Datenbanken und Web-Techniken

Project Term Paper

SUBMITTED BY

Name: Yasin Arafat
Matriculation # 612254

Automotive Software Engineering

July 5, 2021

PROJECT NAME

Bright Star Grading System

Bright Star Grading System is a project for the final exam of "Datenbanken und Web-Techniken". This project is targeted for the advanced online student grading system managed by school. In this project, a simple grading system is developed where teachers and students can find information about classes, subjects, and test grades. There were various other functionalities implemented according to the project requirements.

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Contents

Figures

Tables

1 Introduction

The aim of this project is to develop a system for school where the teachers can easily manage test results of students to a degree of digital conversion and can come out of the traditional paper-based grades management system. The system is initially managed and maintained by an administrative panel. Secondarily, the system also gives necessary access to the teachers and students with separate user-friendly interfaces. So, this project will offer a platform to modernize school grading system.

2 Project

2.1 Primary Requirements

The primary requirement of the project is to build a system where there must be a local database, which is communicated through API by the use of a backend. There should also be a frontend just to provide a user interface to the end users. The front end never communicates directly with the database.

Figure-1: Structure [5]

2.2 Database

2.2.1 Relational Schema

Relational schema denotes to the meta-data that describes the erection of data within a convinced domain. A relational schema for a database is the framework of how data is prearranged. It typically stipulates which columns in which tables comprise orientations to data in other tables, frequently by comprising primary keys from other table so that rows can be easily joined.

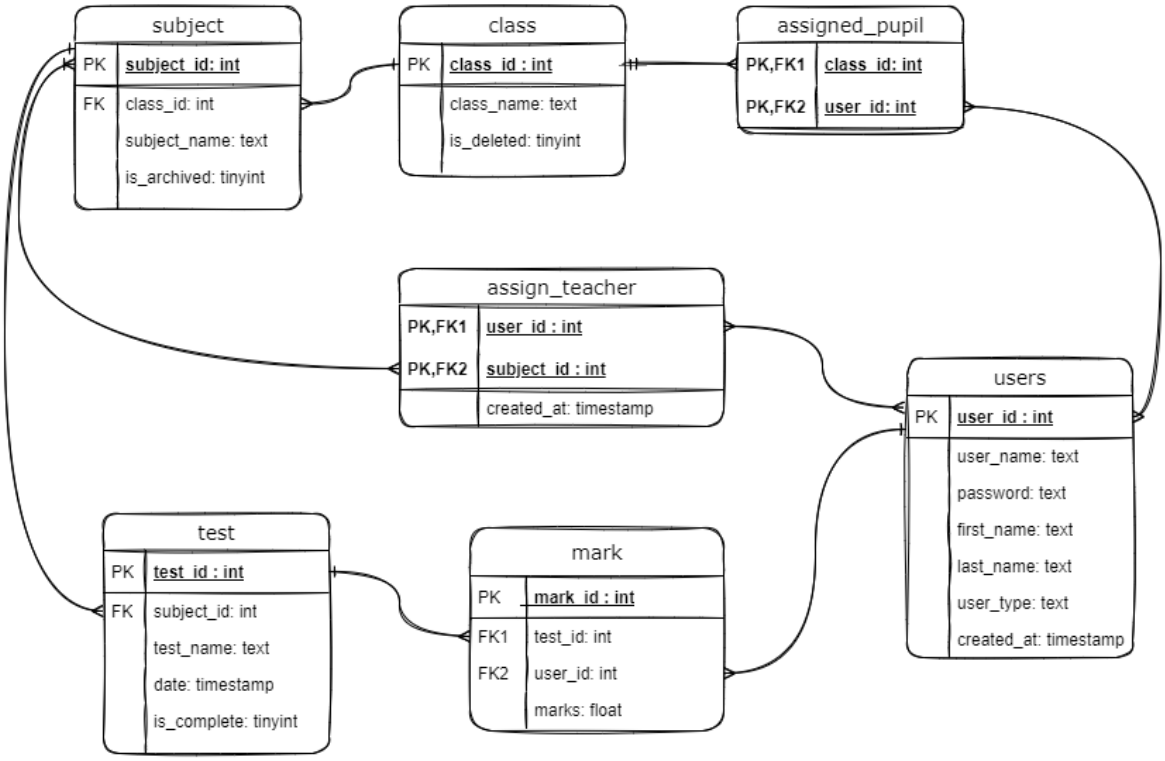


Figure-2: Relational Schema

Relational schema has a very important role for developing the project. First we did this relational schema even before starting to write program. Because relational schema helps us to understand how the tables of the database are connected to each other using foreign key and primary key.

2.3 Functionalities

This project is mainly consisting of three user interfaces. The main interface is for the admin and the other two is for teachers and students. According to the requirements of the project

- All the data were stored in a database and no additional storages were used.
- The frontend communicates with the backend using API routes and finally the backend communicates with the database to run the system.

User	Task
Admin view	• Login • Add/edit user • Add/edit class • Add/edit subject • Assign/de-assign student • Assign teacher
Teacher view	• Login • Edit personal details • Add/edit tests • Add/edit grades
Student view	• Login • Edit personal details

Table-1: UI – Functionalities

2.4 Project operations

This project is mainly consisting of three user interfaces. The main interface is for the admin and the other two is for teachers and students. According to the requirements of the project, all the data were stored in a database and no additional storage were used.

2.4.1 Home

The home menu consists of a single login panel for all the users. The provided login data dynamically identifies the user and leads them to their respective views. There are three different views for three different user types. Those are:

- Admin view
- Teacher view

- Student View

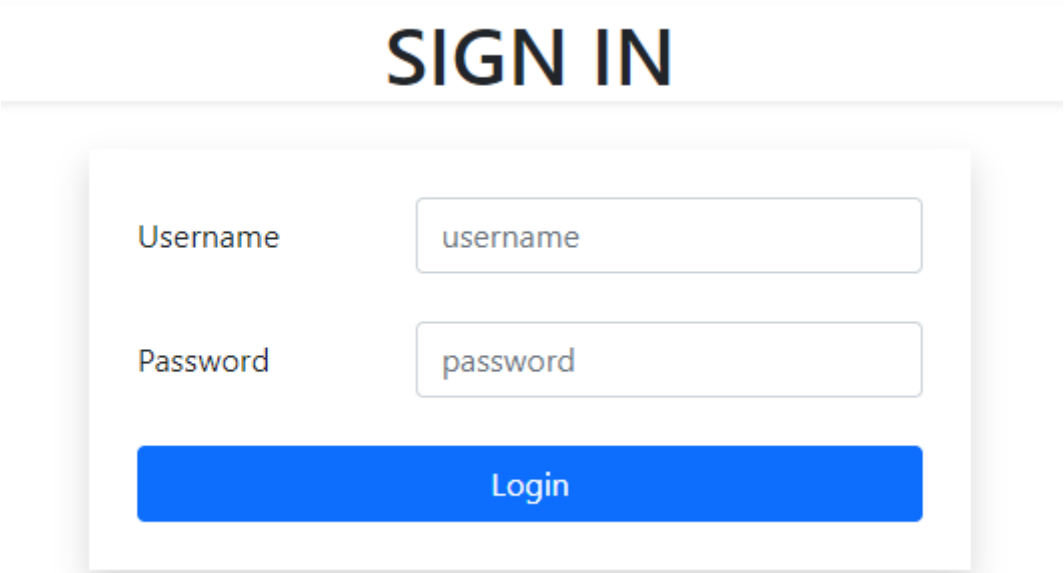


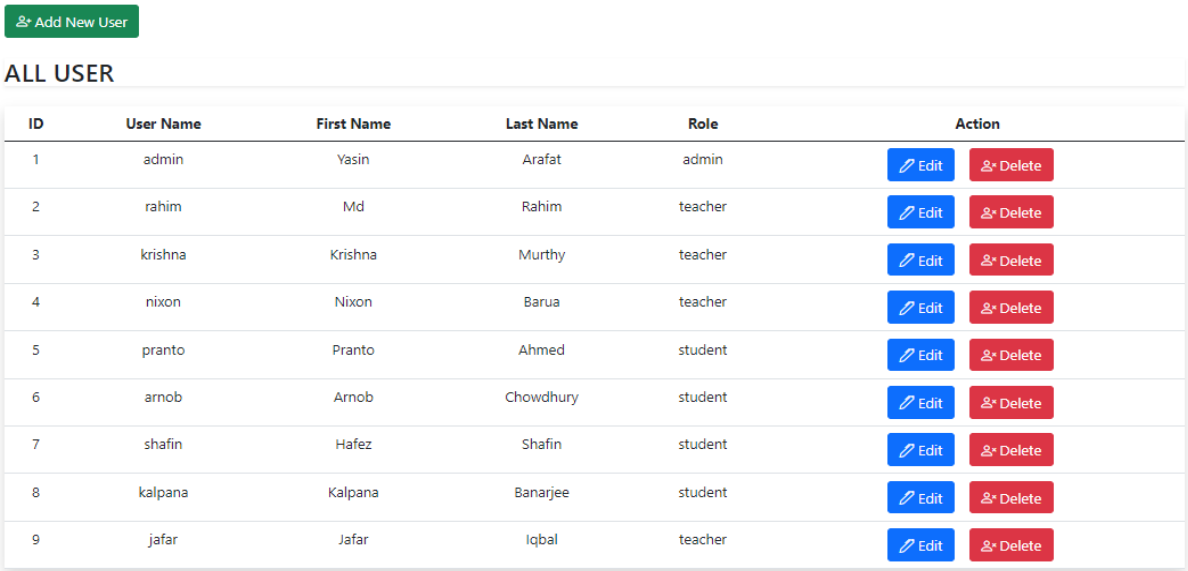
Figure-3: User login

2.4.2 Admin view

The login data of an admin will lead to the admin view. Here admins can see the list of all available user. They can also see all the classes which includes the subjects of their respective classes.

Login:

- Admins have their predefined username and password to initiate the process.

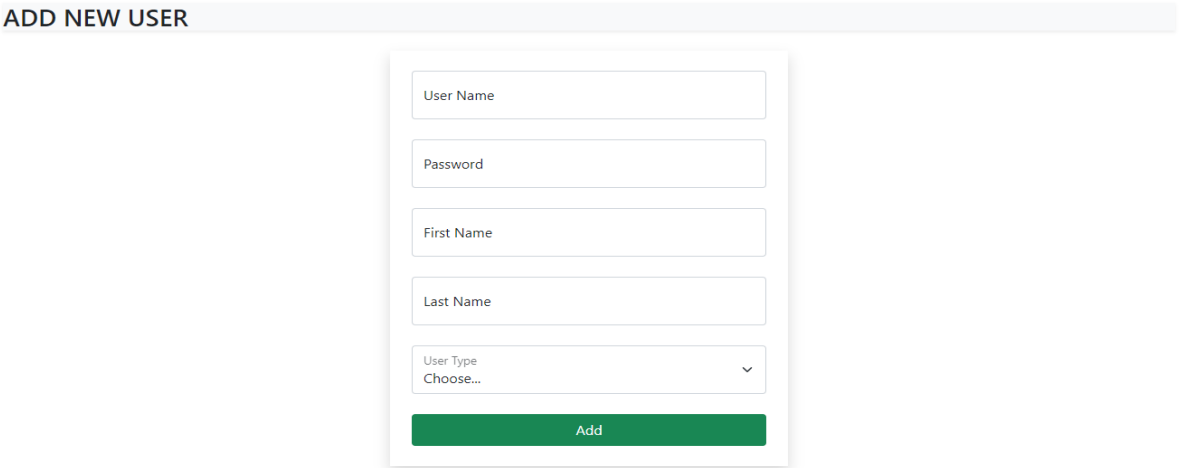


ID	User Name	First Name	Last Name	Role	Action
1	admin	Yasin	Arafat	admin	Edit Delete
2	rahim	Md	Rahim	teacher	Edit Delete
3	krishna	Krishna	Murthy	teacher	Edit Delete
4	nixon	Nixon	Barua	teacher	Edit Delete
5	pranto	Pranto	Ahmed	student	Edit Delete
6	arnob	Arnob	Chowdhury	student	Edit Delete
7	shafin	Hafez	Shafin	student	Edit Delete
8	kalpana	Kalpana	Banarjee	student	Edit Delete
9	jafar	Jafar	Iqbal	teacher	Edit Delete

- Figure-4: Admin view: Users

Add user:

- Admin can add new user to the system.
- Users are identified by a unique system provided user Id.
- User can be of three types: Admin, teacher and student.
- Admin provides all the information while creating a user with their respective types.
- Initially admin creates a password for the user which the user can change later. No plain text password is stored in the database.



- Figure-5: Add user

Modifying user attributes:

- Admin can change some basic attributes of the user which might include first name, last name and so forth.
- They cannot change the role or password of an already existing user.
- All the modifications are synchronized properly with the other views.

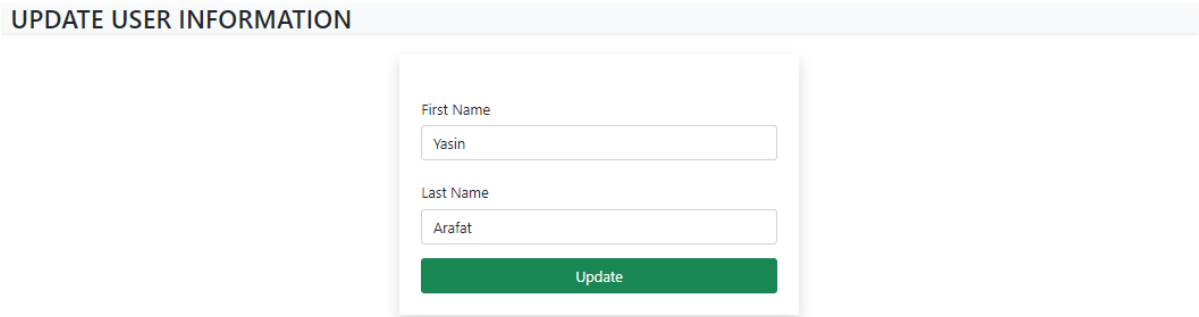


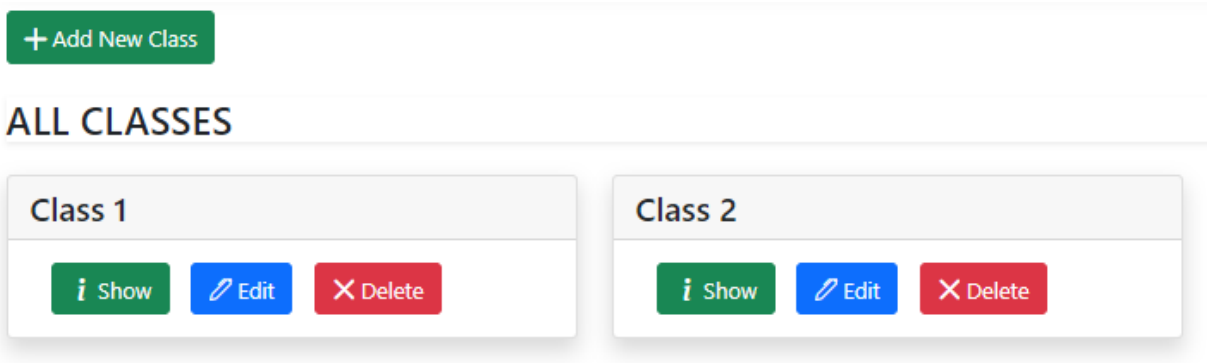
Figure-6: Update user

Delete user:

- Admin can delete an existing user anytime.
- System can restrict deletion in special cases.
- Teacher cannot be deleted if he/she is assigned to atleast one non-archived subject.
- While deleting a student, all the related information is also deleted from the system.

Add class:

- Admin can create new classes.
- Each class has a unique name and Id.



- Figure-7: Admin view: Classes

Modifying class attributes:

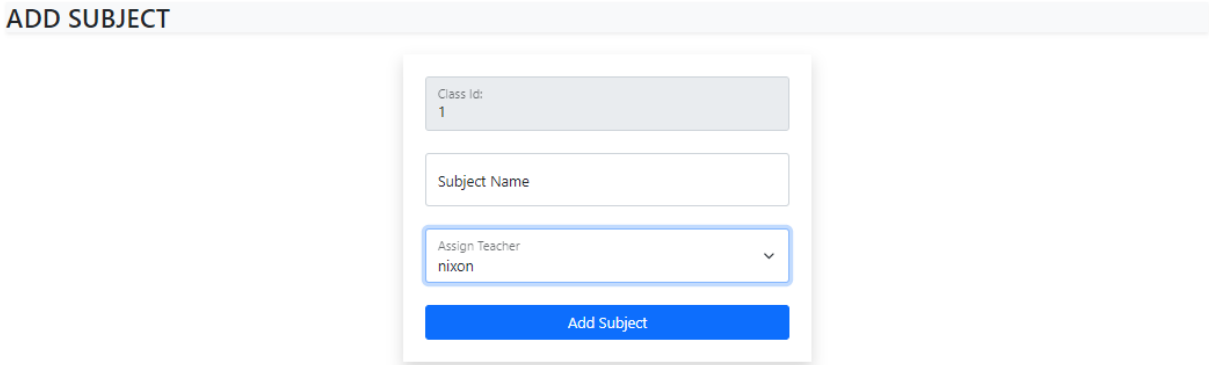
- Admin can change the name of a class.
- System always checks for the uniqueness of the name.

Delete class:

- Admin can delete a class anytime with necessary conditions.
- Deleting a class de-assigns all the included students.
- Related subjects either gets deleted or archived according to the requirements.

Add Subject:

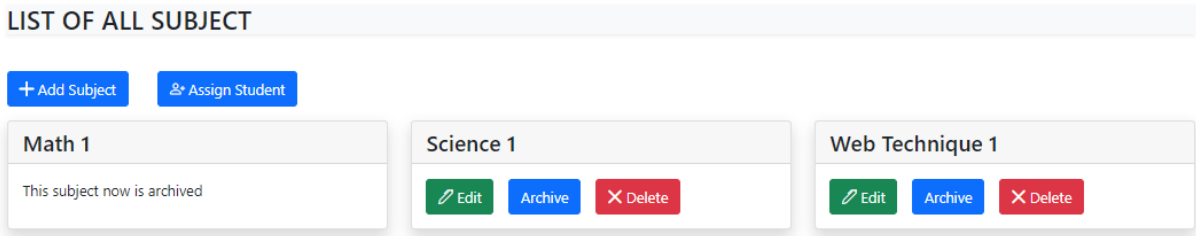
- Admin creates different subjects inside of a class.
- Each subject has a class wide unique name and Id.
- An already existing user is also assigned while creating the subject.



- Figure-8: Add Subject

Modifying subject attributes:

- Admin has full control over subject attributes.
- They can change the name of the subject which is automatically checked for unique name.
- Admin can assign new teacher to any class which leads to removal of the existing teacher from the subject.
- They cannot modify archived subjects.



- Figure-9: Subject view

Archive subject:

- Admin can archive an existing subject.
- Only subjects with completed test can be archived.
- Archiving of a subject cannot be altered and cannot be deleted as well.

Delete subject:

- Admin can entirely remove an existing subject.

- Subjects that does not contain any incomplete tests can only be deleted.

Assign teacher:

- Admin initially assigns an existing teacher to a new subject.
- They can also change the teacher later on.
- Only one teacher can be assigned to a single subject.

Assign/Deassign pupil:

- Students are normally assigned to a class which leads to assigning to the containing subject.
- An admin can also de-assign students from a class.
- Assigning student to a new class will automatically de-assign them from the previous one.

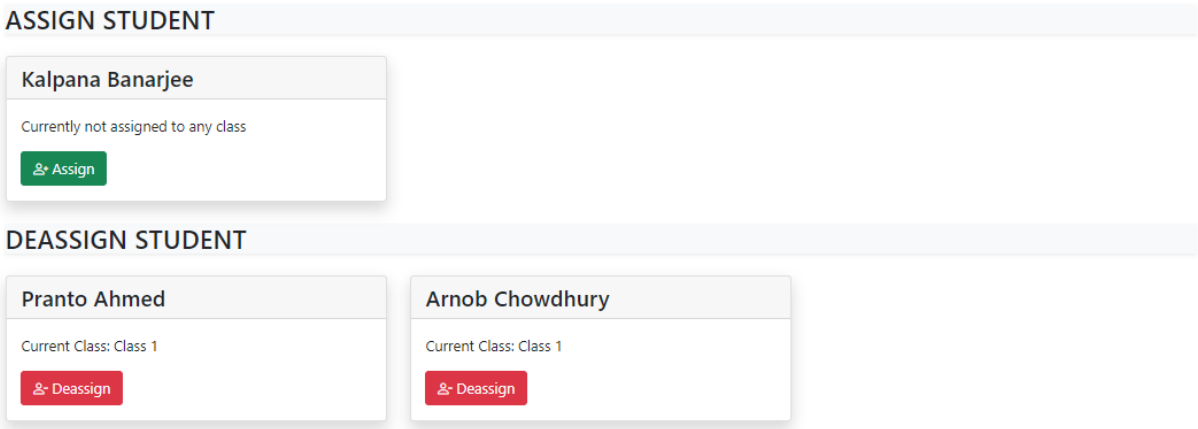


Figure-10: Assign/de-assign student

2.4.3 Teacher

Teacher view provides the list of all the subject that he/she is assigned to. They can also see all the students studying the subjects along with their respective average grades.

Login:

- Teachers have their predefined username and password to initiate the process.

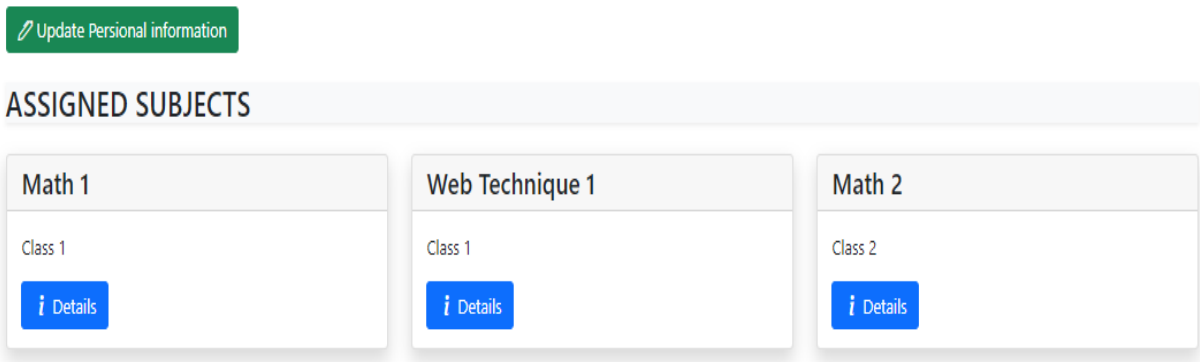


Figure-11: Teacher view

Change personal attributes:

- Teacher can update basic attributes like first name or last name.
- They can also change the initially given password by the admin.

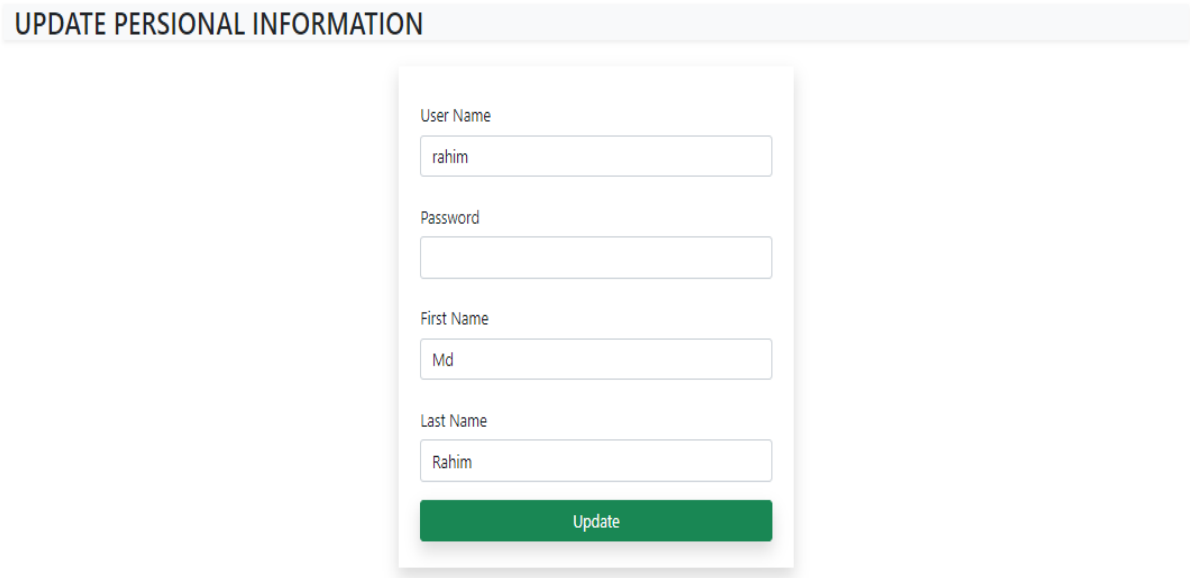


Figure-12: Update personal information

Add test:

- A teacher can add new test for individual subjects as many as they want.
- They need to provide a test name and an expected test date to each test.

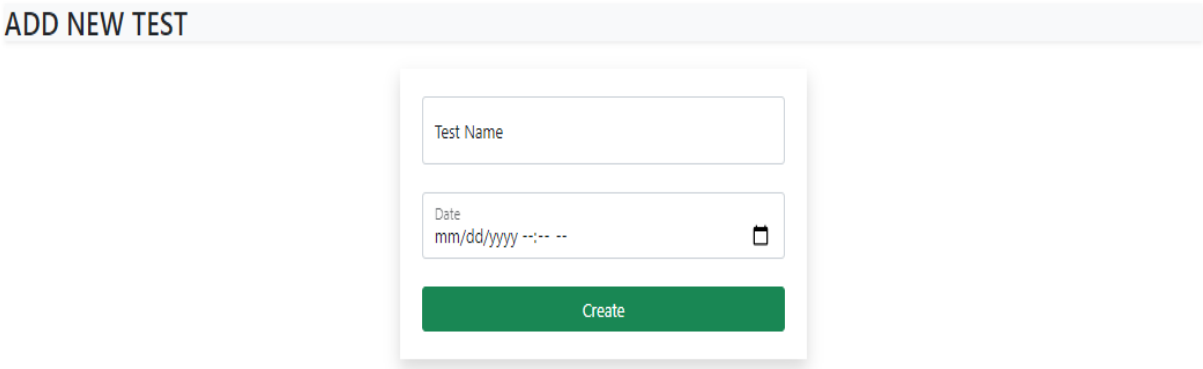


Figure-13: Add test

Add grades:

- Teacher can easily include grades of individual tests by uploading CSV files to the system.
- Test grades of all the students is then visible to the teacher view.

Modify grades:

- Teacher can also update already provided grades of individual students by editing the marks.
- The corresponding student view is automatically updated upon every change.

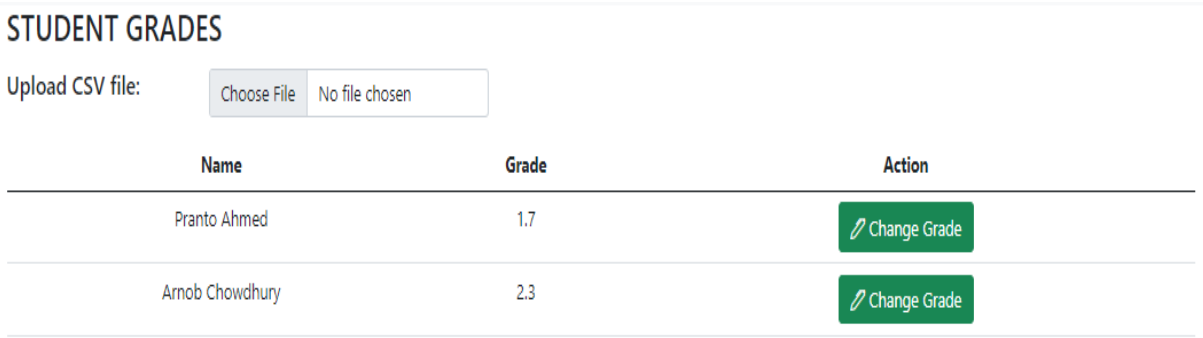


Figure-14: Add/view grades

Modifying test attributes:

- Teacher has full control over test attributes.
- They can change the name of the test.
- They can also change the expected date and time of the test.

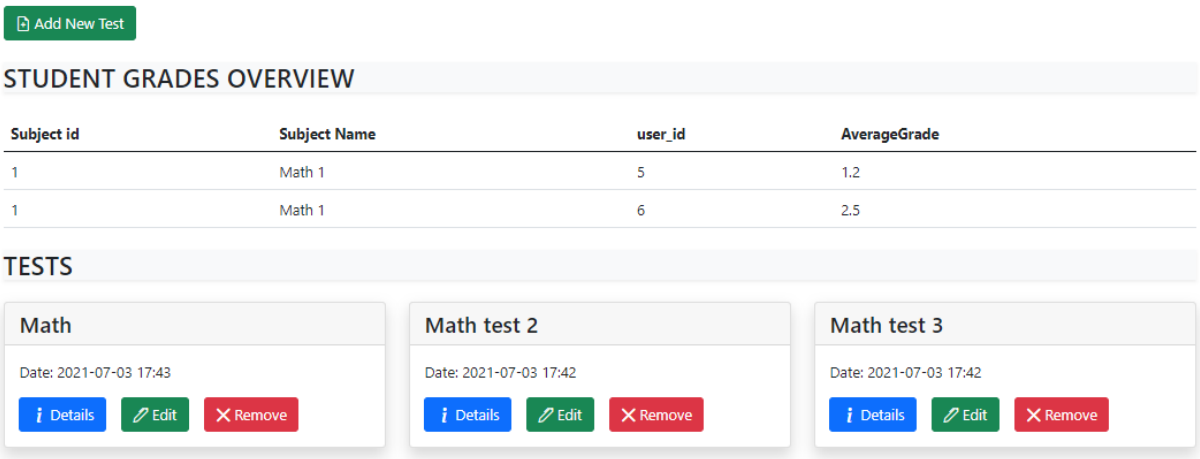


Figure-15: Test and grade overview

Delete test:

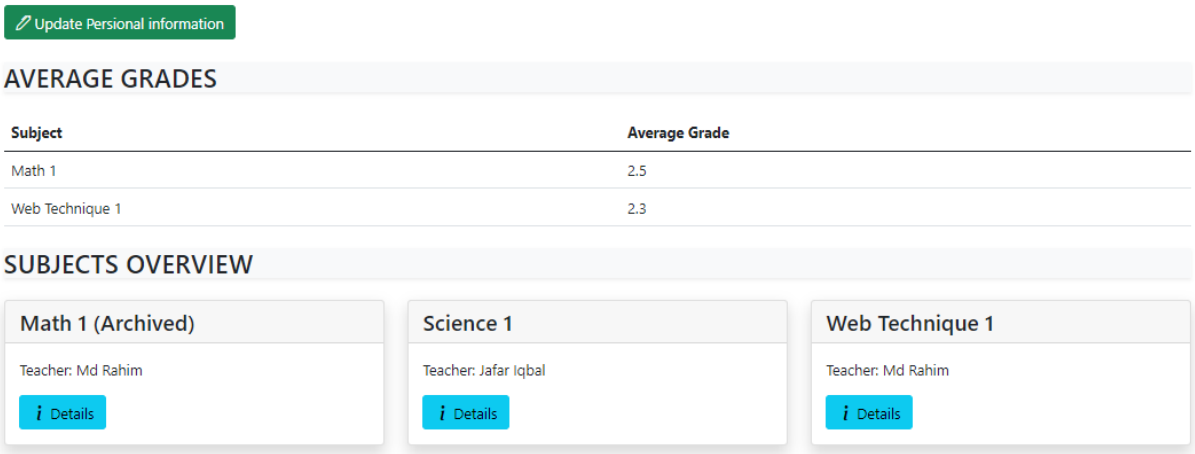
- Teacher view also provides functionality to delete current tests.
- Deleting a test removes all the related information like provided marks.

2.4.4 Student

Student view is the user view of the system. Here students can login to access the system and see meetings. They can easily join existing study groups under individual meeting or create a new one. Students also have full access to their created study groups where they can edit some attributes. Below we will see some more functionalities in details.

Login:

- Students have their predefined username and password to initiate the process.



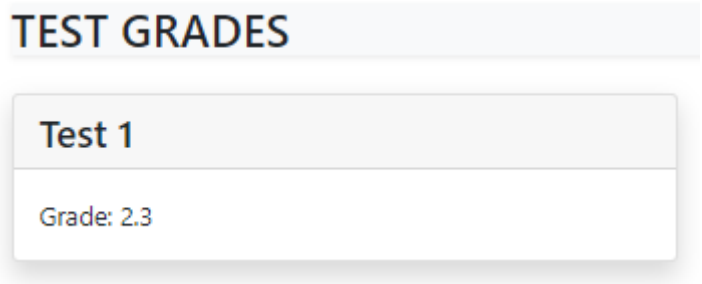


Figure-16: Student view

Change personal attributes:

- Students can update basic attributes like first name or last name.
- They can also change the initially given password by the admin.

3 Technology Overview

3.1 PHP

PHP is a general-purpose scripting language especially suited to web development. PHP code is usually processed on a web server by a PHP interpreter implemented as a module, a daemon or as a Common Gateway Interface (CGI) executable. On a web server, the result of the interpreted and executed PHP code – which may be any type of data, such as generated HTML or binary image data – would form the whole or part of an HTTP response. [4]

The main interface of this project is entirely created using PHP. All the database logic are implemented using PL/pgSQL. The executable data were shown in the application using PHP. Working with PHP is really easy and the development environment is also very user friendly. So we used PHP as our main language for user interface interaction.

3.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

3.2.1 Bootstrap

Bootstrap is a free front-end - CSS framework. It contains HTML and CSS- based design templates for typography, forms, buttons, tables, grid systems, navigation and other interface design elements as well as additional, optional JavaScript extensions[2].

Initially we did not use this language because it is mainly used to stylize the project. We almost entirely coded our project interface on PHP. After implementing all the logics and running a number of tests to verify all the required objectives and tasks, we finally used CSS to give the project a visuall aesthetic look.

3.3 Javascript

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. [3]

We used javascript for some important functionalities like exception handling warnings, page transformation functions and debugging.

3.3.1 Node.js

Node.js is a cross-platform, open-source back-end JavaScript runtime environment that uses the V8 engine to execute JavaScript code outside of a web browser[1]. Node.js allows developers to utilize JavaScript to create command-line tools and server-side scripting, which involves running scripts on the server before sending the page to the user's browser. As a result, Node.js is a good choice.

bcrypt : we use this module to hash our users passwords. With help of this module user can login securely into their corresponding pages.

cors: CORS is a node.js package that provides a Connect/Express middleware for enabling CORS with a variety of options. we use this to access from another server to our backend server.

express: Express is a Node.js backend web application framework. with this, I was implementing our backend and API.

fast-CSV: In the project, the teacher can upload a CSV file. That works done by fast-CSV.

3.3.2 Ajax

Ajax is a collection of web development approaches that uses a variety of client-side web technologies to construct asynchronous online applications. Ajax allows web applications to transmit and receive data from a server asynchronously without interfering with the existing page's presentation and behavior.

With help of this we send to API with method like put, delete request, and get response from API. That way, we handled our error response like 404, 403 etc.

4 Conclusion

This project "Bright Star Grading System" is an API based project where most of the functional logic was implemented in the backend which was communicated by an API with the database. Working on this project was entirely a new experience for me. Implementing logics and fetching them using API to cast on the interface without writing any logic in the frontend was a bit challenging at start. But day by day, I learned, and things got easier. This project was cordially created to help the cause to create a simple grading system for schools.

References

[1] N. (2021, July 1). July 2021 Security Releases. Node.Js. https://nodejs.org/en/blog/vulnerability/july-2021-security-releases/ "PHP: News Archive - 2021." PHP 8.0.6 Released, 6 May 2021, https://php.net/archive/2021.php#2021-05-06-2/

[2] "Bootstrap 5.0.1." Bootstrap Blog, 13 May 2021, https://blog.getbootstrap.com/2021/05/13/bootstrap-5-0-1/

[3] Wikipedia contributors. "JavaScript." Wikipedia, 19 Nov. 2001, https://wikipedia.org/wiki/JavaScript

[4] "PHP." Wikipedia, 27 May 2021, https://wikipedia.org/wiki/PHP

[5] Daniel Richter (2021. May 31) "DBW Project Task" [PDF File]. Retrieved from https://www.tu-chemnitz.de/informatik/DVS/lehre/AMD/Project/AMD%20-%20Project2.pdf

API Documentation

List of Endpoints:

Resources	Method	Endpoints
User	POST	/users/register
	POST	/users/login
	GET	/users/show/{Id}
	PUT	/users/{Id}
	DELETE	/users/{Id}
	GET	/users
	GET	/users/student/{Id}
	GET	/users/teacher
	GET	/users/student
	GET	/users/subjects/{Id}
	GET	/users/student/meanResult/{Id}
	GET	/users/teacher/subjects/{Id}
	GET	/users/subject/students/{Id}/{subject_id}
	GET	/users/student/grades/{Id}
Class	POST	/classes
	GET	/classes
	GET	/classes/{Id}
	PUT	/classes/{class_Id}/{user_Id}
	DELETE	/classes/{Id}
	GET	/classes/studentAssign/{Id}/{class_id}
	GET	/classes/studentDeassign/{Id/{class_id}
	GET	/classes/subjects/{class_id}
	GET	/classes/assign/{class_id}
	GET	/classes/deassign/{class_id}
Subject	GET	/subjects/{user_Id}
	GET	/subjects/show/{subject_id}
	GET	/subjects/details/{userId/{subjectId}
	DELETE	/subjects/{subject_id}
	POST	/subjects/teacherAssign/{subject_id}
	POST	/subjects/addEditSubject/{operation_type}
	POST	/subjects/archive/{subject_id}
	GET	/subjects/show/{Id/{subject_id}
	GET	/subjects/tests/{Id/{subject_id}
Test	POST	/tests/{subject_id}
	GET	/tests/{test_id}
	PUT	/tests/{Id}
	DELETE	/tests/{test_id}
	GET	/tests/details/{Id}/{test_id}
	POST	/tests/importGrades/{Id}
	GET	/tests/student_grades/{Id}/{test_id}
	PUT	/tests/grades/{Id}

Details of Endpoints

POST	/users/register
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	

Sample Body	{
	"user_name": "rahim",
	"password": "123",
	"first_name ": "Md",
	"last_name": "Rahim",
	"user_type": "student",
	"created_at ": "2021-07-03 17:26:38"
	}

Response	
Status	201 (Success), 409 (Conflict), 500 (Error)
Headers	location:
Sample Body	"Created successfully"

POST	/users/login
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	

Sample Body	{
	"username": "rahim",
	"password": "123"
	}

Response	
Status	202 (Success), 500 (Error)
Headers	location:
Sample Body	"Logged successfully"

GET	/users/show/{Id}
Request	
Headers	Content-Type: application/json

Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"user_id": 2,
	"user_name": "rahim",
	"password": "$2b$10$2lQbut7sT9xAJLpM2U9.b.iTSxCXDlIYJ90BPGsqbOXG530bQuJk6",
	"first_name": "Md",
	"last_name": "Rahim",
	"user_type": "student",
	"created_at": "2021-07-03T15:29:52.000Z"
	}

PUT	/users/{Id}
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	id: int (required)

Sample Body	{
	"first_name ": "Md",
	"last_name": "Rahim",
	"password": "123"
	}

Response	
Status	202 (Success), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Updated successfully"

DELETE	/users/{Id}
Request	
Headers	Content-Type: text/html
Params	id: int (required)
Sample Body	
Response	

Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Deleted successfully"

GET	/users
Request	
Headers	Content-Type: application/json
Params	
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"user_id": 1,
	"user_name": "admin",
	"password": "$2b$10$caHdrkuytvqnHfT/6hBoAunX5Vnc4WzRMiKFvU2Ju6fWwgT9gMioO",
	"first_name": "Admin",
	"last_name": "Admin",
	"user_type": "admin",
	"created_at": "2021-07-03T15:26:38.000Z"
	},
	{
	"user_id": 2,
	"user_name": "rahim",
	"password": "$2b$10$2lQbut7sT9xAJLpM2U9.b.iTSxCXDlIYJ90BPGsqbOXG530bQuJk6",
	"first_name": "Md",
	"last_name": "Rahim",
	"user_type": "student",
	"created_at": "2021-07-03T15:29:52.000Z"
	},
	{
	"user_id": 3,
	"user_name": "krishna",
	"password": "$2b$10$otDIlcDJ6ly.B1WABEhcO.hfMmupDTmC50LjnZ6VLewKIPJe87X9m",
	"first_name": "Krishna",
	"last_name": "Murthy",
	"user_type": "teacher",
	"created_at": "2021-07-03T15:30:19.000Z"
	}
	]

GET	/users/teacher
Request	

Headers	Content-Type: application/json
Params	
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"user_id": 3,
	"user_name": "krishna",
	"password": "$2b$10$otDIlcDJ6ly.B1WABEhcO.hfMmupDTmC50LjnZ6VLewKIPJe87X9m",
	"first_name": "Krishna",
	"last_name": "Murthy",
	"user_type": "teacher",
	"created_at": "2021-07-03T15:30:19.000Z"
	},
	{
	"user_id": 4,
	"user_name": "nixon",
	"password": "$2b$10$6/b24puRrIeSy3G11EjxEeCCgcjfqMrhUvMiW2K8IwHCCqLyAQtlK",
	"first_name": "Nixon",
	"last_name": "Barua",
	"user_type": "teacher",
	"created_at": "2021-07-03T15:30:57.000Z"
	}
	]

GET	/users/student
Request	
Headers	Content-Type: application/json
Params	
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:
Sample Body	[{ "user_id": 5, "user_name": "pranto", "password": "$2b$10$otDIlcDJ6ly.B1WABEhcO.hfMmupDTmC50LjnZ6VLewKIPJe87X9m", "first_name": "Pranto", "last_name": "Ahmed", "user_type": "student", "created_at": "2021-07-03T15:30:19.000Z"

	},
	{
	"user_id": 6,
	"user_name": "arnob",
	"password": "$2b$10$6/b24puRrIeSy3G11EjxEeCCgcjfqMrhUvMiW2K8IwHCCqLyAQtlK",
	"first_name": "Arnob",
	"last_name": "Chowdhury",
	"user_type": "student",
	"created_at": "2021-07-03T15:30:57.000Z"
	}
	]

GET	/users/subjects/{Id}
Request	
Headers	Content-Type: application/json
Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 1,
	"subject_name": "Math 1",
	"class_id": 1,
	"is_archived": 0,
	"class_name": "Class 1",
	"test_id": 1,
	"test_name": "Math",
	"is_complete": 1,
	"user_id": 6,
	"date": "2021-07-03T15:29:52.000Z"
	}

GET	/users/student/meanResult/{Id}
Request	
Headers	Content-Type: application/json
Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"AVG(m.marks)": 1.2
	}

GET	/users/teacher/subjects/{Id}
Request	
Headers	Content-Type: application/json
Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 2,
	"subject_name": "Math 2",
	"class_id": 2,
	"is_archived": 0,
	"class_name": "Class 2"
	}

GET	/users/subject/students/{Id}/{subject_id}
Request	
Headers	Content-Type: application/json
Params	id: int (required); subject_id: int (required);
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 2,
	"subject_name": "Math 2",
	"class_id": 2,
	"is_archived": 0,
	"class_name": "Class 2",
	"is_archived": 0,
	"user_id": 6,
	"AverageGrade": 2.5
	}

GET	/users/student/grades/{Id}
Request	
Headers	Content-Type: application/json
Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 2,
	"subject_name": "Math 2",
	"class_id": 2,
	"is_archived": 0,
	"class_name": "Class 2",
	"AverageGrade": 2.3
	}

POST	/classes
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	

Sample Body	{
	" Class 4": "Class 3"
	}

Response	
Status	202 (Success), 409 (Conflict), 500 (Error)
Headers	location:
Sample Body	"Logged successfully"

GET	/classes
Request	
Headers	Content-Type: application/json
Params	
Sample Body	

Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"class_id": 1,
	"class_name": "Class 1",
	"is_archived": 0
	},
	{
	"class_id": 2,
	"class_name": "Class 2",
	"is_archived": 1
	}
	]

GET	/classes/{id}
Request	
Headers	Content-Type: application/json
Params	id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"class_id": 1,
	"class_name": "Class 1",
	"is_archived": 0
	}
	]

PUT	/classes/{class_Id}/{user_Id}
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	class_Id: int (required); user_Id (required)

Sample Body	{
	" class_name ": "Class 6"
	}

Response	

Status	202 (Success), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Updated successfully"

DELETE	/classes/{Id}
Request	
Headers	Content-Type: text/html
Params	id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Deleted successfully"

GET	/classes/studentAssign/{Id}/{class_id}
Request	
Headers	Content-Type: application/json
Params	Id: int (required); class _id (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:
Sample Body	"Assigned successfully"

GET	/classes/studentDeassign/{Id/{class_id}
Request	
Headers	Content-Type: application/json
Params	Id: int (required); class _id (required)
Sample Body	

Response	
Status	200 (Success), 500 (Error)
Headers	location:
Sample Body	"Deassigned successfully"

GET	/classes/subjects/{class_id}
Request	
Headers	Content-Type: application/json
Params	class _id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"subject_id": 2,
	"subject_name": "Math 2",
	"class_id": 2,
	"class_name": "Class 2",
	"is_archived": 0
	},
	{
	"subject_id": 4,
	"subject_name": "English 2",
	"class_id": 2,
	"class_name": "Class 2",
	"is_archived": 0
	}
	]

GET	/classes/assign/{class_id}
Request	
Headers	Content-Type: application/json
Params	class _id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)

Headers	location:

Sample Body	[
	{
	"user_id": 5
	},
	{
	"user_id": 6
	}
	]

GET	/classes/deassign/{class_id}
Request	
Headers	Content-Type: application/json
Params	class _id: int (required)
Sample Body	
Response	
Status	200 (Success), 500 (Error)
Headers	location:

Sample Body	[
	{
	"user_id": 7
	},
	{
	"user_id": 8
	},
	{
	"user_id": 9
	}
	]

GET	**/subjects/{user_Id}**
Request	
Headers	Content-Type: application/json
Params	user_Id: int (required)
Sample Body	
Response	

Status	200 (Success), 401(Unauthorized), 403(Forbidden), 500 (Error)
Headers	location:

Sample Body	[
	{
	"subject_id": 1,
	"subject_name": "Math 1",
	"class_id": 1,
	"is_archived": 0,
	"class_name": "Class 1",
	"user_id": 2,
	"user_name": "rahim"
	},
	{
	"subject_id": 5,
	"subject_name": "Science 1",
	"class_id": 1,
	"is_archived": 0,
	"class_name": "Class 1",
	"user_id": 9,
	"user_name": "jafar"
	},
	{
	"subject_id": 6,
	"subject_name": "Sport 1",
	"class_id": 1,
	"is_archived": 0,
	"class_name": "Class 1",
	"user_id": 9,
	"user_name": "jafar"
	},
	{
	"subject_id": 2,
	"subject_name": "Math 2",
	"class_id": 2,
	"is_archived": 0,
	"class_name": "Class 2",
	"user_id": 2,
	"user_name": "rahim"
	},
	{
	"subject_id": 3,
	"subject_name": "Bengali 2",
	"class_id": 2,
	"is_archived": 0,
	"class_name": "Class 2",
	"user_id": 3,
	"user_name": "krishna"
	}
	]

GET	/subjects/show/{subject_id}
Request	
Headers	Content-Type: application/json
Params	subjects_Id: int (required)
Sample Body	
Response	
Status	200 (Success), 401(Unauthorized), 403(Forbidden), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 1,
	"subject_name": "Math 1",
	"class_id": 1,
	"is_archived": 0
	}

GET	/subjects/details/{userId/{subjectId}
Request	
Headers	Content-Type: application/json
Params	user_Id: int (required) subjects_Id: int (required)

Sample Body	{
	"user_id": 2,
	"user_name": "rahim",
	"password": "$2b$10$2lQbut7sT9xAJLpM2U9.b.iTSxCXDlIYJ90BPGsqbOXG530bQuJk6",
	"first_name": "Md",
	}

Response	
Status	200 (Ok),204(No Content),404(Error), 500 (Error)
Headers	location:

Sample Body	{
	"test_name": "Math",
	"marks": 1.2
	}

DELETE	/subjects/{subject_id}
Request	
Headers	Content-Type: text/html
Params	subject_id: int (required)

Sample Body	{
	"first_name ": "Md",
	"last_name": "Rahim",
	"password": "123"
	}

Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Deleted successfully"

POST	/subjects/addEditSubject/{operation_type}
Request	
Headers	application/x-www-form-urlencoded
Params	operation_type: string (required)

Sample Body	{
	"user_id":1
	"subject_name:Math" 4
	"class_id":1
	"subject_id":1
	}

Response	
Status	200 (Ok), 201 (Success), 404 (Error), 500 (Error)
Headers	location:

Sample Body	"Deleted successfully"

POST	/subjects/archive/{subject_id}
Request	
Headers	application/x-www-form-urlencoded
Params	subject_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Archived Successfully"

GET	/subjects/show/{Id}/{subject_id}
Request	
Headers	Content-Type: application/json
Params	Id: int (required) subjects_Id: int (required)
Sample Body	
Response	
Status	200 (Ok), 401 (Unauthorized), 404 (Error), 500 (Error)
Headers	location:

Sample Body	{
	"subject_id": 1,
	"subject_name": "Math 4",
	"is_archived": 1,
	"class_id": 1,
	"class_name": "Class 1",
	"user_id": 2,
	"user_name": "rahim"
	}

GET	/subjects/tests/{Id}/{subject_id}

Request	
Headers	Content-Type: application/json
Params	Id: int (required) subjects_Id: int (required)
Sample Body	
Response	
Status	200 (Ok), 204(No Content), 404 (Error), 500 (Error)
Headers	location:

Sample Body	[
	{
	"test_id": 1,
	"test_name": "Math",
	"subject_id": 1,
	"date": "2021-07-03T15:43:12.000Z",
	"is_complete": 1
	},
	{
	"test_id": 2,
	"test_name": "Math test 2",
	"subject_id": 1,
	"date": "2021-07-03T15:42:23.000Z",
	"is_complete": 0
	},
	{
	"test_id": 3,
	"test_name": "Math test 3",
	"subject_id": 1,
	"date": "2021-07-03T15:42:50.000Z",
	"is_complete": 0
	}
	]

POST	**/tests/{subject_id}**
Request	
Headers	application/x-www-form-urlencoded
Params	subject_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)

Headers	location:

Sample Body	[
	{
	"test_id": 2,
	"test_name": "Math test 2",
	"subject_id": 1,
	"date": "2021-07-03T15:42:23.000Z",
	"is_complete": 0
	}
	]

GET	/tests/{test_id}
Request	
Headers	Content-Type: application/json
Params	test_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:

Sample Body	[
	{
	"test_id": 1,
	"test_name": "Math",
	"subject_id": 1,
	"date": "2021-07-03T15:43:12.000Z",
	"is_complete": 1
	}
	]

PUT	/tests/{Id}
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	Id: int (required)

Sample Body	{
	"test_id": 2,
	"test_name": "Math test 33",
	"subject_id": 1,
	"date": "2021-07-06T15:42:23.000Z"
	}

Response	
Status	200 (Success), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Updated successfully"

DELETE	**/tests/{test_id}**
Request	
Headers	Content-Type: text/html
Params	test_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Deleted successfully"

GET	**/tests/details/{Id}/{test_id}**
Request	
Headers	Content-Type: application/json
Params	test_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:

Sample Body	[
	{
	"user_id": 5,
	"user_name": "pranto",
	"first_name": "Pranto",
	"last_name": "Ahmed",
	"test_id": 1,
	"test_name": "Math",
	"marks": 1.2
	},
	{
	"user_id": 6,
	"user_name": "arnob",
	"first_name": "Arnob",
	"last_name": "Chowdhury",
	"test_id": 1,
	"test_name": "Math",
	"marks": 2.5
	}
	]

POST	/tests/importGrades/{Id}
Request	
Headers	application/x-www-form-urlencoded
Params	Id: int (required)

Sample Body	{
	"test_id": 1,
	"file": "Mark.csv",
	}

Response	
Status	200 (Ok), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"CSV file uploaded successfully"

GET	/tests/student_grades/{Id}/{test_id}
Request	
Headers	Content-Type: application/json

Params	Id: int (required) test_id: int (required)
Sample Body	
Response	
Status	200 (Ok), 204(No Content), 404 (Error), 500 (Error)
Headers	location:

Sample Body	[
	{
	"mark_id": 2,
	"test_id": 1,
	"user_id": 6,
	"marks": 2.5
	}
	]

PUT	**/tests/grades/{Id}**
Request	
Headers	Content-Type: application/x-www-form-urlencoded
Params	Id: int (required)

Sample Body	{
	" marks ": "2.0",
	" test_id ": "1"
	}

Response	
Status	200 (Success), 404 (Error), 500 (Error)
Headers	location:
Sample Body	"Updated successfully"