = ssg.q1FinalGrade;
4079           }
4080         } else {
4081           if (subjectType == "NON_SHS") {
4082             score = ssg.q4FinalGrade;

```

```

4081                         } else if (subjectType == "2ND_SEM") {
4082                             score = ssg.q2FinalGrade;
4083                         }
4084                     }
4085                 }
4086             return {
4087                 score,
4088                 subjectName
4089             };
4090         }
4091     );
4092     if (typeof score !== "undefined" && typeof score
4093         !== "null") {
4094         data2.push({
4095             teacher,
4096             classRecordID,
4097             status,
4098             imageUrl,
4099             studentID,
4100             sex,
4101             score,
4102             subjectName,
4103             name,
4104             studsectID
4105         });
4106     }
4107 }
4108 return data2;
4109 }
4110 });
4111 data3.push(grades);
4112 }
4113 const resData = utils.formatCondensedDeliberationGrade(data3)
4114 ;
4115 res.status(200).json({
4116     data: [
4117         ...resData.data.filter(a => a.sex == "M"),
4118         ...resData.data.filter(a => a.sex == "F")
4119     ],
4120     columns: resData.columns
4121 });
4122 }
4123 );
4124 );
4125
4126 // @route POST api/registrar/getsectionname
4127 // @desc Get section name by studentID
4128 // @access Private
4129
4130 router.post(
4131     "/getsectionname",
4132     passport.authenticate(directorRegistrar, {
4133         session: false
4134     }),
4135     async(req, res) => {
4136         const response = await utils.getSectionName(req.body.sectionID);
4137         res.status(200).json({
4138             sectionName: response
4139         });
4140     }
4141 );
4142
4143 // @route POST api/registrar/getsynname
4144 // @desc Get school year name by schoolYearID
4145 // @access Private
4146
4147 router.post(
4148     "/getsynname",
4149     passport.authenticate(directorRegistrar, {
4150         session: false
4151     }),

```

```

4152     async(req, res) => {
4153       const response = await utils.getSYname(req.body.schoolYearID);
4154       res.status(200).json({
4155         schoolYear: response
4156       });
4157     }
4158   );
4159
4160 // @route POST api/registrar/getnotsubmittedsubsect
4161 // @desc Get not submitted subject section where class record status
4162 // = 'D'
4163 // @access Private
4164
4165 router.post(
4166   "/getnotsubmittedsubsect",
4167   passport.authenticate("registrar", {
4168     session: false
4169   }),
4170   async(req, res) => {
4171     let {
4172       accountID,
4173       page,
4174       pageSize,
4175       quarter
4176     } = req.body;
4177     page = page - 1;
4178     let offset = page * pageSize;
4179     let limit = offset + pageSize;
4180     const {
4181       schoolYearID
4182     } = await utils.getActiveSY();
4183     const teacherID = await utils.getTeacherID(accountID);
4184     const deadline = await SubmissionDeadline.findOne({
4185       where: {
4186         teacherID,
4187         isActive: 1
4188       }
4189     }).then(sd => {
4190       if (sd) {
4191         return sd.deadline;
4192       } else {
4193         return "NOT SET";
4194       }
4195     });
4196     const classRecordIDs = await SubjectSection.findAll({
4197       where: {
4198         teacherID,
4199         schoolYearID,
4200         subjectType: {
4201           [Op.in]: quarter == "Q1" || quarter == "Q2" ?
4202             ["NON_SHS", "1ST_SEM"] : ["NON_SHS", "2ND_SEM"]
4203         }
4204       }
4205     }).then(async ss => {
4206       if (ss) {
4207         let data = [];
4208         for (const [i, v] of ss.entries()) {
4209           const crs = await ClassRecordStatus.findOne({
4210             where: {
4211               classRecordID: v.classRecordID
4212             }
4213           });
4214           if (quarter == "Q1" || quarter == "Q2") {
4215             if (crs[quarter.toLowerCase()] == "E") {
4216               data.push(v.classRecordID);
4217             }
4218           } else {
4219             if (quarter == "Q3") {
4220               if (v.subjectType == "NON_SHS") {
4221                 if (crs[quarter.toLowerCase()] == "E") {
4222                   data.push(v.classRecordID);
4223                 }
4224               }
4225             }
4226           }
4227         }
4228       }
4229     });
4230     res.json(data);
4231   }
4232 );

```

```

4223     } else {
4224         if (crs["q1"] == "E") {
4225             data.push(v.classRecordID);
4226         }
4227     }
4228 } else {
4229     if (v.subjectType == "NON_SHS") {
4230         if (crs[quarter.toLowerCase()] == "E") {
4231             data.push(v.classRecordID);
4232         }
4233     } else {
4234         if (crs["q2"] == "E") {
4235             data.push(v.classRecordID);
4236         }
4237     }
4238 }
4239 }
4240 return data;
4241 }
4242 });
4243 if (classRecordIDs.length == 0) {
4244     res.status(200).json({
4245         classRecordList: [],
4246         numOfPages: 1,
4247         deadline: "NOT SET"
4248     });
4249 } else {
4250     let condition = {};
4251     condition["classRecordID"] = {
4252         [Op.in]: classRecordIDs
4253     };
4254     await ClassRecordStatus.findAll({
4255         limit,
4256         offset,
4257         where: condition
4258     }).then(async crs => {
4259         if (crs) {
4260             if (crs.length == 0) {
4261                 res.status(200).json({
4262                     classRecordList: [],
4263                     numOfPages: 1,
4264                     deadline: "NOT SET"
4265                 });
4266             } else {
4267                 let data2 = [];
4268                 let numOfPages = 1;
4269                 crs.slice(0, pageSize).forEach(async (v, k, r) => {
4270                     const {
4271                         classRecordID
4272                     } = v;
4273                     const subjectType = await utils.
4274                         getSubjectTypeByClassRecordID(
4275                             classRecordID
4276                         );
4277                     let dateSubmitted = v['${quarter.toLowerCase()}'
4278                         DateSubmitted'];
4279                     if (subjectType == "2ND_SEM") {
4280                         if (quarter == "Q3") {
4281                             dateSubmitted = v.q1DateSubmitted;
4282                         } else if (quarter == "Q4") {
4283                             dateSubmitted = v.q2DateSubmitted;
4284                         }
4285                     }
4286                     const {
4287                         subjectCode,
4288                         subjectName,
4289                         section,
4290                         subsectID
4291                     } = await SubjectSection.findOne({
4292                         where: {
4293                             classRecordID

```

```

4293         }
4294     }).then(async cr => {
4295         if (cr) {
4296             const subjectCode = await utils.getSubjectCode(cr.
4297                 subjectID);
4298             const {
4299                 subsectID
4300             } = cr;
4301             const subjectName = await utils.getSubjectName(cr.
4302                 subjectID);
4303             const section = await utils.getSectionName(cr.
4304                 sectionID);
4305             return {
4306                 subjectCode,
4307                 subsectID,
4308                 subjectName,
4309                 section
4310             };
4311         }
4312     });
4313     data2.push({
4314         classRecordID,
4315         dateSubmitted,
4316         subjectCode,
4317         subjectName,
4318         section,
4319         subsectID
4320     });
4321     if (k == r.length - 1) {
4322         numOfPages = await ClassRecordStatus.findAndCountAll
4323         ({
4324             where: condition
4325         }).then(count => {
4326             return Math.ceil(count.count / pageSize);
4327         });
4328         res.status(200).json({
4329             classRecordList: data2,
4330             numOfPages,
4331             deadline
4332         });
4333     }
4334 }
4335 );
4336 );
4337
4338 // @route POST api/registrar/getclassrecinfo
4339 // @desc Get class record info by classRecordID and quarter
4340 // @access Private
4341
4342 router.post(
4343     "/getclassrecinfo",
4344     passport.authenticate(directorRegistrar, {
4345         session: false
4346     }),
4347     async(req, res) => {
4348         let {
4349             classRecordID,
4350             quarter
4351         } = req.body;
4352         let {
4353             page,
4354             pageSize
4355         } = req.body;
4356         page = page - 1;
4357         let offset = page * pageSize;
4358         let limit = offset + pageSize;
4359         const subjectType = await utils.getSubjectTypeByClassRecordID(
4360             classRecordID
4361         );

```

```

4362
4363     const {
4364       IDs,
4365       data,
4366       numOfPages
4367     } = await StudentSubjectGrades.findAll({
4368       where: {
4369         classRecordID
4370       }
4371     }).then(async sg => {
4372       if (sg) {
4373         if (sg.length == 0) {
4374           res.status(200).json({
4375             numOfPages: 1,
4376             studentList: []
4377           });
4378         } else {
4379           let data = [];
4380           let IDs = [];
4381           for (const [index, value] of sg.entries()) {
4382             const {
4383               subsectstudID
4384             } = value;
4385             let add = {};
4386             let grade = value['${quarter.toLowerCase()}FinalGrade'];
4387             var i = 0;
4388             for (i = 1; i < parseInt(quarter.charAt(1)); i = i + 1) {
4389               add[`q${i}Grade`] = value[`q${i}FinalGrade`]
4390             }
4391             // if (quarter != "Q1") {
4392             //   add[`q${parseInt(quarter.charAt(1))-1}Grade`] =
4393             //     value[`q${parseInt(quarter.charAt(1))-1}FinalGrade`]
4394             // }
4395             if (subjectType == "2ND_SEM") {
4396               if (quarter == "Q3") {
4397                 grade = value["q1FinalGrade"];
4398               } else if (quarter == "Q4") {
4399                 grade = value["q2FinalGrade"];
4400                 add['q1Grade'] = value['q1FinalGrade']
4401               } else {
4402                 grade = -1;
4403               }
4404             } else if (subjectType == "1ST_SEM") {
4405               if (quarter == "Q3" || quarter == "Q4") {
4406                 grade = -1;
4407               }
4408             const accountID = await utils.getAccountIDBySubsectstudID
4409               (
4410                 subsectstudID
4411               );
4412             data.push({
4413               ...add,
4414               subsectstudID,
4415               grade,
4416               accountID
4417             });
4418             IDs.push(accountID);
4419           }
4420           return {
4421             IDs,
4422             data,
4423             numOfPages: Math.ceil(data.length / pageSize)
4424           };
4425         }
4426       });
4427
4428       UserAccount.findAll({
4429         where: {
4430           accountID: [
4431             [Op.in]: IDs

```

```

4432         }
4433     },
4434     order: [
4435       ["lastName", "ASC"],
4436       ["sex", "DESC"]
4437     ]
4438   }).then(async ua => {
4439     if (ua) {
4440       let newData = [];
4441       for (const [index, value] of ua.entries()) {
4442         const name = `${utils.capitalize(value.lastName)}, ${utils.
4443           capitalize(
4444             value.firstName
4445           )}${value.middleName.charAt(0).toUpperCase()}.`;
4446         const subsectID = data.find(a => a.accountID == value.
4447           accountID)
4448           .subsectID;
4449         const grade = data.find(a => a.accountID == value.accountID
4450           ).grade;
4451         const q1Grade = data.find(a => a.accountID == value.
4452           accountID).q1Grade;
4453         const q2Grade = data.find(a => a.accountID == value.
4454           accountID).q2Grade;
4455         const q3Grade = data.find(a => a.accountID == value.
4456           accountID).q3Grade;
4457         const q4Grade = data.find(a => a.accountID == value.
4458           accountID).q4Grade;
4459         const {
4460           imageUrl,
4461           sex
4462         } = value;
4463         newData.push({
4464           q1Grade,
4465           q2Grade,
4466           q3Grade,
4467           q4Grade,
4468           sex,
4469           name,
4470           subsectID,
4471           grade,
4472           imageUrl
4473         });
4474       }
4475     }
4476   });
4477 }
4478 );
4479
4480 // @route POST api/registrar/postclassrecord
4481 // @desc Post class record by classRecordID and quarter
4482 // @access Private
4483
4484 router.post(
4485   "/postclassrecord",
4486   passport.authenticate("registrar", {
4487     session: false
4488   }),
4489   async(req, res) => {
4490     let {
4491       classRecordID,
4492       quarter
4493     } = req.body;
4494     const {
4495       subjectID,
4496       sectionID,
4497       subjectType,

```

```

4498     subsectID
4499   } = await SubjectSection.findOne({
4500     where: {
4501       classRecordID
4502     }
4503   }).then(ss => {
4504     if (ss) {
4505       return {
4506         subjectID: ss.subjectID,
4507         sectionID: ss.sectionID,
4508         subjectType: ss.subjectType,
4509         subsectID: ss.subsectID
4510       };
4511     }
4512   });
4513   if (subjectType == "2ND_SEM") {
4514     if (quarter == "Q3") {
4515       quarter = "Q1";
4516     } else {
4517       quarter = "Q2";
4518     }
4519   }
4520
4521   ClassRecordStatus.findOne({
4522     where: {
4523       classRecordID
4524     }
4525   }).then(async crs => {
4526     if (crs) {
4527       let quarterObj = {};
4528       quarterObj[quarter.toLowerCase()] = "F";
4529       let invalid =
4530         crs[quarter.toLowerCase()] == "L" ||
4531         crs[quarter.toLowerCase()] == "F" ||
4532         crs[quarter.toLowerCase()] == "E";
4533       if (invalid) {
4534         res.status(400).json({
4535           msg: "You are not authorized to edit this class record."
4536         });
4537       } else {
4538         crs
4539           .update(quarterObj, {
4540             position: "Registrar",
4541             classRecordID,
4542             type: "CHANGE_STATUS",
4543             accountID: req.user.accountID,
4544             subjectID,
4545             sectionID,
4546             oldVal: "D",
4547             newVal: "F",
4548             quarter
4549           })
4550           .then(async() => {
4551             await utils.refreshStudentGradesBySubsectID(subsectID);
4552             res.status(200).json({
4553               msg: "Class record is now posted!"
4554             });
4555           });
4556         }
4557       }
4558     });
4559   }
4560 );
4561
4562 // @route POST api/registrar/revertclassrecord
4563 // @desc Revert class record by classRecordID and quarter
4564 // @access Private
4565
4566 router.post(
4567   "/revertclassrecord",
4568   passport.authenticate("registrar", {
4569     session: false
4570   }),

```

```

4571     async(req, res) => {
4572       let {
4573         classRecordID,
4574         quarter
4575       } = req.body;
4576       const {
4577         subjectID,
4578         sectionID,
4579         subjectType,
4580         subsectID
4581       } = await SubjectSection.findOne({
4582         where: {
4583           classRecordID
4584         }
4585       }).then(ss => {
4586         if (ss) {
4587           return {
4588             subjectID: ss.subjectID,
4589             sectionID: ss.sectionID,
4590             subjectType: ss.subjectType,
4591             subsectID: ss.subsectID
4592           };
4593         }
4594       });
4595       if (subjectType == "2ND_SEM") {
4596         if (quarter == "Q3") {
4597           quarter = "Q1";
4598         } else {
4599           quarter = "Q2";
4600         }
4601       }
4602
4603       ClassRecordStatus.findOne({
4604         where: {
4605           classRecordID
4606         }
4607       }).then(async crs => {
4608         if (crs) {
4609           let quarterObj = {};
4610           quarterObj[quarter.toLowerCase()] = "D";
4611           let invalid =
4612             crs[quarter.toLowerCase()] == "L" ||
4613             crs[quarter.toLowerCase()] == "D" ||
4614             crs[quarter.toLowerCase()] == "E";
4615           if (invalid) {
4616             res.status(400).json({
4617               msg: "You are not authorized to edit this class record."
4618             });
4619           } else {
4620             crs
4621               .update(quarterObj, {
4622                 position: "Registrar",
4623                 classRecordID,
4624                 type: "CHANGE_STATUS",
4625                 accountID: req.user.accountID,
4626                 subjectID,
4627                 sectionID,
4628                 oldVal: "F",
4629                 newVal: "D",
4630                 quarter
4631               })
4632               .then(async() => {
4633                 await utils.refreshStudentGradesBySubsectID(subsectID);
4634                 res.status(200).json({
4635                   msg: "Class record is now ready for deliberation!"
4636                 });
4637               });
4638           }
4639         }
4640       });
4641     );
4642   );
4643 
```

```

4644 // @route POST api/registrar/getsubjecttype
4645 // @desc Get subject type by classRecordID
4646 // @access Private
4647
4648 router.post(
4649   "/getsubjecttype",
4650   passport.authenticate(directorRegistrar, {
4651     session: false
4652   }),
4653   async(req, res) => {
4654     const {
4655       classRecordID
4656     } = req.body;
4657     SubjectSection.findOne({
4658       where: {
4659         classRecordID
4660       }
4661     }).then(ss => {
4662       if (ss) {
4663         res.status(200).json({
4664           subjectType: ss.subjectType
4665         });
4666       } else {
4667         res.status(404).json({
4668           msg: "Subject section not found!"
4669         });
4670       }
4671     });
4672   }
4673 );
4674
4675 // @route POST api/registrar/getcomponents
4676 // @desc Get components by classRecordID and quarter
4677 // @access Private
4678
4679 router.post(
4680   "/getcomponents",
4681   passport.authenticate(directorRegistrar, {
4682     session: false
4683   }),
4684   async(req, res) => {
4685     const {
4686       classRecordID,
4687       quarter
4688     } = req.body;
4689     SubjectSection.findOne({
4690       where: {
4691         classRecordID
4692       }
4693     }).then(subjectsection => {
4694       if (subjectsection) {
4695         Component.findOne({
4696           where: {
4697             subjectID: subjectsection.subjectID,
4698             component: "FA"
4699           }
4700         }).then(comp1 => {
4701           Component.findOne({
4702             where: {
4703               subjectID: subjectsection.subjectID,
4704               component: "WW"
4705             }
4706           }).then(comp2 => {
4707             Component.findOne({
4708               where: {
4709                 subjectID: subjectsection.subjectID,
4710                 component: "PT"
4711               }
4712             }).then(comp3 => {
4713               Component.findOne({
4714                 where: {
4715                   subjectID: subjectsection.subjectID,

```

```

4716         component: "QE"
4717     }
4718 }).then(async comp4 => {
4719     let faSubcompData = [];
4720     let wwSubcompData = [];
4721     let ptSubcompData = [];
4722     let qeSubcompData = [];
4723     await Subcomponent.findAll({
4724         where: {
4725             classRecordID: subjectsection.classRecordID,
4726             componentID: comp1.componentID,
4727             quarter
4728         }
4729     }).then(async subcomp1 => {
4730         if (subcomp1.length != 0) {
4731             let i = 0;
4732             for (i = 0; i < subcomp1.length; i++) {
4733                 let {
4734                     name,
4735                     compWeight,
4736                     subcompID
4737                 } = subcomp1[i];
4738                 faSubcompData.push({
4739                     name,
4740                     compWeight,
4741                     subcompID
4742                 });
4743             }
4744         }
4745     });
4746     await Subcomponent.findAll({
4747         where: {
4748             classRecordID: subjectsection.classRecordID,
4749             componentID: comp2.componentID,
4750             quarter
4751         }
4752     }).then(async subcomp2 => {
4753         if (subcomp2.length != 0) {
4754             let i = 0;
4755             for (i = 0; i < subcomp2.length; i++) {
4756                 let {
4757                     name,
4758                     compWeight,
4759                     subcompID
4760                 } = subcomp2[i];
4761                 wwSubcompData.push({
4762                     name,
4763                     compWeight,
4764                     subcompID
4765                 });
4766             }
4767         }
4768     });
4769     await Subcomponent.findAll({
4770         where: {
4771             classRecordID: subjectsection.classRecordID,
4772             componentID: comp3.componentID,
4773             quarter
4774         }
4775     }).then(async subcomp3 => {
4776         if (subcomp3.length != 0) {
4777             let i = 0;
4778             for (i = 0; i < subcomp3.length; i++) {
4779                 let {
4780                     name,
4781                     compWeight,
4782                     subcompID
4783                 } = subcomp3[i];
4784                 ptSubcompData.push({
4785                     name,
4786                     compWeight,
4787                     subcompID
4788                 });
4789             }

```

```

4790         }
4791     });
4792     await Subcomponent.findAll({
4793       where: {
4794         classRecordID: subjectsection.classRecordID,
4795         componentID: comp4.componentID,
4796         quarter
4797       }
4798     }).then(async subcomp4 => {
4799       if (subcomp4.length != 0) {
4800         let i = 0;
4801         for (i = 0; i < subcomp4.length; i++) {
4802           let {
4803             name,
4804             compWeight,
4805             subcompID
4806           } = subcomp4[i];
4807           qeSubcompData.push({
4808             name,
4809             compWeight,
4810             subcompID
4811           });
4812         }
4813       }
4814     });
4815     const sectionName = await utils.getSectionName(
4816       subjectsection.sectionID
4817     );
4818     const subjectName = await utils.getSubjectName(
4819       subjectsection.subjectID
4820     );
4821     const schoolYear = await utils.getSYname(
4822       subjectsection.schoolYearID
4823     );
4824     const subjectCode = await utils.getSubjectCode(
4825       subjectsection.subjectID
4826     );
4827     res.status(200).json({
4828       subjectCode,
4829       sectionName,
4830       subjectName,
4831       schoolYear,
4832       classRecordID: subjectsection.classRecordID,
4833       FA: {
4834         componentID: comp1.componentID,
4835         weight: comp1.compWeight,
4836         subcomponents: faSubcompData
4837       },
4838       WW: {
4839         componentID: comp2.componentID,
4840         weight: comp2.compWeight,
4841         subcomponents: wwSubcompData
4842       },
4843       PT: {
4844         componentID: comp3.componentID,
4845         weight: comp3.compWeight,
4846         subcomponents: ptSubcompData
4847       },
4848       QE: {
4849         componentID: comp4.componentID,
4850         weight: comp4.compWeight,
4851         subcomponents: qeSubcompData
4852       }
4853     });
4854   });
4855 });
4856 });
4857 });
4858 } else {
4859   res.status(404).json({
4860     msg: "Subject Section not found!"
4861   });
4862 }
4863 );

```

```

4864      }
4865  );
4866
4867 // @route POST api/teacher/editsubcomp
4868 // @desc Edit subcomponent name, weight by subcomponentID
4869 // @access Private
4870
4871 router.post(
4872   "/editsubcomp",
4873   passport.authenticate("registrar", {
4874     session: false
4875   }),
4876   async(req, res) => {
4877     const reg = /^-?[0-9]*(\.[0-9]*)?$/;
4878     const {
4879       payload,
4880       classRecordID,
4881       quarter
4882     } = req.body;
4883     ClassRecordStatus.findOne({
4884       where: {
4885         classRecordID
4886       }
4887     }).then(async crs => {
4888       if (crs) {
4889         let invalid = false;
4890         if (quarter == "Q1") {
4891           invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 == "F";
4892         } else if (quarter == "Q2") {
4893           invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 == "F";
4894         } else if (quarter == "Q3") {
4895           invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 == "F";
4896         } else {
4897           invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 == "F";
4898         }
4899         if (invalid) {
4900           res.status(400).json({
4901             msg: "You are not authorized to edit this class record."
4902           });
4903         } else {
4904           let i = 0;
4905           let sum = 0;
4906           for (i = 0; i < payload.length; i++) {
4907             let {
4908               errors,
4909               isValid
4910             } = validateSubcomponent({
4911               name: payload[i].name
4912             });
4913             if (!isValid) {
4914               return res.status(400).json({
4915                 msg: errors.msg
4916               });
4917             } else {
4918               let valid =
4919                 (!isNaN(payload[i].compWeight) &&
4920                  reg.test(payload[i].compWeight)) ||
4921                  payload[i].compWeight === "" ||
4922                  payload[i].compWeight === "-";
4923               if (
4924                 payload[i].compWeight > 100 ||
4925                 payload[i].compWeight < 0 ||
4926                 !valid
4927               ) {
4928                 return res.status(400).json({
4929                   msg: "Invalid component weight input"
4930                 });
4931               } else {
4932                 sum = sum + parseFloat(payload[i].compWeight);
4933               }
4934             }
4935           }

```

```

4936
4937     if (sum != 100) {
4938         return res.status(400).json({
4939             msg: "Sum of component weights must be 100!"
4940         });
4941     }
4942
4943     payload.forEach(async(val, arr, index) => {
4944         await Subcomponent.findOne({
4945             where: {
4946                 subcompID: val.subcompID
4947             }
4948         }).then(subcomp => {
4949             if (subcomp) {
4950                 SubjectSection.findOne({
4951                     where: {
4952                         classRecordID: subcomp.classRecordID
4953                     }
4954                 }).then(async subjectsection => {
4955                     if (subjectsection) {
4956                         let {
4957                             name,
4958                             compWeight
4959                         } = val;
4960                         const componentName = await utils.
4961                             getComponentName(
4962                                 subcomp.componentID
4963                             );
4964
4965                         subcomp
4966                             .update({
4967                                 name,
4968                                 compWeight
4969                             }, {
4970                                 showLog: true,
4971                                 position: "Registrar",
4972                                 type: "SUBCOMP_UPDATE",
4973                                 accountID: req.user.accountID,
4974                                 sectionID: subjectsection.sectionID,
4975                                 subjectID: subjectsection.subjectID,
4976                                 quarter,
4977                                 subcompName: subcomp.name,
4978                                 componentName,
4979                                 classRecordID,
4980                                 oldVal: subcomp.compWeight,
4981                                 newVal: compWeight,
4982                                 oldName: subcomp.name,
4983                                 newName: name
4984                             })
4985                         .then(async() => {
4986                             await utils.
4987                                 refreshStudentWeightedScoreBySubsectID(
4988                                     subjectsection.subsectID
4989                                 );
4990                         } else {
4991                             res.status(400).json({
4992                                 msg: "Subject Section not found!"
4993                             });
4994                         });
4995                     } else {
4996                         res.status(400).json({
4997                             msg: "Subcomponent not found!"
4998                         });
4999                     }
5000                 });
5001             });
5002             res.status(200).json({
5003                 msg: "Subcomponent updated successfully!"
5004             });
5005         }
5006     } else {
5007         res.status(404).json({

```

```

5008             msg: "Class record status not found!"
5009         });
5010     });
5011   });
5012 }
5013 );
5014
5015 // @route POST api/registrar/addnewsubcomp
5016 // @desc Add new subcomponent by classRecordID and componentID
5017 // @access Private
5018
5019 router.post(
5020   "/addnewsubcomp",
5021   passport.authenticate("registrar", {
5022     session: false
5023   }),
5024   async(req, res) => {
5025     const {
5026       classRecordID,
5027       componentID,
5028       name,
5029       quarter
5030     } = req.body;
5031     const {
5032       errors,
5033       isValid
5034     } = validateSubcomponent({
5035       name
5036     );
5037
5038     if (!isValid) {
5039       return res.status(400).json({
5040         msg: errors.msg
5041       });
5042     }
5043
5044     SubjectSection.findOne({
5045       where: {
5046         classRecordID
5047       }
5048     }).then(subjectsection => {
5049       if (subjectsection) {
5050         ClassRecordStatus.findOne({
5051           where: {
5052             classRecordID
5053           }
5054         }).then(async crs => {
5055           if (crs) {
5056             if (quarter == "Q1") {
5057               if (crs.q1 == "L" || crs.q1 == "E" || crs.q1 == "F") {
5058                 res.status(400).json({
5059                   msg: "You are not authorized to edit this class
5060                     record."
5061                 });
5062               } else {
5063                 const componentName = await utils.getComponentName(
5064                   componentID);
5065                 Subcomponent.create({
5066                   name,
5067                   componentID,
5068                   classRecordID,
5069                   compWeight: 0,
5070                   quarter
5071                 }, {
5072                   showLog: true,
5073                   position: "Registrar",
5074                   type: "SUBCOMP_ADD",
5075                   accountID: req.user.accountID,
5076                   sectionID: subjectsection.sectionID,
5077                   subjectID: subjectsection.subjectID,
5078                   quarter,
5079                   subcompName: name,
5080                   componentName,
5081                   classRecordID

```

```

5080     }).then(subcomponent => {
5081         res.status(200).json({
5082             msg: "Successfully added a new subcomponent!"
5083         });
5084     });
5085 }
5086 } else if (quarter == "Q2") {
5087     if (crs.q2 == "L" || crs.q2 == "E" || crs.q2 == "F") {
5088         res.status(400).json({
5089             msg: "You are not authorized to edit this class
5090             record."
5091         });
5092     } else {
5093         const componentName = await utils.getComponentName(
5094             componentID);
5095         Subcomponent.create({
5096             name,
5097             componentID,
5098             classRecordID,
5099             compWeight: 0,
5100             quarter
5101         }, {
5102             showLog: true,
5103             position: "Registrar",
5104             type: "SUBCOMP_ADD",
5105             accountID: req.user.accountID,
5106             sectionID: subjectsection.sectionID,
5107             subjectID: subjectsection.subjectID,
5108             quarter,
5109             subcompName: name,
5110             componentName,
5111             classRecordID
5112         }).then(subcomponent => {
5113             res.status(200).json({
5114                 msg: "Successfully added a new subcomponent!"
5115             });
5116         });
5117     } else if (quarter == "Q3") {
5118         if (crs.q3 == "L" || crs.q3 == "E" || crs.q3 == "F") {
5119             res.status(400).json({
5120                 msg: "You are not authorized to edit this class
5121                 record."
5122             });
5123     } else {
5124         const componentName = await utils.getComponentName(
5125             componentID);
5126         Subcomponent.create({
5127             name,
5128             componentID,
5129             classRecordID,
5130             compWeight: 0,
5131             quarter
5132         }, {
5133             showLog: true,
5134             position: "Registrar",
5135             type: "SUBCOMP_ADD",
5136             accountID: req.user.accountID,
5137             sectionID: subjectsection.sectionID,
5138             subjectID: subjectsection.subjectID,
5139             quarter,
5140             subcompName: name,
5141             componentName,
5142             classRecordID
5143         }).then(subcomponent => {
5144             res.status(200).json({
5145                 msg: "Successfully added a new subcomponent!"
5146             });
5147         });
5148     } else if (crs.q4 == "L" || crs.q4 == "E" || crs.q4 == "F") {
5149         res.status(400).json({
5150             msg: "You are not authorized to edit this class
5151             record."
5152         });
5153     };
5154 }

```

```

5150           });
5151     } else {
5152       const componentName = await utils.getComponentName(
5153         componentID);
5154       Subcomponent.create({
5155         name,
5156         componentID,
5157         classRecordID,
5158         compWeight: 0,
5159         quarter
5160       }, {
5161         showLog: true,
5162         position: "Registrar",
5163         type: "SUBCOMP_ADD",
5164         accountID: req.user.accountID,
5165         sectionID: subjectsection.sectionID,
5166         subjectID: subjectsection.subjectID,
5167         quarter,
5168         subcompName: name,
5169         componentName,
5170         classRecordID
5171       }).then(subcomponent => {
5172         res.status(200).json({
5173           msg: "Successfully added a new subcomponent!"
5174         });
5175       });
5176     }
5177   } else {
5178     res.status(404).json({
5179       msg: "Class record status does not exist."
5180     });
5181   }
5182 });
5183 } else {
5184   res.status(404).json({
5185     msg: "Subject section not found!"
5186   });
5187 }
5188 );
5189 );
5190 );
5191
5192 // @route POST api/registrar/deletesubcomp
5193 // @desc Delete subcomponent by subcomponentID
5194 // @access Private
5195
5196 router.post(
5197   "/deletesubcomp",
5198   passport.authenticate("registrar", {
5199     session: false
5200   }),
5201   async(req, res) => {
5202     const {
5203       subcompID,
5204       quarter,
5205       classRecordID
5206     } = req.body;
5207     ClassRecordStatus.findOne({
5208       where: {
5209         classRecordID
5210       }
5211     }).then(async crs => {
5212       if (crs) {
5213         let invalid = false;
5214         if (quarter == "Q1") {
5215           invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 == "F";
5216         } else if (quarter == "Q2") {
5217           invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 == "F";
5218         } else if (quarter == "Q3") {
5219           invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 == "F";
5220         } else {
5221           invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 == "F";

```

```

5222
5223     }
5224     if (invalid) {
5225         res.status(400).json({
5226             msg: "You are not authorized to edit this class record."
5227         });
5228     } else {
5229         Subcomponent.findOne({
5230             where: {
5231                 subcompID
5232             }
5233         }).then(subcomponent => {
5234             if (subcomponent) {
5235                 Subcomponent.findAll({
5236                     where: {
5237                         classRecordID: subcomponent.classRecordID,
5238                         componentID: subcomponent.componentID
5239                     }
5240                 }).then(c => {
5241                     if (c.length == 1) {
5242                         res.status(400).json({
5243                             msg: "Subcomponent can't be deleted. There must
5244                             be at least one subcomponent."
5245                         });
5246                     } else {
5247                         let compweight = subcomponent.compWeight;
5248                         if (compweight != 0) {
5249                             res.status(400).json({
5250                                 msg: "Subcomponent can't be deleted. Set
5251                                 component weight to 0 first."
5252                             });
5253                         } else {
5254                             Grade.findOne({
5255                                 where: {
5256                                     subcomponentID: subcomponent.subcompID
5257                                 }
5258                             }).then(async g => {
5259                                 if (g) {
5260                                     res.status(400).json({
5261                                         msg: "Operation could not be completed.
5262                                         Delete all grades under this
5263                                         subcomponent first."
5264                                     });
5265                                 } else {
5266                                     const {
5267                                         sectionID,
5268                                         subjectID
5269                                     } = await SubjectSection.findOne({
5270                                         where: {
5271                                             classRecordID
5272                                         }
5273                                     }).then(ss => {
5274                                         if (ss) {
5275                                             return {
5276                                                 sectionID: ss.sectionID,
5277                                                 subjectID: ss.subjectID,
5278                                                 subjectType: ss.subjectType
5279                                             };
5280                                         }
5281                                     });
5282                                     const {
5283                                         accountID
5284                                     } = req.user;
5285                                     const subcompName = subcomponent.name;
5286                                     const componentName = await utils.
5287                                         getComponentName(
5288                                         subcomponent.componentID
5289                                     );
5290                                     subcomponent
5291                                         .destroy({
5292                                             showLog: true,
5293                                             position: "Registrar",
5294                                             type: "SUBCOMP_DELETE",
5295                                             classRecordID,
5296                                             componentID
5297                                         });
5298                                     res.json({
5299                                         msg: "Subcomponent deleted successfully."
5300                                     });
5301                                 }
5302                             });
5303                         }
5304                     }
5305                 });
5306             }
5307         });
5308     }
5309 
```

```

5290                               accountID ,
5291                               sectionID ,
5292                               subjectID ,
5293                               quarter ,
5294                               subcompName ,
5295                               componentName
5296                         })
5297                         .then(() => {
5298                           res.status(200).json({
5299                             msg: "Subcomponent deleted successfully
5300                               !
5301                               });
5302                               }
5303                               });
5304                               }
5305                               }
5306                               });
5307                           } else {
5308                             res.status(404).json({
5309                               msg: "Subcomponent not found!"
5310                             });
5311                           }
5312                           });
5313                         }
5314                       } else {
5315                         res.status(404).json({
5316                           msg: "Class record status does not exist."
5317                         });
5318                       }
5319                     );
5320                   );
5321                 );
5322               );
5323 // @route POST api/registrar/getactivitylog
5324 // @desc Get activity log by classRecordID
5325 // @access Private
5326
5327 router.post(
5328   "/getactivitylog",
5329   passport.authenticate("registrar", {
5330     session: false
5331   }),
5332   async(req, res) => {
5333     const {
5334       classRecordID,
5335       quarter
5336     } = req.body;
5337     const responseData = await ActivityLog.findAll({
5338       where: {
5339         classRecordID,
5340         quarter
5341       },
5342       order: [
5343         ["timestamp", "DESC"]
5344       ]
5345     }).then(async activitylogs => {
5346       if (activitylogs) {
5347         let activitylogData = [];
5348         for (const [index, al] of activitylogs.entries()) {
5349           const {
5350             position,
5351             name,
5352             section,
5353             subject,
5354             type,
5355             logID,
5356             timestamp
5357           } = al;
5358           const logDetails = await LogDetails.findAll({
5359             where: {
5360               logID
5361             }

```

```

5362     }).then(async lds => {
5363         if (lds) {
5364             let logDetailsData = [];
5365             for (const [index2, ld] of lds.entries()) {
5366                 const {
5367                     student,
5368                     component,
5369                     subcomponent,
5370                     description,
5371                     oldValue,
5372                     newValue
5373                 } = ld;
5374                 logDetailsData.push({
5375                     student,
5376                     component,
5377                     subcomponent,
5378                     description,
5379                     oldValue,
5380                     newValue
5381                 });
5382             }
5383             return logDetailsData;
5384         }
5385     });
5386 });
5387
5388     activitylogData.push({
5389         position,
5390         name,
5391         section,
5392         subject,
5393         type,
5394         timestamp,
5395         logDetails
5396     });
5397 }
5398
5399     return activitylogData;
5400 }
5401 });
5402
5403     res.status(200).json({
5404         activityLog: responseData
5405     });
5406 }
5407 );
5408
5409 // @router POST api/registrar/compinfo
5410 // @desc Get component information by classRecordID and componentID
5411
5412 router.post(
5413     "/compinfo",
5414     passport.authenticate(directorRegistrar, {
5415         session: false
5416     }),
5417     async(req, res) => {
5418         const {
5419             classRecordID,
5420             componentID,
5421             quarter
5422         } = req.body;
5423         const {
5424             accountID
5425         } = req.user;
5426         const teacherID = await utils.getTeacherID(accountID);
5427         SubjectSection.findOne({
5428             where: {
5429                 classRecordID
5430             }
5431         }).then(async subjectsection => {
5432             if (subjectsection) {
5433                 Component.findOne({
5434                     where: {
5435                         componentID

```

```

5436
5437     }
5438   }).then(async component => {
5439     if (component) {
5440       let component = await utils.getComponentName(componentID)
5441       ;
5442       let grades = await Subcomponent.findAll({
5443         where: {
5444           componentID,
5445           classRecordID: subjectsection.classRecordID,
5446           quarter
5447         }
5448       }).then(async subcomp => {
5449         if (subcomp.length == 0) {
5450           res.status(404).json({
5451             msg: "Subcomponents not found!"
5452           });
5453         } else {
5454           let i = 0;
5455           let data1 = [];
5456           for (i = 0; i < subcomp.length; i++) {
5457             const {
5458               subcompID
5459             } = subcomp[i];
5460             let name = await utils.getSubcomponentName(
5461               subcompID);
5462             name = name + ' (' + subcomp[i].compWeight + '%)';
5463             const data2 = await SubjectSectionStudent.findAll({
5464               where: {
5465                 subsectID: subjectsection.subsectID
5466               }
5467             }).then(async subsectstud => {
5468               if (subsectstud.length == 0) {
5469                 res.status(404).json({
5470                   msg: "Subject section student not found!"
5471                 });
5472               } else {
5473                 let j = 0;
5474                 let studentData = [];
5475                 for (j = 0; j < subsectstud.length; j++) {
5476                   const {
5477                     subsectstudID
5478                   } = subsectstud[j];
5479                   const name = await utils.
5480                     getStudentNameBySubsectstudID(
5481                       subsectstudID
5482                     );
5483                   let sex = await utils.
5484                     getStudentSexBySubsectstudID(
5485                       subsectstudID
5486                     );
5487                   const imageUrl = await utils.
5488                     getStudentImageUrlBySubsectstudID(
5489                       subsectstudID
5490                     );
5491                   const grades = await Grade.findAll({
5492                     where: {
5493                       componentID,
5494                       subcomponentID: subcomp[i].subcompID,
5495                       classRecordID: subjectsection.
5496                         classRecordID,
5497                       subsectstudID: subsectstud[j].
5498                         subsectstudID,
5499                         quarter
5500                     },
5501                     order: [
5502                       ["dateGiven", "DESC"]
5503                     ]
5504                   }).then(async grades => {
5505                     if (grades.length == 0) {
5506                       return [];
5507                     } else {
5508                       let k = 0;
5509                       let gradesData = [];

```

```

5502             for (k = 0; k < grades.length; k++) {
5503                 const {
5504                     gradeID,
5505                     dateGiven,
5506                     score,
5507                     total,
5508                     attendance
5509                 } = grades[k];
5510                 gradesData.push({
5511                     gradeID,
5512                     dateGiven,
5513                     score,
5514                     total,
5515                     attendance
5516                 });
5517             }
5518             return gradesData;
5519         }
5520     );
5521     let k = 0;
5522     let sumScores = 0;
5523     let sumItems = 0;
5524     let excused = 0;
5525     let ps = -1;
5526     for (k = 0; k < grades.length; k++) {
5527         if (grades[k].attendance == "E") {
5528             excused = excused + 1;
5529         } else if (grades[k].attendance == "A") {
5530             sumScores = sumScores + 0;
5531             sumItems = sumItems + parseFloat(grades[k].total);
5532         } else {
5533             sumScores = sumScores + parseFloat(grades[k].score);
5534             sumItems = sumItems + parseFloat(grades[k].total);
5535         }
5536     }
5537     if (grades.length != 0) {
5538         if (grades.length == excused) {
5539             ps = -1;
5540         } else {
5541             ps = (sumScores / sumItems) * 100;
5542         }
5543     }
5544     studentData.push({
5545         sex,
5546         imageUrl,
5547         subsectstudID,
5548         name,
5549         grades,
5550         ps
5551     });
5552 }
5553     return studentData;
5554 }
5555 }
5556 );
5557 data2.sort((a, b) => (a.name > b.name ? 1 : -1));
5558 let tempData = [
5559     ...data2.filter(a => a.sex == "M"),
5560     ...data2.filter(a => a.sex == "F")
5561 ];
5562 data1.push({
5563     subcompID,
5564     name,
5565     data: tempData
5566 });
5567 }
5568     return data1;
5569 }
5570 );
5571 let ave = await StudentWeightedScore.findAll({
5572     where: {

```

```

5573         classRecordID: subjectsection.classRecordID ,
5574         quarter
5575     }
5576 }).then(async studweightedscore => {
5577     if (studweightedscore.length == 0) {
5578         return [];
5579     } else {
5580         let l = 0;
5581         const aveData = [];
5582         for (l = 0; l < studweightedscore.length; l++) {
5583             const {
5584                 subsectstudID ,
5585                 faWS ,
5586                 wwWS ,
5587                 ptWS ,
5588                 qeWS ,
5589                 actualGrade
5590             } = studweightedscore[l];
5591             const name = await utils.
5592                 getStudentNameBySubsectstudID(
5593                     subsectstudID
5594                 );
5595             let sex = await utils.getStudentSexBySubsectstudID(
5596                 subsectstudID
5597             );
5598             let ws = 0;
5599             switch (component) {
5600                 case "Formative Assessment":
5601                     {
5602                         ws = faWS;
5603                         break;
5604                     }
5605                 case "Written Works":
5606                     {
5607                         ws = wwWS;
5608                         break;
5609                     }
5610                 case "Performance Task":
5611                     {
5612                         ws = ptWS;
5613                         break;
5614                     }
5615                 case "Quarterly Exam":
5616                     {
5617                         ws = qeWS;
5618                         break;
5619                     }
5620                 default:
5621                     break;
5622             }
5623             aveData.push({
5624                 sex ,
5625                 subsectstudID ,
5626                 ws ,
5627                 name
5628             });
5629         }
5630     }
5631 });
5632 ave.sort((a, b) => (a.name > b.name ? 1 : -1));
5633 let tempData2 = [
5634     ...ave.filter(a => a.sex == "M"),
5635     ...ave.filter(a => a.sex == "F")
5636 ];
5637 res.status(200).json({
5638     componentID ,
5639     component ,
5640     grades ,
5641     ave: tempData2
5642 });
5643 } else {
5644     res.status(404).json({
5645         msg: "Component not found!"

```

```

5646         });
5647     });
5648   });
5649 } else {
5650   res.status(404).json({
5651     msg: "Subject section not found!"
5652   });
5653 }
5654 );
5655 }
5656 );
5657
5658 // @router POST api/registrar/subcompinfo
5659 // @desc Get component information by componentID and classRecordID
5660 // @access Private
5661
5662 router.post(
5663   "/subcompinfo",
5664   passport.authenticate(directorRegistrar, {
5665     session: false
5666   }),
5667   async(req, res) => {
5668     const {
5669       classRecordID,
5670       componentID,
5671       subcompID,
5672       quarter
5673     } = req.body;
5674     SubjectSection.findOne({
5675       where: {
5676         classRecordID
5677       }
5678     }).then(async subjectsection => {
5679       if (subjectsection) {
5680         if (false) {
5681           res.status(400).json({
5682             msg: "Not authorized!"
5683           });
5684         } else {
5685           Component.findOne({
5686             where: {
5687               componentID
5688             }
5689           }).then(async component => {
5690             if (component) {
5691               let component = await utils.getComponentName(
5692                 componentID);
5693               let {
5694                 subcomponentID,
5695                 name,
5696                 data
5697               } = await Subcomponent.findOne({
5698                 where: {
5699                   componentID,
5700                   classRecordID: subjectsection.classRecordID,
5701                   subcompID,
5702                   quarter
5703                 }
5704               ).then(async subcomp => {
5705                 if (!subcomp) {
5706                   res.status(404).json({
5707                     msg: "Subcomponents not found!"
5708                   });
5709                 } else {
5710                   const subcomponentID = subcomp.subcompID;
5711                   let name = await utils.getSubcomponentName(
5712                     subcomponentID);
5713                   name = name + ' (' + subcomp.compWeight + '%)';
5714                   const data2 = await SubjectSectionStudent.findAll({
5715                     where: {
5716                       subsectID: subjectsection.subsectID
5717                     }
5718                   }).then(async subsectstud => {

```

```

5717     if (subsectstud.length == 0) {
5718       res.status(404).json({
5719         msg: "Subject section student not found!"
5720       });
5721     } else {
5722       let j = 0;
5723       let studentData = [];
5724       for (j = 0; j < subsectstud.length; j++) {
5725         const {
5726           subsectstudID
5727         } = subsectstud[j];
5728         const name = await utils.
5729           getStudentNameBySubsectstudID(
5730             subsectstudID
5731           );
5732         const sex = await utils.
5733           getStudentSexBySubsectstudID(
5734             subsectstudID
5735           );
5736         const imageUrl = await utils.
5737           getStudentImageUrlBySubsectstudID(
5738             subsectstudID
5739           );
5740         const grades = await Grade.findAll({
5741           where: {
5742             componentID,
5743             subcomponentID: subcomp.subcompID,
5744             classRecordID: subjectsection.
5745               classRecordID,
5746             subsectstudID: subsectstud[j].
5747               subsectstudID,
5748               quarter
5749             },
5750             order: [
5751               ["date", "DESC"]
5752             ]
5753           }).then(async grades => {
5754             if (grades.length == 0) {
5755               return [];
5756             } else {
5757               let k = 0;
5758               let gradesData = [];
5759               for (k = 0; k < grades.length; k++) {
5760                 const {
5761                   gradeID,
5762                   dateGiven,
5763                   score,
5764                   total,
5765                   description,
5766                   attendance
5767                 } = grades[k];
5768                 gradesData.push({
5769                   gradeID,
5770                   dateGiven,
5771                   score,
5772                   total,
5773                   description,
5774                   attendance
5775                 });
5776               let k = 0;
5777               let sumScores = 0;
5778               let sumItems = 0;
5779               let excused = 0;
5780               let ps = -1;
5781               for (k = 0; k < grades.length; k++) {
5782                 if (grades[k].attendance == "E") {
5783                   excused = excused + 1;
5784                 } else if (grades[k].attendance == "A") {
5785                   sumScores = sumScores + 0;
5786                   sumItems = sumItems + parseFloat(grades[k]

```

```

                ].total);
5786     } else {
5787         sumScores = sumScores + parseFloat(grades
5788             [k].score);
5789         sumItems = sumItems + parseFloat(grades[k
5790             ].total);
5791     }
5792     if (grades.length != 0) {
5793         if (grades.length == excused) {
5794             ps = -1;
5795         } else {
5796             ps = (sumScores / sumItems) * 100;
5797         }
5798     }
5799     studentData.push({
5800         sex,
5801         imageUrl,
5802         subsectstudID,
5803         name,
5804         grades,
5805         ps
5806     });
5807 }
5808 studentData.sort((a, b) => (a.name > b.name ? 1
5809 : -1));
5810 return [
5811     ...studentData.filter(a => a.sex == "M"),
5812     ...studentData.filter(a => a.sex == "F")
5813 ];
5814 );
5815
5816 return {
5817     subcomponentID,
5818     name,
5819     data: data2
5820 };
5821 }
5822 );
5823 let ave = await StudentWeightedScore.findAll({
5824     where: {
5825         classRecordID: subjectsection.classRecordID,
5826         quarter
5827     }
5828 }).then(async studweightedscore => {
5829     if (studweightedscore.length == 0) {
5830         return [];
5831     } else {
5832         let l = 0;
5833         const aveData = [];
5834         for (l = 0; l < studweightedscore.length; l++) {
5835             const {
5836                 subsectstudID,
5837                 faWS,
5838                 wwWS,
5839                 ptWS,
5840                 qeWS,
5841                 actualGrade
5842             } = studweightedscore[l];
5843             const name = await utils.
5844                 getStudentNameBySubsectstudID(
5845                     subsectstudID
5846                 );
5847             let sex = await utils.
5848                 getStudentSexBySubsectstudID(
5849                     subsectstudID
5850                 );
5851             let ws = 0;
5852             switch (component) {
5853                 case "Formative Assessment":
5854                 {
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
60100
60101
60102
60103
60104
60105
60106
60107
60108
60109
60110
60111
60112
60113
60114
60115
60116
60117
60118
60119
60120
60121
60122
60123
60124
60125
60126
60127
60128
60129
60130
60131
60132
60133
60134
60135
60136
60137
60138
60139
60140
60141
60142
60143
60144
60145
60146
60147
60148
60149
60150
60151
60152
60153
60154
60155
60156
60157
60158
60159
60160
60161
60162
60163
60164
60165
60166
60167
60168
60169
60170
60171
60172
60173
60174
60175
60176
60177
60178
60179
60180
60181
60182
60183
60184
60185
60186
60187
60188
60189
60190
60191
60192
60193
60194
60195
60196
60197
60198
60199
60200
60201
60202
60203
60204
60205
60206
60207
60208
60209
60210
60211
60212
60213
60214
60215
60216
60217
60218
60219
60220
60221
60222
60223
60224
60225
60226
60227
60228
60229
60230
60231
60232
60233
60234
60235
60236
60237
60238
60239
60240
60241
60242
60243
60244
60245
60246
60247
60248
60249
60250
60251
60252
60253
60254
60255
60256
60257
60258
60259
60260
60261
60262
60263
60264
60265
60266
60267
60268
60269
60270
60271
60272
60273
60274
60275
60276
60277
60278
60279
60280
60281
60282
60283
60284
60285
60286
60287
60288
60289
60290
60291
60292
60293
60294
60295
60296
60297
60298
60299
60300
60301
60302
60303
60304
60305
60306
60307
60308
60309
60310
60311
60312
60313
60314
60315
60316
60317
60318
60319
60320
60321
60322
60323
60324
60325
60326
60327
60328
60329
60330
60331
60332
60333
60334
60335
60336
60337
60338
60339
60340
60341
60342
60343
60344
60345
60346
60347
60348
60349
60350
60351
60352
60353
60354
60355
60356
60357
60358
60359
60360
60361
60362
60363
60364
60365
60366
60367
60368
60369
60370
60371
60372
60373
60374
60375
60376
60377
60378
60379
60380
60381
60382
60383
60384
60385
60386
60387
60388
60389
60390
60391
60392
60393
60394
60395
60396
60397
60398
60399
60400
60401
60402
60403
60404
60405
60406
60407
60408
60409
60410
60411
60412
60413
60414
60415
60416
60417
60418
60419
60420
60421
60422
60423
60424
60425
60426
60427
60428
60429
60430
60431
60432
60433
60434
60435
60436
60437
60438
60439
60440
60441
60442
60443
60444
60445
60446
60447
60448
60449
60450
60451
60452
60453
60454
60455
60456
60457
60458
60459
60460
60461
60462
60463
60464
60465
60466
60467
60468
60469
60470
60471
60472
60473
60474
60475
60476
60477
60478
60479
60480
60481
60482
60483
60484
60485
60486
60487
60488
60489
60490
60491
60492
60493
60494
60495
60496
60497
60498
60499
60500
60501
60502
60503
60504
60505
60506
60507
60508
60509
60510
60511
60512
60513
60514
60515
60516
60517
60518
60519
60520
60521
60522
60523
60524
60525
60526
60527
60528
60529
60530
60531
60532
60533
60534
60535
60536
60537
60538
60539
60540
60541
60542
60543
60544
60545
60546
60547
60548
60549
60550
60551
60552
60553
60554
60555
60556
60557
60558
60559
60560
60561
60562
60563
60564
60565
60566
60567
60568
60569
60570
60571
60572
60573
60574
60575
60576
60577
60578
60579
60580
60581
60582
60583
60584
60585
60586
60587
60588
60589
60590
60591
60592
60593
60594
60595
60596
60597
60598
60599
60600
60601
60602
60603
60604
60605
60606
60607
60608
60609
60610
60611
60612
60613
60614
60615
60616
60617
60618
60619
60620
60621
60622
60623
60624
60625
60626
60627
60628
60629
60630
60631
60632
60633
60634
60635
60636
60637
60638
60639
60640
60641
60642
60643
60644
60645
60646
60647
60648
60649
60650
60651
60652
60653
60654
60655
60656
60657
60658
60659
60660
60661
60662
60663
60664
60665
60666
60667
60668
60669
60670
60671
60672
60673
60674
60675
60676
60677
60678
60679
60680
60681
60682
60683
60684
60685
60686
60687
60688
60689
60690
60691
60692
60693
60694
60695
60696
60697
60698
60699
60700
60701
60702
60703
60704
60705
60706
60707
60708
60709
60710
60711
60712
60713
60714
60715
60716
60717
60718
60719
60720
60721
60722
60723
60724
60725
60726
60727
60728
60729
60730
60731
60732
60733
60734
60735
60736
60737
60738
60739
60740
60741
60742
60743
60744
60745
60746
60747
60748
60749
60750
60751
60752
60753
60754
60755
60756
60757
60758
60759
60760
60761
60762
60763
60764
60765
60766
60767
60768
60769
60770
60771
60772
60773
60774
60775
60776
60777
60778
60779
60780
60781
60782
60783
60784
60785
60786
60787
60788
60789
60790
60791
60792
60793
60794
60795
60796
60797
60798
60799
60800
60801
60802
60803
60804
60805
60806
60807
60808
60809
60810
60811
60812
60813
60814
60815
60816
60817
60818
60819
60820
60821
60822
60823
60824
60825
60826
60827
60828
60829
60830
60831
60832
60833
60834
60835
60836
60837
60838
60839
60840
60841
60842
60843
60844
60845
60846
60847
60848
60849
60850
60851
60852
60853
60854
60855
60856
60857
60858
60859
60860
60861
60862
60863
60864
60865
60866
60867
60868
60869
60870
60871
60872
60873
60874
60875
60876
60877
60878
60879
60880
60881
60882
60883
60884
60885
60886
60887
60888
60889
60890
60891
60892
60893
60894
60895
60896
60897
60898
60899
60900
60901
60902
60903
60904
60905
60906
60907
60908
60909
60910
60911
60912
60913
60914
60915
60916
60917
60918
60919
60920
60921
60922
60923
60924
60925
60926
60927
60928
60929
60930
60931
60932
60933
60934
60935
60936
60937
60938
60939
60940
60941
60942
60943
60944
60945
60946
60947
60948
60949
60950
60951
60952
60953
60954
60955
60956
60957
60958
60959
60960
60961
60962
60963
60964
60965
60966
60967
60968
60969
60970
60971
60972
60973
60974
60975
60976
60977
60978
60979
60980
60981
60982
60983
60984
60985
60986
60987
60988
60989
60990
60991
60992
60993
60994
60995
60996
60997
60998
60999
609999
61000
61001
61002
61003
61004
61005
61006
61007
61008
61009
610010
610011
610012
610013
610014
610015
610016
610017
610018
610019
610020
610021
610022
610023
610024
610025
610026
610027
610028
610029
610030
610031
610032
610033
610034
610035
610036
610037
610038
610039
610040
610041
610042
610043
610044
610045
610046
610047
610048
610049
610050
610051
610052
610053
610054
610055
610056
610057
610058
610059
610060
610061
610062
610063
610064
610065
610066
610067
610068
610069
610070
610071
610072
610073
610074
610075
610076
610077
610078
610079
610080
610081
610082
610083
610084
610085
610086
610087
610088
610089
610090
610091
610092
610093
610094
610095
610096
610097
610098
610099
6100100
6100101
6100102
6100103
6100104
6100105
6100106
6100107
6100108
6100109
6100110
6100111
6100112
6100113
6100114
6100115
6100116
6100117
6100118
6100119
6100120
6100121
6100122
6100123
6100124
6100125
6100126
6100127
6100128
6100129
6100130
6100131
6100132
6100133
6100134
6100135
6100136
6100137
6100138
6100139
6100140
6100141
6100142
6100143
6100144
6100145
6100146
6100147
6100148
6100149
6100150
6100151
6100152
6100153
6100154
6100155
6100156
6100157
6100158
6100159
6100160
6100161
6100162
6100163
6100164
6100165
6100166
6100167
6100168
6100169
6100170
6100171
6100172
6100173
6100174
6100175
6100176
6100177
6100178
6100179
6100180
6100181
6100182
6100183
6100184
6100185
6100186
6100187
6100188
6100189
6100190
6100191
6100192
6100193
6100194
6100195
6100196
6100197
6100198
6100199
6100200
6100201
6100202
6100203
6100204
6100205
6100206
6100207
6100208
6100209
6100210
6100211
6100212
6100213
6100214
6100215
6100216
6100217
6100218
6100219
6100220
6100221
6100222
6100223
6100224
6100225
6100226
6100227
6100228
6100229
6100230
6100231
6100232
6100233
6100234
6100235
6100236
6100237
6100238
6100239
6100240
6100241
6100242
6100243
6100244
6100245
6100246
6100247
6100248
6100249
6100250
6100251
6100252
6100253
6100254
6100255
6100256
6100257
6100258
6100259
6100260
6100261
6100262
6100263
6100264
6100265
6100266
6100267
6100268
6100269
6100270
6100271
6100272
6100273
6100274
6100275
6100276
6100277
6100278
6100279
6100280
6100281
6100282
6100283
6100284
6100285
6100286
6100287
6100288
6100289
6100290
6100291
6100292
6100293
6100294
6100295
6100296
6100297
6100298
6100299
6100300
6100301
6100302
6100303
6100304
6100305
6100306
6100307
6100308
6100309
6100310
6100311
6100312
6100313
6100314
6100315
6100316
6100317
6100318
6100319
6100320
6100321
6100322
6100323
6100324
6100325
6100326
6100327
6100328
6100329
6100330
6100331
6100332
6100333
6100334
6100335
6100336
6100337
6100338
6100339
61003310
61003311
61003312
61003313
61003314
61003315
61003316
61003317
61003318
61003319
61003320
61003321
61003322
61003323
61003324
61003325
61003326
61003327
61003328
61003329
61003330
61003331
61003332
61003333
61003334
61003335
61003336
61003337
61003338
61003339
610033310
610033311
610033312
610033313
610033314
610033315
610033316
610033317
610033318
610033319
610033320
610033321
610033322
610033323
610033324
610033325
610033326
610033327
610033328
610033329
610033330
610033331
610033332
610033333
610033334
610033335
610033336
610033337
610033338
610033339
610033340
610033341
610033342
610033343
610033344
610033345
610033346
610033347
610033348
610033349
610033350
610033351
610033352
610033353
610033354
610033355
610033356
610033357
610033358
610033359
610033360
610033361
610033362
610033363
610033364
610033365
610033366
610033367
610033368
610033369
610033370
610033371
610033372
610033373
610033374
610033375
610033376
610033377
610033378
610033379
610033380
610033381
610033382
610033383
610033384
610033385
610033386
610033387
610033388
610033389
610033390
610033391
610033392
610033393
610033394
610033395
610033396
610033397
610033398
610033399
6100333100
6100333101
6100333102
6100333103
6100333104
6100333
```

```

5853             ws = faWS;
5854         }
5855     case "Written Works":
5856     {
5857         ws = wwWS;
5858     }
5859     case "Performance Task":
5860     {
5861         ws = ptWS;
5862     }
5863     case "Quarterly Exam":
5864     {
5865         ws = qeWS;
5866     }
5867     default:
5868     "";
5869 }
5870 aveData.push({
5871     sex,
5872     subsectstudID,
5873     ws,
5874     name
5875 });
5876 }
5877 return aveData;
5878 }
5879 });
5880 ave.sort((a, b) => (a.name > b.name ? 1 : -1));
5881 res.status(200).json({
5882     componentID,
5883     component,
5884     subcompID: subcomponentID,
5885     subcompName: name,
5886     data,
5887     ave: [
5888         ...ave.filter(a => a.sex == "M"),
5889         ...ave.filter(a => a.sex == "F")
5890     ]
5891 });
5892 } else {
5893     res.status(404).json({
5894         msg: "Component not found!"
5895     });
5896 }
5897 });
5898 }
5899 } else {
5900     res.status(404).json({
5901         msg: "Subject section not found!"
5902     });
5903 }
5904 });
5905 }
5906 );
5907
5908 // @route POST api/registrar/deleterecord
5909 // @desc Delete a record
5910 // @access Private
5911
5912 router.post(
5913     "/deleterecord",
5914     passport.authenticate("registrar", {
5915         session: false
5916     }),
5917     async(req, res) => {
5918         const {
5919             classRecordID,
5920             description,
5921             dateGiven,
5922             subcompID,
5923             componentID,
5924             quarter
5925         } = req.body;

```

```

5926     SubjectSection.findOne({
5927       where: {
5928         classRecordID
5929       }
5930     }).then(subsect => {
5931       if (subsect) {
5932         if (false) {
5933           res.status(401).json({
5934             msg: "Not authorized!"
5935           });
5936         } else {
5937           ClassRecordStatus.findOne({
5938             where: {
5939               classRecordID: subsect.classRecordID
5940             }
5941           }).then(async crs => {
5942             if (crs) {
5943               let invalid = false;
5944               if (quarter == "Q1") {
5945                 invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 ==
5946                 "F";
5947               } else if (quarter == "Q2") {
5948                 invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 ==
5949                 "F";
5950               } else if (quarter == "Q3") {
5951                 invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 ==
5952                 "F";
5953               } else {
5954                 invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 ==
5955                 "F";
5956               }
5957             if (invalid) {
5958               res.status(400).json({
5959                 msg: "You are not authorized to edit this class
5960                   record."
5961               });
5962             } else {
5963               Grade.findAll({
5964                 where: {
5965                   description,
5966                   dateGiven,
5967                   subcomponentID: subcompID,
5968                   componentID,
5969                   quarter
5970                 }
5971               }).then(async grades => {
5972                 if (grades.length != 0) {
5973                   const name = await utils.getAccountName(req.user.
5974                     accountID);
5975                   const section = await utils.getSectionName(
5976                     subsect.sectionID
5977                   );
5978                   const subject = await utils.getSubjectName(
5979                     subsect.subjectID
5980                   );
5981                   const logID = await ActivityLog.create({
5982                     type: "DELETE",
5983                     classRecordID,
5984                     position: "Registrar",
5985                     name,
5986                     section,
5987                     subject,
5988                     quarter,
5989                     timestamp: new Date()
5990                   }).then(async al => {
5991                     if (al) {
5992                       return al.logID;
5993                     }
5994                   });
5995                   for (const [index, value] of grades.entries()) {
5996                     await value.destroy({
5997                       showLog: true,
5998                       logID,
5999                     });
6000                 }
6001               }
6002             }
6003           }
6004         }
6005       }
6006     }
6007   }
6008 
```

```

5993         subsectstudID: value.subsectstudID,
5994         componentID,
5995         subcompID,
5996         value: value.score
5997     });
5998 }
5999 await utils.
6000     refreshStudentWeightedScoreBySubsectID(
6001     subsect.subsectID
6002 );
6003     res.status(200).json({
6004         msg: "Deleted successfully!"
6005     });
6006 } else {
6007     res.status(404).json({
6008         msg: "Grades not found!"
6009     });
6010 });
6011 }
6012 } else {
6013     res.status(404).json({
6014         msg: "Class record status does not exist"
6015     });
6016 }
6017 });
6018 }
6019 } else {
6020     res.status(404).json({
6021         msg: "Subject section not found!"
6022     });
6023 }
6024 });
6025 );
6026 );
6027
6028 // @route api/teacher/getrecinfo
6029 // @desc Get grade record info by one gradeID
6030 // @access Private
6031
6032 router.post(
6033     "/getrecinfo",
6034     passport.authenticate("registrar", {
6035         session: false
6036     }),
6037     async(req, res) => {
6038         const {
6039             classRecordID,
6040             componentID,
6041             subcompID,
6042             quarter,
6043             gradeID
6044         } = req.body;
6045         SubjectSection.findOne({
6046             where: {
6047                 classRecordID
6048             }
6049         }).then(async ss => {
6050             if (ss) {
6051                 if (false) {
6052                     res.status(401).json({
6053                         msg: "Not authorized!"
6054                     });
6055                 } else {
6056                     Grade.findOne({
6057                         where: {
6058                             gradeID
6059                         }
6060                     }).then(async grade => {
6061                         if (grade) {
6062                             const {
6063                                 description,
6064                                 dateGiven,

```

```

6065         total
6066     } = grade;
6067     const component = await utils.getComponentName(
6068         componentID);
6069     const weight = await Subcomponent.findOne({
6070         where: {
6071             subcompID
6072         }
6073     }).then(sc => {
6074         if (sc) {
6075             return sc.compWeight;
6076         }
6077     });
6078     let subcompName = await utils.getSubcomponentName(
6079         subcompID);
6080     subcompName = `${subcompName} (${weight}%}`;
6081     Grade.findAll({
6082         where: {
6083             quarter,
6084             subcomponentID: subcompID,
6085             componentID,
6086             description: grade.description,
6087             dateGiven: grade.dateGiven
6088         }
6089     }).then(async grades => {
6090         if (grades.length == 0) {
6091             res.status(404).json({
6092                 msg: "Grades not found!"
6093             });
6094         } else {
6095             let data = [];
6096             for (const [index, value] of grades.entries()) {
6097                 const {
6098                     subsectstudID,
6099                     score,
6100                     gradeID,
6101                     attendance
6102                 } = value;
6103                 const name = await utils.
6104                     getStudentNameBySubsectstudID(
6105                         subsectstudID
6106                     );
6107                 const imageUrl = await utils.
6108                     getStudentImageUrlBySubsectstudID(
6109                         subsectstudID
6110                     );
6111                 data.push({
6112                     gradeID,
6113                     subsectstudID,
6114                     score: attendance == "A" ?
6115                         "A" : attendance == "E" ?
6116                         "E" : score,
6117                     name,
6118                     imageUrl
6119                 });
6120             }
6121             res.status(200).json({
6122                 componentID,
6123                 component,
6124                 subcompID,
6125                 subcompName,
6126                 dateGiven,
6127                 total,
6128                 description,
6129                 data
6130             });
6131         }
6132     });
6133 }
6134

```

```

6135         }
6136     } else {
6137         res.status(404).json({
6138             msg: "Subject section not found!"
6139         });
6140     }
6141   });
6142 }
6143 );
6144
6145 // @route api/registrar/editrecord
6146 // @desc Edit record
6147 // @access Private
6148
6149 router.post(
6150   "/editrecord",
6151   passport.authenticate("registrar", {
6152     session: false
6153   }),
6154   async(req, res) => {
6155     const reg = /^-?[0-9]*(\.[0-9]*)?$/;
6156     const {
6157       description,
6158       dateGiven,
6159       total,
6160       classRecordID,
6161       subcompID,
6162       componentID,
6163       quarter,
6164       payload
6165     } = req.body;
6166     let totalValid = !isNaN(total) && reg.test(total) && total !== ""
6167       && total !== "-";
6168     SubjectSection.findOne({
6169       where: {
6170         classRecordID
6171       }
6172     }).then(async ss => {
6173       if (ss) {
6174         ClassRecordStatus.findOne({
6175           where: {
6176             classRecordID: ss.classRecordID
6177           }
6178         }).then(async crs => {
6179           if (crs) {
6180             let invalid = false;
6181             if (quarter == "Q1") {
6182               invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 == "F";
6183             } else if (quarter == "Q2") {
6184               invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 == "F";
6185             } else if (quarter == "Q3") {
6186               invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 == "F";
6187             } else {
6188               invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 == "F";
6189             }
6190             if (invalid) {
6191               res.status(400).json({
6192                 msg: "You are not authorized to edit this class
6193                   record."
6194               });
6195             } else {
6196               if (!totalValid) {
6197                 res.status(400).json({
6198                   msg: "Total score is invalid. Enter numbers only."
6199                 });
6200               } else {
6201                 if (total <= 0) {
6202                   return res.status(400).json({
6203                     msg: "Total must be more than 0"
6204                   });
6205                 }
6206               }
6207             }
6208           }
6209         });
6210       }
6211     });
6212   });
6213 }
6214
6215 // @route api/registrar/getrecords
6216 // @desc Get records
6217 // @access Private
6218
6219 router.get(
6220   "/getrecords",
6221   passport.authenticate("registrar", {
6222     session: false
6223   })
6224   )
6225 );
6226
6227 // @route api/registrar/getrecords
6228 // @desc Get records
6229 // @access Private
6230
6231 router.get(
6232   "/getrecords",
6233   passport.authenticate("registrar", {
6234     session: false
6235   })
6236   )
6237 );
6238
6239 // @route api/registrar/getrecords
6240 // @desc Get records
6241 // @access Private
6242
6243 router.get(
6244   "/getrecords",
6245   passport.authenticate("registrar", {
6246     session: false
6247   })
6248   )
6249 );
6250
6251 // @route api/registrar/getrecords
6252 // @desc Get records
6253 // @access Private
6254
6255 router.get(
6256   "/getrecords",
6257   passport.authenticate("registrar", {
6258     session: false
6259   })
6260   )
6261 );
6262
6263 // @route api/registrar/getrecords
6264 // @desc Get records
6265 // @access Private
6266
6267 router.get(
6268   "/getrecords",
6269   passport.authenticate("registrar", {
6270     session: false
6271   })
6272   )
6273 );
6274
6275 // @route api/registrar/getrecords
6276 // @desc Get records
6277 // @access Private
6278
6279 router.get(
6280   "/getrecords",
6281   passport.authenticate("registrar", {
6282     session: false
6283   })
6284   )
6285 );
6286
6287 // @route api/registrar/getrecords
6288 // @desc Get records
6289 // @access Private
6290
6291 router.get(
6292   "/getrecords",
6293   passport.authenticate("registrar", {
6294     session: false
6295   })
6296   )
6297 );
6298
6299 // @route api/registrar/getrecords
6300 // @desc Get records
6301 // @access Private
6302
6303 router.get(
6304   "/getrecords",
6305   passport.authenticate("registrar", {
6306     session: false
6307   })
6308   )
6309 );
6310
6311 // @route api/registrar/getrecords
6312 // @desc Get records
6313 // @access Private
6314
6315 router.get(
6316   "/getrecords",
6317   passport.authenticate("registrar", {
6318     session: false
6319   })
6320   )
6321 );
6322
6323 // @route api/registrar/getrecords
6324 // @desc Get records
6325 // @access Private
6326
6327 router.get(
6328   "/getrecords",
6329   passport.authenticate("registrar", {
6330     session: false
6331   })
6332   )
6333 );
6334
6335 // @route api/registrar/getrecords
6336 // @desc Get records
6337 // @access Private
6338
6339 router.get(
6340   "/getrecords",
6341   passport.authenticate("registrar", {
6342     session: false
6343   })
6344   )
6345 );
6346
6347 // @route api/registrar/getrecords
6348 // @desc Get records
6349 // @access Private
6350
6351 router.get(
6352   "/getrecords",
6353   passport.authenticate("registrar", {
6354     session: false
6355   })
6356   )
6357 );
6358
6359 // @route api/registrar/getrecords
6360 // @desc Get records
6361 // @access Private
6362
6363 router.get(
6364   "/getrecords",
6365   passport.authenticate("registrar", {
6366     session: false
6367   })
6368   )
6369 );
6370
6371 // @route api/registrar/getrecords
6372 // @desc Get records
6373 // @access Private
6374
6375 router.get(
6376   "/getrecords",
6377   passport.authenticate("registrar", {
6378     session: false
6379   })
6380   )
6381 );
6382
6383 // @route api/registrar/getrecords
6384 // @desc Get records
6385 // @access Private
6386
6387 router.get(
6388   "/getrecords",
6389   passport.authenticate("registrar", {
6390     session: false
6391   })
6392   )
6393 );
6394
6395 // @route api/registrar/getrecords
6396 // @desc Get records
6397 // @access Private
6398
6399 router.get(
6400   "/getrecords",
6401   passport.authenticate("registrar", {
6402     session: false
6403   })
6404   )
6405 );
6406
6407 // @route api/registrar/getrecords
6408 // @desc Get records
6409 // @access Private
6410
6411 router.get(
6412   "/getrecords",
6413   passport.authenticate("registrar", {
6414     session: false
6415   })
6416   )
6417 );
6418
6419 // @route api/registrar/getrecords
6420 // @desc Get records
6421 // @access Private
6422
6423 router.get(
6424   "/getrecords",
6425   passport.authenticate("registrar", {
6426     session: false
6427   })
6428   )
6429 );
6430
6431 // @route api/registrar/getrecords
6432 // @desc Get records
6433 // @access Private
6434
6435 router.get(
6436   "/getrecords",
6437   passport.authenticate("registrar", {
6438     session: false
6439   })
6440   )
6441 );
6442
6443 // @route api/registrar/getrecords
6444 // @desc Get records
6445 // @access Private
6446
6447 router.get(
6448   "/getrecords",
6449   passport.authenticate("registrar", {
6450     session: false
6451   })
6452   )
6453 );
6454
6455 // @route api/registrar/getrecords
6456 // @desc Get records
6457 // @access Private
6458
6459 router.get(
6460   "/getrecords",
6461   passport.authenticate("registrar", {
6462     session: false
6463   })
6464   )
6465 );
6466
6467 // @route api/registrar/getrecords
6468 // @desc Get records
6469 // @access Private
6470
6471 router.get(
6472   "/getrecords",
6473   passport.authenticate("registrar", {
6474     session: false
6475   })
6476   )
6477 );
6478
6479 // @route api/registrar/getrecords
6480 // @desc Get records
6481 // @access Private
6482
6483 router.get(
6484   "/getrecords",
6485   passport.authenticate("registrar", {
6486     session: false
6487   })
6488   )
6489 );
6490
6491 // @route api/registrar/getrecords
6492 // @desc Get records
6493 // @access Private
6494
6495 router.get(
6496   "/getrecords",
6497   passport.authenticate("registrar", {
6498     session: false
6499   })
6500   )
6501 );
6502
6503 // @route api/registrar/getrecords
6504 // @desc Get records
6505 // @access Private
6506
6507 router.get(
6508   "/getrecords",
6509   passport.authenticate("registrar", {
6510     session: false
6511   })
6512   )
6513 );
6514
6515 // @route api/registrar/getrecords
6516 // @desc Get records
6517 // @access Private
6518
6519 router.get(
6520   "/getrecords",
6521   passport.authenticate("registrar", {
6522     session: false
6523   })
6524   )
6525 );
6526
6527 // @route api/registrar/getrecords
6528 // @desc Get records
6529 // @access Private
6530
6531 router.get(
6532   "/getrecords",
6533   passport.authenticate("registrar", {
6534     session: false
6535   })
6536   )
6537 );
6538
6539 // @route api/registrar/getrecords
6540 // @desc Get records
6541 // @access Private
6542
6543 router.get(
6544   "/getrecords",
6545   passport.authenticate("registrar", {
6546     session: false
6547   })
6548   )
6549 );
6550
6551 // @route api/registrar/getrecords
6552 // @desc Get records
6553 // @access Private
6554
6555 router.get(
6556   "/getrecords",
6557   passport.authenticate("registrar", {
6558     session: false
6559   })
6560   )
6561 );
6562
6563 // @route api/registrar/getrecords
6564 // @desc Get records
6565 // @access Private
6566
6567 router.get(
6568   "/getrecords",
6569   passport.authenticate("registrar", {
6570     session: false
6571   })
6572   )
6573 );
6574
6575 // @route api/registrar/getrecords
6576 // @desc Get records
6577 // @access Private
6578
6579 router.get(
6580   "/getrecords",
6581   passport.authenticate("registrar", {
6582     session: false
6583   })
6584   )
6585 );
6586
6587 // @route api/registrar/getrecords
6588 // @desc Get records
6589 // @access Private
6590
6591 router.get(
6592   "/getrecords",
6593   passport.authenticate("registrar", {
6594     session: false
6595   })
6596   )
6597 );
6598
6599 // @route api/registrar/getrecords
6600 // @desc Get records
6601 // @access Private
6602
6603 router.get(
6604   "/getrecords",
6605   passport.authenticate("registrar", {
6606     session: false
6607   })
6608   )
6609 );
6610
6611 // @route api/registrar/getrecords
6612 // @desc Get records
6613 // @access Private
6614
6615 router.get(
6616   "/getrecords",
6617   passport.authenticate("registrar", {
6618     session: false
6619   })
6620   )
6621 );
6622
6623 // @route api/registrar/getrecords
6624 // @desc Get records
6625 // @access Private
6626
6627 router.get(
6628   "/getrecords",
6629   passport.authenticate("registrar", {
6630     session: false
6631   })
6632   )
6633 );
6634
6635 // @route api/registrar/getrecords
6636 // @desc Get records
6637 // @access Private
6638
6639 router.get(
6640   "/getrecords",
6641   passport.authenticate("registrar", {
6642     session: false
6643   })
6644   )
6645 );
6646
6647 // @route api/registrar/getrecords
6648 // @desc Get records
6649 // @access Private
6650
6651 router.get(
6652   "/getrecords",
6653   passport.authenticate("registrar", {
6654     session: false
6655   })
6656   )
6657 );
6658
6659 // @route api/registrar/getrecords
6660 // @desc Get records
6661 // @access Private
6662
6663 router.get(
6664   "/getrecords",
6665   passport.authenticate("registrar", {
6666     session: false
6667   })
6668   )
6669 );
6670
6671 // @route api/registrar/getrecords
6672 // @desc Get records
6673 // @access Private
6674
6675 router.get(
6676   "/getrecords",
6677   passport.authenticate("registrar", {
6678     session: false
6679   })
6680   )
6681 );
6682
6683 // @route api/registrar/getrecords
6684 // @desc Get records
6685 // @access Private
6686
6687 router.get(
6688   "/getrecords",
6689   passport.authenticate("registrar", {
6690     session: false
6691   })
6692   )
6693 );
6694
6695 // @route api/registrar/getrecords
6696 // @desc Get records
6697 // @access Private
6698
6699 router.get(
6700   "/getrecords",
6701   passport.authenticate("registrar", {
6702     session: false
6703   })
6704   )
6705 );
6706
6707 // @route api/registrar/getrecords
6708 // @desc Get records
6709 // @access Private
6710
6711 router.get(
6712   "/getrecords",
6713   passport.authenticate("registrar", {
6714     session: false
6715   })
6716   )
6717 );
6718
6719 // @route api/registrar/getrecords
6720 // @desc Get records
6721 // @access Private
6722
6723 router.get(
6724   "/getrecords",
6725   passport.authenticate("registrar", {
6726     session: false
6727   })
6728   )
6729 );
6730
6731 // @route api/registrar/getrecords
6732 // @desc Get records
6733 // @access Private
6734
6735 router.get(
6736   "/getrecords",
6737   passport.authenticate("registrar", {
6738     session: false
6739   })
6740   )
6741 );
6742
6743 // @route api/registrar/getrecords
6744 // @desc Get records
6745 // @access Private
6746
6747 router.get(
6748   "/getrecords",
6749   passport.authenticate("registrar", {
6750     session: false
6751   })
6752   )
6753 );
6754
6755 // @route api/registrar/getrecords
6756 // @desc Get records
6757 // @access Private
6758
6759 router.get(
6760   "/getrecords",
6761   passport.authenticate("registrar", {
6762     session: false
6763   })
6764   )
6765 );
6766
6767 // @route api/registrar/getrecords
6768 // @desc Get records
6769 // @access Private
6770
6771 router.get(
6772   "/getrecords",
6773   passport.authenticate("registrar", {
6774     session: false
6775   })
6776   )
6777 );
6778
6779 // @route api/registrar/getrecords
6780 // @desc Get records
6781 // @access Private
6782
6783 router.get(
6784   "/getrecords",
6785   passport.authenticate("registrar", {
6786     session: false
6787   })
6788   )
6789 );
6790
6791 // @route api/registrar/getrecords
6792 // @desc Get records
6793 // @access Private
6794
6795 router.get(
6796   "/getrecords",
6797   passport.authenticate("registrar", {
6798     session: false
6799   })
6800   )
6801 );
6802
6803 // @route api/registrar/getrecords
6804 // @desc Get records
6805 // @access Private
6806
6807 router.get(
6808   "/getrecords",
6809   passport.authenticate("registrar", {
6810     session: false
6811   })
6812   )
6813 );
6814
6815 // @route api/registrar/getrecords
6816 // @desc Get records
6817 // @access Private
6818
6819 router.get(
6820   "/getrecords",
6821   passport.authenticate("registrar", {
6822     session: false
6823   })
6824   )
6825 );
6826
6827 // @route api/registrar/getrecords
6828 // @desc Get records
6829 // @access Private
6830
6831 router.get(
6832   "/getrecords",
6833   passport.authenticate("registrar", {
6834     session: false
6835   })
6836   )
6837 );
6838
6839 // @route api/registrar/getrecords
6840 // @desc Get records
6841 // @access Private
6842
6843 router.get(
6844   "/getrecords",
6845   passport.authenticate("registrar", {
6846     session: false
6847   })
6848   )
6849 );
6850
6851 // @route api/registrar/getrecords
6852 // @desc Get records
6853 // @access Private
6854
6855 router.get(
6856   "/getrecords",
6857   passport.authenticate("registrar", {
6858     session: false
6859   })
6860   )
6861 );
6862
6863 // @route api/registrar/getrecords
6864 // @desc Get records
6865 // @access Private
6866
6867 router.get(
6868   "/getrecords",
6869   passport.authenticate("registrar", {
6870     session: false
6871   })
6872   )
6873 );
6874
6875 // @route api/registrar/getrecords
6876 // @desc Get records
6877 // @access Private
6878
6879 router.get(
6880   "/getrecords",
6881   passport.authenticate("registrar", {
6882     session: false
6883   })
6884   )
6885 );
6886
6887 // @route api/registrar/getrecords
6888 // @desc Get records
6889 // @access Private
6890
6891 router.get(
6892   "/getrecords",
6893   passport.authenticate("registrar", {
6894     session: false
6895   })
6896   )
6897 );
6898
6899 // @route api/registrar/getrecords
6900 // @desc Get records
6901 // @access Private
6902
6903 router.get(
6904   "/getrecords",
6905   passport.authenticate("registrar", {
6906     session: false
6907   })
6908   )
6909 );
6910
6911 // @route api/registrar/getrecords
6912 // @desc Get records
6913 // @access Private
6914
6915 router.get(
6916   "/getrecords",
6917   passport.authenticate("registrar", {
6918     session: false
6919   })
6920   )
6921 );
6922
6923 // @route api/registrar/getrecords
6924 // @desc Get records
6925 // @access Private
6926
6927 router.get(
6928   "/getrecords",
6929   passport.authenticate("registrar", {
6930     session: false
6931   })
6932   )
6933 );
6934
6935 // @route api/registrar/getrecords
6936 // @desc Get records
6937 // @access Private
6938
6939 router.get(
6940   "/getrecords",
6941   passport.authenticate("registrar", {
6942     session: false
6943   })
6944   )
6945 );
6946
6947 // @route api/registrar/getrecords
6948 // @desc Get records
6949 // @access Private
6950
6951 router.get(
6952   "/getrecords",
6953   passport.authenticate("registrar", {
6954     session: false
6955   })
6956   )
6957 );
6958
6959 // @route api/registrar/getrecords
6960 // @desc Get records
6961 // @access Private
6962
6963 router.get(
6964   "/getrecords",
6965   passport.authenticate("registrar", {
6966     session: false
6967   })
6968   )
6969 );
6970
6971 // @route api/registrar/getrecords
6972 // @desc Get records
6973 // @access Private
6974
6975 router.get(
6976   "/getrecords",
6977   passport.authenticate("registrar", {
6978     session: false
6979   })
6980   )
6981 );
6982
6983 // @route api/registrar/getrecords
6984 // @desc Get records
6985 // @access Private
6986
6987 router.get(
6988   "/getrecords",
6989   passport.authenticate("registrar", {
6990     session: false
6991   })
6992   )
6993 );
6994
6995 // @route api/registrar/getrecords
6996 // @desc Get records
6997 // @access Private
6998
6999 router.get(
7000   "/getrecords",
7001   passport.authenticate("registrar", {
7002     session: false
7003   })
7004   )
7005 );
7006
7007 // @route api/registrar/getrecords
7008 // @desc Get records
7009 // @access Private
7010
7011 router.get(
7012   "/getrecords",
7013   passport.authenticate("registrar", {
7014     session: false
7015   })
7016   )
7017 );
7018
7019 // @route api/registrar/getrecords
7020 // @desc Get records
7021 // @access Private
7022
7023 router.get(
7024   "/getrecords",
7025   passport.authenticate("registrar", {
7026     session: false
7027   })
7028   )
7029 );
7030
7031 // @route api/registrar/getrecords
7032 // @desc Get records
7033 // @access Private
7034
7035 router.get(
7036   "/getrecords",
7037   passport.authenticate("registrar", {
7038     session: false
7039   })
7040   )
7041 );
7042
7043 // @route api/registrar/getrecords
7044 // @desc Get records
7045 // @access Private
7046
7047 router.get(
7048   "/getrecords",
7049   passport.authenticate("registrar", {
7050     session: false
7051   })
7052   )
7053 );
7054
7055 // @route api/registrar/getrecords
7056 // @desc Get records
7057 // @access Private
7058
7059 router.get(
7060   "/getrecords",
7061   passport.authenticate("registrar", {
7062     session: false
7063   })
7064   )
7065 );
7066
7067 // @route api/registrar/getrecords
7068 // @desc Get records
7069 // @access Private
7070
7071 router.get(
7072   "/getrecords",
7073   passport.authenticate("registrar", {
7074     session: false
7075   })
7076   )
7077 );
7078
7079 // @route api/registrar/getrecords
7080 // @desc Get records
7081 // @access Private
7082
7083 router.get(
7084   "/getrecords",
7085   passport.authenticate("registrar", {
7086     session: false
7087   })
7088   )
7089 );
7090
7091 // @route api/registrar/getrecords
7092 // @desc Get records
7093 // @access Private
7094
7095 router.get(
7096   "/getrecords",
7097   passport.authenticate("registrar", {
7098     session: false
7099   })
7100   )
7101 );
7102
7103 // @route api/registrar/getrecords
7104 // @desc Get records
7105 // @access Private
7106
7107 router.get(
7108   "/getrecords",
7109   passport.authenticate("registrar", {
7110     session: false
7111   })
7112   )
7113 );
7114
7115 // @route api/registrar/getrecords
7116 // @desc Get records
7117 // @access Private
7118
7119 router.get(
7120   "/getrecords",
7121   passport.authenticate("registrar", {
7122     session: false
7123   })
7124   )
7125 );
7126
7127 // @route api/registrar/getrecords
7128 // @desc Get records
7129 // @access Private
7130
7131 router.get(
7132   "/getrecords",
7133   passport.authenticate("registrar", {
7134     session: false
7135   })
7136   )
7137 );
7138
7139 // @route api/registrar/getrecords
7140 // @desc Get records
7141 // @access Private
7142
7143 router.get(
7144   "/getrecords",
7145   passport.authenticate("registrar", {
7146     session: false
7147   })
7148   )
7149 );
7150
7151 // @route api/registrar/getrecords
7152 // @desc Get records
7153 // @access Private
7154
7155 router.get(
7156   "/getrecords",
7157   passport.authenticate("registrar", {
7158     session: false
7159   })
7160   )
7161 );
7162
7163 // @route api/registrar/getrecords
7164 // @desc Get records
7165 // @access Private
7166
7167 router.get(
7168   "/getrecords",
7169   passport.authenticate("registrar", {
7170     session: false
7171   })
7172   )
7173 );
7174
7175 // @route api/registrar/getrecords
7176 // @desc Get records
7177 // @access Private
7178
7179 router.get(
7180   "/getrecords",
7181   passport.authenticate("registrar", {
7182     session: false
7183   })
7184   )
7185 );
7186
7187 // @route api/registrar/getrecords
7188 // @desc Get records
7189 // @access Private
7190
7191 router.get(
7192   "/getrecords",
7193   passport.authenticate("registrar", {
7194     session: false
7195   })
7196   )
7197 );
7198
7199 // @route api/registrar/getrecords
7200 // @desc Get records
7201 // @access Private
7202
7203 router.get(
7204   "/getrecords",
7205   passport.authenticate("registrar", {
7206     session: false
7207   })
7208   )
7209 );
7210
7211 // @route api/registrar/getrecords
7212 // @desc Get records
7213 // @access Private
7214
7215 router.get(
7216   "/getrecords",
7217   passport.authenticate("registrar", {
7218     session: false
7219   })
7220   )
7221 );
7222
7223 // @route api/registrar/getrecords
7224 // @desc Get records
7225 // @access Private
7226
7227 router.get(
7228   "/getrecords",
7229   passport.authenticate("registrar", {
7230     session: false
7231   })
7232   )
7233 );
7234
7235 // @route api/registrar/getrecords
7236 // @desc Get records
7237 // @access Private
7238
7239 router.get(
7240   "/getrecords",
7241   passport.authenticate("registrar", {
7242     session: false
7243   })
7244   )
7245 );
7246
7247 // @route api/registrar/getrecords
7248 // @desc Get records
7249 // @access Private
7250
7251 router.get(
7252   "/getrecords",
7253   passport.authenticate("registrar", {
7254     session: false
7255   })
7256   )
7257 );
7258
7259 // @route api/registrar/getrecords
7260 // @desc Get records
7261 // @access Private
7262
7263 router.get(
7264   "/getrecords",
7265   passport.authenticate("registrar", {
7266     session: false
7267   })
7268   )
7269 );
7270
7271 // @route api/registrar/getrecords
7272 // @desc Get records
7273 // @access Private
7274
7275 router.get(
7276   "/getrecords",
7277   passport.authenticate("registrar", {
7278     session: false
7279   })
7280   )
7281 );
7282
7283 // @route api/registrar/getrecords
7284 // @desc Get records
7285 // @access Private
7286
7287 router.get(
7288   "/getrecords",
7289   passport.authenticate("registrar", {
7290     session: false
7291   })
7292   )
7293 );
7294
7295 // @route api/registrar/getrecords
7296 // @desc Get records
7297 // @access Private
7298
7299 router.get(
7300   "/getrecords",
7301   passport.authenticate("registrar", {
7302     session: false
7303   })
7304   )
7305 );
7306
7307 // @route api/registrar/getrecords
7308 // @desc Get records
7309 // @access Private
7310
7311 router.get(
7312   "/getrecords",
7313   passport.authenticate("registrar", {
7314     session: false
7315   })
7316   )
7317 );
7318
7319 // @route api/registrar/getrecords
7320 // @desc Get records
7321 // @access Private
7322
7323 router.get(
7324   "/getrecords",
7325   passport.authenticate("registrar", {
7326     session: false
7327   })
7328   )
7329 );
7330
7331 // @route api/registrar/getrecords
7332 // @desc Get records
7333 // @access Private
7334
7335 router.get(
7336   "/getrecords",
7337   passport.authenticate("registrar", {
7338     session: false
7339   })
7340   )
7341 );
7342
7343 // @route api/registrar/getrecords
7344 // @desc Get records
7345 // @access Private
7346
7347 router.get(
7348   "/getrecords",
7349   passport.authenticate("registrar", {
7350     session: false
7351   })
7352   )
7353 );
7354
7355 // @route api/registrar/getrecords
7356 // @desc Get records
7357 // @access Private
7358
7359 router.get(
7360   "/getrecords",
7361   passport.authenticate("registrar", {
7362     session: false
7363   })
7364   )
7365 );
7366
7367 // @route api/registrar/getrecords
7368 // @desc Get records
7369 // @access Private
7370
7371 router.get(
7372   "/getrecords",
7373   passport.authenticate("registrar", {
7374     session: false
7375   })
7376   )
7377 );
7378
7379 // @route api/registrar/getrecords
7380 // @desc Get records
7381 // @access Private
7382
7383 router.get(
7384   "/getrecords",
7385   passport.authenticate("registrar", {
7386     session: false
7387   })
7388   )
7389 );
7390
7391 // @route api/registrar/getrecords
7392 // @desc Get records
7393 // @access Private
7394
7395 router.get(
7396   "/getrecords",
7397   passport.authenticate("registrar", {
7398     session: false
7399   })
7400   )
7401 );
7402
7403 // @route api/registrar/getrecords
7404 // @desc Get records
7405 // @access Private
7406
7407 router.get(
7408   "/getrecords",
7409   passport.authenticate("registrar", {
7410     session: false
7411   })
7412   )
7413 );
7414
7415 // @route api/registrar/getrecords
7416 // @desc Get records
7417 // @access Private
7418
7419 router.get(
7420   "/getrecords",
7421   passport.authenticate("registrar", {
7422     session: false
7423   })
7424   )
7425 );
7426
7427 // @route api/registrar/getrecords
7428 // @desc Get records
7429 // @access Private
7430
7431 router.get(
7432   "/getrecords",
7433   passport.authenticate("registrar", {
7434     session: false
7435   })
7436   )
7437 );
7438
7439 // @route api/registrar/getrecords
7440 // @desc Get records
7441 // @access Private
7442
7443 router.get(
7444   "/getrecords",
7445   passport.authenticate("registrar", {
7446     session: false
7447   })
7448   )
7449 );
7450
7451 // @route api/registrar/getrecords
7452 // @desc Get records
7453 // @access Private
7454
7455 router.get(
7456   "/getrecords",
7457   passport.authenticate("registrar", {
7458     session: false
7459   })
7460   )
7461 );
7462
7463 // @route api/registrar/getrecords
7464 // @desc Get records
7465 // @access Private
7466
7467 router.get(
7468   "/getrecords",
7469   passport.authenticate("registrar", {
7470     session: false
7471   })
7472   )
7473 );
7474
7475 // @route api/registrar/getrecords
7476 // @desc Get records
7477 // @access Private
7478
7479 router.get(
7480   "/getrecords",
7481   passport.authenticate("registrar", {
7482     session: false
7483   })
7484   )
7485 );
7486
7487 // @route api/registrar/getrecords
7488 // @desc Get records
7489 // @access Private
7490
7491 router.get(
7492   "/getrecords",
7493   passport.authenticate("registrar", {
7494     session: false
7495   })
7496   )
7497 );
7498
7499 // @route api/registrar/getrecords
7500 // @desc
```

```

6202     });
6203   } else {
6204     SubjectSection.findOne({
6205       where: {
6206         classRecordID
6207       }
6208     }).then(async subjectsection => {
6209       if (subjectsection) {
6210         if (false) {
6211           return res.status(401).json({
6212             msg: "Not authorized!"
6213           });
6214         } else {
6215           const isSubmitted = await ClassRecord.findOne
6216             ({
6217               where: {
6218                 classRecordID: subjectsection.
6219                   classRecordID
6220               }
6221             }).then(cr => {
6222               if (cr) {
6223                 return cr.isSubmitted;
6224               }
6225             });
6226           let {
6227             errors,
6228             isValid
6229           } = validateSubcomponent({
6230             name: description
6231           });
6232           if (!isValid) {
6233             return res.status(400).json({
6234               msg: "Error input: Description"
6235             });
6236           } else {
6237             let i = 0;
6238             let invalid = false;
6239             let gradeIDchecker = [];
6240             for (const [index, value] of payload.
6241               entries()) {
6242               gradeIDchecker.push(value.gradeID);
6243               await Grade.findAll({
6244                 where: {
6245                   componentID,
6246                   subcomponentID: subcompID,
6247                   quarter,
6248                   subsectstudID: value.subsectstudID,
6249                   gradeID: {
6250                     [Op.ne]: value.gradeID
6251                   }
6252                 }
6253               }).then(async grades2 => {
6254                 let sum = 0;
6255                 let totalTemp = 0;
6256                 for (const [
6257                   index2,
6258                   value2
6259                 ] of grades2.entries()) {
6260                   if (value2.score == "A") {
6261                     sum = sum + 0;
6262                     totalTemp =
6263                     totalTemp + parseFloat(value2.
6264                       total);
6265                   } else if (value2.score == "E") {
6266                     else {
6267                       sum = sum + parseFloat(value2.score
6268                         );
6269                     totalTemp =
6270                     totalTemp + parseFloat(value2.
6271                       total);
6272                   }
6273                 }
6274               }
6275             }
6276           if (

```

```

6268     sum + parseFloat(value.score) >
6269     totalTemp + parseFloat(total) &&
6270     !invalid
6271   ) {
6272     // invalid = true;
6273     // return res.status(400).json({
6274     //   msg:
6275     //     "Invalid score input. Score
6276     //       must accumulate to less than or
6277     //       equal to the total number of
6278     //       items."
6279     // });
6280   });
6281   for (i = 0; i < payload.length; i++) {
6282     let {
6283       score,
6284       subsectstudID
6285     } = payload[i];
6286     let valid =
6287       (!isNaN(score) &&
6288       reg.test(score) &&
6289       score !== "" &&
6290       score !== "-") ||
6291       score == "A" ||
6292       score == "E";
6293     if (!valid) {
6294       invalid = true;
6295     }
6296     return res.status(400).json({
6297       msg: "Invalid score input. Enter
6298         numbers, E, or B only"
6299     });
6300   }
6301   await Grade.findOne({
6302     where: {
6303       description,
6304       dateGiven,
6305       componentID,
6306       subcomponentID: subcompID,
6307       classRecordID: subjectsection.
6308         classRecordID,
6309       quarter,
6310       gradeID: {
6311         [Op.notIn]: gradeIDchecker
6312       }
6313     }).then(async gr => {
6314       if (gr) {
6315         res.status(400).json({
6316           msg: "Description already exists!"
6317         });
6318       } else {
6319         if (!invalid) {
6320           const name = await utils.
6321             getAccountName(
6322               req.user.accountID
6323             );
6324           const section = await utils.
6325             getSectionName(
6326               subjectsection.sectionID
6327             );
6328           const subject = await utils.
6329             getSubjectName(
6330               subjectsection.subjectID
6331             );
6332           const logID = await ActivityLog.
6333             create({
6334               type: "UPDATE",
6335               classRecordID,
6336               position: "Registrar",

```

```

6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344
6345
6346
6347
6348
6349
6350
6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367
6368
6369
6370
6371
6372
6373
6374
6375
6376
6377
6378
6379
6380
6381
6382
6383
6384
6385
6386
6387
6388
6389
6390
6391
6392
6393
6394
6395
6396
6397
6398
6399
6400
6401
6402
6403
6404
          name ,
          section ,
          subject ,
          quarter ,
          timestamp: new Date()
        }).then(async al => {
          if (al) {
            return al.logID;
          }
        });

      let oldDesc = "";
      let oldTotal = -1;
      for (const [
        index,
        value3
      ] of payload.entries()) {
        await Grade.findOne({
          where: {
            gradeID: value3.gradeID
          }
        }).then(async gr2 => {
          if (gr2) {
            const score = gr2.score;
            let attendance = "";
            if (value3.score == "A") {
              attendance = "A";
            } else if (value3.score == "E")
              {
                attendance = "E";
              } else {
                attendance = "P";
              }
            if (index == 0) {
              oldDesc = gr2.description;
              oldTotal = gr2.total;
            }
            await gr2.update({
              description,
              total,
              dateGiven,
              attendance,
              score: value3.score,
              isUpdated: 1,
              showLog: isSubmitted
            }, {
              showLog: true,
              logID,
              subsectstudID: gr2.
                subsectstudID,
              componentID,
              subcompID,
              oldValue: gr2.score,
              newValue: value3.score,
              description
            });
          }
        });
      });
    }
  );
  await LogDetails.findAll({
    where: {
      logID
    }
  }).then(lds => {
    if (lds) {
      if (lds.length == 0) {
        ActivityLog.findOne({
          where: {
            logID
          }
        }).then(async al => {
          if (al) {
            al.destroy({}).then(() => {
              if (oldDesc !=

```

```

        description) {
ActivityLog.create({
  type: "DESC_UPDATE",
  classRecordID,
  position: "Registrar"
  ,
  name,
  section,
  subject,
  quarter,
  timestamp: new Date()
}).then(async al2 => {
  if (al2) {
    const component =
      await utils.
        getComponentName
        (
          componentID
        );
    const subcomponent
      = await utils.
        getSubcomponentName
        (
          subcompID
        );
    await LogDetails.
      create({
        logID: al2.logID,
        description:
          Changed
          description
          from '${oldDesc}' to
          '${description}',
          ,
          subcomponent,
          component
        });
  }
});

if (oldTotal != total) {
  ActivityLog.create({
    type: "TOTAL_UPDATE",
    classRecordID,
    position: "Registrar"
    ,
    name,
    section,
    subject,
    quarter,
    timestamp: new Date()
}).then(async al2 => {
  if (al2) {
    const component =
      await utils.
        getComponentName
        (
          componentID
        );
    const subcomponent
      = await utils.
        getSubcomponentName
        (
          subcompID
        );
    await LogDetails.
      create({
        logID: al2.logID,
        description:
          Changed total
          from '${oldTotal}' to

```

```

6453           ' ${total} , ,
6454           subcomponent ,
6455           component
6456           });
6457           });
6458           });
6459           });
6460           });
6461           });
6462       } else {
6463         if (oldDesc != description) {
6464           ActivityLog.create({
6465             type: "DESC_UPDATE",
6466             classRecordID,
6467             position: "Registrar",
6468             name,
6469             section,
6470             subject,
6471             quarter,
6472             timestamp: new Date()
6473           }).then(async al2 => {
6474             if (al2) {
6475               const component = await
6476                 utils.
6477                   getComponentName(
6478                     componentID
6479                   );
6480               const subcomponent =
6481                 await utils.
6482                   getSubcomponentName(
6483                     subcompID
6484                   );
6485               await LogDetails.create({
6486                 logID: al2.logID,
6487                 description: 'Changed
6488                   description from '$
6489                     {oldDesc}' to '${
6490                       description}' ,
6491                     subcomponent ,
6492                     component
6493                   });
6494             }
6495           });
6496         }
6497         if (oldTotal != total) {
6498           ActivityLog.create({
6499             type: "TOTAL_UPDATE",
6500             classRecordID,
6501             position: "Registrar",
6502             name,
6503             section,
6504             subject,
6505             quarter,
6506             timestamp: new Date()
6507           }).then(async al2 => {
6508             if (al2) {
6509               const component = await
6510                 utils.
6511                   getComponentName(
6512                     componentID
6513                   );
6514               const subcomponent =
6515                 await utils.
6516                   getSubcomponentName(
6517                     subcompID
6518                   );
6519               await LogDetails.create({
6520                 logID: al2.logID,
6521                 description: 'Changed
6522                   total from '${
6523                     oldTotal}' to '${
6524                       total}' ,

```

```

6512                     subcomponent ,
6513                     component
6514                 });
6515             }
6516         });
6517     }
6518 }
6519 });
6520
6521
6522         await utils.
6523             refreshStudentWeightedScoreBySubsectID
6524             (
6525                 subjectsection.subsectID
6526             );
6527
6528         res.status(200).json({
6529             msg: "Record updated successfully!"
6530         });
6531     }
6532 }
6533 } else {
6534     res.status(404).json({
6535         msg: "Subject section not found!"
6536     });
6537 }
6538 });
6539 }
6540 }
6541 }
6542 } else {
6543     res.status(404).json({
6544         msg: "Class record status does not exist"
6545     });
6546 }
6547 });
6548 } else {
6549     res.status(404).json({
6550         msg: "Subject section does not exist"
6551     });
6552 }
6553 });
6554 }
6555 );
6556
6557 // @route POST api/registrar/addnewrecord
6558 // @desc Add new record under subcomponent
6559 // @access Private
6560
6561 router.post(
6562     "/addnewrecord",
6563     passport.authenticate("registrar", {
6564         session: false
6565     }),
6566     async(req, res) => {
6567         const reg = /^-?[0-9]*(\.[0-9]*)?$/;
6568         const {
6569             componentID,
6570             subcompID,
6571             payload,
6572             dateGiven,
6573             description,
6574             total,
6575             classRecordID,
6576             quarter
6577         } = req.body;
6578         let totalValid = !isNaN(total) && reg.test(total) && total !== ""
6579             && total !== "-";
6580         SubjectSection.findOne({
6581             where: {
6582                 classRecordID

```

```

6582         }
6583     }).then(async ss => {
6584     if (ss) {
6585         ClassRecordStatus.findOne({
6586             where: {
6587                 classRecordID: ss.classRecordID
6588             }
6589         }).then(async crs => {
6590         if (crs) {
6591             let invalid = false;
6592             if (quarter == "Q1") {
6593                 invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 == "F";
6594             } else if (quarter == "Q2") {
6595                 invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 == "F";
6596             } else if (quarter == "Q3") {
6597                 invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 == "F";
6598             } else {
6599                 invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 == "F";
6600             }
6601             if (invalid) {
6602                 res.status(400).json({
6603                     msg: "You are not authorized to edit this class
6604                     record."
6605                 });
6606             } else {
6607                 if (!totalValid) {
6608                     res.status(400).json({
6609                         msg: "Total score is invalid. Enter numbers only."
6610                     });
6611                 } else {
6612                     if (total <= 0) {
6613                         res.status(400).json({
6614                             msg: "Total must be more than 0"
6615                         });
6616                     SubjectSection.findOne({
6617                         where: {
6618                             classRecordID
6619                         }
6620                     }).then(async subjectsection => {
6621                     if (subjectsection) {
6622                         if (false) {
6623                             return res.status(400).json({
6624                                 msg: "Not authorized!"
6625                             });
6626                         } else {
6627                             let {
6628                                 errors,
6629                                 isValid
6630                             } = validateSubcomponent({
6631                                 name: description
6632                             });
6633                             if (!isValid) {
6634                                 return res.status(400).json({
6635                                     msg: "Error input: Description"
6636                                 });
6637                             } else {
6638                                 let i = 0;
6639                                 let invalid = false;
6640                                 for (i = 0; i < payload.length; i++) {
6641                                     let {
6642                                         score,
6643                                         subsectstudID
6644                                     } = payload[i];
6645                                     let valid =
6646                                         (!isNaN(score) &&
6647                                         reg.test(score) &&
6648                                         score !== "" &&
6649                                         score !== "-") ||

```

```

6650                     score == "A" ||
6651                     score == "E";
6652             if (!valid) {
6653                 invalid = true;
6654                 return res.status(400).json({
6655                     msg: "Invalid score input. Enter
6656                         numbers, E, or B only"
6657                 });
6658             }
6659         }
6660         for (const [index, value] of payload.entries
6661             ()) {
6662             await Grade.findAll({
6663                 where: {
6664                     componentID,
6665                     subcomponentID: subcompID,
6666                     quarter,
6667                     subsectstudID: value.subsectstudID
6668                 }
6669             }).then(async grades2 => {
6670                 let sum = 0;
6671                 let totalTemp = 0;
6672                 for (const [index2, value2] of grades2.
6673                     entries()) {
6674                     if (value2.score == "A") {
6675                         sum = sum + 0;
6676                         totalTemp =
6677                             totalTemp + parseFloat(value2.total
6678                                 );
6679                     } else if (value2.score == "E") {} else
6680                         {
6681                             sum = sum + parseFloat(value2.score);
6682                             totalTemp =
6683                                 totalTemp + parseFloat(value2.total
6684                                     );
6685                         }
6686                 }
6687                 if (
6688                     sum + parseFloat(value.score) >
6689                     totalTemp + parseFloat(total) &&
6690                     !invalid
6691                 ) {
6692                     // invalid = true;
6693                     // return res.status(400).json({
6694                     //     msg:
6695                     //         "Invalid score input. Scores
6696                         must accumulate to less than or
6697                         equal to the total number of items
6698                         .
6699                     // });
6700                 }
6701             });
6702         }
6703         await Grade.findOne({
6704             where: {
6705                 description,
6706                 dateGiven,
6707                 componentID,
6708                 subcomponentID: subcompID,
6709                 classRecordID: subjectsection.
6710                     classRecordID,
6711                     quarter
6712             }
6713         }).then(async gr => {
6714             if (gr) {
6715                 res.status(400).json({
6716                     msg: "Description already exists"
6717                 });
6718             } else {
6719                 const name = await utils.getAccountName(
6720                     req.user.accountID

```

```

6713 );
6714     const section = await utils.
6715         getSectionName(
6716             subjectsection.sectionID
6717         );
6718     const subject = await utils.
6719         getSubjectName(
6720             subjectsection.subjectID
6721         );
6722     const logID = await ActivityLog.create({
6723         type: "ADD",
6724         classRecordID,
6725         position: "Registrar",
6726         name,
6727         section,
6728         subject,
6729         quarter,
6730         timestamp: new Date()
6731     }).then(async al => {
6732         if (al) {
6733             return al.logID;
6734         }
6735     });
6736     for (i = 0; i < payload.length; i++) {
6737         let {
6738             score,
6739             subsectstudID
6740         } = payload[i];
6741         let attendance = "";
6742         if (score == "A") {
6743             attendance = "A";
6744         } else if (score == "E") {
6745             attendance = "E";
6746         } else {
6747             attendance = "P";
6748         }
6749         if (!invalid) {
6750             Grade.create({
6751                 description,
6752                 dateGiven,
6753                 score,
6754                 attendance,
6755                 total,
6756                 componentID,
6757                 subcomponentID: subcompID,
6758                 date: new Date(),
6759                 classRecordID: subjectsection.
6760                     classRecordID,
6761                     subsectstudID,
6762                     showLog: 0,
6763                     isUpdated: 0,
6764                     quarter: quarter
6765             }, {
6766                 showLog: true,
6767                 logID,
6768                 subsectstudID,
6769                 componentID,
6770                 subcompID,
6771                 value: score
6772             });
6773         }
6774     }
6775     await utils.
6776         refreshStudentWeightedScoreBySubsectID
6777         (
6778             subjectsection.subsectID
6779         );
6780     return res.status(200).json({
6781         msg: "Record has been added
6782             successfully!"
6783     });
6784 });
6785 });
6786 }

```

```

6782         } else {
6783             res.status(404).json({
6784                 msg: "Subject section not found!"
6785             });
6786         }
6787     });
6788 }
6789 }
6790 } else {
6791     res.status(404).json({
6792         msg: "Class record status does not exist"
6793     });
6794 }
6795 );
6796 } else {
6797     res.status(404).json({
6798         msg: "Subject section does not exist"
6799     });
6800 }
6801 );
6802 }
6803 );
6804
6805 // @route api/registrar/changetransmutation
6806 // @desc Change transmutation by classRecordID and quarter
6807 // @access Private
6808
6809 router.post(
6810     "/changetransmutation",
6811     passport.authenticate("registrar", {
6812         session: false
6813     }),
6814     async(req, res) => {
6815         const {
6816             classRecordID,
6817             quarter,
6818             transmutation
6819         } = req.body;
6820         SubjectSection.findOne({
6821             where: {
6822                 classRecordID
6823             }
6824         }).then(async ss => {
6825             if (ss) {
6826                 if (false) {
6827                     res.status(401).json({
6828                         msg: "Not authorized!"
6829                     });
6830                 } else {
6831                     ClassRecordStatus.findOne({
6832                         where: {
6833                             classRecordID: ss.classRecordID
6834                         }
6835                     }).then(async crs => {
6836                         if (crs) {
6837                             let invalid = false;
6838                             if (quarter == "Q1") {
6839                                 invalid = crs.q1 == "L" || crs.q1 == "E" || crs.q1 ==
6840                                     "F";
6841                             } else if (quarter == "Q2") {
6842                                 invalid = crs.q2 == "L" || crs.q2 == "E" || crs.q2 ==
6843                                     "F";
6844                             } else if (quarter == "Q3") {
6845                                 invalid = crs.q3 == "L" || crs.q3 == "E" || crs.q3 ==
6846                                     "F";
6847                             } else {
6848                                 invalid = crs.q4 == "L" || crs.q4 == "E" || crs.q4 ==
6849                                     "F";
6850                             }
6851                             if (invalid) {
6852                                 res.status(400).json({
6853                                     msg: "You are not authorized to edit this class

```

```

        record."
6850    });
6851  } else {
6852    ClassRecord.findOne({
6853      where: {
6854        classRecordID: ss.classRecordID
6855      }
6856    }).then(async cr => {
6857      if (cr) {
6858        switch (quarter) {
6859          case "Q1":
6860            {
6861              await cr.update({
6862                q1Transmu: transmutation
6863              }, {
6864                showLog: true,
6865                position: "Registrar",
6866                type: "TRANSMU_UPDATE",
6867                accountID: req.user.accountID,
6868                sectionID: ss.sectionID,
6869                subjectID: ss.subjectID,
6870                quarter,
6871                classRecordID,
6872                oldVal: cr.q1Transmu,
6873                newVal: transmutation
6874              });
6875              break;
6876            }
6877          case "Q2":
6878            {
6879              await cr.update({
6880                q2Transmu: transmutation
6881              }, {
6882                showLog: true,
6883                position: "Registrar",
6884                type: "TRANSMU_UPDATE",
6885                accountID: req.user.accountID,
6886                sectionID: ss.sectionID,
6887                subjectID: ss.subjectID,
6888                quarter,
6889                classRecordID,
6890                oldVal: cr.q2Transmu,
6891                newVal: transmutation
6892              });
6893              break;
6894            }
6895          case "Q3":
6896            {
6897              await cr.update({
6898                q3Transmu: transmutation
6899              }, {
6900                showLog: true,
6901                position: "Registrar",
6902                type: "TRANSMU_UPDATE",
6903                accountID: req.user.accountID,
6904                sectionID: ss.sectionID,
6905                subjectID: ss.subjectID,
6906                quarter,
6907                classRecordID,
6908                oldVal: cr.q3Transmu,
6909                newVal: transmutation
6910              });
6911              break;
6912            }
6913          case "Q4":
6914            {
6915              await cr.update({
6916                q4Transmu: transmutation
6917              }, {
6918                showLog: true,
6919                position: "Registrar",
6920                type: "TRANSMU_UPDATE",
6921                accountID: req.user.accountID,
6922                sectionID: ss.sectionID,
6923                subjectID: ss.subjectID,
6924                quarter,

```

```

6925                               classRecordID,
6926                               oldVal: cr.q4Transmu,
6927                               newVal: transmutation
6928                           });
6929                           break;
6930                       }
6931               default:
6932                   break;
6933           }
6934           await utils.
6935               refreshStudentWeightedScoreBySubsectID(
6936                   ss.subsectID
6937               );
6938               res.status(200).json({
6939                   msg: "Transmutation changed successfully!"
6940               });
6941           } else {
6942               res.status(404).json({
6943                   msg: "Class Record not found!"
6944               });
6945           });
6946       }
6947   } else {
6948       res.status(404).json({
6949           msg: "Subject Section not found!"
6950       });
6951   }
6952   });
6953   }
6954 } else {
6955     res.status(404).json({
6956         msg: "Class record status does not exist"
6957     });
6958   }
6959   });
6960   );
6961   );
6962   );
6963 // @route POST api/registrar/getquartersummary
6964 // @desc Get quarter summary by classRecordID and quarter
6965 // @access Private
6966
6967 router.post(
6968     "/getquartersummary",
6969     passport.authenticate(directorRegistrar, {
6970         session: false
6971     }),
6972     async(req, res) => {
6973         const {
6974             classRecordID,
6975             quarter
6976         } = req.body;
6977         SubjectSection.findOne({
6978             where: {
6979                 classRecordID
6980             }
6981         }).then(async subsect => {
6982             if (subsect) {
6983                 if (false) {
6984                     res.status(401).json({
6985                         msg: "Not authorized!"
6986                     });
6987                 } else {
6988                     const transmutation = await ClassRecord.findOne({
6989                         where: {
6990                             classRecordID: subsect.classRecordID
6991                         }
6992                     }).then(async cr => {
6993                         if (cr) {
6994                             switch (quarter) {
6995                                 case "Q1":
```

```

6996
6997         {
6998             return cr.q1Transmu;
6999             break;
7000         }
7001     case "Q2":
7002         {
7003             return cr.q2Transmu;
7004             break;
7005         }
7006     case "Q3":
7007         {
7008             return cr.q3Transmu;
7009             break;
7010         }
7011     case "Q4":
7012         {
7013             return cr.q4Transmu;
7014             break;
7015         }
7016     default:
7017         break;
7018     }
7019 });
7020 const subjectName = await utils.getSubjectName(subsect.
7021     subjectID);
7022 const subjectCode = await utils.getSubjectCode(subsect.
7023     subjectID);
7024 const sectionName = await utils.getSectionName(subsect.
7025     sectionID);
7026 const schoolYearID = subsect.schoolYearID;
7027 const schoolYear = await utils.getSYname(schoolYearID);
7028 const subsectstudIDs = await SubjectSectionStudent.findAll
7029     ({
7030         where: {
7031             subsectID: subsect.subsectID
7032         }
7033     }).then(async sss => {
7034         if (sss.length == 0) {
7035             return [];
7036         } else {
7037             let data = [];
7038             for (const [index, value] of sss.entries()) {
7039                 data.push(value.subsectstudID);
7040             }
7041             return data;
7042         }
7043     });
7044 let data = [];
7045 for (const [index, value] of subsectstudIDs.entries()) {
7046     const temp = await StudentWeightedScore.findOne({
7047         where: {
7048             subsectstudID: value,
7049             quarter
7050         }
7051     }).then(async sws => {
7052         if (sws) {
7053             const {
7054                 subsectstudID,
7055                 faWS,
7056                 wwWS,
7057                 ptWS,
7058                 qeWS,
7059                 actualGrade,
7060                 transmutedGrade50,
7061                 transmutedGrade55,
7062                 transmutedGrade60
7063             } = sws;
7064             let finalGrade = 0;
7065             switch (transmutation) {
7066                 case "50":
7067                     {
7068                         finalGrade = Math.round(transmutedGrade50);
7069                     break;

```

```

7066
7067         }
7068     case "55":
7069     {
7070         finalGrade = Math.round(transmutedGrade55);
7071         break;
7072     }
7073     case "60":
7074     {
7075         finalGrade = Math.round(transmutedGrade60);
7076         break;
7077     }
7078     default:
7079         break;
7080     }
7081     const name = await utils.
7082         getStudentNameBySubsectstudID(
7083             subsectstudID
7084         );
7085     const sex = await utils.getStudentSexBySubsectstudID(
7086             subsectstudID
7087         );
7088     const imageUrl = await utils.
7089         getStudentImageUrlBySubsectstudID(
7090             subsectstudID
7091         );
7092     return {
7093         sex,
7094         name,
7095         imageUrl,
7096         subsectstudID,
7097         faWS,
7098         wwWS,
7099         ptWS,
7100         qeWS,
7101         actualGrade,
7102         transmutedGrade50,
7103         transmutedGrade55,
7104         transmutedGrade60,
7105         finalGrade
7106     };
7107 }
7108 });
7109 data.push(temp);
7110 }
7111 data.sort((a, b) => (a.name > b.name ? 1 : -1));
7112 res.status(200).json({
7113     transmutation,
7114     subjectName,
7115     subjectCode,
7116     sectionName,
7117     schoolYearID,
7118     schoolYear,
7119     classRecordID: subsect.classRecordID,
7120     data: [
7121         ...data.filter(a => a.sex === "M"),
7122         ...data.filter(a => a.sex === "F")
7123     ]
7124 });
7125 }
7126 }
7127 }
7128 );
7129 }
7130 );
7131
7132 // @route POST api/registrar/getsubmittedsubsectbysectionid
7133 // @desc Get subject section where class record status = 'D' by
7134 // @access Private
7135
7136 router.post(

```

```

7137     "/getsubmittedsubsectbysectionid",
7138     passport.authenticate("registrar", {
7139       session: false
7140     }),
7141     async(req, res) => {
7142       let = {
7143         sectionID,
7144         page,
7145         pageSize,
7146         quarter
7147       } = req.body;
7148       page = page - 1;
7149       let offset = page * pageSize;
7150       let limit = offset + pageSize;
7151       const sectionName = await utils.getSectionName(sectionID);
7152       const {
7153         schoolYearID
7154       } = await utils.getActiveSY();
7155       const classRecordIDs = await SubjectSection.findAll({
7156         where: {
7157           sectionID,
7158           schoolYearID,
7159           subjectType: {
7160             [Op.in]: quarter == "Q1" || quarter == "Q2" ?
7161               ["NON_SHS", "1ST_SEM"] : ["NON_SHS", "2ND_SEM"]
7162           }
7163         }
7164       }).then(async ss => {
7165         if (ss) {
7166           let data = [];
7167           for (const [i, v] of ss.entries()) {
7168             const crs = await ClassRecordStatus.findOne({
7169               where: {
7170                 classRecordID: v.classRecordID
7171               }
7172             });
7173             if (quarter == "Q1" || quarter == "Q2") {
7174               if (crs[quarter.toLowerCase()] == "D") {
7175                 data.push(v.classRecordID);
7176               }
7177             } else {
7178               if (quarter == "Q3") {
7179                 if (v.subjectType == "NON_SHS") {
7180                   if (crs[quarter.toLowerCase()] == "D") {
7181                     data.push(v.classRecordID);
7182                   }
7183                 } else {
7184                   if (crs["q1"] == "D") {
7185                     data.push(v.classRecordID);
7186                   }
7187                 }
7188               } else {
7189                 if (v.subjectType == "NON_SHS") {
7190                   if (crs[quarter.toLowerCase()] == "D") {
7191                     data.push(v.classRecordID);
7192                   }
7193                 } else {
7194                   if (crs["q2"] == "D") {
7195                     data.push(v.classRecordID);
7196                   }
7197                 }
7198               }
7199             }
7200           }
7201           return data;
7202         }
7203       });
7204       if (classRecordIDs.length == 0) {
7205         res.status(200).json({
7206           sectionName,
7207           classRecordList: [],
7208           numOfPages: 1,

```

```

7209     deadline: "NOT SET"
7210   });
7211 } else {
7212   let condition = {};
7213   condition["classRecordID"] = {
7214     [Op.in]: classRecordIDs
7215   };
7216   await ClassRecordStatus.findAll({
7217     limit,
7218     offset,
7219     where: condition
7220   }).then(async crs => {
7221     if (crs) {
7222       if (crs.length == 0) {
7223         res.status(404).json({
7224           msg: "No class record for deliberation found"
7225         });
7226       } else {
7227         let data2 = [];
7228         let numOfPages = 1;
7229         crs.slice(0, pageSize).forEach(async(v, k, r) => {
7230           const {
7231             classRecordID
7232           } = v;
7233           const subjectType = await utils.
7234             getSubjectTypeByClassRecordID(
7235               classRecordID
7236             );
7237           let dateSubmitted = v['${quarter.toLowerCase()}'
7238             DateSubmitted'];
7239           if (subjectType == "2ND_SEM") {
7240             if (quarter == "Q3") {
7241               dateSubmitted = v.q1DateSubmitted;
7242             } else if (quarter == "Q4") {
7243               dateSubmitted = v.q2DateSubmitted;
7244             }
7245             const {
7246               subjectCode,
7247               subjectName,
7248               section,
7249               subsectID,
7250               deadline,
7251               teacher
7252             } = await SubjectSection.findOne({
7253               where: {
7254                 classRecordID
7255               }
7256             }).then(async cr => {
7257               if (cr) {
7258                 const deadline = await SubmissionDeadline.findOne({
7259                   where: {
7260                     teacherID: cr.teacherID,
7261                     isActive: 1
7262                   }
7263                 }).then(sd => {
7264                   if (sd) {
7265                     return sd.deadline;
7266                   } else {
7267                     return "NOT SET";
7268                   }
7269                 });
7270               const subjectCode = await utils.getSubjectCode(cr.
7271                 subjectID);
7272               const {
7273                 subsectID
7274               } = cr;
7275               const teacher = await utils.getTeacherName(cr.
7276                 teacherID);
7277               const subjectName = await utils.getSubjectName(cr.
7278                 subjectID);
7279               const section = await utils.getSectionName(cr.
7280                 sectionID);

```



```

7348     }
7349   }).then(async ss => {
7350     if (ss) {
7351       let data = [];
7352       for (const [i, v] of ss.entries()) {
7353         const crs = await ClassRecordStatus.findOne({
7354           where: {
7355             classRecordID: v.classRecordID
7356           }
7357         });
7358         if (quarter == "Q1" || quarter == "Q2") {
7359           if (crs[quarter.toLowerCase()] == "E") {
7360             data.push(v.classRecordID);
7361           }
7362         } else {
7363           if (quarter == "Q3") {
7364             if (v.subjectType == "NON_SHS") {
7365               if (crs[quarter.toLowerCase()] == "E") {
7366                 data.push(v.classRecordID);
7367               }
7368             } else {
7369               if (crs["q1"] == "E") {
7370                 data.push(v.classRecordID);
7371               }
7372             }
7373           } else {
7374             if (v.subjectType == "NON_SHS") {
7375               if (crs[quarter.toLowerCase()] == "E") {
7376                 data.push(v.classRecordID);
7377               }
7378             } else {
7379               if (crs["q2"] == "E") {
7380                 data.push(v.classRecordID);
7381               }
7382             }
7383           }
7384         }
7385       }
7386       return data;
7387     }
7388   });
7389   if (classRecordIDs.length == 0) {
7390     res.status(404).json({
7391       msg: "Class Record not found!"
7392     });
7393   } else {
7394     let condition = {};
7395     condition["classRecordID"] = {
7396       [Op.in]: classRecordIDs
7397     };
7398     await ClassRecordStatus.findAll({
7399       limit,
7400       offset,
7401       where: condition
7402     }).then(async crs => {
7403       if (crs) {
7404         if (crs.length == 0) {
7405           res.status(404).json({
7406             msg: "No class record for deliberation found"
7407           });
7408         } else {
7409           let data2 = [];
7410           let numOfPages = 1;
7411           crs.slice(0, pageSize).forEach(async (v, k, r) => {
7412             const {
7413               classRecordID
7414             } = v;
7415             const dateSubmitted = v['${quarter.toLowerCase()}'
7416               DateSubmitted'];
7417             const {
7418               subjectCode,

```

```

7418     subjectName ,
7419     section ,
7420     subsectID ,
7421     deadline ,
7422     teacher
7423   } = await SubjectSection.findOne({
7424     where: {
7425       classRecordID
7426     }
7427   }).then(async cr => {
7428     if (cr) {
7429       const deadline = await SubmissionDeadline.findOne({
7430         where: {
7431           teacherID: cr.teacherID ,
7432           isActive: 1
7433         }
7434       }).then(sd => {
7435         if (sd) {
7436           return sd.deadline;
7437         } else {
7438           return "NOT SET";
7439         }
7440       });
7441       const subjectCode = await utils.getSubjectCode(cr.
7442         subjectID);
7443       const {
7444         subsectID
7445       } = cr;
7446       const subjectName = await utils.getSubjectName(cr.
7447         subjectID);
7448       const teacher = await utils.getTeacherName(cr.
7449         teacherID);
7450       const section = await utils.getSectionName(cr.
7451         sectionID);
7452       return {
7453         teacher ,
7454         subjectCode ,
7455         subsectID ,
7456         subjectName ,
7457         section ,
7458         deadline
7459       };
7460     }
7461   });
7462   data2.push({
7463     teacher ,
7464     classRecordID ,
7465     dateSubmitted ,
7466     subjectCode ,
7467     subjectName ,
7468     section ,
7469     subsectID ,
7470     deadline
7471   });
7472   if (k == r.length - 1) {
7473     numOfPages = await ClassRecordStatus.findAndCountAll
7474     ({
7475       where: condition
7476     }).then(count => {
7477       return Math.ceil(count.count / pageSize);
7478     });
7479     res.status(200).json({
7480       sectionName ,
7481       classRecordList: data2 ,
7482       numOfPages ,
7483       deadline
7484     });
7485   });
7486 }

```

```

7487 );
7488 // @route POST api/registrar/studentdeliberationrecord
7489 // @desc Get deliberation grade of student by studsectID
7490 // @access Private
7491
7492 router.post(
7493   "/studentdeliberationrecord",
7494   passport.authenticate("registrar", {
7495     session: false
7496   }),
7497   async(req, res) => {
7498     const {
7499       studsectID
7500     } = req.body;
7501     const {
7502       schoolYearID,
7503       quarter
7504     } = await utils.getActiveSY();
7505     const data = await SubjectSectionStudent.findAll({
7506       where: {
7507         studsectID
7508       }
7509     }).then(async sss2 => {
7510       if (sss2) {
7511         let data2 = [];
7512         for (const [i, v] of sss2.entries()) {
7513           let {
7514             subjectType,
7515             classRecordID,
7516             subjectID
7517           } = await SubjectSection.findOne({
7518             where: {
7519               subsectID: v.subsectID,
7520               schoolYearID
7521             }
7522           })
7523         }).then(async ss => {
7524           if (ss) {
7525             return {
7526               subjectType: ss.subjectType,
7527               classRecordID: ss.classRecordID,
7528               subjectID: ss.subjectID
7529             };
7530           }
7531         });
7532       let status = await ClassRecordStatus.findOne({
7533         where: {
7534           classRecordID
7535         }
7536       }).then(async crs => {
7537         if (crs) {
7538           if (quarter == "Q1") {
7539             if (subjectType == "NON_SHS" || subjectType == "1
7540               ST_SEM") {
7541                 return crs.q1;
7542               }
7543             } else if (quarter == "Q2") {
7544               if (subjectType == "NON_SHS" || subjectType == "1
7545               ST_SEM") {
7546                 return crs.q2;
7547               }
7548             } else if (quarter == "Q3") {
7549               if (subjectType == "NON_SHS") {
7550                 return crs.q3;
7551               } else if (subjectType == "2ND_SEM") {
7552                 return crs.q1;
7553               }
7554             } else {
7555               if (subjectType == "NON_SHS") {
7556                 return crs.q4;
7557               } else if (subjectType == "2ND_SEM") {
7558                 return crs.q2;
7559               }
7560             }
7561           }
7562         }
7563       }
7564     }
7565   }
7566 );
7567 
```

```

7558         }
7559     }
7560   }
7561 });
7562 let subjectName = await utils.getSubjectName(subjectID);
7563 if (typeof status !== "undefined") {
7564   let score = await StudentSubjectGrades.findOne({
7565     where: {
7566       classRecordID,
7567       subsectstudID: v.subsectstudID
7568     }
7569   }).then(ssg => {
7570     if (status == "F" || status == "D") {
7571       if (quarter == "Q1") {
7572         if (subjectType == "NON_SHS" || subjectType == "1
7573           ST_SEM") {
7574           return ssg.q1FinalGrade;
7575         }
7576       } else if (quarter == "Q2") {
7577         if (subjectType == "NON_SHS" || subjectType == "1
7578           ST_SEM") {
7579           return ssg.q2FinalGrade;
7580         }
7581       } else if (quarter == "Q3") {
7582         if (subjectType == "NON_SHS") {
7583           return ssg.q3FinalGrade;
7584         } else if (subjectType == "2ND_SEM") {
7585           return ssg.q1FinalGrade;
7586         }
7587       } else {
7588         if (subjectType == "NON_SHS") {
7589           return ssg.q4FinalGrade;
7590         } else if (subjectType == "2ND_SEM") {
7591           return ssg.q2FinalGrade;
7592         }
7593       }
7594     })
7595     if (typeof score !== "undefined") {
7596       data2.push({
7597         score,
7598         subjectName
7599       });
7600     }
7601   }
7602   return data2;
7603 }
7604 });
7605 res.status(200).json({
7606   data
7607 });
7608 }
7609 );
7610
7611 // @route POST api/registrar/studentfinalgrade
7612 // @desc Get student final grade of student by studentID ,
7613 // schoolYearID
7614 // @access Private
7615 router.post(
7616   "/studentfinalgrade",
7617   passport.authenticate(directorRegistrar , {
7618     session: false
7619   }),
7620   async(req, res) => {
7621     const {
7622       studentID ,
7623       schoolYearID
7624     } = req.body;
7625     const stsectID = await StudentSection.findOne({
7626       where: {

```

```

7627         studentID ,
7628         schoolYearID
7629     }
7630   }).then(std => {
7631     if (std) {
7632       return std.studsectID;
7633     } else return -1;
7634   });
7635   if (studsectID == -1) {
7636     res.status(200).json({
7637       finalGrade: -1,
7638       data: []
7639     });
7640   } else {
7641     let finalGrade = await StudentFinalGrade.findOne({
7642       where: {
7643         studsectID,
7644         schoolYearID
7645       }
7646     }).then(sfg => {
7647       if (sfg) {
7648         return sfg.grade;
7649       }
7650     });
7651     let data = await StudentGrades.findAll({
7652       where: {
7653         studsectID,
7654         schoolYearID
7655       }
7656     }).then(sg => {
7657       if (sg) {
7658         let data2 = [];
7659         for (const [index, value] of sg.entries()) {
7660           data2.push({
7661             quarter: value.quarter,
7662             grade: value.grade
7663           });
7664         }
7665         return data2;
7666       }
7667     });
7668     data.sort((a, b) => (a.quarter > b.quarter ? 1 : -1));
7669     res.status(200).json({
7670       finalGrade,
7671       data
7672     });
7673   }
7674 }
7675 );
7676
7677 // @route POST api/registrar/studentfinalrecord
7678 // @desc Get finalgrade grade of student by studentID , schoolYearID ,
7679 // @access Private
7680
7681 router.post(
7682   "/studentfinalrecord",
7683   passport.authenticate(directorRegistrar, {
7684     session: false
7685   }),
7686   async(req, res) => {
7687     const {
7688       studentID
7689     } = req.body;
7690     const {
7691       schoolYearID,
7692       quarter
7693     } = req.body;
7694     const studsectID = await StudentSection.findOne({
7695       where: {
7696         studentID,
7697         schoolYearID
7698       }

```

```

7699     }).then(std => {
7700       if (std) {
7701         return std.studsectID;
7702       } else {
7703         return -1;
7704       }
7705     });
7706     if (studsectID == -1) {
7707       res.status(200).json({
7708         data: [],
7709         finalGrade: -1
7710       });
7711     } else {
7712       const data = await SubjectSectionStudent.findAll({
7713         where: {
7714           studsectID
7715         }
7716       }).then(async sss2 => {
7717         if (sss2) {
7718           let data2 = [];
7719           for (const [i, v] of sss2.entries()) {
7720             let {
7721               subjectType,
7722               classRecordID,
7723               subjectID
7724             } = await SubjectSection.findOne({
7725               where: {
7726                 subsectID: v.subsectID,
7727                 schoolYearID
7728               }
7729             }).then(async ss => {
7730               if (ss) {
7731                 return {
7732                   subjectType: ss.subjectType,
7733                   classRecordID: ss.classRecordID,
7734                   subjectID: ss.subjectID
7735                 };
7736               }
7737             });
7738             let status = await ClassRecordStatus.findOne({
7739               where: {
7740                 classRecordID
7741               }
7742             }).then(async crs => {
7743               if (crs) {
7744                 if (quarter == "Q1") {
7745                   if (subjectType == "NON_SHS" || subjectType == "1
7746                     ST_SEM") {
7747                     return crs.q1;
7748                   }
7749                 } else if (quarter == "Q2") {
7750                   if (subjectType == "NON_SHS" || subjectType == "1
7751                     ST_SEM") {
7752                     return crs.q2;
7753                   }
7754                 } else if (quarter == "Q3") {
7755                   if (subjectType == "NON_SHS") {
7756                     return crs.q3;
7757                   } else if (subjectType == "2ND_SEM") {
7758                     return crs.q1;
7759                   }
7760                 } else {
7761                   if (subjectType == "NON_SHS") {
7762                     return crs.q4;
7763                   } else if (subjectType == "2ND_SEM") {
7764                     return crs.q2;
7765                   }
7766                 }
7767               });
7768             let subjectName = await utils.getSubjectName(subjectID);

```

```

7769         if (typeof status !== "undefined") {
7770             let score = await StudentSubjectGrades.findOne({
7771                 where: {
7772                     classRecordID,
7773                     subsectstudID: v.subsectstudID
7774                 }
7775             }).then(ssg => {
7776                 if (status == "F") {
7777                     if (quarter == "Q1") {
7778                         if (subjectType == "NON_SHS" || subjectType == "1
7779                             ST_SEM") {
7780                             return ssg.q1FinalGrade;
7781                         }
7782                     } else if (quarter == "Q2") {
7783                         if (subjectType == "NON_SHS" || subjectType == "1
7784                             ST_SEM") {
7785                             return ssg.q2FinalGrade;
7786                         }
7787                     } else if (quarter == "Q3") {
7788                         if (subjectType == "NON_SHS") {
7789                             return ssg.q3FinalGrade;
7790                         } else if (subjectType == "2ND_SEM") {
7791                             return ssg.q1FinalGrade;
7792                         }
7793                     } else {
7794                         if (subjectType == "NON_SHS") {
7795                             return ssg.q4FinalGrade;
7796                         } else if (subjectType == "2ND_SEM") {
7797                             return ssg.q2FinalGrade;
7798                         }
7799                     });
7800                 if (typeof score !== "undefined") {
7801                     data2.push({
7802                         score,
7803                         subjectName
7804                     });
7805                 }
7806             }
7807         }
7808     }
7809     return data2;
7810 );
7811     let finalGrade = await StudentGrades.findOne({
7812         where: {
7813             studsectID,
7814             quarter
7815         }
7816     }).then(sg => {
7817         if (sg) {
7818             return sg.grade;
7819         }
7820     });
7821     res.status(200).json({
7822         data,
7823         finalGrade
7824     });
7825 }
7826 }
7827 );
7828
7829 // @router POST api/registrar/getsectionstudent
7830 // @desc Get student section by studentID , schoolYearID
7831 // @access Private
7832
7833 router.post(
7834     "/getsectionstudent",
7835     passport.authenticate(directorRegistrar , {
7836         session: false
7837     }),
7838     async(req, res) => {

```

```

7839         const {
7840             studentID,
7841             schoolYearID
7842         } = req.body;
7843         const section = await StudentSection.findOne({
7844             where: [
7845                 studentID,
7846                 schoolYearID
7847             ]
7848         }).then(async ss => await utils.getSectionName(ss.sectionID));
7849         res.status(200).json({
7850             sectionName: section
7851         });
7852     }
7853 );
7854
7855 // @router POST api/registrar/reportcardstudent
7856 // @desc Generate report card pdf per student
7857 // @access Private
7858
7859 router.post(
7860     "/reportcardstudent",
7861     passport.authenticate("registrar", {
7862         session: false
7863     }),
7864     async(req, res) => {
7865         let doc = new PDFDocument();
7866         const {
7867             studentID,
7868             schoolYearID,
7869             quarter
7870         } = req.body;
7871         let filename = encodeURIComponent("final_grade") + ".pdf";
7872         res.setHeader(
7873             "Content-disposition",
7874             'attachment; filename="' + filename + "'"
7875         );
7876         res.setHeader("Content-type", "application/pdf");
7877         doc.pipe(res);
7878         await utils.generateReportCardStudent(
7879             doc,
7880             res,
7881             studentID,
7882             schoolYearID,
7883             quarter,
7884             true
7885         );
7886     }
7887 );
7888
7889 // @route POST api/registrar/reportcardsection
7890 // @desc Generate report card pdf per section
7891 // @access Private
7892
7893 router.post(
7894     "/reportcardsection",
7895     passport.authenticate("registrar", {
7896         session: false
7897     }),
7898     async(req, res) => {
7899         let doc = new PDFDocument();
7900         let headerSent = false
7901         const {
7902             studentIDs,
7903             schoolYearID,
7904             quarter
7905         } = req.body;
7906         let filename = encodeURIComponent("final_grade") + ".pdf";
7907         res.setHeader(
7908             "Content-disposition",
7909             'attachment; filename="' + filename + "'"
7910         );
7911         res.setHeader("Content-type", "application/pdf");

```

```

7912     doc.pipe(res);
7913     for (const [index, value] of studentIDs.entries()) {
7914       if (!headerSent) {
7915         await utils.generateReportCardStudent(
7916           doc,
7917           res,
7918           value,
7919           schoolYearID,
7920           quarter,
7921           index + 1 == studentIDs.length,
7922           headerSent
7923         );
7924       }
7925     }
7926   );
7927 )
7928
7929 // @route POST api/registrar/getpassedfailed
7930 // @desc Get number of passed/failed students by schoolYearID,
7931 // quarter, and gradeLevel
7932 // @access Private
7933 router.post('/getpassedfailed', passport.authenticate(
7934   directorRegistrar, {
7935     session: false
7936   }), async(req, res) => {
7937   const {
7938     schoolYearID,
7939     quarter,
7940     gradeLevel
7941   } = req.body
7942   Section.findAll({
7943     where: {
7944       gradeLevel
7945     }
7946   }).then(async section => {
7947     if (section) {
7948       let data = []
7949       for (const [index, value] of section.entries()) {
7950         let name = await utils.getSectionName(value.sectionID)
7951         let sectionID = value.sectionID
7952         let failedInfo = []
7953         const {
7954           numOfPassed,
7955           numOfFailed
7956         } = await StudentSection.findAll({
7957           where: {
7958             sectionID: value.sectionID,
7959             schoolYearID
7960           }
7961         }).then(async ss => {
7962           if (ss) {
7963             let numOfPassed = 0
7964             let numOfFailed = 0
7965             for (const [index2, value2] of ss.entries()) {
7966               const passed = await SubjectSectionStudent.findAll({
7967                 where: {
7968                   studsectID: value2.studsectID
7969                 }
7970               }).then(async sss => {
7971                 if (sss) {
7972                   let passed2 = 0;
7973                   for (const [index3, value3] of sss.entries()) {
7974                     passed2 = await StudentSubjectGrades.findAll({
7975                       where: {
7976                         subsectstudID: value3.subsectstudID
7977                       }
7978                     }).then(async ssg => {
7979                       if (ssg) {
7980                         let passed4 = 0
7981                         for (const [index4, value4] of ssg.entries())
7982                         {
7983                           let subjectType = await utils.

```

```

        getSubjectTypeByClassRecordID(value4.
    classRecordID)
7982    if (subjectType == 'NON_SHS') {
7983        let status = await ClassRecordStatus.
            findOne({
7984            where: {
7985                classRecordID: value4.classRecordID
7986            }
7987        }).then(crs => crs['${quarter.toLowerCase()
    ()}'])
7988        if (status == 'D' || status == 'F') {
7989            if (parseFloat(value4['${quarter.
    toLowerCase()}FinalGrade']) < 75) {
7990                let name = await utils.
                    getStudentNameBySubsectstudID(
                        value4.subsectstudID)
7991                let classRecordID = value4.
                    classRecordID
7992                let grade = parseFloat(value4['${
        quarter.toLowerCase()}FinalGrade
    '])
7993                let subjectID = await utils.
                    getSubjectIDBySubsectstudID(
                        value4.subsectstudID)
7994                let subjectName = await utils.
                    getSubjectName(subjectID)
7995                let teacher = await
                    SubjectSectionStudent.findOne({
7996                    where: {
7997                        subsectstudID: value4.
                            subsectstudID
7998                    }
7999                }).then(async subsectstud => await
                    SubjectSection.findOne({
7999                    where: {
8000                        subsectID: subsectstud.subsectID
8001                    }
8002                })).then(async ss2 => await utils.
                    getTeacherName(ss2.teacherID))
8003                failedInfo.push({
8004                    name,
8005                    classRecordID,
8006                    grade,
8007                    subjectName,
8008                    teacher
8009                })
8010                passed4 = -1
8011            } else {
8012                passed4 = 1
8013            }
8014        }
8015    }
8016    } else if (subjectType == '1ST_SEM') {
8017        if (quarter == 'Q1' || quarter == 'Q2') {
8018            let status = await ClassRecordStatus.
                findOne({
8019                where: {
8020                    classRecordID: value4.classRecordID
8021                }
8022            }).then(crs => crs['${quarter.
                toLowerCase()}'])
8023            if (status == 'D' || status == 'F') {
8024                if (parseFloat(value4['${quarter.
                    toLowerCase()}FinalGrade']) < 75)
                    {
8025                    let name = await utils.
                        getStudentNameBySubsectstudID(
                            value4.subsectstudID)
8026                    let classRecordID = value4.
                        classRecordID
8027                    let grade = parseFloat(value4['${
                        quarter.toLowerCase()}'
                    FinalGrade'])
8028                    let subjectID = await utils.

```

```

8029           getSubjectIDBySubsectstudID(
8029             value4.subsectstudID)
8030           let subjectName = await utils.
8030             getSubjectName(subjectID)
8031           let teacher = await
8031             SubjectSectionStudent.findOne({
8032               where: {
8032                 subsectstudID: value4.
8032                   subsectstudID
8033               }
8034             }).then(async subsectstud => await
8034               SubjectSection.findOne({
8035                 where: {
8035                   subsectID: subsectstud.
8035                     subsectID
8036               }
8037             }).then(async ss2 => await utils.
8037               getTeacherName(ss2.teacherID))
8038             failedInfo.push({
8039               name,
8040               classRecordID,
8041               grade,
8042               subjectName,
8043               teacher
8044             })
8045             passed4 = -1
8046           } else {
8047             passed4 = 1
8048           }
8049         }
8050       }
8051     }
8052   } else {
8053     if (quarter == 'Q3' || quarter == 'Q4') {
8054       let status = await ClassRecordStatus.
8054         findOne({
8055           where: {
8055             classRecordID: value4.classRecordID
8056           }
8057         }).then(crs => crs['q${parseInt(quarter
8057           .charAt(1))-2}'])
8058       if (status == 'D' || status == 'F') {
8059         if (parseFloat(value4['q${parseInt(
8060           quarter.charAt(1))-2}FinalGrade
8060             ']]) < 75) {
8061           let name = await utils.
8061             getStudentNameBySubsectstudID(
8061               value4.subsectstudID)
8062           let classRecordID = value4.
8062             classRecordID
8063           let grade = parseFloat(value4['q${
8063             parseInt(quarter.charAt(1))-2}
8063               FinalGrade']])
8064           let subjectID = await utils.
8064             getSubjectIDBySubsectstudID(
8064               value4.subsectstudID)
8065           let subjectName = await utils.
8065             getSubjectName(subjectID)
8066           let teacher = await
8066             SubjectSectionStudent.findOne({
8067               where: {
8067                 subsectstudID: value4.
8067                   subsectstudID
8068               }
8069             }).then(async subsectstud => await
8069               SubjectSection.findOne({
8070                 where: {
8070                   subsectID: subsectstud.
8070                     subsectID
8071               }
8072             }).then(async ss2 => await utils.
8072               getTeacherName(ss2.teacherID))
8073             failedInfo.push({
8073               name,
8074             })
8075           }
8076         }

```

```

8077         classRecordID ,
8078         grade ,
8079         subjectName ,
8080         teacher
8081     })
8082     passed4 = -1
8083 } else {
8084     passed4 = 1
8085 }
8086 }
8087 }
8088 }
8089 }
8090     return passed4
8091 }
8092 }
8093 }
8094     return passed2
8095 }
8096 }
8097 if (passed == 1) {
8098     numOfPassed = numOfPassed + 1
8099 } else if (passed == -1) {
8100     numOfFailed = numOfFailed + 1
8101 }
8102 }
8103     return {
8104         numOfFailed ,
8105         numOfPassed
8106     }
8107 }
8108 }
8109     data.push({
8110         failedInfo ,
8111         numOfPassed ,
8112         numOfFailed ,
8113         name ,
8114         sectionID
8115     })
8116 }
8117     res.status(200).json({
8118         data ,
8119         failed: data.reduce((a, b) => [...a, b.numOfFailed], []).reduce((a, b) => a + parseInt(b), 0)
8120     })
8121 }
8122 }
8123 }
8124 }
8125 // @route POST api/registrar/gethonorstudents
8126 // @desc Get number of honor students by schoolYearID , and gradeLevel
8127 // @access Private
8128
8129 router.post('/gethonorstudents', passport.authenticate(
8130     directorRegistrar , {
8131         session: false
8132     }), async(req, res) => {
8133     const {
8134         schoolYearID ,
8135         gradeLevel
8136     } = req.body
8137     Section.findAll({
8138         where: {
8139             gradeLevel
8140         }
8141     }).then(async section => {
8142         if (section) {
8143             let data = []
8144             for (const [index, value] of section.entries()) {
8145                 let name = await utils.getSectionName(value.sectionID)
8146                 let sectionID = value.sectionID
8147                 let honorInfo = []
8148                 const numOfHonors = await StudentSection.findAll({

```

```

8148     where: {
8149       sectionID: value.sectionID,
8150       schoolYearID
8151     }
8152   }).then(async ss => {
8153     if (ss) {
8154       let numOfHonors = 0
8155       for (const [index2, value2] of ss.entries()) {
8156         const isHonors = await StudentFinalGrade.findOne({
8157           where: {
8158             studsectID: value2.studsectID,
8159             schoolYearID
8160           }
8161         }).then(async sfg => {
8162           if (sfg) {
8163             if (parseFloat(sfg.grade) >= 90) {
8164               let name = await utils.getStudentNameByStudsectID
8165               (value2.studsectID)
8166               let grade = sfg.grade
8167               honorInfo.push({
8168                 name,
8169                 grade,
8170                 studentID: ss.studentID
8171               })
8172             return true
8173           } else {
8174             return false
8175           }
8176         })
8177         if (isHonors) {
8178           numOfHonors = numOfHonors + 1
8179         }
8180       }
8181       return numOfHonors
8182     }
8183   })
8184   data.push({
8185     name,
8186     sectionID,
8187     numOfHonors,
8188     honorInfo
8189   })
8190 }
8191 res.status(200).json({
8192   data,
8193   numOfHonors: data.reduce((a, b) => [...a, b.numOfHonors], [])
8194     .reduce((a, b) => a + parseInt(b), 0)
8195   })
8196 })
8197 })
8198 module.exports = router;

```

Listing 34: student.js

```
1 const express = require("express");
2 const PDFDocument = require("pdfkit");
3 const router = express.Router();
4 const UserAccount = require("../models/UserAccount");
5 const passport = require("passport");
6 const Sequelize = require("sequelize");
7 const Teacher = require("../models/Teacher");
8 const SubmissionDeadline = require("../models/SubmissionDeadline")
9     ;
10 const Section = require("../models/Section");
11 const StudentSection = require("../models/StudentSection");
12 const SubjectSection = require("../models/SubjectSection");
13 const ClassRecord = require("../models/ClassRecord");
14 const SubjectSectionStudent = require("../models/
15     SubjectSectionStudent");
16 const StudentWeightedScore = require("../models/
17     StudentWeightedScore");
18 const Student = require("../models/Student");
19 const TeacherSection = require("../models/TeacherSection");
20 const Subject = require("../models/Subject");
21 const Component = require("../models/Component");
22 const Subcomponent = require("../models/Subcomponent");
23 const StudentSubjectGrades = require("../models/
24     StudentSubjectGrades");
25 const ParentGuardian = require("../models/ParentGuardian");
26 const SchoolYear = require("../models/SchoolYear");
27 const Grade = require("../models/Grade");
28 const ClassRecordStatus = require("../models/ClassRecordStatus");
29 const ActivityLog = require("../models/ActivityLog");
30 const LogDetails = require("../models/LogDetails");
31 const StudentGrades = require("../models/StudentGrades");
32 const StudentFinalGrade = require("../models/StudentFinalGrade");
33 const Op = Sequelize.Op;
34
35 // Input Validation
36 const validateEditProfileParent = require("../validation/
37     editprofileparent");
38
39 // Import Utility Functions
40 const utils = require("../utils");
41
42 // @router POST api/student/updateprofile
43 // @desc Update profile
44 // @access Private
45
46 router.post(
47     "/updateprofile",
48     passport.authenticate("student", {
49         session: false
50     }),
51     (req, res) => {
52         const {
53             firstName,
54             lastName,
55             middleName,
56             suffix,
57             nickname,
58             contactNum,
59             address,
60             province,
61             city,
62             region,
63             zipcode,
64             civilStatus,
65             sex,
66             citizenship,
67             birthDate,
68             birthPlace,
69             religion,
```

```

65         emergencyName ,
66         emergencyAddress ,
67         emergencyTelephone ,
68         emergencyCellphone ,
69         emergencyEmail ,
70         emergencyRelationship
71     } = req.body;
72
73     const {
74         errors,
75         isValid
76     } = validateEditProfileParent(req.body);
77
78     if (!isValid) {
79         return res.status(400).json(errors);
80     }
81
82     UserAccount.findOne({
83         where: {
84             accountID: req.user.accountID
85         }
86     }).then(user => {
87         if (user) {
88             user
89                 .update({
90                     firstName ,
91                     lastName ,
92                     middleName ,
93                     suffix ,
94                     nickname ,
95                     contactNum ,
96                     address ,
97                     province ,
98                     city ,
99                     region ,
100                    zipcode ,
101                    civilStatus ,
102                    sex ,
103                    citizenship ,
104                    birthDate ,
105                    birthPlace ,
106                    religion ,
107                    emergencyName ,
108                    emergencyAddress ,
109                    emergencyTelephone ,
110                    emergencyCellphone ,
111                    emergencyEmail ,
112                    emergencyRelationship
113                })
114                 .then(user2 => {
115                     res.status(200).json({
116                         msg: "Profile updated successfully!"
117                     });
118                 });
119             }
120         });
121     });
122 );
123
124 // @route GET api/student/getsy
125 // @desc Get current school year
126 // @access Private
127 router.get(
128     "/getsy",
129     passport.authenticate(["student", "guardian"], {
130         session: false
131     }),
132     async(req, res) => {
133         let {
134             schoolYearID,
135             quarter
136         } = await utils.getActiveSY();
137         if (schoolYearID != 0) {
138             let schoolYear = await utils.getSYname(schoolYearID);
139             res.status(200).json({

```

```

140             schoolYear,
141             schoolYearID,
142             quarter
143         });
144     } else {
145         res.status(404).json({
146             msg: "There is no active school year"
147         });
148     }
149 }
150 );
151
152 // @route GET api/student/getallsy
153 // @desc Get all school year
154 // @access Private
155
156 router.get(
157     "/getallsy",
158     passport.authenticate(["student", "guardian"], {
159         session: false
160     }),
161     async(req, res) => {
162         SchoolYear.findAll().then(async sys => {
163             if (sys) {
164                 let data = [];
165                 for (const [index, value] of sys.entries()) {
166                     data.push({
167                         schoolYearID: value.schoolYearID,
168                         schoolYear: value.schoolYear
169                     });
170                 }
171                 data.sort((a, b) => (a.schoolYear < b.schoolYear ? 1 : -1));
172                 res.status(200).json({
173                     schoolYearList: data
174                 });
175             }
176         });
177     }
178 );
179
180 // @route POST api/student/studentfinalrecord
181 // @desc Get finalgrade grade of student by studentID, schoolYearID,
182 // quarter
183 // @access Private
184
185 router.post(
186     "/studentfinalrecord",
187     passport.authenticate(["student", "guardian"], {
188         session: false
189     }),
190     async(req, res) => {
191         if (req.body.parent == true) {
192             const {
193                 accountID
194             } = req.user;
195             const accountIDs = await ParentGuardian.findOne({
196                 where: {
197                     accountID
198                 }
199             }).then(pg => {
200                 if (pg) {
201                     return JSON.parse(pg.studentIDs);
202                 }
203             });
204             const {
205                 schoolYearID,
206                 quarter
207             } = req.body;
208             let outData = [];
209             for (const [index, value] of accountIDs.entries()) {
210                 const studentID = await utils.getStudentID(value);
211                 const name = await utils.getStudentName(studentID)

```

```

211     let section, gradeLevel;
212     const teacher = await StudentSection.findOne({
213       where: {
214         studentID,
215         schoolYearID
216       }
217     }).then(async ss => {
218       return await TeacherSection.findOne({
219         where: {
220           sectionID: ss.sectionID,
221           schoolYearID
222         }
223       }).then(async ts => {
224         if (ts) {
225           section = await utils.getSectionName(ss.sectionID);
226           gradeLevel = await utils.getGradeLevelBySectionID(ss.
227             sectionID);
228           return await utils.getTeacherName(ts.teacherID);
229         } else {
230           section = await utils.getSectionName(ss.sectionID);
231           gradeLevel = await utils.getGradeLevelBySectionID(ss.
232             sectionID);
233           return "No Adviser";
234         }
235       });
236       const studsectID = await StudentSection.findOne({
237         where: {
238           studentID,
239           schoolYearID
240         }
241       }).then(std => {
242         if (std) {
243           return std.studsectID;
244         } else {
245           return -1;
246         }
247       });
248       if (studsectID == -1) {
249         res.status(200).json({
250           data
251         });
252       } else {
253         const data = await SubjectSectionStudent.findAll({
254           where: {
255             studsectID
256           }
257         }).then(async sss2 => {
258           if (sss2) {
259             let data2 = [];
260             for (const [i, v] of sss2.entries()) {
261               let {
262                 subjectType,
263                 classRecordID,
264                 subjectID
265               } = await SubjectSection.findOne({
266                 where: {
267                   subsectID: v.subsectID,
268                   schoolYearID
269                 }
270               }).then(async ss => {
271                 if (ss) {
272                   return {
273                     subjectType: ss.subjectType,
274                     classRecordID: ss.classRecordID,
275                     subjectID: ss.subjectID
276                   };
277                 }
278               });
279             let status = await ClassRecordStatus.findOne({
280               where: {
281                 classRecordID

```

```

282     }
283   }).then(async crs => {
284     if (crs) {
285       if (quarter == "Q1") {
286         if (
287           subjectType == "NON_SHS" ||
288           subjectType == "1ST_SEM"
289         ) {
290           return crs.q1;
291         }
292       } else if (quarter == "Q2") {
293         if (
294           subjectType == "NON_SHS" ||
295           subjectType == "1ST_SEM"
296         ) {
297           return crs.q2;
298         }
299       } else if (quarter == "Q3") {
300         if (subjectType == "NON_SHS") {
301           return crs.q3;
302         } else if (subjectType == "2ND_SEM") {
303           return crs.q1;
304         }
305       } else {
306         if (subjectType == "NON_SHS") {
307           return crs.q4;
308         } else if (subjectType == "2ND_SEM") {
309           return crs.q2;
310         }
311       }
312     }
313   });
314   let subjectName = await utils.getSubjectName(
315     subjectID);
316   if (typeof status !== "undefined") {
317     let score = await StudentSubjectGrades.findOne({
318       where: {
319         classRecordID,
320         subsectstudID: v.subsectstudID
321       }
322     }).then(ssg => {
323       if (status == "F") {
324         if (quarter == "Q1") {
325           if (
326             subjectType == "NON_SHS" ||
327             subjectType == "1ST_SEM"
328           ) {
329             return ssg.q1FinalGrade;
330           }
331         } else if (quarter == "Q2") {
332           if (
333             subjectType == "NON_SHS" ||
334             subjectType == "1ST_SEM"
335           ) {
336             return ssg.q2FinalGrade;
337           }
338         } else if (quarter == "Q3") {
339           if (subjectType == "NON_SHS") {
340             return ssg.q3FinalGrade;
341           } else if (subjectType == "2ND_SEM") {
342             return ssg.q1FinalGrade;
343           }
344         } else {
345           if (subjectType == "NON_SHS") {
346             return ssg.q4FinalGrade;
347           } else if (subjectType == "2ND_SEM") {
348             return ssg.q2FinalGrade;
349           }
350         }
351       });
352       if (typeof score !== "undefined") {

```

```

353         data2.push({
354             score,
355             subjectName
356         });
357     }
358 }
359 }
360 return data2;
361 }
362 });
363 let finalGrade = await StudentGrades.findOne({
364     where: {
365         studsectID,
366         quarter
367     }
368 }).then(sg => {
369     if (sg) {
370         return sg.grade;
371     }
372 });
373 outData.push({
374     name,
375     data,
376     finalGrade,
377     section,
378     teacher,
379     gradeLevel
380 });
381 }
382 }
383 res.status(200).json({
384     data: outData
385 });
386 } else {
387     const {
388         accountID
389     } = req.user;
390     const studentID = await utils.getStudentID(accountID);
391     const {
392         schoolYearID,
393         quarter
394     } = req.body;
395     let section, gradeLevel;
396     const teacher = await StudentSection.findOne({
397         where: {
398             studentID,
399             schoolYearID
400         }
401     }).then(async ss => {
402         return await TeacherSection.findOne({
403             where: {
404                 sectionID: ss.sectionID,
405                 schoolYearID
406             }
407         }).then(async ts => {
408             if (ts) {
409                 section = await utils.getSectionName(ss.sectionID);
410                 gradeLevel = await utils.getGradeLevelBySectionID(ss.
411                     sectionID);
412                 return await utils.getTeacherName(ts.teacherID);
413             } else {
414                 section = await utils.getSectionName(ss.sectionID);
415                 gradeLevel = await utils.getGradeLevelBySectionID(ss.
416                     sectionID);
417                 return "No Adviser";
418             }
419         });
420         const studsectID = await StudentSection.findOne({
421             where: {
422                 studentID,
423                 schoolYearID
424             }
425         });

```

```

424     }).then(std => {
425       if (std) {
426         return std.studsectID;
427       } else {
428         return -1;
429       }
430     });
431     if (studsectID == -1) {
432       res.status(200).json({
433         data: [],
434         finalGrade: -1
435       });
436     } else {
437       const data = await SubjectSectionStudent.findAll({
438         where: {
439           studsectID
440         }
441       }).then(async sss2 => {
442         if (sss2) {
443           let data2 = [];
444           for (const [i, v] of sss2.entries()) {
445             let {
446               subjectType,
447               classRecordID,
448               subjectID
449             } = await SubjectSection.findOne({
450               where: {
451                 subsectID: v.subsectID,
452                 schoolYearID
453               }
454             }).then(async ss => {
455               if (ss) {
456                 return {
457                   subjectType: ss.subjectType,
458                   classRecordID: ss.classRecordID,
459                   subjectID: ss.subjectID
460                 };
461               }
462             });
463
464             let status = await ClassRecordStatus.findOne({
465               where: {
466                 classRecordID
467               }
468             }).then(async crs => {
469               if (crs) {
470                 if (quarter == "Q1") {
471                   if (subjectType == "NON_SHS" || subjectType == "1
472                     ST_SEM") {
473                     return crs.q1;
474                   }
475                 } else if (quarter == "Q2") {
476                   if (subjectType == "NON_SHS" || subjectType == "1
477                     ST_SEM") {
478                     return crs.q2;
479                   }
480                 } else if (quarter == "Q3") {
481                   if (subjectType == "NON_SHS") {
482                     return crs.q3;
483                   } else if (subjectType == "2ND_SEM") {
484                     return crs.q1;
485                   }
486                 }
487               }
488             });
489             data2.push(data);
490           }
491         }
492       });
493       let subjectName = await utils.getSubjectName(subjectID)
494     };

```

```

494     if (typeof status !== "undefined") {
495       let score = await StudentSubjectGrades.findOne({
496         where: {
497           classRecordID,
498           subsectstudID: v.subsectstudID
499         }
500       }).then(ssg => {
501         if (status == "F") {
502           if (quarter == "Q1") {
503             if (
504               subjectType == "NON_SHS" ||
505               subjectType == "1ST_SEM"
506             ) {
507               return ssg.q1FinalGrade;
508             }
509           } else if (quarter == "Q2") {
510             if (
511               subjectType == "NON_SHS" ||
512               subjectType == "1ST_SEM"
513             ) {
514               return ssg.q2FinalGrade;
515             }
516           } else if (quarter == "Q3") {
517             if (subjectType == "NON_SHS") {
518               return ssg.q3FinalGrade;
519             } else if (subjectType == "2ND_SEM") {
520               return ssg.q1FinalGrade;
521             }
522           } else {
523             if (subjectType == "NON_SHS") {
524               return ssg.q4FinalGrade;
525             } else if (subjectType == "2ND_SEM") {
526               return ssg.q2FinalGrade;
527             }
528           }
529         }
530       });
531       if (typeof score !== "undefined") {
532         data2.push({
533           score,
534           subjectName
535         });
536       }
537     }
538   }
539   return data2;
540 }
541 );
542 let finalGrade = await StudentGrades.findOne({
543   where: {
544     studsectID,
545     quarter
546   }
547 }).then(sg => {
548   if (sg) {
549     return sg.grade;
550   }
551 });
552 res.status(200).json({
553   data,
554   finalGrade,
555   section,
556   teacher,
557   gradeLevel
558 });
559 }
560 }
561 );
562 );
563
564 module.exports = router;

```

Listing 35: teacher.js

```
1 const express = require("express");
2 const router = express.Router();
3 const UserAccount = require("../models/UserAccount");
4 const Grade = require("../models/Grade");
5 const SubjectSection = require("../models/SubjectSection");
6 const SubjectSectionStudent = require("../models/
    SubjectSectionStudent");
7 const StudentWeightedScore = require("../models/
    StudentWeightedScore");
8 const StudentSubjectGrades = require("../models/
    StudentSubjectGrades");
9 const Teacher = require("../models/Teacher");
10 const Component = require("../models/Component");
11 const ClassRecord = require("../models/ClassRecord");
12 const ClassRecordStatus = require("../models/ClassRecordStatus");
13 const Subcomponent = require("../models/Subcomponent");
14 const ActivityLog = require("../models/ActivityLog");
15 const LogDetails = require("../models/LogDetails");
16 const SchoolYear = require('../models/SchoolYear')
17 const passport = require("passport");
18 const Sequelize = require("sequelize");
19 const Op = Sequelize.Op;
20
21 // Input Validation
22 const validateEditProfileNonacademic = require("../validation/
    editprofilenonacademic");
23 const validateSubcomponent = require("../validation/subcomponents"
    );
24
25 // Import Utility Functions
26 const utils = require("../utils");
27
28 // @route POST api/teacher/updateprofile
29 // @desc Update profile
30 // @access Private
31
32 router.post(
33     "/updateprofile",
34     passport.authenticate("teacher", {
35         session: false
36     }),
37     (req, res) => {
38         const {
39             firstName,
40             lastName,
41             middleName,
42             suffix,
43             nickname,
44             contactNum,
45             address,
46             province,
47             city,
48             region,
49             zipcode,
50             civilStatus,
51             sex,
52             citizenship,
53             birthDate,
54             birthPlace,
55             religion,
56             emergencyName,
57             emergencyAddress,
58             emergencyTelephone,
59             emergencyCellphone,
60             emergencyEmail,
61             emergencyRelationship
62         } = req.body;
63
64         console.log(req.body.firstName);
65     }
66 )
```

```

66      const {
67        errors,
68        isValid
69      } = validateEditProfileNonacademic(req.body);
70
71      if (!isValid) {
72        return res.status(400).json(errors);
73      }
74
75      UserAccount.findOne({
76        where: {
77          accountID: req.user.accountID
78        }
79      }).then(user => {
80        if (user) {
81          user
82            .update({
83              firstName,
84              lastName,
85              middleName,
86              suffix,
87              nickname,
88              contactNum,
89              address,
90              province,
91              city,
92              region,
93              zipcode,
94              civilStatus,
95              sex,
96              citizenship,
97              birthDate,
98              birthPlace,
99              religion,
100             emergencyName,
101             emergencyAddress,
102             emergencyTelephone,
103             emergencyCellphone,
104             emergencyEmail,
105             emergencyRelationship
106           })
107         .then(user2 => {
108           res.status(200).json({
109             msg: "Profile updated successfully!"
110           });
111         });
112       }
113     );
114   );
115
116 // @route POST api/teacher/listssubjectsection
117 // @desc List subject section of teacher by school year
118 // @access Private
119
120 router.post(
121   "/listssubjectsection",
122   passport.authenticate("teacher", {
123     session: false
124   }),
125   async(req, res) => {
126     let {
127       teacherID,
128       schoolYear,
129       page,
130       pageSize
131     } = req.body;
132     let sessionTeacherID = await utils.getTeacherID(req.user.
133       accountID);
134     if (sessionTeacherID != teacherID) {
135       res.status(400).json({
136         msg: "Not authorized!"
137       });
138     }
139     let offset = page * pageSize;

```

```

140     let limit = offset + pageSize;
141     SubjectSection.findAll({
142       limit,
143       offset,
144       where: {
145         teacherID,
146         schoolYear
147       }
148     })
149     .then(subjectsections => {
150       let subjectsectionData = [];
151       if (subjectsections.length != 0) {
152         subjectsections
153           .slice(0, pageSize)
154           .forEach(async(subjectsection, key, arr) => {
155             const keyID = subjectsection.subsectID;
156             const subjectName = await utils.getSubjectName(
157               subjectsection.subjectID
158             );
159             const sectionName = await utils.getSectionName(
160               subjectsection.sectionID
161             );
162             const classRecordID = subjectsection.classRecordID;
163             subjectsectionData.push({
164               key: keyID,
165               subjectName,
166               sectionName,
167               classRecordID
168             });
169             if (key == arr.length - 1) {
170               SubjectSection.findAndCountAll({
171                 where: {
172                   teacherID,
173                   schoolYear
174                 }
175               })
176               .then(count => {
177                 subjectsectionData.sort((a, b) => {
178                   a.key > b.key ? 1 : -1;
179                 });
180                 res.status(200).json({
181                   numOfPages: Math.ceil(count.count / pageSize),
182                   subjectsectionList: subjectsectionData
183                 });
184               })
185               .catch(err => {
186                 res.status(404);
187               });
188             }
189           });
190         } else {
191           req.status(404).json({
192             msg: "Not found"
193           });
194         }
195       }
196       .catch(err => {
197         res.status(404);
198       });
199     });
200   );
201
202 // @route GET api/registrar/getsy
203 // @desc Get current school year
204 // @access Private
205 router.get(
206   "/getsy",
207   passport.authenticate("teacher", {
208     session: false
209   }),
210   async(req, res) => {
211     let {
212       schoolYearID,

```

```

213         quarter
214     } = await utils.getActiveSY();
215     if (schoolYearID != 0) {
216         let schoolYear = await utils.getSYname(schoolYearID);
217         res.status(200).json({
218             schoolYear,
219             schoolYearID,
220             quarter
221         });
222     } else {
223         res.status(404).json({
224             msg: "There is no active school year"
225         });
226     }
227 }
228 );
229
230 // @route GET api/teacher/getallsy
231 // @desc Get all school year
232 // @access Private
233
234 router.get(
235     "/getallsy",
236     passport.authenticate(["teacher"], {
237         session: false
238     }),
239     async(req, res) => {
240         SchoolYear.findAll().then(async sys => {
241             if (sys) {
242                 let data = [];
243                 for (const [index, value] of sys.entries()) {
244                     data.push({
245                         schoolYearID: value.schoolYearID,
246                         schoolYear: value.schoolYear
247                     });
248                 }
249                 data.sort((a, b) => (a.schoolYear < b.schoolYear ? 1 : -1));
250                 res.status(200).json({
251                     schoolYearList: data
252                 });
253             }
254         });
255     }
256 );
257
258 // @route POST api/registrar/getsubjectload
259 // @desc Get subject load of teacher
260 // @access Private
261
262 router.post(
263     "/getsubjectload",
264     passport.authenticate("teacher", {
265         session: false
266     }),
267     async(req, res) => {
268         const {
269             accountID
270         } = req.user;
271         let {
272             page,
273             pageSize,
274             schoolYearID,
275             quarter
276         } = req.body;
277         page = page - 1;
278         let offset = page * pageSize;
279         let limit = offset + pageSize;
280         const teacherID = await utils.getTeacherID(accountID);
281         SubjectSection.findAll({
282             limit,
283             offset,
284             where: {
285                 teacherID,

```

```

286         schoolYearID
287     }
288 }).then(async subjectsections => {
289     if (subjectsections.length == 0) {
290         res.status(404).json({
291             msg: "No record"
292         });
293     } else {
294         let subjectsectionData = [];
295         let i = 0;
296         for (i; i < subjectsections.slice(0, pageSize).length; i++) {
297             const key = subjectsections[i].subsectID;
298             const subjectCode = await utils.getSubjectCode(
299                 subjectsections[i].subjectID
300             );
301             const subjectName = await utils.getSubjectName(
302                 subjectsections[i].subjectID
303             );
304             const gradeLevel = await utils.getSectionGradeLevel(
305                 subjectsections[i].sectionID
306             );
307             const sectionName = await utils.getSectionName(
308                 subjectsections[i].sectionID
309             );
310             if (subjectsections[i].subjectType == "NON_SHS") {
311                 subjectsectionData.push({
312                     key,
313                     subjectCode,
314                     subjectName,
315                     gradeLevel,
316                     sectionName
317                 });
318             } else {
319                 if (subjectsections[i].subjectType == "1ST_SEM") {
320                     if (quarter == "Q1" || quarter == "Q2") {
321                         subjectsectionData.push({
322                             key,
323                             subjectCode,
324                             subjectName,
325                             gradeLevel,
326                             sectionName
327                         });
328                     }
329                 } else if (subjectsections[i].subjectType == "2ND_SEM") {
330                     if (quarter == "Q3" || quarter == "Q4") {
331                         subjectsectionData.push({
332                             key,
333                             subjectCode,
334                             subjectName,
335                             gradeLevel,
336                             sectionName
337                         });
338                     }
339                 }
340             }
341         }
342         SubjectSection.findAndCountAll({
343             where: {
344                 teacherID,
345                 schoolYearID
346             }
347         }).then(count => {
348             res.status(200).json({
349                 numOfPages: Math.ceil(count.count / pageSize),
350                 subjectSectionData: subjectsectionData
351             });
352         });
353     }
354   });
355 });
356 );
357
358 // @route POST api/teacher/getsubsectinfo

```

```

359 // @desc Get subject section info
360 // @access Private
361
362 router.post(
363   "/getsubsectinfo",
364   passport.authenticate("teacher", {
365     session: false
366   }),
367   async(req, res) => {
368     const {
369       subsectID
370     } = req.body;
371     const {
372       accountID
373     } = req.user;
374     const teacherID = await utils.getTeacherID(accountID);
375     let {
376       page,
377       pageSize
378     } = req.body;
379     page = page - 1;
380     let offset = page * pageSize;
381     let limit = offset + pageSize;
382     SubjectSection.findOne({
383       where: {
384         subsectID
385       }
386     }).then(subjectsection => {
387       if (subjectsection) {
388         if (subjectsection.teacherID != teacherID) {
389           res.status(400).json({
390             msg: "Not authorized!"
391           });
392         } else {
393           SubjectSectionStudent.findAll({
394             where: {
395               subsectID
396             }
397           }).then(async subjectsectionstudents => {
398             if (subjectsectionstudents.length == 0) {
399               const sectionName = await utils.getSectionName(
400                 subjectsection.sectionID
401               );
402               const gradeLevel = await utils.getSectionGradeLevel(
403                 subjectsection.sectionID
404               );
405               const subjectName = await utils.getSubjectName(
406                 subjectsection.subjectID
407               );
408               res.status(200).json({
409                 numOfPages: 1,
410                 sectionName,
411                 gradeLevel,
412                 subjectName,
413                 studentList: []
414               });
415             } else {
416               let i = 0;
417               let studarr = [];
418               const sectionName = await utils.getSectionName(
419                 subjectsection.sectionID
420               );
421               const gradeLevel = await utils.getSectionGradeLevel(
422                 subjectsection.sectionID
423               );
424               const subjectName = await utils.getSubjectName(
425                 subjectsection.subjectID
426               );
427               for (
428                 i; i < subjectsectionstudents.length; i++
429               ) {
430                 let key = subjectsectionstudents[i].subsectstudID;
431                 let name = await utils.getStudentNameByStudsectID(

```

```

432         subjectsectionstudents[i].studsectID
433     );
434     let sex = await utils.getStudentSexByStudsectID(
435         subjectsectionstudents[i].studsectID)
436     let email = await utils.getStudentEmailByStudsectID(
437         subjectsectionstudents[i].studsectID
438     );
439     let sectionName = await utils.
440         getSectionNameByStudsectID(
441             subjectsectionstudents[i].studsectID
442         );
443     let imageUrl = await utils.
444         getStudentImageUrlByStudsectID(
445             subjectsectionstudents[i].studsectID
446         );
447     let gradeLevel = await utils.
448         getSectionGradeLevelByStudsectID(
449             subjectsectionstudents[i].studsectID
450         );
451     studarr.push({
452         sex,
453         key,
454         name,
455         email,
456         sectionName,
457         imageUrl,
458         gradeLevel
459     });
460 }
461 SubjectSectionStudent.findAndCountAll({
462     limit,
463     offset,
464     where: {
465         subsectID
466     }
467 }).then(count => {
468     studarr.sort((a, b) => (a.name > b.name ? 1 : -1));
469     let sortedArr = [...studarr.filter(a => a.sex == "M")
470                     , ...studarr.filter(a => a.sex == "F")]
471     res.status(200).json({
472         numOfPages: Math.ceil(count.count / pageSize),
473         sectionName,
474         gradeLevel,
475         subjectName,
476         studentList: sortedArr.slice(pageSize * page,
477                                         pageSize * (page + 1))
478     });
479 });
480 }
481 } else {
482     res.status(404).json({
483         msg: "Subject section not found!"
484     });
485 }
486 });
487 });
488 );
489 );
490 // @route POST api/teacher/getcomponents
491 // @desc Get components per subject section
492 // @access Private
493 router.post(
494     "/getcomponents",
495     passport.authenticate("teacher", {
496         session: false
497     }),
498     async(req, res) => {
499         const {
500             subsectID,
501             quarter
502         }
503     );
504     let components = await Component.find({
505         subsectID
506     });
507     res.json(components);
508 });

```

```

499     } = req.body;
500     const {
501       accountID
502     } = req.user;
503     const teacherID = await utils.getTeacherID(accountID);
504     SubjectSection.findOne({
505       where: {
506         subsectID
507       }
508     }).then(subjectsection => {
509       if (subjectsection) {
510         if (subjectsection.teacherID != teacherID) {
511           res.status(400).json({
512             msg: "Not authorized!"
513           });
514         } else {
515           Component.findOne({
516             where: {
517               subjectID: subjectsection.subjectID,
518               component: "FA"
519             }
520           }).then(comp1 => {
521             Component.findOne({
522               where: {
523                 subjectID: subjectsection.subjectID,
524                 component: "WW"
525               }
526             }).then(comp2 => {
527               Component.findOne({
528                 where: {
529                   subjectID: subjectsection.subjectID,
530                   component: "PT"
531                 }
532               }).then(comp3 => {
533                 Component.findOne({
534                   where: {
535                     subjectID: subjectsection.subjectID,
536                     component: "QE"
537                   }
538                 }).then(async comp4 => {
539                   let faSubcompData = [];
540                   let wwSubcompData = [];
541                   let ptSubcompData = [];
542                   let qeSubcompData = [];
543                   await Subcomponent.findAll({
544                     where: {
545                       classRecordID: subjectsection.classRecordID,
546                       componentID: comp1.componentID,
547                       quarter
548                     }
549                   }).then(async subcomp1 => {
550                     if (subcomp1.length != 0) {
551                       let i = 0;
552                       for (i = 0; i < subcomp1.length; i++) {
553                         let {
554                           name,
555                           compWeight,
556                           subcompID
557                         } = subcomp1[i];
558                         faSubcompData.push({
559                           name,
560                           compWeight,
561                           subcompID
562                         });
563                       }
564                     }
565                   });
566                   await Subcomponent.findAll({
567                     where: {
568                       classRecordID: subjectsection.classRecordID,
569                       componentID: comp2.componentID,
570                       quarter
571                     }
572                   })
573                 })
574               })
575             })
576           })
577         })
578       })
579     })
580   })
581 
```

```

572     }).then(async subcomp2 => {
573       if (subcomp2.length != 0) {
574         let i = 0;
575         for (i = 0; i < subcomp2.length; i++) {
576           let {
577             name,
578             compWeight,
579             subcompID
580           } = subcomp2[i];
581           wwSubcompData.push({
582             name,
583             compWeight,
584             subcompID
585           });
586         }
587       }
588     });
589     await Subcomponent.findAll({
590       where: {
591         classRecordID: subjectsection.classRecordID,
592         componentID: comp3.componentID,
593         quarter
594       }
595     }).then(async subcomp3 => {
596       if (subcomp3.length != 0) {
597         let i = 0;
598         for (i = 0; i < subcomp3.length; i++) {
599           let {
600             name,
601             compWeight,
602             subcompID
603           } = subcomp3[i];
604           ptSubcompData.push({
605             name,
606             compWeight,
607             subcompID
608           });
609         }
610       }
611     });
612     await Subcomponent.findAll({
613       where: {
614         classRecordID: subjectsection.classRecordID,
615         componentID: comp4.componentID,
616         quarter
617       }
618     }).then(async subcomp4 => {
619       if (subcomp4.length != 0) {
620         let i = 0;
621         for (i = 0; i < subcomp4.length; i++) {
622           let {
623             name,
624             compWeight,
625             subcompID
626           } = subcomp4[i];
627           qeSubcompData.push({
628             name,
629             compWeight,
630             subcompID
631           });
632         }
633       }
634     });
635     const sectionName = await utils.getSectionName(
636       subjectsection.sectionID
637     );
638     const subjectName = await utils.getSubjectName(
639       subjectsection.subjectID
640     );
641     const schoolYear = await utils.getSYname(
642       subjectsection.schoolYearID
643     );
644     const subjectCode = await utils.getSubjectCode(
645       subjectsection.subjectID

```

```

646
647     );
648     res.status(200).json({
649       subjectCode,
650       sectionName,
651       subjectName,
652       schoolYear,
653       classRecordID: subjectsection.classRecordID,
654       FA: {
655         componentID: comp1.componentID,
656         weight: comp1.compWeight,
657         subcomponents: faSubcompData
658       },
659       WW: {
660         componentID: comp2.componentID,
661         weight: comp2.compWeight,
662         subcomponents: wwSubcompData
663       },
664       PT: {
665         componentID: comp3.componentID,
666         weight: comp3.compWeight,
667         subcomponents: ptSubcompData
668       },
669       QE: {
670         componentID: comp4.componentID,
671         weight: comp4.compWeight,
672         subcomponents: qeSubcompData
673       }
674     });
675   });
676 });
677 });
678 }
679 } else {
680   res.status(404).json({
681     msg: "Subject section not found!"
682   });
683 }
684 });
685 );
686 );
687
688 // @route POST api/teacher/addnewsubcomp
689 // @desc Add new subcomponent by classRecordID and componentID
690 // @access Private
691
692 router.post(
693   "/addnewsubcomp",
694   passport.authenticate("teacher", {
695     session: false
696   }),
697   async(req, res) => {
698     const {
699       classRecordID,
700       componentID,
701       name,
702       quarter
703     } = req.body;
704     const {
705       accountID
706     } = req.user;
707     const teacherID = await utils.getTeacherID(accountID);
708     const {
709       errors,
710       isValid
711     } = validateSubcomponent({
712       name
713     });
714
715     if (!isValid) {
716       return res.status(400).json({
717         msg: errors.msg
718       });
719     }

```

```

720     SubjectSection.findOne({
721       where: {
722         classRecordID
723       }
724     }).then(subjectsection => {
725       if (subjectsection) {
726         if (subjectsection.teacherID == teacherID) {
727           ClassRecordStatus.findOne({
728             where: {
729               classRecordID
730             }
731           }).then(async crs => {
732             if (crs) {
733               if (quarter == "Q1") {
734                 if (crs.q1 == "L" || crs.q1 == "D" || crs.q1 == "F") {
735                   res.status(400).json({
736                     msg: "You are not authorized to edit this class
737                     record."
738                   });
739                 } else {
740                   Subcomponent.create({
741                     name,
742                     componentID,
743                     classRecordID,
744                     compWeight: 0,
745                     quarter
746                   }, {
747                     showLog: false
748                   }).then(subcomponent => {
749                     res.status(200).json({
750                       msg: "Successfully added a new subcomponent!"
751                     });
752                   });
753                 } else if (quarter == "Q2") {
754                   if (crs.q2 == "L" || crs.q2 == "D" || crs.q2 == "F") {
755                     res.status(400).json({
756                       msg: "You are not authorized to edit this class
757                     record."
758                     });
759                   } else {
760                     Subcomponent.create({
761                       name,
762                         componentID,
763                         classRecordID,
764                         compWeight: 0,
765                         quarter
766                       }, {
767                         showLog: false
768                       }).then(subcomponent => {
769                         res.status(200).json({
770                           msg: "Successfully added a new subcomponent!"
771                         });
772                       });
773                     } else if (quarter == "Q3") {
774                       if (crs.q3 == "L" || crs.q3 == "D" || crs.q3 == "F") {
775                         res.status(400).json({
776                           msg: "You are not authorized to edit this class
777                             record."
778                           });
779                         } else {
780                           Subcomponent.create({
781                             name,
782                               componentID,
783                               classRecordID,
784                               compWeight: 0,
785                               quarter
786                             }, {
787                               showLog: false

```

```

787             });
788             res.status(200).json({
789                 msg: "Successfully added a new subcomponent!"
790             });
791         });
792     } else {
793         if (crs.q4 == "L" || crs.q4 == "D" || crs.q4 == "F")
794         {
795             res.status(400).json({
796                 msg: "You are not authorized to edit this class
797                     record."
798             });
799         } else {
800             Subcomponent.create({
801                 name,
802                 componentID,
803                 classRecordID,
804                 compWeight: 0,
805                 quarter
806             }, {
807                 showLog: false
808             }).then(subcomponent => {
809                 res.status(200).json({
810                     msg: "Successfully added a new subcomponent!"
811                 });
812             });
813         }
814     } else {
815         res.status(404).json({
816             msg: "Class record status does not exist."
817         });
818     });
819 };
820 } else {
821     res.status(400).json({
822         msg: "Not authorized!"
823     });
824 }
825 } else {
826     res.status(404).json({
827         msg: "Subject section not found!"
828     });
829 }
830 });
831 );
832 );
833
834 // @route POST api/teacher/editsubcomp
835 // @desc Edit subcomponent name, weight by subcomponentID
836 // @access Private
837
838 router.post(
839     "/editsubcomp",
840     passport.authenticate("teacher", {
841         session: false
842     }),
843     async(req, res) => {
844         const reg = /^-?[0-9]*(\.[0-9]*)?$/;
845         const {
846             payload,
847             classRecordID,
848             quarter
849         } = req.body;
850         const {
851             accountID
852         } = req.user;
853         const teacherID = await utils.getTeacherID(accountID);
854         ClassRecordStatus.findOne({
855             where: {
856                 classRecordID
857             }
858         })
859         .then(subcomponent => {
860             if (subcomponent)
861             {
862                 subcomponent.name = payload.name;
863                 subcomponent.compWeight = payload.compWeight;
864                 subcomponent.quarter = payload.quarter;
865                 subcomponent.showLog = payload.showLog;
866                 subcomponent.save();
867             }
868             res.json({
869                 msg: "Subcomponent updated successfully"
870             });
871         })
872     }
873 );

```

```

858     }).then(async crs => {
859       if (crs) {
860         let invalid = false;
861         if (quarter == "Q1") {
862           invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 == "F";
863         } else if (quarter == "Q2") {
864           invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 == "F";
865         } else if (quarter == "Q3") {
866           invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 == "F";
867         } else {
868           invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 == "F";
869         }
870         if (invalid) {
871           res.status(400).json({
872             msg: "You are not authorized to edit this class record."
873           });
874         } else {
875           let i = 0;
876           let sum = 0;
877           for (i = 0; i < payload.length; i++) {
878             let {
879               errors,
880               isValid
881             } = validateSubcomponent({
882               name: payload[i].name
883             });
884             if (!isValid) {
885               return res.status(400).json({
886                 msg: errors.msg
887               });
888             } else {
889               let valid =
890                 (!isNaN(payload[i].compWeight) &&
891                  reg.test(payload[i].compWeight)) ||
892                  payload[i].compWeight === "" ||
893                  payload[i].compWeight === "-";
894               if (
895                 payload[i].compWeight > 100 ||
896                 payload[i].compWeight < 0 ||
897                 !valid
898               ) {
899                 return res.status(400).json({
900                   msg: "Invalid component weight input"
901                 });
902               } else {
903                 sum = sum + parseFloat(payload[i].compWeight);
904               }
905             }
906           }
907           if (sum != 100) {
908             return res.status(400).json({
909               msg: "Sum of component weights must be 100!"
910             });
911           }
912         }
913         payload.forEach(async(val, arr, index) => {
914           await Subcomponent.findOne({
915             where: {
916               subcompID: val.subcompID
917             }
918           }).then(subcomp => {
919             if (subcomp) {
920               SubjectSection.findOne({
921                 where: {
922                   classRecordID: subcomp.classRecordID
923                 }
924               }).then(subjectsection => {
925                 if (subjectsection) {
926                   if (subjectsection.teacherID == teacherID) {
927                     let {
928                       name,

```

```

930                     compWeight
931             } = val;
932             subcomp
933                 .update({
934                     name,
935                     compWeight
936                 }, {
937                     showLog: false
938                 })
939             .then(async() => {
940                 await utils.
941                     refreshStudentWeightedScoreBySubsectID(
942                         subjectsection.subsectID
943                     );
944             });
945         } else {
946             res.status(401).json({
947                 msg: "Not authorized!"
948             });
949         }
950     } else {
951         res.status(400).json({
952             msg: "Subject Section not found!"
953         });
954     });
955     } else {
956         res.status(400).json({
957             msg: "Subcomponent not found!"
958         });
959     });
960 });
961 });
962 res.status(200).json({
963     msg: "Subcomponent updated successfully!"
964 });
965 }
966 } else {
967     res.status(404).json({
968         msg: "Class record status not found!"
969     });
970 }
971 );
972 );
973 );
974
975 // @router POST api/teacher/deletesubcomp
976 // @desc Delete subcomponent by subcomponentID
977 // @access Private
978
979 router.post(
980     "/deletesubcomp",
981     passport.authenticate("teacher", {
982         session: false
983     }),
984     async(req, res) => {
985         const {
986             subcompID,
987             quarter,
988             classRecordID
989         } = req.body;
990         const {
991             accountID
992         } = req.user;
993         const teacherID = await utils.getTeacherID(accountID);
994         ClassRecordStatus.findOne({
995             where: {
996                 classRecordID
997             }
998         }).then(crs => {
999             if (crs) {
1000                 let invalid = false;

```

```

1001     if (quarter == "Q1") {
1002         invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 == "F";
1003     } else if (quarter == "Q2") {
1004         invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 == "F";
1005     } else if (quarter == "Q3") {
1006         invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 == "F";
1007     } else {
1008         invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 == "F";
1009     }
1010     if (invalid) {
1011         res.status(400).json({
1012             msg: "You are not authorized to edit this class record."
1013         });
1014     } else {
1015         Subcomponent.findOne({
1016             where: {
1017                 subcompID
1018             }
1019         }).then(subcomponent => {
1020             if (subcomponent) {
1021                 Subcomponent.findAll({
1022                     where: {
1023                         classRecordID: subcomponent.classRecordID,
1024                         componentID: subcomponent.componentID
1025                     }
1026                 }).then(c => {
1027                     if (c.length == 1) {
1028                         res.status(400).json({
1029                             msg: "Subcomponent can't be deleted. There must
1030                                 be at least one subcomponent."
1031                         });
1032                     } else {
1033                         SubjectSection.findOne({
1034                             where: {
1035                                 classRecordID: subcomponent.classRecordID
1036                             }
1037                         }).then(subjectsection => {
1038                             if (subjectsection) {
1039                                 if (subjectsection.teacherID != teacherID) {
1040                                     res.status(400).json({
1041                                         msg: "Not authorized!"
1042                                     });
1043                                 } else {
1044                                     let compweight = subcomponent.compWeight;
1045                                     if (compweight != 0) {
1046                                         res.status(400).json({
1047                                             msg: "Subcomponent can't be deleted. Set
1048                                                 component weight to 0 first."
1049                                         });
1050                                     } else {
1051                                         Grade.findOne({
1052                                             where: {
1053                                                 subcomponentID: subcomponent.subcompID
1054                                             }
1055                                         }).then(g => {
1056                                             if (g) {
1057                                                 res.status(400).json({
1058                                                     msg: "Operation could not be
1059                                         completed. Delete all grades
1060                                         under this subcomponent first."
1061                                         });
1062                                             } else
1063                                                 subcomponent
1064                                                 .destroy({
1065                                                     showLog: false
1066                                                 })
1067                                                 .then(() => {
1068                                                     res.status(200).json({
1069                                                         msg: "Subcomponent deleted
1070                                                 successfully!"
1071                                                     });
1072                                                 });
1073                                         });
1074                                     }
1075                                 }
1076                             }
1077                         });
1078                     }
1079                 });
1080             }
1081         });
1082     }
1083 
```

```

1068             });
1069         });
1070     });
1071   } else {
1072     res.status(404).json({
1073       msg: "Subject section not found!"
1074     });
1075   });
1076 });
1077 });
1078 });
1079 } else {
1080   res.status(404).json({
1081     msg: "Subcomponent not found!"
1082   });
1083 });
1084 });
1085 });
1086 } else {
1087   res.status(404).json({
1088     msg: "Class record status does not exist."
1089   });
1090 });
1091 });
1092 });
1093 );
1094 );
1095 // @router POST api/teacher/compinfo
1096 // @desc Get component information by componentID and subseectID
1097 // @access Private
1098
1099 router.post(
1100   "/compinfo",
1101   passport.authenticate("teacher", {
1102     session: false
1103   }),
1104   async(req, res) => {
1105     const {
1106       subsectID,
1107       componentID,
1108       quarter
1109     } = req.body;
1110     const {
1111       accountID
1112     } = req.user;
1113     const teacherID = await utils.getTeacherID(accountID);
1114     SubjectSection.findOne({
1115       where: {
1116         subsectID
1117       }
1118     }).then(async subjectsection => {
1119       if (subjectsection) {
1120         if (subjectsection.teacherID != teacherID) {
1121           res.status(400).json({
1122             msg: "Not authorized!"
1123           });
1124         } else {
1125           Component.findOne({
1126             where: {
1127               componentID
1128             }
1129           }).then(async component => {
1130             if (component) {
1131               let component = await utils.getComponentName(
1132                 componentID);
1133               let grades = await Subcomponent.findAll({
1134                 where: {
1135                   componentID,
1136                   classRecordID: subjectsection.classRecordID,
1137                   quarter
1138                 }
1139               }).then(async subcomp => {

```

```

1139     if (subcomp.length == 0) {
1140       res.status(404).json({
1141         msg: "Subcomponents not found!"
1142       });
1143     } else {
1144       let i = 0;
1145       let data1 = [];
1146       for (i = 0; i < subcomp.length; i++) {
1147         const {
1148           subcompID
1149         } = subcomp[i];
1150         let name = await utils.getSubcomponentName(
1151           subcompID);
1152         name = name + ' (${subcomp[i].compWeight}%)';
1153         const data2 = await SubjectSectionStudent.findAll(
1154           {
1155             where: {
1156               subsectID
1157             }
1158           }).then(async subsectstud => {
1159             if (subsectstud.length == 0) {
1160               res.status(404).json({
1161                 msg: "Subject section student not found!"
1162               });
1163             } else {
1164               let j = 0;
1165               let studentData = [];
1166               for (j = 0; j < subsectstud.length; j++) {
1167                 const {
1168                   subsectstudID
1169                 } = subsectstud[j];
1170                 const name = await utils.
1171                   getStudentNameBySubsectstudID(
1172                     subsectstudID
1173                   );
1174                 const sex = await utils.
1175                   getStudentSexBySubsectstudID(
1176                     subsectstudID)
1177                   );
1178                 const grades = await Grade.findAll({
1179                   where: {
1180                     componentID ,
1181                     subcomponentID: subcomp[i].subcompID ,
1182                     classRecordID: subjectsection .
1183                     classRecordID ,
1184                     subsectstudID: subsectstud[j] .
1185                     subsectstudID ,
1186                     quarter
1187                   },
1188                   order: [
1189                     ["dateGiven", "DESC"]
1190                   ]
1191                 }).then(async grades => {
1192                   if (grades.length == 0) {
1193                     return [];
1194                   } else {
1195                     let k = 0;
1196                     let gradesData = [];
1197                     for (k = 0; k < grades.length; k++) {
1198                       const {
1199                         gradeID ,
1200                         dateGiven ,
1201                         score ,
1202                         total ,
1203                         attendance
1204                       } = grades[k];
1205                       gradesData.push({
1206                         gradeID ,
1207                         dateGiven ,
1208                         score ,

```

```

1204             total ,
1205             attendance
1206         });
1207     }
1208     return gradesData;
1209 }
1210 });
1211 let k = 0;
1212 let sumScores = 0;
1213 let sumItems = 0;
1214 let excused = 0;
1215 let ps = -1;
1216 for (k = 0; k < grades.length; k++) {
1217     if (grades[k].attendance == "E") {
1218         excused = excused + 1;
1219     } else if (grades[k].attendance == "A") {
1220         sumScores = sumScores + 0;
1221         sumItems = sumItems + parseFloat(grades
1222             [k].total);
1223     } else {
1224         sumScores =
1225             sumScores + parseFloat(grades[k].
1226                 score);
1227         sumItems = sumItems + parseFloat(grades
1228             [k].total);
1229     }
1230 }
1231 if (grades.length != 0) {
1232     if (grades.length == excused) {
1233         ps = -1;
1234     } else {
1235         ps = (sumScores / sumItems) * 100;
1236     }
1237     studentData.push({
1238         sex,
1239         imageUrl,
1240         subsectstudID,
1241         name,
1242         grades,
1243         ps
1244     });
1245 }
1246 return studentData;
1247 });
1248 data2.sort((a, b) => (a.name > b.name ? 1 : -1));
1249 let tempData2 = [...data2.filter(a => a.sex == "M"
1250     "), ...data2.filter(a => a.sex == "F")]
1251 data1.push({
1252     subcompID,
1253     name,
1254     data: tempData2
1255 });
1256 return data1;
1257 }
1258 });
1259 let ave = await StudentWeightedScore.findAll({
1260     where: {
1261         classRecordID: subjectsection.classRecordID,
1262         quarter
1263     }
1264 }).then(async studweightedscore => {
1265     if (studweightedscore.length == 0) {
1266         return [];
1267     } else {
1268         let l = 0;
1269         const aveData = [];
1270         for (l = 0; l < studweightedscore.length; l++) {
1271             const {
1272                 subsectstudID,

```

```

1273         faWS ,
1274         wwWS ,
1275         ptWS ,
1276         qeWS ,
1277         actualGrade
1278     } = studweightedscore[1];
1279     const name = await utils.
1280             getStudentNameBySubsectstudID(
1281             subsectstudID
1282         );
1283     let sex = await utils.
1284             getStudentSexBySubsectstudID(subsectstudID)
1285     let ws = 0;
1286     switch (component) {
1287         case "Formative Assessment":
1288         {
1289             ws = faWS;
1290             break;
1291         }
1292         case "Written Works":
1293         {
1294             ws = wwWS;
1295             break;
1296         }
1297         case "Performance Task":
1298         {
1299             ws = ptWS;
1300             break;
1301         }
1302         case "Quarterly Exam":
1303         {
1304             ws = qeWS;
1305             break;
1306         }
1307         default:
1308             break;
1309     }
1310     aveData.push({
1311         sex,
1312         subsectstudID,
1313         ws,
1314         name
1315     });
1316     return aveData;
1317 });
1318 });
1319 ave.sort((a, b) => (a.name > b.name ? 1 : -1));
1320 let tempData = [...ave.filter(a => a.sex == "M"), ...
1321                 ave.filter(a => a.sex == "F")]
1322 res.status(200).json({
1323     componentID,
1324     component,
1325     grades,
1326     ave: tempData
1327 });
1328 } else {
1329     res.status(404).json({
1330         msg: "Component not found!"
1331     });
1332 }
1333 } else {
1334     res.status(404).json({
1335         msg: "Subject section not found!"
1336     });
1337 }
1338 });
1339 );
1340 );
1341
1342 // @router POST api/teacher/subcompinfo
1343 // @desc Get component information by componentID and subseectID

```

```

1344 // @access Private
1345
1346 router.post(
1347   "/subcompinfo",
1348   passport.authenticate("teacher", {
1349     session: false
1350   }),
1351   async(req, res) => {
1352     const {
1353       subsectID,
1354       componentID,
1355       subcompID,
1356       quarter
1357     } = req.body;
1358     const {
1359       accountID
1360     } = req.user;
1361     const teacherID = await utils.getTeacherID(accountID);
1362     SubjectSection.findOne({
1363       where: {
1364         subsectID
1365       }
1366     }).then(async subjectsection => {
1367       if (subjectsection) {
1368         if (subjectsection.teacherID != teacherID) {
1369           res.status(400).json({
1370             msg: "Not authorized!"
1371           });
1372         } else {
1373           Component.findOne({
1374             where: {
1375               componentID
1376             }
1377           }).then(async component => {
1378             if (component) {
1379               let component = await utils.getComponentName(
1380                 componentID);
1381               let {
1382                 subcomponentID,
1383                 name,
1384                 data
1385               } = await Subcomponent.findOne({
1386                 where: {
1387                   componentID,
1388                   classRecordID: subjectsection.classRecordID,
1389                   subcompID,
1390                   quarter
1391                 }
1392               }).then(async subcomp => {
1393                 if (!subcomp) {
1394                   res.status(404).json({
1395                     msg: "Subcomponents not found!"
1396                   });
1397                 } else {
1398                   const subcomponentID = subcomp.subcompID;
1399                   let name = await utils.getSubcomponentName(
1400                     subcomponentID);
1401                   name = name + ` (${subcomp.compWeight}%)`;
1402                   const data2 = await SubjectSectionStudent.findAll({
1403                     where: {
1404                       subsectID
1405                     }
1406                   }).then(async subsectstud => {
1407                     if (subsectstud.length == 0) {
1408                       res.status(404).json({
1409                         msg: "Subject section student not found!"
1410                       });
1411                     } else {
1412                       let j = 0;
1413                       let studentData = [];
1414                       for (j = 0; j < subsectstud.length; j++) {
1415                         const {
1416                           subsectstudID

```

```

1415     } = subsectstud[j];
1416     const name = await utils.
1417         getStudentNameBySubsectstudID(
1418             subsectstudID
1419         );
1420     let sex = await utils.
1421         getStudentSexBySubsectstudID(
1422             subsectstudID)
1423     const imageUrl = await utils.
1424         getStudentImageUrlBySubsectstudID(
1425             subsectstudID
1426         );
1427     const grades = await Grade.findAll({
1428         where: {
1429             componentID,
1430             subcomponentID: subcomp.subcompID,
1431             classRecordID: subjectsection.
1432                 classRecordID,
1433             subsectstudID: subsectstud[j].
1434                 subsectstudID,
1435                 quarter
1436         },
1437         order: [
1438             ["date", "DESC"]
1439         ]
1440     }).then(async grades => {
1441         if (grades.length == 0) {
1442             return [];
1443         } else {
1444             let k = 0;
1445             let gradesData = [];
1446             for (k = 0; k < grades.length; k++) {
1447                 const {
1448                     gradeID,
1449                     dateGiven,
1450                     score,
1451                     total,
1452                     description,
1453                     attendance
1454                 } = grades[k];
1455                 gradesData.push({
1456                     gradeID,
1457                     dateGiven,
1458                     score,
1459                     total,
1460                     description,
1461                     attendance
1462                 });
1463             }
1464             return gradesData;
1465         }
1466     });
1467     let k = 0;
1468     let sumScores = 0;
1469     let sumItems = 0;
1470     let excused = 0;
1471     let ps = -1;
1472     for (k = 0; k < grades.length; k++) {
1473         if (grades[k].attendance == "E") {
1474             excused = excused + 1;
1475         } else if (grades[k].attendance == "A") {
1476             sumScores = sumScores + 0;
1477             sumItems = sumItems + parseFloat(grades[k].
1478                 .total);
1479         } else {
1480             sumScores = sumScores + parseFloat(grades
1481                 [k].score);
1482             sumItems = sumItems + parseFloat(grades[k].
1483                 .total);
1484         }
1485     }
1486     if (grades.length != 0) {
1487         if (grades.length == excused) {
1488             ps = -1;

```

```

1480
1481         } else {
1482             ps = (sumScores / sumItems) * 100;
1483         }
1484     }
1485     studentData.push({
1486         sex,
1487         imageUrl,
1488         subsectstudID,
1489         name,
1490         grades,
1491         ps
1492     });
1493 }
1494 studentData.sort((a, b) => (a.name > b.name ? 1
1495 : -1));
1496 return [...studentData.filter(a => a.sex == "M"
1497 ), ...studentData.filter(a => a.sex == "F")
1498 ];
1499 }
1500 }
1501 return {
1502     subcomponentID,
1503     name,
1504     data: data2
1505 };
1506 });
1507 let ave = await StudentWeightedScore.findAll({
1508     where: {
1509         classRecordID: subjectsection.classRecordID,
1510         quarter
1511     }
1512 }).then(async studweightedscore => {
1513     if (studweightedscore.length == 0) {
1514         return [];
1515     } else {
1516         let l = 0;
1517         const aveData = [];
1518         for (l = 0; l < studweightedscore.length; l++) {
1519             const {
1520                 subsectstudID,
1521                 faWS,
1522                 wwWS,
1523                 ptWS,
1524                 qeWS,
1525                 actualGrade
1526             } = studweightedscore[l];
1527             const name = await utils.
1528                 getStudentNameBySubsectstudID(
1529                     subsectstudID
1530                 );
1531             let sex = await utils.
1532                 getStudentSexBySubsectstudID(subsectstudID)
1533             let ws = 0;
1534             switch (component) {
1535                 case "Formative Assessment":
1536                     {
1537                         ws = faWS;
1538                     }
1539                 case "Written Works":
1540                     {
1541                         ws = wwWS;
1542                     }
1543                 case "Performance Task":
1544                     {
1545                         ws = ptWS;
1546                     }
1547                 case "Quarterly Exam":
1548                     {
1549                         ws = qeWS;
1550                     }
1551             default:

```

```

1549                     " ";
1550                 }
1551             aveData.push({
1552                 sex,
1553                 subsectstudID,
1554                 ws,
1555                 name
1556             });
1557         }
1558     return aveData;
1559 }
1560 );
1561 ave.sort((a, b) => a.name > b.name ? 1 : -1)
1562 res.status(200).json({
1563     componentID,
1564     component,
1565     subcompID: subcomponentID,
1566     subcompName: name,
1567     data,
1568     ave: [...ave.filter(a => a.sex == "M"), ...ave.filter
1569         (a => a.sex == "F")]
1570     });
1571 } else {
1572     res.status(404).json({
1573         msg: "Component not found!"
1574     });
1575 };
1576 }
1577 } else {
1578     res.status(404).json({
1579         msg: "Subject section not found!"
1580     });
1581 }
1582 );
1583 );
1584 );
1585
// @route POST api/teacher/submitclassrecord
// @desc Submit class record by classRecordID and quarter
// @access Private
1589
router.post(
    "/submitclassrecord",
    passport.authenticate("teacher", {
        session: false
    }),
    async(req, res) => {
        const {
            classRecordID,
            quarter
        } = req.body;
        const {
            schoolYearID,
            quarter: compareQuarter
        } = await utils.getActiveSY();
        const {
            accountID
        } = req.user;
        const teacherID = await utils.getTeacherID(accountID);
        SubjectSection.findOne({
            where: {
                classRecordID
            }
        }).then(async ss => {
            if (ss) {
                if (teacherID != ss.teacherID) {
                    res.status(400).json({
                        msg: "You are unauthorized to make the changes."
                    });
                } else {
                    if (schoolYearID != ss.schoolYearID) {
                        res.status(400).json({

```

```

1621             msg: "You are unauthorized to make the changes."
1622         });
1623     } else {
1624         let comp = [];
1625         if (ss.subjectType == "2ND_SEM") {
1626             if (compareQuarter == "Q3") {
1627                 comp.push("Q1");
1628             } else {
1629                 comp.push("Q2");
1630             }
1631         } else {
1632             comp.push(compareQuarter)
1633         }
1634         if (!comp.includes(quarter)) {
1635             res.status(400).json({
1636                 msg: "You are unauthorized to make the changes."
1637             });
1638         } else {
1639             ClassRecordStatus.findOne({
1640                 where: {
1641                     classRecordID
1642                 }
1643             }).then(async crs => {
1644                 if (crs) {
1645                     let invalid = false;
1646                     let quarterObj = {};
1647                     if (quarter == "Q1") {
1648                         quarterObj["q1"] = "D";
1649                         quarterObj["q1DateSubmitted"] = utils.getPHTime()
1650                         ;
1651                         invalid = crs.q1 == "L" || crs.q1 == "D" || crs.
1652                             q1 == "F";
1653                     } else if (quarter == "Q2") {
1654                         quarterObj["q2"] = "D";
1655                         quarterObj["q2DateSubmitted"] = utils.getPHTime()
1656                         ;
1657                         invalid = crs.q2 == "L" || crs.q2 == "D" || crs.
1658                             q2 == "F";
1659                     } else if (quarter == "Q3") {
1660                         quarterObj["q3"] = "D";
1661                         quarterObj["q3DateSubmitted"] = utils.getPHTime()
1662                         ;
1663                         invalid = crs.q3 == "L" || crs.q3 == "D" || crs.
1664                             q3 == "F";
1665                     } else {
1666                         quarterObj["q4"] = "D";
1667                         quarterObj["q4DateSubmitted"] = utils.getPHTime()
1668                         ;
1669                         invalid = crs.q4 == "L" || crs.q4 == "D" || crs.
1670                             q4 == "F";
1671                     }
1672                     if (invalid) {
1673                         res.status(400).json({
1674                             msg: "You are not authorized to edit this class
1675                             record."
1676                         });
1677                     } else {
1678                         crs
1679                             .update(quarterObj, {
1680                                 position: "Teacher",
1681                                 classRecordID,
1682                                 type: "CHANGE_STATUS",
1683                                 accountID: req.user.accountID,
1684                                 sectionID: ss.sectionID,
1685                                 subjectID: ss.subjectID,
1686                                 oldVal: "E",
1687                                 newVal: "D",
1688                                 quarter
1689                             })
1690                         .then(() => {
1691                             res.status(200).json({
1692                                 msg: "Class record is now submitted for
1693                                 deliberation!"
1694                         })
1695                     }
1696                 }
1697             })
1698         }
1699     }
2000 }

```

```

1684         });
1685         });
1686     }
1687   } else {
1688     res.status(404).json({
1689       msg: "Class record status does not exist"
1690     });
1691   }
1692 });
1693 }
1694 }
1695 }
1696 } else {
1697   res.status(404).json({
1698     msg: "Subject section not found!"
1699   });
1700 }
1701 );
1702 );
1703 );
1704
1705 // @route POST api/teacher/addnewrecord
1706 // @desc Add new record under subcomponent
1707 // @access Private
1708
1709 router.post(
1710   "/addnewrecord",
1711   passport.authenticate("teacher", {
1712     session: false
1713   }),
1714   async(req, res) => {
1715     const reg = /^-?[0-9]*(\.[0-9]*)?$/;
1716     const {
1717       componentID,
1718       subcompID,
1719       payload,
1720       dateGiven,
1721       description,
1722       total,
1723       subsectID,
1724       quarter
1725     } = req.body;
1726     let totalValid = !isNaN(total) && reg.test(total) && total !== ""
1727       && total !== "-";
1728     SubjectSection.findOne({
1729       where: {
1730         subsectID
1731       }
1732     }).then(async ss => {
1733       if (ss) {
1734         ClassRecordStatus.findOne({
1735           where: {
1736             classRecordID: ss.classRecordID
1737           }
1738         }).then(async crs => {
1739           if (crs) {
1740             let invalid = false;
1741             if (quarter == "Q1") {
1742               invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 == "F";
1743             } else if (quarter == "Q2") {
1744               invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 == "F";
1745             } else if (quarter == "Q3") {
1746               invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 == "F";
1747             } else {
1748               invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 == "F";
1749             }
1750             if (invalid) {
1751               res.status(400).json({
1752                 msg: "You are not authorized to edit this class

```

```

        record."
1752 });
1753 } else {
1754     if (!totalValid) {
1755         res.status(400).json({
1756             msg: "Total score is invalid. Enter numbers only."
1757         });
1758     } else {
1759         const {
1760             accountID
1761         } = req.user;
1762         const teacherID = await utils.getTeacherID(accountID)
1763         ;
1764         if (total <= 0) {
1765             res.status(400).json({
1766                 msg: "Total must be more than 0"
1767             });
1768         }
1769         SubjectSection.findOne({
1770             where: {
1771                 subsectID
1772             }
1773         }).then(async subjectsection => {
1774             if (subjectsection) {
1775                 if (subjectsection.teacherID != teacherID) {
1776                     return res.status(400).json({
1777                         msg: "Not authorized!"
1778                     });
1779                 } else {
1780                     let {
1781                         errors,
1782                         isValid
1783                     } = validateSubcomponent({
1784                         name: description
1785                     });
1786                     if (!isValid) {
1787                         return res.status(400).json({
1788                             msg: "Error input: Description"
1789                         });
1790                     } else {
1791                         let i = 0;
1792                         let invalid = false;
1793                         for (i = 0; i < payload.length; i++) {
1794                             let {
1795                                 score,
1796                                 subsectstudID
1797                             } = payload[i];
1798                             let valid =
1799                             (!isNaN(score) &&
1800                             reg.test(score) &&
1801                             score !== "" &&
1802                             score !== "-") ||
1803                             score == "A" ||
1804                             score == "E";
1805                             if (!valid) {
1806                                 invalid = true;
1807                                 return res.status(400).json({
1808                                     msg: "Invalid score input. Enter
1809                                         numbers, E, or B only"
1810                                 });
1811                             }
1812                         for (const [index, value] of payload.entries
1813 () ) {
1814                             await Grade.findAll({
1815                             where: {
1816                                 componentID,
1817                                 subcomponentID: subcompID,
1818                                 quarter,
1819                                 subsectstudID: value.subsectstudID
1820                             }
1821                         }).then(async grades2 => {

```

```

1821     let sum = 0;
1822     let totalTemp = 0;
1823     for (const [index2, value2] of grades2.
1824         entries()) {
1825         if (value2.score == "A") {
1826             sum = sum + 0;
1827             totalTemp =
1828                 totalTemp + parseFloat(value2.total
1829                 );
1830             } else if (value2.score == "E") {} else
1831                 {
1832                     sum = sum + parseFloat(value2.score);
1833                     totalTemp =
1834                         totalTemp + parseFloat(value2.total
1835                             );
1836                     }
1837             }
1838             console.log(sum + parseFloat(value.score)
1839                 );
1840             console.log(totalTemp + parseFloat(total)
1841                 );
1842             if (
1843                 sum + parseFloat(value.score) >
1844                 totalTemp + parseFloat(total) &&
1845                 !invalid
1846             ) {
1847                 // invalid = true;
1848                 // return res.status(400).json({
1849                 //   msg:
1850                 //     "Invalid score input. Scores
1851                 //       must accumulate to less than or
1852                 //       equal to the total number of items
1853                 //   ");
1854             }
1855         }
1856     });
1857     await Grade.findOne({
1858         where: {
1859             description,
1860             dateGiven,
1861             componentID,
1862             subcomponentID: subcompID,
1863             classRecordID: subjectsection.
1864                 classRecordID,
1865             quarter
1866         }
1867     }).then(async gr => {
1868         if (gr) {
1869             res.status(400).json({
1870                 msg: "Description already exists"
1871             });
1872         } else {
1873             for (i = 0; i < payload.length; i++) {
1874                 let {
1875                     score,
1876                     subsectstudID
1877                 } = payload[i];
1878                 let attendance = "";
1879                 if (score == "A") {
1880                     attendance = "A";
1881                 } else if (score == "E") {
1882                     attendance = "E";
1883                 } else {
1884                     attendance = "P";
1885                 }
1886                 if (!invalid) {
1887                     Grade.create({
1888                         description,
1889                         dateGiven,
1890                         score,
1891                         subsectstudID,
1892                         attendance
1893                     });
1894                 }
1895             }
1896         }
1897     });

```

```

1884                               attendance ,
1885                               total ,
1886                               componentID ,
1887                               subcomponentID: subcompID ,
1888                               date: new Date(),
1889                               classRecordID: subjectsection.
1890                               classRecordID ,
1891                               subsectstudID ,
1892                               showLog: 0 ,
1893                               isUpdated: 0 ,
1894                               quarter: quarter
1895                         }, {
1896                           showLog: false
1897                         );
1898                     }
1899                   await utils.
1900                     refreshStudentWeightedScoreBySubsectID
1901                     (
1902                       subsectID
1903                     );
1904                     return res.status(200).json({
1905                       msg: "Record has been added
1906                         successfully!"
1907                     });
1908                   }
1909                 } else {
1910                   res.status(404).json({
1911                     msg: "Subject section not found!"
1912                   });
1913                 }
1914               );
1915             }
1916           }
1917         } else {
1918           res.status(404).json({
1919             msg: "Class record status does not exist"
1920           });
1921         }
1922       );
1923     } else {
1924       res.status(404).json({
1925         msg: "Subject section does not exist"
1926       });
1927     }
1928   );
1929 }
1930 );
1931
1932 // @route POST api/teacher/getactivitylog
1933 // @desc Get activity log by classRecordID
1934 // @access Private
1935
1936 router.post(
1937   "/getactivitylog",
1938   passport.authenticate("teacher", {
1939     session: false
1940   }),
1941   async(req, res) => {
1942     const {
1943       classRecordID,
1944       quarter
1945     } = req.body;
1946     const responseData = await ActivityLog.findAll({
1947       where: {
1948         classRecordID,
1949         quarter
1950       },
1951       order: [
1952         ["timestamp", "DESC"]

```

```

1953     ]
1954   }).then(async activitylogs => {
1955     if (activitylogs) {
1956       let activitylogData = [];
1957       for (const [index, al] of activitylogs.entries()) {
1958         const {
1959           position,
1960           name,
1961           section,
1962           subject,
1963           type,
1964           logID,
1965           timestamp
1966         } = al;
1967         const logDetails = await LogDetails.findAll({
1968           where: {
1969             logID
1970           }
1971         }).then(async lds => {
1972           if (lds) {
1973             let logDetailsData = [];
1974             for (const [index2, ld] of lds.entries()) {
1975               const {
1976                 student,
1977                 component,
1978                 subcomponent,
1979                 description,
1980                 oldValue,
1981                 newValue
1982               } = ld;
1983               logDetailsData.push({
1984                 student,
1985                 component,
1986                 subcomponent,
1987                 description,
1988                 oldValue,
1989                 newValue
1990               });
1991             }
1992             return logDetailsData;
1993           }
1994         });
1995       }
1996       activitylogData.push({
1997         position,
1998         name,
1999         section,
2000         subject,
2001         type,
2002         timestamp,
2003         logDetails
2004       });
2005     }
2006   }
2007   return activitylogData;
2008 }
2009 );
2010 );
2011
2012 res.status(200).json({
2013   activityLog: responseData
2014 });
2015 }
2016 );
2017
2018 // @route POST api/teacher/deleterecord
2019 // @desc Delete a record
2020 // @access Private
2021
2022 router.post(
2023   "/deleterecord",
2024   passport.authenticate("teacher", {
2025     session: false
2026   }),

```

```

2027     async(req, res) => {
2028       const {
2029         accountID
2030       } = req.user;
2031       const {
2032         subsectID,
2033         description,
2034         dateGiven,
2035         subcompID,
2036         componentID,
2037         quarter
2038       } = req.body;
2039       const teacherID = await utils.getTeacherID(accountID);
2040       SubjectSection.findOne({
2041         where: {
2042           subsectID
2043         }
2044       }).then(subsect => {
2045         if (subsect) {
2046           if (subsect.teacherID != teacherID) {
2047             res.status(401).json({
2048               msg: "Not authorized!"
2049             });
2050           } else {
2051             ClassRecordStatus.findOne({
2052               where: {
2053                 classRecordID: subsect.classRecordID
2054               }
2055             }).then(async crs => {
2056               if (crs) {
2057                 let invalid = false;
2058                 if (quarter == "Q1") {
2059                   invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 ==
2060                     "F";
2061                 } else if (quarter == "Q2") {
2062                   invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 ==
2063                     "F";
2064                 } else if (quarter == "Q3") {
2065                   invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 ==
2066                     "F";
2067                 } else {
2068                   invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 ==
2069                     "F";
2070                 }
2071                 if (invalid) {
2072                   res.status(400).json({
2073                     msg: "You are not authorized to edit this class
2074                       record."
2075                   });
2076                 } else {
2077                   Grade.findAll({
2078                     where: {
2079                       description,
2080                       dateGiven,
2081                       subcomponentID: subcompID,
2082                       componentID,
2083                       quarter
2084                     }
2085                   }).then(async grades => {
2086                     if (grades.length != 0) {
2087                       for (const [index, value] of grades.entries()) {
2088                         await value.destroy({
2089                           showLog: false
2090                         });
2091                       await utils.
2092                         refreshStudentWeightedScoreBySubsectID(
2093                           subsectID
2094                         );
2095                       res.status(200).json({
2096                         msg: "Deleted successfully!"
2097                       });
2098                     } else {
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
2999

```

```

2094             res.status(404).json({
2095                 msg: "Grades not found!"
2096             });
2097         }
2098     });
2099 }
2100 } else {
2101     res.status(404).json({
2102         msg: "Class record status does not exist"
2103     });
2104 }
2105 });
2106 }
2107 } else {
2108     res.status(404).json({
2109         msg: "Subject section not found!"
2110     });
2111 }
2112 });
2113 }
2114 );
2115
2116 // @route POST api/teacher/getquartersummary
2117 // @desc Get quarter summary by subsectID and quarter
2118 // @access Private
2119
2120 router.post(
2121     "/getquartersummary",
2122     passport.authenticate("teacher", {
2123         session: false
2124     }),
2125     async(req, res) => {
2126         const {
2127             accountID
2128         } = req.user;
2129         const {
2130             subsectID,
2131             quarter
2132         } = req.body;
2133         const teacherID = await utils.getTeacherID(accountID);
2134         SubjectSection.findOne({
2135             where: {
2136                 subsectID
2137             }
2138         }).then(async subsect => {
2139             if (subsect) {
2140                 if (subsect.teacherID != teacherID) {
2141                     res.status(401).json({
2142                         msg: "Not authorized!"
2143                     });
2144                 } else {
2145                     const transmutation = await ClassRecord.findOne({
2146                         where: {
2147                             classRecordID: subsect.classRecordID
2148                         }
2149                     }).then(async cr => {
2150                         if (cr) {
2151                             switch (quarter) {
2152                                 case "Q1":
2153                                     {
2154                                         return cr.q1Transmu;
2155                                         break;
2156                                     }
2157                                 case "Q2":
2158                                     {
2159                                         return cr.q2Transmu;
2160                                         break;
2161                                     }
2162                                 case "Q3":
2163                                     {
2164                                         return cr.q3Transmu;
2165                                         break;
2166                                     }

```

```

2167         case "Q4":
2168             {
2169                 return cr.q4Transmu;
2170                 break;
2171             }
2172         default:
2173             break;
2174         }
2175     }
2176 });
2177 const subjectName = await utils.getSubjectName(subsect.
2178     subjectID);
2179 const subjectCode = await utils.getSubjectCode(subsect.
2180     subjectID);
2181 const sectionName = await utils.getSectionName(subsect.
2182     sectionID);
2183 const schoolYearID = subsect.schoolYearID;
2184 const schoolYear = await utils.getSYname(schoolYearID);
2185 const subsectstudIDs = await SubjectSectionStudent.findAll
2186     ({
2187         where: {
2188             subsectID
2189         }
2190     }).then(async sss => {
2191         if (sss.length == 0) {
2192             return [];
2193         } else {
2194             let data = [];
2195             for (const [index, value] of sss.entries()) {
2196                 data.push(value.subsectstudID);
2197             }
2198             return data;
2199         }
2200     });
2201 let data = [];
2202 for (const [index, value] of subsectstudIDs.entries()) {
2203     const temp = await StudentWeightedScore.findOne({
2204         where: {
2205             subsectstudID: value,
2206             quarter
2207         }
2208     }).then(async sws => {
2209         if (sws) {
2210             const {
2211                 subsectstudID,
2212                 faWS,
2213                 wwWS,
2214                 ptWS,
2215                 qeWS,
2216                 actualGrade,
2217                 transmutedGrade50,
2218                 transmutedGrade55,
2219                 transmutedGrade60
2220             } = sws;
2221             let finalGrade = 0;
2222             switch (transmutation) {
2223                 case "50":
2224                     {
2225                         finalGrade = Math.round(transmutedGrade50);
2226                         break;
2227                     }
2228                 case "55":
2229                     {
2230                         finalGrade = Math.round(transmutedGrade55);
2231                         break;
2232                     }
2233                 case "60":
2234                     {
2235                         finalGrade = Math.round(transmutedGrade60);
2236                         break;
2237                     }
2238             default:
2239                 break;

```

```

2236         }
2237         const name = await utils.
2238             getStudentNameBySubsectstudID(
2239                 subsectstudID
2240             );
2241         const imageUrl = await utils.
2242             getStudentImageUrlBySubsectstudID(
2243                 subsectstudID
2244             );
2245         const sex = await utils.getStudentSexBySubsectstudID(
2246             subsectstudID)
2247         return {
2248             sex,
2249             name,
2250             imageUrl,
2251             subsectstudID,
2252             faWS,
2253             wwWS,
2254             ptWS,
2255             qeWS,
2256             actualGrade,
2257             transmutedGrade50 ,
2258             transmutedGrade55 ,
2259             transmutedGrade60 ,
2260             finalGrade
2261         };
2262     });
2263     data.push(temp);
2264 }
2265 data.sort((a, b) => (a.name > b.name ? 1 : -1));
2266 res.status(200).json({
2267     transmutation,
2268     subjectName,
2269     subjectCode,
2270     sectionName,
2271     schoolYearID,
2272     schoolYear,
2273     classRecordID: subsect.classRecordID,
2274     data: [...data.filter(a => a.sex == "M"), ...data.filter(
2275         a => a.sex == "F")]
2276     });
2277 } else {
2278     res.status(404).json({
2279         msg: "Subject section not found!"
2280     });
2281 }
2282 );
2283 );
2284 );
2285 // @route api/teacher/changetransmutation
2286 // @desc Change transmutation by subsectID and quarter
2287 // @access Private
2288
2289 router.post(
2290     "/changetransmutation",
2291     passport.authenticate("teacher", {
2292         session: false
2293     }),
2294     async(req, res) => {
2295         const {
2296             subsectID,
2297             quarter,
2298             transmutation
2299         } = req.body;
2300         const {
2301             accountID
2302         } = req.user;
2303         const teacherID = await utils.getTeacherID(accountID);
2304         SubjectSection.findOne({
2305             where: {

```

```

2306         subsectID
2307     }
2308   }).then(async ss => {
2309     if (ss) {
2310       if (ss.teacherID != teacherID) {
2311         res.status(401).json({
2312           msg: "Not authorized!"
2313         });
2314     } else {
2315       ClassRecordStatus.findOne({
2316         where: {
2317           classRecordID: ss.classRecordID
2318         }
2319       }).then(async crs => {
2320         if (crs) {
2321           let invalid = false;
2322           if (quarter == "Q1") {
2323             invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 ==
2324             "F";
2325           } else if (quarter == "Q2") {
2326             invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 ==
2327             "F";
2328           } else if (quarter == "Q3") {
2329             invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 ==
2330             "F";
2331           } else {
2332             invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 ==
2333             "F";
2334           }
2335           if (invalid) {
2336             res.status(400).json({
2337               msg: "You are not authorized to edit this class
2338                 record."
2339             });
2340           } else {
2341             ClassRecord.findOne({
2342               where: {
2343                 classRecordID: ss.classRecordID
2344               }
2345             }).then(async cr => {
2346               if (cr) {
2347                 switch (quarter) {
2348                   case "Q1":
2349                     {
2350                       await cr.update({
2351                         q1Transmu: transmutation
2352                       },
2353                         {
2354                           showLog: false
2355                         });
2356                     break;
2357                   }
2358                   case "Q2":
2359                     {
2360                       await cr.update({
2361                         q2Transmu: transmutation
2362                       },
2363                         {
2364                           showLog: false
2365                         });
2366                     break;
2367                   }
2368                   case "Q3":
2369                     {
2370                       await cr.update({
2371                         q3Transmu: transmutation
2372                       },
2373                         {
2374                           showLog: false
2375                         });
2376                     break;
2377                   }
2378                   case "Q4":
2379                     {
2380                       await cr.update({
2381                         q4Transmu: transmutation
2382                       },
2383                         {
2384                           showLog: false
2385                         });
2386                     break;
2387                   }
2388                 }
2389               }
2390             }
2391           }
2392         }
2393       }
2394     }
2395   }
2396 }

```

```

2374             },
2375             showLog: false
2376         });
2377         break;
2378     }
2379     default:
2380     break;
2381 }
2382 await utils.
2383     refreshStudentWeightedScoreBySubsectID(
2384     subsectID
2385 );
2386 res.status(200).json({
2387     msg: "Transmutation changed successfully!"
2388 });
2389 } else {
2390     res.status(404).json({
2391     msg: "Class Record not found!"
2392 });
2393 }
2394 }
2395 } else {
2396     res.status(404).json({
2397     msg: "Subject Section not found!"
2398 });
2399 }
2400 });
2401 }
2402 } else {
2403     res.status(404).json({
2404     msg: "Class record status does not exist"
2405 });
2406 }
2407 });
2408 );
2409 );
2410
2411 // @route api/teacher/getrecinfo
2412 // @desc Get grade record info by one gradeID
2413 // @access Private
2414
2415 router.post(
2416     "/getrecinfo",
2417     passport.authenticate("teacher", {
2418         session: false
2419     }),
2420     async(req, res) => {
2421         const {
2422             accountID
2423         } = req.user;
2424         const {
2425             subsectID,
2426             componentID,
2427             subcompID,
2428             quarter,
2429             gradeID
2430         } = req.body;
2431         const teacherID = await utils.getTeacherID(accountID);
2432         SubjectSection.findOne({
2433             where: {
2434                 subsectID
2435             }
2436         }).then(async ss => {
2437             if (ss) {
2438                 if (ss.teacherID != teacherID) {
2439                     res.status(401).json({
2440                         msg: "Not authorized!"
2441                     });
2442                 } else {
2443                     Grade.findOne({
2444                         where: {
2445                             gradeID

```

```

2446
2447     }
2448   }).then(async grade => {
2449     if (grade) {
2450       const {
2451         description,
2452         dateGiven,
2453         total
2454       } = grade;
2455       const component = await utils.getComponentName(
2456         componentID);
2457       const weight = await Subcomponent.findOne({
2458         where: {
2459           subcompID
2460         }
2461       }).then(sc => {
2462         if (sc) {
2463           return sc.compWeight;
2464         }
2465       });
2466       let subcompName = await utils.getSubcomponentName(
2467         subcompID);
2468       subcompName = `${subcompName} (${weight}%)`;
2469       Grade.findAll({
2470         where: {
2471           quarter,
2472           subcomponentID: subcompID,
2473           componentID,
2474           description: grade.description,
2475           dateGiven: grade.dateGiven
2476         }
2477       }).then(async grades => {
2478         if (grades.length == 0) {
2479           res.status(404).json({
2480             msg: "Grades not found!"
2481           });
2482         } else {
2483           let data = [];
2484           for (const [index, value] of grades.entries()) {
2485             const {
2486               subsectstudID,
2487               score,
2488               gradeID,
2489               attendance
2490             } = value;
2491             const name = await utils.
2492               getStudentNameBySubsectstudID(
2493                 subsectstudID
2494               );
2495             const imageUrl = await utils.
2496               getStudentImageUrlBySubsectstudID(
2497                 subsectstudID
2498               );
2499             data.push({
2500               gradeID,
2501               subsectstudID,
2502               score: attendance == "A" ?
2503                 "A" : attendance == "E" ?
2504                   "E" : score,
2505               name,
2506               imageUrl
2507             });
2508           }
2509           res.status(200).json({
2510             componentID,
2511             component,
2512             subcompID,
2513             subcompName,
2514             dateGiven,
2515             total,
2516             description,
2517             data
2518           });
2519         }
2520       });
2521     }
2522   });

```

```

2516         } else {
2517             res.status(404).json({
2518                 msg: "Grade not found!"
2519             });
2520         }
2521     });
2522 }
2523 } else {
2524     res.status(404).json({
2525         msg: "Subject section not found!"
2526     });
2527 }
2528 });
2529 }
2530 );
2531
2532 // @route api/teacher/editrecord
2533 // @desc Edit record
2534 // @access Private
2535
2536 router.post(
2537     "/editrecord",
2538     passport.authenticate("teacher", {
2539         session: false
2540     }),
2541     async(req, res) => {
2542         const reg = /^-?[0-9]*(\.[0-9]*)?$/;
2543         const {
2544             description,
2545             dateGiven,
2546             total,
2547             subsectID,
2548             subcompID,
2549             componentID,
2550             quarter,
2551             payload
2552         } = req.body;
2553         let totalValid = !isNaN(total) && reg.test(total) && total !== ""
2554             && total !== "-";
2555         SubjectSection.findOne({
2556             where: {
2557                 subsectID
2558             }
2559         }).then(async ss => {
2560             if (ss) {
2561                 ClassRecordStatus.findOne({
2562                     where: {
2563                         classRecordID: ss.classRecordID
2564                     }
2565                 }).then(async crs => {
2566                     if (crs) {
2567                         let invalid = false;
2568                         if (quarter === "Q1") {
2569                             invalid = crs.q1 === "L" || crs.q1 === "D" || crs.q1 === "F";
2570                         } else if (quarter === "Q2") {
2571                             invalid = crs.q2 === "L" || crs.q2 === "D" || crs.q2 === "F";
2572                         } else if (quarter === "Q3") {
2573                             invalid = crs.q3 === "L" || crs.q3 === "D" || crs.q3 === "F";
2574                         } else {
2575                             invalid = crs.q4 === "L" || crs.q4 === "D" || crs.q4 === "F";
2576                         }
2577                         if (invalid) {
2578                             res.status(400).json({
2579                                 msg: "You are not authorized to edit this class
2580                                     record."
2581                             });
2582                         } else {
2583                             if (!totalValid) {
2584                                 res.status(400).json({

```

```

2583     msg: "Total score is invalid. Enter numbers only."
2584   });
2585 } else {
2586   const {
2587     accountID
2588   } = req.user;
2589   const teacherID = await utils.getTeacherID(accountID)
2590   ;
2591   if (total <= 0) {
2592     return res.status(400).json({
2593       msg: "Total must be more than 0"
2594     });
2595   } else {
2596     SubjectSection.findOne({
2597       where: {
2598         subsectID
2599       }
2600     }).then(async subjectsection => {
2601       if (subjectsection) {
2602         if (subjectsection.teacherID != teacherID) {
2603           return res.status(401).json({
2604             msg: "Not authorized!"
2605           });
2606         } else {
2607           const isSubmitted = await ClassRecord.findOne
2608             ({
2609               where: {
2610                 classRecordID: subjectsection.
2611                   classRecordID
2612               }
2613             }).then(cr => {
2614               if (cr) {
2615                 return cr.isSubmitted;
2616               }
2617             });
2618           let {
2619             errors,
2620             isValid
2621           } = validateSubcomponent({
2622             name: description
2623           });
2624           if (!isValid) {
2625             return res.status(400).json({
2626               msg: "Error input: Description"
2627             });
2628           } else {
2629             let i = 0;
2630             let invalid = false;
2631             let gradeIDchecker = [];
2632             for (const [index, value] of payload.
2633               entries()) {
2634               gradeIDchecker.push(value.gradeID);
2635             await Grade.findAll({
2636               where: {
2637                 componentID,
2638                 subcomponentID: subcompID,
2639                 quarter,
2640                 subsectstudID: value.subsectstudID,
2641                 gradeID: {
2642                   [Op.ne]: value.gradeID
2643                 }
2644               }
2645             }).then(async grades2 => {
2646               let sum = 0;
2647               let totalTemp = 0;
2648               for (const [
2649                 index2,
2650                 value2
2651               ] of grades2.entries()) {
2652                 if (value2.score == "A") {
2653                   sum = sum + 0;
2654                   totalTemp =
2655                     totalTemp + parseFloat(value2.

```

```

                total);
2652         } else if (value2.score == "E") {}
2653             else {
2654                 sum = sum + parseFloat(value2.score
2655                                         );
2656                 totalTemp =
2657                     totalTemp + parseFloat(value2.
2658                         total);
2659             }
2660         }
2661     if (
2662         sum + parseFloat(value.score) >
2663             totalTemp + parseFloat(total) &&
2664             !invalid
2665     ) {
2666         // invalid = true;
2667         // return res.status(400).json({
2668             //   msg:
2669             //     "Invalid score input. Score
2670             //       must accumulate to less than or
2671             //       equal to the total number of
2672             //       items."
2673         // });
2674     }
2675   );
2676 }
2677
2678 for (i = 0; i < payload.length; i++) {
2679   let {
2680     score,
2681     subsectstudID
2682   } = payload[i];
2683   let valid =
2684     (!isNaN(score) &&
2685       reg.test(score) &&
2686       score !== "" &&
2687       score !== "-") ||
2688       score == "A" ||
2689       score == "E";
2690   if (!valid) {
2691     invalid = true;
2692     return res.status(400).json({
2693       msg: "Invalid score input. Enter
2694           numbers, E, or B only"
2695     });
2696   }
2697
2698 await Grade.findOne({
2699   where: {
2700     description,
2701     dateGiven,
2702     componentID,
2703     subcomponentID: subcompID,
2704     classRecordID: subjectsection.
2705         classRecordID,
2706     quarter,
2707     gradeID: {
2708       [Op.notIn]: gradeIDchecker
2709     }
2710   }
2711 }).then(async gr => {
2712   if (gr) {
2713     res.status(400).json({
2714       msg: "Description already exists!"
2715     });
2716   } else {
2717     if (!invalid) {
2718       for (const [
2719         index,
2720         value3
2721       ] of payload.entries()) {
2722         Grade.findOne({
2723

```

```

2716                               where: {
2717                                 gradeID: value3.gradeID
2718                               }
2719                               }).then(async gr2 => {
2720                                 if (gr2) {
2721                                   const score = gr2.score;
2722                                   let attendance = "";
2723                                   if (value3.score == "A") {
2724                                     attendance = "A";
2725                                   } else if (value3.score == "E")
2726                                     {
2727                                       attendance = "E";
2728                                     } else {
2729                                       attendance = "P";
2730                                     }
2731                                   await gr2.update({
2732                                     description,
2733                                     total,
2734                                     dateGiven,
2735                                     attendance,
2736                                     score: value3.score,
2737                                     isUpdated: 1,
2738                                     showLog: isSubmitted
2739                                     },
2740                                     {
2741                                       showLog: false
2742                                     });
2743                               }
2744                               }
2745                               await utils.
2746                               refreshStudentWeightedScoreBySubsectID
2747                               (
2748                                 subsectID
2749                               );
2750                               res.status(200).json({
2751                                 msg: "Record updated successfully!"
2752                               });
2753                               }
2754                               });
2755                               }
2756                               }
2757                               } else {
2758                                 res.status(404).json({
2759                                   msg: "Subject section not found!"
2760                                 });
2761                               }
2762                               });
2763                               }
2764                               }
2765                               }
2766                               } else {
2767                                 res.status(404).json({
2768                                   msg: "Class record status does not exist"
2769                                 });
2770                               }
2771                               });
2772                               } else {
2773                                 res.status(404).json({
2774                                   msg: "Subject section does not exist"
2775                                 });
2776                               }
2777                               });
2778                               );
2779                               );
2780                               }
2781 // @route POST api/teacher/getsubjecttype
2782 // @desc Get subject type by subsectID
2783 // @access Private
2784
2785 router.post(

```

```

2786     "/getsubjecttype",
2787     passport.authenticate("teacher", {
2788       session: false
2789     }),
2790     async(req, res) => {
2791       const {
2792         subsectID
2793       } = req.body;
2794       SubjectSection.findOne({
2795         where: {
2796           subsectID
2797         }
2798       }).then(ss => {
2799         if (ss) {
2800           res.status(200).json({
2801             subjectType: ss.subjectType
2802           });
2803         } else {
2804           res.status(404).json({
2805             msg: "Subject section not found!"
2806           });
2807         }
2808       );
2809     }
2810   );
2811
2812 // @route POST api/teacher/ifclassrecordlocked
2813 // @desc Determine of the class record is locked by classRecordID and
2814 // quarter
2815 // @access Private
2816
2817 router.post(
2818   "/ifclassrecordlocked",
2819   passport.authenticate("teacher", {
2820     session: false
2821   }),
2822   async(req, res) => {
2823     const {
2824       classRecordID,
2825       quarter
2826     } = req.body;
2827     ClassRecordStatus.findOne({
2828       where: {
2829         classRecordID
2830       }
2831     }).then(async crs => {
2832       if (crs) {
2833         let invalid = false;
2834         let status = "E";
2835         if (quarter == "Q1") {
2836           status = crs.q1;
2837           invalid = crs.q1 == "L" || crs.q1 == "D" || crs.q1 == "F";
2838         } else if (quarter == "Q2") {
2839           status = crs.q2;
2840           invalid = crs.q2 == "L" || crs.q2 == "D" || crs.q2 == "F";
2841         } else if (quarter == "Q3") {
2842           status = crs.q3;
2843           invalid = crs.q3 == "L" || crs.q3 == "D" || crs.q3 == "F";
2844         } else {
2845           status = crs.q4;
2846           invalid = crs.q4 == "L" || crs.q4 == "D" || crs.q4 == "F";
2847         }
2848         res.status(200).json({
2849           locked: invalid,
2850           status
2851         });
2852       } else {
2853         res.status(404).json({
2854           msg: "Class record status does not exist"
2855         });
2856       }
2857     });

```

```

2858 );
2859 // @route POST api/teacher/getsummary
2860 // @desc Get summary by subsectID
2861 // @access Private
2862
2863 router.post(
2864   "/getsummary",
2865   passport.authenticate("teacher", {
2866     session: false
2867   }),
2868   async(req, res) => {
2869     const {
2870       accountID
2871     } = req.user;
2872     const {
2873       subsectID
2874     } = req.body;
2875     const teacherID = await utils.getTeacherID(accountID);
2876     SubjectSection.findOne({
2877       where: {
2878         subsectID
2879       }
2880     }).then(async subsect => {
2881       if (subsect) {
2882         if (subsect.teacherID != teacherID) {
2883           res.status(401).json({
2884             msg: "Not authorized!"
2885           });
2886         } else {
2887           const {
2888             q1Transmu,
2889             q2Transmu,
2890             q3Transmu,
2891             q4Transmu
2892           } = await ClassRecord.findOne({
2893             where: {
2894               classRecordID: subsect.classRecordID
2895             }
2896           });
2897           .then(async cr => {
2898             if (cr) {
2899               const {
2900                 q1Transmu,
2901                 q2Transmu,
2902                 q3Transmu,
2903                 q4Transmu
2904               } = cr;
2905               return {
2906                 q1Transmu,
2907                 q2Transmu,
2908                 q3Transmu,
2909                 q4Transmu
2910               };
2911             }
2912           });
2913           const subjectName = await utils.getSubjectName(subsect.
2914             subjectID);
2915           const subjectCode = await utils.getSubjectCode(subsect.
2916             subjectID);
2917           const sectionName = await utils.getSectionName(subsect.
2918             sectionID);
2919           const schoolYearID = subsect.schoolYearID;
2920           const schoolYear = await utils.getSYname(schoolYearID);
2921           StudentSubjectGrades.findAll({
2922             where: {
2923               classRecordID: subsect.classRecordID
2924             }
2925           }).then(async sg => {
2926             if (sg.length != 0) {
2927               let data = [];
2928               for (const [index, value] of sg.entries()) {
2929                 const {
2930                   subsectstudID,

```

```

2928             studsubjgradesID ,
2929             q1FinalGrade ,
2930             q2FinalGrade ,
2931             q3FinalGrade ,
2932             q4FinalGrade ,
2933             ave
2934         } = value;
2935         const name = await utils.
2936             getStudentNameBySubsectstudID(
2937                 subsectstudID
2938             );
2939         const sex = await utils.getStudentSexBySubsectstudID(
2940             subsectstudID)
2941         const imageUrl = await utils.
2942             getStudentImageUrlBySubsectstudID(
2943                 subsectstudID
2944             );
2945         data.push({
2946             sex ,
2947             subsectstudID ,
2948             studsubjgradesID ,
2949             q1FinalGrade ,
2950             q2FinalGrade ,
2951             q3FinalGrade ,
2952             q4FinalGrade ,
2953             ave ,
2954             name ,
2955             imageUrl
2956         });
2957     }
2958     data.sort((a , b) => (a.name > b.name ? 1 : -1));
2959     res.status(200).json({
2960         subjectName ,
2961         subjectCode ,
2962         sectionName ,
2963         schoolYear ,
2964         schoolYearID ,
2965         data: [...data.filter(a => a.sex == "M") , ...data.
2966             filter(a => a.sex == "F")],
2967             q1Transmu ,
2968             q2Transmu ,
2969             q3Transmu ,
2970             q4Transmu
2971         });
2972     } else {
2973         res.status(404).json({
2974             msg: "Student Grades not found!"
2975         });
2976     }
2977     });
2978   );
2979   );
2980 );
2981 );
2982 );
2983 );
2984 module.exports = router;

```

Listing 36: user.js

```

1  const express = require("express");
2  const router = express.Router();
3  const UserAccount = require("../models/UserAccount");
4  const Nonacademic = require("../models/Nonacademic");
5  const Teacher = require("../models/Teacher");
6  const Student = require("../models/Student");
7  const ParentGuardian = require("../models/ParentGuardian");

```

```
8  const Subject = require("../models/Subject");
9  const Component = require("../models/Component");
10 const SubjectSection = require('../models/SubjectSection')
11 const StudentSection = require('../models/StudentSection')
12 const SubjectSectionStudent = require('../models/
    SubjectSectionStudent')
13 const bcrypt = require("bcryptjs");
14 const jwt = require("jsonwebtoken");
15 const key = require("../config/key");
16 const passport = require("passport");
17 const multer = require("multer");
18 const fs = require("fs");
19
20 // Input Validation
21 const validateAdminCreateInput = require("../validation/register")
    ;
22 const validateLoginInput = require("../validation/login");
23 const validateChangePassword = require("../validation/password");
24 const isEmpty = require("../validation/is-empty");
25
26 // Import Utility Functions
27 const utils = require("../utils");
28
29 // @route POST api/users/createAdmin
30 // @desc Initializing admin account. Will be deleted afterwards
31 // @access Private
32
33 router.post("/createAdmin", (req, res) => {
34   const {
35     email,
36     password
37   } = req.body;
38   const {
39     errors,
40     isValid
41   } = validateAdminCreateInput(req.body);
42
43   // Validation Process
44   if (!isValid) {
45     return res.status(400).json(errors);
46   }
47
48   UserAccount.findOne({
49     where: {
50       email
51     }
52 }).then(user => {
53   if (user) {
54     errors.email = "Email already exists";
55     return res.status(400).json(errors);
56   } else {
57     bcrypt.genSalt(10, (err, salt) => {
58       bcrypt.hash(password, salt, (err, hash) => {
59         if (err) throw err;
60         utils
61           .createUser({
62             email,
63             password: hash,
64             isActive: 1,
65             position: 0
66           })
67           .then(user => {
68             utils
69               .createNonAcademic({
70                 accountID: user.accountID
71               })
72               .then(user => {
73                 res.json({
74                   msg: "Account created successfully!"
75                 });
76               });
77             });
78           });
79 });
80
81 module.exports = router;
```

```

79         });
80     });
81   });
82 });
83
84 // @route POST api/users/login
85 // @desc Login user account
86 // @access Public
87
88 router.post("/login", (req, res) => {
89   const {
90     email,
91     password
92   } = req.body;
93   const {
94     errors,
95     isValid
96   } = validateLoginInput(req.body);
97
98   if (!isValid) {
99     return res.status(400).json(errors);
100 }
101 if (email && password) {
102   UserAccount.findOne({
103     where: {
104       email
105     }
106   }).then(user => {
107     if (!user) {
108       errors.email = "User account not found!";
109       res.status(400).json(errors);
110     } else {
111       bcrypt.compare(password, user.password).then(isMatch => {
112         if (isMatch) {
113           switch (user.position) {
114             case false:
115             case true:
116             case 2:
117             case 6:
118               Nonacademic.findOne({
119                 where: {
120                   accountID: user.accountID
121                 }
122               }).then(async user2 => {
123                 const payload = {
124                   accountID: user.accountID,
125                   email: user.email,
126                   position: user.position,
127                   facultyID: user2.facultyID
128                 };
129
130                 if (!user.isActive) {
131                   errors.isActive =
132                     "Account is disabled. Please contact your
133                     school cashier for details.";
134                   errors.notice = await utils.
135                     getAccountNoticeByAccountID(user.accountID)
136                   res.status(400).json(errors);
137                 }
138
139                 // Sign Nonacademic Token
140                 jwt.sign(
141                   payload,
142                   key.secretKey, {
143                     expiresIn: "2 days"
144                   },
145                   (err, token) => {
146                     res.json({
147                       token: "Bearer " + token
148                     });
149                   });
150               });
151             }
152           }
153         }
154       });
155     }
156   });
157 });

```

```

150         break;
151     case 3:
152         Teacher.findOne({
153             where: {
154                 accountID: user.accountID
155             }
156         }).then(async user2 => {
157             const payload = {
158                 accountID: user.accountID,
159                 email: user.email,
160                 position: user.position,
161                 teacherID: user2.teacherID,
162                 isAdviser: user2.isAdviser
163             };
164
165             if (!user.isActive) {
166                 errors.isActive =
167                     "Account is disabled. Please contact your
168                     school cashier for details.";
169                 errors.notice = await utils.
170                     getAccountNoticeByAccountID(user.accountID)
171                 res.status(400).json(errors);
172             }
173             const {
174                 schoolYearID
175             } = await utils.getActiveSY()
176             SubjectSection.findOne({
177                 where: {
178                     teacherID: payload.teacherID,
179                     schoolYearID
180                 }
181             }).then(ss => {
182                 if (ss) {
183                     jwt.sign(
184                         payload,
185                         key.secretKey,
186                         { expiresIn: "2 days"
187                         },
188                         (err, token) => {
189                             res.json({
190                                 token: "Bearer " + token
191                             });
192                         });
193                 } else {
194                     res.status(400).json({
195                         msg: "You are not currently assigned to any
196                         subject for teaching. Wait for your
197                         school registrar to assign subject load
198                         for you to access the system."
199                     });
200                 }
201             }
202         })
203         // Sign Teacher Token
204     );
205     break;
206     case 4:
207         Student.findOne({
208             where: {
209                 accountID: user.accountID
210             }
211         }).then(async user2 => {
212             const payload = {
213                 accountID: user.accountID,
214                 email: user.email,
215                 position: user.position,
216                 studentID: user2.studentID
217             };
218
219             if (!user.isActive) {
220                 errors.isActive =
221                     "Account is disabled. Please contact your
222                     school cashier for details.";

```

```

218         errors.notice = await utils.
219             getAccountNoticeByAccountID(user.accountID)
220         res.status(400).json(errors);
221     }
222     const {
223         schoolYearID
224     } = await utils.getActiveSY()
225     StudentSection.findOne({
226         where: {
227             schoolYearID,
228             studentID: payload.studentID
229         }
230     }).then(async ss => {
231         if (ss) {
232             SubjectSectionStudent.findOne({
233                 where: {
234                     studsectID: ss.studsectID
235                 }
236             }).then(ss2 => {
237                 if (ss2) {
238                     // Sign Student Token
239                     jwt.sign(
240                         payload,
241                         key.secretKey, {
242                             expiresIn: "2 days"
243                         },
244                         (err, token) => {
245                             res.json({
246                                 token: "Bearer " + token
247                             });
248                         }
249                     );
250                 } else {
251                     res.status(400).json({
252                         msg: "You are not currently enrolled to
253                             any subject this school year. Wait
254                             for your school registrar to enroll
255                             you in a subject."
256                         });
257                     }
258                 });
259             }
260         }
261     });
262     break;
263 case 5:
264     ParentGuardian.findOne({
265         where: {
266             accountID: user.accountID
267         }
268     }).then(async user2 => {
269         const payload = {
270             accountID: user.accountID,
271             email: user.email,
272             position: user.position,
273             parentID: user2.parentID
274         };
275
276         if (!user.isActive) {
277             errors.isActive =
278                 "Account is disabled. Please contact your
279                 school cashier for details.";
280             errors.notice = await utils.
281                 getAccountNoticeByAccountID(user.accountID)
282             res.status(400).json(errors);

```

```

282         }
283
284         // Sign ParentGuardian Token
285         jwt.sign(
286             payload,
287             key.secretKey, {
288                 expiresIn: "2 days"
289             },
290             (err, token) => {
291                 res.json({
292                     token: "Bearer " + token
293                 });
294             }
295         );
296     } ;
297     break;
298 default:
299     res.json(400);
300 }
301 } else {
302     errors.password = "Password incorrect";
303     return res.status(400).json(errors);
304 }
305 });
306 }
307 });
308 }
309 });
310
311 // @route GET api/users/profile
312 // @desc Get all personal information
313 // @access Private
314
315 router.get(
316     "/profile",
317     passport.authenticate("jwt", {
318         session: false
319     }),
320     (req, res) => {
321         const {
322             accountID,
323             position
324         } = req.user;
325         UserAccount.findOne({
326             where: {
327                 accountID
328             }
329         }).then(user => {
330             const payload = {
331                 firstName: user.firstName,
332                 lastName: user.lastName,
333                 middleName: user.middleName,
334                 suffix: user.suffix,
335                 nickname: user.nickname,
336                 imageUrl: user.imageUrl,
337                 contactNum: user.contactNum,
338                 address: user.address,
339                 province: user.province,
340                 city: user.city,
341                 region: user.region,
342                 zipcode: user.zipcode,
343                 civilStatus: user.civilStatus,
344                 sex: user.sex,
345                 citizenship: user.citizenship,
346                 birthDate: user.birthDate,
347                 birthPlace: user.birthPlace,
348                 religion: user.religion,
349                 emergencyName: user.emergencyName,
350                 emergencyAddress: user.emergencyAddress,
351                 emergencyTelephone: user.emergencyTelephone,
352                 emergencyCellphone: user.emergencyCellphone,
353                 emergencyEmail: user.emergencyEmail,
354                 emergencyRelationship: user.emergencyRelationship
355             };

```

```

356     if (
357         position == false ||
358         position == true ||
359         position == 2 ||
360         position == 6
361     ) {
362         // Get Nonacademic's Profile
363         Nonacademic.findOne({
364             where: {
365                 accountID
366             }
367         }).then(user2 => {
368             res.json({
369                 ...payload,
370                 facultyID: user2.faultyID
371             });
372         });
373     } else if (position == 3) {
374         // Get Teacher's Profile
375         Teacher.findOne({
376             where: {
377                 accountID
378             }
379         }).then(user2 => {
380             res.json({
381                 ...payload,
382                 teacherID: user2.teacherID,
383                 isActive: user2.isAdviser
384             });
385         });
386     } else if (position == 4) {
387         // Get Student's Profile
388         Student.findOne({
389             where: {
390                 accountID
391             }
392         }).then(user2 => {
393             res.json({
394                 ...payload,
395                 studentID: user2.studentID,
396                 fatherName: user2.fatherName,
397                 fatherAddress: user2.fatherAddress,
398                 fatherEmail: user2.fatherEmail,
399                 fatherOccupation: user2.fatherOccupation,
400                 fatherEmployer: user2.fatherEmployer,
401                 fatherBusinessAdd: user2.fatherBusinessAdd,
402                 fatherOfficeNum: user2.fatherOfficeNum,
403                 motherName: user2.motherName,
404                 motherAddress: user2.motherAddress,
405                 motherEmail: user2.motherEmail,
406                 motherOccupation: user2.motherOccupation,
407                 motherEmployer: user2.motherEmployer,
408                 motherBusinessAdd: user2.motherBusinessAdd,
409                 motherOfficeNum: user2.motherOfficeNum
410             });
411         });
412     } else {
413         // Get Parent's Profile
414         ParentGuardian.findOne({
415             where: {
416                 accountID
417             }
418         }).then(user2 => {
419             res.json({
420                 ...payload,
421                 parentID: user2.parentID,
422                 studentID: user2.studentID
423             });
424         });
425     });
426 };
427 );
428

```

```

429 // @route POST api/users/upload
430 // @desc Upload photo
431 // @access Private
432
433 const storage = multer.diskStorage({
434   destination: (req, file, cb) => {
435     cb(null, './public/images');
436   },
437   filename: (req, file, cb) => {
438     var filetype = "";
439     if (file.mimetype === "image/gif") {
440       filetype = "gif";
441     }
442     if (file.mimetype === "image/png") {
443       filetype = "png";
444     }
445     if (file.mimetype === "image/jpeg") {
446       filetype = "jpg";
447     }
448     cb(null, "image-" + Date.now() + "." + filetype);
449   }
450 });
451
452 const upload = multer({
453   storage
454 });
455 router.post(
456   "/upload",
457   upload.single("file"),
458   passport.authenticate("jwt", {
459     session: false
460   }),
461   (req, res) => {
462     if (!req.file) {
463       res.status(500);
464     } else {
465       const {
466         accountID
467       } = req.user;
468       UserAccount.findOne({
469         where: {
470           accountID
471         }
472       }).then(user => {
473         try {
474           fs.unlinkSync(`./public/${user.imageUrl}`);
475         } catch (err) {
476           console.log(err);
477         }
478         user
479           .update({
480             imageUrl: "images/" + req.file.filename
481           })
482           .then(result => {
483             res.status(200).json({
484               msg: "Success"
485             });
486           });
487         });
488       }
489     }
490   );
491
492 // @route POST api/users/changepassword
493 // @desc Change password
494 // @access Private
495
496 router.post(
497   "/changepassword",
498   passport.authenticate("jwt", {
499     session: false
500   }),

```

```

501     (req, res) => {
502       const {
503         password,
504         currentPassword
505       } = req.body;
506       const {
507         errors,
508         isValid
509       } = validateChangePassword(req.body);
510
511       UserAccount.findOne({
512         where: {
513           accountID: req.user.accountID
514         }
515       }).then(user => {
516         bcrypt.compare(currentPassword, user.password).then(isMatch =>
517         {
518           if (isMatch) {
519             if (!isValid) {
520               return res.status(400).json(errors);
521             }
522             bcrypt.genSalt(10, (err, salt) => {
523               bcrypt.hash(password, salt, (err, hash) => {
524                 if (err) throw err;
525                 user.update({
526                   password: hash
527                 }).then(() => {
528                   res.status(200).json({
529                     msg: "Password updated successfully!"
530                   });
531                 });
532               });
533             } else {
534               errors.currentPassword = isEmpty(currentPassword) ?
535                 "Current password field is required" :
536                 "Password incorrect";
537               return res.status(400).json(errors);
538             }
539           });
540         });
541       );
542     );
543
544     // @route GET api/users/populateweights
545     // @desc Populate component weights for all defined subjects
546     // @access Public
547
548     router.get("/populateweights", (req, res) => {
549       const nonSHSLanguageIDs = [
550         1,
551         2,
552         3,
553         4,
554         5,
555         6,
556         7,
557         8,
558         9,
559         10,
560         11,
561         12,
562         13,
563         14,
564         15,
565         16,
566         17,
567         18,
568         19,
569         20,
570         21,
571         22,
572         23
573       ];

```

```

574     const nonSHSAPIDs = [42, 43, 44, 45, 46, 47, 48, 49, 50, 51];
575     const nonSHSEsPIDs = [52, 53, 54, 55, 56, 57, 58, 59, 60, 61];
576     const nonSHSScienceIDs = [34, 35, 36, 37, 38, 39, 40, 41];
577     const nonSHSMathIDs = [24, 25, 26, 27, 28, 29, 30, 31, 32, 33];
578     const nonSHSMAPEHIDs = [
579         62,
580         63,
581         64,
582         65,
583         66,
584         67,
585         68,
586         69,
587         70,
588         71,
589         72,
590         73,
591         74,
592         75,
593         76,
594         77,
595         78,
596         79,
597         80,
598         81,
599         82,
600         83,
601         84,
602         85,
603         86,
604         87,
605         88,
606         89,
607         90,
608         91,
609         92,
610         93,
611         94,
612         95,
613         96,
614         97,
615         98,
616         99,
617         100,
618         101
619     ];
620     const nonSHSEPPTLEIDs = [102, 103, 104, 105, 106, 107, 108];
621     const SHSCoreIDs = [
622         109,
623         110,
624         111,
625         112,
626         113,
627         114,
628         115,
629         116,
630         117,
631         118,
632         119,
633         120,
634         121,
635         122,
636         123,
637         124,
638         125
639     ];
640     const SHSAcadSpecializedIDs = [];
641     const SHSAcadAppliedIDs = [];
642     const SHSTVLSpecializedIDs = [];
643     const SHSTVLApliedIDs = [];
644     const SHSSportsSpecializedIDs = [];
645     const SHSSportsAppliedIDs = [];
646     const SHSAADSpecializedIDs = [];
647     const SHSAADAppliedIDs = [];
648
649     let i = 0;
650     for (i = 0; i < nonSHSLanguageIDs.length; i++) {

```

```

651     Component.create({
652       subjectID: nonSHSLanguageIDs[i],
653       component: "FA",
654       compWeight: 0
655     });
656     Component.create({
657       subjectID: nonSHSLanguageIDs[i],
658       component: "WW",
659       compWeight: 30
660     });
661     Component.create({
662       subjectID: nonSHSLanguageIDs[i],
663       component: "PT",
664       compWeight: 50
665     });
666     Component.create({
667       subjectID: nonSHSLanguageIDs[i],
668       component: "QE",
669       compWeight: 20
670     });
671   }
672   for (i = 0; i < nonSHSAPIDs.length; i++) {
673     Component.create({
674       subjectID: nonSHSAPIDs[i],
675       component: "FA",
676       compWeight: 0
677     });
678     Component.create({
679       subjectID: nonSHSAPIDs[i],
680       component: "WW",
681       compWeight: 30
682     });
683     Component.create({
684       subjectID: nonSHSAPIDs[i],
685       component: "PT",
686       compWeight: 50
687     });
688     Component.create({
689       subjectID: nonSHSAPIDs[i],
690       component: "QE",
691       compWeight: 20
692     });
693   }
694   for (i = 0; i < nonSHSEsPidS.length; i++) {
695     Component.create({
696       subjectID: nonSHSEsPidS[i],
697       component: "FA",
698       compWeight: 0
699     });
700     Component.create({
701       subjectID: nonSHSEsPidS[i],
702       component: "WW",
703       compWeight: 30
704     });
705     Component.create({
706       subjectID: nonSHSEsPidS[i],
707       component: "PT",
708       compWeight: 50
709     });
710     Component.create({
711       subjectID: nonSHSEsPidS[i],
712       component: "QE",
713       compWeight: 20
714     });
715   }
716   for (i = 0; i < nonSHSScienceIDs.length; i++) {
717     Component.create({
718       subjectID: nonSHSScienceIDs[i],
719       component: "FA",
720       compWeight: 0
721     });
722     Component.create({
723       subjectID: nonSHSScienceIDs[i],

```

```

724         component: "WW",
725         compWeight: 40
726     });
727     Component.create({
728         subjectID: nonSHSScienceIDs[i],
729         component: "PT",
730         compWeight: 40
731     });
732     Component.create({
733         subjectID: nonSHSScienceIDs[i],
734         component: "QE",
735         compWeight: 20
736     });
737 }
738 for (i = 0; i < nonSHSMathIDs.length; i++) {
739     Component.create({
740         subjectID: nonSHSMathIDs[i],
741         component: "FA",
742         compWeight: 0
743     });
744     Component.create({
745         subjectID: nonSHSMathIDs[i],
746         component: "WW",
747         compWeight: 40
748     });
749     Component.create({
750         subjectID: nonSHSMathIDs[i],
751         component: "PT",
752         compWeight: 40
753     });
754     Component.create({
755         subjectID: nonSHSMathIDs[i],
756         component: "QE",
757         compWeight: 20
758     });
759 }
760 for (i = 0; i < nonSHSMAPEHIDs.length; i++) {
761     Component.create({
762         subjectID: nonSHSMAPEHIDs[i],
763         component: "FA",
764         compWeight: 0
765     });
766     Component.create({
767         subjectID: nonSHSMAPEHIDs[i],
768         component: "WW",
769         compWeight: 20
770     });
771     Component.create({
772         subjectID: nonSHSMAPEHIDs[i],
773         component: "PT",
774         compWeight: 60
775     });
776     Component.create({
777         subjectID: nonSHSMAPEHIDs[i],
778         component: "QE",
779         compWeight: 20
780     });
781 }
782 for (i = 0; i < nonSHSEPPTLEIDs.length; i++) {
783     Component.create({
784         subjectID: nonSHSEPPTLEIDs[i],
785         component: "FA",
786         compWeight: 0
787     });
788     Component.create({
789         subjectID: nonSHSEPPTLEIDs[i],
790         component: "WW",
791         compWeight: 20
792     });
793     Component.create({
794         subjectID: nonSHSEPPTLEIDs[i],
795         component: "PT",
796         compWeight: 60

```

```

797     });
798     Component.create({
799         subjectID: nonSHSEPPTLEIDs[i],
800         component: "QE",
801         compWeight: 20
802     });
803 }
804 for (i = 0; i < SHSCoreIDs.length; i++) {
805     Component.create({
806         subjectID: SHSCoreIDs[i],
807         component: "FA",
808         compWeight: 0
809     });
810     Component.create({
811         subjectID: SHSCoreIDs[i],
812         component: "WW",
813         compWeight: 25
814     });
815     Component.create({
816         subjectID: SHSCoreIDs[i],
817         component: "PT",
818         compWeight: 50
819     });
820     Component.create({
821         subjectID: SHSCoreIDs[i],
822         component: "QE",
823         compWeight: 25
824     });
825 }
826 for (i = 0; i < SHSAcadSpecializedIDs.length; i++) {
827     Component.create({
828         subjectID: SHSAcadSpecializedIDs[i],
829         component: "FA",
830         compWeight: 0
831     });
832     Component.create({
833         subjectID: SHSAcadSpecializedIDs[i],
834         component: "WW",
835         compWeight: 25
836     });
837     Component.create({
838         subjectID: SHSAcadSpecializedIDs[i],
839         component: "PT",
840         compWeight: 45
841     });
842     Component.create({
843         subjectID: SHSAcadSpecializedIDs[i],
844         component: "QE",
845         compWeight: 30
846     });
847 }
848 for (i = 0; i < SHSAcadAppliedIDs.length; i++) {
849     Component.create({
850         subjectID: SHSAcadAppliedIDs[i],
851         component: "FA",
852         compWeight: 0
853     });
854     Component.create({
855         subjectID: SHSAcadAppliedIDs[i],
856         component: "WW",
857         compWeight: 35
858     });
859     Component.create({
860         subjectID: SHSAcadAppliedIDs[i],
861         component: "PT",
862         compWeight: 40
863     });
864     Component.create({
865         subjectID: SHSAcadAppliedIDs[i],
866         component: "QE",
867         compWeight: 25
868     });
869 }

```

```

870     for (i = 0; i < SHSTVLSpecializedIDs.length; i++) {
871         Component.create({
872             subjectID: SHSTVLSpecializedIDs[i],
873             component: "FA",
874             compWeight: 0
875         });
876         Component.create({
877             subjectID: SHSTVLSpecializedIDs[i],
878             component: "WW",
879             compWeight: 20
880         });
881         Component.create({
882             subjectID: SHSTVLSpecializedIDs[i],
883             component: "PT",
884             compWeight: 60
885         });
886         Component.create({
887             subjectID: SHSTVLSpecializedIDs[i],
888             component: "QE",
889             compWeight: 20
890         });
891     }
892     for (i = 0; i < SHSTVLApliedIDs.length; i++) {
893         Component.create({
894             subjectID: SHSTVLApliedIDs[i],
895             component: "FA",
896             compWeight: 0
897         });
898         Component.create({
899             subjectID: SHSTVLApliedIDs[i],
900             component: "WW",
901             compWeight: 20
902         });
903         Component.create({
904             subjectID: SHSTVLApliedIDs[i],
905             component: "PT",
906             compWeight: 60
907         });
908         Component.create({
909             subjectID: SHSTVLApliedIDs[i],
910             component: "QE",
911             compWeight: 20
912         });
913     }
914     for (i = 0; i < SHSSportsAppliedIDs.length; i++) {
915         Component.create({
916             subjectID: SHSSportsAppliedIDs[i],
917             component: "FA",
918             compWeight: 0
919         });
920         Component.create({
921             subjectID: SHSSportsAppliedIDs[i],
922             component: "WW",
923             compWeight: 20
924         });
925         Component.create({
926             subjectID: SHSSportsAppliedIDs[i],
927             component: "PT",
928             compWeight: 60
929         });
930         Component.create({
931             subjectID: SHSSportsAppliedIDs[i],
932             component: "QE",
933             compWeight: 20
934         });
935     }
936     for (i = 0; i < SHSSportsSpecializedIDs.length; i++) {
937         Component.create({
938             subjectID: SHSSportsSpecializedIDs[i],
939             component: "FA",
940             compWeight: 0
941         });
942         Component.create({

```

```

943     subjectID: SHSSportsSpecializedIDs[i],
944     component: "WW",
945     compWeight: 20
946   });
947   Component.create({
948     subjectID: SHSSportsSpecializedIDs[i],
949     component: "PT",
950     compWeight: 60
951   });
952   Component.create({
953     subjectID: SHSSportsSpecializedIDs[i],
954     component: "QE",
955     compWeight: 20
956   });
957 }
958 for (i = 0; i < SHSAADAppliedIDs.length; i++) {
959   Component.create({
960     subjectID: SHSAADAppliedIDs[i],
961     component: "FA",
962     compWeight: 0
963   });
964   Component.create({
965     subjectID: SHSAADAppliedIDs[i],
966     component: "WW",
967     compWeight: 20
968   });
969   Component.create({
970     subjectID: SHSAADAppliedIDs[i],
971     component: "PT",
972     compWeight: 60
973   });
974   Component.create({
975     subjectID: SHSAADAppliedIDs[i],
976     component: "QE",
977     compWeight: 20
978   });
979 }
980 for (i = 0; i < SHSAADSpecializedIDs.length; i++) {
981   Component.create({
982     subjectID: SHSAADSpecializedIDs[i],
983     component: "FA",
984     compWeight: 0
985   });
986   Component.create({
987     subjectID: SHSAADSpecializedIDs[i],
988     component: "WW",
989     compWeight: 20
990   });
991   Component.create({
992     subjectID: SHSAADSpecializedIDs[i],
993     component: "PT",
994     compWeight: 60
995   });
996   Component.create({
997     subjectID: SHSAADSpecializedIDs[i],
998     component: "QE",
999     compWeight: 20
1000 });
1001 }
1002 });
1003
1004 module.exports = router;

```

Listing 37: utilities/index.js

```

1 const UserAccount = require("../models/UserAccount");
2 const Nonacademic = require("../models/Nonacademic");
3 const SchoolYear = require("../models/SchoolYear");
4 const Section = require("../models/Section");
5 const Student = require("../models/Student");

```

```

6   const StudentSection = require("../models/StudentSection");
7   const SubjectSection = require("../models/SubjectSection");
8   const SubjectSectionStudent = require("../models/
      SubjectSectionStudent");
9   const StudentWeightedScore = require("../models/StudentWeightedScore"
    );
10  const StudentSubjectGrades = require("../models/StudentSubjectGrades"
    );
11  const ClassRecord = require("../models/ClassRecord");
12  const Component = require("../models/Component");
13  const Subcomponent = require("../models/Subcomponent");
14  const TeacherSection = require("../models/TeacherSection");
15  const Teacher = require("../models/Teacher");
16  const Sequelize = require("sequelize");
17  const Subject = require("../models/Subject");
18  const Grade = require("../models/Grade");
19  const ClassRecordStatus = require("../models/ClassRecordStatus");
20  const StudentGrades = require("../models/StudentGrades");
21  const StudentFinalGrade = require("../models/StudentFinalGrade");
22  const AccountNotice = require('../models/AccountNotice')
23  const logo = require("../assets/pdf/logo");
24  const Op = Sequelize.Op;
25
26 // Utility Functions
27 exports.getPHTime = function(date = "") {
28   const d = date === "" ? new Date() : new Date(date);
29   const localOffset = d.getTimezoneOffset() * 60000;
30   const UTC = d.getTime() + localOffset;
31   const PHT = UTC + 3600000 * 16;
32   return new Date(PHT);
33 };
34
35 exports.createUser = async({
36   email,
37   password,
38   isActive,
39   position
40 }) => {
41   const na = "NA";
42   return await UserAccount.create({
43     email,
44     password,
45     isActive,
46     position,
47     firstName: na,
48     lastName: na,
49     middleName: na,
50     suffix: na,
51     nickname: na,
52     imageUrl: na,
53     contactNum: na,
54     address: na,
55     province: na,
56     city: na,
57     region: na,
58     zipcode: na,
59     civilStatus: "SINGLE",
60     sex: "M",
61     citizenship: na,
62     birthDate: 0,
63     birthPlace: na,
64     religion: na,
65     emergencyName: na,
66     emergencyAddress: na,
67     emergencyTelephone: na,
68     emergencyCellphone: na,
69     emergencyEmail: na,
70     emergencyRelationship: na
71   });
72 };
73
74 exports.getSubjectNameBySubsectstudID = async subsectstudID => {
75   return await SubjectSectionStudent.findOne({
76     where: {
77       subsectstudID

```

```

78         }
79     }).then(
80         async sss => {
81             return await SubjectSection.findOne({
82                 where: {
83                     subsectID: sss.subsectID
84                 }
85             }).then(async ss => {
86                 return await this.getSubjectName(ss.subjectID);
87             });
88         );
89     );
90 };
91
92 exports.getStudentSexByStudentId = async studentID => {
93     return await Student.findOne({
94         where: {
95             studentID
96         }
97     }).then(async student => {
98         return await UserAccount.findOne({
99             where: {
100                 accountID: student.accountID
101             }
102         }).then(user => {
103             return user.sex;
104         });
105     );
106 };
107
108 exports.createNonAcademic = async({
109     accountID
110 }) => {
111     return await Nonacademic.create({
112     accountID
113 });
114 };
115
116 exports.capitalize = string => {
117     return string;
118 };
119
120 const capitalize = string => {
121     return string;
122 };
123
124 exports.getAccountName = async accountID => {
125     return await UserAccount.findOne({
126         where: {
127             accountID
128         }
129     }).then(user => {
130         return `${capitalize(user.lastName)}, ${capitalize(
131             user.firstName
132         )}${user.middleName.charAt(0).toUpperCase()}.`;
133     });
134 };
135
136 exports.getTeacherName = async teacherID => {
137     return await Teacher.findOne({
138         where: {
139             teacherID
140         }
141     }).then(teacher => {
142         return UserAccount.findOne({
143             where: {
144                 accountID: teacher.accountID
145             }
146         }).then(user => {
147             return `${capitalize(user.lastName)}, ${capitalize(
148                 user.firstName
149             )}${user.middleName.charAt(0).toUpperCase()}.`;

```

```

150      });
151  });
152 };
153
154 exports.getAccountIDBySubsectstudID = async subsectstudID => {
155   return await SubjectSectionStudent.findOne({
156     where: {
157       subsectstudID
158     }
159   }).then(
160     async ss => {
161       if (ss) {
162         return await StudentSection.findOne({
163           where: {
164             studsectID: ss.studsectID
165           }
166         }).then(async ss => {
167           if (ss) {
168             return await Student.findOne({
169               where: {
170                 studentID: ss.studentID
171               }
172             }).then(async s => {
173               if (s) {
174                 return s.accountID;
175               }
176             });
177           }
178         );
179       }
180     );
181   );
182 };
183
184 exports.getNonacademicName = async facultyID => {
185   return await Nonacademic.findOne({
186     where: {
187       facultyID
188     }
189   }).then(
190     nonacademic => {
191       return UserAccount.findOne({
192         where: {
193           accountID: nonacademic.accountID
194         }
195       }).then(user => {
196         return `${capitalize(user.lastName)}, ${capitalize(
197           user.firstName
198         )} ${user.middleName.charAt(0).toUpperCase()}.`;
199       });
200     );
201   );
202 };
203
204 exports.getStudentName = async studentID => {
205   return await Student.findOne({
206     where: {
207       studentID
208     }
209   }).then(student => {
210     return UserAccount.findOne({
211       where: {
212         accountID: student.accountID
213       }
214     }).then(user => {
215       return `${capitalize(user.lastName)}, ${capitalize(
216         user.firstName
217       )} ${user.middleName.charAt(0).toUpperCase()}.`;
218     });
219   );
220 };
221

```

```

222     exports.getStudentEmail = async studentID => {
223         return await Student.findOne({
224             where: {
225                 studentID
226             }
227         }).then(student => {
228             return UserAccount.findOne({
229                 where: {
230                     accountID: student.accountID
231                 }
232             }).then(user => {
233                 return user.email;
234             });
235         });
236     };
237
238     exports.getSubjectCode = async subjectID => {
239         return await Subject.findOne({
240             where: {
241                 subjectID
242             }
243         }).then(subject => {
244             return subject.subjectCode;
245         });
246     };
247
248     exports.getSubjectName = async subjectID => {
249         return await Subject.findOne({
250             where: {
251                 subjectID
252             }
253         }).then(subject => {
254             return subject.subjectName;
255         });
256     };
257
258     exports.getSectionName = async sectionID => {
259         return await Section.findOne({
260             where: {
261                 sectionID
262             }
263         }).then(section => {
264             return section.sectionName;
265         });
266     };
267
268     exports.getSectionGradeLevel = async sectionID => {
269         return await Section.findOne({
270             where: {
271                 sectionID
272             }
273         }).then(section => {
274             return section.gradeLevel;
275         });
276     };
277
278     exports.displayPosition = position => {
279         switch (position) {
280             case false:
281                 return "Administrator";
282             case true:
283                 return "Director";
284             case 2:
285                 return "Registrar";
286             case 3:
287                 return "Teacher";
288             case 4:
289                 return "Student";
290             case 5:
291                 return "Guardian";
292             case 6:
293                 return "Cashier";
294             default:
295                 return "";

```

```

296     }
297   };
298
299   exports.getActiveSY = async() => {
300     return await SchoolYear.findOne({
301       where: {
302         isActive: 1
303       }
304     }).then(
305       schoolYear => {
306         if (schoolYear) {
307           const {
308             schoolYearID,
309             quarter
310           } = schoolYear;
311           return {
312             schoolYearID,
313             quarter
314           };
315         } else return {
316           schoolYearID: 0,
317           quarter: "Q1"
318         };
319       }
320     );
321   };
322
323   exports.getStudentSectionBySYAndSectionID = async({
324     sectionID,
325     schoolYearID,
326     page,
327     pageSize,
328     keyword
329   }) => {
330     page = page - 1;
331     let offset = page * pageSize;
332     let limit = offset + pageSize;
333     let studentIDs = [];
334     let numOfPages = 0;
335
336     await UserAccount.findAll({
337       where: {
338         position: 4,
339         [Op.or]: [
340           {
341             firstName: {
342               [Op.like]: `%"${keyword}"%
343             }
344           },
345           {
346             lastName: {
347               [Op.like]: `%"${keyword}"%
348             }
349           }
350         ]
351       }
352     }).then(async useraccounts => {
353       if (useraccounts.length != 0) {
354         var i = 0;
355         for (i; i < useraccounts.length; i++) {
356           let studentID = await this.getStudentID(useraccounts[i].accountID);
357           if (studentID != "") {
358             studentIDs.push(studentID);
359           }
360         }
361       }
362     });
363     return await StudentSection.findAll({
364       limit,
365       offset,
366       where: {
367         sectionID,
368         schoolYearID,
369         studentID: studentIDs
370       }
371     }
372   );

```

```

368     })
369     .then(async studentsections => {
370       let data = [];
371       if (studentsections.length != 0) {
372         var i = 0;
373         for (i; i < studentsections.slice(0, pageSize).length; i++) {
374           let sectionName = await this.getSectionName(
375             studentsections[i].sectionID
376           );
377           let studentName = await this.getStudentName(
378             studentsections[i].studentID
379           );
380           let studentID = studentsections[i].studentID;
381           let imageUrl = await this.getStudentImageUrl(
382             studentsections[i].studentID
383           );
384           let email = await this.getStudentEmail(studentsections[i].
385             studentID);
386           data.push({
387             sectionName,
388             studentName,
389             studentID,
390             email,
391             imageUrl
392           });
393         await StudentSection.findAndCountAll({
394           where: {
395             sectionID,
396             schoolYearID,
397             studentID: studentIDs
398           }
399         }).then(count => {
400           data.sort((a, b) => (a.studentName > b.studentName ? 1 :
401             -1));
402           numOfPages = Math.ceil(count.count / pageSize);
403         });
404         return {
405           studentData: data,
406           numOfPages
407         };
408       } else {
409         return {
410           studentData: [],
411           numOfPages: 1
412         };
413       }
414     .catch(err => {
415       return {
416         studentData: [],
417         numOfPages: 1
418       };
419     });
420   };
421
422 exports.getSectionsIDByGradeLevel = async gradeLevel => {
423   return await Section.findAll({
424     where: {
425       gradeLevel
426     }
427   }).then(
428     async sections => {
429       if (sections.length == 0) {
430         return [];
431       } else {
432         let sectionData = [];
433         await sections.forEach(async(section, key, arr) => {
434           sectionData.push(section.sectionID);
435         });
436         return sectionData;
437       }
438     }
439   );
440 }

```

```

438         }
439     );
440   };
441
442   exports.getStudentImageUrl = async studentID => {
443     return await Student.findOne({
444       where: {
445         studentID
446       }
447     }).then(async student => {
448       if (student) {
449         return await UserAccount.findOne({
450           where: {
451             accountID: student.accountID
452           }
453         }).then(user => {
454           if (user) {
455             return user.imageUrl;
456           } else {
457             return "";
458           }
459         });
460       } else {
461         return "";
462       }
463     });
464   };
465
466   exports.getTeacherSectionBySchoolYearID = async schoolYearID => {
467     return await TeacherSection.findAll({
468       where: {
469         schoolYearID
470       }
471     }).then(
472       async advisers => {
473         if (advisers.length == 0) {
474           return [];
475         } else {
476           let advisersData = [];
477           await advisers.forEach(async(adviser, key, arr) => {
478             let sectionID = adviser.sectionID;
479             let teacherName = await this.getTeacherName(adviser.
480               teacherID);
481             let teacherID = adviser.teacherID;
482             advisersData.push({
483               sectionID,
484               teacherName,
485               teacherID
486             });
487           });
488           return advisersData;
489         }
490       );
491     };
492
493   exports.getSectionsID = async({
494     limit,
495     offset,
496     keyword
497   }) => {
498     return await Section.findAll({
499       limit,
500       offset,
501       where: {
502         archived: 0,
503         sectionName: {
504           [Op.like]: `%"${keyword}"%
505         }
506       }
507     }).then(async sections => {
508       if (sections.length == 0) {
509         return [];

```

```

510         } else {
511             let sectionData = [];
512             await sections.forEach(async(section, key, arr) => {
513                 sectionData.push(section.sectionID);
514             });
515             return sectionData;
516         }
517     });
518 };
519
520 exports.getGradeLevelBySectionID = async sectionID => {
521     return await Section.findOne({
522         where: {
523             sectionID
524         }
525     }).then(section => {
526         if (section) {
527             return section.gradeLevel;
528         } else {
529             return "";
530         }
531     });
532 };
533
534 exports.getSYID = async schoolYear => {
535     return await SchoolYear.findOne({
536         where: {
537             schoolYear
538         }
539     }).then(sy => {
540         if (sy) return sy.schoolYearID;
541         else return 0;
542     });
543 };
544
545 const getActiveSY = async() => {
546     return await SchoolYear.findOne({
547         where: {
548             isActive: 1
549         }
550     }).then(
551         schoolYear => {
552             if (schoolYear) return schoolYear.schoolYearID;
553             else return 0;
554         }
555     );
556 };
557
558 exports.getPastSY = async() => {
559     let currSyID = await getActiveSY();
560     let currSy = await getSYname(currSyID);
561     return (
562         (parseInt(currSy.slice(0, 4)) - 1).toString() +
563         "-" +
564         (parseInt(currSy.slice(5, 9)) - 1).toString()
565     );
566 };
567
568 exports.getSYname = async schoolYearID => {
569     return await SchoolYear.findOne({
570         where: {
571             schoolYearID
572         }
573     }).then(
574         schoolYear => {
575             if (schoolYear) return schoolYear.schoolYear;
576             else return "";
577         }
578     );
579 };
580
581 const getSYname = async schoolYearID => {

```

```

582     return await SchoolYear.findOne({
583         where: {
584             schoolYearID
585         }
586     }).then(
587         schoolYear => {
588             if (schoolYear) return schoolYear.schoolYear;
589             else return "";
590         }
591     );
592 };
593
594 exports.getStudentID = async accountID => {
595     return await Student.findOne({
596         where: {
597             accountID
598         }
599     }).then(student => {
600         if (student) return student.studentID;
601         else return "";
602     });
603 };
604
605 exports.getTeacherID = async accountID => {
606     return await Teacher.findOne({
607         where: {
608             accountID
609         }
610     }).then(teacher => {
611         if (teacher) return teacher.teacherID;
612         else return "";
613     });
614 };
615
616 exports.getStudentsIDByKeyword = async keyword => {
617     return await UserAccount.findAll({
618         where: {
619             position: 4,
620             [Op.or]: [
621                 {
622                     firstName: {
623                         [Op.like]: `%"${keyword}"%
624                     }
625                 },
626                 {
627                     lastName: {
628                         [Op.like]: `%"${keyword}"%
629                     }
630                 }
631             ]
632         }
633     }).then(async students => {
634         if (students.length == 0) {
635             return [];
636         } else {
637             var i = 0;
638             let studarr = [];
639             for (i = 0; i < students.length; i++) {
640                 const studentID = await this.getStudentID(students[i].accountID);
641                 studarr.push(studentID);
642             }
643             return studarr;
644         }
645     });
646
647     exports.getStudentNameByStudsectID = async studsectID => {
648         return await StudentSection.findOne({
649             where: {
650                 studsectID
651             }
652         }).then(
653             async studentsection => {
654                 if (studentsection) {

```

```

653         return await this.getStudentName(studentsection.studentID);
654     } else {
655         return "";
656     }
657 }
658 );
659 };
660
661 exports.getStudentSexBySubsectstudID = async subsectstudID => {
662     return await SubjectSectionStudent.findOne({
663         where: {
664             subsectstudID
665         }
666     }).then(
667         async sss => {
668             return await this.getStudentSexByStudsectID(sss.studsectID);
669         }
670     );
671 };
672
673 exports.getStudentSexByStudsectID = async studsectID => {
674     return await StudentSection.findOne({
675         where: {
676             studsectID
677         }
678     }).then(
679         async studentsection => {
680             if (studentsection) {
681                 return await this.getStudentSexByStudentID(studentsection.
682                     studentID);
683             } else {
684                 return "";
685             }
686         }
687     );
688 };
689
690 exports.getStudentEmailByStudsectID = async studsectID => {
691     return await StudentSection.findOne({
692         where: {
693             studsectID
694         }
695     }).then(
696         async studentsection => {
697             if (studentsection) {
698                 return await this.getStudentEmail(studentsection.studentID);
699             } else {
700                 return "";
701             }
702         }
703     );
704 };
705
706 exports.getStudentImageUrlByStudsectID = async studsectID => {
707     return await StudentSection.findOne({
708         where: {
709             studsectID
710         }
711     }).then(
712         async studentsection => {
713             if (studentsection) {
714                 return await this.getStudentImageUrl(studentsection.studentID
715                     );
716             } else {
717                 return "";
718             }
719         }
720     );
721
722 exports.getSectionNameByStudsectID = async studsectID => {
723     return await StudentSection.findOne({

```

```

723         where: {
724             studsectID
725         }
726     }).then(
727         async studentsection => {
728             if (studentsection) {
729                 return await this.getSectionName(studentsection.sectionID);
730             } else {
731                 return "";
732             }
733         }
734     );
735 };
736
737 exports.getAccountNoticeByAccountID = async accountID => {
738     return await AccountNotice.findOne({
739         where: {
740             accountID
741         }
742     }).then(an => an.message)
743 }
744
745 exports.getSectionGradeLevelByStudsectID = async studsectID => {
746     return await StudentSection.findOne({
747         where: {
748             studsectID
749         }
750     }).then(
751         async studentsection => {
752             if (studentsection) {
753                 return await this.getSectionGradeLevel(studentsection.
754                     sectionID);
755             } else {
756                 return "";
757             }
758         }
759     );
760 };
761
762 exports.getStudentNameBySubsectstudID = async subsectstudID => {
763     return await SubjectSectionStudent.findOne({
764         where: {
765             subsectstudID
766         }
767     }).then(
768         async subsectstud => {
769             if (subsectstud) {
770                 return await StudentSection.findOne({
771                     where: {
772                         studsectID: subsectstud.studsectID
773                     }
774                 }).then(async studsect => {
775                     if (studsect) {
776                         return await this.getStudentName(studsect.studentID);
777                     } else {
778                         return "";
779                     }
780                 });
781             } else {
782                 return "";
783             }
784         }
785     );
786 };
787
788 exports.getComponentName = async componentID => {
789     return await Component.findOne({
790         where: {
791             componentID
792         }
793     }).then(
794         async comp => {

```

```

794     if (comp) {
795         return comp.component == "FA" ?
796             "Formative Assessment" :
797             comp.component == "WW" ?
798                 "Written Works" :
799                 comp.component == "PT" ?
800                     "Performance Task" :
801                     comp.component == "QE" ?
802                         "Quarterly Exam" :
803                         "";
804     } else {
805         return "";
806     }
807 }
808 );
809 };
810
811 exports.getSubcomponentName = async subcompID => {
812     return await Subcomponent.findOne({
813         where: {
814             subcompID
815         }
816     }).then(
817         async subcomp => {
818             if (subcomp) {
819                 return subcomp.name;
820             } else {
821                 return "";
822             }
823         }
824     );
825 };
826
827 exports.getStudentImageUrlBySubsectstudID = async subsectstudID => {
828     return await SubjectSectionStudent.findOne({
829         where: {
830             subsectstudID
831         }
832     }).then(
833         async subsectstud => {
834             if (subsectstud) {
835                 return await StudentSection.findOne({
836                     where: {
837                         studsectID: subsectstud.studsectID
838                     }
839                 }).then(async studsect => {
840                     if (studsect) {
841                         return await this.getStudentImageUrl(studsect.studentID);
842                     } else {
843                         return "";
844                     }
845                 });
846             } else {
847                 return "";
848             }
849         }
850     );
851 };
852
853 exports.getSubcompWsBySubsectstudIDAndSubcompIDAndQuarter = async({
854     subcompID,
855     subsectstudID,
856     quarter
857 }) => {
858     return await Grade.findAll({
859         where: {
860             subcomponentID: subcompID,
861             subsectstudID,
862             quarter
863         }
864     }).then(async grades => {
865         if (grades.length == 0) {
866             return -1;

```

```

867     } else {
868         let score = 0;
869         let i = 0;
870         let total = 0;
871         let ps = 0;
872         let excused = 0;
873         for (i = 0; i < grades.length; i++) {
874             if (grades[i].attendance == "E") {
875                 excused = excused + 1;
876             } else if (grades[i].attendance == "A") {
877                 score = score + 0;
878                 total = total + parseFloat(grades[i].total);
879             } else {
880                 score = score + parseFloat(grades[i].score);
881                 total = total + parseFloat(grades[i].total);
882             }
883         }
884         if (grades.length != 0) {
885             if (grades.length == excused) {
886                 ps = -1;
887             } else {
888                 ps = (score / total) * 100;
889             }
890         }
891         return ps;
892     }
893 });
894 };
895
896 exports.refreshStudentWeightedScoreBySubsectID = async subsectID => {
897     const {
898         subjectID,
899         classRecordID,
900         subjectType
901     } = await SubjectSection.findOne({
902         where: {
903             subsectID
904         }
905     }).then(subsect => {
906         if (subsect) {
907             const {
908                 subjectID,
909                 classRecordID,
910                 subjectType
911             } = subsect;
912             return {
913                 subjectID,
914                 classRecordID,
915                 subjectType
916             };
917         } else {
918             return -1;
919         }
920     });
921     return await SubjectSectionStudent.findAll({
922         where: {
923             subsectID
924         }
925     }).then(
926         async subsectstud => {
927             if (subsectstud.length == 0) {
928                 return false;
929             } else {
930                 let q =
931                     subjectType == "NON_SHS" ? ["Q1", "Q2", "Q3", "Q4"] : ["Q1",
932                     , "Q2"];
933                 let subsectstudIDs = [];
934                 let i = 0;
935                 for (i = 0; i < subsectstud.length; i++) {
936                     let {
937                         subsectstudID
938                     } = subsectstud[i];
939                     subsectstudIDs.push(subsectstudID);

```

```

939     }
940     Component.findOne({
941       where: {
942         subjectID,
943         component: "FA"
944       }
945     }).then(async comp1 => {
946       if (comp1) {
947         Component.findOne({
948           where: {
949             subjectID,
950             component: "WW"
951           }
952         }).then(async comp2 => {
953           if (comp2) {
954             Component.findOne({
955               where: {
956                 subjectID,
957                 component: "PT"
958               }
959             }).then(async comp3 => {
960               if (comp3) {
961                 Component.findOne({
962                   where: {
963                     subjectID,
964                     component: "QE"
965                   }
966                 }).then(async comp4 => {
967                   if (comp4) {
968                     //Refreshing all FA records
969                     await Subcomponent.findAll({
970                       where: {
971                         componentID: comp1.componentID,
972                         classRecordID
973                       }
974                     }).then(async fasc => {
975                       if (fasc.length != 0) {
976                         for (const [
977                           index3,
978                           value3
979                         ] of subsectstudIDs.entries()) {
980                           for (const [index2, value2] of q.
981                             entries()) {
982                             let faData = [];
983                             let hasNegativeOne = false;
984                             for (const [index, value] of fasc.
985                               entries()) {
986                               if (value.quarter == value2) {
987                                 const {
988                                   compWeight
989                                 } = value;
990                                 const ws = await this.
991                                   getSubcompWsBySubsectstudIDAndSubcompIDAndQ
992                                   ({
993                                     subcompID: value.subcompID,
994                                     subsectstudID: value3,
995                                     quarter: value2
996                                   });
997                                 if (ws == -1) {
998                                   if (parseFloat(compWeight) !=
999                                     0)
999                                     hasNegativeOne = true;
999                                   }
999                                 faData.push({
999                                   compWeight,
999                                   ws
999                                 });
999                               }
999                             }
999                           }
999                         }
999                       }
999                     if (!hasNegativeOne) {
999                       let sum = 0;
999                       for (const [
999                         index,
999                         value
999                         ] of subsectstudIDs.entries()) {
999                           sum += value.quarter;
999                         }
999                         }
999                       }
999                     }
999                   }
999                 }
999               }
999             }
999           }
999         }
999       }
999     }
999   }
999 
```



```

1075     });
1076     if (ws == -1) {
1077         if (parseFloat(compWeight) != 0)
1078             hasNegativeOne = true;
1079     }
1080     wwData.push({
1081         compWeight,
1082         ws
1083     });
1084 }
1085 }
1086 if (!hasNegativeOne) {
1087     let sum = 0;
1088     for (const [
1089         index,
1090         value4
1091     ] of wwData.entries()) {
1092         sum =
1093             sum +
1094                 parseFloat(value4.compWeight)
1095                     / 100) *
1096                 parseFloat(value4.ws);
1097 }
1098 StudentWeightedScore.findOne({
1099     where: {
1100         subsectstudID: value3,
1101         quarter: value2,
1102         classRecordID
1103     }
1104 }).then(async sws => {
1105     if (sws) {
1106         sws
1107             .update({
1108                 wwWS: sum
1109             })
1110             .then(async () => {});
1111     }
1112 });
1113 else {
1114     StudentWeightedScore.findOne({
1115         where: {
1116             subsectstudID: value3,
1117             quarter: value2,
1118             classRecordID
1119         }
1120     }).then(async sws => {
1121         if (sws) {
1122             sws
1123                 .update({
1124                     wwWS: -1
1125                 })
1126                 .then(async () => {});
1127             }
1128         });
1129     }
1130   }
1131 });
1132 });
1133 });

//Refreshing all PT records
1134 await Subcomponent.findAll({
1135     where: {
1136         componentID: comp3.componentID,
1137         classRecordID
1138     }
1139 }).then(async ptsc => {
1140     if (ptsc.length != 0) {
1141         for (const [
1142             index3,
1143             value3
1144         ] of subsectstudIDs.entries()) {
1145

```

```

1146     for (const [index2, value2] of q.
1147         entries()) {
1148             let ptdata = [];
1149             let hasNegativeOne = false;
1150             for (const [index, value] of ptsc.
1151                 entries()) {
1152                 if (value.quarter == value2) {
1153                     const {
1154                         compWeight
1155                     } = value;
1156                     const ws = await this.
1157                         getSubcompWsBySubsectstudIDAndSubcompIDAndQ
1158                         ({
1159                             subcompID: value.subcompID,
1160                             subsectstudID: value3,
1161                             quarter: value2
1162                         });
1163                         if (ws == -1) {
1164                             if (parseFloat(compWeight) != 0)
1165                                 hasNegativeOne = true;
1166                         ptdata.push({
1167                             compWeight,
1168                             ws
1169                         });
1170                     }
1171                     if (!hasNegativeOne) {
1172                         let sum = 0;
1173                         for (const [
1174                             index,
1175                             value4
1176                         ] of ptdata.entries()) {
1177                             sum =
1178                                 sum +
1179                                     (parseFloat(value4.compWeight)
1180                                         / 100) *
1181                                         parseFloat(value4.ws);
1182                     }
1183                     StudentWeightedScore.findOne({
1184                         where: {
1185                             subsectstudID: value3,
1186                             quarter: value2,
1187                             classRecordID
1188                         }
1189                     }).then(async sws => {
1190                         if (sws) {
1191                             sws
1192                                 .update({
1193                                     ptWS: sum
1194                                 })
1195                                 .then(async () => {});
1196                         }
1197                     } else {
1198                         StudentWeightedScore.findOne({
1199                             where: {
1200                                 subsectstudID: value3,
1201                                 quarter: value2,
1202                                 classRecordID
1203                             }
1204                         }).then(async sws => {
1205                             if (sws) {
1206                                 sws
1207                                     .update({
1208                                         ptWS: -1
1209                                     })
1210                                     .then(async () => {});
1211                             }
1212                         }
1213                     }
1214                 }
1215             }
1216         }
1217     }

```

```

1213         }
1214     }
1215   });
1216
1217   //Refreshing all QE records
1218   await Subcomponent.findAll({
1219     where: {
1220       componentID: comp4.componentID,
1221       classRecordID
1222     }
1223   }).then(async qesc => {
1224     if (qesc.length != 0) {
1225       for (const [
1226         index3,
1227         value3
1228       ] of subsectstudIDs.entries()) {
1229         for (const [index2, value2] of q.
1230           entries()) {
1231           let qeData = [];
1232           let hasNegativeOne = false;
1233           for (const [index, value] of qesc.
1234             entries()) {
1235             if (value.quarter == value2) {
1236               const {
1237                 compWeight
1238               } = value;
1239               const ws = await this.
1240                 getSubcompWsBySubsectstudIDAndSubcompIDAndQ
1241                 ({
1242                   subcompID: value.subcompID,
1243                   subsectstudID: value3,
1244                   quarter: value2
1245                 });
1246               if (ws == -1) {
1247                 if (parseFloat(compWeight) != 0)
1248                   hasNegativeOne = true;
1249               }
1250             }
1251           if (!hasNegativeOne) {
1252             let sum = 0;
1253             for (const [
1254               index,
1255               value4
1256             ] of qeData.entries()) {
1257               sum =
1258                 sum +
1259                 (parseFloat(value4.compWeight)
1260                   / 100) *
1261                   parseFloat(value4.ws);
1262             }
1263             StudentWeightedScore.findOne({
1264               where: {
1265                 subsectstudID: value3,
1266                 quarter: value2,
1267                 classRecordID
1268               }
1269             }).then(async sws => {
1270               if (sws) {
1271                 sws
1272                   .update({
1273                     qeWS: sum
1274                   })
1275                   .then(async () => {});
1276               }
1277             });
1278           } else {
1279             StudentWeightedScore.findOne({

```

```

1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
        where: {
          subsectstudID: value3,
          quarter: value2,
          classRecordID
        }
      }).then(async sws => {
        if (sws) {
          sws
            .update({
              qeWS: -1
            })
            .then(async () => {});
        }
      });
    }
  );
}

for (const [index5, value5] of q.entries()) {
  for (const [
    index6,
    value6
  ] of subsectstudIDs.entries()) {
    await StudentWeightedScore.findOne({
      where: {
        subsectstudID: value6,
        quarter: value5
      }
    }).then(async sws2 => {
      if (sws2) {
        await this.studentWeightedScoreUpdate(
          (
            sws2.weightedScoreID
          );
        }
      }
    });

    await StudentSubjectGrades.findOne({
      where: {
        subsectstudID: value6
      }
    }).then(async sg => {
      if (sg) {
        await this.studentGradesUpdate(
          sg.studsubjgradesID,
          subjectType
        );
      }
    });
  }
} else {
  res
    .status(404)
    .json({
      msg: "QE component not found!"
    });
}
};

} else {
  res.status(404).json({
    msg: "PT component not found!"
  });
}
};

} else {
  res.status(404).json({
    msg: "WW component not found!"
  });
}
}

```

```

1351         });
1352     } else {
1353       res.status(404).json({
1354         msg: "FA component not found!"
1355       });
1356     }
1357   });
1358 }
1359 );
1360 );
1361 };
1362
1363 exports.formatCondensedDeliberationGrade = data => {
1364   let flags = [],
1365     flags2 = [],
1366     name = [],
1367     subjectName = [],
1368     output = [];
1369   for (const [i, v] of data.entries()) {
1370     for (const [i2, v2] of v.entries()) {
1371       if (flags[v2.name]) continue;
1372       flags[v2.name] = true;
1373       name.push(v2.name);
1374     }
1375     for (const [i2, v2] of v.entries()) {
1376       if (flags2[v2.subjectName]) continue;
1377       flags2[v2.subjectName] = true;
1378       subjectName.push({
1379         subjectName: v2.subjectName,
1380         status: v2.status,
1381         classRecordID: v2.classRecordID,
1382         teacher: v2.teacher
1383       });
1384     }
1385   }
1386   for (const [i, v] of name.entries()) {
1387     let name = v,
1388       grades = [];
1389     let temparr = data.find(a => a[0].name == v);
1390     let studsectID = temparr[0].studsectID;
1391     let sex = temparr[0].sex;
1392     let studentID = temparr[0].studentID;
1393     let imageUrl = temparr[0].imageUrl;
1394     for (const [i2, v2] of subjectName.entries()) {
1395       let grade =
1396         typeof temparr.find(a => a.subjectName == v2.subjectName) ===
1397         "undefined" ?
1398           "N/A" :
1399             temparr.find(a => a.subjectName == v2.subjectName).score;
1400       grades.push({
1401         subjectName: v2.subjectName,
1402         grade,
1403       });
1404     }
1405     output.push({
1406       imageUrl,
1407       studentID,
1408       sex,
1409       name,
1410       grades,
1411       studsectID
1412     });
1413   }
1414   output.sort((a, b) => (a.name > b.name ? 1 : -1));
1415   return {
1416     data: output,
1417     columns: subjectName
1418   };
1419 };
1420
1421 exports.refreshStudentGradesBySubsectID = async subsectID => {
1422   await SubjectSectionStudent.findAll({

```

```

1422     where: {
1423       subsectID
1424     }
1425   }).then(async sss => {
1426     if (sss) {
1427       sss.forEach(async(x, k, r) => {
1428         for (const [index, q] of ["Q1", "Q2", "Q3", "Q4"].entries()) {
1429           const grades = await SubjectSectionStudent.findAll({
1430             where: {
1431               studsectID: x.studsectID
1432             }
1433           }).then(async sss2 => {
1434             if (sss2) {
1435               let data2 = [];
1436               for (const [i, v] of sss2.entries()) {
1437                 let {
1438                   subjectType,
1439                   classRecordID
1440                 } = await SubjectSection.findOne({
1441                   where: {
1442                     subsectID: v.subsectID
1443                   }
1444                 }).then(async ss => {
1445                   if (ss) {
1446                     return {
1447                       subjectType: ss.subjectType,
1448                       classRecordID: ss.classRecordID
1449                     };
1450                   }
1451                 });
1452               let status = await ClassRecordStatus.findOne({
1453                 where: {
1454                   classRecordID
1455                 }
1456               }).then(async crs => {
1457                 if (crs) {
1458                   if (q == "Q1") {
1459                     if (
1460                       subjectType == "NON_SHS" ||
1461                       subjectType == "1ST_SEM"
1462                     ) {
1463                       return crs.q1;
1464                     }
1465                   } else if (q == "Q2") {
1466                     if (
1467                       subjectType == "NON_SHS" ||
1468                       subjectType == "1ST_SEM"
1469                     ) {
1470                       return crs.q2;
1471                     }
1472                   } else if (q == "Q3") {
1473                     if (subjectType == "NON_SHS") {
1474                       return crs.q3;
1475                     } else if (subjectType == "2ND_SEM") {
1476                       return crs.q1;
1477                     }
1478                   } else {
1479                     if (subjectType == "NON_SHS") {
1480                       return crs.q4;
1481                     } else if (subjectType == "2ND_SEM") {
1482                       return crs.q2;
1483                     }
1484                   }
1485                 }
1486               });
1487             if (typeof status !== "undefined") {
1488               let score = await StudentSubjectGrades.findOne({
1489                 where: {
1490                   classRecordID,
1491                   subsectstudID: v.subsectstudID
1492                 }
1493               }).then(ssg => {

```

```

1494         if (ssg) {
1495             if (status == "F") {
1496                 if (q == "Q1") {
1497                     if (
1498                         subjectType == "NON_SHS" ||
1499                         subjectType == "1ST_SEM"
1500                     ) {
1501                         return ssg.q1FinalGrade;
1502                     }
1503                 } else if (q == "Q2") {
1504                     if (
1505                         subjectType == "NON_SHS" ||
1506                         subjectType == "1ST_SEM"
1507                     ) {
1508                         return ssg.q2FinalGrade;
1509                     }
1510                 } else if (q == "Q3") {
1511                     if (subjectType == "NON_SHS") {
1512                         return ssg.q3FinalGrade;
1513                     } else if (subjectType == "2ND_SEM") {
1514                         return ssg.q1FinalGrade;
1515                     }
1516                 } else {
1517                     if (subjectType == "NON_SHS") {
1518                         return ssg.q4FinalGrade;
1519                     } else if (subjectType == "2ND_SEM") {
1520                         return ssg.q2FinalGrade;
1521                     }
1522                 }
1523             }
1524         }
1525     );
1526     if (typeof score !== "undefined") {
1527         data2.push(score);
1528     }
1529 }
1530 }
1531 return data2;
1532 }
1533 });
1534 await StudentGrades.findOne({
1535     where: {
1536         studsectID: x.studsectID,
1537         quarter: q
1538     }
1539 }).then(async sg => {
1540     if (sg) {
1541         if (grades.length == 0) {
1542             await sg.update({
1543                 grade: -1
1544             });
1545         } else {
1546             let av = 0;
1547             let denom = 0;
1548             for (const [i2, v2] of grades.entries()) {
1549                 if (v2 != -1) {
1550                     denom = denom + 1;
1551                     av = av + parseFloat(v2);
1552                 }
1553             }
1554             if (denom != 0) {
1555                 av = parseFloat(av / denom);
1556                 await sg.update({
1557                     grade: av
1558                 });
1559             } else {
1560                 await sg.update({
1561                     grade: -1
1562                 });
1563             }
1564         }
1565     }
}

```

```

1566         });
1567     }
1568
1569     await StudentGrades.findAll({
1570       where: {
1571         studsectID: x.studsectID
1572       }
1573     }).then(async sg2 => {
1574       if (sg2) {
1575         let invalid = false;
1576         let sum = 0;
1577         for (const [index, value] of sg2.entries()) {
1578           if (value.grade == -1) {
1579             invalid = true;
1580           } else {
1581             sum = sum + parseFloat(value.grade);
1582           }
1583         }
1584         await StudentFinalGrade.findOne({
1585           where: {
1586             studsectID: x.studsectID
1587           }
1588         }).then(async sfg => {
1589           if (sfg) {
1590             if (!invalid) {
1591               let ave = parseFloat(sum / 4);
1592               await sfg.update({
1593                 grade: ave
1594               });
1595             } else {
1596               await sfg.update({
1597                 grade: -1
1598               });
1599             }
1600           }
1601         });
1602       }
1603     });
1604   );
1605 }
1606 );
1607 };
1608 exports.getSubjectIDBySubsectstudID = async subsectstudID => {
1609   return await SubjectSectionStudent.findOne({
1610     where: {
1611       subsectstudID
1612     }
1613   }).then(
1614     async sss => {
1615       if (sss) {
1616         return await SubjectSection.findOne({
1617           where: {
1618             subsectID: sss.subsectID
1619           }
1620         }).then(async ss => {
1621           if (ss) {
1622             return ss.subjectID;
1623           } else {
1624             return -1;
1625           }
1626         });
1627       } else {
1628         return -1;
1629       }
1630     }
1631   );
1632 };
1633
1634 exports.studentGradesUpdate = async(studsubjgradesID, subjectType) =>
1635   {
1636     return await StudentSubjectGrades.findOne({
1637       where: {

```

```

1637         studsubjgradesID
1638     }
1639   }).then(async sg => {
1640     if (sg) {
1641       const q1Transmutation = await ClassRecord.findOne({
1642         where: {
1643           classRecordID: sg.classRecordID
1644         }
1645       }).then(async cr => {
1646         if (cr) {
1647           return cr.q1Transmu;
1648         }
1649       });
1650       const q2Transmutation = await ClassRecord.findOne({
1651         where: {
1652           classRecordID: sg.classRecordID
1653         }
1654       }).then(async cr => {
1655         if (cr) {
1656           return cr.q2Transmu;
1657         }
1658       });
1659       const q3Transmutation = await ClassRecord.findOne({
1660         where: {
1661           classRecordID: sg.classRecordID
1662         }
1663       }).then(async cr => {
1664         if (cr) {
1665           return cr.q3Transmu;
1666         }
1667       });
1668       const q4Transmutation = await ClassRecord.findOne({
1669         where: {
1670           classRecordID: sg.classRecordID
1671         }
1672       }).then(async cr => {
1673         if (cr) {
1674           return cr.q4Transmu;
1675         }
1676       });
1677       const q1TransmuGrade = await StudentWeightedScore.findOne({
1678         where: {
1679           classRecordID: sg.classRecordID,
1680           subsectstudID: sg.subsectstudID,
1681           quarter: "Q1"
1682         }
1683       }).then(async sws => {
1684         if (sws) {
1685           switch (q1Transmutation) {
1686             case "50":
1687               {
1688                 return sws.transmutedGrade50;
1689                 break;
1690               }
1691             case "55":
1692               {
1693                 return sws.transmutedGrade55;
1694                 break;
1695               }
1696             case "60":
1697               {
1698                 return sws.transmutedGrade60;
1699                 break;
1700               }
1701             default:
1702               break;
1703             }
1704           }
1705         });
1706       const q2TransmuGrade = await StudentWeightedScore.findOne({
1707         where: {
1708           classRecordID: sg.classRecordID,

```

```

1710         subsectstudID: sg.subsectstudID ,
1711         quarter: "Q2"
1712     }
1713 }).then(async sws => {
1714     if (sws) {
1715         switch (q2Transmutation) {
1716             case "50":
1717                 {
1718                     return sws.transmutedGrade50;
1719                     break;
1720                 }
1721             case "55":
1722                 {
1723                     return sws.transmutedGrade55;
1724                     break;
1725                 }
1726             case "60":
1727                 {
1728                     return sws.transmutedGrade60;
1729                     break;
1730                 }
1731             default:
1732                 break;
1733             }
1734         }
1735     );
1736
1737     const q3TransmuGrade = await StudentWeightedScore.findOne({
1738         where: {
1739             classRecordID: sg.classRecordID ,
1740             subsectstudID: sg.subsectstudID ,
1741             quarter: "Q3"
1742         }
1743     }).then(async sws => {
1744         if (sws) {
1745             switch (q3Transmutation) {
1746                 case "50":
1747                     {
1748                         return sws.transmutedGrade50;
1749                         break;
1750                     }
1751                 case "55":
1752                     {
1753                         return sws.transmutedGrade55;
1754                         break;
1755                     }
1756                 case "60":
1757                     {
1758                         return sws.transmutedGrade60;
1759                         break;
1760                     }
1761             default:
1762                 break;
1763             }
1764         }
1765     );
1766
1767     const q4TransmuGrade = await StudentWeightedScore.findOne({
1768         where: {
1769             classRecordID: sg.classRecordID ,
1770             subsectstudID: sg.subsectstudID ,
1771             quarter: "Q4"
1772         }
1773     }).then(async sws => {
1774         if (sws) {
1775             switch (q4Transmutation) {
1776                 case "50":
1777                     {
1778                         return sws.transmutedGrade50;
1779                         break;
1780                     }
1781                 case "55":
1782                     {
1783                         return sws.transmutedGrade55;

```

```

1784             break;
1785         }
1786         case "60":
1787         {
1788             return sws.transmutedGrade60;
1789             break;
1790         }
1791         default:
1792             break;
1793         }
1794     }
1795 });
1796
1797 if (parseFloat(q1TransmuGrade) != -1) {
1798     await sg.update({
1799         q1FinalGrade: Math.round(parseFloat(q1TransmuGrade))
1800     });
1801 } else {
1802     await sg.update({
1803         q1FinalGrade: -1
1804     });
1805 }
1806
1807 if (parseFloat(q2TransmuGrade) != -1) {
1808     await sg.update({
1809         q2FinalGrade: Math.round(parseFloat(q2TransmuGrade))
1810     });
1811 } else {
1812     await sg.update({
1813         q2FinalGrade: -1
1814     });
1815 }
1816
1817 if (parseFloat(q3TransmuGrade) != -1) {
1818     await sg.update({
1819         q3FinalGrade: Math.round(parseFloat(q3TransmuGrade))
1820     });
1821 } else {
1822     await sg.update({
1823         q3FinalGrade: -1
1824     });
1825 }
1826
1827 if (parseFloat(q4TransmuGrade) != -1) {
1828     await sg.update({
1829         q4FinalGrade: Math.round(parseFloat(q4TransmuGrade))
1830     });
1831 } else {
1832     await sg.update({
1833         q4FinalGrade: -1
1834     });
1835 }
1836
1837 if (subjectType == "NON_SHS") {
1838     if (
1839         parseFloat(q1TransmuGrade) != -1 &&
1840         parseFloat(q2TransmuGrade) != -1 &&
1841         parseFloat(q3TransmuGrade) != -1 &&
1842         parseFloat(q4TransmuGrade) != -1
1843     ) {
1844         let ave =
1845             (Math.round(parseFloat(q1TransmuGrade)) +
1846             Math.round(parseFloat(q2TransmuGrade)) +
1847             Math.round(parseFloat(q3TransmuGrade)) +
1848             Math.round(parseFloat(q4TransmuGrade))) /
1849             4;
1850         await sg.update({
1851             ave
1852         });
1853     } else {
1854         await sg.update({
1855             ave: -1

```

```

1856         });
1857     }
1858   } else {
1859     if (
1860       parseFloat(q1TransmuGrade) != -1 &&
1861       parseFloat(q2TransmuGrade) != -1
1862     ) {
1863       let ave =
1864         (Math.round(parseFloat(q1TransmuGrade)) +
1865          Math.round(parseFloat(q2TransmuGrade))) /
1866          2;
1867       await sg.update({
1868         ave
1869       });
1870     } else {
1871       await sg.update({
1872         ave: -1
1873       });
1874     }
1875   }
1876 }
1877 );
1878 };
1879
1880 exports.getSubjectTypeByClassRecordID = async classRecordID => {
1881   return await SubjectSection.findOne({
1882     where: {
1883       classRecordID
1884     }
1885   }).then(ss => {
1886     if (ss) {
1887       return ss.subjectType;
1888     }
1889   });
1890 };
1891
1892 exports.studentWeightedScoreUpdate = async weightedScoreID => {
1893   return await StudentWeightedScore.findOne({
1894     where: {
1895       weightedScoreID
1896     }
1897   }).then(async sws => {
1898     if (sws) {
1899       const subjectID = await this.getSubjectIDBySubsectstudID(
1900         sws.subsectstudID
1901       );
1902       Component.findAll({
1903         where: {
1904           subjectID
1905         }
1906       }).then(async comps => {
1907         if (comps.length != 0) {
1908           let data = [];
1909           let hasNegativeOne =
1910             sws.wwWS == -1 || sws.ptWS == -1 || sws.qeWS == -1;
1911           for (const [index, value] of comps.entries()) {
1912             const {
1913               component,
1914               compWeight
1915             } = value;
1916             data.push({
1917               component,
1918               compWeight
1919             });
1920           }
1921           let sum = 0;
1922           for (const [index2, value2] of data.entries()) {
1923             switch (value2.component) {
1924               case "FA":
1925                 {
1926                   sum =
1927                     sum +

```

```

1928             (parseFloat(sws.faWS) * parseFloat(value2.
1929                 compWeight)) / 100;
1930         }
1931     case "WW":
1932     {
1933         sum =
1934         sum +
1935         (parseFloat(sws.wwWS) * parseFloat(value2.
1936             compWeight)) / 100;
1937         break;
1938     case "PT":
1939     {
1940         sum =
1941         sum +
1942         (parseFloat(sws.ptWS) * parseFloat(value2.
1943             compWeight)) / 100;
1944         break;
1945     case "QE":
1946     {
1947         sum =
1948         sum +
1949         (parseFloat(sws.qeWS) * parseFloat(value2.
1950             compWeight)) / 100;
1951         break;
1952     default:
1953         break;
1954     }
1955 }
1956 if (!hasNegativeOne) {
1957     await sws.update({
1958         actualGrade: sum
1959     });
1960 } else {
1961     await sws.update({
1962         actualGrade: -1
1963     });
1964 }
1965
1966 let transmuGrade50 =
1967     75 +
1968     (parseFloat(sws.actualGrade) - parseFloat(50)) /
1969     ((100 - parseFloat(50)) / 25);
1970 let transmuGrade55 =
1971     75 +
1972     (parseFloat(sws.actualGrade) - parseFloat(55)) /
1973     ((100 - parseFloat(55)) / 25);
1974 let transmuGrade60 =
1975     75 +
1976     (parseFloat(sws.actualGrade) - parseFloat(60)) /
1977     ((100 - parseFloat(60)) / 25);
1978
1979 if (sws.actualGrade != -1) {
1980     await sws.update({
1981         transmutedGrade50: transmuGrade50
1982     });
1983     await sws.update({
1984         transmutedGrade55: transmuGrade55
1985     });
1986     await sws.update({
1987         transmutedGrade60: transmuGrade60
1988     });
1989 } else {
1990     await sws.update({
1991         transmutedGrade50: -1
1992     });
1993     await sws.update({
1994         transmutedGrade55: -1
1995     });
1996     await sws.update({
1997         transmutedGrade60: -1

```

```

1998         });
1999     });
2000   });
2001 });
2002 }
2003 });
2004 };
2005 };
2006 exports.generateReportCardStudent = async(
2007   doc,
2008   res,
2009   studentID,
2010   schoolYearID,
2011   quarter,
2012   end,
2013   headerSent
2014 ) => {
2015   const displayQuarter = q => `Quarter ${q.charAt(1)}`;
2016   const displayGradeLevel = gradeLevel => {
2017     switch (gradeLevel) {
2018       case "N":
2019         return "Nursery";
2020       case "K1":
2021         return "Kinder 1";
2022       case "K2":
2023         return "Kinder 2";
2024       case "G1":
2025         return "Grade 1";
2026       case "G2":
2027         return "Grade 2";
2028       case "G3":
2029         return "Grade 3";
2030       case "G4":
2031         return "Grade 4";
2032       case "G5":
2033         return "Grade 5";
2034       case "G6":
2035         return "Grade 6";
2036       case "G7":
2037         return "Grade 7";
2038       case "G8":
2039         return "Grade 8";
2040       case "G9":
2041         return "Grade 9";
2042       case "G10":
2043         return "Grade 10";
2044       case "G11":
2045         return "Grade 11";
2046       case "G12":
2047         return "Grade 12";
2048       default:
2049         return "";
2050     }
2051   };
2052   const name = await this.getStudentName(studentID);
2053   const schoolYear = await this.getSYname(schoolYearID);
2054   const studsectID = await StudentSection.findOne({
2055     where: {
2056       studentID,
2057       schoolYearID
2058     }
2059   }).then(ss => {
2060     if (ss) {
2061       return ss.studsectID;
2062     } else {
2063       return -1;
2064     }
2065   });
2066   if (studsectID == -1) {
2067     headerSent = true
2068     res.status(404).json({
2069       msg: "Student is not enrolled this school year."
2070     });
2071     // doc.end()
2072   } else {

```

```

2073     const section = await this.getSectionNameByStudsectID(studsectID)
2074         ;
2075     const gradeLevel = await this.getSectionGradeLevelByStudsectID(
2076         studsectID);
2077     const subsectstudIDs = await SubjectSectionStudent.findAll({
2078         where: {
2079             studsectID
2080         }
2081     }).then(async sss => {
2082         if (sss) {
2083             let data = [];
2084             for (const [index, x] of sss.entries()) {
2085                 const subjectType = await SubjectSectionStudent.findOne({
2086                     where: {
2087                         subsectstudID: x.subsectstudID
2088                     }
2089                 }).then(
2090                     async subsectstud =>
2091                     await SubjectSection.findOne({
2092                         where: {
2093                             subsectID: subsectstud.subsectID
2094                         }
2095                     }).then(subsect => subsect.subjectType)
2096                 );
2097                 const compare = ["Q1", "Q2"].includes(quarter) ?
2098                     ["NON_SHS", "1ST_SEM"] :
2099                     ["NON_SHS", "2ND_SEM"];
2100                 if (compare.includes(subjectType)) {
2101                     data.push(x.subsectstudID);
2102                 }
2103             }
2104         }
2105         return data;
2106     });
2107     if (subsectstudIDs.length == 0) {
2108         headerSent = true
2109         res.status(404).json({
2110             msg: "Student is not enrolled in a subject this school year."
2111         });
2112         // doc.end()
2113     } else {
2114         StudentSubjectGrades.findAll({
2115             where: {
2116                 subsectstudID: {
2117                     [Op.in]: subsectstudIDs
2118                 }
2119             }
2120         }).then(async ssg => {
2121             if (ssg) {
2122                 let data = {
2123                     q1: [],
2124                     q2: [],
2125                     q3: [],
2126                     q4: [],
2127                     final: []
2128                 };
2129                 for (const [i, y] of ssg.entries()) {
2130                     const classRecordStatus = await ClassRecordStatus.findOne(
2131                         {
2132                             where: {
2133                                 classRecordID: y.classRecordID
2134                             }
2135                         });
2136                     const subjectType = await this.
2137                         getSubjectTypeByClassRecordID(
2138                             y.classRecordID
2139                         );
2140                     let subjectName = await this.
2141                         getSubjectNameBySubsectstudID(
2142                             y.subsectstudID
2143                         );

```

```

2140     if (subjectType == "NON_SHS") {
2141       let hasNegativeOne = false;
2142       for (const [i2, q] of ["Q1", "Q2", "Q3", "Q4"].entries())
2143         }
2144         let score = y['${q.toLowerCase()}FinalGrade'];
2145         if (classRecordStatus[q.toLowerCase()] == "F" &&
2146             quarter >= q) {
2147           data[q.toLowerCase()].push({
2148             subjectName,
2149             score
2150           });
2151         } else {
2152           hasNegativeOne = true;
2153           data[q.toLowerCase()].push({
2154             subjectName,
2155             score: -1
2156           });
2157         }
2158       if (!hasNegativeOne) {
2159         data.final.push({
2160           subjectName,
2161           score: y.ave
2162         });
2163       } else {
2164         data.final.push({
2165           subjectName,
2166           score: -1
2167         });
2168     }
2169   } else if (subjectType == "1ST_SEM") {
2170     let hasNegativeOne = false;
2171     for (const [i2, q] of ["Q1", "Q2"].entries()) {
2172       let score = y['${q.toLowerCase()}FinalGrade'];
2173       if (classRecordStatus[q.toLowerCase()] == "F" &&
2174           quarter >= q) {
2175         data[q.toLowerCase()].push({
2176           subjectName,
2177           score
2178         });
2179       } else {
2180         hasNegativeOne = true;
2181         data[q.toLowerCase()].push({
2182           subjectName,
2183           score: -1
2184         });
2185       }
2186     if (!hasNegativeOne) {
2187       data.final.push({
2188         subjectName,
2189         score: y.ave
2190       });
2191     } else {
2192       data.final.push({
2193         subjectName,
2194         score: -1
2195       });
2196     }
2197   } else if (subjectType == "2ND_SEM") {
2198     let hasNegativeOne = false;
2199     for (const [i2, q] of ["Q1", "Q2"].entries()) {
2200       let score = y['${q.toLowerCase()}FinalGrade'];
2201       if (
2202         classRecordStatus[q.toLowerCase()] == "F" &&
2203         'Q${parseInt(q.charAt(1)) - 2}' >= q
2204       ) {
2205         data['Q${parseInt(q.charAt(1)) + 2}'].push({
2206           subjectName,
2207           score
2208         });
2209       } else {
2210         hasNegativeOne = true;

```

```

2210         data['q${parseInt(q.charAt(1)) + 2}'].push({
2211             subjectName,
2212             score: -1
2213         });
2214     }
2215 }
2216 if (!hasNegativeOne) {
2217     data.final.push({
2218         subjectName,
2219         score: y.ave
2220     });
2221 } else {
2222     data.final.push({
2223         subjectName,
2224         score: -1
2225     });
2226 }
2227 }
2228 }
2229 if (!["G11", "G12"].includes(gradeLevel)) {
2230     let finalGrade = -1;
2231     if (quarter == "Q4") {
2232         finalGrade = await StudentFinalGrade.findOne({
2233             where: {
2234                 schoolYearID,
2235                 studsectID
2236             }
2237         }).then(sfg => sfg.grade);
2238     }
2239     const {
2240         q1,
2241         q2,
2242         q3,
2243         q4,
2244         final
2245     } = data;
2246     let flags = [];
2247     let subjectName = [];
2248     //PDF CREATION
2249     for (const [index, value] of [
2250         ...q1,
2251         ...q2,
2252         ...q3,
2253         ...q4
2254     ].entries()) {
2255         if (flags[value.subjectName]) continue;
2256         flags[value.subjectName] = true;
2257         subjectName.push(value.subjectName);
2258     }
2259     doc.image(logo, 234.5, 30, {
2260         width: 140,
2261         align: "center"
2262     });
2263     doc.x = 30;
2264     doc.y = 90;
2265     doc.text("Name: " + name);
2266     doc.moveDown();
2267     doc.text("Section: " + section);
2268     doc.moveDown();
2269     doc.text("Grade Level: " + displayGradeLevel(gradeLevel))
2270     ;
2271     doc.moveDown();
2272     doc
2273         .fontSize(14)
2274         .text(
2275             'Report Card of S.Y. ${schoolYear} ${displayQuarter(
2276                 quarter)}', {
2277                 width: 549,
2278                 align: "center"
2279             }
2280         );
2281     doc.moveDown();
2282     doc.fontSize(12);

```

```

2281     let y = doc.y;
2282     let x = doc.x;
2283     doc.text("Subject Name", {
2284         width: 200,
2285         align: "center"
2286     });
2287     let newX = doc.x;
2288     let newY = doc.y;
2289     doc.x = 230;
2290     doc.y = y;
2291     doc.rect(x, y - 5, 200, newY - y + 5).stroke();
2292     doc.text("Q1", {
2293         width: 69.8,
2294         align: "center"
2295     });
2296     doc.x = 299.8;
2297     doc.y = y;
2298     doc.rect(doc.x - 69.8, y - 5, 69.8, newY - y + 5).stroke()
2299         ();
2300     doc.text("Q2", {
2301         width: 69.8,
2302         align: "center"
2303     });
2304     doc.x = 369.6;
2305     doc.y = y;
2306     doc.rect(doc.x - 69.8, y - 5, 69.8, newY - y + 5).stroke()
2307         ();
2308     doc.text("Q3", {
2309         width: 69.8,
2310         align: "center"
2311     });
2312     doc.x = 439.4;
2313     doc.y = y;
2314     doc.rect(doc.x - 69.8, y - 5, 69.8, newY - y + 5).stroke()
2315         ();
2316     doc.text("Q4", {
2317         width: 69.8,
2318         align: "center"
2319     });
2320     doc.x = 509.2;
2321     doc.y = y;
2322     doc.rect(doc.x - 69.8, y - 5, 69.8, newY - y + 5).stroke()
2323         ();
2324     doc.text("Final", {
2325         width: 69.8,
2326         align: "center"
2327     });
2328     doc.x = newX;
2329     doc.y = newY;

2330     doc.moveDown();
2331     for (const [index, value] of subjectName.entries()) {
2332         let border = 14;
2333         let y = doc.y;
2334         let x = doc.x;
2335         doc.x = doc.x + 5;
2336         doc.text(value, {
2337             width: 200
2338         });
2339         let newX = doc.x - 5;
2340         let newY = doc.y;
2341         doc.x = 230;
2342         doc.y = y;
2343         doc.rect(x, y - border, 200, newY - y + border).stroke()
2344             ();
2345         doc.text(
2346             q1.find(a => a.subjectName == value) ?
2347             q1.find(a => a.subjectName == value).score == -1 ?
2348             "" :
2349             q1.find(a => a.subjectName == value).score :
2350             "", {

```

```

2350           width: 69.8,
2351           align: "center"
2352       }
2353   );
2354   doc.x = 299.8;
2355   doc.y = y;
2356   doc
2357       .rect(doc.x - 69.8, y - border, 69.8, newY - y +
2358             border)
2359       .stroke();
2360   doc.text(
2361     q2.find(a => a.subjectName == value) ?
2362     q2.find(a => a.subjectName == value).score == -1 ?
2363     "" :
2364     q2.find(a => a.subjectName == value).score :
2365     "", {
2366       width: 69.8,
2367       align: "center"
2368     }
2369   );
2370   doc.x = 369.6;
2371   doc.y = y;
2372   doc
2373       .rect(doc.x - 69.8, y - border, 69.8, newY - y +
2374             border)
2375       .stroke();
2376   doc.text(
2377     q3.find(a => a.subjectName == value) ?
2378     q3.find(a => a.subjectName == value).score == -1 ?
2379     "" :
2380     q3.find(a => a.subjectName == value).score :
2381     "", {
2382       width: 69.8,
2383       align: "center"
2384     }
2385   );
2386   doc.x = 439.4;
2387   doc.y = y;
2388   doc
2389       .rect(doc.x - 69.8, y - border, 69.8, newY - y +
2390             border)
2391       .stroke();
2392   doc.text(
2393     q4.find(a => a.subjectName == value) ?
2394     q4.find(a => a.subjectName == value).score == -1 ?
2395     "" :
2396     q4.find(a => a.subjectName == value).score :
2397     "", {
2398       width: 69.8,
2399       align: "center"
2400     }
2401   );
2402   doc.x = 509.2;
2403   doc.y = y;
2404   doc
2405       .rect(doc.x - 69.8, y - border, 69.8, newY - y +
2406             border)
2407       .stroke();
2408   doc.text(
2409     final.find(a => a.subjectName == value) ?
2410     final.find(a => a.subjectName == value).score == -1 ?
2411     "" :
2412     final.find(a => a.subjectName == value).score :
2413     "", {
2414       width: 69.8,
2415       align: "center"
2416     }
2417   );
2418   doc.x = 579;
2419   doc.y = y;
2420   doc
2421       .rect(doc.x - 69.8, y - border, 69.8, newY - y +
2422             border)

```

```

2418     .stroke();
2419     doc.x = newX;
2420     doc.y = newY;
2421     doc.moveDown();
2422 }
2423 doc.text('Final Grade: ${finalGrade == -1 ? "N/A" :
2424     finalGrade}');
2425 if (end) doc.end();
2426 else doc.addPage();
2427 } else {
2428     if (["Q1", "Q2"].includes(quarter)) {
2429         // 1st Sem SHS
2430         let finalGrade = -1;
2431         if (quarter == "Q4") {
2432             finalGrade = await StudentFinalGrade.findOne({
2433                 where: [
2434                     schoolYearID,
2435                     studsectID
2436                 ]
2437             }).then(sfg => sfg.grade);
2438         }
2439         const {
2440             q1,
2441             q2,
2442             final
2443         } = data;
2444         let flags = [];
2445         let subjectName = [];
2446         //PDF CREATION
2447         for (const [index, value] of [...q1, ...q2].entries()) {
2448             if (flags[value.subjectName]) continue;
2449             flags[value.subjectName] = true;
2450             subjectName.push(value.subjectName);
2451         }
2452         doc.image(logo, 234.5, 30, {
2453             width: 140,
2454             align: "center"
2455         });
2456         doc.x = 30;
2457         doc.y = 90;
2458         doc.text("Name: " + name);
2459         doc.moveDown();
2460         doc.text("Section: " + section);
2461         doc.moveDown();
2462         doc.text("Grade Level: " + displayGradeLevel(gradeLevel
2463             ));
2464         doc.moveDown();
2465         doc
2466             .fontSize(14)
2467             .text(
2468                 'Report Card of S.Y. ${schoolYear} - First Semester
2469                 ${displayQuarter(
2470                     quarter
2471                 )}', {
2472                     width: 549,
2473                     align: "center"
2474                 }
2475             );
2476         doc.moveDown();
2477         doc.fontSize(12);
2478         let y = doc.y;
2479         let x = doc.x;
2480         doc.text("Subject Name", {
2481             width: 200,
2482             align: "center"
2483         });
2484         let newX = doc.x;
2485         let newY = doc.y;
2486         doc.x = 230;
2487         doc.y = y;
2488         doc.rect(x, y - 5, 200, newY - y + 5).stroke();
2489         doc.text("Midterm", {
2490             width: 116.33,

```

```

2488         align: "center"
2489     });
2490     doc.x = 346.33;
2491     doc.y = y;
2492     doc.rect(doc.x - 116.33, y - 5, 116.33, newY - y + 5).
2493         stroke();
2494     doc.text("Finals", {
2495         width: 116.33,
2496         align: "center"
2497     });
2498     doc.x = 462.66;
2499     doc.y = y;
2500     doc.rect(doc.x - 116.33, y - 5, 116.3, newY - y + 5).
2501         stroke();
2502     doc.text("Final Grade", {
2503         width: 116.33,
2504         align: "center"
2505     });
2506     doc.x = 579;
2507     doc.y = y;
2508     doc.rect(doc.x - 116.33, y - 5, 116.3, newY - y + 5).
2509         stroke();
2510     doc.x = newX;
2511     doc.y = newY;
2512     doc.moveDown();
2513     for (const [index, value] of subjectName.entries()) {
2514         let border = 14;
2515         let y = doc.y;
2516         let x = doc.x;
2517         doc.x = doc.x + 5;
2518         doc.text(value, {
2519             width: 200
2520         });
2521         let newX = doc.x - 5;
2522         let newY = doc.y;
2523         doc.x = 230;
2524         doc.y = y;
2525         doc.rect(x, y - border, 200, newY - y + border).
2526             stroke();
2527         doc.text(
2528             q1.find(a => a.subjectName == value) ?
2529             q1.find(a => a.subjectName == value).score == -1 ?
2530             "" :
2531             q1.find(a => a.subjectName == value).score :
2532             "", {
2533                 width: 116.33,
2534                 align: "center"
2535             }
2536         );
2537         doc.x = 346.33;
2538         doc.y = y;
2539         doc
2540             .rect(doc.x - 116.33, y - border, 116.33, newY - y
2541                 + border)
2542                 .stroke();
2543         doc.text(
2544             q2.find(a => a.subjectName == value) ?
2545             q2.find(a => a.subjectName == value).score == -1 ?
2546             "" :
2547             q2.find(a => a.subjectName == value).score :
2548             "", {
2549                 width: 116.33,
2550                 align: "center"
2551             }
2552         );
2553         doc.x = 462.66;
2554         doc.y = y;
2555         doc
2556             .rect(doc.x - 116.33, y - border, 116.33, newY - y
2557                 + border)
2558                 .stroke();
2559         doc.text(
2560             final.find(a => a.subjectName == value) ?
2561             final.find(a => a.subjectName == value).score == -1

```

```

    ?
2556    "": {
2557        final.find(a => a.subjectName == value).score :
2558        "", {
2559            width: 116.33,
2560            align: "center"
2561        }
2562    );
2563    doc.x = 579;
2564    doc.y = y;
2565    doc
2566        .rect(doc.x - 116.33, y - border, 116.33, newY - y
2567            + border)
2568        .stroke();
2569    doc.x = newX;
2570    doc.y = newY;
2571    doc.moveDown();
2572}
2573let enumerator = 0;
2574let denominator = 0;
2575let ave = 0;
2576for (const [i3, v3] of final.entries()) {
2577    if (v3.score != -1) {
2578        enumerator = enumerator + parseFloat(v3.score);
2579        denominator = denominator + 1;
2580    }
2581}
2582ave =
2583denominator == 0 ?
2584-1 :
2585    Math.round(parseFloat(enumerator / denominator));
2586doc.text(`Semester Grade: ${ave == -1 ? "N/A" : ave}`);
2587doc.moveDown();
2588doc.text(`Final Grade: ${finalGrade == -1 ? "N/A" :
2589    finalGrade}`);
2590if (end) doc.end();
2591else doc.addPage();
2592} else {
2593    // 2nd Sem SHS
2594    let finalGrade = -1;
2595    if (quarter == "Q4") {
2596        finalGrade = await StudentFinalGrade.findOne({
2597            where: {
2598                schoolYearID,
2599                studsectID
2600            }
2601        }).then(sfg => sfg.grade);
2602    }
2603    const {
2604        q3,
2605        q4,
2606        final
2607    } = data;
2608    let flags = [];
2609    let subjectName = [];
2610    //PDF CREATION
2611    for (const [index, value] of [...q3, ...q4].entries()) {
2612        if (flags[value.subjectName]) continue;
2613        flags[value.subjectName] = true;
2614        subjectName.push(value.subjectName);
2615    }
2616    doc.image(logo, 234.5, 30, {
2617        width: 140,
2618        align: "center"
2619    });
2620    doc.x = 30;
2621    doc.y = 90;
2622    doc.text("Name: " + name);
2623    doc.moveDown();
2624    doc.text("Section: " + section);
2625    doc.moveDown();
2626    doc.text("Grade Level: " + displayGradeLevel(gradeLevel)

```

```

        });
2626 doc.moveDown();
2627 doc
2628     .fontSize(14)
2629     .text(
2630         'Report Card of S.Y. ${schoolYear} - Second
2631             Semester ${displayQuarter(
2632                 quarter
2633             )}', {
2634                 width: 549,
2635                 align: "center"
2636             }
2637         );
2638 doc.moveDown();
2639 doc.fontSize(12);
2640 let y = doc.y;
2641 let x = doc.x;
2642 doc.text("Subject Name", {
2643     width: 200,
2644     align: "center"
2645 });
2646 let newX = doc.x;
2647 let newY = doc.y;
2648 doc.x = 230;
2649 doc.y = y;
2650 doc.rect(x, y - 5, 200, newY - y + 5).stroke();
2651 doc.text("Midterm", {
2652     width: 116.33,
2653     align: "center"
2654 });
2655 doc.x = 346.33;
2656 doc.y = y;
2657 doc.rect(doc.x - 116.33, y - 5, 116.33, newY - y + 5).
2658     stroke();
2659 doc.text("Finals", {
2660     width: 116.33,
2661     align: "center"
2662 });
2663 doc.x = 462.66;
2664 doc.y = y;
2665 doc.rect(doc.x - 116.33, y - 5, 116.3, newY - y + 5).
2666     stroke();
2667 doc.text("Final Grade", {
2668     width: 116.33,
2669     align: "center"
2670 });
2671 doc.x = 579;
2672 doc.y = y;
2673 doc.rect(doc.x - 116.33, y - 5, 116.3, newY - y + 5).
2674     stroke();
2675 doc.x = newX;
2676 doc.y = newY;
2677 doc.moveDown();
2678 for (const [index, value] of subjectName.entries()) {
2679     let border = 14;
2680     let y = doc.y;
2681     let x = doc.x;
2682     doc.x = doc.x + 5;
2683     doc.text(value, {
2684         width: 200
2685     });
2686     let newX = doc.x - 5;
2687     let newY = doc.y;
2688     doc.x = 230;
2689     doc.y = y;
2690     doc.rect(x, y - border, 200, newY - y + border).
2691         stroke();
2692     doc.text(
2693         q3.find(a => a.subjectName == value) ?
2694             q3.find(a => a.subjectName == value).score == -1 ?
2695                 "" :
2696                 q3.find(a => a.subjectName == value).score :
2697                     "", {
2698                         width: 116.33,

```

```

2694         align: "center"
2695     }
2696 );
2697 doc.x = 346.33;
2698 doc.y = y;
2699 doc
2700     .rect(doc.x - 116.33, y - border, 116.33, newY - y
2701         + border)
2702     .stroke();
2703 doc.text(
2704     q4.find(a => a.subjectName == value) ?
2705     q4.find(a => a.subjectName == value).score == -1 ?
2706     "" :
2707     q4.find(a => a.subjectName == value).score :
2708     "", {
2709         width: 116.33,
2710         align: "center"
2711     }
2712 );
2713 doc.x = 462.66;
2714 doc.y = y;
2715 doc
2716     .rect(doc.x - 116.33, y - border, 116.33, newY - y
2717         + border)
2718     .stroke();
2719 doc.text(
2720     final.find(a => a.subjectName == value) ?
2721     final.find(a => a.subjectName == value).score == -1 ?
2722     "" :
2723     final.find(a => a.subjectName == value).score :
2724     "", {
2725         width: 116.33,
2726         align: "center"
2727     }
2728 );
2729 doc.x = 579;
2730 doc.y = y;
2731 doc
2732     .rect(doc.x - 116.33, y - border, 116.33, newY - y
2733         + border)
2734     .stroke();
2735 doc.x = newX;
2736 doc.y = newY;
2737 doc.moveDown();
2738 }
2739 let enumerator = 0;
2740 let denominator = 0;
2741 let ave = 0;
2742 for (const [i3, v3] of final.entries()) {
2743     if (v3.score != -1) {
2744         enumerator = enumerator + parseFloat(v3.score);
2745         denominator = denominator + 1;
2746     }
2747 }
2748 ave =
2749     denominator == 0 ?
2750     -1 :
2751     Math.round(parseFloat(enumerator / denominator));
2752 doc.text(`Semester Grade: ${ave == -1 ? "N/A" : ave}`);
2753 doc.moveDown();
2754 doc.text(`Final Grade: ${finalGrade == -1 ? "N/A" :
2755     finalGrade}`);
2756 if (end) doc.end();
2757 else doc.addPage();
2758 }
2759 }
2760 };

```

Listing 38: addsection.js

```
1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateAddSection(data) {
5     let errors = {};
6
7     data.sectionName = !isEmpty(data.sectionName) ? data.sectionName :
8         "";
9
10    if (!Validator.isLength(data.sectionName, { min: 2, max: 255 })) {
11        errors.sectionName = "Section name must be between 2 and 255
12            characters";
13    }
14
15    if (Validator.isEmpty(data.sectionName)) {
16        errors.sectionName = "Section name field is required";
17    }
18
19    return {
20        errors,
21        isValid: isEmpty(errors)
22    };
23}
```

Listing 39: createaccount.js

```
1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateCreateAccount(data) {
5     let errors = {};
6
7     data.email = !isEmpty(data.email) ? data.email : "";
8     data.firstName = !isEmpty(data.firstName) ? data.firstName : "";
9     data.middleName = !isEmpty(data.middleName) ? data.middleName : "";
10    data.lastName = !isEmpty(data.lastName) ? data.lastName : "";
11
12    if (!Validator.isEmail(data.email)) {
13        errors.email = "Email is invalid";
14    }
15    if (
16        !Validator.isLength(data.firstName, {
17            min: 2,
18            max: 255
19        })
20    ) {
21        errors.firstName = "First name must be between 2 and 255
22            characters";
23    }
24
25    if (
26        !Validator.isLength(data.lastName, {
27            min: 2,
28            max: 255
29        })
30    ) {
31        errors.lastName = "Last name must be between 2 and 255 characters
32            ";
33    }
34
35    if (
36        !Validator.isLength(data.middleName, {
37            min: 2,
38            max: 255
39        })
40    ) {
41        errors.middleName = "Middle name must be between 2 and 255 characters
42            ";
43    }
44}
```

```

39         errors.middleName = "Middle name must be between 2 and 255
40             characters";
41     }
42     if (Validator.isEmpty(data.firstName)) {
43         errors.firstName = "First name field is required";
44     }
45     if (Validator.isEmpty(data.lastName)) {
46         errors.lastName = "Last name field is required";
47     }
48     if (Validator.isEmpty(data.middleName)) {
49         errors.middleName = "Middle name field is required";
50     }
51     if (Validator.isEmpty(data.email)) {
52         errors.email = "Email field is required";
53     }
54
55     return {
56         errors,
57         isValid: isEmpty(errors)
58     };

```

Listing 40: editprofile.js

```

1  const Validator = require("validator");
2  const isEmpty = require("./is-empty");
3
4  module.exports = function validateEditProfile(data) {
5      let errors = {};
6
7      data.firstName = !isEmpty(data.firstName) ? data.firstName : "";
8      data.lastName = !isEmpty(data.lastName) ? data.lastName : "";
9      data.middleName = !isEmpty(data.middleName) ? data.middleName : "";
10     data.suffix = !isEmpty(data.suffix) ? data.suffix : "";
11     data.imageUrl = !isEmpty(data.imageUrl) ? data.imageUrl : "";
12     data.contactNum = !isEmpty(data.contactNum) ? data.contactNum : "";
13     data.email = !isEmpty(data.email) ? data.email : "";
14     data.nickname = !isEmpty(data.nickname) ? data.nickname : "";
15     data.address = !isEmpty(data.address) ? data.address : "";
16     data.bday = !isEmpty(data.bday) ? data.bday : "";
17     data.sex = !isEmpty(data.sex) ? data.sex : "";
18
19     if (
20         !Validator.minLength(data.firstName, {
21             min: 2,
22             max: 255
23         })
24     ) {
25         errors.firstName = "First name must be between 2 and 255
26             characters";
27     }
28
29     if (
30         !Validator.minLength(data.lastName, {
31             min: 2,
32             max: 255
33         })
34     ) {
35         errors.lastName = "Last name must be between 2 and 255 characters
36             ";
37     }
38
39     if (
40         !Validator.minLength(data.middleName, {
41             min: 2,
42             max: 255
43         })
44     ) {
45         errors.middleName = "Middle name must be between 2 and 255
46             characters";
47     }

```

```

46     if (
47         !Validator.isLength(data.nickname, {
48             min: 2,
49             max: 255
50         })
51     ) {
52         errors.nickname = "Nickname must be between 2 and 255 characters"
53         ;
54     }
55     if (
56         !Validator.isLength(data.address, {
57             min: 2,
58             max: 255
59         })
60     ) {
61         errors.address = "Middle name must be between 2 and 255
62             characters";
63     }
64     if (
65         !Validator.isLength(data.contactNum, {
66             min: 11,
67             max: 11
68         })
69     ) {
70         errors.contactNum = "Contact number must be exactly 11 digits";
71     }
72     if (Validator.isEmpty(data.firstName)) {
73         errors.firstName = "First name field is required";
74     }
75     if (Validator.isEmpty(data.lastName)) {
76         errors.lastName = "Last name field is required";
77     }
78     if (Validator.isEmpty(data.middleName)) {
79         errors.middleName = "Middle name field is required";
80     }
81     if (Validator.isEmpty(data.imageUrl)) {
82         errors.imageUrl = "Image Url field is required";
83     }
84     if (Validator.isEmpty(data.contactNum)) {
85         errors.contactNum = "Contact number field is required";
86     }
87     if (Validator.isEmpty(data.email)) {
88         errors.email = "Email field is required";
89     }
90     if (Validator.isEmpty(data.nickname)) {
91         errors.nickname = "Nickname field is required";
92     }
93     if (Validator.isEmpty(data.address)) {
94         errors.address = "Address field is required";
95     }
96     if (Validator.isEmpty(data.bday)) {
97         errors.bday = "Birthday field is required";
98     }
99     if (Validator.isEmpty(data.sex)) {
100        errors.sex = "Sex field is required";
101    }
102    if (!Validator.isEmail(data.email)) {
103        errors.email = "Email is invalid";
104    }
105
106    return {
107        errors,
108        isValid: isEmpty(errors)
109    };
110};

```

Listing 41: editprofilenonacademic.js

```
1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = validateEditProfileNonacademic = data => {
5     let errors = {};
6
7     data.firstName = !isEmpty(data.firstName) ? data.firstName : "";
8     data.lastName = !isEmpty(data.lastName) ? data.lastName : "";
9     data.middleName = !isEmpty(data.middleName) ? data.middleName : "";
10    data.suffix = !isEmpty(data.suffix) ? data.suffix : "";
11    data.nickname = !isEmpty(data.nickname) ? data.nickname : "";
12    data.contactNum = !isEmpty(data.contactNum) ? data.contactNum : "";
13    data.address = !isEmpty(data.address) ? data.address : "";
14    data.province = !isEmpty(data.province) ? data.province : "";
15    data.city = !isEmpty(data.city) ? data.city : "";
16    data.region = !isEmpty(data.region) ? data.region : "";
17    data.zipcode = !isEmpty(data.zipcode) ? data.zipcode : "";
18    data.civilStatus = !isEmpty(data.civilStatus) ? data.civilStatus :
19        "";
20    data.sex = !isEmpty(data.sex) ? data.sex : "";
21    data.citizenship = !isEmpty(data.citizenship) ? data.citizenship :
22        "";
23    data.birthDate = !isEmpty(data.birthDate) ? data.birthDate : "";
24    data.birthPlace = !isEmpty(data.birthPlace) ? data.birthPlace : "";
25    data.religion = !isEmpty(data.religion) ? data.religion : "";
26    data.emergencyName = !isEmpty(data.emergencyName) ? data.
27        emergencyName : "";
28    data.emergencyAddress = !isEmpty(data.emergencyAddress)
29        ? data.emergencyAddress
30        : "";
31    data.emergencyTelephone = !isEmpty(data.emergencyTelephone)
32        ? data.emergencyTelephone
33        : "";
34    data.emergencyCellphone = !isEmpty(data.emergencyCellphone)
35        ? data.emergencyCellphone
36        : "";
37    data.emergencyEmail = !isEmpty(data.emergencyEmail)
38        ? data.emergencyEmail
39        : "";
40    data.emergencyRelationship = !isEmpty(data.emergencyRelationship)
41        ? data.emergencyRelationship
42        : "";
43    if (
44        !Validator.isLength(data.firstName, {
45            min: 2,
46            max: 255
47        })
48    ) {
49        errors.firstName = "First name must be between 2 and 255
50            characters";
51    }
52
53    if (
54        !Validator.isLength(data.lastName, {
55            min: 2,
56            max: 255
57        })
58    ) {
59        errors.lastName = "Last name must be between 2 and 255 characters
60            ";
61    }
62
63    if (
64        !Validator.isLength(data.middleName, {
65            min: 2,
66            max: 255
67        })
68    ) {
69        errors.middleName = "Middle name must be between 2 and 255
70            characters";
71    }
72}
```

```
67     if (
68         !Validator.isLength(data.suffix, {
69             min: 2,
70             max: 10
71         })
72     ) {
73         errors.suffix = "Suffix must be between 2 and 10 characters";
74     }
75
76     if (
77         !Validator.isLength(data.nickname, {
78             min: 2,
79             max: 255
80         })
81     ) {
82         errors.nickname = "Nickname must be between 2 and 255 characters";
83     }
84
85     if (
86         !Validator.isLength(data.address, {
87             min: 2,
88             max: 255
89         })
90     ) {
91         errors.address = "Middle name must be between 2 and 255
92             characters";
93     }
94
95     if (
96         !Validator.isLength(data.contactNum, {
97             min: 11,
98             max: 11
99         })
100    ) {
101        errors.contactNum = "Contact number must be exactly 11 digits";
102    }
103
104    if (
105        !Validator.isLength(data.province, {
106            min: 2,
107            max: 20
108        })
109    ) {
110        errors.province = "Province must be between 2 and 20 characters";
111    }
112
113    if (
114        !Validator.isLength(data.city, {
115            min: 2,
116            max: 25
117        })
118    ) {
119        errors.city = "City must be between 2 and 25 characters";
120    }
121
122    if (
123        !Validator.isLength(data.zipcode, {
124            min: 4,
125            max: 4
126        })
127    ) {
128        errors.zipcode = "Postal code must be exactly 4 digits";
129    }
130    if (
131        !Validator.isLength(data.region, {
132            min: 2,
133            max: 15
134        })
135    ) {
136        errors.region = "Region must be between 2 and 15 characters";
137    }
138    if (
139        !Validator.isLength(data.citizenship, {
```

```

139         min: 2,
140         max: 20
141     })
142 }
143 errors.citizenship = "Citizenship must be between 2 and 20
144         characters";
145 }
146 if (
147     !Validator.isLength(data.birthPlace, {
148         min: 2,
149         max: 25
150     })
151 )
152 errors.birthPlace = "Birth place must be between 2 and 25
153         characters";
154 }
155 if (
156     !Validator.isLength(data.religion, {
157         min: 2,
158         max: 20
159     })
160 )
161 errors.religion = "Religion must be between 2 and 20 characters";
162 }
163 if (
164     !Validator.isLength(data.emergencyName, {
165         min: 2,
166         max: 30
167     })
168 )
169 errors.emergencyName = "Contact person be between 2 and 30
170         characters";
171 }
172 if (
173     !Validator.isLength(data.emergencyAddress, {
174         min: 2,
175         max: 50
176     })
177 )
178 errors.emergencyAddress = "Contact address be between 2 and 50
179         characters";
180 }
181 if (
182     !Validator.isLength(data.emergencyTelephone, {
183         min: 2,
184         max: 50
185     })
186 )
187 errors.emergencyTelephone =
188     "Contact telephone no. be between 2 and 50 characters";
189 }
190 if (
191     !Validator.isLength(data.emergencyCellphone, {
192         min: 2,
193         max: 50
194     })
195 )
196 errors.emergencyCellphone =
197     "Contact cellphone no. be between 2 and 50 characters";
198 }
199 if (
200     !Validator.isLength(data.emergencyEmail, {
201         min: 2,
202         max: 30
203     })
204 )
205 errors.emergencyEmail = "Contact email be between 2 and 30
206
207
208

```

```

                characters";
209
210
211     }
212     if (
213         !Validator.isLength(data.emergencyRelationship, {
214             min: 2,
215             max: 15
216         })
217     ) {
218         errors.emergencyRelationship =
219             "Contact relationship no. be between 2 and 15 characters";
220     }
221
222     if (Validator.isEmpty(data.firstName)) {
223         errors.firstName = "First name field is required";
224     }
225     if (Validator.isEmpty(data.lastName)) {
226         errors.lastName = "Last name field is required";
227     }
228     if (Validator.isEmpty(data.middleName)) {
229         errors.middleName = "Middle name field is required";
230     }
231
232     if (Validator.isEmpty(data.suffix)) {
233         errors.suffix = "Suffix field is required";
234     }
235     if (Validator.isEmpty(data.contactNum)) {
236         errors.contactNum = "Contact number field is required";
237     }
238
239     if (Validator.isEmpty(data.nickname)) {
240         errors.nickname = "Nickname field is required";
241     }
242     if (Validator.isEmpty(data.address)) {
243         errors.address = "Address field is required";
244     }
245     if (Validator.isEmpty(data.province)) {
246         errors.province = "Province field is required";
247     }
248
249     if (Validator.isEmpty(data.city)) {
250         errors.city = "City field is required";
251     }
252
253     if (Validator.isEmpty(data.region)) {
254         errors.region = "Region field is required";
255     }
256
257     if (Validator.isEmpty(data.zipcode)) {
258         errors.zipcode = "Postal code field is required";
259     }
260
261     if (Validator.isEmpty(data.citizenship)) {
262         errors.citizenship = "Citizenship field is required";
263     }
264
265     if (Validator.isEmpty(data.birthPlace)) {
266         errors.birthPlace = "Birth place field is required";
267     }
268
269     if (Validator.isEmpty(data.religion)) {
270         errors.religion = "Religion field is required";
271     }
272
273     if (Validator.isEmpty(data.emergencyName)) {
274         errors.emergencyName = "Contact person field is required";
275     }
276
277     if (Validator.isEmpty(data.emergencyAddress)) {
278         errors.emergencyAddress = "Contact address field is required";
279     }
280
281     if (Validator.isEmpty(data.emergencyTelephone)) {
282         errors.emergencyTelephone = "Contact telephone no. field is

```

```

        required";
282    }
283
284    if (Validator.isEmpty(data.emergencyCellphone)) {
285        errors.emergencyCellphone = "Contact cellphone no. field is
286            required";
287    }
288
289    if (Validator.isEmpty(data.emergencyEmail)) {
290        errors.emergencyEmail = "Contact email field is required";
291    }
292
293    if (Validator.isEmpty(data.emergencyRelationship)) {
294        errors.emergencyRelationship = "Contact relationship field is
295            required";
296    }
297
298    if (!Validator.isEmail(data.emergencyEmail)) {
299        errors.emergencyEmail = "Email is invalid";
300    }
301
302    return {
303        errors,
304        isValid: isEmpty(errors)
305    };
306}

```

Listing 42: editprofileparent.js

```

1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = validateEditProfileParent = data => {
5     let errors = {};
6
7     data.firstName = !isEmpty(data.firstName) ? data.firstName : "";
8     data.lastName = !isEmpty(data.lastName) ? data.lastName : "";
9     data.middleName = !isEmpty(data.middleName) ? data.middleName : "";
10    data.suffix = !isEmpty(data.suffix) ? data.suffix : "";
11    data.nickname = !isEmpty(data.nickname) ? data.nickname : "";
12    data.contactNum = !isEmpty(data.contactNum) ? data.contactNum : "";
13    data.address = !isEmpty(data.address) ? data.address : "";
14    data.province = !isEmpty(data.province) ? data.province : "";
15    data.city = !isEmpty(data.city) ? data.city : "";
16    data.region = !isEmpty(data.region) ? data.region : "";
17    data.zipcode = !isEmpty(data.zipcode) ? data.zipcode : "";
18    data.civilStatus = !isEmpty(data.civilStatus) ? data.civilStatus :
19        "";
20    data.sex = !isEmpty(data.sex) ? data.sex : "";
21    data.citizenship = !isEmpty(data.citizenship) ? data.citizenship :
22        "";
23    data.birthDate = !isEmpty(data.birthDate) ? data.birthDate : "";
24    data.birthPlace = !isEmpty(data.birthPlace) ? data.birthPlace : "";
25    data.religion = !isEmpty(data.religion) ? data.religion : "";
26    data.emergencyName = !isEmpty(data.emergencyName) ? data.
27        emergencyName : "";
28    data.emergencyAddress = !isEmpty(data.emergencyAddress)
29        ? data.emergencyAddress
30        : "";
31    data.emergencyTelephone = !isEmpty(data.emergencyTelephone)
32        ? data.emergencyTelephone
33        : "";
34    data.emergencyCellphone = !isEmpty(data.emergencyCellphone)
35        ? data.emergencyCellphone
36        : "";
37    data.emergencyEmail = !isEmpty(data.emergencyEmail)
38        ? data.emergencyEmail
39        : "";
40    data.emergencyRelationship = !isEmpty(data.emergencyRelationship)
41        ? data.emergencyRelationship
42        : "";
43
44    if (
45        !Validator.minLength(data.firstName, {

```

```

42             min: 2,
43             max: 255
44         })
45     ) {
46         errors.firstName = "First name must be between 2 and 255
47             characters";
48     }
49
50     if (
51         !Validator.isLength(data.lastName, {
52             min: 2,
53             max: 255
54         })
55     ) {
56         errors.lastName = "Last name must be between 2 and 255 characters
57             ";
58     }
59
60     if (
61         !Validator.isLength(data.middleName, {
62             min: 2,
63             max: 255
64         })
65     ) {
66         errors.middleName = "Middle name must be between 2 and 255
67             characters";
68     }
69
70     if (
71         !Validator.isLength(data.suffix, {
72             min: 2,
73             max: 10
74         })
75     ) {
76         errors.suffix = "Suffix must be between 2 and 10 characters";
77     }
78
79     if (
80         !Validator.isLength(data.nickname, {
81             min: 2,
82             max: 255
83         })
84     ) {
85         errors.nickname = "Nickname must be between 2 and 255 characters"
86             ;
87     }
88
89     if (
90         !Validator.isLength(data.address, {
91             min: 2,
92             max: 255
93         })
94     ) {
95         errors.address = "Address must be between 2 and 255
96             characters";
97     }
98
99     if (
100        !Validator.isLength(data.contactNum, {
101            min: 11,
102            max: 11
103        })
104    ) {
105        errors.contactNum = "Contact number must be exactly 11 digits";
106    }
107
108    if (
109        !Validator.isLength(data.province, {
110            min: 2,
111            max: 20
112        })
113    ) {
114        errors.province = "Province must be between 2 and 20 characters";
115    }

```

```

111
112     if (
113         !Validator.isLength(data.city, {
114             min: 2,
115             max: 25
116         })
117     ) {
118         errors.city = "City must be between 2 and 25 characters";
119     }
120
121     if (
122         !Validator.isLength(data.zipcode, {
123             min: 4,
124             max: 4
125         })
126     ) {
127         errors.zipcode = "Postal code must be exactly 4 digits";
128     }
129     if (
130         !Validator.isLength(data.region, {
131             min: 2,
132             max: 15
133         })
134     ) {
135         errors.region = "Region must be between 2 and 15 characters";
136     }
137     if (
138         !Validator.isLength(data.citizenship, {
139             min: 2,
140             max: 20
141         })
142     ) {
143         errors.citizenship = "Citizenship must be between 2 and 20
144             characters";
145     }
146
147     if (
148         !Validator.isLength(data.birthPlace, {
149             min: 2,
150             max: 25
151         })
152     ) {
153         errors.birthPlace = "Birth place must be between 2 and 25
154             characters";
155     }
156
157     if (
158         !Validator.isLength(data.religion, {
159             min: 2,
160             max: 20
161         })
162     ) {
163         errors.religion = "Religion must be between 2 and 20 characters";
164     }
165
166     if (
167         !Validator.isLength(data.emergencyName, {
168             min: 2,
169             max: 30
170         })
171     ) {
172         errors.emergencyName = "Contact person be between 2 and 30
173             characters";
174     }
175
176     if (
177         !Validator.isLength(data.emergencyAddress, {
178             min: 2,
179             max: 50
180         })
181     ) {
182         errors.emergencyAddress = "Contact address be between 2 and 50
183             characters";
184     }

```

```

181
182     if (
183         !Validator.isLength(data.emergencyTelephone, {
184             min: 2,
185             max: 50
186         })
187     ) {
188         errors.emergencyTelephone =
189             "Contact telephone no. be between 2 and 50 characters";
190     }
191
192     if (
193         !Validator.isLength(data.emergencyCellphone, {
194             min: 2,
195             max: 50
196         })
197     ) {
198         errors.emergencyCellphone =
199             "Contact cellphone no. be between 2 and 50 characters";
200     }
201
202     if (
203         !Validator.isLength(data.emergencyEmail, {
204             min: 2,
205             max: 30
206         })
207     ) {
208         errors.emergencyEmail = "Contact email be between 2 and 30
209             characters";
210     }
211
212     if (
213         !Validator.isLength(data.emergencyRelationship, {
214             min: 2,
215             max: 15
216         })
217     ) {
218         errors.emergencyRelationship =
219             "Contact relationship no. be between 2 and 15 characters";
220     }
221
222     if (Validator.isEmpty(data.firstName)) {
223         errors.firstName = "First name field is required";
224     }
225     if (Validator.isEmpty(data.lastName)) {
226         errors.lastName = "Last name field is required";
227     }
228     if (Validator.isEmpty(data.middleName)) {
229         errors.middleName = "Middle name field is required";
230     }
231
232     if (Validator.isEmpty(data.suffix)) {
233         errors.suffix = "Suffix field is required";
234     }
235     if (Validator.isEmpty(data.contactNum)) {
236         errors.contactNum = "Contact number field is required";
237     }
238
239     if (Validator.isEmpty(data.nickname)) {
240         errors.nickname = "Nickname field is required";
241     }
242     if (Validator.isEmpty(data.address)) {
243         errors.address = "Address field is required";
244     }
245     if (Validator.isEmpty(data.province)) {
246         errors.province = "Province field is required";
247     }
248
249     if (Validator.isEmpty(data.city)) {
250         errors.city = "City field is required";
251     }
252
253     if (Validator.isEmpty(data.region)) {
254         errors.region = "Region field is required";

```

```

254     }
255
256     if (Validator.isEmpty(data.zipcode)) {
257       errors.zipcode = "Postal code field is required";
258     }
259
260     if (Validator.isEmpty(data.citizenship)) {
261       errors.citizenship = "Citizenship field is required";
262     }
263
264     if (Validator.isEmpty(data.birthPlace)) {
265       errors.birthPlace = "Birth place field is required";
266     }
267
268     if (Validator.isEmpty(data.religion)) {
269       errors.religion = "Religion field is required";
270     }
271
272     if (Validator.isEmpty(data.emergencyName)) {
273       errors.emergencyName = "Contact person field is required";
274     }
275
276     if (Validator.isEmpty(data.emergencyAddress)) {
277       errors.emergencyAddress = "Contact address field is required";
278     }
279
280     if (Validator.isEmpty(data.emergencyTelephone)) {
281       errors.emergencyTelephone = "Contact telephone no. field is
282       required";
283     }
284
285     if (Validator.isEmpty(data.emergencyCellphone)) {
286       errors.emergencyCellphone = "Contact cellphone no. field is
287       required";
288
289     if (Validator.isEmpty(data.emergencyEmail)) {
290       errors.emergencyEmail = "Contact email field is required";
291     }
292
293     if (Validator.isEmpty(data.emergencyRelationship)) {
294       errors.emergencyRelationship = "Contact relationship field is
295       required";
296
297     if (!Validator.isEmail(data.emergencyEmail)) {
298       errors.emergencyEmail = "Email is invalid";
299     }
300
301     return {
302       errors,
303       isValid: isEmpty(errors)
304     };
305   };

```

Listing 43: login.js

```

1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateLoginInput(data) {
5   let errors = {};
6
7   data.email = !isEmpty(data.email) ? data.email : "";
8   data.password = !isEmpty(data.password) ? data.password : "";
9
10  if (!Validator.isEmail(data.email)) {
11    errors.email = "Email is invalid";
12  }
13
14  if (Validator.isEmpty(data.password)) {

```

```

15         errors.password = "Password field is required";
16     }
17
18     if (Validator.isEmpty(data.email)) {
19         errors.email = "Email field is required";
20     }
21
22     return {
23         errors,
24         isValid: isEmpty(errors)
25     };
26 }

```

Listing 44: password.js

```

1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateChangePassword(data) {
5     let errors = {};
6     data.password = !isEmpty(data.password) ? data.password : "";
7     data.password2 = !isEmpty(data.password2) ? data.password2 : "";
8
9     if (Validator.isEmpty(data.password)) {
10         errors.password = "Password field is required";
11     }
12
13     if (Validator.isEmpty(data.password2)) {
14         errors.password2 = "Confirm Password field is required";
15     }
16
17     if (!Validator.equals(data.password, data.password2)) {
18         errors.password2 = "Passwords must match";
19     }
20
21     if (
22         !Validator.minLength(data.password, {
23             min: 6,
24             max: 30
25         })
26     ) {
27         errors.password = "Password must be at least 6 characters";
28     }
29
30     return {
31         errors,
32         isValid: isEmpty(errors)
33     };
34 }

```

Listing 45: register.js

```

1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateAdminCreateInput(data) {
5     let errors = {};
6
7     data.email = !isEmpty(data.email) ? data.email : "";
8     data.password = !isEmpty(data.password) ? data.password : "";
9
10    if (Validator.isEmpty(data.email)) {
11        errors.email = "Email field is required";
12    }
13
14    if (!Validator.isEmail(data.email)) {
15        errors.email = "Email is invalid";
16    }
17

```

```

18     if (Validator.isEmpty(data.password)) {
19         errors.password = "Password field is required";
20     }
21     if (
22         !Validator.isLength(data.password, {
23             min: 6,
24             max: 30
25         })
26     ) {
27         errors.password = "Password must be at least 6 characters";
28     }
29
30     return {
31         errors,
32         isValid: isEmpty(errors)
33     };
34 }

```

Listing 46: subcomponents.js

```

1 const Validator = require("validator");
2 const isEmpty = require("./is-empty");
3
4 module.exports = function validateSubcomponent(data) {
5     let errors = {};
6     data.name = !isEmpty(data.name) ? data.name : "";
7     if (!Validator.isLength(data.name, { min: 2, max: 255 })) {
8         errors.msg = "Subcomponent name must be between 2 and 255
9             characters";
10    }
11    if (Validator.isEmpty(data.name)) {
12        errors.msg = "Subcomponent name field is required";
13    }
14    return {
15        errors,
16        isValid: isEmpty(errors)
17    };
17 }

```

Listing 47: server.js

```

1 // Import Dependencies
2
3 const express = require("express");
4 const bodyParser = require("body-parser");
5 const passport = require("passport");
6 const cors = require("cors");
7
8 // Import Models (Database table names)
9 const databaseTable = require("./config/erd");
10 const app = express();
11
12 //Enable CORS policy
13 app.use(cors());
14 // Body-parser middleware
15 app.use(
16     bodyParser.urlencoded({
17         extended: false
18     })
19 );
20 app.use(bodyParser.json());
21
22 // MySQL Connection Configuration
23 const sequelize = require("./config/db/database");
24 sequelize
25     .authenticate()
26     .then(() => console.log("Connection has been established
27         successfully."))
28     .catch(err => console.error("Unable to connect to the database."))
28

```

```
29 // Create/Sync MySQL table names from models
30 databaseTable.init();
31
32 // Passport middleware initialization
33 app.use(passport.initialize());
34 require("./config/passport")(passport);
35
36 // Routers
37 const users = require("./routes/api/users");
38 const admin = require("./routes/api/admin");
39 const teacher = require("./routes/api/teacher");
40 const director = require("./routes/api/director");
41 const registrar = require("./routes/api/registrar");
42 const parent = require("./routes/api/parent");
43 const cashier = require("./routes/api/cashier");
44 const student = require('./routes/api/student')
45 app.use("/api/users", users);
46 app.use("/api/admin", admin);
47 app.use("/api/teacher", teacher);
48 app.use("/api/director", director);
49 app.use("/api/registrar", registrar);
50 app.use("/api/parent", parent);
51 app.use("/api/cashier", cashier);
52 app.use('/api/student', student)
53 app.use(express.static(__dirname + "/public"));
54
55 const port = process.env.PORT || 5000;
56
57 app.listen(port, () => console.log("Server is running on localhost
:5000"));
```

Listing 48: redux actions (compiled in one source code)

```

1   export { setCurrentUser } from
2     './setCurrentUser';          29
3   export { sampleAction } from
4     './sampleAction';           30
5   export { logoutUser } from
6     './logoutUser';             31
7   export { setLoading,
8     setLoadingFalse,
9     setLoadingTrue } from
10    './setLoading';            32
11  export { setLoading' };
12  export { loginUser } from
13    './loginUser';              34
14  export { getErrors } from
15    './getErrors';
16  export { getCurrentProfile } from
17    './getCurrentProfile';      35
18  export { updateAdminProfile } from
19    './updateAdminProfile';     36
20  export { changePassword } from
21    './changePassword';         37
22  export { createAccount } from
23    './createAccount';          38
24  export { getAccountList } from
25    './getAccountList';         39
26  export { activateAccount } from
27    './activateAccount';        40
28  export { deactivateAccount } from
29    './deactivateAccount';      41
30  export { updateDirectorProfile } from
31    './
32    updateDirectorProfile';
33  export { updateTeacherProfile } from
34    './
35    updateTeacherProfile';      43
36  export { updateRegistrarProfile } from
37    './
38    updateRegistrarProfile';    44
39  export { updateParentProfile } from
40    './updateParentProfile'
41    ;
42  export { addSection } from
43    './
44    addSection';
45  export { editSection } from
46    './
47    editSection';
48  export { deleteSection } from
49    './
50    deleteSection';
51  export { createStudentSection } from
52    './
53    createStudentSection';      50
54  export { deleteStudentSection } from
55    './
56    deleteStudentSection';      51
57  export { assignAdvisorySection } from
58    './
59    assignAdvisorySection';     52
60  export {
61    unassignAdvisorySection } from
62    './
63    unassignAdvisorySection';   53
64  export { createSubjectSection } from
65    './
66    createSubjectSection';      54
67  export { deleteSubjectSection } from
68    './
69    deleteSubjectSection';      55
70  export {
71    addSubjectSectionStudent } from
72    './
73    addSubjectSectionStudent';
74  export {
75    deleteSubjectSectionStudent } from
76    './
77    deleteSubjectSectionStudent';
78  export {
79    addNewSubcomponent } from
80    './
81    addNewSubcomponent';
82  export { editSubcomponent } from
83    './
84    editSubcomponent';
85  export { deleteSubcomponent } from
86    './
87    deleteSubcomponent';
88  export { addNewRecord } from
89    './
90    addNewRecord';
91  export { deleteRecord } from
92    './
93    deleteRecord';
94  export { changeTransmutation } from
95    './
96    changeTransmutation';
97  export { editRecord } from
98    './
99    editRecord';
100 export { setDeadlineAll } from
101   './
102   setDeadlineAll';
103 export { setDeadline } from
104   './
105   setDeadline';
106 export { removeDeadline } from
107   './
108   removeDeadline';
109 export { createSchoolYear } from
110   './
111   createSchoolYear';
112 export { changeQuarterSy } from
113   './
114   changeQuarterSy';
115 export { endSchoolYear } from
116   './
117   endSchoolYear';
118 export {
119   addSubjectSectionStudentBulk
120   }
121   from './
122   addSubjectSectionStudentBulk';
123 export { submitClassRecord } from
124   './
125   submitClassRecord';
126 export { postClassRecord } from
127   './
128   postClassRecord';
129 export { revertClassRecord } from
130   './
131   revertClassRecord';
132 export { generatePdfStudent } from
133   './
134   generatePdfStudent';
135 export { generatePdfSection } from
136   './
137   generatePdfSection';
138 export { restrictAccount } from
139   './
140   restrictAccount';
141 export { unrestrictAccount } from
142   './
143   unrestrictAccount';
144 export { updateCashierProfile } from
145   './
146   updateCashierProfile';
147 export { updateStudentProfile } from
148   './
149   updateStudentProfile';
150 export { assignParent } from
151   './
152   assignParent';
153 export { unassignParent } from
154   './
155   unassignParent';
156 // Rekit uses a new approach to
157 // organizing actions and
158 // reducers. That is
159 // putting related actions and
160 // reducers in one file. See
161 // more at:
162 // https://medium.com/
163 @nate_wang/a-new-approach-

```

```

      for-managing-redux-actions 100
      -91c26ce8b5da
57      import { APP_ACTIVATE_ACCOUNT,
58          APP_GET_ERRORS } from './ 102
      constants';
59      import axios from 'axios';
60      import * as actions from './ 103
      actions';
61      import { message } from 'antd' 104
62
63      export const activateAccount = 105
          userData => dispatch => { 106
              axios
                  .post('api/admin/activate',
                      userData)
                  .then(res => { 107
                      message.success('The
                          account has been 108
                          activated successfully
                          !');
                      dispatch({ type:
                          APP_ACTIVATE_ACCOUNT 111
                      });
                  })
                  .catch(err => { 113
                      dispatch({ type:
                          APP_GET_ERRORS, 114
                          payload: err.response,
                          data });
                  });
              });
59      export function reducer(state, 121
          action) {
76          switch (action.type) { 123
77              case APP_ACTIVATE_ACCOUNT:
78                  return {
79                      ...state,
80                  };
81
82              default: 125
83                  return state;
84          }
85      }
86      // Rekit uses a new approach to 126
      organizing actions and
      reducers. That is
87      // putting related actions and 127
      reducers in one file. See
      more at:
88      // https://medium.com/ 129
      @nate_wang/a-new-approach- 130
      for-managing-redux-actions 131
      -91c26ce8b5da 132
89
90      import { APP_ADD_NEW_RECORD,
91          APP_GET_ERRORS } from './ 133
      constants';
92      import * as actions from './ 134
      actions';
93      import axios from 'axios'; 135
94      import { message } from 'antd'; 136
95
96      export const addNewRecord = ( 137
          data, position) => dispatch 138
          => {
97          dispatch(actions.setLoading(139
              true));
98          axios
              .post('api/${position.
                  toLowerCase()}/
                  addnewrecord', data) 141
              .then(res => { 142

```

```

      message.success(res.data.
          msg);
      dispatch(actions.
          setLoading(false));
      dispatch({ type:
          APP_GET_ERRORS,
          payload: {} });
      dispatch({ type:
          APP_ADD_NEW_RECORD });
    })
    .catch(err => {
        dispatch({ type:
            APP_GET_ERRORS,
            payload: err.response.
            data });
        dispatch(actions.
            setLoading(false));
        message.error(err.
            response.data.msg);
    });
}

export function reducer(state,
    action) {
    switch (action.type) {
        case APP_ADD_NEW_RECORD:
            return {
                ...state,
            };
        default:
            return state;
    }
}
// Rekit uses a new approach to
// organizing actions and
// reducers. That is
// putting related actions and
// reducers in one file. See
// more at:
// https://medium.com/
// @nate_wang/a-new-approach-
// for-managing-redux-actions
// -91c26ce8b5da

import {
    APP_ADD_NEW_SUBCOMPONENT,
    APP_GET_ERRORS } from './
    constants';
import * as actions from './
    actions';
import axios from 'axios';
import { message } from 'antd';

export const addNewSubcomponent
    = (data, position) =>
        dispatch => {
        dispatch(actions.setLoading(
            true));
        if (position == 'Teacher') {
            axios
                .post('api/teacher/
                    addnewsubcomp', data)
                .then(res => {
                    message.success(res.
                        data.msg);
                    dispatch(actions.
                        setLoading(false));
                    dispatch({ type:
                        APP_GET_ERRORS,
                        payload: {} });
                })
                .catch(err => {

```

```

143         dispatch({ type: 186
144             APP_GET_ERRORS,
145             payload: err.
146             response.data}); 188
147             dispatch(actions.
148                 setLoading(false));
149             message.error(err.
150                 response.data.msg);
151             });
152         } else {
153             axios
154                 .post('api/registrar/
155                     addnewsubcomp', data)
156                 .then(res => {
157                     message.success(res.
158                         data.msg);
159                     dispatch(actions.
160                         setLoading(false));
161                     dispatch({ type:
162                         APP_GET_ERRORS,
163                         payload: {} });
164                     });
165                 .catch(err => {
166                     dispatch({ type:
167                         APP_GET_ERRORS,
168                         payload: err.
169                         response.data });
170                     dispatch(actions.
171                         setLoading(false));
172                     message.error(err.
173                         response.data.msg);
174                 });
175             );
176         });
177     );
178     export function reducer(state,
179         action) {
180         switch (action.type) {
181             case
182                 APP_ADD_NEW_SUBCOMPONENT
183                 :
184                     return {
185                         ...state,
186                     };
187                 default:
188                     return state;
189             }
190         // Rekit uses a new approach to
191         // organizing actions and
192         // reducers. That is
193         // putting related actions and
194         // reducers in one file. See
195         // more at:
196         // https://medium.com/
197         // @nate_wang/a-new-approach-
198         // for-managing-redux-actions
199         // -91c26ce8b5da
200         // Rekit uses a new approach to
201         // organizing actions and
202         // reducers. That is
203         // putting related actions and
204         // reducers in one file. See
205         // more at:
206         // https://medium.com/
207         // @nate_wang/a-new-approach-
208         // for-managing-redux-actions
209         // -91c26ce8b5da
210         import { APP_ADD_SECTION,
211             APP_GET_ERRORS } from './
212             constants';
213         import * as actions from './
214             actions';
215         import axios from 'axios';
216         import { message } from 'antd';
217         export const addSection = data
218             => dispatch => {
219                 dispatch(actions.setLoading(
220                     true));
221                 dispatch({ type:
222                     APP_ADD_SECTION_STUDENT
223                     });
224                 axios
225                     .post('api/registrar/
226                         addsubsectstud', data)
227                     .then(res => {
228                         dispatch(actions.
229                             setLoading(false));
230                     });
231             );
232         export function reducer(state,
233             action) {
234             switch (action.type) {
235                 case APP_ADD_SECTION:
236                     return {
237                         ...state,
238                     };
239                 default:
240                     return state;
241             }
242         }
243         // Rekit uses a new approach to
244         // organizing actions and
245         // reducers. That is
246         // putting related actions and
247         // reducers in one file. See
248         // more at:
249         // https://medium.com/
250         // @nate_wang/a-new-approach-
251         // for-managing-redux-actions
252         // -91c26ce8b5da
253         import {
254             APP_ADD_SUBJECT_SECTION_STUDENT
255             , APP_GET_ERRORS } from './
256             constants';
257         import * as actions from './
258             actions';
259         import axios from 'axios';
260         import { message } from 'antd';
261         export const addSubjectSectionStudent =
262             data => dispatch => {
263                 dispatch(actions.setLoading(
264                     true));
265                 dispatch({ type:
266                     APP_ADD_SUBJECT_SECTION_STUDENT
267                     });
268                 axios
269                     .post('api/registrar/
270                         addsubsectstud', data)
271                     .then(res => {
272                         dispatch(actions.
273                             setLoading(false));
274                     });
275             );
276         export function reducer(state,
277             action) {
278             switch (action.type) {
279                 case APP_ADD_SUBJECT_SECTION_STUDENT:
280                     return {
281                         ...state,
282                     };
283                 default:
284                     return state;
285             }
286         }
287     );
288 
```

```

229     message.success('Subject268
230         load added
231         successfully!');    269
232     dispatch({ type:
233         APP_GET_ERRORS,
234         payload: {} });
235     });
236     .catch(err => {
237         dispatch(actions.
238             setLoading(false));
239         dispatch({ type:
240             APP_GET_ERRORS,
241             payload: err.response
242             data });
243         message.error(err.
244             response.data.msg);
245     });
246     );
247     );
248     );
249     );
250 // Rekit uses a new approach to
251 // organizing actions and
252 // reducers. That is
253 // putting related actions and
254 // reducers in one file. See
255 // more at:
256 // https://medium.com/
257 // @nate_wang/a-new-approach-
258 // for-managing-redux-actions
259 -91c26ce8b5da
260 import {
261     APP_ADD SUBJECT SECTION STUDEN
262     , APP_GET_ERRORS } from './
263     constants';
264 import * as actions from './
265     actions';
266 import axios from 'axios';
267 import { message } from 'antd';
268 export const
269     addSubjectSectionStudentBulk
270     = data => async dispatch
271     {
272         dispatch(actions.setLoading(
273             true));
274         dispatch({ type:
275             APP_ADD SUBJECT SECTION STUDEN
276             });
277     );
278     );
279     );
280 export function reducer(state,
281     action) {
282     switch (action.type) {
283         case
284             APP_ADD SUBJECT SECTION STUDEN
285             :
286             return {
287                 ...state,
288             };
289         default:
290             return state;
291     }
292     );
293     );
294     );
295     );
296     );
297     );
298     );
299     );
300     );
301     );
302     );
303     );
304     );
305     );
306     );
307     );
308     );
309     );
310     );
311     );
312     );
313     );
314     );
315     );
316     );
317     );
318     );
319     );
320     );
321     );
322     );
323     );
324     );
325     );
326     );
327     );
328     );
329     );
330     );
331     );
332     );
333     );
334     );
335     );
336     );
337     );
338     );
339     );
340     );
341     );
342     );
343     );
344     );
345     );
346     );
347     );
348     );
349     );
350     );
351     );
352     );
353     );
354     );
355     );
356     );
357     );
358     );
359     );
360     );
361     );
362     );
363     );
364     );
365     );
366     );
367     );
368     );
369     );
370     );
371     );
372     );
373     );
374     );
375     );
376     );
377     );
378     );
379     );
380     );
381     );
382     );
383     );
384     );
385     );
386     );
387     );
388     );
389     );
390     );
391     );
392     );
393     );
394     );
395     );
396     );
397     );
398     );
399     );
400     );
401     );
402     );
403     );
404     );
405     );
406     );
407     );
408     );
409     );
410     );
411     );
412     );
413     );
414     );
415     );
416     );
417     );
418     );
419     );
420     );
421     );
422     );
423     );
424     );
425     );
426     );
427     );
428     );
429     );
430     );
431     );
432     );
433     );
434     );
435     );
436     );
437     );
438     );
439     );
440     );
441     );
442     );
443     );
444     );
445     );
446     );
447     );
448     );
449     );
450     );
451     );
452     );
453     );
454     );
455     );
456     );
457     );
458     );
459     );
460     );
461     );
462     );
463     );
464     );
465     );
466     );
467     );
468     );
469     );
470     );
471     );
472     );
473     );
474     );
475     );
476     );
477     );
478     );
479     );
480     );
481     );
482     );
483     );
484     );
485     );
486     );
487     );
488     );
489     );
490     );
491     );
492     );
493     );
494     );
495     );
496     );
497     );
498     );
499     );
500     );
501     );
502     );
503     );
504     );
505     );
506     );
507     );
508     );
509     );
510     );
511     );
512     );
513     );
514     );
515     );
516     );
517     );
518     );
519     );
520     );
521     );
522     );
523     );
524     );
525     );
526     );
527     );
528     );
529     );
530     );
531     );
532     );
533     );
534     );
535     );
536     );
537     );
538     );
539     );
540     );
541     );
542     );
543     );
544     );
545     );
546     );
547     );
548     );
549     );
550     );
551     );
552     );
553     );
554     );
555     );
556     );
557     );
558     );
559     );
560     );
561     );
562     );
563     );
564     );
565     );
566     );
567     );
568     );
569     );
570     );
571     );
572     );
573     );
574     );
575     );
576     );
577     );
578     );
579     );
580     );
581     );
582     );
583     );
584     );
585     );
586     );
587     );
588     );
589     );
590     );
591     );
592     );
593     );
594     );
595     );
596     );
597     );
598     );
599     );
599     );
600     );
601     );
602     );
603     );
604     );
605     );
606     );
607     );
608     );
609     );
609     );
610     );
611     );
612     );
613     );
614     );
615     );
616     );
617     );
618     );
619     );
619     );
620     );
621     );
622     );
623     );
624     );
625     );
626     );
627     );
628     );
629     );
629     );
630     );
631     );
632     );
633     );
634     );
635     );
636     );
637     );
638     );
639     );
639     );
640     );
641     );
642     );
643     );
644     );
645     );
646     );
647     );
648     );
649     );
649     );
650     );
651     );
652     );
653     );
654     );
655     );
656     );
657     );
658     );
659     );
659     );
660     );
661     );
662     );
663     );
664     );
665     );
666     );
667     );
668     );
669     );
669     );
670     );
671     );
672     );
673     );
674     );
675     );
676     );
677     );
678     );
679     );
679     );
680     );
681     );
682     );
683     );
684     );
685     );
686     );
687     );
688     );
689     );
689     );
690     );
691     );
692     );
693     );
694     );
695     );
696     );
697     );
698     );
699     );
699     );
700     );
701     );
702     );
703     );
704     );
705     );
706     );
707     );
708     );
709     );
709     );
710     );
711     );
712     );
713     );
714     );
715     );
716     );
717     );
718     );
719     );
719     );
720     );
721     );
722     );
723     );
724     );
725     );
726     );
727     );
728     );
729     );
729     );
730     );
731     );
732     );
733     );
734     );
735     );
736     );
737     );
738     );
739     );
739     );
740     );
741     );
742     );
743     );
744     );
745     );
746     );
747     );
748     );
749     );
749     );
750     );
751     );
752     );
753     );
754     );
755     );
756     );
757     );
758     );
759     );
759     );
760     );
761     );
762     );
763     );
764     );
765     );
766     );
767     );
768     );
769     );
769     );
770     );
771     );
772     );
773     );
774     );
775     );
776     );
777     );
778     );
779     );
779     );
780     );
781     );
782     );
783     );
784     );
785     );
786     );
787     );
788     );
789     );
789     );
790     );
791     );
792     );
793     );
794     );
795     );
796     );
797     );
798     );
799     );
799     );
800     );
801     );
802     );
803     );
804     );
805     );
806     );
807     );
808     );
809     );
809     );
810     );
811     );
812     );
813     );
814     );
815     );
816     );
817     );
818     );
819     );
819     );
820     );
821     );
822     );
823     );
824     );
825     );
826     );
827     );
828     );
829     );
829     );
830     );
831     );
832     );
833     );
834     );
835     );
836     );
837     );
838     );
839     );
839     );
840     );
841     );
842     );
843     );
844     );
845     );
846     );
847     );
848     );
849     );
849     );
850     );
851     );
852     );
853     );
854     );
855     );
856     );
857     );
858     );
859     );
859     );
860     );
861     );
862     );
863     );
864     );
865     );
866     );
867     );
868     );
869     );
869     );
870     );
871     );
872     );
873     );
874     );
875     );
876     );
877     );
878     );
879     );
879     );
880     );
881     );
882     );
883     );
884     );
885     );
886     );
887     );
888     );
889     );
889     );
890     );
891     );
892     );
893     );
894     );
895     );
896     );
897     );
898     );
899     );
899     );
900     );
901     );
902     );
903     );
904     );
905     );
906     );
907     );
908     );
909     );
909     );
910     );
911     );
912     );
913     );
914     );
915     );
916     );
917     );
918     );
919     );
919     );
920     );
921     );
922     );
923     );
924     );
925     );
926     );
927     );
928     );
929     );
929     );
930     );
931     );
932     );
933     );
934     );
935     );
936     );
937     );
938     );
939     );
939     );
940     );
941     );
942     );
943     );
944     );
945     );
946     );
947     );
948     );
949     );
949     );
950     );
951     );
952     );
953     );
954     );
955     );
956     );
957     );
958     );
959     );
959     );
960     );
961     );
962     );
963     );
964     );
965     );
966     );
967     );
968     );
969     );
969     );
970     );
971     );
972     );
973     );
974     );
975     );
976     );
977     );
978     );
979     );
979     );
980     );
981     );
982     );
983     );
984     );
985     );
986     );
987     );
988     );
989     );
989     );
990     );
991     );
992     );
993     );
994     );
995     );
996     );
997     );
998     );
999     );
999     );

```

```
307     teacherName); 354
308   });
309   );
310   export function reducer(state, 355
311     action) { 356
312       switch (action.type) { 357
313         case APP_ASSIGN_ADVISORY_SECTIO: 358
314           : 359
315             return { 360
316               ...state, 361
317             };
318           default: 362
319             return state; 363
320         }
321       // Rekit uses a new approach to 364
322         organizing actions and 365
323         reducers. That is 366
324       // putting related actions and 367
325         reducers in one file. See 368
326         more at: 369
327       // https://medium.com/ 367
328         @nate_wang/a-new-approach- 368
329         for-managing-redux-actions 369
330         -91c26ce8b5da
331
332 import { APP_ASSIGN_PARENT, 370
333   APP_GET_ERRORS } from './ 371
334   constants';
335 import * as actions from './ 372
336   actions';
337 import { message } from 'antd'; 372
338 import axios from 'axios'; 373
339
340 export const assignParent = 373
341   data => dispatch => { 374
342     dispatch({ type: 374
343       APP_ASSIGN_PARENT }); 375
344     dispatch(actions.setLoading( 375
345       true)); 376
346     axios 376
347       .post('api/admin/ 377
348         assignparent', data) 377
349       .then(res => { 378
350         message.success(res.data. 378
351           msg); 379
352         dispatch(actions. 379
353           setLoading(false)); 379
354       })
355       .catch(err => { 380
356         message.error(err. 380
357           response.data.msg); 381
358         dispatch(actions. 381
359           setLoading(false)); 382
360         dispatch({ type: 382
361           APP_GET_ERRORS, 383
362             payload: err.response. 383
363               data }); 384
364       });
365   };
366
367 export function reducer(state, 392
368   action) { 393
369   switch (action.type) { 394
370     case APP_ASSIGN_PARENT: 394
371       return { 395
372         ...state, 395
373       };
374     default: 396
375   }
376 }
377
378 // Rekit uses a new approach to 396
379   organizing actions and 397
380   reducers. That is 398
381 // putting related actions and 399
382   reducers in one file. See 400
383   more at: 401
384 // https://medium.com/ 401
385   @nate_wang/a-new-approach- 402
386   for-managing-redux-actions 403
387   -91c26ce8b5da
388
389 import { APP_CHANGE_PASSWORD, 404
390   APP_GET_ERRORS } from './ 405
391   constants';
392 import * as actions from './ 406
393   actions';
394 import axios from 'axios'; 407
395
396 export const changePassword = 407
397   userData => dispatch => { 408
398     dispatch(actions.setLoading( 408
399       true)); 410
400     axios 410
401       .post('api/users/ 411
402         changepassword', 411
403           userData) 412
404       .then(res => { 412
405         message.success('You have 412
406           successfully updated 413
407             your password!'); 414
408         dispatch(actions. 414
409           setLoading(false)); 415
410         dispatch({ type: 415
411           APP_CHANGE_PASSWORD }); 416
412       });
413       .catch(err => { 416
414         dispatch(actions. 416
415           setLoading(false)); 417
416         dispatch({ type: 417
417           APP_GET_ERRORS, 418
418             payload: err.response. 418
419               data }); 420
421       });
422     });
423
424 export function reducer(state, 424
425   action) { 425
426   switch (action.type) { 426
427     case APP_CHANGE_PASSWORD: 427
428       return { 428
429         ...state, 429
430       };
431     default: 430
432       return state; 431
433     }
434   }
435
436 // Rekit uses a new approach to 435
437   organizing actions and 438
438   reducers. That is 439
439 // putting related actions and 440
440   reducers in one file. See 441
441   more at: 442
442 // https://medium.com/ 442
443   @nate_wang/a-new-approach- 443
444   for-managing-redux-actions 444
445   -91c26ce8b5da
```

```

        for-managing-redux-actions 440
          -91c26ce8b5da
397      441
398      import { APP_CHANGE_QUARTER_SY 442
          APP_GET_ERRORS } from './
          constants';
399      import * as actions from './
          actions'; 443
400      import { message } from 'antd' 444
401      import axios from 'axios';
402
403      export const changeQuarterSy = 445
          data => dispatch => {
404          dispatch({ type:
              APP_CHANGE_QUARTER_SY });
405          dispatch(actions.setLoading( 447
              true));
406          axios
407            .post('api/registrar/
              changequartersy', data) 448
408            .then(res => {
              message.success(res.data.
                  msg);
409            dispatch(actions.
              setLoading(false)); 451
410        })
411        .catch(err => { 452
412          message.error(err.
              response.data.msg); 453
413          dispatch(actions.
              setLoading(false)); 455
414          dispatch({ type:
              APP_GET_ERRORS ,
              payload: err.response 457
              data }); 458
415      });
416    );
417  );
418
419  export function reducer(state, 460
          action) { 461
420    switch (action.type) { 462
421      case APP_CHANGE_QUARTER_SY 463
422        return { 464
423          ...state, 465
424        }; 466
425      default: 467
426        return state;
427    }
428  }
429 // Rekit uses a new approach to 469
        organizing actions and 470
        reducers. That is
430 // putting related actions and 471
        reducers in one file. See
        more at:
431 // https://medium.com/
        @nate_wang/a-new-approach-473
        for-managing-redux-actions
        -91c26ce8b5da
432
433 import {
434   APP_CHANGE_TRANSMUTATION ,
        APP_GET_ERRORS } from './ 475
        constants';
435 import * as actions from './
        actions'; 476
436 import axios from 'axios';
437 import { message } from 'antd';
438
439 export const
        changeTransmutation = (data,
          position) => dispatch => {
dispatch(actions.setLoading(
  true));
axios
  .post('api/${position.
    toLowerCase()}/
    changetransmutation',
  data)
  .then(res => {
    message.success(res.data.
      msg);
    dispatch(actions.
      setLoading(false));
    dispatch({ type:
      APP_GET_ERRORS ,
      payload: {} });
    dispatch({ type:
      APP_CHANGE_TRANSMUTATION
    });
  })
  .catch(err => {
    dispatch({ type:
      APP_GET_ERRORS ,
      payload: err.response.
        data });
    dispatch(actions.
      setLoading(false));
    message.error(err.
      response.data.msg);
  });
}

export function reducer(state,
  action) {
  switch (action.type) {
    case APP_CHANGE_TRANSMUTATION
      :
        return {
          ...state,
        };
    default:
        return state;
  }
}

export const
  APP_SET_CURRENT_USER = ,
  APP_SET_CURRENT_USER';
export const APP_SAMPLE_ACTION
  = 'APP_SAMPLE_ACTION';
export const APP_LOGOUT_USER =
  'APP_LOGOUT_USER';
export const APP_SET_LOADING =
  'APP_SET_LOADING';
export const APP_LOGIN_USER =
  'APP_LOGIN_USER';
export const APP_GET_ERRORS =
  'APP_GET_ERRORS';
export const
  APP_GET_CURRENT_PROFILE = ,
  APP_GET_CURRENT_PROFILE';
export const
  APP_UPDATE_ADMIN_PROFILE = ,
  APP_UPDATE_ADMIN_PROFILE';
export const
  APP_CHANGE_PASSWORD = ,
  APP_CHANGE_PASSWORD';
export const APP_CREATE_ACCOUNT
  = 'APP_CREATE_ACCOUNT';
export const
  APP_GET_ACCOUNT_LIST = ,
  APP_GET_ACCOUNT_LIST';
export const
  APP_ACTIVATE_ACCOUNT = ,
  APP_ACTIVATE_ACCOUNT';

```

```

479     export const          'APP_EDIT_RECORD';
480     APP_DEACTIVATE_ACCOUNT = '502
481     APP_DEACTIVATE_ACCOUNT';
482     export const          APP_UPDATE_DIRECTOR_PROFILE03
483     = ',           APP_UPDATE_DIRECTOR_PROFILE04
484     ;           export const          APP_UPDATE_TEACHER_PROFILE05
485     'APP_UPDATE_TEACHER_PROFILE
486     ';
487     export const          506
488     APP_UPDATE_REGISTRAR_PROFILE
489     = ',           APP_UPDATE_REGISTRAR_PROFILE07
490     ';
491     export const          APP_UPDATE_PARENT_PROFILE 508
492     'APP_UPDATE_PARENT_PROFILE';
493     export const APP_ADD_SECTION =
494     'APP_ADD_SECTION';
495     export const APP_EDIT_SECTION =
496     'APP_EDIT_SECTION';           509
497     export const APP_DELETE_SECTION
498     = 'APP_DELETE_SECTION';
499     export const          510
500     APP_CREATE_STUDENT_SECTION =
501     'APP_CREATE_STUDENT_SECTION
502     ';
503     export const          APP_DELETE_STUDENT_SECTION =
504     'APP_DELETE_STUDENT_SECTION02
505     ';
506     export const          APP_ASSIGN_ADVISORY_SECTION13
507     = ',           APP_ASSIGN_ADVISORY_SECTION
508     ';
509     export const          APP_UNASSIGN_ADVISORY_SECTION
510     = ',           APP_UNASSIGN_ADVISORY_SECTION02
511     ';
512     export const          APP_CREATE_SUBJECT_SECTION =
513     'APP_CREATE_SUBJECT_SECTION
514     ';
515     export const          APP_DELETE_SUBJECT_SECTION =
516     'APP_DELETE_SUBJECT_SECTION
517     ';
518     export const          APP_ADD_SUBJECT_SECTION_STUDEN
519     = ',           APP_ADD_SUBJECT_SECTION_STUDEN'
520     ;
521     export const          APP_DELETE_SUBJECT_SECTION_STU
522     = ',           APP_DELETE_SUBJECT_SECTION_STU'
523     ;
524     export const          APP_ADD_NEW_SUBCOMPONENT =522
525     APP_ADD_NEW_SUBCOMPONENT';
526     export const APP_EDIT_SUBCOMPONENT =
527     'APP_EDIT_SUBCOMPONENT';
528     export const          APP_DELETE_SUBCOMPONENT =
529     'APP_DELETE_SUBCOMPONENT';
530     export const APP_ADD_NEW_RECORD =
531     'APP_ADD_NEW_RECORD';           525
532     export const APP_DELETE_RECORD
533     = 'APP_DELETE_RECORD';           526
534     export const          APP_CHANGE_TRANSMUTATION =528
535     APP_CHANGE_TRANSMUTATION';
536     export const APP_EDIT_RECORD =529
537     'APP_SET_DEADLINE_ALL';
538     APP_SET_DEADLINE_ALL = '
539     APP_SET_DEADLINE_ALL';
540     export const APP_SET_DEADLINE =
541     'APP_SET_DEADLINE';
542     export const          APP_REMOVE_DEADLINE =
543     'APP_REMOVE_DEADLINE';
544     export const          APP_CREATE SCHOOL_YEAR =
545     'APP_CREATE SCHOOL_YEAR';
546     export const          APP_CHANGE_QUARTER_SY =
547     'APP_CHANGE_QUARTER_SY';
548     export const          APP_END SCHOOL_YEAR =
549     'APP_END SCHOOL_YEAR';
550     export const APP_ADD SUBJECT SECTION STUDENT BULK
551     = ,
552     APP_ADD SUBJECT SECTION STUDENT BULK
553     ;
554     export const          APP_SUBMIT CLASS RECORD =
555     'APP_SUBMIT CLASS RECORD';
556     export const          APP_POST CLASS RECORD =
557     'APP_POST CLASS RECORD';
558     export const          APP_REVERT CLASS RECORD =
559     'APP_REVERT CLASS RECORD';
560     export const          APP_GENERATE PDF STUDENT =
561     'APP_GENERATE PDF STUDENT';
562     export const          APP_GENERATE PDF SECTION =
563     'APP_GENERATE PDF SECTION';
564     export const          APP_RESTRICT ACCOUNT =
565     'APP_RESTRICT ACCOUNT';
566     export const          APP_UNRESTRICT ACCOUNT =
567     'APP_UNRESTRICT ACCOUNT';
568     export const          APP_UPDATE CASHIER PROFILE =
569     'APP_UPDATE CASHIER PROFILE
570     ';
571     export const          APP_UPDATE STUDENT PROFILE =
572     'APP_UPDATE STUDENT PROFILE
573     ';
574     export const APP_ASSIGN PARENT
575     = 'APP_ASSIGN PARENT';
576     export const          APP_UNASSIGN PARENT =
577     'APP_UNASSIGN PARENT';
578     // Rekit uses a new approach to
579     // organizing actions and
580     // reducers. That is
581     // putting related actions and
582     // reducers in one file. See
583     // more at:
584     // https://medium.com/
585     @nate_wang/a-new-approach-
586     for-managing-redux-actions
587     -91c26ce8b5da
588     import { APP_CREATE_ACCOUNT,
589     APP_GET_ERRORS } from './
590     constants';
591     import * as actions from './
592     actions';
593     import axios from 'axios';
594     import { message } from 'antd';
595     export const createAccount =
596     userData => dispatch => {
597     dispatch(actions.setLoading(

```



```

615             showModal: true,
616             selectedStudSectID657
617                 res.data.
618                     studsectID,
619                     selectedSubsect: 658
620                         [],
621                         659
622                     );
623             660
624         );
625             .catch(err => {
626                 dispatch(actions.
627                     setLoading(false));
628                 dispatch({ type:
629                     APP_GET_ERRORS,
630                     payload: err.response
631                     data });
632                 message.error(err.
633                     response.data.
634                     studentName);
635             });
636             667
637         );
638     );
639     668
640     export function reducer(state,669
641         action) {
642             switch (action.type) { 670
643                 case
644                     APP_CREATE_STUDENT_SECTION
645                         :
646                         674
647                         return {
648                             ...state,
649                         };
650                         675
651                         676
652                         677
653                         default:
654                             return state;
655                         678
656                     }
657                     // Rekit uses a new approach to
658                     // organizing actions and
659                     // reducers. That is
660                     // putting related actions and
661                     // reducers in one file. See
662                     // more at:
663                     // https://medium.com/
664                     // @nate_wang/a-new-approach-
665                     // for-managing-redux-actions
666                     -91c26ce8b5da
667                     679
668                     import {
669                         APP_CREATE_SUBJECT_SECTION680
670                         APP_GET_ERRORS } from './681
671                         constants';
672                     import * as actions from './682
673                         actions';
674                     import axios from 'axios';
675                     import { message } from 'antd';
676                     683
677                     684
678                     export const
679                         createSubjectSection = data
680                         => dispatch => {
681                             dispatch(actions.setLoading(681
682                             true));
683                             dispatch({ type:
684                             APP_CREATE_SUBJECT_SECTION
685                             });
686                             693
687                             axios
688                             .post('api/registrar/
689                                 createsubjectsection',
690                                 data)
691                             .then(res => {
692                                 695
693                                 dispatch(actions.
694                                     setLoading(false));
695                                 message.success('Subject
696                                     load added
697                                     successfully!'));
698                                     698
699                                     dispatch({ type:
700                                         APP_GET_ERRORS,
701                                         payload: {} });
702                                     }
703                                     .catch(err => {
704                                         dispatch(actions.
705                                             setLoading(false));
706                                         dispatch({ type:
707                                             APP_GET_ERRORS,
708                                             payload: err.response.
709                                             data });
710                                         message.error(err.
711                                             response.data.msg);
712                                         });
713                                     }
714                                     );
715                                     export function reducer(state,
716                                         action) {
717                                         switch (action.type) {
718                                             case
719                                                 APP_CREATE_SUBJECT_SECTION
720                                                 :
721                                                 return {
722                                                     ...state,
723                                                 };
724                                             default:
725                                                 return state;
726                                         }
727                                         // Rekit uses a new approach to
728                                         // organizing actions and
729                                         // reducers. That is
730                                         // putting related actions and
731                                         // reducers in one file. See
732                                         // more at:
733                                         // https://medium.com/
734                                         // @nate_wang/a-new-approach-
735                                         // for-managing-redux-actions
736                                         -91c26ce8b5da
737                                         import { APP_DEACTIVATE_ACCOUNT
738                                             , APP_GET_ERRORS } from './
739                                             constants';
740                                         import * as actions from './
741                                             actions';
742                                         import { message } from 'antd';
743                                         import axios from 'axios';
744                                         export const deactivateAccount
745                                             = userData => dispatch => {
746                                             axios
747                                                 .post('api/admin/deactivate
748                                                 ', userData)
749                                                 .then(res => {
750                                                     message.success('The
751                                                         account has been
752                                                         deactivated
753                                                         successfully!'));
754                                                     dispatch({ type:
755                                                         APP_DEACTIVATE_ACCOUNT
756                                                         });
757                                                 })
758                                                 .catch(err => {
759                                                     dispatch({ type:
760                                                         APP_GET_ERRORS,
761                                                         payload: err.response.
762                                                         data });
763                                                 });
764                                         }
765                                         export function reducer(state,
766                                         action) {

```

```

699     switch (action.type) {    742
700       case APP_DEACTIVATE_ACCOUNT
701         :
702           return {
703             ...state,
704           };
705         default:
706           return state;
707         }
708     } // Rekit uses a new approach to
709       organizing actions and
710       reducers. That is
711     // putting related actions and
712       reducers in one file. See
713       more at:
714     // https://medium.com/
715       @nate_wang/a-new-approach-
716       for-managing-redux-actions
717       -91c26ce8b5da
718     import { APP_DELETE_RECORD,
719       APP_GET_ERRORS } from './
720       constants';
721     import * as actions from './
722       actions';
723     import axios from 'axios';
724     import { message } from 'antd';
725
726     export const deleteRecord = (
727       data, position) => dispatch
728         => {
729           dispatch(actions.setLoading(
730             true));
731           if (position == 'Teacher') {
732             axios
733               .post('api/teacher/
734                 deleterecord', data)
735               .then(res => {
736                 message.success(res.
737                   data.msg);
738                 dispatch(actions.
739                   setLoading(false));
740                 dispatch({ type:
741                   APP_GET_ERRORS,
742                   payload: {} });
743               })
744               .catch(err => {
745                 dispatch({ type:
746                   APP_GET_ERRORS,
747                   payload: err.
748                     response.data });
749                 dispatch(actions.
750                   setLoading(false));
751                 message.error(err.
752                   response.data.msg);
753               });
754             } else if (position == 'Registrar') {
755               axios
756                 .post('api/registrar/
757                   deleterecord', data)
758                 .then(res => {
759                   message.success(res.
760                     data.msg);
761                   dispatch(actions.
762                     setLoading(false));
763                   dispatch({ type:
764                     APP_GET_ERRORS,
765                     payload: {} });
766                 })
767                 .catch(err => {
768                   dispatch({ type:
769                     APP_GET_ERRORS,
770                     payload: err.
771                       response.data });
772                   dispatch(actions.
773                     setLoading(false));
774                   message.error(err.
775                     response.data.msg);
776                 });
777             }
778           } else if (position == 'Section') {
779             axios
780               .post('api/section/
781                 deleterecord', data)
782               .then(res => {
783                 message.success(res.
784                   data.msg);
785               })
786               .catch(err => {
787                 dispatch({ type:
788                   APP_GET_ERRORS,
789                   payload: err.
790                     response.data });
791                 message.error(err.
792                   response.data.msg);
793               });
794             }
795           }
796         }
797       }
798     );
799   }
800
801   dispatch({ type:
802     APP_GET_ERRORS,
803     payload: err.
804       response.data });
805   dispatch(actions.
806     setLoading(false));
807   message.error(err.
808     response.data.msg);
809   response.data.msg);
810 });
811
812 export function reducer(state,
813   action) {
814   switch (action.type) {
815     case APP_DELETE_RECORD:
816       return {
817         ...state,
818       };
819     default:
820       return state;
821     }
822   }
823 // Rekit uses a new approach to
824   organizing actions and
825   reducers. That is
826 // putting related actions and
827   reducers in one file. See
828   more at:
829 // https://medium.com/
830   @nate_wang/a-new-approach-
831   for-managing-redux-actions
832   -91c26ce8b5da
833
834 import { APP_DELETE_SECTION,
835   APP_GET_ERRORS } from './
836   constants';
837 import * as actions from './
838   actions';
839 import axios from 'axios';
840 import { message } from 'antd';
841
842 export const deleteSection =
843   data => dispatch => {
844   dispatch(actions.setLoading(
845     true));
846   axios
847     .post('api/registrar/
848       deletesection', data)
849     .then(res => {
850       message.success('You have
851         successfully deleted
852         the section!');
853       dispatch(actions.
854         setLoading(false));
855       dispatch({ type:
856         APP_GET_ERRORS,
857         payload: {} });
858     })
859     .catch(err => {
860       dispatch({ type:
861         APP_GET_ERRORS,
862         payload: err.
863           response.data });
864       message.error(err.
865         response.data.msg);
866       dispatch(actions.
867         setLoading(false));
868     });
869   return {
870     type: APP_DELETE_SECTION,
871   };
872 }

```

```

786     };                                     827
787     export function reducer(state,           828
    action) {                                829
    switch (action.type) {                   830
        case APP_DELETE_SECTION:            831
            return {                      832
                ...state,                 833
            };                           834
        default:                         835
            return state;               836
    }                                         837
    // Rekit uses a new approach to       838
    // organizing actions and          839
    // reducers. That is              840
    // putting related actions and      841
    // reducers in one file. See       842
    // more at:                      843
    // https://medium.com/             844
    // @nate_wang/a-new-approach-      845
    // for-managing-redux-actions     846
    -91c26ce8b5da                     847
801
802     import {                          848
    APP_DELETE_STUDENT_SECTION,         849
    APP_GET_ERRORS } from './          850
    constants';                        851
803     import * as actions from './       852
    actions';                          853
804     import axios from 'axios';        854
805     import { message } from 'antd';    855
806
807     export const                      856
        deleteStudentSection = data856
        => dispatch => {               857
        dispatch(actions.setLoading(857
            true));                  858
        axios                           859
            .post('api/registrar/     860
                deletestudentsection', 861
            data)                    862
            .then(res => {           863
                dispatch({ type:     864
                    APP_GET_ERRORS,     865
                    payload: {} });   866
                dispatch(actions.     867
                    setLoading(false)); 868
                dispatch({ type:     867
                    APP_DELETE_STUDENT_SECTI... 868
                });
                message.success('Student 869
                    unenrolled           870
                    successfully!');    871
            })
            .catch(err => {           872
                dispatch({ type:     873
                    APP_DELETE_STUDENT_SECTI... 874
                });
                dispatch(actions.     875
                    setLoading(false)); 876
                dispatch({ type:     877
                    APP_GET_ERRORS,     878
                    payload: err.response 879
                });
                message.error(err.     880
                    response.data.msg); 881
            });
        );
    };
825     export function reducer(state,           881
    action) {                                882
    switch (action.type) {                   883
        case APP_DELETE_STUDENT_SECTION: 884
            :                           885
            return {                      886
                ...state,                 887
            };                           888
        default:                         889
            return state;               890
    }
    // Rekit uses a new approach to       891
    // organizing actions and          892
    // reducers. That is              893
    // putting related actions and      894
    // reducers in one file. See       895
    // more at:                      896
    // https://medium.com/             897
    // @nate_wang/a-new-approach-      898
    // for-managing-redux-actions     899
    -91c26ce8b5da                     900
901
902     import {                          903
    APP_DELETE_SUBCOMPONENT,          904
    APP_GET_ERRORS } from './          905
    constants';
903     import * as actions from './       906
    actions';
904     import axios from 'axios';
905     import { message } from 'antd';
906
907     export const deleteSubcomponent = (data, position) =>
908         dispatch => {
909         dispatch(actions.setLoading(true));
910         if (position == 'Teacher') {
911             axios
912                 .post('api/teacher/
913                     deletesubcomp', data)
914                 .then(res => {
915                     dispatch({ type:
916                         APP_GET_ERRORS,
917                         payload: {} });
918                     dispatch(actions.
919                         setLoading(false));
920                     dispatch({ type:
921                         APP_GET_ERRORS,
922                         payload: {} });
923                 })
924                 .catch(err => {
925                     dispatch({ type:
926                         APP_GET_ERRORS,
927                         payload: err.
928                         response.data });
929                     dispatch(actions.
930                         setLoading(false));
931                     message.error(err.
932                         response.data.msg);
933                 });
934             } else {
935                 axios
936                     .post('api/registrar/
937                         deletesubcomp', data)
938                     .then(res => {
939                         message.success(res.
940                             data.msg);
941                         dispatch(actions.
942                             setLoading(false));
943                         dispatch({ type:
944                             APP_GET_ERRORS,
945                             payload: {} });
946                     })
947                     .catch(err => {

```

```

869         dispatch({ type: 909
870             APP_GET_ERRORS,
871             payload: err.
872             response.data });
873         dispatch(actions.
874             setLoading(false));
875         message.error(err.
876             response.data.msg);
877     });
878 };
879 export function reducer(state, 917
880     action) {
881     switch (action.type) {
882         case
883             APP_DELETE_SUBCOMPONENT:
884             return {
885                 ...state,
886             };
887         default:
888             return state;
889     }
890 // Rekit uses a new approach to
891 // organizing actions and
892 // reducers. That is
893 // putting related actions and
894 // reducers in one file. See
895 // more at:
896 // https://medium.com/
897 // @nate_wang/a-new-approach-
898 // for-managing-redux-actions
899 // -91c26ce8b5da
900 import {
901     APP_DELETE SUBJECT SECTION,
902     APP_GET_ERRORS } from './
903     constants';
904 import * as actions from './
905     actions';
906 import axios from 'axios';
907 import { message } from 'antd';
908 export const
909     deleteSubjectSection = data
910     => dispatch => {
911         dispatch(actions.setLoading(
912             true));
913         dispatch({ type:
914             APP_DELETE SUBJECT SECTION});
915         axios
916             .post('api/registrar/
917                 deletesubjectsection',
918                 data)
919             .then(res => {
920                 dispatch(actions.
921                     setLoading(false));
922                 message.success('Subject
923                     load deleted
924                     successfully!');
925             })
926             .catch(err => {
927                 dispatch(actions.
928                     setLoading(false));
929                 dispatch({ type:
930                     APP_GET_ERRORS,
931                     payload: {} });
932             })
933             .catch(err => {
934                 dispatch(actions.
935                     setLoading(false));
936                 dispatch({ type:
937                     APP_GET_ERRORS,
938                     payload: err.response.
939                     data });
940             })
941             .then(res => {
942                 dispatch(actions.
943                     setLoading(false));
944                 message.success('Student
945                     dropped successfully!');
946             })
947             .catch(err => {
948                 dispatch(actions.
949                     setLoading(false));
950                 dispatch({ type:
951                     APP_GET_ERRORS,
952                     payload: err.response.
953                     data });
954             })
955         message.error(err.
956             response.data.msg);
957     });
958 };
959 export function reducer(state,

```

```

951     action) { 996
952       switch (action.type) { 997
953         case 998
954           APP_DELETE_SUBJECT_SECTION
955             : 999
956             ...state,
957             }; 1000
958             default:
959               return state; 1001
960           } 1001
961           // Rekit uses a new approach to
962             organizing actions and 1002
963             reducers. That is 1002
964             // putting related actions and 1003
965               reducers in one file. See 1003
966               more at: 1004
967               // https://medium.com/ 1005
968                 @nate_wang/a-new-approach 1006
969                 for-managing-redux-actions 1006
970                 -91c26ce8b5da 1007
971
972   import { APP_EDIT_RECORD, 1008
973     APP_GET_ERRORS } from './ 1009
974     constants'; 1009
975   import * as actions from './ 1010
976     actions'; 1010
977   import axios from 'axios'; 1011
978   import { message } from 'antd'; 1011
979   export const editRecord = (data 1012
980     , position) => dispatch => 1013
981     dispatch(actions.setLoading( 1013
982       true)); 1014
983     axios 1014
984       .post(`api/${position. 1015
985        toLowerCase()}/ 1015
986         editrecord`, data) 1016
987       .then(res => { 1016
988         message.success(res.data. 1017
989           msg); 1017
990         dispatch(actions. 1017
991           setLoading(false)); 1018
992         dispatch({ type: 1018
993           APP_GET_ERRORS, 1019
994             payload: {} }); 1020
995         dispatch({ type: 1021
996           APP_EDIT_RECORD}); 1022
997       }) 1023
998       .catch(err => { 1024
999         dispatch({ type: 1025
1000           APP_GET_ERRORS, 1026
1001             payload: err.response 1027
1002               data}); 1027
1003         dispatch(actions. 1028
1004           setLoading(false)); 1029
1005         message.error(err. 1030
1006           response.data.msg); 1031
1007       });
1008   };
1009   export function reducer(state, 1033
1010     action) {
1011     switch (action.type) { 1034
1012       case APP_EDIT_RECORD: 1034
1013         return {
1014           ...state,
1015           }; 1035
1016         default:
1017           return state; 1036
1018     } 1037
1019     }
1020     }
1021     // Rekit uses a new approach to
1022       organizing actions and 1022
1023       reducers. That is 1022
1024       // putting related actions and 1023
1025         reducers in one file. See 1023
1026         more at: 1024
1027         // https://medium.com/ 1025
1028           @nate_wang/a-new-approach 1026
1029             for-managing-redux-actions 1026
1030             -91c26ce8b5da 1027
1031
1032   import { APP_EDIT_SUBCOMPONENT, 1027
1033     APP_GET_ERRORS } from './ 1028
1034     constants'; 1029
1035   import * as actions from './ 1030

```

```
1038     actions';
1039     import axios from 'axios';    1084
1040     import { message } from 'antd';
1041
1042     export const editSubcomponent1085
1043         (data, position) =>
1044             dispatch => {
1045                 dispatch(actions.setLoading(
1046                     true));          1086
1047                 if (position == 'Teacher') 1087
1048                     axios
1049                         .post('api/teacher/
1050                             editsubcomp', data) 1088
1051                         .then(res => {
1052                             message.success(res. 1089
1053                                 data.msg);           1090
1054                             dispatch(actions. 1091
1055                                 setLoading(false));
1056                             dispatch({ type: 1092
1057                                 APP_GET_ERRORS,
1058                                 payload: {} }); 1093
1059                         })
1060                         .catch(err => {
1061                             dispatch({ type: 1095
1062                                 APP_GET_ERRORS,
1063                                 payload: err. 1096
1064                                     response.data}); 1097
1065                         })
1066                         .then(res => {
1067                             message.success(res. 1104
1068                                 data.msg);           1105
1069                             dispatch(actions. 1106
1070                                 setLoading(false));
1071                             dispatch({ type: 1107
1072                                 APP_GET_ERRORS,
1073                                 payload: {} }); 1108
1074                         })
1075                         .catch(err => {
1076                             dispatch({ type: 1109
1077                                 APP_GET_ERRORS,
1078                                 payload: err. 1110
1079                                     response.data}); 1111
1080                             dispatch(actions. 1112
1081                                 setLoading(false));
1082                             message.error(err. 1113
1083                                 response.data.msg); 1114
1084                         })
1085                     });
1086
1087     export function reducer(state, 1120
1088         action) {
1089         switch (action.type) {
1090             case APP_EDIT_SUBCOMPONENT121
1091                 return {
1092                     ...state,
1093                 };
1094                 default: 1122
1095                     return state;
1096                 }
1097             }
1098
1099 // Rekit uses a new approach 1124
1100 // organizing actions and 1125
```



```

1215     .catch(err => {           1260           ...state,
1216       dispatch({ type:      1261         profile: action.payload
1217         APP_GET_ERRORS,
1218         payload: err.response
1219         data });
1220       });
1221     );
1222   );
1223   export function reducer(state, 1267
1224     action) {
1225     switch (action.type) {
1226       case APP_GET_ACCOUNT_LIST:
1227         return {
1228           ...state,
1229           accounts: action.
1230             payload,
1231           };
1232         default:
1233           return state;
1234         }
1235       // Rekit uses a new approach 1273
1236       organizing actions and 1274
1237       reducers. That is
1238       // putting related actions and 1275
1239       reducers in one file. See
1240       more at:
1241       // https://medium.com/ 1276
1242       @nate_wang/a-new-approach 1277
1243       for-managing-redux-actions 1278
1244       -91c26ce8b5da
1245     import {
1246       APP_GET_CURRENT_PROFILE, 1280
1247       APP_GET_ERRORS,          1281
1248       APP_SET_LOADING } from '. 1282
1249       A283
1250       constants';
1251     import axios from 'axios'; 1284
1252     export const getCurrentProfile = () => dispatch => { 1285
1253       dispatch({ type: 1286
1254         APP_SET_LOADING, payload: 1287
1255         true });
1256       axios
1257         .get('api/users/profile') 1288
1258         .then(res => {
1259           dispatch({ type: 1289
1260             APP_GET_CURRENT_PROFILE, 1290
1261             payload: res.data 1291
1262             });
1263           dispatch({ type: 1293
1264             APP_SET_LOADING, 1294
1265             payload: false });
1266         });
1267         .catch(err => {
1268           dispatch({ 1302
1269             type: APP_GET_ERRORS,
1270             payload: err.response.
1271               data,
1272             });
1273           dispatch({ type: 1303
1274             APP_SET_LOADING,
1275             payload: false });
1276         });
1277       );
1278     export function reducer(state, 1306
1279     action) {
1280       switch (action.type) {
1281         case
1282           APP_GET_CURRENT_PROFILE:
1283             return { 1307
1284               ...state,
1285               profile: action.payload
1286             };
1287           default:
1288             return state;
1289         }
1290       // Rekit uses a new approach to
1291       organizing actions and
1292       reducers. That is
1293       // putting related actions and
1294       reducers in one file. See
1295       more at:
1296       // https://medium.com/
1297       @nate_wang/a-new-approach-
1298       for-managing-redux-actions
1299       -91c26ce8b5da
1300     import { APP_GET_ERRORS } from
1301       './constants';
1302     export const getErrors = data
1303       => dispatch => {
1304       dispatch({ type:
1305         APP_GET_ERRORS, payload:
1306           data });
1307     };
1308     export function reducer(state,
1309     action) {
1310       switch (action.type) {
1311         case APP_GET_ERRORS:
1312           return {
1313             ...state,
1314             errors: action.payload,
1315           };
1316         default:
1317           return state;
1318         }
1319       const initialState = {
1320         showLoading: false,
1321         auth: {
1322           isAuthenticated: false,
1323           user: {},
1324         },
1325         profile: {},
1326         errors: {}
1327       };
1328       export default initialState;
1329       // Rekit uses a new approach
1330       to organizing actions and
1331       reducers. That is
1332       // putting related actions
1333       and reducers in one file.
1334       See more at:
1335       // https://medium.com/
1336       @nate_wang/a-new-approach-
1337       for-managing-redux-actions
1338       -91c26ce8b5da
1339       import { APP_LOGIN_USER,
1340         APP_GET_ERRORS } from './
1341         constants';
1342       import * as actions from './
1343         actions';
1344       import setAuthToken from '.
1345         ../../common/
1346         setAuthToken';
1347       import axios from 'axios';
1348       import jwt_decode from 'jwt-
1349         decode';

```

```

1310     decode';           1354
1311     import { message } from 'antd'    1355
1312     ';
1313     export const loginUser = 1356
1314     userData => dispatch => {
1315     dispatch(actions.setLoading
1316     (true));           1357
1317     axios             1358
1318     .post('api/users/login',
1319     userData)        1359
1320     .then(res => {      1360
1321     dispatch(actions.
1322     setLoading(false)) 1361
1323     const { token } = res 1362
1324     const decoded =       1363
1325     jwt_decode(token) 1364
1326     dispatch(actions.
1327     setCurrentUser( 1369
1328     decoded));         1370
1329     dispatch(actions.
1330     getCurrentProfile 1371
1331     ;                 1372
1332     })                1373
1333     .catch(err => {      1374
1334     dispatch(actions.
1335     setLoading(false)) 1375
1336     dispatch(actions.
1337     getErrors(err. 1376
1338     response.data)); 1377
1339     if (err.response.data 1378
1340     isActive) {       1379
1341     message.error(err. 1380
1342     response.data. 1381
1343     isActive);       1382
1344     } else {           1383
1345     message.error(err. 1384
1346     response.data.msg)
1347     ;                 1385
1348     }                  1386
1349     });
1350     return {
1351     type: APP_LOGIN_USER,
1352     };
1353   };

1354   import { APP_LOGOUT_USER }
1355   from './constants';
1356   import * as actions from './
1357   actions';
1358   import setAuthToken from '
1359   ../../common/
1360   setAuthToken';

1361   export function logoutUser()
1362   {
1363     actions.setLoading(true);
1364     localStorage.removeItem(
1365     'jwtToken');
1366     setAuthToken(false);
1367     actions.setLoading(false);
1368     return {
1369     type: APP_LOGOUT_USER,
1370     };
1371   }

1372   export function reducer(state
1373   , action) {
1374     switch (action.type) {
1375     case APP_LOGOUT_USER:
1376       return {
1377       ...state,
1378       auth: {
1379       isAuthenticated:
1380       false,
1381       user: {},
1382       profile: {}
1383     };
1384     default:
1385       return state;
1386     }
1387   }
1388 }

1389   export function reducer(state
1390   , action) {
1391     switch (action.type) {
1392     case APP_LOGIN_USER:
1393       return {
1394       ...state,
1395       };
1396     default:
1397       return state;
1398     }
1399   }

1400   // Rekit uses a new approach
1401   // to organizing actions and
1402   // reducers. That is
1403   // putting related actions
1404   // and reducers in one file.
1405   // See more at:
1406   // https://medium.com/
1407   // @nate_wang/a-new-approach-
1408   // for-managing-redux-actions
1409   // -91c26ce8b5da
1410   import {
1411     APP_POST_CLASS_RECORD,
1412     APP_GET_ERRORS } from './
1413     constants';
1414   import * as actions from './
1415     actions';
1416   import { message } from 'antd'
1417   ;
1418   import axios from 'axios';

1419   export const postClassRecord
1420   = (data,
1421   resetClassRecordInfo) =>
1422   dispatch => {
1423     dispatch({ type:
1424     APP_POST_CLASS_RECORD })
1425     ;
1426     dispatch(actions.setLoading
1427     (true));
1428     axios
1429     .post('api/registrar/
1430     postclassrecord', data
1431     )
1432     .then(res => {
1433     message.success(res.
1434     );
1435   }

```

```

1400         data.msg);
1401     resetClassRecordInfo();
1402     dispatch(actions.setLoading(false));
1403   })
1404   .catch(err => {
1405     message.error(err.response.data.msg);
1406     dispatch(actions.setLoading(false));
1407   dispatch({ type: APP_GET_ERRORS, payload: err.response.data });
1408 });
1409
1410 export function reducer(state,
1411   action) {
1412   switch (action.type) {
1413     case APP_POST_CLASS_RECORD:
1414       return {
1415         ...state,
1416       };
1417     default:
1418       return state;
1419     }
1420   }
1421   import initialState from './initialState';
1422   import { reducer as setCurrentUserReducer } from './setCurrentUser';
1423   import { reducer as sampleActionReducer } from './sampleAction';
1424   import { reducer as logoutUserReducer } from './logoutUser';
1425   import { reducer as setLoadingReducer } from './setLoading';
1426   import { reducer as loginUserReducer } from './loginUser';
1427   import { reducer as getErrorsReducer } from './getErrors';
1428   import { reducer as getCurrentProfileReducer } from './getCurrentProfile';
1429   import { reducer as updateAdminProfileReducer } from './updateAdminProfile';
1430   import { reducer as changePasswordReducer } from './changePassword';
1431   import { reducer as createAccountReducer } from './createAccount';
1432   import { reducer as getAccountListReducer } from './getAccountList';
1433   import { reducer as activateAccountReducer } from './activateAccount';
1434   import { reducer as deactivateAccountReducer } from './deactivateAccount';
1435   import { reducer as updateDirectorProfileReducer } from './updateDirectorProfile';
1436   import { reducer as updateTeacherProfileReducer } from './updateTeacherProfile';
1437   import { reducer as updateRegistrarProfileReducer } from './updateRegistrarProfile';
1438   import { reducer as updateParentProfileReducer } from './updateParentProfile';
1439   import { reducer as addSectionReducer } from './addSection';
1440   import { reducer as editSectionReducer } from './editSection';
1441   import { reducer as deleteSectionReducer } from './deleteSection';
1442   import { reducer as createStudentSectionReducer } from './createStudentSection';
1443   import { reducer as deleteStudentSectionReducer } from './deleteStudentSection';
1444   import { reducer as assignAdvisorySectionReducer } from './assignAdvisorySection';
1445   import { reducer as unassignAdvisorySectionReducer } from './unassignAdvisorySection';
1446   import { reducer as createSubjectSectionReducer } from './createSubjectSection';
1447   import { reducer as deleteSubjectSectionReducer } from './deleteSubjectSection';
1448   import { reducer as addSubjectSectionStudentReducer } from './addSubjectSectionStudent';
1449   import { reducer as deleteSubjectSectionStudentReducer } from './deleteSubjectSectionStudent';
1450   import { reducer as addNewSubcomponentReducer } from './addNewSubcomponent';
1451   import { reducer as editSubcomponentReducer } from './editSubcomponent';
1452   import { reducer as deleteSubcomponentReducer } from './deleteSubcomponent';
1453   import { reducer as addNewRecordReducer } from './addNewRecord';

```

```

1454 import { reducer as
1455     deleteRecordReducer } from
1456         './deleteRecord'; 1476
1457 import { reducer as 1477
1458     changeTransmutationReducer 1478
1459     } from './
1460         changeTransmutation'; 1480
1461 import { reducer as 1481
1462     editRecordReducer } from
1463         './editRecord'; 1483
1464 import { reducer as 1484
1465     setDeadlineAllReducer 1485
1466     } from './setDeadlineAll' 1486
1467 import { reducer as 1488
1468     setDeadlineReducer } from
1469         './setDeadline'; 1490
1470 import { reducer as 1491
1471     removeDeadlineReducer } 1492
1472     from './removeDeadline' 1492
1473 import { reducer as 1493
1474     createSchoolYearReducer 1494
1475     } from './createSchoolYear' 1494
1476 import { reducer as 1495
1477     changeQuarterSyReducer 1496
1478     } from './changeQuarterSy' 1497
1479 import { reducer as 1498
1480     endSchoolYearReducer } 1498
1481     from './endSchoolYear'; 1499
1482 import { reducer as 1500
1483     addSubjectSectionStudentBulk 1500
1484     } from './
1485     addSubjectSectionStudentBulk' 1500
1486     ;
1487 import { reducer as 1502
1488     submitClassRecordReducer 1503
1489     } from './submitClassRecord
1490     ';
1491 import { reducer as 1504
1492     postClassRecordReducer 1505
1493     } from './postClassRecord 1506
1494
1495 import { reducer as 1507
1496     revertClassRecordReducer 1508
1497     } from './revertClassRecord
1498     ';
1499 import { reducer as 1511
1500     generatePdfStudentReducer 1512
1501     } from './
1502     generatePdfStudent'; 1514
1503 import { reducer as 1515
1504     generatePdfSectionReducer 1516
1505     } from './
1506     generatePdfSection'; 1518
1507 import { reducer as 1519
1508     restrictAccountReducer 1520
1509     } from './restrictAccount 1521
1510
1511 import { reducer as 1522
1512     unrestrictAccountReducer 1523
1513     } from './unrestrictAccou
1514     nt';
1515 import { reducer as 1526
1516     updateCashierProfileReduc
1517     } from './
1518     updateCashierProfile'; 1528
1519 import { reducer as 1529
1520     updateStudentProfileReduc
1521     } from './
1522     updateStudentProfile'; 1532
1523 import { reducer as 1533
1524     assignParentReducer } from
1525         './assignParent'; 1533
1526 import { reducer as 1534
1527     unassignParentReducer } 1535
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2438
2439
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2489
2490
2491
2492
2493
2494
2495
2496
2497
2497
2498
2499
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2538
2539
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2548
2549
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2697
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2797
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3388
3389
3389
```

```

1536         actions here
1537     default:
1538         newState = state;    1581
1539         break;
1540     }
1541     /* istanbul ignore next */
1542     return reducers.reduce((s1, 1582
1543         r) => r(s, action), 1583
1544         newState);
1545     }
1546     // Rekit uses a new approach 1584
1547     to organizing actions and 1585
1548     reducers. That is 1586
1549     // putting related actions 1587
1550     and reducers in one file 1588
1551     See more at: 1589
1552     // https://medium.com/ 1588
1553     @nate_wang/a-new-approach- 1589
1554     for-managing-redux-actions 1590
1555     -91c26ce8b5da
1556     import { APP_REMOVE_DEADLINE, 1590
1557         APP_GET_ERRORS } from 1591
1558         'constants';
1559     import * as actions from './ 1592
1560         actions';
1561     import { message } from 'antd' 1593
1562         ';
1563     import axios from 'axios'; 1594
1564
1565     export const removeDeadline = 1595
1566         data => dispatch => { 1596
1567             dispatch({ type: 1597
1568                 APP_REMOVE_DEADLINE }, 1598
1569             dispatch(actions.setLoading 1599
1570                 (true));
1571             axios
1572                 .post('api/registrar/ 1600
1573                     removedeadline', data)
1574                 .then(res => {
1575                     message.success(res. 1601
1576                         data.msg);
1577                     dispatch(actions. 1602
1578                         setLoading(false)} 1603
1579                 );
1580                 .catch(err => {
1581                     message.error(err. 1605
1582                         response.data.msg);
1583                     dispatch(actions. 1606
1584                         setLoading(false)} 1608
1585                     dispatch({ type: 1609
1586                         APP_GET_ERRORS, 1610
1587                         payload: err. 1611
1588                         response.data }); 1612
1589                 );
1590             });
1591         );
1592     }
1593     export function reducer(state 1615
1594         , action) {
1595         switch (action.type) { 1616
1596             case APP_REMOVE_DEADLINE: 1617
1597                 return {
1598                     ...state,
1599                 };
1600             default:
1601                 return state; 1618
1602             }
1603         }
1604         // Rekit uses a new approach 1620
1605         to organizing actions and 1621
1606         reducers. That is 1622
1607         // putting related actions 1623
1608         and reducers in one file. 1624
1609         See more at:
1610         // https://medium.com/ 1625
1611         @nate_wang/a-new-approach- 1626
1612         for-managing-redux-actions 1627
1613         -91c26ce8b5da
1614         import {
1615             APP_REVERT_CLASS_RECORD, 1628
1616             APP_GET_ERRORS } from './ 1629
1617             constants';
1618         import * as actions from './ 1630
1619             actions';

```

```

1621     import { message } from 'antd'          1667
1622     ;                                     1668
1623     import axios from 'axios';           1669
1624     export const                         1670
1625       revertClassRecord = (data) => {      1671
1626         dispatch({ type: APP_REVERT_CLASS_RECORD }) 1672
1627         dispatch(actions.setLoading(true));    1673
1628         axios.post('api/registrar/revertclassrecord', data) 1674
1629         .then(res => {                      1675
1630           message.success(res.data.msg);      1676
1631           resetClassRecordInfo();           1677
1632           dispatch(actions.setLoading(false)) 1678
1633         })
1634         .catch(err => {                    1679
1635           message.error(err.response.data.msg); 1680
1636           dispatch(actions.setLoading(false)); 1681
1637           dispatch({ type: APP_GET_ERRORS, payload: err.response.data }); 1682
1638         });
1639       };
1640     );
1641     export function reducer(state, action) { 1691
1642       switch (action.type) {                  1692
1643         case APP_REVERT_CLASS_RECORD:        1693
1644           :                                1694
1645           return { ...state,                1695
1646             };                           1696
1647           default:                      1697
1648             return state;                 1698
1649           }
1650         }
1651       );
1652     // Rekit uses a new approach to organizing actions and reducers. That is
1653     // putting related actions and reducers in one file. See more at:
1654     // https://medium.com/@nate_wang/a-new-approach-for-managing-redux-actions-91c26ce8b5da
1655     import { APP_SAMPLE_ACTION,           1709
1656   } from './constants';                  1710
1657   export function sampleAction() {        1711
1658     return { type: APP_SAMPLE_ACTION }; 1712
1659   }
1660   export function reducer(state, action) { 1713
1661     , action) {
1662       switch (action.type) {
1663         case APP_SAMPLE_ACTION:
1664           return { ...state,
1665             showLoading: true
1666           };
1667         default:
1668           return state;
1669         }
1670       }
1671     // Rekit uses a new approach to organizing actions and reducers. That is
1672     // putting related actions and reducers in one file. See more at:
1673     // https://medium.com/@nate_wang/a-new-approach-for-managing-redux-actions-91c26ce8b5da
1674     import { APP_SET_CURRENT_USER,        1714
1675   } from './constants';
1676   import * as actions from './actions';
1677   export function setCurrentUser(payload) {
1678     actions.getCurrentProfile();
1679     return {
1680       type: APP_SET_CURRENT_USER,
1681       payload,
1682     };
1683   }
1684   export function reducer(state, action) {
1685     switch (action.type) {
1686       case APP_SET_CURRENT_USER:
1687         :
1688         return { ...state,
1689           auth: {
1690             isAuthenticated: true,
1691             user: action.payload,
1692           },
1693         };
1694       default:
1695         return state;
1696       }
1697     }
1698   // Rekit uses a new approach to organizing actions and reducers. That is
1699   // putting related actions and reducers in one file. See more at:
1700   // https://medium.com/@nate_wang/a-new-approach-for-managing-redux-actions-91c26ce8b5da
1701   import { APP_SET_DEADLINE,           1709
1702   APP_GET_ERRORS } from './constants';
1703   import * as actions from './actions';
1704   import { message } from 'antd'

```

```

        ';
1715    import axios from 'axios'; 1760
1716
1717    export const setDeadline = 1761
1718      data => dispatch => { 1762
1719        dispatch({ type: 1763
1720          APP_SET_DEADLINE });
1721        dispatch(actions.setLoading 1764
1722          (true));
1723        axios 1765
1724          .post('api/registrar/ 1766
1725            setdeadline', data) 1767
1726          .then(res => { 1768
1727            message.success(res. 1766
1728              data.msg); 1767
1729            dispatch(actions. 1768
1730              setLoading(false)) 1769
1731          });
1732          1770
1733        .catch(err => { 1771
1734          message.error(err. 1771
1735            response.data.msg);
1736          dispatch(actions. 1772
1737            setLoading(false)); 1773
1738          dispatch({ type: 1774
1739            APP_GET_ERRORS, 1775
1740            payload: err. 1776
1741            response.data }); 1777
1742        });
1743        1778
1744      });
1745      1779
1746      1780
1747      export function reducer(state
1748        , action) {
1749        switch (action.type) { 1781
1750          case APP_SET_DEADLINE:
1751            return {
1752              ...state,
1753            };
1754            default:
1755              return state;
1756            }
1757          // Rekit uses a new approach
1758          to organizing actions and
1759          reducers. That is
1760          // putting related actions
1761          and reducers in one file.
1762          See more at:
1763          // https://medium.com/
1764          @nate_wang/a-new-approach-
1765          for-managing-redux-actions
1766          -91c26ce8b5da
1767
1768      import { APP_SET_DEADLINE_ALL
1769        , APP_GET_ERRORS } from
1770        './constants';
1771      import * as actions from './actions';
1772      import { message } from 'antd';
1773      import axios from 'axios';
1774
1775      export const setDeadlineAll =
1776        data => dispatch => { 1779
1777          dispatch({ type: 1800
1778            APP_SET_DEADLINE_ALL });
1779          dispatch(actions.setLoading 1801
1780            (true));
1781          axios 1803
1782            .post('api/registrar/ 1804
1783              setdeadlineall', data);
1784          .then(res => {
1785            message.success(res. 1805
1786              data.msg);
1787            dispatch(actions.setLoading 1806
1788              (false));
1789          });
1790        });
1791
1792        data.msg);
1793        dispatch(actions.setLoading(false));
1794      });
1795      .catch(err => {
1796        message.error(err.
1797          response.data.msg);
1798        dispatch(actions.setLoading(false));
1799        dispatch({ type:
1800          APP_GET_ERRORS ,
1801          payload: err.
1802          response.data });
1803      });
1804
1805      export function reducer(state
1806        , action) {
1807        switch (action.type) {
1808          case APP_SET_DEADLINE_ALL:
1809            :
1810            return {
1811              ...state,
1812            };
1813          default:
1814            return state;
1815          }
1816        }
1817
1818        // Rekit uses a new approach
1819        to organizing actions and
1820        reducers. That is
1821        // putting related actions
1822        and reducers in one file.
1823        See more at:
1824        // https://medium.com/
1825        @nate_wang/a-new-approach-
1826        for-managing-redux-actions
1827        -91c26ce8b5da
1828
1829        import { APP_SET_LOADING }
1830          from './constants';
1831
1832        export function setLoading(
1833          input) {
1834          return {
1835            type: APP_SET_LOADING ,
1836            payload: input,
1837          };
1838        }
1839
1840        export const setLoadingTrue =
1841          () => dispatch => {
1842          dispatch({ type:
1843            APP_SET_LOADING , payload
1844              : true });
1845        };
1846
1847        export const setLoadingFalse
1848          = () => dispatch => {
1849          dispatch({ type:
1850            APP_SET_LOADING , payload
1851              : false });
1852        };
1853
1854        export function reducer(state
1855        , action) {
1856        switch (action.type) {
1857          case APP_SET_LOADING:
1858            return {
1859              ...state,
1860              showLoading: action.
1861                payload,
1862              };
1863        };

```

```

1808
1809         default:
1810             return state;           1851
1811     }
1812 }
1813 // Rekit uses a new approach
1814 // to organizing actions and
1815 // reducers. That is
1816 // putting related actions
1817 // and reducers in one file.
1818 // See more at:
1819 // https://medium.com/
1820 // @nate_wang/a-new-approach-
1821 // for-managing-redux-actions
1822 // -91c26ce8b5da
1823
1824 import {
1825     APP_SUBMIT_CLASS_RECORD,
1826     APP_GET_ERRORS } from './
1827     constants';
1828 import * as actions from './
1829     actions';
1830 import { message } from 'antd';
1831 import axios from 'axios';
1832
1833 export const
1834     submitClassRecord = data
1835     => dispatch => {
1836         dispatch({ type:
1837             APP_SUBMIT_CLASS_RECORD
1838         });
1839         dispatch(actions.setLoading(
1840             true));
1841         axios
1842             .post('api/teacher/
1843                 submitclassrecord',
1844                 data)
1845             .then(res => {
1846                 message.success(res.
1847                     data.msg);
1848                 dispatch(actions.
1849                     setLoading(false));
1850             })
1851             .catch(err => {
1852                 message.error(err.
1853                     response.data.msg);
1854                 dispatch(actions.
1855                     setLoading(false));
1856                 dispatch({ type:
1857                     APP_GET_ERRORS,
1858                     payload: err.
1859                     response.data });
1860             });
1861     };
1862 }
1863
1864 export function reducer(state
1865     , action) {
1866     switch (action.type) {
1867         case
1868             APP_SUBMIT_CLASS_RECORD
1869             :
1870             return {
1871                 ...state,
1872             };
1873         default:
1874             return state;
1875     }
1876 }
1877
1878 // Rekit uses a new approach
1879 // to organizing actions and
1880 // reducers. That is
1881 // putting related actions
1882 // and reducers in one file.
1883 // See more at:
1884 // https://medium.com/
1885 // @nate_wang/a-new-approach-
1886 // putting related actions
1887
1888 // and reducers in one file.
1889 // See more at:
1890 // https://medium.com/
1891 // @nate_wang/a-new-approach-
1892 // for-managing-redux-actions
1893 // -91c26ce8b5da
1894
1895 import {
1896     APP_UNASSIGN_ADVISORY_SECTION
1897     , APP_GET_ERRORS } from './
1898     constants';
1899 import * as actions from './
1900     actions';
1901 import axios from 'axios';
1902 import { message } from 'antd';
1903
1904
1905 export const
1906     unassignAdvisorySection =
1907     data => dispatch => {
1908         dispatch(actions.setLoading(
1909             true));
1910         dispatch({ type:
1911             APP_UNASSIGN_ADVISORY_SECTION
1912         });
1913         axios
1914             .post('api/registrar/
1915                 unassignadviser', data
1916             )
1917             .then(res => {
1918                 dispatch(actions.
1919                     setLoading(false));
1920                 message.success('
1921                     Teacher unassigned
1922                     successfully!');
1923                 dispatch({ type:
1924                     APP_GET_ERRORS,
1925                     payload: {} });
1926             })
1927             .catch(err => {
1928                 dispatch(actions.
1929                     setLoading(false));
1930                 dispatch({ type:
1931                     APP_GET_ERRORS,
1932                     payload: err.
1933                     response.data });
1934                 message.error(err.
1935                     response.data.
1936                     teacherName);
1937             });
1938
1939         export function reducer(state
1940             , action) {
1941             switch (action.type) {
1942                 case
1943                     APP_UNASSIGN_ADVISORY_SECTION
1944                     :
1945                     return {
1946                         ...state,
1947                     };
1948                 default:
1949                     return state;
1950             }
1951         }
1952
1953 // Rekit uses a new approach
1954 // to organizing actions and
1955 // reducers. That is
1956 // putting related actions
1957 // and reducers in one file.
1958 // See more at:
1959 // https://medium.com/
1960 // @nate_wang/a-new-approach-

```



```

        !');
2015
1973   dispatch(actions.
           setLoading(false));
2016
1974   dispatch(actions.
           getCurrentProfile());
2017
1975   dispatch({ type:
               APP_GET_ERRORS,
               payload: {} });
2018
1976   .catch(err => {
2019     dispatch({
2020       type: APP_GET_ERRORS,
2021       payload: err.response
2022         .data,
2023       });
2024   dispatch(actions.
               setLoading(false));
2025
1983   });
2026
1984   return {
2027     type:
       APP_UPDATE_ADMIN_PROFILE
       ,
2028     );
2029   };
2030
1988   2031
1989   export function reducer(state
       , action) {
2032     switch (action.type) {
2033       case
         APP_UPDATE_ADMIN_PROFILE
         :
2034         return {
2035           ...state,
2036         };
2037       default:
2038         return state;
2040     }
2041   }
2042
2000   // Rekit uses a new approach
       to organizing actions and
       reducers. That is
2043
2001   // putting related actions
       and reducers in one file.
       See more at:
2044
2002   // https://medium.com/
       @nate_wang/a-new-approach-
       for-managing-redux-actions
       -91c26ce8b5da
2045
2046
2004   import {
       APP_UPDATE_CASHIER_PROFILE
       , APP_GET_ERRORS } from
       './constants';
2047
2005   import * as actions from './
       actions';
2048
2006   import axios from 'axios';
2049
2007   import { message } from 'antd
       ';
2050
2008
2009   export const
       updateCashierProfile =
       data => dispatch => {
2052     dispatch(actions.setLoading
       (true));
2053
2011   axios
       .post('api/cashier/
             updateprofile', data);
2054
2012   .then(res => {
2055     message.success('You
       have successfully
       updated your profile
       !'));
2056
2013
2014
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2798
2799
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2898
2899
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2909
2910
2911
2912
2913
2914
2915
2916
2917
2918
2919
2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2949
2950
2951
2952
2953
2954
2955
2956
2957
2958
2959
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2998
2999
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3098
3099
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3198
3199
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3298
3299
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3359
3360
3361
3362
3363
3364
3365
3366
3367
3368
3369
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3398
3399
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3429
3430
3431
3432
3433
3434
3435
3436
3437
3438
3439
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3449
3450
3451
3452
3453
3454
3455
3456
3457
3458
3459
3459
3460
3461
3462
3463
3464
3465
3466
3467
3468
3469
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478
3479
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3498
3499
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529
3529
3530
3531
3532
3533
3534
3535
3536
3537
3538
3539
3539
3540
3541
3542
3543
3544
3545
3546
3547
3548
3549
3549
3550
3551
3552
3553
3554
3555
3556
3557
3558
3559
3559
3560
3561
3562
3563
3564
3565
3566
3567
3568
3569
3569
3570
3571
3572
3573
3574
3575
3576
3577
3578
3579
3579
3580
3581
3582
3583
3584
3585
3586
3587
3588
3589
3589
3590
3591
3592
3593
3594
3595
3596
3597
3598
3598
3599
3599
3600
3601
3602
3603
3604
3605
3606
3607
3608
3609
3609
3610
3611
3612
3613
3614
3615
3616
3617
3618
3619
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3698
3699
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3739
3740
3741
3742
3743
3744
3745
3746
3747
3748
3749
3749
3750
3751
3752
3753
3754
3755
3756
3757
3758
3759
3759
3760
3761
3762
3763
3764
3765
3766
3767
3768
3769
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3779
3780
3781
3782
3783
3784
3785
3786
3787
3788
3789
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3798
3799
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858
3859
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3898
3899
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3998
3999
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4
```

```
2058     setLoading(false) 2100
2059     dispatch(actions.
2060         getCurrentProfile())
2061         ;
2062     dispatch({ type:
2063         APP_GET_ERRORS ,
2064         payload: {} });
2065     });
2066     .catch(err => {
2067         dispatch({
2068             type: APP_GET_ERRORS ,
2069             payload: err.response
2070             .data,
2071         });
2072         dispatch(actions.
2073             setLoading(false));
2074     });
2075     return {
2076         type:
2077             APP_UPDATE_DIRECTOR_PROFILE
2078             ,
2079         };
2080     );
2081     export function reducer(state, action) {
2082         switch (action.type) {
2083             case
2084                 APP_UPDATE_DIRECTOR_PROFILE
2085                 :
2086                     return {
2087                         ...state,
2088                     };
2089             default:
2090                 return state;
2091         }
2092         // Rekit uses a new approach
2093         to organizing actions and
2094         reducers. That is
2095         // putting related actions
2096         and reducers in one file.
2097         See more at:
2098         // https://medium.com/
2099         @nate_wang/a-new-approach-
2100         for-managing-redux-actions
2101         -91c26ce8b5da
2102
2103     import {
2104         APP_UPDATE_PARENT_PROFILE ,
2105         APP_GET_ERRORS } from
2106         constants';
2107     import * as actions from './actions';
2108     import axios from 'axios';
2109     import { message } from 'antd';
2110         ';
2111     );
2112     export const
2113         updateParentProfile = data
2114             => dispatch => {
2115             dispatch(actions.setLoading(true));
2116             axios
2117                 .post('api/parent/
2118                     updateprofile', data);
2119             .then(res => {
2120                 message.success('You
2121                     have successfully
2122                     updated your profile
2123                     !');
2124                 dispatch(actions.
2125                     setLoading(false));
2126             });
2127             .catch(err => {
2128                 dispatch({
2129                     type: APP_GET_ERRORS ,
2130                     payload: err.response
2131                     .data,
2132                 });
2133                 dispatch(actions.setLoading(true));
2134                 axios
2135                     .post('api/registrar/
2136                         updateprofile', data);
2137                     .then(res => {
2138                         message.success('You
2139                             have successfully
2140                             updated your profile
2141                             !');
2142                         dispatch(actions.
2143                             setLoading(false));
2144                         dispatch(actions.
2145                             setCurrentProfile());
2146                     });
2147                 });
2148             );
2149             .catch(err => {
2150                 dispatch({
2151                     type: APP_GET_ERRORS ,
2152                     payload: err.response
2153                     .data,
2154                 });
2155                 dispatch(actions.setLoading(false));
2156             });
2157             return {
2158                 type:
2159                     APP_UPDATE_PARENT_PROFILE
2160                     ,
2161             };
2162         );
2163         export function reducer(state
2164             , action) {
2165             switch (action.type) {
2166                 case
2167                     APP_UPDATE_PARENT_PROFILE
2168                     :
2169                         return {
2170                             ...state,
2171                         };
2172                 default:
2173                     return state;
2174             }
2175             // Rekit uses a new approach
2176             to organizing actions and
2177             reducers. That is
2178             // putting related actions
2179             and reducers in one file.
2180             See more at:
2181             // https://medium.com/
2182             @nate_wang/a-new-approach-
2183             for-managing-redux-actions
2184             -91c26ce8b5da
2185
2186         import {
2187             APP_UPDATE_REGISTRAR_PROFILE
2188             , APP_GET_ERRORS } from
2189             './constants';
2190         import * as actions from './actions';
2191         import axios from 'axios';
2192         import { message } from 'antd';
2193             ';
2194         );
2195         export const
2196             updateRegistrarProfile =
2197                 data => dispatch => {
2198                 dispatch(actions.setLoading(true));
2199                 axios
2200                     .post('api/registrar/
2201                         updateprofile', data)
2202                         .then(res => {
2203                             message.success('You
2204                                 have successfully
2205                                 updated your profile
2206                                 !');
2207                             dispatch(actions.
2208                                 setLoading(false));
2209                             dispatch(actions.
2210                                 setCurrentProfile());
2211                         });
2212                 );
2213             );
2214         );
2215         export function reducer(state
2216             , action) {
2217             switch (action.type) {
2218                 case
2219                     APP_UPDATE_REGISTRAR_PROFILE
2220                     :
2221                         return {
2222                             ...state,
2223                         };
2224                 default:
2225                     return state;
2226             }
2227             // Rekit uses a new approach
2228             to organizing actions and
2229             reducers. That is
2230             // putting related actions
2231             and reducers in one file.
2232             See more at:
2233             // https://medium.com/
2234             @nate_wang/a-new-approach-
2235             for-managing-redux-actions
2236             -91c26ce8b5da
2237
2238         import {
2239             APP_UPDATE_REGISTRAR_PROFILE
2240             , APP_GET_ERRORS } from
2241             './constants';
2242         import * as actions from './actions';
2243         import axios from 'axios';
2244         import { message } from 'antd';
2245             ';
2246         );
2247         export const
2248             updateRegistrarProfile =
2249                 data => dispatch => {
2250                 dispatch(actions.setLoading(true));
2251                 axios
2252                     .post('api/registrar/
2253                         updateprofile', data)
2254                         .then(res => {
2255                             message.success('You
2256                                 have successfully
2257                                 updated your profile
2258                                 !');
2259                             dispatch(actions.
2260                                 setLoading(false));
2261                             dispatch(actions.
2262                                 setCurrentProfile());
2263                         });
2264                 );
2265             );
2266         );
2267         export function reducer(state
2268             , action) {
2269             switch (action.type) {
2270                 case
2271                     APP_UPDATE_REGISTRAR_PROFILE
2272                     :
2273                         return {
2274                             ...state,
2275                         };
2276                 default:
2277                     return state;
2278             }
2279             // Rekit uses a new approach
2280             to organizing actions and
2281             reducers. That is
2282             // putting related actions
2283             and reducers in one file.
2284             See more at:
2285             // https://medium.com/
2286             @nate_wang/a-new-approach-
2287             for-managing-redux-actions
2288             -91c26ce8b5da
2289
2290         import {
2291             APP_UPDATE_REGISTRAR_PROFILE
2292             , APP_GET_ERRORS } from
2293             './constants';
2294         import * as actions from './actions';
2295         import axios from 'axios';
2296         import { message } from 'antd';
2297             ';
2298         );
2299         export const
2300             updateRegistrarProfile =
2301                 data => dispatch => {
2302                 dispatch(actions.setLoading(true));
2303                 axios
2304                     .post('api/registrar/
2305                         updateprofile', data)
2306                         .then(res => {
2307                             message.success('You
2308                                 have successfully
2309                                 updated your profile
2310                                 !');
2311                             dispatch(actions.
2312                                 setLoading(false));
2313                             dispatch(actions.
2314                                 setCurrentProfile());
2315                         });
2316                 );
2317             );
2318         );
2319         export function reducer(state
2320             , action) {
2321             switch (action.type) {
2322                 case
2323                     APP_UPDATE_REGISTRAR_PROFILE
2324                     :
2325                         return {
2326                             ...state,
2327                         };
2328                 default:
2329                     return state;
2330             }
2331             // Rekit uses a new approach
2332             to organizing actions and
2333             reducers. That is
2334             // putting related actions
2335             and reducers in one file.
2336             See more at:
2337             // https://medium.com/
2338             @nate_wang/a-new-approach-
2339             for-managing-redux-actions
2340             -91c26ce8b5da
2341
2342         import {
2343             APP_UPDATE_REGISTRAR_PROFILE
2344             , APP_GET_ERRORS } from
2345             './constants';
2346         import * as actions from './actions';
2347         import axios from 'axios';
2348         import { message } from 'antd';
2349             ';
2350         );
2351         export const
2352             updateRegistrarProfile =
2353                 data => dispatch => {
2354                 dispatch(actions.setLoading(true));
2355                 axios
2356                     .post('api/registrar/
2357                         updateprofile', data)
2358                         .then(res => {
2359                             message.success('You
2360                                 have successfully
2361                                 updated your profile
2362                                 !');
2363                             dispatch(actions.
2364                                 setLoading(false));
2365                             dispatch(actions.
2366                                 setCurrentProfile());
2367                         });
2368                 );
2369             );
2370         );
2371         export function reducer(state
2372             , action) {
2373             switch (action.type) {
2374                 case
2375                     APP_UPDATE_REGISTRAR_PROFILE
2376                     :
2377                         return {
2378                             ...state,
2379                         };
2380                 default:
2381                     return state;
2382             }
2383             // Rekit uses a new approach
2384             to organizing actions and
2385             reducers. That is
2386             // putting related actions
2387             and reducers in one file.
2388             See more at:
2389             // https://medium.com/
2390             @nate_wang/a-new-approach-
2391             for-managing-redux-actions
2392             -91c26ce8b5da
2393
2394         import {
2395             APP_UPDATE_REGISTRAR_PROFILE
2396             , APP_GET_ERRORS } from
2397             './constants';
2398         import * as actions from './actions';
2399         import axios from 'axios';
2400         import { message } from 'antd';
2401             ';
2402         );
2403         export const
2404             updateRegistrarProfile =
2405                 data => dispatch => {
2406                 dispatch(actions.setLoading(true));
2407                 axios
2408                     .post('api/registrar/
2409                         updateprofile', data)
2410                         .then(res => {
2411                             message.success('You
2412                                 have successfully
2413                                 updated your profile
2414                                 !');
2415                             dispatch(actions.
2416                                 setLoading(false));
2417                             dispatch(actions.
2418                                 setCurrentProfile());
2419                         });
2420                 );
2421             );
2422         );
2423         export function reducer(state
2424             , action) {
2425             switch (action.type) {
2426                 case
2427                     APP_UPDATE_REGISTRAR_PROFILE
2428                     :
2429                         return {
2430                             ...state,
2431                         };
2432                 default:
2433                     return state;
2434             }
2435             // Rekit uses a new approach
2436             to organizing actions and
2437             reducers. That is
2438             // putting related actions
2439             and reducers in one file.
2440             See more at:
2441             // https://medium.com/
2442             @nate_wang/a-new-approach-
2443             for-managing-redux-actions
2444             -91c26ce8b5da
2445
2446         import {
2447             APP_UPDATE_REGISTRAR_PROFILE
2448             , APP_GET_ERRORS } from
2449             './constants';
2450         import * as actions from './actions';
2451         import axios from 'axios';
2452         import { message } from 'antd';
2453             ';
2454         );
2455         export const
2456             updateRegistrarProfile =
2457                 data => dispatch => {
2458                 dispatch(actions.setLoading(true));
2459                 axios
2460                     .post('api/registrar/
2461                         updateprofile', data)
2461                         .then(res => {
2462                             message.success('You
2463                                 have successfully
2464                                 updated your profile
2465                                 !');
2466                             dispatch(actions.
2467                                 setLoading(false));
2468                             dispatch(actions.
2469                                 setCurrentProfile());
2470                         });
2471                 );
2472             );
2473         );
2474         export function reducer(state
2475             , action) {
2476             switch (action.type) {
2477                 case
2478                     APP_UPDATE_REGISTRAR_PROFILE
2479                     :
2480                         return {
2481                             ...state,
2482                         };
2483                 default:
2484                     return state;
2485             }
2486             // Rekit uses a new approach
2487             to organizing actions and
2488             reducers. That is
2489             // putting related actions
2490             and reducers in one file.
2491             See more at:
2492             // https://medium.com/
2493             @nate_wang/a-new-approach-
2494             for-managing-redux-actions
2495             -91c26ce8b5da
2496
2497         import {
2498             APP_UPDATE_REGISTRAR_PROFILE
2499             , APP_GET_ERRORS } from
2500             './constants';
2501         import * as actions from './actions';
2502         import axios from 'axios';
2503         import { message } from 'antd';
2504             ';
2505         );
2506         export const
2507             updateRegistrarProfile =
2508                 data => dispatch => {
2509                 dispatch(actions.setLoading(true));
2510                 axios
2511                     .post('api/registrar/
2512                         updateprofile', data)
2512                         .then(res => {
2513                             message.success('You
2514                                 have successfully
2515                                 updated your profile
2516                                 !');
2517                             dispatch(actions.
2518                                 setLoading(false));
2519                             dispatch(actions.
2520                                 setCurrentProfile());
2521                         });
2522                 );
2523             );
2524         );
2525         export function reducer(state
2526             , action) {
2527             switch (action.type) {
2528                 case
2529                     APP_UPDATE_REGISTRAR_PROFILE
2530                     :
2531                         return {
2532                             ...state,
2533                         };
2534                 default:
2535                     return state;
2536             }
2537             // Rekit uses a new approach
2538             to organizing actions and
2539             reducers. That is
2540             // putting related actions
2541             and reducers in one file.
2542             See more at:
2543             // https://medium.com/
2544             @nate_wang/a-new-approach-
2545             for-managing-redux-actions
2546             -91c26ce8b5da
2547
2548         import {
2549             APP_UPDATE_REGISTRAR_PROFILE
2550             , APP_GET_ERRORS } from
2551             './constants';
2552         import * as actions from './actions';
2553         import axios from 'axios';
2554         import { message } from 'antd';
2555             ';
2556         );
2557         export const
2558             updateRegistrarProfile =
2559                 data => dispatch => {
2560                 dispatch(actions.setLoading(true));
2561                 axios
2562                     .post('api/registrar/
2563                         updateprofile', data)
2563                         .then(res => {
2564                             message.success('You
2565                                 have successfully
2566                                 updated your profile
2567                                 !');
2568                             dispatch(actions.
2569                                 setLoading(false));
2570                             dispatch(actions.
2571                                 setCurrentProfile());
2572                         });
2573                 );
2574             );
2575         );
2576         export function reducer(state
2577             , action) {
2578             switch (action.type) {
2579                 case
2580                     APP_UPDATE_REGISTRAR_PROFILE
2581                     :
2582                         return {
2583                             ...state,
2584                         };
2585                 default:
2586                     return state;
2587             }
2588             // Rekit uses a new approach
2589             to organizing actions and
2590             reducers. That is
2591             // putting related actions
2592             and reducers in one file.
2593             See more at:
2594             // https://medium.com/
2595             @nate_wang/a-new-approach-
2596             for-managing-redux-actions
2597             -91c26ce8b5da
2598
2599         import {
2600             APP_UPDATE_REGISTRAR_PROFILE
2601             , APP_GET_ERRORS } from
2602             './constants';
2603         import * as actions from './actions';
2604         import axios from 'axios';
2605         import { message } from 'antd';
2606             ';
2607         );
2608         export const
2609             updateRegistrarProfile =
2610                 data => dispatch => {
2611                 dispatch(actions.setLoading(true));
2612                 axios
2613                     .post('api/registrar/
2614                         updateprofile', data)
2614                         .then(res => {
2615                             message.success('You
2616                                 have successfully
2617                                 updated your profile
2618                                 !');
2619                             dispatch(actions.
2620                                 setLoading(false));
2621                             dispatch(actions.
2622                                 setCurrentProfile());
2623                         });
2624                 );
2625             );
2626         );
2627         export function reducer(state
2628             , action) {
2629             switch (action.type) {
2630                 case
2631                     APP_UPDATE_REGISTRAR_PROFILE
2632                     :
2633                         return {
2634                             ...state,
2635                         };
2636                 default:
2637                     return state;
2638             }
2639             // Rekit uses a new approach
2640             to organizing actions and
2641             reducers. That is
2642             // putting related actions
2643             and reducers in one file.
2644             See more at:
2645             // https://medium.com/
2646             @nate_wang/a-new-approach-
2647             for-managing-redux-actions
2648             -91c26ce8b5da
2649
2650         import {
2651             APP_UPDATE_REGISTRAR_PROFILE
2652             , APP_GET_ERRORS } from
2653             './constants';
2654         import * as actions from './actions';
2655         import axios from 'axios';
2656         import { message } from 'antd';
2657             ';
2658         );
2659         export const
2660             updateRegistrarProfile =
2661                 data => dispatch => {
2662                 dispatch(actions.setLoading(true));
2663                 axios
2664                     .post('api/registrar/
2665                         updateprofile', data)
2665                         .then(res => {
2666                             message.success('You
2667                                 have successfully
2668                                 updated your profile
2669                                 !');
2670                             dispatch(actions.
2671                                 setLoading(false));
2672                             dispatch(actions.
2673                                 setCurrentProfile());
2674                         });
2675                 );
2676             );
2677         );
2678         export function reducer(state
2679             , action) {
2680             switch (action.type) {
2681                 case
2682                     APP_UPDATE_REGISTRAR_PROFILE
2683                     :
2684                         return {
2685                             ...state,
2686                         };
2687                 default:
2688                     return state;
2689             }
2690             // Rekit uses a new approach
2691             to organizing actions and
2692             reducers. That is
2693             // putting related actions
2694             and reducers in one file.
2695             See more at:
2696             // https://medium.com/
2697             @nate_wang/a-new-approach-
2698             for-managing-redux-actions
2699             -91c26ce8b5da
2700
2701         import {
2702             APP_UPDATE_REGISTRAR_PROFILE
2703             , APP_GET_ERRORS } from
2704             './constants';
2705         import * as actions from './actions';
2706         import axios from 'axios';
2707         import { message } from 'antd';
2708             ';
2709         );
2710         export const
2711             updateRegistrarProfile =
2712                 data => dispatch => {
2713                 dispatch(actions.setLoading(true));
2714                 axios
2715                     .post('api/registrar/
2716                         updateprofile', data)
2716                         .then(res => {
2717                             message.success('You
2718                                 have successfully
2719                                 updated your profile
2720                                 !');
2721                             dispatch(actions.
2722                                 setLoading(false));
2723                             dispatch(actions.
2724                                 setCurrentProfile());
2725                         });
2726                 );
2727             );
2728         );
2729         export function reducer(state
2730             , action) {
2731             switch (action.type) {
2732                 case
2733                     APP_UPDATE_REGISTRAR_PROFILE
2734                     :
2735                         return {
2736                             ...state,
2737                         };
2738                 default:
2739                     return state;
2740             }
2741             // Rekit uses a new approach
2742             to organizing actions and
2743             reducers. That is
2744             // putting related actions
2745             and reducers in one file.
2746             See more at:
2747             // https://medium.com/
2748             @nate_wang/a-new-approach-
2749             for-managing-redux-actions
2750             -91c26ce8b5da
2751
2752         import {
2753             APP_UPDATE_REGISTRAR_PROFILE
2754             , APP_GET_ERRORS } from
2755             './constants';
2756         import * as actions from './actions';
2757         import axios from 'axios';
2758         import { message } from 'antd';
2759             ';
2760         );
2761         export const
2762             updateRegistrarProfile =
2763                 data => dispatch => {
2764                 dispatch(actions.setLoading(true));
2765                 axios
2766                     .post('api/registrar/
2767                         updateprofile', data)
2767                         .then(res => {
2768                             message.success('You
2769                                 have successfully
2770                                 updated your profile
2771                                 !');
2772                             dispatch(actions.
2773                                 setLoading(false));
2774                             dispatch(actions.
2775                                 setCurrentProfile());
2776                         });
2777                 );
2778             );
2779         );
2780         export function reducer(state
2781             , action) {
2782             switch (action.type) {
2783                 case
2784                     APP_UPDATE_REGISTRAR_PROFILE
2785                     :
2786                         return {
2787                             ...state,
2788                         };
2789                 default:
2790                     return state;
2791             }
2792             // Rekit uses a new approach
2793             to organizing actions and
2794             reducers. That is
2795             // putting related actions
2796             and reducers in one file.
2797             See more at:
2798             // https://medium.com/
2799             @nate_wang/a-new-approach-
2800             for-managing-redux-actions
2801             -91c26ce8b5da
2802
2803         import {
2804             APP_UPDATE_REGISTRAR_PROFILE
2805             , APP_GET_ERRORS } from
2806             './constants';
2807         import * as actions from './actions';
2808         import axios from 'axios';
2809         import { message } from 'antd';
2810             ';
2811         );
2812         export const
2813             updateRegistrarProfile =
2814                 data => dispatch => {
2815                 dispatch(actions.setLoading(true));
2816                 axios
2817                     .post('api/registrar/
2818                         updateprofile', data)
2818                         .then(res => {
2819                             message.success('You
2820                                 have successfully
2821                                 updated your profile
2822                                 !');
2823                             dispatch(actions.
2824                                 setLoading(false));
2825                             dispatch(actions.
2826                                 setCurrentProfile());
2827                         });
2828                 );
2829             );
2830         );
2831         export function reducer(state
2832             , action) {
2833             switch (action.type) {
2834                 case
2835                     APP_UPDATE_REGISTRAR_PROFILE
2836                     :
2837                         return {
2838                             ...state,
2839                         };
2840                 default:
2841                     return state;
2842             }
2843             // Rekit uses a new approach
2844             to organizing actions and
2845             reducers. That is
2846             // putting related actions
2847             and reducers in one file.
2848             See more at:
2849             // https://medium.com/
2850             @nate_wang/a-new-approach-
2851             for-managing-redux-actions
2852             -91c26ce8b5da
2853
2854         import {
2855             APP_UPDATE_REGISTRAR_PROFILE
2856             , APP_GET_ERRORS } from
2857             './constants';
2858         import * as actions from './actions';
2859         import axios from 'axios';
2860         import { message } from 'antd';
2861             ';
2862         );
2863         export const
2864             updateRegistrarProfile =
2865                 data => dispatch => {
2866                 dispatch(actions.setLoading(true));
2867                 axios
2868                     .post('api/registrar/
2869                         updateprofile', data)
2869                         .then(res => {
2870                             message.success('You
2871                                 have successfully
2872                                 updated your profile
2873                                 !');
2874                             dispatch(actions.
2875                                 setLoading(false));
2876                             dispatch(actions.
2877                                 setCurrentProfile());
2878                         });
2879                 );
2880             );
2881         );
2882         export function reducer(state
2883             , action) {
2884             switch (action.type) {
2885                 case
2886                     APP_UPDATE_REGISTRAR_PROFILE
2887                     :
2888                         return {
2889                             ...state,
2890                         };
2891                 default:
2892                     return state;
2893             }
2894             // Rekit uses a new approach
2895             to organizing actions and
2896             reducers. That is
2897             // putting related actions
2898             and reducers in one file.
2899             See more at:
2900             // https://medium.com/
2901             @nate_wang/a-new-approach-
2902             for-managing-redux-actions
2903             -91c26ce8b5da
2904
2905         import {
2906             APP_UPDATE_REGISTRAR_PROFILE
2907             , APP_GET_ERRORS } from
2908             './constants';
2909         import * as actions from './actions';
2910         import axios from 'axios';
2911         import { message } from 'antd';
2912             ';
2913         );
2914         export const
2915             updateRegistrarProfile =
2916                 data => dispatch => {
2917                 dispatch(actions.setLoading(true));
2918                 axios
2919                     .post('api/registrar/
2920                         updateprofile', data)
2920                         .then(res => {
2921                             message.success('You
2922                                 have successfully
2923                                 updated your profile
2924                                 !');
2925                             dispatch(actions.
2926                                 setLoading(false));
2927                             dispatch(actions.
2928                                 setCurrentProfile());
2929                         });
2930                 );
2931             );
2932         );
2933         export function reducer(state
2934             , action) {
2935             switch (action.type) {
2936                 case
2937                     APP_UPDATE_REGISTRAR_PROFILE
2938                     :
2939                         return {
2940                             ...state,
2941                         };
2942                 default:
2943                     return state;
2944             }
2945             // Rekit uses a new approach
2946             to organizing actions and
2947             reducers. That is
2948             // putting related actions
2949             and reducers in one file.
2950             See more at:
2951             // https://medium.com/
2952             @nate_wang/a-new-approach-
2953             for-managing-redux-actions
2954             -91c26ce8b5da
2955
2956         import {
2957             APP_UPDATE_REGISTRAR_PROFILE
2958             , APP_GET_ERRORS } from
2959             './constants';
2960         import * as actions from './actions';
2961         import axios from 'axios';
2962         import { message } from 'antd';
2963             ';
2964         );
2965         export const
2966             updateRegistrarProfile =
2967                 data => dispatch => {
2968                 dispatch(actions.setLoading(true));
2969                 axios
2970                     .post('api/registrar/
2971                         updateprofile', data)
2971                         .then(res => {
2972                             message.success('You
2973                                 have successfully
2974                                 updated your profile
2975                                 !');
2976                             dispatch(actions.
2977                                 setLoading(false));
2978                             dispatch(actions.
2979                                 setCurrentProfile());
2980                         });
2981                 );
2982             );
2983         );
2984         export function reducer(state
2985             , action) {
2986             switch (action.type) {
2987                 case
2988                     APP_UPDATE_REGISTRAR_PROFILE
2989                     :
2990                         return {
2991                             ...state,
2992                         };
2993                 default:
2994                     return state;
2995             }
2996             // Rekit uses a new approach
2997             to organizing actions and
2998             reducers. That is
2999             // putting related actions
3000             and reducers in one file.
3001             See more at:
3002             // https://medium.com/
3003             @nate_wang/a-new-approach-
3004             for-managing-redux-actions
3005             -91c26ce8b5da
3006
3007         import {
3008             APP_UPDATE_REGISTRAR_PROFILE
3009             , APP_GET_ERRORS } from
3010             './constants';
3011         import * as actions from './actions';
3012         import axios from 'axios';
3013         import { message } from 'antd';
3014             ';
3015         );
3016         export const
3017             updateRegistrarProfile =
3018                 data => dispatch => {
3019                 dispatch(actions.setLoading(true));
3020                 axios
3021                     .post('api/registrar/
3022                         updateprofile', data)
3022                         .then(res => {
3023                             message.success('You
3024                                 have successfully
3025                                 updated your profile
3026                                 !');
3027                             dispatch(actions.
3028                                 setLoading(false));
3029                             dispatch(actions.
3030                                 setCurrentProfile());
3031                         });
3032                 );
3033             );
3034         );
3035         export function reducer(state
3036             , action) {
3037             switch (action.type) {
3038                 case
3039                     APP_UPDATE_REGISTRAR_PROFILE
3040                     :
3041                         return {
3042                             ...state,
3043                         };
3044                 default:
3045                     return state;
3046             }
3047             // Rekit uses a new approach
3048             to organizing actions and
3049             reducers. That is
3050             // putting related actions
3051             and reducers in one file.
3052             See more at:
3053             // https://medium.com/
3054             @nate_wang/a-new-approach-
3055             for-managing-redux-actions
3056             -91c26ce8b5da
3057
3058         import {
3059             APP_UPDATE_REGISTRAR_PROFILE
3060             , APP_GET_ERRORS } from
3061             './constants';
3062         import * as actions from './actions';
3063         import axios from 'axios';
3064         import { message } from 'antd';
3065             ';
3066         );
3067         export const
3068             updateRegistrarProfile =
3069                 data => dispatch => {
3070                 dispatch(actions.setLoading(true));
3071                 axios
3072                     .post('api/registrar/
3073                         updateprofile', data)
3073                         .then(res => {
3074                             message.success('You
3075                                 have successfully
3076                                 updated your profile
3077                                 !');
3078                             dispatch(actions.
3079                                 setLoading(false));
3080                             dispatch(actions.
3081                                 setCurrentProfile());
3082                         });
3083                 );
3084             );
3085         );
3086         export function reducer(state
3087             , action) {
3088             switch (action.type) {
3089                 case
3090                     APP_UPDATE_REGISTRAR_PROFILE
3091                     :
3092                         return {
3093                             ...state,
3094                         };
3095                 default:
3096                     return state;
3097             }
3098             // Rekit uses a new approach
3099             to organizing actions and
3100             reducers. That is
3101             // putting related actions
3102             and reducers in one file.
3103             See more at:
3104             // https://medium.com/
3105             @nate_wang/a-new-approach-
3106             for-managing-redux-actions
3107             -91c26ce8b5da
3108
3109         import {
3110             APP_UPDATE_REGISTRAR_PROFILE
3111             , APP_GET_ERRORS } from
3112             './constants';
3113         import * as actions from './actions';
3114         import axios from 'axios';
3115         import { message } from 'antd';
3116             ';
3117         );
3118         export const
3119             updateRegistrarProfile =
3120                 data => dispatch => {
3121                 dispatch(actions.setLoading(true));
3122                 axios
3123                     .post('api/registrar/
3124                         updateprofile', data)
3124                         .then(res => {
3125                             message.success('You
3126                                 have successfully
3127                                 updated your profile
3128                                 !');
3129                             dispatch(actions.
3130                                 setLoading(false));
3131                             dispatch(actions.
3132                                 setCurrentProfile());
3133                         });
3134                 );
3135             );
3136         );
3137         export function reducer(state
3138             , action) {
3139             switch (action.type) {
3140                 case
3141                     APP_UPDATE_REGISTRAR_PROFILE
3142                     :
3143                         return {
3144                             ...state,
3145                         };
3146                 default:
3147                     return state;
3148             }
3149             // Rekit uses a new approach
3150             to organizing actions and
3151             reducers. That is
3152             // putting related actions
3153             and reducers in one file.
3154             See more at:
3155             // https://medium.com/
3156             @nate_wang/a-new-approach-
3157             for-managing-redux-actions
3158             -91c26ce8b5da
3159
3160         import {
3161             APP_UPDATE_REGISTRAR_PROFILE
3162             , APP_GET_ERRORS } from
3163             './constants';
3164         import * as actions from './actions';
3165         import axios from 'axios';
3166         import { message } from 'antd';
3167             ';
3168         );
3169         export const
3170             updateRegistrarProfile =
3171                 data => dispatch => {
3172                 dispatch(actions.setLoading(true));
3173                 axios
3174                     .post('api/registrar/
3175                         updateprofile', data)
3175                         .then(res => {
3176                             message.success('You
3177                                 have successfully
3178                                 updated your profile
3179                                 !');
3180                             dispatch(actions.
3181                                 setLoading(false));
3182                             dispatch(actions.
3183                                 setCurrentProfile());
3184                         });
3185                 );
3186             );
3187         );
3188         export function reducer(state
3189             , action) {
3190             switch (action.type) {
3191                 case
3192                     APP_UPDATE_REGISTRAR_PROFILE
3193                     :
3194                         return {
3195                             ...state,
3196                         };
3197                 default:
3198                     return state;
3199             }
3200             // Rekit uses a new approach
3201             to organizing actions and
3202             reducers. That is
3203             // putting related actions
3204             and reducers in one file.
3205             See more at:
3206             // https://medium.com/
3207             @nate_wang/a-new-approach-
3208             for-managing-redux-actions
3209             -91c26ce8b5da
3210
3211         import {
3212             APP_UPDATE_REGISTRAR_PROFILE
3213             , APP_GET_ERRORS } from
3214             './constants';
3215         import * as actions from './actions';
3216         import axios from 'axios';
3217         import { message } from 'antd';
3218             ';
3219         );
3220         export const
3221             updateRegistrarProfile =
3222                 data => dispatch => {
3223                 dispatch(actions.setLoading(true));
3224                 axios
3225                     .post('api/registrar/
3226                         updateprofile', data)
3226                         .then(res => {
3227                             message.success('You
3228                                 have successfully
3229                                 updated your profile
3230                                 !');
3231                             dispatch(actions.
3232                                 setLoading(false));
3233                             dispatch(actions.
3234                                 setCurrentProfile());
3235                         });
3236                 );
3237             );
3238         );
3239         export function reducer(state
3240             , action) {
3241             switch (action.type) {
3242                 case
3243                     APP_UPDATE_REGISTRAR_PROFILE
3244                     :
3245                         return {
3246                             ...state,
3247                         };
3248                 default:
3249                     return state;
3250             }
3251             // Rekit uses a new approach
3252             to organizing actions and
3253             reducers. That is
3254             // putting related actions
3255             and reducers in one file.
3256             See more at:
3257             // https://medium.com/
3258             @nate_wang/a-new-approach-
3259             for-managing-redux-actions
3260             -91c26ce8b5da
3261
3262         import {
3263             APP_UPDATE_REGISTRAR_PROFILE
3264             , APP_GET_ERRORS } from
3265             './constants';
3266         import * as actions from './actions';
3267         import axios from 'axios';
3268         import { message } from 'antd';
3269             ';
3270         );
3271         export const
3272             updateRegistrarProfile =
3273                 data => dispatch => {
3274                 dispatch(actions.setLoading(true));
3275                 axios
3276                     .post('api/registrar/
3277                         updateprofile', data)
3277                         .then(res => {
3278                             message.success('You
3279                                 have successfully
3280                                 updated your profile
3281                                 !');
3282                             dispatch(actions.
3283                                 setLoading(false));
3284                             dispatch(actions.
3285                                 setCurrentProfile());
3286                         });
3287                 );
3288             );
3289         );
3290         export function reducer(state
3291             , action) {
3292             switch (action.type) {
3293                 case
3294                     APP_UPDATE_REGISTRAR_PROFILE
3295                     :
3296                         return {
3297                             ...state,
3298                         };
3299                 default:
3300                     return state;
3301             }
3302             // Rekit uses a new approach
3303             to organizing actions and
3304             reducers. That is
3305             // putting related actions
3306             and reducers in one file.
3307             See more at:
3308             // https://medium.com/
3309             @nate_wang/a-new-approach-
3310             for-managing-redux-actions
3311             -91c26ce8b5da
3312
3313         import {
3314             APP_UPDATE_REGISTRAR_PROFILE
3315             , APP_GET_ERRORS } from
3316             './constants';
3317         import * as actions from './actions';
3318         import axios from 'axios';
3319         import { message } from 'antd';
3320             ';
3321         );
3322         export const
3323             updateRegistrarProfile =
3324                 data => dispatch => {
3325                 dispatch(actions.setLoading(true));
3326                 axios
3327                     .post('api/registrar/
3328                         updateprofile', data)
3328                         .then(res => {
3329                             message.success('You
3330                                 have successfully
3331                                 updated your profile
3332                                 !');
3333                             dispatch(actions.
3334                                 setLoading(false));
3335                             dispatch(actions.
3336                                 setCurrentProfile());
3337                         });
3338                 );
3339             );
3340         );
3341         export function reducer(state
3342             , action) {
3343             switch (action.type) {
3344                 case
3345                     APP_UPDATE_REGISTRAR_PROFILE
3346                     :
3347                         return {
3348                             ...state,
3349                         };
3350                 default:
3351                     return state;
3352             }
3353             // Rekit uses a new approach
3354             to organizing actions and
3355             reducers. That is
3356             // putting related actions
3357             and reducers in one file.
3358             See more at:
3359             // https://medium.com/
3360             @nate_wang/a-new-approach-
3361             for-managing-redux-actions
3362             -91c26ce8b5da
3363
3364         import {
3365             APP_UPDATE_REGISTRAR_PROFILE
3366             , APP_GET_ERRORS } from
3367             './constants';
3368         import * as actions from './actions';
3369         import axios from 'axios';
3370         import { message } from 'antd';
3371             ';
3372         );
3373         export const
3374             updateRegistrarProfile =
3375                 data => dispatch => {
3376                 dispatch(actions.setLoading(true));
3377                 axios
3378                     .post('api/reg
```

```

2140 ; 2182
2140     dispatch({ type: 2182
2140         APP_GET_ERRORS, 2183
2140         payload: {} });
2141     }) 2184
2142     .catch(err => { 2185
2143         dispatch({
2144             type: APP_GET_ERRORS, 2186
2144             payload: err.response
2145                 .data,
2146             });
2147             dispatch(actions. 2187
2147                 setLoading(false));
2148         });
2149         return {
2150             type:
2150                 APP_UPDATE_REGISTRAR;
2150             };
2151         );
2152     );
2153
2154     export function reducer(state, 2194
2154         action) {
2155             switch (action.type) {
2156                 case
2156                     APP_UPDATE_REGISTRAR;
2156                     :
2157                     return {
2158                         ...state,
2159                         };
2160                     };
2161                     default:
2162                         return state;
2163                     );
2164     }
2165     // Rekit uses a new approach
2165     to organizing actions and
2165     reducers. That is
2166     // putting related actions
2166     and reducers in one file.
2166     See more at:
2167     // https://medium.com/
2167     @nate_wang/a-new-approach-
2167     for-managing-redux-actions
2167     -91c26ce8b5da
2168     import {
2169         APP_UPDATE_STUDENT_PROFILE
2169         , APP_GET_ERRORS } from
2169         './constants';
2170     import * as actions from './actions';
2171     import axios from 'axios';
2172     import { message } from 'antd';
2172         ';
2173
2174     export const
2174         updateStudentProfile = 2214
2174         data => dispatch => {
2175             dispatch(actions.setLoading(true));
2176             axios
2176                 .post('api/student/ 2217
2176                     updateprofile', data);
2177             .then(res => {
2177                 message.success('You
2177                     have successfully
2177                     updated your profile
2177                     !');
2178                 dispatch(actions. 2220
2178                     setLoading(false));
2179                 dispatch(actions.
2179                     getCurrentProfile());
2180
2181                 dispatch({ type:
2181                     APP_GET_ERRORS,

```

```
2222         payload: {} });
2223     })
2224     .catch(err => {
2225       dispatch({ type:
2226         APP_GET_ERRORS,
2227         payload: err,
2228         response.data });
2229       dispatch(actions.
2230         setLoading(false));
2231     });
2232   return {
2233     type:
2234       APP_UPDATE_TEACHER_PROFILE
2235       ,
2236     };
2237   );
2238 }
2239 default:
2240   return state;
2241 }
2242 }
```

Listing 49: React Components (Client)

```
1 import React, { Component } from 'react';
2 import PropTypes from 'prop-types';
3 import { bindActionCreators } from 'redux';
4 import { connect } from 'react-redux';
5 import * as actions from './redux/actions';
6 import { Card, Button, Grid, Avatar, Table, Form, Container } from 'tabler-react';
7 import { Pagination, Spin, Popconfirm, Modal, Tooltip } from 'antd';
8 import { getImageUrl } from '../../../../../utils';
9 import placeholder from '../../../../../images/placeholder.jpg';
10 import axios from 'axios';
11
12 export class AdminActivateAccount extends Component {
13   static propTypes = {
14     app: PropTypes.object.isRequired,
15     actions: PropTypes.object.isRequired,
16   };
17
18   constructor(props) {
19     super(props);
20     this.state = {
21       isLoading: true,
22       isLoading2: true,
23       keyword: '',
24       page: 1,
25       pageSize: 5,
26       numOfPages: 1,
27       data: [],
28       selectedKey: -1,
29       showModal: false,
30       selectedID: -1,
31       parentKeyword: '',
32       parentPage: 1,
33       parentPageSize: 5,
34       parentNumOfPages: 1,
35       parentData: [],
36       listParents: [],
37     };
38     this.onChange = this.onChange.bind(this);
39   }
40
41   handleParentSearch = () => {
42     axios
43       .post('api/admin/getallparents', {
44         accountID: this.state.selectedID,
45         keyword: this.state.parentKeyword,
46         page: this.state.parentPage,
47         pageSize: this.state.parentPageSize,
48       })
49       .then(res =>
50         this.setState({
51           isLoading2: false,
52           parentNumOfPages: res.data.numOfPages,
53           parentData: res.data.accountList,
54           listParents: res.data.listParents,
55         }),
56       )
57       .catch(err =>
58         this.setState({
59           isLoading2: false,
60           parentNumOfPages: 1,
61           parentData: [],
62           parentPage: 1,
63           listParents: [],
64         }),
65       );
66   }
67
68   paginate2 = page => {
69     axios
```

```

70         .post('api/admin/getallparents', {
71             keyword: this.state.parentKeyword,
72             page: page,
73             pageSize: this.state.parentPageSize,
74         })
75         .then(res =>
76             this.setState({
77                 isLoading2: false,
78                 parentNumOfPages: res.data.numOfPages,
79                 parentData: res.data.accountList,
80                 listParents: res.data.listParents,
81             }),
82         )
83         .catch(err =>
84             this.setState({
85                 isLoading2: false,
86                 parentNumOfPages: 1,
87                 parentData: [],
88                 parentPage: 1,
89                 listParents: [],
90             }),
91         );
92     );
93
94     paginate = page => {
95         this.setState({
96             page,
97         });
98         this.setState({ isLoading: true });
99         axios
100            .post('api/admin/getaccounts', {
101                keyword: this.state.keyword,
102                page,
103                pageSize: this.state.pageSize,
104            })
105            .then(res => {
106                this.setState({ isLoading: false });
107                this.setState({
108                    numOfPages: res.data.numOfPages,
109                    data: res.data.accountList,
110                });
111            })
112            .catch(err => {
113                this.setState({ isLoading: false, numOfPages: 1, data: [] , page : 1 });
114            });
115        );
116
117        onChange(event) {
118            this.setState({
119                isLoading: true,
120                page: 1,
121            });
122            this.setState({ [event.target.name]: event.target.value });
123            axios
124                .post('api/admin/getaccounts', { keyword: event.target.value,
125                    page: 1, pageSize: 5 })
126                .then(res => {
127                    this.setState({ isLoading: false });
128                    this.setState({
129                        numOfPages: res.data.numOfPages,
130                        data: res.data.accountList,
131                    });
132                })
133                .catch(err => {
134                    this.setState({ isLoading: false, data: [] });
135                });
136
137        deactivateAccount(accountID) {
138            this.props.actions.deactivateAccount({ accountID });
139        }
140

```

```

141 handleDeleteAccount = () => {
142   Modal.confirm({
143     title: 'Delete an Account',
144     content: 'Do you want to delete this account?',
145     onOk: () => this.deactivateAccount(this.state.selectedKey),
146   });
147 };
148 componentWillReceiveProps() {
149   this.setState({ isLoading: true });
150   axios
151     .post('api/admin/getaccounts', { keyword: '', page: this.state.
152       page, pageSize: 5 })
153     .then(res => {
154       this.setState({ isLoading: false });
155       this.setState({
156         numOfPages: res.data.numOfPages,
157         data: res.data.accountList,
158       });
159     })
160     .catch(err => {
161       this.setState({ isLoading: false });
162     });
163 }
164 componentWillMount() {
165   this.setState({ isLoading: true });
166   axios
167     .post('api/admin/getaccounts', { keyword: '', page: 1, pageSize:
168       5 })
169     .then(res => {
170       this.setState({ isLoading: false });
171       this.setState({
172         numOfPages: res.data.numOfPages,
173         data: res.data.accountList,
174       });
175     })
176     .catch(err => {
177       this.setState({ isLoading: false });
178     });
179 }
180 render() {
181   const DisplayData = [];
182   for (const [index, value] of this.state.data.entries()) {
183     DisplayData.push(
184       <Table.Row>
185         <Table.Col className="w-1">
186           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
187             getImageUrl(value.imageUrl)} />
188         </Table.Col>
189         <Table.Col>{value.name}</Table.Col>
190         <Table.Col>{value.email}</Table.Col>
191         <Table.Col>{value.position}</Table.Col>
192         <Table.Col>
193           <Button.List>
194             {value.position == 'Student' && (
195               <Tooltip title="Assign a parent">
196                 <Button
197                   icon="user"
198                   size="sm"
199                   pill
200                   color="primary"
201                   onClick={() =>
202                     this.setState(
203                       { showModal: true, selectedID: value.key,
204                         parentData: [], listParents: [] },
205                         () => this.handleParentSearch(),
206                           )
207                         }
208                         ></Button>
209                         </Tooltip>
210                       )}
211                     <Button>

```

```

210     icon="trash"
211     size="sm"
212     pill
213     color="danger"
214     value={value.key}
215     onClick={() =>
216       this.setState({ selectedKey: value.key }, () => this.
217         handleDeleteAccount())
218     }
219   ></Button>
220 </Button.List>
221 </Table.Col>
222 </Table.Row>,
223 );
224 }
225 <div className="app-admin-activate-account card">
226   <Modal
227     title="Assign Parent to Student"
228     visible={this.state.showModal}
229     onOk={() => {
230       this.handleAssignParent();
231     }}
232     onCancel={() => this.setState({ showModal: false })}
233     footer={[
234       <Button onClick={() => this.setState({ showModal: false ,
235         listParents: [] })}>
236         Close
237       </Button>,
238     ]}
239     cancelButton="Cancel"
240   >
241     <Grid.Row>
242       <Grid.Col sm={12} xs={12} md={12}>
243         <Form.Group>
244           <Form.Label>Search for parent</Form.Label>
245           <Form.Input
246             value={this.state.parentKeyword}
247             onChange={e =>
248               this.setState({ parentKeyword: e.target.value }, () =>
249                 this.handleParentSearch(),
250               )
251             }
252             placeholder="Search for..." />
253           </Form.Group>
254         </Grid.Col>
255       <Grid.Col sm={12} xs={12} md={12}>
256         <Spin spinning={this.state.isLoading2}>
257           <Table highlightRowOnHover={true} responsive={true}>
258             <Table.Header>
259               <Table.ColHeader></Table.ColHeader>
260               <Table.ColHeader>Name</Table.ColHeader>
261               <Table.ColHeader>Email</Table.ColHeader>
262               <Table.ColHeader>Actions</Table.ColHeader>
263             </Table.Header>
264             <Table.Body>
265               {this.state.parentData.length == 0 ? (
266                 <Table.Row>
267                   <Table.Col colSpan={4} alignContent="center">
268                     No entries.
269                   </Table.Col>
270                 </Table.Row>
271               ) : (
272                 this.state.parentData.map(value => (
273                   <Table.Row>
274                     <Table.Col className="w-1">
275                       <Avatar
276                         imageURL={
277                           value.imageUrl == 'NA' ? placeholder :
278                           getImageUrl(value.imageUrl)
279                         }

```

```

279          />
280      </Table.Col>
281      <Table.Col>{value.name}</Table.Col>
282      <Table.Col>{value.email}</Table.Col>
283      <Table.Col>
284          {this.state.listParents.length == 0 ? (
285              <Button
286                  icon="user"
287                  color="primary"
288                  pill
289                  size="sm"
290                  onClick={() =>
291                      this.props.actions.assignParent({
292                          accountID: this.state.selectedID,
293                          parentID: value.key,
294                      })
295                  }
296              >
297                  Assign
298              </Button>
299          ) : !this.state.listParents.find(a => a.
300              accountID == value.key) ? (
301                  <Button
302                      icon="user"
303                      color="primary"
304                      pill
305                      size="sm"
306                      onClick={() =>
307                          this.props.actions.assignParent({
308                              accountID: this.state.selectedID,
309                              parentID: value.key,
310                          })
311                  }
312              >
313                  Assign
314              </Button>
315          ) : (
316              ;,
317          )
318      )}
319      {this.state.listParents.length != 0 &&
320          this.state.listParents.find(a => a.
321              accountID == value.key) && (
322                  <Button
323                      icon="user"
324                      color="danger"
325                      pill
326                      size="sm"
327                      onClick={() =>
328                          this.props.actions.unassignParent({
329                              accountID: this.state.selectedID,
330                              parentID: value.key,
331                          })
332                  }
333              >
334                  Unassign
335              </Button>
336          ))
337      )
338  </Table.Body>
339 </Table>
340 <Pagination
341     size="large"
342     current={this.state.parentPage}
343     pageSize={this.state.parentPageSize}
344     total={this.state.parentPageSize * this.state.
345         parentNumOfPages}
346     onChange={this.paginate2}
347     />
348     </Spin>
349 </Grid.Col>
</Grid.Row>

```

```

350         </Modal>
351     <Card.Body>
352         <Card.Title>Accounts List</Card.Title>
353         <Grid.Row>
354             <Grid.Col sm={12} md={12} xs={12}>
355                 <Form.Group>
356                     <Form.Input
357                         icon="search"
358                         placeholder="Search for..."
359                         position="append"
360                         name="keyword"
361                         value={this.state.keyword}
362                         onChange={this.onChange}
363                     />
364                 </Form.Group>
365                 <Spin spinning={this.state.isLoading}>
366                     <Table highlightRowOnHover={true} responsive={true}>
367                         <Table.Header>
368                             <Table.ColHeader colSpan={2}>Name</Table.ColHeader>
369                             <Table.ColHeader>Email</Table.ColHeader>
370                             <Table.ColHeader>Position</Table.ColHeader>
371                             <Table.ColHeader>Action</Table.ColHeader>
372                         </Table.Header>
373                         <Table.Body>
374                             {DisplayData.length !== 0 ? (
375                                 DisplayData
376                             ) : (
377                                 <Table.Row>
378                                     <Table.Col colSpan={5} alignContent="center">
379                                         No data to display.
380                                     </Table.Col>
381                                 </Table.Row>
382                             )}
383                         </Table.Body>
384                     </Table>
385                     <Pagination
386                         size="large"
387                         current={this.state.page}
388                         pageSize={this.state.pageSize}
389                         total={this.state.pageSize * this.state.numOfPages}
390                         onChange={this.paginate}
391                     />
392                     </Spin>
393                 </Grid.Col>
394             </Grid.Row>
395         <Card.Body>
396     </div>
397 );
398 }
399 }
400 /* istanbul ignore next */
401 function mapStateToProps(state) {
402     return {
403         app: state.app,
404     };
405 }
406 /* istanbul ignore next */
407 function mapDispatchToProps(dispatch) {
408     return {
409         actions: bindActionCreators({ ...actions }, dispatch),
410     };
411 }
412
413
414
415 export default connect(mapStateToProps, mapDispatchToProps)(
416     AdminActivateAccount);
417 import React, { Component } from 'react';
418 import PropTypes from 'prop-types';
419 import { bindActionCreators } from 'redux';
420 import { connect } from 'react-redux';
421 import * as actions from './redux/actions';

```

```

421 import { Card, Form, Button, Grid, Alert } from 'tabler-react';
422
423 export class AdminCreateAccount extends Component {
424   static propTypes = {
425     app: PropTypes.object.isRequired,
426     actions: PropTypes.object.isRequired,
427   };
428
429   constructor(props) {
430     super(props);
431     this.state = {
432       email: '',
433       firstName: '',
434       lastName: '',
435       position: '',
436       middleName: '',
437       errors: {}
438     };
439     this.onSubmit = this.onSubmit.bind(this);
440     this.onChange = this.onChange.bind(this);
441   }
442
443   onSubmit(event) {
444     event.preventDefault();
445     const userData = {
446       email: this.state.email,
447       firstName: this.state.firstName,
448       lastName: this.state.lastName,
449       position: this.state.position,
450       middleName: this.state.middleName,
451       errors: {}
452     };
453     this.props.actions.createAccount(userData);
454   }
455
456   componentDidMount() {
457     this.setState({
458       email: '',
459       firstName: '',
460       lastName: '',
461       position: 1,
462       middleName: '',
463       errors: {}
464     });
465   }
466   componentWillReceiveProps(nextProps) {
467     this.setState({ errors: nextProps.app.errors });
468   }
469
470   onChange(event) {
471     this.setState({ [event.target.name]: event.target.value });
472   }
473   render() {
474     const { email, firstName, middleName, lastName, position, errors } =
475       this.state;
476     return (
477       <div className="app-admin-create-account">
478         <Form className="card" onSubmit={this.onSubmit}>
479           <Card.Body>
480             <Card.Title>Create an account</Card.Title>
481             <Grid.Row>
482               <Grid.Col xs={12} sm={12} md={12}>
483                 <Alert icon="bell" type="success">
484                   Default password for user accounts: first name + last
485                   name + 123 (example: Juan
486                   Dela Cruz, password: "juandela cruz123")
487                 </Alert>
488               </Grid.Col>
489               <Grid.Col xs={12} sm={12} md={7}>
490                 <Form.Group>
491                   <Form.Label>Email</Form.Label>
492                   <Form.Input
493                     type="email"
494                     name="email"

```

```

493         placeholder="Email"
494         value={email}
495         error={errors.email}
496         onChange={this.onChange}
497     ></Form.Input>
498   </Form.Group>
499 </Grid.Col>
500 <Grid.Col xs={12} sm={12} md={5}>
501   <Form.Group>
502     <Form.Label>Position</Form.Label>
503     <Form.Select value={position} onChange={this.onChange}>
504       ><input name="position">
505       <option value="1">Director</option>
506       <option value="2">Registrar</option>
507       <option value="3">Teacher</option>
508       <option value="4">Student</option>
509       <option value="5">Parent or Guardian</option>
510       <option value="6">Cashier</option>
511     </Form.Select>
512   </Form.Group>
513 </Grid.Col>
514 <Grid.Col xs={12} sm={12} md={4}>
515   <Form.Group>
516     <Form.Label>First Name</Form.Label>
517     <Form.Input
518       name="firstName"
519       placeholder="First Name"
520       value={firstName}
521       error={errors.firstName}
522       onChange={this.onChange}>
523     </Form.Input>
524   </Form.Group>
525 </Grid.Col>
526 <Grid.Col xs={12} sm={12} md={4}>
527   <Form.Group>
528     <Form.Label>Middle Name</Form.Label>
529     <Form.Input
530       name="middleName"
531       placeholder="Middle Name"
532       value={middleName}
533       error={errors.middleName}
534       onChange={this.onChange}>
535     </Form.Input>
536   </Form.Group>
537 </Grid.Col>
538 <Grid.Col xs={12} sm={12} md={4}>
539   <Form.Group>
540     <Form.Label>Last Name</Form.Label>
541     <Form.Input
542       name="lastName"
543       placeholder="Last Name"
544       value={lastName}
545       error={errors.lastName}
546       onChange={this.onChange}>
547     </Form.Input>
548   </Form.Group>
549 </Grid.Col>
550 </Grid.Row>
551 <Card.Body>
552   <Card.Footer className="text-right">
553     <Button type="submit" color="primary">
554       Create an account
555     </Button>
556   </Card.Footer>
557 </Form>
558   </div>
559 );
560 }
561 /* istanbul ignore next */
562 function mapStateToProps(state) {

```

```

564     return {
565       app: state.app,
566     };
567   }
568
569   /* istanbul ignore next */
570   function mapDispatchToProps(dispatch) {
571     return {
572       actions: bindActionCreators({ ...actions }, dispatch),
573     };
574   }
575
576   export default connect(mapStateToProps, mapDispatchToProps)(
577     AdminCreateAccount);
578   import React, { Component } from 'react';
579   import PropTypes from 'prop-types';
580   import { bindActionCreators } from 'redux';
581   import { connect } from 'react-redux';
582   import * as actions from './redux/actions';
583   import { Container, Grid, Card, Button, Form, Header, List } from 'tabler-react';
584   import AdminCreateAccount from './AdminCreateAccount';
585   import AdminActivateAccount from './AdminActivateAccount';
586   import AdminViewUpdateLog from './AdminViewUpdateLog';
587
588   export class AdminDashboard extends Component {
589     static propTypes = {
590       app: PropTypes.object.isRequired,
591       actions: PropTypes.object.isRequired,
592     };
593
594     render() {
595       return (
596         <div className="app-admin-dashboard my-3 my-md-5">
597           <Container>
598             <Grid.Row>
599               <Grid.Col xs={12} sm={12} md={5}>
600                 <AdminCreateAccount />
601               </Grid.Col>
602               <Grid.Col sm={12} md={7}>
603                 <AdminActivateAccount />
604               </Grid.Col>
605             </Grid.Row>
606           </Container>
607         </div>
608     );
609   }
610
611   /* istanbul ignore next */
612   function mapStateToProps(state) {
613     return {
614       app: state.app,
615     };
616   }
617
618   /* istanbul ignore next */
619   function mapDispatchToProps(dispatch) {
620     return {
621       actions: bindActionCreators({ ...actions }, dispatch),
622     };
623   }
624
625   export default connect(mapStateToProps, mapDispatchToProps)(
626     AdminDashboard);
627   import React, { Component } from 'react';
628   import PropTypes from 'prop-types';
629   import { bindActionCreators } from 'redux';
630   import { connect } from 'react-redux';
631   import * as actions from './redux/actions';
632   import moment from 'moment';
633   import axios from 'axios';

```

```

633 import { Container, Grid, Card, Button, Form, Header, List } from 'react-table';
634 import { Alert, Upload, message } from 'antd';
635 import cn from 'classnames';
636 import placeholder from '/images/placeholder.jpg';
637 import bg from '/images/BG.png';
638 import { getImageUrl } from '/utils';
639 function ProfileImage({ avatarURL }) {
640   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
641 }
642
643 function Profile({
644   className,
645   children,
646   name,
647   avatarURL = '',
648   twitterURL = '',
649   backgroundURL = '',
650   bio,
651 }) {
652   const classes = cn('card-profile', className);
653   return (
654     <Card className={classes}>
655       <Card.Header backgroundURL={backgroundURL} />
656       <Card.Body className="text-center">
657         <ProfileImage avatarURL={avatarURL} />
658         <Header.H3 className="mb-3">{name}</Header.H3>
659         <p className="mb-4">{bio} || {children}</p>
660       </Card.Body>
661     </Card>
662   );
663 }
664 export class AdminProfile extends Component {
665   static propTypes = {
666     app: PropTypes.object.isRequired,
667     actions: PropTypes.object.isRequired,
668   };
669
670   constructor(props) {
671     super(props);
672     this.state = {
673       loading: true,
674       email: '',
675       position: '',
676       firstName: '',
677       lastName: '',
678       middleName: '',
679       suffix: '',
680       nickname: '',
681       imageUrl: '',
682       contactNum: '',
683       address: '',
684       province: '',
685       city: '',
686       region: '',
687       zipcode: '',
688       civilStatus: '',
689       sex: '',
690       citizenship: '',
691       birthDate: new Date(),
692       birthPlace: '',
693       religion: '',
694       emergencyName: '',
695       emergencyAddress: '',
696       emergencyTelephone: '',
697       emergencyCellphone: '',
698       emergencyEmail: '',
699       emergencyRelationship: '',
700     };
701
702     this.onSubmit = this.onSubmit.bind(this);
703     this.onChange = this.onChange.bind(this);
704     this.onDateChange = this.onDateChange.bind(this);
705   }

```

```

706     componentWillReceiveProps(nextProps) {
707       if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
708         .length == 0)
709         this.setState({
710           email: nextProps.app.auth.user.email,
711           position: nextProps.app.auth.user.position,
712           firstName: nextProps.app.profile.firstName,
713           lastName: nextProps.app.profile.lastName,
714           middleName: nextProps.app.profile.middleName,
715           suffix: nextProps.app.profile.suffix,
716           nickname: nextProps.app.profile.nickname,
717           imageUrl: nextProps.app.profile.imageUrl,
718           contactNum: nextProps.app.profile.contactNum,
719           address: nextProps.app.profile.address,
720           province: nextProps.app.profile.province,
721           city: nextProps.app.profile.city,
722           region: nextProps.app.profile.region,
723           zipcode: nextProps.app.profile.zipcode,
724           civilStatus: nextProps.app.profile.civilStatus,
725           sex: nextProps.app.profile.sex,
726           citizenship: nextProps.app.profile.citizenship,
727           birthDate: nextProps.app.profile.birthDate,
728           birthPlace: nextProps.app.profile.birthPlace,
729           religion: nextProps.app.profile.religion,
730           emergencyName: nextProps.app.profile.emergencyName,
731           emergencyAddress: nextProps.app.profile.emergencyAddress,
732           emergencyTelephone: nextProps.app.profile.emergencyTelephone,
733           emergencyCellphone: nextProps.app.profile.emergencyCellphone,
734           emergencyEmail: nextProps.app.profile.emergencyEmail,
735           emergencyRelationship: nextProps.app.profile.
736             emergencyRelationship,
737             loading: false,
738         });
739     }
740     onChange(event) {
741       this.setState({ [event.target.name]: event.target.value });
742     }
743     onDateChange(event) {
744       this.setState({ birthDate: event });
745     }
746     onSubmit(event) {
747       event.preventDefault();
748       const {
749         firstName,
750         lastName,
751         middleName,
752         suffix,
753         nickname,
754         contactNum,
755         address,
756         province,
757         city,
758         region,
759         zipcode,
760         civilStatus,
761         sex,
762         citizenship,
763         birthDate,
764         birthPlace,
765         religion,
766         emergencyName,
767         emergencyAddress,
768         emergencyTelephone,
769         emergencyCellphone,
770         emergencyEmail,
771         emergencyRelationship,
772       } = this.state;
773     const profileData = {
774       firstName,
775       lastName,
776       middleName,
777       suffix,
778       nickname,
779       contactNum,
780     }

```

```

781     address ,
782     province ,
783     city ,
784     region ,
785     zipcode ,
786     civilStatus ,
787     sex ,
788     citizenship ,
789     birthDate ,
790     birthPlace ,
791     religion ,
792     emergencyName ,
793     emergencyAddress ,
794     emergencyTelephone ,
795     emergencyCellphone ,
796     emergencyEmail ,
797     emergencyRelationship ,
798   };
799   console.log(profileData);
800   this.props.actions.updateAdminProfile(profileData);
801 }
802
803 render() {
804   const {
805     email ,
806     position ,
807     firstName ,
808     lastName ,
809     middleName ,
810     suffix ,
811     nickname ,
812     imageUrl ,
813     contactNum ,
814     address ,
815     province ,
816     city ,
817     region ,
818     zipcode ,
819     civilStatus ,
820     sex ,
821     citizenship ,
822     birthPlace ,
823     religion ,
824     emergencyName ,
825     emergencyAddress ,
826     emergencyTelephone ,
827     emergencyCellphone ,
828     emergencyEmail ,
829     emergencyRelationship ,
830   } = this.state;
831   const birthDate = new Date(this.state.birthDate);
832   const defaultDate = birthDate.toISOString();
833   const { errors } = this.props.app;
834   const uploadLoading = false;
835   const displayPosition = position => {
836     switch (position) {
837       case false:
838         return 'Administrator';
839       case true:
840         return 'Director';
841       case 2:
842         return 'Registrar';
843       case 3:
844         return 'Teacher';
845       case 4:
846         return 'Student';
847       case 5:
848         return 'Guardian';
849       default:
850         return '';
851     }
852   };
853   const capitalize = string => {
854     return string.charAt(0).toUpperCase() + string.slice(1).
855       toLowerCase();
856   };

```

```

856     return (
857       <div className="app-admin-profile my-3 my-md-5">
858         {this.state.loading ? (
859           ,
860         ) : (
861           <Container>
862             <Grid.Row>
863               <Grid.Col sm={12} lg={4}>
864                 <Profile
865                   name={`${firstName} ${middleName.charAt(0).
866                     toUpperCase()}. ${lastName}`}
867                     avatarURL={imageUrl == 'NA' ? placeholder :
868                       getImageUrl(imageUrl)}
869                     backgroundURL={bg}
870                   >
871                     <Upload
872                       name="file"
873                         multiple={false}
874                           accept=".png, .jpg"
875                           customRequest={({req, uploadLoading) => {
876                             // Axios Photo Upload
877                             this.props.actions.setLoading(true);
878                             const bodyFormData = new FormData();
879                             bodyFormData.set('file', req.file);
880                             axios({
881                               method: 'post',
882                                 url: 'api/users/upload',
883                                   data: bodyFormData,
884                                 })
885                               .then(res => {
886                                 req.onSuccess();
887                                   this.props.actions.setLoading(false);
888                                     message.success('Uploaded Successfully!').
889                                       then(() => {
890                                         window.location.href = '/profile';
891                                       }));
892                                     })
893                                       .catch(err => {
894                                         req.onError();
895                                           this.props.actions.setLoading(false);
896                                             message.error('Upload failed!');
897                                           }));
898                                         }
899                                         >
900                                           <Header.H4>{displayPosition(position)}</Header.H4
901                                         >
902                                         <Button loading={this.props.app.showLoading} icon="
903                                           upload" pill color="primary">
904                                             Upload Image
905                                           </Button>
906                                         </Upload>
907                                         </Profile>
908                                         </Grid.Col>
909                                         <Grid.Col xs={12} sm={12} lg={8}>
910                                           <Form className="card" onSubmit={this.onSubmit}>
911                                             <Card.Body>
912                                               <Card.Title>Edit Profile</Card.Title>
913                                                 <Grid.Row>
914                                                   <Grid.Col xs={12} sm={12} md={6}>
915                                                     <Form.Group>
916                                                       <Form.Label>Email</Form.Label>
917                                                       <Form.Input
918                                                         type="email"
919                                                       disabled
920                                                         placeholder="Email"
921                                                       value={email}
922                                                         error={errors.email}
923                                                       />
924                                                     </Form.Group>
925                                                   </Grid.Col>
926                                                 <Grid.Col sm={12} md={6}>

```

```

924         <Form.Group>
925             <Form.Label>Position</Form.Label>
926             <Form.Input
927                 type="text"
928                 placeholder="Position"
929                 disabled
930                 value={displayPosition(position)}
931                 error={errors.position}
932             />
933         </Form.Group>
934     </Grid.Col>
935     <Grid.Col sm={12} md={4}>
936         <Form.Group>
937             <Form.Label>First Name</Form.Label>
938             <Form.Input
939                 type="text"
940                 name="firstName"
941                 placeholder="First Name"
942                 value={firstName}
943                 onChange={this.onChange}
944                 error={errors.firstName}
945             />
946         </Form.Group>
947     </Grid.Col>
948     <Grid.Col sm={12} md={4}>
949         <Form.Group>
950             <Form.Label>Middle Name</Form.Label>
951             <Form.Input
952                 type="text"
953                 placeholder="Middle Name"
954                 value={middleName}
955                 onChange={this.onChange}
956                 name="middleName"
957                 error={errors.middleName}
958             />
959         </Form.Group>
960     </Grid.Col>
961     <Grid.Col sm={12} md={4}>
962         <Form.Group>
963             <Form.Label>Last Name</Form.Label>
964             <Form.Input
965                 type="text"
966                 placeholder="Last Name"
967                 value={lastName}
968                 onChange={this.onChange}
969                 name="lastName"
970                 error={errors.lastName}
971             />
972         </Form.Group>
973     </Grid.Col>
974     <Grid.Col sm={12} md={3}>
975         <Form.Group>
976             <Form.Label>Nickname</Form.Label>
977             <Form.Input
978                 type="text"
979                 placeholder="Nickname"
980                 value={nickname}
981                 onChange={this.onChange}
982                 name="nickname"
983                 error={errors.nickname}
984             />
985         </Form.Group>
986     </Grid.Col>
987     <Grid.Col sm={12} md={3}>
988         <Form.Group>
989             <Form.Label>Suffix</Form.Label>
990             <Form.Input
991                 type="text"
992                 placeholder="Suffix"
993                 value={suffix}
994                 onChange={this.onChange}
995                 name="suffix"
996                 error={errors.suffix}

```

```

997          />
998      </Form.Group>
999  </Grid.Col>
1000 <Grid.Col sm={12} md={6}>
1001   <Form.Group>
1002     <Form.Label>Contact Number</Form.Label>
1003     <Form.Input
1004       type="text"
1005       placeholder="Contact Number"
1006       value={contactNum}
1007       onChange={this.onChange}
1008       name="contactNum"
1009       error={errors.contactNum}
1010     />
1011   </Form.Group>
1012 </Grid.Col>
1013 <Grid.Col sm={12} md={12}>
1014   <Form.Group>
1015     <Form.Label>Address</Form.Label>
1016     <Form.Input
1017       type="text"
1018       placeholder="Home Address"
1019       value={address}
1020       onChange={this.onChange}
1021       name="address"
1022       error={errors.address}
1023     />
1024   </Form.Group>
1025 </Grid.Col>
1026 <Grid.Col sm={12} md={4}>
1027   <Form.Group>
1028     <Form.Label>Province</Form.Label>
1029     <Form.Input
1030       type="text"
1031       placeholder="Province"
1032       value={province}
1033       onChange={this.onChange}
1034       name="province"
1035       error={errors.province}
1036     />
1037   </Form.Group>
1038 </Grid.Col>
1039 <Grid.Col sm={12} md={4}>
1040   <Form.Group>
1041     <Form.Label>City</Form.Label>
1042     <Form.Input
1043       type="text"
1044       placeholder="City"
1045       value={city}
1046       onChange={this.onChange}
1047       name="city"
1048       error={errors.city}
1049     />
1050   </Form.Group>
1051 </Grid.Col>
1052 <Grid.Col sm={12} md={2}>
1053   <Form.Group>
1054     <Form.Label>Region</Form.Label>
1055     <Form.Input
1056       type="text"
1057       placeholder="Region"
1058       value={region}
1059       onChange={this.onChange}
1060       name="region"
1061       error={errors.region}
1062     />
1063   </Form.Group>
1064 </Grid.Col>
1065 <Grid.Col sm={12} md={2}>
1066   <Form.Group>
1067     <Form.Label>Zipcode</Form.Label>
1068     <Form.Input
1069       type="text"

```

```

1070           placeholder="Postal Code"
1071           value={zipcode}
1072           onChange={this.onChange}
1073           name="zipcode"
1074           error={errors.zipcode}
1075         />
1076       </Form.Group>
1077     </Grid.Col>
1078   <Grid.Col sm={12} md={4}>
1079     <Form.Group>
1080       <Form.Label>Civil Status</Form.Label>
1081       <Form.Select
1082         value={civilStatus}
1083         onChange={this.onChange}
1084         name="civilStatus"
1085       >
1086         <option>SINGLE</option>
1087         <option>MARRIED</option>
1088         <option>WIDOWED</option>
1089         <option>OTHERS</option>
1090       </Form.Select>
1091     </Form.Group>
1092   </Grid.Col>
1093   <Grid.Col sm={12} md={2}>
1094     <Form.Group>
1095       <Form.Label>Sex</Form.Label>
1096       <Form.Select value={sex} onChange={this.
1097         onChange} name="sex">
1098         <option>M</option>
1099         <option>F</option>
1100       </Form.Select>
1101     </Form.Group>
1102   </Grid.Col>
1103   <Grid.Col sm={12} md={6}>
1104     <Form.Group>
1105       <Form.Label>Citizenship</Form.Label>
1106       <Form.Input
1107         type="text"
1108         placeholder="Citizenship"
1109         value={citizenship}
1110         onChange={this.onChange}
1111         name="citizenship"
1112         error={errors.citizenship}
1113       />
1114     </Form.Group>
1115   </Grid.Col>
1116   <Grid.Col sm={12} md={4}>
1117     <Form.Group>
1118       <Form.Label>Birth Date</Form.Label>
1119       <Form.DatePicker
1120         defaultDate={new Date(defaultDate)}
1121         format="mm/dd/yyyy"
1122         onChange={this.onDateChange}
1123         name="birthDate"
1124         maxYear={2020}
1125         minYear={1897}
1126         monthLabels={[
1127           'January',
1128           'February',
1129           'March',
1130           'April',
1131           'May',
1132           'June',
1133           'July',
1134           'August',
1135           'September',
1136           'October',
1137           'November',
1138           'December',
1139         ]}>
1140       </Form.DatePicker>
1141     </Form.Group>
   </Grid.Col>

```

```

1142 <Grid.Col sm={12} md={4}>
1143   <Form.Group>
1144     <Form.Label>Birth Place</Form.Label>
1145     <Form.Input
1146       type="text"
1147       placeholder="Birth Place"
1148       value={birthPlace}
1149       onChange={this.onChange}
1150       name="birthPlace"
1151       error={errors.birthPlace}>
1152   />
1153   </Form.Group>
1154 </Grid.Col>
1155 <Grid.Col sm={12} md={4}>
1156   <Form.Group>
1157     <Form.Label>Religion</Form.Label>
1158     <Form.Input
1159       type="text"
1160       placeholder="Religion"
1161       value={religion}
1162       onChange={this.onChange}
1163       name="religion"
1164       error={errors.religion}>
1165   />
1166   </Form.Group>
1167 </Grid.Col>
1168 </Grid.Row>
1169 </Card.Body>
1170 <Card.Body>
1171   <Card.Title>Emergency Contact Information</Card.
1172     Title>
1173   <Grid.Row>
1174     <Grid.Col sm={12} md={6}>
1175       <Form.Group>
1176         <Form.Label>Contact Person</Form.Label>
1177         <Form.Input
1178           type="text"
1179           placeholder="Contact Person"
1180           value={emergencyName}
1181           onChange={this.onChange}
1182           name="emergencyName"
1183           error={errors.emergencyName}>
1184       />
1185     </Form.Group>
1186   </Grid.Col>
1187   <Grid.Col sm={12} md={6}>
1188     <Form.Group>
1189       <Form.Label>Contact Address</Form.Label>
1190       <Form.Input
1191         type="text"
1192         placeholder="Contact Address"
1193         value={emergencyAddress}
1194         onChange={this.onChange}
1195         name="emergencyAddress"
1196         error={errors.emergencyAddress}>
1197     />
1198   </Form.Group>
1199   </Grid.Col>
1200   <Grid.Col sm={12} md={6}>
1201     <Form.Group>
1202       <Form.Label>Contact Telephone No.</Form.Label
1203         >
1204       <Form.Input
1205         type="text"
1206         placeholder="Contact Telephone No."
1207         value={emergencyTelephone}
1208         onChange={this.onChange}
1209         name="emergencyTelephone"
1210         error={errors.emergencyTelephone}>
1211     />
1212   </Form.Group>
1213   </Grid.Col>
1214   <Grid.Col sm={12} md={6}>
1215     <Form.Group>

```

```

1214         <Form.Label>Contact Cellphone No.</Form.Label>
1215             >
1216             <Form.Input
1217                 type="text"
1218                 placeholder="Contact Cellphone No."
1219                 value={emergencyCellphone}
1220                 onChange={this.onChange}
1221                 name="emergencyCellphone"
1222                 error={errors.emergencyCellphone}>
1223             />
1224         </Form.Group>
1225     </Grid.Col>
1226     <Grid.Col sm={12} md={6}>
1227         <Form.Group>
1228             <Form.Label>Contact Email</Form.Label>
1229             <Form.Input
1230                 type="text"
1231                 placeholder="Contact Email"
1232                 value={emergencyEmail}
1233                 onChange={this.onChange}
1234                 name="emergencyEmail"
1235                 error={errors.emergencyEmail}>
1236             />
1237         </Form.Group>
1238     </Grid.Col>
1239     <Grid.Col sm={12} md={6}>
1240         <Form.Group>
1241             <Form.Label>Contact Relationship</Form.Label>
1242             <Form.Input
1243                 type="text"
1244                 placeholder="Contact Relationship"
1245                 value={emergencyRelationship}
1246                 onChange={this.onChange}
1247                 name="emergencyRelationship"
1248                 error={errors.emergencyRelationship}>
1249             />
1250         </Form.Group>
1251     </Grid.Row>
1252 </Card.Body>
1253 <Card.Footer className="text-right">
1254     <Button type="submit" color="primary">
1255         Update Profile
1256     </Button>
1257 </Card.Footer>
1258 </Form>
1259 </Grid.Col>
1260 </Grid.Row>
1261 </Container>
1262     )}>
1263     </div>
1264   );
1265 }
1266 }
1267 /* istanbul ignore next */
1268 function mapStateToProps(state) {
1269   return {
1270     app: state.app,
1271   };
1272 }
1273
1274 /* istanbul ignore next */
1275 function mapDispatchToProps(dispatch) {
1276   return {
1277     actions: bindActionCreators({ ...actions }, dispatch),
1278   };
1279 }
1280
1281
1282 export default connect(mapStateToProps, mapDispatchToProps)(
1283   AdminProfile);
1284 import React, { Component } from 'react';
1285 import PropTypes from 'prop-types';

```

```

1285 import { bindActionCreators } from 'redux';
1286 import { connect } from 'react-redux';
1287 import * as actions from './redux/actions';
1288 import { Card, Button, Grid, Avatar, Table, Form } from 'tabler-react';
1289 import { Spin, Pagination } from 'antd';
1290 import moment from 'moment';
1291 import axios from 'axios';
1292
1293 export class AdminViewUpdateLog extends Component {
1294   static propTypes = {
1295     app: PropTypes.object.isRequired,
1296     actions: PropTypes.object.isRequired,
1297   };
1298
1299   constructor(props) {
1300     super(props);
1301     this.state = {
1302       isLoading: false,
1303       page: 1,
1304       pageSize: 5,
1305       numOfPages: 1,
1306       data: [],
1307     };
1308   }
1309
1310   componentWillMount() {
1311     this.setState({ isLoading: true });
1312     axios
1313       .post('api/admin/getupdatelog', { page: 1, pageSize: 5 })
1314       .then(res => {
1315         this.setState({ isLoading: false });
1316         this.setState({
1317           numOfPages: res.data.numOfPages,
1318           data: res.data.updateLog,
1319         });
1320       })
1321       .catch(err => {
1322         this.setState({ isLoading: false });
1323       });
1324   }
1325
1326   paginate = page => {
1327     this.setState({
1328       page,
1329     });
1330     this.setState({ isLoading: true });
1331     axios
1332       .post('api/admin/getupdatelog', { page, pageSize: this.state.
1333         pageSize })
1334       .then(res => {
1335         this.setState({ isLoading: false });
1336         this.setState({
1337           numOfPages: res.data.numOfPages,
1338           data: res.data.updateLog,
1339         });
1340       })
1341       .catch(err => {
1342         this.setState({ isLoading: false });
1343       });
1344   };
1345
1346   render() {
1347     const DisplayData = [];
1348     for (const [index, value] of this.state.data.entries()) {
1349       DisplayData.push(
1350         <Table.Row>
1351           <Table.Col>{moment(value.dateUpdated).format('MMM d YYYY hh:
1352             mm:ss')}</Table.Col>
1353           <Table.Col>{value.teacherName}</Table.Col>
1354           <Table.Col>{value.studentName}</Table.Col>
1355           <Table.Col>{value.sectionName}</Table.Col>
1356           <Table.Col>{value.subjectName}</Table.Col>
1357           <Table.Col>
```

```

1356         {value.teacherName != '' && (
1357             <Button icon="info" size="sm" pill color="info">
1358                 Details
1359             </Button>
1360         )}
1361     </Table.Col>
1362   </Table.Row>,
1363 );
1364 }
1365 return (
1366   <div className="app-admin-view-update-log card">
1367     <Card.Body>
1368       <Card.Title>Grades Update Log</Card.Title>
1369       <Grid.Row>
1370         <Grid.Col sm={12} md={12} xs={12}>
1371           <Spin spinning={this.state.isLoading}>
1372             <Table highlightRowOnHover={true} responsive={true}>
1373               <Table.Header>
1374                 <Table.ColHeader>Date</Table.ColHeader>
1375                 <Table.ColHeader>Teacher</Table.ColHeader>
1376                 <Table.ColHeader>Student</Table.ColHeader>
1377                 <Table.ColHeader>Section</Table.ColHeader>
1378                 <Table.ColHeader>Subject</Table.ColHeader>
1379                 <Table.ColHeader>Details</Table.ColHeader>
1380               </Table.Header>
1381               <Table.Body>{DisplayData}</Table.Body>
1382             </Table>
1383             <Pagination
1384               size="large"
1385               current={this.state.page}
1386               pageSize={this.state.pageSize}
1387               total={this.state.pageSize * this.state.numOfPages}
1388               onChange={this.paginate}
1389             />
1390           </Spin>
1391         </Grid.Col>
1392       </Grid.Row>
1393     </Card.Body>
1394   </div>
1395 );
1396 }
1397 }
1398 */
1399 /* istanbul ignore next */
1400 function mapStateToProps(state) {
1401   return {
1402     app: state.app,
1403   };
1404 }
1405 */
1406 /* istanbul ignore next */
1407 function mapDispatchToProps(dispatch) {
1408   return {
1409     actions: bindActionCreators({ ...actions }, dispatch),
1410   };
1411 }
1412 */
1413 export default connect(
1414   mapStateToProps,
1415   mapDispatchToProps,
1416 )(AdminViewUpdateLog);
1417 import React, { Component } from 'react';
1418 import PropTypes from 'prop-types';
1419 import { bindActionCreators } from 'redux';
1420 import { connect } from 'react-redux';
1421 import * as actions from './redux/actions';
1422 import { Link } from 'react-router-dom';
1423 import { Card, Button, Grid, Table, Container } from 'tabler-react';
1424 import axios from 'axios';
1425 import { Pagination, Spin, Tooltip, Modal, Typography } from 'antd';
1426 */
1427 export class AllStudentDeliberationGrades extends Component {

```

```

1428     static propTypes = {
1429       app: PropTypes.object.isRequired,
1430       actions: PropTypes.object.isRequired,
1431     };
1432
1433     constructor(props) {
1434       super(props);
1435       this.state = {
1436         showModal: false,
1437         isLoading: false,
1438         data: [],
1439       };
1440     }
1441
1442     handleViewGrades() {
1443       this.setState({ isLoading: true }, () => {
1444         axios
1445           .post('api/registrar/studentdeliberationrecord', { studsectID:
1446             this.props.studsectID })
1447           .then(res => {
1448             this.setState({ isLoading: false, data: res.data.data });
1449           });
1450     }
1451
1452     render() {
1453       const DisplayData = [];
1454       const classStanding = () => {
1455         let enumerator = 0,
1456           denom = 0;
1457         for (const [index, value] of this.state.data.entries()) {
1458           if (value.score != -1) {
1459             denom = denom + 1;
1460             enumerator = enumerator + parseFloat(value.score);
1461           }
1462         }
1463         if (denom != 0) {
1464           return parseFloat(enumerator / denom);
1465         } else return 'Not yet available';
1466       };
1467       for (const [index, value] of this.state.data.entries()) {
1468         DisplayData.push(
1469           <Table.Row>
1470             <Table.Col>{value.subjectName}</Table.Col>
1471             <Table.Col>{value.score == -1 ? 'Not yet available' : value.
1472               score}</Table.Col>
1473           </Table.Row>,
1474         );
1475       }
1476       return (
1477         <React.Fragment>
1478           <Modal
1479             title="Student Grades"
1480             visible={this.state.showModal}
1481             footer={[<Button onClick={() => this.setState({ showModal:
1482               false })}>Close</Button>]}
1483             onCancel={() => {
1484               this.setState({ showModal: false });
1485             }}
1486           >
1487             <Spin spinning={this.state.isLoading}>
1488               <Card statusColor="success">
1489                 <Card.Body>
1490                   <Card.Title>Current Student Record of {this.props.name
1491                     }</Card.Title>
1492                   <Container>
1493                     <Grid.Row>
1494                       <Grid.Col sm={12} xs={12} md={12}>
1495                         <Table highlightRowOnHover={true} responsive={
1496                           true}>
1497                           <Table.Header>
1498                             <Table.ColHeader>Subject Name</Table.
1499                             ColHeader>

```

```

1495             <Table.ColHeader>Grade</Table.ColHeader>
1496         </Table.Header>
1497         <Table.Body>{DisplayData}</Table.Body>
1498     </Table>
1499 </Grid.Col>
1500 <Grid.Col sm={12} xs={12} md={12}>
1501     <Typography.Text style={{ fontSize: '18px' }}>
1502         Current Class Standing (Current Quarter): <b>{
1503             classStanding()</b>
1504         </Typography.Text>
1505     </Grid.Col>
1506 </Grid.Row>
1507 </Container>
1508 </Card.Body>
1509 </Card>
1510 </Spin>
1511 </Modal>
1512 <Tooltip placement="top" title="View All Student Grades">
1513     <Button
1514         icon="file"
1515         size="sm"
1516         color="primary"
1517         outline
1518         pill
1519         onClick={() => this.setState({ showModal: true }), () =>
1520             this.handleViewGrades()}></Button>
1521     </Tooltip>
1522     <React.Fragment>
1523     );
1524 }
1525 /* istanbul ignore next */
1526 function mapStateToProps(state) {
1527     return {
1528         app: state.app,
1529     };
1530 }
1531 /* istanbul ignore next */
1532 function mapDispatchToProps(dispatch) {
1533     return {
1534         actions: bindActionCreators({ ...actions }, dispatch),
1535     };
1536 }
1537
1538
1539 export default connect(mapStateToProps, mapDispatchToProps)(
1540     AllStudentDeliberationGrades);
1541 import React, { Component } from 'react';
1542 import PropTypes from 'prop-types';
1543 import { bindActionCreators } from 'redux';
1544 import { connect } from 'react-redux';
1545 import * as actions from './redux/actions';
1546 import { Link } from 'react-router-dom';
1547 import { Card, Button, Grid, Avatar, Table, Form, Header, Container,
1548         Text } from 'tabler-react';
1549 import axios from 'axios';
1550 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
1551 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
1552         message } from 'antd';
1553 import cn from 'classnames';
1554 import placeholder from '../../../../../images/placeholder.jpg';
1555 import bg from '../../../../../images/BG.png';
1556 import { getImageUrl } from '../../../../../utils';
1557 const { Option } = AutoComplete;
1558
1559 export class AllStudentFinalGrades extends Component {
1560     static propTypes = {
1561         app: PropTypes.object.isRequired,
1562         actions: PropTypes.object.isRequired,
1563     };

```

```

1562
1563     constructor(props) {
1564         super(props);
1565         this.state = {
1566             showModal: false,
1567             isLoading: false,
1568             data: [],
1569             finalGrade: -1,
1570         };
1571     }
1572
1573     handleShowGrades = () => {
1574         this.setState({ isLoading: true }, () => {
1575             axios
1576                 .post('api/registrar/studentfinalgrade', {
1577                     studentID: this.props.id,
1578                     schoolYearID: this.props.schoolYearID,
1579                 })
1580                 .then(res => {
1581                     this.setState({
1582                         data: res.data.data,
1583                         finalGrade: res.data.finalGrade,
1584                         isLoading: false,
1585                     });
1586                 });
1587             });
1588         );
1589     }
1590     render() {
1591         const displayQuarter = quarter =>
1592             quarter == 'Q1'
1593                 ? '1st Quarter'
1594                 : quarter == 'Q2'
1595                     ? '2nd Quarter'
1596                     : quarter == 'Q3'
1597                         ? '3rd Quarter'
1598                         : '4th Quarter';
1599         const DisplayData = [];
1600         for (const [index, value] of this.state.data.entries()) {
1601             DisplayData.push(
1602                 <Table.Row>
1603                     <Table.Col alignContent={'center'}>{displayQuarter(value.
1604                         quarter)}</Table.Col>
1605                     <Table.Col alignContent={'center'}>
1606                         {value.grade == -1 ? 'Not yet available' : value.grade}
1607                     </Table.Col>
1608                 </Table.Row>,
1609             );
1610         }
1611         return (
1612             <React.Fragment>
1613                 <Modal
1614                     title="Student Grades"
1615                     onCancel={() => this.setState({ showModal: false })}
1616                     footer={[<Button onClick={() => this.setState({ showModal:
1617                         false })}>Close</Button>]}
1618                     visible={this.state.showModal}
1619                 >
1620                     <Spin spinning={this.state.isLoading}>
1621                         <Grid.Row>
1622                             <Grid.Col sm={12} xs={12} md={12}>
1623                                 <Card statusColor={'success'}>
1624                                     <Card.Body>
1625                                         <Card.Title>Student Final Grade for S.Y. {this.
1626                                             props.schoolYear}</Card.Title>
1627                                         <Table highlightRowOnHover={true} responsive={true
1628                                             }>
1629                                             <Table.Header>
1630                                                 <Table.ColHeader alignContent={'center'}>
1631                                                     Quarter</Table.ColHeader>
1632                                                 <Table.ColHeader alignContent={'center'}>Grade
1633                                                     </Table.ColHeader>
1634                                             </Table.Header>

```

```

1629             <Table.Body>
1630                 {DisplayData.length == 0 ? (
1631                     <Table.Row>
1632                         <Table.Col colSpan={2} alignContent={'center'}>
1633                             No entries.
1634                         </Table.Col>
1635                     </Table.Row>
1636                 ) : (
1637                     DisplayData
1638                 )}
1639             </Table.Body>
1640         </Table>
1641     </Card.Body>
1642     <Card.Footer>
1643         <Text style={{ fontSize: '18px' }}>
1644             Final Grade for S.Y. {this.props.schoolYear}:{' '}
1645             <br>
1646             {this.state.finalGrade == -1 ? 'Not yet
1647                 available' : this.state.finalGrade}
1648             </b>
1649         </Text>
1650     </Card.Footer>
1651 </Grid.Col>
1652 </Grid.Row>
1653 </Spin>
1654 </Modal>
1655 <Button
1656     icon="eye"
1657     color="success"
1658     onClick={() =>
1659         this.setState({ showModal: true }, () => {
1660             this.handleShowGrades();
1661         })
1662     }
1663     >
1664     {this.props.text}
1665     </Button>
1666 </React.Fragment>
1667 );
1668 }
1669 }
1670 */
1671 /* istanbul ignore next */
1672 function mapStateToProps(state) {
1673     return {
1674         app: state.app,
1675     };
1676 }
1677 */
1678 /* istanbul ignore next */
1679 function mapDispatchToProps(dispatch) {
1680     return {
1681         actions: bindActionCreators({ ...actions }, dispatch),
1682     };
1683 }
1684 */
1685 export default connect(mapStateToProps, mapDispatchToProps)(
1686     AllStudentFinalGrades);
1687 import React, { Component } from 'react';
1688 import { bindActionCreators } from 'redux';
1689 import * as actions from './redux/actions';
1690 import PropTypes from 'prop-types';
1691 import { Spin } from 'antd';
1692 import { connect } from 'react-redux';
1693 */
1694     This is the root component of your app. Here you define the overall
1695     layout
1696     and the container of the react router.

```

```

1696     You should adjust it according to the requirement of your app.
1697 */
1698 export class App extends Component {
1699   static propTypes = {
1700     children: PropTypes.node,
1701   };
1702
1703   static defaultProps = {
1704     children: '',
1705   };
1706
1707   render() {
1708     const { showLoading } = this.props;
1709     return (
1710       <div className="app-app">
1711         <div className="page-container">
1712           <Spin spinning={showLoading} size="large">
1713             {this.props.children}
1714           </Spin>
1715         </div>
1716       </div>
1717     );
1718   }
1719 }
1720
1721 /* istanbul ignore next */
1722 function mapStateToProps(state) {
1723   return {
1724     showLoading: state.app.showLoading,
1725   };
1726 }
1727
1728 /* istanbul ignore next */
1729 function mapDispatchToProps(dispatch) {
1730   return {
1731     actions: bindActionCreators({ ...actions }, dispatch),
1732   };
1733 }
1734
1735 export default connect(
1736   mapStateToProps,
1737   mapDispatchToProps,
1738 )(App);
1739 import React, { Component } from 'react';
1740 import PropTypes from 'prop-types';
1741 import { bindActionCreators } from 'redux';
1742 import { connect } from 'react-redux';
1743 import * as actions from './redux/actions';
1744 import { Container, Grid, Card, Button, Form, Header } from 'tabler-react';
1745 import cn from 'classnames';
1746 import placeholder from '../..../images/placeholder.jpg';
1747 import { Spin } from 'antd';
1748 import bg from '../..../images/BG.png';
1749 import { getImageUrl } from '../..../utils';
1750 import RestrictAccount from './RestrictAccount';
1751 function ProfileImage({ avatarURL }) {
1752   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
1753 }
1754
1755 function Profile({
1756   className,
1757   children,
1758   name,
1759   avatarURL = '',
1760   twitterURL = '',
1761   backgroundURL = '',
1762   bio,
1763 }) {
1764   const classes = cn('card-profile', className);
1765   return (
1766     <Card className={classes}>

```

```

1767         <Card.Header backgroundURL={backgroundURL} />
1768         <Card.Body className="text-center">
1769             <ProfileImage avatarURL={avatarURL} />
1770             <Header.H3 className="mb-3">{name}</Header.H3>
1771             <p className="mb-4">{bio || children}</p>
1772         </Card.Body>
1773     </Card>
1774   );
1775 }
1776
1777 export class CashierDashboard extends Component {
1778   static propTypes = {
1779     app: PropTypes.object.isRequired,
1780     actions: PropTypes.object.isRequired,
1781   };
1782
1783   constructor(props) {
1784     super(props);
1785     this.state = {
1786       isLoading: false,
1787       email: '',
1788       position: '',
1789       firstName: '',
1790       lastName: '',
1791       middleName: '',
1792       suffix: '',
1793       nickname: '',
1794       imageUrl: '',
1795       contactNum: '',
1796       address: '',
1797       province: '',
1798       city: '',
1799       region: '',
1800       zipcode: '',
1801       civilStatus: '',
1802       sex: '',
1803       citizenship: '',
1804       birthDate: 0,
1805       birthPlace: '',
1806       religion: '',
1807       emergencyName: '',
1808       emergencyAddress: '',
1809       emergencyTelephone: '',
1810       emergencyCellphone: '',
1811       emergencyEmail: '',
1812       emergencyRelationship: '',
1813     };
1814   }
1815
1816   componentWillReceiveProps(nextProps) {
1817     this.setState({
1818       email: nextProps.app.auth.user.email,
1819       position: nextProps.app.auth.user.position,
1820       firstName: nextProps.app.profile.firstName,
1821       lastName: nextProps.app.profile.lastName,
1822       middleName: nextProps.app.profile.middleName,
1823       suffix: nextProps.app.profile.suffix,
1824       nickname: nextProps.app.profile.nickname,
1825       imageUrl: nextProps.app.profile.imageUrl,
1826       contactNum: nextProps.app.profile.contactNum,
1827       address: nextProps.app.profile.address,
1828       province: nextProps.app.profile.province,
1829       city: nextProps.app.profile.city,
1830       region: nextProps.app.profile.region,
1831       zipcode: nextProps.app.profile.zipcode,
1832       civilStatus: nextProps.app.profile.civilStatus,
1833       sex: nextProps.app.profile.sex,
1834       citizenship: nextProps.app.profile.citizenship,
1835       birthDate: nextProps.app.profile.birthDate,
1836       birthPlace: nextProps.app.profile.birthPlace,
1837       religion: nextProps.app.profile.religion,
1838       emergencyName: nextProps.app.profile.emergencyName,
1839       emergencyAddress: nextProps.app.profile.emergencyAddress,
1840       emergencyTelephone: nextProps.app.profile.emergencyTelephone,
1841       emergencyCellphone: nextProps.app.profile.emergencyCellphone,
1842       emergencyEmail: nextProps.app.profile.emergencyEmail,
1843       emergencyRelationship: nextProps.app.profile.

```

```

        emergencyRelationship ,
1844    });
1845 }
1846
1847 componentDidMount() {
1848   if (Object.keys(this.props.app.profile).length !== 0) {
1849     this.setState({
1850       email: this.props.app.auth.user.email,
1851       position: this.props.app.auth.user.position,
1852       firstName: this.props.app.profile.firstName,
1853       lastName: this.props.app.profile.lastName,
1854       middleName: this.props.app.profile.middleName,
1855       suffix: this.props.app.profile.suffix,
1856       nickname: this.props.app.profile.nickname,
1857       imageUrl: this.props.app.profile.imageUrl,
1858       contactNum: this.props.app.profile.contactNum,
1859       address: this.props.app.profile.address,
1860       province: this.props.app.profile.province,
1861       city: this.props.app.profile.city,
1862       region: this.props.app.profile.region,
1863       zipcode: this.props.app.profile.zipcode,
1864       civilStatus: this.props.app.profile.civilStatus,
1865       sex: this.props.app.profile.sex,
1866       citizenship: this.props.app.profile.citizenship,
1867       birthDate: this.props.app.profile.birthDate,
1868       birthPlace: this.props.app.profile.birthPlace,
1869       religion: this.props.app.profile.religion,
1870       emergencyName: this.props.app.profile.emergencyName,
1871       emergencyAddress: this.props.app.profile.emergencyAddress,
1872       emergencyTelephone: this.props.app.profile.emergencyTelephone,
1873       emergencyCellphone: this.props.app.profile.emergencyCellphone,
1874       emergencyEmail: this.props.app.profile.emergencyEmail,
1875       emergencyRelationship: this.props.app.profile.
1876         emergencyRelationship,
1877     });
1878   }
1879 }
1880
1881 render() {
1882   const {
1883     email,
1884     position,
1885     firstName,
1886     lastName,
1887     middleName,
1888     suffix,
1889     nickname,
1890     imageUrl,
1891     contactNum,
1892     address,
1893     province,
1894     city,
1895     region,
1896     zipcode,
1897     civilStatus,
1898     sex,
1899     citizenship,
1900     birthPlace,
1901     religion,
1902     emergencyName,
1903     emergencyAddress,
1904     emergencyTelephone,
1905     emergencyCellphone,
1906     emergencyEmail,
1907     emergencyRelationship,
1908   } = this.state;
1909   const birthDate = new Date(this.state.birthDate);
1910   const { errors } = this.props.app;
1911   const uploadLoading = false;
1912   const displayPosition = position => {
1913     switch (position) {
1914       case false:
1915         return 'Administrator';
1916       case true:
1917         return 'Director';
1918       case 2:
1919         return 'Registrar';

```

```

1919         case 3:
1920             return 'Teacher';
1921         case 4:
1922             return 'Student';
1923         case 5:
1924             return 'Guardian';
1925         case 6:
1926             return 'Cashier';
1927         default:
1928             return '';
1929     }
1930 };
1931
1932 const getPHTime = (date = '') => {
1933     const d = date === '' ? new Date() : new Date(date);
1934     const localOffset = d.getTimezoneOffset() * 60000;
1935     const UTC = d.getTime() + localOffset;
1936     const PHT = UTC + 3600000 * 16;
1937     return new Date(PHT);
1938 };
1939 const capitalize = string => {
1940     return string.charAt(0).toUpperCase() + string.slice(1).
1941         toLowerCase();
1942 };
1943 return (
1944     <div className="app-cashier-dashboard my-3 my-md-5">
1945         <Container>
1946             <Grid.Row>
1947                 <Grid.Col xs={12} md={12} md={5}>
1948                     <Profile
1949                         name={`${firstName} ${middleName.charAt(0).toUpperCase()
1950                             ()}. ${lastName}`}
1951                         avatarUrl={imageUrl === 'NA' ? placeholder :
1952                             getImageUrl(imageUrl)}
1953                         backgroundURL={bg}
1954                     >
1955                         <div>
1956                             <Header.H5>{displayPosition(position)}</Header.H5>
1957                         </div>
1958                     </Profile>
1959                 </Grid.Col>
1960                 <Grid.Col xs={12} md={12} md={7}>
1961                     <RestrictAccount />
1962                 </Grid.Col>
1963             </Grid.Row>
1964         </Container>
1965     </div>
1966 );
1967 }
1968 /* istanbul ignore next */
1969 function mapStateToProps(state) {
1970     return {
1971         app: state.app,
1972     };
1973 }
1974 /* istanbul ignore next */
1975 function mapDispatchToProps(dispatch) {
1976     return {
1977         actions: bindActionCreators({ ...actions }, dispatch),
1978     };
1979 }
1980
1981 export default connect(mapStateToProps, mapDispatchToProps)(
1982     CashierDashboard);
1983 import React, { Component } from 'react';
1984 import PropTypes from 'prop-types';
1985 import { bindActionCreators } from 'redux';
1986 import { connect } from 'react-redux';
1987 import * as actions from './redux/actions';
import moment from 'moment';

```

```

1988 import axios from 'axios';
1989 import { Container, Grid, Card, Button, Form, Header, List } from 'tabler-react';
1990 import { Alert, Upload, message } from 'antd';
1991 import cn from 'classnames';
1992 import placeholder from '../images/placeholder.jpg';
1993 import bg from '../images/BG.png';
1994 import { getImageUrl } from '../utils';
1995 function ProfileImage({ avatarURL }) {
1996   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
1997 }
1998
1999 function Profile({
2000   className,
2001   children,
2002   name,
2003   avatarURL = '',
2004   twitterURL = '',
2005   backgroundURL = '',
2006   bio,
2007 }) {
2008   const classes = cn('card-profile', className);
2009   return (
2010     <Card className={classes}>
2011       <Card.Header backgroundURL={backgroundURL} />
2012       <Card.Body className="text-center">
2013         <ProfileImage avatarURL={avatarURL} />
2014         <Header.H3 className="mb-3">{name}</Header.H3>
2015         <p className="mb-4">{bio || children}</p>
2016       </Card.Body>
2017     </Card>
2018   );
2019 }
2020 export class CashierProfile extends Component {
2021   static propTypes = {
2022     app: PropTypes.object.isRequired,
2023     actions: PropTypes.object.isRequired,
2024   };
2025
2026   constructor(props) {
2027     super(props);
2028     this.state = {
2029       loading: true,
2030       email: '',
2031       position: '',
2032       firstName: '',
2033       lastName: '',
2034       middleName: '',
2035       suffix: '',
2036       nickname: '',
2037       imageUrl: '',
2038       contactNum: '',
2039       address: '',
2040       province: '',
2041       city: '',
2042       region: '',
2043       zipcode: '',
2044       civilStatus: '',
2045       sex: '',
2046       citizenship: '',
2047       birthDate: new Date(),
2048       birthPlace: '',
2049       religion: '',
2050       emergencyName: '',
2051       emergencyAddress: '',
2052       emergencyTelephone: '',
2053       emergencyCellphone: '',
2054       emergencyEmail: '',
2055       emergencyRelationship: '',
2056     };
2057
2058     this.onSubmit = this.onSubmit.bind(this);
2059     this.onChange = this.onChange.bind(this);
2060     this.onDateChange = this.onDateChange.bind(this);

```

```

2061     }
2062     componentWillReceiveProps(nextProps) {
2063       if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
2064         .length == 0)
2065         this.setState({
2066           email: nextProps.app.auth.user.email,
2067           position: nextProps.app.auth.user.position,
2068           firstName: nextProps.app.profile.firstName,
2069           lastName: nextProps.app.profile.lastName,
2070           middleName: nextProps.app.profile.middleName,
2071           suffix: nextProps.app.profile.suffix,
2072           nickname: nextProps.app.profile.nickname,
2073           imageUrl: nextProps.app.profile.imageUrl,
2074           contactNum: nextProps.app.profile.contactNum,
2075           address: nextProps.app.profile.address,
2076           province: nextProps.app.profile.province,
2077           city: nextProps.app.profile.city,
2078           region: nextProps.app.profile.region,
2079           zipcode: nextProps.app.profile.zipcode,
2080           civilStatus: nextProps.app.profile.civilStatus,
2081           sex: nextProps.app.profile.sex,
2082           citizenship: nextProps.app.profile.citizenship,
2083           birthDate: nextProps.app.profile.birthDate,
2084           birthPlace: nextProps.app.profile.birthPlace,
2085           religion: nextProps.app.profile.religion,
2086           emergencyName: nextProps.app.profile.emergencyName,
2087           emergencyAddress: nextProps.app.profile.emergencyAddress,
2088           emergencyTelephone: nextProps.app.profile.emergencyTelephone,
2089           emergencyCellphone: nextProps.app.profile.emergencyCellphone,
2090           emergencyEmail: nextProps.app.profile.emergencyEmail,
2091           emergencyRelationship: nextProps.app.profile.
2092             emergencyRelationship,
2093             loading: false,
2094         });
2095     }
2096     onChange(event) {
2097       this.setState({ [event.target.name]: event.target.value });
2098     }
2099     onDateChange(event) {
2100       this.setState({ birthDate: event });
2101     }
2102     onSubmit(event) {
2103       event.preventDefault();
2104       const {
2105         firstName,
2106         lastName,
2107         middleName,
2108         suffix,
2109         nickname,
2110         contactNum,
2111         address,
2112         province,
2113         city,
2114         region,
2115         zipcode,
2116         civilStatus,
2117         sex,
2118         citizenship,
2119         birthDate,
2120         birthPlace,
2121         religion,
2122         emergencyName,
2123         emergencyAddress,
2124         emergencyTelephone,
2125         emergencyCellphone,
2126         emergencyEmail,
2127         emergencyRelationship,
2128       } = this.state;
2129     }
2130     const profileData = {
2131       firstName,
2132       lastName,
2133       middleName,
2134       suffix,
2135       nickname,

```

```

2136     contactNum ,
2137     address ,
2138     province ,
2139     city ,
2140     region ,
2141     zipcode ,
2142     civilStatus ,
2143     sex ,
2144     citizenship ,
2145     birthDate ,
2146     birthPlace ,
2147     religion ,
2148     emergencyName ,
2149     emergencyAddress ,
2150     emergencyTelephone ,
2151     emergencyCellphone ,
2152     emergencyEmail ,
2153     emergencyRelationship ,
2154   };
2155   console.log(profileData);
2156   this.props.actions.updateCashierProfile(profileData);
2157 }
2158
2159 render() {
2160   const {
2161     email ,
2162     position ,
2163     firstName ,
2164     lastName ,
2165     middleName ,
2166     suffix ,
2167     nickname ,
2168     imageUrl ,
2169     contactNum ,
2170     address ,
2171     province ,
2172     city ,
2173     region ,
2174     zipcode ,
2175     civilStatus ,
2176     sex ,
2177     citizenship ,
2178     birthPlace ,
2179     religion ,
2180     emergencyName ,
2181     emergencyAddress ,
2182     emergencyTelephone ,
2183     emergencyCellphone ,
2184     emergencyEmail ,
2185     emergencyRelationship ,
2186   } = this.state;
2187   const birthDate = new Date(this.state.birthDate);
2188   const defaultDate = birthDate.toISOString();
2189   const { errors } = this.props.app;
2190   const uploadLoading = false;
2191   const displayPosition = position => {
2192     switch (position) {
2193       case false:
2194         return 'Administrator';
2195       case true:
2196         return 'Director';
2197       case 2:
2198         return 'Registrar';
2199       case 3:
2200         return 'Teacher';
2201       case 4:
2202         return 'Student';
2203       case 5:
2204         return 'Guardian';
2205       default:
2206         return '';
2207     }
2208   };
2209   const capitalize = string => {
2210     return string.charAt(0).toUpperCase() + string.slice(1).
2211     toLowerCase();
2211   };

```

```

2212     return (
2213       <div className="app-admin-profile my-3 my-md-5">
2214         {this.state.loading ? (
2215           ,
2216         ) : (
2217           <Container>
2218             <Grid.Row>
2219               <Grid.Col sm={12} lg={4}>
2220                 <Profile
2221                   name={`${firstName} ${middleName.charAt(0).
2222                     toUpperCase()} ${lastName}`}
2223                     avatarURL={imageUrl == 'NA' ? placeholder :
2224                       getImageUrl(imageUrl)}
2225                     backgroundURL={bg}
2226                   >
2227                     <Upload
2228                       name="file"
2229                         multiple={false}
2230                           accept=".png, .jpg"
2231                           customRequest={({req, uploadLoading) => {
2232                             // Axios Photo Upload
2233                             this.props.actions.setLoading(true);
2234                             const bodyFormData = new FormData();
2235                             bodyFormData.set('file', req.file);
2236                             axios({
2237                               method: 'post',
2238                                 url: 'api/users/upload',
2239                                   data: bodyFormData,
2240                                 })
2241                               .then(res => {
2242                                 req.onSuccess();
2243                                   this.props.actions.setLoading(false);
2244                                     message.success('Uploaded Successfully!').
2245                                       then(() => {
2246                                         window.location.href = '/profile';
2247                                       });
2248                                     })
2249                                     .catch(err => {
2250                                       req.onError();
2251                                         this.props.actions.setLoading(false);
2252                                           message.error('Upload failed!');
2253                                         });
2254                                       })
2255                                     >
2256                                       <Header.H4>{displayPosition(position)}</Header.H4
2257                                     >
2258                                       <Button loading={this.props.app.showLoading} icon="
2259                                         upload" pill color="primary">
2260                                           Upload Image
2261                                         </Button>
2262                                         </Upload>
2263                                       </Profile>
2264                                     </Grid.Col>
2265                                     <Grid.Col xs={12} sm={12} lg={8}>
2266                                       <Form className="card" onSubmit={this.onSubmit}>
2267                                         <Card.Body>
2268                                           <Card.Title>Edit Profile</Card.Title>
2269                                         <Grid.Row>
2270                                           <Grid.Col xs={12} sm={12} md={6}>
2271                                             <Form.Group>
2272                                               <Form.Label>Email</Form.Label>
2273                                                 <Form.Input
2274                                                   type="email"
2275                                                     disabled
2276                                                       placeholder="Email"
2277                                                         value={email}
2278                                                       error={errors.email}
2279                                                     />
2280                                                 </Form.Group>
2281                                               </Grid.Col>
2282                                             <Grid.Col sm={12} md={6}>

```

```

2280
2281     <Form.Group>
2282         <Form.Label>Position</Form.Label>
2283         <Form.Input
2284             type="text"
2285             placeholder="Position"
2286             disabled
2287             value={displayPosition(position)}
2288             error={errors.position}
2289         />
2290     </Form.Group>
2291 </Grid.Col>
2292 <Grid.Col sm={12} md={4}>
2293     <Form.Group>
2294         <Form.Label>First Name</Form.Label>
2295         <Form.Input
2296             type="text"
2297             name="firstName"
2298             placeholder="First Name"
2299             value={firstName}
2300             onChange={this.onChange}
2301             error={errors.firstName}
2302         />
2303     </Form.Group>
2304 </Grid.Col>
2305 <Grid.Col sm={12} md={4}>
2306     <Form.Group>
2307         <Form.Label>Middle Name</Form.Label>
2308         <Form.Input
2309             type="text"
2310             placeholder="Middle Name"
2311             value={middleName}
2312             onChange={this.onChange}
2313             name="middleName"
2314             error={errors.middleName}
2315         />
2316     </Form.Group>
2317 </Grid.Col>
2318 <Grid.Col sm={12} md={4}>
2319     <Form.Group>
2320         <Form.Label>Last Name</Form.Label>
2321         <Form.Input
2322             type="text"
2323             placeholder="Last Name"
2324             value={lastName}
2325             onChange={this.onChange}
2326             name="lastName"
2327             error={errors.lastName}
2328         />
2329     </Form.Group>
2330 </Grid.Col>
2331 <Grid.Col sm={12} md={3}>
2332     <Form.Group>
2333         <Form.Label>Nickname</Form.Label>
2334         <Form.Input
2335             type="text"
2336             placeholder="Nickname"
2337             value={nickname}
2338             onChange={this.onChange}
2339             name="nickname"
2340             error={errors.nickname}
2341         />
2342     </Form.Group>
2343 </Grid.Col>
2344 <Grid.Col sm={12} md={3}>
2345     <Form.Group>
2346         <Form.Label>Suffix</Form.Label>
2347         <Form.Input
2348             type="text"
2349             placeholder="Suffix"
2350             value={suffix}
2351             onChange={this.onChange}
2352             name="suffix"
2353             error={errors.suffix}

```

```

2353         />
2354     </Form.Group>
2355   </Grid.Col>
2356   <Grid.Col sm={12} md={6}>
2357     <Form.Group>
2358       <Form.Label>Contact Number</Form.Label>
2359       <Form.Input
2360         type="text"
2361         placeholder="Contact Number"
2362         value={contactNum}
2363         onChange={this.onChange}
2364         name="contactNum"
2365         error={errors.contactNum}
2366       />
2367     </Form.Group>
2368   </Grid.Col>
2369   <Grid.Col sm={12} md={12}>
2370     <Form.Group>
2371       <Form.Label>Address</Form.Label>
2372       <Form.Input
2373         type="text"
2374         placeholder="Home Address"
2375         value={address}
2376         onChange={this.onChange}
2377         name="address"
2378         error={errors.address}
2379       />
2380     </Form.Group>
2381   </Grid.Col>
2382   <Grid.Col sm={12} md={4}>
2383     <Form.Group>
2384       <Form.Label>Province</Form.Label>
2385       <Form.Input
2386         type="text"
2387         placeholder="Province"
2388         value={province}
2389         onChange={this.onChange}
2390         name="province"
2391         error={errors.province}
2392       />
2393     </Form.Group>
2394   </Grid.Col>
2395   <Grid.Col sm={12} md={4}>
2396     <Form.Group>
2397       <Form.Label>City</Form.Label>
2398       <Form.Input
2399         type="text"
2400         placeholder="City"
2401         value={city}
2402         onChange={this.onChange}
2403         name="city"
2404         error={errors.city}
2405       />
2406     </Form.Group>
2407   </Grid.Col>
2408   <Grid.Col sm={12} md={2}>
2409     <Form.Group>
2410       <Form.Label>Region</Form.Label>
2411       <Form.Input
2412         type="text"
2413         placeholder="Region"
2414         value={region}
2415         onChange={this.onChange}
2416         name="region"
2417         error={errors.region}
2418       />
2419     </Form.Group>
2420   </Grid.Col>
2421   <Grid.Col sm={12} md={2}>
2422     <Form.Group>
2423       <Form.Label>Zipcode</Form.Label>
2424       <Form.Input
2425         type="text"

```

```

2426           placeholder="Postal Code"
2427           value={zipcode}
2428           onChange={this.onChange}
2429           name="zipcode"
2430           error={errors.zipcode}
2431       />
2432     </Form.Group>
2433   </Grid.Col>
2434   <Grid.Col sm={12} md={4}>
2435     <Form.Group>
2436       <Form.Label>Civil Status</Form.Label>
2437       <Form.Select
2438         value={civilStatus}
2439         onChange={this.onChange}
2440         name="civilStatus"
2441       >
2442         <option>SINGLE</option>
2443         <option>MARRIED</option>
2444         <option>WIDOWED</option>
2445         <option>OTHERS</option>
2446       </Form.Select>
2447     </Form.Group>
2448   </Grid.Col>
2449   <Grid.Col sm={12} md={2}>
2450     <Form.Group>
2451       <Form.Label>Sex</Form.Label>
2452       <Form.Select value={sex} onChange={this.
2453         onChange} name="sex">
2454         <option>M</option>
2455         <option>F</option>
2456       </Form.Select>
2457     </Form.Group>
2458   </Grid.Col>
2459   <Grid.Col sm={12} md={6}>
2460     <Form.Group>
2461       <Form.Label>Citizenship</Form.Label>
2462       <Form.Input
2463         type="text"
2464         placeholder="Citizenship"
2465         value={citizenship}
2466         onChange={this.onChange}
2467         name="citizenship"
2468         error={errors.citizenship}
2469       />
2470     </Form.Group>
2471   </Grid.Col>
2472   <Grid.Col sm={12} md={4}>
2473     <Form.Group>
2474       <Form.Label>Birth Date</Form.Label>
2475       <Form.DatePicker
2476         defaultDate={new Date(defaultDate)}
2477         format="mm/dd/yyyy"
2478         onChange={this.onDateChange}
2479         name="birthDate"
2480         maxYear={2020}
2481         minYear={1897}
2482         monthLabels={[
2483           'January',
2484           'February',
2485           'March',
2486           'April',
2487           'May',
2488           'June',
2489           'July',
2490           'August',
2491           'September',
2492           'October',
2493           'November',
2494           'December',
2495         ]}
2496       ></Form.DatePicker>
2497     </Form.Group>
   </Grid.Col>

```

```

2498 <Grid.Col sm={12} md={4}>
2499   <Form.Group>
2500     <Form.Label>Birth Place</Form.Label>
2501     <Form.Input
2502       type="text"
2503       placeholder="Birth Place"
2504       value={birthPlace}
2505       onChange={this.onChange}
2506       name="birthPlace"
2507       error={errors.birthPlace}
2508     />
2509   </Form.Group>
2510 </Grid.Col>
2511 <Grid.Col sm={12} md={4}>
2512   <Form.Group>
2513     <Form.Label>Religion</Form.Label>
2514     <Form.Input
2515       type="text"
2516       placeholder="Religion"
2517       value={religion}
2518       onChange={this.onChange}
2519       name="religion"
2520       error={errors.religion}
2521     />
2522   </Form.Group>
2523 </Grid.Col>
2524 </Grid.Row>
2525 </Card.Body>
2526 <Card.Body>
2527   <Card.Title>Emergency Contact Information</Card.
2528     Title>
2529   <Grid.Row>
2530     <Grid.Col sm={12} md={6}>
2531       <Form.Group>
2532         <Form.Label>Contact Person</Form.Label>
2533         <Form.Input
2534           type="text"
2535           placeholder="Contact Person"
2536           value={emergencyName}
2537           onChange={this.onChange}
2538           name="emergencyName"
2539           error={errors.emergencyName}
2540         />
2541       </Form.Group>
2542     </Grid.Col>
2543     <Grid.Col sm={12} md={6}>
2544       <Form.Group>
2545         <Form.Label>Contact Address</Form.Label>
2546         <Form.Input
2547           type="text"
2548           placeholder="Contact Address"
2549           value={emergencyAddress}
2550           onChange={this.onChange}
2551           name="emergencyAddress"
2552           error={errors.emergencyAddress}
2553         />
2554       </Form.Group>
2555     </Grid.Col>
2556     <Grid.Col sm={12} md={6}>
2557       <Form.Group>
2558         <Form.Label>Contact Telephone No.</Form.Label
2559         >
2560         <Form.Input
2561           type="text"
2562           placeholder="Contact Telephone No."
2563           value={emergencyTelephone}
2564           onChange={this.onChange}
2565           name="emergencyTelephone"
2566           error={errors.emergencyTelephone}
2567         />
2568       </Form.Group>
2569     </Grid.Col>

```

```

2570 <Form.Label>Contact Cellphone No.</Form.Label>
2571 <Form.Input
2572   type="text"
2573   placeholder="Contact Cellphone No."
2574   value={emergencyCellphone}
2575   onChange={this.onChange}
2576   name="emergencyCellphone"
2577   error={errors.emergencyCellphone}>
2578 />
2579 </Form.Group>
2580 </Grid.Col>
2581 <Grid.Col sm={12} md={6}>
2582 <Form.Group>
2583 <Form.Label>Contact Email</Form.Label>
2584 <Form.Input
2585   type="text"
2586   placeholder="Contact Email"
2587   value={emergencyEmail}
2588   onChange={this.onChange}
2589   name="emergencyEmail"
2590   error={errors.emergencyEmail}>
2591 />
2592 </Form.Group>
2593 </Grid.Col>
2594 <Grid.Col sm={12} md={6}>
2595 <Form.Group>
2596 <Form.Label>Contact Relationship</Form.Label>
2597 <Form.Input
2598   type="text"
2599   placeholder="Contact Relationship"
2600   value={emergencyRelationship}
2601   onChange={this.onChange}
2602   name="emergencyRelationship"
2603   error={errors.emergencyRelationship}>
2604 />
2605 </Form.Group>
2606 </Grid.Col>
2607 </Grid.Row>
2608 </Card.Body>
2609 <Card.Footer className="text-right">
2610   <Button type="submit" color="primary">
2611     Update Profile
2612   </Button>
2613 </Card.Footer>
2614 </Form>
2615 </Grid.Col>
2616 </Grid.Row>
2617 </Container>
2618 )
2619 );
2620 );
2621 }
2622 }
2623 /* istanbul ignore next */
2624 function mapStateToProps(state) {
2625   return {
2626     app: state.app,
2627   };
2628 }
2629 }
2630 /* istanbul ignore next */
2631 function mapDispatchToProps(dispatch) {
2632   return {
2633     actions: bindActionCreators({ ...actions }, dispatch),
2634   };
2635 }
2636 }
2637
2638 export default connect(mapStateToProps, mapDispatchToProps)(
2639   CashierProfile);
2640 import React, { Component } from 'react';
2641 import PropTypes from 'prop-types';

```

```

2641 import { bindActionCreators } from 'redux';
2642 import { connect } from 'react-redux';
2643 import * as actions from './redux/actions';
2644 import { Link } from 'react-router-dom';
2645 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
2646     from 'tabler-react';
2647 import axios from 'axios';
2648 import { Pagination, Spin, Tooltip, Descriptions, Typography } from 'antd';
2649 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
2650     message, Empty } from 'antd';
2651 import cn from 'classnames';
2652 import placeholder from '../images/placeholder.jpg';
2653 import bg from '../images/BG.png';
2654 import { getImageUrl } from '../utils';
2655 const { Option } = AutoComplete;
2656
2657 export class ClassRecordInformation extends Component {
2658     static propTypes = {
2659         app: PropTypes.object.isRequired,
2660         actions: PropTypes.object.isRequired,
2661     };
2662
2663     constructor(props) {
2664         super(props);
2665         this.state = {
2666             isLoading: false,
2667             data: [],
2668             hasData: false,
2669             page: 1,
2670             pageSize: 10,
2671             numOfPages: 1,
2672         };
2673     }
2674
2675     handlePostGrade = () => {
2676         Modal.confirm({
2677             title: 'Post Grades',
2678             content:
2679                 'Do you want to post this class record? Once posted, it will be
2680                 accessed by the students enrolled in the subject.',
2681             okText: 'Post',
2682             cancelText: 'Cancel',
2683             onCancel: () => {},
2684             onOk: () => {
2685                 this.props.actions.postClassRecord(
2686                     {
2687                         classRecordID: this.props.classRecordID,
2688                         quarter: this.props.quarter,
2689                     },
2690                     () => this.props.resetClassRecordInfo(),
2691                 );
2692             },
2693         });
2694     }
2695
2696     handleRevertToDeliberation = () => {
2697         Modal.confirm({
2698             title: 'Revert to Deliberation Status',
2699             content:
2700                 'Do you want to revert the class record status? It will not
2701                 longer be accessed by the students enrolled in the subject.
2702                 ',
2703             okText: 'Revert',
2704             cancelText: 'Cancel',
2705             onCancel: () => {},
2706             onOk: () => {
2707                 this.props.actions.revertClassRecord(
2708                     {
2709                         classRecordID: this.props.classRecordID,
2710                         quarter: this.props.quarter,
2711                     },
2712                     () => this.props.resetClassRecordInfo(),
2713                 );
2714             },
2715         });
2716     }

```

```

2709     },
2710   });
2711 };
2712
2713 componentWillMount(nextProps) {
2714   if (nextProps.classRecordID != -1) {
2715     this.setState({ isLoading: true });
2716     axios
2717       .post('api/registrar/getclassrecinfo', {
2718         classRecordID: nextProps.classRecordID,
2719         quarter: nextProps.quarter,
2720         page: 1,
2721         pageSize: this.state.pageSize,
2722       })
2723       .then(res => {
2724         this.setState({
2725           isLoading: false,
2726           hasData: true,
2727           page: 1,
2728           data: res.data.studentList,
2729           numOfPages: res.data.numOfPages,
2730         });
2731       })
2732       .catch(err => {
2733         this.setState({ isLoading: false, hasData: true, data: [], numOfPages: 1, page: 1 });
2734       });
2735     } else {
2736       this.setState({ hasData: false, data: [], page: 1 });
2737     }
2738   }
2739
2740 paginate = page => {
2741   this.setState({ page, isLoading: true }, () => {
2742     axios
2743       .post('api/registrar/getclassrecinfo', {
2744         classRecordID: this.props.classRecordID,
2745         quarter: this.props.quarter,
2746         page: page,
2747         pageSize: this.state.pageSize,
2748       })
2749       .then(res => {
2750         this.setState({
2751           isLoading: false,
2752           hasData: true,
2753           data: res.data.studentList,
2754           numOfPages: res.data.numOfPages,
2755         });
2756       })
2757       .catch(err => {
2758         this.setState({ isLoading: false, hasData: true, data: [], numOfPages: 1, page: 1 });
2759       });
2760     });
2761   }
2762
2763 render() {
2764   const DisplayData = [];
2765   for (const [index, value] of this.state.data.entries()) {
2766     let check = value.q1Grade
2767       ? value.q2Grade
2768         ? value.q3Grade
2769           ? 'q3Grade'
2770             : 'q2Grade'
2771           : 'q1Grade'
2772         : '';
2773     DisplayData.push(
2774       <Table.Row>
2775         <Table.Col className="w-1">
2776           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getImageUrl(value.imageUrl)} />
2777         </Table.Col>
2778         <Table.Col>{value.name}</Table.Col>
2779         <Table.Col alignContent="center">

```

```

2780     {(value.grade != -1 || value.grade != 'N/A') && (
2781       <span
2782         className={'status-icon bg-${
2783           parseFloat(value.grade) >=
2784             75 ? 'green' : 'red'}'}
2785         />
2786       )}
2787     {value.grade == -1 ? 'N/A' : value.grade}
2788   </Table.Col>
2789   {check != '' && (
2790     <Table.Col alignContent="center">
2791       {value[check] == -1 ? (
2792         'N/A',
2793         ) : (
2794           <React.Fragment>
2795             <span
2796               className={'status-icon bg-${
2797                 parseFloat(value.grade) - parseFloat(value[check])
2798                   } >= 0 ? 'green' : 'red'
2799               }'
2800             />
2801             {parseFloat(value.grade) - parseFloat(value[check])}
2802           </React.Fragment>
2803         )
2804       </Table.Col>
2805     )}
2806     {value.q3Grade && (
2807       <React.Fragment>
2808         <Table.Col alignContent="center">
2809           {value.q3Grade && (value.q3Grade != -1 || value.q3Grade
2810             != 'N/A') && (
2811               <span
2812                 className={'status-icon bg-${
2813                   parseFloat(value.q3Grade) >= 75 ? 'green' : 'red'
2814                 }'}
2815               />
2816             )
2817             {value.q3Grade == -1 ? 'N/A' : value.q3Grade}
2818           </Table.Col>
2819         </React.Fragment>
2820     )}
2821     {value.q2Grade && (
2822       <React.Fragment>
2823         <Table.Col alignContent="center">
2824           {value.q2Grade && (value.q2Grade != -1 || value.q2Grade
2825             != 'N/A') && (
2826               <span
2827                 className={'status-icon bg-${
2828                   parseFloat(value.q2Grade) >= 75 ? 'green' : 'red'
2829                 }'}
2830               />
2831             )
2832             {value.q2Grade == -1 ? 'N/A' : value.q2Grade}
2833           </Table.Col>
2834         </React.Fragment>
2835     )}
2836     {value.q1Grade && (
2837       <React.Fragment>
2838         <Table.Col alignContent="center">
2839           {value.q1Grade && (value.q1Grade != -1 || value.q1Grade
2840             != 'N/A') && (
2841               <span
2842                 className={'status-icon bg-${
2843                   parseFloat(value.q1Grade) >= 75 ? 'green' : 'red'
2844                 }'}
2845               />
2846             )
2847             {value.q1Grade == -1 ? 'N/A' : value.q1Grade}
2848           </Table.Col>
2849         </React.Fragment>
2850     )}
2851   </Table.Row>, 2852 );

```

```

2847     }
2848     return (
2849       <Card>
2850         <Card.Body>
2851           <Card.Title>Class Record Information</Card.Title>
2852           <Spin spinning={this.state.isLoading}>
2853             {!this.state.hasData ? (
2854               <Container>
2855                 <Empty description="There is no selected subject." />
2856               </Container>
2857             ) : (
2858               <Container>
2859                 <Grid.Row>
2860                   <Grid.Col sm={12} xs={12} md={12}>
2861                     <Descriptions
2862                       style={{ marginBottom: '15px', marginTop: '15px' }}
2863                     bordered
2864                     title="Class Record Information"
2865                   >
2866                     <Descriptions.Item span={3} label="Subject Code">
2867                       {this.props.subjectCode}
2868                     </Descriptions.Item>
2869                     <Descriptions.Item span={3} label="Subject Name">
2870                       {this.props.subjectName}
2871                     </Descriptions.Item>
2872                     <Descriptions.Item span={3} label="Section Name">
2873                       {this.props.section}
2874                     </Descriptions.Item>
2875                     <Descriptions.Item span={3} label="Deadline">
2876                       {this.props.deadline}
2877                     </Descriptions.Item>
2878                   </Descriptions>
2879                 </Grid.Col>
2880               <Table highlightRowOnHover={true} responsive={true}>
2881                 <Table.Header>
2882                   <Table.ColHeader></Table.ColHeader>
2883                   <Table.ColHeader>Student Name</Table.ColHeader>
2884                   <Table.ColHeader alignContent="center">Final
2885                     Grade</Table.ColHeader>
2886                     {this.state.data.length != 0 && this.state.data
2887                       [0].q1Grade && (
2888                         <Table.ColHeader alignContent="center">
2889                           Deliberation Column</Table.ColHeader>
2890                         )
2891                     {this.state.data.length != 0 && this.state.data
2892                       [0].q3Grade && (
2893                         <React.Fragment>
2894                           <Table.ColHeader alignContent="center">Q3
2895                             Grade</Table.ColHeader>
2896                           </React.Fragment>
2897                         )
2898                     {this.state.data.length != 0 && this.state.data
2899                       [0].q2Grade && (
2900                         <React.Fragment>
2901                           <Table.ColHeader alignContent="center">Q2
2902                             Grade</Table.ColHeader>
2903                           </React.Fragment>
2904                         )
2905                     {this.state.data.length != 0 && this.state.data
2906                       [0].q1Grade && (
2907                         <React.Fragment>
2908                           <Table.ColHeader alignContent="center">Q1
2909                             Grade</Table.ColHeader>
2910                           </React.Fragment>
2911                         )
2912                     )
2913                   </Table.Header>
2914                   <Table.Body>
2915                     {DisplayData.length == 0 ? (
2916                       <Table.Row>
2917                         <Table.Col colSpan={5} alignContent="center">
2918                           No entries.
2919                         </Table.Col>

```

```

2910                     </Table.Row>
2911                 ) : (
2912                     DisplayData
2913                 )}
2914             </Table.Body>
2915         </Table>
2916     <Pagination
2917         size="large"
2918         current={this.state.page}
2919         pageSize={this.state.pageSize}
2920         total={this.state.pageSize * this.state.numOfPages}
2921         onChange={this.paginate}
2922     />
2923   </Grid.Row>
2924   </Container>
2925 }
2926 </Spin>
2927 </Card.Body>
2928 {!this.state.isLoading ? (
2929     this.state.hasData ? (
2930       <Card.Footer>
2931           <Grid.Col sm={12} xs={12} md={12}>
2932             {this.props.status == 'deliberation' && (
2933               <Button.List align="right">
2934                 <Button icon="file" color="primary">
2935                   <Link
2936                     style={{ color: 'white' }}
2937                     to={`/individualdeliberation/${this.props.id}/
2938                         managegrade/${this.props.classRecordID}/
2939                         quarter/${this.props.quarter}'}
2940                     target="_blank"
2941                   >
2942                     Edit Class Record
2943                     </Link>
2944                   </Button>
2945                 <Button icon="check" color="success" onClick={() =>
2946                     this.handlePostGrade()}>
2947                     Post Grades
2948                     </Button>
2949               </Button.List>
2950             )
2951           </Grid.Col>
2952           <Grid.Col sm={12} xs={12} md={12}>
2953             <Grid.Row>
2954               <Container>
2955                 {this.props.status == 'final' && (
2956                   <Button.List align="right">
2957                     <Button icon="eye" color="primary">
2958                       <Link
2959                         style={{ color: 'white' }}
2960                         to={`/viewstudentrecord/classrecord/${this.
2961                             props.classRecordID}/q/${this.props.
2962                             quarter}'}
2963                         target="_blank"
2964                       >
2965                         View Class Record
2966                         </Link>
2967                   </Button>
2968                 {!this.props.showRevert && this.props.auth.user
2969                   .position == 2 && (
2970                     <Button
2971                       icon="arrow-left-circle"
2972                       color="danger"
2973                       onClick={() => this.
2974                         handleRevertToDeliberation()}>
2975                         >
2976                           Revert to Deliberation Status
2977                         </Button>
2978                   )
2979                 )
2980               </Container>
2981             </Grid.Row>

```

```

2976             </Grid.Col>
2977         </Card.Footer>
2978     ) : (
2979     ,
2980     )
2981   ) : (
2982   ,
2983   )
2984   </Card>
2985 );
2986 }
2987 }
2988 */
2989 function mapStateToProps(state) {
2990   return {
2991     auth: state.app.auth,
2992   };
2993 }
2994 */
2995 */
2996 /* istanbul ignore next */
2997 function mapDispatchToProps(dispatch) {
2998   return {
2999     actions: bindActionCreators({ ...actions }, dispatch),
3000   };
3001 }
3002 */
3003 export default connect(mapStateToProps, mapDispatchToProps)(
3004   ClassRecordInformation);
3005 import React, { Component } from 'react';
3006 import PropTypes from 'prop-types';
3007 import { bindActionCreators } from 'redux';
3008 import { connect } from 'react-redux';
3009 import * as actions from './redux/actions';
3010 import NavBar from './NavBar';
3011 import AdminDashboard from './AdminDashboard';
3012 import DirectorDashboard from './DirectorDashboard';
3013 import RegistrarDashboard from './RegistrarDashboard';
3014 import TeacherDashboard from './TeacherDashboard';
3015 import StudentDashboard from './StudentDashboard';
3016 import ParentDashboard from './ParentDashboard';
3017 import CashierDashboard from './CashierDashboard';
3018 import { withRouter } from 'react-router-dom';
3019 */
3020 function DashboardComponent(props) {
3021   const { position } = props;
3022   switch (position) {
3023     case false:
3024       return <AdminDashboard />;
3025     case true:
3026       return <DirectorDashboard />;
3027     case 2:
3028       return <RegistrarDashboard />;
3029     case 3:
3030       return <TeacherDashboard />;
3031     case 4:
3032       return <StudentDashboard />;
3033     case 5:
3034       return <ParentDashboard />;
3035     case 6:
3036       return <CashierDashboard />;
3037     default:
3038       return <div></div>;
3039   }
3040 }
3041 */
3042 export class DashboardView extends Component {
3043   static propTypes = {
3044     app: PropTypes.object.isRequired,
3045     actions: PropTypes.object.isRequired,
3046   };
3047 }

```

```

3047     componentDidMount() {
3048       if (!this.props.app.auth.isAuthenticated) {
3049         this.props.history.push('/login');
3050       } else {
3051         this.props.actions.getCurrentProfile();
3052       }
3053     }
3054   }
3055   render() {
3056     return (
3057       <div className="app-dashboard-view fh">
3058         {this.props.app.showLoading ? (
3059           ,
3060         ) : (
3061           <NavBar>
3062             <DashboardComponent position={this.props.app.auth.user.
3063               position} />
3064           </NavBar>
3065         )}
3066       </div>
3067     );
3068   }
3069 }
3070 /* istanbul ignore next */
3071 function mapStateToProps(state) {
3072   return {
3073     app: state.app,
3074   };
3075 }
3076
3077 /* istanbul ignore next */
3078 function mapDispatchToProps(dispatch) {
3079   return {
3080     actions: bindActionCreators({ ...actions }, dispatch),
3081   };
3082 }
3083
3084 export default connect(mapStateToProps, mapDispatchToProps)(
  DashboardView);
3085 import React, { Component } from 'react';
3086 import PropTypes from 'prop-types';
3087 import { bindActionCreators } from 'redux';
3088 import { connect } from 'react-redux';
3089 import * as actions from './redux/actions';
3090 import { Container, Grid, Card, Button, Form, Header } from 'tabler-
  react';
3091 import cn from 'classnames';
3092 import placeholder from '../../images/placeholder.jpg';
3093 import { Spin } from 'antd';
3094 import bg from '../../images/BG.png';
3095 import { getImageUrl } from '../../utils';
3096 function ProfileImage({ avatarURL }) {
3097   return <img className="card-profile-img" alt="Profile" src={avatarURL
3098     } />;
3099 }
3100
3101 function Profile({
3102   className,
3103   children,
3104   name,
3105   avatarURL = '',
3106   twitterURL = '',
3107   backgroundURL = '',
3108   bio,
3109 }) {
3110   const classes = cn('card-profile', className);
3111   return (
3112     <Card className={classes}>
3113       <Card.Header backgroundURL={backgroundURL} />
3114       <Card.Body className="text-center">
3115         <ProfileImage avatarURL={avatarURL} />

```

```

3115         <Header.H3 className="mb-3">{name}</Header.H3>
3116         <p className="mb-4">{bio || children}</p>
3117     </Card.Body>
3118   </Card>
3119   );
3120 }
3121
3122 export class DirectorDashboard extends Component {
3123   static propTypes = {
3124     app: PropTypes.object.isRequired,
3125     actions: PropTypes.object.isRequired,
3126   };
3127
3128   constructor(props) {
3129     super(props);
3130     this.state = {
3131       isLoading: false,
3132       email: '',
3133       position: '',
3134       firstName: '',
3135       lastName: '',
3136       middleName: '',
3137       suffix: '',
3138       nickname: '',
3139       imageUrl: '',
3140       contactNum: '',
3141       address: '',
3142       province: '',
3143       city: '',
3144       region: '',
3145       zipcode: '',
3146       civilStatus: '',
3147       sex: '',
3148       citizenship: '',
3149       birthDate: 0,
3150       birthPlace: '',
3151       religion: '',
3152       emergencyName: '',
3153       emergencyAddress: '',
3154       emergencyTelephone: '',
3155       emergencyCellphone: '',
3156       emergencyEmail: '',
3157       emergencyRelationship: '',
3158     };
3159   }
3160
3161   componentWillReceiveProps(nextProps) {
3162     this.setState({
3163       email: nextProps.app.auth.user.email,
3164       position: nextProps.app.auth.user.position,
3165       firstName: nextProps.app.profile.firstName,
3166       lastName: nextProps.app.profile.lastName,
3167       middleName: nextProps.app.profile.middleName,
3168       suffix: nextProps.app.profile.suffix,
3169       nickname: nextProps.app.profile.nickname,
3170       imageUrl: nextProps.app.profile.imageUrl,
3171       contactNum: nextProps.app.profile.contactNum,
3172       address: nextProps.app.profile.address,
3173       province: nextProps.app.profile.province,
3174       city: nextProps.app.profile.city,
3175       region: nextProps.app.profile.region,
3176       zipcode: nextProps.app.profile.zipcode,
3177       civilStatus: nextProps.app.profile.civilStatus,
3178       sex: nextProps.app.profile.sex,
3179       citizenship: nextProps.app.profile.citizenship,
3180       birthDate: nextProps.app.profile.birthDate,
3181       birthPlace: nextProps.app.profile.birthPlace,
3182       religion: nextProps.app.profile.religion,
3183       emergencyName: nextProps.app.profile.emergencyName,
3184       emergencyAddress: nextProps.app.profile.emergencyAddress,
3185       emergencyTelephone: nextProps.app.profile.emergencyTelephone,
3186       emergencyCellphone: nextProps.app.profile.emergencyCellphone,
3187       emergencyEmail: nextProps.app.profile.emergencyEmail,
3188       emergencyRelationship: nextProps.app.profile.
3189     });
3190   }

```

```

3191
3192     componentDidMount() {
3193         if (Object.keys(this.props.app.profile).length !== 0) {
3194             this.setState({
3195                 email: this.props.app.auth.user.email,
3196                 position: this.props.app.auth.user.position,
3197                 firstName: this.props.app.profile.firstName,
3198                 lastName: this.props.app.profile.lastName,
3199                 middleName: this.props.app.profile.middleName,
3200                 suffix: this.props.app.profile.suffix,
3201                 nickname: this.props.app.profile.nickname,
3202                 imageUrl: this.props.app.profile.imageUrl,
3203                 contactNum: this.props.app.profile.contactNum,
3204                 address: this.props.app.profile.address,
3205                 province: this.props.app.profile.province,
3206                 city: this.props.app.profile.city,
3207                 region: this.props.app.profile.region,
3208                 zipcode: this.props.app.profile.zipcode,
3209                 civilStatus: this.props.app.profile.civilStatus,
3210                 sex: this.props.app.profile.sex,
3211                 citizenship: this.props.app.profile.citizenship,
3212                 birthDate: this.props.app.profile.birthDate,
3213                 birthPlace: this.props.app.profile.birthPlace,
3214                 religion: this.props.app.profile.religion,
3215                 emergencyName: this.props.app.profile.emergencyName,
3216                 emergencyAddress: this.props.app.profile.emergencyAddress,
3217                 emergencyTelephone: this.props.app.profile.emergencyTelephone,
3218                 emergencyCellphone: this.props.app.profile.emergencyCellphone,
3219                 emergencyEmail: this.props.app.profile.emergencyEmail,
3220                 emergencyRelationship: this.props.app.profile.
3221                     emergencyRelationship,
3222             });
3223         }
3224     }
3225     render() {
3226         const {
3227             email,
3228             position,
3229             firstName,
3230             lastName,
3231             middleName,
3232             suffix,
3233             nickname,
3234             imageUrl,
3235             contactNum,
3236             address,
3237             province,
3238             city,
3239             region,
3240             zipcode,
3241             civilStatus,
3242             sex,
3243             citizenship,
3244             birthPlace,
3245             religion,
3246             emergencyName,
3247             emergencyAddress,
3248             emergencyTelephone,
3249             emergencyCellphone,
3250             emergencyEmail,
3251             emergencyRelationship,
3252         } = this.state;
3253         const birthDate = new Date(this.state.birthDate);
3254         const { errors } = this.props.app;
3255         const uploadLoading = false;
3256         const displayPosition = position => {
3257             switch (position) {
3258                 case false:
3259                     return 'Administrator';
3260                 case true:
3261                     return 'Director';
3262                 case 2:
3263                     return 'Registrar';
3264                 case 3:
3265                     return 'Teacher';
3266                 case 4:

```

```

3267         return 'Student';
3268     case 5:
3269         return 'Guardian';
3270     case 6:
3271         return 'Cashier';
3272     default:
3273         return '';
3274     }
3275   };
3276
3277   const getPHTime = (date = '') => {
3278     const d = date === '' ? new Date() : new Date(date);
3279     const localOffset = d.getTimezoneOffset() * 60000;
3280     const UTC = d.getTime() + localOffset;
3281     const PHT = UTC + 3600000 * 16;
3282     return new Date(PHT);
3283   };
3284   const capitalize = string => {
3285     return string.charAt(0).toUpperCase() + string.slice(1).
3286      toLowerCase();
3287   };
3288   return (
3289     <div className="app-teacher-dashboard my-3 my-md-5">
3290       <Spin spinning={this.props.app.showLoading}>
3291         <Container>
3292           <Grid.Row>
3293             <Grid.Col sm={12} lg={4}>
3294               <Profile
3295                 name={`${firstName} ${middleName.charAt(0) .
3296                  toUpperCase()} ${lastName}`}
3297                 avatarURL={imageUrl === 'NA' ? placeholder :
3298                   getImageUrl(imageUrl)}
3299                 backgroundURL={bg}
3300               >
3301                 <div>
3302                   <Header.H5>{displayPosition(position)}</Header.H5>
3303                 </div>
3304               </Profile>
3305             </Grid.Col>
3306             <Grid.Col xs={12} sm={12} lg={8}>
3307               <Form className="card" onSubmit={this.onSubmit}>
3308                 <Card.Body>
3309                   <Card.Title>Account Information</Card.Title>
3310                   <Grid.Row>
3311                     <Grid.Col xs={12} sm={12} md={6}>
3312                       <Form.Group>
3313                         <Form.Label>Email</Form.Label>
3314                         <Form.Input type="email" disabled placeholder
3315                           ="Email" value={email} />
3316                       </Form.Group>
3317                     </Grid.Col>
3318                     <Grid.Col sm={12} md={6}>
3319                       <Form.Group>
3320                         <Form.Label>Position</Form.Label>
3321                         <Form.Input
3322                           type="text"
3323                           placeholder="Position"
3324                           disabled
3325                           value={displayPosition(position)}>
3326                         </Form.Input>
3327                       </Form.Group>
3328                     </Grid.Col>
3329                     <Grid.Col sm={12} md={4}>
3330                       <Form.Group>
3331                         <Form.Label>First Name</Form.Label>
3332                         <Form.Input
3333                           disabled
3334                           type="text"
3335                           name="firstName"
3336                           placeholder="First Name"
3337                           value={firstName}>
3338                         </Form.Input>
3339                       </Form.Group>

```

```

3336 </Grid.Col>
3337 <Grid.Col sm={12} md={4}>
3338   <Form.Group>
3339     <Form.Label>Middle Name</Form.Label>
3340     <Form.Input
3341       disabled
3342       type="text"
3343       placeholder="Middle Name"
3344       value={middleName}
3345       name="middleName"
3346     />
3347   </Form.Group>
3348 </Grid.Col>
3349 <Grid.Col sm={12} md={4}>
3350   <Form.Group>
3351     <Form.Label>Last Name</Form.Label>
3352     <Form.Input
3353       type="text"
3354       disabled
3355       placeholder="Last Name"
3356       value={lastName}
3357       name="lastName"
3358     />
3359   </Form.Group>
3360 </Grid.Col>
3361 <Grid.Col sm={12} md={3}>
3362   <Form.Group>
3363     <Form.Label>Nickname</Form.Label>
3364     <Form.Input
3365       type="text"
3366       disabled
3367       placeholder="Nickname"
3368       value={nickname}
3369       name="nickname"
3370     />
3371   </Form.Group>
3372 </Grid.Col>
3373 <Grid.Col sm={12} md={3}>
3374   <Form.Group>
3375     <Form.Label>Suffix</Form.Label>
3376     <Form.Input
3377       type="text"
3378       disabled
3379       placeholder="Suffix"
3380       value={suffix}
3381       name="suffix"
3382     />
3383   </Form.Group>
3384 </Grid.Col>
3385 <Grid.Col sm={12} md={6}>
3386   <Form.Group>
3387     <Form.Label>Contact Number</Form.Label>
3388     <Form.Input
3389       type="text"
3390       disabled
3391       placeholder="Contact Number"
3392       value={contactNum}
3393       name="contactNum"
3394     />
3395   </Form.Group>
3396 </Grid.Col>
3397 <Grid.Col sm={12} md={12}>
3398   <Form.Group>
3399     <Form.Label>Address</Form.Label>
3400     <Form.Input
3401       type="text"
3402       disabled
3403       placeholder="Home Address"
3404       value={address}
3405       name="address"
3406     />
3407   </Form.Group>
3408 </Grid.Col>
3409 <Grid.Col sm={12} md={4}>

```

```

3410 <Form.Group>
3411   <Form.Label>Province</Form.Label>
3412   <Form.Input
3413     type="text"
3414     disabled
3415     placeholder="Province"
3416     value={province}
3417     name="province"
3418   />
3419 </Form.Group>
3420 </Grid.Col>
3421 <Grid.Col sm={12} md={4}>
3422   <Form.Group>
3423     <Form.Label>City</Form.Label>
3424     <Form.Input
3425       type="text"
3426       disabled
3427       placeholder="City"
3428       value={city}
3429       name="city"
3430     />
3431   </Form.Group>
3432 </Grid.Col>
3433 <Grid.Col sm={12} md={2}>
3434   <Form.Group>
3435     <Form.Label>Region</Form.Label>
3436     <Form.Input
3437       type="text"
3438       disabled
3439       placeholder="Region"
3440       value={region}
3441       name="region"
3442     />
3443   </Form.Group>
3444 </Grid.Col>
3445 <Grid.Col sm={12} md={2}>
3446   <Form.Group>
3447     <Form.Label>Zipcode</Form.Label>
3448     <Form.Input
3449       type="text"
3450       disabled
3451       placeholder="Postal Code"
3452       value={zipcode}
3453       name="zipcode"
3454     />
3455   </Form.Group>
3456 </Grid.Col>
3457 <Grid.Col sm={12} md={4}>
3458   <Form.Group>
3459     <Form.Label>Civil Status</Form.Label>
3460     <Form.Input
3461       type="text"
3462       disabled
3463       placeholder="Civil Status"
3464       value={civilStatus}
3465       name="civilStatus"
3466     />
3467   </Form.Group>
3468 </Grid.Col>
3469 <Grid.Col sm={12} md={2}>
3470   <Form.Group>
3471     <Form.Label>Sex</Form.Label>
3472     <Form.Input
3473       type="text"
3474       disabled
3475       placeholder="Sex"
3476       value={sex}
3477       name="sex"
3478     />
3479   </Form.Group>
3480 </Grid.Col>
3481 <Grid.Col sm={12} md={6}>
3482   <Form.Group>
3483     <Form.Label>Citizenship</Form.Label>

```

```

3484             <Form.Input
3485                 disabled
3486                 type="text"
3487                 placeholder="Citizenship"
3488                 value={citizenship}
3489                 name="citizenship"
3490             />
3491         </Form.Group>
3492     </Grid.Col>
3493     <Grid.Col sm={12} md={4}>
3494         <Form.Group>
3495             <Form.Label>Birth Date</Form.Label>
3496             <Form.Input
3497                 disabled
3498                 type="text"
3499                 placeholder="Birth date"
3500                 value={birthDate}
3501             ></Form.Input>
3502         </Form.Group>
3503     </Grid.Col>
3504     <Grid.Col sm={12} md={4}>
3505         <Form.Group>
3506             <Form.Label>Birth Place</Form.Label>
3507             <Form.Input
3508                 disabled
3509                 type="text"
3510                 placeholder="Birth Place"
3511                 value={birthPlace}
3512                 name="birthPlace"
3513             />
3514         </Form.Group>
3515     </Grid.Col>
3516     <Grid.Col sm={12} md={4}>
3517         <Form.Group>
3518             <Form.Label>Religion</Form.Label>
3519             <Form.Input
3520                 disabled
3521                 type="text"
3522                 placeholder="Religion"
3523                 value={religion}
3524                 name="religion"
3525             />
3526         </Form.Group>
3527     </Grid.Col>
3528 </Grid.Row>
3529 </Card.Body>
3530 <Card.Body>
3531     <Card.Title>Emergency Contact Information</Card.
3532         Title>
3533     <Grid.Row>
3534         <Grid.Col sm={12} md={6}>
3535             <Form.Group>
3536                 <Form.Label>Contact Person</Form.Label>
3537                 <Form.Input
3538                     type="text"
3539                     disabled
3540                     placeholder="Contact Person"
3541                     value={emergencyName}
3542                     name="emergencyName"
3543             />
3544         </Form.Group>
3545     </Grid.Col>
3546     <Grid.Col sm={12} md={6}>
3547         <Form.Group>
3548             <Form.Label>Contact Address</Form.Label>
3549             <Form.Input
3550                 type="text"
3551                 disabled
3552                 placeholder="Contact Address"
3553                 value={emergencyAddress}
3554                 name="emergencyAddress"
3555             />
3556         </Form.Group>

```

```

3557         <Grid.Col sm={12} md={6}>
3558             <Form.Group>
3559                 <Form.Label>Contact Telephone No.</Form.Label>
3560                     >
3561                         <Form.Input
3562                             type="text"
3563                             disabled
3564                             placeholder="Contact Telephone No."
3565                             value={emergencyTelephone}
3566                             name="emergencyTelephone"
3567                         />
3568                     </Form.Group>
3569             </Grid.Col>
3570             <Grid.Col sm={12} md={6}>
3571                 <Form.Group>
3572                     <Form.Label>Contact Cellphone No.</Form.Label>
3573                         >
3574                             <Form.Input
3575                                 type="text"
3576                                 disabled
3577                                 placeholder="Contact Cellphone No."
3578                                 value={emergencyCellphone}
3579                                 name="emergencyCellphone"
3580                         />
3581                     </Form.Group>
3582             </Grid.Col>
3583             <Grid.Col sm={12} md={6}>
3584                 <Form.Group>
3585                     <Form.Label>Contact Email</Form.Label>
3586                         <Form.Input
3587                             disabled
3588                             type="text"
3589                             placeholder="Contact Email"
3590                             value={emergencyEmail}
3591                             name="emergencyEmail"
3592                         />
3593                     </Form.Group>
3594             </Grid.Col>
3595             <Grid.Col sm={12} md={6}>
3596                 <Form.Group>
3597                     <Form.Label>Contact Relationship</Form.Label>
3598                         <Form.Input
3599                             type="text"
3600                             disabled
3601                             placeholder="Contact Relationship"
3602                             value={emergencyRelationship}
3603                             name="emergencyRelationship"
3604                         />
3605                     </Form.Group>
3606             </Grid.Col>
3607         </Grid.Row>
3608     </Card.Body>
3609   </Form>
3610 </Grid>
3611 </Container>
3612 </Spin>
3613 </div>
3614 );
3615 }
3616
3617 /* istanbul ignore next */
3618 function mapStateToProps(state) {
3619     return {
3620         app: state.app,
3621     };
3622 }
3623
3624 /* istanbul ignore next */
3625 function mapDispatchToProps(dispatch) {
3626     return {
3627         actions: bindActionCreators({ ...actions }, dispatch),

```

```

3628     };
3629 }
3630
3631 export default connect(mapStateToProps, mapDispatchToProps)(
  DirectorDashboard);
3632 import React, { Component } from 'react';
3633 import PropTypes from 'prop-types';
3634 import { bindActionCreators } from 'redux';
3635 import { connect } from 'react-redux';
3636 import * as actions from './redux/actions';
3637 import moment from 'moment';
3638 import axios from 'axios';
3639 import { Container, Grid, Card, Button, Form, Header, List } from 'table-react';
3640 import { Alert, Upload, message } from 'antd';
3641 import cn from 'classnames';
3642 import placeholder from '../../images/placeholder.jpg';
3643 import bg from '../../images/BG.png';
3644 import { getImageUrl } from '../../utils';
3645 function ProfileImage({ avatarURL }) {
3646   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
3647 }
3648
3649 function Profile({
3650   className,
3651   children,
3652   name,
3653   avatarURL = '',
3654   twitterURL = '',
3655   backgroundURL = '',
3656   bio,
3657 }) {
3658   const classes = cn('card-profile', className);
3659   return (
3660     <Card className={classes}>
3661       <Card.Header backgroundURL={backgroundURL} />
3662       <Card.Body className="text-center">
3663         <ProfileImage avatarURL={avatarURL} />
3664         <Header.H3 className="mb-3">{name}</Header.H3>
3665         <p className="mb-4">{bio || children}</p>
3666       </Card.Body>
3667     </Card>
3668   );
3669 }
3670 export class DirectorProfile extends Component {
3671   static propTypes = {
3672     app: PropTypes.object.isRequired,
3673     actions: PropTypes.object.isRequired,
3674   };
3675
3676   constructor(props) {
3677     super(props);
3678     this.state = {
3679       loading: true,
3680       email: '',
3681       position: '',
3682       firstName: '',
3683       lastName: '',
3684       middleName: '',
3685       suffix: '',
3686       nickname: '',
3687       imageUrl: '',
3688       contactNum: '',
3689       address: '',
3690       province: '',
3691       city: '',
3692       region: '',
3693       zipcode: '',
3694       civilStatus: '',
3695       sex: '',
3696       citizenship: '',
3697       birthDate: new Date(),
3698       birthPlace: ''
3699     };

```

```

3699         religion: '',
3700         emergencyName: '',
3701         emergencyAddress: '',
3702         emergencyTelephone: '',
3703         emergencyCellphone: '',
3704         emergencyEmail: '',
3705         emergencyRelationship: '',
3706     );
3707
3708     this.onSubmit = this.onSubmit.bind(this);
3709     this.onChange = this.onChange.bind(this);
3710     this.onDateChange = this.onDateChange.bind(this);
3711 }
3712 componentWillReceiveProps(nextProps) {
3713     if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
3714         .length == 0)
3715         this.setState({
3716             email: nextProps.app.auth.user.email,
3717             position: nextProps.app.auth.user.position,
3718             firstName: nextProps.app.profile.firstName,
3719             lastName: nextProps.app.profile.lastName,
3720             middleName: nextProps.app.profile.middleName,
3721             suffix: nextProps.app.profile.suffix,
3722             nickname: nextProps.app.profile.nickname,
3723             imageUrl: nextProps.app.profile.imageUrl,
3724             contactNum: nextProps.app.profile.contactNum,
3725             address: nextProps.app.profile.address,
3726             province: nextProps.app.profile.province,
3727             city: nextProps.app.profile.city,
3728             region: nextProps.app.profile.region,
3729             zipcode: nextProps.app.profile.zipcode,
3730             civilStatus: nextProps.app.profile.civilStatus,
3731             sex: nextProps.app.profile.sex,
3732             citizenship: nextProps.app.profile.citizenship,
3733             birthDate: nextProps.app.profile.birthDate,
3734             birthPlace: nextProps.app.profile.birthPlace,
3735             religion: nextProps.app.profile.religion,
3736             emergencyName: nextProps.app.profile.emergencyName,
3737             emergencyAddress: nextProps.app.profile.emergencyAddress,
3738             emergencyTelephone: nextProps.app.profile.emergencyTelephone,
3739             emergencyCellphone: nextProps.app.profile.emergencyCellphone,
3740             emergencyEmail: nextProps.app.profile.emergencyEmail,
3741             emergencyRelationship: nextProps.app.profile.
3742                 emergencyRelationship,
3743             loading: false,
3744         });
3745     onChange(event) {
3746         this.setState({ [event.target.name]: event.target.value });
3747     }
3748     onDateChange(event) {
3749         this.setState({ birthDate: event });
3750     }
3751     onSubmit(event) {
3752         event.preventDefault();
3753         const {
3754             firstName,
3755             lastName,
3756             middleName,
3757             suffix,
3758             nickname,
3759             contactNum,
3760             address,
3761             province,
3762             city,
3763             region,
3764             zipcode,
3765             civilStatus,
3766             sex,
3767             citizenship,
3768             birthDate,
3769             birthPlace,
3770             religion,
3771             emergencyName,
3772             emergencyAddress,
3773             emergencyTelephone,
3774             emergencyCellphone,
3775             emergencyEmail,
3776             emergencyRelationship
3777         } = this.state;
3778         const errors = {};
3779         let valid = true;
3780
3781         if (!firstName) {
3782             errors.firstName = 'First name is required';
3783             valid = false;
3784         }
3785         if (!lastName) {
3786             errors.lastName = 'Last name is required';
3787             valid = false;
3788         }
3789         if (!middleName) {
3790             errors.middleName = 'Middle name is required';
3791             valid = false;
3792         }
3793         if (!suffix) {
3794             errors.suffix = 'Suffix is required';
3795             valid = false;
3796         }
3797         if (!nickname) {
3798             errors.nickname = 'Nickname is required';
3799             valid = false;
3800         }
3801         if (!contactNum) {
3802             errors.contactNum = 'Contact number is required';
3803             valid = false;
3804         }
3805         if (!address) {
3806             errors.address = 'Address is required';
3807             valid = false;
3808         }
3809         if (!province) {
3810             errors.province = 'Province is required';
3811             valid = false;
3812         }
3813         if (!city) {
3814             errors.city = 'City is required';
3815             valid = false;
3816         }
3817         if (!region) {
3818             errors.region = 'Region is required';
3819             valid = false;
3820         }
3821         if (!zipcode) {
3822             errors.zipcode = 'Zipcode is required';
3823             valid = false;
3824         }
3825         if (!civilStatus) {
3826             errors.civilStatus = 'Civil status is required';
3827             valid = false;
3828         }
3829         if (!sex) {
3830             errors.sex = 'Sex is required';
3831             valid = false;
3832         }
3833         if (!citizenship) {
3834             errors.citizenship = 'Citizenship is required';
3835             valid = false;
3836         }
3837         if (!birthDate) {
3838             errors.birthDate = 'Birth date is required';
3839             valid = false;
3840         }
3841         if (!birthPlace) {
3842             errors.birthPlace = 'Birth place is required';
3843             valid = false;
3844         }
3845         if (!religion) {
3846             errors.religion = 'Religion is required';
3847             valid = false;
3848         }
3849         if (!emergencyName) {
3850             errors.emergencyName = 'Emergency name is required';
3851             valid = false;
3852         }
3853         if (!emergencyAddress) {
3854             errors.emergencyAddress = 'Emergency address is required';
3855             valid = false;
3856         }
3857         if (!emergencyTelephone) {
3858             errors.emergencyTelephone = 'Emergency telephone is required';
3859             valid = false;
3860         }
3861         if (!emergencyCellphone) {
3862             errors.emergencyCellphone = 'Emergency cellphone is required';
3863             valid = false;
3864         }
3865         if (!emergencyEmail) {
3866             errors.emergencyEmail = 'Emergency email is required';
3867             valid = false;
3868         }
3869         if (!emergencyRelationship) {
3870             errors.emergencyRelationship = 'Emergency relationship is required';
3871             valid = false;
3872         }
3873
3874         if (valid) {
3875             this.props.onSubmit(this.state);
3876         } else {
3877             this.setState({ errors });
3878         }
3879     }
3880 }

```

```

3773     emergencyName ,
3774     emergencyAddress ,
3775     emergencyTelephone ,
3776     emergencyCellphone ,
3777     emergencyEmail ,
3778     emergencyRelationship ,
3779   } = this.state;
3780   const profileData = {
3781     firstName ,
3782     lastName ,
3783     middleName ,
3784     suffix ,
3785     nickname ,
3786     contactNum ,
3787     address ,
3788     province ,
3789     city ,
3790     region ,
3791     zipcode ,
3792     civilStatus ,
3793     sex ,
3794     citizenship ,
3795     birthDate ,
3796     birthPlace ,
3797     religion ,
3798     emergencyName ,
3799     emergencyAddress ,
3800     emergencyTelephone ,
3801     emergencyCellphone ,
3802     emergencyEmail ,
3803     emergencyRelationship ,
3804   };
3805   console.log(profileData);
3806   this.props.actions.updateDirectorProfile(profileData);
3807 }
3808
3809   render() {
3810     const {
3811       email ,
3812       position ,
3813       firstName ,
3814       lastName ,
3815       middleName ,
3816       suffix ,
3817       nickname ,
3818       imageUrl ,
3819       contactNum ,
3820       address ,
3821       province ,
3822       city ,
3823       region ,
3824       zipcode ,
3825       civilStatus ,
3826       sex ,
3827       citizenship ,
3828       birthPlace ,
3829       religion ,
3830       emergencyName ,
3831       emergencyAddress ,
3832       emergencyTelephone ,
3833       emergencyCellphone ,
3834       emergencyEmail ,
3835       emergencyRelationship ,
3836     } = this.state;
3837   const birthDate = new Date(this.state.birthDate);
3838   const defaultDate = birthDate.toISOString();
3839   const { errors } = this.props.app;
3840   const uploadLoading = false;
3841   const displayPosition = position => {
3842     switch (position) {
3843       case false:
3844         return 'Administrator';
3845       case true:
3846         return 'Director';
3847       case 2:
3848         return 'Registrar';
3849       case 3:

```

```

3850     return 'Teacher';
3851     case 4:
3852       return 'Student';
3853     case 5:
3854       return 'Guardian';
3855     default:
3856       return '';
3857   }
3858 };
3859 const capitalize = string => {
3860   return string.charAt(0).toUpperCase() + string.slice(1).
3861   toLowerCase();
3862 };
3863 return (
3864   <div className="app-admin-profile my-3 my-md-5">
3865     {this.state.loading ? (
3866       '',
3867     ) : (
3868       <Container>
3869         <Grid.Row>
3870           <Grid.Col sm={12} lg={4}>
3871             <Profile
3872               name={`${firstName} ${middleName.charAt(0) .
3873                 .toUpperCase()} ${lastName}`}
3874               avatarURL={imageUrl == 'NA' ? placeholder :
3875                 getImageUrl(imageUrl)}
3876               backgroundURL={bg}
3877             >
3878               <Upload
3879                 name="file"
3880                 multiple={false}
3881                 accept=".png, .jpg"
3882                 customRequest={(req, uploadLoading) => {
3883                   // Axios Photo Upload
3884                   this.props.actions.setLoading(true);
3885                   const bodyFormData = new FormData();
3886                   bodyFormData.set('file', req.file);
3887                   axios({
3888                     method: 'post',
3889                     url: 'api/users/upload',
3890                     data: bodyFormData,
3891                   })
3892                     .then(res => {
3893                       req.onSuccess();
3894                       this.props.actions.setLoading(false);
3895                       message.success('Uploaded Successfully!');
3896                         then(() => {
3897                           window.location.href = '/profile';
3898                         });
3899                     });
3900                   .catch(err => {
3901                     req.onError();
3902                     this.props.actions.setLoading(false);
3903                     message.error('Upload failed!');
3904                   });
3905                 })
3906               >
3907                 <div>
3908                   <Header.H4>{displayPosition(position)}</Header.H4
3909                 >
3910               </div>
3911               <Button loading={this.props.app.showLoading} icon="
3912                 upload" pill color="primary">
3913                   Upload Image
3914                 </Button>
3915               </Upload>
3916             </Profile>
3917           </Grid.Col>
3918           <Grid.Col xs={12} sm={12} lg={8}>
3919             <Form className="card" onSubmit={this.onSubmit}>
3920               <Card.Body>
3921                 <Card.Title>Edit Profile</Card.Title>
3922               <Grid.Row>

```

```

3917 <Grid.Col xs={12} sm={12} md={6}>
3918   <Form.Group>
3919     <Form.Label>Email</Form.Label>
3920     <Form.Input
3921       type="email"
3922       disabled
3923       placeholder="Email"
3924       value={email}
3925       error={errors.email}
3926     />
3927   </Form.Group>
3928 </Grid.Col>
3929 <Grid.Col sm={12} md={6}>
3930   <Form.Group>
3931     <Form.Label>Position</Form.Label>
3932     <Form.Input
3933       type="text"
3934       placeholder="Position"
3935       disabled
3936       value={displayPosition(position)}
3937       error={errors.position}
3938     />
3939   </Form.Group>
3940 </Grid.Col>
3941 <Grid.Col sm={12} md={4}>
3942   <Form.Group>
3943     <Form.Label>First Name</Form.Label>
3944     <Form.Input
3945       type="text"
3946       name="firstName"
3947       placeholder="First Name"
3948       value={firstName}
3949       onChange={this.onChange}
3950       error={errors.firstName}
3951     />
3952   </Form.Group>
3953 </Grid.Col>
3954 <Grid.Col sm={12} md={4}>
3955   <Form.Group>
3956     <Form.Label>Middle Name</Form.Label>
3957     <Form.Input
3958       type="text"
3959       placeholder="Middle Name"
3960       value={middleName}
3961       onChange={this.onChange}
3962       name="middleName"
3963       error={errors.middleName}
3964     />
3965   </Form.Group>
3966 </Grid.Col>
3967 <Grid.Col sm={12} md={4}>
3968   <Form.Group>
3969     <Form.Label>Last Name</Form.Label>
3970     <Form.Input
3971       type="text"
3972       placeholder="Last Name"
3973       value={lastName}
3974       onChange={this.onChange}
3975       name="lastName"
3976       error={errors.lastName}
3977     />
3978   </Form.Group>
3979 </Grid.Col>
3980 <Grid.Col sm={12} md={3}>
3981   <Form.Group>
3982     <Form.Label>Nickname</Form.Label>
3983     <Form.Input
3984       type="text"
3985       placeholder="Nickname"
3986       value={nickname}
3987       onChange={this.onChange}
3988       name="nickname"
3989       error={errors.nickname}
3990     />

```

```

3991         </Form.Group>
3992     </Grid.Col>
3993     <Grid.Col sm={12} md={3}>
3994         <Form.Group>
3995             <Form.Label>Suffix</Form.Label>
3996             <Form.Input
3997                 type="text"
3998                 placeholder="Suffix"
3999                 value={suffix}
4000                 onChange={this.onChange}
4001                 name="suffix"
4002                 error={errors.suffix}
4003             />
4004         </Form.Group>
4005     </Grid.Col>
4006     <Grid.Col sm={12} md={6}>
4007         <Form.Group>
4008             <Form.Label>Contact Number</Form.Label>
4009             <Form.Input
4010                 type="text"
4011                 placeholder="Contact Number"
4012                 value={contactNum}
4013                 onChange={this.onChange}
4014                 name="contactNum"
4015                 error={errors.contactNum}
4016             />
4017         </Form.Group>
4018     </Grid.Col>
4019     <Grid.Col sm={12} md={12}>
4020         <Form.Group>
4021             <Form.Label>Address</Form.Label>
4022             <Form.Input
4023                 type="text"
4024                 placeholder="Home Address"
4025                 value={address}
4026                 onChange={this.onChange}
4027                 name="address"
4028                 error={errors.address}
4029             />
4030         </Form.Group>
4031     </Grid.Col>
4032     <Grid.Col sm={12} md={4}>
4033         <Form.Group>
4034             <Form.Label>Province</Form.Label>
4035             <Form.Input
4036                 type="text"
4037                 placeholder="Province"
4038                 value={province}
4039                 onChange={this.onChange}
4040                 name="province"
4041                 error={errors.province}
4042             />
4043         </Form.Group>
4044     </Grid.Col>
4045     <Grid.Col sm={12} md={4}>
4046         <Form.Group>
4047             <Form.Label>City</Form.Label>
4048             <Form.Input
4049                 type="text"
4050                 placeholder="City"
4051                 value={city}
4052                 onChange={this.onChange}
4053                 name="city"
4054                 error={errors.city}
4055             />
4056         </Form.Group>
4057     </Grid.Col>
4058     <Grid.Col sm={12} md={2}>
4059         <Form.Group>
4060             <Form.Label>Region</Form.Label>
4061             <Form.Input
4062                 type="text"
4063                 placeholder="Region"

```

```

4064           value={region}
4065           onChange={this.onChange}
4066           name="region"
4067           error={errors.region}
4068       />
4069     </Form.Group>
4070   </Grid.Col>
4071   <Grid.Col sm={12} md={2}>
4072     <Form.Group>
4073       <Form.Label>Zipcode</Form.Label>
4074       <Form.Input
4075         type="text"
4076         placeholder="Postal Code"
4077         value={zipcode}
4078         onChange={this.onChange}
4079         name="zipcode"
4080         error={errors.zipcode}
4081       />
4082     </Form.Group>
4083   </Grid.Col>
4084   <Grid.Col sm={12} md={4}>
4085     <Form.Group>
4086       <Form.Label>Civil Status</Form.Label>
4087       <Form.Select
4088         value={civilStatus}
4089         onChange={this.onChange}
4090         name="civilStatus"
4091       >
4092         <option>SINGLE</option>
4093         <option>MARRIED</option>
4094         <option>WIDOWED</option>
4095         <option>OTHERS</option>
4096       </Form.Select>
4097     </Form.Group>
4098   </Grid.Col>
4099   <Grid.Col sm={12} md={2}>
4100     <Form.Group>
4101       <Form.Label>Sex</Form.Label>
4102       <Form.Select value={sex} onChange={this.
4103         onChange} name="sex">
4104         <option>M</option>
4105         <option>F</option>
4106       </Form.Select>
4107     </Form.Group>
4108   </Grid.Col>
4109   <Grid.Col sm={12} md={6}>
4110     <Form.Group>
4111       <Form.Label>Citizenship</Form.Label>
4112       <Form.Input
4113         type="text"
4114         placeholder="Citizenship"
4115         value={citizenship}
4116         onChange={this.onChange}
4117         name="citizenship"
4118         error={errors.citizenship}
4119       />
4120     </Form.Group>
4121   </Grid.Col>
4122   <Grid.Col sm={12} md={4}>
4123     <Form.Group>
4124       <Form.Label>Birth Date</Form.Label>
4125       <Form.DatePicker
4126         defaultDate={new Date(defaultDate)}
4127         format="mm/dd/yyyy"
4128         onChange={this.onDateChange}
4129         name="birthDate"
4130         maxYear={2020}
4131         minYear={1897}
4132         monthLabels={[
4133           'January',
4134           'February',
4135           'March',
4136           'April',

```

```

4136          'May',
4137          'June',
4138          'July',
4139          'August',
4140          'September',
4141          'October',
4142          'November',
4143          'December',
4144      ]}
4145    ></Form.DatePicker>
4146  </Form.Group>
4147</Grid.Col>
4148<Grid.Col sm={12} md={4}>
4149  <Form.Group>
4150    <Form.Label>Birth Place</Form.Label>
4151    <Form.Input
4152      type="text"
4153      placeholder="Birth Place"
4154      value={birthPlace}
4155      onChange={this.onChange}
4156      name="birthPlace"
4157      error={errors.birthPlace}
4158    />
4159  </Form.Group>
4160</Grid.Col>
4161<Grid.Col sm={12} md={4}>
4162  <Form.Group>
4163    <Form.Label>Religion</Form.Label>
4164    <Form.Input
4165      type="text"
4166      placeholder="Religion"
4167      value={religion}
4168      onChange={this.onChange}
4169      name="religion"
4170      error={errors.religion}
4171    />
4172  </Form.Group>
4173</Grid.Col>
4174</Grid.Row>
4175</Card.Body>
4176<Card.Body>
4177  <Card.Title>Emergency Contact Information</Card.
4178  Title>
4179  <Grid.Row>
4180    <Grid.Col sm={12} md={6}>
4181      <Form.Group>
4182        <Form.Label>Contact Person</Form.Label>
4183        <Form.Input
4184          type="text"
4185          placeholder="Contact Person"
4186          value={emergencyName}
4187          onChange={this.onChange}
4188          name="emergencyName"
4189          error={errors.emergencyName}
4190        />
4191      </Form.Group>
4192    <Grid.Col sm={12} md={6}>
4193      <Form.Group>
4194        <Form.Label>Contact Address</Form.Label>
4195        <Form.Input
4196          type="text"
4197          placeholder="Contact Address"
4198          value={emergencyAddress}
4199          onChange={this.onChange}
4200          name="emergencyAddress"
4201          error={errors.emergencyAddress}
4202        />
4203      </Form.Group>
4204    <Grid.Col sm={12} md={6}>
4205      <Form.Group>
4206        <Form.Label>Contact Telephone No.</Form.Label
4207        >

```

```

4208             <Form.Input
4209                 type="text"
4210                 placeholder="Contact Telephone No."
4211                 value={emergencyTelephone}
4212                 onChange={this.onChange}
4213                 name="emergencyTelephone"
4214                 error={errors.emergencyTelephone}
4215             />
4216         </Form.Group>
4217     </Grid.Col>
4218     <Grid.Col sm={12} md={6}>
4219         <Form.Group>
4220             <Form.Label>Contact Cellphone No.</Form.Label>
4221             >
4222             <Form.Input
4223                 type="text"
4224                 placeholder="Contact Cellphone No."
4225                 value={emergencyCellphone}
4226                 onChange={this.onChange}
4227                 name="emergencyCellphone"
4228                 error={errors.emergencyCellphone}
4229             />
4230         </Form.Group>
4231     </Grid.Col>
4232     <Grid.Col sm={12} md={6}>
4233         <Form.Group>
4234             <Form.Label>Contact Email</Form.Label>
4235             <Form.Input
4236                 type="text"
4237                 placeholder="Contact Email"
4238                 value={emergencyEmail}
4239                 onChange={this.onChange}
4240                 name="emergencyEmail"
4241                 error={errors.emergencyEmail}
4242             />
4243         </Form.Group>
4244     </Grid.Col>
4245     <Grid.Col sm={12} md={6}>
4246         <Form.Group>
4247             <Form.Label>Contact Relationship</Form.Label>
4248             <Form.Input
4249                 type="text"
4250                 placeholder="Contact Relationship"
4251                 value={emergencyRelationship}
4252                 onChange={this.onChange}
4253                 name="emergencyRelationship"
4254                 error={errors.emergencyRelationship}
4255             />
4256         </Form.Group>
4257     </Grid.Row>
4258 </Card.Body>
4259 <Card.Footer className="text-right">
4260     <Button type="submit" color="primary">
4261         Update Profile
4262     </Button>
4263 </Card.Footer>
4264 </Form>
4265 </Grid.Col>
4266 </Grid.Row>
4267 </Container>
4268     )
4269     </div>
4270   );
4271 }
4272 }
4273 /* istanbul ignore next */
4274 function mapStateToProps(state) {
4275   return {
4276     app: state.app,
4277   };
4278 }
4279 */

```

```

4280
4281 /* istanbul ignore next */
4282 function mapDispatchToProps(dispatch) {
4283   return {
4284     actions: bindActionCreators({ ...actions }, dispatch),
4285   };
4286 }
4287
4288 export default connect(mapStateToProps, mapDispatchToProps)(  

4289   DirectorProfile);
4290 import React, { Component } from 'react';
4291 import PropTypes from 'prop-types';
4292 import { bindActionCreators } from 'redux';
4293 import { connect } from 'react-redux';
4294 import { Link } from 'react-router-dom';
4295 import reactLogo from '../../images/react-logo.svg';
4296 import rekitLogo from '../../images/rekit-logo.svg';
4297 import * as actions from './redux/actions';
4298
4299 export class HomeView extends Component {
4300   static propTypes = {
4301     home: PropTypes.object.isRequired,
4302     actions: PropTypes.object.isRequired,
4303   };
4304   componentDidMount() {
4305     if (!this.props.isAuthenticated) {
4306       this.props.history.push('/login');
4307     } else {
4308       this.props.history.push('/dashboard');
4309     }
4310   }
4311   render() {
4312     return <div className="app-home-view"></div>;
4313   }
4314 }
4315
4316 /* istanbul ignore next */
4317 function mapStateToProps(state) {
4318   return {
4319     home: state.home,
4320   };
4321 }
4322
4323 /* istanbul ignore next */
4324 function mapDispatchToProps(dispatch) {
4325   return {
4326     actions: bindActionCreators({ ...actions }, dispatch),
4327   };
4328 }
4329
4330
4331 export default connect(mapStateToProps, mapDispatchToProps)(HomeView);
4332 import React, { Component } from 'react';
4333 import PropTypes from 'prop-types';
4334 import { bindActionCreators } from 'redux';
4335 import { connect } from 'react-redux';
4336 import * as actions from './redux/actions';
4337 import DHLALogo from '../../images/logo.png';
4338 import { FormCard, FormTextInput, Site, Alert } from 'tabler-react';
4339 import { withRouter } from 'react-router-dom';
4340
4341 function StandaloneFormPage(props) {
4342   return (
4343     <div className="page">
4344       <div className="page-single">
4345         <div className="container">
4346           <div className="row">
4347             <div className="col col-login mx-auto">
4348               <div className="text-center mb-6"></div>
4349               {props.children}
4350             </div>

```

```

4351         </div>
4352     </div>
4353     </div>
4354     </div>
4355   );
4356 }
4357
4358 export class Login extends Component {
4359   static propTypes = {
4360     app: PropTypes.object.isRequired,
4361     actions: PropTypes.object.isRequired,
4362   };
4363
4364   constructor(props) {
4365     super(props);
4366     this.state = { email: '', password: '', errors: {} };
4367     this.onSubmit = this.onSubmit.bind(this);
4368     this.onChange = this.onChange.bind(this);
4369   }
4370
4371   onChange(event) {
4372     this.setState({ [event.target.name]: event.target.value });
4373   }
4374
4375   onSubmit(event) {
4376     event.preventDefault();
4377     const userData = {
4378       email: this.state.email,
4379       password: this.state.password,
4380     };
4381     this.props.actions.loginUser(userData);
4382   }
4383
4384   componentDidMount() {
4385     if (this.props.app.auth.isAuthenticated) {
4386       this.props.history.push('/dashboard');
4387     }
4388   }
4389
4390   componentWillReceiveProps(nextProps) {
4391     if (nextProps.app.auth.isAuthenticated) {
4392       this.props.history.push('/dashboard');
4393     }
4394     if (nextProps.app.errors) {
4395       this.setState({ errors: nextProps.app.errors });
4396     }
4397   }
4398
4399   render() {
4400     const { errors } = this.props.app;
4401     return (
4402       <div className="app-login">
4403         <Site>
4404           <StandaloneFormPage>
4405             <img src={DHLALogo} style={{ marginBottom: '60px' }} alt="logo" />
4406             <FormCard buttonText="Login" title="Login to your account" onSubmit={this.onSubmit}>
4407               {errors.notice && (
4408                 <Alert icon="info" type="warning">
4409                   Message from the Cashier: {errors.notice}
4410                 </Alert>
4411               )}
4412               <FormTextInput
4413                 name="email"
4414                 label="Email Address"
4415                 placeholder="Enter your email address"
4416                 onChange={this.onChange}
4417                 value={this.state.email}
4418                 error={errors.email}
4419               />
4420               <FormTextInput
4421                 name="password"

```

```

4422         type="password"
4423         label="Password"
4424         placeholder="Enter your password"
4425         onChange={this.onChange}
4426         value={this.state.password}
4427         error={errors.password}
4428     />
4429     </FormCard>
4430   <StandaloneFormPage>
4431   </Site>
4432   </div>
4433 );
4434 }
4435 }
4436
4437 /* istanbul ignore next */
4438 function mapStateToProps(state) {
4439   return {
4440     app: state.app,
4441   };
4442 }
4443
4444 /* istanbul ignore next */
4445 function mapDispatchToProps(dispatch) {
4446   return {
4447     actions: bindActionCreators({ ...actions }, dispatch),
4448   };
4449 }
4450
4451 export default withRouter(connect(mapStateToProps, mapDispatchToProps)(
  Login));
4452 import React, { Component, Fragment } from 'react';
4453 import PropTypes from 'prop-types';
4454 import { bindActionCreators } from 'redux';
4455 import { connect } from 'react-redux';
4456 import * as actions from './redux/actions';
4457 import { Site, Form, Grid, Container, RouterContextProvider } from ,
  tabler-react';
4458 import { NavLink, withRouter } from 'react-router-dom';
4459 import DHLALogo from '../../images/logo.png';
4460 import { Modal, message } from 'antd';
4461 import { Spin } from 'antd';
4462 import placeholder from '../../images/placeholder.jpg';
4463 import { Header } from 'tabler-react';
4464 import { getImageUrl } from '../../utils';
4465 import axios from 'axios';
4466
4467 const showNavBarItems = position => {
4468   switch (position) {
4469     case false:
4470       // Navbar items for Administrator
4471       return [
4472         {
4473           value: 'Dashboard',
4474           to: '/dashboard',
4475           icon: 'home',
4476           LinkComponent: withRouter(NavLink),
4477         },
4478       ];
4479     case true:
4480       // Navbar items for Director
4481       return [
4482         {
4483           value: 'Home',
4484           to: '/dashboard',
4485           icon: 'home',
4486           LinkComponent: withRouter(NavLink),
4487         },
4488         {
4489           value: 'Grades',
4490           icon: 'book',
4491           subItems: [

```

```

4492      {
4493        value: 'View Student Records',
4494        to: '/viewstudentrecord',
4495        LinkComponent: withRouter(NavLink),
4496      },
4497    ],
4498  },
4499];
4500 case 2:
4501   // Navbar items for Registrar
4502   return [
4503     {
4504       value: 'Home',
4505       to: '/dashboard',
4506       icon: 'home',
4507       LinkComponent: withRouter(NavLink),
4508     },
4509     {
4510       value: 'Sections',
4511       icon: 'file',
4512       subItems: [
4513         {
4514           value: 'Sections List',
4515           to: '/sectionslist',
4516           LinkComponent: withRouter(NavLink),
4517         },
4518         {
4519           value: 'Manage Students',
4520           to: '/managestudents',
4521           LinkComponent: withRouter(NavLink),
4522         },
4523       ],
4524     },
4525   },
4526   {
4527     value: 'Teachers',
4528     icon: 'user',
4529     subItems: [
4530       {
4531         value: 'Assign Advisory Section',
4532           to: '/assignadvisorysection',
4533           LinkComponent: withRouter(NavLink),
4534         },
4535         {
4536           value: 'Assign Subject Load',
4537           to: '/assignsubjectload',
4538           LinkComponent: withRouter(NavLink),
4539         },
4540       ],
4541     },
4542     {
4543       value: 'Deliberation',
4544       icon: 'file',
4545       subItems: [
4546         {
4547           value: 'Individual Deliberation',
4548             to: '/individualdeliberation',
4549             LinkComponent: withRouter(NavLink),
4550           },
4551           {
4552             value: 'Group Deliberation',
4553               to: '/groupdeliberation',
4554               LinkComponent: withRouter(NavLink),
4555             },
4556       ],
4557     },
4558     {
4559       value: 'Grades',
4560       icon: 'book',
4561       subItems: [
4562         {
4563           value: 'View Student Records',
4564             to: '/viewstudentrecord',

```

```

4564             LinkComponent: withRouter(NavLink),
4565         },
4566     ],
4567   },
4568   {
4569     value: 'School Information',
4570     icon: 'info',
4571   },
4572 ];
4573 case 3:
4574   // Navbar items for Teacher
4575   return [
4576     {
4577       value: 'Home',
4578       to: '/dashboard',
4579       icon: 'home',
4580       LinkComponent: withRouter(NavLink),
4581     },
4582     {
4583       value: 'Subjects',
4584       icon: 'file',
4585       subItems: [
4586         {
4587           value: 'View Subject Load',
4588           to: '/viewsubjectload',
4589           LinkComponent: withRouter(NavLink),
4590         },
4591       ],
4592     },
4593   {
4594     value: 'Adviser',
4595     icon: 'user',
4596     subItems: [
4597       {
4598         value: 'View Student Grades',
4599         to: '/viewadviseegrades',
4600         LinkComponent: withRouter(NavLink),
4601       },
4602       {
4603         value: 'View Report Cards',
4604         to: '/viewreportcards',
4605         LinkComponent: withRouter(NavLink),
4606       },
4607     ],
4608   },
4609 ];
4610 case 4:
4611   // Navbar items for Student
4612   return [
4613     {
4614       value: 'Home',
4615       to: '/dashboard',
4616       icon: 'home',
4617       LinkComponent: withRouter(NavLink),
4618     },
4619     {
4620       value: 'Grade',
4621       icon: 'file',
4622       to: '/viewstudentgrade',
4623       LinkComponent: withRouter(NavLink),
4624     },
4625   ];
4626 case 5:
4627   // Navbar items for Parent
4628   return [
4629     {
4630       value: 'Home',
4631       to: '/dashboard',
4632       icon: 'home',
4633       LinkComponent: withRouter(NavLink),
4634     },
4635     {
4636       value: 'Grade',
4637     }

```

```

4637         icon: 'file',
4638         to: '/viewstudentgrade',
4639         LinkComponent: withRouter(NavLink),
4640       },
4641     ];
4642   case 6:
4643     // Navbar items for Cashier
4644     return [
4645       {
4646         value: 'Home',
4647         to: '/dashboard',
4648         icon: 'home',
4649         LinkComponent: withRouter(NavLink),
4650       },
4651     ];
4652   default:
4653     return [];
4654   }
4655 };
4656
4657 export class NavBar extends Component {
4658   propTypes = {
4659     app: PropTypes.object.isRequired,
4660   };
4661   constructor(props) {
4662     super(props);
4663     this.state = {
4664       firstName: 'NA',
4665       lastName: 'NA',
4666       position: '',
4667       currentPassword: '',
4668       password: '',
4669       password2: '',
4670       errors: {},
4671       showChangePw: false,
4672       showLoading: true,
4673       showLoading2: false,
4674     };
4675     this.logoutClick = this.logoutClick.bind(this);
4676     this.onChange = this.onChange.bind(this);
4677     this.showChangePw = this.showChangePw.bind(this);
4678     this.hideChangePw = this.hideChangePw.bind(this);
4679     this.changePassword = this.changePassword.bind(this);
4680   }
4681
4682   changePassword() {
4683     // this.props.actions.changePassword({
4684     //   currentPassword: this.state.currentPassword,
4685     //   password: this.state.password,
4686     //   password2: this.state.password2,
4687     // });
4688     this.setState({ showLoading2: true });
4689     axios
4690       .post('api/users/changepassword', {
4691         currentPassword: this.state.currentPassword,
4692         password: this.state.password,
4693         password2: this.state.password2,
4694       })
4695       .then(res => {
4696         message.success('You have successfully changed your password!')
4697         ;
4698         this.setState({
4699           showLoading2: false,
4700           showChangePw: false,
4701           errors: {},
4702         });
4703       })
4704       .catch(err => {
4705         this.setState({
4706           showLoading2: false,
4707           errors: err.response.data,
4708         });
4709       });
4710 
```

```

4709     }
4710
4711     onChange(event) {
4712       this.setState({ [event.target.name]: event.target.value });
4713     }
4714     componentWillMount() {
4715       if (Object.keys(this.props.app.profile).length !== 0) {
4716         this.setState({
4717           firstName: this.props.app.profile.firstName,
4718           lastName: this.props.app.profile.lastName,
4719           imageUrl: this.props.app.profile.imageUrl,
4720           showLoading: false,
4721           position: this.props.app.auth.user.position,
4722         });
4723       }
4724     }
4725
4726     componentWillReceiveProps(nextProps) {
4727       this.setState({
4728         firstName: nextProps.app.profile.firstName,
4729         lastName: nextProps.app.profile.lastName,
4730         imageUrl: nextProps.app.profile.imageUrl,
4731         showLoading: false,
4732         position: nextProps.app.auth.user.position,
4733       });
4734     }
4735
4736     showChangePw() {
4737       this.setState({
4738         showChangePw: true,
4739         currentPassword: '',
4740         password: '',
4741         password2: '',
4742         errors: {},
4743       });
4744       this.props.actions.getErrors({});
4745     }
4746
4747     hideChangePw() {
4748       this.setState({
4749         showChangePw: false,
4750         errors: {},
4751       });
4752     }
4753
4754     logoutClick() {
4755       this.props.actions.setLoadingTrue();
4756       this.props.actions.logoutUser().then((window.location.href = '/login'));
4757     }
4758
4759     render() {
4760       const capitalize = string => {
4761         return string.charAt(0).toUpperCase() + string.slice(1).toLowerCase();
4762       };
4763       const { firstName, lastName, imageUrl } = this.state;
4764       const { position } = this.state;
4765       const { errors } = this.state;
4766       const displayPosition = position => {
4767         switch (position) {
4768           case false:
4769             return 'Administrator';
4770           case true:
4771             return 'Director';
4772           case 2:
4773             return 'Registrar';
4774           case 3:
4775             return 'Teacher';
4776           case 4:
4777             return 'Student';
4778           case 5:
4779             return 'Guardian';
4780           case 6:

```

```

4781         return 'Cashier';
4782     default:
4783         return '';
4784     }
4785 };
4786 const accountDropdownProps = {
4787     avatarURL: imageUrl === 'NA' ? placeholder : getImageUrl(imageUrl
4788         ),
4789     name: firstName + ' ' + lastName,
4790     description: displayPosition(position),
4791     options: [
4792         {
4793             icon: 'user',
4794             value: 'Edit Profile',
4795             to: '/profile',
4796             LinkComponent: withRouter(NavLink),
4797         },
4798         {
4799             icon: 'lock',
4800             value: 'Change password',
4801             onClick: this.showChangePw,
4802         },
4803         {
4804             icon: 'log-out',
4805             value: 'Sign Out',
4806             onClick: this.logoutClick,
4807         },
4808     ],
4809 };
4810
4811     return (
4812         <div className="app-nav-bar">
4813             {this.state.showLoading ? (
4814                 '',
4815             ) : (
4816                 <div>
4817                     <Modal
4818                         title="Change Password"
4819                         visible={this.state.showChangePw}
4820                         onOk={this.changePassword}
4821                         confirmLoading={this.state.showLoading2}
4822                         onCancel={this.hideChangePw}
4823                         okText="Change Password"
4824                         cancelText="Close"
4825                     >
4826                         <Spin spinning={this.state.showLoading}>
4827                             <Container>
4828                                 <Grid.Row>
4829                                     <Grid.Col sm={12} md={12}>
4830                                         <Form.Group>
4831                                             <Form.Label>Current Password</Form.Label>
4832                                             <Form.Input
4833                                                 type="password"
4834                                                 name="currentPassword"
4835                                                 placeholder="Current Password"
4836                                                 value={this.state.currentPassword}
4837                                                 error={errors.currentPassword}
4838                                                 onChange={this.onChange}
4839                                         />
4840                                         </Form.Group>
4841                                     </Grid.Col>
4842                                     <Grid.Col sm={12} md={12}>
4843                                         <Form.Group>
4844                                             <Form.Label>New Password</Form.Label>
4845                                             <Form.Input
4846                                                 type="password"
4847                                                 name="password"
4848                                                 placeholder="New Password"
4849                                                 value={this.state.password}
4850                                                 error={errors.password}
4851                                                 onChange={this.onChange}
4852                                         />
4853                                     </Form.Group>
4854                                 </Grid.Col>

```

```

4854
4855     <Grid.Col sm={12} md={12}>
4856         <Form.Group>
4857             <Form.Label>Confirm Password</Form.Label>
4858             <Form.Input
4859                 type="password"
4860                 name="password2"
4861                 placeholder="Confirm Password"
4862                 value={this.state.password2}
4863                 error={errors.password2}
4864                 onChange={this.onChange}
4865             />
4866         </Form.Group>
4867     </Grid.Col>
4868 </Grid.Row>
4869 </Container>
4870 </Modal>
4871 <Site.Wrapper
4872     headerProps={{
4873         href: '/',
4874         alt: 'DHLA Web App',
4875         imageURL: DHLALogo,
4876         accountDropdown: accountDropdownProps,
4877     }}
4878     navProps={{ itemsObjects: showNavBarItems(position) }}
4879     routerContextComponentType={withRouter(
4880         RouterContextProvider)}
4881     footerProps={{
4882         copyright: (
4883             <div>
4884                 Copyright © 2019
4885                 <a href="."> Dee Hwa Liong Academy</a>. All rights
4886                 reserved.
4887             </div>
4888         ),
4889     )}
4890 >
4891     <Container className="fh">{this.props.children}</
4892         Container>
4893     </Site.Wrapper>
4894 </div>
4895 );
4896 }
4897 */
4898 /* istanbul ignore next */
4899 function mapStateToProps(state) {
4900     return {
4901         app: state.app,
4902     };
4903 }
4904 */
4905 /* istanbul ignore next */
4906 function mapDispatchToProps(dispatch) {
4907     return {
4908         actions: bindActionCreators({ ...actions }, dispatch),
4909     };
4910 }
4911 */
4912 export default connect(mapStateToProps, mapDispatchToProps)(NavBar);
4913 import React, { Component } from 'react';
4914 import PropTypes from 'prop-types';
4915 import { bindActionCreators } from 'redux';
4916 import { connect } from 'react-redux';
4917 import * as actions from './redux/actions';
4918 import { Container, Grid, Card, Button, Form, Header } from 'tabler-
4919     react';
4920 import cn from 'classnames';
4921 import placeholder from '../../../../../images/placeholder.jpg';
4922 import { Spin } from 'antd';
4923 import bg from '../../../../../images/BG.png';

```

```

4923 import { getImageUrl } from '../../../../../utils';
4924 function ProfileImage({ avatarURL }) {
4925   return <img className="card-profile-img" alt="Profile" src={avatarURL
4926     } />;
4927 }
4928 function Profile({
4929   className,
4930   children,
4931   name,
4932   avatarURL = '',
4933   twitterURL = '',
4934   backgroundURL = '',
4935   bio,
4936 }) {
4937   const classes = cn('card-profile', className);
4938   return (
4939     <Card className={classes}>
4940       <Card.Header backgroundURL={backgroundURL} />
4941       <Card.Body className="text-center">
4942         <ProfileImage avatarURL={avatarURL} />
4943         <Header.H3 className="mb-3">{name}</Header.H3>
4944         <p className="mb-4">{bio || children}</p>
4945       </Card.Body>
4946     </Card>
4947   );
4948 }
4949
4950 export class ParentDashboard extends Component {
4951   static propTypes = {
4952     app: PropTypes.object.isRequired,
4953     actions: PropTypes.object.isRequired,
4954   };
4955
4956   constructor(props) {
4957     super(props);
4958     this.state = {
4959       isLoading: false,
4960       email: '',
4961       position: '',
4962       firstName: '',
4963       lastName: '',
4964       middleName: '',
4965       suffix: '',
4966       nickname: '',
4967       imageUrl: '',
4968       contactNum: '',
4969       address: '',
4970       province: '',
4971       city: '',
4972       region: '',
4973       zipcode: '',
4974       civilStatus: '',
4975       sex: '',
4976       citizenship: '',
4977       birthDate: 0,
4978       birthPlace: '',
4979       religion: '',
4980       emergencyName: '',
4981       emergencyAddress: '',
4982       emergencyTelephone: '',
4983       emergencyCellphone: '',
4984       emergencyEmail: '',
4985       emergencyRelationship: '',
4986     };
4987   }
4988
4989   componentWillMountReceiveProps(nextProps) {
4990     this.setState({
4991       email: nextProps.app.auth.user.email,
4992       position: nextProps.app.auth.user.position,
4993       firstName: nextProps.app.profile.firstName,
4994       lastName: nextProps.app.profile.lastName,
4995       middleName: nextProps.app.profile.middleName,
4996       suffix: nextProps.app.profile.suffix,
4997       nickname: nextProps.app.profile.nickname,

```

```

4998     imageUrl: nextProps.app.profile.imageUrl,
4999     contactNum: nextProps.app.profile.contactNum,
5000     address: nextProps.app.profile.address,
5001     province: nextProps.app.profile.province,
5002     city: nextProps.app.profile.city,
5003     region: nextProps.app.profile.region,
5004     zipcode: nextProps.app.profile.zipcode,
5005     civilStatus: nextProps.app.profile.civilStatus,
5006     sex: nextProps.app.profile.sex,
5007     citizenship: nextProps.app.profile.citizenship,
5008     birthDate: nextProps.app.profile.birthDate,
5009     birthPlace: nextProps.app.profile.birthPlace,
5010     religion: nextProps.app.profile.religion,
5011     emergencyName: nextProps.app.profile.emergencyName,
5012     emergencyAddress: nextProps.app.profile.emergencyAddress,
5013     emergencyTelephone: nextProps.app.profile.emergencyTelephone,
5014     emergencyCellphone: nextProps.app.profile.emergencyCellphone,
5015     emergencyEmail: nextProps.app.profile.emergencyEmail,
5016     emergencyRelationship: nextProps.app.profile.
5017         emergencyRelationship,
5018   );
5019 }
5020 componentDidMount() {
5021   if (Object.keys(this.props.app.profile).length !== 0) {
5022     this.setState({
5023       email: this.props.app.auth.user.email,
5024       position: this.props.app.auth.user.position,
5025       firstName: this.props.app.profile.firstName,
5026       lastName: this.props.app.profile.lastName,
5027       middleName: this.props.app.profile.middleName,
5028       suffix: this.props.app.profile.suffix,
5029       nickname: this.props.app.profile.nickname,
5030       imageUrl: this.props.app.profile.imageUrl,
5031       contactNum: this.props.app.profile.contactNum,
5032       address: this.props.app.profile.address,
5033       province: this.props.app.profile.province,
5034       city: this.props.app.profile.city,
5035       region: this.props.app.profile.region,
5036       zipcode: this.props.app.profile.zipcode,
5037       civilStatus: this.props.app.profile.civilStatus,
5038       sex: this.props.app.profile.sex,
5039       citizenship: this.props.app.profile.citizenship,
5040       birthDate: this.props.app.profile.birthDate,
5041       birthPlace: this.props.app.profile.birthPlace,
5042       religion: this.props.app.profile.religion,
5043       emergencyName: this.props.app.profile.emergencyName,
5044       emergencyAddress: this.props.app.profile.emergencyAddress,
5045       emergencyTelephone: this.props.app.profile.emergencyTelephone,
5046       emergencyCellphone: this.props.app.profile.emergencyCellphone,
5047       emergencyEmail: this.props.app.profile.emergencyEmail,
5048       emergencyRelationship: this.props.app.profile.
5049         emergencyRelationship,
5050     });
5051   }
5052 }
5053 render() {
5054   const {
5055     email,
5056     position,
5057     firstName,
5058     lastName,
5059     middleName,
5060     suffix,
5061     nickname,
5062     imageUrl,
5063     contactNum,
5064     address,
5065     province,
5066     city,
5067     region,
5068     zipcode,
5069     civilStatus,
5070     sex,
5071     citizenship,
5072     birthPlace,
5073     religion,

```

```

5074     emergencyName ,
5075     emergencyAddress ,
5076     emergencyTelephone ,
5077     emergencyCellphone ,
5078     emergencyEmail ,
5079     emergencyRelationship ,
5080   } = this.state;
5081   const birthDate = new Date(this.state.birthDate);
5082   const { errors } = this.props.app;
5083   const uploadLoading = false;
5084   const displayPosition = position => {
5085     switch (position) {
5086       case false:
5087         return 'Administrator';
5088       case true:
5089         return 'Director';
5090       case 2:
5091         return 'Registrar';
5092       case 3:
5093         return 'Teacher';
5094       case 4:
5095         return 'Student';
5096       case 5:
5097         return 'Guardian';
5098       case 6:
5099         return 'Cashier';
5100     default:
5101       return '';
5102   }
5103 };
5104
5105 const getPHTime = (date = '') => {
5106   const d = date === '' ? new Date() : new Date(date);
5107   const localOffset = d.getTimezoneOffset() * 60000;
5108   const UTC = d.getTime() + localOffset;
5109   const PHT = UTC + 3600000 * 16;
5110   return new Date(PHT);
5111 };
5112 const capitalize = string => {
5113   return string.charAt(0).toUpperCase() + string.slice(1).
5114     toLowerCase();
5115 };
5116 return (
5117   <div className="app-teacher-dashboard my-3 my-md-5">
5118     <Spin spinning={this.props.app.showLoading}>
5119       <Container>
5120         <Grid.Row>
5121           <Grid.Col sm={12} lg={4}>
5122             <Profile
5123               name={`${firstName} ${middleName.charAt(0).
5124                 toUpperCase()} ${lastName}`}
5125               avatarURL={imageUrl === 'NA' ? placeholder :
5126                 getImageUrl(imageUrl)}
5127               backgroundURL={bg}
5128             >
5129               <div>
5130                 <Header.H5>{displayPosition(position)}</Header.H5>
5131               </div>
5132             </Profile>
5133           </Grid.Col>
5134           <Grid.Col xs={12} sm={12} lg={8}>
5135             <Form className="card" onSubmit={this.onSubmit}>
5136               <Card.Body>
5137                 <Card.Title>Account Information</Card.Title>
5138                 <Grid.Row>
5139                   <Grid.Col xs={12} sm={12} md={6}>
5140                     <Form.Group>
5141                       <Form.Label>Email</Form.Label>
5142                         <Form.Input type="email" disabled placeholder
5143                           ="Email" value={email} />
5144                     </Form.Group>
5145                   </Grid.Col>
5146                   <Grid.Col sm={12} md={6}>
5147                     <Form.Group>

```

```

5144 <Form.Label>Position</Form.Label>
5145 <Form.Input
5146   type="text"
5147   placeholder="Position"
5148   disabled
5149   value={displayPosition(position)}
5150   />
5151 </Form.Group>
5152 </Grid.Col>
5153 <Grid.Col sm={12} md={4}>
5154 <Form.Group>
5155   <Form.Label>First Name</Form.Label>
5156   <Form.Input
5157     disabled
5158     type="text"
5159     name="firstName"
5160     placeholder="First Name"
5161     value={firstName}
5162   />
5163 </Form.Group>
5164 </Grid.Col>
5165 <Grid.Col sm={12} md={4}>
5166 <Form.Group>
5167   <Form.Label>Middle Name</Form.Label>
5168   <Form.Input
5169     disabled
5170     type="text"
5171     placeholder="Middle Name"
5172     value={middleName}
5173     name="middleName"
5174   />
5175 </Form.Group>
5176 </Grid.Col>
5177 <Grid.Col sm={12} md={4}>
5178 <Form.Group>
5179   <Form.Label>Last Name</Form.Label>
5180   <Form.Input
5181     type="text"
5182     disabled
5183     placeholder="Last Name"
5184     value={lastName}
5185     name="lastName"
5186   />
5187 </Form.Group>
5188 </Grid.Col>
5189 <Grid.Col sm={12} md={3}>
5190 <Form.Group>
5191   <Form.Label>Nickname</Form.Label>
5192   <Form.Input
5193     type="text"
5194     disabled
5195     placeholder="Nickname"
5196     value={nickname}
5197     name="nickname"
5198   />
5199 </Form.Group>
5200 </Grid.Col>
5201 <Grid.Col sm={12} md={3}>
5202 <Form.Group>
5203   <Form.Label>Suffix</Form.Label>
5204   <Form.Input
5205     type="text"
5206     disabled
5207     placeholder="Suffix"
5208     value={suffix}
5209     name="suffix"
5210   />
5211 </Form.Group>
5212 </Grid.Col>
5213 <Grid.Col sm={12} md={6}>
5214 <Form.Group>
5215   <Form.Label>Contact Number</Form.Label>
5216   <Form.Input
5217     type="text"

```

```

5218             disabled
5219             placeholder="Contact Number"
5220             value={contactNum}
5221             name="contactNum"
5222         />
5223     </Form.Group>
5224   </Grid.Col>
5225   <Grid.Col sm={12} md={12}>
5226     <Form.Group>
5227       <Form.Label>Address</Form.Label>
5228       <Form.Input
5229         type="text"
5230         disabled
5231         placeholder="Home Address"
5232         value={address}
5233         name="address"
5234       />
5235     </Form.Group>
5236   </Grid.Col>
5237   <Grid.Col sm={12} md={4}>
5238     <Form.Group>
5239       <Form.Label>Province</Form.Label>
5240       <Form.Input
5241         type="text"
5242         disabled
5243         placeholder="Province"
5244         value={province}
5245         name="province"
5246       />
5247     </Form.Group>
5248   </Grid.Col>
5249   <Grid.Col sm={12} md={4}>
5250     <Form.Group>
5251       <Form.Label>City</Form.Label>
5252       <Form.Input
5253         type="text"
5254         disabled
5255         placeholder="City"
5256         value={city}
5257         name="city"
5258       />
5259     </Form.Group>
5260   </Grid.Col>
5261   <Grid.Col sm={12} md={2}>
5262     <Form.Group>
5263       <Form.Label>Region</Form.Label>
5264       <Form.Input
5265         type="text"
5266         disabled
5267         placeholder="Region"
5268         value={region}
5269         name="region"
5270       />
5271     </Form.Group>
5272   </Grid.Col>
5273   <Grid.Col sm={12} md={2}>
5274     <Form.Group>
5275       <Form.Label>Zipcode</Form.Label>
5276       <Form.Input
5277         type="text"
5278         disabled
5279         placeholder="Postal Code"
5280         value={zipcode}
5281         name="zipcode"
5282       />
5283     </Form.Group>
5284   </Grid.Col>
5285   <Grid.Col sm={12} md={4}>
5286     <Form.Group>
5287       <Form.Label>Civil Status</Form.Label>
5288       <Form.Input
5289         type="text"
5290         disabled
5291         placeholder="Civil Status"

```

```

5292                     value={civilStatus}
5293                     name="civilStatus"
5294                 />
5295             </Form.Group>
5296         </Grid.Col>
5297         <Grid.Col sm={12} md={2}>
5298             <Form.Group>
5299                 <Form.Label>Sex</Form.Label>
5300                 <Form.Input
5301                     type="text"
5302                     disabled
5303                     placeholder="Sex"
5304                     value={sex}
5305                     name="sex"
5306                 />
5307             </Form.Group>
5308         </Grid.Col>
5309         <Grid.Col sm={12} md={6}>
5310             <Form.Group>
5311                 <Form.Label>Citizenship</Form.Label>
5312                 <Form.Input
5313                     disabled
5314                     type="text"
5315                     placeholder="Citizenship"
5316                     value={citizenship}
5317                     name="citizenship"
5318                 />
5319             </Form.Group>
5320         </Grid.Col>
5321         <Grid.Col sm={12} md={4}>
5322             <Form.Group>
5323                 <Form.Label>Birth Date</Form.Label>
5324                 <Form.Input
5325                     disabled
5326                     type="text"
5327                     placeholder="Birth date"
5328                     value={birthDate}
5329                 ></Form.Input>
5330             </Form.Group>
5331         </Grid.Col>
5332         <Grid.Col sm={12} md={4}>
5333             <Form.Group>
5334                 <Form.Label>Birth Place</Form.Label>
5335                 <Form.Input
5336                     disabled
5337                     type="text"
5338                     placeholder="Birth Place"
5339                     value={birthPlace}
5340                     name="birthPlace"
5341                 />
5342             </Form.Group>
5343         </Grid.Col>
5344         <Grid.Col sm={12} md={4}>
5345             <Form.Group>
5346                 <Form.Label>Religion</Form.Label>
5347                 <Form.Input
5348                     type="text"
5349                     disabled
5350                     placeholder="Religion"
5351                     value={religion}
5352                     name="religion"
5353                 />
5354             </Form.Group>
5355         </Grid.Col>
5356     </Grid.Row>
5357 </Card.Body>
5358 <Card.Body>
5359     <Card.Title>Emergency Contact Information</Card.
5360     Title>
5361     <Grid.Row>
5362         <Grid.Col sm={12} md={6}>
5363             <Form.Group>
5364                 <Form.Label>Contact Person</Form.Label>

```

```

5365           type="text"
5366           disabled
5367           placeholder="Contact Person"
5368           value={emergencyName}
5369           name="emergencyName"
5370       />
5371   </Form.Group>
5372 </Grid.Col>
5373 <Grid.Col sm={12} md={6}>
5374   <Form.Group>
5375     <Form.Label>Contact Address</Form.Label>
5376     <Form.Input
5377       type="text"
5378       disabled
5379       placeholder="Contact Address"
5380       value={emergencyAddress}
5381       name="emergencyAddress"
5382   />
5383 </Form.Group>
5384 </Grid.Col>
5385 <Grid.Col sm={12} md={6}>
5386   <Form.Group>
5387     <Form.Label>Contact Telephone No.</Form.Label>
5388   >
5389     <Form.Input
5390       type="text"
5391       disabled
5392       placeholder="Contact Telephone No."
5393       value={emergencyTelephone}
5394       name="emergencyTelephone"
5395   />
5396 </Form.Group>
5397 </Grid.Col>
5398 <Grid.Col sm={12} md={6}>
5399   <Form.Group>
5400     <Form.Label>Contact Cellphone No.</Form.Label>
5401   >
5402     <Form.Input
5403       type="text"
5404       disabled
5405       placeholder="Contact Cellphone No."
5406       value={emergencyCellphone}
5407       name="emergencyCellphone"
5408   />
5409 </Form.Group>
5410 </Grid.Col>
5411 <Grid.Col sm={12} md={6}>
5412   <Form.Group>
5413     <Form.Label>Contact Email</Form.Label>
5414     <Form.Input
5415       disabled
5416       type="text"
5417       placeholder="Contact Email"
5418       value={emergencyEmail}
5419       name="emergencyEmail"
5420   />
5421 </Form.Group>
5422 </Grid.Col>
5423 <Grid.Col sm={12} md={6}>
5424   <Form.Group>
5425     <Form.Label>Contact Relationship</Form.Label>
5426     <Form.Input
5427       type="text"
5428       disabled
5429       placeholder="Contact Relationship"
5430       value={emergencyRelationship}
5431       name="emergencyRelationship"
5432   />
5433 </Form.Group>
5434 </Grid.Col>
5435 </Grid.Row>
5436 </Card.Body>
5437 </Form>
5438 </Grid.Col>

```

```

5437             </Grid.Row>
5438         </Container>
5439     </Spin>
5440   </div>
5441 );
5442 }
5443 }
5444 */
5445 /* istanbul ignore next */
5446 function mapStateToProps(state) {
5447   return {
5448     app: state.app,
5449   };
5450 }
5451 */
5452 /* istanbul ignore next */
5453 function mapDispatchToProps(dispatch) {
5454   return {
5455     actions: bindActionCreators({ ...actions }, dispatch),
5456   };
5457 }
5458 */
5459 export default connect(mapStateToProps, mapDispatchToProps)(
  ParentDashboard);
5460 import React, { Component } from 'react';
5461 import PropTypes from 'prop-types';
5462 import { bindActionCreators } from 'redux';
5463 import { connect } from 'react-redux';
5464 import * as actions from './redux/actions';
5465 import moment from 'moment';
5466 import axios from 'axios';
5467 import { Container, Grid, Card, Button, Form, Header, List } from 'tabler-react';
5468 import { Alert, Upload, message } from 'antd';
5469 import cn from 'classnames';
5470 import bg from '../images/BG.png';
5471 import { getImageUrl, getLocalUrl } from '../utils';
5472 import placeholder from '../images/placeholder.jpg';
5473 function ProfileImage({ avatarURL }) {
5474   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
5475 }
5476 */
5477 function Profile({
5478   className,
5479   children,
5480   name,
5481   avatarURL = '',
5482   twitterURL = '',
5483   backgroundURL = '',
5484   bio,
5485 }) {
5486   const classes = cn('card-profile', className);
5487   return (
5488     <Card className={classes}>
5489       <Card.Header backgroundURL={backgroundURL} />
5490       <Card.Body className="text-center">
5491         <ProfileImage avatarURL={avatarURL} />
5492         <Header.H3 className="mb-3">{name}</Header.H3>
5493         <p className="mb-4">{bio} || {children}</p>
5494       </Card.Body>
5495     </Card>
5496   );
5497 }
5498 export class ParentProfile extends Component {
5499   static propTypes = {
5500     app: PropTypes.object.isRequired,
5501     actions: PropTypes.object.isRequired,
5502   };
5503 }
5504 constructor(props) {
5505   super(props);
5506   this.state = {

```

```

5507     loading: false,
5508     email: '',
5509     position: '',
5510     firstName: '',
5511     lastName: '',
5512     middleName: '',
5513     suffix: '',
5514     nickname: '',
5515     imageUrl: '',
5516     contactNum: '',
5517     address: '',
5518     province: '',
5519     city: '',
5520     region: '',
5521     zipcode: '',
5522     civilStatus: '',
5523     sex: '',
5524     citizenship: '',
5525     birthDate: new Date(),
5526     birthPlace: '',
5527     religion: '',
5528     emergencyName: '',
5529     emergencyAddress: '',
5530     emergencyTelephone: '',
5531     emergencyCellphone: '',
5532     emergencyEmail: '',
5533     emergencyRelationship: '',
5534   );
5535
5536   this.onSubmit = this.onSubmit.bind(this);
5537   this.onChange = this.onChange.bind(this);
5538   this.onDateChange = this.onDateChange.bind(this);
5539 }
5540 componentWillReceiveProps(nextProps) {
5541   if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
5542       .length == 0)
5543     this.setState({
5544       email: nextProps.app.auth.user.email,
5545       position: nextProps.app.auth.user.position,
5546       firstName: nextProps.app.profile.firstName,
5547       lastName: nextProps.app.profile.lastName,
5548       middleName: nextProps.app.profile.middleName,
5549       suffix: nextProps.app.profile.suffix,
5550       nickname: nextProps.app.profile.nickname,
5551       imageUrl: nextProps.app.profile.imageUrl,
5552       contactNum: nextProps.app.profile.contactNum,
5553       address: nextProps.app.profile.address,
5554       province: nextProps.app.profile.province,
5555       city: nextProps.app.profile.city,
5556       region: nextProps.app.profile.region,
5557       zipcode: nextProps.app.profile.zipcode,
5558       civilStatus: nextProps.app.profile.civilStatus,
5559       sex: nextProps.app.profile.sex,
5560       citizenship: nextProps.app.profile.citizenship,
5561       birthDate: nextProps.app.profile.birthDate,
5562       birthPlace: nextProps.app.profile.birthPlace,
5563       religion: nextProps.app.profile.religion,
5564       emergencyName: nextProps.app.profile.emergencyName,
5565       emergencyAddress: nextProps.app.profile.emergencyAddress,
5566       emergencyTelephone: nextProps.app.profile.emergencyTelephone,
5567       emergencyCellphone: nextProps.app.profile.emergencyCellphone,
5568       emergencyEmail: nextProps.app.profile.emergencyEmail,
5569       emergencyRelationship: nextProps.app.profile.
5570         emergencyRelationship,
5571       loading: false,
5572     });
5573   onChange(event) {
5574     this.setState({ [event.target.name]: event.target.value });
5575   }
5576   onDateChange(event) {
5577     this.setState({ birthDate: event });
5578   }
5579 }
5580

```

```

5581     onSubmit(event) {
5582         event.preventDefault();
5583         const {
5584             firstName,
5585             lastName,
5586             middleName,
5587             suffix,
5588             nickname,
5589             contactNum,
5590             address,
5591             province,
5592             city,
5593             region,
5594             zipcode,
5595             civilStatus,
5596             sex,
5597             citizenship,
5598             birthDate,
5599             birthPlace,
5600             religion,
5601             emergencyName,
5602             emergencyAddress,
5603             emergencyTelephone,
5604             emergencyCellphone,
5605             emergencyEmail,
5606             emergencyRelationship,
5607         } = this.state;
5608         const profileData = {
5609             firstName,
5610             lastName,
5611             middleName,
5612             suffix,
5613             nickname,
5614             contactNum,
5615             address,
5616             province,
5617             city,
5618             region,
5619             zipcode,
5620             civilStatus,
5621             sex,
5622             citizenship,
5623             birthDate,
5624             birthPlace,
5625             religion,
5626             emergencyName,
5627             emergencyAddress,
5628             emergencyTelephone,
5629             emergencyCellphone,
5630             emergencyEmail,
5631             emergencyRelationship,
5632         };
5633         console.log(profileData);
5634         this.props.actions.updateParentProfile(profileData);
5635     }
5636
5637     render() {
5638         const {
5639             email,
5640             position,
5641             firstName,
5642             lastName,
5643             middleName,
5644             suffix,
5645             nickname,
5646             imageUrl,
5647             contactNum,
5648             address,
5649             province,
5650             city,
5651             region,
5652             zipcode,
5653             civilStatus,
5654             sex,
5655             citizenship,
5656             birthPlace,
5657             religion,
5658             emergencyName,

```

```

5659         emergencyAddress ,
5660         emergencyTelephone ,
5661         emergencyCellphone ,
5662         emergencyEmail ,
5663         emergencyRelationship ,
5664     } = this.state;
5665     const birthDate = new Date(this.state.birthDate);
5666     const defaultDate = birthDate.toISOString();
5667     const { errors } = this.props.app;
5668     const uploadLoading = false;
5669     const displayPosition = position => {
5670         switch (position) {
5671             case false:
5672                 return 'Administrator';
5673             case true:
5674                 return 'Director';
5675             case 2:
5676                 return 'Registrar';
5677             case 3:
5678                 return 'Teacher';
5679             case 4:
5680                 return 'Student';
5681             case 5:
5682                 return 'Guardian';
5683             default:
5684                 return '';
5685         }
5686     };
5687     const capitalize = string => {
5688         return string.charAt(0).toUpperCase() + string.slice(1).
5689             toLowerCase();
5690     };
5691     return (
5692         <div className="app-admin-profile my-3 my-md-5">
5693             {this.state.loading ? (
5694                 '',
5695             ) : (
5696                 <Container>
5697                     <Grid.Row>
5698                         <Grid.Col sm={12} lg={4}>
5699                             <Profile
5700                                 name={`${firstName} ${middleName.charAt(0).
5701                                     toUpperCase()} ${lastName}`}
5702                                 avatarURL={imageUrl == 'NA' ? placeholder :
5703                                     getImageUrl(imageUrl)}
5704                                 backgroundURL={bg}
5705                         >
5706                             <Upload
5707                                 name="file"
5708                                 multiple={false}
5709                                 accept=".png, .jpg"
5710                                 customRequest={({req, uploadLoading}) => {
5711                                     // Axios Photo Upload
5712                                     this.props.actions.setLoading(true);
5713                                     const bodyFormData = new FormData();
5714                                     bodyFormData.set('file', req.file);
5715                                     axios({
5716                                         method: 'post',
5717                                         url: 'api/users/upload',
5718                                         data: bodyFormData,
5719                                     })
5720                                         .then(res => {
5721                                             req.onSuccess();
5722                                             this.props.actions.setLoading(false);
5723                                             message.success('Uploaded Successfully!').
5724                                                 then(() => {
5725                                                 window.location.href = '/profile';
5726                                             });
5727                                         })
5728                                         .catch(err => {
5729                                             req.onError();
5730                                             this.props.actions.setLoading(false);
5731                                             message.error('Upload failed!');
5732                                         });
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
5999

```

```

5729      }
5730    >
5731      <div>
5732        <Header.H4>{displayPosition(position)}</Header.H4
5733      >
5734      </div>
5735      <Button loading={this.props.app.showLoading} icon="upload" pill color="primary">
5736        Upload Image
5737      </Button>
5738      </Upload>
5739    </Profile>
5740  </Grid.Col>
5741  <Grid.Col xs={12} sm={12} lg={8}>
5742    <Form className="card" onSubmit={this.onSubmit}>
5743      <Card.Body>
5744        <Card.Title>Edit Profile</Card.Title>
5745        <Grid.Row>
5746          <Grid.Col xs={12} sm={12} md={6}>
5747            <Form.Group>
5748              <Form.Label>Email</Form.Label>
5749              <Form.Input
5750                type="email"
5751                disabled
5752                placeholder="Email"
5753                value={email}
5754                error={errors.email}
5755              />
5756            </Form.Group>
5757          </Grid.Col>
5758          <Grid.Col sm={12} md={6}>
5759            <Form.Group>
5760              <Form.Label>Position</Form.Label>
5761              <Form.Input
5762                type="text"
5763                placeholder="Position"
5764                disabled
5765                value={displayPosition(position)}
5766                error={errors.position}
5767              />
5768            </Form.Group>
5769          </Grid.Col>
5770          <Grid.Col sm={12} md={4}>
5771            <Form.Group>
5772              <Form.Label>First Name</Form.Label>
5773              <Form.Input
5774                type="text"
5775                name="firstName"
5776                placeholder="First Name"
5777                value={firstName}
5778                onChange={this.onChange}
5779                error={errors.firstName}
5780              />
5781            </Form.Group>
5782          </Grid.Col>
5783          <Grid.Col sm={12} md={4}>
5784            <Form.Group>
5785              <Form.Label>Middle Name</Form.Label>
5786              <Form.Input
5787                type="text"
5788                placeholder="Middle Name"
5789                value={middleName}
5790                onChange={this.onChange}
5791                name="middleName"
5792                error={errors.middleName}
5793              />
5794            </Form.Group>
5795          </Grid.Col>
5796          <Grid.Col sm={12} md={4}>
5797            <Form.Group>
5798              <Form.Label>Last Name</Form.Label>
5799              <Form.Input
5800                type="text"
                placeholder="Last Name"

```

```

5801           value={lastName}
5802           onChange={this.onChange}
5803           name="lastName"
5804           error={errors.lastName}
5805         />
5806       </Form.Group>
5807     </Grid.Col>
5808   <Grid.Col sm={12} md={3}>
5809     <Form.Group>
5810       <Form.Label>Nickname</Form.Label>
5811       <Form.Input
5812         type="text"
5813         placeholder="Nickname"
5814         value={nickname}
5815         onChange={this.onChange}
5816         name="nickname"
5817         error={errors.nickname}
5818       />
5819     </Form.Group>
5820   </Grid.Col>
5821   <Grid.Col sm={12} md={3}>
5822     <Form.Group>
5823       <Form.Label>Suffix</Form.Label>
5824       <Form.Input
5825         type="text"
5826         placeholder="Suffix"
5827         value={suffix}
5828         onChange={this.onChange}
5829         name="suffix"
5830         error={errors.suffix}
5831       />
5832     </Form.Group>
5833   </Grid.Col>
5834   <Grid.Col sm={12} md={6}>
5835     <Form.Group>
5836       <Form.Label>Contact Number</Form.Label>
5837       <Form.Input
5838         type="text"
5839         placeholder="Contact Number"
5840         value={contactNum}
5841         onChange={this.onChange}
5842         name="contactNum"
5843         error={errors.contactNum}
5844       />
5845     </Form.Group>
5846   </Grid.Col>
5847   <Grid.Col sm={12} md={12}>
5848     <Form.Group>
5849       <Form.Label>Address</Form.Label>
5850       <Form.Input
5851         type="text"
5852         placeholder="Home Address"
5853         value={address}
5854         onChange={this.onChange}
5855         name="address"
5856         error={errors.address}
5857       />
5858     </Form.Group>
5859   </Grid.Col>
5860   <Grid.Col sm={12} md={4}>
5861     <Form.Group>
5862       <Form.Label>Province</Form.Label>
5863       <Form.Input
5864         type="text"
5865         placeholder="Province"
5866         value={province}
5867         onChange={this.onChange}
5868         name="province"
5869         error={errors.province}
5870       />
5871     </Form.Group>
5872   </Grid.Col>
5873   <Grid.Col sm={12} md={4}>

```

```

5874 <Form.Group>
5875   <Form.Label>City</Form.Label>
5876   <Form.Input
5877     type="text"
5878     placeholder="City"
5879     value={city}
5880     onChange={this.onChange}
5881     name="city"
5882     error={errors.city}
5883   />
5884 </Form.Group>
5885 </Grid.Col>
5886 <Grid.Col sm={12} md={2}>
5887   <Form.Group>
5888     <Form.Label>Region</Form.Label>
5889     <Form.Input
5890       type="text"
5891       placeholder="Region"
5892       value={region}
5893       onChange={this.onChange}
5894       name="region"
5895       error={errors.region}
5896     />
5897   </Form.Group>
5898 </Grid.Col>
5899 <Grid.Col sm={12} md={2}>
5900   <Form.Group>
5901     <Form.Label>Zipcode</Form.Label>
5902     <Form.Input
5903       type="text"
5904       placeholder="Postal Code"
5905       value={zipcode}
5906       onChange={this.onChange}
5907       name="zipcode"
5908       error={errors.zipcode}
5909     />
5910   </Form.Group>
5911 </Grid.Col>
5912 <Grid.Col sm={12} md={4}>
5913   <Form.Group>
5914     <Form.Label>Civil Status</Form.Label>
5915     <Form.Select
5916       value={civilStatus}
5917       onChange={this.onChange}
5918       name="civilStatus"
5919     >
5920       <option>SINGLE</option>
5921       <option>MARRIED</option>
5922       <option>WIDOWED</option>
5923       <option>OTHERS</option>
5924     </Form.Select>
5925   </Form.Group>
5926 </Grid.Col>
5927 <Grid.Col sm={12} md={2}>
5928   <Form.Group>
5929     <Form.Label>Sex</Form.Label>
5930     <Form.Select value={sex} onChange={this.
5931       onChange} name="sex">
5932       <option>M</option>
5933       <option>F</option>
5934     </Form.Select>
5935   </Form.Group>
5936 </Grid.Col>
5937 <Grid.Col sm={12} md={6}>
5938   <Form.Group>
5939     <Form.Label>Citizenship</Form.Label>
5940     <Form.Input
5941       type="text"
5942       placeholder="Citizenship"
5943       value={citizenship}
5944       onChange={this.onChange}
5945       name="citizenship"
       error={errors.citizenship}

```

```

5946          />
5947      </Form.Group>
5948  </Grid.Col>
5949  <Grid.Col sm={12} md={4}>
5950      <Form.Group>
5951          <Form.Label>Birth Date</Form.Label>
5952          <Form.DatePicker
5953              defaultValue={new Date(defaultDate)}
5954              format="mm/dd/yyyy"
5955              onChange={this.onDateChange}
5956              name="birthDate"
5957              maxYear={2020}
5958              minYear={1897}
5959              monthLabels={[
5960                  'January',
5961                  'February',
5962                  'March',
5963                  'April',
5964                  'May',
5965                  'June',
5966                  'July',
5967                  'August',
5968                  'September',
5969                  'October',
5970                  'November',
5971                  'December',
5972              ]}
5973          ></Form.DatePicker>
5974      </Form.Group>
5975  </Grid.Col>
5976  <Grid.Col sm={12} md={4}>
5977      <Form.Group>
5978          <Form.Label>Birth Place</Form.Label>
5979          <Form.Input
5980              type="text"
5981              placeholder="Birth Place"
5982              value={birthPlace}
5983              onChange={this.onChange}
5984              name="birthPlace"
5985              error={errors.birthPlace}
5986          />
5987      </Form.Group>
5988  </Grid.Col>
5989  <Grid.Col sm={12} md={4}>
5990      <Form.Group>
5991          <Form.Label>Religion</Form.Label>
5992          <Form.Input
5993              type="text"
5994              placeholder="Religion"
5995              value={religion}
5996              onChange={this.onChange}
5997              name="religion"
5998              error={errors.religion}
5999          />
6000      </Form.Group>
6001  </Grid.Col>
6002  </Grid.Row>
6003  </Card.Body>
6004  <Card.Body>
6005      <Card.Title>Emergency Contact Information</Card.
6006          Title>
6007      <Grid.Row>
6008          <Grid.Col sm={12} md={6}>
6009              <Form.Group>
6010                  <Form.Label>Contact Person</Form.Label>
6011                  <Form.Input
6012                      type="text"
6013                      placeholder="Contact Person"
6014                      value={emergencyName}
6015                      onChange={this.onChange}
6016                      name="emergencyName"
6017                      error={errors.emergencyName}
6018          />
6019      </Form.Group>

```

```

6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090

```

```

        </Grid.Col>
        <Grid.Col sm={12} md={6}>
            <Form.Group>
                <Form.Label>Contact Address</Form.Label>
                <Form.Input
                    type="text"
                    placeholder="Contact Address"
                    value={emergencyAddress}
                    onChange={this.onChange}
                    name="emergencyAddress"
                    error={errors.emergencyAddress}
                />
            </Form.Group>
        </Grid.Col>
        <Grid.Col sm={12} md={6}>
            <Form.Group>
                <Form.Label>Contact Telephone No.</Form.Label>
            <Form.Input
                type="text"
                placeholder="Contact Telephone No."
                value={emergencyTelephone}
                onChange={this.onChange}
                name="emergencyTelephone"
                error={errors.emergencyTelephone}
            />
            </Form.Group>
        </Grid.Col>
        <Grid.Col sm={12} md={6}>
            <Form.Group>
                <Form.Label>Contact Cellphone No.</Form.Label>
            <Form.Input
                type="text"
                placeholder="Contact Cellphone No."
                value={emergencyCellphone}
                onChange={this.onChange}
                name="emergencyCellphone"
                error={errors.emergencyCellphone}
            />
            </Form.Group>
        </Grid.Col>
        <Grid.Col sm={12} md={6}>
            <Form.Group>
                <Form.Label>Contact Email</Form.Label>
                <Form.Input
                    type="text"
                    placeholder="Contact Email"
                    value={emergencyEmail}
                    onChange={this.onChange}
                    name="emergencyEmail"
                    error={errors.emergencyEmail}
                />
            </Form.Group>
        </Grid.Col>
        <Grid.Col sm={12} md={6}>
            <Form.Group>
                <Form.Label>Contact Relationship</Form.Label>
                <Form.Input
                    type="text"
                    placeholder="Contact Relationship"
                    value={emergencyRelationship}
                    onChange={this.onChange}
                    name="emergencyRelationship"
                    error={errors.emergencyRelationship}
                />
            </Form.Group>
        </Grid.Col>
    </Grid.Row>
</Card.Body>
<Card.Footer className="text-right">
    <Button type="submit" color="primary">
        Update Profile
    </Button>

```

```

6091             </Card.Footer>
6092         </Form>
6093     </Grid.Col>
6094   </Grid.Row>
6095 </Container>
6096   )
6097 </div>
6098 );
6099 }
6100 }
6101
/* istanbul ignore next */
6102 function mapStateToProps(state) {
6103   return {
6104     app: state.app,
6105   };
6106 }
6107
/* istanbul ignore next */
6108 function mapDispatchToProps(dispatch) {
6109   return {
6110     actions: bindActionCreators({ ...actions }, dispatch),
6111   };
6112 }
6113
6114
6115 export default connect(mapStateToProps, mapDispatchToProps)(
6116   ParentProfile);
6117 import React, { Component } from 'react';
6118 import PropTypes from 'prop-types';
6119 import { bindActionCreators } from 'redux';
6120 import { connect } from 'react-redux';
6121 import * as actions from './redux/actions';
6122 import NavBar from './NavBar';
6123 import AdminProfile from './AdminProfile';
6124 import DirectorProfile from './DirectorProfile';
6125 import RegistrarProfile from './RegistrarProfile';
6126 import TeacherProfile from './TeacherProfile';
6127 import StudentProfile from './StudentProfile';
6128 import ParentProfile from './ParentProfile';
6129 import CashierProfile from './CashierProfile';
6130
6131 function ProfileComponent(props) {
6132   const { position } = props;
6133   switch (position) {
6134     case false:
6135       return <AdminProfile />;
6136     case true:
6137       return <DirectorProfile />;
6138     case 2:
6139       return <RegistrarProfile />;
6140     case 3:
6141       return <TeacherProfile />;
6142     case 4:
6143       return <StudentProfile />;
6144     case 5:
6145       return <ParentProfile />;
6146     case 6:
6147       return <CashierProfile />;
6148     default:
6149       return <div></div>;
6150   }
6151 }
6152 export class ProfileView extends Component {
6153   static propTypes = {
6154     app: PropTypes.object.isRequired,
6155     actions: PropTypes.object.isRequired,
6156   };
6157
6158   componentDidMount() {
6159     if (!this.props.app.auth.isAuthenticated) {
6160       this.props.history.push('/login');
6161     }

```

```

6162     }
6163
6164     componentWillReceiveProps(nextProps) {
6165       if (!nextProps.app.auth.isAuthenticated) {
6166         this.props.history.push('/login');
6167       }
6168     }
6169
6170     render() {
6171       return (
6172         <div className="app-profile-view">
6173           <NavBar>
6174             <ProfileComponent position={this.props.app.auth.user.position}>
6175               </ProfileComponent>
6176             </NavBar>
6177           </div>
6178         );
6179     }
6180   }
6181   /* istanbul ignore next */
6182   function mapStateToProps(state) {
6183     return {
6184       app: state.app,
6185     };
6186   }
6187
6188   /* istanbul ignore next */
6189   function mapDispatchToProps(dispatch) {
6190     return {
6191       actions: bindActionCreators({ ...actions }, dispatch),
6192     };
6193   }
6194
6195   export default connect(mapStateToProps, mapDispatchToProps)(ProfileView);
6196
6197   import React, { Component } from 'react';
6198   import PropTypes from 'prop-types';
6199   import { bindActionCreators } from 'redux';
6200   import { connect } from 'react-redux';
6201   import * as actions from './redux/actions';
6202   import { Card, Button, Grid, Avatar, Table, Form, Header, Container } from 'tabler-react';
6203   import axios from 'axios';
6204   import { Pagination, Spin, Tooltip } from 'antd';
6205   import {
6206     Modal,
6207     Popconfirm,
6208     Select,
6209     Search,
6210     Breadcrumb,
6211     AutoComplete,
6212     Input,
6213     message,
6214     Descriptions,
6215   } from 'antd';
6216   import placeholder from '../../images/placeholder.jpg';
6217   import { getImageUrl } from '../../utils';
6218   const { Option } = Select;
6219
6220   export class RegistrarAddNewLoad extends Component {
6221     static propTypes = {
6222       app: PropTypes.object.isRequired,
6223       actions: PropTypes.object.isRequired,
6224     };
6225
6226     constructor(props) {
6227       super(props);
6228       this.state = {
6229         isLoadingTable: false,
6230         isLoading: true,
6231         sectionOption: [],
6232         isSectionLoading: true,

```

```

6232     subjectOption: [],
6233     isSubjectLoading: true,
6234     gradeLevel: 'N',
6235     noSectionData: true,
6236     noSubjectData: true,
6237     selectedSubject: 0,
6238     selectedSection: 0,
6239     sectionName: 'NA',
6240     subjectName: 'NA',
6241     studentData: [],
6242     page: 1,
6243     pageSize: 100,
6244     keyword: '',
6245     selectedKey: -1,
6246     options: [],
6247     optionLoading: false,
6248   );
6249
6250   this.onChangeGradeLevel = this.onChangeGradeLevel.bind(this);
6251   this.onChangeSection = this.onChangeSection.bind(this);
6252   this.onChangeSubject = this.onChangeSubject.bind(this);
6253   this.importStudent = this.importStudent.bind(this);
6254   this.removeStudent = this.removeStudent.bind(this);
6255   this.handleSearch = this.handleSearch.bind(this);
6256   this.onSelect = this.onSelect.bind(this);
6257   this.addStudent = this.addStudent.bind(this);
6258 }
6259
6260 addStudent() {
6261   let arr = this.state.studentData;
6262   let add = this.state.options.find(value => value.key == this.state.
6263     selectedKey);
6264   if (arr.findIndex(val => val.key == this.state.selectedKey) == -1)
6265   {
6266     arr.push(add);
6267     this.setState({ studentData: arr });
6268   } else {
6269     message.error('Student already in the list');
6270   }
6271   onSelect(key) {
6272     this.setState({ selectedKey: key });
6273   }
6274
6275   handleSearch(query) {
6276     this.setState({ selectedKey: -1 });
6277     this.setState({
6278       options: [],
6279       optionLoading: true,
6280     });
6281     axios
6282       .post('api/registrar/searchenrolled', { keyword: query })
6283       .then(res => {
6284         if (query == '') {
6285           this.setState({ options: [], optionLoading: false });
6286         } else {
6287           this.setState({ options: res.data.studentList, optionLoading:
6288             false });
6289         }
6290       })
6291       .catch(err => {
6292         this.setState({
6293           options: [],
6294           optionLoading: false,
6295         });
6296       });
6297
6298   removeStudent(key) {
6299     const index = this.state.studentData.findIndex(value => value.key
6300       == key);
6301     if (index != -1) {

```

```

6301         let arr = this.state.studentData;
6302         arr.splice(index, 1);
6303         this.setState({ studentData: arr });
6304     }
6305   }
6306
6307   importStudent() {
6308     this.setState({ isLoadingTable: true });
6309     axios
6310       .post('api/registrar/getcurrentenrolled', {
6311         page: this.state.page,
6312         pageSize: this.state.pageSize,
6313         sectionID: this.state.selectedSection,
6314       })
6315       .then(res => {
6316         this.setState({ isLoadingTable: false, studentData: res.data.
6317           studentList });
6318       })
6319       .catch(err => {
6320         this.setState({ isLoadingTable: false, studentData: [] });
6321       });
6322   }
6323
6324   onChangeSection(value) {
6325     this.setState({ selectedSection: value });
6326     const sectionName =
6327       this.state.sectionOption.length != 0
6328         ? this.state.sectionOption.find(val => val.sectionID == value).
6329           sectionName
6330         : '';
6331     this.setState({ sectionName });
6332   }
6333
6334   onChangeSubject(value) {
6335     this.setState({ selectedSubject: value });
6336     const subjectName =
6337       this.state.subjectOption.length != 0
6338         ? this.state.subjectOption.find(val => val.subjectID == value).
6339           subjectName
6340         : '';
6341     this.setState({ subjectName });
6342   }
6343
6344   onChangeGradeLevel(value) {
6345     this.setState({ studentData: [] });
6346     this.setState({ gradeLevel: value });
6347     this.setState({ isSubjectLoading: true, isSectionLoading: true });
6348     axios
6349       .post('api/registrar/getsubjectbygradelevel', { gradeLevel: value
6350         })
6351       .then(res => {
6352         this.setState({ isSubjectLoading: false });
6353         this.setState({ subjectOption: res.data.subjectsList });
6354         this.setState({ noSubjectData: false });
6355         this.setState({ selectedSubject: res.data.subjectsList[0].
6356           subjectID });
6357         this.setState({ subjectName: res.data.subjectsList[0].
6358           subjectName });
6359       })
6360       .catch(err => {
6361         this.setState({ subjectOption: [] });
6362         this.setState({ isSubjectLoading: false });
6363         this.setState({ noSubjectData: true });
6364         this.setState({ selectedSubject: 0 });
6365         this.setState({ subjectName: 'NA' });
6366       });
6367     axios
6368       .post('api/registrar/getsectionbygradelevel', { gradeLevel: value
6369         })
6370       .then(res => {
6371         this.setState({ isSectionLoading: false });
6372         this.setState({ sectionOption: res.data.sectionsList });

```

```

6366     this.setState({ noSectionData: false });
6367     this.setState({ selectedSection: res.data.sectionsList[0].
6368         sectionID });
6369     this.setState({ sectionName: res.data.sectionsList[0].
6370         sectionName });
6371   })
6372   .catch(err => {
6373     this.setState({ isSectionLoading: false });
6374     this.setState({ sectionOption: [] });
6375     this.setState({ noSectionData: true });
6376     this.setState({ selectedSection: 0 });
6377     this.setState({ sectionName: 'NA' });
6378   });
6379
6380 componentDidMount() {
6381   axios
6382     .post('api/registrar/getsubjectbygradelevel', { gradeLevel: this.
6383       state.gradeLevel })
6384     .then(res => {
6385       this.setState({ isSubjectLoading: false });
6386       this.setState({ subjectOption: res.data.subjectsList });
6387       this.setState({ noSubjectData: false });
6388       this.setState({ selectedSubject: res.data.subjectsList[0].
6389         subjectID });
6390       this.setState({ subjectName: res.data.subjectsList[0].
6391         subjectName });
6392     })
6393     .catch(err => {
6394       this.setState({ subjectOption: [] });
6395       this.setState({ isSubjectLoading: false });
6396       this.setState({ noSubjectData: true });
6397       this.setState({ selectedSubject: 0 });
6398       this.setState({ subjectName: 'NA' });
6399     });
6400   axios
6401     .post('api/registrar/getsectionbygradelevel', { gradeLevel: this.
6402       state.gradeLevel })
6403     .then(res => {
6404       this.setState({ isSectionLoading: false });
6405       this.setState({ sectionOption: res.data.sectionsList });
6406       this.setState({ noSectionData: false });
6407       this.setState({ selectedSection: res.data.sectionsList[0].
6408         sectionID });
6409       this.setState({ sectionName: res.data.sectionsList[0].
6410         sectionName });
6411     })
6412     .catch(err => {
6413       this.setState({ isSectionLoading: false });
6414       this.setState({ sectionOption: [] });
6415       this.setState({ noSectionData: true });
6416       this.setState({ selectedSection: 0 });
6417       this.setState({ sectionName: 'NA' });
6418     });
6419
6420 render() {
6421   const displayGradeLevel = gradeLevel => {
6422     switch (gradeLevel) {
6423       case 'N':
6424         return 'Nursery';
6425       case 'K1':
6426         return 'Kinder 1';
6427       case 'K2':
6428         return 'Kinder 2';
6429       case 'G1':
6430         return 'Grade 1';
6431       case 'G2':
6432         return 'Grade 2';
6433       case 'G3':
6434         return 'Grade 3';
6435       case 'G4':
6436         return 'Grade 4';
6437     }
6438   }
6439 }

```

```

6430         return 'Grade 4';
6431     case 'G5':
6432         return 'Grade 5';
6433     case 'G6':
6434         return 'Grade 6';
6435     case 'G7':
6436         return 'Grade 7';
6437     case 'G8':
6438         return 'Grade 8';
6439     case 'G9':
6440         return 'Grade 9';
6441     case 'G10':
6442         return 'Grade 10';
6443     case 'G11':
6444         return 'Grade 11';
6445     case 'G12':
6446         return 'Grade 12';
6447     default:
6448         return '';
6449     }
6450   };
6451   let displayOptions = [];
6452   const displayStudentData = [];
6453   const displaySubjectOption = [];
6454   const displaySectionOption = [];
6455   for (const [index, value] of this.state.subjectOption.entries()) {
6456     displaySubjectOption.push(<Option value={value.subjectID}>{value.
6457       subjectCode}</Option>);
6458   }
6459   for (const [index, value] of this.state.sectionOption.entries()) {
6460     displaySectionOption.push(<Option value={value.sectionID}>{value.
6461       sectionName}</Option>);
6462   }
6463   for (const [index, value] of this.state.options.entries()) {
6464     displayOptions.push(
6465       <Option key={value.key} text={value.name}>
6466         <div>
6467           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
6468             getImageUrl(value.imageUrl)} />
6469           <span style={{ margin: '16px', verticalAlign: 'text-top'
6470             }}>{value.name}</span>
6471         </div>
6472       </Option>,
6473     );
6474   }
6475   for (const [index, value] of this.state.studentData.entries()) {
6476     displayStudentData.push(
6477       <Table.Row>
6478         <Table.Col className="w-1">
6479           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
6480             getImageUrl(value.imageUrl)} />
6481         </Table.Col>
6482         <Table.Col>{value.name}</Table.Col>
6483         <Table.Col>{value.email}</Table.Col>
6484         <Table.Col>{value.sectionName}</Table.Col>
6485         <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
6486         <Table.Col alignContent="center">
6487           <Button
6488             pill
6489             icon="trash"
6490             color="danger"
6491             onClick={() => this.removeStudent(value.key)}
6492           ></Button>
6493         </Table.Col>
6494       </Table.Row>,
6495     );
6496   }
6497   if (displaySubjectOption.length == 0) {
6498     displaySubjectOption.push(<Option value={0}>No records.</Option>)
6499   }
6500   if (displaySectionOption.length == 0) {
6501     displaySectionOption.push(<Option value={0}>No records.</Option>)

```

```

        ;
6497    }
6498    if (displayStudentData.length == 0) {
6499      displayStudentData.push(
6500        <Table.Row>
6501          <Table.Col colSpan={6} alignContent="center">
6502            No result.
6503          </Table.Col>
6504        </Table.Row>,
6505      );
6506    }
6507
6508    if (this.state.optionLoading) {
6509      displayOptions = [
6510        <Option key={0} text="">
6511          <div>
6512            <Spin spinning={true}></Spin>
6513          </div>
6514        </Option>,
6515      ];
6516    } else {
6517      if (displayOptions.length == 0) {
6518        displayOptions = [
6519          <Option key={0} text="">
6520            <div>No data.</div>
6521          </Option>,
6522        ];
6523      }
6524    }
6525    return (
6526      <Table className="app-registrar-add-new-load card">
6527        <Card.Body>
6528          <Card.Title>Add a New Subject Load</Card.Title>
6529          <Container>
6530            <Grid.Row>
6531              <Grid.Col sm={12} lg={2}>
6532                <Form.Group>
6533                  <Form.Label>Grade Level</Form.Label>
6534                  <Input.Group>
6535                    <Select
6536                      style={{ width: '100%' }}
6537                      defaultValue={this.state.gradeLevel}
6538                      onChange={this.onChangeGradeLevel}
6539                    >
6540                      <Option value="N">Nursery</Option>
6541                      <Option value="K1">Kinder 1</Option>
6542                      <Option value="K2">Kinder 2</Option>
6543                      <Option value="G1">Grade 1</Option>
6544                      <Option value="G2">Grade 2</Option>
6545                      <Option value="G3">Grade 3</Option>
6546                      <Option value="G4">Grade 4</Option>
6547                      <Option value="G5">Grade 5</Option>
6548                      <Option value="G6">Grade 6</Option>
6549                      <Option value="G7">Grade 7</Option>
6550                      <Option value="G8">Grade 8</Option>
6551                      <Option value="G9">Grade 9</Option>
6552                      <Option value="G10">Grade 10</Option>
6553                      <Option value="G11">Grade 11</Option>
6554                      <Option value="G12">Grade 12</Option>
6555                    </Select>
6556                  </Input.Group>
6557                </Form.Group>
6558              </Grid.Col>
6559              <Grid.Col sm={12} lg={3}>
6560                <Form.Group>
6561                  <Form.Label>Section Name</Form.Label>
6562                  <Input.Group>
6563                    <Select
6564                      style={{ width: '100%' }}
6565                      defaultValue="0"
6566                      value={this.state.selectedSection}
6567                      loading={this.state.isSectionLoading}

```

```

6568         disabled={this.state.noSectionData}
6569         onChange={this.onChangeSection}
6570     >
6571         {displaySectionOption}
6572     </Select>
6573     </Input.Group>
6574   </Form.Group>
6575 </Grid.Col>
6576 <Grid.Col sm={12} lg={3}>
6577   <Form.Group>
6578     <Form.Label>Subject Name</Form.Label>
6579     <Input.Group>
6580       <Select
6581         style={{ width: '100%' }}
6582         value={this.state.selectedSubject}
6583         loading={this.state.isSubjectLoading}
6584         disabled={this.state.noSubjectData}
6585         onChange={this.onChangeSubject}
6586     >
6587       {displaySubjectOption}
6588     </Select>
6589   </Input.Group>
6590 </Form.Group>
6591 </Grid.Col>
6592 <Grid.Col sm={12} lg={4}>
6593   <Form.Label>Action</Form.Label>
6594   <Button
6595     color="primary"
6596     block
6597     icon="plus"
6598     onClick={this.importStudent}
6599     disabled={this.state.noSectionData}
6600   >
6601     Import Students
6602   </Button>
6603 </Grid.Col>
6604 <Grid.Col sm={12} xs={12} md={12}>
6605   <Descriptions
6606     style={{ marginBottom: '15px', marginTop: '15px' }}
6607     bordered
6608     title="Subject Load Information"
6609   >
6610     <Descriptions.Item span={3} label="Section Name">
6611       {this.state.sectionName}
6612     </Descriptions.Item>
6613     <Descriptions.Item span={3} label="Grade Level">
6614       {displayGradeLevel(this.state.gradeLevel)}
6615     </Descriptions.Item>
6616     <Descriptions.Item span={3} label="Subject Name">
6617       {this.state.subjectName}
6618     </Descriptions.Item>
6619     <Descriptions.Item span={3} label="Number of Students
6620       ">
6621       {this.state.studentData.length}
6622     </Descriptions.Item>
6623   </Descriptions>
6624 </Grid.Col>
6625 </Grid.Row>
6626 <Grid.Row>
6627   <Grid.Col sm={12} xs={12} md={10}>
6628     <AutoComplete
6629       style={{ width: '100%', marginBottom: '10px' }}
6630       optionLabelProp="text"
6631       onSearch={this.handleSearch}
6632       dataSource={displayOptions}
6633       onSelect={this.onSelect}
6634     >
6635       <Input placeholder="Search for students" enterButton
6636         />
6637     </AutoComplete>
6638 </Grid.Col>
6639 <Grid.Col sm={12} xs={12} md={2}>
6640   <Button

```

```

6639         block
6640             icon="plus"
6641             color="primary"
6642             onClick={() => this.addStudent()}
6643             disabled={this.state.selectedKey == -1}
6644         >
6645             Add
6646         </Button>
6647     </Grid.Col>
6648 </Grid.Row>
6649 <Grid.Row>
6650     <Grid.Col xs={12} sm={12} md={12}>
6651         <Spin spinning={this.state.isLoadingTable}>
6652             <Table highlightRowOnHover={true} responsive={true}>
6653                 <Table.Header>
6654                     <Table.ColHeader alignContent="center" colSpan={2}>
6655                         Student Name
6656                     </Table.ColHeader>
6657                     <Table.ColHeader>Email</Table.ColHeader>
6658                     <Table.ColHeader>Section Name</Table.ColHeader>
6659                     <Table.ColHeader>Grade Level</Table.ColHeader>
6660                     <Table.ColHeader alignContent="center">Actions</Table.ColHeader>
6661                 </Table.Header>
6662                 <Table.Body>{displayStudentData}</Table.Body>
6663             </Table>
6664         </Spin>
6665     </Grid.Col>
6666 </Grid.Row>
6667 <Grid.Row>
6668     <Button
6669         disabled={
6670             this.state.studentData.length == 0 ||
6671             this.state.selectedSection == 0 ||
6672             this.state.selectedSubject == 0 ||
6673             this.state.isLoadingTable ||
6674             this.state.isSectionLoading ||
6675             this.state.isSubjectLoading
6676         }
6677         icon="file"
6678         block
6679         color="primary"
6680         onClick={() => {
6681             this.props.actions.createSubjectSection({
6682                 payload: this.state.studentData,
6683                 sectionID: this.state.selectedSection,
6684                 subjectID: this.state.selectedSubject,
6685                 accountID: this.props.accountID,
6686             });
6687         }}
6688     >
6689         Create Subject Load
6690     </Button>
6691 </Grid.Row>
6692 </Container>
6693 </Card.Body>
6694 </Table>
6695 );
6696 }
6697 }
6698 /* istanbul ignore next */
6699 function mapStateToProps(state) {
6700     return {
6701         app: state.app,
6702     };
6703 }
6704 }
6705 /* istanbul ignore next */
6706 function mapDispatchToProps(dispatch) {
6707     return {
6708         actions: bindActionCreators({ ...actions }, dispatch),
6709

```

```

6710     };
6711 }
6712
6713 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarAddNewLoad);
6714 import React, { Component } from 'react';
6715 import PropTypes from 'prop-types';
6716 import { bindActionCreators } from 'redux';
6717 import { connect } from 'react-redux';
6718 import * as actions from './redux/actions';
6719 import axios from 'axios';
6720 import { Pagination, Spin, Tooltip } from 'antd';
6721 import {
  Modal,
  Popconfirm,
  Search,
  Breadcrumb,
  AutoComplete,
  Input,
  message,
  Descriptions,
  Popover,
} from 'antd';
6731 import cn from 'classnames';
6732 import placeholder from '../../../../../images/placeholder.jpg';
6733 import {
  Card,
  Button,
  Grid,
  Avatar,
  Table,
  Form,
  Header,
  Container,
  Text,
  Alert,
} from 'tabler-react';
6745 import { Link } from 'react-router-dom';
6746 import moment from 'moment';
6748 import { getImageUrl } from '../../../../../utils';
6749 const { Option } = AutoComplete;
6750
6751 export class RegistrarAddRecord extends Component {
6752   static propTypes = {
6753     app: PropTypes.object.isRequired,
6754     actions: PropTypes.object.isRequired,
6755   };
6756
6757   constructor(props) {
6758     super(props);
6759     this.state = {
6760       componentID: 0,
6761       component: '',
6762       subcompName: '',
6763       subcompID: 0,
6764       isLoading: true,
6765       quarter: 'Q1',
6766       sectionName: '',
6767       subjectName: '',
6768       schoolYear: '',
6769       subjectCode: '',
6770       studList: [],
6771       itemDesc: '',
6772       dateGiven: new Date(),
6773       total: 0,
6774     };
6775     this.addNewRecord = this.addNewRecord.bind(this);
6776   }
6777
6778   addNewRecord() {
6779     this.props.actions.addNewRecord(
6780       {
6781         componentID: this.state.componentID,
6782         subcompID: this.state.subcompID,
6783         payload: this.state.studList,

```

```

6784     dateGiven: this.state.dateGiven,
6785     description: this.state.itemDesc,
6786     total: this.state.total,
6787     classRecordID: this.props.classRecordID,
6788     quarter: this.props.quarter,
6789   },
6790   'Registrar',
6791 );
6792 }
6793
6794 componentDidMount() {
6795   this.setState({ isLoading: true });
6796   axios
6797     .post('api/registrar/getcomponents', {
6798       classRecordID: this.props.classRecordID,
6799       quarter: this.props.quarter,
6800     })
6801     .then(res => {
6802       this.setState({
6803         sectionName: res.data.sectionName,
6804         subjectName: res.data.subjectName,
6805         schoolYear: res.data.schoolYear,
6806         subjectCode: res.data.subjectCode,
6807       });
6808       axios
6809         .post('api/registrar/subcompinfo', {
6810           classRecordID: this.props.classRecordID,
6811           componentID: this.props.componentID,
6812           quarter: this.props.quarter,
6813           subcompID: this.props.subcompID,
6814         })
6815         .then(res2 => {
6816           let studList = [];
6817           for (const [index, value] of res2.data.data.entries()) {
6818             const { subsectstudID, name, imageUrl } = value;
6819             studList.push({ subsectstudID, name, score: 0, imageUrl
6820           });
6821           this.setState({
6822             isLoading: false,
6823             componentID: this.props.componentID,
6824             component: res2.data.component,
6825             subcompName: res2.data.subcompName,
6826             subcompID: this.props.subcompID,
6827             quarter: this.props.quarter,
6828             studList,
6829           });
6830         });
6831       });
6832     });
6833   }
6834   render() {
6835     const birthDate = new Date(this.state.dateGiven);
6836     const defaultDate = birthDate.toISOString();
6837     let tableData = [];
6838     for (const [index, value] of this.state.studList.entries()) {
6839       tableData.push(
6840         <Table.Row>
6841           <Table.Col className="w-1">
6842             <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getimageUrl(value.imageUrl)} />
6843           </Table.Col>
6844           <Table.Col>{value.name}</Table.Col>
6845           <Table.Col>
6846             <Form.Group>
6847               <Form.Input
6848                 placeholder="Score"
6849                 position="append"
6850                 value={value.score}
6851                 onChange={e => {
6852                   const reg = /^-?[0-9]*(\.[0-9]*)?$/;
6853                   let temparr = this.state.studList;
6854                   if (
6855

```

```

6856             (!isNaN(e.target.value) && reg.test(e.target.value)
6857                 ) ||
6858                 (e.target.value === '' && e.target.value !== '-')
6859                     ||
6860                     e.target.value == 'A' ||
6861                     e.target.value == 'E'
6862             ) {
6863                 temparr[index].score = e.target.value;
6864                 this.setState({ studList: temparr });
6865             }
6866         )}
6867     />
6868     </Form.Group>
6869     </Table.Col>
6870     </Table.Row>,
6871 );
6872 }
6873 return (
6874     <div className="app-teacher-add-record my-3 my-md-5">
6875         <Container>
6876             <Grid.Row>
6877                 <Card>
6878                     <Card.Body>
6879                         {this.state.isLoading ? (
6880                             <Spin spinning={true}></Spin>
6881                         ) : (
6882                             <div>
6883                                 <Card.Title>
6884                                     <Breadcrumb>
6885                                         <Breadcrumb.Item>Subjects</Breadcrumb.Item>
6886                                         <Breadcrumb.Item>View Subject Load</Breadcrumb.
6887                                             Item>
6888                                         <Breadcrumb.Item>Manage Grades</Breadcrumb.Item>
6889                                         >
6890                                         <Breadcrumb.Item>
6891                                         {this.state.sectionName} - {this.state.
6892                                             subjectName}
6893                                         </Breadcrumb.Item>
6894                                         <Breadcrumb.Item>{this.state.component}</
6895                                             Breadcrumb.Item>
6896                                         <Breadcrumb.Item>{this.state.subcompName}</
6897                                             Breadcrumb.Item>
6898                                         </Breadcrumb>
6899                                     </Card.Title>
6900                                     <Header.H3>
6901                                         {this.state.component} - {this.state.
6902                                             subcompName} - Add New Record
6903                                         </Header.H3>
6904                                     </Card.Title>
6905                                     <Card.Body>
6906                                         </Card>
6907                                     </Grid.Row>
6908                                     <Grid.Row>
6909                                         <Grid.Col sm={12} xs={12} md={6}>
6910                                         <Card>
6911                                         <Card.Body>
6912                                         <Grid.Row>
6913                                         <Grid.Col sm={12} xs={12} md={12}>
6914                                         <Descriptions
6915                                             style={{ marginBottom: '15px', marginTop: '15px
6916                                                 ,
6917                                                 }}>
6918                                         bordered
6919                                         title="Add a New Record"
6920                                     >

```

```

6919 <Descriptions.Item span={3} label="Item
6920   Description">
6921   <Form.Group>
6922     <Form.Input
6923       placeholder="Description"
6924       value={this.state.itemDesc}
6925       onChange={e => {
6926         this.setState({ itemDesc: e.target.
6927           value });
6928       }}
6929     />
6930   </Form.Group>
6931 </Descriptions.Item>
6932 <Descriptions.Item span={3} label="Date Given">
6933   <Form.Group>
6934     <Form.DatePicker
6935       defaultDate={new Date(defaultDate)}
6936       format="mm/dd/yyyy"
6937       onChange={e => {
6938         this.setState({ dateGiven: e });
6939       }}
6940       name="birthDate"
6941       maxYear={2020}
6942       minYear={1897}
6943       monthLabels={[
6944         'January',
6945         'February',
6946         'March',
6947         'April',
6948         'May',
6949         'June',
6950         'July',
6951         'August',
6952         'September',
6953         'October',
6954         'November',
6955         'December',
6956       ]}
6957     ></Form.DatePicker>
6958   </Form.Group>
6959 </Descriptions.Item>
6960 <Descriptions.Item span={3} label="Total number
6961   of items">
6962   <Form.Group>
6963     <Form.Input
6964       placeholder="Total"
6965       value={this.state.total}
6966       onChange={e => {
6967         const reg = /^-?[0-9]*(\.[0-9]*)?$/;
6968
6969         if (
6970           (!isNaN(e.target.value) && reg.test(e
6971             .target.value)) ||
6972             e.target.value === ',' ||

6973             e.target.value === '-',
6974           )
6975           {
6976             this.setState({ total: e.target.value
6977               });
6978           }
6979         }
6980       }>
6981     </Form.Group>
6982   </Descriptions.Item>
6983   </Grid.Col>
6984 </Grid.Row>
6985 </Card.Body>
6986 </Card>
6987 <Grid.Col sm={12} xs={12} md={6}>
6988   <Card>
6989     <Card.Body>
6990       <Card.Title>Student Scores</Card.Title>

```

```

6987             <Container>
6988                 <Alert type="primary">
6989                     <b>Note:</b> Scores with value of <b>E</b> are
6990                         considered{' '}
6991                         <b>excused</b>
6992                     </i>
6993                         . Enter a value of <b>A</b> for students who are{
6994                             , ,}
6995                         <b>absent</b>
6996                     </i>
6997                         .
6998                 </Alert>
6999             </Container>
7000             <Table highlightRowOnHover={true} responsive={true}>
7001                 <Table.Header>
7002                     <Table.ColHeader colSpan={2}>Student Name</Table.
7003                         ColHeader>
7004                     <Table.ColHeader>Score</Table.ColHeader>
7005                 <Table.Body>{tableData}</Table.Body>
7006             </Table>
7007         </Card.Body>
7008         <Card.Footer>
7009             <Button.List align="right">
7010                 <Button icon="plus" color="success" onClick={this.
7011                     addNewRecord}>
7012                     Add New Record
7013                 </Button>
7014             </Button.List>
7015         </Card.Footer>
7016     </Grid.Col>
7017     </Grid.Row>
7018 </Container>
7019 </div>
7020 );
7021 }
7022 }
7023 */
7024 /* istanbul ignore next */
7025 function mapStateToProps(state) {
7026     return {
7027         app: state.app,
7028     };
7029 }
7030 */
7031 /* istanbul ignore next */
7032 function mapDispatchToProps(dispatch) {
7033     return {
7034         actions: bindActionCreators({ ...actions }, dispatch),
7035     };
7036 }
7037 */
7038 export default connect(mapStateToProps, mapDispatchToProps)(
    RegistrarAddRecord);
7039 import React, { Component } from 'react';
7040 import PropTypes from 'prop-types';
7041 import { bindActionCreators } from 'redux';
7042 import { connect } from 'react-redux';
7043 import * as actions from './redux/actions';
7044 import NavBar from './NavBar';
7045 import RegistrarAddRecord from './RegistrarAddRecord';
7046 import { Container, Grid } from 'tabler-react';
7047 import { Result, Button } from 'antd';
7048 import axios from 'axios';
7049 import { Link } from 'react-router-dom';
7050 */
7051 export class RegistrarAddRecordView extends Component {
7052     static propTypes = {
7053         app: PropTypes.object.isRequired,

```

```

7054     actions: PropTypes.object.isRequired,
7055   };
7056
7057   constructor(props) {
7058     super(props);
7059     this.state = {
7060       locked: false,
7061       isLoading: true,
7062     };
7063   }
7064
7065   componentDidMount() {
7066     this.props.actions.setLoadingTrue();
7067     if (!this.props.app.auth.isAuthenticated) {
7068       this.props.history.push('/login');
7069     } else {
7070       if (this.props.app.auth.user.position != 2) {
7071         this.props.history.push('/page401');
7072       } else {
7073         axios
7074           .get('api/registrar/getsy')
7075           .then(res => {
7076             this.props.actions.setLoadingFalse();
7077             this.setState({ locked: false, isLoading: false });
7078           })
7079           .catch(err => {
7080             this.props.actions.setLoadingFalse();
7081             this.setState({ locked: true, isLoading: false });
7082           });
7083         }
7084       }
7085     }
7086
7087   render() {
7088     return (
7089       <div className="app-registrar-add-record-view fh">
7090         {this.state.isLoading ? (
7091           '',
7092         ) : this.state.locked ? (
7093           <NavBar>
7094             <Container>
7095               <Grid.Row>
7096                 <Grid.Col xs={12} sm={12} md={12}>
7097                   <Result
7098                     status="403"
7099                     title="No Active School Year"
7100                     subTitle="Sorry, you are not authorized to access
7101                       this page now. Please contact your system
7102                         administrator for details."
7103                     extra={
7104                       <Link to="/">
7105                         <Button type="primary">Back Home</Button>
7106                         </Link>
7107                       }
7108                     />
7109                   </Grid.Col>
7110                 </Grid.Row>
7111               </Container>
7112             </NavBar>
7113           ) : (
7114             <NavBar>
7115               <Container>
7116                 <Grid.Row>
7117                   <Grid.Col xs={12} sm={12} md={12}>
7118                     <RegistrarAddRecord
7119                       classRecordID={this.props.match.params.
7120                         classRecordID}
7121                         componentID={this.props.match.params.comp}
7122                           subcompID={this.props.match.params.subcomp}
7123                             quarter={this.props.match.params.Q}
7124                               id={this.props.match.params.id}
7125                                 history={this.props.history}
7126                               />

```

```

7124             </Grid.Col>
7125         </Grid.Row>
7126     </Container>
7127     <NavBar>
7128         )}
7129     );
7130   );
7131 }
7132 }
7133 */
7134 /* istanbul ignore next */
7135 function mapStateToProps(state) {
7136   return {
7137     app: state.app,
7138   };
7139 }
7140 */
7141 /* istanbul ignore next */
7142 function mapDispatchToProps(dispatch) {
7143   return {
7144     actions: bindActionCreators({ ...actions }, dispatch),
7145   };
7146 }
7147 */
7148 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarAddRecordView);
7149 import React, { Component } from 'react';
7150 import PropTypes from 'prop-types';
7151 import { bindActionCreators } from 'redux';
7152 import { connect } from 'react-redux';
7153 import * as actions from './redux/actions';
7154 import { Card, Button, Grid, Table, Form, Container, Avatar } from 'tabler-react';
7155 import { Pagination, Spin, Breadcrumb, Modal } from 'antd';
7156 import axios from 'axios';
7157 import { Popconfirm, Input, message, AutoComplete } from 'antd';
7158 import { getImageUrl } from '../../../../../utils';
7159 import placeholder from '../../../../../images/placeholder.jpg';
7160 const { Option } = AutoComplete;
7161 */
7162 export class RegistrarAssignAdvisorySection extends Component {
7163   static propTypes = {
7164     app: PropTypes.object.isRequired,
7165     actions: PropTypes.object.isRequired,
7166   };
7167   */
7168   constructor(props) {
7169     super(props);
7170     this.state = {
7171       isLoading: true,
7172       isLoadingTable: true,
7173       keyword: '',
7174       page: 1,
7175       pageSize: 10,
7176       numOfPages: 1,
7177       data: [],
7178       schoolYearID: 0,
7179       schoolYear: '',
7180       assignAdviserLoading: false,
7181       showAssignAdviser: false,
7182       sectionName: '',
7183       selectedKey: -1,
7184       selectedSection: -1,
7185       teacherKeyword: '',
7186       options: [],
7187     };
7188   */
7189   this.onChangeSearch = this.onChangeSearch.bind(this);
7190   this.showAssignAdviser = this.showAssignAdviser.bind(this);
7191   this.hideAssignAdviser = this.hideAssignAdviser.bind(this);
7192   this.onTeacherChange = this.onTeacherChange.bind(this);
7193   this.onSelect = this.onSelect.bind(this);
7194   this.assignAdviser = this.assignAdviser.bind(this);

```

```

7195     this.unassignAdviser = this.unassignAdviser.bind(this);
7196 }
7197
7198 assignAdviser() {
7199   if (this.state.selectedKey == -1) {
7200     message.error('You must select a teacher');
7201   } else {
7202     this.props.actions.assignAdvisorySection({
7203       schoolYearID: this.state.schoolYearID,
7204       teacherID: this.state.selectedKey,
7205       sectionID: this.state.selectedSection,
7206     });
7207   }
7208 }
7209
7210 unassignAdviser() {
7211   this.props.actions.unassignAdvisorySection({
7212     schoolYearID: this.state.schoolYearID,
7213     teacherID: this.state.selectedKey,
7214     sectionID: this.state.selectedSection,
7215   });
7216 }
7217
7218 onTeacherChange(query) {
7219   this.setState({ teacherKeyword: query });
7220   this.setState({ selectedKey: -1 });
7221   this.setState({
7222     options: [
7223       <Option key={0} text="">
7224         <div>
7225           <Spin spinning={true}></Spin>
7226         </div>
7227       </Option>,
7228     ],
7229   });
7230   axios
7231     .post('api/registrar/searchteacher', { keyword: query })
7232     .then(res => {
7233       if (query == '') {
7234         this.setState({ options: [] });
7235       } else {
7236         let optionData = res.data.accountList.map(data => (
7237           <Option key={data.key} text={data.name}>
7238             <div>
7239               <Avatar
7240                 imageURL={data.imageUrl == 'NA' ? placeholder :
7241                   getImageUrl(data.imageUrl)}
7242                 />
7243                 <span style={{ margin: '16px', verticalAlign: 'text-top
7244                   }}>{data.name}</span>
7245               </div>
7246             </Option>
7247         )));
7248         this.setState({ options: optionData });
7249       }
7250     })
7251     .catch(err => {
7252       this.setState({
7253         options: [
7254           <Option key={0} text="">
7255             <div>No data.</div>
7256           </Option>,
7257         ],
7258       });
7259     })
7260   showAssignAdviser(key) {
7261     this.setState({
7262       showAssignAdviser: true,
7263       sectionName: this.state.data.filter(section => section.sectionID
7264         === key)[0].sectionName,

```

```

7264         selectedSection: key ,
7265     });
7266 }
7267
7268 hideAssignAdviser() {
7269     this.setState({ showAssignAdviser: false, selectedSection: -1 });
7270 }
7271
7272 componentDidMount() {
7273     this.setState({ isLoading: true, isLoadingTable: true });
7274     axios.get('api/registrar/getsy').then(res => {
7275         this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
7276             res.data.schoolYear });
7277         this.setState({ isLoading: false });
7278         axios
7279             .post('api/registrar/advisorytable', {
7280                 keyword: this.state.keyword,
7281                 page: this.state.page,
7282                 pageSize: this.state.pageSize,
7283             })
7284             .then(res2 => {
7285                 this.setState({
7286                     numOfPages: res2.data.numOfPages,
7287                     data: res2.data.advisoryData,
7288                     isLoadingTable: false,
7289                 });
7290             })
7291             .catch(err => {
7292                 this.setState({ isLoadingTable: false });
7293             });
7294     });
7295 }
7296
7297 componentWillMountReceiveProps() {
7298     this.setState({ isLoading: true, isLoadingTable: true });
7299     axios.get('api/registrar/getsy').then(res => {
7300         this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
7301             res.data.schoolYear });
7302         this.setState({ isLoading: false });
7303         axios
7304             .post('api/registrar/advisorytable', {
7305                 keyword: this.state.keyword,
7306                 page: this.state.page,
7307                 pageSize: this.state.pageSize,
7308             })
7309             .then(res2 => {
7310                 this.setState({
7311                     numOfPages: res2.data.numOfPages,
7312                     data: res2.data.advisoryData,
7313                     isLoadingTable: false,
7314                 });
7315             })
7316             .catch(err => {
7317                 this.setState({ isLoadingTable: false });
7318             });
7319     });
7320 }
7321
7322 onChangeSearch(event) {
7323     this.setState({ [event.target.name]: event.target.value });
7324     this.setState({ isLoadingTable: true, page: 1 });
7325     axios
7326         .post('api/registrar/advisorytable', {
7327             keyword: event.target.value,
7328             page: this.state.page,
7329             pageSize: this.state.pageSize,
7330         })
7331         .then(res2 => {
7332             this.setState({
7333                 numOfPages: res2.data.numOfPages,
7334                 data: res2.data.advisoryData,
7335                 isLoadingTable: false,
7336             });
7337
7338
7339
7340
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7799

```

```

7335         })
7336     .catch(err => {
7337       this.setState({ isLoadingTable: false });
7338       this.setState({ data: [] });
7339     });
7340   }
7341
7342   paginate = page => {
7343     this.setState({ page });
7344     this.setState({ isLoadingTable: true });
7345     axios
7346       .post('api/registrar/advisorytable', {
7347         keyword: this.state.keyword,
7348         page: page,
7349         pageSize: this.state.pageSize,
7350       })
7351       .then(res2 => {
7352         this.setState({
7353           numOfPages: res2.data.numOfPages,
7354           data: res2.data.advisoryData,
7355           isLoadingTable: false,
7356         });
7357       })
7358       .catch(err => {
7359         this.setState({ isLoadingTable: false });
7360         this.setState({ data: [] });
7361       });
7362   };
7363
7364   onSelect(key) {
7365     this.setState({ selectedKey: key });
7366   }
7367
7368   render() {
7369     const { options } = this.state;
7370     const displayGradeLevel = gradeLevel => {
7371       switch (gradeLevel) {
7372         case 'N':
7373           return 'Nursery';
7374         case 'K1':
7375           return 'Kinder 1';
7376         case 'K2':
7377           return 'Kinder 2';
7378         case 'G1':
7379           return 'Grade 1';
7380         case 'G2':
7381           return 'Grade 2';
7382         case 'G3':
7383           return 'Grade 3';
7384         case 'G4':
7385           return 'Grade 4';
7386         case 'G5':
7387           return 'Grade 5';
7388         case 'G6':
7389           return 'Grade 6';
7390         case 'G7':
7391           return 'Grade 7';
7392         case 'G8':
7393           return 'Grade 8';
7394         case 'G9':
7395           return 'Grade 9';
7396         case 'G10':
7397           return 'Grade 10';
7398         case 'G11':
7399           return 'Grade 11';
7400         case 'G12':
7401           return 'Grade 12';
7402         default:
7403           return '';
7404       }
7405     };
7406     const DisplayData = [];
7407     for (const [index, value] of this.state.data.entries()) {
7408       DisplayData.push(
7409         <Table.Row>

```

```

7410 <Table.Col>{value.sectionName}</Table.Col>
7411 <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
7412 <Table.Col>{value.adviser}</Table.Col>
7413 <Table.Col alignContent="center">
7414   {value.adviser === 'NO ADVISER' ? (
7415     <Button
7416       icon="plus"
7417       color="primary"
7418       value={value.sectionID}
7419       onClick={() => {
7420         this.showAssignAdviser(value.sectionID);
7421       }}
7422     >
7423       Assign Adivser
7424     </Button>
7425   ) : (
7426     <Popconfirm
7427       title="Do you want to unassign this teacher?"
7428       onConfirm={this.unassignAdviser}
7429       okText="Unassign"
7430       cancelText="cancel"
7431     >
7432       <Button
7433         icon="trash"
7434         color="danger"
7435         onClick={() => {
7436           this.setState({
7437             selectedSection: value.sectionID,
7438             selectedKey: value.teacherID,
7439           });
7440         }}
7441       >
7442         Unassign Adivser
7443       </Button>
7444     </Popconfirm>
7445   )}
7446 </Table.Col>
7447 </Table.Row>,
7448 );
7449 }
7450 return (
7451   <Table.Body className="app-registrar-assign-advisory-section my-3
7452     my-md-5 card">
7453     <Modal
7454       title="Assign Adivser"
7455       visible={this.state.showAssignAdviser}
7456       onOk={this.assignAdviser}
7457       onCancel={this.hideAssignAdviser}
7458       cancelText="close"
7459       okText="Assign Adivser"
7460     >
7461       <Spin spinning={this.props.app.showLoading}>
7462         <Container>
7463           <Grid.Row>
7464             <Grid.Col sm={12} md={12} xs={12}>
7465               <AutoComplete
7466                 style={{ width: '100%', marginBottom: '10px' }}
7467                 onSearch={this.onTeacherChange}
7468                 dataSource={options}
7469                 onSelect={this.onSelect}
7470                 optionLabelProp="text"
7471               >
7472                 <Input placeholder="Search for teachers"
7473                   enterButton />
7474               </AutoComplete>
7475             </Grid.Col>
7476           </Grid.Row>
7477         </Container>
7478       </Spin>
7479     </Modal>
7480   <Card.Body>
7481     {this.state.isLoading ? (
7482       '',
7483     ) : (

```

```

7482     <div>
7483       <Card.Title>
7484         <Breadcrumb>
7485           <Breadcrumb.Item>Teachers</Breadcrumb.Item>
7486           <Breadcrumb.Item>Assign Advisory Section</Breadcrumb.
7487             Item>
7488         </Breadcrumb>
7489       </Card.Title>
7490       <Card.Title>Advisory Table for S.Y. {this.state.
7491         schoolYear}</Card.Title>
7492     </div>
7493   )
7494   <Grid.Row>
7495     <Grid.Col sm={12} md={12} xs={12}>
7496       <Grid.Row>
7497         <Grid.Col sm={12} md={12} xs={12}>
7498           <Form.Group>
7499             <Form.Input
7500               icon="search"
7501               placeholder="Search for..."
7502               position="append"
7503               name="keyword"
7504               value={this.state.keyword}
7505               onChange={this.onChangeSearch}
7506             />
7507           </Form.Group>
7508         </Grid.Col>
7509       </Grid.Row>
7510       <Spin spinning={this.state.isLoadingTable}>
7511         <Table highlightRowOnHover={true} responsive={true}>
7512           <Table.Header>
7513             <Table.ColHeader>Section Name</Table.ColHeader>
7514             <Table.ColHeader>Grade Level</Table.ColHeader>
7515             <Table.ColHeader>Adviser</Table.ColHeader>
7516             <Table.ColHeader alignContent="center">Action</
7517               Table.ColHeader>
7518           </Table.Header>
7519           <Table.Body>
7520             {DisplayData.length == 0 ? (
7521               <Table.Row>
7522                 <Table.Col colSpan={3} alignContent="center">
7523                   No entries.
7524                 </Table.Col>
7525               </Table.Row>
7526             ) : (
7527               DisplayData
7528             )}
7529           </Table.Body>
7530         </Table>
7531         <Pagination
7532           size="large"
7533           current={this.state.page}
7534           pageSize={this.state.pageSize}
7535           total={this.state.pageSize * this.state.numOfPages}
7536           onChange={this.paginate}
7537         />
7538       </Spin>
7539     </Grid.Col>
7540   </Grid.Row>
7541   <Card.Body>
7542     <Table.Body>
7543   );
7544 }
7545 /* istanbul ignore next */
7546 function mapStateToProps(state) {
7547   return {
7548     app: state.app,
7549   };
7550 }
7551 /* istanbul ignore next */

```

```

7552     function mapDispatchToProps(dispatch) {
7553       return {
7554         actions: bindActionCreators({ ...actions }, dispatch),
7555       };
7556     }
7557   }
7558   export default connect(mapStateToProps, mapDispatchToProps)(
7559     RegistrarAssignAdvisorySection);
7560   import React, { Component } from 'react';
7561   import PropTypes from 'prop-types';
7562   import { bindActionCreators } from 'redux';
7563   import { connect } from 'react-redux';
7564   import * as actions from './redux/actions';
7565   import { Container, Grid } from 'tabler-react';
7566   import NavBar from './NavBar';
7567   import RegistrarAssignAdvisorySection from './
7568     RegistrarAssignAdvisorySection';
7569   import { Result, Button } from 'antd';
7570   import axios from 'axios';
7571   import { Link } from 'react-router-dom';
7572
7573   export class RegistrarAssignAdvisorySectionView extends Component {
7574     static propTypes = {
7575       app: PropTypes.object.isRequired,
7576       actions: PropTypes.object.isRequired,
7577     };
7578
7579     constructor(props) {
7580       super(props);
7581       this.state = {
7582         locked: false,
7583         isLoading: true,
7584       };
7585     }
7586
7587     componentDidMount() {
7588       this.props.actions.setLoadingTrue();
7589       if (!this.props.app.auth.isAuthenticated) {
7590         this.props.history.push('/login');
7591       } else {
7592         if (this.props.app.auth.user.position != 2) {
7593           this.props.history.push('/page401');
7594         } else {
7595           axios
7596             .get('api/registrar/getsy')
7597             .then(res => {
7598               this.props.actions.setLoadingFalse();
7599               this.setState({ locked: false, isLoading: false });
7600             })
7601             .catch(err => {
7602               this.props.actions.setLoadingFalse();
7603               this.setState({ locked: true, isLoading: false });
7604             });
7605       }
7606     }
7607     render() {
7608       return (
7609         <div className="app-registrar-assign-advisory-section-view fh">
7610           {this.state.isLoading ? (
7611             '',
7612           ) : this.state.locked ? (
7613             <NavBar>
7614               <Container>
7615                 <Grid.Row>
7616                   <Grid.Col xs={12} sm={12} md={12}>
7617                     <Result
7618                       status="403"
7619                       title="No Active School Year"
7620                       subTitle="Sorry, you are not authorized to access
7621                         this page now. Please contact your system
7622                           administrator for details."

```

```

7621             extra={
7622                 <Link to="/">
7623                     <Button type="primary">Back Home</Button>
7624                 </Link>
7625             }
7626         />
7627     </Grid.Col>
7628   </Grid.Row>
7629 </Container>
7630 </NavBar>
7631 ) : (
7632   <NavBar>
7633     <Container>
7634       <Grid.Row>
7635         <Grid.Col xs={12} sm={12} md={12}>
7636           <RegistrarAssignAdvisorySection />
7637         </Grid.Col>
7638       </Grid.Row>
7639     </Container>
7640   </NavBar>
7641 )
7642   </div>
7643 );
7644 }
7645 }
7646
7647 /* istanbul ignore next */
7648 function mapStateToProps(state) {
7649   return {
7650     app: state.app,
7651   };
7652 }
7653
7654 /* istanbul ignore next */
7655 function mapDispatchToProps(dispatch) {
7656   return {
7657     actions: bindActionCreators({ ...actions }, dispatch),
7658   };
7659 }
7660
7661 export default connect(mapStateToProps, mapDispatchToProps)(
7662   RegistrarAssignAdvisorySectionView);
7663 import React, { Component } from 'react';
7664 import PropTypes from 'prop-types';
7665 import { bindActionCreators } from 'redux';
7666 import { connect } from 'react-redux';
7667 import * as actions from './redux/actions';
7668 import { Card, Button, Grid, Avatar, Table, Form } from 'tabler-react';
7669 import { Pagination, Spin, Breadcrumb } from 'antd';
7670 import axios from 'axios';
7671 import placeholder from '../../../../../images/placeholder.jpg';
7672 import { getImageUrl } from '../../../../../utils';
7673
7674 export class RegistrarAssignSubjectLoad extends Component {
7675   static propTypes = {
7676     app: PropTypes.object.isRequired,
7677     actions: PropTypes.object.isRequired,
7678   };
7679
7680   constructor(props) {
7681     super(props);
7682     this.state = {
7683       isLoading: true,
7684       isLoadingTable: true,
7685       keyword: '',
7686       page: 1,
7687       pageSize: 10,
7688       numOfPages: 1,
7689       data: [],
7690       schoolYearID: 0,
7691       schoolYear: '',
7692     };

```

```

7693     this.onChangeSearch = this.onChangeSearch.bind(this);
7694 }
7695
7696
7697 onChangeSearch(event) {
7698     this.setState({ [event.target.name]: event.target.value });
7699     this.setState({ isLoadingTable: true, page: 1 });
7700     axios
7701         .post('api/registrar/getallteachers', {
7702             keyword: event.target.value,
7703             page: 1,
7704             pageSize: this.state.pageSize,
7705         })
7706         .then(res => {
7707             this.setState({
7708                 numOfPages: res.data.numOfPages,
7709                 data: res.data.accountList,
7710                 isLoadingTable: false,
7711             });
7712         })
7713         .catch(err => {
7714             this.setState({ isLoadingTable: false, data: [] });
7715         });
7716     }
7717
7718 paginate = page => {
7719     this.setState({ page });
7720     this.setState({ isLoadingTable: true });
7721     axios
7722         .post('api/registrar/getallteachers', {
7723             keyword: this.state.keyword,
7724             page: page,
7725             pageSize: this.state.pageSize,
7726         })
7727         .then(res => {
7728             this.setState({
7729                 numOfPages: res.data.numOfPages,
7730                 data: res.data.accountList,
7731                 isLoadingTable: false,
7732             });
7733         })
7734         .catch(err => {
7735             this.setState({ isLoadingTable: false, data: [] });
7736         });
7737     };
7738
7739 componentDidMount() {
7740     this.setState({ isLoading: true, isLoadingTable: true });
7741     axios.get('api/registrar/getsy').then(res => {
7742         this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
7743             res.data.schoolYear });
7744         this.setState({ isLoading: false });
7745         axios
7746             .post('api/registrar/getallteachers', {
7747                 keyword: this.state.keyword,
7748                 page: this.state.page,
7749                 pageSize: this.state.pageSize,
7750             })
7751             .then(res2 => {
7752                 this.setState({
7753                     numOfPages: res2.data.numOfPages,
7754                     data: res2.data.accountList,
7755                     isLoadingTable: false,
7756                 });
7757             })
7758             .catch(err => {
7759                 this.setState({ isLoadingTable: false });
7760             });
7761     });
7762
7763     render() {
7764         const DisplayData = [];

```

```

7765     for (const [index, value] of this.state.data.entries()) {
7766       DisplayData.push(
7767         <Table.Row>
7768           <Table.Col className="w-1">
7769             <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getimageUrl(value.imageUrl)} />
7770           </Table.Col>
7771           <Table.Col>{value.name}</Table.Col>
7772           <Table.Col>{value.email}</Table.Col>
7773           <Table.Col alignContent="center">
7774             <Link to={`/assignsubjectload/viewload/${value.key}}>
7775               <Button icon="user" color="primary">
7776                 View Subject Load
7777               </Button>
7778             </Link>
7779           </Table.Col>
7780         </Table.Row>,
7781       );
7782     }
7783   return (
7784     <Table.Body className="app-registrar-assign-subject-load card my-3 my-md-5">
7785       <Card.Body>
7786         {this.state.isLoading ? (
7787           ,
7788         ) : (
7789           <div>
7790             <Card.Title>
7791               <Breadcrumb>
7792                 <Breadcrumb.Item>Teachers</Breadcrumb.Item>
7793                 <Breadcrumb.Item>Assign Subject Load</Breadcrumb.Item>
7794               </Breadcrumb>
7795             </Card.Title>
7796             <Card.Title>Assign Subject Load for S.Y. {this.state.schoolYear}</Card.Title>
7797           </div>
7798         )}
7799       <Grid.Row>
7800         <Grid.Col sm={12} md={12} xs={12}>
7801           <Grid.Row>
7802             <Grid.Col sm={12} md={12} xs={12}>
7803               <Form.Group>
7804                 <Form.Input
7805                   icon="search"
7806                   placeholder="Search for..."
7807                   position="append"
7808                   name="keyword"
7809                   value={this.state.keyword}
7810                   onChange={this.onChangeSearch}
7811                 />
7812               </Form.Group>
7813             </Grid.Col>
7814           </Grid.Row>
7815           <Spin spinning={this.state.isLoadingTable}>
7816             <Table highlightRowOnHover={true} responsive={true}>
7817               <Table.Header>
7818                 <Table.ColHeader colSpan={2}>Teacher Name</Table.ColHeader>
7819                 <Table.ColHeader>Email</Table.ColHeader>
7820                 <Table.ColHeader alignContent="center">Action</Table.ColHeader>
7821               </Table.Header>
7822               <Table.Body>
7823                 {DisplayData.length == 0 ? (
7824                   <Table.Row>
7825                     <Table.Col colSpan={4} alignContent="center">
7826                       No entries.
7827                     </Table.Col>
7828                   </Table.Row>
7829                 ) : (
7830                   DisplayData
7831                 )}
```

```

7832             </Table.Body>
7833         </Table>
7834         <Pagination
7835             size="large"
7836             current={this.state.page}
7837             pageSize={this.state.pageSize}
7838             total={this.state.pageSize * this.state.numOfPages}
7839             onChange={this.paginate}
7840         />
7841     </Spin>
7842     </Grid.Col>
7843   </Grid.Row>
7844   </Card.Body>
7845 </Table.Body>
7846 );
7847 }
7848 }
7849 /* istanbul ignore next */
7850 function mapStateToProps(state) {
7851   return {
7852     app: state.app,
7853   };
7854 }
7855 }
7856 /* istanbul ignore next */
7857 function mapDispatchToProps(dispatch) {
7858   return {
7859     actions: bindActionCreators({ ...actions }, dispatch),
7860   };
7861 }
7862 }
7863 }
7864 export default connect(mapStateToProps, mapDispatchToProps)(
7865   RegistrarAssignSubjectLoad);
7866 import React, { Component } from 'react';
7867 import PropTypes from 'prop-types';
7868 import { bindActionCreators } from 'redux';
7869 import { connect } from 'react-redux';
7870 import * as actions from './redux/actions';
7871 import { Container, Grid } from 'tabler-react';
7872 import NavBar from './NavBar';
7873 import RegistrarAssignSubjectLoad from './RegistrarAssignSubjectLoad';
7874 import { Result, Button } from 'antd';
7875 import axios from 'axios';
7876 import { Link } from 'react-router-dom';
7877
7878 export class RegistrarAssignSubjectLoadView extends Component {
7879   static propTypes = {
7880     app: PropTypes.object.isRequired,
7881     actions: PropTypes.object.isRequired,
7882   };
7883   constructor(props) {
7884     super(props);
7885     this.state = {
7886       locked: false,
7887       isLoading: true,
7888     };
7889   }
7890   componentDidMount() {
7891     this.props.actions.setLoadingTrue();
7892     if (!this.props.app.auth.isAuthenticated) {
7893       this.props.history.push('/login');
7894     } else {
7895       if (this.props.app.auth.user.position != 2) {
7896         this.props.history.push('/page401');
7897       } else {
7898         axios
7899           .get('api/registrar/getsy')
7900           .then(res => {
7901             this.props.actions.setLoadingFalse();

```

```

7903     this.setState({ locked: false, isLoading: false });
7904   })
7905   .catch(err => {
7906     this.props.actions.setLoadingFalse();
7907     this.setState({ locked: true, isLoading: false });
7908   });
7909 }
7910 }
7911 }
7912
7913 render() {
7914   return (
7915     <div className="app-registrar-assign-subject-load-view fh">
7916       {this.state.isLoading ? (
7917         ,
7918       ) : this.state.locked ? (
7919         <NavBar>
7920           <Container>
7921             <Grid.Row>
7922               <Grid.Col xs={12} sm={12} md={12}>
7923                 <Result
7924                   status="403"
7925                   title="No Active School Year"
7926                   subTitle="Sorry, you are not authorized to access
7927                     this page now. Please contact your system
7928                     administrator for details."
7929                   extra={
7930                     <Link to="/">
7931                       <Button type="primary">Back Home</Button>
7932                     </Link>
7933                   }
7934                 </Grid.Col>
7935               </Grid.Row>
7936             </Container>
7937           </NavBar>
7938         ) : (
7939           <NavBar>
7940             <Container>
7941               <Grid.Row>
7942                 <Grid.Col xs={12} sm={12} md={12}>
7943                   <RegistrarAssignSubjectLoad />
7944                 </Grid.Col>
7945               </Grid.Row>
7946             </Container>
7947           </NavBar>
7948         )
7949       );
7950     );
7951   }
7952
7953 /* istanbul ignore next */
7954 function mapStateToProps(state) {
7955   return {
7956     app: state.app,
7957   };
7958 }
7959
7960 /* istanbul ignore next */
7961 function mapDispatchToProps(dispatch) {
7962   return {
7963     actions: bindActionCreators({ ...actions }, dispatch),
7964   };
7965 }
7966
7967 export default connect(mapStateToProps, mapDispatchToProps)(
7968   RegistrarAssignSubjectLoadView);
7969 import React, { Component } from 'react';
7970 import PropTypes from 'prop-types';
7971 import { bindActionCreators } from 'redux';
7972 import { connect } from 'react-redux';

```

```

7972 import * as actions from './redux/actions';
7973 import axios from 'axios';
7974 import { Pagination, Spin, Tooltip } from 'antd';
7975 import {
7976   Modal,
7977   Popconfirm,
7978   Search,
7979   Breadcrumb,
7980   AutoComplete,
7981   Input,
7982   message,
7983   Descriptions,
7984   Popover,
7985 } from 'antd';
7986 import cn from 'classnames';
7987 import placeholder from '../../../../../images/placeholder.jpg';
7988 import {
7989   Card,
7990   Button,
7991   Grid,
7992   Avatar,
7993   Table,
7994   Form,
7995   Header,
7996   Container,
7997   Text,
7998   Alert,
7999 } from 'tabler-react';
8000 import { Link } from 'react-router-dom';
8001 import { getImageUrl } from '../../../../../utils';
8002 import ViewEditLog from './ViewEditLog';
8003 const { Option } = AutoComplete;
8004
8005 export class RegistrarComponent extends Component {
8006   static propTypes = {
8007     app: PropTypes.object.isRequired,
8008     actions: PropTypes.object.isRequired,
8009   };
8010
8011   constructor(props) {
8012     super(props);
8013     this.state = {
8014       componentID: 0,
8015       component: '',
8016       grades: [],
8017       ave: [],
8018       isLoading: true,
8019       quarter: 'Q1',
8020       sectionName: '',
8021       subjectName: '',
8022       schoolYear: '',
8023       subjectCode: '',
8024     };
8025   }
8026
8027   componentDidMount() {
8028     this.setState({ isLoading: true });
8029     axios
8030       .post('api/registrar/getcomponents', {
8031         classRecordID: this.props.classRecordID,
8032         quarter: this.props.quarter,
8033       })
8034       .then(res => {
8035         this.setState({
8036           sectionName: res.data.sectionName,
8037           subjectName: res.data.subjectName,
8038           schoolYear: res.data.schoolYear,
8039           subjectCode: res.data.subjectCode,
8040         });
8041         axios
8042           .post('api/registrar/compinfo', {
8043             classRecordID: this.props.classRecordID,
8044             componentID: this.props.componentID,
8045             quarter: this.props.quarter,
8046           })

```

```

8047     .then(res2 => {
8048       this.setState({
8049         isLoading: false,
8050         componentID: this.props.componentID,
8051         component: res2.data.component,
8052         grades: res2.data.grades,
8053         ave: res2.data.ave,
8054         quarter: this.props.quarter,
8055       });
8056     });
8057   );
8058 }
8059
8060 render() {
8061   let headerData = [];
8062   let tableData = [];
8063   headerData.push(<Table.ColHeader colSpan={2}>Student Name</Table.
8064     ColHeader>);
8065   if (!this.state.isLoading) {
8066     for (const [index, value] of this.state.grades.entries()) {
8067       headerData.push(
8068         <Table.ColHeader alignContent="center">
8069           {value.name}
8070           <span style={{ marginLeft: '5px' }}>
8071             <Button outline size="sm" color="primary">
8072               <Link
8073                 to={
8074                   !this.props.locked
8075                     ? '/individualdeliberation/${this.props.id}/
8076                       managegrade/${this.props.classRecordID}/
8077                         quarter/${this.props.quarter}/comp/${this.
8078                           props.componentID}/subcomp/${value.subcompID}
8079                     }
8080                     : '/viewstudentrecord/classrecord/${this.props.
8081                       classRecordID}/q/${this.props.quarter}/comp/$
8082                         {this.props.componentID}/subcomp/${value.
8083                           subcompID}'
8084                 }
8085               >
8086                 View
8087               </Link>
8088             </Button>
8089           </span>
8090         </Table.ColHeader>,
8091       );
8092     }
8093     headerData.push(<Table.ColHeader alignContent="center">Percentage
8094       Score</Table.ColHeader>);
8095   for (const [index, value] of this.state.grades[0].data.entries())
8096     {
8097       let name = value.name;
8098       let imageUrl = value.imageUrl;
8099       let subsectstudID = value.subsectstudID;
8100       let tempRow = [];
8101       for (const [index, value] of this.state.grades.entries()) {
8102         let ps = value.data.find(value => value.subsectstudID ==
8103           subsectstudID).ps;
8104         tempRow.push(
8105           <Table.Col alignContent="center">
8106             {ps == -1 ? 'Not yet available' : Number(Math.round(ps +
8107               'e2') + 'e-2')}
8108           </Table.Col>,
8109         );
8110       }
8111       let ave = this.state.ave.find(value => value.subsectstudID ==
8112         subsectstudID).ws;
8113       tempRow.push(
8114         <Table.Col alignContent="center">
8115           {ave == -1 ? 'Not yet available' : Number(Math.round(ave +
8116             'e2') + 'e-2')}
8117           </Table.Col>,
8118         );
8119     }

```

```

8105     tableData.push(
8106       <Table.Row>
8107         <Table.Col className="w-1">
8108           <Avatar imageUrl={imageUrl == 'NA' ? placeholder : getImageUrl(imageUrl)} />
8109         </Table.Col>
8110         <Table.Col>{name}</Table.Col>
8111         {tempRow}
8112       </Table.Row>,
8113     );
8114   }
8115 }
8116
8117 return (
8118   <div className="app-teacher-component my-3 my-md-5">
8119     <Container>
8120       <Grid.Row>
8121         <Card>
8122           <Card.Body>
8123             {this.state.isLoading ? (
8124               <Spin spinning={true}></Spin>
8125             ) : (
8126               <div>
8127                 <Card.Title>
8128                   <Breadcrumb>
8129                     <Breadcrumb.Item>Subjects</Breadcrumb.Item>
8130                     <Breadcrumb.Item>View Subject Load</Breadcrumb.Item>
8131                     <Breadcrumb.Item>Manage Grades</Breadcrumb.Item>
8132                   <Breadcrumb.Item>
8133                     {this.state.sectionName} - {this.state.subjectName}
8134                   <Breadcrumb.Item>
8135                     <Breadcrumb.Item>{this.state.component}</Breadcrumb.Item>
8136               </Breadcrumb>
8137             </Card.Title>
8138             <Card.Title>
8139               <Header.H3>{this.state.component}</Header.H3>
8140             </Card.Title>
8141             <Card.Title>
8142               <Text.Small>
8143                 {this.state.sectionName} {this.props.quarter} S
8144                 .Y. {this.state.schoolYear}
8145               </Text.Small>
8146             </Card.Title>
8147           )
8148         </Card.Body>
8149         {this.state.isLoading ? (
8150           <Spin spinning={true}></Spin>
8151         ) : (
8152           <Card.Body>
8153             <Grid.Row>
8154               <Grid.Col sm={12} xs={12} md={12}>
8155                 <Descriptions
8156                   style={{ marginBottom: '15px', marginTop: '15px' }}
8157                   bordered
8158                   title="Subject Load Information"
8159                 >
8160                   <Descriptions.Item span={3} label="Section Name">
8161                     {this.state.sectionName}
8162                   </Descriptions.Item>
8163                   <Descriptions.Item span={3} label="Subject Name">
8164                     {this.state.subjectName}
8165                   </Descriptions.Item>
8166                   <Descriptions.Item span={3} label="Number of Students">
8167                     {tableData.length}

```

```

8168             </Descriptions.Item>
8169         </Descriptions>
8170     </Grid.Col>
8171 </Grid.Row>
8172 <Grid.Row>
8173     <Grid.Col sm={12} xs={12} md={12}>
8174         <Table highlightRowOnHover={true} responsive={true}>
8175             <Table.Header>{headerData}</Table.Header>
8176             <Table.Body>{tableData}</Table.Body>
8177         </Table>
8178     </Grid.Col>
8179 </Grid.Row>
8180 </Card.Body>
8181 )
8182 <Card.Footer>
8183     <Button.List align="right">
8184         {(
8185             this.props.app.auth.user.position == 2 ||
8186             this.props.app.auth.user.position == 3) && (
8187                 <ViewEditLog
8188                     classRecordID={this.props.classRecordID}
8189                     quarter={this.state.quarter}
8190                     position="Registrar"
8191                 />
8192             )
8193         )>
8194     </Button.List>
8195 </Card.Footer>
8196 </Card>
8197 </Grid.Row>
8198 </Container>
8199 </div>
8200 );
8201 }
8202 /* istanbul ignore next */
8203 function mapStateToProps(state) {
8204     return {
8205         app: state.app,
8206     };
8207 }
8208 /* istanbul ignore next */
8209 function mapDispatchToProps(dispatch) {
8210     return {
8211         actions: bindActionCreators({ ...actions }, dispatch),
8212     };
8213 }
8214 }
8215
8216 export default connect(mapStateToProps, mapDispatchToProps)(
8217     RegistrarComponent);
8218 import React, { Component } from 'react';
8219 import PropTypes from 'prop-types';
8220 import { bindActionCreators } from 'redux';
8221 import { connect } from 'react-redux';
8222 import * as actions from './redux/actions';
8223 import NavBar from './NavBar';
8224 import RegistrarComponent from './RegistrarComponent';
8225 import { Container, Grid } from 'table-react';
8226 import { Result, Button } from 'antd';
8227 import axios from 'axios';
8228 import { Link } from 'react-router-dom';
8229
8230 export class RegistrarComponentView extends Component {
8231     static propTypes = {
8232         app: PropTypes.object.isRequired,
8233         actions: PropTypes.object.isRequired,
8234     };
8235     constructor(props) {
8236         super(props);
8237         this.state = {

```

```

8238         locked: false,
8239         isLoading: true,
8240     );
8241 }
8242
8243 componentDidMount() {
8244     this.props.actions.setLoadingTrue();
8245     if (!this.props.app.auth.isAuthenticated) {
8246         this.props.history.push('/login');
8247     } else {
8248         if (this.props.app.auth.user.position != 2) {
8249             this.props.history.push('/page401');
8250         } else {
8251             axios
8252                 .get('api/registrar/getsy')
8253                 .then(res => {
8254                     this.props.actions.setLoadingFalse();
8255                     this.setState({ locked: false, isLoading: false });
8256                 })
8257                 .catch(err => {
8258                     this.props.actions.setLoadingFalse();
8259                     this.setState({ locked: true, isLoading: false });
8260                 });
8261     }
8262 }
8263
8264
8265 render() {
8266     return (
8267         <div className="app-registrar-component-view fh">
8268             {this.state.isLoading ? (
8269                 '',
8270             ) : this.state.locked ? (
8271                 <NavBar>
8272                     <Container>
8273                         <Grid.Row>
8274                             <Grid.Col xs={12} sm={12} md={12}>
8275                                 <Result
8276                                     status="403"
8277                                     title="No Active School Year"
8278                                     subTitle="Sorry, you are not authorized to access
8279                                         this page now. Please contact your system
8280                                         administrator for details."
8281                                     extra={
8282                                         <Link to="/">
8283                                             <Button type="primary">Back Home</Button>
8284                                         </Link>
8285                                     }
8286                                 />
8287                             </Grid.Col>
8288                         </Grid.Row>
8289                     </Container>
8290                 </NavBar>
8291             ) : (
8292                 <NavBar>
8293                     <Container>
8294                         <Grid.Row>
8295                             <Grid.Col xs={12} sm={12} md={12}>
8296                                 <RegistrarComponent
8297                                     classRecordID={this.props.match.params.
8298                                         classRecordID}
8299                                     componentID={this.props.match.params.comp}
8300                                     quarter={this.props.match.params.Q}
8301                                     id={this.props.match.params.id}
8302                                     locked={false}
8303                                 />
8304                             </Grid.Col>
8305                         </Grid.Row>
8306                     </Container>
8307                 </NavBar>
8308             )}
8309         </div>

```

```

8307         );
8308     }
8309 }
8310
8311 /* istanbul ignore next */
8312 function mapStateToProps(state) {
8313     return {
8314         app: state.app,
8315     };
8316 }
8317
8318 /* istanbul ignore next */
8319 function mapDispatchToProps(dispatch) {
8320     return {
8321         actions: bindActionCreators({ ...actions }, dispatch),
8322     };
8323 }
8324
8325 export default connect(mapStateToProps, mapDispatchToProps)(
8326     RegistrarComponentView);
8327 import React, { Component } from 'react';
8328 import PropTypes from 'prop-types';
8329 import { bindActionCreators } from 'redux';
8330 import { connect } from 'react-redux';
8331 import * as actions from './redux/actions';
8332 import { Container, Grid } from 'tabler-react';
8333 import RegistrarSetSubmissionDeadline from './
8334     RegistrarSetSubmissionDeadline';
8335 import SchoolYearInfo from './SchoolYearInfo';
8336 import RegistrarViewUpdateLog from './RegistrarViewUpdateLog';
8337 import { Spin } from 'antd';
8338
8339 export class RegistrarDashboard extends Component {
8340     static propTypes = {
8341         app: PropTypes.object.isRequired,
8342         actions: PropTypes.object.isRequired,
8343     };
8344     constructor(props) {
8345         super(props);
8346         this.state = {
8347             isLoading: false,
8348         };
8349     }
8350     render() {
8351         return (
8352             <div className="app-admin-profile my-3 my-md-5">
8353                 <Spin spinning={this.props.app.showLoading}>
8354                     <Container>
8355                         <Grid.Row>
8356                             <Grid.Col sm={12} xs={12} md={6} lg={6}>
8357                                 <RegistrarSetSubmissionDeadline />
8358                             </Grid.Col>
8359                         </Grid.Row>
8360                         <Grid.Col sm={12} xs={12} md={6} lg={6}>
8361                             <Grid.Row>
8362                                 <SchoolYearInfo />
8363                             </Grid.Row>
8364                         </Grid.Col>
8365                     </Grid.Row>
8366                 </Container>
8367                 </Spin>
8368             </div>
8369         );
8370     }
8371 }
8372
8373 /* istanbul ignore next */
8374 function mapStateToProps(state) {
8375     return {
8376         app: state.app,

```

```

8377     };
8378 }
8379
8380 /* istanbul ignore next */
8381 function mapDispatchToProps(dispatch) {
8382     return {
8383         actions: bindActionCreators({ ...actions }, dispatch),
8384     };
8385 }
8386
8387 export default connect(mapStateToProps, mapDispatchToProps)(
8388     RegistrarDashboard);
8389 import React, { Component } from 'react';
8390 import PropTypes from 'prop-types';
8391 import { bindActionCreators } from 'redux';
8392 import { connect } from 'react-redux';
8393 import * as actions from './redux/actions';
8394 import axios from 'axios';
8395 import { Pagination, Spin, Tooltip } from 'antd';
8396 import {
8397     Modal,
8398     Popconfirm,
8399     Search,
8400     Breadcrumb,
8401     AutoComplete,
8402     Input,
8403     message,
8404     Descriptions,
8405     Popover,
8406 } from 'antd';
8407 import cn from 'classnames';
8408 import placeholder from '../../images/placeholder.jpg';
8409 import {
8410     Card,
8411     Button,
8412     Grid,
8413     Avatar,
8414     Table,
8415     Form,
8416     Header,
8417     Container,
8418     Text,
8419     Alert,
8420 } from 'tabler-react';
8421 import { Link } from 'react-router-dom';
8422 import moment from 'moment';
8423 import { getImageUrl } from '../../utils';
8424 const { Option } = AutoComplete;
8425
8426 export class RegistrarEditRecord extends Component {
8427     static propTypes = {
8428         app: PropTypes.object.isRequired,
8429         actions: PropTypes.object.isRequired,
8430     };
8431     constructor(props) {
8432         super(props);
8433         this.state = {
8434             componentID: 0,
8435             component: '',
8436             subcompName: '',
8437             subcompID: 0,
8438             isLoading: true,
8439             quarter: 'Q1',
8440             sectionName: '',
8441             subjectName: '',
8442             schoolYear: '',
8443             subjectCode: '',
8444             studList: [],
8445             dateGiven: new Date(),
8446             total: 0,
8447             recID: 0,
8448             classRecordID: 0,
8449     };
8450

```

```

8451     this.editRecord = this.editRecord.bind(this);
8452 }
8453
8454 editRecord() {
8455   const {
8456     description,
8457     dateGiven,
8458     total,
8459     classRecordID,
8460     subcompID,
8461     componentID,
8462     quarter,
8463     studList,
8464   } = this.state;
8465   const payload = studList;
8466   this.props.actions.editRecord(
8467     {
8468       description,
8469       dateGiven,
8470       total,
8471       classRecordID,
8472       subcompID,
8473       componentID,
8474       quarter,
8475       payload,
8476     },
8477     'Registrar',
8478   );
8479 }
8480
8481 componentDidMount() {
8482   this.setState({ isLoading: true });
8483   axios
8484     .post('api/registrar/getcomponents', {
8485       classRecordID: this.props.classRecordID,
8486       quarter: this.props.quarter,
8487     })
8488     .then(res => {
8489       this.setState({
8490         sectionName: res.data.sectionName,
8491         subjectName: res.data.subjectName,
8492         schoolYear: res.data.schoolYear,
8493         subjectCode: res.data.subjectCode,
8494       });
8495       axios
8496         .post('api/registrar/getrecinfo', {
8497           classRecordID: this.props.classRecordID,
8498           componentID: this.props.componentID,
8499           subcompID: this.props.subcompID,
8500           quarter: this.props.quarter,
8501           gradeID: this.props.rec,
8502         })
8503         .then(res => {
8504           this.setState({
8505             componentID: res.data.componentID,
8506             component: res.data.component,
8507             subcompName: res.data.subcompName,
8508             subcompID: res.data.subcompID,
8509             dateGiven: res.data.dateGiven,
8510             total: res.data.total,
8511             description: res.data.description,
8512             studList: res.data.data,
8513             isLoading: false,
8514             classRecordID: this.props.classRecordID,
8515             quarter: this.props.quarter,
8516             recID: this.props.rec,
8517           });
8518         });
8519       });
8520     }
8521
8522   render() {
8523     const birthDate = new Date(this.state.dateGiven);
8524     const defaultDate = birthDate.toISOString();
8525     let tableData = [];

```

```

8526     for (const [index, value] of this.state.studList.entries()) {
8527       tableData.push(
8528         <Table.Row>
8529           <Table.Col className="w-1">
8530             <Avatar imageUrl={value.imageUrl == 'NA' ? placeholder : getimageUrl(value.imageUrl)} />
8531           </Table.Col>
8532           <Table.Col>{value.name}</Table.Col>
8533           <Table.Col>
8534             <Form.Group>
8535               <Form.Input
8536                 placeholder="Score"
8537                 position="append"
8538                 value={value.score}
8539                 onChange={e => {
8540                   const reg = /^-?[0-9]*(\.[0-9]*)?$/;
8541                   let temparr = this.state.studList;
8542
8543                   if (
8544                     isNaN(e.target.value) && reg.test(e.target.value)
8545                     ) ||
8546                     (e.target.value === '' && e.target.value !== '-')
8547                     ||
8548                     e.target.value == 'A' ||
8549                     e.target.value == 'E'
8550                   ) {
8551                     temparr[index].score = e.target.value;
8552                     this.setState({ studList: temparr });
8553                   }
8554                 }>
8555               </Form.Group>
8556             </Table.Col>
8557           </Table.Row>,
8558         );
8559       return (
8560         <div className="app-teacher-add-record my-3 my-md-5">
8561           <Container>
8562             <Grid.Row>
8563               <Card>
8564                 <Card.Body>
8565                   {this.state.isLoading ? (
8566                     <Spin spinning={true}></Spin>
8567                   ) : (
8568                     <div>
8569                       <Card.Title>
8570                         <Breadcrumb>
8571                           <Breadcrumb.Item>Subjects</Breadcrumb.Item>
8572                           <Breadcrumb.Item>View Subject Load</Breadcrumb.
8573                             Item>
8574                           <Breadcrumb.Item>Manage Grades</Breadcrumb.Item
8575                             >
8576                           <Breadcrumb.Item>
8577                             {this.state.sectionName} - {this.state.
8578                               subjectName}
8579                           </Breadcrumb.Item>
8580                           <Breadcrumb.Item>{this.state.component}</
8581                             Breadcrumb.Item>
8582                           <Breadcrumb.Item>{this.state.subcompName}</
8583                             Breadcrumb.Item>
8584                         </Breadcrumb>
8585                       </Card.Title>
8586                       <Card.Title>
8587                         <Header.H3>
8588                           {this.state.component} - {this.state.

```

```

8589             </Text.Small>
8590         </Card.Title>
8591     </div>
8592   )
8593   </Card.Body>
8594 </Card>
8595 </Grid.Row>
8596 <Grid.Row>
8597   <Grid.Col sm={12} xs={12} md={6}>
8598     <Spin spinning={this.state.isLoading}>
8599       <Card>
8600         <Card.Body>
8601           <Grid.Row>
8602             <Grid.Col sm={12} xs={12} md={12}>
8603               <Descriptions
8604                 style={{ marginBottom: '15px', marginTop: '15
8605                   px' }}
8606                 bordered
8607                 title="Edit Record"
8608               >
8609                 <Descriptions.Item span={3} label="Item
8610                   Description">
8611                   <Form.Group>
8612                     <Form.Input
8613                       placeholder="Description"
8614                       value={this.state.description}
8615                       onChange={e => {
8616                         this.setState({ description: e.target
8617                           .value });
8618                       }
8619                     />
8620                   </Form.Group>
8621                 </Descriptions.Item>
8622                 <Descriptions.Item span={3} label="Date Given
8623                   ">
8624                   <Form.Group>
8625                     <Form.DatePicker
8626                       defaultDate={new Date(defaultDate)}
8627                       format="mm/dd/yyyy"
8628                       onChange={e => {
8629                         this.setState({ dateGiven: e });
8630                       }
8631                     name="birthDate"
8632                     maxYear={2020}
8633                     minYear={1897}
8634                     monthLabels={[
8635                       'January',
8636                       'February',
8637                       'March',
8638                       'April',
8639                       'May',
8640                       'June',
8641                       'July',
8642                       'August',
8643                       'September',
8644                       'October',
8645                       'November',
8646                       'December',
8647                     ]}
8648                   ></Form.DatePicker>
8649                 </Form.Group>
8650               </Descriptions.Item>
8651               <Descriptions.Item span={3} label="Total
8652                 number of items">
8653                 <Form.Group>
8654                   <Form.Input
8655                     placeholder="Total"
8656                     value={this.state.total}
8657                     onChange={e => {
8658                       const reg = /^-?[0-9]*(\.[0-9]*)?$/;
8659                       if (
8660                         (!isNaN(e.target.value) && reg.test
8661                           (e.target.value)) ||

```

```

8657             e.target.value === '' ||  

8658             e.target.value === '--'  

8659         ) {  

8660             this.setState({ total: e.target.  

8661                 value });
8662         }
8663     }
8664   
```

```

8665     </Form.Group>  

8666     </Descriptions.Item>
8667   
```

```

8668     </Grid.Col>
8669   
```

```

8670   
```

```

8671     </Spin>
8672   
```

```

8673   
```

```

8674     <Spin spinning={this.state.isLoading}>
8675   
```

```

8676     <Card>
8677       <Card.Body>
8678         <Card.Title>Student Scores</Card.Title>
8679         <Container>
8680           <Alert type="primary">
8681             <b>Note:</b> Scores with value of <b>E</b> are  

8682               considered{' '}
8683             <i>
8684               <b>excused</b>
8685             </i>
8686             . Enter a value of <b>A</b> for students who  

8687               are{' '}
8688             <i>
8689               <b>absent</b>
8690             </i>
8691             .
8692           </Alert>
8693         </Container>
8694         <Table highlightRowOnHover={true} responsive={true
8695           }>
8696           <Table.Header>
8697             <Table.ColHeader colSpan={2}>Student Name</
8698               Table.ColHeader>
8699             <Table.ColHeader>Score</Table.ColHeader>
8700           </Table.Header>
8701           <Table.Body>{tableData}</Table.Body>
8702         </Table>
8703       
```

```

8704     <Card.Body>
8705       <Card.Footer>
8706         <Button.List align="right">
8707           <Button icon="file" color="success" onClick={this
8708             .editRecord}>
8709             Edit Record
8710           </Button>
8711         </Card.Footer>
8712       </Card>
8713     </Spin>
8714   
```

```

8715   
```

```

8716   
```

```

8717     /* istanbul ignore next */
8718     function mapStateToProps(state) {
8719       return {
8720         app: state.app,
8721       };
8722     }

```

```

8723 /* istanbul ignore next */
8724 function mapDispatchToProps(dispatch) {
8725   return {
8726     actions: bindActionCreators({ ...actions }, dispatch),
8727   };
8728 }
8729
8730 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarEditRecord);
8731 import React, { Component } from 'react';
8732 import PropTypes from 'prop-types';
8733 import { bindActionCreators } from 'redux';
8734 import { connect } from 'react-redux';
8735 import * as actions from './redux/actions';
8736 import NavBar from './NavBar';
8737 import RegistrarEditRecord from './RegistrarEditRecord';
8738 import { Container, Grid } from 'tabler-react';
8739 import { Result, Button } from 'antd';
8740 import axios from 'axios';
8741 import { Link } from 'react-router-dom';
8742
8743 export class RegistrarEditRecordView extends Component {
8744   static propTypes = {
8745     app: PropTypes.object.isRequired,
8746     actions: PropTypes.object.isRequired,
8747   };
8748
8749   constructor(props) {
8750     super(props);
8751     this.state = {
8752       locked: false,
8753       isLoading: true,
8754     };
8755   }
8756
8757   componentDidMount() {
8758     this.props.actions.setLoadingTrue();
8759     if (!this.props.app.auth.isAuthenticated) {
8760       this.props.history.push('/login');
8761     } else {
8762       if (this.props.app.auth.user.position != 2) {
8763         this.props.history.push('/page401');
8764       } else {
8765         axios
8766           .get('api/registrar/getsy')
8767           .then(res => {
8768             this.props.actions.setLoadingFalse();
8769             this.setState({ locked: false, isLoading: false });
8770           })
8771           .catch(err => {
8772             this.props.actions.setLoadingFalse();
8773             this.setState({ locked: true, isLoading: false });
8774           });
8775       }
8776     }
8777   }
8778
8779   render() {
8780     return (
8781       <div className="app-registrar-edit-record-view fh">
8782         {this.state.isLoading ? (
8783           '',
8784         ) : this.state.locked ? (
8785           <NavBar>
8786             <Container>
8787               <Grid.Row>
8788                 <Grid.Col xs={12} sm={12} md={12}>
8789                   <Result
8790                     status="403"
8791                     title="No Active School Year"
8792                     subTitle="Sorry, you are not authorized to access
8793                     this page now. Please contact your system
8794                     administrator for details."

```

```

8793             extra={
8794                 <Link to="/">
8795                     <Button type="primary">Back Home</Button>
8796                 </Link>
8797             }
8798         />
8799     </Grid.Col>
8800   </Grid.Row>
8801 </Container>
8802 </NavBar>
8803 ) : (
8804     <NavBar>
8805       <Container>
8806         <Grid.Row>
8807           <Grid.Col xs={12} sm={12} md={12}>
8808             <RegistrarEditRecord
8809               classRecordID={this.props.match.params.
8810                 classRecordID}
8811               componentID={this.props.match.params.comp}
8812               subcompID={this.props.match.params.subcomp}
8813               quarter={this.props.match.params.Q}
8814               id={this.props.match.params.id}
8815               history={this.props.history}
8816               rec={this.props.match.params.rec}
8817             />
8818           </Grid.Col>
8819         </Grid.Row>
8820       </Container>
8821     </NavBar>
8822   )
8823 );
8824 }
8825 }
8826 /* istanbul ignore next */
8827 function mapStateToProps(state) {
8828   return {
8829     app: state.app,
8830   };
8831 }
8832 }
8833 /* istanbul ignore next */
8834 function mapDispatchToProps(dispatch) {
8835   return {
8836     actions: bindActionCreators({ ...actions }, dispatch),
8837   };
8838 }
8839 }
8840
8841 export default connect(mapStateToProps, mapDispatchToProps)(
8842   RegistrarEditRecordView);
8843 import React, { Component } from 'react';
8844 import PropTypes from 'prop-types';
8845 import { bindActionCreators } from 'redux';
8846 import { connect } from 'react-redux';
8847 import * as actions from './redux/actions';
8848 import { Link } from 'react-router-dom';
8849 import { Card, Button, Grid, Avatar, Table, Form, Header, Container } from 'tabler-react';
8850 import axios from 'axios';
8851 import { Pagination, Spin, Tooltip } from 'antd';
8852 import {
8853   Modal,
8854   Popconfirm,
8855   Search,
8856   Breadcrumb,
8857   AutoComplete,
8858   Input,
8859   message,
8860   Descriptions,
8861 } from 'antd';
8862 import RegistrarAddNewLoad from './RegistrarAddNewLoad';

```

```

8862 import cn from 'classnames';
8863 import placeholder from '../../images/placeholder.jpg';
8864 import bg from '../../images/BG.png';
8865 import { getImageUrl } from '../../utils';
8866 const { Option } = AutoComplete;
8867
8868 function ProfileImage({ avatarURL }) {
8869   return <img className="card-profile-img" alt="Profile" src={avatarURL
8870     } />;
8871 }
8872
8873 function Profile({ className, children, name, avatarURL = '',
8874   backgroundURL = '', bio }) {
8875   const classes = cn('card-profile', className);
8876   return (
8877     <Card className={classes}>
8878       <Card.Header backgroundURL={backgroundURL} />
8879       <Card.Body className="text-center">
8880         <ProfileImage avatarURL={avatarURL} />
8881         <Header.H3 className="mb-3">{name}</Header.H3>
8882         <p className="mb-4">{bio || children}</p>
8883       </Card.Body>
8884     </Card>
8885   );
8886 }
8887
8888 export class RegistrarEditSubjectLoad extends Component {
8889   static propTypes = {
8890     app: PropTypes.object.isRequired,
8891     actions: PropTypes.object.isRequired,
8892   };
8893
8894   constructor(props) {
8895     super(props);
8896     this.state = {
8897       isLoading: true,
8898       isLoadingTable: true,
8899       name: '',
8900       email: '',
8901       accountID: '',
8902       imageUrl: '',
8903       data: [],
8904       gradeLevel: 'N',
8905       sectionName: '',
8906       subjectName: '',
8907       selectedKey: -1,
8908       options: [],
8909       optionLoading: false,
8910     };
8911
8912     this.handleSearch = this.handleSearch.bind(this);
8913     this.onSelect = this.onSelect.bind(this);
8914     this.addStudent = this.addStudent.bind(this);
8915   }
8916
8917   addStudent() {
8918     this.props.actions.addSubjectSectionStudent({
8919       studsectID: this.state.selectedKey,
8920       subsectID: this.props.subsectID,
8921     });
8922
8923   deleteStudent(key) {
8924     this.props.actions.deleteSubjectSectionStudent({ subsectstudID: key
8925       });
8926   }
8927
8928   onSelect(key) {
8929     this.setState({ selectedKey: key });
8930   }
8931
8932   handleSearch(query) {
8933     this.setState({ selectedKey: -1 });

```

```

8932     this.setState({
8933       options: [],
8934       optionLoading: true,
8935     });
8936     axios
8937       .post('api/registrar/searchenrolled', { keyword: query })
8938       .then(res => {
8939         if (query == '') {
8940           this.setState({ options: [], optionLoading: false });
8941         } else {
8942           this.setState({ options: res.data.studentList, optionLoading:
8943             false });
8944         }
8945       })
8946       .catch(err => {
8947         this.setState({
8948           options: [],
8949           optionLoading: false,
8950         });
8951       });
8952     }
8953   componentDidMount() {
8954     axios.post('api/registrar/userinfo', { accountID: this.props.
8955       accountID }).then(res => {
8956       this.setState({
8957         email: res.data.email,
8958         imageUrl: res.data.imageUrl,
8959         name: res.data.name,
8960       });
8961     axios.post('api/registrar/getsubsectinfo', { subsectID: this.
8962       props.subsectID }).then(res2 => {
8963       this.setState({
8964         isLoading: false,
8965         isLoadingTable: false,
8966         data: res2.data.studentList,
8967         gradeLevel: res2.data.gradeLevel,
8968         sectionName: res2.data.sectionName,
8969         subjectName: res2.data.subjectName,
8970       });
8971     });
8972   }
8973   componentWillMount() {
8974     this.setState({ isLoadingTable: true });
8975     axios.post('api/registrar/getsubsectinfo', { subsectID: this.props.
8976       subsectID }).then(res2 => {
8977       this.setState({
8978         isLoadingTable: false,
8979         data: res2.data.studentList,
8980         gradeLevel: res2.data.gradeLevel,
8981         sectionName: res2.data.sectionName,
8982         subjectName: res2.data.subjectName,
8983       });
8984     });
8985   }
8986   render() {
8987     const displayGradeLevel = gradeLevel => {
8988       switch (gradeLevel) {
8989         case 'N':
8990           return 'Nursery';
8991         case 'K1':
8992           return 'Kinder 1';
8993         case 'K2':
8994           return 'Kinder 2';
8995         case 'G1':
8996           return 'Grade 1';
8997         case 'G2':
8998           return 'Grade 2';
8999         case 'G3':
9000           return 'Grade 3';
9001         case 'G4':

```

```

9002         return 'Grade 4';
9003     case 'G5':
9004         return 'Grade 5';
9005     case 'G6':
9006         return 'Grade 6';
9007     case 'G7':
9008         return 'Grade 7';
9009     case 'G8':
9010         return 'Grade 8';
9011     case 'G9':
9012         return 'Grade 9';
9013     case 'G10':
9014         return 'Grade 10';
9015     case 'G11':
9016         return 'Grade 11';
9017     case 'G12':
9018         return 'Grade 12';
9019     default:
9020         return '';
9021     }
9022   };
9023   let displayOptions = [];
9024   const displayStudentData = [];
9025   for (const [index, value] of this.state.options.entries()) {
9026     displayOptions.push(
9027       <Option key={value.studsectID} text={value.name}>
9028         <div>
9029           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getImageUrl(value.imageUrl)} />
9030           <span style={{ margin: '16px', verticalAlign: 'text-top' }}>{value.name}</span>
9031         </div>
9032       </Option>,
9033     );
9034   }
9035   for (const [index, value] of this.state.data.entries()) {
9036     displayStudentData.push(
9037       <Table.Row>
9038         <Table.Col className="w-1">
9039           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getImageUrl(value.imageUrl)} />
9040         </Table.Col>
9041         <Table.Col>{value.name}</Table.Col>
9042         <Table.Col>{value.email}</Table.Col>
9043         <Table.Col>{value.sectionName}</Table.Col>
9044         <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
9045         <Table.Col alignContent="center">
9046           <Popconfirm
9047             title="Do you want to drop this student?"
9048             onConfirm={() => this.deleteStudent(value.key)}
9049             okText="Delete"
9050             cancelText="Cancel"
9051           >
9052             <Button pill size="sm" icon="trash" color="danger"></
9053               Button>
9054             </Popconfirm>
9055           </Table.Col>
9056         </Table.Row>,
9057       );
9058     }
9059     if (displayStudentData.length == 0) {
9060       displayStudentData.push(
9061         <Table.Row>
9062           <Table.Col colSpan={6} alignContent="center">
9063             No result.
9064           </Table.Col>
9065         </Table.Row>,
9066       );
9067     }
9068     if (this.state.optionLoading) {
9069       displayOptions = [
9070         <Option key={0} text="">
9071           <div>
9072             <Spin spinning={true}></Spin>

```

```

9072         </div>
9073     </Option>,
9074   ];
9075 } else {
9076   if (displayOptions.length == 0) {
9077     displayOptions = [
9078       <Option key={0} text="">
9079         <div>No data.</div>
9080       </Option>,
9081     ];
9082   }
9083 }
9084 return (
9085   <div className="app-registrar-edit-subject-load my-3 my-md-5">
9086     <Container>
9087       <Grid.Row>
9088         <Grid.Col sm={12} lg={4}>
9089           {this.state.isLoading ? (
9090             <Spin spinning={this.state.isLoading}>
9091               <Profile name="" avatarURL={placeholder}>
9092                 backgroundURL={bg}></Profile>
9093             ) : (
9094               <Profile
9095                 name={this.state.name}
9096                 avatarURL={
9097                   this.state.imageUrl === 'NA' ? placeholder :
9098                     getImageUrl(this.state.imageUrl)
9099                 }
9100                 backgroundURL={bg}
9101               ></Profile>
9102             )
9103           </Grid.Col>
9104           <Grid.Col sm={12} lg={8}>
9105             <Card>
9106               <Card.Body>
9107                 {this.state.isLoading ? (
9108                   '',
9109                 ) : (
9110                   <div>
9111                     <Card.Title>
9112                       <Breadcrumb>
9113                         <Breadcrumb.Item>Teachers</Breadcrumb.Item>
9114                         <Breadcrumb.Item>Assign Subject Load</
9115                           <Breadcrumb.Item>
9116                             <Breadcrumb.Item>Edit</Breadcrumb.Item>
9117                               <Breadcrumb.Item>{this.state.subjectName}</
9118                                 <Breadcrumb.Item>
9119                               </Breadcrumb>
9120                             </Card.Title>
9121                           </div>
9122             )
9123             <Descriptions
9124               style={{ marginBottom: '15px', marginTop: '15px' }}
9125               bordered
9126               title="Subject Load Information"
9127             >
9128               <Descriptions.Item span={3} label="Section Name">
9129                 {this.state.sectionName}
9130               </Descriptions.Item>
9131               <Descriptions.Item span={3} label="Grade Level">
9132                 {displayGradeLevel(this.state.gradeLevel)}
9133               </Descriptions.Item>
9134               <Descriptions.Item span={3} label="Subject Name">
9135                 {this.state.subjectName}
9136               </Descriptions.Item>
9137               <Descriptions.Item span={3} label="Teacher">
9138                 {this.state.name}
               </Descriptions.Item>

```

```

9139         <Descriptions.Item span={3} label="Number of
9140             Students">
9141                 {this.state.data.length}
9142             </Descriptions.Item>
9143         </Descriptions>
9144         <Grid.Row>
9145             <Grid.Col sm={12} xs={12} md={10}>
9146                 <AutoComplete
9147                     style={{ width: '100%', marginBottom: '10px' }}
9148                     optionLabelProp="text"
9149                     onSearch={this.handleSearch}
9150                     dataSource={displayOptions}
9151                     onSelect={this.onSelect}
9152                 >
9153                     <Input placeholder="Search for students"
9154                         enterButton />
9155                 </AutoComplete>
9156             </Grid.Col>
9157             <Grid.Col sm={12} xs={12} md={2}>
9158                 <Button
9159                     block
9160                     icon="plus"
9161                     color="primary"
9162                     onClick={() => this.addStudent()}
9163                     disabled={this.state.selectedKey == -1}
9164                 >
9165                     Add
9166                 </Button>
9167             </Grid.Col>
9168         </Grid.Row>
9169     </Grid.Row>
9170     <Grid.Col xs={12} sm={12} md={12}>
9171         <Spin spinning={this.state.isLoadingTable}>
9172             <Table highlightRowOnHover={true} responsive={
9173                 true}>
9174                 <Table.Header>
9175                     <Table.ColHeader alignContent="center"
9176                         colSpan={2}>
9177                         Student Name
9178                     </Table.ColHeader>
9179                     <Table.ColHeader>Email</Table.ColHeader>
9180                     <Table.ColHeader>Section Name</Table.
9181                         ColHeader>
9182                     <Table.ColHeader>Grade Level</Table.
9183                         ColHeader>
9184                     <Table.ColHeader alignContent="center">
9185                         Actions</Table.ColHeader>
9186                 </Table.Header>
9187                 <Table.Body>{displayStudentData}</Table.Body>
9188             </Table>
9189         </Spin>
9190     </Grid.Col>
9191     </Grid.Row>
9192     </Container>
9193     </div>
9194     );
9195 }
9196 /* istanbul ignore next */
9197 function mapStateToProps(state) {
9198     return {
9199         app: state.app,
9200     };
9201 }
9202 /* istanbul ignore next */
9203 function mapDispatchToProps(dispatch) {
9204     return {

```

```

9205     actions: bindActionCreators({ ...actions }, dispatch),
9206   };
9207 }
9208
9209 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarEditSubjectLoad);
9210 import React, { Component } from 'react';
9211 import PropTypes from 'prop-types';
9212 import { bindActionCreators } from 'redux';
9213 import { connect } from 'react-redux';
9214 import * as actions from './redux/actions';
9215 import NavBar from './NavBar';
9216 import RegistrarEditSubjectLoad from './RegistrarEditSubjectLoad';
9217 import { Container, Grid } from 'tabler-react';
9218 import { Result, Button } from 'antd';
9219 import axios from 'axios';
9220 import { Link } from 'react-router-dom';
9221
9222 export class RegistrarEditSubjectLoadView extends Component {
9223   static propTypes = {
9224     app: PropTypes.object.isRequired,
9225     actions: PropTypes.object.isRequired,
9226   };
9227
9228   constructor(props) {
9229     super(props);
9230     this.state = {
9231       locked: false,
9232       isLoading: true,
9233     };
9234   }
9235
9236   componentDidMount() {
9237     this.props.actions.setLoadingTrue();
9238     if (!this.props.app.auth.isAuthenticated) {
9239       this.props.history.push('/login');
9240     } else {
9241       if (this.props.app.auth.user.position != 2) {
9242         this.props.history.push('/page401');
9243       } else {
9244         axios
9245           .get('api/registrar/getsy')
9246           .then(res => {
9247             this.props.actions.setLoadingFalse();
9248             this.setState({ locked: false, isLoading: false });
9249           })
9250           .catch(err => {
9251             this.props.actions.setLoadingFalse();
9252             this.setState({ locked: true, isLoading: false });
9253           });
9254       }
9255     }
9256   }
9257
9258   render() {
9259     return (
9260       <div className="app-registrar-edit-subject-load-view fh">
9261         {this.state.isLoading ? (
9262           '',
9263         ) : this.state.locked ? (
9264           <NavBar>
9265             <Container>
9266               <Grid.Row>
9267                 <Grid.Col xs={12} sm={12} md={12}>
9268                   <Result
9269                     status="403"
9270                     title="No Active School Year"
9271                     subTitle="Sorry, you are not authorized to access
9272                       this page now. Please contact your system
9273                         administrator for details."
9274                     extra={
9275                       <Link to="/">
9276                         <Button type="primary">Back Home</Button>

```

```

9275             </Link>
9276         }
9277     }
9278     </Grid.Col>
9279   </Grid.Row>
9280   </Container>
9281 </NavBar>
9282 ) : (
9283   <NavBar>
9284     <Container>
9285       <Grid.Row>
9286         <Grid.Col xs={12} sm={12} md={12}>
9287           <RegistrarEditSubjectLoad
9288             accountID={this.props.match.params.id}
9289             subsectID={this.props.match.params.id2}
9290           />
9291         </Grid.Col>
9292       </Grid.Row>
9293     </Container>
9294   </NavBar>
9295 )
9296 </div>
9297 );
9298 }
9299 }
9300 */
9301 function mapStateToProps(state) {
9302   return {
9303     app: state.app,
9304   };
9305 }
9306 */
9307 */
9308 function mapDispatchToProps(dispatch) {
9309   return {
9310     actions: bindActionCreators({ ...actions }, dispatch),
9311   };
9312 }
9313 */
9314
9315 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarEditSubjectLoadView);
9316 import React, { Component } from 'react';
9317 import PropTypes from 'prop-types';
9318 import { bindActionCreators } from 'redux';
9319 import { connect } from 'react-redux';
9320 import * as actions from './redux/actions';
9321 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
  from 'tabler-react';
9322 import axios from 'axios';
9323 import { Pagination, Spin } from 'antd';
9324 import { Breadcrumb } from 'antd';
9325 import { Link } from 'react-router-dom';
9326
9327 export class RegistrarGroupDeliberation extends Component {
9328   static propTypes = {
9329     app: PropTypes.object.isRequired,
9330     actions: PropTypes.object.isRequired,
9331   };
9332
9333   constructor(props) {
9334     super(props);
9335     this.state = {
9336       isLoading: false,
9337       keyword: '',
9338       page: 1,
9339       pageSize: 10,
9340       numOfPages: 1,
9341       data: [],
9342       showAddSection: false,
9343       addSectionLoading: false,
9344       errors: {},
9345       sectionName: ''
9346     };
9347   }
9348
9349   componentDidMount() {
9350     this.fetchData();
9351   }
9352
9353   componentDidUpdate(prevProps) {
9354     if (prevProps.app !== this.props.app) {
9355       this.fetchData();
9356     }
9357   }
9358
9359   fetchData() {
9360     const { app, actions } = this.props;
9361     const { keyword, page, pageSize } = this.state;
9362
9363     axios.get(`https://api.example.com/search?keyword=${keyword}&page=${page}&pageSize=${pageSize}`)
9364       .then(response => {
9365         const { data } = response;
9366
9367         actions.setSearchResults(data);
9368       })
9369       .catch(error => {
9370         console.error(error);
9371       });
9372   }
9373
9374   render() {
9375     const { app, isLoading, keyword, page, pageSize, data, errors, sectionName } = this.state;
9376
9377     return (
9378       <div>
9379         <Grid>
9380           <Grid.Col xs={12}>
9381             <Card>
9382               <Form>
9383                 <Input type="text" value={keyword} onChange={e => this.setState({ keyword: e.target.value })} />
9384                 <Button type="button" onClick={this.fetchData}>Search</Button>
9385               </Form>
9386             </Card>
9387           </Grid.Col>
9388         </Grid>
9389       </div>
9390     );
9391   }
9392 }
9393
9394 
```

```

9346     gradeLevel: 'N',
9347     showEditSection: false,
9348     editSectionLoading: false,
9349     selectedKey: 0,
9350   );
9351   this.onChange = this.onChange.bind(this);
9352   this.onChangeSearch = this.onChangeSearch.bind(this),
9353 }
9354
9355 onChange(event) {
9356   this.setState({ [event.target.name]: event.target.value });
9357 }
9358
9359 onChangeSearch(event) {
9360   this.setState({ [event.target.name]: event.target.value });
9361   this.setState({ isLoading: true, page: 1 });
9362   axios
9363     .post('api/registrar/getsections', {
9364       keyword: event.target.value,
9365       page: 1,
9366       pageSize: this.state.pageSize,
9367     })
9368     .then(res => {
9369       this.setState({ isLoading: false });
9370       this.setState({ numOfPages: res.data.numOfPages, data: res.data
9371         .sectionList });
9372     })
9373     .catch(err => {
9374       this.setState({ isLoading: false });
9375       this.setState({ data: [] });
9376     });
9377
9378 paginate = page => {
9379   this.setState({
9380     page,
9381   });
9382   this.setState({ isLoading: true });
9383   axios
9384     .post('api/registrar/getsections', {
9385       keyword: this.state.keyword,
9386       page,
9387       pageSize: this.state.pageSize,
9388     })
9389     .then(res => {
9390       this.setState({ isLoading: false });
9391       this.setState({
9392         numOfPages: res.data.numOfPages,
9393         data: res.data.sectionList,
9394       });
9395     })
9396     .catch(err => {
9397       this.setState({ isLoading: false });
9398     });
9399   };
9400
9401 componentWillMount() {
9402   this.setState({ isLoading: true });
9403   axios
9404     .post('api/registrar/getsections', { keyword: '', page: 1,
9405       pageSize: 10 })
9406     .then(res => {
9407       this.setState({ isLoading: false });
9408       this.setState({ numOfPages: res.data.numOfPages, data: res.data
9409         .sectionList });
9410     })
9411     .catch(err => {
9412       this.setState({ isLoading: false });
9413     });
9414
9415 componentWillReceiveProps() {

```

```

9415     this.setState({ isLoading: true });
9416     axios
9417       .post('api/registrar/getsections', {
9418         keyword: this.state.keyword,
9419         page: this.state.page,
9420         pageSize: 10,
9421       })
9422       .then(res => {
9423         this.setState({ isLoading: false });
9424         this.setState({ numOfPages: res.data.numOfPages, data: res.data
9425           .sectionList });
9426       })
9427       .catch(err => {
9428         this.setState({ isLoading: false });
9429         this.setState({ data: [] });
9430       });
9431   }
9432   render() {
9433     const displayGradeLevel = gradeLevel => {
9434       switch (gradeLevel) {
9435         case 'N':
9436           return 'Nursery';
9437         case 'K1':
9438           return 'Kinder 1';
9439         case 'K2':
9440           return 'Kinder 2';
9441         case 'G1':
9442           return 'Grade 1';
9443         case 'G2':
9444           return 'Grade 2';
9445         case 'G3':
9446           return 'Grade 3';
9447         case 'G4':
9448           return 'Grade 4';
9449         case 'G5':
9450           return 'Grade 5';
9451         case 'G6':
9452           return 'Grade 6';
9453         case 'G7':
9454           return 'Grade 7';
9455         case 'G8':
9456           return 'Grade 8';
9457         case 'G9':
9458           return 'Grade 9';
9459         case 'G10':
9460           return 'Grade 10';
9461         case 'G11':
9462           return 'Grade 11';
9463         case 'G12':
9464           return 'Grade 12';
9465         default:
9466           return '';
9467       }
9468     };
9469     const { errors } = this.props.app;
9470     const DisplayData = [];
9471     for (const [index, value] of this.state.data.entries()) {
9472       DisplayData.push(
9473         <Table.Row>
9474           <Table.Col>{value.name}</Table.Col>
9475           <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
9476           <Table.Col>
9477             <Link to={`/groupdeliberation/${value.key}`}>
9478               <Button icon="file" pill color="primary" value={value.key
9479                 }>
9480                 View Class Record
9481               </Button>
9482             </Link>
9483           </Table.Col>
9484         </Table.Row>,
9485       );
9486     }
9487     return (
9488       <Table.Body className="app-registrar-add-section my-3 my-md-5

```

```

        card">
9488    <Card.Body>
9489      <Card.Title>
9490        <Breadcrumb>
9491          <Breadcrumb.Item>Group Deliberation</Breadcrumb.Item>
9492          <Breadcrumb.Item>Sections List</Breadcrumb.Item>
9493        </Breadcrumb>
9494      </Card.Title>
9495      <Card.Title>Sections List</Card.Title>
9496      <Grid.Row>
9497        <Grid.Col sm={12} md={12} xs={12}>
9498          <Grid.Row>
9499            <Grid.Col sm={12} md={12} xs={12}>
9500              <Form.Group>
9501                <Form.Input
9502                  icon="search"
9503                  placeholder="Search for..."
9504                  position="append"
9505                  name="keyword"
9506                  value={this.state.keyword}
9507                  onChange={this.onChangeSearch}
9508                />
9509              </Form.Group>
9510            </Grid.Col>
9511          </Grid.Row>
9512        <Grid.Row>
9513          <Grid.Col sm={12} xs={12} md={12} lg={12}>
9514            <Spin spinning={this.state.isLoading}>
9515              <Table highlightRowOnHover={true} responsive={true}
9516                >
9517                  <Table.Header>
9518                    <Table.ColHeader>Section Name</Table.ColHeader>
9519                    <Table.ColHeader>Grade Level</Table.ColHeader>
9520                    <Table.ColHeader>Action</Table.ColHeader>
9521                  </Table.Header>
9522                  <Table.Body>
9523                    {DisplayData.length == 0 ? (
9524                      <Table.Row>
9525                        <Table.Col colSpan={3} alignContent="center"
9526                          >
9527                            No entries.
9528                          </Table.Col>
9529                        </Table.Row>
9530                      ) : (
9531                        DisplayData
9532                      )}
9533                    </Table.Body>
9534                  </Table>
9535                  <Pagination
9536                    size="large"
9537                    current={this.state.page}
9538                    pageSize={this.state.pageSize}
9539                    total={this.state.pageSize * this.state.
9540                      numOfPages}
9541                    onChange={this.paginate}
9542                  />
9543                </Spin>
9544              </Grid.Col>
9545            </Grid.Row>
9546          </Card.Body>
9547        </Table.Body>
9548      );
9549    }
9550
9551    /* istanbul ignore next */
9552    function mapStateToProps(state) {
9553      return {
9554        app: state.app,
9555      };
9556    }

```

```

9557
9558     /* istanbul ignore next */
9559     function mapDispatchToProps(dispatch) {
9560         return {
9561             actions: bindActionCreators({ ...actions }, dispatch),
9562         };
9563     }
9564
9565     export default connect(mapStateToProps, mapDispatchToProps)(
9566         RegistrarGroupDeliberation);
9567     import React, { Component } from 'react';
9568     import PropTypes from 'prop-types';
9569     import { bindActionCreators } from 'redux';
9570     import { connect } from 'react-redux';
9571     import * as actions from './redux/actions';
9572     import { Link } from 'react-router-dom';
9573     import { Card, Button, Grid, Table, Container, Form, Avatar, Alert } from
9574         'tabler-react';
9575     import axios from 'axios';
9576     import { Pagination, Spin, Tooltip, Modal } from 'antd';
9577     import AllStudentDeliberationGrades from './
9578         AllStudentDeliberationGrades';
9579     import placeholder from '../.../images/placeholder.jpg';
9580     import { getImageUrl } from '../.../utils';
9581
9582     export class RegistrarGroupDeliberationInfo extends Component {
9583         static propTypes = {
9584             app: PropTypes.object.isRequired,
9585             actions: PropTypes.object.isRequired,
9586         };
9587
9588         constructor(props) {
9589             super(props);
9590             this.state = {
9591                 isLoading: false,
9592                 isLoadingTable: true,
9593                 isLoadingTable2: true,
9594                 isLoadingTable3: true,
9595                 name: '',
9596                 email: '',
9597                 accountID: '',
9598                 imageUrl: '',
9599                 data: [],
9600                 data2: [],
9601                 data3: [],
9602                 columns: [],
9603                 schoolYearID: 0,
9604                 schoolYear: '',
9605                 page: 1,
9606                 keyword: '',
9607                 pageSize: 10,
9608                 page2: 1,
9609                 numOfPages: 1,
9610                 deadline: '',
9611                 selectedClassRecordID: -1,
9612                 selectedSubjectCode: '',
9613                 selectedSubjectName: '',
9614                 selectedSection: '',
9615                 selectedDeadline: '',
9616                 quarter: 'Q1',
9617                 sectionName: '',
9618             };
9619         }
9620
9621         handlePostGrade = () => {
9622             Modal.confirm({
9623                 title: 'Post Grades',
9624                 content:
9625                     'Do you want to post this class record? Once posted, it will be
9626                     accessed by the students enrolled in the subject.',
9627                 okText: 'Post',
9628                 cancelText: 'Cancel',
9629                 onCancel: () => {},

```

```

9628     onOk: () => {
9629       this.props.actions.postClassRecord(
9630         {
9631           classRecordID: this.state.selectedClassRecordID,
9632           quarter: this.state.quarter,
9633         },
9634         () => {},
9635       );
9636     },
9637   );
9638 };
9639
9640 resetClassRecordInfo = () => {
9641   this.setState({
9642     selectedClassRecordID: -1,
9643     selectedSubjectCode: '',
9644     selectedSubjectName: '',
9645     selectedSection: '',
9646     selectedDeadline: '',
9647   });
9648 };
9649
9650 componentDidMount() {
9651   axios.get('api/registrar/getsy').then(res => {
9652     this.setState({
9653       schoolYearID: res.data.schoolYearID,
9654       schoolYear: res.data.schoolYear,
9655       quarter: res.data.quarter,
9656     });
9657     axios
9658       .post('api/registrar/getsubmittedsubsectbysectionid', {
9659         sectionID: this.props.id,
9660         page: this.state.page,
9661         pageSize: this.state.pageSize,
9662         quarter: res.data.quarter,
9663       })
9664       .then(res2 => {
9665         this.setState({
9666           sectionName: res2.data.sectionName,
9667           data: res2.data.classRecordList,
9668           numOfPages: res2.data.numOfPages,
9669           isLoadingTable: false,
9670         });
9671       })
9672       .catch(err => {
9673         this.setState({ data: [], numOfPages: 1, isLoadingTable:
9674           false });
9675       });
9676     axios
9677       .post('api/registrar/getnotsubmittedsubsectbysectionid', {
9678         sectionID: this.props.id,
9679         page: this.state.page2,
9680         pageSize: this.state.pageSize2,
9681         quarter: res.data.quarter,
9682       })
9683       .then(res3 => {
9684         this.setState({
9685           data2: res3.data.classRecordList,
9686           numOfPages2: res3.data.numOfPages,
9687           isLoadingTable2: false,
9688         });
9689       })
9690       .catch(err => {
9691         this.setState({ data2: [], numOfPages2: 1, isLoadingTable2:
9692           false });
9693       });
9694     axios
9695       .post('api/registrar/condenseddeliberationgrade', {
9696         sectionID: this.props.id,
9697         quarter: res.data.quarter,
9698       })
9699       .then(res3 => {
9700         this.setState({
9701           isLoadingTable3: false,

```

```

9700         data3: res3.data.data,
9701         columns: res3.data.columns,
9702     });
9703   });
9704 }
9705
9706
9707 handleChangeQuarter = () => {
9708   this.setState({ isLoadingTable: true, isLoadingTable2: true,
9709     isLoadingTable3: true });
9710   axios
9711     .post('api/registrar/getsubmittedsubsectbysectionid', {
9712       sectionID: this.props.id,
9713       page: this.state.page,
9714       pageSize: this.state.pageSize,
9715       quarter: this.state.quarter,
9716     })
9717     .then(res2 => {
9718       this.setState({
9719         data: res2.data.classRecordList,
9720         numOfPages: res2.data.numOfPages,
9721         isLoadingTable: false,
9722       });
9723     })
9724     .catch(err => {
9725       this.setState({ data: [], numOfPages: 1, isLoadingTable: false
9726     });
9727   axios
9728     .post('api/registrar/condenseddeliberationgrade', {
9729       sectionID: this.props.id,
9730       quarter: this.state.quarter,
9731     })
9732     .then(res3 => {
9733       this.setState({
9734         isLoadingTable3: false,
9735         data3: res3.data.data,
9736         columns: res3.data.columns,
9737       });
9738   axios
9739     .post('api/registrar/getnotsubmittedsubsectbysectionid', {
9740       sectionID: this.props.id,
9741       page: this.state.page2,
9742       pageSize: this.state.pageSize2,
9743       quarter: this.state.quarter,
9744     })
9745     .then(res3 => {
9746       this.setState({
9747         data2: res3.data.classRecordList,
9748         numOfPages2: res3.data.numOfPages,
9749         isLoadingTable2: false,
9750       });
9751     })
9752     .catch(err => {
9753       this.setState({ data2: [], numOfPages2: 1, isLoadingTable2:
9754         false });
9755     });
9756
9757 componentWillReceiveProps() {
9758   axios
9759     .post('api/registrar/getsubmittedsubsectbysectionid', {
9760       sectionID: this.props.id,
9761       page: this.state.page,
9762       pageSize: this.state.pageSize,
9763       quarter: this.state.quarter,
9764     })
9765     .then(res2 => {
9766       this.setState({
9767         data: res2.data.classRecordList,
9768         numOfPages: res2.data.numOfPages,
9769         isLoadingTable: false,
9770       });

```

```

9771     })
9772     .catch(err => {
9773       this.setState({ data: [] , numOfPages: 1, isLoadingTable: false
9774     });
9775   axios
9776     .post('api/registrar/getnotsubmittedsubsectbysectionid' , {
9777       sectionID: this.props.id,
9778       page: this.state.page2,
9779       pageSize: this.state.pageSize2,
9780       quarter: this.state.quarter,
9781     })
9782     .then(res3 => {
9783       this.setState({
9784         data2: res3.data.classRecordList ,
9785         numOfPages2: res3.data.numOfPages ,
9786         isLoadingTable2: false ,
9787       });
9788     })
9789     .catch(err => {
9790       this.setState({ data2: [] , numOfPages2: 1, isLoadingTable2:
9791         false });
9792     });
9793
9794 paginate = page => {
9795   this.setState({ page, isLoadingTable: true }, async () => {
9796     axios
9797       .post('api/registrar/getsubmittedsubsectbysectionid' , {
9798         sectionID: this.props.id,
9799         page: page ,
9800         pageSize: this.state.pageSize ,
9801         quarter: this.state.quarter,
9802       })
9803       .then(res2 => {
9804         this.setState({
9805           data: res2.data.classRecordList ,
9806           numOfPages: res2.data.numOfPages ,
9807           isLoadingTable: false ,
9808         });
9809       })
9810       .catch(err => {
9811         this.setState({ data: [] , numOfPages: 1, isLoadingTable:
9812           false });
9813       });
9814     );
9815
9816 paginate2 = page => {
9817   this.setState({ page2: page, isLoadingTable2: true }, async () => {
9818     axios
9819       .post('api/registrar/getnotsubmittedsubsectbysectionid' , {
9820         sectionID: this.props.id,
9821         page: page ,
9822         pageSize: this.state.pageSize2,
9823         quarter: this.state.quarter,
9824       })
9825       .then(res2 => {
9826         this.setState({
9827           data2: res2.data.classRecordList ,
9828             numOfPages2: res2.data.numOfPages ,
9829             isLoadingTable2: false ,
9830           });
9831         })
9832         .catch(err => {
9833           this.setState({ data2: [] , numOfPages2: 1, isLoadingTable2:
9834             false });
9835         });
9836       );
9837
9838     render() {
9839       const DisplayColumns = [];

```

```

9840     DisplayColumns.push(<Table.ColHeader></Table.ColHeader>);
9841     DisplayColumns.push(<Table.ColHeader>Name</Table.ColHeader>);
9842     const DisplayData = [];
9843     const DisplayData2 = [];
9844     const DisplayData3 = [];
9845     for (const [index, value] of this.state.data3.entries()) {
9846       let tempCol = [];
9847       for (const [index2, value2] of value.grades.entries()) {
9848         tempCol.push(
9849           <Table.Col>
9850             {value2.grade != -1 && (
9851               <span
9852                 className={'status-icon bg-$parseFloat(value2.grade)
9853                   >= 75 ? 'green' : 'red'}{'}
9854                 />
9855               )} {value2.grade == -1 ? 'N/A' : value2.grade == 'N/A' ? 'Not
9856                 enrolled' : value2.grade}
9857             </Table.Col>,
9858           );
9859       DisplayData3.push(
9860         <Table.Row>
9861           <Table.Col className="w-1">
9862             <Avatar imageUrl={value.imageUrl == 'NA' ? placeholder :
9863               getImageUrl(value.imageUrl)} />
9864             <Table.Col>{value.name}</Table.Col>
9865             {tempCol}
9866             <Table.Col>
9867               <Tooltip placement="top" title="View all grades">
9868                 <AllStudentDeliberationGrades name={value.name}
9869                   studsectID={value.studsectID} />
9870               </Tooltip>
9871             </Table.Col>
9872           </Table.Row>,
9873         );
9874     for (const [index, value] of this.state.columns.entries()) {
9875       DisplayColumns.push(
9876         <Table.ColHeader>
9877           {value.status == 'F' && <span className="status-icon bg-green
9878             " />}
9879             {value.subjectName}
9880           </Table.ColHeader>,
9881         );
9882     DisplayColumns.push(<Table.ColHeader>Actions</Table.ColHeader>);
9883     for (const [index, value] of this.state.data.entries()) {
9884       const dateSubmitted = new Date(value.dateSubmitted);
9885       let status = '';
9886       if (value.deadline == 'NOT SET') {
9887         status = 'NOT SET';
9888       } else {
9889         let deadline = new Date(value.deadline);
9890         let relativeTime = new Date(Math.abs(deadline.getTime() -
9891           dateSubmitted.getTime()));
9892         let s = relativeTime / 1000;
9893         let m = s / 60;
9894         let h = m / 60;
9895         let d = h / 24;
9896         let overDueText = Math.floor(d) == 0 ? '' : Math.floor(d) + ' day/s';
9897         overDueText =
9898           overDueText +
9899             Math.floor(h % 24) +
9900               ' hr ' +
9901                 Math.floor(m % 60) +
9902                   ' min ' +
9903                     Math.floor(s % 60) +
9904                       ' secs';

```

```

9905     status = deadline.getTime() < dateSubmitted.getTime() ? '${  

9906         overDueText} late' : 'On Time';  

9907     DisplayData.push(  

9908         <Table.Row>  

9909             <Table.Col>{value.subjectCode}</Table.Col>  

9910             <Table.Col>{value.teacher}</Table.Col>  

9911             <Table.Col>  

9912                 <Tooltip placement="top" title="View Class Record">  

9913                     <Link  

9914                         to={`/individualdeliberation/${this.props.id}/  

9915                             managegrade/${value.classRecordID}/quarter/${this.  

9916                                 state.quarter}`}  

9917                         target="_blank"  

9918                     >  

9919                     <Button icon="eye" size="sm" pill color="primary"></  

9920                         Button>  

9921                     </Link>  

9922                 </Tooltip>  

9923                 <span style={{ marginLeft: '2px' }}>  

9924                     <Tooltip placement="top" title="Post Grade">  

9925                         <Button  

9926                             icon="check"  

9927                             size="sm"  

9928                             pill  

9929                             color="success"  

9930                             onClick={() =>  

9931                                 this.setState({ selectedClassRecordID: value.  

9932                                     classRecordID }, () =>  

9933                                     this.handlePostGrade(),  

9934                                         )  

9935                                 }  

9936                         ></Button>  

9937                     </Tooltip>  

9938                 </span>  

9939             </Table.Col>  

9940         </Table.Row>,  

9941     );  

9942 }
9943 for (const [index, value] of this.state.data2.entries()) {  

9944     const displayDate = new Date(value.deadline);  

9945     DisplayData2.push(  

9946         <Table.Row>  

9947             <Table.Col>{value.subjectCode}</Table.Col>  

9948             <Table.Col>{value.teacher}</Table.Col>  

9949             <Table.Col>{displayDate.toDateString()}</Table.Col>  

9950         </Table.Row>,  

9951     );  

9952 }
9953 return (
9954     <div className="app-registrar-individual-deliberation-info my-3  

9955         my-md-5">  

9956         <Container>  

9957             <Grid.Row>  

9958                 <Grid.Col sm={12} lg={6}>  

9959                     <Grid.Row>
9960                         <Container>
9961                             {this.state.isLoading ? (
9962                                 '',
9963                                 ) : (
9964                                     <Card statusColor="warning">
9965                                         <Card.Body>
9966                                             <Card.Title>
9967                                                 Class Record of {this.state.sectionName} S.Y.
9968                                                 {this.state.schoolYear}
9969                                         </Card.Title>
9970                                         <Grid.Row>
9971                                             <Grid.Col sm={12} md={12} xs={12}>
9972                                                 <Spin spinning={this.state.isLoadingTable3
9973                                                     }>
9974                                                 {' '}
9975                                         <Form.Group>
9976                                             <Form.Label>Select Quarter</Form.Label>
```

```

9970 <Form.Select
9971   onChange={e =>
9972     this.setState({ quarter: e.target.
9973       value }, () =>
9974       this.handleChangeQuarter(),
9975     )
9976   value={this.state.quarter}
9977 >
9978   <option value="Q1">Quarter 1</option>
9979   <option value="Q2">Quarter 2</option>
9980   <option value="Q3">Quarter 3</option>
9981   <option value="Q4">Quarter 4</option>
9982 </Form.Select>
9983   </Form.Group>
9984 </Spin>
9985 </Grid.Col>
9986 <Grid.Col sm={12} md={12} xs={12}>
9987   <Spin spinning={this.state.isLoadingTable}>
9988     <Table highlightRowOnHover={true}>
9989       responsive={true}>
9990         <Table.Header>
9991           <Table.ColHeader>Subject</Table.
9992             ColHeader>
9993           <Table.ColHeader>Teacher</Table.
9994             ColHeader>
9995           <Table.ColHeader>Actions</Table.
9996             ColHeader>
9997 </Table.Header>
9998   <Table.Body>
9999     {DisplayData.length == 0 ? (
1000       <Table.Row>
1001         <Table.Col colSpan={5}>
1002           alignContent="center">
1003             No submitted class record.
1004           </Table.Col>
1005         </Table.Row>
1006       ) : (
1007         DisplayData
1008       )}
1009     </Table.Body>
1010   </Table>
1011   <Pagination
1012     size="large"
1013     current={this.state.page}
1014     pageSize={this.state.pageSize}
1015     total={this.state.pageSize * this.state
1016       .numOfPages}
1017     onChange={this.paginate}
1018   />
1019   </Spin>
1020 </Grid.Col>
1021 </Grid.Row>
1022 <Grid.Col sm={12} xs={12} md={6}>
1023   <Grid.Row>
1024     <Container>
1025       {this.state.isLoading ? (
1026         '',
1027       ) : (
1028         <Card statusColor="danger">
1029           <Card.Body>
1030             <Card.Title>
1031               List of subjects which are{' '}
1032             <b>
1033               <i>not</i>
1034             </b>{' '}

```

```

10035           yet submitted for deliberation
10036     </Card.Title>
10037     <Grid.Row>
10038       <Grid.Col sm={12} md={12} xs={12}>
10039         <Spin spinning={this.state.isLoadingTable}>
10040           <Table highlightRowOnHover={true}
10041             responsive={true}>
10042             <Table.Header>
10043               <Table.ColHeader>Subject</Table.
10044                 ColHeader>
10045               <Table.ColHeader>Teacher</Table.
10046                 ColHeader>
10047               <Table.ColHeader>Deadline</Table.
10048                 ColHeader>
10049             </Table.Header>
10050           <Table.Body>
10051             {DisplayData2.length == 0 ? (
10052               <Table.Row>
10053                 <Table.Col colSpan={5}
10054                   alignContent="center">
10055                   No entries.
10056                 </Table.Col>
10057               </Table.Row>
10058             ) : (
10059               DisplayData2
10060             )}
10061           </Table.Body>
10062         </Table>
10063         <Pagination
10064           size="large"
10065           current={this.state.page2}
10066           pageSize={this.state.pageSize2}
10067           total={this.state.pageSize2 * this.
10068             state.numOfPages2}
10069           onChange={this.paginate2}
10070         />
10071       </Spin>
10072     </Grid.Col>
10073   </Grid.Row>
10074 </Grid.Col sm={12} xs={12}>
10075   <Grid.Row>
10076     <Container>
10077       <Card statusColor="info">
10078         <Card.Body>
10079           <Card.Title>
10080             Condensed Grades of {this.state.sectionName} -
10081             Quarter{' '}
10082             {this.state.quarter.charAt(1)}
10083           </Card.Title>
10084         <Card.Body>
10085           <Spin spinning={this.state.isLoadingTable3}>
10086             <Grid.Row>
10087               <Grid.Col sm={12} xs={12} md={12}>
10088                 {' '}
10089                 <Alert type="info" icon="info">
10090                   Note: Column names with status color{' '}
10091                   <span className="status-icon bg-green" />
10092                   <b>
10093                     <i>green</i>
10094                   </b>{' '}
10095                   are subjects which are now posted.{' '}
10096                   <b>
10097                     <i>View All Student Grades'</i>
10098                   </b>{' '}
10099                   button shows the deliberation grade (
10099                     grades under deliberation process{', '

```

```

10100                                     }
10101             <b>
10102                 <i>included</i>
10103             </b>
10104             ) of the student in current quarter.
10105         </Alert>
10106     </Grid.Col>
10107     <Table highlightRowOnHover={true} responsive
10108         ={true}>
10109         <Table.Header>{DisplayColumns}</Table.
10110             Header>
10111             <Table.Body>{DisplayData3}</Table.Body>
10112         </Table>
10113     </Grid.Row>
10114         <Spin>
10115             </Card.Body>
10116         </Card>
10117     </Container>
10118         </Grid.Row>
10119         </Grid.Col>
10120     </Grid.Row>
10121     </Container>
10122     </div>
10123   );
10124 }
10125 /* istanbul ignore next */
10126 function mapStateToProps(state) {
10127   return {
10128     app: state.app,
10129   };
10130 }
10131 /* istanbul ignore next */
10132 function mapDispatchToProps(dispatch) {
10133   return {
10134     actions: bindActionCreators({ ...actions }, dispatch),
10135   };
10136 }
10137
10138 export default connect(mapStateToProps, mapDispatchToProps)(
10139   RegistrarGroupDeliberationInfo);
10140 import React, { Component } from 'react';
10141 import PropTypes from 'prop-types';
10142 import { bindActionCreators } from 'redux';
10143 import { connect } from 'react-redux';
10144 import * as actions from './redux/actions';
10145 import NavBar from './NavBar';
10146 import RegistrarGroupDeliberationInfo from './
10147   RegistrarGroupDeliberationInfo';
10148 import { Container, Grid } from 'tabler-react';
10149 import { Result, Button } from 'antd';
10150 import axios from 'axios';
10151 import { Link } from 'react-router-dom';
10152
10153 export class RegistrarGroupDeliberationInfoView extends Component {
10154   static propTypes = {
10155     app: PropTypes.object.isRequired,
10156     actions: PropTypes.object.isRequired,
10157   };
10158   constructor(props) {
10159     super(props);
10160     this.state = {
10161       locked: false,
10162       isLoading: true,
10163     };
10164   }
10165   componentDidMount() {
10166     this.props.actions.setLoadingTrue();

```

```

10167     if (!this.props.app.auth.isAuthenticated) {
10168       this.props.history.push('/login');
10169     } else {
10170       if (this.props.app.auth.user.position != 2) {
10171         this.props.history.push('/page401');
10172       } else {
10173         axios
10174           .get('api/registrar/getsy')
10175           .then(res => {
10176             this.props.actions.setLoadingFalse();
10177             this.setState({ locked: false, isLoading: false });
10178           })
10179           .catch(err => {
10180             this.props.actions.setLoadingFalse();
10181             this.setState({ locked: true, isLoading: false });
10182           });
10183       }
10184     }
10185   }
10186
10187   render() {
10188     return (
10189       <div className="app-registrar-group-deliberation-info-view fh">
10190         {this.state.isLoading ? (
10191           ''
10192         ) : this.state.locked ? (
10193           <NavBar>
10194             <Container>
10195               <Grid.Row>
10196                 <Grid.Col xs={12} sm={12} md={12}>
10197                   <Result
10198                     status="403"
10199                     title="No Active School Year"
10200                     subTitle="Sorry, you are not authorized to access
                           this page now. Please contact your system
                           administrator for details."
10201                     extra={
10202                       <Link to="/">
10203                         <Button type="primary">Back Home</Button>
10204                       </Link>
10205                     }
10206                   />
10207                 </Grid.Col>
10208               </Grid.Row>
10209             </Container>
10210           </NavBar>
10211         ) : (
10212           <NavBar>
10213             <Container>
10214               <Grid.Row>
10215                 <Grid.Col xs={12} sm={12} md={12}>
10216                   <RegistrarGroupDeliberationInfo id={this.props.match.
                           params.id} />
10217                 </Grid.Col>
10218               </Grid.Row>
10219             </Container>
10220           </NavBar>
10221         )}
10222       </div>
10223     );
10224   }
10225 }
10226
10227 /* istanbul ignore next */
10228 function mapStateToProps(state) {
10229   return {
10230     app: state.app,
10231   };
10232 }
10233
10234 /* istanbul ignore next */
10235 function mapDispatchToProps(dispatch) {

```

```

10236     return {
10237       actions: bindActionCreators({ ...actions }, dispatch),
10238     };
10239   }
10240
10241   export default connect(mapStateToProps, mapDispatchToProps)(
10242     RegistrarGroupDeliberationInfoView);
10243   import React, { Component } from 'react';
10244   import PropTypes from 'prop-types';
10245   import { bindActionCreators } from 'redux';
10246   import { connect } from 'react-redux';
10247   import * as actions from './redux/actions';
10248   import NavBar from './NavBar';
10249   import RegistrarGroupDeliberation from './RegistrarGroupDeliberation';
10250   import { Container, Grid } from 'tabler-react';
10251   import { Result, Button } from 'antd';
10252   import axios from 'axios';
10253   import { Link } from 'react-router-dom';
10254
10255   export class RegistrarGroupDeliberationView extends Component {
10256     static propTypes = {
10257       app: PropTypes.object.isRequired,
10258       actions: PropTypes.object.isRequired,
10259     };
10260
10261     constructor(props) {
10262       super(props);
10263       this.state = {
10264         locked: false,
10265         isLoading: true,
10266       };
10267     }
10268
10269     componentDidMount() {
10270       this.props.actions.setLoadingTrue();
10271       if (!this.props.app.auth.isAuthenticated) {
10272         this.props.history.push('/login');
10273       } else {
10274         if (this.props.app.auth.user.position != 2) {
10275           this.props.history.push('/page401');
10276         } else {
10277           axios
10278             .get('api/registrar/getsy')
10279             .then(res => {
10280               this.props.actions.setLoadingFalse();
10281               this.setState({ locked: false, isLoading: false });
10282             })
10283             .catch(err => {
10284               this.props.actions.setLoadingFalse();
10285               this.setState({ locked: true, isLoading: false });
10286             });
10287         }
10288       }
10289     }
10290
10291     render() {
10292       return (
10293         <div className="app-registrar-group-deliberation-view fh">
10294           {this.state.isLoading ? (
10295             '',
10296             ) : this.state.locked ? (
10297               <NavBar>
10298                 <Container>
10299                   <Grid.Row>
10300                     <Grid.Col xs={12} sm={12} md={12}>
10301                       <Result
10302                         status="403"
10303                           title="No Active School Year"
10304                           subTitle="Sorry, you are not authorized to access
10305                             this page now. Please contact your system
                               administrator for details."
                           extra={
                             <Link to="/">
```

```

10306                               <Button type="primary">Back Home</Button>
10307                           </Link>
10308                       }
10309                   />
10310               </Grid.Col>
10311           </Grid.Row>
10312       </Container>
10313   </NavBar>
10314 ) : (
10315     <NavBar>
10316       <Container>
10317         <Grid.Row>
10318           <Grid.Col xs={12} sm={12} md={12}>
10319             <RegistrarGroupDeliberation />
10320           </Grid.Col>
10321         </Grid.Row>
10322       </Container>
10323     </NavBar>
10324   )}
10325   </div>
10326 );
10327 }
10328 }
10329 /* istanbul ignore next */
10330 function mapStateToProps(state) {
10331   return {
10332     app: state.app,
10333   };
10334 }
10335 }
10336 /* istanbul ignore next */
10337 function mapDispatchToProps(dispatch) {
10338   return {
10339     actions: bindActionCreators({ ...actions }, dispatch),
10340   };
10341 }
10342 }
10343 }
10344 export default connect(mapStateToProps, mapDispatchToProps)(
10345   RegistrarGroupDeliberationView);
10346 import React, { Component } from 'react';
10347 import PropTypes from 'prop-types';
10348 import { bindActionCreators } from 'redux';
10349 import { connect } from 'react-redux';
10350 import * as actions from './redux/actions';
10351 import { Card, Button, Grid, Avatar, Table, Form } from 'tabler-react';
10352 import { Pagination, Spin, Breadcrumb } from 'antd';
10353 import axios from 'axios';
10354 import { Link } from 'react-router-dom';
10355 import placeholder from '../../../../../images/placeholder.jpg';
10356 import { getImageUrl } from '../../../../../utils';
10357
10358 export class RegistrarIndividualDeliberation extends Component {
10359   static propTypes = {
10360     app: PropTypes.object.isRequired,
10361     actions: PropTypes.object.isRequired,
10362   };
10363   constructor(props) {
10364     super(props);
10365     this.state = {
10366       isLoading: true,
10367       isLoadingTable: true,
10368       keyword: '',
10369       page: 1,
10370       pageSize: 10,
10371       numOfPages: 1,
10372       data: [],
10373       schoolYearID: 0,
10374       schoolYear: '',
10375     };
10376     this.onChangeSearch = this.onChangeSearch.bind(this);
10377   }

```

```

10378     }
10379
10380     onChangeSearch(event) {
10381         this.setState({ [event.target.name]: event.target.value });
10382         this.setState({ isLoadingTable: true, page: 1 });
10383         axios
10384             .post('api/registrar/getallteachers', {
10385                 keyword: event.target.value,
10386                 page: 1,
10387                 pageSize: this.state.pageSize,
10388             })
10389             .then(res => {
10390                 this.setState({
10391                     numOfPages: res.data.numOfPages,
10392                     data: res.data.accountList,
10393                     isLoadingTable: false,
10394                 });
10395             })
10396             .catch(err => {
10397                 this.setState({ isLoadingTable: false, data: [] });
10398             });
10399     }
10400
10401     paginate = page => {
10402         this.setState({ page });
10403         this.setState({ isLoadingTable: true });
10404         axios
10405             .post('api/registrar/getallteachers', {
10406                 keyword: this.state.keyword,
10407                 page: page,
10408                 pageSize: this.state.pageSize,
10409             })
10410             .then(res => {
10411                 this.setState({
10412                     numOfPages: res.data.numOfPages,
10413                     data: res.data.accountList,
10414                     isLoadingTable: false,
10415                 });
10416             })
10417             .catch(err => {
10418                 this.setState({ isLoadingTable: false, data: [] });
10419             });
10420     };
10421
10422     componentDidMount() {
10423         this.setState({ isLoading: true, isLoadingTable: true });
10424         axios.get('api/registrar/getsy').then(res => {
10425             this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
10426                 res.data.schoolYear });
10427             this.setState({ isLoading: false });
10428             axios
10429                 .post('api/registrar/getallteachers', {
10430                     keyword: this.state.keyword,
10431                     page: this.state.page,
10432                     pageSize: this.state.pageSize,
10433                 })
10434                 .then(res2 => {
10435                     this.setState({
10436                         numOfPages: res2.data.numOfPages,
10437                         data: res2.data.accountList,
10438                         isLoadingTable: false,
10439                     });
10440                 })
10441                 .catch(err => {
10442                     this.setState({ isLoadingTable: false });
10443                 });
10444             });
10445     }
10446     render() {
10447         const DisplayData = [];
10448         for (const [index, value] of this.state.data.entries()) {
10449             DisplayData.push(

```

```

10450     <Table.Row>
10451         <Table.Col className="w-1">
10452             <Avatar imageUrl={value.imageUrl == 'NA' ? placeholder : getimageUrl(value.imageUrl)} />
10453         </Table.Col>
10454         <Table.Col>{value.name}</Table.Col>
10455         <Table.Col>{value.email}</Table.Col>
10456         <Table.Col alignContent="center">
10457             <Link to={`/individualdeliberation/${value.key}}>
10458                 <Button icon="file" color="primary">
10459                     View Class Record
10460                 </Button>
10461             </Link>
10462         </Table.Col>
10463     </Table.Row>,
10464 );
10465 }
10466 return (
10467     <Table.Body className="app-registrar-assign-subject-load card my-3 my-md-5">
10468         <Card.Body>
10469             {this.state.isLoading ? (
10470                 ,
10471             ) : (
10472                 <div>
10473                     <Card.Title>
10474                         <Breadcrumb>
10475                             <Breadcrumb.Item>Individual Deliberation</Breadcrumb.Item>
10476                             <Breadcrumb.Item>Teachers List</Breadcrumb.Item>
10477                         </Breadcrumb>
10478                     </Card.Title>
10479                     <Card.Title>Individual Deliberation for S.Y. {this.state.schoolYear}</Card.Title>
10480                 </div>
10481             )}
10482         <Grid.Row>
10483             <Grid.Col sm={12} md={12} xs={12}>
10484                 <Grid.Row>
10485                     <Grid.Col sm={12} md={12} xs={12}>
10486                         <Form.Group>
10487                             <Form.Input
10488                                 icon="search"
10489                                 placeholder="Search for..."
10490                                 position="append"
10491                                 name="keyword"
10492                                 value={this.state.keyword}
10493                                 onChange={this.onChangeSearch}
10494                             />
10495                         </Form.Group>
10496                     </Grid.Col>
10497                 </Grid.Row>
10498                 <Spin spinning={this.state.isLoadingTable}>
10499                     <Table highlightRowOnHover={true} responsive={true}>
10500                         <Table.Header>
10501                             <Table.ColHeader colSpan={2}>Teacher Name</Table.ColHeader>
10502                             <Table.ColHeader>Email</Table.ColHeader>
10503                             <Table.ColHeader alignContent="center">Action</Table.ColHeader>
10504                         </Table.Header>
10505                         <Table.Body>
10506                             {DisplayData.length == 0 ? (
10507                                 <Table.Row>
10508                                     <Table.Col colSpan={4} alignContent="center">
10509                                         No entries.
10510                                     </Table.Col>
10511                                 </Table.Row>
10512                             ) : (
10513                                 DisplayData
10514                             )}
10515                         </Table.Body>
10516                     </Table>

```

```

10517             <Pagination
10518                 size="large"
10519                     current={this.state.page}
10520                     pageSize={this.state.pageSize}
10521                     total={this.state.pageSize * this.state.numOfPages}
10522                     onChange={this.paginate}
10523                 />
10524             </Spin>
10525         </Grid.Col>
10526     </Grid.Row>
10527     </Card.Body>
10528   </Table.Body>
10529 );
10530 }
10531 }
10532 /* istanbul ignore next */
10533 function mapStateToProps(state) {
10534     return {
10535         app: state.app,
10536     };
10537 }
10538 }
10539 /* istanbul ignore next */
10540 function mapDispatchToProps(dispatch) {
10541     return {
10542         actions: bindActionCreators({ ...actions }, dispatch),
10543     };
10544 }
10545 }
10546
10547 export default connect(mapStateToProps, mapDispatchToProps)(
    RegistrarIndividualDeliberation);
10548 import React, { Component } from 'react';
10549 import PropTypes from 'prop-types';
10550 import { bindActionCreators } from 'redux';
10551 import { connect } from 'react-redux';
10552 import * as actions from './redux/actions';
10553 import { Link } from 'react-router-dom';
10554 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
    from 'tabler-react';
10555 import axios from 'axios';
10556 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
10557 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
    message } from 'antd';
10558 import cn from 'classnames';
10559 import placeholder from '../../images/placeholder.jpg';
10560 import bg from '../../images/BG.png';
10561 import { getImageUrl } from '../../utils';
10562 import ClassRecordInformation from './ClassRecordInformation';
10563 const { Option } = AutoComplete;
10564
10565 function ProfileImage({ avatarURL }) {
10566     return <img className="card-profile-img" alt="Profile" src={avatarURL
        } />;
10567 }
10568
10569 function Profile({ className, children, name, avatarURL = '',
    backgroundURL = '', bio }) {
10570     const classes = cn('card-profile', className);
10571     return (
10572         <Card className={classes}>
10573             <Card.Header backgroundURL={backgroundURL} />
10574             <Card.Body className="text-center">
10575                 <ProfileImage avatarURL={avatarURL} />
10576                 <Header.H3 className="mb-3">{name}</Header.H3>
10577                 <p className="mb-4">{bio || children}</p>
10578             </Card.Body>
10579         </Card>
10580     );
10581 }
10582
10583 export class RegistrarIndividualDeliberationInfo extends Component {

```

```

10584     static propTypes = {
10585       app: PropTypes.object.isRequired,
10586       actions: PropTypes.object.isRequired,
10587     };
10588
10589   constructor(props) {
10590     super(props);
10591     this.state = {
10592       isLoading: true,
10593       isLoadingTable: true,
10594       isLoadingTable2: true,
10595       name: '',
10596       email: '',
10597       accountID: '',
10598       imageUrl: '',
10599       data: [],
10600       data2: [],
10601       schoolYearID: 0,
10602       schoolYear: '',
10603       page: 1,
10604       page2: 1,
10605       keyword: '',
10606       pageSize: 10,
10607       pageSize2: 10,
10608       numOfPages: 1,
10609       numOfPages2: 1,
10610       deadline: '',
10611       selectedClassRecordID: -1,
10612       selectedSubjectCode: '',
10613       selectedSubjectName: '',
10614       selectedSection: '',
10615       selectedDeadline: '',
10616       quarter: '',
10617     };
10618   }
10619
10620   resetClassRecordInfo = () => {
10621     this.setState({
10622       selectedClassRecordID: -1,
10623       selectedSubjectCode: '',
10624       selectedSubjectName: '',
10625       selectedSection: '',
10626       selectedDeadline: '',
10627     });
10628   };
10629
10630   componentDidMount() {
10631     axios.get('api/registrar/getsy').then(res => {
10632       this.setState({
10633         schoolYearID: res.data.schoolYearID,
10634         schoolYear: res.data.schoolYear,
10635         quarter: res.data.quarter,
10636       });
10637       axios.post('api/registrar/userinfo', { accountID: this.props.id
10638         }).then(res2 => {
10639           this.setState({
10640             email: res2.data.email,
10641             imageUrl: res2.data.imageUrl,
10642             name: res2.data.name,
10643             isLoading: false,
10644           });
10645           axios
10646             .post('api/registrar/getsubmittedsubsect', {
10647               accountID: this.props.id,
10648               page: this.state.page,
10649               pageSize: this.state.pageSize,
10650               quarter: res.data.quarter,
10651             })
10652             .then(res3 => {
10653               this.setState({
10654                 data: res3.data.classRecordList,
10655                 numOfPages: res3.data.numOfPages,
10656                 deadline: res3.data.deadline,
10657                 isLoadingTable: false,
10658               });
10659             })

```

```

10659         .catch(err => {
10660             this.setState({ data: [], numOfPages: 1, isLoadingTable:
10661                 false });
10662         });
10663         axios
10664             .post('api/registrar/getnotsubmittedsubsect', {
10665                 accountID: this.props.id,
10666                 page: this.state.page2,
10667                 pageSize: this.state.pageSize2,
10668                 quarter: res.data.quarter,
10669             })
10670             .then(res3 => {
10671                 this.setState({
10672                     data2: res3.data.classRecordList,
10673                     deadline: res3.data.deadline,
10674                     numOfPages2: res3.data.numOfPages,
10675                     isLoadingTable2: false,
10676                 });
10677             })
10678             .catch(err => {
10679                 this.setState({ data: [], numOfPages2: 1, isLoadingTable2:
10680                     false });
10681             });
10682         }
10683     }
10684     componentWillReceiveProps() {
10685         axios
10686             .post('api/registrar/getsubmittedsubsect', {
10687                 accountID: this.props.id,
10688                 page: this.state.page,
10689                 pageSize: this.state.pageSize,
10690                 quarter: this.state.quarter,
10691             })
10692             .then(res3 => {
10693                 this.setState({
10694                     data: res3.data.classRecordList,
10695                     numOfPages: res3.data.numOfPages,
10696                     deadline: res3.data.deadline,
10697                     isLoadingTable: false,
10698                 });
10699             })
10700             .catch(err => {
10701                 this.setState({ data: [], numOfPages: 1, isLoadingTable: false
10702                     });
10703         });
10704         axios
10705             .post('api/registrar/getnotsubmittedsubsect', {
10706                 accountID: this.props.id,
10707                 page: this.state.page2,
10708                 pageSize: this.state.pageSize2,
10709                 quarter: this.state.quarter,
10710             })
10711             .then(res3 => {
10712                 this.setState({
10713                     data2: res3.data.classRecordList,
10714                     numOfPages2: res3.data.numOfPages,
10715                     isLoadingTable2: false,
10716                 });
10717             })
10718             .catch(err => {
10719                 this.setState({ data: [], numOfPages2: 1, isLoadingTable2:
10720                     false });
10721             });
10722         paginate = page => {
10723             this.setState({ page, isLoadingTable: true }, async () => {
10724                 axios
10725                     .post('api/registrar/getsubmittedsubsect', {
10726                         accountID: this.props.id,
10727                         page: page,
10728                         pageSize: this.state.pageSize,

```

```

10729         quarter: this.state.quarter,
10730     })
10731     .then(res3 => {
10732       this.setState({
10733         data: res3.data.classRecordList,
10734         numOfPages: res3.data.numOfPages,
10735         deadline: res3.data.deadline,
10736         isLoadingTable: false,
10737       });
10738     });
10739   );
10740 };
10741
10742 paginate2 = page => {
10743   this.setState({ page2: page, isLoadingTable2: true }, async () => {
10744     axios
10745       .post('api/registrar/getnotsubmittedsubsect', {
10746         accountID: this.props.id,
10747         page: page,
10748         pageSize: this.state.pageSize2,
10749         quarter: this.state.quarter,
10750       })
10751       .then(res3 => {
10752         this.setState({
10753           data2: res3.data.classRecordList,
10754           numOfPages2: res3.data.numOfPages,
10755           isLoadingTable2: false,
10756         });
10757       })
10758       .catch(err => {
10759         this.setState({ data: [], numOfPages2: 1, isLoadingTable2:
10760           false });
10761       });
10762     );
10763   };
10764 handleQuarterChange = () => {
10765   this.setState({ isLoadingTable: true, isLoadingTable2: true });
10766   axios
10767     .post('api/registrar/getsubmittedsubsect', {
10768       accountID: this.props.id,
10769       page: this.state.page,
10770       pageSize: this.state.pageSize,
10771       quarter: this.state.quarter,
10772     })
10773     .then(res3 => {
10774       this.setState({
10775         data: res3.data.classRecordList,
10776         numOfPages: res3.data.numOfPages,
10777         deadline: res3.data.deadline,
10778         isLoadingTable: false,
10779       });
10780     })
10781     .catch(err => {
10782       this.setState({ data: [], numOfPages: 1, isLoadingTable: false
10783       });
10784     });
10785   axios
10786     .post('api/registrar/getnotsubmittedsubsect', {
10787       accountID: this.props.id,
10788       page: this.state.page2,
10789       pageSize: this.state.pageSize2,
10790       quarter: this.state.quarter,
10791     })
10792     .then(res3 => {
10793       this.setState({
10794         data2: res3.data.classRecordList,
10795         numOfPages2: res3.data.numOfPages,
10796         isLoadingTable2: false,
10797       });
10798     })
10799     .catch(err => {
10800       this.setState({ data: [], numOfPages2: 1, isLoadingTable2:
10801         false });

```

```

10800         });
10801     };
10802
10803     render() {
10804       const DisplayData = [];
10805       const DisplayData2 = [];
10806       for (const [index, value] of this.state.data.entries()) {
10807         const dateSubmitted = new Date(value.dateSubmitted);
10808         let status = '';
10809         if (this.state.deadline == 'NOT SET') {
10810           status = 'NOT SET';
10811         } else {
10812           let deadline = new Date(this.state.deadline);
10813           let relativeTime = new Date(Math.abs(deadline.getTime() -
10814             dateSubmitted.getTime()));
10815           let s = relativeTime / 1000;
10816           let m = s / 60;
10817           let h = m / 60;
10818           let d = h / 24;
10819           let overDueText = Math.floor(d) == 0 ? '' : Math.floor(d) +
10820             'day/s ';
10821           overDueText =
10822             overDueText +
10823               Math.floor(h % 24) +
10824                 ' hr ' +
10825                   Math.floor(m % 60) +
10826                     ' min ' +
10827                       Math.floor(s % 60) +
10828                         ' secs';
10829
10830           status = deadline.getTime() < dateSubmitted.getTime() ? ${
10831             overDueText} late' : 'On Time';
10832       }
10833       DisplayData.push(
10834         <Table.Row>
10835           <Table.Col>{value.subjectCode}</Table.Col>
10836           <Table.Col>{value.subjectName}</Table.Col>
10837           <Table.Col>{value.section}</Table.Col>
10838           <Table.Col>
10839             <Button
10840               icon="eye"
10841               size="sm"
10842               outline
10843               pill
10844               color="primary"
10845               onClick={() =>
10846                 this.setState({
10847                   selectedClassRecordID: value.classRecordID,
10848                   selectedSubjectCode: value.subjectCode,
10849                   selectedSubjectName: value.subjectName,
10850                   selectedSection: value.section,
10851                   selectedDeadline: status,
10852                 })
10853               }
10854             >
10855               View
10856             </Button>
10857           </Table.Col>
10858         </Table.Row>,
10859       );
10860     }
10861     for (const [index, value] of this.state.data2.entries()) {
10862       let displayDate = 'NOT SET';
10863       if (this.state.deadline != 'NOT SET') {
10864         displayDate = new Date(this.state.deadline);
10865         displayDate = displayDate.toDateString();
10866       }
10867       DisplayData2.push(
10868         <Table.Row>
10869           <Table.Col>{value.subjectCode}</Table.Col>

```

```

10870         <Table.Col>{displayDate}</Table.Col>
10871     </Table.Row>,
10872   );
10873 }
10874
10875 return (
10876   <div className="app-registrar-individual-deliberation-info my-3
10877     my-md-5">
10878     <Container>
10879       <Grid.Row>
10880         <Grid.Col sm={12} lg={6}>
10881           <Spin spinning={this.state.isLoading}>
10882             <Container>
10883               <Grid.Row>
10884                 {this.state.isLoading ? (
10885                   <Profile name="" avatarURL={placeholder}
10886                     backgroundImage=""></Profile>
10887                 ) : (
10888                   <Profile
10889                     name={this.state.name}
10890                     avatarURL={
10891                       this.state.imageUrl === 'NA'
10892                         ? placeholder
10893                           : getImageUrl(this.state.imageUrl)
10894                     }
10895                     backgroundURL={bg}
10896                   >
10897                     <Container>
10898                       <Grid.Row>
10899                         <Grid.Col sm={12} xs={12} md={12}>
10900                           <Form.Group>
10901                             <Form.Label>Select Quarter</Form.Label>
10902                             <Form.Select
10903                               value={this.state.quarter}
10904                               onChange={e =>
10905                                 this.setState({ quarter: e.target.
10906                                   value }), () =>
10907                                     this.handleQuarterChange(),
10908                               )
10909                           }
10910                         >
10911                           <option value="Q1">Quarter 1</option>
10912                           <option value="Q2">Quarter 2</option>
10913                           <option value="Q3">Quarter 3</option>
10914                           <option value="Q4">Quarter 4</option>
10915                         </Form.Select>
10916                         </Form.Group>
10917                       </Grid.Col>
10918                     </Container>
10919                   </Profile>
10920                 )
10921               </Grid.Row>
10922             </Container>
10923           <Spin>
10924             <Grid.Col sm={12} xs={12} md={6}>
10925               {', '}
10926               <Grid.Row>
10927                 <Container>
10928                   {this.state.isLoading ? (
10929                     '',
10930                   ) : (
10931                     <Card statusColor="warning">
10932                       <Card.Body>
10933                         <Card.Title>
10934                           Class Record of {this.state.name} S.Y. {this.
10935                             state.schoolYear}
10936                         </Card.Title>
10937                         <Grid.Row>
10938                           <Grid.Col sm={12} md={12} xs={12}>
10939                             <Spin spinning={this.state.isLoadingTable}>
10940                               <Table highlightRowOnHover={true}>
```

```

10939             responsive={true}>
10940         <Table.Header>
10941             <Table.ColHeader>Code</Table.
10942                 ColHeader>
10943             <Table.ColHeader>Subject</Table.
10944                 ColHeader>
10945             <Table.ColHeader>Section</Table.
10946                 ColHeader>
10947             <Table.ColHeader>Actions</Table.
10948                 ColHeader>
10949         </Table.Header>
10950     <Table.Body>
10951         {DisplayData.length == 0 ? (
10952             <Table.Row>
10953                 <Table.Col colSpan={5}
10954                     alignContent="center">
10955                         No submitted class record.
10956                     </Table.Col>
10957                 </Table.Row>
10958             ) : (
10959                 DisplayData
10960             )}
10961         </Table.Body>
10962     </Table>
10963     <Pagination
10964         size="large"
10965         current={this.state.page}
10966         pageSize={this.state.pageSize}
10967         total={this.state.pageSize * this.state
10968             .numOfPages}
10969         onChange={this.paginate}
10970     />
10971         <Spin>
10972             </Grid.Col>
10973         </Grid.Row>
10974     </Card.Body>
10975     </Card>
10976     )
10977 </Container>
10978 <Container>
10979     {this.state.isLoading ? (
10980         '',
10981     ) : (
10982         <Card statusColor="danger">
10983             <Card.Body>
10984                 <Card.Title>
10985                     List of subjects which are{' '}
10986                     <b>
10987                         <i>not</i>
10988                     </b>{' '}
10989                     yet submitted for deliberation
10990                 </Card.Title>
10991             <Grid.Row>
10992                 <Grid.Col sm={12} md={12} xs={12}>
10993                     <Spin spinning={this.state.isLoadingTable}>
10994                         <Table highlightRowOnHover={true}
10995                             responsive={true}>
10996                             <Table.Header>
10997                                 <Table.ColHeader>Code</Table.
10998                                     ColHeader>
10999                                 <Table.ColHeader>Subject</Table.
11000                                     ColHeader>
11001                                 <Table.ColHeader>Section</Table.
11002                                     ColHeader>
11003                                 <Table.ColHeader>Deadline</Table.
11004                                     ColHeader>
11005                             </Table.Header>
11006                         <Table.Body>
11007                             {DisplayData2.length == 0 ? (
11008                                 <Table.Row>
11009                                     <Table.Col colSpan={5}
11010                                         alignContent="center">
11011                                             No entries.

```

```

10999                     </Table.Col>
11000                 </Table.Row>
11001             ) : (
11002                 DisplayData2
11003             )}
11004         </Table.Body>
11005     </Table>
11006     <Pagination
11007         size="large"
11008         current={this.state.page2}
11009         pageSize={this.state.pageSize2}
11010         total={this.state.pageSize2 * this.
11011             state.numOfPages2}
11012         onChange={this.paginate2}
11013     />
11014     <Spin>
11015         </Grid.Col>
11016         </Grid.Row>
11017             <Card.Body>
11018         </Card>
11019     )
11020 </Container>
11021 </Grid.Row>
11022 </Grid.Col>
11023 <Grid.Col sm={12} xs={12}>
11024     <Grid.Row>
11025         <Container>
11026             <ClassRecordInformation
11027                 classRecordID={this.state.selectedClassRecordID}
11028                 quarter={this.state.quarter}
11029                 subjectCode={this.state.selectedSubjectCode}
11030                 subjectName={this.state.selectedSubjectName}
11031                 section={this.state.selectedSection}
11032                 deadline={this.state.selectedDeadline}
11033                 resetClassRecordInfo={this.resetClassRecordInfo}
11034                 id={this.props.id}
11035                 status="deliberation"
11036             />
11037         </Container>
11038     </Grid.Row>
11039     </Grid.Col>
11040     </Container>
11041     </div>
11042 );
11043 }
11044 }
11045
11046 /* istanbul ignore next */
11047 function mapStateToProps(state) {
11048     return {
11049         app: state.app,
11050     };
11051 }
11052
11053 /* istanbul ignore next */
11054 function mapDispatchToProps(dispatch) {
11055     return {
11056         actions: bindActionCreators({ ...actions }, dispatch),
11057     };
11058 }
11059
11060 export default connect(mapStateToProps, mapDispatchToProps)(
11061     RegistrarIndividualDeliberationInfo);
11062 import React, { Component } from 'react';
11063 import PropTypes from 'prop-types';
11064 import { bindActionCreators } from 'redux';
11065 import { connect } from 'react-redux';
11066 import * as actions from './redux/actions';
11067 import NavBar from './NavBar';
11068 import RegistrarIndividualDeliberationInfo from './
11069     RegistrarIndividualDeliberationInfo';

```

```

11068 import { Container, Grid } from 'tabler-react';
11069 import { Result, Button } from 'antd';
11070 import axios from 'axios';
11071 import { Link } from 'react-router-dom';
11072
11073 export class RegistrarIndividualDeliberationInfoView extends Component
11074 {
11075     static propTypes = {
11076         app: PropTypes.object.isRequired,
11077         actions: PropTypes.object.isRequired,
11078     };
11079
11080     constructor(props) {
11081         super(props);
11082         this.state = {
11083             locked: false,
11084             isLoading: true,
11085         };
11086     }
11087
11088     componentDidMount() {
11089         this.props.actions.setLoadingTrue();
11090         if (!this.props.app.auth.isAuthenticated) {
11091             this.props.history.push('/login');
11092         } else {
11093             if (this.props.app.auth.user.position != 2) {
11094                 this.props.history.push('/page401');
11095             } else {
11096                 axios
11097                     .get('api/registrar/getsy')
11098                     .then(res => {
11099                         this.props.actions.setLoadingFalse();
11100                         this.setState({ locked: false, isLoading: false });
11101                     })
11102                     .catch(err => {
11103                         this.props.actions.setLoadingFalse();
11104                         this.setState({ locked: true, isLoading: false });
11105                     });
11106             }
11107         }
11108     }
11109
11110     render() {
11111         return (
11112             <div className="app-registrar-individual-deliberation-info-view
11113                 fh">
11114                 {this.state.isLoading ? (
11115                     ''
11116                 ) : this.state.locked ? (
11117                     <NavBar>
11118                         <Container>
11119                             <Grid.Row>
11120                                 <Grid.Col xs={12} sm={12} md={12}>
11121                                     <Result
11122                                         status="403"
11123                                         title="No Active School Year"
11124                                         subTitle="Sorry, you are not authorized to access
11125                                         this page now. Please contact your system
11126                                         administrator for details."
11127                                         extra={
11128                                             <Link to="/">
11129                                                 <Button type="primary">Back Home</Button>
11130                                             </Link>
11131                                         }
11132                                     />
11133                                 </Grid.Col>
11134                             </Grid.Row>
11135                         </Container>
11136                     ) : (
                     <NavBar>
                         <Container>
                             <Grid.Row>

```

```

11137         <Grid.Col xs={12} sm={12} md={12}>
11138             <RegistrarIndividualDeliberationInfo id={this.props.
11139                 match.params.id} />
11140         </Grid.Col>
11141     </Grid.Row>
11142   </Container>
11143 </NavBar>
11144 )
11145 </div>
11146 );
11147 }
11148
11149 /* istanbul ignore next */
11150 function mapStateToProps(state) {
11151     return {
11152         app: state.app,
11153     };
11154 }
11155
11156 /* istanbul ignore next */
11157 function mapDispatchToProps(dispatch) {
11158     return {
11159         actions: bindActionCreators({ ...actions }, dispatch),
11160     };
11161 }
11162
11163 export default connect(
11164     mapStateToProps,
11165     mapDispatchToProps,
11166 )(RegistrarIndividualDeliberationInfoView);
11167 import React, { Component } from 'react';
11168 import PropTypes from 'prop-types';
11169 import { bindActionCreators } from 'redux';
11170 import { connect } from 'react-redux';
11171 import * as actions from './redux/actions';
11172 import NavBar from './NavBar';
11173 import RegistrarIndividualDeliberation from './
11174     RegistrarIndividualDeliberation';
11175 import { Container, Grid } from 'tabler-react';
11176 import { Result, Button } from 'antd';
11177 import axios from 'axios';
11178 import { Link } from 'react-router-dom';
11179
11180 export class RegistrarIndividualDeliberationView extends Component {
11181     static propTypes = {
11182         app: PropTypes.object.isRequired,
11183         actions: PropTypes.object.isRequired,
11184     };
11185     constructor(props) {
11186         super(props);
11187         this.state = {
11188             locked: false,
11189             isLoading: true,
11190         };
11191     }
11192
11193     componentDidMount() {
11194         this.props.actions.setLoadingTrue();
11195         if (!this.props.app.auth.isAuthenticated) {
11196             this.props.history.push('/login');
11197         } else {
11198             if (this.props.app.auth.user.position != 2) {
11199                 this.props.history.push('/page401');
11200             } else {
11201                 axios
11202                     .get('api/registrar/getsy')
11203                     .then(res => {
11204                         this.props.actions.setLoadingFalse();
11205                         this.setState({ locked: false, isLoading: false });
11206                     })
11207             }
11208         }
11209     }
11210 }

```

```

11207         .catch(err => {
11208             this.props.actions.setLoadingFalse();
11209             this.setState({ locked: true, isLoading: false });
11210         });
11211     }
11212   );
11213 }
11214
11215 render() {
11216   return (
11217     <div className="app-registrar-individual-deliberation-view fh">
11218       {this.state.isLoading ? (
11219         '',
11220       ) : this.state.locked ? (
11221         <NavBar>
11222           <Container>
11223             <Grid.Row>
11224               <Grid.Col xs={12} sm={12} md={12}>
11225                 <Result
11226                   status="403"
11227                     title="No Active School Year"
11228                     subTitle="Sorry, you are not authorized to access
11229                       this page now. Please contact your system
11230                         administrator for details."
11231                         extra={
11232                           <Link to="/">
11233                             <Button type="primary">Back Home</Button>
11234                           </Link>
11235                         }
11236                       />
11237                     </Grid.Col>
11238                   </Grid.Row>
11239                 </Container>
11240               <NavBar>
11241                 <Container>
11242                   <Grid.Row>
11243                     <Grid.Col xs={12} sm={12} md={12}>
11244                       <RegistrarIndividualDeliberation />
11245                     </Grid.Col>
11246                   </Grid.Row>
11247                 </Container>
11248               </NavBar>
11249             )
11250           </div>
11251         );
11252       );
11253     }
11254   )
11255   /* istanbul ignore next */
11256   function mapStateToProps(state) {
11257     return {
11258       app: state.app,
11259     };
11260   }
11261
11262   /* istanbul ignore next */
11263   function mapDispatchToProps(dispatch) {
11264     return {
11265       actions: bindActionCreators({ ...actions }, dispatch),
11266     };
11267   }
11268
11269   export default connect(mapStateToProps, mapDispatchToProps)(
11270     RegistrarIndividualDeliberationView);
11271   import React, { Component } from 'react';
11272   import PropTypes from 'prop-types';
11273   import { bindActionCreators } from 'redux';
11274   import { connect } from 'react-redux';
11275   import * as actions from './redux/actions';
11276   import axios from 'axios';
11277   import { Pagination, Spin, Tooltip } from 'antd';

```

```

11277 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
11278   message } from 'antd';
11279 import cn from 'classnames';
11280 import {
11281   Card,
11282   Button,
11283   Grid,
11284   Avatar,
11285   Table,
11286   Form,
11287   Header,
11288   Container,
11289   Text,
11290   Alert,
11291 } from 'tabler-react';
11292 import { Link } from 'react-router-dom';
11293 import ViewEditLog from './ViewEditLog';
11294 const { Option } = AutoComplete;
11295
11296 export class RegistrarManageGrades extends Component {
11297   static propTypes = {
11298     app: PropTypes.object.isRequired,
11299     actions: PropTypes.object.isRequired,
11300   };
11301
11302   constructor(props) {
11303     super(props);
11304     this.state = {
11305       isLoading: true,
11306       sectionName: '',
11307       subjectCode: '',
11308       subjectName: '',
11309       schoolYear: '',
11310       classRecordID: 0,
11311       faData: [],
11312       wwData: [],
11313       ptData: [],
11314       qeData: [],
11315       showAddNewSubcomponent: false,
11316       selectedCompID: -1,
11317       selectedSubcompID: -1,
11318       keyword: '',
11319       showConfirmDelete: false,
11320       deleteText: '',
11321       editFA: false,
11322       editWW: false,
11323       editPT: false,
11324       faLoading: false,
11325       ptLoading: false,
11326       wwLoading: false,
11327       hasErrors: true,
11328       quarter: 'Q1',
11329       currentUpdated: '',
11330       subjectType: '',
11331       locked: false,
11332       status: '',
11333     };
11334   this.showAddNewSubcomponent = this.showAddNewSubcomponent.bind(this)
11335   );
11336   this.hideAddNewSubcomponent = this.hideAddNewSubcomponent.bind(this)
11337   );
11338   this.onChange = this.onChange.bind(this);
11339   this.addNewSubcomponent = this.addNewSubcomponent.bind(this);
11340   this.deleteSubcomponent = this.deleteSubcomponent.bind(this);
11341   this.onReset = this.onReset.bind(this);
11342   this.editSubcomponent = this.editSubcomponent.bind(this);
11343 }
11344
11345   editSubcomponent(payload) {
11346     this.props.actions.editSubcomponent(
11347       {
11348         payload,
11349         classRecordID: this.props.classRecordID,
11350         quarter:

```

```

11349         this.state.subjectType == 'NON_SHS' || this.state.subjectType
11350             == '1ST_SEM',
11351             ? this.props.quarter
11352             : this.props.quarter == 'Q3',
11353             ? 'Q1'
11354             : 'Q2',
11355         },
11356         'Registrar',
11357     );
11358 }
11359 deleteSubcomponent() {
11360     if (this.state.deleteText != 'DELETE') {
11361         message.error('You must type DELETE to confirm');
11362     } else {
11363         this.props.actions.deleteSubcomponent(
11364             {
11365                 subcompID: this.state.selectedSubcompID,
11366                 classRecordID: this.props.classRecordID,
11367                 quarter:
11368                     this.state.subjectType == 'NON_SHS' || this.state.
11369                         subjectType == '1ST_SEM',
11370                         ? this.props.quarter
11371                         : this.props.quarter == 'Q3',
11372                         ? 'Q1'
11373                         : 'Q2',
11374                     },
11375                     'Registrar',
11376                 );
11377         this.setState({ showConfirmDelete: false });
11378     }
11379 }
11380 addNewSubcomponent() {
11381     this.props.actions.addNewSubcomponent(
11382         {
11383             classRecordID: this.props.classRecordID,
11384             name: this.state.keyword,
11385             componentID: this.state.selectedCompID,
11386             quarter:
11387                 this.state.subjectType == 'NON_SHS' || this.state.subjectType
11388                     == '1ST_SEM',
11389                     ? this.props.quarter
11390                     : this.props.quarter == 'Q3',
11391                     ? 'Q1'
11392                     : 'Q2',
11393                 },
11394                 'Registrar',
11395             );
11396         this.setState({ showAddNewSubcomponent: false });
11397     }
11398     onChange(event) {
11399         this.setState({ [event.target.name]: event.target.value });
11400     }
11401     showAddNewSubcomponent(key) {
11402         this.setState({ showAddNewSubcomponent: true, selectedCompID: key,
11403             keyword: '' });
11404     }
11405     hideAddNewSubcomponent() {
11406         this.setState({ showAddNewSubcomponent: false, selectedCompID: -1
11407             });
11408     }
11409     onReset(key) {
11410         switch (key) {
11411             case 'FA':
11412                 {
11413                     this.setState({ faLoading: true });
11414                     axios
11415                         .post('api/registrar/getcomponents', {
11416                             classRecordID: this.props.classRecordID,
11417                             quarter:

```

```

11418     this.state.subjectType == 'NON_SHS' || this.state.
11419         subjectType == '1ST_SEM'
11420         ? this.props.quarter
11421         : this.props.quarter == 'Q3'
11422         ? 'Q1'
11423         : 'Q2',
11424     })
11425     .then(res => {
11426       this.setState({
11427         sectionName: res.data.sectionName,
11428         subjectName: res.data.subjectName,
11429         schoolYear: res.data.schoolYear,
11430         subjectCode: res.data.subjectCode,
11431         classRecordID: res.data.classRecordID,
11432         faData: res.data.FA,
11433         wwData: res.data.WW,
11434         ptData: res.data.PT,
11435         qeData: res.data.QE,
11436         faLoading: false,
11437       });
11438     });
11439   case 'WW': {
11440     this.setState({ wwLoading: true });
11441     axios
11442       .post('api/registrar/getcomponents', {
11443         classRecordID: this.props.classRecordID,
11444         quarter:
11445           this.state.subjectType == 'NON_SHS' || this.state.
11446             subjectType == '1ST_SEM'
11447             ? this.props.quarter
11448             : this.props.quarter == 'Q3'
11449             ? 'Q1'
11450             : 'Q2',
11451       })
11452       .then(res => {
11453         this.setState({
11454           sectionName: res.data.sectionName,
11455           subjectName: res.data.subjectName,
11456           schoolYear: res.data.schoolYear,
11457           subjectCode: res.data.subjectCode,
11458           classRecordID: res.data.classRecordID,
11459           faData: res.data.FA,
11460           wwData: res.data.WW,
11461           ptData: res.data.PT,
11462           qeData: res.data.QE,
11463           wwLoading: false,
11464         });
11465       });
11466   case 'PT': {
11467     this.setState({ ptLoading: true });
11468     axios
11469       .post('api/registrar/getcomponents', {
11470         classRecordID: this.props.classRecordID,
11471         quarter:
11472           this.state.subjectType == 'NON_SHS' || this.state.
11473             subjectType == '1ST_SEM'
11474             ? this.props.quarter
11475             : this.props.quarter == 'Q3'
11476             ? 'Q1'
11477             : 'Q2',
11478       })
11479       .then(res => {
11480         this.setState({
11481           sectionName: res.data.sectionName,
11482           subjectName: res.data.subjectName,
11483           schoolYear: res.data.schoolYear,
11484           subjectCode: res.data.subjectCode,
11485           classRecordID: res.data.classRecordID,
11486           faData: res.data.FA,
11487           wwData: res.data.WW,
11488           ptData: res.data.PT,
11489           qeData: res.data.QE,
11490           ptLoading: false,
11491         });
11492       });
11493     }

```

```

11491         });
11492     }
11493     default: {
11494     }
11495   }
11496 }
11497
11498 componentDidMount() {
11499   this.setState({ isLoading: true });
11500   axios
11501     .post('api/registrar/getsubjecttype', { classRecordID: this.props
11502       .classRecordID })
11503     .then(res => {
11504       this.setState({ subjectType: res.data.subjectType }, () => {
11505         axios
11506           .post('api/registrar/getcomponents', {
11507             classRecordID: this.props.classRecordID,
11508             quarter:
11509               this.state.subjectType == 'NON_SHS' || this.state.
11510                 subjectType == '1ST_SEM'
11511               ? this.props.quarter
11512               : this.props.quarter == 'Q3'
11513               ? 'Q1'
11514               : 'Q2',
11515             })
11516             .then(res2 => {
11517               this.setState({
11518                 sectionName: res2.data.sectionName,
11519                 subjectName: res2.data.subjectName,
11520                 schoolYear: res2.data.schoolYear,
11521                 subjectCode: res2.data.subjectCode,
11522                 classRecordID: res2.data.classRecordID,
11523                 faData: res2.data.FA,
11524                 wwData: res2.data.WW,
11525                 ptData: res2.data.PT,
11526                 qeData: res2.data.QE,
11527                 quarter: this.props.quarter,
11528                 isLoading: false,
11529               });
11530             });
11531           });
11532         });
11533       componentWillReceiveProps(nextProps) {
11534         if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
11535           .length == 0) {
11536           this.setState({ isLoading: true });
11537           axios
11538             .post('api/registrar/getsubjecttype', { classRecordID: this.
11539               props.classRecordID })
11540             .then(res => {
11541               this.setState({ subjectType: res.data.subjectType }, () => {
11542                 axios
11543                   .post('api/registrar/getcomponents', {
11544                     classRecordID: this.props.classRecordID,
11545                     quarter:
11546                       this.state.subjectType == 'NON_SHS' || this.state.
11547                         subjectType == '1ST_SEM'
11548                         ? this.props.quarter
11549                         : this.props.quarter == 'Q3'
11550                         ? 'Q1'
11551                         : 'Q2',
11552                     })
11553                     .then(res2 => {
11554                       this.setState({
11555                         sectionName: res2.data.sectionName,
11556                         subjectName: res2.data.subjectName,
11557                         schoolYear: res2.data.schoolYear,
11558                         subjectCode: res2.data.subjectCode,
11559                         classRecordID: res2.data.classRecordID,
11560                           faData: res2.data.FA,
                           wwData: res2.data.WW,
                           ptData: res2.data.PT,
                           qeData: res2.data.QE,

```

```

11561             quarter: this.props.quarter,
11562             isLoading: false,
11563         });
11564     });
11565   });
11566 }
11567 }
11568 }
11569
11570 render() {
11571   const { locked } = this.props;
11572   let displayFaData = [];
11573   let displayWwData = [];
11574   let displayPtData = [];
11575   let displayQeData = [];
11576   if (!this.state.isLoading) {
11577     if (this.state.editFA) {
11578       for (const [index, value] of this.state.faData.subcomponents.
11579         entries()) {
11580         displayFaData.push(
11581           <Table.Row>
11582             <Table.Col>
11583               <Form.Input
11584                 value={value.name}
11585                 onChange={e => {
11586                   let temparr = this.state.faData;
11587                   temparr.subcomponents[index].name = e.target.value;
11588                   this.setState({ faData: temparr });
11589                 }}
11590               />
11591             </Table.Col>
11592             <Table.Col>
11593               <Form.Input
11594                 value={value.compWeight}
11595                 onChange={e => {
11596                   const reg = /^-?[0-9]*(\.[0-9]*)?$/;
11597                   if (
11598                     isNaN(e.target.value) && reg.test(e.target.
11599                       value)) ||
11600                     e.target.value === '' ||
11601                     e.target.value === '-',
11602                   ) {
11603                     let temparr = this.state.faData;
11604                     temparr.subcomponents[index].compWeight = e.
11605                       target.value;
11606                     this.setState({ faData: temparr });
11607                   }
11608                 />
11609               </Table.Col>
11610             <Table.Col alignContent="center"></Table.Col>
11611           </Table.Row>,
11612         );
11613     } else {
11614       for (const [index, value] of this.state.faData.subcomponents.
11615         entries()) {
11616         displayFaData.push(
11617           <Table.Row>
11618             <Table.Col>{value.name}</Table.Col>
11619             <Table.Col>{value.compWeight}%</Table.Col>
11620             <Table.Col alignContent="center">
11621               {!locked && (
11622                 <span>
11623                   <Button
11624                     icon="trash"
11625                     color="danger"
11626                     onClick={() =>
11627                       this.setState({
11628                         selectedSubcompID: value.subcompID,
11629                         showConfirmDelete: true,
11630                         deleteText: '',

```

```

11629                               })
11630                         }
11631                         pill
11632                         size="sm"
11633                         ></Button>
11634                     </span>
11635                 )
11636             </Table.Col>
11637         </Table.Row>,
11638     );
11639 }
11640 }
11641 if (this.state.editWW) {
11642     for (const [index, value] of this.state.wwData.subcomponents.
11643         entries()) {
11644         displayWwData.push(
11645             <Table.Row>
11646                 <Table.Col>
11647                     <Form.Input
11648                         value={value.name}
11649                         onChange={e => {
11650                             let temparr = this.state.wwData;
11651                             temparr.subcomponents[index].name = e.target.value;
11652                             this.setState({ wwData: temparr });
11653                         }}
11654                     />
11655                 <Table.Col>
11656                     <Form.Input
11657                         value={value.compWeight}
11658                         onChange={e => {
11659                             const reg = /^-?[0-9]*(\.[0-9]*)?$/;
11660                             if (
11661                                 isNaN(e.target.value) && reg.test(e.target.
11662                                     value)) ||
11663                                     e.target.value === '' ||
11664                                     e.target.value === '--'
11665                             ) {
11666                             let temparr = this.state.wwData;
11667                             temparr.subcomponents[index].compWeight = e.
11668                                 target.value;
11669                             this.setState({ wwData: temparr });
11670                         }
11671                     />
11672                     <Table.Col alignContent="center"></Table.Col>
11673                 </Table.Row>,
11674             );
11675         }
11676     } else {
11677         for (const [index, value] of this.state.wwData.subcomponents.
11678             entries()) {
11679             displayWwData.push(
11680                 <Table.Row>
11681                     <Table.Col>{value.name}</Table.Col>
11682                     <Table.Col>{value.compWeight}%</Table.Col>
11683                     <Table.Col alignContent="center">
11684                         {!locked && (
11685                             <span>
11686                             <Button
11687                                 icon="trash"
11688                                 color="danger"
11689                                 onClick={() =>
11690                                     this.setState({
11691                                         selectedSubcompID: value.subcompID,
11692                                         showConfirmDelete: true,
11693                                         deleteText: '',
11694                                     })
11695                                 }
11696                                 pill
11697                                 size="sm"
11698                             ></Button>

```

```

11698             </span>
11699         )}
11700     </Table.Col>
11701   </Table.Row>,
11702 );
11703 }
11704 }
11705 if (this.state.editPT) {
11706   for (const [index, value] of this.state.ptData.subcomponents.
11707     entries()) {
11708     displayPtData.push(
11709       <Table.Row>
11710         <Table.Col>
11711           <Form.Input
11712             value={value.name}
11713             onChange={e => {
11714               let temparr = this.state.ptData;
11715               temparr.subcomponents[index].name = e.target.value;
11716               this.setState({ ptData: temparr });
11717             }}
11718           />
11719         <Table.Col>
11720           <Table.Col>
11721             <Form.Input
11722               value={value.compWeight}
11723               onChange={e => {
11724                 const reg = /^-?[0-9]*(\.[0-9]*)?$/;
11725                 if (
11726                   (!isNaN(e.target.value) && reg.test(e.target.
11727                     value)) ||
11728                     e.target.value === '' || 
11729                     e.target.value === '-'
11730                 ) {
11731                   let temparr = this.state.ptData;
11732                   temparr.subcomponents[index].compWeight = e.
11733                     target.value;
11734                   this.setState({ ptData: temparr });
11735                 }
11736               />
11737             <Table.Col alignContent="center"></Table.Col>
11738           </Table.Row>,
11739         );
11740   }
11741 } else {
11742   for (const [index, value] of this.state.ptData.subcomponents.
11743     entries()) {
11744     displayPtData.push(
11745       <Table.Row>
11746         <Table.Col>{value.name}</Table.Col>
11747         <Table.Col>{value.compWeight}%</Table.Col>
11748         <Table.Col alignContent="center">
11749           {!locked && (
11750             <span>
11751               <Button
11752                 icon="trash"
11753                 color="danger"
11754                 onClick={() =>
11755                   this.setState({
11756                     selectedSubcompID: value.subcompID,
11757                     showConfirmDelete: true,
11758                     deleteText: '',
11759                   })
11760                 }
11761               pill
11762               size="sm"
11763             ></Button>
11764           </span>
11765         )
11766       </Table.Col>
11767     </Table.Row>,

```

```

11766         );
11767     }
11768 }
11769
11770     for (const [index, value] of this.state.qeData.subcomponents.
11771       entries()) {
11772       displayQeData.push(
11773         <Table.Row>
11774           <Table.Col>{value.name}</Table.Col>
11775           <Table.Col>{value.compWeight}%</Table.Col>
11776         </Table.Row>,
11777       );
11778     }
11779     if (displayFaData.length == 0) {
11780       displayFaData.push(
11781         <Table.Row>
11782           <Table.Col colSpan={2} alignContent="center">
11783             No subcomponents.
11784           </Table.Col>
11785         </Table.Row>,
11786       );
11787     }
11788     if (displayWwData.length == 0) {
11789       displayWwData.push(
11790         <Table.Row>
11791           <Table.Col colSpan={2} alignContent="center">
11792             No subcomponents.
11793           </Table.Col>
11794         </Table.Row>,
11795       );
11796     }
11797     if (displayPtData.length == 0) {
11798       displayPtData.push(
11799         <Table.Row>
11800           <Table.Col colSpan={2} alignContent="center">
11801             No subcomponents.
11802           </Table.Col>
11803         </Table.Row>,
11804       );
11805     }
11806     if (displayQeData.length == 0) {
11807       displayQeData.push(
11808         <Table.Row>
11809           <Table.Col colSpan={2} alignContent="center">
11810             No subcomponents.
11811           </Table.Col>
11812         </Table.Row>,
11813       );
11814     }
11815   return (
11816     <div className="app-teacher-manage-grades my-3 my-md-5">
11817       <Container>
11818         <Modal
11819           title="Delete Subcomponent"
11820           visible={this.state.showConfirmDelete}
11821           onOk={this.deleteSubcomponent}
11822           onCancel={() => this.setState({ showConfirmDelete: false })}
11823           okText="Delete"
11824           confirmLoading={this.props.app.showLoading}
11825           cancelText="Cancel"
11826         >
11827           <Container>
11828             <Grid.Row>
11829               <Grid.Col sm={12} xs={12} md={12} lg={12}>
11830                 <Header.H5>
11831                   Are you sure you want to delete this subcomponent?
11832                   Type 'DELETE' to confirm.
11833                 </Header.H5>
11834               </Grid.Col>
11835             </Grid.Row>
             <Grid.Row>

```

```

11836 <Grid.Col sm={12} xs={12} md={12} lg={12}>
11837   <Form.Group>
11838     <Form.Input
11839       autoComplete="off"
11840       value={this.state.deleteText}
11841       name="deleteText"
11842       onChange={this.onChange}
11843       placeholder="Type DELETE"
11844     />
11845   </Form.Group>
11846 </Grid.Col>
11847 </Grid.Row>
11848 </Container>
11849 </Modal>
11850 <Modal
11851   title="Add New Subcomponent"
11852   visible={this.state.showAddNewSubcomponent}
11853   onOk={this.addNewSubcomponent}
11854   onCancel={this.hideAddNewSubcomponent}
11855   okText="Add"
11856   confirmLoading={this.props.app.showLoading}
11857   cancelText="Close"
11858 >
11859   <Grid.Row>
11860     <Grid.Col sm={12} md={12} xs={12}>
11861       <Form.Group>
11862         <Form.Label>Subcomponent Name</Form.Label>
11863         <Form.Input
11864           placeholder="Subcomponent name"
11865           name="keyword"
11866           value={this.state.keyword}
11867           onChange={this.onChange}
11868         />
11869       </Form.Group>
11870     </Grid.Col>
11871   </Grid.Row>
11872 </Modal>
11873 </Grid.Row>
11874 <Card>
11875   <Card.Body>
11876     {this.state.isLoading ? (
11877       <Spin spinning={true}></Spin>
11878     ) : (
11879       <div>
11880         <Card.Title>
11881           <Breadcrumb>
11882             <Breadcrumb.Item>Subjects</Breadcrumb.Item>
11883             <Breadcrumb.Item>View Subject Load</Breadcrumb.
11884               Item>
11885             <Breadcrumb.Item>Manage Grades</Breadcrumb.Item
11886               >
11887             <Breadcrumb.Item>
11888               {this.state.sectionName} - {this.state.
11889                 subjectName}
11890               </Breadcrumb.Item>
11891             </Breadcrumb>
11892           </Card.Title>
11893           <Header.H3>
11894             {this.state.subjectCode} - {this.state.
11895               subjectName}
11896           </Header.H3>
11897           <Card.Title>
11898             <Text.Small>
11899               {this.state.sectionName}{', '}
11900               {this.state.subjectType == 'NON_SHS'
11901                 ? this.props.quarter
11902                   : this.props.quarter == 'Q1' || this.props.
11903                     quarter == 'Q2'
11904                     ? 'FIRST SEMESTER'
11905                     : 'SECOND SEMESTER'}{', '}
11906           S.Y. {this.state.schoolYear}

```

```

11905         </Text.Small>
11906     <Grid.Row>
11907         <Grid.Col xs={12} sm={12} md={3}></Grid.Col>
11908         <Grid.Col xs={12} sm={12} md={3}></Grid.Col>
11909         <Grid.Col xs={12} sm={12} md={6}>
11910             <Button.List align="right">
11911                 <Link
11912                     to={
11913                         !locked
11914                         ? '/individualdeliberation/${this.
11915                             props.id}/managegrade/${
11916                             this.props.classRecordID
11917                         }/quarter/${{
11918                             this.state.subjectType == ,
11919                             NON_SHS' ||
11920                             this.state.subjectType == '1
11921                             ST_SEM'
11922                             ? this.props.quarter
11923                             : this.props.quarter == 'Q3'
11924                             ? 'Q1',
11925                             : 'Q2'
11926                         }/summaryreport'
11927                         : '/viewstudentrecord/classrecord/${{
11928                             this.props.classRecordID}/q/${{
11929                             this.state.subjectType == ,
11930                             NON_SHS' ||
11931                             this.state.subjectType == '1
11932                             ST_SEM'
11933                             ? this.props.quarter
11934                             : this.props.quarter == 'Q3'
11935                             ? 'Q1',
11936                             : 'Q2'
11937                         }/summaryreport'
11938                     }
11939                 >
11940                     <Button color="success" icon="file">
11941                         View{' '}
11942                         {this.state.subjectType == 'NON_SHS'
11943                             ? this.props.quarter
11944                             : this.props.quarter == 'Q1' || this.
11945                             props.quarter == 'Q3'
11946                             ? 'MIDTERM'
11947                             : 'FINAL'S'{ ' '}
11948                         Summary Report
11949                     </Button>
11950                 </Link>
11951             </Button.List>
11952         </Grid.Col>
11953     </Grid.Row>
11954     </Card.Title>
11955     </div>
11956     )>
11957 </Card.Body>
11958 {this.state.isLoading ? (
11959     <Spin spinning={this.state.isLoading}></Spin>
11960 ) : (
11961     <Card.Body>
11962         <Grid.Row>
11963             {Object.keys(this.props.app.errors).length != 0 &&
11964                 (
11965                     <Alert type="danger" icon="alert-triangle">
11966                         There is an error updating the subcomponents.
11967                         Make sure that no field is
11968                         empty and weights are between 0 to 100.
11969                         Subcomponent names must be between 2
11970                         to 255 characters.
11971                     </Alert>
11972                 )
11973             </Grid.Row>
11974         <Spin spinning={this.state.isLoading}>
11975             <Grid.Row>
11976                 <Grid.Col sm={12} xs={12} md={6}>
11977                     <Spin spinning={this.state.faLoading}>

```

```

11969 <Card statusColor={this.state.editFA ? 'yellow' : 'blue'}>
11970   <Card.Header>
11971     <Card.Title>
11972       Formative Assessment - {this.state.
11973         faData.weight}%
11974   </Card.Title>
11975   <Card.Options>
11976     {this.state.editFA ? (
11977       ,
11978     ) : (
11979       <Button.List>
11980         <Button color="info" size="sm">
11981           <Link
11982             style={{ color: 'white' }}
11983             to={
11984               !locked
11985                 ? '/individualdeliberation/
11986                   ${
11987                     this.props.id
11988                   }/managegrade/${this.
11989                     props.classRecordID}/
11990                     quarter/${{
11991                       this.state.subjectType
11992                         == 'NON_SHS' ||
11993                         this.state.subjectType
11994                           == '1ST_SEM'
11995                             ? this.props.quarter
11996                               : this.props.quarter
11997                                 == 'Q3'
11998                                   ? 'Q1'
11999                                     : 'Q2'
12000                               }/comp/${this.state.
12001                                 faData.componentID}'+
12002                                   : '/viewstudentrecord/
12003                                     classrecord/${{
12004                                       this.props.
12005                                         classRecordID
12006                                         }/q/${{
12007                                           this.state.subjectType
12008                                             == 'NON_SHS' ||
12009                                               this.state.subjectType
12010                                                 == '1ST_SEM'
12011                                                 ? this.props.quarter
12012                                                   : this.props.quarter
12013                                                     == 'Q3'
12014                                                       ? 'Q1'
12015                                                         : 'Q2'
12016                                         }/comp/${this.state.
12017                                           faData.componentID}+
12018                                         }
12019                                         >
12020                                           View
12021                                         </Link>
12022                                         </Button>
12023                                         {!locked && (
12024                                           <Button
12025                                             color="primary"
12026                                               size="sm"
12027                                               onClick={() => {
12028                                                 this.setState({
12029                                                   editFA: true,
12030                                                 });
12031                                               })}
12032                                         }
12033                                         >
12034                                           Edit
12035                                         </Button>
12036                                         )}
12037                                         </Button.List>
12038                                         )
12039                                         </Card.Options>
12040                                         </Card.Header>
12041                                         <Card.Body>
12042                                           <Table responsive={true}>

```

```
12028 <Table.Header>
12029   <Table.ColHeader>Subcomponent Name</
12030     Table.ColHeader>
12031   <Table.ColHeader>Weight </Table.
12032     ColHeader>
12033   <Table.ColHeader>Actions </Table.
12034     ColHeader>
12035   </Table.Header>
12036   <Table.Body>{displayFaData}</Table.Body
12037   >
12038 </Table>
12039 </Card.Body>
12040 {!locked && (
12041   <Card.Footer>
12042     {this.state.editFA ? (
12043       <Button.List align="right">
12044         <Button
12045           color="info"
12046           size="sm"
12047           onClick={() => {
12048             this.setState({ editFA: false
12049           });
12050             this.onReset('FA');
12051           }}}
12052         >
12053           Back
12054         </Button>
12055         <Button
12056           color="primary"
12057           size="sm"
12058           onClick={() => {
12059             this.setState({ currentUpdated:
12060               'FA' }, () => {
12061                 this.editSubcomponent(this.
12062                   state.faData.
12063                     subcomponents);
12064               });
12065             })
12066           >
12067             Save
12068           </Button>
12069         </Button.List>
12070     ) : (
12071       <Button.List align="right">
12072         <Button
12073           color="primary"
12074           size="sm"
12075           icon="plus"
12076           onClick={() =>
12077             this.showAddNewSubcomponent(
12078               this.state.faData.
12079                 componentID)
12080           });
12081       >
12082         Add New Subcomponent
12083       </Button>
12084     </Button.List>
12085   ) }
12086   </Card.Footer>
12087 </Card>
12088 </Spin>
12089 </Grid.Col>
12090 <Grid.Col sm={12} xs={12} md={6}>
12091   <Spin spinning={this.state.wwLoading}>
12092     <Card statusColor={this.state.editWW ? 'yellow' : 'blue'}>
12093       <Card.Header>
12094         <Card.Title>Written Works - {this.state.
12095           wwData.weight}%</Card.Title>
12096       <Card.Options>
12097         {this.state.editWW ? (
12098           ,
12099           ,
12100           ,
12101           ,
12102           ,
12103           ,
12104           ,
12105           ,
12106           ,
12107           ,
12108           ,
12109           ,
12110           ,
12111           ,
12112           ,
12113           ,
12114           ,
12115           ,
12116           ,
12117           ,
12118           ,
12119           ,
12120           ,
12121           ,
12122           ,
12123           ,
12124           ,
12125           ,
12126           ,
12127           ,
12128           ,
12129           ,
12130           ,
12131           ,
12132           ,
12133           ,
12134           ,
12135           ,
12136           ,
12137           ,
12138           ,
12139           ,
12140           ,
12141           ,
12142           ,
12143           ,
12144           ,
12145           ,
12146           ,
12147           ,
12148           ,
12149           ,
12150           ,
12151           ,
12152           ,
12153           ,
12154           ,
12155           ,
12156           ,
12157           ,
12158           ,
12159           ,
12160           ,
12161           ,
12162           ,
12163           ,
12164           ,
12165           ,
12166           ,
12167           ,
12168           ,
12169           ,
12170           ,
12171           ,
12172           ,
12173           ,
12174           ,
12175           ,
12176           ,
12177           ,
12178           ,
12179           ,
12180           ,
12181           ,
12182           ,
12183           ,
12184           ,
12185           ,
12186           ,
12187           ,
12188           ,
12189           ,
12190           ,
12191           ,
12192           ,
12193           ,
12194           ,
12195           ,
12196           ,
12197           ,
12198           ,
12199           ,
12200           ,
12201           ,
12202           ,
12203           ,
12204           ,
12205           ,
12206           ,
12207           ,
12208           ,
12209           ,
12210           ,
12211           ,
12212           ,
12213           ,
12214           ,
12215           ,
12216           ,
12217           ,
12218           ,
12219           ,
12220           ,
12221           ,
12222           ,
12223           ,
12224           ,
12225           ,
12226           ,
12227           ,
12228           ,
12229           ,
12230           ,
12231           ,
12232           ,
12233           ,
12234           ,
12235           ,
12236           ,
12237           ,
12238           ,
12239           ,
12240           ,
12241           ,
12242           ,
12243           ,
12244           ,
12245           ,
12246           ,
12247           ,
12248           ,
12249           ,
12250           ,
12251           ,
12252           ,
12253           ,
12254           ,
12255           ,
12256           ,
12257           ,
12258           ,
12259           ,
12260           ,
12261           ,
12262           ,
12263           ,
12264           ,
12265           ,
12266           ,
12267           ,
12268           ,
12269           ,
12270           ,
12271           ,
12272           ,
12273           ,
12274           ,
12275           ,
12276           ,
12277           ,
12278           ,
12279           ,
12280           ,
12281           ,
12282           ,
12283           ,
12284           ,
12285           ,
12286           ,
12287           ,
12288           ,
12289           ,
12290           ,
12291           ,
12292           ,
12293           ,
12294           ,
12295           ,
12296           ,
12297           ,
12298           ,
12299           ,
12300           ,
12301           ,
12302           ,
12303           ,
12304           ,
12305           ,
12306           ,
12307           ,
12308           ,
12309           ,
12310           ,
12311           ,
12312           ,
12313           ,
12314           ,
12315           ,
12316           ,
12317           ,
12318           ,
12319           ,
12320           ,
12321           ,
12322           ,
12323           ,
12324           ,
12325           ,
12326           ,
12327           ,
12328           ,
12329           ,
12330           ,
12331           ,
12332           ,
12333           ,
12334           ,
12335           ,
12336           ,
12337           ,
12338           ,
12339           ,
12340           ,
12341           ,
12342           ,
12343           ,
12344           ,
12345           ,
12346           ,
12347           ,
12348           ,
12349           ,
12350           ,
12351           ,
12352           ,
12353           ,
12354           ,
12355           ,
12356           ,
12357           ,
12358           ,
12359           ,
12360           ,
12361           ,
12362           ,
12363           ,
12364           ,
12365           ,
12366           ,
12367           ,
12368           ,
12369           ,
12370           ,
12371           ,
12372           ,
12373           ,
12374           ,
12375           ,
12376           ,
12377           ,
12378           ,
12379           ,
12380           ,
12381           ,
12382           ,
12383           ,
12384           ,
12385           ,
12386           ,
12387           ,
12388           ,
12389           ,
12390           ,
12391           ,
12392           ,
12393           ,
12394           ,
12395           ,
12396           ,
12397           ,
12398           ,
12399           ,
12400           ,
12401           ,
12402           ,
12403           ,
12404           ,
12405           ,
12406           ,
12407           ,
12408           ,
12409           ,
12410           ,
12411           ,
12412           ,
12413           ,
12414           ,
12415           ,
12416           ,
12417           ,
12418           ,
12419           ,
12420           ,
12421           ,
12422           ,
12423           ,
12424           ,
12425           ,
12426           ,
12427           ,
12428           ,
12429           ,
12430           ,
12431           ,
12432           ,
12433           ,
12434           ,
12435           ,
12436           ,
12437           ,
12438           ,
12439           ,
12440           ,
12441           ,
12442           ,
12443           ,
12444           ,
12445           ,
12446           ,
12447           ,
12448           ,
12449           ,
12450           ,
12451           ,
12452           ,
12453           ,
12454           ,
12455           ,
12456           ,
12457           ,
12458           ,
12459           ,
12460           ,
12461           ,
12462           ,
12463           ,
12464           ,
12465           ,
12466           ,
12467           ,
12468           ,
12469           ,
12470           ,
12471           ,
12472           ,
12473           ,
12474           ,
12475           ,
12476           ,
12477           ,
12478           ,
12479           ,
12480           ,
12481           ,
12482           ,
12483           ,
12484           ,
12485           ,
12486           ,
12487           ,
12488           ,
12489           ,
12490           ,
12491           ,
12492           ,
12493           ,
12494           ,
12495           ,
12496           ,
12497           ,
12498           ,
12499           ,
12500           ,
12501           ,
12502           ,
12503           ,
12504           ,
12505           ,
12506           ,
12507           ,
12508           ,
12509           ,
12510           ,
12511           ,
12512           ,
12513           ,
12514           ,
12515           ,
12516           ,
12517           ,
12518           ,
12519           ,
12520           ,
12521           ,
12522           ,
12523           ,
12524           ,
12525           ,
12526           ,
12527           ,
12528           ,
12529           ,
12530           ,
12531           ,
12532           ,
12533           ,
12534           ,
12535           ,
12536           ,
12537           ,
12538           ,
12539           ,
12540           ,
12541           ,
12542           ,
12543           ,
12544           ,
12545           ,
12546           ,
12547           ,
12548           ,
12549           ,
12550           ,
12551           ,
12552           ,
12553           ,
12554           ,
12555           ,
12556           ,
12557           ,
12558           ,
12559           ,
12560           ,
12561           ,
12562           ,
12563           ,
12564           ,
12565           ,
12566           ,
12567           ,
12568           ,
12569           ,
12570           ,
12571           ,
12572           ,
12573           ,
12574           ,
12575           ,
12576           ,
12577           ,
12578           ,
12579           ,
12580           ,
12581           ,
12582           ,
12583           ,
12584           ,
12585           ,
12586           ,
12587           ,
12588           ,
12589           ,
12590           ,
12591           ,
12592           ,
12593           ,
12594           ,
12595           ,
12596           ,
12597           ,
12598           ,
12599           ,
12600           ,
12601           ,
12602           ,
12603           ,
12604           ,
12605           ,
12606           ,
12607           ,
12608           ,
12609           ,
12610           ,
12611           ,
12612           ,
12613           ,
12614           ,
12615           ,
12616           ,
12617           ,
12618           ,
12619           ,
12620           ,
12621           ,
12622           ,
12623           ,
12624           ,
12625           ,
12626           ,
12627           ,
12628           ,
12629           ,
12630           ,
12631           ,
12632           ,
12633           ,
12634           ,
12635           ,
12636           ,
12637           ,
12638           ,
12639           ,
12640           ,
12641           ,
12642           ,
12643           ,
12644           ,
12645           ,
12646           ,
12647           ,
12648           ,
12649           ,
12650           ,
12651           ,
12652           ,
12653           ,
12654           ,
12655           ,
12656           ,
12657           ,
12658           ,
12659           ,
12660           ,
12661           ,
12662           ,
12663           ,
12664           ,
12665           ,
12666           ,
12667           ,
12668           ,
12669           ,
12670           ,
12671           ,
12672           ,
12673           ,
12674           ,
12675           ,
12676           ,
12677           ,
12678           ,
12679           ,
12680           ,
12681           ,
12682           ,
12683           ,
12684           ,
12685           ,
12686           ,
12687           ,
12688           ,
12689           ,
12690           ,
12691           ,
12692           ,
12693           ,
12694           ,
12695           ,
12696           ,
12697           ,
12698           ,
12699           ,
12700           ,
12701           ,
12702           ,
12703           ,
12704           ,
12705           ,
12706           ,
12707           ,
12708           ,
12709           ,
12710           ,
12711           ,
12712           ,
12713           ,
12714           ,
12715           ,
12716           ,
12717           ,
12718           ,
12719           ,
12720           ,
12721           ,
12722           ,
12723           ,
12724           ,
12725           ,
12726           ,
12727           ,
12728           ,
12729           ,
12730           ,
12731           ,
12732           ,
12733           ,
12734           ,
12735           ,
12736           ,
12737           ,
12738           ,
12739           ,
12740           ,
12741           ,
12742           ,
12743           ,
12744           ,
12745           ,
12746           ,
12747           ,
12748           ,
12749           ,
12750           ,
12751           ,
12752           ,
12753           ,
12754           ,
12755           ,
12756           ,
12757           ,
12758           ,
12759           ,
12760           ,
12761           ,
12762           ,
12763           ,
12764           ,
12765           ,
12766           ,
12767           ,
12768           ,
12769           ,
12770           ,
12771           ,
12772           ,
12773           ,
12774           ,
12775           ,
12776           ,
12777           ,
12778           ,
12779           ,
12780           ,
12781           ,
12782           ,
12783           ,
12784           ,
12785           ,
12786           ,
12787           ,
12788           ,
12789           ,
12790           ,
12791           ,
12792           ,
12793           ,
12794           ,
12795           ,
12796           ,
12797           ,
12798           ,
12799           ,
12800           ,
12801           ,
12802           ,
12803           ,
12804           ,
12805           ,
12806           ,
12807           ,
12808           ,
12809           ,
12810           ,
12811           ,
12812           ,
12813           ,
12814           ,
12815           ,
12816           ,
12817           ,
12818           ,
12819           ,
12820           ,
12821           ,
12822           ,
12823           ,
12824           ,
12825           ,
12826           ,
12827           ,
12828           ,
12829           ,
12830           ,
12831           ,
12832           ,
12833           ,
12834           ,
12835           ,
12836           ,
12837           ,
12838           ,
12839           ,
12840           ,
12841           ,
12842           ,
12843           ,
12844           ,
12845           ,
12846           ,
12847           ,
12848           ,
12849           ,
12850           ,
12851           ,
12852           ,
12853           ,
12854           ,
12855           ,
12856           ,
12857           ,
12858           ,
12859           ,
12860           ,
12861           ,
12862           ,
12863           ,
12864           ,
12865           ,
12866           ,
12867           ,
12868           ,
12869           ,
12870           ,
12871           ,
12872           ,
12873           ,
12874           ,
12875           ,
12876           ,
12877           ,
12878           ,
12879           ,
12880           ,
12881           ,
12882           ,
12883           ,
12884           ,
12885           ,
12886           ,
12887           ,
12888           ,
12889           ,
12890           ,
12891           ,
12892           ,
12893           ,
12894           ,
12895           ,
12896           ,
12897           ,
12898           ,
12899           ,
12900           ,
12901           ,
12902           ,
12903           ,
12904           ,
12905           ,
12906           ,
12907           ,
12908           ,
12909           ,
12910           ,
12911           ,
12912           ,
12913           ,
12914           ,
12915           ,
12916           ,
12917           ,
12918           ,
12919           ,
12920           ,
12921           ,
12922           ,
12923           ,
12924           ,
12925           ,
12926           ,
12927           ,
12928           ,
12929           ,
12930           ,
12931           ,
12932           ,
12933           ,
12934           ,
12935           ,
12936           ,
12937           ,
12938           ,
12939           ,
12940           ,
12941           ,
12942           ,
12943           ,
12944           ,
12945           ,
12946           ,
12947           ,
12948           ,
12949           ,
12950           ,
12951           ,
12952           ,
12953           ,
12954           ,
12955           ,
12956           ,
12957           ,
12958           ,
12959           ,
12960           ,
12961           ,
12962           ,
12963           ,
12964           ,
12965           ,
12966           ,
12967           ,
12968           ,
12969           ,
12970           ,
12971           ,
12972           ,
12973           ,
12974           ,
12975           ,
12976           ,
12977           ,
12978           ,
12979           ,
12980           ,
12981           ,
12982           ,
12983           ,
12984           ,
12985           ,
12986           ,
12987           ,
12988           ,
12989           ,
12990           ,
12991           ,
12992           ,
12993           ,
12994           ,
12995           ,
12996           ,
12997           ,
12998           ,
12999           ,
129999
```

```

12089 ) : (
12090   <Button.List>
12091     <Button color="info" size="sm">
12092       <Link
12093         style={{ color: 'white' }}
12094         to={_
12095           !locked
12096             ? '/individualdeliberation/
12097               ${
12098                 this.props.id
12099               }/managegrade/${this.
12100                 props.classRecordID}/
12101                 quarter/${_
12102                   this.state.subjectType
12103                     == 'NON_SHS' ||
12104                     this.state.subjectType
12105                     == '1ST_SEM'
12106                     ? this.props.quarter
12107                     : this.props.quarter
12108                     == 'Q3'
12109                     ? 'Q1'
12110                     : 'Q2'
12111               }/comp/${this.state.
12112                 wwData.componentID}‘
12113               : ‘/viewstudentrecord/
12114                 classrecord/${_
12115                   this.props.
12116                     classRecordID
12117               }/q/${_
12118                 this.state.subjectType
12119                   == 'NON_SHS' ||
12120                     this.state.subjectType
12121                     == '1ST_SEM'
12122                     ? this.props.quarter
12123                     : this.props.quarter
12124                     == 'Q3'
12125                     ? 'Q1'
12126                     : 'Q2'
12127               }/comp/${this.state.
12128                 wwData.componentID}‘
12129             }
12130           >
12131             View
12132           </Link>
12133         </Button>
12134       {!locked && (
12135         <Button
12136           color="primary"
12137             size="sm"
12138             onClick={() => {
12139               this.setState({
12140                 editWW: true,
12141               });
12142             }}
12143           >
12144             Edit
12145           </Button>
12146         )
12147       }
12148     </Card.Options>
12149   </Card.Header>
12150   <Card.Body>
12151     <Table responsive={true}>
12152       <Table.Header>
12153         <Table.ColHeader>Subcomponent Name</
12154           Table.ColHeader>
12155         <Table.ColHeader>Weight</Table.
12156           ColHeader>
12157         <Table.ColHeader>Actions</Table.
12158           ColHeader>
12159       </Table.Header>
12160       <Table.Body>{displayWwData}</Table.Body
12161     >

```

```

12146         </Table>
12147     </Card.Body>
12148     {!locked && (
12149         <Card.Footer>
12150             {this.state.editWW ? (
12151                 <Button.List align="right">
12152                     <Button
12153                         color="info"
12154                         size="sm"
12155                         onClick={() => {
12156                             this.setState({ editWW: false
12157                             });
12158                             this.onReset('WW');
12159                         }}
12160                     >
12161                         Back
12162                     </Button>
12163                     <Button
12164                         color="primary"
12165                         size="sm"
12166                         onClick={() => {
12167                             this.setState({ currentUpdated:
12168                                 'WW' }, () => {
12169                                 this.editSubcomponent(this.
12170                                     state.wwData.
12171                                         subcomponents);
12172                         });
12173                     >
12174                         Save
12175                     </Button>
12176                 ) : (
12177                     <Button.List align="right">
12178                         <Button
12179                             color="primary"
12180                             size="sm"
12181                             icon="plus"
12182                             onClick={() =>
12183                                 this.showAddNewSubcomponent(
12184                                     this.state.wwData.
12185                                         componentID)
12186                         }
12187                     >
12188                         Add New Subcomponent
12189                     </Button.List>
12190                 ) }
12191             </Card.Footer>
12192         </Card>
12193     </Spin>
12194     </Grid.Col>
12195     <Grid.Col sm={12} xs={12} md={6}>
12196         <Spin spinning={this.state.ptLoading}>
12197             <Card statusColor={this.state.editPT ? 'yellow' : 'blue'}>
12198                 <Card.Header>
12199                     <Card.Title>
12200                         Performance Task - {this.state.ptData.
12201                             weight}%
12202                     </Card.Title>
12203                     <Card.Options>
12204                         {this.state.editPT ? (
12205                             '',
12206                         ) : (
12207                             <Button.List>
12208                                 <Button color="info" size="sm">
12209                                     <Link
12210                                         style={{ color: 'white' }}
12211                                         to={
12212                                             !locked
12213                                                 ? '/individualdeliberation/

```

```

12211           ${
12212             this.props.id
12213           }/managegrade/${this.
12214             props.classRecordID}/
12215               quarter/${{
12216                 this.state.subjectType
12217                   == 'NON_SHS' || 
12218                     this.state.subjectType
12219                       == '1ST_SEM'
12220                         ? this.props.quarter
12221                           : this.props.quarter
12222                             == 'Q3',
12223                               ? 'Q1'
12224                                 : 'Q2'
12225           }/comp/${this.state.
12226             ptData.componentID}`
12227           : '/viewstudentrecord/
12228             classrecord/${{
12229               this.props.
12230                 classRecordID
12231           }/q/${{
12232             this.state.subjectType
12233               == 'NON_SHS' || 
12234                 this.state.subjectType
12235                   == '1ST_SEM'
12236                     ? this.props.quarter
12237                       : this.props.quarter
12238                         == 'Q3',
12239                           ? 'Q1'
12240                             : 'Q2'
12241           }/comp/${this.state.
12242             ptData.componentID}`
12243           }
12244         >
12245           View
12246         </Link>
12247       </Button>
12248     {!locked && (
12249       <Button
12250         color="primary"
12251           size="sm"
12252             onClick={() => {
12253               this.setState({
12254                 editPT: true,
12255               });
12256             }}
12257           >
12258             Edit
12259           </Button>
12260         )
12261       </Card.Options>
12262     </Card.Header>
12263     <Card.Body>
12264       <Table responsive={true}>
12265         <Table.Header>
12266           <Table.ColHeader>Subcomponent Name</
12267             Table.ColHeader>
12268           <Table.ColHeader>Weight</Table.
12269             ColHeader>
12270           <Table.ColHeader>Actions</Table.
12271             ColHeader>
12272         </Table.Header>
12273         <Table.Body>{displayPtData}</Table.Body
12274         >
12275       </Table>
12276     </Card.Body>
12277   {!locked && (
12278     <Card.Footer>
12279       {this.state.editPT ? (
12280         <Button.List align="right">
12281           <Button
12282             color="info"

```

```

12268           size="sm"
12269           onClick={() => {
12270             this.setState({ editPT: false
12271               });
12272             this.onReset('PT');
12273           }
12274         >
12275           Back
12276         </Button>
12277         <Button
12278           color="primary"
12279           size="sm"
12280           onClick={() => {
12281             this.setState({ currentUpdated:
12282               'PT' }, () => {
12283               this.editSubcomponent(this.
12284                 state.ptData.
12285                 subcomponents);
12286             });
12287           });
12288         >
12289           Save
12290         </Button>
12291       </Button.List>
12292     ) : (
12293       <Button.List align="right">
12294         <Button
12295           color="primary"
12296           size="sm"
12297           icon="plus"
12298           onClick={() =>
12299             this.showAddNewSubcomponent(
12300               this.state.ptData.
12301                 componentID)
12302             });
12303           </Card.Footer>
12304         );
12305       </Spin>
12306     </Grid.Col>
12307     <Grid.Col sm={12} xs={12} md={6}>
12308       <Card statusColor="blue">
12309         <Card.Header>
12310           <Card.Title>
12311             Quarterly Assessment - {this.state.qeData
12312               .weight}%
12313           </Card.Title>
12314           <Card.Options>
12315             <Button.List>
12316               <Button color="info" size="sm">
12317                 <Link
12318                   style={{ color: 'white' }}
12319                   to={
12320                     !locked
12321                     ? '/individualdeliberation/${
12322                       this.props.id}/managegrade/
12323                     ${
12324                       this.props.classRecordID
12325                     }/quarter/${
12326                       this.state.subjectType == '
12327                         NON_SHS' ||
12328                           this.state.subjectType == '
12329                             1ST_SEM'
12330                           ? this.props.quarter
12331                             : this.props.quarter == '
12332                               Q3'
12333                               ? 'Q1'
12334                                 : 'Q2'
12335                               }/comp/${this.state.qeData.

```

```

12330                               componentID}`
12331 : '/viewstudentrecord/
12332   classrecord/${
12333     this.props.classRecordID
12334   }/q/${{
12335     this.state.subjectType == 'NON_SHS' ||
12336     this.state.subjectType == '1ST_SEM'
12337     ? this.props.quarter
12338     : this.props.quarter == 'Q3'
12339     ? 'Q1'
12340     : 'Q2'
12341   }/comp/${this.state.qeData.
12342     componentID}`
12343   }
12344 >
12345   import React, { Component } from 'react';
12346   import PropTypes from 'prop-types';
12347   import { bindActionCreators } from 'redux';
12348   import { connect } from 'react-redux';
12349   import * as actions from './redux/actions';
12350   import NavBar from './NavBar';
12351   import RegistrarManageGrades from './RegistrarManageGrades';
12352   import { Container, Grid } from 'tabler-react';
12353   import { Result, Button } from 'antd';
12354   import axios from 'axios';
12355   import { Link } from 'react-router-dom';
12356
12357   export class
12358     RegistrarManageGradesView extends
12359       Component {
12360       static propTypes = {
12361         app: PropTypes.object.isRequired,
12362         actions: PropTypes.object.
12363          isRequired,
12364       };
12365
12366       constructor(props) {
12367         super(props);
12368         this.state = {
12369           locked: false,
12370           isLoading: true,
12371         };
12372
12373       componentDidMount() {
12374         this.props.actions.setLoadingTrue()
12375         if (!this.props.app.auth.
12376           isAuthenticated) {
12377             this.props.history.push('/login')
12378           }
12379         } else {
12380           if (this.props.app.auth.user.
12381             position != 2) {
12382             this.props.history.push('/page401');
12383           } else {
12384             axios
12385               .get('api/registrar/getsy')
12386               .then(res => {
12387                 this.props.actions.
12388                   setLoadingFalse();
12389                 this.setState({ locked:

```

```

        false, isLoading:
        false });
    })
    .catch(err => {
      this.props.actions.
        setLoadingFalse();
      this.setState({ locked:
        true, isLoading:
        false });
    });
  }
}

render() {
  return (
    <div className="app-registrar-
      manage-grades-view fh">
      {this.state.isLoading ? (
        '',
      ) : this.state.locked ? (
        <NavBar>
          <Container>
            <Grid.Row>
              <Grid.Col xs={12} sm
                ={12} md={12}>
                <Result
                  status="403"
                  title="No Active
                    School Year"
                  subTitle="Sorry,
                    you are not
                      authorized to
                        access this
                          page now.
                            Please
                              contact your
                                system
                                  administrator
                                    for details.
          "
        extra={
          <Link to="/">
            <Button type=
              "primary"
              >Back
                Home </
                  Button>
            </Link>
          }
        />
      </Grid.Col>
    </Grid.Row>
  </Container>
</NavBar>
) : (
  <NavBar>
    <Container>
      <Grid.Row>
        <Grid.Col xs={12} sm
          ={12} md={12}>
            <
              RegistrarManageGrades
              classRecordID={

                this.props.
                  match.params.
                  classRecordID
              }
              quarter={this.
                props.match.
                params.Q}
              id={this.props.
                match.params.
                id}
            </

```

```

12423                               locked={false}
12424                               />
12425                               </Grid.Col>
12426                               </Grid.Row>
12427                               </Container>
12428                               </NavBar>
12429                               )}           )
12430                               </div>
12431                               );
12432                           }
12433                       }
12434
12435 /* istanbul ignore next */
12436 function mapStateToProps(state) {
12437     return {
12438         app: state.app,
12439     };
12440
12441 /* istanbul ignore next */
12442 function mapDispatchToProps(dispatch)
12443     {
12444         return {
12445             actions: bindActionCreators({ ...
12446             actions }, dispatch),
12447         };
12448
12449         export default connect(
12450             mapStateToProps ,
12451             mapDispatchToProps)(
12452                 RegistrarManageGradesView);
12453                     View
12454                     </Link>
12455                     </Button>
12456                     </Button.List>
12457                     </Card.Options>
12458                     </Card.Header>
12459                     <Card.Body>
12460                         <Table responsive={true}>
12461                             <Table.Header>
12462                                 <Table.ColHeader>Subcomponent Name</
12463                                     Table.ColHeader>
12464                                 <Table.ColHeader>Weight </Table.
12465                                     ColHeader>
12466                             </Table.Header>
12467                             <Table.Body>{displayQeData}</Table.Body>
12468                         </Table>
12469                     </Card.Body>
12470                     </Card>
12471                     </Grid.Col>
12472                     </Grid.Row>
12473                     </Spin>
12474                     </Card.Body>
12475                 )}
12476             <Card.Footer>
12477                 <Button.List align="right">
12478                     {((this.props.app.auth.user.position == 2 ||
12479                         this.props.app.auth.user.position == 3) && (
12480                             <ViewEditLog
12481                             classRecordID={this.props.classRecordID}
12482                             quarter={
12483                                 this.state.subjectType == 'NON_SHS' || this.
12484                                     state.subjectType == '1ST_SEM'
12485                                     ? this.props.quarter
12486                                     : this.props.quarter == 'Q3'
12487                                     ? 'Q1'
12488                                     : 'Q2'
12489                             }
12490                             position="Registrar"
12491                         />
12492                     )}           )

```

```

12487             </Button.List>
12488         </Card.Footer>
12489     </Card>
12490     </Grid.Row>
12491   </Container>
12492 </div>
12493 );
12494 }
12495 }
12496 */
12497 /* istanbul ignore next */
12498 function mapStateToProps(state) {
12499   return {
12500     app: state.app,
12501   };
12502 }
12503 */
12504 /* istanbul ignore next */
12505 function mapDispatchToProps(dispatch) {
12506   return {
12507     actions: bindActionCreators({ ...actions }, dispatch),
12508   };
12509 }
12510 */
12511 export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarManageGrades);
12512 import React, { Component } from 'react';
12513 import PropTypes from 'prop-types';
12514 import { bindActionCreators } from 'redux';
12515 import { connect } from 'react-redux';
12516 import * as actions from './redux/actions';
12517 import { Card, Button, Grid, Table, Form, Container } from 'tabler-react';
12518 import axios from 'axios';
12519 import { Pagination, Spin, Breadcrumb } from 'antd';
12520 import { Link } from 'react-router-dom';
12521 export class RegistrarManageStudents extends Component {
12522   static propTypes = {
12523     app: PropTypes.object.isRequired,
12524     actions: PropTypes.object.isRequired,
12525   };
12526   constructor(props) {
12527     super(props);
12528     this.state = {
12529       isLoading: false,
12530       keyword: '',
12531       page: 1,
12532       pageSize: 10,
12533       numOfPages: 1,
12534       data: [],
12535       selectedKey: 0,
12536     };
12537   };
12538   this.onChangeSearch = this.onChangeSearch.bind(this);
12539 }
12540 */
12541 onChangeSearch(event) {
12542   this.setState({ [event.target.name]: event.target.value });
12543   this.setState({ isLoading: true, page: 1 });
12544   axios
12545     .post('api/registrar/getsections', {
12546       keyword: event.target.value,
12547       page: 1,
12548       pageSize: this.state.pageSize,
12549     })
12550     .then(res => {
12551       this.setState({ isLoading: false });
12552       this.setState({ numOfPages: res.data.numOfPages, data: res.data
12553         .sectionList });
12554     })
12555     .catch(err => {
12556       this.setState({ isLoading: false });

```

```

12557         this.setState({ data: [] });
12558     });
12559 }
12560
12561 paginate = page => {
12562     this.setState({
12563     page,
12564   });
12565     this.setState({ isLoading: true });
12566     axios
12567       .post('api/registrar/getsections', {
12568         keyword: this.state.keyword,
12569         page,
12570         pageSize: this.state.pageSize,
12571     })
12572       .then(res => {
12573         this.setState({ isLoading: false });
12574         this.setState({
12575           numOfPages: res.data.numOfPages,
12576           data: res.data.sectionList,
12577         });
12578     })
12579       .catch(err => {
12580         this.setState({ isLoading: false });
12581     });
12582 };
12583
12584 componentWillMount() {
12585     this.setState({ isLoading: true });
12586     axios
12587       .post('api/registrar/getsections', { keyword: '' , page: 1 ,
12588         pageSize: 10 })
12589       .then(res => {
12590         this.setState({ isLoading: false });
12591         this.setState({ numOfPages: res.data.numOfPages , data: res.data
12592           .sectionList });
12593     })
12594       .catch(err => {
12595         this.setState({ isLoading: false });
12596     });
12597 }
12598
12599 componentWillMountReceiveProps() {
12600     this.setState({ isLoading: true });
12601     axios
12602       .post('api/registrar/getsections', {
12603         keyword: this.state.keyword,
12604         page: this.state.page,
12605         pageSize: 10,
12606     })
12607       .then(res => {
12608         this.setState({ isLoading: false });
12609         this.setState({ numOfPages: res.data.numOfPages , data: res.data
12610           .sectionList });
12611     })
12612       .catch(err => {
12613         this.setState({ isLoading: false });
12614     });
12615 }
12616
12617 render() {
12618   const displayGradeLevel = gradeLevel => {
12619     switch (gradeLevel) {
12620       case 'N':
12621         return 'Nursery';
12622       case 'K1':
12623         return 'Kinder 1';
12624       case 'K2':
12625         return 'Kinder 2';
12626       case 'G1':
12627         return 'Grade 1';
12628       case 'G2':
12629         return 'Grade 2';

```

```

12627     case 'G3':
12628         return 'Grade 3';
12629     case 'G4':
12630         return 'Grade 4';
12631     case 'G5':
12632         return 'Grade 5';
12633     case 'G6':
12634         return 'Grade 6';
12635     case 'G7':
12636         return 'Grade 7';
12637     case 'G8':
12638         return 'Grade 8';
12639     case 'G9':
12640         return 'Grade 9';
12641     case 'G10':
12642         return 'Grade 10';
12643     case 'G11':
12644         return 'Grade 11';
12645     case 'G12':
12646         return 'Grade 12';
12647     default:
12648         return '';
12649     }
12650 };
12651 const { errors } = this.props.app;
12652 const DisplayData = [];
12653 for (const [index, value] of this.state.data.entries()) {
12654     DisplayData.push(
12655         <Table.Row>
12656             <Table.Col>{value.name}</Table.Col>
12657             <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
12658             <Table.Col alignContent="center">
12659                 <Link to={`/managestudents/viewenrolled/${value.key}`}>
12660                     <Button icon="user" size="sm" pill color="primary">
12661                         View Enrolled Students
12662                     </Button>
12663                 </Link>
12664             </Table.Col>
12665         </Table.Row>,
12666     );
12667 }
12668 return (
12669     <Table.Body className="app-registrar-enroll-students my-3 my-md-5
12670         card">
12671         <Card.Body>
12672             <Card.Title>
12673                 <Breadcrumb>
12674                     <Breadcrumb.Item>Sections</Breadcrumb.Item>
12675                     <Breadcrumb.Item>Manage Students</Breadcrumb.Item>
12676                 </Breadcrumb>
12677             </Card.Title>
12678             <Card.Title>Manage Students</Card.Title>
12679             <Grid.Row>
12680                 <Grid.Col sm={12} md={12} xs={12}>
12681                     <Grid.Col sm={12} md={12} xs={12}>
12682                         <Form.Group>
12683                             <Form.Input
12684                                 icon="search"
12685                                 placeholder="Search for..."
12686                                 position="append"
12687                                 name="keyword"
12688                                 value={this.state.keyword}
12689                                 onChange={this.onChangeSearch}
12690                             />
12691                         </Form.Group>
12692                     </Grid.Col>
12693                 </Grid.Row>
12694                 <Spin spinning={this.state.isLoading}>
12695                     <Table highlightRowOnHover={true} responsive={true}>
12696                         <Table.Header>
12697                             <Table.ColHeader>Section Name</Table.ColHeader>
12698                             <Table.ColHeader>Grade Level</Table.ColHeader>
12699                             <Table.ColHeader alignContent="center">Action</
12700                         <Table.ColHeader>
```

```

12700     </Table.Header>
12701     <Table.Body>
12702       {DisplayData.length == 0 ? (
12703         <Table.Row>
12704           <Table.Col colSpan={3} alignContent="center">
12705             No entries.
12706           </Table.Col>
12707         ) : (
12708           DisplayData
12709         )}
12710       </Table.Body>
12711     </Table>
12712     <Pagination
12713       size="large"
12714       current={this.state.page}
12715       pageSize={this.state.pageSize}
12716       total={this.state.pageSize * this.state.numOfPages}
12717       onChange={this.paginate}
12718     />
12719     </Spin>
12720   </Grid.Col>
12721 </Grid.Row>
12722   </Card.Body>
12723 </Table.Body>
12724 );
12725 }
12726 }
12727 */
12728 /* istanbul ignore next */
12729 function mapStateToProps(state) {
12730   return {
12731     app: state.app,
12732   };
12733 }
12734 */
12735 /* istanbul ignore next */
12736 function mapDispatchToProps(dispatch) {
12737   return {
12738     actions: bindActionCreators({ ...actions }, dispatch),
12739   };
12740 }
12741 */
12742 export default connect(mapStateToProps, mapDispatchToProps)(
12743   RegistrarManageStudents);
12744 import React, { Component } from 'react';
12745 import PropTypes from 'prop-types';
12746 import { bindActionCreators } from 'redux';
12747 import { connect } from 'react-redux';
12748 import * as actions from './redux/actions';
12749 import moment from 'moment';
12750 import axios from 'axios';
12751 import { Container, Grid, Card, Button, Form, Header, List } from 'tabler-react';
12752 import { Alert, Upload, message } from 'antd';
12753 import cn from 'classnames';
12754 import bg from '../images/BG.png';
12755 import { getImageUrl, getLocalUrl } from '../../utils';
12756 import placeholder from '../../images/placeholder.jpg';
12757 function ProfileImage({ avatarURL }) {
12758   return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
12759 }
12760 */
12761 function Profile({
12762   className,
12763   children,
12764   name,
12765   avatarURL = '',
12766   twitterURL = '',
12767   backgroundURL = '',
12768   bio,
12769 }) {

```

```

12770     const classes = cn('card-profile', className);
12771     return (
12772       <Card className={classes}>
12773         <Card.Header backgroundURL={backgroundURL} />
12774         <Card.Body className="text-center">
12775           <ProfileImage avatarURL={avatarURL} />
12776           <Header.H3 className="mb-3">{name}</Header.H3>
12777           <p className="mb-4">{bio || children}</p>
12778         </Card.Body>
12779       </Card>
1280     );
1281   }
1282   export class RegistrarProfile extends Component {
1283     static propTypes = {
1284       app: PropTypes.object.isRequired,
1285       actions: PropTypes.object.isRequired,
1286     };
1287     constructor(props) {
1288       super(props);
1289       this.state = {
1290         loading: true,
1291         email: '',
1292         position: '',
1293         firstName: '',
1294         lastName: '',
1295         middleName: '',
1296         suffix: '',
1297         nickname: '',
1298         imageUrl: '',
1299         contactNum: '',
1300         address: '',
1301         province: '',
1302         city: '',
1303         region: '',
1304         zipcode: '',
1305         civilStatus: '',
1306         sex: '',
1307         citizenship: '',
1308         birthDate: new Date(),
1309         birthPlace: '',
1310         religion: '',
1311         emergencyName: '',
1312         emergencyAddress: '',
1313         emergencyTelephone: '',
1314         emergencyCellphone: '',
1315         emergencyEmail: '',
1316         emergencyRelationship: '',
1317       };
1318     };
1319     this.onSubmit = this.onSubmit.bind(this);
1320     this.onChange = this.onChange.bind(this);
1321     this.onDateChange = this.onDateChange.bind(this);
1322   }
1323   componentWillReceiveProps(nextProps) {
1324     if (!nextProps.app.showLoading && Object.keys(nextProps.app.errors)
1325       .length == 0)
1326       this.setState({
1327         email: nextProps.app.auth.user.email,
1328         position: nextProps.app.auth.user.position,
1329         firstName: nextProps.app.profile.firstName,
1330         lastName: nextProps.app.profile.lastName,
1331         middleName: nextProps.app.profile.middleName,
1332         suffix: nextProps.app.profile.suffix,
1333         nickname: nextProps.app.profile.nickname,
1334         imageUrl: nextProps.app.profile.imageUrl,
1335         contactNum: nextProps.app.profile.contactNum,
1336         address: nextProps.app.profile.address,
1337         province: nextProps.app.profile.province,
1338         city: nextProps.app.profile.city,
1339         region: nextProps.app.profile.region,
1340         zipcode: nextProps.app.profile.zipcode,
1341         civilStatus: nextProps.app.profile.civilStatus,
1342         sex: nextProps.app.profile.sex,
1343         citizenship: nextProps.app.profile.citizenship,
1344         birthDate: nextProps.app.profile.birthDate,

```

```

12845     birthPlace: nextProps.app.profile.birthPlace,
12846     religion: nextProps.app.profile.religion,
12847     emergencyName: nextProps.app.profile.emergencyName,
12848     emergencyAddress: nextProps.app.profile.emergencyAddress,
12849     emergencyTelephone: nextProps.app.profile.emergencyTelephone,
12850     emergencyCellphone: nextProps.app.profile.emergencyCellphone,
12851     emergencyEmail: nextProps.app.profile.emergencyEmail,
12852     emergencyRelationship: nextProps.app.profile.
12853         emergencyRelationship,
12854     loading: false,
12855   );
12856 }
12857
12858   onChange(event) {
12859     this.setState({ [event.target.name]: event.target.value });
12860   }
12861
12862   onDateChange(event) {
12863     this.setState({ birthDate: event });
12864   }
12865
12866   onSubmit(event) {
12867     event.preventDefault();
12868     const {
12869       firstName,
12870       lastName,
12871       middleName,
12872       suffix,
12873       nickname,
12874       contactNum,
12875       address,
12876       province,
12877       city,
12878       region,
12879       zipcode,
12880       civilStatus,
12881       sex,
12882       citizenship,
12883       birthDate,
12884       birthPlace,
12885       religion,
12886       emergencyName,
12887       emergencyAddress,
12888       emergencyTelephone,
12889       emergencyCellphone,
12890       emergencyEmail,
12891       emergencyRelationship,
12892     } = this.state;
12893     const profileData = {
12894       firstName,
12895       lastName,
12896       middleName,
12897       suffix,
12898       nickname,
12899       contactNum,
12900       address,
12901       province,
12902       city,
12903       region,
12904       zipcode,
12905       civilStatus,
12906       sex,
12907       citizenship,
12908       birthDate,
12909       birthPlace,
12910       religion,
12911       emergencyName,
12912       emergencyAddress,
12913       emergencyTelephone,
12914       emergencyCellphone,
12915       emergencyEmail,
12916       emergencyRelationship,
12917     };
12918     console.log(profileData);
12919     this.props.actions.updateRegistrarProfile(profileData);
12920   }

```

```

12921     render() {
12922       const {
12923         email,
12924         position,
12925         firstName,
12926         lastName,
12927         middleName,
12928         suffix,
12929         nickname,
12930         imageUrl,
12931         contactNum,
12932         address,
12933         province,
12934         city,
12935         region,
12936         zipcode,
12937         civilStatus,
12938         sex,
12939         citizenship,
12940         birthPlace,
12941         religion,
12942         emergencyName,
12943         emergencyAddress,
12944         emergencyTelephone,
12945         emergencyCellphone,
12946         emergencyEmail,
12947         emergencyRelationship,
12948     } = this.state;
12949     const birthDate = new Date(this.state.birthDate);
12950     const defaultDate = birthDate.toISOString();
12951     const { errors } = this.props.app;
12952     const uploadLoading = false;
12953     const displayPosition = position => {
12954       switch (position) {
12955         case false:
12956           return 'Administrator';
12957         case true:
12958           return 'Director';
12959         case 2:
12960           return 'Registrar';
12961         case 3:
12962           return 'Teacher';
12963         case 4:
12964           return 'Student';
12965         case 5:
12966           return 'Guardian';
12967         default:
12968           return '';
12969     }
12970   };
12971   const capitalize = string => {
12972     return string.charAt(0).toUpperCase() + string.slice(1).
12973      toLowerCase();
12974   };
12975   return (
12976     <div className="app-admin-profile my-3 my-md-5">
12977       {this.state.loading ? (
12978         '',
12979       ) : (
12980         <Container>
12981           <Grid.Row>
12982             <Grid.Col sm={12} lg={4}>
12983               <Profile
12984                 name={`${firstName} ${middleName.charAt(0).
12985                  toUpperCase()} ${lastName}`}
12986                 avatarURL={imageUrl === 'NA' ? placeholder :
12987                   getImageUrl(imageUrl)}
12988                 backgroundURL={bg}
12989               >
12990                 <Upload
12991                   name="file"
12992                     multiple={false}
12993                     accept=".png, .jpg"
12994                     customRequest={(req, uploadLoading) => {
12995                       // Axios Photo Upload
12996                       this.props.actions.setLoading(true);
12997                     }
12998                   }
12999                 </Upload>
13000               </Profile>
13001             </Grid.Col>
13002           </Grid.Row>
13003         </Container>
13004       )
13005     </div>
13006   );
13007 
```

```

12994         const bodyFormData = new FormData();
12995         bodyFormData.set('file', req.file);
12996         axios({
12997             method: 'post',
12998             url: 'api/users/upload',
12999             data: bodyFormData,
13000         })
13001         .then(res => {
13002             req.onSuccess();
13003             this.props.actions.setLoading(false);
13004             message.success('Uploaded Successfully!');
13005                 then(() => {
13006                     window.location.href = '/profile';
13007                 });
13008             })
13009             .catch(err => {
13010                 req.onError();
13011                 this.props.actions.setLoading(false);
13012                 message.error('Upload failed!');
13013             });
13014         })
13015     > <div>
13016         <Header.H4>{displayPosition(position)}</Header.H4>
13017     </div>
13018     <Button loading={this.props.app.showLoading} icon="upload" pill color="primary">
13019         Upload Image
13020     </Button>
13021     </Upload>
13022     </Profile>
13023 </Grid.Col>
13024 <Grid.Col xs={12} sm={12} lg={8}>
13025     <Form className="card" onSubmit={this.onSubmit}>
13026         <Card.Body>
13027             <Card.Title>Edit Profile</Card.Title>
13028             <Grid.Row>
13029                 <Grid.Col xs={12} sm={12} md={6}>
13030                     <Form.Group>
13031                         <Form.Label>Email</Form.Label>
13032                         <Form.Input
13033                             type="email"
13034                             disabled
13035                             placeholder="Email"
13036                             value={email}
13037                             error={errors.email}
13038                         />
13039                     </Form.Group>
13040                 </Grid.Col>
13041                 <Grid.Col sm={12} md={6}>
13042                     <Form.Group>
13043                         <Form.Label>Position</Form.Label>
13044                         <Form.Input
13045                             type="text"
13046                             placeholder="Position"
13047                             disabled
13048                             value={displayPosition(position)}
13049                             error={errors.position}
13050                         />
13051                     </Form.Group>
13052                 </Grid.Col>
13053                 <Grid.Col sm={12} md={4}>
13054                     <Form.Group>
13055                         <Form.Label>First Name</Form.Label>
13056                         <Form.Input
13057                             type="text"
13058                             name="firstName"
13059                             placeholder="First Name"
13060                             value={firstName}
13061                             onChange={this.onChange}
13062                             error={errors.firstName}
13063                         />

```

```

13064      </Form.Group>
13065    </Grid.Col>
13066    <Grid.Col sm={12} md={4}>
13067      <Form.Group>
13068        <Form.Label>Middle Name</Form.Label>
13069        <Form.Input
13070          type="text"
13071            placeholder="Middle Name"
13072            value={middleName}
13073            onChange={this.onChange}
13074            name="middleName"
13075            error={errors.middleName}
13076          />
13077      </Form.Group>
13078    </Grid.Col>
13079    <Grid.Col sm={12} md={4}>
13080      <Form.Group>
13081        <Form.Label>Last Name</Form.Label>
13082        <Form.Input
13083          type="text"
13084            placeholder="Last Name"
13085            value={lastName}
13086            onChange={this.onChange}
13087            name="lastName"
13088            error={errors.lastName}
13089          />
13090      </Form.Group>
13091    </Grid.Col>
13092    <Grid.Col sm={12} md={3}>
13093      <Form.Group>
13094        <Form.Label>Nickname</Form.Label>
13095        <Form.Input
13096          type="text"
13097            placeholder="Nickname"
13098            value={nickname}
13099            onChange={this.onChange}
13100            name="nickname"
13101            error={errors.nickname}
13102          />
13103      </Form.Group>
13104    </Grid.Col>
13105    <Grid.Col sm={12} md={3}>
13106      <Form.Group>
13107        <Form.Label>Suffix</Form.Label>
13108        <Form.Input
13109          type="text"
13110            placeholder="Suffix"
13111            value={suffix}
13112            onChange={this.onChange}
13113            name="suffix"
13114            error={errors.suffix}
13115          />
13116      </Form.Group>
13117    </Grid.Col>
13118    <Grid.Col sm={12} md={6}>
13119      <Form.Group>
13120        <Form.Label>Contact Number</Form.Label>
13121        <Form.Input
13122          type="text"
13123            placeholder="Contact Number"
13124            value={contactNum}
13125            onChange={this.onChange}
13126            name="contactNum"
13127            error={errors.contactNum}
13128          />
13129      </Form.Group>
13130    </Grid.Col>
13131    <Grid.Col sm={12} md={12}>
13132      <Form.Group>
13133        <Form.Label>Address</Form.Label>
13134        <Form.Input
13135          type="text"
13136            placeholder="Home Address"

```

```

13137           value={address}
13138           onChange={this.onChange}
13139           name="address"
13140           error={errors.address}
13141       />
13142     </Form.Group>
13143   </Grid.Col>
13144   <Grid.Col sm={12} md={4}>
13145     <Form.Group>
13146       <Form.Label>Province</Form.Label>
13147       <Form.Input
13148         type="text"
13149         placeholder="Province"
13150         value={province}
13151         onChange={this.onChange}
13152         name="province"
13153         error={errors.province}
13154       />
13155     </Form.Group>
13156   </Grid.Col>
13157   <Grid.Col sm={12} md={4}>
13158     <Form.Group>
13159       <Form.Label>City</Form.Label>
13160       <Form.Input
13161         type="text"
13162         placeholder="City"
13163         value={city}
13164         onChange={this.onChange}
13165         name="city"
13166         error={errors.city}
13167       />
13168     </Form.Group>
13169   </Grid.Col>
13170   <Grid.Col sm={12} md={2}>
13171     <Form.Group>
13172       <Form.Label>Region</Form.Label>
13173       <Form.Input
13174         type="text"
13175         placeholder="Region"
13176         value={region}
13177         onChange={this.onChange}
13178         name="region"
13179         error={errors.region}
13180       />
13181     </Form.Group>
13182   </Grid.Col>
13183   <Grid.Col sm={12} md={2}>
13184     <Form.Group>
13185       <Form.Label>Zipcode</Form.Label>
13186       <Form.Input
13187         type="text"
13188         placeholder="Postal Code"
13189         value={zipcode}
13190         onChange={this.onChange}
13191         name="zipcode"
13192         error={errors.zipcode}
13193       />
13194     </Form.Group>
13195   </Grid.Col>
13196   <Grid.Col sm={12} md={4}>
13197     <Form.Group>
13198       <Form.Label>Civil Status</Form.Label>
13199       <Form.Select
13200         value={civilStatus}
13201         onChange={this.onChange}
13202         name="civilStatus"
13203       >
13204         <option>SINGLE</option>
13205         <option>MARRIED</option>
13206         <option>WIDOWED</option>
13207         <option>OTHERS</option>
13208       </Form.Select>
13209     </Form.Group>

```

```

13210 </Grid.Col>
13211 <Grid.Col sm={12} md={2}>
13212 <Form.Group>
13213 <Form.Label>Sex</Form.Label>
13214 <Form.Select value={sex} onChange={this.
13215   onChange} name="sex">
13216   <option>M</option>
13217   <option>F</option>
13218 </Form.Select>
13219 </Form.Group>
13220 </Grid.Col>
13221 <Grid.Col sm={12} md={6}>
13222 <Form.Group>
13223 <Form.Label>Citizenship</Form.Label>
13224 <Form.Input
13225   type="text"
13226   placeholder="Citizenship"
13227   value={citizenship}
13228   onChange={this.onChange}
13229   name="citizenship"
13230   error={errors.citizenship}
13231 />
13232 </Form.Group>
13233 </Grid.Col>
13234 <Grid.Col sm={12} md={4}>
13235 <Form.Group>
13236 <Form.Label>Birth Date</Form.Label>
13237 <Form.DatePicker
13238   defaultDate={new Date(defaultDate)}
13239   format="mm/dd/yyyy"
13240   onChange={this.onDateChange}
13241   name="birthDate"
13242   maxYear={2020}
13243   minYear={1897}
13244   monthLabels={[
13245     'January',
13246     'February',
13247     'March',
13248     'April',
13249     'May',
13250     'June',
13251     'July',
13252     'August',
13253     'September',
13254     'October',
13255     'November',
13256     'December',
13257   ]}
13258 ></Form.DatePicker>
13259 </Form.Group>
13260 </Grid.Col>
13261 <Grid.Col sm={12} md={4}>
13262 <Form.Group>
13263 <Form.Label>Birth Place</Form.Label>
13264 <Form.Input
13265   type="text"
13266   placeholder="Birth Place"
13267   value={birthPlace}
13268   onChange={this.onChange}
13269   name="birthPlace"
13270   error={errors.birthPlace}
13271 />
13272 </Form.Group>
13273 </Grid.Col>
13274 <Grid.Col sm={12} md={4}>
13275 <Form.Group>
13276 <Form.Label>Religion</Form.Label>
13277 <Form.Input
13278   type="text"
13279   placeholder="Religion"
13280   value={religion}
13281   onChange={this.onChange}
13282   name="religion"
13283   error={errors.religion}

```

```

13283                     />
13284             </Form.Group>
13285         </Grid.Col>
13286     </Grid.Row>
13287 </Card.Body>
13288 <Card.Body>
13289     <Card.Title>Emergency Contact Information</Card.
13290     Title>
13291     <Grid.Row>
13292         <Grid.Col sm={12} md={6}>
13293             <Form.Group>
13294                 <Form.Label>Contact Person</Form.Label>
13295                 <Form.Input
13296                     type="text"
13297                     placeholder="Contact Person"
13298                     value={emergencyName}
13299                     onChange={this.onChange}
13300                     name="emergencyName"
13301                     error={errors.emergencyName}
13302                 />
13303             </Form.Group>
13304         </Grid.Col>
13305         <Grid.Col sm={12} md={6}>
13306             <Form.Group>
13307                 <Form.Label>Contact Address</Form.Label>
13308                 <Form.Input
13309                     type="text"
13310                     placeholder="Contact Address"
13311                     value={emergencyAddress}
13312                     onChange={this.onChange}
13313                     name="emergencyAddress"
13314                     error={errors.emergencyAddress}
13315                 />
13316             </Form.Group>
13317         </Grid.Col>
13318         <Grid.Col sm={12} md={6}>
13319             <Form.Group>
13320                 <Form.Label>Contact Telephone No.</Form.Label
13321                 >
13322                 <Form.Input
13323                     type="text"
13324                     placeholder="Contact Telephone No."
13325                     value={emergencyTelephone}
13326                     onChange={this.onChange}
13327                     name="emergencyTelephone"
13328                     error={errors.emergencyTelephone}
13329                 />
13330             </Form.Group>
13331         <Grid.Col sm={12} md={6}>
13332             <Form.Group>
13333                 <Form.Label>Contact Cellphone No.</Form.Label
13334                 >
13335                 <Form.Input
13336                     type="text"
13337                     placeholder="Contact Cellphone No."
13338                     value={emergencyCellphone}
13339                     onChange={this.onChange}
13340                     name="emergencyCellphone"
13341                     error={errors.emergencyCellphone}
13342                 />
13343             </Form.Group>
13344         <Grid.Col sm={12} md={6}>
13345             <Form.Group>
13346                 <Form.Label>Contact Email</Form.Label>
13347                 <Form.Input
13348                     type="text"
13349                     placeholder="Contact Email"
13350                     value={emergencyEmail}
13351                     onChange={this.onChange}
13352                     name="emergencyEmail"
13353                     error={errors.emergencyEmail}
13354                 />

```

```

13354             </Form.Group>
13355         </Grid.Col>
13356         <Grid.Col sm={12} md={6}>
13357             <Form.Group>
13358                 <Form.Label>Contact Relationship</Form.Label>
13359                 <Form.Input
13360                     type="text"
13361                     placeholder="Contact Relationship"
13362                     value={emergencyRelationship}
13363                     onChange={this.onChange}
13364                     name="emergencyRelationship"
13365                     error={errors.emergencyRelationship}
13366                 />
13367             </Form.Group>
13368         </Grid.Col>
13369     </Grid.Row>
13370   </Card.Body>
13371   <Card.Footer className="text-right">
13372     <Button type="submit" color="primary">
13373       Update Profile
13374     </Button>
13375   </Card.Footer>
13376 </Form>
13377     </Grid.Col>
13378   </Grid.Row>
13379 </Container>
13380   )
13381   );
13382 );
13383 }
13384 }
13385 */
13386 /* istanbul ignore next */
13387 function mapStateToProps(state) {
13388   return {
13389     app: state.app,
13390   };
13391 }
13392 */
13393 /* istanbul ignore next */
13394 function mapDispatchToProps(dispatch) {
13395   return {
13396     actions: bindActionCreators({ ...actions }, dispatch),
13397   };
13398 }
13399 */
13400 export default connect(mapStateToProps, mapDispatchToProps)(
13401   RegistrarProfile);
13402 import React, { Component } from 'react';
13403 import PropTypes from 'prop-types';
13404 import { bindActionCreators } from 'redux';
13405 import { connect } from 'react-redux';
13406 import * as actions from './redux/actions';
13407 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
13408   from 'tabler-react';
13409 import axios from 'axios';
13410 import { Pagination, Spin } from 'antd';
13411 import { Modal, Popconfirm, Breadcrumb } from 'antd';
13412 */
13413 export class RegistrarSectionsList extends Component {
13414   static propTypes = {
13415     app: PropTypes.object.isRequired,
13416     actions: PropTypes.object.isRequired,
13417   };
13418   constructor(props) {
13419     super(props);
13420     this.state = {
13421       isLoading: false,
13422       keyword: '',
13423       page: 1,
13424       pageSize: 10,
13425       numOfPages: 1,

```

```

13425         data: [],
13426         showAddSection: false,
13427         addSectionLoading: false,
13428         errors: {},
13429         sectionName: '',
13430         gradeLevel: 'N',
13431         showEditSection: false,
13432         editSectionLoading: false,
13433         selectedKey: 0,
13434     );
13435
13436     this.showAddSection = this.showAddSection.bind(this);
13437     this.hideAddSection = this.hideAddSection.bind(this);
13438     this.onChange = this.onChange.bind(this);
13439     this.onChangeSearch = this.onChangeSearch.bind(this);
13440     this.addSection = this.addSection.bind(this);
13441     this.hideEditSection = this.hideEditSection.bind(this);
13442     this.editSection = this.editSection.bind(this);
13443     this.deleteSection = this.deleteSection.bind(this);
13444 }
13445
13446     showEditSection(key) {
13447         this.setState({
13448             showEditSection: true,
13449             sectionName: this.state.data.filter(section => section.key ===
13450                 key)[0].name,
13451             gradeLevel: this.state.data.filter(section => section.key === key
13452                 )[0].gradeLevel,
13453             selectedKey: key,
13454             errors: {}
13455         });
13456     }
13457
13458     editSection() {
13459         this.props.actions.editSection({
13460             sectionID: this.state.selectedKey,
13461             sectionName: this.state.sectionName,
13462             gradeLevel: this.state.gradeLevel,
13463         });
13464     }
13465
13466     hideEditSection() {
13467         this.setState({
13468             showEditSection: false,
13469             editSectionLoading: false,
13470             sectionName: '',
13471             gradeLevel: 'N',
13472             errors: {}
13473         });
13474     }
13475
13476     deleteSection() {
13477         this.props.actions.deleteSection({ sectionID: this.state.
13478             selectedKey });
13479     }
13480
13481     showAddSection() {
13482         this.setState({
13483             showAddSection: true,
13484             sectionName: '',
13485             gradeLevel: 'N',
13486             errors: {}
13487         });
13488     }
13489
13490     hideAddSection() {
13491         this.setState({ showAddSection: false, sectionName: '', gradeLevel:
13492             'N', errors: {} });
13493         this.props.actions.getErrors({});
13494     }
13495
13496     onChange(event) {
13497         this.setState({ [event.target.name]: event.target.value });

```

```

13494     }
13495
13496     onChangeSearch(event) {
13497       this.setState({ [event.target.name]: event.target.value });
13498       this.setState({ isLoading: true, page: 1 });
13499       axios
13500         .post('api/registrar/getsections', {
13501           keyword: event.target.value,
13502           page: 1,
13503           pageSize: this.state.pageSize,
13504         })
13505         .then(res => {
13506           this.setState({ isLoading: false });
13507           this.setState({ numOfPages: res.data.numOfPages, data: res.data
13508             .sectionList });
13509         })
13510         .catch(err => {
13511           this.setState({ isLoading: false });
13512           this.setState({ data: [] });
13513         });
13514     }
13515     addSection() {
13516       this.props.actions.addSection({
13517         sectionName: this.state.sectionName,
13518         gradeLevel: this.state.gradeLevel,
13519       });
13520     }
13521
13522     paginate = page => {
13523       this.setState({
13524         page,
13525       });
13526       this.setState({ isLoading: true });
13527       axios
13528         .post('api/registrar/getsections', {
13529           keyword: this.state.keyword,
13530           page,
13531           pageSize: this.state.pageSize,
13532         })
13533         .then(res => {
13534           this.setState({ isLoading: false });
13535           this.setState({
13536             numOfPages: res.data.numOfPages,
13537             data: res.data.sectionList,
13538           });
13539         })
13540         .catch(err => {
13541           this.setState({ isLoading: false });
13542         });
13543     };
13544
13545     componentWillMount() {
13546       this.setState({ isLoading: true });
13547       axios
13548         .post('api/registrar/getsections', { keyword: '', page: 1,
13549           pageSize: 10 })
13550         .then(res => {
13551           this.setState({ isLoading: false });
13552           this.setState({ numOfPages: res.data.numOfPages, data: res.data
13553             .sectionList });
13554         })
13555         .catch(err => {
13556           this.setState({ isLoading: false });
13557         });
13558     }
13559     componentWillReceiveProps() {
13560       this.setState({ isLoading: true });
13561       axios
13562         .post('api/registrar/getsections', {
13563           keyword: this.state.keyword,
13564           page: this.state.page,

```

```

13564     pageSize: 10,
13565   })
13566   .then(res => {
13567     this.setState({ isLoading: false });
13568     this.setState({ numOfPages: res.data.numOfPages, data: res.data
13569       .sectionList });
13570   })
13571   .catch(err => {
13572     this.setState({ isLoading: false });
13573     this.setState({ data: [] });
13574   });
13575 }
13576
13577 render() {
13578   const displayGradeLevel = gradeLevel => {
13579     switch (gradeLevel) {
13580       case 'N':
13581         return 'Nursery';
13582       case 'K1':
13583         return 'Kinder 1';
13584       case 'K2':
13585         return 'Kinder 2';
13586       case 'G1':
13587         return 'Grade 1';
13588       case 'G2':
13589         return 'Grade 2';
13590       case 'G3':
13591         return 'Grade 3';
13592       case 'G4':
13593         return 'Grade 4';
13594       case 'G5':
13595         return 'Grade 5';
13596       case 'G6':
13597         return 'Grade 6';
13598       case 'G7':
13599         return 'Grade 7';
13600       case 'G8':
13601         return 'Grade 8';
13602       case 'G9':
13603         return 'Grade 9';
13604       case 'G10':
13605         return 'Grade 10';
13606       case 'G11':
13607         return 'Grade 11';
13608       case 'G12':
13609         return 'Grade 12';
13610       default:
13611         return '';
13612     }
13613   };
13614   const { errors } = this.props.app;
13615   const DisplayData = [];
13616   for (const [index, value] of this.state.data.entries()) {
13617     DisplayData.push(
13618       <Table.Row>
13619         <Table.Col>{value.name}</Table.Col>
13620         <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
13621         <Table.Col>
13622           <Button
13623             icon="edit"
13624             size="sm"
13625             pill
13626             color="primary"
13627             value={value.key}
13628             onClick={() => this.showEditSection(value.key)}
13629           />
13630           <span style={{ marginLeft: '10px' }}>
13631             <Popconfirm
13632               title="Do you want to delete this section?"
13633               onConfirm={this.deleteSection}
13634               okText="Delete"
13635               cancelText="Cancel"
13636             >
13637               <Button
13638                 icon="trash"
13639                 size="sm"

```

```

13639         pill
13640         color="danger"
13641         value={value.key}
13642         onClick={() => {
13643             this.setState({ selectedKey: value.key });
13644         }}
13645     />
13646     </Popconfirm>
13647     </span>
13648   </Table.Col>
13649   </Table.Row>,
13650 );
13651 }
13652 return (
13653   <Table.Body className="app-registrar-add-section my-3 my-md-5
13654   card">
13655     <Modal
13656       title="Edit Section"
13657       visible={this.state.showEditSection}
13658       onOk={this.editSection}
13659       onCancel={this.hideEditSection}
13660       okText="Edit section"
13661       confirmLoading={this.props.app.showLoading}
13662       cancelText="Close"
13663     >
13664       <Spin spinning={this.props.app.showLoading}>
13665         <Container>
13666           <Grid.Row>
13667             <Grid.Col sm={12} md={12}>
13668               <Form.Group>
13669                 <Form.Label>Section Name</Form.Label>
13670                 <Form.Input
13671                   type="text"
13672                     name="sectionName"
13673                     placeholder="Section Name"
13674                     value={this.state.sectionName}
13675                     error={errors.sectionName}
13676                     onChange={this.onChange}
13677                 />
13678               </Form.Group>
13679             </Grid.Col>
13680             <Grid.Col sm={12} md={12}>
13681               <Form.Group>
13682                 <Form.Label>Grade Level</Form.Label>
13683                 <Form.Select
13684                   value={this.state.gradeLevel}
13685                   onChange={this.onChange}
13686                   name="gradeLevel"
13687                 >
13688                   <option value="N">Nursery</option>
13689                   <option value="K1">Kinder 1</option>
13690                   <option value="K2">Kinder 2</option>
13691                   <option value="G1">Grade 1</option>
13692                   <option value="G2">Grade 2</option>
13693                   <option value="G3">Grade 3</option>
13694                   <option value="G4">Grade 4</option>
13695                   <option value="G5">Grade 5</option>
13696                   <option value="G6">Grade 6</option>
13697                   <option value="G7">Grade 7</option>
13698                   <option value="G8">Grade 8</option>
13699                   <option value="G9">Grade 9</option>
13700                   <option value="G10">Grade 10</option>
13701                   <option value="G11">Grade 11</option>
13702                   <option value="G12">Grade 12</option>
13703                 </Form.Select>
13704               </Form.Group>
13705             </Grid.Col>
13706           </Grid.Row>
13707         </Container>
13708       </Spin>
13709     <Modal
13710       title="Add a Section"

```

```

13711   visible={this.state.showAddSection}
13712   onOk={this.addSection}
13713   onCancel={this.hideAddSection}
13714   okText="Add section"
13715   confirmLoading={this.props.app.showLoading}
13716   cancelText="Close"
13717 >
13718   <Spin spinning={this.props.app.showLoading}>
13719     <Container>
13720       <Grid.Row>
13721         <Grid.Col sm={12} md={12}>
13722           <Form.Group>
13723             <Form.Label>Section Name</Form.Label>
13724             <Form.Input
13725               type="text"
13726               name="sectionName"
13727               placeholder="Section Name"
13728               value={this.state.sectionName}
13729               error={errors.sectionName}
13730               onChange={this.onChange}
13731             />
13732           </Form.Group>
13733         </Grid.Col>
13734         <Grid.Col sm={12} md={12}>
13735           <Form.Group>
13736             <Form.Label>Grade Level</Form.Label>
13737             <Form.Select
13738               value={this.state.gradeLevel}
13739               onChange={this.onChange}
13740               name="gradeLevel"
13741             >
13742               <option value="N">Nursery</option>
13743               <option value="K1">Kinder 1</option>
13744               <option value="K2">Kinder 2</option>
13745               <option value="G1">Grade 1</option>
13746               <option value="G2">Grade 2</option>
13747               <option value="G3">Grade 3</option>
13748               <option value="G4">Grade 4</option>
13749               <option value="G5">Grade 5</option>
13750               <option value="G6">Grade 6</option>
13751               <option value="G7">Grade 7</option>
13752               <option value="G8">Grade 8</option>
13753               <option value="G9">Grade 9</option>
13754               <option value="G10">Grade 10</option>
13755               <option value="G11">Grade 11</option>
13756               <option value="G12">Grade 12</option>
13757             </Form.Select>
13758           </Form.Group>
13759         </Grid.Col>
13760       </Grid.Row>
13761     </Container>
13762   </Spin>
13763 </Modal>
13764 <Card.Body>
13765   <Card.Title>
13766     <Breadcrumb>
13767       <Breadcrumb.Item>Sections</Breadcrumb.Item>
13768       <Breadcrumb.Item>Sections List</Breadcrumb.Item>
13769     </Breadcrumb>
13770   </Card.Title>
13771   <Card.Title>Sections List</Card.Title>
13772 <Grid.Row>
13773   <Grid.Col sm={12} md={12} xs={12}>
13774     <Grid.Row>
13775       <Grid.Col sm={12} md={10} xs={12}>
13776         <Form.Group>
13777           <Form.Input
13778             icon="search"
13779             placeholder="Search for..."
13780             position="append"
13781             name="keyword"
13782             value={this.state.keyword}
13783             onChange={this.onChangeSearch}>
```

```

13784         />
13785     </Form.Group>
13786   </Grid.Col>
13787   <Grid.Col sm={12} md={2} xs={12}>
13788     <Button icon="plus" block color="primary" onClick={
13789       this.showAddSection}>
13790       Add Section
13791     </Button>
13792   </Grid.Col>
13793 </Grid.Row>
13794 <Spin spinning={this.state.isLoading}>
13795   <Table highlightRowOnHover={true} responsive={true}>
13796     <Table.Header>
13797       <Table.ColHeader>Section Name</Table.ColHeader>
13798       <Table.ColHeader>Grade Level</Table.ColHeader>
13799       <Table.ColHeader>Action</Table.ColHeader>
13800     </Table.Header>
13801     <Table.Body>
13802       {DisplayData.length == 0 ? (
13803         <Table.Row>
13804           <Table.Col colSpan={3} alignContent="center">
13805             No entries.
13806           </Table.Col>
13807         ) : (
13808           DisplayData
13809         )}
13810     </Table.Body>
13811   </Table>
13812   <Pagination
13813     size="large"
13814     current={this.state.page}
13815     pageSize={this.state.pageSize}
13816     total={this.state.pageSize * this.state.numOfPages}
13817     onChange={this.paginate}
13818   />
13819   </Spin>
13820   </Grid.Col>
13821 </Grid.Row>
13822 </Card.Body>
13823 </Table.Body>
13824 );
13825 }
13826 }
13827
13828 /* istanbul ignore next */
13829 function mapStateToProps(state) {
13830   return {
13831     app: state.app,
13832   };
13833 }
13834
13835 /* istanbul ignore next */
13836 function mapDispatchToProps(dispatch) {
13837   return {
13838     actions: bindActionCreators({ ...actions }, dispatch),
13839   };
13840 }
13841
13842 export default connect(mapStateToProps, mapDispatchToProps)(
13843   RegistrarSectionsList);
13844 import React, { Component } from 'react';
13845 import PropTypes from 'prop-types';
13846 import { bindActionCreators } from 'redux';
13847 import { connect } from 'react-redux';
13848 import * as actions from './redux/actions';
13849 import { Card, Container, Form, Grid, Button, Table, Avatar } from 'tabler-react';
13850 import { Spin, Pagination, Modal } from 'antd';
13851 import { getImageUrl } from '../../../../../utils';
13852 import placeholder from '../../../../../images/placeholder.jpg';
13853 import axios from 'axios';

```

```

13853 import moment from 'moment';
13854
13855 export class RegistrarSetSubmissionDeadline extends Component {
13856   static propTypes = {
13857     app: PropTypes.object.isRequired,
13858     actions: PropTypes.object.isRequired,
13859   };
13860
13861   constructor(props) {
13862     super(props);
13863     this.state = {
13864       isLoading: false,
13865       keyword: '',
13866       page: 1,
13867       pageSize: 10,
13868       numOfPagaes: 1,
13869       data: [],
13870       selectedKey: -1,
13871       selectedDeadlineID: -1,
13872       errors: {},
13873       showModal: false,
13874       modalLoading: false,
13875       teacher: '',
13876       date: new Date(),
13877       deadline: '',
13878     };
13879     this.onChange = this.onChange.bind(this);
13880     this.onDateChange = this.onDateChange.bind(this);
13881     this.removeDeadline = this.removeDeadline.bind(this);
13882     this.showDeleteModal = this.showDeleteModal.bind(this);
13883   }
13884
13885   removeDeadline = () => {
13886     this.props.actions.removeDeadline({ deadlineID: this.state.
13887       selectedDeadlineID });
13888   }
13889
13890   showDeleteModal = () => {
13891     Modal.confirm({
13892       title: 'Are you sure you want to remove the deadline?',
13893       okText: 'Remove',
13894       okType: 'danger',
13895       cancelText: 'Cancel',
13896       onOk: () => {
13897         this.removeDeadline();
13898       },
13899       onCancel: () => {
1400         console.log('closed');
1401       },
1402     });
1403   }
1404
1405   onDateChange = event => {
1406     this.setState({ date: event });
1407   }
1408
1409   setDeadline = () => {
1410     if (this.state.selectedKey == 0) {
1411       let { date } = this.state;
1412       date.setHours(0);
1413       date.setMinutes(0);
1414       date.setSeconds(0);
1415       this.props.actions.setDeadlineAll({ deadline: date.toISOString()
1416         });
1417     } else {
1418       let { date } = this.state;
1419       date.setHours(0);
1420       date.setMinutes(0);
1421       date.setSeconds(0);
1422       this.props.actions.setDeadline({
1423         teacherID: this.state.selectedKey,
1424         deadline: date.toISOString(),
1425       });
1426     }
1427   }

```

```

13924     };
13925
13926     componentDidMount() {
13927       this.setState({ isLoading: true }, async () => {
13928         axios
13929           .post('api/registrar/getallteachers', {
13930             page: this.state.page,
13931             pageSize: this.state.pageSize,
13932             keyword: this.state.keyword,
13933           })
13934           .then(res => {
13935             this.setState({ isLoading: false }, () => {
13936               this.setState({ numOfPages: res.data.numOfPages, data: res.
13937                 data.accountList });
13938             });
13939           });
13940     }
13941
13942     paginate = page => {
13943       this.setState({ isLoading: true, page }, () => {
13944         axios
13945           .post('api/registrar/getallteachers', {
13946             page,
13947             pageSize: this.state.pageSize,
13948             keyword: this.state.keyword,
13949           })
13950           .then(res => {
13951             this.setState({ isLoading: false }, () => {
13952               this.setState({ numOfPages: res.data.numOfPages, data: res.
13953                 data.accountList });
13954             });
13955           });
13956     );
13957
13958     onChange = e => {
13959       this.setState({ isLoading: true, page: 1, [e.target.name]: e.target
13960         .value }, () => {
13961         axios
13962           .post('api/registrar/getallteachers', {
13963             page: this.state.page,
13964             pageSize: this.state.pageSize,
13965             keyword: this.state.keyword,
13966           })
13967           .then(res => {
13968             this.setState({ isLoading: false }, () => {
13969               this.setState({ numOfPages: res.data.numOfPages, data: res.
13970                 data.accountList });
13971             });
13972           .catch(err => {
13973             this.setState({ isLoading: false }, () => {
13974               this.setState({ numOfPages: 1, data: [] });
13975             });
13976           });
13977     );
13978
13979     render() {
13980       const displayDate = date => {
13981         const disp = new Date(date);
13982         return disp.toDateString();
13983       };
13984       const DisplayData = [];
13985       for (const [index, value] of this.state.data.entries()) {
13986         DisplayData.push(
13987           <Table.Row>
13988             <Table.Col className="w-1">
13989               <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
13990                 getImageUrl(value.imageUrl)} />
13991             </Table.Col>

```

```

13991 <Table.Col>{value.name}</Table.Col>
13992 <Table.Col>{value.email}</Table.Col>
13993 <Table.Col>
13994   <Button
13995     size="sm"
13996     icon="edit"
13997     color="primary"
13998     onClick={() =>
13999       this.setState({
14000         selectedKey: value.teacherID,
14001         teacher: value.name,
14002         showModal: true,
14003         deadline: value.deadline ? value.deadline : 'NOT SET'
14004       ,
14005         date: new Date(),
14006         selectedDeadlineID: value.deadlineID,
14007       })
14008     }
14009   >
14010     Set deadline
14011   </Button>
14012 </Table.Col>
14013 </Table.Row>,
14014 );
14015 return (
14016   <Container>
14017     <div className="app-registrar-set-submission-deadline card">
14018       <Modal
14019         title={'Set deadline for ${this.state.teacher}'}
14020         visible={this.state.showModal}
14021         onOk={this.setDeadline}
14022         onCancel={() => {
14023           this.setState({ selectedKey: -1, showModal: false,
14024             teacher: '' });
14025         }}
14026         okText="Set deadline"
14027         confirmLoading={this.props.app.showLoading}
14028         cancelText="Close"
14029       >
14030         <Spin spinning={this.props.app.showLoading}>
14031           <Container>
14032             <Grid.Row>
14033               <Grid.Col sm={12} md={12}>
14034                 <Form.Group>
14035                   {this.state.deadline == 'all' ? (
14036                     ,
14037                   ) : (
14038                     <h4>
14039                       Current deadline set:{' '}
14040                         <b>
14041                           {this.state.deadline == 'NOT SET'
14042                             ? 'NOT SET'
14043                             : displayDate(this.state.deadline)}
14044                         </b>
14045                     </h4>
14046                   <Form.Label>Deadline</Form.Label>
14047                   <Form.DatePicker
14048                     defaultDate={new Date()}
14049                     format="mm/dd/yyyy"
14050                     onChange={this.onDateChange}
14051                     name="deadline"
14052                     monthLabels={[
14053                       'January',
14054                       'February',
14055                       'March',
14056                       'April',
14057                       'May',
14058                       'June',
14059                       'July',
14060                       'August',
14061                       'September',
14062                       'October',
14063                       'November',

```

```

14064          'December',
14065      ]}
14066      maxYear={3000}
14067    ></Form.DatePicker>
14068  </Form.Group>
14069</Grid.Col>
14070 {this.state.selectedDeadlineID != -1 && (
14071   <Grid.Col sm={12} md={12}>
14072     <Form.Group>
14073       <Button color="danger" icon="trash" onClick={this.showDeleteModal}>
14074         Remove {this.state.selectedDeadlineID == 0 ?
14075           'all' : ''} deadline
14076       </Button>
14077     </Form.Group>
14078   </Grid.Col>
14079 )
14080 </Grid.Row>
14081 </Container>
14082 </Spin>
14083 </Modal>
14084 <Card.Header>
14085   <Card.Title>Set Submission Deadline</Card.Title>
14086 </Card.Header>
14087 <Card.Body>
14088   <Grid.Row>
14089     <Grid.Col sm={12} md={8} xs={12}>
14090       <Form.Group>
14091         <Form.Input
14092           icon="search"
14093           placeholder="Search for..."
14094           position="append"
14095           name="keyword"
14096           value={this.state.keyword}
14097           onChange={this.onChange}
14098         />
14099       </Form.Group>
14100     </Grid.Col>
14101     <Grid.Col sm={12} md={4} xs={12}>
14102       <Button
14103         block
14104         color="primary"
14105         icon="edit"
14106         onClick={() => {
14107           this.setState({
14108             showModal: true,
14109             selectedKey: 0,
14110             teacher: 'all teachers',
14111             deadline: 'all',
14112             date: new Date(),
14113             selectedDeadlineID: 0,
14114           });
14115         }}
14116       >
14117         Set to all
14118       </Button>
14119     </Grid.Col>
14120   </Grid.Row>
14121   <Grid.Row>
14122     <Grid.Col sm={12} xs={12} md={12}>
14123       <Spin spinning={this.state.isLoading}>
14124         <Table highlightRowOnHover={true} responsive={true}>
14125           <Table.Header>
14126             <Table.ColHeader colSpan={2}>Name</Table.
14127               ColHeader>
14128             <Table.ColHeader>Email</Table.ColHeader>
14129             <Table.ColHeader>Action</Table.ColHeader>
14130           </Table.Header>
14131           <Table.Body>
14132             {DisplayData.length !== 0 ? (
14133               DisplayData
14134             ) : (
14135               <Table.Row>
```

```

14134             <Table.Col colSpan={4} alignContent="center">
14135                 No data to display.
14136             </Table.Col>
14137         </Table.Row>
14138     )}
14139     </Table.Body>
14140 </Table>
14141 <Pagination
14142   size="large"
14143   current={this.state.page}
14144   pageSize={this.state.pageSize}
14145   total={this.state.pageSize * this.state.numOfPages}
14146   onChange={this.paginate}
14147 />
14148 </Spin>
14149 </Grid.Col>
14150 </Grid.Row>
14151 </Card.Body>
14152 </div>
14153 </Container>
14154 );
14155 }
14156 }
14157 /* istanbul ignore next */
14158 function mapStateToProps(state) {
14159     return {
14160       app: state.app,
14161     };
14162 }
14163 }
14164 /* istanbul ignore next */
14165 function mapDispatchToProps(dispatch) {
14166     return {
14167       actions: bindActionCreators({ ...actions }, dispatch),
14168     };
14169 }
14170 }
14171
14172 export default connect(mapStateToProps, mapDispatchToProps)(
14173   RegistrarSetSubmissionDeadline);
14174 import React, { Component } from 'react';
14175 import PropTypes from 'prop-types';
14176 import { bindActionCreators } from 'redux';
14177 import { connect } from 'react-redux';
14178 import * as actions from './redux/actions';
14179 import axios from 'axios';
14180 import { Pagination, Spin, Tooltip } from 'antd';
14181 import {
14182   Modal,
14183   Popconfirm,
14184   Search,
14185   Breadcrumb,
14186   AutoComplete,
14187   Input,
14188   message,
14189   Descriptions,
14190   Popover,
14191 } from 'antd';
14192 import cn from 'classnames';
14193 import placeholder from '../../images/placeholder.jpg';
14194 import {
14195   Card,
14196   Button,
14197   Grid,
14198   Avatar,
14199   Table,
14200   Form,
14201   Header,
14202   Container,
14203   Text,
14204   Alert,
14205 } from 'tabler-react';
14206 import { Link } from 'react-router-dom';

```

```

14206 import { getImageUrl } from '../../../../../utils';
14207 import ViewEditLog from './ViewEditLog';
14208 const { Option } = AutoComplete;
14209
14210 export class RegistrarSubcomponent extends Component {
14211   static propTypes = {
14212     app: PropTypes.object.isRequired,
14213     actions: PropTypes.object.isRequired,
14214   };
14215
14216   constructor(props) {
14217     super(props);
14218     this.state = {
14219       componentID: 0,
14220       component: '',
14221       subcompName: '',
14222       subcompID: 0,
14223       data: [],
14224       ave: [],
14225       isLoading: true,
14226       quarter: 'Q1',
14227       sectionName: '',
14228       subjectName: '',
14229       schoolYear: '',
14230       subjectCode: '',
14231       showConfirmDelete: false,
14232       selectedDateGiven: new Date(),
14233       selectedDescription: '',
14234       deleteText: '',
14235       subsectID: 0,
14236       locked: false,
14237     };
14238
14239     this.deleteRecord = this.deleteRecord.bind(this);
14240     this.onChange = this.onChange.bind(this);
14241   }
14242
14243   onChange(event) {
14244     this.setState({ [event.target.name]: event.target.value });
14245   }
14246
14247   deleteRecord() {
14248     if (this.state.deleteText != 'DELETE') {
14249       message.error('You must type DELETE to confirm');
14250     } else {
14251       this.props.actions.deleteRecord(
14252         {
14253           classRecordID: this.props.classRecordID,
14254           description: this.state.selectedDescription,
14255           dateGiven: this.state.selectedDateGiven,
14256           subcompID: this.state.subcompID,
14257           componentID: this.state.componentID,
14258           quarter: this.state.quarter,
14259         },
14260         'Registrar',
14261       );
14262     }
14263   }
14264
14265   componentDidMount() {
14266     this.setState({ isLoading: true });
14267     axios
14268       .post('api/registrar/getcomponents', {
14269         classRecordID: this.props.classRecordID,
14270         quarter: this.props.quarter,
14271       })
14272       .then(res => {
14273         this.setState({
14274           sectionName: res.data.sectionName,
14275           subjectName: res.data.subjectName,
14276           schoolYear: res.data.schoolYear,
14277           subjectCode: res.data.subjectCode,
14278         });
14279       axios

```

```

14280     .post('api/registrar/subcompinfo', {
14281       classRecordID: this.props.classRecordID,
14282       componentID: this.props.componentID,
14283       quarter: this.props.quarter,
14284       subcompID: this.props.subcompID,
14285     })
14286     .then(res2 => {
14287       this.setState({
14288         isLoading: false,
14289         componentID: this.props.componentID,
14290         component: res2.data.component,
14291         subcompName: res2.data.subcompName,
14292         subcompID: this.props.subcompID,
14293         data: res2.data.data,
14294         ave: res2.data.ave,
14295         quarter: this.props.quarter,
14296       });
14297     });
14298   );
14299 }
14300
14301 componentWillReceiveProps() {
14302   this.setState({ isLoading: true, showConfirmDelete: false });
14303   axios
14304     .post('api/registrar/getcomponents', {
14305       classRecordID: this.props.classRecordID,
14306       quarter: this.props.quarter,
14307     })
14308     .then(res => {
14309       this.setState({
14310         sectionName: res.data.sectionName,
14311         subjectName: res.data.subjectName,
14312         schoolYear: res.data.schoolYear,
14313         subjectCode: res.data.subjectCode,
14314       });
14315       axios
14316         .post('api/registrar/subcompinfo', {
14317           classRecordID: this.props.classRecordID,
14318           componentID: this.props.componentID,
14319           quarter: this.props.quarter,
14320           subcompID: this.props.subcompID,
14321         })
14322         .then(res2 => {
14323           this.setState({
14324             isLoading: false,
14325             componentID: this.props.componentID,
14326             component: res2.data.component,
14327             subcompName: res2.data.subcompName,
14328             subcompID: this.props.subcompID,
14329             data: res2.data.data,
14330             ave: res2.data.ave,
14331             quarter: this.props.quarter,
14332           });
14333         });
14334       );
14335     );
14336   }
14337   render() {
14338     const { locked } = this.props;
14339     let headerData = [];
14340     let tableData = [];
14341     headerData.push(<Table.ColHeader colSpan={2}>Student Name</Table.
14342       ColHeader>);
14343     if (!this.state.isLoading) {
14344       for (const [index, value] of this.state.data[0].grades.entries())
14345       {
14346         headerData.push(
14347           <Table.ColHeader alignContent="center">
14348             <Popover
14349               content={
14350                 <div>
14351                   <p>
14352                     <b>Description:</b> {value.description}
14353                   </p>
14354                 <p>
```

```

14353             <b>Number of items:</b> {value.total}
14354         </p>
14355     </div>
14356   }
14357 >      Item #{index + 1}
14358 </Popover>
14359 {!locked && (
14360     <span style={{ marginLeft: '15px' }}>
14361       <Button.List>
14362         <Button
14363           icon="edit"
14364           size="sm"
14365           outline
14366           color="primary"
14367           onClick={() => {
14368             this.props.history.push(
14369               `/individualdeliberation/${this.props.id}/
14370                 managegrade/${this.props.classRecordID}/
14371                   quarter/${this.props.quarter}/comp/${this.
14372                     props.componentID}/subcomp/${this.props.
14373                       subcompID}/editrecord/${value.gradeID}`,
14374                     );
14375               })
14376             ></Button>
14377             <Button
14378               icon="trash"
14379               size="sm"
14380               color="danger"
14381               onClick={() => {
14382                 this.setState({
14383                   deleteText: '',
14384                     showConfirmDelete: true,
14385                     selectedDescription: value.description,
14386                     selectedDateGiven: value.dateGiven,
14387                   });
14388               })
14389             ></Button>
14390           </Button.List>
14391         </span>
14392       )
14393     );
14394   headerData.push(<Table.ColHeader alignContent="center">Percentage
14395     Score</Table.ColHeader>);
14396   for (const [index, value] of this.state.data.entries()) {
14397     let name = value.name;
14398     let imageUrl = value.imageUrl;
14399     let ps = value.ps;
14400     let tempRow = [];
14401     for (const [index2, value2] of value.grades.entries()) {
14402       tempRow.push(
14403         <Table.Col alignContent="center">
14404           <Popover
14405             content={
14406               <div>
14407                 <p>
14408                   <b>Description:</b> {value2.description}
14409                 </p>
14410                 <p>
14411                   <b>Number of items:</b> {value2.total}
14412                 </p>
14413               </div>
14414             }
14415           >
14416             {value2.attendance == 'A'
14417               ? 'Absent'
14418               : value2.attendance == 'E'
14419               ? 'Excused'
14420               : `${value2.score}/${value2.total}`}
14421           </Popover>

```

```

14421             </Table.Col>,
14422         );
14423     }
14424     tempRow.push(
14425         <Table.Col alignContent="center">
14426             {ps == -1 ? 'Not yet available' : Number(Math.round(ps + ,
14427                 e2) + 'e-2')}
14428         </Table.Col>,
14429     );
14430     tableData.push(
14431         <Table.Row>
14432             <Table.Col className="w-1">
14433                 <Avatar imageUrl={imageUrl == 'NA' ? placeholder :
14434                     getImageUrl(imageUrl)} />
14435             </Table.Col>
14436             <Table.Col>{name}</Table.Col>
14437             {tempRow}
14438         </Table.Row>,
14439     );
14440 }
14441 return (
14442     <div className="app-teacher-subcomponent my-3 my-md-5">
14443         <Container>
14444             <Modal
14445                 title="Delete Record"
14446                 visible={this.state.showConfirmDelete}
14447                 onOk={this.deleteRecord}
14448                 onCancel={() =>
14449                     this.setState({
14450                         showConfirmDelete: false,
14451                         selectedDateGiven: 0,
14452                         selectedDescription: '',
14453                     })
14454                 }
14455                 okText="Delete"
14456                 confirmLoading={this.props.app.showLoading}
14457                 cancelText="Cancel"
14458             >
14459                 <Container>
14460                     <Grid.Row>
14461                         <Grid.Col sm={12} xs={12} md={12} lg={12}>
14462                             <Header.H5>
14463                                 Are you sure you want to delete this record
14464                                     Deleting this will delete ALL
14465                                         STUDENT RECORDS. Type 'DELETE' to confirm.
14466                             </Header.H5>
14467                         </Grid.Col>
14468                     </Grid.Row>
14469                     <Grid.Row>
14470                         <Grid.Col sm={12} xs={12} md={12} lg={12}>
14471                             <Form.Group>
14472                                 <Form.Input
14473                                     autoComplete="off"
14474                                     value={this.state.deleteText}
14475                                     name="deleteText"
14476                                     onChange={this.onChange}
14477                                     placeholder="Type DELETE"
14478                                 />
14479                             </Form.Group>
14480                         </Grid.Col>
14481                     </Grid.Row>
14482                 </Container>
14483             </Modal>
14484             <Grid.Row>
14485                 <Card>
14486                     <Card.Body>
14487                         {this.state.isLoading ? (
14488                             <Spin spinning={true}></Spin>
14489                         ) : (
14490                             <div>
14491                                 <Card.Title>
14492                                     <Breadcrumb>
14493                                         <Breadcrumb.Item>Subjects</Breadcrumb.Item>

```

```

14492 <Breadcrumb.Item>View Subject Load</Breadcrumb.
14493   Item>
14494   <Breadcrumb.Item>Manage Grades</Breadcrumb.Item>
14495   <Breadcrumb.Item>
14496     {this.state.sectionName} - {this.state.
14497       subjectName}
14498   </Breadcrumb.Item>
14499   <Breadcrumb.Item>{this.state.component}</
14500     Breadcrumb.Item>
14501   </Breadcrumb>
14502 </Card.Title>
14503 <Grid.Row>
14504   <Grid.Col sm={12} xs={12} md={8}>
14505     <Card.Title>
14506       <Header.H3>
14507         {this.state.component} - {this.state.
14508           subcompName}
14509       </Header.H3>
14510     </Card.Title>
14511     <Text.Small>
14512       {this.state.sectionName} {this.props.
14513         quarter} S.Y.{' '}
14514       {this.state.schoolYear}
14515     </Text.Small>
14516   </Card.Title>
14517   <Grid.Col sm={12} xs={12} md={4}>
14518     <Link
14519       to={`/individualdeliberation/${this.props.id}
14520         `/managegrade/${this.props.classRecordID}
14521         `/quarter/${this.props.quarter}/comp/${
14522           this.props.componentID}/subcomp/${this.
14523             props.subcompID}/addrecord`}
14524   >
14525     {!locked && (
14526       <Button.List align="right">
14527         <Button style={{ margin: '20px' }} icon="
14528           plus" color="primary">
14529             Add New Record
14530           </Button>
14531         </Button.List>
14532       )
14533     }
14534   </Link>
14535 </Grid.Col>
14536 </Grid.Row>
14537 </div>
14538 )
14539 </Card.Body>
14540 {this.state.isLoading ? (
14541   <Spin spinning={true}></Spin>
14542 ) : (
14543   <Card.Body>
14544     <Grid.Row>
14545       <Grid.Col sm={12} xs={12} md={12}>
14546         <Descriptions
14547           style={{ marginBottom: '15px', marginTop: '15px
14548             ' }}
14549             bordered
14550             title="Subject Load Information"
14551           >
14552             <Descriptions.Item span={3} label="Section Name
14553               ">
14554               {this.state.sectionName}
14555             </Descriptions.Item>
14556             <Descriptions.Item span={3} label="Subject Name
14557               ">
14558               {this.state.subjectName}
14559             </Descriptions.Item>
14560             <Descriptions.Item span={3} label="Number of

```

```

        Students">
    {tableData.length}
  </Descriptions.Item>
  <Descriptions.Item span={3} label="Number of
    items">
    {this.state.data[0].grades.length}
  </Descriptions.Item>
  <Descriptions.Item span={3} label="Total Number
    of items">
    {this.state.data[0].grades != 0
      ? this.state.data[0].grades.reduce((sum,
        val) => {
          let tempArr = JSON.parse(JSON.stringify
            (sum));
          tempArr.total = sum.total + val.total;
          return tempArr;
        }).total
      : 0}
    </Descriptions.Item>
  </Descriptions>
</Grid.Col>
</Grid.Row>
<Grid.Row>
  <Grid.Col sm={12} xs={12} md={12}>
    <Table highlightRowOnHover={true} responsive={true}>
      <Table.Header>{headerData}</Table.Header>
      <Table.Body>{tableData}</Table.Body>
    </Table>
  </Grid.Col>
</Grid.Row>
</Card.Body>
)}
<Card.Footer>
  <Button.List align="right">
    {((this.props.app.auth.user.position == 2 ||
      this.props.app.auth.user.position == 3) && (
        <ViewEditLog
          classRecordID={this.props.classRecordID}
          quarter={this.state.quarter}
          position="Registrar"
        />
      ))}
    </Button.List>
  </Card.Footer>
</Grid.Row>
</Container>
</div>
);
}
}

/* istanbul ignore next */
function mapStateToProps(state) {
  return {
    app: state.app,
  };
}

/* istanbul ignore next */
function mapDispatchToProps(dispatch) {
  return {
    actions: bindActionCreators({ ...actions }, dispatch),
  };
}

export default connect(mapStateToProps, mapDispatchToProps)(
  RegistrarSubcomponent);
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { bindActionCreators } from 'redux';

```

```

14615 import { connect } from 'react-redux';
14616 import * as actions from './redux/actions';
14617 import axios from 'axios';
14618 import { Pagination, Spin, Tooltip } from 'antd';
14619 import {
14620   Modal,
14621   Popconfirm,
14622   Search,
14623   Breadcrumb,
14624   AutoComplete,
14625   Input,
14626   message,
14627   Descriptions,
14628 } from 'antd';
14629 import cn from 'classnames';
14630 import placeholder from '../../images/placeholder.jpg';
14631 import {
14632   Card,
14633   Button,
14634   Grid,
14635   Avatar,
14636   Table,
14637   Form,
14638   Header,
14639   Container,
14640   Text,
14641   Alert,
14642   Tag,
14643   Badge,
14644 } from 'tabler-react';
14645 import { Link } from 'react-router-dom';
14646 import { getImageUrl } from '../../utils';
14647 import ViewEditLog from './ViewEditLog';
14648 const { Option } = AutoComplete;
14649
14650 export class RegistrarSummaryReport extends Component {
14651   static propTypes = {
14652     app: PropTypes.object.isRequired,
14653     actions: PropTypes.object.isRequired,
14654   };
14655
14656   constructor(props) {
14657     super(props);
14658     this.state = {
14659       isLoading: true,
14660       subjectName: '',
14661       subjectCode: '',
14662       sectionName: '',
14663       schoolYearID: 0,
14664       schoolYear: '',
14665       data: [],
14666       quarter: 'Q1',
14667       subsectID: 0,
14668       transmutation: '50',
14669       trans: '50',
14670       subjectType: '',
14671       locked: false,
14672     };
14673
14674     this.changeTransmutation = this.changeTransmutation.bind(this);
14675   }
14676
14677   changeTransmutation() {
14678     this.props.actions.changeTransmutation(
14679       {
14680         classRecordID: this.props.classRecordID,
14681         quarter: this.state.quarter,
14682         transmutation: this.state.transmutation,
14683       },
14684       'Registrar',
14685     );
14686   }
14687
14688   componentDidMount() {
14689     this.setState({ isLoading: true });

```

```

14690     axios
14691       .post('api/registrar/getquartersummary', {
14692         classRecordID: this.props.classRecordID,
14693         quarter: this.props.quarter,
14694       })
14695       .then(res => {
14696         this.setState({
14697           isLoading: false,
14698           subjectName: res.data.subjectName,
14699           subjectCode: res.data.subjectCode,
14700           sectionName: res.data.sectionName,
14701           schoolYearID: res.data.schoolYearID,
14702           schoolYear: res.data.schoolYear,
14703           data: res.data.data,
14704           quarter: this.props.quarter,
14705           classRecordID: this.props.classRecordID,
14706           transmutation: res.data.transmutation,
14707           trans: res.data.transmutation,
14708         });
14709       });
14710     }
14711   }
14712   componentWillReceiveProps() {
14713     this.setState({ isLoading: true });
14714     axios
14715       .post('api/registrar/getquartersummary', {
14716         classRecordID: this.props.classRecordID,
14717         quarter: this.props.quarter,
14718       })
14719       .then(res => {
14720         this.setState({
14721           isLoading: false,
14722           subjectName: res.data.subjectName,
14723           subjectCode: res.data.subjectCode,
14724           sectionName: res.data.sectionName,
14725           schoolYearID: res.data.schoolYearID,
14726           schoolYear: res.data.schoolYear,
14727           data: res.data.data,
14728           quarter: this.props.quarter,
14729           classRecordID: this.props.classRecordID,
14730           transmutation: res.data.transmutation,
14731           trans: res.data.transmutation,
14732         });
14733       });
14734     }
14735   }
14736   render() {
14737     const { locked } = this.props;
14738     let displayData = [];
14739     for (const [index, value] of this.state.data.entries()) {
14740       displayData.push(
14741         <Table.Row>
14742           <Table.Col className="w-1">
14743             <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getImageUrl(value.imageUrl)} />
14744           </Table.Col>
14745           <Table.Col>{value.name}</Table.Col>
14746           {/* <Table.Col alignContent="center">
14747             {value.faWS == -1 ? 'Not yet available' : Number(Math.round
14748               (value.faWS + 'e2') + 'e-2')}
14749           </Table.Col>
14750           <Table.Col alignContent="center">
14751             {value.wwWS == -1 ? 'Not yet available' : Number(Math.round
14752               (value.wwWS + 'e2') + 'e-2')}
14753           </Table.Col>
14754           <Table.Col alignContent="center">
14755             {value.ptWS == -1 ? 'Not yet available' : Number(Math.round
14756               (value.ptWS + 'e2') + 'e-2')}
14757           </Table.Col> */
14758           <Table.Col alignContent="center">
```

```

14759
14760      <b>
14761          {value.actualGrade == -1
14762              ? 'Not yet available'
14763              : Number(Math.round(value.actualGrade + 'e2') + 'e-2')}
14764      </b>
14765  </Table.Col>
14766  <Table.Col alignContent="center">
14767      {value.transmutedGrade50 == -1
14768          ? 'Not yet available'
14769          : Number(Math.round(value.transmutedGrade50 + 'e2') + 'e
14770              -2')}
14771  </Table.Col>
14772  <Table.Col alignContent="center">
14773      {value.transmutedGrade55 == -1
14774          ? 'Not yet available'
14775          : Number(Math.round(value.transmutedGrade55 + 'e2') + 'e
14776              -2')}
14777  </Table.Col>
14778  <Table.Col alignContent="center">
14779      {value.transmutedGrade60 == -1
14780          ? 'Not yet available'
14781          : Number(Math.round(value.transmutedGrade60 + 'e2') + 'e
14782              -2')}
14783      <b>
14784          <Text color={value.finalGrade < 75 ? 'red' : 'black'}>
14785              <span className={'status-icon bg-$\{value.finalGrade <
14786                  75 ? 'red' : 'green'\}' />
14787          {value.finalGrade == -1
14788              ? 'Not yet available'
14789              : Number(Math.round(value.finalGrade + 'e2') + 'e-2')
14790      }
14791      </Table.Col>
14792  );
14793  return (
14794      <div className="app-teacher-summary-per-quarter my-3 my-md-5">
14795          <Card>
14796              <Card.Body>
14797                  {this.state.isLoading ? (
14798                      <Spin spinning={true}></Spin>
14799                  ) : (
14800                      <div>
14801                          <Card.Title>
14802                              <Grid.Row>
14803                                  <Grid.Col xs={12} sm={12} md={6}>
14804                                      <Breadcrumb>
14805                                          <Breadcrumb.Item>Subjects</Breadcrumb.Item>
14806                                          <Breadcrumb.Item>View Subject Load</Breadcrumb.
14807                                              Item>
14808                                          <Breadcrumb.Item>Summary Report</Breadcrumb.
14809                                              Item>
14810                                          <Breadcrumb.Item>
14811                                              {this.state.sectionName} - {this.state.
14812                                              subjectName}
14813                                              </Breadcrumb.Item>
14814                                      </Grid.Col>
14815                                  <Grid.Col xs={12} sm={12} md={6}></Grid.Col>
14816                              </Grid.Row>
14817          </Card.Title>
14818          <Card.Title>
14819              <Header.H3>
14820                  {this.state.subjectCode} - {this.state.subjectName}
14821              </Header.H3>
14822          </Card.Title>
14823              <Text.Small>S.Y. {this.state.schoolYear}</Text.Small>
              <Grid.Row>

```

```

14824 <Grid.Col xs={12} sm={12} md={3}></Grid.Col>
14825 <Grid.Col xs={12} sm={12} md={3}></Grid.Col>
14826 <Grid.Col xs={12} sm={12} md={3}>
14827   {!locked && (
14828     <Form.Select
14829       onChange={e => {
14830         this.setState({ transmutation: e.target.
14831           value });
14832       }}
14833       value={this.state.transmutation}
14834     >
14835       <option value="50">50% (Yellow)</option>
14836       <option value="55">55% (Orange)</option>
14837       <option value="60">60% (Green)</option>
14838     </Form.Select>
14839   )
14840 </Grid.Col>
14841 <Grid.Col xs={12} sm={12} md={3}>
14842   {!locked && (
14843     <Button
14844       color="primary"
14845       block
14846       onClick={() => {
14847         this.changeTransmutation();
14848       }}
14849     >
14850       Change Transmutation
14851     </Button>
14852   )
14853 </Grid.Row>
14854 <Grid.Row>
14855 <Grid.Col xs={12} sm={12} md={6}>
14856   <Descriptions
14857     style={{ marginBottom: '15px', marginTop: '15px
14858       ' }}
14859     bordered
14860     title="Frequency Distribution"
14861   >
14862     <Descriptions.Item span={3} label="95-100">
14863       {this.state.data.length != 0 &&
14864         parseFloat(
14865           this.state.data.reduce((sum, val) => {
14866             const tempobj = JSON.parse(JSON.stringify
14867               (sum));
14868             tempobj.transmutedGrade60 =
14869               sum.transmutedGrade60 + val.
14870                 transmutedGrade60;
14871             return tempobj;
14872           }) .transmutedGrade60,
14873         ) /
14874           this.state.data.length !=
14875             -1
14876             ? this.state.data.filter(
14877               val => val.finalGrade >= 95 && val.
14878                 finalGrade <= 100,
14879               ).length
14880             : 'Not yet available'
14881           </Descriptions.Item>
14882         <Descriptions.Item span={3} label="90-94">
14883           {this.state.data.length != 0 &&
14884             parseFloat(
14885               this.state.data.reduce((sum, val) => {
14886                 const tempobj = JSON.parse(JSON.stringify
14887                   (sum));
14888                 tempobj.transmutedGrade60 =
14889                   sum.transmutedGrade60 + val.
14890                     transmutedGrade60;
14891                   return tempobj;
14892                 }) .transmutedGrade60,
14893               ) /
14894                 this.state.data.length !=
14895                   -1

```

```

14890           ? this.state.data.filter(
14891             val => val.finalGrade >= 90 && val.
14892               finalGrade < 95,
14893               ).length
14894               : 'Not yet available'}
14895             
```

```

14896             <Descriptions.Item span={3} label="85-89">
14897               {this.state.data.length != 0 &&
14898                 parseFloat(
14899                   this.state.data.reduce((sum, val) => {
14900                     const tempobj = JSON.parse(JSON.stringify
14901                         (sum));
14902                     tempobj.transmutedGrade60 =
14903                         sum.transmutedGrade60 + val.
14904                             transmutedGrade60;
14905                             return tempobj;
14906               }) .transmutedGrade60 ,
14907               ) /
14908                 this.state.data.length !=
14909                   -1
14910                   ? this.state.data.filter(
14911                     val => val.finalGrade >= 85 && val.
14912                         finalGrade < 90,
14913                         ).length
14914                         : 'Not yet available'}
14915             
```

```

14916             <Descriptions.Item span={3} label="80-84">
14917               {this.state.data.length != 0 &&
14918                 parseFloat(
14919                   this.state.data.reduce((sum, val) => {
14920                     const tempobj = JSON.parse(JSON.stringify
14921                         (sum));
14922                     tempobj.transmutedGrade60 =
14923                         sum.transmutedGrade60 + val.
14924                             transmutedGrade60;
14925                             return tempobj;
14926               }) .transmutedGrade60 ,
14927               ) /
14928                 this.state.data.length !=
14929                   -1
14930                   ? this.state.data.filter(
14931                     val => val.finalGrade >= 80 && val.
14932                         finalGrade < 85,
14933                         ).length
14934                         : 'Not yet available'}
14935             
```

```

14936             <Descriptions.Item span={3} label="75-79">
14937               {this.state.data.length != 0 &&
14938                 parseFloat(
14939                   this.state.data.reduce((sum, val) => {
14940                     const tempobj = JSON.parse(JSON.stringify
14941                         (sum));
14942                     tempobj.transmutedGrade60 =
14943                         sum.transmutedGrade60 + val.
14944                             transmutedGrade60;
14945                             return tempobj;
14946               }) .transmutedGrade60 ,
14947               ) /
14948                 this.state.data.length !=
14949                   -1
14950                   ? this.state.data.filter(
14951                     val => val.finalGrade >= 75 && val.
                           finalGrade < 80,
                           ).length
                           : 'Not yet available'}
                           
```

```

14952             <Descriptions.Item span={3} label="Below 75">
14953               {this.state.data.length != 0 &&
14954                 parseFloat(
14955                   this.state.data.reduce((sum, val) => {
14956                     const tempobj = JSON.parse(JSON.stringify
14957                         (sum));
14958                     tempobj.transmutedGrade60 =
14959                         sum.transmutedGrade60 + val.
14960                             transmutedGrade60;
14961                             return tempobj;
14962               }) .transmutedGrade60 ,
14963               ) /
14964                 this.state.data.length !=
14965                   -1
14966                   ? this.state.data.filter(
14967                     val => val.finalGrade < 75,
14968                         ).length
14969                         : 'Not yet available'}
                           
```

```

14952             sum.transmutedGrade60 + val.
14953                 transmutedGrade60;
14954             return tempobj;
14955         }).transmutedGrade60,
14956     ) /
14957         this.state.data.length !=
14958             -1
14959             ? this.state.data.filter(val => val.
14960                 finalGrade < 75).length
14961             : 'Not yet available'
14962         </Descriptions.Item>
14963     </Descriptions>
14964 </Grid.Col>
14965 <Grid.Col xs={12} sm={12} md={6}>
14966     <Descriptions
14967         style={{ marginBottom: '15px', marginTop: '15px
14968             ,
14969             }}>
14970         bordered
14971         title="Average"
14972     >
14973         <Descriptions.Item span={3} label="Actual Grade
14974             ">
14975             {this.state.data.length != 0 &&
14976             parseFloat(
14977                 this.state.data.reduce((sum, val) => {
14978                     const tempobj = JSON.parse(JSON.stringify
14979                         (sum));
14980                     tempobj.actualGrade = sum.actualGrade +
14981                         val.actualGrade;
14982                     return tempobj;
14983                 }).actualGrade,
14984             ) /
14985             this.state.data.length
14986             -1
14987             ? parseFloat(
14988                 this.state.data.reduce((sum, val) => {
14989                     const tempobj = JSON.parse(JSON.
14990                         stringify(sum));
14991                     tempobj.actualGrade = sum.actualGrade +
14992                         val.actualGrade;
14993                     return tempobj;
14994                 }).actualGrade,
14995             ) / this.state.data.length
14996             : 'Not yet available'
14997         </Descriptions.Item>
14998         <Descriptions.Item span={3} label="Transmuted
14999             Grade (50%)">
15000             {this.state.data.length != 0 &&
15001             parseFloat(
15002                 this.state.data.reduce((sum, val) => {
15003                     const tempobj = JSON.parse(JSON.stringify
15004                         (sum));
15005                     tempobj.transmutedGrade50 =
15006                         sum.transmutedGrade50 + val.
15007                             transmutedGrade50;
15008                     return tempobj;
15009                 }).transmutedGrade50,
15010             ) /
15011             this.state.data.length
15012             -1
15013             ? parseFloat(
15014                 this.state.data.reduce((sum, val) => {
15015                     const tempobj = JSON.parse(JSON.
15016                         stringify(sum));
15017                     tempobj.transmutedGrade50 =
15018                         sum.transmutedGrade50 + val.
15019                             transmutedGrade50;
15020                     return tempobj;
15021                 }).transmutedGrade50,
15022             ) / this.state.data.length
15023             : 'Not yet available'
15024         </Descriptions.Item>
15025         <Descriptions.Item span={3} label="Transmuted

```

```

Grade (55%)">
15012     {this.state.data.length != 0 &&
15013       parseFloat(
15014         this.state.data.reduce((sum, val) => {
15015           const tempobj = JSON.parse(JSON.stringify(
15016             (sum)));
15017             tempobj.transmutedGrade55 =
15018               sum.transmutedGrade55 + val.
15019                 transmutedGrade55;
15020               return tempobj;
15021             )).transmutedGrade55,
15022           ) /
15023             this.state.data.length !=
15024               -1
15025             ? parseFloat(
15026               this.state.data.reduce((sum, val) => {
15027                 const tempobj = JSON.parse(JSON.
15028                   stringify(sum));
15029                     tempobj.transmutedGrade55 =
15030                       sum.transmutedGrade55 + val.
15031                         transmutedGrade55;
15032                           return tempobj;
15033                         )).transmutedGrade55,
15034                           ) / this.state.data.length
15035                             : 'Not yet available'}
15036 </Descriptions.Item>
15037 <Descriptions.Item span={3} label="Transmuted
15038   Grade (60%)">
15039     {this.state.data.length != 0 &&
15040       parseFloat(
15041         this.state.data.reduce((sum, val) => {
15042           const tempobj = JSON.parse(JSON.stringify(
15043             (sum)));
15044               tempobj.transmutedGrade60 =
15045                 sum.transmutedGrade60 + val.
15046                   transmutedGrade60;
15047                     return tempobj;
15048                   )).transmutedGrade60,
15049                     ) /
15050           this.state.data.length !=
15051             -1
15052             ? parseFloat(
15053               this.state.data.reduce((sum, val) => {
15054                 const tempobj = JSON.parse(JSON.
15055                   stringify(sum));
15056                     tempobj.transmutedGrade60 =
15057                       sum.transmutedGrade60 + val.
15058                         transmutedGrade60;
15059                           return tempobj;
15060                         )).transmutedGrade60,
15061                           ) / this.state.data.length
15062                             : 'Not yet available'}
15063 </Descriptions.Item>
15064 </Descriptions>
15065 </Grid.Col>
15066 </Grid.Row>
15067 </Card.Title>
15068 </div>
15069   )
15070 </Card.Body>
<Card.Body>
  <Grid.Row>
    <Grid.Col xs={12} md={12} sm={12}>
      <Table responsive={true} highlightRowOnHover={true}>
        <Table.Header>
          <Table.ColHeader colSpan={2}>Student Name</Table.
            ColHeader>
          {/* <Table.ColHeader alignContent="center">
            Formative Assessment</Table.ColHeader>
          <Table.ColHeader alignContent="center">Written
            Works</Table.ColHeader>
          <Table.ColHeader alignContent="center">Performance
            Tasks</Table.ColHeader>
```

```

15071             <Table.ColHeader alignContent="center">Quarterly
15072                         Assessment</Table.ColHeader> *}
15073             <Table.ColHeader alignContent="center">Actual Grade
15074                         </Table.ColHeader>
15075             <Table.ColHeader alignContent="center">
15076                         Transmuted Grade (50%) <Tag color="yellow">Yellow
15077                         </Tag>
15078             </Table.ColHeader>
15079             <Table.ColHeader alignContent="center">
15080                         Transmuted Grade (55%) <Tag color="orange">Orange
15081                         </Tag>
15082             </Table.ColHeader>
15083             <Table.ColHeader alignContent="center">Final Grade
15084                         </Table.ColHeader>
15085             </Table.Header>
15086             <Table.Body>{displayData}</Table.Body>
15087             </Table>
15088             </Grid.Col>
15089             </Grid.Row>
15090             <Card.Body>
15091             <Button.List align="right">
15092                 {(
15093                     this.props.app.auth.user.position == 2 ||
15094                     this.props.app.auth.user.position == 3) && (
15095                     <ViewEditLog
15096                         classRecordID={this.props.classRecordID}
15097                         quarter={this.state.quarter}
15098                         position="Registrar"
15099                         />
15100                     )}
15101             </Button.List>
15102             <Card.Footer>
15103             </Card>
15104         );
15105     }
15106
15107 /* istanbul ignore next */
15108 function mapStateToProps(state) {
15109     return {
15110         app: state.app,
15111     };
15112 }
15113
15114 /* istanbul ignore next */
15115 function mapDispatchToProps(dispatch) {
15116     return {
15117         actions: bindActionCreators({ ...actions }, dispatch),
15118     };
15119 }
15120
15121 export default connect(mapStateToProps, mapDispatchToProps)(
    RegistrarSummaryReport);
15122 import React, { Component } from 'react';
15123 import PropTypes from 'prop-types';
15124 import { bindActionCreators } from 'redux';
15125 import { connect } from 'react-redux';
15126 import StudentTooltip from './StudentTooltip';
15127 import * as actions from './redux/actions';
15128 import { Link } from 'react-router-dom';
15129 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
    from 'tabler-react';
15130 import axios from 'axios';
15131 import { Pagination, Spin, Tooltip } from 'antd';
15132 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
    message } from 'antd';
15133 import placeholder from '../../../../../images/placeholder.jpg';

```

```

15134 import { getImageUrl } from '../../../../../utils';
15135 import { Table as AntdTable } from 'antd';
15136 const { Option } = AutoComplete;
15137
15138 export class RegistrarViewEnrolled extends Component {
15139   static propTypes = {
15140     app: PropTypes.object.isRequired,
15141     actions: PropTypes.object.isRequired,
15142   };
15143
15144   constructor(props) {
15145     super(props);
15146     this.state = {
15147       isLoading: true,
15148       isLoadingTable: true,
15149       data: [],
15150       options: [],
15151       page: 1,
15152       pageSize: 10,
15153       numOfPages: 1,
15154       keyword: '',
15155       selectedKey: -1,
15156       errors: {},
15157       studentID: 0,
15158       gradeLevel: '',
15159       sectionName: '',
15160       schoolYear: '',
15161       schoolYearID: 0,
15162       showModal: false,
15163       subsectList: [],
15164       selectedSubsect: [],
15165       selectedStudSectID: -1,
15166     };
15167
15168   this.handleSearch = this.handleSearch.bind(this);
15169   this.onSelect = this.onSelect.bind(this);
15170   this.enrollStudent = this.enrollStudent.bind(this);
15171   this.unenrollStudent = this.unenrollStudent.bind(this);
15172   this.studentAddSubject = this.studentAddSubject.bind(this);
15173 }
15174
15175   enrollStudent() {
15176     if (this.state.selectedKey == -1) {
15177       message.error('You must select a student');
15178     } else {
15179       this.props.actions.createStudentSection(
15180         {
15181           sectionID: this.props.id,
15182           schoolYearID: this.state.schoolYearID,
15183           studentID: this.state.selectedKey,
15184         },
15185         s => this.setState(s),
15186         this.state.subsectList,
15187       );
15188     }
15189   }
15190
15191   studentAddSubject() {
15192     if (this.state.selectedSubsect.length == 0) {
15193       message.error('You must select a student');
15194     } else {
15195       this.props.actions.addSubjectSectionStudentBulk({
15196         studsectID: this.state.selectedStudSectID,
15197         subsectIDs: this.state.selectedSubsect,
15198       });
15199       this.setState({ showModal: false });
15200     }
15201   }
15202
15203   unenrollStudent() {
15204     this.props.actions.deleteStudentSection({
15205       sectionID: this.props.id,
15206       schoolYearID: this.state.schoolYearID,
15207       studentID: this.state.studentID,

```

```

15208     });
15209 }
15210
15211     componentWillMount() {
15212         this.setState({ isLoadingTable: true });
15213         axios
15214             .post('api/registrar/getcurrentenrolled', { sectionID: this.props
15215                 .id, page: 1, pageSize: 10 })
15216             .then(res => {
15217                 this.setState({ numOfPages: res.data.numOfPages, data: res.data
15218                     .studentList });
15219                 this.setState({ isLoadingTable: false });
15220             })
15221             .catch(err => {
15222                 this.setState({ data: [] });
15223                 this.setState({ isLoadingTable: false });
15224             });
15225     }
15226
15227     componentDidMount() {
15228         axios.get('api/registrar/getsy').then(res => {
15229             axios.post('api/registrar/sectionname', { sectionID: this.props.
15230                 id }).then(res2 => {
15231                     this.setState({ sectionName: res2.data.sectionName });
15232                     this.setState({ schoolYearID: res.data.schoolYearID });
15233                     this.setState({ schoolYear: res.data.schoolYear });
15234                     this.setState({ isLoading: false });
15235                     axios
15236                         .post('api/registrar/getcurrentenrolled', {
15237                             sectionID: this.props.id,
15238                             page: 1,
15239                             pageSize: this.state.pageSize,
15240                         })
15241                         .then(res3 => {
15242                             axios
15243                                 .post('api/registrar/listsubjectsection', { sectionID:
15244                                     this.props.id })
15245                                 .then(res4 => {
15246                                     this.setState({ gradeLevel: res3.data.gradeLevel });
15247                                     this.setState({ numOfPages: res3.data.numOfPages });
15248                                     this.setState({ data: res3.data.studentList });
15249                                     this.setState({ isLoadingTable: false });
15250                                     this.setState({ subsectList: res4.data.subjectList });
15251                                 });
15252                             })
15253                         .catch(err => {
15254                             this.setState({ isLoadingTable: false });
15255                         });
15256                     });
15257                     paginate = page => {
15258                         this.setState({
15259                             page,
15260                         });
15261                         this.setState({ isLoadingTable: true });
15262                         axios
15263                             .post('api/registrar/getcurrentenrolled', {
15264                                 page,
15265                                 pageSize: this.state.pageSize,
15266                                 sectionID: this.props.id,
15267                             })
15268                             .then(res => {
15269                                 this.setState({ isLoadingTable: false });
15270                                 this.setState({
15271                                     numOfPages: res.data.numOfPages,
15272                                     data: res.data.studentList,
15273                                 });
15274                             })
15275                             .catch(err => {
15276                                 this.setState({ isLoadingTable: false });

```

```

15276         });
15277     };
15278
15279     handleSearch(query) {
15280       this.setState({ selectedKey: -1 });
15281       this.setState({
15282         options: [
15283           <Option key={0} text="">
15284             <div>
15285               <Spin spinning={true}></Spin>
15286             </div>
15287           </Option>,
15288         ],
15289       });
15290       axios
15291         .post('api/registrar/searchstudent', { keyword: query })
15292         .then(res => {
15293           if (query == '') {
15294             this.setState({ options: [] });
15295           } else {
15296             let optionData = res.data.accountList.map(data => (
15297               <Option key={data.key} text={data.name}>
15298                 <div>
15299                   <Avatar
15300                     imageURL={data.imageUrl == 'NA' ? placeholder :
15301                       getImageUrl(data.imageUrl)}
15302                     />
15303                     <span style={{ margin: '16px', verticalAlign: 'text-top
15304                         }}>{data.name}</span>
15305                   </div>
15306                 </Option>
15307               )));
15308             this.setState({ options: optionData });
15309           }
15310         .catch(err => {
15311           this.setState({
15312             options: [
15313               <Option key={0} text="">
15314                 <div>No data.</div>
15315               </Option>,
15316             ],
15317           });
15318         }
15319
15320         onSelect(key) {
15321           this.setState({ selectedKey: key });
15322         }
15323
15324         render() {
15325           const { options } = this.state;
15326           const DisplayData = [];
15327           for (const [index, value] of this.state.data.entries()) {
15328             DisplayData.push(
15329               <Table.Row>
15330                 <Table.Col className="w-1">
15331                   <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
15332                       getImageUrl(value.imageUrl)} />
15333                 </Table.Col>
15334                 <Table.Col>
15335                   <Tooltip title={<StudentTooltip id={value.key} />}>{value.
15336                     name}</Tooltip>
15337                 </Table.Col>
15338                 <Table.Col>{value.email}</Table.Col>
15339                 <Table.Col>
15340                   <span style={{ marginLeft: '10px' }}>
15341                     <Popconfirm
15342                       title="Do you want to remove this student?"
15343                       onConfirm={this.unenrollStudent}
15344                       okText="Delete"
15345                       cancelText="Cancel"

```

```

15344      >          <Button
15345          icon="trash"
15346          size="sm"
15347          pill
15348          color="danger"
15349          value={value.key}
15350          onClick={() => {
15351              this.setState({ studentID: value.key });
15352          }}
15353      />
15354      </Popconfirm>
15355      </span>
15356      </Table.Col>
15357      </Table.Row>,
15358  );
15359 }
15360 }
15361 return (
15362     <div className="app-registrar-view-enrolled my-3 my-md-5 card">
15363         <Modal
15364             title="Enroll to subjects"
15365             visible={this.state.showModal}
15366             onOk={this.studentAddSubject}
15367             onCancel={() =>
15368                 this.setState({ showModal: false, selectedSubsect: [] ,
15369                     selectedStudSectID: -1 })
15370             }
15371             okText="Enroll student"
15372             confirmLoading={this.props.app.showLoading}
15373             cancelText="Close"
15374         >
15375             <Container>
15376                 <Grid.Row>
15377                     <Grid.Col sm={12} xs={12} md={12}>
15378                         Do you want to enroll this student to the following
15379                         subjects?
15380                     </Grid.Col>
15381                     <Grid.Col sm={12} xs={12} md={12}>
15382                         <AntdTable
15383                             columns={[
15384                             {
15385                                 title: 'Subject',
15386                                 dataIndex: 'subjectName',
15387                                 render: text => text ,
15388                             },
15389                             {
15390                                 title: 'Teacher',
15391                                 dataIndex: 'teacher',
15392                                 render: text => text ,
15393                             },
15394                             ]}
15395                             rowSelection={{
15396                                 onChange: (selectedRowKeys , selectedRows) => {
15397                                     this.setState({ selectedSubsect: selectedRows });
15398                                 }
15399                             }}
15400                             dataSource={this.state.subsectList}
15401                         />
15402                     </Grid.Col>
15403                 </Container>
15404             </Modal>
15405             <Card.Body>
15406                 {this.state.isLoading ? (
15407                     '',
15408                 ) : (
15409                     <div>
15410                         {' '}
15411                         <Card.Title>
15412                             <Breadcrumb>
15413                                 <Breadcrumb.Item>Sections</Breadcrumb.Item>
15414                                 <Breadcrumb.Item>Manage Students</Breadcrumb.Item>
15415                                 <Breadcrumb.Item>{this.state.sectionName}</Breadcrumb

```

```

        .Item>
    </Breadcrumb>
</Card.Title>
<Card.Title>
    {this.state.sectionName} S.Y. {this.state.schoolYear}
</Card.Title>
</div>
)
<Grid.Row>
    <Grid.Col sm={12} md={12} xs={12}>
        <Grid.Row>
            <Grid.Col sm={12} md={6} xs={12}>
                <AutoComplete
                    style={{ width: '100%', marginBottom: '10px' }}
                    onSearch={this.handleSearch}
                    dataSource={options}
                    onSelect={this.onSelect}
                    optionLabelProp="text"
                >
                    <Input placeholder="Search for students"
                        enterButton />
                </AutoComplete>
            </Grid.Col>
            <Grid.Col sm={6} md={3} xs={6}>
                <Button color="primary" block onClick={this.enrollStudent}>
                    Add to Section
                </Button>
            </Grid.Col>
            <Grid.Col sm={6} md={3} xs={6}>
                <Link to={`/managestudents/viewpastrecords/${this.props.id}}>
                    <Button
                        icon="user"
                        disabled={this.state.gradeLevel == 'N'}
                        color="primary"
                        block
                    >
                        View past records
                    </Button>
                </Link>
            </Grid.Col>
        </Grid.Row>
        <Spin spinning={this.state.isLoadingTable}>
            <Table highlightRowOnHover={true} responsive={true}>
                <Table.Header>
                    <Table.ColHeader colSpan={2}>Student Name</Table.ColHeader>
                    <Table.ColHeader>Email address</Table.ColHeader>
                    <Table.ColHeader>Action</Table.ColHeader>
                </Table.Header>
                <Table.Body>
                    {DisplayData.length == 0 ? (
                        <Table.Row>
                            <Table.Col colSpan={4} alignContent="center">
                                No entries.
                            </Table.Col>
                        </Table.Row>
                    ) : (
                        DisplayData
                    )}
                </Table.Body>
            </Table>
            <Pagination
                size="large"
                current={this.state.page}
                pageSize={this.state.pageSize}
                total={this.state.pageSize * this.state.numOfPages}
                onChange={this.paginate}
            />
        </Spin>
    </Grid.Col>
</Grid.Row>

```

```

15483             </Card.Body>
15484         </div>
15485     );
15486 }
15487 }
15488
15489 /* istanbul ignore next */
15490 function mapStateToProps(state) {
15491     return {
15492         app: state.app,
15493     };
15494 }
15495
15496 /* istanbul ignore next */
15497 function mapDispatchToProps(dispatch) {
15498     return {
15499         actions: bindActionCreators({ ...actions }, dispatch),
15500     };
15501 }
15502
15503 export default connect(mapStateToProps, mapDispatchToProps)(
15504     RegistrarViewEnrolled);
15505 import React, { Component } from 'react';
15506 import PropTypes from 'prop-types';
15507 import { bindActionCreators } from 'redux';
15508 import { connect } from 'react-redux';
15509 import * as actions from './redux/actions';
15510 import axios from 'axios';
15511 import { Spin, Breadcrumb, Pagination } from 'antd';
15512 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
15513     from 'tabler-react';
15514 import placeholder from '../../../../../images/placeholder.jpg';
15515 import { getImageUrl } from '../../../../../utils';
15516
15517     export class RegistrarViewPastRecords extends Component {
15518         static propTypes = {
15519             app: PropTypes.object.isRequired,
15520             actions: PropTypes.object.isRequired,
15521         };
15522
15523         constructor(props) {
15524             super(props);
15525             this.state = {
15526                 isLoading: true,
15527                 isLoadingTable: true,
15528                 schoolYearID: 0,
15529                 sectionName: '',
15530                 schoolYear: '',
15531                 gradeLevel: '',
15532                 pastGradeLevel: '',
15533                 data: [],
15534                 keyword: '',
15535                 page: 1,
15536                 pageSize: 10,
15537                 numOfPages: 1,
15538                 selectedKey: 0,
15539             };
15540
15541             this.onChangeSearch = this.onChangeSearch.bind(this);
15542             this.enrollStudent = this.enrollStudent.bind(this);
15543         }
15544
15545         enrollStudent(studentID) {
15546             axios.get(`api/registrar/getsy`).then(res => {
15547                 this.props.actions.createStudentSection({
15548                     sectionID: this.props.id,
15549                     schoolYearID: res.data.schoolYearID,
15550                     studentID,
15551                 });
15552             });
15553         }
15554
15555         onChangeSearch(event) {

```

```

15554     this.setState({ [event.target.name]: event.target.value });
15555     this.setState({ page: 1 });
15556     this.setState({ isLoadingTable: true });
15557     axios
15558       .post('api/registrar/getpastrecords', {
15559         page: 1,
15560         pageSize: this.state.pageSize,
15561         keyword: event.target.value,
15562         gradeLevel: this.state.gradeLevel,
15563       })
15564       .then(res => {
15565         this.setState({ numOfPages: res.data.numOfPages });
15566         this.setState({ data: res.data.studentData });
15567         this.setState({ isLoadingTable: false });
15568       });
15569     }
15570   }
15571   paginate = page => {
15572     this.setState({ page });
15573     this.setState({ isLoadingTable: true });
15574     axios
15575       .post('api/registrar/getpastrecords', {
15576         page,
15577         pageSize: this.state.pageSize,
15578         keyword: this.state.keyword,
15579       })
15580       .then(res => {
15581         this.setState({ numOfPages: res.data.numOfPages });
15582         this.setState({ data: res.data.studentData });
15583         this.setState({ isLoadingTable: false });
15584       });
15585     }
15586   }
15587   componentWillMount() {
15588     this.setState({ isLoading: true });
15589     axios.get('api/registrar/getpastsy').then(res => {
15590       this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
15591         res.data.schoolYear });
15592       axios.post('api/registrar/sectiongradelevel', { sectionID: this.
15593         props.id }).then(res => {
15594         this.setState({ gradeLevel: res.data.gradeLevel });
15595         axios
15596           .post('api/registrar/getpastgradelevel', { gradeLevel: res.
15597             data.gradeLevel })
15598           .then(res2 => {
15599             this.setState({ pastGradeLevel: res2.data.gradeLevel });
15600             this.setState({ isLoading: false });
15601             axios
15602               .post('api/registrar/getpastrecords', {
15603                 page: this.state.page,
15604                 pageSize: this.state.pageSize,
15605                 keyword: this.state.keyword,
15606                 gradeLevel: res.data.gradeLevel,
15607               })
15608               .then(res3 => {
15609                 axios.post('api/registrar/sectionname', { sectionID:
15610                   this.props.id }).then(res4 => {
15611                     this.setState({ sectionName: res4.data.sectionName })
15612                     ;
15613                     this.setState({ numOfPages: res3.data.numOfPages });
15614                     this.setState({ data: res3.data.studentData });
15615                     this.setState({ isLoadingTable: false });
15616                   });
15617                 })
15618               .catch(err => {
15619                 this.setState({ isLoadingTable: false });
15620               });
15621             });
15622           });
15623         });
15624       });
15625     }

```

```

15621 render() {
15622     const displayGradeLevel = gradeLevel => {
15623         switch (gradeLevel) {
15624             case 'N':
15625                 return 'Nursery';
15626             case 'K1':
15627                 return 'Kinder 1';
15628             case 'K2':
15629                 return 'Kinder 2';
15630             case 'G1':
15631                 return 'Grade 1';
15632             case 'G2':
15633                 return 'Grade 2';
15634             case 'G3':
15635                 return 'Grade 3';
15636             case 'G4':
15637                 return 'Grade 4';
15638             case 'G5':
15639                 return 'Grade 5';
15640             case 'G6':
15641                 return 'Grade 6';
15642             case 'G7':
15643                 return 'Grade 7';
15644             case 'G8':
15645                 return 'Grade 8';
15646             case 'G9':
15647                 return 'Grade 9';
15648             case 'G10':
15649                 return 'Grade 10';
15650             case 'G11':
15651                 return 'Grade 11';
15652             case 'G12':
15653                 return 'Grade 12';
15654             default:
15655                 return '';
15656         }
15657     };
15658     const DisplayData = [];
15659     for (const [index, value] of this.state.data.entries()) {
15660         DisplayData.push(
15661             <Table.Row>
15662                 <Table.Col className="w-1">
15663                     <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getimageUrl(value.imageUrl)} />
15664                 </Table.Col>
15665                 <Table.Col>{value.studentName}</Table.Col>
15666                 <Table.Col>{value.email}</Table.Col>
15667                 <Table.Col>{value.sectionName}</Table.Col>
15668                 <Table.Col alignContent="center">
15669                     <Button
15670                         icon="plus"
15671                         color="primary"
15672                         value="ASD"
15673                         onClick={() => {
15674                             this.enrollStudent(value.studentID);
15675                         }}}
15676                     >
15677                         {'Enroll Student to ${this.state.sectionName}'}
15678                     </Button>
15679                 </Table.Col>
15680             </Table.Row>,
15681         );
15682     }
15683     return (
15684         <Table.Body className="app-registrar-view-past-records my-3 my-md-5 card">
15685             <Card.Body>
15686                 {this.state.isLoading ? (
15687                     '',
15688                 ) : (
15689                     <div>
15690                         <Card.Title>
15691                             <Breadcrumb>
15692                                 <Breadcrumb.Item>Sections </Breadcrumb.Item>
15693                                 <Breadcrumb.Item>Manage Students </Breadcrumb.Item>

```

```

15694             <Breadcrumb.Item>{this.state.sectionName}</Breadcrumb
15695                 .Item>
15696             <Breadcrumb.Item>View Past Records</Breadcrumb.Item>
15697             <Breadcrumb.Item>{'${displayGradeLevel(this.state.
15698                 pastGradeLevel)} S.Y. ${this.state.schoolYear
15699                     }'</Breadcrumb.Item>
15700         </Breadcrumb>
15701     </Card.Title>
15702     <Card.Title>{'All Students from ${displayGradeLevel(
15703         this.state.pastGradeLevel,
15704         )} S.Y. ${this.state.schoolYear}'</Card.Title>
15705     </div>
15706 <Grid.Row>
15707     <Grid.Col sm={12} md={12} xs={12}>
15708         <Grid.Row>
15709             <Grid.Col sm={12} md={12} xs={12}>
15710                 <Form.Group>
15711                     <Form.Input
15712                         icon="search"
15713                         placeholder="Search for..."
15714                         position="append"
15715                         name="keyword"
15716                         value={this.state.keyword}
15717                         onChange={this.onChangeSearch}
15718                     />
15719                 </Form.Group>
15720             </Grid.Col>
15721         </Grid.Row>
15722     <Spin spinning={this.state.isLoadingTable}>
15723         <Table highlightRowOnHover={true} responsive={true}>
15724             <Table.Header>
15725                 <Table.ColHeader colSpan={2}>Student Name</Table.
15726                     ColHeader>
15727                     <Table.ColHeader>Email Address</Table.ColHeader>
15728                     <Table.ColHeader>Section</Table.ColHeader>
15729                     <Table.ColHeader alignContent="center">Action</
15730                         Table.ColHeader>
15731             <Table.Body>
15732                 {DisplayData.length == 0 ? (
15733                     <Table.Row>
15734                         <Table.Col colSpan={5} alignContent="center">
15735                             No entries.
15736                         </Table.Col>
15737                     ) : (
15738                         DisplayData
15739                     )}
15740             </Table.Body>
15741         </Table>
15742         <Pagination
15743             size="large"
15744             current={this.state.page}
15745             pageSize={this.state.pageSize}
15746             total={this.state.pageSize * this.state.numOfPages}
15747             onChange={this.paginate}
15748         />
15749         </Spin>
15750     </Grid.Col>
15751 </Grid.Row>
15752 </Card.Body>
15753 </Table.Body>
15754 );
15755 }
15756 }
15757 /* istanbul ignore next */
15758 function mapStateToProps(state) {
15759     return {
15760         app: state.app,
15761     };

```

```

15763     }
15764
15765     /* istanbul ignore next */
15766     function mapDispatchToProps(dispatch) {
15767         return {
15768             actions: bindActionCreators({ ...actions }, dispatch),
15769         };
15770     }
15771
15772     export default connect(mapStateToProps, mapDispatchToProps)(
15773         RegistrarViewPastRecords);
15774     import React, { Component } from 'react';
15775     import PropTypes from 'prop-types';
15776     import { bindActionCreators } from 'redux';
15777     import { connect } from 'react-redux';
15778     import * as actions from './redux/actions';
15779     import { Link } from 'react-router-dom';
15780     import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
15781         from 'tabler-react';
15782     import axios from 'axios';
15783     import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
15784     import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
15785         message } from 'antd';
15786     import cn from 'classnames';
15787     import placeholder from '../../images/placeholder.jpg';
15788     import bg from '../../images/BG.png';
15789     import { getImageUrl } from '../../utils';
15790     import ClassRecordInformation from './ClassRecordInformation';
15791     import ViewHonorStudents from './ViewHonorStudents';
15792     import ViewFailedStudents from './ViewFailedStudents';
15793     const { Option } = AutoComplete;
15794
15795     export class RegistrarViewStudentRecord extends Component {
15796         static propTypes = {
15797             app: PropTypes.object.isRequired,
15798             actions: PropTypes.object.isRequired,
15799         };
15800
15801         constructor(props) {
15802             super(props);
15803             this.state = {
15804                 isLoading: true,
15805                 isLoadingTable: true,
15806                 selectedSearchBy: 'Teacher',
15807                 selectedQuarter: 'Q1',
15808                 selectedSchoolYear: '',
15809                 schoolYears: [],
15810                 data: [],
15811                 keyword: '',
15812                 page: 1,
15813                 pageSize: 10,
15814                 numOfPages: 1,
15815             };
15816
15817             this.handleDropdownChange = this.handleDropdownChange.bind(this);
15818             this.onChangeSearch = this.onChangeSearch.bind(this);
15819         }
15820
15821         componentDidMount() {
15822             this.setState({ isLoading: true }, () => {
15823                 axios.get('api/registrar/getallsy').then(res => {
15824                     this.setState(
15825                         {
15826                             schoolYears: res.data.schoolYearList,
15827                             isLoading: false,
15828                             selectedSchoolYear: res.data.schoolYearList[0].schoolYearID
15829                         },
15830                         () => {
15831                             axios
15832                                 .post('api/registrar/getallteachers', {
15833                                     keyword: this.state.keyword,
15834                                     page: this.state.page,

```

```

15832         pageSize: this.state.pageSize,
15833     })
15834     .then(res2 => {
15835       this.setState({
15836         isLoadingTable: false,
15837         numOfPages: res2.data.numOfPages,
15838         data: res2.data.accountList,
15839       });
15840   })
15841   .catch(err => {
15842     this.setState({ isLoadingTable: false, data: [], numOfPages: 1 });
15843   });
15844   },
15845   );
15846   });
15847   });
15848 }
15849
15850 handleDropdownChange = () => {
15851   this.setState({ isLoadingTable: true });
15852   const { selectedSearchBy } = this.state;
15853   if (selectedSearchBy === 'Teacher' || selectedSearchBy === 'Student')
15854     {
15855       axios
15856         .post('api/registrar/getall${selectedSearchBy.toLowerCase()}s',
15857           {
15858             keyword: '',
15859             page: 1,
15860             pageSize: this.state.pageSize,
15861           })
15862         .then(res => {
15863           this.setState({
15864             isLoadingTable: false,
15865             numOfPages: res.data.numOfPages,
15866             data: res.data.accountList,
15867           });
15868         })
15869         .catch(err => {
15870           this.setState({ isLoadingTable: false, data: [], numOfPages: 1 });
15871         });
15872     } else {
15873       axios
15874         .post('api/registrar/getsections', {
15875           keyword: '',
15876           page: 1,
15877           pageSize: this.state.pageSize,
15878         })
15879         .then(res => {
15880           this.setState({
15881             isLoadingTable: false,
15882             numOfPages: res.data.numOfPages,
15883             data: res.data.sectionList,
15884           });
15885         });
15886     };
15887   onChangeSearch(e) {
15888     this.setState({ keyword: e.target.value }, () => {
15889       this.setState({ isLoadingTable: true });
15890       const { selectedSearchBy } = this.state;
15891       if (selectedSearchBy === 'Teacher' || selectedSearchBy === 'Student')
15892         {
15893           axios
15894             .post('api/registrar/getall${selectedSearchBy.toLowerCase()}s',
15895               {
15896                 keyword: this.state.keyword,
15897                 page: 1,
15898                 pageSize: this.state.pageSize,
15899               })
15900             .then(res => {

```

```

15899         this.setState({
15900             isLoadingTable: false,
15901             numOfPages: res.data.numOfPages ,
15902             data: res.data.accountList ,
15903         });
15904     })
15905     .catch(err => {
15906         this.setState({ isLoadingTable: false , data: [] , numOfPages
15907             : 1 });
15908     });
15909 } else {
15910     axios
15911         .post('api/registrar/getsections' , {
15912             keyword: this.state.keyword ,
15913             page: 1 ,
15914             pageSize: this.state.pageSize ,
15915         })
15916         .then(res => {
15917             this.setState({
15918                 isLoadingTable: false ,
15919                 numOfPages: res.data.numOfPages ,
15920                 data: res.data.sectionList ,
15921             });
15922         })
15923         .catch(err => {
15924             this.setState({ isLoadingTable: false , data: [] , numOfPages
15925                 : 1 });
15926         });
15927     }
15928
15929     paginate = page => {
15930         this.setState({
15931             page ,
15932         });
15933         this.setState({ isLoadingTable: true }, () => {
15934             const { selectedSearchBy } = this.state;
15935             if (selectedSearchBy == 'Teacher' || selectedSearchBy == 'Student
15936                 ') {
15937                 axios
15938                     .post('api/registrar/getall${selectedSearchBy.toLowerCase()}s
15939                         ' , {
15940                             keyword: this.state.keyword ,
15941                             page ,
15942                             pageSize: this.state.pageSize ,
15943                         })
15944                     .then(res => {
15945                         this.setState({
15946                             isLoadingTable: false ,
15947                             numOfPages: res.data.numOfPages ,
15948                             data: res.data.accountList ,
15949                         });
15950                     })
15951                     .catch(err => {
15952                         this.setState({ isLoadingTable: false , data: [] , numOfPages
15953                             : 1 });
15954                     });
15955     } else {
15956         axios
15957             .post('api/registrar/getsections' , {
15958                 keyword: this.state.keyword ,
15959                 page ,
15960                 pageSize: this.state.pageSize ,
15961             })
15962             .then(res => {
15963                 this.setState({
15964                     isLoadingTable: false ,
15965                     numOfPages: res.data.numOfPages ,
15966                     data: res.data.sectionList ,
15967                 });
15968             })
15969             .catch(err => {

```

```

15967         this.setState({ isLoadingTable: false, data: [], numOfPages
15968             : 1 });
15969     );
15970   );
15971 };
15972
15973 render() {
15974     const displayGradeLevel = gradeLevel => {
15975         switch (gradeLevel) {
15976             case 'N':
15977                 return 'Nursery';
15978             case 'K1':
15979                 return 'Kinder 1';
15980             case 'K2':
15981                 return 'Kinder 2';
15982             case 'G1':
15983                 return 'Grade 1';
15984             case 'G2':
15985                 return 'Grade 2';
15986             case 'G3':
15987                 return 'Grade 3';
15988             case 'G4':
15989                 return 'Grade 4';
15990             case 'G5':
15991                 return 'Grade 5';
15992             case 'G6':
15993                 return 'Grade 6';
15994             case 'G7':
15995                 return 'Grade 7';
15996             case 'G8':
15997                 return 'Grade 8';
15998             case 'G9':
15999                 return 'Grade 9';
16000             case 'G10':
16001                 return 'Grade 10';
16002             case 'G11':
16003                 return 'Grade 11';
16004             case 'G12':
16005                 return 'Grade 12';
16006             default:
16007                 return '';
16008         }
16009     };
16010     const schoolYearOptions = [];
16011     const displayData = [];
16012     for (const [index, value] of this.state.schoolYears.entries()) {
16013         schoolYearOptions.push(<option value={value.schoolYearID}>{value.
16014             schoolYear}</option>);
16015     }
16016     for (const [index, value] of this.state.data.entries()) {
16017         if (this.state.selectedSearchBy == 'Teacher' || this.state.
16018             selectedSearchBy == 'Student') {
16019             displayData.push(
16020                 <Table.Row>
16021                     <Table.Col className="w-1">
16022                         <Avatar
16023                             imageURL={value.imageUrl == 'NA' ? placeholder :
16024                               getImageUrl(value.imageUrl)}
16025                         />
16026                     </Table.Col>
16027                     <Table.Col>{value.name}</Table.Col>
16028                     <Table.Col>{value.email}</Table.Col>
16029                     <Table.Col>
16030                         <Link
16031                             to={`/viewstudentrecord/${this.state.selectedSearchBy.
16032                               toLowerCase()}/${value['${this.state.selectedSearchBy.toLowerCase()}ID
16033                               ']}
16034                           }/sy/${this.state.selectedSchoolYear}/q/${this.state.
16035                               selectedQuarter}>
16036                         <Button color="primary" icon="file" size="sm" pill>
16037                           View Grades

```

```

16035                     </Button>
16036                 </Link>
16037             </Table.Col>
16038         </Table.Row>,
16039     );
16040 } else {
16041     displayData.push(
16042         <Table.Row>
16043             <Table.Col>{value.name}</Table.Col>
16044             <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
16045         >
16046         <Table.Col>
16047             <Link
16048                 to={`/viewstudentrecord/section/${value.key}/sy/${this.state.selectedSchoolYear}/q/${this.state.selectedQuarter}`}
16049                 >
16050                 <Button color="primary" icon="file" size="sm" pill>
16051                     View Grades
16052                 </Button>
16053             </Link>
16054         </Table.Col>
16055     );
16056 }
16057 }
16058
16059     return (
16060         <div className="app-registrar-view-student-record card my-3 my-md-5">
16061             <Card.Body>
16062                 <Card.Title>
16063                     <Breadcrumb>
16064                         <Breadcrumb.Item>View Student Records</Breadcrumb.Item>
16065                         <Breadcrumb.Item>{this.state.selectedSearchBy}s List</Breadcrumb.Item>
16066                 </Breadcrumb>
16067             </Card.Title>
16068             <Grid.Row>
16069                 <Grid.Col sm={12} xs={12} md={12}>
16070                     <Container>
16071                         <Button.List align="right" style={{ marginBottom: '10px' }}>
16072                             <ViewHonorStudents />
16073                             <ViewFailedStudents />
16074                         </Button.List>
16075                     </Container>
16076                 </Grid.Col>
16077             </Grid.Row>
16078             <Card.Title>View Student Records</Card.Title>
16079             <Container>
16080                 <Spin spinning={this.state.isLoading}>
16081                     <Grid.Row>
16082                         <Grid.Col sm={12} xs={12} md={3}>
16083                             <Form.Group>
16084                                 <Form.Label>Search by</Form.Label>
16085                                 <Form.Select
16086                                     onChange={e =>
16087                                         this.setState({ selectedSearchBy: e.target.value }, () =>
16088                                         this.handleDropdownChange(),
16089                                     )
16090                                 }
16091                         >
16092                             <option>Teacher</option>
16093                             <option>Section</option>
16094                             <option>Student</option>
16095                         </Form.Select>
16096                     </Form.Group>
16097                 </Grid.Col>
16098                 <Grid.Col sm={12} xs={12} md={5}>
16099                     <Form.Group>
16100                         <Form.Label>Select School Year</Form.Label>

```

```

16101 <Form.Select
16102   onChange={e => this.setState({ selectedSchoolYear
16103     : e.target.value })}
16104   >
16105     {schoolYearOptions}
16106   </Form.Select>
16107 </Form.Group>
16108 </Grid.Col>
16109 <Grid.Col sm={12} xs={12} md={4}>
16110   <Form.Group>
16111     <Form.Label>Select Quarter</Form.Label>
16112     <Form.Select onChange={e => this.setState({
16113       selectedQuarter: e.target.value })}>
16114       <option>Q1</option>
16115       <option>Q2</option>
16116       <option>Q3</option>
16117       <option>Q4</option>
16118     </Form.Select>
16119   </Form.Group>
16120 </Grid.Col>
16121 </Grid.Row>
16122 </Spin>
16123 </Container>
16124 <Container>
16125   <Grid.Row>
16126     <Grid.Col sm={12} md={12} xs={12}>
16127       <Form.Group>
16128         <Form.Input
16129           icon="search"
16130           placeholder="Search for..."
16131           position="append"
16132           name="keyword"
16133           value={this.state.keyword}
16134           onChange={this.onChangeSearch}
16135         />
16136       </Form.Group>
16137     </Grid.Col>
16138   </Grid.Row>
16139   <Spin spinning={this.state.isLoadingTable}>
16140     <Grid.Row>
16141       <Table highlightRowOnHover={true} responsive={true}>
16142         {this.state.selectedSearchBy == 'Teacher' && (
16143           <Table.Header>
16144             <Table.ColHeader></Table.ColHeader>
16145             <Table.ColHeader>Teacher Name</Table.ColHeader>
16146             <Table.ColHeader>Email</Table.ColHeader>
16147             <Table.ColHeader>Action</Table.ColHeader>
16148           </Table.Header>
16149         {this.state.selectedSearchBy == 'Student' && (
16150           <Table.Header>
16151             <Table.ColHeader></Table.ColHeader>
16152             <Table.ColHeader>Student Name</Table.ColHeader>
16153             <Table.ColHeader>Email</Table.ColHeader>
16154             <Table.ColHeader>Action</Table.ColHeader>
16155           </Table.Header>
16156         {this.state.selectedSearchBy == 'Section' && (
16157           <Table.Header>
16158             <Table.ColHeader>Section Name</Table.ColHeader>
16159             <Table.ColHeader>Grade Level</Table.ColHeader>
16160             <Table.ColHeader>Action</Table.ColHeader>
16161           </Table.Header>
16162         )}>
16163       <Table.Body>
16164         {displayData.length == 0 ? (
16165           this.selectedSearchBy == 'Section' ? (
16166             <Table.Row>
16167               <Table.Col colSpan={3} alignContent="center">
16168                 No entries.
16169               </Table.Col>
16170             </Table.Row>

```

```

16171         ) : (
16172             <Table.Row>
16173                 <Table.Col colSpan={4}>No entries.</Table.Col>
16174             )
16175         )
16176     ) : (
16177         displayData
16178     )
16179     </Table.Body>
16180   </Table>
16181   <Pagination
16182     size="large"
16183     current={this.state.page}
16184     pageSize={this.state.pageSize}
16185     total={this.state.pageSize * this.state.numOfPages}
16186     onChange={this.paginate}
16187   />
16188   </Grid.Row>
16189   </Spin>
16190   </Container>
16191   </Card.Body>
16192   </div>
16193 );
16194 }
16195 }
16196
16197 /* istanbul ignore next */
16198 function mapStateToProps(state) {
16199   return {
16200     app: state.app,
16201   };
16202 }
16203
16204 /* istanbul ignore next */
16205 function mapDispatchToProps(dispatch) {
16206   return {
16207     actions: bindActionCreators({ ...actions }, dispatch),
16208   };
16209 }
16210
16211 export default connect(mapStateToProps, mapDispatchToProps)(
16212   RegistrarViewStudentRecord);
16213 import React, { Component } from 'react';
16214 import PropTypes from 'prop-types';
16215 import { bindActionCreators } from 'redux';
16216 import { connect } from 'react-redux';
16217 import * as actions from './redux/actions';
16218 import { Link } from 'react-router-dom';
16219 import { Card, Button, Grid, Table, Container, Text, Avatar, Alert } from 'tabler-react';
16220 import axios from 'axios';
16221 import { Pagination, Spin, Tooltip, Modal, Popover } from 'antd';
16222 import AllStudentFinalGrades from './AllStudentFinalGrades';
16223 import placeholder from '../../images/placeholder.jpg';
16224 import { getImageUrl } from '../../utils';
16225
16226 export class RegistrarViewStudentRecordSection extends Component {
16227   static propTypes = {
16228     app: PropTypes.object.isRequired,
16229     actions: PropTypes.object.isRequired,
16230   };
16231   constructor(props) {
16232     super(props);
16233     this.state = {
16234       isLoading: false,
16235       data: [],
16236       columns: [],
16237       sectionName: '',
16238       schoolYear: '',
16239       studentIDs: []

```

```

16240     };
16241 }
16242
16243     componentWillMount() {
16244         this.setState({ isLoading: true }, () => {
16245             axios
16246                 .post('api/registrar/getsynname', { schoolYearID: this.props.
16247                     schoolYearID })
16248                 .then(res3 => {
16249                     this.setState({ schoolYear: res3.data.schoolYear });
16250                 });
16251             axios.post('api/registrar/getsectionname', { sectionID: this.
16252                 props.id }).then(res2 => {
16253                     this.setState({ sectionName: res2.data.sectionName });
16254                 axios
16255                     .post('api/registrar/condensedfinalgrade', {
16256                         sectionID: this.props.id,
16257                         quarter: this.props.quarter,
16258                     })
16259                     .then(res => {
16260                         this.setState({
16261                             isLoading: false,
16262                             data: res.data.data,
16263                             columns: res.data.columns,
16264                             studentIDs: res.data.data.map(a => a.studentID),
16265                         });
16266                     });
16267                 });
16268             }
16269             render() {
16270                 const DisplayColumns = [];
16271                 const DisplayData3 = [];
16272                 DisplayColumns.push(<Table.ColHeader></Table.ColHeader>);
16273                 DisplayColumns.push(<Table.ColHeader>Name</Table.ColHeader>);
16274                 for (const [index, value] of this.state.data.entries()) {
16275                     let tempCol = [];
16276                     for (const [index2, value2] of value.grades.entries()) {
16277                         tempCol.push(
16278                             <Table.Col>
16279                                 <Popover
16280                                     placement="top"
16281                                     content={
16282                                         <div>
16283                                             {this.state.columns.length != 0 && this.state.data.
16284                                                 length != 0 && (
16285                                         <React.Fragment>
16286                                             <p>Subject Name: {value2.subjectName}</p>
16287                                             <p>
16288                                                 Teacher Name:{' '}
16289                                                 {this.state.columns.find(a => value2.
16290                                                     subjectName == a.subjectName).teacher}
16291                                             </p>
16292                                             <p>
16293                                                 <Link
16294                                                     to={`/viewstudentrecord/classrecord/${
16295                                                         this.state.columns.find(a => value2.
16296                                                         subjectName == a.subjectName)
16297                                                         .classRecordID
16298                                                         }/q/${this.props.quarter}`}
16299                                                         target="_blank"
16300                                                 >
16301                                                 <Button icon="eye" block color="primary" size
16302                                                     ="sm" pill>
16303                                                     View Class Record
16304                                                 </Button>
16305                                                 </Link>
16306                                             </p>
16307                                         </React.Fragment>
16308                                     )
16309                                 </div>
16310                             );
16311                         );
16312                     );
16313                 );
16314             );
16315         );
16316     );
16317 
```

```

16307      >
16308      { (value2.grade != -1 || value2.grade != 'N/A') && (
16309        <span
16310          className={'status-icon bg-$\{parseFloat(value2.grade)
16311            >= 75 ? 'green' : 'red'\}'}
16312        />
16313      ) }
16314      { value2.grade == -1 ? 'N/A' : value2.grade == 'N/A' ? '
16315        Not enrolled' : value2.grade}
16316    </Popover>
16317  </Table.Col>,
16318 }
16319 DisplayData3.push(
16320   <Table.Row>
16321     <Table.Col className="w-1">
16322       <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
16323         getImageUrl(value.imageUrl)} />
16324     </Table.Col>
16325     <Table.Col>
16326       <Tooltip placement="top" title="View all grades">
16327         <AllStudentFinalGrades
16328           text="View Condensed Grade"
16329           schoolYear={this.state.schoolYear}
16330           id={value.studentID}
16331           schoolYearID={this.props.schoolYearID}
16332         />
16333       </Tooltip>
16334     </Table.Col>
16335   </Table.Row>,
16336 );
16337 }
16338 for (const [index, value] of this.state.columns.entries()) {
16339   DisplayColumns.push(
16340     <Table.ColumnHeader>
16341       {value.status == 'D' && <span className={'status-icon bg-
16342         yellow'} />}
16343       {value.subjectName}
16344     </Table.ColumnHeader>,
16345   );
16346 }
16347 return (
16348   <div className="app-registrar-view-student-record-section my-3 my-
16349     -md-5">
16350     <Grid.Col sm={12} sm={12}>
16351       <Grid.Row>
16352         <Container>
16353           <Card statusColor="info">
16354             <Card.Body>
16355               <Card.Title>Condensed Grades of {this.state.
16356                 sectionName}</Card.Title>
16357               <Card.Title>
16358                 <Grid.Row>
16359                   <Grid.Col sm={12} xs={12} md={12}>
16360                     <Button.List align="right">
16361                       {this.props.app.auth.user.position == 2 && (
16362                         <Button
16363                           disabled={DisplayData3.length == 0}
16364                           icon="file"
16365                           color="primary"
16366                           onClick={() =>
16367                             this.props.actions.generatePdfSection(
16368                               {
16369                                 studentIDs: this.state.studentIDs,
16370                                 quarter: this.props.quarter,
16371                                 schoolYearID: this.props.
16372                                   schoolYearID,
16373                               },
16374                               'Registrar',
16375                             )
16376                           }
16377                         )
16378                       )
16379                     )
16380                   </Grid.Col>
16381                 </Grid.Row>
16382               </Card>
16383             </Card.Body>
16384           </Card>
16385         </Container>
16386       </Grid.Row>
16387     </Grid.Col>
16388   </div>

```

```

16373                               >
16374                         Generate Report Card
16375                     </Button>
16376                   )}
16377             </Button.List>
16378           </Grid.Col>
16379         </Grid.Row>
16380       </Card.Title>
16381     <Card.Title>
16382       <Text.Small>
16383         S.Y. {this.state.schoolYear} {this.props.quarter}
16384       </Text.Small>
16385     </Card.Title>
16386   </Card.Body>
16387 <Card.Body>
16388   <Spin spinning={this.state.isLoading}>
16389     <Grid.Row>
16390       <Grid.Col sm={12} xs={12} md={12}>
16391         <Alert type="info" icon="info">
16392           Note: Column names with status color{' '}
16393           <span className="status-icon bg-yellow" />
16394         <b>
16395           <i>yellow</i>
16396         </b>{' '}
16397         are subjects which are still under
16398           deliberation process.{ ' '}
16399         <b>
16400           <i>'View Condensed Grade'</i>
16401         </b>{' '}
16402         button shows the final grade (grades under
16403           deliberation process{ ' '}
16404         <b>
16405           <i>not included.</i>
16406         </b>
16407           ) of the student per quarter.
16408       </Alert>
16409     </Grid.Col>
16410   <Table highlightRowOnHover={true} responsive={true}>
16411     <Table.Header>{DisplayColumns}</Table.Header>
16412     <Table.Body>
16413       {DisplayData3.length == 0 ? (
16414         <Table.Row>
16415           <Table.Col colSpan={10} alignContent="center">
16416             No data available.
16417           </Table.Col>
16418         </Table.Row>
16419       ) : (
16420         DisplayData3
16421       )}
16422     </Table.Body>
16423   </Table>
16424 </Grid.Row>
16425   <Spin>
16426     </Card.Body>
16427   </Card>
16428 </Container>
16429   </Grid.Row>
16430   </Grid.Col>
16431 </div>
16432 );
16433 }
16434 /* istanbul ignore next */
16435 function mapStateToProps(state) {
16436   return {
16437     app: state.app,
16438   };
16439 }
16440

```

```

16441 /* istanbul ignore next */
16442 function mapDispatchToProps(dispatch) {
16443   return {
16444     actions: bindActionCreators({ ...actions }, dispatch),
16445   };
16446 }
16447
16448 export default connect(mapStateToProps, mapDispatchToProps)(
16449   RegistrarViewStudentRecordSection);
16450 import React, { Component } from 'react';
16451 import PropTypes from 'prop-types';
16452 import { bindActionCreators } from 'redux';
16453 import { connect } from 'react-redux';
16454 import * as actions from './redux/actions';
16455 import { Link } from 'react-router-dom';
16456 import { Card, Button, Grid, Avatar, Table, Form, Header, Container,
16457   Text } from 'tabler-react';
16458 import axios from 'axios';
16459 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
16460 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
16461   message } from 'antd';
16462 import cn from 'classnames';
16463 import placeholder from '../../images/placeholder.jpg';
16464 import bg from '../../images/BG.png';
16465 import { getImageUrl } from '../../utils';
16466 import AllStudentFinalGrades from './AllStudentFinalGrades';
16467 const { Option } = AutoComplete;
16468
16469 function ProfileImage({ avatarURL }) {
16470   return <img className="card-profile-img" alt="Profile" src={avatarURL}
16471     />;
16472 }
16473
16474 function Profile({ className, children, name, avatarURL = '',
16475   backgroundURL = '', bio }) {
16476   const classes = cn('card-profile', className);
16477   return (
16478     <Card className={classes}>
16479       <Card.Header backgroundURL={backgroundURL} />
16480       <Card.Body className="text-center">
16481         <ProfileImage avatarURL={avatarURL} />
16482         <Header.H3 className="mb-3">{name}</Header.H3>
16483         <p className="mb-4">{bio || children}</p>
16484       </Card.Body>
16485     </Card>
16486   );
16487 }
16488
16489 export class RegistrarViewStudentRecordStudent extends Component {
16490   static propTypes = {
16491     app: PropTypes.object.isRequired,
16492     actions: PropTypes.object.isRequired,
16493   };
16494
16495   constructor(props) {
16496     super(props);
16497     this.state = {
16498       isLoading: true,
16499       isLoadingTable: true,
16500       name: '',
16501       email: '',
16502       accountID: '',
16503       imageUrl: '',
16504       data: [],
16505       schoolYearID: 0,
16506       schoolYear: '',
16507       page: 1,
16508       pageSize: 10,
16509       numPages: 1,
16510       quarter: '',
16511       finalGrade: -1,
16512       sectionName: '',
16513     };
16514   }

```

```

16509     }
16510
16511     componentDidMount() {
16512       axios.post('api/registrar/getsy', { schoolYearID: this.props.
16513         schoolYearID }).then(res => {
16514           this.setState({
16515             schoolYearID: res.data.schoolYearID,
16516             schoolYear: res.data.schoolYear,
16517           });
16518         axios.post('api/registrar/studentinfo', { studentID: this.props.
16519           id }).then(res2 => {
16520           axios
16521             .post('api/registrar/getsectionstudent', {
16522               studentID: this.props.id,
16523                 schoolYearID: this.props.schoolYearID,
16524               })
16525             .then(res5 => {
16526               this.setState({
16527                 email: res2.data.email,
16528                   imageUrl: res2.data.imageUrl,
16529                     name: res2.data.name,
16530                       isLoading: false,
16531                         sectionName: res5.data.sectionName,
16532                           });
16533             axios
16534               .post('api/registrar/studentfinalrecord', {
16535                 studentID: this.props.id,
16536                   schoolYearID: this.props.schoolYearID,
16537                     quarter: this.props.quarter,
16538                   })
16539                     .then(res3 => {
16540                       this.setState({
16541                         isLoadingTable: false,
16542                           data: res3.data.data,
16543                             finalGrade: res3.data.finalGrade,
16544                               });
16545                     .catch(err => {
16546                       this.setState({
16547                         isLoadingTable: false,
16548                           data: [],
16549                             finalGrade: -1,
16550                               });
16551                     });
16552                   });
16553                 );
16554   }
16555
16556   render() {
16557     const DisplayData = [];
16558     for (const [index, value] of this.state.data.entries()) {
16559       DisplayData.push(
16560         <Table.Row>
16561           <Table.Col>{value.subjectName}</Table.Col>
16562             <Table.Col>
16563               {value.score != -1 || value.score != 'N/A') && (
16564                 <span
16565                   className={'status-icon bg-$parseFloat(value.score) >=
16566                     75 ? 'green' : 'red'}'}
16567                   />
16568                 )}
16569                   {value.score == -1 ? 'Not yet available' : value.score}
16570                     </Table.Col>
16571                   </Table.Row>,
16572                 );
16573   }
16574   return (
16575     <div className="app-registrar-view-student-record-student my-3 my-
16576       -md-5">
16577       <Container>
16578         <Grid.Row>
16579           <Grid.Col sm={12} lg={4} md={8}>

```

```

16578 <Spin spinning={this.state.isLoading}>
16579   <Grid.Row>
16580     <Container>
16581       <Grid.Row>
16582         {this.state.isLoading ? (
16583           <Profile name="" avatarURL={placeholder}
16584             backgroundUrl=""></Profile>
16585         ) : (
16586           <Profile
16587             name={this.state.name}
16588             avatarURL={
16589               this.state.imageUrl === 'NA'
16590                 ? placeholder
16591                   : getImageUrl(this.state.imageUrl)
16592             }
16593             backgroundURL={bg}
16594           ></Profile>
16595         )}
16596       </Grid.Row>
16597     </Container>
16598   </Grid.Row>
16599 </Spin>
16600 </Grid.Col>
16601 <Grid.Col sm={12} lg={8} md={8}>
16602   <Container>
16603     <Spin spinning={this.state.isLoading}>
16604       <Grid.Row>
16605         <Card statusColor="success">
16606           <Card.Body>
16607             <Grid.Row>
16608               <Grid.Col sm={12} xs={12} md={8}>
16609                 {', '}
16610               <Card.Title>Student Record of {this.state.
16611                 name}</Card.Title>
16612               <Card.Title>
16613                 <Text.Small>
16614                   S.Y {this.state.schoolYear} {this.props
16615                     .quarter}
16616                   </Text.Small>
16617                 </Card.Title>
16618               <AllStudentFinalGrades
16619                 text={'Grades for S.Y. ' + this.state.
16620                   schoolYear}
16621                   schoolYear={this.state.schoolYear}
16622                   id={this.props.id}
16623                   schoolYearID={this.props.schoolYearID}
16624                 />
16625               </Grid.Col>
16626             <Card.Body>
16627               <Spin spinning={this.state.isLoadingTable}>
16628                 <Grid.Row>
16629                   <Table highlightRowOnHover={true}>
16630                     responsive={true}>
16631                     <Table.Header>
16632                       <Table.ColumnHeader>Subject Name</Table.
16633                         ColHeader>
16634                       <Table.ColumnHeader>Grade</Table.
16635                         ColHeader>
16636                     </Table.Header>
16637                     <Table.Body>
16638                       {DisplayData.length == 0 ? (
16639                         <Table.Row>
16640                           <Table.Col colSpan={2}
16641                             alignContent="center">
16642                               No entries.
16643                           </Table.Col>
16644                         </Table.Row>
16645                       ) : (
16646                         DisplayData
16647                       )}>

```

```

16643         </Table.Body>
16644     </Table>
16645   </Grid.Row>
16646   <Grid.Row>
16647     <span style={{ fontSize: '18px' }}>
16648       Final Grade for {this.props.quarter}:{'}
16649       '
16650     <b>
16651       {this.state.finalGrade == -1
16652         ? 'Not yet available'
16653           : this.state.finalGrade}
16654     </b>
16655   </span>
16656 </Grid.Row>
16657   </Spin>
16658 </Card.Body>
16659 </Card.Body>
16660 <Card.Footer>
16661   <Button.List align="right">
16662     {this.props.app.auth.user.position == 2 && (
16663       <Button
16664         color="primary"
16665           icon="file"
16666           onClick={() =>
16667             this.props.actions.generatePdfStudent(
16668               {
16669                 studentID: this.props.id,
16670                   schoolYearID: this.props.
16671                     schoolYearID,
16672                     quarter: this.props.quarter,
16673                     },
16674                     'Registrar',
16675                   )
16676             >
16677               Generate Report Card
16678             </Button>
16679           )})
16680         </Button.List>
16681       </Card.Footer>
16682     </Grid.Row>
16683   </Spin>
16684   </Container>
16685   </Grid.Col>
16686 </Grid.Row>
16687 </Container>
16688 </div>
16689 );
16690 }
16692 }
16693 /* istanbul ignore next */
16694 function mapStateToProps(state) {
16695   return {
16696     app: state.app,
16697   };
16698 }
16699 /* istanbul ignore next */
16700 function mapDispatchToProps(dispatch) {
16701   return {
16702     actions: bindActionCreators({ ...actions }, dispatch),
16703   };
16704 }
16705 }
16706
16707 export default connect(mapStateToProps, mapDispatchToProps)(
16708   RegistrarViewStudentRecordStudent);
16709 import React, { Component } from 'react';
16710 import PropTypes from 'prop-types';
16711 import { bindActionCreators } from 'redux';
16712 import { connect } from 'react-redux';

```

```

16712 import * as actions from './redux/actions';
16713 import { Link } from 'react-router-dom';
16714 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
16715     from 'tabler-react';
16716 import axios from 'axios';
16717 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
16718 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
16719     message } from 'antd';
16720 import cn from 'classnames';
16721 import placeholder from '../images/placeholder.jpg';
16722 import bg from '../images/BG.png';
16723 import { getImageUrl } from '../utils';
16724 import ClassRecordInformation from './ClassRecordInformation';
16725 const { Option } = AutoComplete;
16726
16727 function ProfileImage({ avatarURL }) {
16728     return <img className="card-profile-img" alt="Profile" src={avatarURL}
16729         />;
16730 }
16731
16732 function Profile({ className, children, name, avatarURL = '',
16733     backgroundURL = '', bio }) {
16734     const classes = cn('card-profile', className);
16735     return (
16736         <Card className={classes}>
16737             <Card.Header backgroundURL={backgroundURL} />
16738             <Card.Body className="text-center">
16739                 <ProfileImage avatarURL={avatarURL} />
16740                 <Header.H3 className="mb-3">{name}</Header.H3>
16741                 <p className="mb-4">{bio || children}</p>
16742             </Card.Body>
16743         </Card>
16744     );
16745 }
16746
16747 export class RegistrarViewStudentRecordTeacher extends Component {
16748     static propTypes = {
16749         app: PropTypes.object.isRequired,
16750         actions: PropTypes.object.isRequired,
16751     };
16752
16753     constructor(props) {
16754         super(props);
16755         this.state = {
16756             isLoading: true,
16757             isLoadingTable: true,
16758             isLoadingTable2: true,
16759             name: '',
16760             email: '',
16761             accountID: '',
16762             imageUrl: '',
16763             data: [],
16764             data2: [],
16765             schoolYearID: 0,
16766             schoolYear: '',
16767             page: 1,
16768             page2: 1,
16769             keyword: '',
16770             pageSize: 10,
16771             pageSize2: 10,
16772             numOfPages: 1,
16773             numOfPages2: 1,
16774             deadline: '',
16775             selectedClassRecordID: -1,
16776             selectedSubjectCode: '',
16777             selectedSubjectName: '',
16778             selectedSection: '',
16779             selectedDeadline: '',
16780             quarter: '',
16781             showRevert: false,
16782         };
16783     }
16784
16785     resetClassRecordInfo = () => {

```

```

16782     this.setState({
16783       selectedClassRecordID: -1,
16784       selectedSubjectCode: '',
16785       selectedSubjectName: '',
16786       selectedSection: '',
16787       selectedDeadline: '',
16788     });
16789   };
16790
16791   componentDidMount() {
16792     axios.get('api/registrar/getsy').then(res => {
16793       this.setState({
16794         schoolYearID: res.data.schoolYearID,
16795         schoolYear: res.data.schoolYear,
16796         quarter: res.data.quarter,
16797       });
16798       axios.post('api/registrar/teacherinfo', { teacherID: this.props.
16799         id }).then(res2 => {
16800         this.setState({
16801           email: res2.data.email,
16802           imageUrl: res2.data.imageUrl,
16803           name: res2.data.name,
16804           isLoading: false,
16805         });
16806         axios
16807           .post('api/registrar/getfinalsubsect', {
16808             teacherID: this.props.id,
16809             page: this.state.page,
16810             pageSize: this.state.pageSize,
16811             quarter: this.props.quarter,
16812           })
16813           .then(res3 => {
16814             this.setState({
16815               data: res3.data.classRecordList,
16816               numOfPages: res3.data.numOfPages,
16817               deadline: res3.data.deadline,
16818               isLoadingTable: false,
16819             });
16820           })
16821           .catch(err => {
16822             this.setState({ data: [], numOfPages: 1, isLoadingTable:
16823               false });
16824           });
16825         });
16826
16827       componentWillMount() {
16828         axios.get('api/registrar/getsy').then(res => {
16829           this.setState({
16830             schoolYearID: res.data.schoolYearID,
16831             schoolYear: res.data.schoolYear,
16832             quarter: res.data.quarter,
16833           });
16834           axios.post('api/registrar/teacherinfo', { teacherID: this.props.
16835             id }).then(res2 => {
16836             this.setState({
16837               email: res2.data.email,
16838               imageUrl: res2.data.imageUrl,
16839               name: res2.data.name,
16840               isLoading: false,
16841             });
16842             axios
16843               .post('api/registrar/getfinalsubsect', {
16844                 teacherID: this.props.id,
16845                 page: this.state.page,
16846                 pageSize: this.state.pageSize,
16847                 quarter: this.props.quarter,
16848               })
16849               .then(res3 => {
16850                 this.setState({
16851                   data: res3.data.classRecordList,
16852                     numOfPages: res3.data.numOfPages,
16853                     deadline: res3.data.deadline,
16854                     isLoadingTable: false,
16855                   });
16856                 });
16857               });
16858             });
16859           });
16860         });
16861       });
16862     });
16863   }

```

```

16854         });
16855     })
16856     .catch(err => {
16857       this.setState({ data: [], numOfPages: 1, isLoadingTable:
16858         false });
16859     });
16860   });
16861 }
16862
16863 paginate = page => {
16864   this.setState({ page, isLoadingTable: true }, async () => {
16865     axios
16866       .post('api/registrar/getsubmittedsubsect', {
16867         teacherID: this.props.id,
16868         page: page,
16869         pageSize: this.state.pageSize,
16870         quarter: this.state.quarter,
16871       })
16872       .then(res3 => {
16873         this.setState({
16874           data: res3.data.classRecordList,
16875           numOfPages: res3.data.numOfPages,
16876           deadline: res3.data.deadline,
16877           isLoadingTable: false,
16878         });
16879       });
16880     });
16881   );
16882
16883 render() {
16884   const DisplayData = [];
16885   const DisplayData2 = [];
16886   for (const [index, value] of this.state.data.entries()) {
16887     const dateSubmitted = new Date(value.dateSubmitted);
16888     let status = '';
16889     if (this.state.deadline == 'NOT SET') {
16890       status = 'NOT SET';
16891     } else {
16892       let deadline = new Date(this.state.deadline);
16893       let relativeTime = new Date(Math.abs(deadline.getTime() -
16894         dateSubmitted.getTime())));
16895       let s = relativeTime / 1000;
16896       let m = s / 60;
16897       let h = m / 60;
16898       let d = h / 24;
16899       let overDueText = Math.floor(d) == 0 ? '' : Math.floor(d) +
1700         ' day/s ';
1701       overDueText =
1702         overDueText +
1703         Math.floor(h % 24) +
1704         ' hr ' +
1705         Math.floor(m % 60) +
1706         ' min ' +
1707         Math.floor(s % 60) +
1708         ' secs';
1709
1710       status = deadline.getTime() < dateSubmitted.getTime() ? ${
1711         overDueText} late' : 'On Time';
1712     }
1713     DisplayData.push(
1714       <Table.Row>
1715         <Table.Col>{value.subjectCode}</Table.Col>
1716         <Table.Col>{value.subjectName}</Table.Col>
1717         <Table.Col>{value.section}</Table.Col>
1718         <Table.Col>{value.status == 'D' ? 'Deliberation' : 'Final'}</
1719           Table.Col>
1720         <Table.Col>
1721           <Button
1722             icon="eye"
1723             size="sm"
1724             outline
1725             pill

```

```

16922         color="primary"
16923         onClick={() =>
16924             this.setState({
16925                 selectedClassRecordID: value.classRecordID,
16926                 selectedSubjectCode: value.subjectCode,
16927                 selectedSubjectName: value.subjectName,
16928                 selectedSection: value.section,
16929                 selectedDeadline: status,
16930                 showRevert: value.status != 'F',
16931             })
16932         }
16933     ></Button>
16934   </Table.Col>
16935 </Table.Row>,
16936 );
16937 }
16938
16939 for (const [index, value] of this.state.data2.entries()) {
16940     let displayDate = 'NOT SET';
16941     if (this.state.deadline != 'NOT SET') {
16942         displayDate = new Date(this.state.deadline);
16943         displayDate = displayDate.toDateString();
16944     }
16945     DisplayData2.push(
16946       <Table.Row>
16947         <Table.Col>{value.subjectCode}</Table.Col>
16948         <Table.Col>{value.subjectName}</Table.Col>
16949         <Table.Col>{value.section}</Table.Col>
16950         <Table.Col>{displayDate}</Table.Col>
16951     </Table.Row>,
16952   );
16953 }
16954
16955 return (
16956   <div className="app-registrar-individual-deliberation-info my-3
16957     my-md-5">
16958     <Container>
16959       <Grid.Row>
16960         <Grid.Col sm={12} lg={5}>
16961           <Spin spinning={this.state.isLoading}>
16962             <Grid.Row>
16963               <Container>
16964                 {this.state.isLoading ? (
16965                   <Profile name="" avatarURL={placeholder}
16966                     backgroundImage=""></Profile>
16967                   ) : (
16968                     <Profile
16969                       name={this.state.name}
16970                         avatarURL={
16971                           this.state.imageUrl === 'NA'
16972                             ? placeholder
16973                             : getImageUrl(this.state.imageUrl)
16974                         }
16975                         backgroundURL={bg}
16976                     ></Profile>
16977                   )
16978                 </Grid.Row>
16979               </Container>
16980             </Spin>
16981           </Grid.Col>
16982           <Grid.Col sm={12} xs={12} md={7}>
16983             {' '}
16984             <Grid.Row>
16985               <Container>
16986                 {this.state.isLoading ? (
16987                   '',
16988                 ) : (
16989                   <Card statusColor="warning">
16990                     <Card.Body>
16991                       <Card.Title>
16992                         Class Record of {this.state.name} S.Y. {this.

```

```
state.schoolYear}
        </Card.Title>
        <Grid.Row>
            <Grid.Col sm={12} md={12} xs={12}>
                <Spin spinning={this.state.isLoadingTable}>
                    <Table highlightRowOnHover={true}>
                        responsive={true}>
                        <Table.Header>
                            <Table.ColHeader>Code</Table.
                                ColHeader>
                            <Table.ColHeader>Subject </Table.
                                ColHeader>
                            <Table.ColHeader>Section </Table.
                                ColHeader>
                            <Table.ColHeader>Status </Table.
                                ColHeader>
                            <Table.ColHeader>Actions </Table.
                                ColHeader>
                        </Table.Header>
                        <Table.Body>
                            {DisplayData.length == 0 ? (
                                <Table.Row>
                                    <Table.Col colSpan={5}
                                        alignContent="center">
                                        No submitted class record.
                                    </Table.Col>
                                </Table.Row>
                            ) : (
                                DisplayData
                            )}
                        </Table.Body>
                    </Table>
                    <Pagination
                        size="large"
                        current={this.state.page}
                        pageSize={this.state.pageSize}
                        total={this.state.pageSize * this.state
                            .numOfPages}
                        onChange={this.paginate}
                    />
                    </Spin>
                </Grid.Col>
            </Grid.Row>
            <Card.Body>
                </Card>
            )
        </Container>
    </Grid.Row>
    </Grid.Col>
    <Grid.Col sm={12} xs={12} md={12}>
        <Grid.Row>
            <Container>
                <ClassRecordInformation
                    classRecordID={this.state.selectedClassRecordID}
                    quarter={this.props.quarter}
                    subjectCode={this.state.selectedSubjectCode}
                    subjectName={this.state.selectedSubjectName}
                    section={this.state.selectedSection}
                    deadline={this.state.selectedDeadline}
                    resetClassRecordInfo={this.resetClassRecordInfo}
                    id={this.props.id}
                    status="final"
                    showRevert={this.state.showRevert}
                />
            </Container>
        </Grid.Row>
        </Grid.Col>
    </Grid.Row>
    <Container>
        </div>
    );
}
```

```

17056    }
17057
17058    /* istanbul ignore next */
17059    function mapStateToProps(state) {
17060        return {
17061            app: state.app,
17062        };
17063    }
17064
17065    /* istanbul ignore next */
17066    function mapDispatchToProps(dispatch) {
17067        return {
17068            actions: bindActionCreators({ ...actions }, dispatch),
17069        };
17070    }
17071
17072    export default connect(mapStateToProps, mapDispatchToProps)(
17073        RegistrarViewStudentRecordTeacher);
17074    import React, { Component } from 'react';
17075    import PropTypes from 'prop-types';
17076    import { bindActionCreators } from 'redux';
17077    import { connect } from 'react-redux';
17078    import * as actions from './redux/actions';
17079    import { Link } from 'react-router-dom';
17080    import { Card, Button, Grid, Avatar, Table, Form, Header, Container } from 'tabler-react';
17081    import axios from 'axios';
17082    import { Pagination, Spin, Tooltip } from 'antd';
17083    import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input, message } from 'antd';
17084    import RegistrarAddNewLoad from './RegistrarAddNewLoad';
17085    import cn from 'classnames';
17086    import placeholder from '../../images/placeholder.jpg';
17087    import bg from '../../images/BG.png';
17088    import { getImageUrl } from '../../utils';
17089    const { Option } = AutoComplete;
17090
17091    function ProfileImage({ avatarURL }) {
17092        return <img className="card-profile-img" alt="Profile" src={avatarURL} />;
17093    }
17094
17095    function Profile({ className, children, name, avatarURL = '',
17096        backgroundURL = '', bio }) {
17097        const classes = cn('card-profile', className);
17098        return (
17099            <Card className={classes}>
17100                <Card.Header backgroundURL={backgroundURL} />
17101                <Card.Body className="text-center">
17102                    <ProfileImage avatarURL={avatarURL} />
17103                    <Header.H3 className="mb-3">{name}</Header.H3>
17104                    <p className="mb-4">{bio || children}</p>
17105                </Card.Body>
17106            </Card>
17107        );
17108    }
17109
17110    export class RegistrarViewSubjectLoad extends Component {
17111        static propTypes = {
17112            app: PropTypes.object.isRequired,
17113            actions: PropTypes.object.isRequired,
17114        };
17115
17116        constructor(props) {
17117            super(props);
17118            this.state = {
17119                isLoading: true,
17120                isLoadingTable: true,
17121                name: '',
17122                email: '',
17123                accountID: '',
17124                imageUrl: '',
17125                data: [],

```

```

17124     schoolYearID: 0,
17125     schoolYear: '',
17126     page: 1,
17127     keyword: '',
17128     pageSize: 10,
17129     numOfPages: 1,
17130   );
17131 }
17132
17133 componentDidMount() {
17134   axios.get('api/registrar/getsy').then(res => {
17135     this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:
17136       res.data.schoolYear });
17137     axios.post('api/registrar/userinfo', { accountID: this.props.id
17138       }).then(res2 => {
17139         this.setState({
17140           email: res2.data.email,
17141           imageUrl: res2.data.imageUrl,
17142           name: res2.data.name,
17143           isLoading: false,
17144         });
17145         axios
17146           .post('api/registrar/getsubjectsection', {
17147             page: this.state.page,
17148             pageSize: this.state.pageSize,
17149             accountID: this.props.id,
17150             schoolYearID: res.data.schoolYearID,
17151           })
17152             .then(res3 => {
17153               this.setState({
17154                 numOfPages: res3.data.numOfPages,
17155                 data: res3.data.subjectSectionData,
17156                 isLoadingTable: false,
17157               });
17158               .catch(err => {
17159                 this.setState({ isLoadingTable: false, data: [] });
17160               });
17161             });
17162           });
17163
17164 paginate = page => {
17165   this.setState({
17166     page,
17167   });
17168   this.setState({ isLoadingTable: true });
17169   axios
17170     .post('api/registrar/getsubjectsection', {
17171       page,
17172       pageSize: this.state.pageSize,
17173       accountID: this.props.id,
17174       schoolYearID: this.state.schoolYearID,
17175     })
17176     .then(res => {
17177       this.setState({ isLoadingTable: false });
17178       this.setState({
17179         numOfPages: res.data.numOfPages,
17180         data: res.data.subjectSectionData,
17181       });
17182     })
17183     .catch(err => {
17184       this.setState({ isLoadingTable: false, data: [] });
17185     });
17186   };
17187
17188 deleteSubjectSection(key) {
17189   this.props.actions.deleteSubjectSection({ subsectID: key });
17190 }
17191
17192 componentWillMountReceiveProps() {
17193   axios.get('api/registrar/getsy').then(res => {
17194     this.setState({ schoolYearID: res.data.schoolYearID, schoolYear:

```

```

        res.data.schoolYear });
17195    axios.post('api/registrar/userinfo', { accountID: this.props.id
17196      }).then(res2 => {
17197        this.setState({
17198          email: res2.data.email,
17199          imageUrl: res2.data.imageUrl,
17200          name: res2.data.name,
17201          isLoading: false,
17202        });
17203        axios
17204          .post('api/registrar/getsubjectsection', {
17205            page: this.state.page,
17206            pageSize: this.state.pageSize,
17207            accountID: this.props.id,
17208            schoolYearID: res.data.schoolYearID,
17209          })
17210          .then(res3 => {
17211            this.setState({
17212              numOfpages: res3.data.numOfPages,
17213              data: res3.data.subjectSectionData,
17214              isLoadingTable: false,
17215            });
17216            .catch(err => {
17217              this.setState({ isLoadingTable: false, data: [] });
17218            });
17219          });
17220        });
17221      }
17222
17223    render() {
17224      const displayGradeLevel = gradeLevel => {
17225        switch (gradeLevel) {
17226          case 'N':
17227            return 'Nursery';
17228          case 'K1':
17229            return 'Kinder 1';
17230          case 'K2':
17231            return 'Kinder 2';
17232          case 'G1':
17233            return 'Grade 1';
17234          case 'G2':
17235            return 'Grade 2';
17236          case 'G3':
17237            return 'Grade 3';
17238          case 'G4':
17239            return 'Grade 4';
17240          case 'G5':
17241            return 'Grade 5';
17242          case 'G6':
17243            return 'Grade 6';
17244          case 'G7':
17245            return 'Grade 7';
17246          case 'G8':
17247            return 'Grade 8';
17248          case 'G9':
17249            return 'Grade 9';
17250          case 'G10':
17251            return 'Grade 10';
17252          case 'G11':
17253            return 'Grade 11';
17254          case 'G12':
17255            return 'Grade 12';
17256          default:
17257            return '';
17258        }
17259      };
17260      const DisplayData = [];
17261      for (const [index, value] of this.state.data.entries()) {
17262        DisplayData.push(
17263          <Table.Row>
17264            <Table.Col>{value.subjectCode}</Table.Col>
17265            <Table.Col>{value.subjectName}</Table.Col>
17266            <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
17267            <Table.Col>{value.sectionName}</Table.Col>

```

```

17268 <Table.Col>
17269   <Link to={`/assignsubjectload/viewload/${this.props.id}/
17270     edit/${value.key}`}>
17271     <Button pill size="sm" icon="edit" color="primary"></
17272       Button>
17273     </Link>
17274     <Popconfirm
17275       title="Do you want to remove this subject load?"
17276       onConfirm={() => this.deleteSubjectSection(value.key)}
17277       okText="Delete"
17278       cancelText="Cancel"
17279     >
17280       <Button pill size="sm" icon="trash" color="danger"></
17281         Button>
17282     </Popconfirm>
17283   </Table.Col>
17284 </Table.Row>,
17285 );
17286 }
17287 return (
17288   <div className="app-registrar-view-subject-load my-3 my-md-5">
17289     <Container>
17290       <Grid.Row>
17291         <Grid.Col sm={12} lg={4}>
17292           {this.state.isLoading ? (
17293             <Spin spinning={this.state.isLoading}>
17294               <Profile name="" avatarURL={placeholder}
17295                 backgroundImage=""></Profile>
17296             </Spin>
17297           ) : (
17298             <Profile
17299               name={this.state.name}
17300               avatarURL={
17301                 this.state.imageUrl === 'NA' ? placeholder :
17302                   getImageUrl(this.state.imageUrl)
17303               }
17304               backgroundURL={bg}
17305             ></Profile>
17306           )}
17307         </Grid.Col>
17308         <Grid.Col sm={12} lg={8}>
17309           <Grid.Row>
17310             <Card>
17311               <Card.Body>
17312                 {this.state.isLoading ? (
17313                   ,
17314                 ) : (
17315                   <div>
17316                     <Card.Title>
17317                       <Breadcrumb>
17318                         <Breadcrumb.Item>Teachers</Breadcrumb.Item>
17319                         <Breadcrumb.Item>Assign Subject Load</
17320                           Breadcrumb.Item>
17321                         <Breadcrumb.Item>View Load</Breadcrumb.Item>
17322                       >
17323                         <Breadcrumb.Item>{this.state.name}</
17324                           Breadcrumb.Item>
17325                       </Breadcrumb>
17326                     </Card.Title>
17327                     <Card.Title>
17328                       Subject Load of {this.state.name} S.Y. {this.
17329                         state.schoolYear}
17330                     </Card.Title>
17331                   </div>
17332                 )
17333               <Grid.Row>
17334                 <Grid.Col sm={12} md={12} xs={12}>
17335                   <Spin spinning={this.state.isLoadingTable}>
17336                     <Table highlightRowOnHover={true} responsive
17337                       ={true}>
17338                       <Table.Header>
17339                         <Table.ColHeader>Subject Code</Table.
                           ColHeader>

```

```

17330             <Table.ColHeader>Subject Name</Table.
17331                     ColHeader>
17332             <Table.ColHeader>Grade Level</Table.
17333                     ColHeader>
17334             <Table.ColHeader>Section</Table.ColHeader
17335                     >
17336             <Table.ColHeader>Actions</Table.ColHeader
17337                     >
17338     </Table.Header>
17339     <Table.Body>
17340         {DisplayData.length == 0 ? (
17341             <Table.Row>
17342                 <Table.Col colSpan={5} alignContent="center">
17343                     No entries.
17344                 </Table.Col>
17345             </Table.Row>
17346         ) : (
17347             DisplayData
17348         )}
17349     </Table.Body>
17350 </Table>
17351     <Pagination
17352         size="large"
17353         current={this.state.page}
17354         pageSize={this.state.pageSize}
17355         total={this.state.pageSize * this.state.
17356             numOfPages}
17357         onChange={this.paginate}
17358     />
17359     </Spin>
17360     </Grid.Col>
17361     </Grid.Row>
17362     <Card.Body>
17363     </Card>
17364 </Grid.Row>
17365 <Grid.Row>
17366     {this.state.isLoading ? (
17367         <Spin spinning={true}>
17368             <Card></Card>
17369         </Spin>
17370     ) : (
17371         <RegistrarAddNewLoad
17372             schoolYearID={this.state.schoolYearID}
17373             accountID={this.props.id}
17374         />
17375     )}
17376     </Grid.Row>
17377     </Grid.Col>
17378     </Grid.Row>
17379     </Container>
17380     </div>
17381   );
17382 }
17383 }
17384 */
17385 function mapStateToProps(state) {
17386   return {
17387     app: state.app,
17388   };
17389 }
17390 */
17391 function mapDispatchToProps(dispatch) {
17392   return {
17393     actions: bindActionCreators({ ...actions }, dispatch),
17394   };
17395 }
17396 */
17397 export default connect(mapStateToProps, mapDispatchToProps)(
17398   RegistrarViewSubjectLoad);

```

```

17395 import React, { Component } from 'react';
17396 import PropTypes from 'prop-types';
17397 import { bindActionCreators } from 'redux';
17398 import { connect } from 'react-redux';
17399 import * as actions from './redux/actions';
17400 import { Card, Button, Grid, Avatar, Table, Form, Alert } from 'tabler-react';
17401 import { Pagination, Spin, Popconfirm, Modal, message } from 'antd';
17402 import { getImageUrl } from '../../utils';
17403 import placeholder from '../../images/placeholder.jpg';
17404 import axios from 'axios';
17405
17406 export class RestrictAccount extends Component {
17407   static propTypes = {
17408     app: PropTypes.object.isRequired,
17409     actions: PropTypes.object.isRequired,
17410   };
17411
17412   constructor(props) {
17413     super(props);
17414     this.state = {
17415       isLoading: true,
17416       page: 1,
17417       numOfPages: 1,
17418       pageSize: 10,
17419       keyword: '',
17420       data: [],
17421       showModal: false,
17422       message: '',
17423       selectedID: -1,
17424     };
17425   }
17426
17427   handleUnrestrict = () => {
17428     Modal.confirm({
17429       title: 'Unrestrict Account',
17430       content: 'Do you want to unrestrict this account?',
17431       okText: 'Unrestrict',
17432       cancelText: 'Cancel',
17433       onCancel: () => this.setState({ selectedID: -1 }),
17434       onOk: () => this.props.actions.unrestrictAccount({ accountID:
17435         this.state.selectedID }),
17436     });
17437   }
17438
17439   handleOnSearch = () => {
17440     this.setState({ isLoading: true });
17441     axios
17442       .post('api/cashier/getallstudents', {
17443         page: this.state.page,
17444         pageSize: this.state.pageSize,
17445         keyword: this.state.keyword,
17446       })
17447       .then(res => {
17448         this.setState({
17449           isLoading: false,
17450           data: res.data.accountList,
17451           numOfPages: res.data.numOfPages,
17452         });
17453       })
17454       .catch(err => {
17455         this.setState({ isLoading: false, data: [], numOfPages: 1 });
17456       });
17457   }
17458
17459   paginate = page => {
17460     this.setState({ page, isLoading: true }, () => {
17461       axios
17462         .post('api/cashier/getallstudents', {
17463           page: this.state.page,
17464           pageSize: this.state.pageSize,
17465           keyword: this.state.keyword,
17466         })
17467         .then(res => {

```

```

17467     this.setState({
17468       isLoading: false,
17469       data: res.data.accountList,
17470       numOfPages: res.data.numOfPages,
17471     });
17472   })
17473   .catch(err => {
17474     this.setState({ isLoading: false, data: [] , numOfPages: 1 });
17475   });
17476 );
17477 }
17478
17479 componentDidMount() {
17480   axios
17481     .post('api/cashier/getallstudents', {
17482       page: this.state.page,
17483       pageSize: this.state.pageSize,
17484       keyword: this.state.keyword,
17485     })
17486     .then(res => {
17487       this.setState({
17488         isLoading: false,
17489         data: res.data.accountList,
17490         numOfPages: res.data.numOfPages,
17491       });
17492     })
17493     .catch(err => {
17494       this.setState({ isLoading: false, data: [] , numOfPages: 1 });
17495     });
17496 }
17497
17498 render() {
17499   const DisplayData = [];
17500   for (const [index, value] of this.state.data.entries()) {
17501     DisplayData.push(
17502       <Table.Row>
17503         <Table.Col className="w-1">
17504           <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
17505             getImageUrl(value.imageUrl)} />
17506         <Table.Col>{value.name}</Table.Col>
17507         <Table.Col>{value.email}</Table.Col>
17508         <Table.Col>
17509           {value.isActive ? (
17510             <Button
17511               size="sm"
17512               icon="lock"
17513               color="danger"
17514               onClick={() => this.setState({ selectedID: value.key ,
17515                 showModal: true })}
17516               pill
17517             ></Button>
17518           ) : (
17519             <Button
17520               size="sm"
17521               icon="unlock"
17522               color="success"
17523               onClick={() =>
17524                 this.setState({ selectedID: value.key }, () => this.
17525                   handleUnrestrict())
17526               }
17527               pill
17528             ></Button>
17529           )}
17530         </Table.Col>
17531       );
17532     return (
17533       <React.Fragment>
17534         <Modal
17535           title="Restrict Account"
17536           visible={this.state.showModal}

```

```

17537     onOk={() => {
17538       this.props.actions.restrictAccount({
17539         accountID: this.state.selectedID,
17540         message: this.state.message,
17541       });
17542     }}
17543     onCancel={() => this.setState({ showModal: false, selectedID:
17544       -1 })}
17545     cancelText="Close"
17546     okText="Restrict Account"
17547   >
17548     <Grid.Row>
17549       <Grid.Col sm={12} xs={12} md={12}>
17550         <Alert icon="info" type="danger">
17551           Do you want to restrict this account? Leave a message
17552             for the student.
17553         </Alert>
17554         <Form.Group>
17555           <Form.Input
17556             placeholder="Enter message here"
17557             value={this.state.message}
17558             onChange={e => this.setState({ message: e.target.
17559               value })}>
17560             />
17561           </Form.Group>
17562         </Grid.Col>
17563       </Grid.Row>
17564     </Modal>
17565     <Card statusColor="warning">
17566       <Card.Body>
17567         <Card.Title>Restrict an Account</Card.Title>
17568         <Card.Body>
17569           <Grid.Row>
17570             <Grid.Col xs={12} md={12} sm={12}>
17571               <Form.Group>
17572                 <Form.Label>Search Student</Form.Label>
17573                 <Form.Input
17574                   value={this.state.keyword}
17575                     onChange={e =>
17576                       this.setState({ keyword: e.target.value }), ()
17577                         => this.handleOnSearch())
17578                     }
17579                     placeholder="Search for...">
17580                   />
17581                 </Form.Group>
17582               </Grid.Col>
17583             <Grid.Col sm={12} xs={12} md={12}>
17584               <Spin spinning={this.state.isLoading}>
17585                 <Table highlightRowOnHover={true} responsive={true
17586                   }>
17587                   <Table.Header>
17588                     <Table.ColHeader></Table.ColHeader>
17589                     <Table.ColHeader>Name</Table.ColHeader>
17590                     <Table.ColHeader>Email</Table.ColHeader>
17591                     <Table.ColHeader>Actions</Table.ColHeader>
17592                   </Table.Header>
17593                   <Table.Body>
17594                     {DisplayData.length == 0 ? (
17595                       <Table.Row>
17596                         <Table.Col colSpan={4} alignContent="center
17597                           ">
17598                             No entries found.
17599                           </Table.Col>
17600                         </Table.Row>
17601                     ) : (
17602                       DisplayData
17603                     )}
17604                   </Table.Body>
17605                 </Table>
17606                 <Pagination
17607                   size="large"
17608                   current={this.state.page}
17609                   pageSize={this.state.pageSize}>

```

```

17604             total={this.state.pageSize * this.state.
17605                 numOfPages}
17606             onChange={this.paginate}
17607         />
17608     </Spin>
17609     </Grid.Col>
17610   </Grid.Row>
17611   </Card.Body>
17612   </Card>
17613 </React.Fragment>
17614 );
17615 }
17616 }
17617 */
17618 /* istanbul ignore next */
17619 function mapStateToProps(state) {
17620   return {
17621     app: state.app,
17622   };
17623 }
17624 */
17625 /* istanbul ignore next */
17626 function mapDispatchToProps(dispatch) {
17627   return {
17628     actions: bindActionCreators({ ...actions }, dispatch),
17629   };
17630 }
17631 */
17632 export default connect(mapStateToProps, mapDispatchToProps)(
17633   RestrictAccount);
17634 import React, { Component } from 'react';
17635 import PropTypes from 'prop-types';
17636 import { bindActionCreators } from 'redux';
17637 import { connect } from 'react-redux';
17638 import * as actions from './redux/actions';
17639 import { Card, Container, Grid, Button, Form } from 'tabler-react';
17640 import { Spin, Typography, Modal, Descriptions } from 'antd';
17641 import axios from 'axios';
17642 export class SchoolYearInfo extends Component {
17643   static propTypes = {
17644     app: PropTypes.object.isRequired,
17645     actions: PropTypes.object.isRequired,
17646   };
17647   constructor(props) {
17648     super(props);
17649     this.state = {
17650       isLoading: false,
17651       schoolYear: '',
17652       schoolYearID: 0,
17653       quarter: 'Q1',
17654       hasActiveSY: false,
17655       syInput: '',
17656       showModal: false,
17657     };
17658     this.onChange = this.onChange.bind(this);
17659     this.showModal = this.showModal.bind(this);
17660   }
17661   */
17662   showModal = () => {
17663     Modal.confirm({
17664       title: 'End School Year',
17665       content: 'Do you want to end this school year?',
17666       okText: 'End',
17667       cancelText: 'Cancel',
17668       onOk: () => {
17669         this.props.actions.endSchoolYear({ schoolYearID: this.state.
17670           schoolYearID });
17671       },
17672       onCancel: () => {},
17673     });

```

```

17674     };
17675
17676     onChange = e => {
17677       this.setState({ [e.target.name]: e.target.value });
17678     };
17679
17680     componentDidMount() {
17681       this.setState({ isLoading: true }, () => {
17682         axios
17683           .get('api/registrar/getsy')
17684           .then(res => {
17685             this.setState({
17686               isLoading: false,
17687               schoolYear: res.data.schoolYear,
17688               schoolYearID: res.data.schoolYearID,
17689               quarter: res.data.quarter,
17690               hasActiveSY: true,
17691             });
17692           })
17693           .catch(err => {
17694             this.setState({ isLoading: false, hasActiveSY: false });
17695           });
17696         });
17697       }
17698
17699     componentWillReceiveProps() {
17700       this.setState({ isLoading: true }, () => {
17701         axios
17702           .get('api/registrar/getsy')
17703           .then(res => {
17704             this.setState({
17705               isLoading: false,
17706               schoolYear: res.data.schoolYear,
17707               schoolYearID: res.data.schoolYearID,
17708               quarter: res.data.quarter,
17709               hasActiveSY: true,
17710             });
17711           })
17712           .catch(err => {
17713             this.setState({ isLoading: false, hasActiveSY: false });
17714           });
17715         });
17716       }
17717
17718     render() {
17719       return (
17720         <Container>
17721           <Modal
17722             title="Create a New School Year"
17723             visible={this.state.showModal}
17724             onOk={() => this.props.actions.createSchoolYear({ schoolYear:
17725               this.state.syInput })}
17726             onCancel={() => this.setState({ showModal: false })}
17727             okText="Create"
17728             cancelText="Close"
17729           >
17730             <Grid.Col>
17731               <Grid.Row>
17732                 <Grid.Col sm={12} xs={12} md={12}>
17733                   <Form.Group>
17734                     <Form.Label>Enter school year:</Form.Label>
17735                     <Form.Input
17736                       placeholder="Enter school year"
17737                       value={this.state.syInput}
17738                       name="syInput"
17739                       onChange={this.onChange}
17740                     />
17741                     Format must be '{<year>-<year+1> (e.g. 2001-2002)'}
17742                   </Form.Group>
17743                 </Grid.Col>
17744               </Grid.Row>
17745             </Grid.Col>
17746           </Modal>

```

```

17746 <Spin spinning={this.state.isLoading}>
17747   <div className="app-school-year-info card">
17748     <Card.Header>
17749       <Card.Title>School Year Information</Card.Title>
17750     </Card.Header>
17751     <Card.Body>
17752       <Grid.Row>
17753         {!this.state.isLoading && !this.state.hasActiveSY && (
17754           <React.Fragment>
17755             <Grid.Col sm={12} xs={12} md={12}>
17756               <Typography.Title level={3} style={{ textAlign: 'center' }}>
17757                 There is no active school year.
17758               </Typography.Title>
17759             </Grid.Col>
17760             <Grid.Col sm={12} xs={12} md={12}>
17761               <Button.List align="center">
17762                 <Button
17763                   icon="add"
17764                     pill
17765                     outline
17766                     color="primary"
17767                     onClick={() => this.setState({ showModal: true })}>
17768                   >
17769                     Create a New School Year
17770                   </Button>
17771                 </Button.List>
17772               </Grid.Col>
17773             </React.Fragment>
17774         )}>
17775         {!this.state.isLoading && this.state.hasActiveSY && (
17776           <React.Fragment>
17777             <Grid.Col sm={12} xs={12} md={12}>
17778               <Descriptions style={{ marginBottom: '15px', marginTop: '15px' }} bordered>
17779                 <Descriptions.Item span={3} label="Current School Year">
17780                   <b>{this.state.schoolYear}</b>
17781                 </Descriptions.Item>
17782                 <Descriptions.Item span={3} label="Current Quarter">
17783                   <Form.Select
17784                     style={{ marginBottom: '10px' }}
17785                     onChange={e => this.setState({ quarter: e.target.value })}
17786                     value={this.state.quarter}>
17787                     >
17788                       <option>Q1</option>
17789                       <option>Q2</option>
17790                       <option>Q3</option>
17791                       <option>Q4</option>
17792                     </Form.Select>
17793
17794               <span style={{ marginLeft: '10px', marginTop: '10px' }}>
17795                 <Button
17796                   style={{ marginTop: '10px' }}
17797                   icon="edit"
17798                   size="sm"
17799                   onClick={() => {
17800                     this.props.actions.changeQuarterSy({
17801                       schoolYearID: this.state.schoolYearID
17802                         ,
17803                         quarter: this.state.quarter,
17804                       });
17805                     }
17806                     color="primary"
17807                     outline
17808                     block
17809                   >
17810                     Change
17811                   </Button>
17812                 </span>

```

```

17812                     </Descriptions.Item>
17813                 </Descriptions>
17814                 <Button.List align="right">
17815                     <Button color="danger" size="sm" onClick={this.
17816                         showModal}>
17817                         End School Year
17818                     </Button>
17819                 </Button.List>
17820             </Grid.Col>
17821         </React.Fragment>
17822     )
17823     </Grid.Row>
17824   </Card.Body>
17825 </div>
17826   <Spin>
17827 </Container>
17828 );
17829 }
17830
17831 /* istanbul ignore next */
17832 function mapStateToProps(state) {
17833   return {
17834     app: state.app,
17835   };
17836 }
17837
17838 /* istanbul ignore next */
17839 function mapDispatchToProps(dispatch) {
17840   return {
17841     actions: bindActionCreators({ ...actions }, dispatch),
17842   };
17843 }
17844
17845 export default connect(mapStateToProps, mapDispatchToProps)(
17846   SchoolYearInfo);
17847 import React, { Component } from 'react';
17848 import PropTypes from 'prop-types';
17849 import { bindActionCreators } from 'redux';
17850 import { connect } from 'react-redux';
17851 import * as actions from './redux/actions';
17852 import { Container, Grid, Card, Button, Form, Header } from 'tabler-
17853   react';
17854 import cn from 'classnames';
17855 import placeholder from '../..../images/placeholder.jpg';
17856 import { Spin } from 'antd';
17857 import bg from '../..../images/BG.png';
17858 import { getImageUrl } from '../..../utils';
17859 function ProfileImage({ avatarURL }) {
17860   return <img className="card-profile-img" alt="Profile" src={avatarURL
17861     } />;
17862 }
17863
17864 function Profile({
17865   className,
17866   children,
17867   name,
17868   avatarURL = '',
17869   twitterURL = '',
17870   backgroundURL = '',
17871   bio,
17872 }) {
17873   const classes = cn('card-profile', className);
17874   return (
17875     <Card className={classes}>
17876       <Card.Header backgroundURL={backgroundURL} />
17877       <Card.Body className="text-center">
17878         <ProfileImage avatarURL={avatarURL} />
17879         <Header.H3 className="mb-3">{name}</Header.H3>
17880         <p className="mb-4">{bio || children}</p>
17881       </Card.Body>
17882     </Card>

```

```

17880      );
17881  }
17882
17883  export class StudentDashboard extends Component {
17884    static propTypes = {
17885      app: PropTypes.object.isRequired,
17886      actions: PropTypes.object.isRequired,
17887    };
17888
17889  constructor(props) {
17890    super(props);
17891    this.state = {
17892      isLoading: false,
17893      email: '',
17894      position: '',
17895      firstName: '',
17896      lastName: '',
17897      middleName: '',
17898      suffix: '',
17899      nickname: '',
17900      imageUrl: '',
17901      contactNum: '',
17902      address: '',
17903      province: '',
17904      city: '',
17905      region: '',
17906      zipcode: '',
17907      civilStatus: '',
17908      sex: '',
17909      citizenship: '',
17910      birthDate: 0,
17911      birthPlace: '',
17912      religion: '',
17913      emergencyName: '',
17914      emergencyAddress: '',
17915      emergencyTelephone: '',
17916      emergencyCellphone: '',
17917      emergencyEmail: '',
17918      emergencyRelationship: '',
17919    };
17920  }
17921
17922  componentWillReceiveProps(nextProps) {
17923    this.setState({
17924      email: nextProps.app.auth.user.email,
17925      position: nextProps.app.auth.user.position,
17926      firstName: nextProps.app.profile.firstName,
17927      lastName: nextProps.app.profile.lastName,
17928      middleName: nextProps.app.profile.middleName,
17929      suffix: nextProps.app.profile.suffix,
17930      nickname: nextProps.app.profile.nickname,
17931      imageUrl: nextProps.app.profile.imageUrl,
17932      contactNum: nextProps.app.profile.contactNum,
17933      address: nextProps.app.profile.address,
17934      province: nextProps.app.profile.province,
17935      city: nextProps.app.profile.city,
17936      region: nextProps.app.profile.region,
17937      zipcode: nextProps.app.profile.zipcode,
17938      civilStatus: nextProps.app.profile.civilStatus,
17939      sex: nextProps.app.profile.sex,
17940      citizenship: nextProps.app.profile.citizenship,
17941      birthDate: nextProps.app.profile.birthDate,
17942      birthPlace: nextProps.app.profile.birthPlace,
17943      religion: nextProps.app.profile.religion,
17944      emergencyName: nextProps.app.profile.emergencyName,
17945      emergencyAddress: nextProps.app.profile.emergencyAddress,
17946      emergencyTelephone: nextProps.app.profile.emergencyTelephone,
17947      emergencyCellphone: nextProps.app.profile.emergencyCellphone,
17948      emergencyEmail: nextProps.app.profile.emergencyEmail,
17949      emergencyRelationship: nextProps.app.profile.
17950        emergencyRelationship,
17951    });
17952  }
17953  componentDidMount() {
17954    if (Object.keys(this.props.app.profile).length !== 0) {
17955      this.setState({

```

```

17956   email: this.props.app.auth.user.email,
17957   position: this.props.app.auth.user.position,
17958   firstName: this.props.app.profile.firstName,
17959   lastName: this.props.app.profile.lastName,
17960   middleName: this.props.app.profile.middleName,
17961   suffix: this.props.app.profile.suffix,
17962   nickname: this.props.app.profile.nickname,
17963   imageUrl: this.props.app.profile.imageUrl,
17964   contactNum: this.props.app.profile.contactNum,
17965   address: this.props.app.profile.address,
17966   province: this.props.app.profile.province,
17967   city: this.props.app.profile.city,
17968   region: this.props.app.profile.region,
17969   zipcode: this.props.app.profile.zipcode,
17970   civilStatus: this.props.app.profile.civilStatus,
17971   sex: this.props.app.profile.sex,
17972   citizenship: this.props.app.profile.citizenship,
17973   birthDate: this.props.app.profile.birthDate,
17974   birthPlace: this.props.app.profile.birthPlace,
17975   religion: this.props.app.profile.religion,
17976   emergencyName: this.props.app.profile.emergencyName,
17977   emergencyAddress: this.props.app.profile.emergencyAddress,
17978   emergencyTelephone: this.props.app.profile.emergencyTelephone,
17979   emergencyCellphone: this.props.app.profile.emergencyCellphone,
17980   emergencyEmail: this.props.app.profile.emergencyEmail,
17981   emergencyRelationship: this.props.app.profile.
17982           emergencyRelationship,
17983     );
17984   }
17985 }
17986 render() {
17987   const {
17988     email,
17989     position,
17990     firstName,
17991     lastName,
17992     middleName,
17993     suffix,
17994     nickname,
17995     imageUrl,
17996     contactNum,
17997     address,
17998     province,
17999     city,
18000     region,
18001     zipcode,
18002     civilStatus,
18003     sex,
18004     citizenship,
18005     birthPlace,
18006     religion,
18007     emergencyName,
18008     emergencyAddress,
18009     emergencyTelephone,
18010     emergencyCellphone,
18011     emergencyEmail,
18012     emergencyRelationship,
18013   } = this.state;
18014   const birthDate = new Date(this.state.birthDate);
18015   const { errors } = this.props.app;
18016   const uploadLoading = false;
18017   const displayPosition = position => {
18018     switch (position) {
18019       case false:
18020         return 'Administrator';
18021       case true:
18022         return 'Director';
18023       case 2:
18024         return 'Registrar';
18025       case 3:
18026         return 'Teacher';
18027       case 4:
18028         return 'Student';
18029       case 5:
18030         return 'Guardian';
18031       case 6:
18032         return 'Cashier';

```

```

18033     default:
18034         return '';
18035     }
18036   };
18037
18038   const getPHTime = (date = '') => {
18039     const d = date === '' ? new Date() : new Date(date);
18040     const localOffset = d.getTimezoneOffset() * 60000;
18041     const UTC = d.getTime() + localOffset;
18042     const PHT = UTC + 3600000 * 16;
18043     return new Date(PHT);
18044   };
18045   const capitalize = string => {
18046     return string.charAt(0).toUpperCase() + string.slice(1).
18047      toLowerCase();
18048   };
18049   return (
18050     <div className="app-teacher-dashboard my-3 my-md-5">
18051       <Spin spinning={this.props.app.showLoading}>
18052         <Container>
18053           <Grid.Row>
18054             <Grid.Col sm={12} lg={4}>
18055               <Profile
18056                 name={`${firstName} ${middleName.charAt(0) .
18057                  toUpperCase()} ${lastName}`}
18058                 avatarURL={imageUrl === 'NA' ? placeholder :
18059                   getImageUrl(imageUrl)}
18060                 backgroundURL={bg}
18061               >
18062                 <div>
18063                   <Header.H5>{displayPosition(position)}</Header.H5>
18064                 </div>
18065               </Profile>
18066             </Grid.Col>
18067             <Grid.Col xs={12} sm={12} lg={8}>
18068               <Form className="card" onSubmit={this.onSubmit}>
18069                 <Card.Body>
18070                   <Card.Title>Account Information</Card.Title>
18071                   <Grid.Row>
18072                     <Grid.Col xs={12} sm={12} md={6}>
18073                       <Form.Group>
18074                         <Form.Label>Email</Form.Label>
18075                         <Form.Input type="email" disabled placeholder
18076                           ="Email" value={email} />
18077                       </Form.Group>
18078                     </Grid.Col>
18079                     <Grid.Col sm={12} md={6}>
18080                       <Form.Group>
18081                         <Form.Label>Position</Form.Label>
18082                         <Form.Input
18083                           type="text"
18084                           placeholder="Position"
18085                           disabled
18086                           value={displayPosition(position)}>
18087                         </Form.Input>
18088                       </Form.Group>
18089                     </Grid.Col>
18090                     <Grid.Col sm={12} md={4}>
18091                       <Form.Group>
18092                         <Form.Label>First Name</Form.Label>
18093                         <Form.Input
18094                           disabled
18095                           type="text"
18096                           name="firstName"
18097                           placeholder="First Name"
18098                           value={firstName}>
18099                         </Form.Input>
18100                       </Form.Group>
18101                     </Grid.Col>

```

```

18102          disabled
18103          type="text"
18104          placeholder="Middle Name"
18105          value={middleName}
18106          name="middleName"
18107      />
18108  </Form.Group>
18109  </Grid.Col>
18110  <Grid.Col sm={12} md={4}>
18111      <Form.Group>
18112          <Form.Label>Last Name</Form.Label>
18113          <Form.Input
18114              type="text"
18115              disabled
18116              placeholder="Last Name"
18117              value={lastName}
18118              name="lastName"
18119      />
18120  </Form.Group>
18121  </Grid.Col>
18122  <Grid.Col sm={12} md={3}>
18123      <Form.Group>
18124          <Form.Label>Nickname</Form.Label>
18125          <Form.Input
18126              type="text"
18127              disabled
18128              placeholder="Nickname"
18129              value={nickname}
18130              name="nickname"
18131      />
18132  </Form.Group>
18133  </Grid.Col>
18134  <Grid.Col sm={12} md={3}>
18135      <Form.Group>
18136          <Form.Label>Suffix</Form.Label>
18137          <Form.Input
18138              type="text"
18139              disabled
18140              placeholder="Suffix"
18141              value={suffix}
18142              name="suffix"
18143      />
18144  </Form.Group>
18145  </Grid.Col>
18146  <Grid.Col sm={12} md={6}>
18147      <Form.Group>
18148          <Form.Label>Contact Number</Form.Label>
18149          <Form.Input
18150              type="text"
18151              disabled
18152              placeholder="Contact Number"
18153              value={contactNum}
18154              name="contactNum"
18155      />
18156  </Form.Group>
18157  </Grid.Col>
18158  <Grid.Col sm={12} md={12}>
18159      <Form.Group>
18160          <Form.Label>Address</Form.Label>
18161          <Form.Input
18162              type="text"
18163              disabled
18164              placeholder="Home Address"
18165              value={address}
18166              name="address"
18167      />
18168  </Form.Group>
18169  </Grid.Col>
18170  <Grid.Col sm={12} md={4}>
18171      <Form.Group>
18172          <Form.Label>Province</Form.Label>
18173          <Form.Input
18174              type="text"
18175              disabled

```

```

18176          placeholder="Province"
18177          value={province}
18178          name="province"
18179      />
18180    </Form.Group>
18181  </Grid.Col>
18182  <Grid.Col sm={12} md={4}>
18183    <Form.Group>
18184      <Form.Label>City</Form.Label>
18185      <Form.Input
18186        type="text"
18187        disabled
18188        placeholder="City"
18189        value={city}
18190        name="city"
18191      />
18192    </Form.Group>
18193  </Grid.Col>
18194  <Grid.Col sm={12} md={2}>
18195    <Form.Group>
18196      <Form.Label>Region</Form.Label>
18197      <Form.Input
18198        type="text"
18199        disabled
18200        placeholder="Region"
18201        value={region}
18202        name="region"
18203      />
18204    </Form.Group>
18205  </Grid.Col>
18206  <Grid.Col sm={12} md={2}>
18207    <Form.Group>
18208      <Form.Label>Zipcode</Form.Label>
18209      <Form.Input
18210        type="text"
18211        disabled
18212        placeholder="Postal Code"
18213        value={zipcode}
18214        name="zipcode"
18215      />
18216    </Form.Group>
18217  </Grid.Col>
18218  <Grid.Col sm={12} md={4}>
18219    <Form.Group>
18220      <Form.Label>Civil Status</Form.Label>
18221      <Form.Input
18222        type="text"
18223        disabled
18224        placeholder="Civil Status"
18225        value={civilStatus}
18226        name="civilStatus"
18227      />
18228    </Form.Group>
18229  </Grid.Col>
18230  <Grid.Col sm={12} md={2}>
18231    <Form.Group>
18232      <Form.Label>Sex</Form.Label>
18233      <Form.Input
18234        type="text"
18235        disabled
18236        placeholder="Sex"
18237        value={sex}
18238        name="sex"
18239      />
18240    </Form.Group>
18241  </Grid.Col>
18242  <Grid.Col sm={12} md={6}>
18243    <Form.Group>
18244      <Form.Label>Citizenship</Form.Label>
18245      <Form.Input
18246        disabled
18247        type="text"
18248        placeholder="Citizenship"
18249        value={citizenship}

```

```

18250          name="citizenship"
18251      />
18252  </Form.Group>
18253 </Grid.Col>
18254 <Grid.Col sm={12} md={4}>
18255   <Form.Group>
18256     <Form.Label>Birth Date</Form.Label>
18257     <Form.Input
18258       disabled
18259       type="text"
18260       placeholder="Birth date"
18261       value={birthDate}
18262     ></Form.Input>
18263   </Form.Group>
18264 </Grid.Col>
18265 <Grid.Col sm={12} md={4}>
18266   <Form.Group>
18267     <Form.Label>Birth Place</Form.Label>
18268     <Form.Input
18269       disabled
18270       type="text"
18271       placeholder="Birth Place"
18272       value={birthPlace}
18273       name="birthPlace"
18274     />
18275   </Form.Group>
18276 </Grid.Col>
18277 <Grid.Col sm={12} md={4}>
18278   <Form.Group>
18279     <Form.Label>Religion</Form.Label>
18280     <Form.Input
18281       type="text"
18282       disabled
18283       placeholder="Religion"
18284       value={religion}
18285       name="religion"
18286     />
18287   </Form.Group>
18288 </Grid.Col>
18289 </Grid.Row>
18290 </Card.Body>
18291 <Card.Body>
18292   <Card.Title>Emergency Contact Information</Card.
18293     Title>
18294   <Grid.Row>
18295     <Grid.Col sm={12} md={6}>
18296       <Form.Group>
18297         <Form.Label>Contact Person</Form.Label>
18298         <Form.Input
18299           type="text"
18300           disabled
18301           placeholder="Contact Person"
18302           value={emergencyName}
18303           name="emergencyName"
18304         />
18305       </Form.Group>
18306     </Grid.Col>
18307     <Grid.Col sm={12} md={6}>
18308       <Form.Group>
18309         <Form.Label>Contact Address</Form.Label>
18310         <Form.Input
18311           type="text"
18312           disabled
18313           placeholder="Contact Address"
18314           value={emergencyAddress}
18315           name="emergencyAddress"
18316         />
18317       </Form.Group>
18318     </Grid.Col>
18319     <Grid.Col sm={12} md={6}>
18320       <Form.Group>
18321         <Form.Label>Contact Telephone No.</Form.Label

```

```

18322                     type="text"
18323                     disabled
18324                     placeholder="Contact Telephone No."
18325                     value={emergencyTelephone}
18326                     name="emergencyTelephone"
18327                 />
18328             </Form.Group>
18329         </Grid.Col>
18330     <Grid.Col sm={12} md={6}>
18331         <Form.Group>
18332             <Form.Label>Contact Cellphone No.</Form.Label>
18333             >
18334             <Form.Input
18335                 type="text"
18336                 disabled
18337                 placeholder="Contact Cellphone No."
18338                 value={emergencyCellphone}
18339                 name="emergencyCellphone"
18340             />
18341             </Form.Group>
18342         </Grid.Col>
18343     <Grid.Col sm={12} md={6}>
18344         <Form.Group>
18345             <Form.Label>Contact Email</Form.Label>
18346             <Form.Input
18347                 disabled
18348                 type="text"
18349                 placeholder="Contact Email"
18350                 value={emergencyEmail}
18351                 name="emergencyEmail"
18352             />
18353         </Form.Group>
18354     </Grid.Col>
18355     <Grid.Col sm={12} md={6}>
18356         <Form.Group>
18357             <Form.Label>Contact Relationship</Form.Label>
18358             <Form.Input
18359                 type="text"
18360                 disabled
18361                 placeholder="Contact Relationship"
18362                 value={emergencyRelationship}
18363                 name="emergencyRelationship"
18364             />
18365         </Form.Group>
18366     </Grid.Row>
18367     </Card.Body>
18368   </Form>
18369   </Grid.Col>
18370 </Grid.Row>
18371 </Container>
18372   </Spin>
18373 </div>
18374 );
18375 }
18376 }
18377 */
18378 /* istanbul ignore next */
18379 function mapStateToProps(state) {
18380     return {
18381         app: state.app,
18382     };
18383 }
18384 */
18385 /* istanbul ignore next */
18386 function mapDispatchToProps(dispatch) {
18387     return {
18388         actions: bindActionCreators({ ...actions }, dispatch),
18389     };
18390 }
18391 */
18392 export default connect(mapStateToProps, mapDispatchToProps)(
    StudentDashboard);

```

```

18393 import React, { Component } from 'react';
18394 import PropTypes from 'prop-types';
18395 import { bindActionCreators } from 'redux';
18396 import { connect } from 'react-redux';
18397 import * as actions from './redux/actions';
18398 import NavBar from './NavBar';
18399 import { Result, Descriptions } from 'antd';
18400 import axios from 'axios';
18401 import { Link } from 'react-router-dom';
18402 import { Spin, Breadcrumb, Pagination } from 'antd';
18403 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
18404     from 'tabler-react';
18405 import placeholder from '../../../../../images/placeholder.jpg';
18406 import { getImageUrl } from '../../../../../utils';
18407
18408 export class StudentViewGrade extends Component {
18409     static propTypes = {
18410         app: PropTypes.object.isRequired,
18411         actions: PropTypes.object.isRequired,
18412     };
18413
18414     constructor(props) {
18415         super(props);
18416         this.state = {
18417             locked: false,
18418             isLoading: true,
18419             isLoading2: true,
18420             isLoadingTable: true,
18421             selectedSchoolYearID: -1,
18422             selectedQuarter: 'Q1',
18423             schoolYearOptions: [],
18424             data: [],
18425             finalGrade: -1,
18426             section: '',
18427             teacher: '',
18428         };
18429     }
18430
18431     handleChange = () => {
18432         this.setState({ isLoadingTable: true });
18433         axios
18434             .post('api/student/studentfinalrecord', {
18435                 schoolYearID: this.state.selectedSchoolYearID,
18436                 quarter: this.state.selectedQuarter,
18437                 parent: this.props.app.auth.user.position == 5,
18438             })
18439             .then(res3 => {
18440                 this.setState({
18441                     isLoadingTable: false,
18442                     data: res3.data.data,
18443                     finalGrade: res3.data.finalGrade,
18444                     section: res3.data.section,
18445                     teacher: res3.data.teacher,
18446                 });
18447             });
18448     }
18449
18450     componentDidMount() {
18451         this.props.actions.setLoadingTrue();
18452         if (!this.props.app.auth.isAuthenticated) {
18453             this.props.history.push('/login');
18454         } else {
18455             if (this.props.app.auth.user.position != 4 && this.props.app.auth
18456                 .user.position != 5) {
18457                 this.props.history.push('/page401');
18458             } else {
18459                 axios
18460                     .get('api/student/gettsy')
18461                     .then(res => {
18462                         this.props.actions.setLoadingFalse();
18463                         this.setState({
18464                             locked: false,
18465                             isLoading: false,
18466                         });
18467                     });
18468             }
18469         }
18470     }

```

```

18465 axios.get('api/student/getallsy').then(res2 => {
18466     this.setState({
18467         isLoading2: false,
18468         schoolYearOptions: res2.data.schoolYearList,
18469         selectedSchoolYearID: res2.data.schoolYearList[0].schoolYearID,
18470     });
18471     axios
18472         .post('api/student/studentfinalrecord', {
18473             schoolYearID: res2.data.schoolYearList[0].schoolYearID,
18474             quarter: res.data.quarter,
18475             parent: this.props.app.auth.user.position == 5,
18476         })
18477         .then(res3 => {
18478             this.setState({
18479                 isLoadingTable: false,
18480                 data: res3.data.data,
18481                 finalGrade: res3.data.finalGrade,
18482                 section: res3.data.section,
18483                 teacher: res3.data.teacher,
18484             });
18485         });
18486     });
18487 }
18488 .catch(err => {
18489     this.props.actions.setLoadingFalse();
18490     this.setState({ locked: true, isLoading: false });
18491 });
18492 }
18493 }
18494 }
18495
18496 render() {
18497     const DisplayData = [];
18498     if (this.props.app.auth.user.position == 4) {
18499         for (const [index, value] of this.state.data.entries()) {
18500             DisplayData.push(
18501                 <Table.Row>
18502                     <Table.Col>{value.subjectName}</Table.Col>
18503                     <Table.Col>
18504                         {(value.score != -1 || value.score != 'N/A') && (
18505                             <span
18506                                 className={'status-icon bg-$\{parseFloat(value.score)
18507                                     >= 75 ? 'green' : 'red'\}'}
18508                             />
18509                         )}
18510                         {value.score == -1 ? 'Not yet available' : value.score}
18511                     </Table.Col>
18512                 </Table.Row>,
18513             );
18514         }
18515     }
18516     return (
18517         <div className="app-student-view-grade">
18518             {' '}
18519             {this.state.isLoading ? (
18520                 ''
18521             ) : this.state.locked ? (
18522                 <NavBar>
18523                     <Container>
18524                         <Grid.Row>
18525                             <Grid.Col xs={12} sm={12} md={12}>
18526                                 <Result
18527                                     status="403"
18528                                     title="No Active School Year"
18529                                     subTitle="Sorry, you are not authorized to access
18530                                         this page now. Please contact your system
18531                                         administrator for details."
18532                                     extra={
18533                                         <Link to="/">
18534                                             <Button color="primary">Back Home</Button>
18535                                         </Link>

```

```

18533
18534
18535
18536
18537
18538
18539
18540
18541
18542
18543
18544
18545
18546
18547
18548
18549
18550
18551
18552
18553
18554
18555
18556
18557
18558
18559
18560
18561
18562
18563
18564
18565
18566
18567
18568
18569
18570
18571
18572
18573
18574
18575
18576
18577
18578
18579
18580
18581
18582
18583
18584
18585
18586
18587
18588
18589
18590
18591
18592
18593
18594
}
      />
    </Grid.Col>
  </Grid.Row>
</Container>
<NavBar>
) : (
<NavBar>
  <Container>
    <Grid.Row>
      <Grid.Col xs={12} sm={12} md={12}>
        <Card statusColor="success" className="my-3 my-md-5 card">
          <Spin spinning={this.state.isLoading2}>
            {' '}
          <Card.Body>
            <Card.Title>Student Grades List</Card.Title>
            <Grid.Row>
              <Grid.Col sm={12} xs={12} md={6}>
                <Form.Group>
                  <Form.Label>Select School Year</Form.Label>
                  <Form.Select
                    onChange={e =>
                      this.setState({ selectedSchoolYearID: e.target.value }, () =>
                      this.handleChange(),
                    )
                  }
                >
                  {this.state.schoolYearOptions.map(a =>
                    (
                      <option value={a.schoolYearID}>{a.schoolYear}</option>
                    )));
                  </Form.Select>
                </Form.Group>
              </Grid.Col>
              <Grid.Col sm={12} xs={12} md={6}>
                <Form.Group>
                  <Form.Label>Select Quarter</Form.Label>
                  <Form.Select
                    onChange={e =>
                      this.setState({ selectedQuarter: e.target.value }, () =>
                      this.handleChange(),
                    )
                  }
                >
                  <option value="Q1">Quarter 1</option>
                  <option value="Q2">Quarter 2</option>
                  <option value="Q3">Quarter 3</option>
                  <option value="Q4">Quarter 4</option>
                </Form.Select>
              </Form.Group>
            </Grid.Col>
            <Grid.Col sm={12} xs={12} md={12}>
              {this.props.app.auth.user.position == 4 &&
              (
                <Spin spinning={this.state.isLoadingTable}>
              )>
              <Grid.Row>
                <Grid.Col xs={12} md={12} sm={12}>
                  <Descriptions
                    style={{ marginBottom: '15px',
                      marginTop: '15px' }}
                    bordered
                    title="Student Information"
                  >
                    <Descriptions.Item label="Student Name" span={3}>
                      {this.props.app.profile.

```

```

18595           firstName}{' '}
18596           {this.props.app.profile.
18597             middleName}{' '}
18598           {this.props.app.profile.
18599             lastName}
18600         </Descriptions.Item>
18601       <Descriptions.Item label="Section
18602         Name" span={3}>
18603       {this.state.section}
18604     </Descriptions.Item>
18605   <Descriptions.Item label="Adviser
18606         " span={3}>
18607   {this.state.teacher}
18608 </Descriptions.Item>
18609 <Descriptions.Item label="Quarter
18610   Final Grade" span={3}>
18611   {this.state.finalGrade}
18612 </Descriptions.Item>
18613 </Grid.Col>
18614 <Table highlightRowOnHover={true}
18615   responsive={true}>
18616     <Table.Header>
18617       <Table.ColHeader>Subject Name</
18618       Table.ColHeader>
18619       <Table.ColHeader>Grade</Table.
18620         ColHeader>
18621       </Table.Header>
18622     <Table.Body>
18623       {DisplayData.length == 0 ? (
18624         <Table.Row>
18625           <Table.Col colSpan={2}
18626             alignContent="center">
18627               No entries.
18628           </Table.Col>
18629         </Table.Row>
18630       ) : (
18631         DisplayData
18632       )}
18633     </Table.Body>
18634   </Table>
18635 </Grid.Row>
18636 </Grid>
18637 <Card.Body>
18638 </Spin>
18639 </Card>
18640 <this.props.app.auth.user.position == 5 && (
18641   <Spin spinning={this.state.isLoadingTable}>
18642     {this.state.data.map(value =>
18643       <Card statusColor="success">
18644         <Card.Body>
18645           <Grid.Row>
18646             <Grid.Col xs={12} md={12} sm={12}>
18647               <Descriptions
18648                 style={{ marginBottom: '15px',
18649                   marginTop: '15px' }}
18650                 bordered
18651                 title="Student Information"
18652               >
18653                 <Descriptions.Item label="Student
18654                   Name" span={3}>
18655                   {value.name}
18656                 </Descriptions.Item>
18657                 <Descriptions.Item label="Section
18658                   Name" span={3}>
18659                   {value.section}
18660                 </Descriptions.Item>
18661               <Descriptions.Item label="Adviser"
18662                 span={3}>
18663                 {value.teacher}

```

```

18654
18655      </Descriptions.Item>
18656      <Descriptions.Item label="Quarter
18657          Final Grade" span={3}>
18658          {value.finalGrade}
18659      </Descriptions.Item>
18660      </Grid.Col>
18661      <Table highlightRowOnHover={true}>
18662          responsive={true}>
18663          <Table.Header>
18664              <Table.ColHeader>Subject Name</Table.
18665                  ColHeader>
18666              <Table.ColHeader>Grade</Table.
18667                  ColHeader>
18668          </Table.Header>
18669          <Table.Body>
18670              {value.data.length == 0 ? (
18671                  <Table.Row>
18672                      <Table.Col colSpan={2}
18673                          alignContent="center">
18674                          No entries.
18675                      </Table.Col>
18676                  </Table.Row>
18677          ) : (
18678              value.data.map(v => (
18679                  <Table.Row>
18680                      <Table.Col>{v.subjectName}</
18681                          Table.Col>
18682                      <Table.Col>
18683                          {(v.score != -1 || v.score !=
18684                              'N/A') && (
18685                              <span
18686                                  className={'status-icon'
18687                                      bg-${(
18688                                          parseFloat(v.score) >=
18689                                              75 ? 'green' : 'red
18690                                      ,
18691                                      )})
18692                                  )}
18693                      </Table.Col>
18694                  </Table.Row>
18695          ))
18696      </Table.Body>
18697      </Table>
18698      </Grid.Row>
18699      </Container>
18700      </NavBar>
18701      </div>
18702  );
18703);
18704);
18705);
18706;
18707 /* istanbul ignore next */
18708 function mapStateToProps(state) {
18709     return {
18710         app: state.app,
18711     };
18712 }
18713;
18714 /* istanbul ignore next */

```

```

18715     function mapDispatchToProps(dispatch) {
18716       return {
18717         actions: bindActionCreators({ ...actions }, dispatch),
18718       };
18719     }
18720   }
18721   export default connect(mapStateToProps, mapDispatchToProps)(  

18722     StudentViewGrade);
18723   import React, { Component } from 'react';
18724   import PropTypes from 'prop-types';
18725   import { bindActionCreators } from 'redux';
18726   import { connect } from 'react-redux';
18727   import * as actions from './redux/actions';
18728   import axios from 'axios';
18729   import { Pagination, Spin, Tooltip } from 'antd';
18730   import {
18731     Modal,
18732     Popconfirm,
18733     Search,
18734     Breadcrumb,
18735     AutoComplete,
18736     Input,
18737     message,
18738     Descriptions,
18739     Popover,
18740   } from 'antd';
18741   import cn from 'classnames';
18742   import placeholder from '../../../../../images/placeholder.jpg';
18743   import {
18744     Card,
18745     Button,
18746     Grid,
18747     Avatar,
18748     Table,
18749     Form,
18750     Header,
18751     Container,
18752     Text,
18753     Alert,
18754   } from 'tabler-react';
18755   import { Link } from 'react-router-dom';
18756   import moment from 'moment';
18757   import { getImageUrl } from '../../../../../utils';
18758   const { Option } = AutoComplete;
18759
18760   export class TeacherAddRecord extends Component {
18761     static propTypes = {
18762       app: PropTypes.object.isRequired,
18763       actions: PropTypes.object.isRequired,
18764     };
18765     constructor(props) {
18766       super(props);
18767       this.state = {
18768         componentID: 0,
18769         component: '',
18770         subcompName: '',
18771         subcompID: 0,
18772         isLoading: true,
18773         quarter: 'Q1',
18774         sectionName: '',
18775         subjectName: '',
18776         schoolYear: '',
18777         subjectCode: '',
18778         studList: [],
18779         itemDesc: '',
18780         dateGiven: new Date(),
18781         total: 0,
18782       };
18783       this.addNewRecord = this.addNewRecord.bind(this);
18784     }
18785
18786     addNewRecord() {
18787       this.props.actions.addNewRecord(
18788         {

```

```

18789     componentID: this.state.componentID,
18790     subcompID: this.state.subcompID,
18791     payload: this.state.studList,
18792     dateGiven: this.state.dateGiven,
18793     description: this.state.itemDesc,
18794     total: this.state.total,
18795     subsectID: this.props.subsectID,
18796     quarter: this.props.quarter,
18797   },
18798   'Teacher',
18799 );
18800 }
18801
18802 componentDidMount() {
18803   this.setState({ isLoading: true });
18804   axios
18805     .post('api/teacher/getcomponents', {
18806       subsectID: this.props.subsectID,
18807       quarter: this.props.quarter,
18808     })
18809     .then(res => {
18810       this.setState({
18811         sectionName: res.data.sectionName,
18812         subjectName: res.data.subjectName,
18813         schoolYear: res.data.schoolYear,
18814         subjectCode: res.data.subjectCode,
18815       });
18816       axios
18817         .post('api/teacher/subcompinfo', {
18818           subsectID: this.props.subsectID,
18819           componentID: this.props.componentID,
18820           quarter: this.props.quarter,
18821           subcompID: this.props.subcompID,
18822         })
18823         .then(res2 => {
18824           let studList = [];
18825           for (const [index, value] of res2.data.data.entries()) {
18826             const { subsectstudID, name, imageUrl } = value;
18827             studList.push({ subsectstudID, name, score: 0, imageUrl });
18828           }
18829           this.setState({
18830             isLoading: false,
18831             componentID: this.props.componentID,
18832             component: res2.data.component,
18833             subcompName: res2.data.subcompName,
18834             subcompID: this.props.subcompID,
18835             quarter: this.props.quarter,
18836             studList,
18837           });
18838         });
18839       });
18840     });
18841   }
18842   render() {
18843     const birthDate = new Date(this.state.dateGiven);
18844     const defaultDate = birthDate.toISOString();
18845     let tableData = [];
18846     for (const [index, value] of this.state.studList.entries()) {
18847       tableData.push(
18848         <Table.Row>
18849           <Table.Col className="w-1">
18850             <Avatar imageURL={value.imageUrl == 'NA' ? placeholder : getImageUrl(value.imageUrl)} />
18851           </Table.Col>
18852           <Table.Col>{value.name}</Table.Col>
18853           <Table.Col>
18854             <Form.Group>
18855               <Form.Input
18856                 placeholder="Score"
18857                 position="append"
18858                 value={value.score}
18859                 onChange={e => {
18860                   const reg = /^-?[0-9]*(\.[0-9]*)?$/;

```

```

18861     let temparr = this.state.studList;
18862
18863     if (
18864       (!isNaN(e.target.value) && reg.test(e.target.value)
18865         ) ||
18866         (e.target.value === '' && e.target.value !== '-')
18867         ||
18868         e.target.value == 'A' ||
18869         e.target.value == 'E'
18870     ) {
18871       temparr[index].score = e.target.value;
18872       this.setState({ studList: temparr });
18873     }
18874   
```

/

```

18875   </Form.Group>
18876   </Table.Col>
18877   </Table.Row>,
18878 );
18879
18880 return (
18881   <div className="app-teacher-add-record my-3 my-md-5">
18882     <Container>
18883       <Grid.Row>
18884         <Card>
18885           <Card.Body>
18886             {this.state.isLoading ? (
18887               <Spin spinning={true}></Spin>
18888             ) : (
18889               <div>
18890                 <Card.Title>
18891                   <Breadcrumb>
18892                     <Breadcrumb.Item>Subjects</Breadcrumb.Item>
18893                     <Breadcrumb.Item>View Subject Load</Breadcrumb.
18894                       Item>
18895                     <Breadcrumb.Item>Manage Grades</Breadcrumb.Item
18896                       >
18897                     <Breadcrumb.Item>
18898                       {this.state.sectionName} - {this.state.
18899                         subjectName}
18900                     </Breadcrumb.Item>
18901                     <Breadcrumb.Item>{this.state.component}</
18902                       Breadcrumb.Item>
18903                     <Breadcrumb.Item>{this.state.subcompName}</
18904                       Breadcrumb.Item>
18905                     </Breadcrumb>
18906           <Card.Title>
18907             <Header.H3>
18908               {this.state.component} - {this.state.
18909                 subcompName} - Add New Record
18910               </Header.H3>
18911             <Card.Title>
18912               <Text.Small>
18913                 {this.state.sectionName} S.Y. {this.state.
18914                   schoolYear}
18915                 </Text.Small>
18916               </Card.Title>
18917             </div>
18918           )
18919         </Card.Body>
18920       </Card>
18921     </Grid.Row>
18922     <Grid.Row>
18923       <Grid.Col sm={12} xs={12} md={6}>
18924         <Card>
18925           <Card.Body>
18926             <Grid.Row>
18927               <Grid.Col sm={12} xs={12} md={12}>
18928                 <Descriptions
18929                   style={{ marginBottom: '15px', marginTop: '15px
18930                     , }}>
```

```

18924     bordered
18925     title="Add a New Record"
18926   >
18927     <Descriptions.Item span={3} label="Item
18928       Description">
18929       <Form.Group>
18930         <Form.Input
18931           placeholder="Description"
18932           value={this.state.itemDesc}
18933           onChange={e => {
18934             this.setState({ itemDesc: e.target.
18935               value });
18936           }
18937         />
18938       </Form.Group>
18939     </Descriptions.Item>
18940     <Descriptions.Item span={3} label="Date Given">
18941       <Form.Group>
18942         <Form.DatePicker
18943           defaultDate={new Date(defaultDate)}
18944           format="mm/dd/yyyy"
18945           onChange={e => {
18946             this.setState({ dateGiven: e });
18947           }
18948           name="birthDate"
18949           maxYear={2020}
18950           minYear={1897}
18951           monthLabels={[
18952             'January',
18953             'February',
18954             'March',
18955             'April',
18956             'May',
18957             'June',
18958             'July',
18959             'August',
18960             'September',
18961             'October',
18962             'November',
18963             'December',
18964           ]}
18965         ></Form.DatePicker>
18966       </Form.Group>
18967     </Descriptions.Item>
18968     <Descriptions.Item span={3} label="Total number
18969       of items">
18970       <Form.Group>
18971         <Form.Input
18972           placeholder="Total"
18973           value={this.state.total}
18974           onChange={e => {
18975             const reg = /^-?[0-9]*(\.[0-9]*)?$/;
18976             if (
18977               (!isNaN(e.target.value) && reg.test(e
18978                 .target.value)) ||
18979                 e.target.value === ',' ||

18980               e.target.value === '-',
18981             ) {
18982               this.setState({ total: e.target.value
18983                 });
18984             }
18985           }
18986         >
18987       </Form.Group>
18988     </Descriptions.Item>
18989   </Descriptions>
18990 </Grid.Col>
18991 </Grid.Row>
18992 </Card.Body>
18993 </Card>
18994 </Grid.Col>
18995 <Grid.Col sm={12} xs={12} md={6}>

```

```

18992         <Card>
18993             <Card.Body>
18994                 <Card.Title>Student Scores</Card.Title>
18995                 <Container>
18996                     <Alert type="primary">
18997                         <b>Note:</b> Scores with value of <b>E</b> are
18998                             considered{' '}
18999                         <i>
19000                             <b>excused</b>
19001                         </i>
19002                             . Enter a value of <b>A</b> for students who are{
19003                                 , ,}
19004                         <i>
19005                             <b>absent</b>
19006                         </i>
19007                     .
19008                 </Alert>
19009             </Container>
19010             <Table highlightRowOnHover={true} responsive={true}>
19011                 <Table.Header>
19012                     <Table.ColHeader colSpan={2}>Student Name</Table.
19013                         ColHeader>
19014                     <Table.ColHeader>Score</Table.ColHeader>
19015                 </Table.Header>
19016                 <Table.Body>{tableData}</Table.Body>
19017             </Table>
19018         </Card.Body>
19019     <Card.Footer>
19020         <Button.List align="right">
19021             <Button icon="plus" color="success" onClick={this.
19022                 addNewRecord}>
19023                     Add New Record
19024                 </Button>
19025             </Button.List>
19026         </Card.Footer>
19027     </Card>
19028 );
19029 }
19030 }
19031 */
19032 /* istanbul ignore next */
19033 function mapStateToProps(state) {
19034     return {
19035         app: state.app,
19036     };
19037 }
19038 */
19039 /* istanbul ignore next */
19040 function mapDispatchToProps(dispatch) {
19041     return {
19042         actions: bindActionCreators({ ...actions }, dispatch),
19043     };
19044 }
19045 */
19046 export default connect(mapStateToProps, mapDispatchToProps)(
    TeacherAddRecord);
19047 import React, { Component } from 'react';
19048 import PropTypes from 'prop-types';
19049 import { bindActionCreators } from 'redux';
19050 import { connect } from 'react-redux';
19051 import * as actions from './redux/actions';
19052 import axios from 'axios';
19053 import { Pagination, Spin, Tooltip } from 'antd';
19054 import {
19055     Modal,
19056     Popconfirm,
19057     Search,
19058     Breadcrumb,
19059     AutoComplete,

```

```

19060     Input ,
19061     message ,
19062     Descriptions ,
19063   } from 'antd';
19064   import cn from 'classnames';
19065   import placeholder from '../../../../../images/placeholder.jpg';
19066   import {
19067     Card ,
19068     Button ,
19069     Grid ,
19070     Avatar ,
19071     Table ,
19072     Form ,
19073     Header ,
19074     Container ,
19075     Text ,
19076     Alert ,
19077     Tag ,
19078     Badge ,
19079   } from 'tabler-react';
19080   import { Link } from 'react-router-dom';
19081   import { getImageUrl } from '../../../../../utils';
19082   import ViewEditLog from './ViewEditLog';
19083   const { Option } = AutoComplete;
19084
19085   export class TeacherSummaryPerQuarter extends Component {
19086     static propTypes = {
19087       app: PropTypes.object.isRequired ,
19088       actions: PropTypes.object.isRequired ,
19089     };
19090
19091   constructor(props) {
19092     super(props);
19093     this.state = {
19094       isLoading: true ,
19095       subjectName: '' ,
19096       subjectCode: '' ,
19097       sectionName: '' ,
19098       schoolYearID: 0 ,
19099       schoolYear: '' ,
19100       data: [] ,
19101       quarter: 'Q1' ,
19102       subsectID: 0 ,
19103       transmutation: '50' ,
19104       trans: '50' ,
19105       subjectType: '' ,
19106       locked: true ,
19107       classRecordID: -1 ,
19108     };
19109
19110     this.changeTransmutation = this.changeTransmutation.bind(this);
19111   }
19112
19113   changeTransmutation() {
19114     this.props.actions.changeTransmutation(
19115       {
19116         subsectID: this.state.subsectID ,
19117         quarter: this.state.quarter ,
19118         transmutation: this.state.transmutation ,
19119       },
19120       'Teacher' ,
19121     );
19122   }
19123
19124   componentWillReceiveProps() {
19125     this.setState({ isLoading: true });
19126     axios.post('api/teacher/getsubjecttype', { subsectID: this.props.
19127       subsectID }).then(res2 => {
19128       axios
19129         .post('api/teacher/getquartersummary', {
19130           subsectID: this.props.subsectID ,
19131           quarter: this.props.quarter ,
19132         })
19133         .then(res => {
19134           axios

```

```

19134     .post('api/teacher/ifclassrecordlocked', {
19135       classRecordID: res.data.classRecordID,
19136       quarter: this.props.quarter,
19137     })
19138     .then(res3 => {
19139       this.setState({
19140         isLoading: false,
19141         subjectName: res.data.subjectName,
19142         subjectCode: res.data.subjectCode,
19143         sectionName: res.data.sectionName,
19144         schoolYearID: res.data.schoolYearID,
19145         schoolYear: res.data.schoolYear,
19146         data: res.data.data,
19147         quarter: this.props.quarter,
19148         subsectID: this.props.subsectID,
19149         transmutation: res.data.transmutation,
19150         trans: res.data.transmutation,
19151         subjectType: res2.data.subjectType,
19152         locked: res3.data.locked,
19153         classRecordID: res.data.classRecordID,
19154       });
19155     });
19156   });
19157 );
19158 );
19159 }
19160
19161   componentDidMount() {
19162     this.setState({ isLoading: true });
19163     axios.post('api/teacher/getsubjecttype', { subsectID: this.props.
19164       subsectID }).then(res2 => {
19165       axios
19166         .post('api/teacher/getquartersummary', {
19167           subsectID: this.props.subsectID,
19168           quarter: this.props.quarter,
19169         })
19170         .then(res => {
19171           axios
19172             .post('api/teacher/ifclassrecordlocked', {
19173               classRecordID: res.data.classRecordID,
19174               quarter: this.props.quarter,
19175             })
19176             .then(res3 => {
19177               this.setState({
19178                 isLoading: false,
19179                 subjectName: res.data.subjectName,
19180                 subjectCode: res.data.subjectCode,
19181                 sectionName: res.data.sectionName,
19182                 schoolYearID: res.data.schoolYearID,
19183                 schoolYear: res.data.schoolYear,
19184                 data: res.data.data,
19185                 quarter: this.props.quarter,
19186                 subsectID: this.props.subsectID,
19187                 transmutation: res.data.transmutation,
19188                 trans: res.data.transmutation,
19189                 subjectType: res2.data.subjectType,
19190                 locked: res3.data.locked,
19191                 classRecordID: res.data.classRecordID,
19192               });
19193             });
19194           });
19195         });
19196       render() {
19197         const { locked } = this.state;
19198         ('green');
19199         let displayData = [];
19200         for (const [index, value] of this.state.data.entries()) {
19201           displayData.push(
19202             <Table.Row>
19203               <Table.Col className="w-1">
19204                 <Avatar imageURL={value.imageUrl == 'NA' ? placeholder :
19205                   getImageUrl(value.imageUrl)} />
19206               </Table.Col>
19207             <Table.Col>{value.name}</Table.Col>

```

```

19207    {/* <Table.Col alignContent="center">
19208      {value.faWS == -1 ? 'Not yet available' : Number(Math.round
19209        (value.faWS + 'e2') + 'e-2')}
19210    </Table.Col>
19211    <Table.Col alignContent="center">
19212      {value.wwWS == -1 ? 'Not yet available' : Number(Math.round
19213        (value.wwWS + 'e2') + 'e-2')}
19214    </Table.Col>
19215    <Table.Col alignContent="center">
19216      {value.ptWS == -1 ? 'Not yet available' : Number(Math.round
19217        (value.ptWS + 'e2') + 'e-2')}
19218    </Table.Col> */
19219    <Table.Col alignContent="center">
19220      <b>
19221        {value.actualGrade == -1
19222          ? 'Not yet available'
19223            : Number(Math.round(value.actualGrade + 'e2') + 'e-2')}
19224      </b>
19225    </Table.Col>
19226    <Table.Col alignContent="center">
19227      {value.transmutedGrade50 == -1
19228        ? 'Not yet available'
19229          : Number(Math.round(value.transmutedGrade50 + 'e2') + 'e
19230            -2')}
19231    </Table.Col>
19232    <Table.Col alignContent="center">
19233      {value.transmutedGrade55 == -1
19234        ? 'Not yet available'
19235          : Number(Math.round(value.transmutedGrade55 + 'e2') + 'e
19236            -2')}
19237    </Table.Col>
19238    <Table.Col alignContent="center">
19239      {value.transmutedGrade60 == -1
19240        ? 'Not yet available'
19241          : Number(Math.round(value.transmutedGrade60 + 'e2') + 'e
19242            -2')}
19243    </Table.Col>
19244    <Table.Col style={{ color: 'red' }} alignContent="center">
19245      <b>
19246        <Text color={value.finalGrade < 75 ? 'red' : 'black'}>
19247          <span className={'status-icon bg-${
19248            value.finalGrade <
19249              75 ? 'red' : 'green'}' />
19250        {value.finalGrade == -1
19251          ? 'Not yet available'
19252            : Number(Math.round(value.finalGrade + 'e2') + 'e-2')}
19253      </Text>
19254    </b>
19255  </Table.Col>
19256  </Table.Row>,
```

);

}

return (

```

19257    <div className="app-teacher-summary-per-quarter my-3 my-md-5">
19258      <Card>
19259        <Card.Body>
19260          {this.state.isLoading ?
19261            <Spin spinning={true}></Spin>
19262          ) :
19263            <div>
19264              <Card.Title>
19265                <Grid.Row>
19266                  <Grid.Col xs={12} sm={12} md={6}>
19267                    <Breadcrumb>
19268                      <Breadcrumb.Item>Subjects</Breadcrumb.Item>
19269                      <Breadcrumb.Item>View Subject Load</Breadcrumb.
19270                        Item>
19271                      <Breadcrumb.Item>Summary Report</Breadcrumb.
19272                        Item>
```

```

19269
19270          <Breadcrumb.Item>
19271              {this.state.sectionName} - {this.state.
19272                  subjectName}
19273          </Breadcrumb.Item>
19274      </Grid.Col>
19275      <Grid.Col xs={12} sm={12} md={6}>
19276          <Button.List align="right">
19277              <Link to={'/summaryreportall/${this.state.
19278                  subsectID}'>
19279                  <Button icon="file" color="success">
19280                      View Summary Report
19281                  </Button>
19282          </Link>
19283      </Button.List>
19284  </Grid.Col>
19285  </Grid.Row>
19286  <Card.Title>
19287      <Card.Title>
19288          <Header.H3>
19289              {this.state.subjectCode} - {this.state.subjectName}
19290          </Header.H3>
19291  <Card.Title>
19292      <Text.Small>
19293          {this.state.sectionName}{' '}
19294          {this.state.subjectType == 'NON_SHS'
19295              ? this.props.quarter
19296                  : this.props.quarter == 'Q1' || this.props.
19297                      quarter == 'Q2',
19298                  ? 'FIRST SEMESTER'
19299                      : 'SECOND SEMESTER'}{' '}
19300          S.Y. {this.state.schoolYear}
19301  </Text.Small>
19302  <Grid.Row>
19303      <Grid.Col xs={12} sm={12} md={3}>
19304          <Form.Select
19305              onChange={e => {
19306                  this.setState({ quarter: e.target.value });
19307              }}
19308              value={this.state.quarter}
19309          >
19310              {this.state.subjectType == 'NON_SHS' ? (
19311                  <React.Fragment>
19312                      <option>Q1</option>
19313                      <option>Q2</option>
19314                      <option>Q3</option>
19315                      <option>Q4</option>
19316                  </React.Fragment>
19317              ) : (
19318                  <React.Fragment>
19319                      <option value={'Q1'}>Midterm</option>
19320                      <option value={'Q2'}>Finals</option>
19321                  </React.Fragment>
19322          )}
19323      </Form.Select>
19324  <Grid.Col xs={12} sm={12} md={3}>
19325      <a href={'/summaryreport/${this.props.subsectID}/
19326          ${this.state.quarter}'>
19327          <Button block color="primary">
19328              Change
19329          </Button>
19330      </a>
19331  <Grid.Col xs={12} sm={12} md={3}>
19332      {!locked && (
19333          <Form.Select
19334              onChange={e => {
19335                  this.setState({ transmutation: e.target.
19336                      value });
19337              }}>
```

```

19336          value={this.state.transmutation}
19337      >
19338          <option value="50">50% (Yellow)</option>
19339          <option value="55">55% (Orange)</option>
19340          <option value="60">60% (Green)</option>
19341      </Form.Select>
19342  )
19343  </Grid.Col>
19344  <Grid.Col xs={12} sm={12} md={3}>
19345    {!locked && (
19346      <Button
19347        color="primary"
19348        block
19349        onClick={() => {
19350          this.changeTransmutation();
19351        }}
19352      >
19353        Change Transmutation
19354      </Button>
19355    )}
19356  </Grid.Col>
19357  </Grid.Row>
19358  </Grid.Row>
19359  <Grid.Col xs={12} sm={12} md={6}>
19360    <Descriptions
19361      style={{ marginBottom: '15px', marginTop: '15px',
19362        , }}
19363        bordered
19364        title="Frequency Distribution"
19365    >
19366      <Descriptions.Item span={3} label="95-100">
19367        {this.state.data.length != 0 &&
19368          parseFloat(
19369            this.state.data.reduce((sum, val) => {
19370              const tempobj = JSON.parse(JSON.stringify
19371                (sum));
19372              tempobj.transmutedGrade60 =
19373                sum.transmutedGrade60 + val.
19374                  transmutedGrade60;
19375              return tempobj;
19376            }).transmutedGrade60,
19377          ) /
19378            this.state.data.length !=
19379            -1
19380            ? this.state.data.filter(
19381              val => val.finalGrade >= 95 && val.
19382                finalGrade <= 100,
19383                  ).length
19384                  : 'Not yet available'
19385            </Descriptions.Item>
19386      <Descriptions.Item span={3} label="90-94">
19387        {this.state.data.length != 0 &&
19388          parseFloat(
19389            this.state.data.reduce((sum, val) => {
19390              const tempobj = JSON.parse(JSON.stringify
19391                (sum));
19392              tempobj.transmutedGrade60 =
19393                sum.transmutedGrade60 + val.
19394                  transmutedGrade60;
19395              return tempobj;
19396            }).transmutedGrade60,
19397          ) /
19398            this.state.data.length !=
19399            -1
19400            ? this.state.data.filter(
19401              val => val.finalGrade >= 90 && val.
                finalGrade < 95,
                  ).length
19402                  : 'Not yet available'
19403            </Descriptions.Item>
19404      <Descriptions.Item span={3} label="85-89">
19405        {this.state.data.length != 0 &&
19406          parseFloat(

```

```

19402     this.state.data.reduce((sum, val) => {
19403       const tempobj = JSON.parse(JSON.stringify
19404         (sum));
19405       tempobj.transmutedGrade60 =
19406         sum.transmutedGrade60 + val.
19407           transmutedGrade60;
19408         return tempobj;
19409       }).transmutedGrade60,
19410     ) /
19411     this.state.data.length !=
19412     -1
19413     ? this.state.data.filter(
19414       val => val.finalGrade >= 85 && val.
19415         finalGrade < 90,
19416         ).length
19417         : 'Not yet available'}
19418 </Descriptions.Item>
19419 <Descriptions.Item span={3} label="80-84">
19420   {this.state.data.length != 0 &&
19421     parseFloat(
19422       this.state.data.reduce((sum, val) => {
19423         const tempobj = JSON.parse(JSON.stringify
19424           (sum));
19425         tempobj.transmutedGrade60 =
19426           sum.transmutedGrade60 + val.
19427             transmutedGrade60;
19428           return tempobj;
19429         }).transmutedGrade60,
19430       ) /
19431       this.state.data.length !=
19432       -1
19433       ? this.state.data.filter(
19434         val => val.finalGrade >= 80 && val.
19435           finalGrade < 85,
19436             ).length
19437             : 'Not yet available'}
19438 </Descriptions.Item>
19439 <Descriptions.Item span={3} label="75-79">
19440   {this.state.data.length != 0 &&
19441     parseFloat(
19442       this.state.data.reduce((sum, val) => {
19443         const tempobj = JSON.parse(JSON.stringify
19444           (sum));
19445         tempobj.transmutedGrade60 =
19446           sum.transmutedGrade60 + val.
19447             transmutedGrade60;
19448             return tempobj;
19449           }).transmutedGrade60,
19450         ) /
19451         this.state.data.length !=
19452         -1
19453         ? this.state.data.filter(
19454           val => val.finalGrade >= 75 && val.
19455             finalGrade < 80,
19456               ).length
19457               : 'Not yet available'}
19458 </Descriptions.Item>
19459 <Descriptions.Item span={3} label="Below 75">
19460   {this.state.data.length != 0 &&
19461     parseFloat(
19462       this.state.data.reduce((sum, val) => {
19463         const tempobj = JSON.parse(JSON.stringify
19464           (sum));
19465         tempobj.transmutedGrade60 =
19466           sum.transmutedGrade60 + val.
19467             transmutedGrade60;
19468             return tempobj;
19469           }).transmutedGrade60,
19470         ) /
19471         this.state.data.length !=
19472         -1
19473         ? this.state.data.filter(val => val.
19474           finalGrade < 75).length
19475         : 'Not yet available'}

```

```

19464             </Descriptions.Item>
19465         </Descriptions>
19466     </Grid.Col>
19467     <Grid.Col xs={12} sm={12} md={6}>
19468         <Descriptions
19469             style={{ marginBottom: '15px', marginTop: '15px
19470                 ,
19471             }
19472             bordered
19473             title="Average"
19474         >
19475             <Descriptions.Item span={3} label="Actual Grade
19476                 "
19477                 {this.state.data.length != 0 &&
19478                     parseFloat(
19479                         this.state.data.reduce((sum, val) => {
19480                             const tempobj = JSON.parse(JSON.stringify
19481                                 (sum));
19482                             tempobj.actualGrade = sum.actualGrade +
19483                                 val.actualGrade;
19484                             return tempobj;
19485                         }).actualGrade,
19486                     )
19487                 / this.state.data.length !=
19488                 -1
19489                 ? parseFloat(
19490                     this.state.data.reduce((sum, val) => {
19491                         const tempobj = JSON.parse(JSON.
19492                             stringify(sum));
19493                             tempobj.actualGrade = sum.actualGrade +
19494                                 val.actualGrade;
19495                             return tempobj;
19496                         }).actualGrade,
19497                     ) / this.state.data.length
19498                     : 'Not yet available'
19499             </Descriptions.Item>
19500             <Descriptions.Item span={3} label="Transmuted
19501                 Grade (50%)">
19502                 {this.state.data.length != 0 &&
19503                     parseFloat(
19504                         this.state.data.reduce((sum, val) => {
19505                             const tempobj = JSON.parse(JSON.stringify
19506                                 (sum));
19507                             tempobj.transmutedGrade50 =
19508                                 sum.transmutedGrade50 + val.
19509                                     transmutedGrade50;
19510                             return tempobj;
19511                         }).transmutedGrade50,
19512                     )
19513                 / this.state.data.length !=
19514                 -1
19515                 ? parseFloat(
19516                     this.state.data.reduce((sum, val) => {
19517                         const tempobj = JSON.parse(JSON.
19518                             stringify(sum));
19519                             tempobj.transmutedGrade50 =
19520                                 sum.transmutedGrade50 + val.
19521                                     transmutedGrade50;
19522                             return tempobj;

```

```

19523                               }) .transmutedGrade55 ,
19524             ) /
19525             this.state.data.length !=
19526             -1
19527             ? parseFloat(
19528               this.state.data.reduce((sum, val) => {
19529                 const tempobj = JSON.parse(JSON.
19530                   stringify(sum));
19531                 tempobj.transmutedGrade55 =
19532                   sum.transmutedGrade55 + val.
19533                     transmutedGrade55;
19534                     return tempobj;
19535               )).transmutedGrade55,
19536             ) / this.state.data.length
19537             : 'Not yet available'}
19538 </Descriptions.Item>
19539 <Descriptions.Item span={3} label="Transmuted
19540   Grade (60%)">
19541   {this.state.data.length != 0 &&
19542     parseFloat(
19543       this.state.data.reduce((sum, val) => {
19544         const tempobj = JSON.parse(JSON.stringify
19545           (sum));
19546         tempobj.transmutedGrade60 =
19547           sum.transmutedGrade60 + val.
19548             transmutedGrade60;
19549             return tempobj;
19550       )).transmutedGrade60,
19551     ) /
19552       this.state.data.length !=
19553       -1
19554       ? parseFloat(
19555         this.state.data.reduce((sum, val) => {
19556           const tempobj = JSON.parse(JSON.
19557             stringify(sum));
19558           tempobj.transmutedGrade60 =
19559             sum.transmutedGrade60 + val.
19560               transmutedGrade60;
19561               return tempobj;
19562             )).transmutedGrade60,
19563             ) / this.state.data.length
19564             : 'Not yet available'}
19565 </Descriptions.Item>
19566 </Descriptions>
19567 </Grid.Col>
19568 </Grid.Row>
19569 </Card.Title>
19570 </div>
19571 )}
19572 </Card.Body>
19573 <Card.Body>
19574 <Grid.Row>
19575   <Grid.Col xs={12} md={12} sm={12}>
19576     <Table responsive={true} highlightRowOnHover={true}>
19577       <Table.Header>
19578         <Table.ColHeader colSpan={2}>Student Name</Table.
19579           ColHeader>
19580           /* <Table.ColHeader alignContent="center">
19581             Formative Assessment</Table.ColHeader>
19582             <Table.ColHeader alignContent="center">Written
19583               Works</Table.ColHeader>
19584             <Table.ColHeader alignContent="center">Performance
19585               Tasks</Table.ColHeader>
19586             <Table.ColHeader alignContent="center">Quarterly
19587               Assessment</Table.ColHeader> */
19588             <Table.ColHeader alignContent="center">Actual Grade
19589             </Table.ColHeader>
19590             <Table.ColHeader alignContent="center">
19591               Transmuted Grade (50%) <Tag color="yellow">Yellow
19592               </Tag>
19593             </Table.ColHeader>
19594             <Table.ColHeader alignContent="center">
19595               Transmuted Grade (55%) <Tag color="orange">Orange

```

```

19582                     </Tag>
19583             </Table.ColHeader>
19584             <Table.ColHeader alignContent="center">
19585                 Transmuted Grade (60%) <Tag color="green">Green</
19586                 Tag>
19587             </Table.ColHeader>
19588             <Table.ColHeader alignContent="center">Final Grade
19589         </Table.Header>
19590         <Table.Body>{displayData}</Table.Body>
19591     </Grid.Col>
19592     </Grid.Row>
19593   </Card.Body>
19594   <Card.Footer>
19595     <Button.List align="right">
19596       <ViewEditLog
19597         classRecordID={this.state.classRecordID}
19598         quarter={this.state.quarter}
19599         position="Teacher"
19600       />
19601     </Button.List>
19602   </Card.Footer>
19603 </div>
19604 );
19605 }
19606 }
19607 /* istanbul ignore next */
19608 function mapStateToProps(state) {
19609   return {
19610     app: state.app,
19611   };
19612 }
19613
19614 /* istanbul ignore next */
19615 function mapDispatchToProps(dispatch) {
19616   return {
19617     actions: bindActionCreators({ ...actions }, dispatch),
19618   };
19619 }
19620
19621
19622 export default connect(mapStateToProps, mapDispatchToProps)(
19623   TeacherSummaryPerQuarter);
19624 import React, { Component } from 'react';
19625 import PropTypes from 'prop-types';
19626 import { bindActionCreators } from 'redux';
19627 import { connect } from 'react-redux';
19628 import * as actions from './redux/actions';
19629 import axios from 'axios';
19630 import { Pagination, Spin, Tooltip } from 'antd';
19631 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
19632   message } from 'antd';
19633 import RegistrarAddNewLoad from './RegistrarAddNewLoad';
19634 import cn from 'classnames';
19635 import placeholder from '../../../../../images/placeholder.jpg';
19636 import { Card, Button, Grid, Avatar, Table, Form, Header, Container }
19637   from 'tabler-react';
19638 import { Link } from 'react-router-dom';
19639 const { Option } = AutoComplete;
19640
19641
19642
19643
19644
19645
19646
19647
19648
19649
19650
19651
19652
19653
19654
19655
19656
19657
19658
19659
19660
19661
19662
19663
19664
19665
19666
19667
19668
19669
19670
19671
19672
19673
19674
19675
19676
19677
19678
19679
19680
19681
19682
19683
19684
19685
19686
19687
19688
19689
19690
19691
19692
19693
19694
19695
19696
19697
19698
19699
19700
19701
19702
19703
19704
19705
19706
19707
19708
19709
19710
19711
19712
19713
19714
19715
19716
19717
19718
19719
19720
19721
19722
19723
19724
19725
19726
19727
19728
19729
19730
19731
19732
19733
19734
19735
19736
19737
19738
19739
19740
19741
19742
19743
19744
19745
19746
19747
19748
19749
19750
19751
19752
19753
19754
19755
19756
19757
19758
19759
19760
19761
19762
19763
19764
19765
19766
19767
19768
19769
19770
19771
19772
19773
19774
19775
19776
19777
19778
19779
19780
19781
19782
19783
19784
19785
19786
19787
19788
19789
19790
19791
19792
19793
19794
19795
19796
19797
19798
19799
19800
19801
19802
19803
19804
19805
19806
19807
19808
19809
19810
19811
19812
19813
19814
19815
19816
19817
19818
19819
19820
19821
19822
19823
19824
19825
19826
19827
19828
19829
19830
19831
19832
19833
19834
19835
19836
19837
19838
19839
19840
19841
19842
19843
19844
19845
19846
19847
19848
19849
19850
19851
19852
19853
19854
19855
19856
19857
19858
19859
19860
19861
19862
19863
19864
19865
19866
19867
19868
19869
19870
19871
19872
19873
19874
19875
19876
19877
19878
19879
19880
19881
19882
19883
19884
19885
19886
19887
19888
19889
19890
19891
19892
19893
19894
19895
19896
19897
19898
19899
19900
19901
19902
19903
19904
19905
19906
19907
19908
19909
19910
19911
19912
19913
19914
19915
19916
19917
19918
19919
19920
19921
19922
19923
19924
19925
19926
19927
19928
19929
19930
19931
19932
19933
19934
19935
19936
19937
19938
19939
19940
19941
19942
19943
19944
19945
19946
19947
19948
19949
19950
19951
19952
19953
19954
19955
19956
19957
19958
19959
19960
19961
19962
19963
19964
19965
19966
19967
19968
19969
19970
19971
19972
19973
19974
19975
19976
19977
19978
19979
19980
19981
19982
19983
19984
19985
19986
19987
19988
19989
19990
19991
19992
19993
19994
19995
19996
19997
19998
19999
199999

```

```

19649         data: [],
19650         schoolYearID: 0,
19651         schoolYear: '',
19652         page: 1,
19653         pageSize: 10,
19654         numOfPages: 1,
19655         schoolYearList: [],
19656         selectedSchoolYearID: -1,
19657         selectedQuarter: 'Q1',
19658     );
19659 }
19660
19661     onDropdownChange = () => {
19662         this.setState({ isLoadingTable: true });
19663         axios
19664             .post('api/teacher/getsubjectload', {
19665                 schoolYearID: this.state.selectedSchoolYearID,
19666                 quarter: this.state.selectedQuarter,
19667                 page: this.state.page,
19668                 pageSize: this.state.pageSize,
19669             })
19670             .then(res2 => {
19671                 this.setState({
19672                     isLoadingTable: false,
19673                     data: res2.data.subjectSectionData,
19674                     numOfPages: res2.data.numOfPages,
19675                 });
19676             })
19677             .catch(err => {
19678                 this.setState({ isLoadingTable: false, data: [] });
19679             });
19680     };
19681
19682     componentDidMount() {
19683         axios.get('api/teacher/getpsy').then(res => {
19684             axios.get('api/teacher/getallsy').then(res3 => {
19685                 this.setState(
19686                     {
19687                         isLoading: false,
19688                         schoolYear: res.data.schoolYear,
19689                         schoolYearID: res.data.schoolYearID,
19690                         schoolYearList: res3.data.schoolYearList,
19691                         selectedSchoolYearID: res3.data.schoolYearList[0].schoolYearID,
19692                     },
19693                     () => {
19694                         axios
19695                             .post('api/teacher/getsubjectload', {
19696                                 schoolYearID: this.state.selectedSchoolYearID,
19697                                 quarter: this.state.selectedQuarter,
19698                                 page: this.state.page,
19699                                 pageSize: this.state.pageSize,
19700                             })
19701                             .then(res2 => {
19702                                 this.setState({
19703                                     isLoadingTable: false,
19704                                     data: res2.data.subjectSectionData,
19705                                     numOfPages: res2.data.numOfPages,
19706                                 });
19707                             })
19708                             .catch(err => {
19709                                 this.setState({ isLoadingTable: false, data: [] });
19710                             });
19711                     },
19712                     );
19713                 });
19714             });
19715         }
19716
19717         paginate = page => {
19718             this.setState({
19719                 page,
19720             });
19721             this.setState({ isLoadingTable: true });

```

```

19722     axios
19723       .post('api/teacher/getsubjectload', {
19724         page,
19725         pageSize: this.state.pageSize,
19726       })
19727       .then(res => {
19728         this.setState({ isLoadingTable: false });
19729         this.setState({
19730           numOfPages: res.data.numOfPages,
19731           data: res.data.subjectSectionData,
19732         });
19733       })
19734       .catch(err => {
19735         this.setState({ isLoadingTable: false, data: [] });
19736       });
19737     );
19738   )
19739   render() {
19740     const displayGradeLevel = gradeLevel => {
19741       switch (gradeLevel) {
19742         case 'N':
19743           return 'Nursery';
19744         case 'K1':
19745           return 'Kinder 1';
19746         case 'K2':
19747           return 'Kinder 2';
19748         case 'G1':
19749           return 'Grade 1';
19750         case 'G2':
19751           return 'Grade 2';
19752         case 'G3':
19753           return 'Grade 3';
19754         case 'G4':
19755           return 'Grade 4';
19756         case 'G5':
19757           return 'Grade 5';
19758         case 'G6':
19759           return 'Grade 6';
19760         case 'G7':
19761           return 'Grade 7';
19762         case 'G8':
19763           return 'Grade 8';
19764         case 'G9':
19765           return 'Grade 9';
19766         case 'G10':
19767           return 'Grade 10';
19768         case 'G11':
19769           return 'Grade 11';
19770         case 'G12':
19771           return 'Grade 12';
19772         default:
19773           return '';
19774       }
19775     };
19776     const DisplayData = [];
19777     for (const [index, value] of this.state.data.entries()) {
19778       DisplayData.push(
19779         <Table.Row>
19780           <Table.Col>{value.subjectCode}</Table.Col>
19781           <Table.Col>{value.subjectName}</Table.Col>
19782           <Table.Col>{displayGradeLevel(value.gradeLevel)}</Table.Col>
19783           <Table.Col>{value.sectionName}</Table.Col>
19784           <Table.Col>
19785             <Link to={`/viewsubjectload/${value.key}/${this.state.
19786               selectedQuarter}`}>
19787               <Button pill size="sm" icon="user" color="primary">
19788                 View information
19789               </Button>
19790             </Link>
19791           </Table.Col>
19792         </Table.Row>,
19793       );
19794     }
19795     return (
       <div className="app-teacher-view-subject-load my-3 my-md-5">

```

```

19796 <Container>
19797   <Grid.Row>
19798     <Grid.Col sm={12} lg={12}>
19799       <Grid.Row>
19800         <Card>
19801           <Card.Body>
19802             {this.state.isLoading ? (
19803               '',
19804             ) : (
19805               <div>
19806                 <Card.Title>
19807                   <Breadcrumb>
19808                     <Breadcrumb.Item>Subjects</Breadcrumb.Item>
19809                     <Breadcrumb.Item>View Subject Load</
19810                       Breadcrumb.Item>
19811                     </Breadcrumb>
19812                   </Card.Title>
19813                     <Card.Title>Subject Load List</Card.Title>
19814                   </div>
19815     )
19816     <Grid.Row>
19817       <Grid.Col sm={12} xs={12} md={7}>
19818         <Form.Group>
19819           <Form.Label>Select School Year</Form.Label>
19820           <Form.Select
19821             value={this.state.selectedSchoolYearID}
19822             onChange={e =>
19823               this.setState({ selectedSchoolYearID: e.
19824                 target.value }, () =>
19825                   this.onDropdownChange(),
19826                 )
19827               disabled={this.state.schoolYearList.length
19828                 == 0}
19829             >
19830               {this.state.schoolYearList.map(a => (
19831                 <option value={a.schoolYearID}>{a.
19832                   schoolYear}</option>
19833                 )))
19834             </Form.Select>
19835           </Form.Group>
19836         </Grid.Col>
19837         <Grid.Col sm={12} xs={12} md={5}>
19838           <Form.Group>
19839             <Form.Label>Select Quarter</Form.Label>
19840             <Form.Select
19841               value={this.state.selectedQuarter}
19842               onChange={e =>
19843                 this.setState({ selectedQuarter: e.target
19844                   .value }, () =>
19845                     this.onDropdownChange(),
19846                   )
19847                 }
19848             >
19849               <option value="Q1">Quarter 1</option>
19850               <option value="Q2">Quarter 2</option>
19851               <option value="Q3">Quarter 3</option>
19852               <option value="Q4">Quarter 4</option>
19853             </Form.Select>
19854           </Form.Group>
19855         </Grid.Col>
19856         <Grid.Col sm={12} md={12} xs={12}>
19857           <Spin spinning={this.state.isLoadingTable}>
19858             <Table highlightRowOnHover={true} responsive
19859               ={true}>
19860               <Table.Header>
19861                 <Table.ColHeader>Subject Code</Table.
19862                   ColHeader>
19863                 <Table.ColHeader>Subject Name</Table.
19864                   ColHeader>
19865                 <Table.ColHeader>Grade Level</Table.
19866                   ColHeader>

```

```

19859             <Table.ColHeader>Section</Table.ColHeader>
19860             >
19861             <Table.ColHeader>Actions</Table.ColHeader>
19862             >
19863         </Table.Header>
19864         <Table.Body>
19865             {DisplayData.length == 0 ? (
19866                 <Table.Row>
19867                     <Table.Col colSpan={5} alignContent="center">
19868                         No entries.
19869                     </Table.Col>
19870                 </Table.Row>
19871             ) : (
19872                 DisplayData
19873             )}
19874         </Table.Body>
19875     </Table>
19876     <Pagination
19877         size="large"
19878         current={this.state.page}
19879         pageSize={this.state.pageSize}
19880         total={this.state.pageSize * this.state.
19881             numPages}
19882         onChange={this.paginate}
19883     />
19884     </Spin>
19885     </Grid.Col>
19886     </Grid.Row>
19887     </Card.Body>
19888 </Card>
19889 </Grid.Row>
19890 </Container>
19891 </div>
19892 );
19893 }
19894
19895 /* istanbul ignore next */
19896 function mapStateToProps(state) {
19897     return {
19898         app: state.app,
19899     };
19900 }
19901
19902 /* istanbul ignore next */
19903 function mapDispatchToProps(dispatch) {
19904     return {
19905         actions: bindActionCreators({ ...actions }, dispatch),
19906     };
19907 }
19908
19909 export default connect(mapStateToProps, mapDispatchToProps)(TeacherViewSubjectLoad);
19910 import React, { Component } from 'react';
19911 import PropTypes from 'prop-types';
19912 import { bindActionCreators } from 'redux';
19913 import { connect } from 'react-redux';
19914 import * as actions from './redux/actions';
19915 import { Button, Container, Grid } from 'tabler-react';
19916 import { Modal, Table, Spin } from 'antd';
19917 import moment from 'moment';
19918 import axios from 'axios';
19919
19920 export class ViewEditLog extends Component {
19921     static propTypes = {
19922         app: PropTypes.object.isRequired,
19923         actions: PropTypes.object.isRequired,
19924     };
19925

```

```

19926     constructor(props) {
19927       super(props);
19928       this.state = {
19929         showModal: false,
19930         isLoading: false,
19931         data: [],
19932         showModal2: false,
19933         logDetails: [],
19934         selectedType: '',
19935     };
19936
19937     this.fetchActivityLog = this.fetchActivityLog.bind(this);
19938   }
19939
19940   fetchActivityLog = () => {
19941     this.setState({ isLoading: true });
19942     if (this.props.position == 'Teacher') {
19943       axios
19944         .post('api/teacher/getactivitylog', {
19945           classRecordID: this.props.classRecordID,
19946           quarter: this.props.quarter,
19947         })
19948         .then(res => {
19949           this.setState({ data: res.data.activityLog, isLoading: false
19950             });
19951         } );
19952       } else if (this.props.position == 'Registrar') {
19953         axios
19954           .post('api/registrar/getactivitylog', {
19955             classRecordID: this.props.classRecordID,
19956             quarter: this.props.quarter,
19957           })
19958           .then(res => {
19959             this.setState({ data: res.data.activityLog, isLoading: false
19960               });
19961         } );
19962     }
19963     render() {
19964       return (
19965         <React.Fragment>
19966           <Modal
19967             width={1000}
19968             title="Class Record Log"
19969             visible={this.state.showModal}
19970             footer={[
19971               <Button color="secondary" onClick={() => this.setState({
19972                 showModal: false })}>
19973                 Close
19974               </Button>,
19975             ]}
19976             onCancel={() => this.setState({ showModal: false })}
19977           >
19978             <Modal
19979               width={950}
19980               title="Log Details"
19981               visible={this.state.showModal2}
19982               footer={[
19983                 <Button color="secondary" onClick={() => this.setState({
19984                   showModal2: false })}>
19985                   Close
19986                 </Button>,
19987               ]}
19988               onCancel={() => this.setState({ showModal2: false })}
19989             >
19990               {(this.state.selectedType == 'CHANGE_STATUS' ||
19991                 this.state.selectedType == 'SUBCOMP_UPDATE' ||
19992                 this.state.selectedType == 'SUBCOMP_DELETE' ||
19993                 this.state.selectedType == 'SUBCOMP_ADD' ||
19994                 this.state.selectedType == 'TRANSMU_UPDATE') && (
19995                   <Table
19996                     columns={[{ title: 'Description', dataIndex: '

```

```

        description', render: text => text ]]}
19995    dataSource={this.state.logDetails}
19996  />
19997 )
19998 {((this.state.selectedType == 'TOTAL_UPDATE' ||
19999   this.state.selectedType == 'DESC_UPDATE') && (
20000     <Table
20001       columns={[
20002         { title: 'Description', dataIndex: 'description',
20003           render: text => text },
20004         { title: 'Component', dataIndex: 'component', render:
20005           text => text },
20006         { title: 'Subcomponent', dataIndex: 'subcomponent',
20007           render: text => text },
20008       ]}
20009     dataSource={this.state.logDetails}
20010   />
20011 )
20012 {((this.state.selectedType == 'ADD' ||
20013   this.state.selectedType == 'DELETE' ||
20014   this.state.selectedType == 'UPDATE') && (
20015     <Table
20016       columns={[
20017         { title: 'Student', dataIndex: 'student', render:
20018           text => text },
20019         { title: 'Component', dataIndex: 'component', render:
20020           text => text },
20021         { title: 'Subcomponent', dataIndex: 'subcomponent',
20022           render: text => text },
20023         { title: 'Description', dataIndex: 'description',
20024           render: text => text },
20025         {
20026           title: 'Old Value',
20027           dataIndex: 'oldValue',
20028           render: text => (text == -1 ? 'N/A' : text),
20029         },
20030         {
20031           title: 'New Value',
20032           dataIndex: 'newValue',
20033           render: text => (text == -1 ? 'DELETED' : text),
20034         },
20035       ]}
20036     dataSource={this.state.logDetails}
20037   />
20038 )
20039 </Modal>
20040 <Spin spinning={this.state.isLoading}>
20041   <Table
20042     columns={[
20043       { title: 'Name', dataIndex: 'name', render: text =>
20044         text },
20045       { title: 'Position', dataIndex: 'position', render:
20046         text => text },
20047       {
20048         title: 'Edit Type',
20049         dataIndex: 'type',
20050         render: text =>
20051           text == 'ADD'
20052             ? 'Grade/s Added'
20053             : text == 'UPDATE'
20054             ? 'Grade/s Updated'
20055             : text == 'DELETE'
20056             ? 'Grade/s Deleted'
20057             : text == 'CHANGE_STATUS'
20058             ? 'Record Status Updated'
20059             : text == 'SUBCOMP_UPDATE'
20060             ? 'Subcomponent Updated'
20061             : text == 'TRANSMU_UPDATE'
20062             ? 'Transmutation Updated'
20063             : text == 'SUBCOMP_DELETE'
20064             ? 'Subcomponent Deleted'
20065             : text == 'DESC_UPDATE'
20066             ? 'Description Updated'
20067           }
20068     ]}
20069   </Table>
20070 </Spin>

```

```

20058          : text == 'TOTAL_UPDATE',
20059          ? 'Total Items Updated',
20060          : 'Subcomponent Added',
20061      },
20062      {
20063          title: 'Date',
20064          dataIndex: 'timestamp',
20065          render: text => moment(text).format('LLL'),
20066      },
20067      {
20068          title: 'Details',
20069          dataIndex: 'logDetails',
20070          render: (logDetails, data) => (
20071              <Button
20072                  size="sm"
20073                  color="primary"
20074                  pill
20075                  outline
20076                  onClick={() =>
20077                      this.setState({ logDetails, selectedType: data.
20078                          type, showModal2: true })
20079                  }
20080              >
20081                  View
20082                  </Button>
20083          ),
20084      ],
20085      dataSource={this.state.data}
20086  />
20087  </Spin>
20088  </Modal>
20089  <Button
20090      icon="edit"
20091      outline
20092      color="primary"
20093      onClick={() =>
20094          this.setState({ showModal: true }, () => {
20095              this.fetchActivityLog();
20096          })
20097      }
20098      >
20099          View Edit Log
20100          </Button>
20101      </React.Fragment>
20102  );
20103 }
20104 }
20105 /* istanbul ignore next */
20106 function mapStateToProps(state) {
20107     return {
20108         app: state.app,
20109     };
20110 }
20111 }
20112 /* istanbul ignore next */
20113 function mapDispatchToProps(dispatch) {
20114     return {
20115         actions: bindActionCreators({ ...actions }, dispatch),
20116     };
20117 }
20118 }
20119
20120 export default connect(mapStateToProps, mapDispatchToProps)(ViewEditLog
);
20121 import React, { Component } from 'react';
20122 import PropTypes from 'prop-types';
20123 import { bindActionCreators } from 'redux';
20124 import { connect } from 'react-redux';
20125 import * as actions from './redux/actions';
20126 import { Link } from 'react-router-dom';
20127 import { Card, Button, Grid, Avatar, Table, Form, Header, Container,
Alert } from 'tabler-react';

```

```

20128 import axios from 'axios';
20129 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
20130 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
20131   message } from 'antd';
20132 import cn from 'classnames';
20133 import placeholder from '../../images/placeholder.jpg';
20134 import bg from '../../images/BG.png';
20135 import { getImageUrl } from '../../utils';
20136 const { Option } = AutoComplete;
20137
20138 export class ViewFailedStudents extends Component {
20139   static propTypes = {
20140     app: PropTypes.object.isRequired,
20141     actions: PropTypes.object.isRequired,
20142   };
20143   constructor(props) {
20144     super(props);
20145     this.state = {
20146       showModal: false,
20147       showModal2: false,
20148       isLoading: true,
20149       isLoadingTable: true,
20150       schoolYears: [],
20151       sectionOption: [],
20152       selectedSchoolYear: '',
20153       selectedQuarter: 'Q1',
20154       selectedGradeLevel: 'N',
20155       selectedSection: [],
20156       data: [],
20157       numOfFailed: 'N/A',
20158     };
20159   }
20160
20161   handleFailedStudents = () => {
20162     this.setState({ isLoading: true }, () => {
20163       axios
20164         .get('api/registrar/getallsy')
20165         .then(res => {
20166           this.setState({
20167             schoolYears: res.data.schoolYearList,
20168             selectedSchoolYear: res.data.schoolYearList[0].schoolYearID
20169             ,
20170             isLoading: false,
20171           });
20172         axios
20173           .post('api/registrar/getpassedfailed', {
20174             schoolYearID: res.data.schoolYearList[0].schoolYearID,
20175             gradeLevel: 'N',
20176             quarter: 'Q1',
20177           })
20178           .then(res2 => {
20179             this.setState({
20180               isLoadingTable: false,
20181               data: res2.data.data,
20182               numOfFailed: res2.data.failed,
20183             });
20184           })
20185           .catch(err => {
20186             this.setState({
20187               isLoadingTable: false,
20188               data: [],
20189               numOfFailed: 'N/A',
20190             });
20191           })
20192           .catch(err => {
20193             this.setState({
20194               schoolYears: [],
20195               selectedSchoolYear: -1,
20196               isLoading: false,
20197               numOfFailed: 'N/A',
20198             });

```

```

20199         });
20200     });
20201   };
20202
20203   refetch = () => {
20204     this.setState({ isLoadingTable: true });
20205     axios
20206       .post('api/registrar/getpassedfailed', {
20207         schoolYearID: this.state.selectedSchoolYear,
20208         gradeLevel: this.state.selectedGradeLevel,
20209         quarter: this.state.selectedQuarter,
20210       })
20211       .then(res2 => {
20212         this.setState({
20213           isLoadingTable: false,
20214           data: res2.data.data,
20215           numOfFailed: res2.data.failed,
20216         });
20217       })
20218       .catch(err => {
20219         this.setState({
20220           isLoadingTable: false,
20221           data: [],
20222           numOfFailed: 'N/A',
20223         });
20224       });
20225     );
20226
20227   handleSchoolYearChange = () => {
20228     this.refetch();
20229   };
20230
20231   handleGradeLevelChange = () => {
20232     this.refetch();
20233   };
20234
20235   handleQuarterChange = () => {
20236     this.refetch();
20237   };
20238
20239   render() {
20240     const displayGradeLevel = gradeLevel => {
20241       switch (gradeLevel) {
20242         case 'N':
20243           return 'Nursery';
20244         case 'K1':
20245           return 'Kinder 1';
20246         case 'K2':
20247           return 'Kinder 2';
20248         case 'G1':
20249           return 'Grade 1';
20250         case 'G2':
20251           return 'Grade 2';
20252         case 'G3':
20253           return 'Grade 3';
20254         case 'G4':
20255           return 'Grade 4';
20256         case 'G5':
20257           return 'Grade 5';
20258         case 'G6':
20259           return 'Grade 6';
20260         case 'G7':
20261           return 'Grade 7';
20262         case 'G8':
20263           return 'Grade 8';
20264         case 'G9':
20265           return 'Grade 9';
20266         case 'G10':
20267           return 'Grade 10';
20268         case 'G11':
20269           return 'Grade 11';
20270         case 'G12':
20271           return 'Grade 12';
20272         default:
20273           return '';

```

```

20274      }
20275    };
20276    const DisplayData = [];
20277    const DisplayData2 = [];
20278    for (const [index, value] of this.state.data.entries()) {
20279      DisplayData.push(
20280        <Table.Row>
20281          <Table.Col>{value.name}</Table.Col>
20282          <Table.Col>{value.numOfPassed}</Table.Col>
20283          <Table.Col>{value.numOfFailed}</Table.Col>
20284          <Table.Col>
20285            {value.failedInfo.length != 0 && (
20286              <Tooltip title="View failed subjects">
20287                <Button
20288                  icon="file"
20289                  color="danger"
20290                  size="sm"
20291                  pill
20292                  outline
20293                  onClick={() =>
20294                    this.setState({
20295                      selectedSection: JSON.parse(JSON.stringify(value.
20296                        failedInfo)),
20297                      showModal2: true,
20298                    })
20299                  }>
20300                ></Button>
20301              </Tooltip>
20302            )}>
20303              <Tooltip title="View condensed grades">
20304                <Link
20305                  to={`/viewstudentrecord/section/${value.sectionID}/sy/$
20306                    {this.state.selectedSchoolYear}/q/${this.state.
20307                      selectedQuarter}`}
20308                  target="_blank"
20309                >
20310                  <Button icon="file" pill color="success" size="sm"
20311                    outline></Button>
20312                  </Link>
20313                  </Tooltip>
20314                </Table.Col>
20315              );
20316            );
20317            for (const [index, value] of this.state.selectedSection.entries())
20318            {
20319              DisplayData2.push(
20320                <Table.Row>
20321                  <Table.Col>{value.name}</Table.Col>
20322                  <Table.Col>{value.subjectName}</Table.Col>
20323                  <Table.Col>{value.teacher}</Table.Col>
20324                  <Table.Col>{value.grade}</Table.Col>
20325                  <Table.Col>
20326                    <Link
20327                      to={`/viewstudentrecord/classrecord/${value.classRecordID
20328                        }/q/${this.state.selectedQuarter} `}
20329                      target="_blank"
20330                    >
20331                      <Button icon="eye" color="primary" size="sm">
20332                        View Class Record
20333                      </Button>
20334                    </Link>
20335                  </Table.Col>
20336                );
20337            );
20338            const schoolYearOptions = [];
20339            for (const [index, value] of this.state.schoolYears.entries()) {
20340              schoolYearOptions.push(<option value={value.schoolYearID}>{value.
20341                schoolYear}</option>);
20342            }
20343            return (
20344              <React.Fragment>

```

```

20340 <Modal
20341   title="View Section Information"
20342   visible={this.state.showModal2}
20343   footer={[<Button onClick={() => this.setState({ showModal2:
20344     false })}>Close</Button>]}
20345   onCancel={() => this.setState({ showModal2: false ,
20346     selectedSection: [] })}
20347   width={700}
20348 >
20349   <Grid.Row>
20350     <Grid.Col sm={12} xs={12} md={12}>
20351       <Table highlightRowOnHover={true} responsive={true}>
20352         <Table.Header>
20353           <Table.ColHeader>Student Name</Table.ColHeader>
20354           <Table.ColHeader>Subject</Table.ColHeader>
20355           <Table.ColHeader>Teacher</Table.ColHeader>
20356           <Table.ColHeader>Grade</Table.ColHeader>
20357           <Table.ColHeader>Action</Table.ColHeader>
20358         </Table.Header>
20359         <Table.Body>
20360           {DisplayData2.length == 0 ? (
20361             <Table.Row>
20362               <Table.Col colSpan={5} alignContent="center">
20363                 No data.
20364               </Table.Col>
20365             </Table.Row>
20366           ) : (
20367             DisplayData2
20368           )}
20369         </Table.Body>
20370       </Table>
20371     </Grid.Col>
20372   </Grid.Row>
20373 </Modal>
20374 <Modal
20375   title="View Passed/Failed Students"
20376   visible={this.state.showModal}
20377   footer={[<Button onClick={() => this.setState({ showModal:
20378     false })}>Close</Button>]}
20379   onCancel={() =>
20380     this.setState({
20381       showModal: false ,
20382       selectedQuarter: 'Q1',
20383       data: [],
20384       selectedScholYear: -1,
20385       selectedGradeLevel: 'N',
20386       numOffailed: 'N/A',
20387     })
20388   }
20389   width={700}
20390 >
20391   <Spin spinning={this.state.isLoading}>
20392     <Card statusColor="warning">
20393       <Card.Body>
20394         <Grid.Row>
20395           <Alert icon="info" type="primary">
20396             Note: It will display the number of students with
20397               passed/failed subjects per
20398               section by school year, quarter, and grade level.
20399         </Alert>
20400         <Grid.Col sm={12} xs={12} md={12}>
20401           <Form.Group>
20402             <Form.Label>Select School Year</Form.Label>
20403             <Form.Select
20404               onChange={e =>
20405                 this.setState({ selectedSchoolYear: e.target.
20406                   value }, () =>
20407                     this.handleSchoolYearChange(),
20408                   )
20409             }
20410             {schoolYearOptions}
20411           </Form.Select>

```

```

20408         </Form.Group>
20409     </Grid.Col>
20410     <Grid.Col sm={12} xs={12} md={6}>
20411         <Form.Group>
20412             <Form.Label>Select Quarter</Form.Label>
20413             <Form.Select
20414                 onChange={e =>
20415                     this.setState({ selectedQuarter: e.target.
20416                         value }, () =>
20417                         this.handleQuarterChange(),
20418                     )
20419                 }
20420             <option>Q1</option>
20421             <option>Q2</option>
20422             <option>Q3</option>
20423             <option>Q4</option>
20424         </Form.Select>
20425     </Form.Group>
20426 </Grid.Col>
20427     <Grid.Col sm={12} xs={12} md={6}>
20428         <Form.Group>
20429             <Form.Label>Grade Level</Form.Label>
20430             <Form.Select
20431                 value={this.state.selectedGradeLevel}
20432                 onChange={e =>
20433                     this.setState({ selectedGradeLevel: e.target.
20434                         value }, () =>
20435                         this.handleGradeLevelChange(),
20436                     )
20437                 }
20438             <option value="N">Nursery</option>
20439             <option value="K1">Kinder 1</option>
20440             <option value="K2">Kinder 2</option>
20441             <option value="G1">Grade 1</option>
20442             <option value="G2">Grade 2</option>
20443             <option value="G3">Grade 3</option>
20444             <option value="G4">Grade 4</option>
20445             <option value="G5">Grade 5</option>
20446             <option value="G6">Grade 6</option>
20447             <option value="G7">Grade 7</option>
20448             <option value="G8">Grade 8</option>
20449             <option value="G9">Grade 9</option>
20450             <option value="G10">Grade 10</option>
20451             <option value="G11">Grade 11</option>
20452             <option value="G12">Grade 12</option>
20453         </Form.Select>
20454     </Form.Group>
20455 </Grid.Col>
20456 </Grid.Row>
20457 <Spin spinning={this.state.isLoadingTable}>
20458     {, }
20459     <Grid.Row>
20460         <Grid.Col sm={12} xs={12} md={12}>
20461             <Table highlightRowOnHover={true} responsive={true}>
20462                 <Table.Header>
20463                     <Table.ColHeader>Section</Table.ColHeader>
20464                     <Table.ColHeader>Passed</Table.ColHeader>
20465                     <Table.ColHeader>Failed</Table.ColHeader>
20466                     <Table.ColHeader>Actions</Table.ColHeader>
20467                 </Table.Header>
20468                 <Table.Body>
20469                     {DisplayData.length == 0 ? (
20470                         <Table.Row>
20471                             <Table.Col colSpan={4} alignContent="center">
20472                                 No data.
20473                             </Table.Col>
20474                         </Table.Row>
20475                     ) : (

```

```

20476             DisplayData
20477         )}
20478     </Table.Body>
20479   </Table>
20480   </Grid.Col>
20481   </Grid.Row>
20482   </Spin>
20483 </Card.Body>
20484 <Card.Body>
20485   <Grid.Row>Number of failed students: {this.state.
20486   numOffailed}</Grid.Row>
20487 </Card.Body>
20488 </Card>
20489 </Spin>
20490 </Modal>
20491 <Button
20492   icon="file"
20493   size="sm"
20494   pill
20495   color="info"
20496   onClick={() => this.setState({ showModal: true }), () => this.
20497     handleFailedStudents()})}
20498   >
20499     Passed/Failed Students
20500   </Button>
20501 </React.Fragment>
20502 );
20503 }
20504 /* istanbul ignore next */
20505 function mapStateToProps(state) {
20506   return {
20507     app: state.app,
20508   };
20509 }
20510
20511 /* istanbul ignore next */
20512 function mapDispatchToProps(dispatch) {
20513   return {
20514     actions: bindActionCreators({ ...actions }, dispatch),
20515   };
20516 }
20517
20518 export default connect(mapStateToProps, mapDispatchToProps)(
20519   ViewFailedStudents);
20520 import React, { Component } from 'react';
20521 import PropTypes from 'prop-types';
20522 import { bindActionCreators } from 'redux';
20523 import { connect } from 'react-redux';
20524 import * as actions from './redux/actions';
20525 import { Link } from 'react-router-dom';
20526 import { Card, Button, Grid, Avatar, Table, Form, Header, Container,
20527   Alert } from 'tabler-react';
20528 import axios from 'axios';
20529 import { Pagination, Spin, Tooltip, Descriptions } from 'antd';
20530 import { Modal, Popconfirm, Search, Breadcrumb, AutoComplete, Input,
20531   message } from 'antd';
20532 import cn from 'classnames';
20533 import placeholder from '../../../../../images/placeholder.jpg';
20534 import bg from '../../../../../images/BG.png';
20535 import { getImageUrl } from '../../../../../utils';
20536 const { Option } = AutoComplete;
20537
20538 export class ViewHonorStudents extends Component {
20539   static propTypes = {
20540     app: PropTypes.object.isRequired,
20541     actions: PropTypes.object.isRequired,
20542   };
20543   constructor(props) {
20544     super(props);

```

```

20543     this.state = {
20544       showModal: false,
20545       showModal2: false,
20546       isLoading: true,
20547       isLoadingTable: true,
20548       schoolYears: [],
20549       sectionOption: [],
20550       selectedSchoolYear: '',
20551       selectedGradeLevel: 'N',
20552       selectedSection: [],
20553       data: [],
20554       numOfHonors: 'N/A',
20555       honorInfo: []
20556     );
20557   }
20558
20559   refetch = () => {
20560     this.setState({ isLoadingTable: true });
20561     axios
20562       .post('api/registrar/gethonorstudents', {
20563         schoolYearID: this.state.selectedSchoolYear,
20564         gradeLevel: this.state.selectedGradeLevel,
20565       })
20566       .then(res2 => {
20567         this.setState({
20568           isLoadingTable: false,
20569           data: res2.data.data,
20570           numOfHonors: res2.data.numOfHonors,
20571         });
20572       })
20573       .catch(err => {
20574         this.setState({
20575           isLoadingTable: false,
20576           data: [],
20577           numOfHonors: 'N/A',
20578         });
20579       });
20580   };
20581
20582   handleSchoolYearChange = () => {
20583     this.refetch();
20584   };
20585
20586   handleGradeLevelChange = () => {
20587     this.refetch();
20588   };
20589
20590   handleFetchHonorStudents = () => {
20591     this.setState({ isLoading: true }, () => {
20592       axios
20593         .get('api/registrar/getallsy')
20594         .then(res => {
20595           this.setState({
20596             schoolYears: res.data.schoolYearList,
20597             selectedSchoolYear: res.data.schoolYearList[0].schoolYearID
20598               ,
20599               isLoading: false,
20600             });
20601           axios
20602             .post('api/registrar/gethonorstudents', {
20603               schoolYearID: res.data.schoolYearList[0].schoolYearID,
20604               gradeLevel: 'N',
20605             })
20606             .then(res2 => {
20607               this.setState({
20608                 isLoadingTable: false,
20609                 data: res2.data.data,
20610                 numOfHonors: res2.data.numOfHonors,
20611               });
20612             })
20613             .catch(err => {
20614               this.setState({
20615                 isLoadingTable: false,

```

```

20615             data: [],
20616             numOfHonors: 'N/A',
20617         });
20618     });
20619   })
20620   .catch(err => {
20621     this.setState({
20622       schoolYears: [],
20623       selectedSchoolYear: -1,
20624       isLoading: false,
20625       numOfHonors: 'N/A',
20626     });
20627   });
20628 });
20629 );
20630
20631 render() {
20632   const displayGradeLevel = gradeLevel => {
20633     switch (gradeLevel) {
20634       case 'N':
20635         return 'Nursery';
20636       case 'K1':
20637         return 'Kinder 1';
20638       case 'K2':
20639         return 'Kinder 2';
20640       case 'G1':
20641         return 'Grade 1';
20642       case 'G2':
20643         return 'Grade 2';
20644       case 'G3':
20645         return 'Grade 3';
20646       case 'G4':
20647         return 'Grade 4';
20648       case 'G5':
20649         return 'Grade 5';
20650       case 'G6':
20651         return 'Grade 6';
20652       case 'G7':
20653         return 'Grade 7';
20654       case 'G8':
20655         return 'Grade 8';
20656       case 'G9':
20657         return 'Grade 9';
20658       case 'G10':
20659         return 'Grade 10';
20660       case 'G11':
20661         return 'Grade 11';
20662       case 'G12':
20663         return 'Grade 12';
20664     default:
20665       return '';
20666     }
20667   };
20668   const schoolYearOptions = [];
20669   const DisplayData2 = [];
20670   for (const [index, value] of this.state.selectedSection.entries())
20671   {
20672     DisplayData2.push(
20673       <Table.Row>
20674         <Table.Col>{value.name}</Table.Col>
20675         <Table.Col>{value.grade}</Table.Col>
20676       </Table.Row>,
20677     );
20678   }
20679   for (const [index, value] of this.state.schoolYears.entries()) {
20680     schoolYearOptions.push(<option value={value.schoolYearID}>{value.
20681       schoolYear}</option>);
20682   }
20683   const DisplayData = [];
20684   for (const [index, value] of this.state.data.entries()) {
20685     DisplayData.push(
20686       <Table.Row>
20687         <Table.Col>{value.name}</Table.Col>
20688         <Table.Col>{value.numOfHonors}</Table.Col>

```

```

20687 <Table.Col>
20688   {value.honorInfo.length != 0 && (
20689     <Tooltip title="View honor students">
20690       <Button
20691         icon="file"
20692         color="primary"
20693         size="sm"
20694         pill
20695         outline
20696         onClick={() =>
20697           this.setState({
20698             selectedSection: JSON.parse(JSON.stringify(value.
20699               honorInfo)),
20700             showModal2: true,
20701           })
20702         }
20703       ></Button>
20704     </Tooltip>
20705   <Tooltip title="View condensed grades">
20706     <Link
20707       to={`/viewstudentrecord/section/${value.sectionID}/sy/$
20708         {this.state.selectedSchoolYear}/q/Q4`}
20709       target="_blank"
20710     >
20711       <Button icon="file" pill color="success" size="sm"
20712         outline></Button>
20713     </Link>
20714   </Table.Col>
20715 </Table.Row>,
20716 );
20717 return (
20718   <React.Fragment>
20719     <Modal
20720       title="View Honor Students"
20721       visible={this.state.showModal2}
20722       footer={[<Button onClick={() => this.setState({ showModal2:
20723         false })}>Close</Button>]}
20724       onCancel={() => this.setState({ showModal2: false,
20725         selectedSection: [] })}}
20726       width={700}
20727     >
20728       <Grid.Row>
20729         <Grid.Col sm={12} xs={12} md={12}>
20730           <Table highlightRowOnHover={true} responsive={true}>
20731             <Table.Header>
20732               <Table.ColHeader>Student Name</Table.ColHeader>
20733               <Table.ColHeader>Grade</Table.ColHeader>
20734             </Table.Header>
20735             <Table.Body>
20736               {DisplayData2.length == 0 ? (
20737                 <Table.Row>
20738                   <Table.Col colSpan={3} alignContent="center">
20739                     No data.
20740                   </Table.Col>
20741                 ) : (
20742                   DisplayData2
20743                 )}
20744               </Table.Body>
20745             </Table>
20746           </Grid.Col>
20747         </Grid.Row>
20748       </Modal>
20749       <Modal
20750         title="View Honor Students"
20751         visible={this.state.showModal1}
20752         footer={[<Button onClick={() => this.setState({ showModal1:
20753         false })}>Close</Button>]}
20754         onCancel={() =>
20755           this.setState({
```

```

20754     showModal: false,
20755     selectedGradeLevel: 'N',
20756     selectedSchoolYear: -1,
20757     data: [],
20758   })
20759 }
20760 >
20761   <Spin spinning={this.state.isLoading}>
20762     <Card statusColor="warning">
20763       <Card.Body>
20764         <Grid.Row>
20765           <Alert icon="info" type="primary">
20766             Note: It will display the number of honor students
20767             per school year and grade
20768           level.
20769         </Alert>
20770       <Grid.Col sm={12} xs={12} md={6}>
20771         <Form.Group>
20772           <Form.Label>Select School Year</Form.Label>
20773           <Form.Select
20774             onChange={e =>
20775               this.setState({ selectedSchoolYear: e.target.
20776                 value }, () =>
20777                 this.handleSchoolYearChange(),
20778               )
20779             }
20780           </Form.Select>
20781         </Form.Group>
20782       </Grid.Col>
20783     <Grid.Col sm={12} xs={12} md={6}>
20784       <Form.Group>
20785         <Form.Label>Grade Level</Form.Label>
20786         <Form.Select
20787           value={this.state.selectedGradeLevel}
20788           onChange={e =>
20789             this.setState({ selectedGradeLevel: e.target.
20790               value }, () =>
20791                 this.handleGradeLevelChange(),
20792               )
20793             }
20794           <option value="N">Nursery</option>
20795           <option value="K1">Kinder 1</option>
20796           <option value="K2">Kinder 2</option>
20797           <option value="G1">Grade 1</option>
20798           <option value="G2">Grade 2</option>
20799           <option value="G3">Grade 3</option>
20800           <option value="G4">Grade 4</option>
20801           <option value="G5">Grade 5</option>
20802           <option value="G6">Grade 6</option>
20803           <option value="G7">Grade 7</option>
20804           <option value="G8">Grade 8</option>
20805           <option value="G9">Grade 9</option>
20806           <option value="G10">Grade 10</option>
20807           <option value="G11">Grade 11</option>
20808           <option value="G12">Grade 12</option>
20809         </Form.Select>
20810       </Form.Group>
20811     </Grid.Col>
20812   </Grid.Row>
20813   <Spin spinning={this.state.isLoadingTable}>
20814     {, }
20815   <Grid.Row>
20816     <Grid.Col sm={12} xs={12} md={12}>
20817       <Table highlightRowOnHover={true} responsive={true}>
20818         <Table.Header>
20819           <Table.ColHeader>Section</Table.ColHeader>
20820           <Table.ColHeader>Honor Students</Table.
                                         ColHeader>

```

```

20821 <Table.ColHeader>Actions</Table.ColHeader>
20822 </Table.Header>
20823 <Table.Body>
20824     {DisplayData.length == 0 ? (
20825         <Table.Row>
20826             <Table.Col colSpan={4} alignContent="center">
20827                 No data.
20828             </Table.Col>
20829         </Table.Row>
20830     ) : (
20831         DisplayData
20832     )}
20833     </Table.Body>
20834     </Table>
20835     </Grid.Col>
20836     </Grid.Row>
20837     </Spin>
20838     </Card.Body>
20839     <Card.Body>
20840         <Grid.Row>Number of honor students: {this.state.
20841             numOfHonors}</Grid.Row>
20842     </Card.Body>
20843     </Card>
20844     </Spin>
20845     </Modal>
20846     <Button
20847         icon="file"
20848         size="sm"
20849         pill
20850         color="success"
20851         onClick={() => this.setState({ showModal: true }), () => this.
20852             handleFetchHonorStudents()}
20853         >
20854             Honor Students
20855         </Button>
20856     </React.Fragment>
20857 );
20858 }
20859 /* istanbul ignore next */
20860 function mapStateToProps(state) {
20861     return {
20862         app: state.app,
20863     };
20864 }
20865 /* istanbul ignore next */
20866 function mapDispatchToProps(dispatch) {
20867     return {
20868         actions: bindActionCreators({ ...actions }, dispatch),
20869     };
20870 }
20871 }
20872
20873 export default connect(mapStateToProps, mapDispatchToProps)(
    ViewHonorStudents);

```

V. Bibliography

- [1] M. Okabe, "Where does philippine education go? the "k to 12" program and reform of philippine basic education [abstract]," *Institute of Developing Economics, IDE Discussion Paper No. 425*, vol. 2, 2013.
- [2] O. Gazette, "What is k to 12 program?." <http://www.officialgazette.gov.ph/k-12/>. Accessed on 2019-05-03.
- [3] D. of Education, "E-class record templates." <http://www.deped.gov.ph/resources/downloads/e-class-record-templates/>. Accessed on 2019-05-03.
- [4] D. RM, "Design and evaluation of the electronic class record for lpu-laguna international school." http://www.academia.edu/8002516/Design_and_Evaluation_of_the_Electronic_Class_Record_for_LPU-Laguna_International_School, 2014. Accessed on 2019-05-03.
- [5] Gehlawat, "School management information system: An effective tool for augmenting the school practices," vol. 47, pp. 57–64, 06 2014.
- [6] W. Basri, J. Aalandejani, and F. Almadani, "Ict adoption impact on students' academic performance: Evidence from saudi universities," *Education Research International*, vol. 2018, pp. 1–9, 04 2018.
- [7] G. Hu, "Research into college student information management system based on web," in *International Conference on Education, Management, Computer and Society*, Atlantis Press, 2016/01.
- [8] A. Uche, A. Muhammed, and B. Ahmed, "The need for students information management system (sims) for nigerian universities in a technological age: Challenges and strategies for proper integration," *The International Journal of Social Sciences and Humanities Invention*, 12 2015.
- [9] J. Maggay and J. Guabes, "Student information and accounting system of cagayan state university – lasam campus, philippines," 08 2018.

- [10] M. Fujo and M. Dida, “Web-based admission system for advanced level, private schools: case of kilimanjaro region, tanzania,” *International Journal of Advanced Technology and Engineering Exploration*, vol. 5, pp. 407–418, 10 2018.
- [11] J. Caro, A. Betan, R. Feria, A. Lagman, N. Paje, and M. R. Solamo, “Multi-campus implementation of university information systems,” *Philippine Computing Journal*, vol. 10, pp. 33–39, 12 2015.
- [12] Udeze, “Automated students results management information system (srmis),” 10 2017.
- [13] S. Lubanga, W. Chawinga, F. Majawa, and S. Kapondera, “Web based student information management system in universities: Experiences from mzuzu university,” 05 2018.
- [14] H. Shah and T. Soomro, “Node.js challenges in implementation,” 05 2017.
- [15] Joyent, “Home page of node.js.” <https://nodejs.org/en/>, 2016. Accessed on 2019-05-03.
- [16] A. Mardan, *Building Node.js REST API Servers with Express.js and Hapi: Building Real-World Scalable Web Apps*, pp. 277–305. 01 2018.
- [17] W. Nahar, “Dynamic view rendering using reactjs and jquery,” 08 2016.
- [18] solid IT, “Knowledge base of relational and nosql database management systems.” <https://db-engines.com/en/>, 2019. Accessed on 2019-05-03.
- [19] Charzon, “What is mysql, history and functions,” 12 2018.