“Evaluation of Academic Performance”

A PROJECT REPORT

Submitted By

Nainsi Kansal- 2100290140093

Anshima Saini- 2100290140029

Kiran Chauhan- 2100290140074

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER APPLICATIONS

Under the supervision of
Dr. Vipin Kumar
(Associate Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(JUNE 2023)

DECLARATION

We hereby declare that the work presented in this report entitled "Evaluation of Academic Performance", was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not our original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name: - Kiran Chauhan
Roll no:- 2100290140074

Name: - Nainsi Kansal
Roll no:-2100290140093

Name: - Anshima Saini
Roll no:-2100290140029

Signature of Candidate

CERTIFICATE

Certified that **Nainsi Kansal- 2100290140093, Anshima Saini- 2100290140029, Kiran Chauhan- 2100290140074** have carried out the project work having "**Evaluation of Academic Performance**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Nainsi Kansal- 2100290140093
Anshima Saini- 2100290140029
Kiran Chauhan- 2100290140074

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Vipin Kumar
Associate Professor
Department of Computer Application
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner **Signature of External Examiner**

Dr. Arun Kumar Tripathi

Head Of Department,

Department of Computer Application

KIET Group of Institutions,

Ghaziabad

ABSTRACT

The **"Evaluation of Academic Performance"** project aims to develop an online programme or website that will provide faculty with access to a platform where they can easily keep track of the grades that students receive on internal exams, assignments submitted by the students, and their attendance, in order to evaluate students' performance without the hassle of creating time-consuming spreadsheets. It makes life easier for the staff by consolidating all student data onto a single platform. This website will be useful for the students as well because they can monitor their activities.

It aids a teacher in creating the student's final report card based on how well the student performed on the factors mentioned above. Here, the teacher can identify the subject in which the student is underperforming so that he or she can force the student to study on that poor subject by giving him or her more coursework or classes.

It is built using React, Spring Boot, Figma, and MySQL.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Vipin Kumar** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Introduction of the project

The performance on tests given by universities or institutes is used to gauge a student's academic accomplishment. This system uses a logic-based performance evaluation method to assess students' academic performance. With this methodology, we take into account three factors that go towards assessing a student's ultimate academic performance: attendance, internal grades, and external grades. The fuzzy inference method has also been used to determine student performance for various input values, including student attendance and grades. It's vital to note that the proposed technique's goal is to strengthen the existing system by offering extra information rather than to replace the current traditional method of evaluation.

Students can easily look for project specifics, academic attendance data, and mark/percentage details with the use of this student performance analysis system. Students can use the project title, the guide name, or the academic year to look for projects. The teachers and HODs input all the information regarding the projects, as well as information regarding student attendance and grades. There are three modules in it: student, teacher, and HOD. Students are required to register, log in, fill out their academic information, view projects, check their attendance, and view their grades in a graphed format, along with events and notices. Teachers have access to a secure login page where they may accept students, submit grades, upload attendance, add project details, view scheduled activities, and post notices. HOD can access their account to handle teachers' reports, events, see Attendance, and academic details and manage notice.

1.2 Problem Statement

Implement bootstrap, HTML, CSS, and JavaScript to create an application that students and teachers use to depict and analyze their performance on a monthly, annual basis in tabular and graphical form. Online student performance analysis systems' main objective is to provide high-quality software.

Students are a valuable resource for any school, college, or other educational institution because they excel in academics, practical knowledge, self-development, and creative thinking. Every school, college, and other educational institution must now analyze student performance in order to do this. Academic performance (AP) can be assessed using a variety of tests, evaluations, and other metrics. The grading procedure is made simpler and the teachers have an organized overview when they manage the grades of an entire class in its learning.

1.3 Objective of the project

Providing the online interface for students, faculty etc. Increasing the efficiency of school record management. Decrease time required to access and deliver student records. To make the system more secure. Decrease time spent on non-value-added tasks.

1.4 Hardware / Software Requirements

Operating System: Microsoft Windows

Front End: React

UI/UX: Figma

Software Requirements:

- Windows 10/11 or equivalent
- React
- Spring Boot
- RDBMS (Back end): MySQL

Hardware Requirements:

- Processor – Intel i3 5th generation or higher
- RAM – Minimum 4 GB, recommended 8 GB
- Disk space - Minimum 10 GB of free disk space
- Network Connectivity

1.5 Feasibility Study

A feasibility study assesses the operational, technical, and economic merits of the proposed project. The feasibility study is intended to be a preliminary review of the facts to see if it is worthy of proceeding to the analysis phase. From the systems analyst perspective, the feasibility analysis is the primary tool for recommending whether to proceed to the next phase or to discontinue the project.
The feasibility study is a management-oriented activity.

The objective of a feasibility study is to find out if an information system project can be done and to suggest possible alternative solutions.

1.5.1 Technical Feasibility

Technical Feasibility determines whether the work for the project can be done with the existing equipment's software technology and available personals technical feasibility is concerned with the specified equipment and software that will satisfy the user requirement this project is feasible on technical remark also as the proposed system is beneficiary in term of having a soundproof system with new technical equipment's install on the system that proposed system can run on any machine supporting window and work on the best software and hardware that had been used while designing the system so it would be visible in all technical term of feasibility.

1.5.2 Economic Feasibility

With manual system the operation cost of this system is about 60 lacs for annual this cost comprises salary of Swiss 25 people's stationary, building rent, electricity water telephone etc. but with a new system this offering cost comes out to be about 20 lacs for renewal and the new system is economically feasible.

1.5.3Operational Feasibility

The new solution is possible in all sense but operationally is not the new system demands the explosion of at least 15 people from the company it creates an environment of joblessness and fear among the employees it can lead to indefinite Strike in the company also, so the management must take corrective action prior in advance to start further proceeding.

1.5.2 Behavioral Feasibility

Behavioral Feasibility is the measure of how the society is looking towards our project, what is the reaction of people who are going to use this in upcoming future.

It includes how strong the reaction of user will be towards the development of new system that involves device use in their daily life for connecting with faculties

CHAPTER 2

LITERATURE REVIEW

The features of the existing system are including a user login creator to provide user interface, student performance analyzer, student development card, achieved credit, passing criteria card and wise student performance attribute card.

2.1 WEB BASED STUDENT INFORMATION MANAGEMENT

The design and implementation of a comprehensive student information system and user interface is to replace the current paper records College Staff are able to directly access all aspects of a student’s academic progress through a secure, online interface embedded in the college’s website. The system utilizes user authentication, displaying only information necessary for an individual’s duties. Additionally, each subsystem has authentication allowing authorized users to create or update information in that sub-system. All data is thoroughly reviewed and validated on the server before actual record alteration occurs.

In addition to a staff user interface, the system plans for student user interface, allowing users to access information and submit requests online thus reducing processing time. All data is stored securely on SQL servers managed by the college administrator and ensures highest possible level of security. The system features a complex logging system to track all users access and ensure conformity to data access guidelines and is expected to increase the efficiency of the college’s record management thereby decreasing the work hours needed to access and deliver student records to users.

2.2 EXISTING SYSTEM

In the existing system, all the student information is added manually and the data is stored in the records.

Takes a lot of time and physical effort in searching and adding the information. In the existing system, there is a possibility of losing data and no proper maintenance of data. The use of linear search in file handling might increase time complexity.

CHAPTER 3

SYSTEM DESIGN

System design is the solution of a “how to approach to the creation of the new system. It is composed of several steps. It facilitates the understanding and provides the procedural details necessary for implementation of the system recommended in the feasibility study. Emphasis is given on translating the performance requirements into design specification. Design goes through logical and physical stages of development.

Logical design reviews the present physical system; prepares input and output specification; make editing; security and control specification; details the implementation plan, and prepare logical design walk through. The physical design maps out the details of the physical system; plans the system implementation plan and specifies hardware and software. System design translates the system requirement into the ways of the system as recommended in the feasibility study. Thus, the system design is the translation from user-oriented document to a programmer or a database personal oriented document.

3.1 MODULE DESCRIPTION

The suggested system gives the student quick access to precise project and grade point data. Students may quickly and easily read all the material with only one click. The suggested system keeps an information database where all the data is kept. There is zero danger of data loss with this system. It is quite simple to add information and search for it, and it doesn't take much time or physical work. The system is made up of the following three major components and their supporting modules:

- Dashboard
- Academics
- Attendance marks
- Internal marks
- Assignment
- Final Report

3.1.1 Student

- Sign up: Students can sign up and get credentials.
- Login: Students can use their credentials to log in.
- Profiling and Academic Information: Students may enter their personal.
- See Projects: Students may look at the completed work.
- See Attendance & Academic Marks: Students have the option to view.
- See Events: Students may view events currently taking place or upcoming events.
- See Notice: The notification is also available to students.

3.1.2 HOD

- Login: The HOD may log in with their credentials.
- Managing Teachers: HOD has management capabilities.
- Manage Event: The HOD can assign students to events.
- See Attendance & Academic Information: The HOD can view the information.
- Control Notice: HOD has control over notice.

3.1.3 Teacher

- Login: Teachers can access the website by entering their login information.
- Approve Students: They are able to do this.
- Add Student Marks: Students' academic marks may also be added.
- Upload Attendance: Students' attendance can be uploaded. Also, they can add project specifics.
- See Assigned Events: The assigned events are visible.
- View Notice: The notification may also be viewed by them.

.

3.2 CONCEPTUAL MODELS

3.2.1 USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as we

While a use case itself might drill into a lot of detail about every possibility, a use-case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must do.

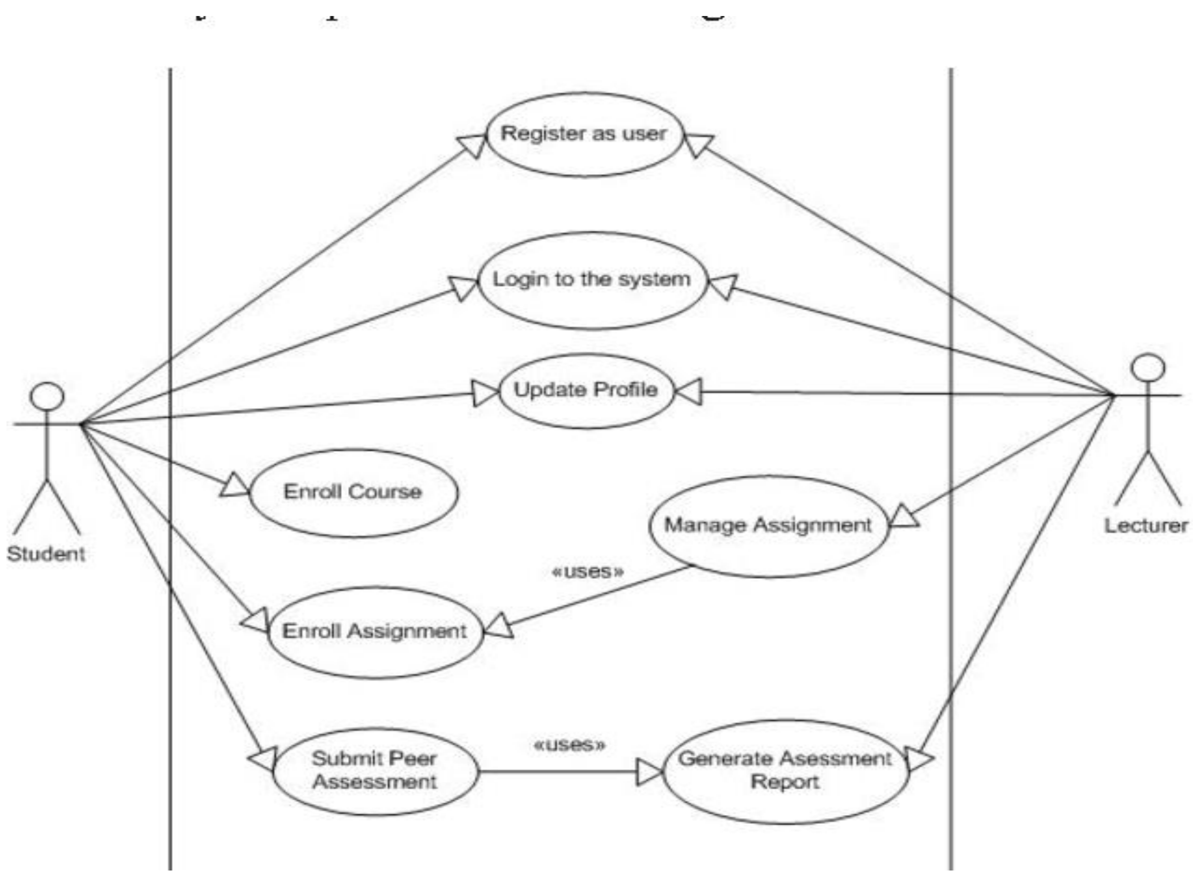


Figure 3.1: Use Case Diagram

3.2.2 DATA FLOW DIAGRAM

The Data Flow Diagram shows the flow of data or information. It can be partitioned into single processes or functions. Data Flow Diagrams can be grouped together or decomposed into multiple processes. There can be physical DFD's that represent the physical files and transactions, or they can be business DFD's (logical, or conceptual).

When it's used?

The DFD is an excellent communication tool for analysts to model processes and functional requirements. One of the primary tools of the structured analysis efforts of the 1970's it was developed and enhanced by the likes of Yourdon, McMenamin, Palmer, Gane and Sarson. It is still considered one of the best modelling techniques for eliciting and representing the processing requirements of a system.

Used effectively, it is a useful and easy to understand modeling tool. It has broad application and usability across most software development projects. It is easily integrated with data modeling, workflow modeling tools, and textual specs. Together with these, it provides analysts and developers with solid models and specs. Alone, however, it has limited usability. It is simple and easy to understand by users and can be easily extended and refined with further specification into a physical version for the design and development teams.

The different versions are Context Diagrams (Level 0), Partitioned Diagrams (single process only -- one level), functionally decomposed, leveled sets of Data Flow Diagrams.

Data Store

It is a repository of information. In the physical model, this represents a file, table, etc. In the logical model, a data store is an object or entity.

Data Flows

DFDs show the flow of data from external entities into the system, showed how the data moved from one process to another, as well as its logical storage. There are only four symbols:

- Squares representing **external entities**, which are sources or destinations of data.
- Rounded rectangles representing **processes**, which take data as input, do something to it, and output it.
- Arrows representing the **data flows**, which can either, be electronic data or physical items.
- Open-ended rectangles representing **data stores**, including electronic stores such as databases or XML files and physical stores such as or filing cabinets or stacks of paper.

There are several common modelling rules for creating DFDs:

- All processes must have at least one data flow in, and one data flow out.
- All processes should modify the incoming data, producing new forms of outgoing data.
- Each data store must be involved with at least one data flow.
- Each external entity must be involved with at least one data flow.
- A data flow must be attached to at least one process.

DFDs are nothing more than a network of related system functions and indicate from where information is received and to where it is sent. It is the starting point in the system that decomposes the requirement specifications down to the lowest level detail.

The four symbols in DFD, each of which has its meaning. They are given below:

- External entities are outside to system but they either supply input data in the system or use the system output. These are represented by square of rectangle. External entities that supply data into a system are sometimes called Sources. External entities that use system data are sometimes called sinks.
- Dataflow models that passage of data in the system and are represented by line by joining system components. An arrow indicates the direction of the flow, and the line is labeled by the name of the dataflow.

- Process show that the systems do. Each process has one or more data inputs and one or data outputs. Circles in DFD represent them. Each high-level process may be consisting of more than one lower-level processes. Process will be expanded in sequent level DFD. A circle or a bubble represents a process that transforms incoming data flow into outgoing dataflow.

  The high-level processes in a system are:

  - Receivable process.
  - Verifiable process.
  - Disposal process.

- File or data store is a repository of data. They contain data that is retained in the system. Process can enter data into data store or retrieved data from the data store. An open rectangle is a data store, data at rest.

0-Level DFD

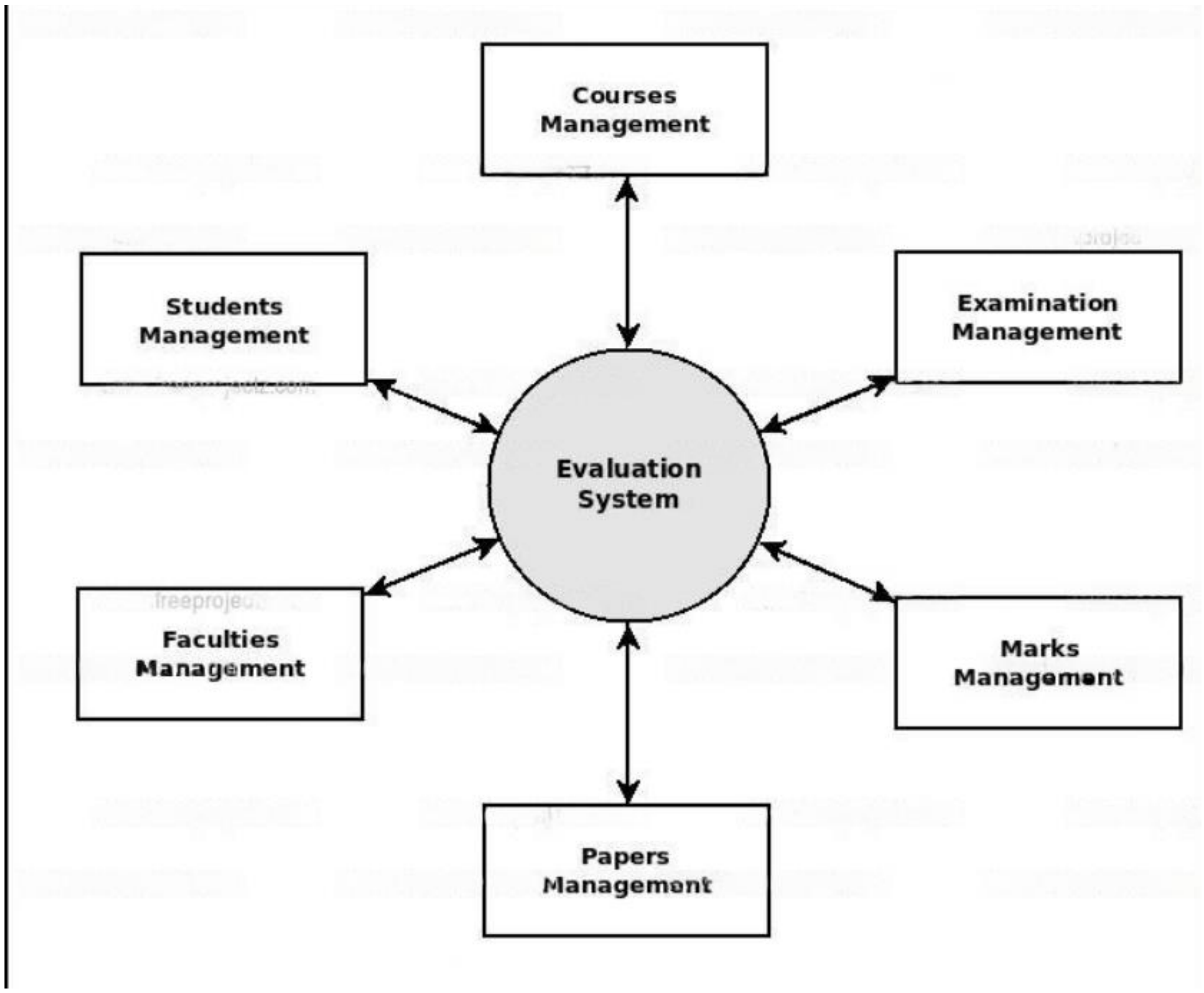


Figure 3.2: Level-0 Data Flow Diagram

1-Level DFD

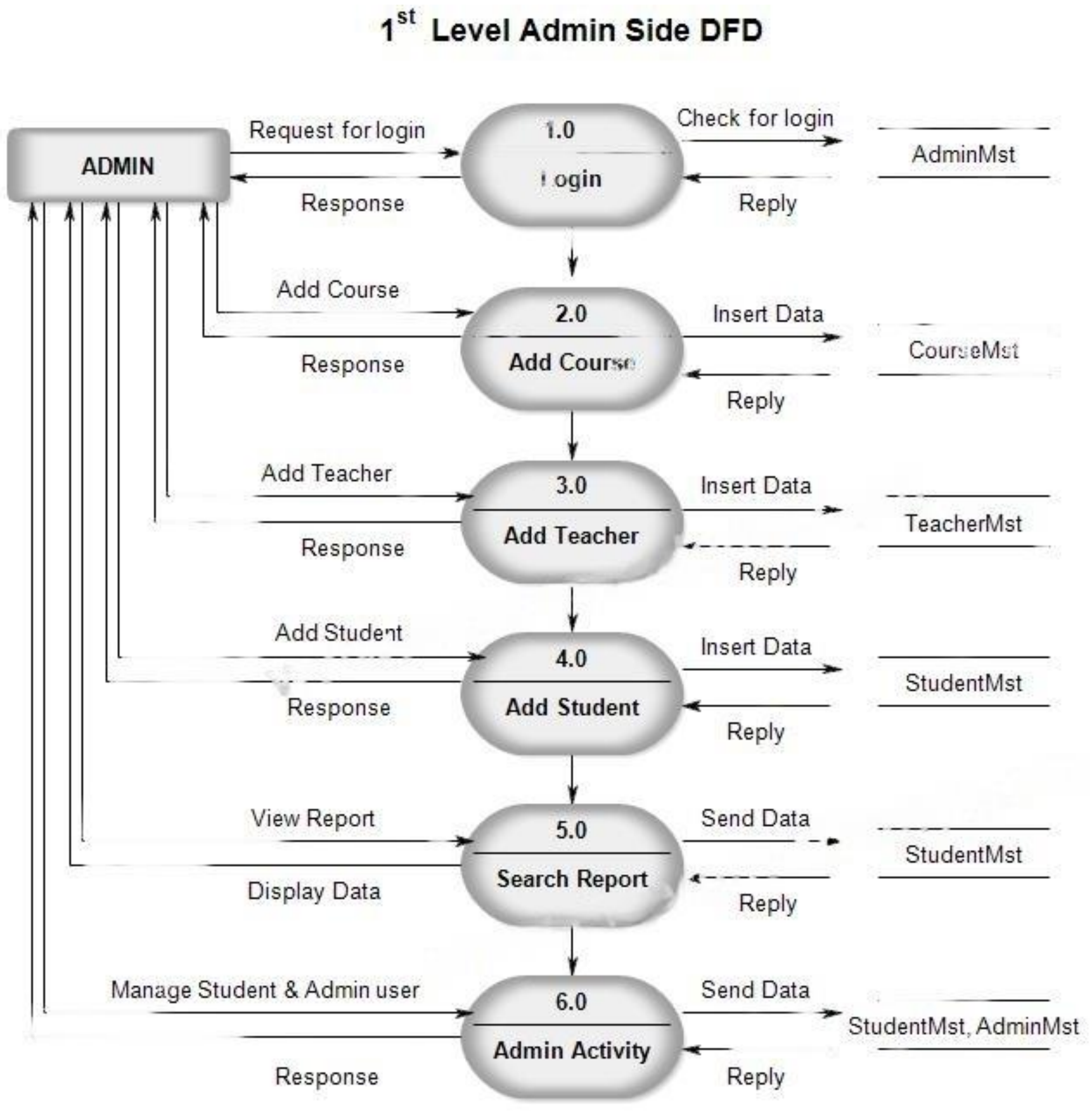


Figure 3.3: Level-1 Data Flow Diagram

3.2.3 Entity-Relationship Model

Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represent data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database.

Basic Constructs of E-R Modelling

The ER model views the real world as a construct of entities and association between entities.

Entities

Entities are the principal data object about which information is to be collected. Entities are classified as independent or dependent (in some methodologies, the terms used are strong and weak, respectively). An independent entity is one that does not rely on another for identification. A dependent entity is one that relies on another for identification. .

Relationships

A Relationship represents an association between two or more entities. Relationships are classified in terms of degree, connectivity, cardinality, and existence.

Attributes

Attributes describe the entity of which they are associated. A particular instance of an attribute is a value. The domain of an attribute is the collection of all possible values an attribute can have. The domain of Name is a character string.

Classifying Relationships

Relationships are classified by their degree, connectivity, cardinality, direction, type, and existence. Not all modeling methodologies use all these classifications.

Degree of a Relationship

The degree of a relationship is the number of entities associated with the relationship. The n-ary relationship is the general form for degree n. Special cases are the binary, and ternary, where the degree is 2 and 3 respectively.

Connectivity and Cardinality

The connectivity of a relationship describes the mapping of associated entity instances in the relationship. The values of connectivity are "one" or "many". The cardinality of a relationship is the actual number of related occurrences for each of the two entities. The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many.

Direction

The direction of a relationship indicates the originating entity of a binary relationship. The entity from which a relationship originates is the parent entity; the entity where the relationship terminates is the child entity.

The direction of a relationship is determined by its connectivity type An identifying relationship is one in which one of the child entities is also a dependent entity. A non-identifying relationship is one in which both entities are independent.

Existence

Existence denotes whether the existence of an entity instance is dependent upon the existence of another, related, entity instance. The existence of an entity in a relationship is defined as either mandatory or optional.

Generalization Hierarchies

A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into a higher-level entity type called a supertype or generic entity. The lower-level of entities become the subtype, or categories, to the supertype**.**

ER Notation

The symbols used for the basic ER constructs are:

- Entities are represented by labelled rectangles. The label is the name of the entity.
- Relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs.
- Attributes, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.
- Cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.
- Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.

Existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional.

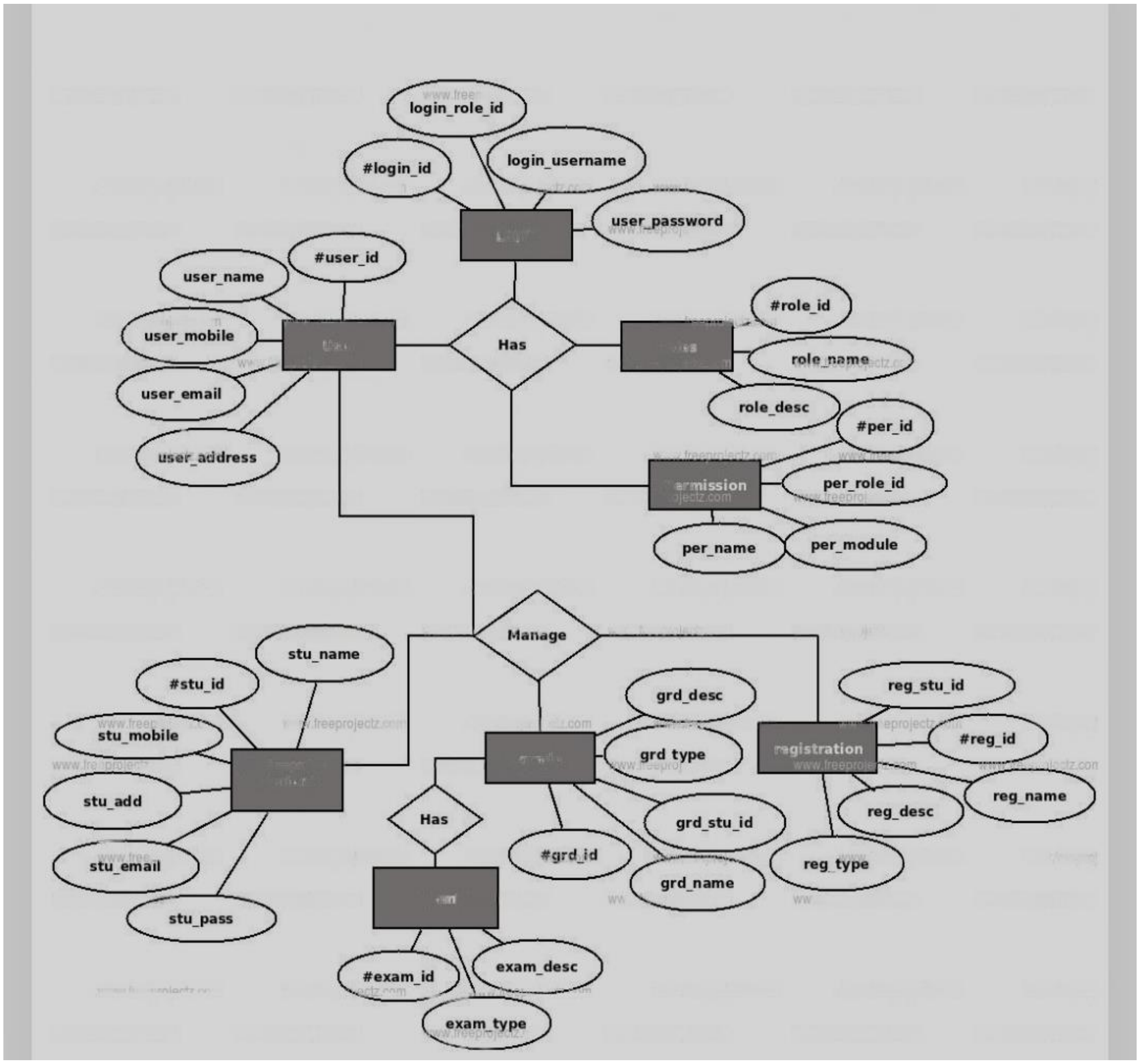


Figure 3.4: E-R Diagram

3.3 SEQUENCE DIAGRAM

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Sequence diagrams are sometimes known as event diagrams or event scenarios.

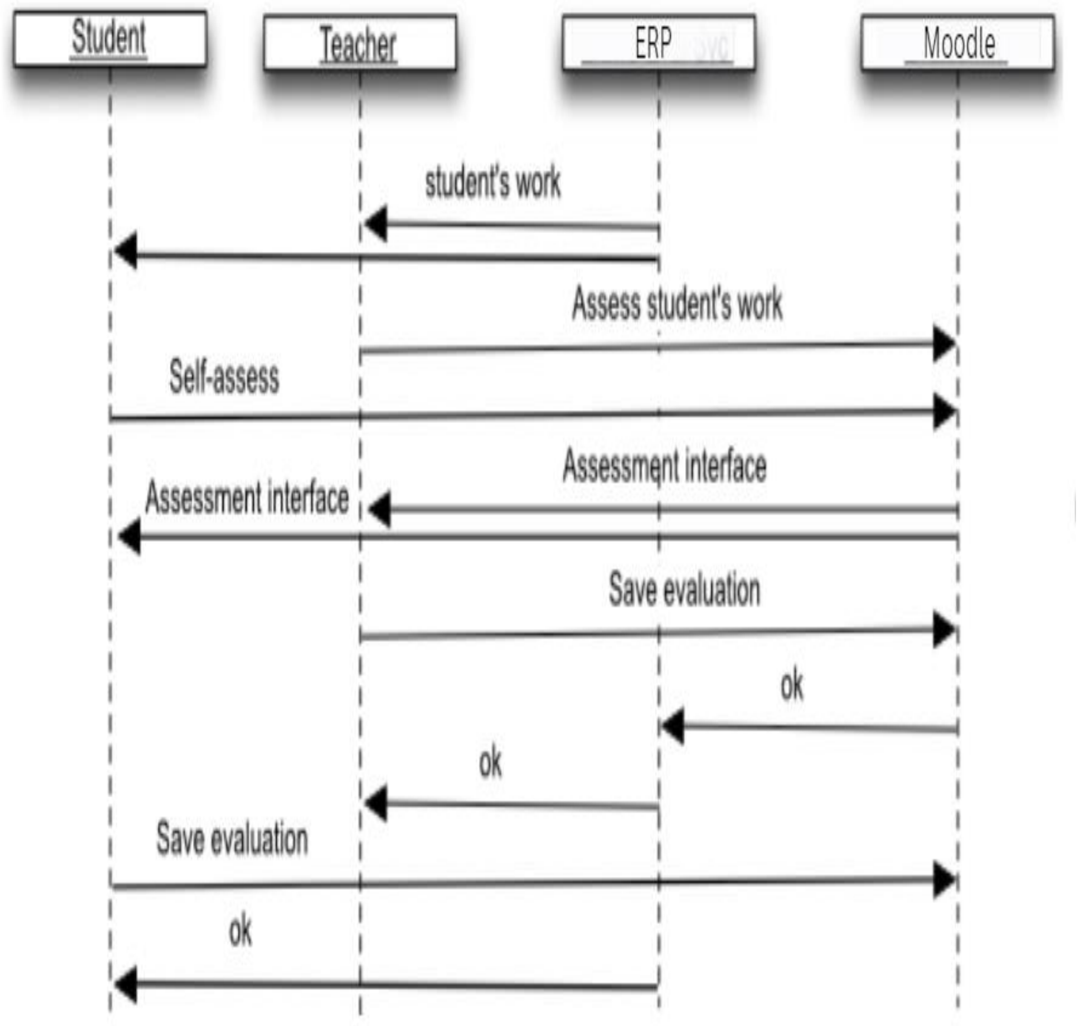


Figure 3.5: Sequence Diagram

3.5 DATABASE DESIGN

The general theme behind a database is to handle information as an integrated whole.

A database is a collection of inter-related data stored with minimum redundancy to serve single users quickly and efficiently. The general objective is to make information necessary, quick, inexpensive, and flexible for the use.

Why is Database Design important?

The important consideration that can be taken into account while emphasizing the importance of database design can be explained in terms of the following points given below.

- Database designs provide the blueprints of how the data is going to be stored in a system. A proper design of a database highly affects the overall performance of any application.
- The designing principles defined for a database give a clear idea of the behavior of any application and how the requests are processed.
- Another instance to emphasize the database design is that a proper database design meets all the requirements of users.
- Lastly, the processing time of an application is greatly reduced if the constraints of designing a highly efficient database are properly implemented.

DATABASE TABLES

Admin Table

Field Name	Data Type	Size	Allow Null	Constrain
email	int	4	No	PK
first_name	varchar	35	No	
last_name	varchar	25	No	
password	varchar	20	No	
phone	varchar	7	No	
role	varchar	35	No	

Table 3.1: Admin Table

Student Table

Field Name	Data Type	Size	Allow Null	Constrain
email	int	4	No	PK
address	varchar	30	No	
alter_email	varchar	4	No	
branch	varchar	5	No	
course	varchar	5	No	
dob	varchar	6	No	
father_name	varchar	20	No	
last_name	varchar	20	No	

mentor	varchar	20	No	
mother_name	varchar	20	No	
password	varchar	10		
phone_num	number	10		
role	varchar	15		
roll_num	number	13		
section	varchar	2		
semester	varchar	5		
year	varchar	4		

Table 3.2: Student Table

Subject Table

Field Name	Data Type	Size	Allow Null	Constrain
sub_id	int	4	No	PK
ass_type	varchar	60	No	
exam_type	varchar	25	No	
max_ass_ marks	longtext	0	No	
max_exam _marks	varchar	40	No	
obt_assign _mrks	varchar	20	No	
sub_code	varchar	10	No	
sub_name	int	5	No	

Table 3.3: Subject Table

Student_subjects Table

Field Name	Data Type	Size	Allow Null	Constrain
subcat_id	int	4	No	PK
Parent_id	Int	4	No	
Subcat_nm	varchar	35	No	

Table 3.4: Student_subjects Table

CHAPTER 4

FORM DESIGN

4.1 HOMEPAGE

With our user-friendly interface, educators and administrators can effortlessly navigate through various features, including grade tracking, attendance monitoring, and personalized performance reports. Our goal is to empower educational institutions to make data-driven decisions that enhance teaching strategies, identify areas of improvement, and support student success.

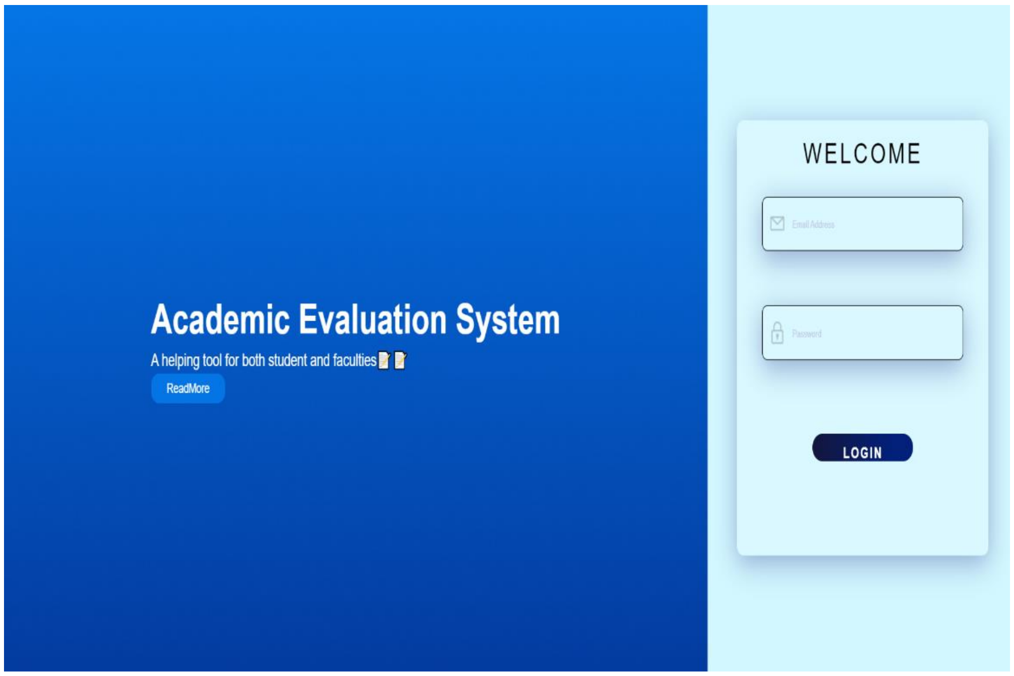
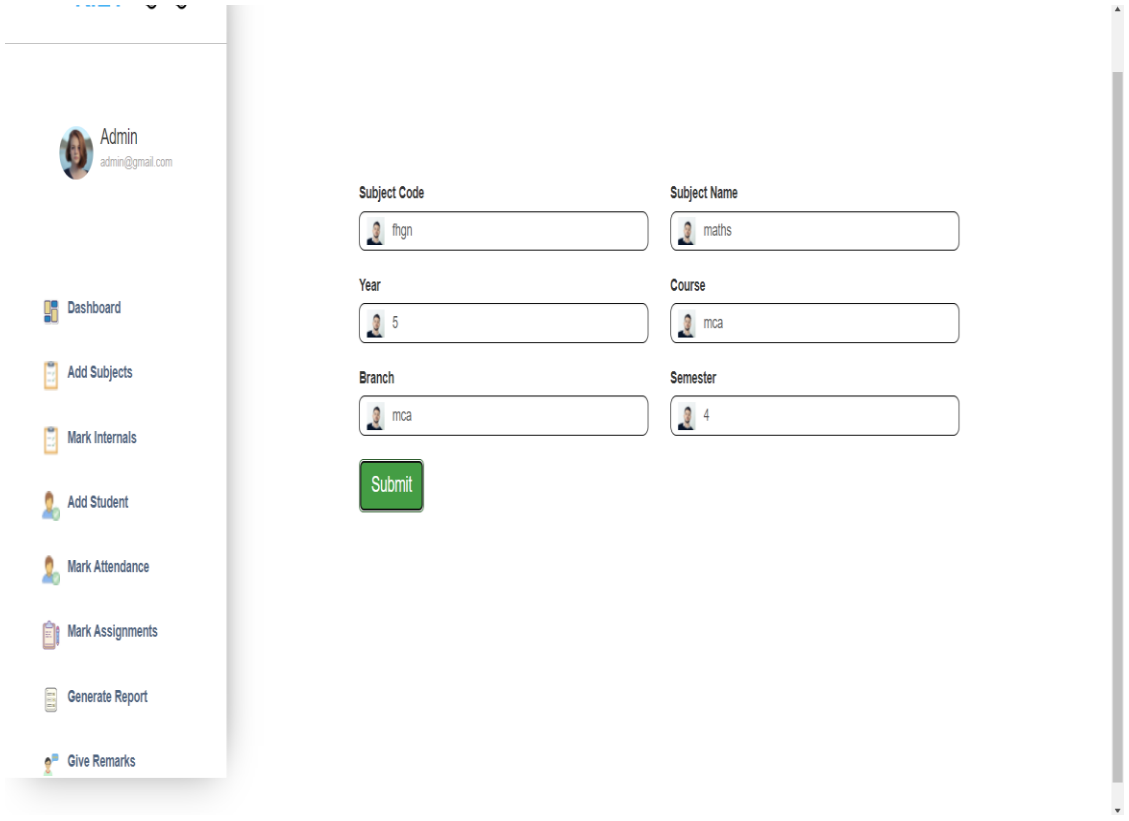

Academic Evaluation System
A helping tool for both student and faculties
ReadMore
WELCOME
Email Address
Password
LOGIN

Figure 4.1: HomePage

4.2 DASHBOARD

Administrators can utilize the dashboard to gain a bird's-eye view of the institution's overall academic performance, allowing them to make informed decisions and allocate resources effectively. The customizable nature of our dashboard ensures that it meets the unique needs of each educational institution, enabling administrators to track specific goals and benchmarks.



Subject Code: fhgn
Subject Name: maths
Year: 5
Course: mca
Branch: mca
Semester: 4

Figure 4.2: Dashboard

4.3 ADD STUDENT FORM

The Add Student Form captures crucial details such as the student's name, age, grade level, contact information, and previous academic history. This information serves as the foundation for assessing and tracking the student's progress throughout their educational journey.

Additionally, the form allows for the inclusion of specific notes or comments that provide valuable insights into the student's individual needs, learning styles, or any other pertinent information that can contribute to a personalized academic experience.

By using this form, educators and administrators can efficiently populate the evaluation system with accurate and up-to-date student information, ensuring that the academic performance tracking process is seamless and comprehensive. This empowers educators to better understand their students, tailor instruction, and provide the necessary support to help them thrive academically.

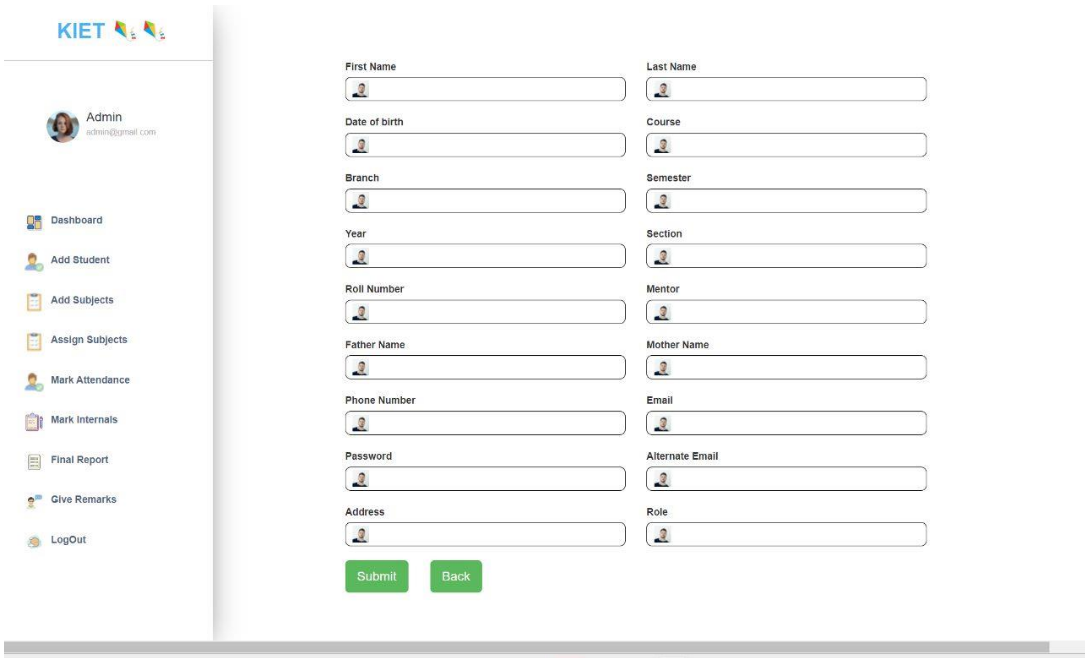


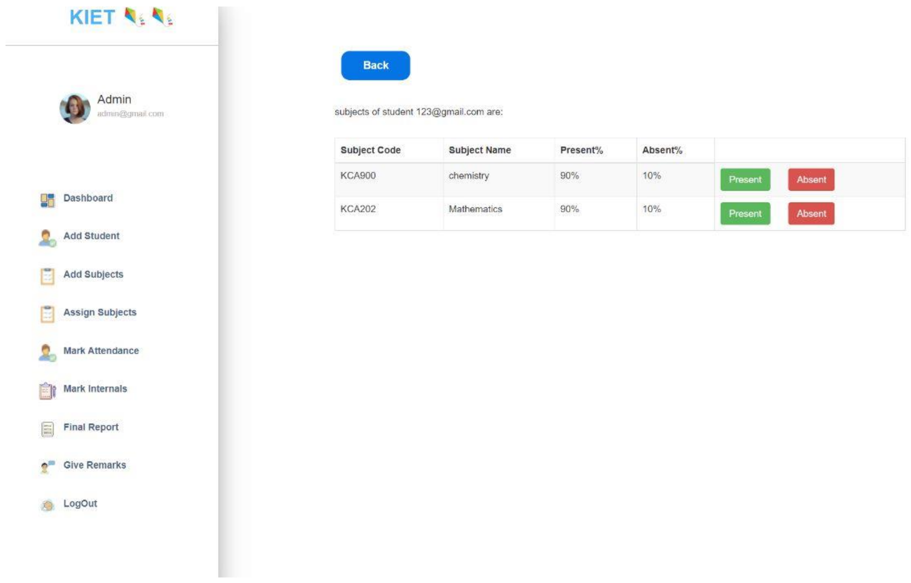
Figure 4.3: Add Student Form

4.4 SHOW ATTENDENCE

Our attendance module allows educators and administrators to easily monitor and track student attendance records. With just a few clicks, you can accurately record and analyze student attendance data to gain valuable insights into their overall engagement and commitment to their education.

Our user-friendly interface provides a clear and concise display of attendance information, allowing you to quickly view attendance records for individual students, entire classes, or specific time periods. By identifying patterns and trends, educators can identify students who may be at risk of falling behind or missing out on crucial instruction.

Through our platform, educators can generate automated attendance reports or send timely notifications to keep parents informed about their child's attendance status. This promotes a collaborative approach and encourages parental involvement in supporting their child's educational journey.



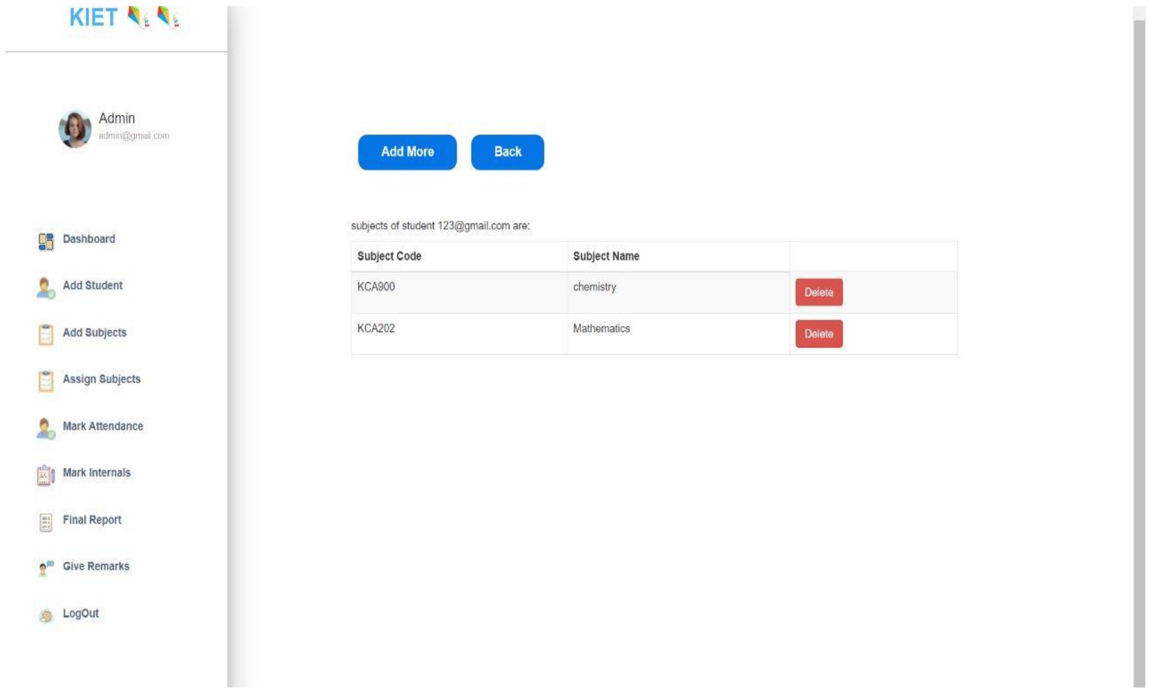
subjects of student 123@gmail.com are:

Subject Code	Subject Name	Present%	Absent%	
KCA900	chemistry	90%	10%	Present Absent
KCA202	Mathematics	90%	10%	Present Absent

Figure 4.4: Show Attendance

4.5 SHOW STUDENT’S SUBJECTS

In this section, you can access a student's profile and navigate to their subjects tab, where you will find a list of all the subjects they are enrolled in. Each subject is accompanied by relevant details such as the subject name, course code, and any additional information provided during the enrollment process.



subjects of student 123@gmail.com are:

Subject Code	Subject Name	
KCA900	chemistry	Delete
KCA202	Mathematics	Delete

Figure 4.5: Show Student’s Subject

4.6 EDIT SUBJECTS

You can easily access and modify subject details such as subject names, course codes, descriptions, and any other relevant information. Whether you need to add new subjects, update existing ones, or remove subjects that are no longer applicable, our user-friendly interface makes the editing process seamless and efficient.

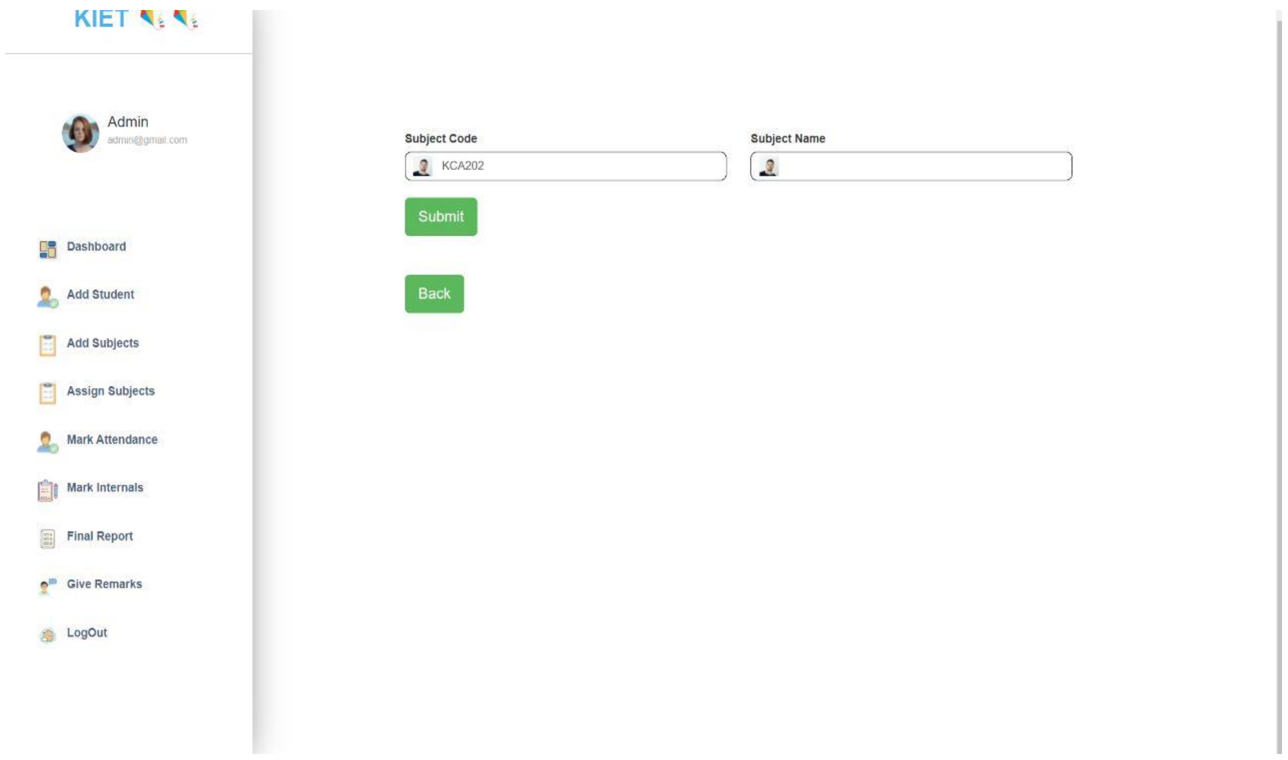
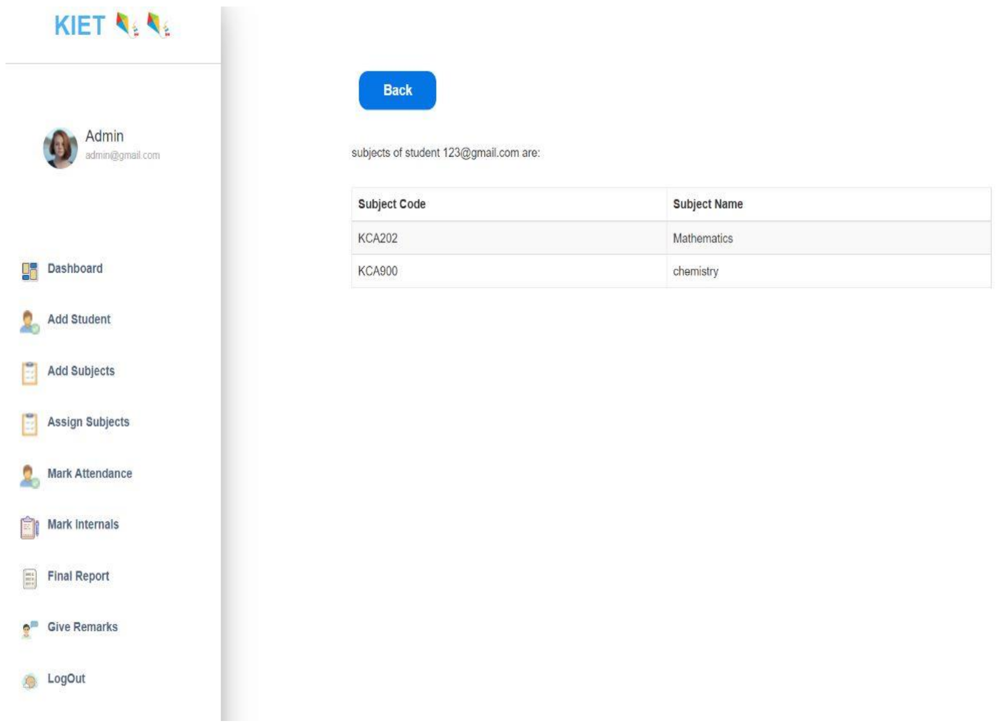


Figure 4.6: Edit Subjects

4.7 ASSIGNED SUBJECTS

This feature provides educators and administrators with a clear overview of the subjects assigned to each student. It allows for efficient management of subject assignments, ensuring that students are enrolled in the appropriate courses based on their grade level, academic program, or individualized learning plans.

Within this section, you can access a student's profile and navigate to their assigned subjects tab. Here, you will find a comprehensive list of the subjects that have been assigned to the student, along with relevant details such as subject names, course codes, and any additional information specified during the enrollment process.



subjects of student 123@gmail.com are:

Subject Code	Subject Name
KCA202	Mathematics
KCA900	chemistry

Figure 4.7: Assigned Subjects

4.8 SHOW ASSIGNMENT

Within this section, you can access a student's profile and navigate to the Assignments tab. Here, you will find a list of all the assignments given to the student, along with relevant details such as assignment names, due dates, descriptions, and any additional instructions or materials provided.

The Assignment feature enables educators to efficiently distribute and collect assignments, monitor submission deadlines, and provide timely feedback to students. By having a centralized location for assignments, both educators and students can stay organized and keep track of their progress and responsibilities.

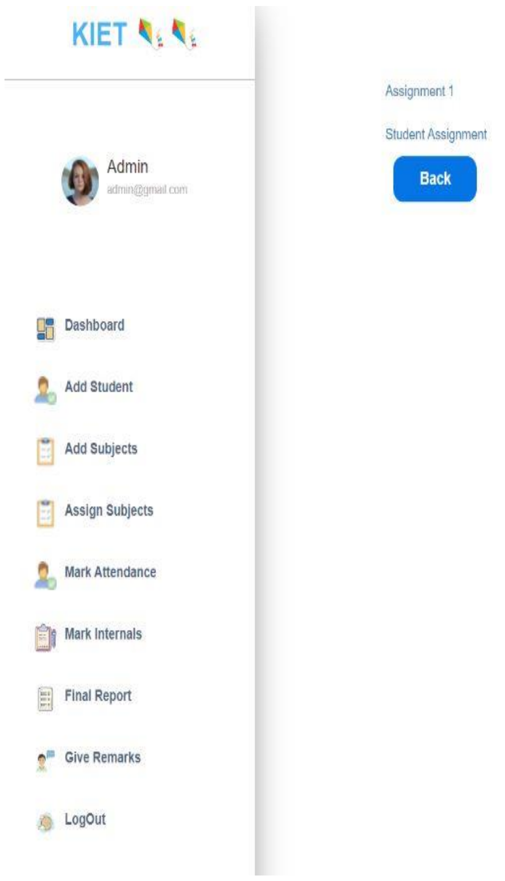


Figure 4.8: Show Assignments

4.9 ADD ASSIGNMENT FORM

Educators can use this feature to create and customize assignments based on their teaching objectives and curriculum requirements. They can attach relevant resources, set grading criteria, and provide clear instructions to ensure students understand the assignment expectations.

Students can access their assigned tasks, view deadlines, and submit completed work through the platform. This promotes accountability and engagement, allowing students to take ownership of their learning and stay on track with their assignments.

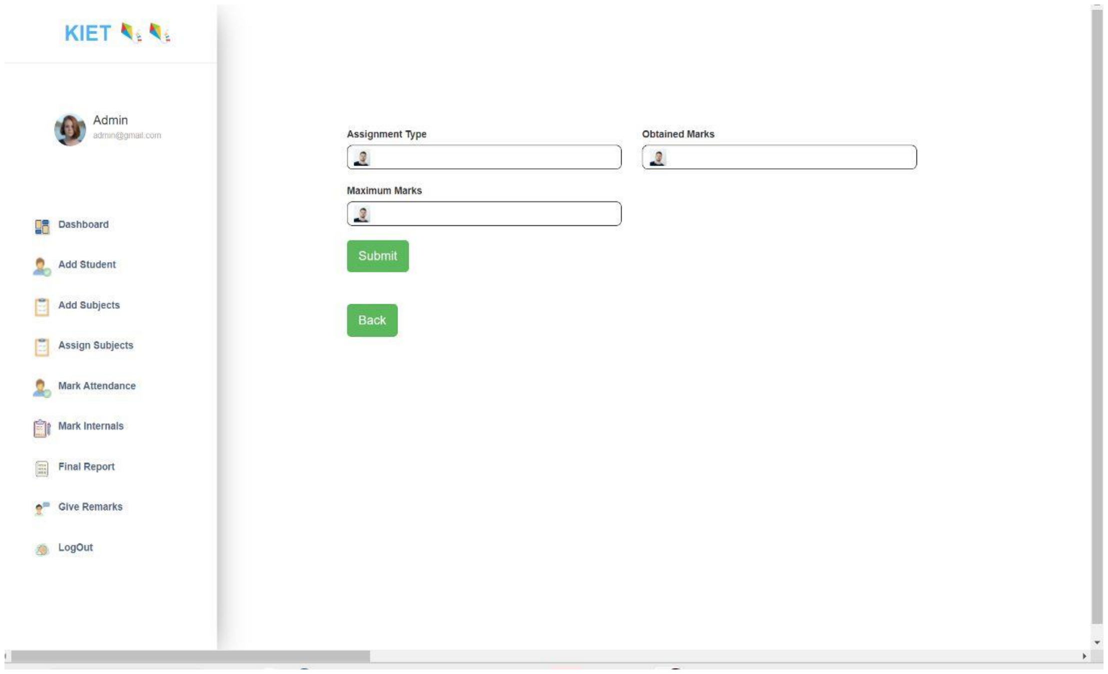
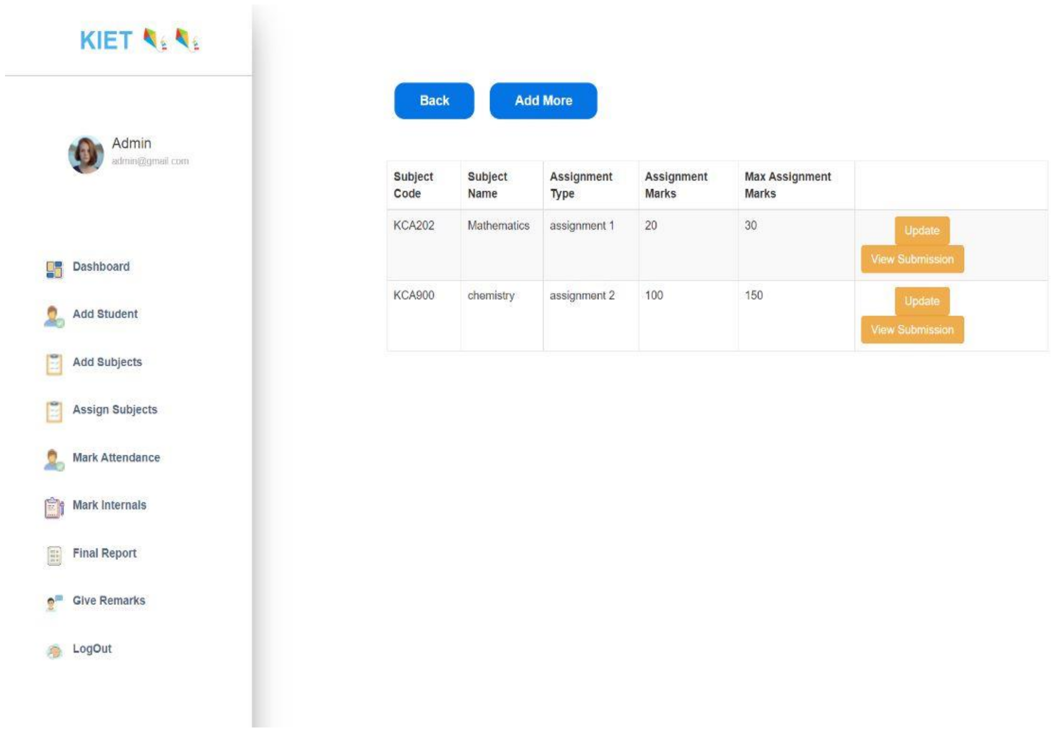


Figure 4.9: Add Assignment Form

4.10 ADD ASSIGNMENT

To add an assignment to the Evaluation of Academic Performance Project, you would typically follow the procedures and interface provided by the project's platform. This may involve logging in as an educator or administrator, accessing the appropriate section for managing assignments, and inputting the necessary details such as assignment name, due date, description, and any additional instructions.



Subject Code	Subject Name	Assignment Type	Assignment Marks	Max Assignment Marks	
KCA202	Mathematics	assignment 1	20	30	Update View Submission
KCA900	chemistry	assignment 2	100	150	Update View Submission

Figure 4.10: Add Assignment

4.11 ADD EXAM FORM

By incorporating the Exam Form Feature into the Evaluation of Academic Performance Project, educators can streamline the exam management process, ensure consistent assessment practices, and provide valuable feedback to support student learning and growth.

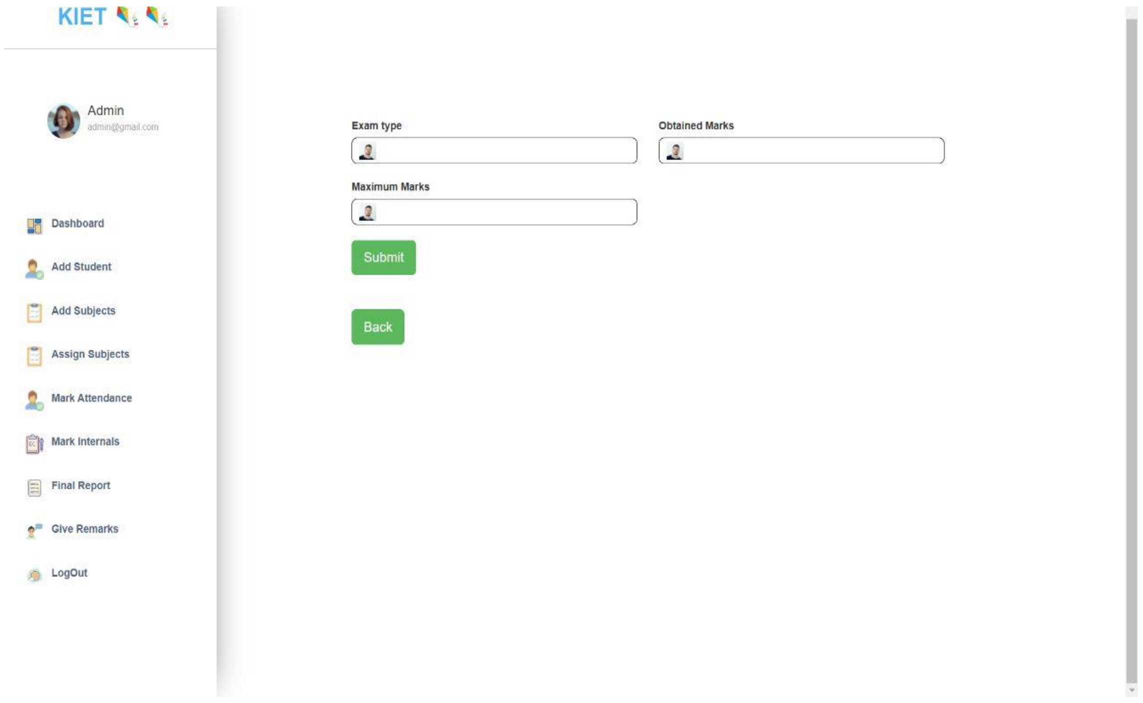


Figure 4.11: Add Exam Form

4.12 EDIT EXAM FORM

The Exam Form Editing feature in the Evaluation of Academic Performance Project empowers educators and administrators to efficiently modify and update existing exam forms. It provides a user-friendly interface to make necessary changes to exam details, question banks, scoring criteria, exam schedule, instructions, and exam sections. This feature ensures that exams remain relevant, accurate, and aligned with learning objectives. Educators can adapt the exam to changing circumstances, refine the grading system, and customize sections based on specific course objectives. They can also review and preview the modified exam before distributing it to students. The Exam Form Editing feature helps maintain the quality of exams and promotes fair and accurate assessment of student knowledge and skills.

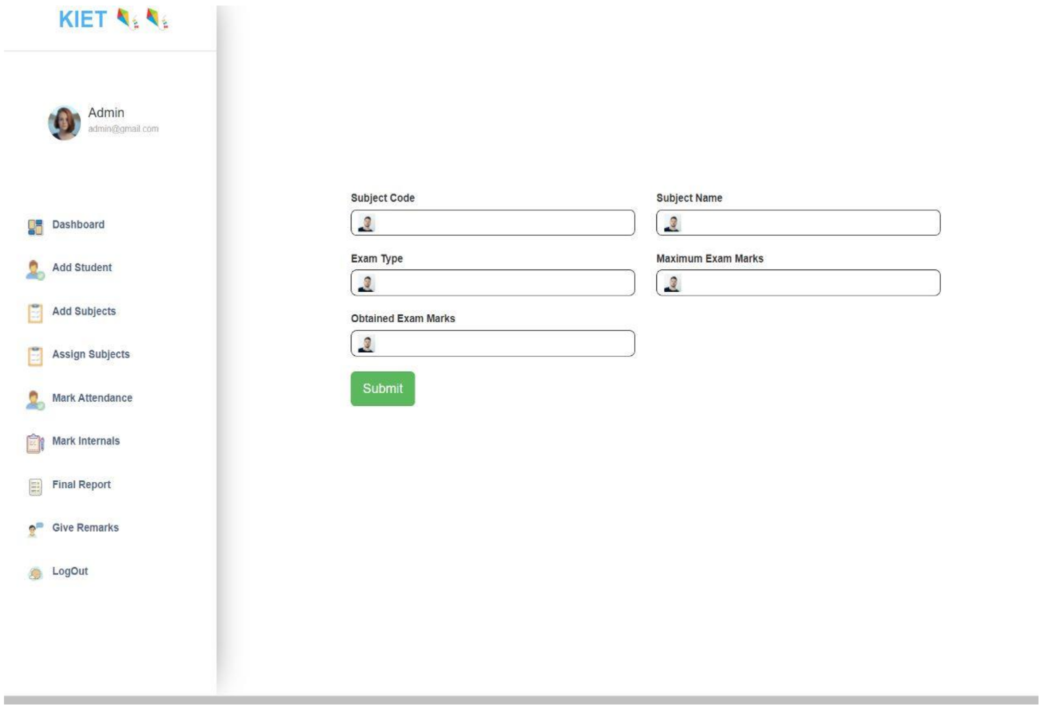
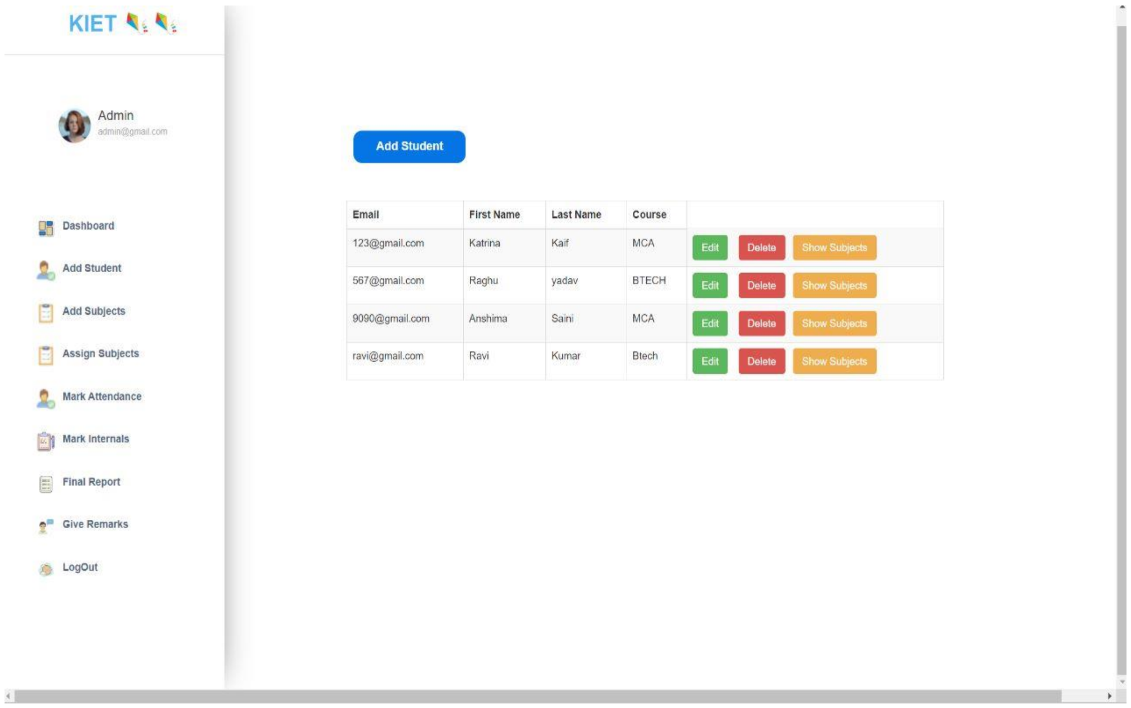


Figure 4.12: Edit Exam Form

4.13 ADD STUDENTS

The "Add Students" feature in the Evaluation of Academic Performance Project simplifies the process of enrolling new students. It allows educators and administrators to input individual student information or import bulk data. This feature ensures accurate student records and enables the creation of comprehensive student profiles within the system.

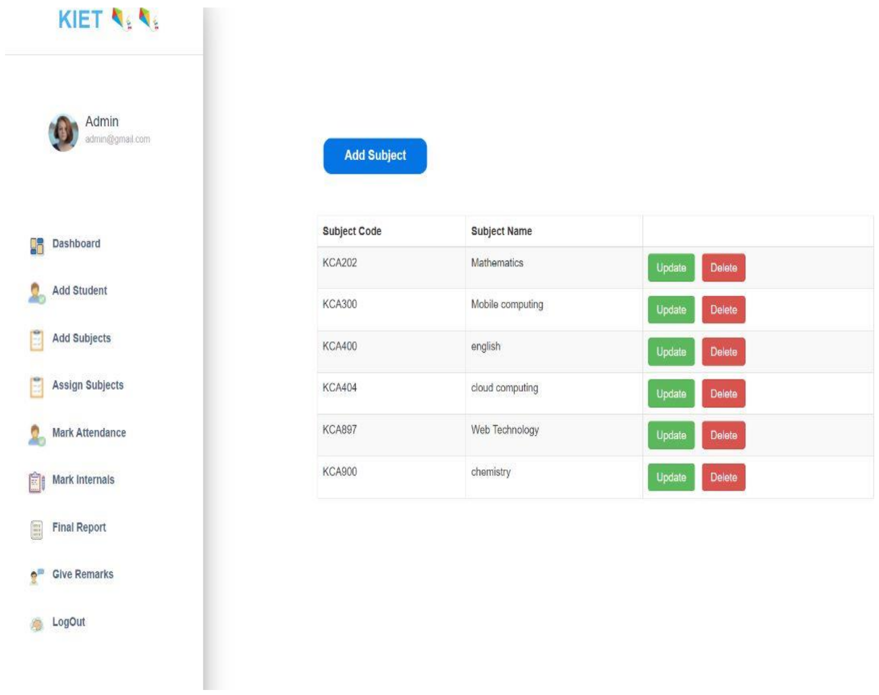


Email	First Name	Last Name	Course	
123@gmail.com	Katrina	Kaif	MCA	Edit Delete Show Subjects
567@gmail.com	Raghu	yadav	BTECH	Edit Delete Show Subjects
9090@gmail.com	Anshima	Saini	MCA	Edit Delete Show Subjects
ravi@gmail.com	Ravi	Kumar	Btech	Edit Delete Show Subjects

Figure 4.13: Add Students

4.14 ADD SUBJECTS

The "Add Subjects" feature in the Evaluation of Academic Performance Project enables educators and administrators to easily add new subjects or courses to the system. This feature provides a user-friendly interface to input subject details, including subject name, code, description, and any additional information. It allows for the efficient management and organization of academic subjects, ensuring accurate and up-to-date records.



Subject Code	Subject Name	
KCA202	Mathematics	Update Delete
KCA300	Mobile computing	Update Delete
KCA400	english	Update Delete
KCA404	cloud computing	Update Delete
KCA897	Web Technology	Update Delete
KCA900	chemistry	Update Delete

Figure 4.14: Add Subjects

CHAPTER 5

CODING

5.1 HOMEPAGE

```
import React, { useRef } from "react";
import styled from "styled-components";
import Bg from "../../assets/images/Bg.png";
import swal from "sweetalert";

import { useNavigate } from "react-router-dom";

const LoginPAge = styled.div`
background-color:#D2F7FF;
`;
const Container = styled.div`
  display: flex;
  justify-content: space-evenly;
  align-items: center;
`;
const LoginLeft = styled.div`
  background: url(${Bg});
  background-size: cover;
  width: 75%;
  margin-right:30px;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
`;
const LoginLeftContents = styled.div``;
const Title = styled.h3`
```

```
  color: #fff;
  font-weight: 700;
  font-size: 45px;
  margin-bottom: 10px;
`;
const Subtitle = styled.h4`
  color: #fff;
`;
const LeftButton = styled.a`
  background-color: #0575e6;
  padding: 8px 24px;
  color: #fff;
  border-radius: 12px;
`;
const LoginRight = styled.div`
  width: 30%;
  padding: 1%;
`;
const LoginContents = styled.div``;
const RIghtTitle = styled.h1`
  font-size: 45px;
  color: #0575e6;
  margin-bottom: 60px;
  margin-left: 80px;
`;
const BoldText = styled.b`
  font-size: 22px;
  color: #585858;
`;
const RightSubtitle = styled.div``;

const Home = () => {
  const emailRef = useRef("");
  const passwordRef = useRef("");
  const navigate = useNavigate();

  async function submitHandler(event) {
    event.preventDefault();
    const user = {
      email: emailRef.current.value,
      password: passwordRef.current.value,
    };

    const response = await fetch("http://localhost:8080/user/login", {
      method: "POST",
      body: JSON.stringify(user),
      headers: {
        "Content-Type": "application/json",
      },
    });

    if (!response.ok) {
      swal({
        title: "Invalid Email or Password",
        icon: "warning",
        buttons: true,
        dangerMode: true,
      });
    }
    if (emailRef.current.value === "admin@gmail.com") {
      navigate("/AdminDashboard");
    } else {
      navigate("/dashboard");
    }
  }

  return (
    <>
      <LoginPAge>
        <Container>
          <LoginLeft>
            <LoginLeftContents>
              <Title>Academic Evaluation System</Title>
              <Subtitle>
                A helping tool for both student and facultiesðŸ“• ðŸ“•
              </Subtitle>
              <LeftButton>ReadMore</LeftButton>
            </LoginLeftContents>
          </LoginLeft>
                <LoginRight>
            <LoginContents>
              <MainContainer>
              <WelcomeText>Welcome</WelcomeText>
                <InputContainer>
                <StyledInput
                  type="email"
                  id="email"
                  name="email"
                  required={true}
                  ref={emailRef}
                  placeholder="Email Address"
                />
                <StyledInput
                  type="password"
                  id="password"
                  name="password"
                  required={true}
                  ref={passwordRef}
                  placeholder="Password"
                />
                </InputContainer>
                <br />
                <ButtonContainer>
                <StyledButton
                  type="button"
                  className="button"
                  onClick={submitHandler}
                >
                  Login
                </StyledButton>
                </ButtonContainer>
              </MainContainer>
            </LoginContents>
          </LoginRight>
        </Container>
      </LoginPAge>
    </>
  );
 };



 const MainContainer = styled.div`
  display: flex;
  margin-top:auto;
  flex-direction: column;
  height: 65vh;
  width: 15vw;
  background: rgba(255, 255, 255, 0.15);
  box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
  backdrop-filter: blur(8.5px);
  -webkit-backdrop-filter: blur(8.5px);
  border-radius: 10px;
  color: #ffffff;
  text-transform: uppercase;
  letter-spacing: 0.4rem;
  @media only screen and (max-width: 320px) {
    width: 75vw;
    height:65vh;
    hr {
      margin-bottom: 0.3rem;
    }
    h4 {
      font-size: small;
    }
  }
  @media only screen and (min-width: 360px) {
    width: 75vw;
    height: 65vh;
    h4 {
      font-size: small;
    }
  }
  @media only screen and (min-width: 411px) {
    width: 75vw;
    height:65vh;
  }

  @media only screen and (min-width: 768px) {
    width: 75vw;
    height: 65vh;
  }
  @media only screen and (min-width: 1024px) {
    width: 65vw;
    height: 65vh;
  }
  @media only screen and (min-width: 1280px) {
    width: 25vw;
    height: 65vh;
  }
`;

const WelcomeText = styled.h2`
  color:black;
  top:10%;
  text-align:center;
`;

const InputContainer = styled.div`
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  align-items: center;
  height:50%;
  width:100%;
`;

const ButtonContainer = styled.div`
  margin: 1rem 0 2rem 0;
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
`;
const StyledButton = styled.button`
  background: linear-gradient(to right, #14163c 0%, #03217b 79%);
  text-transform: uppercase;
  letter-spacing: 0.2rem;
  width: 40%;
  height:3rem;
  border: none;
  color: white;
  border-radius: 2rem;
  cursor: pointer;
`;

const StyledInput = styled.input`
  background: rgba(255, 255, 255, 0.15);
  box-shadow: 0 8px 32px 0 rgba(31, 38, 135, 0.37);
  border-radius: 2rem;
  width: 80%;
  height:25%;
  padding: 1rem;
  border: none;
  outline: none;
  color: #3c354e;
  font-size:1.5rem;
  font-weight: bold;
  &:focus {
    display: inline-block;
    box-shadow: 0 0 0 0.2rem #b9abe0;
    backdrop-filter: blur(12rem);
    border-radius: 2rem;
  }
  &::placeholder {
    color: #b9abe099;
    font-weight: 100;
    font-size: 1rem;
  }
`;
export default Home;
```

5.2 AddStudent

5.2.1 AddStudent.js

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function AddStudent() {
  const [userData, setUserdata] = useState([]);

  useEffect(() => {
    const getUserdata = async () => {
      const reqData = await fetch(
        "http://localhost:8080/student/getAllStudents"
      );
      const resData = await reqData.json();
      setUserdata(resData);
      // console.log(resData);
    };
    getUserdata();
  }, []);

  function refreshPage() {
    window.location.reload(false);
  }

  function handleDelete(email) {
    fetch(`http://localhost:8080/student/deleteStudent/${email}`, {
      method: "DELETE",
    }).then((result) => {
      result.json().then((resp) => {
        console.log(resp);
      });
    });
    refreshPage();
  }

  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          <div className="container">
            <div className="row">
```

```
<div className="col-md-9">
  <div className="d-grid d-md-flex justify-content-md-end mb-3">
    <Link to="/addStudent" className="button">
      Add Student
    </Link>
  </div>
  <br />
  <br />
  <table className="table table-bordered table-striped">
    <thead>
      <tr>
        <th>Email</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Course</th>
      </tr>
    </thead>
    <tbody>
      {userData.map((userData, index) => (
        <tr key={index}>
          <td>{userData.email} </td>
          <td>{userData.firstName} </td>
          <td>{userData.lastName} </td>
          <td>{userData.course} </td>
          <td>
            <button
              onClick={() => alert(userData.email)}
              className="btn btn-success mx-auto"
            >
              Edit
            </button>{" "}
            &nbsp; &nbsp;
            <button
              onClick={() => handleDelete(userData.email)}
              className="btn btn-danger"
            >
              Delete
            </button>
            &nbsp; &nbsp;
            <Link
              to={"/showSub/" + userData.email}
              className="btn btn-warning"
            >
              Show Subjects
            </Link>
          </td>
        </tr>
```

```
          ))}
         </tbody>
        </table>
       </div>
      </div>
     </div>
    </Header>
   </React.Fragment>
  </div>
 );
}
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 20%;
`;

export default AddStudent;
```

5.2.2 ShowSub.js

```
 import React, { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function ShowSubject() {
 const { email } = useParams();
 const [userData, setUserdata] = useState([]);

 useEffect(() => {
  const request = {
   method: "GET",
   headers: {
    "Content-Type": "application/json",
   },
   body: JSON.stringify(),
  };
  fetch("http://localhost:8080/student/getAllSubjects/" + email, request)
   .then((response) => response.json())
   .then((detail) => {
    setUserdata(detail);
   });
 }, []);
 return (
   <div>
     <React.Fragment>
       <AdminDash />
       <Header>
         <div className="container">
           <div className="row">
             <div className="col-md-9">
               <div className="d-grid d-md-flex justify-content-md-end mb-3">
                 <Link id="route" to={"/student"} className="button">
                   Back
                 </Link>
                 <br />
                 <br />
                 <h5>subjects of student {email} are: </h5>
                 <br />
                 <table className="table table-bordered table-striped">
                   <thead>
                     <tr>
                       <th>Subject Code</th>
                       <th>Subject Name</th>
                     </tr>
                   </thead>
                   <tbody>
                     {userData.map((userData, index) => (
                       <tr key={index}>
                         <td>{userData.subjectCode} </td>
                         <td>{userData.subjectName} </td>
                       </tr>
                     ))}
                   </tbody>
                 </table>
               </div>
             </div>
           </div>
         </div>
       </Header>
     </React.Fragment>
   </div>
 );
}
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 10%;
`;
```

```
export default ShowSubject;
```

5.2.3 Student.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef } from "react";
import { useNavigate } from "react-router-dom";

const Student = () => {
  const navigate = useNavigate();
  const emailRef = useRef("");
  const firstNameRef = useRef("");
  const lastNameRef = useRef("");
  const addressRef = useRef("");
  const phoneRef = useRef("");
  const yearRef = useRef("");
  const sectionRef = useRef("");
  const fatherNameRef = useRef("");
  const motherNameRef = useRef("");
  const mentorRef = useRef("");
  const roleRef = useRef("");
  const passwordRef = useRef("");
  const rollNumRef = useRef("");
  const alternateEmailRef = useRef("");
  const dobRef = useRef("");
  const branchRef = useRef("");
  const courseRef = useRef("");
  const semesterRef = useRef("");

  async function submitHandler(event) {
    event.preventDefault();
    const studentData = {
      email: emailRef.current.value,
      firstName: firstNameRef.current.value,
      lastName: lastNameRef.current.value,
      address: addressRef.current.value,
      phone: phoneRef.current.value,
      year: yearRef.current.value,
      section: sectionRef.current.value,
      fatherName: fatherNameRef.current.value,
      motherName: motherNameRef.current.value,
      mentor: mentorRef.current.value,
      role: roleRef.current.value,
    password: passwordRef.current.value,
    rollNum: rollNumRef.current.value,
    alternateEmail: alternateEmailRef.current.value,
    dob: dobRef.current.value,
    branch: branchRef.current.value,
    course: courseRef.current.value,
    semester: semesterRef.current.value,
  };
  const response = await fetch("http://localhost:8080/student/addStudent", {
    method: "POST",
    body: JSON.stringify(studentData),
    headers: {
      "Content-Type": "application/json",
      " charset": "utf-8",
    },
  });
  if (response.ok) {
    navigate("/student");

    return response.json();
  }
}
return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="col-md-9">
            <form onSubmit={submitHandler}>
              <div className="row">
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">First Name</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={firstNameRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Last Name</label>
                    <input
                      type="text"
      name="username"
      className="form-control"
      ref={lastNameRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Date of birth</label>
    <input
      className="form-control"
      type="text"
      ref={dobRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Course</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={courseRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Branch</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={branchRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Semester</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={semesterRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Year</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={yearRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Section</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={sectionRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Roll Number</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={rollNumRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Mentor</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={mentorRef}
    />
  </div>
</div>
```

```
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Father Name</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={fatherNameRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Mother Name</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={motherNameRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Phone Number</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={phoneRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Email</label>
    <input
      type="text"
      name="email"
      className="form-control"
      ref={emailRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Password</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={passwordRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Alternate Email</label>
    <input
      type="text"
      name="phone"
      className="form-control"
      ref={alternateEmailRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Address</label>
    <input
      type="text"
      name="address"
      className="form-control"
      ref={addressRef}
    />
  </div>
</div>
<div className="col-md-6">
  <div className="mb-3">
    <label className="form-lable">Role</label>
    <input
      type="text"
      name="username"
      className="form-control"
      ref={roleRef}
    />
  </div>
</div>

<div className="col-md-12">
  <div className="mb-3">
    <label className="form-lable"></label>
    <button type="submit" className="btn btn-success btn-lg">
      Submit
```

```
          </button>
         </div>
        </div>
       </div>
      </form>
     </div>
    </div>
         </Header>
        </React.Fragment>
       </div>
     );
   };
   const Header = styled.div`
     position: absolute;
     left: 30%;
     top: 10%;
   `;

   export default Student;
```

5.3 AddSubject

5.3.1 AddSubject.js

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function AddSubject() {
  const [userData, setUserdata] = useState([]);

  useEffect(() => {
    const getUserdata = async () => {
      const reqData = await fetch(
        "http://localhost:8080/subject/getAllSubjects"
      );
      const resData = await reqData.json();
      setUserdata(resData);
    // console.log(resData);
  };
  getUserdata();
}, []);

function refreshPage() {
  window.location.reload(false);
}

function handleDelete(id) {
  fetch(`http://localhost:8080/subject/deleteSubject/${id}`, {
    method: "DELETE",
  }).then((result) => {
    result.json().then((resp) => {
      console.log(resp);
    });
  });
  refreshPage();
}

return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="row">
            <div className="col-md-9">
              <div className="d-grid d-md-flex justify-content-md-end mb-3">
                <Link id="route" to="/subject" className="button">
                  Add Subject
                </Link>
              </div>
              <br />
              <br />
              <table className="table table-bordered table-striped">
                <thead>
                  <tr>
```

```
}

                    <th>Subject Code</th>
                    <th>Subject Name</th>
                  </tr>
                </thead>
                <tbody>
                  {userData.map((userData, index) => (
                    <tr key={index}>
                      <td>{userData.subjectCode} </td>
                      <td>{userData.subjectName} </td>
                      <td>
                        <Link
                          to={"/editSubject/" + userData.subjectCode}
                          className="btn btn-success mx-2"
                        >
                          {" "}
                          Update
                        </Link>
                        &nbsp; &nbsp;
                        <button
                          onClick={() => handleDelete(userData.id)}
                          className="btn btn-danger"
                        >
                          {" "}
                          Delete
                        </button>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </Header>
    </React.Fragment>
  </div>
 );
}
```

```
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;

      export default AddSubject;
```

5.3.2 EditSubject.js

```
import React, { useEffect, useState, useRef } from "react";
import { useParams, useNavigate } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

const EditSubject = () => {
  const navigate = useNavigate();
  const { subjectCode } = useParams();
  const subjectNameRef = useRef("");

  const [subEdit, setSubEdit] = useState({
    subjectCode: subjectCode,
    subjectName: "",
  });

  useEffect(() => {
    const editSubject = async () => {
      const reqData = await fetch(
        "http://localhost:8080/subject/findSubject/" + subjectCode
      );
      setSubEdit(reqData);
    };
    editSubject();
  }, []);

  async function submitVal() {
    const item = {
      subjectName: subjectNameRef.current.value,
    };
    await fetch("http://localhost:8080/subject/updateSubject/" + subjectCode, {
      method: "PUT",
      body: JSON.stringify(item),
      headers: {
        "Content-Type": "application/json",
      Accept: "application/json",
    },
  });
}
function handleBack() {
  navigate("/addSubject");
}

return (
  <div>
    <AdminDash />
    <Header>
      <div className="container">
        <div className="col-md-9">
          <form onSubmit={submitVal}>
            <div className="row">
              <div className="col-md-6">
                <h1>Edit subject</h1>
                <div className="mb-3">
                  <label className="form-lable">Subject Code</label>
                  <input
                    type="text"
                    className="form-control"
                    defaultValue={subEdit.subjectCode}
                  />
                </div>
              </div>
              <div className="col-md-6">
                <div className="mb-3">
                  <label className="form-lable">Subject Name</label>
                  <input
                    type="text"
                    className="form-control"
                    ref={subjectNameRef}
                    defaultValue={subEdit.subjectName}
                  />
                </div>
              </div>

              <div className="col-md-12">
                <div className="mb-3">
                  <label className="form-lable"></label>
                  <button type="submit" className="btn btn-success btn-lg">
                    Submit
                  </button>
                </div>
              </div>
        <div className="col-md-12">
          <div className="mb-3">
            <label className="form-lable"></label>
            <button
              type="submit"
              className="btn btn-success btn-lg"
              style={{ marginRight: "500%", marginTop: "3%" }}
              onClick={handleBack}
            >
              Back
            </button>
          </div>
        </div>
      </div>
    </form>
  </div>
</div>
</Header>
</div>
);
};

const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;

            export default EditSubject;
```

5.3.3 Subjects.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef } from "react";
import { useNavigate } from "react-router-dom";

const Subject = () => {
  const navigate = useNavigate();

  const subjectCodeRef = useRef("");
const subjectNameRef = useRef("");

async function submitHandler(event) {
  event.preventDefault();
  const studentData = {
    subjectCode: subjectCodeRef.current.value,
    subjectName: subjectNameRef.current.value,
  };
  const response = await fetch("http://localhost:8080/subject/addSubject", {
    method: "POST",
    body: JSON.stringify(studentData),
    headers: {
      "Content-Type": "application/json",
    },
  });
  if (response.ok) {
    navigate("/addSubject");

    return response.json();
  }
}
return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="col-md-9">
            <form onSubmit={submitHandler}>
              <div className="row">
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Subject Code</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={subjectCodeRef}
                  />
                </div>
              </div>
              <div className="col-md-6">
                <div className="mb-3">
                  <label className="form-lable">Subject Name</label>
                  <input
                    type="text"
                    name="username"
                    className="form-control"
                    ref={subjectNameRef}
                  />
                </div>
              </div>

              <div className="col-md-12">
                <div className="mb-3">
                  <label className="form-lable"></label>
                  <button type="submit" className="btn btn-success btn-lg">
                    Submit
                  </button>
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>
    </Header>
  </React.Fragment>
  </div>
 );
};
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 30%;
`;
```

```
export default Subject;
```

5.4 AdminDashboard

5.4.1 AdminDash.js

```
import React from "react";
import { NavLink } from "react-router-dom";
import styled from "styled-components";
import UserProfile from "../../assets/images/user.jpg";
import Board from "../../assets/images/dashboard.png";
import Exam from "../../assets/images/exam.png";
import Attendance from "../../assets/images/attendance.png";
import Feedback from "../../assets/images/feedback.png";
import Assignment from "../../assets/images/assignments.png";
import Logout from "../../assets/images/logout.png";
import Report from "../../assets/images/report.png";

export default function AdminDash() {
  return (
    <>
      <Fixed>
        <Aside>
          <DivTitle>
            <Text> KIET 🪁🪁</Text>
          </DivTitle>
          <User>
            <UserLeft>
              <ImageContainer>
                <UserImage src={UserProfile} />
              </ImageContainer>
            </UserLeft>
            <UserRight>
              <UserNAme>Admin</UserNAme>
              <UserEmail>admin@gmail.com</UserEmail>
            </UserRight>
```

```
</User>
<Navbar>
  <Items>
    <NavLink
      to="/AdminPro"
      style={({ isActive }) =>
        isActive
          ? {
              color: "#fff",
              background: "#7600dc",
            }
          : { color: "#545e6f", background: "#f0f0f0" }
      }
    >
      <Item>
        <IconContainer>
          <Icon src={Board} />
        </IconContainer>
        <IconTitle>Dashboard</IconTitle>
      </Item>
    </NavLink>

    <NavLink to="/student">
      <Item>
        <IconContainer>
          <Icon src={Attendance} />
        </IconContainer>
        <IconTitle>Add Student</IconTitle>
      </Item>
    </NavLink>

    <NavLink to="/addSubject">
      <Item>
        <IconContainer>
          <Icon src={Exam} />
        </IconContainer>
        <IconTitle>Add Subjects</IconTitle>
      </Item>
```

```
</NavLink>

<NavLink to="/show">
  <Item>
    <IconContainer>
      <Icon src={Exam} />
    </IconContainer>
    <IconTitle>Assign Subjects</IconTitle>
  </Item>
</NavLink>
<NavLink to="/showAttendance">
  <Item>
    <IconContainer>
      <Icon src={Attendance} />
    </IconContainer>
    <IconTitle>Mark Attendance</IconTitle>
  </Item>
</NavLink>
<NavLink to="/mark">
  <Item>
    <IconContainer>
      <Icon src={Assignment} />
    </IconContainer>
    <IconTitle>Mark Internals</IconTitle>
  </Item>
</NavLink>
<NavLink to="/report">
  <Item>
    <IconContainer>
      <Icon src={Report} />
    </IconContainer>
    <IconTitle>Final Report</IconTitle>
  </Item>
</NavLink>
<Item>
  <IconContainer>
    <Icon src={Feedback} />
  </IconContainer>
```

```
                <IconTitle>Give Remarks</IconTitle>
              </Item>

              <NavLink to="/">
                <Item>
                  <IconContainer>
                    <Icon src={Logout} />
                  </IconContainer>
                  <IconTitle>LogOut</IconTitle>
                </Item>
              </NavLink>
            </Items>
          </Navbar>
        </Aside>
      </Fixed>
    </>
  );
}

const Fixed = styled.div`
  display: flex;
`;

const Aside = styled.aside`
  width: 20%;
  height: 100vh;
  box-shadow: 9px 6px 46px -24px rgba(0, 0, 0, 0.75);
`;

const DivTitle = styled.div`
  padding: 15px;
  border-bottom: 1px solid #c0c0c0;
`;
const Text = styled.h2`
  margin-left: 80px;
  color: #49b3f4;
  font-weight: bold;
`;
```

```
const User = styled.div`
  display: flex;
  margin-top: 70px;
  justify-content: center;
  margin-bottom: 2px;
`;
const UserLeft = styled.div`
  margin-right: 10px;
`;
const ImageContainer = styled.div``;
const UserImage = styled.img`
  border-radius: 50%;
`;
const Icon = styled.img`
  display: block;
  width: 100%;
`;
const IconContainer = styled.div`
  width: 10%;
  margin-right: 10px;
`;
const IconTitle = styled.a`
  color: #4e6481;
  font-weight: 600;
  cursor: pointer;
`;
const UserRight = styled.div``;
const UserNAme = styled.h4`
  margin: 0;
`;
const UserEmail = styled.small`
  color: #aca9a9;
`;
const Navbar = styled.div``;
const Items = styled.div`
  margin-top: 100px;
  margin-left: 30px;
`;
```

```
const Item = styled.div`
  display: flex;
  justify-content: start;
  margin-left: 20px;
  margin-bottom: 30px;
`;
```

5.4.2 AdminProfile.css

```
.form-control {
  border: 1px solid #cfd1d8;
  -webkit-border-radius: 2px;
  -moz-border-radius: 2px;
  border-radius: 2px;
  font-size: 0.825rem;
  background: #ffffff;
  color: #2e323c;
}

.card {
  background: #ffffff;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
  border: 0;

  margin-bottom: 1rem;
}
.container {
  margin-top: 50px;
}
input[type="text"],
select {
  background: url("../../assets/images/george.png");
  border: 0.5px solid black;
  padding: 16px 45px;
  margin-bottom: 20px;
  border-radius: 8px;
  background-repeat: no-repeat;
  background-position: left 10px center;
}
```

5.4.2 AdminProfile.js

```
import React, { useEffect, useState } from "react";
import styled from "styled-components";
import AdminDash from "./AdminDash";

const url = "http://localhost:8080/findAdmin/admin@gmail.com";

function AdminProfile() {
  const [userData, setUserData] = useState({});

  useEffect(() => {
    getGitHubUserWithFetch();
  }, []);

  const getGitHubUserWithFetch = async () => {
    const response = await fetch(url);
    const jsonData = await response.json();
    setUserData(jsonData);
  };

  return (
    <div>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="row gutters">
            <div className="col-xl-3 col-lg-3 col-md-12 col-sm-12 col-12"></div>
            <div className="col-xl-9 col-lg-9 col-md-12 col-sm-12 col-12">
              <div className="card h-100">
                <div className="card-body">
                  <div className="row gutters">
```

```
<div className="col-xl-12 col-lg-12 col-md-12 col-sm-12 col-12"></div>
<div className="col-xl-6 col-lg-6 col-md-6 col-sm-6 col-12">
  <div className="form-group">
    <label htmlFor="fullName">First Name</label>
    <input
      type="text"
      className="form-control"
      id="fullName"
      value={userData.firstName}
    />
  </div>
</div>
<div className="col-xl-6 col-lg-6 col-md-6 col-sm-6 col-12">
  <div className="form-group">
    <label htmlFor="eMail">LastName</label>
    <input
      type="email"
      className="form-control"
      id="fullName"
      value={userData.lastName}
    />
  </div>
</div>
<div className="col-xl-6 col-lg-6 col-md-6 col-sm-6 col-12">
  <div className="form-group">
    <label htmlFor="phone">Email</label>
    <input
      type="text"
      className="form-control"
      id="eMail"
      value={userData.email}
    />
  </div>
</div>
<div className="col-xl-6 col-lg-6 col-md-6 col-sm-6 col-12">
  <div className="form-group">
    <label htmlFor="website">Phone</label>
    <input
```

```
                    type="text"
                    className="form-control"
                    id="eMail"
                    value={userData.phone}
                  />
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
 </Header>
</div>
 );
}
const Header = styled.div`
 position: absolute;
 left: 20%;
 top: 20%;
`;

            export default AdminProfile;
```

5.5 AssignSubjects

5.5.1 AssignSubjects.js

```
import React, { useRef } from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useNavigate, useParams } from "react-router-dom";
import swal from "sweetalert";
```

```
const AssignSubject = () => {
  const { email } = useParams();
  const subjectCodeRef = useRef("");
  const navigate = useNavigate();

  async function submitHandler() {
    const item = {
      email: email,
      subjectCode: subjectCodeRef.current.value,
    };
    await fetch("http://localhost:8080/student/assignSubject", {
      method: "PUT",
      body: JSON.stringify(item),
      headers: {
        "Content-Type": "application/json",
        Accept: "application/json",
      },
    }).then((res) => {
      if (!res.ok) {
        swal({
          title: `this is does not exist`,
          icon: "warning",
          buttons: true,
          dangerMode: true,
        });
      } else {
        console.log("successfull");
      }
    });
  }
  function handleBack() {
    navigate("/showSubject/" + email);
  }
  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          <div className="container">
            <div className="col-md-9">
              <form onSubmit={submitHandler}>
                <div className="row">
                  <div className="col-md-6">
                    <div className="mb-3">
                      <label className="form-lable">Subject Code</label>
                      <input
              type="text"
```

```
                  name="username"
                  className="form-control"
                  ref={subjectCodeRef}
                 />
               </div>
              </div>

              <div className="col-md-6">
               <div className="mb-3">
                <label className="form-lable">Student Email</label>
                <input
                  type="text"
                  name="username"
                  className="form-control"
                  defaultValue={email}
                 />
               </div>
              </div>
             </div>
             <button className="button">Submit</button>
             <button className="button" onClick={handleBack}>
              Back
             </button>
            </form>
           </div>
          </div>
         </Header>
        </React.Fragment>
       </div>
  );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;
export default AssignSubject;
```

5.5.2 Show.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef, useState } from "react";
import { Link } from "react-router-dom";
const Show = () => {
  const courseRef = useRef("");
  const semesterRef = useRef("");
  const branchRef = useRef("");
  const yearRef = useRef("");

  const [userData, setUserData] = useState([]);

  async function submitHandler(event) {
    event.preventDefault();

    const item = {
      course: courseRef.current.value,
      year: yearRef.current.value,
      branch: branchRef.current.value,
      semester: semesterRef.current.value,
    };
    const request = {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(item),
    };
    fetch(
      "http://localhost:8080/student//getAllStudentsByCourseAndSemester",
      request
    )
      .then((response) => response.json())
      .then((detail) => {
        setUserData(detail);
      });
  }

  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          <p>Assign Subjects</p>
          <div className="container">
            <div className="col-md-9">
              <form onSubmit={submitHandler}>
                <div className="row">
                  <div className="col-md-6">
                    <div className="mb-3">
        <label className="form-lable">Course</label>
        <input
          type="text"
          name="username"
          className="form-control"
          ref={courseRef}
        />
      </div>
    </div>

    <div className="col-md-6">
      <div className="mb-3">
        <label className="form-lable">Year</label>
        <input
          type="number"
          name="username"
          className="form-control"
          ref={yearRef}
        />
      </div>
    </div>
    <div className="col-md-6">
      <div className="mb-3">
        <label className="form-lable">Branch</label>
        <input
          type="text"
          name="username"
          className="form-control"
          ref={branchRef}
        />
      </div>
    </div>
    <div className="col-md-6">
      <div className="mb-3">
        <label className="form-lable">Semester</label>
        <input
          type="text"
          name="username"
          className="form-control"
          ref={semesterRef}
        />
      </div>
    </div>

    <button className="button">Submit</button>
  </div>
</form>
```

```
              <table className="table table-bordered table-striped">
                <thead>
                  <tr>
                    <th>Email</th>
                    <th>First Name</th>
                    <th>Last Name</th>
                    <th>Course</th>
                  </tr>
                </thead>
                <tbody>
                  {userData.map((userData, index) => (
                    <tr key={index}>
                      <td>{userData.email} </td>
                      <td>{userData.firstName} </td>
                      <td>{userData.lastName} </td>
                      <td>{userData.course} </td>
                      <td>
                        <Link
                          to={"/showSubject/" + userData.email}
                          className="btn btn-warning"
                        >
                          Show Subjects
                        </Link>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </table>
            </div>
          </div>
        </Header>
      </React.Fragment>
    </div>
  );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;

export default Show;
```

5.5.3 ShowSubjects.js

```
import React, { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function ShowSubject() {
  const { email } = useParams();
  const [userData, setUserdata] = useState([]);

  useEffect(() => {
    const request = {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(),
    };
    fetch("http://localhost:8080/student/getAllSubjects/" + email, request)
      .then((response) => response.json())
      .then((detail) => {
        setUserdata(detail);
      });
  }, []);

  function refreshPage() {
    window.location.reload(false);
  }

  function handleDelete(subjectCode) {
    const item = {
      email: email,
      subjectCode: subjectCode,
    };
    console.log(item);
    fetch(`http://localhost:8080/student/delAssignSubject`, {
      method: "DELETE",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(item),
    }).then((result) => {
      result.json().then((resp) => {
        console.log(resp);
      });
  });
  refreshPage();
 }

 return (
   <div>
     <React.Fragment>
       <AdminDash />
       <Header>
         <div className="container">
           <div className="row">
             <div className="col-md-9">
               <div className="d-grid d-md-flex justify-content-md-end mb-3">
                 <Link
                   id="route"
                   to={`/assignSubject/${email}`}
                   className="button"
                 >
                   Add More
                 </Link>
                 <Link id="route" to={`/show`} className="button">
                   Back
                 </Link>
               </div>
               <br />
               <br />
               <h5>subjects of student {email} are: </h5>
               <table className="table table-bordered table-striped">
                 <thead>
                   <tr>
                     <th>Subject Code</th>
                     <th>Subject Name</th>
                   </tr>
                 </thead>
                 <tbody>
                   {userData.map((userData, index) => (
                     <tr key={index}>
                       <td>{userData.subjectCode} </td>
                       <td>{userData.subjectName} </td>
                       <td>
                         <button
                           onClick={() => handleDelete(userData.subjectCode)}
                           className="btn btn-danger"
                         >
                           {" "}
                           Delete
                         </button>
```

```
                    </td>
                  </tr>
                ))}
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </Header>
  </React.Fragment>
  </div>
 );
}
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 20%;
`;

export default ShowSubject;
```

5.6 Attendance

5.6.1 MarkAttendance.js

```
import React, { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function MarkAttendance() {
 const { email } = useParams();
 const [userData, setUserdata] = useState([]);
 let val = false;

 useEffect(() => {
  const request = {
   method: "GET",
   headers: {
    "Content-Type": "application/json",
   },
   body: JSON.stringify(),
  };
  fetch("http://localhost:8080/student/getAllSubjects/" + email, request)
    .then((response) => response.json())
    .then((detail) => {
      setUserdata(detail);
    });
}, []);

function setValue() {
  val = true;
}
console.log(val);

return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="row">
            <div className="col-md-9">
              <div className="d-grid d-md-flex justify-content-md-end mb-3">
                <Link id="route" to={"/"} className="button">
                  Back
                </Link>
                <br />
                <br />
                <h5>subjects of student {email} are: </h5>
                <br />
                <table className="table table-bordered table-striped">
                  <thead>
                    <tr>
                      <th>Subject Code</th>
                      <th>Subject Name</th>
                      <th>Present%</th>
                      <th>Absent%</th>
                    </tr>
                  </thead>
                  <tbody>
                    {userData.map((userData, index) => (
                      <tr key={index}>
                        <td>{userData.subjectCode} </td>
                        <td>{userData.subjectName} </td>
                        <td>90%</td>
                        <td>10%</td>
                        <td>
                          <button
                            type="submit"
              onClick={setValue}
              className="btn btn-success"
             >
              Present
             </button>
             &nbsp; &nbsp; &nbsp; &nbsp;
             <button type="submit" className="btn btn-danger">
              Absent
             </button>
            </td>
           </tr>
          ))}
         </tbody>
        </table>
       </div>
      </div>
     </div>
    </div>
   </Header>
  </React.Fragment>
 </div>
 );
}
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 10%;
`;

export default MarkAttendance;
```

5.6.2 ShowAttendance.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef, useState } from "react";
import { Link } from "react-router-dom";
const ShowAttendance = () => {
 const courseRef = useRef("");
 const semesterRef = useRef("");
 const branchRef = useRef("");
 const yearRef = useRef("");

 const [userData, setUserData] = useState([]);
```

```
async function submitHandler(event) {
  event.preventDefault();

  const item = {
    course: courseRef.current.value,
    year: yearRef.current.value,
    branch: branchRef.current.value,
    semester: semesterRef.current.value,
  };
  const request = {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(item),
  };
  fetch(
    "http://localhost:8080/student//getAllStudentsByCourseAndSemester",
    request
  )
    .then((response) => response.json())
    .then((detail) => {
      setUserData(detail);
    });
}

return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <p>Attendance</p>
        <div className="container">
          <div className="col-md-9">
            <form onSubmit={submitHandler}>
              <div className="row">
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Course</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={courseRef}
                    />
                  </div>
                </div>

                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Year</label>
                    <input
                      type="number"
                      name="username"
                      className="form-control"
                      ref={yearRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Branch</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={branchRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Semester</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={semesterRef}
                    />
                  </div>
                </div>

                <button className="button">Submit</button>
              </div>
            </form>
            <table className="table table-bordered table-striped">
              <thead>
                <tr>
                  <th>Email</th>
                  <th>First Name</th>
                  <th>Last Name</th>
                  <th>Course</th>
                </tr>
```

```
        </thead>
        <tbody>
          {userData.map((userData, index) => (
            <tr key={index}>
              <td>{userData.email} </td>
              <td>{userData.firstName} </td>
              <td>{userData.lastName} </td>
              <td>{userData.course} </td>
              <td>
                <Link
                  to={"/markAttendance/" + userData.email}
                  className="btn btn-warning"
                >
                  Show Attendance
                </Link>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
   </div>
  </Header>
 </React.Fragment>
</div>
 );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;

export default ShowAttendance;
```

5.7 Contact

```
import React from "react";
import Dashboard from "../Dashboard/Dashboard";
import styled from "styled-components";
import "../Home/Home.css";
function Contact() {
  return (
    <div>
      <Dashboard />
      <Header>
        <div className="container min-vh=100">
          <div className="card ">
            <div className="col-xl-5 col-lg-6 col-md-8 col-sm-10 mx-auto text-center form p-4">
              <form>
                <div className="mb-3">
                  <input
                    className="form-control"
                    type="text"
                    id="name"
                    placeholder="Name"
                    required
                  />
                </div>
                <br />
                <div className="mb-3">
                  <input
                    className="form-control"
                    type="email"
                    placeholder="Email"
                    id="email"
                    required
                  />
                </div>
                <div className="mb-3">
                  <textarea
                    className="form-control"
                    rows="10"
                    id="message"
                    placeholder="Enter your message..."
                    required
                  />
                </div>
                <div class="container text-center">
                  <button type="submit" className="button">
                    Send
                  </button>
                </div>
              </form>
            </div>
          </div>
        </div>
      </Header>
    </div>
  );
```

```
}

const Header = styled.div`
  position: absolute;
  left: 40%;
  top: 20%;
`;
export default Contact;
```

5.8 Dashboard

```
import React from "react";
import { NavLink } from "react-router-dom";
import styled from "styled-components";
import UserProfile from "../../assets/images/user.jpg";
import Board from "../../assets/images/dashboard.png";
import Exam from "../../assets/images/exam.png";
import Attendance from "../../assets/images/attendance.png";
import Feedback from "../../assets/images/feedback.png";
import Contacts from "../../assets/images/contacts.png";
import Assignment from "../../assets/images/assignments.png";
import Logout from "../../assets/images/logout.png";
import Report from "../../assets/images/report.png";

export default function Dashboard() {
  return (
    <>
      <Fixed>
        <Aside>
          <DivTitle>
            <Text> KIET 🪁🪁</Text>
          </DivTitle>
          <User>
            <UserLeft>
              <ImageContainer>
                <UserImage src={UserProfile} />
              </ImageContainer>
            </UserLeft>
            <UserRight>
              <UserNAme>Kiran Chauhan</UserNAme>
              <UserEmail>123@gmail.com</UserEmail>
            </UserRight>
          </User>
          <Navbar>
            <Items>
```

```
<Item>
  <IconContainer>
    <Icon src={Board} />
  </IconContainer>
  <IconTitle>Dashboard</IconTitle>
</Item>
<NavLink to="/internal">
  <Item>
    <IconContainer>
      <Icon src={Exam} />
    </IconContainer>

    <IconTitle>Internal Marks</IconTitle>
  </Item>
</NavLink>
<Item>
  <IconContainer>
    <Icon src={Attendance} />
  </IconContainer>
  <IconTitle>Attendance</IconTitle>
</Item>
<Item>
  <IconContainer>
    <Icon src={Assignment} />
  </IconContainer>
  <IconTitle>Assignment</IconTitle>
</Item>
<Item>
  <IconContainer>
    <Icon src={Report} />
  </IconContainer>
  <IconTitle>Final Report</IconTitle>
</Item>
<Item>
  <IconContainer>
    <Icon src={Feedback} />
  </IconContainer>
  <IconTitle>Remarks</IconTitle>
</Item>
<NavLink to="/contact">
  <Item>
    <IconContainer>
      <Icon src={Contacts} />
    </IconContainer>
    <IconTitle>Contacts</IconTitle>
  </Item>
</NavLink>
```

```
              <NavLink to="/">
                <Item>
                  <IconContainer>
                    <Icon src={Logout} />
                  </IconContainer>
                  <IconTitle>LogOut</IconTitle>
                </Item>
              </NavLink>
            </Items>
          </Navbar>
        </Aside>
      </Fixed>
    </>
  );
}

const Fixed = styled.div`
  display: flex;
`;

const Aside = styled.aside`
  width: 20%;
  height: 100vh;
  box-shadow: 9px 6px 46px -24px rgba(0, 0, 0, 0.75);
`;
const DivTitle = styled.div`
  padding: 15px;
  border-bottom: 1px solid #c0c0c0;
`;
const Text = styled.h2`
  margin: 5px;
  color: #49b3f4;
  font-weight: bold;
`;
const User = styled.div`
  display: flex;
  margin-top: 70px;
  justify-content: center;
  margin-bottom: 2px;
`;
const UserLeft = styled.div`
  margin-right: 10px;
`;
const ImageContainer = styled.div``;
const UserImage = styled.img`
  border-radius: 50%;
`;
```

```
const Icon = styled.img`
  display: block;
  width: 100%;
`;
const IconContainer = styled.div`
  width: 10%;
  margin-right: 10px;
`;
const IconTitle = styled.a`
  color: #4e6481;
  font-weight: 600;
  cursor: pointer;
`;
const UserRight = styled.div``;
const UserNAme = styled.h4`
  margin: 0;
`;
const UserEmail = styled.small`
  color: #aca9a9;
`;
const Navbar = styled.div``;
const Items = styled.div`
  margin-top: 100px;
  margin-left: 30px;
`;
const Item = styled.div`
  display: flex;
  justify-content: start;
  margin-left: 20px;
  margin-bottom: 30px;
`;
```

5.8 FinalReport

5.8.1 Report.js

```
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function Report() {
  return (
    <div>
```

```
      <AdminDash />
      <Header>
        <h1>Hello</h1>
      </Header>
    </div>
  );
}
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;
export default Report;
```

5.9 MarkInternals

5.9.1 AddExam.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef } from "react";
import { useNavigate, useParams } from "react-router-dom";

const AddExam = () => {
  const { email } = useParams();
  const navigate = useNavigate();
  const subjectCodeRef = useRef("");
  const subjectNameRef = useRef("");
  const examTypeRef = useRef("");
  const maxExamMarksRef = useRef("");
  const obtainedExamMarksRef = useRef("");

  async function submitHandler(event) {
    event.preventDefault();
    const studentData = {
      subjectCode: subjectCodeRef.current.value,
      subjectName: subjectNameRef.current.value,
      examType: examTypeRef.current.value,
      maxExamMarks: maxExamMarksRef.current.value,
      obtainedExamMarks: obtainedExamMarksRef.current.value,
    };
    const response = await fetch("http://localhost:8080/subject/addSubject", {
      method: "POST",
    body: JSON.stringify(studentData),
    headers: {
      "Content-Type": "application/json",
    },
  });
  if (response.ok) {
    navigate("/markExams/" + email);

    return response.json();
  }
}
return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        <div className="container">
          <div className="col-md-9">
            <form onSubmit={submitHandler}>
              <div className="row">
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Subject Code</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={subjectCodeRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Subject Name</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={subjectNameRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Exam Type</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={examTypeRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Maximum Exam Marks</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={maxExamMarksRef}
                    />
                  </div>
                </div>
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Obtained Exam Marks</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={obtainedExamMarksRef}
                    />
                  </div>
                </div>

                <div className="col-md-12">
                  <div className="mb-3">
                    <label className="form-lable"></label>
                    <button type="submit" className="btn btn-success btn-lg">
                      Submit
                    </button>
                  </div>
                </div>
              </div>
            </form>
          </div>
        </div>
      </Header>
    </React.Fragment>
  </div>
);
};
const Header = styled.div`
```

```
  position: absolute;
  left: 30%;
  top: 30%;
`;

export default AddExam;
```

5.9.2 EditAssignmentMarks.js

```
import React, { useRef } from "react";
import { useParams, useNavigate } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

const EditAssignmentMark = () => {
  const { subjectCode } = useParams();
  const assignTypeRef = useRef("");
  const obtainedAssignMarksRef = useRef("");
  const maxAssignMarksRef = useRef("");
  const navigate = useNavigate();

  async function submitVal() {
    const item = {
      obtainedAssignMarks: obtainedAssignMarksRef.current.value,
      maxAssignMarks: maxAssignMarksRef.current.value,
      assignType: assignTypeRef.current.value,
    };
    await fetch(
      "http://localhost:8080/subject/updateAssignments/" + subjectCode,
      {
        method: "PUT",
        body: JSON.stringify(item),
        headers: {
          "Content-Type": "application/json",
          Accept: "application/json",
        },
      }
    );
  }
  function handleBack() {
    navigate("/markAssignment/123@gmail.com");
  }
  return (
    <div>
```

```
<AdminDash />
<Header>
  <div className="container">
    <div className="col-md-9">
      <form onSubmit={submitVal}>
        <div className="row">
          <div className="col-md-6">
            <div className="mb-3">
              <label className="form-lable">Assignment Type</label>
              <input
                type="text"
                className="form-control"
                ref={assignTypeRef}
              />
            </div>
          </div>
          <div className="col-md-6">
            <div className="mb-3">
              <label className="form-lable">Obtained Marks</label>
              <input
                type="text"
                className="form-control"
                ref={obtainedAssignMarksRef}
              />
            </div>
          </div>
          <div className="col-md-6">
            <div className="mb-3">
              <label className="form-lable">Maximum Marks</label>
              <input
                type="text"
                className="form-control"
                ref={maxAssignMarksRef}
              />
            </div>
          </div>

          <div className="col-md-12">
            <div className="mb-3">
              <label className="form-lable"></label>
              <button type="submit" className="btn btn-success btn-lg">
                Submit
              </button>
            </div>
          </div>

          <div className="col-md-12">
```

```
                <div className="mb-3">
                  <label className="form-lable"></label>
                  <button
                    type="submit"
                    className="btn btn-success btn-lg"
                    style={{ marginRight: "500%", marginTop: "3%" }}
                    onClick={handleBack}
                  >
                    Back
                  </button>
                </div>
              </div>
            </div>
          </form>
        </div>
      </div>
    </Header>
  </div>
 );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;

export default EditAssignmentMark;
```

5.9.3 EditExamMarks.js

```
import React, { useRef } from "react";
import { useParams, useNavigate } from "react-router-dom";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

const EditExamMarks = () => {
  const { subjectCode } = useParams();
  const obtainedExamMarksRef = useRef("");
  const maxExamMarksRef = useRef("");
  const examTypeRef = useRef("");
  const navigate = useNavigate();

  async function submitVal() {
```

```
  const item = {
    obtainedExamMarks: obtainedExamMarksRef.current.value,
    maxExamMarks: maxExamMarksRef.current.value,
    examType: examTypeRef.current.value,
  };
  await fetch("http://localhost:8080/subject/updateMarks/" + subjectCode, {
    method: "PUT",
    body: JSON.stringify(item),
    headers: {
      "Content-Type": "application/json",
      Accept: "application/json",
    },
  });
}
function handleBack() {
  navigate("/markExams/123@gmail.com");
}
return (
  <div>
    <AdminDash />
    <Header>
      <div className="container">
        <div className="col-md-9">
          <form onSubmit={submitVal}>
            <div className="row">
              <div className="col-md-6">
                <div className="mb-3">
                  <label className="form-lable">Exam type</label>
                  <input
                    type="text"
                    className="form-control"
                    ref={examTypeRef}
                  />
                </div>
              </div>
              <div className="col-md-6">
                <div className="mb-3">
                  <label className="form-lable">Obtained Marks</label>
                  <input
                    type="text"
                    className="form-control"
                    ref={obtainedExamMarksRef}
                  />
                </div>
              </div>
              <div className="col-md-6">
                <div className="mb-3">
          <label className="form-lable">Maximum Marks</label>
          <input
            type="text"
            className="form-control"
            ref={maxExamMarksRef}
          />
        </div>
      </div>

      <div className="col-md-12">
        <div className="mb-3">
          <label className="form-lable"></label>
          <button type="submit" className="btn btn-success btn-lg">
            Submit
          </button>
        </div>
      </div>

      <div className="col-md-12">
        <div className="mb-3">
          <label className="form-lable"></label>
          <button
            type="submit"
            className="btn btn-success btn-lg"
            style={{ marginRight: "500%", marginTop: "3%" }}
            onClick={handleBack}
          >
            Back
          </button>
        </div>
      </div>
    </div>
   </form>
  </div>
 </div>
 </Header>
</div>
);
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;

export default EditExamMarks
```

5.9.4 MarkAssignment.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";

const MarkAssignment = () => {
  const { email } = useParams();
  const [userData, setUserdata] = useState([]);

  useEffect(() => {
    const request = {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(),
    };
    fetch("http://localhost:8080/student/getAllSubjects/" + email, request)
      .then((response) => response.json())
      .then((detail) => {
        setUserdata(detail);
      });
  }, []);

  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          {" "}
          <div className="container">
            <div className="row">
              <div className="col-md-9">
                <div className="d-grid d-md-flex justify-content-md-end mb-3">
                  <Link id="route" to={"/mark"} className="button">
                    Back
                  </Link>
                  <Link id="route" to={"/"} className="button">
                    Add More
                  </Link>
                  <br />
          <br />

          <br />
          <table className="table table-bordered table-striped">
            <thead>
              <tr>
                <th>Subject Code</th>
                <th>Subject Name</th>
                <th>Assignment Type</th>
                <th> Assignment Marks</th>
                <th>Max Assignment Marks</th>
              </tr>
            </thead>
            <tbody>
              {userData.map((userData, index) => (
                <tr key={index}>
                  <td>{userData.subjectCode} </td>
                  <td>{userData.subjectName} </td>
                  <td>{userData.assignType}</td>
                  <td>{userData.obtainedAssignMarks} </td>
                  <td>{userData.maxAssignMarks} </td>

                  <td>
                    &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                    <Link
                      to={"/editAssignments/" + userData.id}
                      className="btn btn-warning"
                    >
                      Update
                    </Link>
                    &nbsp; &nbsp; &nbsp; &nbsp;
                    <Link
                      to={"/viewAssignment/" + userData.subjectCode}
                      className="btn btn-warning"
                    >
                      View Submission
                    </Link>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      </div>
    </div>
  </div>
</Header>
```

```
      </React.Fragment>
    </div>
  );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;

export default MarkAssignment;
```

5.9.5 MarkExams.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useEffect, useState } from "react";
import { Link, useParams } from "react-router-dom";

const MarkExams = () => {
  const { email } = useParams();
  const [userData, setUserdata] = useState([]);

  useEffect(() => {
    const request = {
      method: "GET",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(),
    };
    fetch("http://localhost:8080/student/getAllSubjects/" + email, request)
      .then((response) => response.json())
      .then((detail) => {
        setUserdata(detail);
      });
  }, []);

  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          {" "}
```

```
<div className="container">
  <div className="row">
    <div className="col-md-9">
      <div className="d-grid d-md-flex justify-content-md-end mb-3">
        <Link id="route" to={"/addExams/" + email} className="button">
          Add Exams
        </Link>
        <Link id="route" to={"/mark"} className="button">
          Back
        </Link>
        <br />
        <br />
        <h5>Marks of subjects of student with {email} are: </h5>
        <br />
        <table className="table table-bordered table-striped">
          <thead>
            <tr>
              <th>Subject Code</th>
              <th>Subject Name</th>
              <th>Exam Type</th>
              <th>Obtained Marks</th>
              <th>Maximum Marks</th>
            </tr>
          </thead>
          <tbody>
            {userData.map((userData, index) => (
              <tr key={index}>
                <td>{userData.subjectCode} </td>
                <td>{userData.subjectName} </td>
                <td>{userData.examType}</td>
                <td>{userData.obtainedExamMarks} </td>
                <td>{userData.maxExamMarks} </td>
                <td>
                  &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
                  <Link
                    to={"/editExams/" + userData.id}
                    className="btn btn-warning"
                  >
                    Update
                  </Link>
                </td>
              </tr>
            ))}
          </tbody>
        </table>
      </div>
    </div>
```

```
          </div>
        </div>
      </Header>
    </React.Fragment>
  </div>
 );
};
const Header = styled.div`
 position: absolute;
 left: 30%;
 top: 10%;
`;

export default MarkExams;
```

5.9.6 MarkInternals.js

```
import React from "react";
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import { useRef, useState } from "react";
import { Link } from "react-router-dom";

const MarkInternals = () => {
 const courseRef = useRef("");
 const semesterRef = useRef("");
 const branchRef = useRef("");
 const yearRef = useRef("");

 const [userData, setUserData] = useState([]);

 async function submitHandler(event) {
   event.preventDefault();

   const item = {
     course: courseRef.current.value,
     year: yearRef.current.value,
     branch: branchRef.current.value,
     semester: semesterRef.current.value,
   };
   const request = {
     method: "POST",
     headers: {
       "Content-Type": "application/json",
    },
    body: JSON.stringify(item),
  };
  fetch(
    "http://localhost:8080/student//getAllStudentsByCourseAndSemester",
    request
  )
    .then((response) => response.json())
    .then((detail) => {
      setUserData(detail);
    });
}

return (
  <div>
    <React.Fragment>
      <AdminDash />
      <Header>
        {" "}
        <div className="container">
          <div className="col-md-9">
            <form onSubmit={submitHandler}>
              <p>Mark internals</p>
              <div className="row">
                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Course</label>
                    <input
                      type="text"
                      name="username"
                      className="form-control"
                      ref={courseRef}
                    />
                  </div>
                </div>

                <div className="col-md-6">
                  <div className="mb-3">
                    <label className="form-lable">Year</label>
                    <input
                      type="number"
                      name="username"
                      className="form-control"
                      ref={yearRef}
                    />
                  </div>
                </div>
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-lable">Branch</label>
          <input
            type="text"
            name="username"
            className="form-control"
            ref={branchRef}
          />
        </div>
      </div>
      <div className="col-md-6">
        <div className="mb-3">
          <label className="form-lable">Semester</label>
          <input
            type="text"
            name="username"
            className="form-control"
            ref={semesterRef}
          />
        </div>
      </div>

      <button className="button">Submit</button>
    </div>
  </form>
  <table className="table table-bordered table-striped">
    <thead>
      <tr>
        <th>Email</th>
        <th>First Name</th>
        <th>Last Name</th>
        <th>Course</th>
      </tr>
    </thead>
    <tbody>
      {userData.map((userData, index) => (
        <tr key={index}>
          <td>{userData.email} </td>
          <td>{userData.firstName} </td>
          <td>{userData.lastName} </td>
          <td>{userData.course} </td>
          <td>
            <Link
              to={"/markExams/" + userData.email}
              className="btn btn-warning"
            >
```

```
                Exam Marks
              </Link>
              &nbsp; &nbsp;
              <Link
                to={"/markAssignment/" + userData.email}
                className="btn btn-warning"
              >
                Assignment Marks
              </Link>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
 </div>
</Header>
</React.Fragment>
    </div>
  );
};
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;

export default MarkInternals;
```

5.9.7 ViewAssignment.js

```
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";
import React from "react";
import Assignment_1 from "../../assets/images/PDF/Assignment_1.pdf";
import Assignment1_Ans from "../../assets/images/PDF/Assignment1_Ans.pdf";
import { Link } from "react-router-dom";

function ViewAssignment() {
  return (
    <div>
      <React.Fragment>
        <AdminDash />
        <Header>
          <div>
            <a href={Assignment_1} target="_blank" rel="noreferrer">
              Assignment 1
            </a>
          </div>
          <br></br>
          <div>
            <a href={Assignment1_Ans} target="_blank" rel="noreferrer">
              Student Assignment
            </a>{" "}
          </div>
          <Link id="route" to={"/markAssignment"} className="button">
            Back
          </Link>
        </Header>
      </React.Fragment>
    </div>
  );
}
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 10%;
`;

export default ViewAssignment;
```

5.10 Remark

5.10.1 AddRemark.js

```
import AdminDash from "../AdminDashboard/AdminDash";
import styled from "styled-components";

function AddRemark() {
 return (
  <div>
   <AdminDash />
   <Header>
    <h1>Hello</h1>
   </Header>
  </div>
 );
}
```

```
const Header = styled.div`
  position: absolute;
  left: 30%;
  top: 20%;
`;

export default AddRemark;
```

CHAPTER 6

TESTING

Software Testing is evaluation of the software against requirements gathered from users and system specifications. Testing is conducted at the phase level in software development life cycle or at module level in program code. Software testing comprises of Validation and Verification.

6.1 Unit Testing

While coding, the programmer performs some tests on that unit of program to know if it is error free. Testing is performed under white-box testing approach. Unit testing helps developers decide that individual units of the program are working as per requirement and are error free.

6.2 Integration Testing

Even if the units of software are working fine individually, there is a need to find out if the units if integrated together would also work without errors. For example, argument passes and data updating etc.

6.3 System Testing

The software is compiled as product and then it is tested. This can be accomplished using one or more of the following tests

6.4 Functionality testing

Tests all functionalities of the software against the requirement.

6.5 Performance testing

This test proves how efficient the software is. It tests the effectiveness and average time taken by the software to do desired task. Performance testing is done by means of load testing and stress testing where the software is put under high user and data load under various environment conditions.

6.6 Acceptance Testing

When the software is ready to hand over to the customer it must go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

6.7 Alpha testing

The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.

6.7 Beta testing

After the software is tested internally, it is handed over to the users to use it under their features ion environment only for testing purpose. This is not yet the delivered features. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

6.8 Test Cases for Homepage

- Verify that the home page is displayed after login or not.
- Verify that the Username is displayed on homepage or not.
- Verify that displayed features are present on home page or not.
- Verify that Search functionality is present on home page or not.
- Verify the home page of application on different browsers.
- Verify the alignment on the home page.
- Verify that features displayed on home page are clickable or not.
- Verify that features displayed on home page are categorized or not.

6.9 Test Cases for Dashboard

- Verify that the images of features are displayed correctly or not.
- Verify that the price of features is displayed or not.
- Verify that features reviews are mentioned or not.

REFERENCES

[1] Ch. Aswani Kumar; K. Sumangali,” Performance evaluation of employees of an organization using formal cept analysis”, International Conference on Pattern Recognition, Informatics and Medical Engineering (PRIME-2012)

[2] Shaomei Yang; Qian Zhu, “An Evaluation Model on Employee Performance Based on Improved BP Neural Network”, 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing.

[3] TANG Yu-fang, ZHANG Yong-sheng,,—Design and implementation of college student information management system based on the web services‖. Natural Science Foundation of Shandong Province(Y2008G22), 978-1- 4244-3930-0/09 2009 IEEE.

[4] TANG Yu-fang, ZHANG Yong-sheng,,—Design and implementation of college student information management system based on the web services‖. Natural Science Foundation of Shandong Province(Y2008G22), 978-1- 4244-3930-0/09 2009 IEEE.