SAI TIMETABLE GEN

A PROJECT AS A COURSE REQUIREMENT FOR
BACHELOR OF COMPUTER SCIENCE (HONS.)

Prepared by
VIDYASAGER GR
204221
III B.Sc CS (HONS.)

Certificate

This is to certify that this project titled 'SAI Timetable Gen' was submitted by VIDYASAGER GR - 204221, Department of Mathematics and Computer Science (DMACS), Muddenahalli Campus is a bonafide record of the original work done under my supervision as a Course Requirement for the Degree of Bachelor of Science in Computer Science (Hons.)

.....

Sri P Sunil Kumar,
Project Supervisor

Abstract

With the ever increasing usage of technology in almost every aspect of our life, we have been accustomed to the ease that is provided by it. For example, 30 years ago if I wanted to contact a friend who lived faraway the only way for me to converse with him would have been through post but now with a few taps on my phone I can have a video conference with that same friend no matter where in the world he is located.

This increased usage of technology to make a few tasks in our day-to-day life easier lead to the inception of this project when it was observed that the timetables for teachers and students were allocated by hand.

This Project will make the work of the administrator of the campus easier as it provides an user-friendly UI to manage details of Students, Teachers and Courses of the campus and also to allocate schedules for every semester.

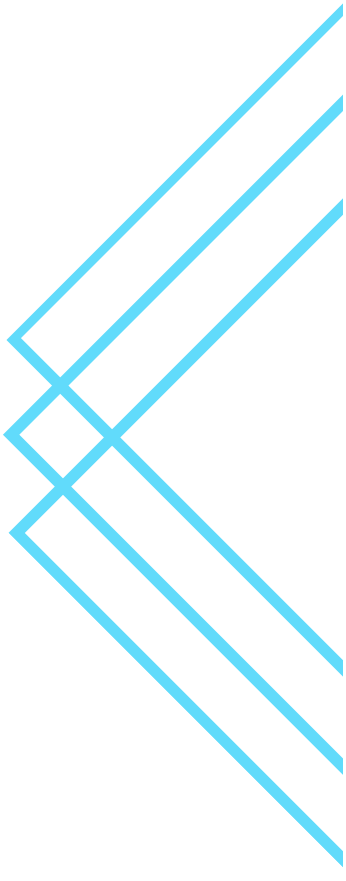


Table of Contents

1. Problem Statement

2. Requirements Analysis

3. Technologies Used

4. Design

5. SAI Timetable Gen

6. Screenshots

7. Future Scope

8. Acknowledgements

9. Sources

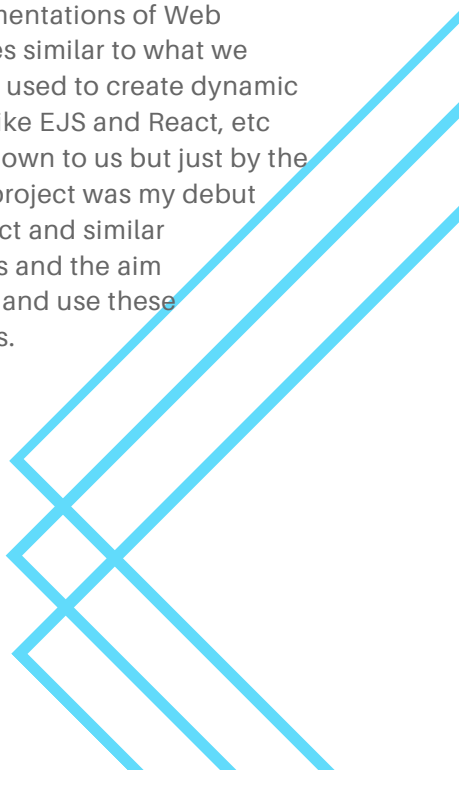
Problem Statement

Motivation

I had wanted to make an application which would serve as a hostel reception system using the technologies we learnt in the previous semester.
Thanks to my Project Mentor's presence of mind, this project came into existence. During discussions it was revealed that our college timetables were allocated manually by the administrator and human errors are very much possible in the process of making the schedules.
So it was decided to make an application which would serve an UI easy enough to work with and since machines are involved in creating and displaying the schedules, there was very little scope for error if not any.

Aim

Ever since we have been introduced to Web Development in the previous semester as a course, we have been excited to learn and implement a few applications or web pages using those technologies.
Though the course was meant as an introduction to basics in Web Development, our interests were piqued. On further research on our own we got to read about more and more implementations of Web Technologies similar to what we learnt. Tools used to create dynamic web pages like EJS and React, etc were also known to us but just by the name. This project was my debut in using React and similar technologies and the aim was to learn and use these technologies.



Requirements Analysis

This is the process used to determine the needs and expectations of a new product

The Required final product was a Web-Application built using MERN stack. The App in itself had to be very user-friendly in terms of the UI and in terms of how fast can the users learn to use the application. Further discussions made it clear that there should be an user-hierarchy in the application and that each type of user would have different permissions and access levels. The Hierarchy goes like this:

Admin

↓

Teacher

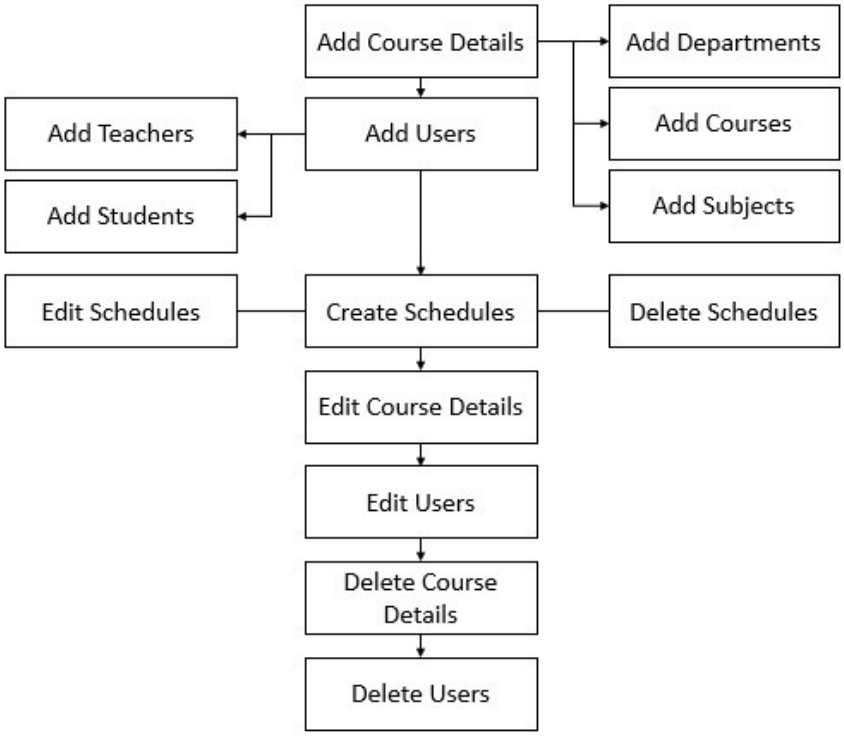
↓

Student

The Admin is at the top of the hierarchy and student at the bottom. This in-turn meant that the Admin will have the highest authority and will have all the permissions to Create, Read, Update and Delete items. The Teacher will have the permissions to Read and Request Admin for an Update (This applies for schedules). The Student will have permissions only to Read. Apart from this, the Teacher and the Student will have the permission to Update their passwords. The flow diagram showing all the actions performed by the users is attached in the following pages.

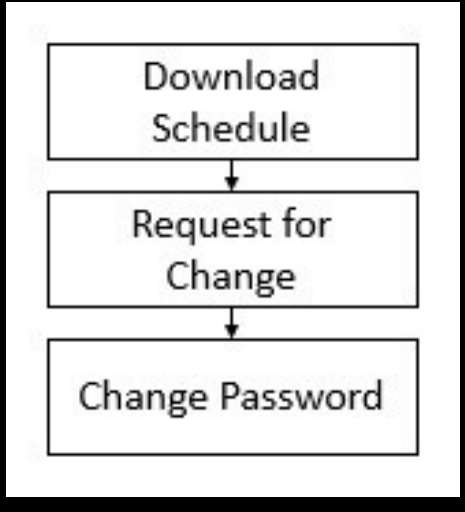
Cont.

Flow Diagram for Admin

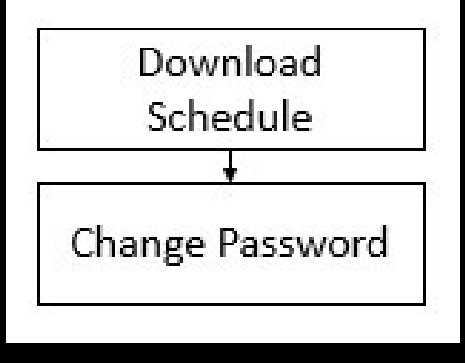


Cont.

Flow Diagram for Teacher



Flow Diagram for Student



What is a Web Application?

There have been a few if not many references to Web Applications and Dynamic Web Pages in this report.

But what is a Web Application? And how different is it from the traditional applications we use in our phones, laptops and desktops?
To put it simply, a Web Application is a software program that is accessed through a web browser and operates over the internet. The difference is that unlike traditional applications, web applications do not require installation on a user's computer or mobile device. Instead, these apps are hosted on a web server and accessed via a web browser, allowing users to access them from anywhere provided they have an internet connection.

A Web Application is also a Web Page or Website but the distinction being that web applications provide some form of utility or functionality to the user. Otherwise a static or a dynamic web page which just displays content is known as Web Pages or Websites. A Web Application can range from simple apps like calculator to more complex ones like social media platforms and online banking systems.

SAI Timetable Gen is also a Web Application according to the definition of a Web Application. Some examples of web applications include Gmail,Amazon,etc.

A Brief Depiction of the functioning of a Web Application

User <-> Web Browser <-> Web Server <-> Application Server <-> Database

- The user accesses the web application through a web browser.
- The web browser sends a request to the web server, which hosts the web application. The request contains information about what the user wants to do, such as submitting a form or viewing a page.

Cont.

- The web server processes the request and sends it to the application server, which hosts the back-end code of the web application.
- The application server retrieves data from the database, processes the request, and generates a response.
- The application server sends the response back to the web server, which in turn sends it back to the user's web browser.
- The user's web browser displays the response, which can include HTML, CSS, JavaScript, and other web technologies, as a web page.

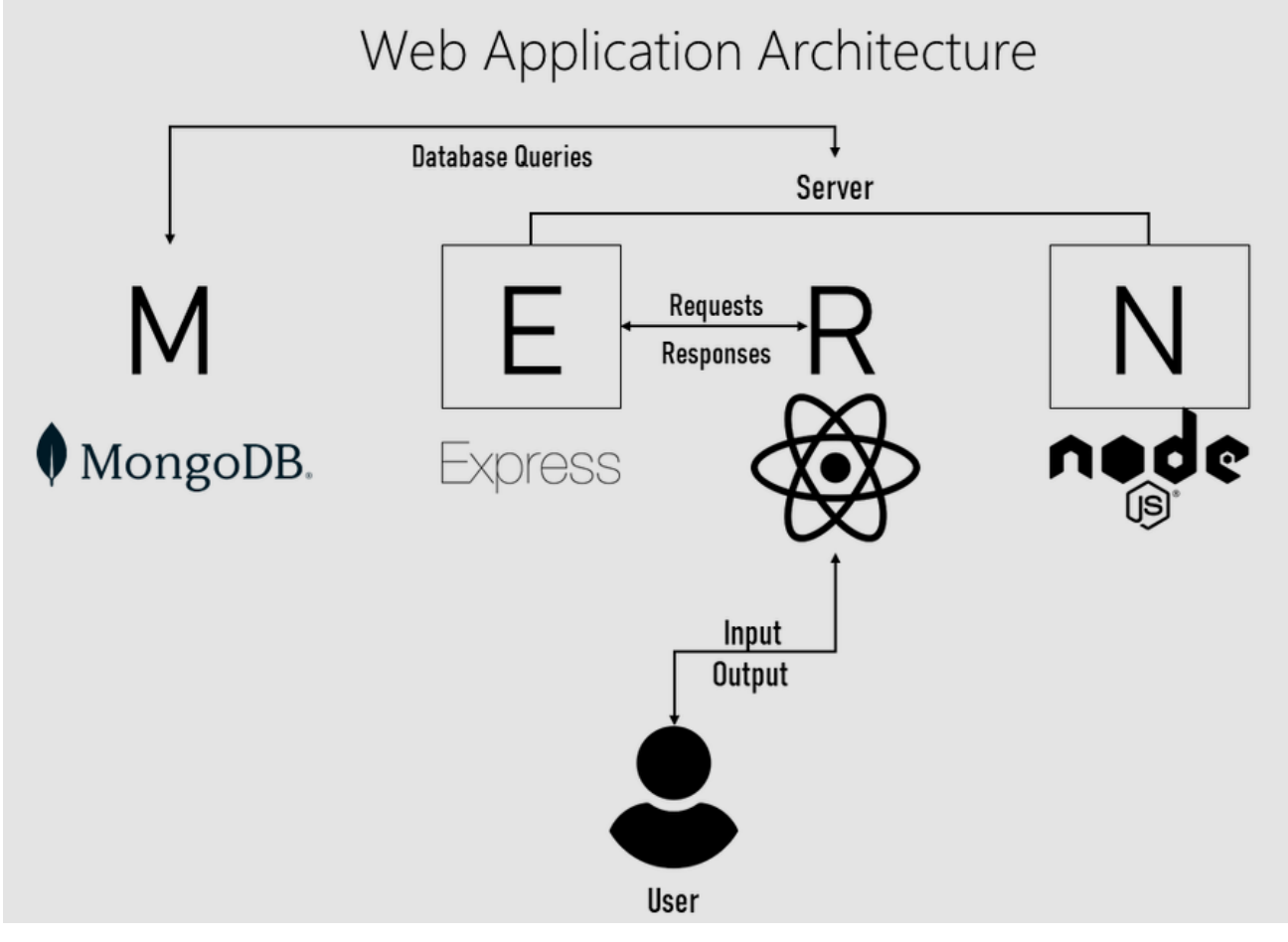
Are Web Applications better?

Web Applications have a lot of advantages and are a better choice than the traditional applications.

- *Cross-platform compatibility: They can be accessed from any device with a web browser and an internet connection, regardless of the operating system or hardware specifications.*
- *No installation required: Web applications do not require installation on a user's computer. This makes them easier to use and maintain, as users do not need to worry about updates or compatibility issues.*
- *Easy to update: Web applications can be updated and maintained centrally on the server, which means that updates can be rolled out to all users at once. This can save time and reduce the risk of errors or inconsistencies across different versions.*
- *Scalability: Web applications can be scaled up or down easily to accommodate changes.*

Technologies used

I had already decided that I would learn Full-Stack Web development as a skill or as a professional competency. This project gave way for me to learn a certain technique in Full-Stack development known as the MERN stack. The MERN stack is a Javascript stack that is used for easier and faster deployment of full-stack web applications. MERN Stack comprises of 4 technologies namely: MongoDB, Express, React and Node.js



HTML, CSS

As already mentioned SAI Timetable Gen is a Web Application. So HTML and CSS are like the building blocks that make up the look and feel of the application.

Now, HTML (Hypertext Markup Language) is the standard markup language used to create web pages and CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of HTML or XML document. Together they form the foundation for modern web development as HTML provides the structure and content of a web page, while CSS provides the style and visual design.

Predefined tags such as <h1>, <p>, etc are used to define the structure of the web pages. It also supports inclusion of media content like images, videos and audio files as well as links. CSS provides developers with range of styling options.
Example: Font SIze, Layout, Colour, etc.
Animations, interactive effects such as hover states, transitions and transformations can also be created.

These technologies are constantly evolving, with new features and standards as well as the support for using frameworks and libraries that simplify the process of building complex web applications. We will be looking about these in the upcoming pages.

Bootstrap

HTML and CSS are basically used to make up the front-end of the Web Application along with a few more tools like Bootstrap for instance.
Bootstrap is a popular open-source front-end framework that was developed by Twitter. It provides developers with a range of pre-built UI components, such as buttons, forms, typography and navigation, as well as CSS and Javascript tools to build responsive web projects.

Almost all the components of SAI Timetable Gen is built using Bootstrap. Bootstrap is used in the project by having a CDN for Bootstrap.min.css and Bootstrap.bundle.min.js in the index.html file.

```
<link
  href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/css/bootstrap.min.css"
  rel="stylesheet"
  integrity="sha384-GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Z13/Jr59b6EGGoI1aFkw7cmDA6j6gD"
  crossorigin="anonymous"
/>
```

```
<script
  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"
  crossorigin="anonymous"
></script>
```

CDN stands for Content Delivery Network. A CDN is a network of servers that are distributed around the world, and which work together to deliver content, such as web pages, images, and videos, to users as quickly and efficiently as possible.

These link and script tags download the Bootstrap.min.css file and Bootstrap.bundle.min.js on every device that uses our Web Application. This way the classes mentioned for all the elements and components that are from Bootstrap.css can be recognized and styled accordingly. For example, the forms, tables, modals, and responsive elements including the Navigation Bar in SAI Timetable Gen is styled using Bootstrap.
Apart from the Bootstrap defined classes used, we have created our own styles for a few components in the application for customization.

Google Fonts

Google Fonts is a free and open-source font library that allows developers and designers to choose from a wide range of web fonts to use in their projects. It's developed by Google and is available to use for both personal and commercial projects. Using it is simple and straightforward and similar to how Bootstrap is used in projects. Developers can browse the library of fonts on the Google Fonts website, select the fonts they want to use, and then copy and paste the embed code into their HTML and CSS files. So another link tag is used to mention the CDN for Google Fonts.

```
<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link
  href="https://fonts.googleapis.com/css2?family=DM+Sans&display=swap"
  rel="stylesheet"
/>
```

In this project the one and only font that is being used is DM Sans which is of the Sans-Serif font family. One of the key benefits and the reason why Google Fonts was chosen to be used in the project is its speed and reliability. Since they are hosted on Google's CDNs, they load quickly and are optimized for performance.
Once the CDN is mentioned, the only task left is to change the font-family attributes of the body of the HTML using CSS selectors in our own index.css stylesheet.

```
body {
  margin: 0;
  padding: 0;
  background-color: #ededed !important;
  font-family: "DM Sans", sans-serif;
  overflow: hidden;
}
```

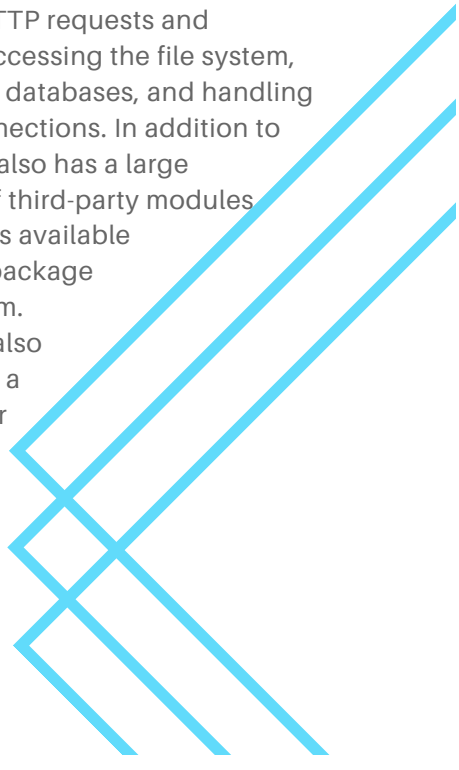
Though, the use of Google Fonts and Bootstrap to style the front-end of the application has proved to be extremely helpful as it reduces the load on the origin/app server and page load times, the loss of internet is a handicap on these technologies as they use internet to load the stylesheets by contacting the CDN servers. Lack of internet connection will inevitably break the front-end which is the UI. Thanks to modern web browsers for their caching systems, the styles and fonts loaded once will not be cleared unless the web browser's cache is cleared.

Javascript

Javascript is one of the three core technologies used in web development, alongside HTML and CSS

It is a popular high-level programming language that is primarily used to add interactivity and dynamic behaviour to web pages. It is client-side scripting language meaning that it is executed in the user's web browser, rather than on the server. This allows Javascript to interact with the web page and make changes to the DOM (Document Object Model) of the page which is a structured representation usually in the form of a tree.
Javascript is also widely used in frameworks and libraries such as Angular, Vue, and React. In our case, we are using React in our front-end.
Before we look into React and the workings and implementation of MERN stack let us look at the main building block of the project. As mentioned already, Javascript is a client-side scripting language and we also know that MERN stack is a Javascript stack which means the back-end is also written in Javascript. So to accomodate this, we have **Node.js**, which is an open-source, cross-platform, server-side runtime environment that allows developers to run JavaScript code outside of a web browser and to use it on the server-side to build scalable, high-performance web applications.

One of the key benefits of Node.js is its asynchronous, event-driven architecture. This means that it can handle a large number of requests concurrently, without blocking the execution of other tasks. Node.js is also known for building APIs. By using Node.js to build APIs, developers can create scalable and efficient interfaces that can handle large amounts of traffic and provide fast responses. The cherry on the top is that Node.js provides a set of built-in modules that enable developers to perform a wide range of tasks, such as managing HTTP requests and responses, accessing the file system, working with databases, and handling network connections. In addition to this, Node.js also has a large ecosystem of third-party modules and packages available through its package manager, npm.
This project also makes use of a good number of such npm modules.



Express

Express is a framework that runs on top of Node.js which makes it easier to build web applications.

Express provides a higher-level set of abstractions and features than what Node.js offers. For example, Express includes built-in support for handling HTTP requests and responses, routing, middleware, and templating engines, which can help reduce the amount of boilerplate code that developers need to write.

A get route written in plain Node.js

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  res.end('Hello World!');
}).listen(8080);
```

Same route written with Express

```
app.listen(8000, () => {
  console.log("Server is running on port 8000");
});

app.get("/", (req, res) => {
  res.send("hello world");
});
```

It is a lot simpler and a lot readable when compared to Node's route. Both of these API endpoints will return Hello World to the browser when port 8080 is accessed.

Express can be easily installed using NPM (Node Package Manager) and once installed, it can be easily used as shown below.

```
const express = require("express");
const app = express();
app.listen(8000, () => {
  console.log("Server is running on port 8000");
});
```

Cont.

And just like that our Express server is up and running. We can add as many API endpoints or routes as referred to as in the express documentation just by mentioning the method of request, url, and responses (if any).
The method of requests would be mostly either GET or POST. A form submission would be an example for a POST request to the server. Asking for a page, media or file, etc would be an example for GET requests. Express also supports a variety of data storage options, including relational databases such as MySQL and PostgreSQL, NoSQL databases such as MongoDB, and file-based storage such as CSV and JSON. This project uses MongoDB as its database as per the standards of MERN stack and both CSV and JSON as the intermediary storage.

MongoDB

A NoSQL Database that is document oriented, flexible, and scalable.

Unlike traditional relational databases, MongoDB stores data in flexible, JSON-like documents, which makes it a great choice for storing and managing complex, unstructured data. Though the database designed for the project is not unstructured and would have worked very well with relational databases, MongoDB was used adhering to MERN stack's standards and also due to pure interest in learning how to use MongoDB. In this venture, it was also learnt that Express and Node offer a native MongoDB driver to communicate with a MongoDB database and perform CRUD operations but a third party driver called Mongoose also works great with Express was discovered. The main use of Mongoose is to define a model or a schema for each collections in the database that enforces consistency and structure in data stored in the database. This ODM (Object Data Modeling) library is extremely powerful as it offers a wide range of features for interactions including CRUD operations.

Another key feature of Mongoose is its support for middleware functions, which can be used to intercept and modify data before it is saved to the database or retrieved from the database. This can be used to add custom validation logic, apply transformations to data, or perform other tasks. The schemas of the database declared using mongoose will be shown in the later parts of this report.

Setup is simple and similar to Express' setup.
NPM is again used to install the Mongoose module and then just two lines of code and connection to the MongoDB database is established.

```
const mongoose = require("mongoose");
mongoose.connect("mongodb://localhost:27017/projectDB");
```

React

React is a JavaScript library for building user interfaces. It is widely used in web development and is one of the key components of the MERN stack

React is a Front-End Library that acts as a server in itself that dynamically renders the Web Pages based on interactions from the user.
One of the key features of React is its use of a virtual DOM, which is a lightweight representation of the actual DOM. This allows React to quickly update the user interface in response to changes in application state, without needing to redraw the entire DOM. React is a very popular library and is widely used. Apps such as Facebook, Instagram, etc are built with React. The reason for its popularity is that it provides a component-based architecture, which allows developers to create reusable UI components that can be composed together to build complex user interfaces. This can greatly simplify the process of building and maintaining large applications, and makes it easy to reuse code across multiple parts of an application. In addition, React provides a wide range of features for handling user input, managing application state, and handling events. This includes support for hooks, which allow developers to easily add stateful logic to functional components, and a powerful set of lifecycle methods that can be used to control the behavior of components.
Thanks to Meta, the creators of React, the quick start sequence makes it easy to start creating a web application with React. All that needs to be done is to type the following line in the Terminal and press Enter.

```
$ npx create-react-app my-app
```

This will create a template app to start with which can be used as a starting point for your app. The template will have an Index.js file whose contents are as follows and the App component that is shown to be rendered will be at the top of the virtual DOM tree that react uses.

```
import React from "react";
import ReactDOM from "react-dom/client";
import App from "./components/App";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(<App />);
```

NPM Packages used

Frontend

- Uniqid
- React-Cookie
- React-to-print
- PapaParse
- EmailJS

Backend

- Body Parser
- Bcrypt
- Cors
- Mongoose
- Nodemon

uniqid

uniqid is a Unique Hexatridecimal ID Generator that is used to create unique IDs based on the current time, process and machine name. This is used to generate Unique cookie IDs for the users to maintain their session in the application. This generates 18 byte unique IDs and the IDs are always unique as:

- With the current time the IDs are always unique in a single process.
- With the Process ID the IDs are unique even if called at the same time from multiple processes.
- With the MAC Address the IDs are unique even if called at the same time from multiple machines and processes.

react-cookie

react-cookie provides the ability to create Universal Cookies meaning they will be stored in the Browser. The usage of this package is similar to the usage of all the React Hooks such as useState or useEffect. We use two functions 'useCookies' and 'setCookie'. The usage of these in our app is shown as follows:

```
const [cookie, setCookie] = useCookies(["userSaved", "username", "password"]);
```

```
setCookie("userSaved", true);
setCookie("username", user.username);
setCookie("cookieID", user.cookieID);
```

This will set the cookieID which is generated by uniqid along with the username so as to accomodate the maintaining of sessions for users.

react-to-print

react-to-print is an NPM package that allows users to print or download a specific React Component along with the CSS styles copied over. This is accomodated by popping up a print window. This module works for both class components and functional components. Since we are using functional components in our React App, we will use the 'useReactToPrint' hook along with the 'useRef' hook.

```
import { useReactToPrint } from "react-to-print";
```

The 'useRef' hook is used to reference a value that is not needed for rendering. By using this we will set the value to be referenced as the values of the component we need to print.

```
const componentRef = useRef();
const handlePrint = useReactToPrint({
  content: () => componentRef.current,
  documentTitle: props.User.username,
});
```

Then this handlePrint function is just called when a button is clicked to print the current component.

Papa Parse

Papa Parse is an in-browser CSV parser tool. It can be used to both convert or parse a CSV to a JSON or a JSON to CSV. It is a powerful tool that offers a lot of customization or personalization like choosing the delimiters, adding headings to the JSON. This package also allows user to parse files that are both local and the ones that are hosted somewhere else. Using the NPM package is that same as others. Install it using NPM and import it into your project.

```
import Papa from "papaparse";
```

To convert from CSV to JSON, we use the parse funcrtion and to convert from JSON to CSV we use the unparse function. This project uses the parse function for the 'Add Cohort' functionality to add Multiple Students, Teachers, or Course Details in one go by using CSV files.

Cont.

```
Papa.parse(event.target.files[0], {
  header: true,
  skipEmptyLines: true,
  complete: function (results) {
    cohort = results.data;
    resolve(cohort);
  },
});
```

EmailJS

EmailJS helps to send emails using client-side technologies only. No server is required. All it requires is to connect EmailJS to one of the supported email services, create an email template, and use their react SDK libraries to trigger an email.

Setup is again similar.

```
import emailjs from "emailjs-com";
```

EmailJs supports getting values from both Forms or single input fields. The methods to be used are sendForm and send respectively. One example from the app is for Request Change which just takes one TextBox as the input from the user.

```
var templateParams = {
  to_name: "Admin",
  to_mail: "vidyasager162@gmail.com",
  from_name: name,
  message: event.target.issue.value,
};
emailjs
  .send("gmail", "req_change", templateParams, "1fuN55z3EW1BC5gy0")
  .then(
    (response) => {
      console.log("SUCCESS!", response.status, response.text);
    },
    (error) => {
      console.log("FAILED...", error);
    }
  );
```

EmailJS is used widely to send mails from front-end libraries like React and Vue.js due to its templating capabilities. The sender, receiver, and the message sent is all made dynamic by sending them as parameters to the send functions including the service id (in our example, its gmail) and template id and public key for authentication.

body-parser

A body parsing middleware which parses incoming request bodies in express before the handler functions

As req.body's shape is based on user-controlled input, all properties and values in this object are untrusted and should be validated before trusting. body-parser accomodates this by having multiple parse factories such as json, url-encoded, etc to populate the req.body property with the parsed body when the Content-Type request header matches the type option.

```
const bodyParser = require("body-parser");
app.use(bodyParser.urlencoded({ extended: true }));
```

This is the setup used in our app. The urlencoded factory returns middleware that only parses urlencoded bodies and only looks at requests where the Content-Type header matches the type option. This parser accepts only UTF-8 encoding of the body. The extended option allows to choose between parsing the URL-encoded data with the querystring library (when false) or the qs library (when true). The "extended" syntax allows for rich objects and arrays to be encoded into the URL-encoded format, allowing for a JSON-like experience with URL-encoded.

bcrypt

bcrypt is a library for node.js that implements the password-hashing function 'bcrypt' designed by Niels Provos and David Mazieres based on the Blowfish Cipher in 1999. The main characteristic of bcrypt is its resistance to brute-force search attacks. There are a number of rounds in which the standard Blowfish keying algorithm is applied, using alternatively the salt(random generated) and the password as the key, each round starting with the subkey state from the previous round. In theory, this is no stronger than the standard Blowfish key schedule, but the number of rekeying rounds is configurable; this process can therefore be made arbitrarily slow, which helps deter brute-force attacks upon the hash or salt. This NPM package does the same using a simple 'hash' function to hash your password with the hash-type, salt and the actual password's hash. The resultant encrypted passwords are 60 characters long.

Cont.

```
const bcrypt = require("bcrypt");
const saltRounds = 10;
```

```
const hashedPassword = await bcrypt.hash(
  req.body.payload[i].password,
  saltRounds
);
```

Salt Rounds talks about the cost of hashing. The module will go through a series of rounds to give you a secure hash. The value you submit is not just the number of rounds the module will go through to hash your data. The module will use the value you enter and go through 2^rounds hashing iterations.

Example of an Hashed Password:

```
$2b$10$nOUIs5kJ7naTuTFkBy1veuK0kSxUFXfuaOKdOKf9xYT0KKIGSJwFa
 |  |  |                     |
 |  |  |                     hash-value = K0kSxUFXfuaOKdOKf9xYT0KKIGSJwFa
 |  |  |
 |  |  salt = nOUIs5kJ7naTuTFkBy1veu
 |  |
 |  cost-factor => 10 = 2^10 rounds
 |
 hash-algorithm identifier => 2b = BCrypt
```

- 2 chars hash algorithm identifier prefix. "$2a$" or "$2b$" indicates BCrypt
- Cost-factor (n). Represents the exponent used to determine how many iterations 2^n
- 16-byte (128-bit) salt, base64 encoded to 22 characters
- 24-byte (192-bit) hash, base64 encoded to 31 characters

Cors

Cross-origin resource sharing is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served

"cross-domain" requests, notably Ajax requests, are forbidden by default by the same-origin security policy. CORS defines a way in which a browser and server can interact to determine whether it is safe to allow the cross-origin request. It allows for more freedom and functionality than purely same-origin requests, but is more secure than simply allowing all cross-origin requests.

```
const cors = require("cors");
```

```
app.use(
  cors({
    origin: "*",
    credentials: true,
  })
);
```

Our implementation of cors is to let all the domains share resources to the server and also to share cookies by setting origin: "*" and credentials: true.

Mongoose

Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment.

Cont.

First, we need to define a connection. If the app uses only one database, then we should use mongoose.connect. If you need to create additional connections, use mongoose.createConnection. Both connect and createConnection take a mongodb:// URI, or the parameters host, database, port, options.

```
mongoose.set("strictQuery", false);
mongoose.connect("mongodb://localhost:27017/projectDB");
```

strictQuery parameter when set to true ensures that values passed to our model constructor that were not specified in our schema do not get saved to the db. This parameter is said to be false in our project and it is by default in newer Mongoose versions set to false. The Schemas defined will be talked about in the upcoming slides.

Nodemon

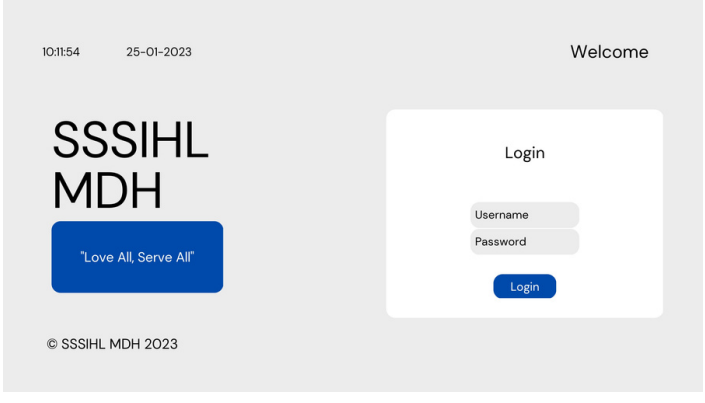
Nodemon is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected.

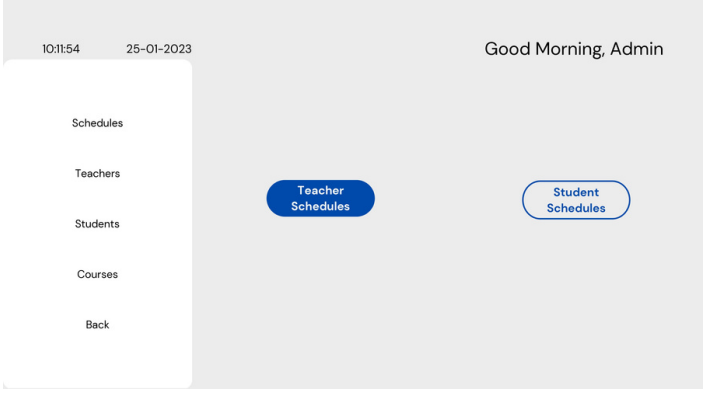
This does not require any additional changes to our code or method of development since it is a replacement wrapper for node. To use it we just replace the word node on the command line when executing our script.

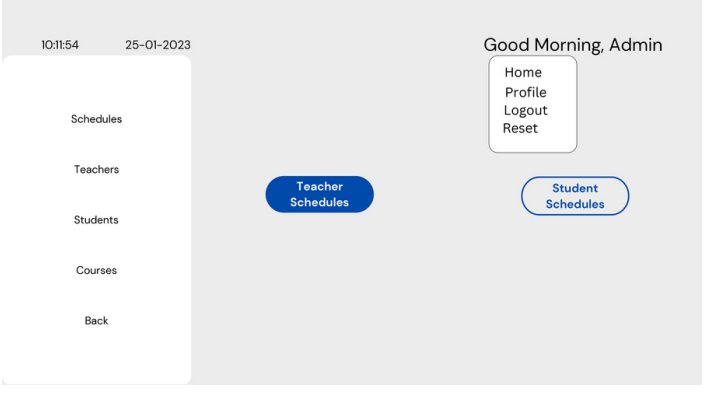
Design

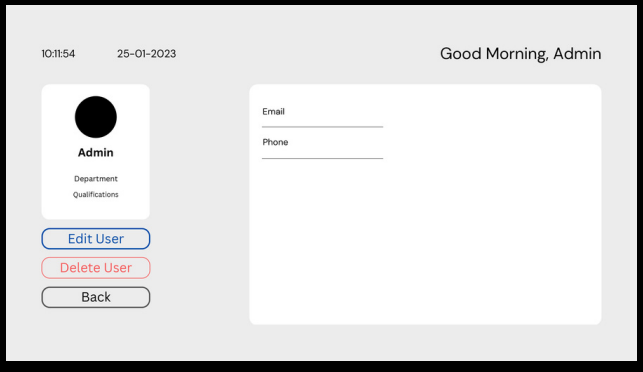
With the Problem Statement and Requirements Analysis ready, it was very easy to visualize and design the frontend of the app and also the database. The design slides are attached here

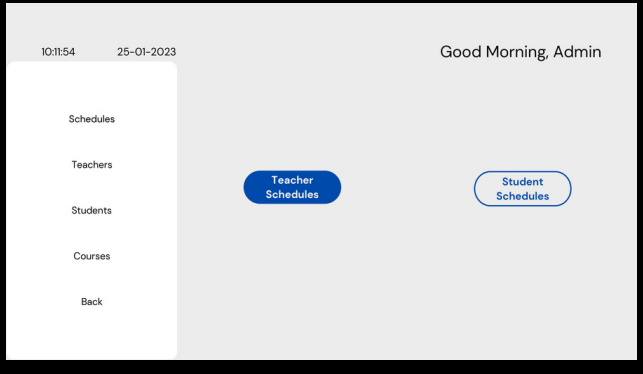
Admin

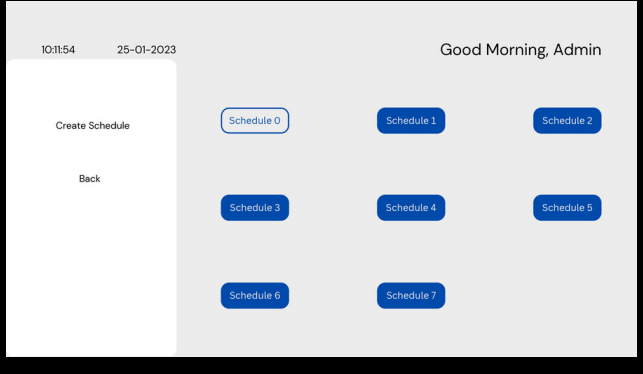


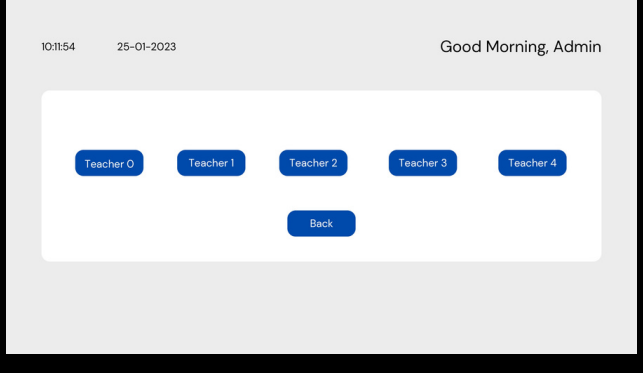


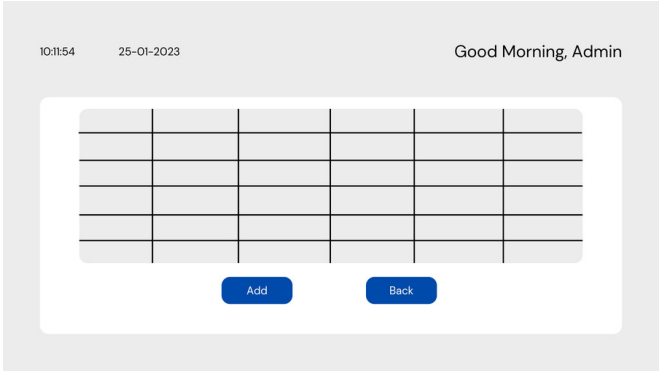


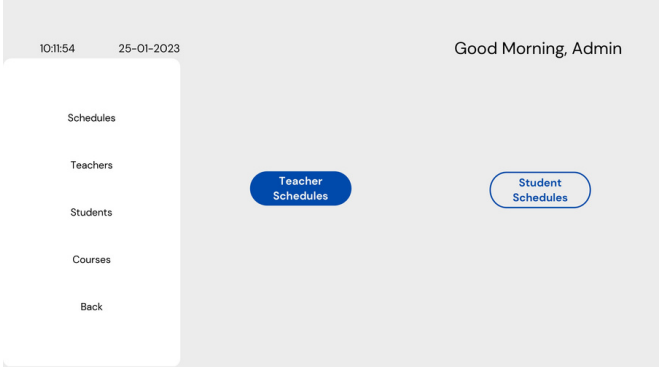

10:11:54
25-01-2023
Good Morning, Admin
Admin
Department
Qualifications
Edit User
Delete User
Back
Email
Phone

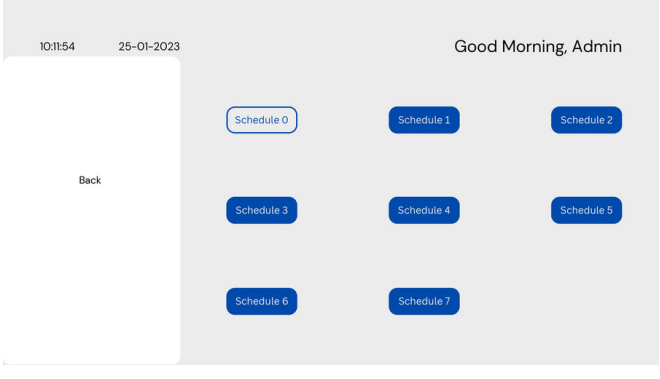

10:11:54
25-01-2023
Good Morning, Admin
Schedules
Teachers
Students
Courses
Back
Teacher Schedules
Student Schedules

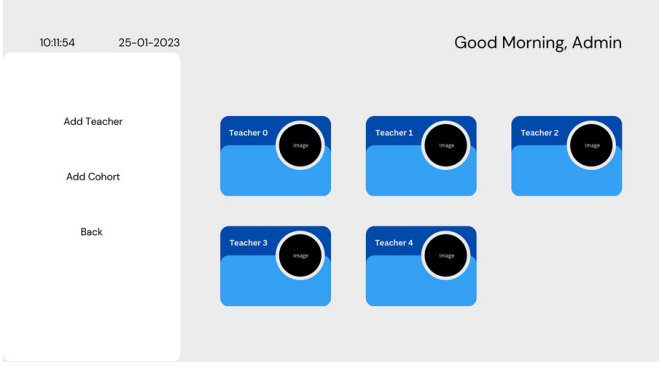

10:11:54
25-01-2023
Good Morning, Admin
Create Schedule
Back
Schedule 0
Schedule 1
Schedule 2
Schedule 3
Schedule 4
Schedule 5
Schedule 6
Schedule 7

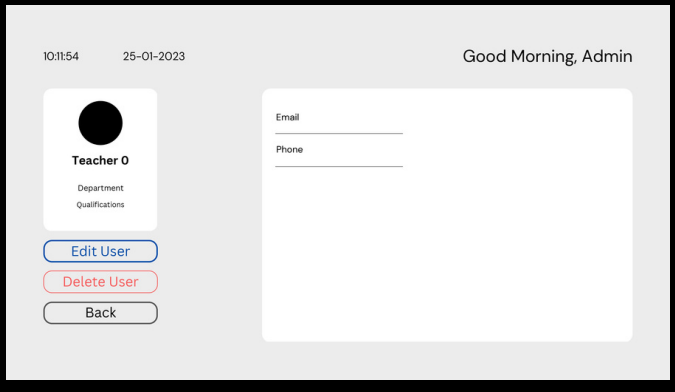

10:11:54
25-01-2023
Good Morning, Admin
Teacher 0
Teacher 1
Teacher 2
Teacher 3
Teacher 4
Back

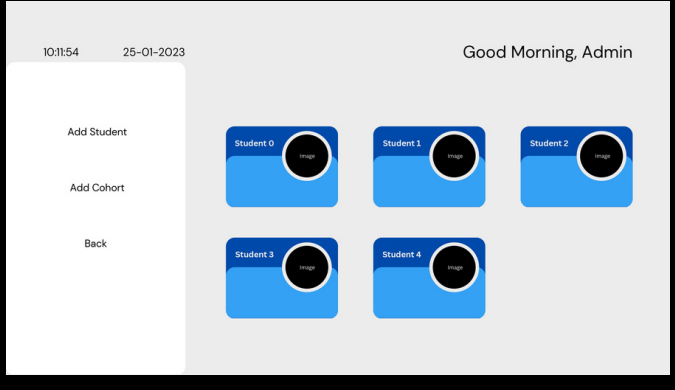

10:11:54
25-01-2023
Good Morning, Admin
Add
Back

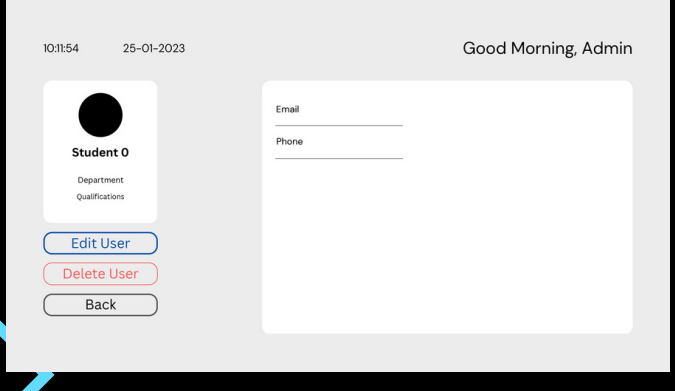

10:11:54
25-01-2023
Good Morning, Admin
Schedules
Teachers
Students
Courses
Back
Teacher Schedules
Student Schedules

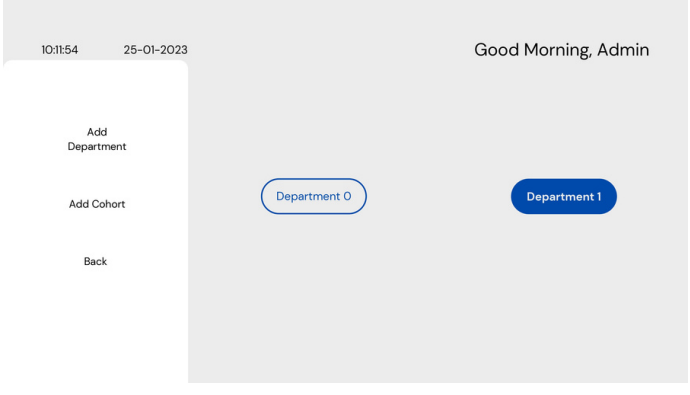

10:11:54
25-01-2023
Good Morning, Admin
Back
Schedule 0
Schedule 1
Schedule 2
Schedule 3
Schedule 4
Schedule 5
Schedule 6
Schedule 7


10:11:54
25-01-2023
Good Morning, Admin
Add Teacher
Add Cohort
Back
Teacher 0
Teacher 1
Teacher 2
Teacher 3
Teacher 4


10:11:54
25-01-2023
Good Morning, Admin
Teacher 0
Department
Qualifications
Edit User
Delete User
Back
Email
Phone


10:11:54
25-01-2023
Good Morning, Admin
Add Student
Add Cohort
Back
Student 0
Image
Student 1
Image
Student 2
Image
Student 3
Image
Student 4
Image

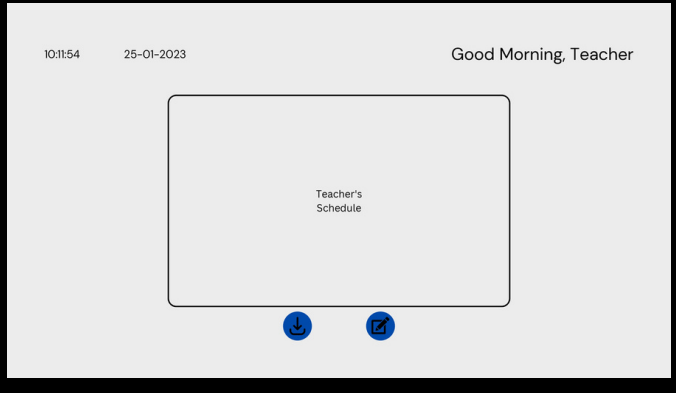

10:11:54
25-01-2023
Good Morning, Admin
Student 0
Department
Qualifications
Edit User
Delete User
Back
Email
Phone

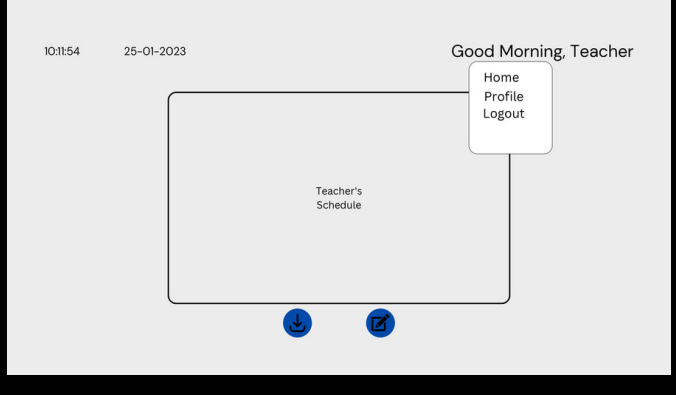

10:11:54
25-01-2023
Good Morning, Admin
Add Department
Add Cohort
Back
Department 0
Department 1

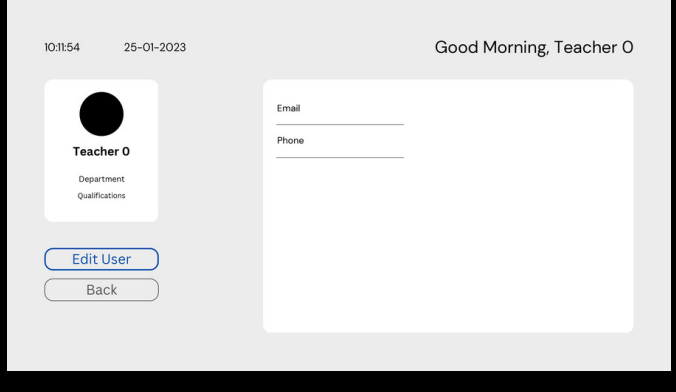

10:11:54
25-01-2023
Good Morning, Admin
Add Course
Add Cohort
Back
Course 0
Course 1
Course 2
Course 3
Course 4
Course 5
Course 6
Course 7


10:11:54
25-01-2023
Good Morning, Admin
Add Subject
Add Cohort
Back
Subject 0
Subject 1
Subject 2
Subject 3
Subject 4
Subject 5
Subject 6
Subject 7

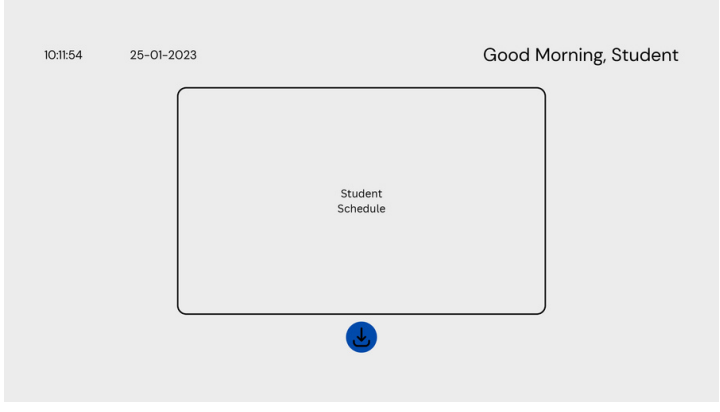
Teacher

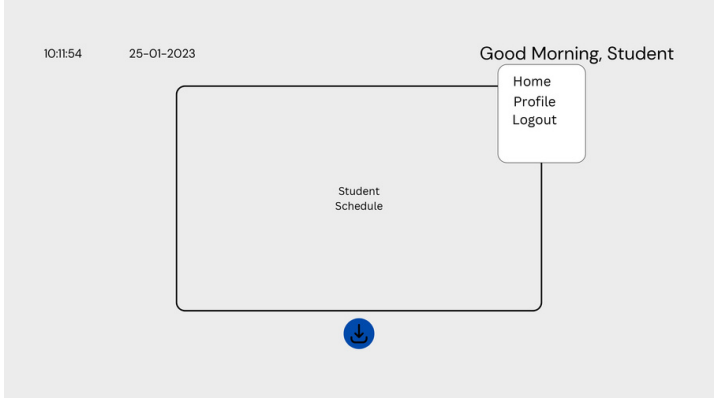

10:11:54
25-01-2023
Good Morning, Teacher
Teacher's
Schedule

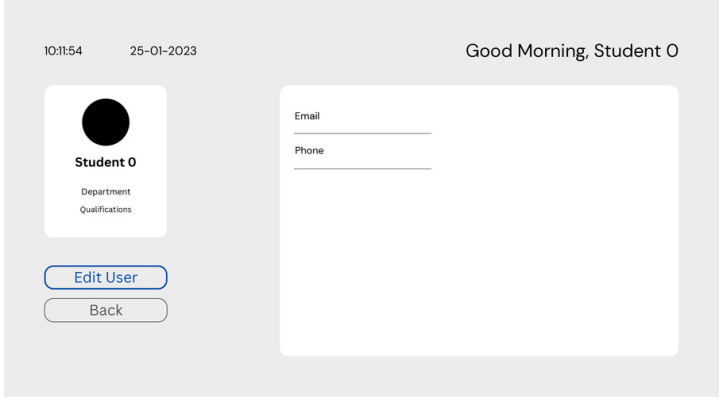

10:11:54
25-01-2023
Good Morning, Teacher
Home
Profile
Logout
Teacher's
Schedule


10:11:54
25-01-2023
Good Morning, Teacher 0
Teacher 0
Department
Qualifications
Edit User
Back
Email
Phone

Student


10:11:54
25-01-2023
Good Morning, Student
Student
Schedule


10:11:54
25-01-2023
Good Morning, Student
Home
Profile
Logout
Student
Schedule

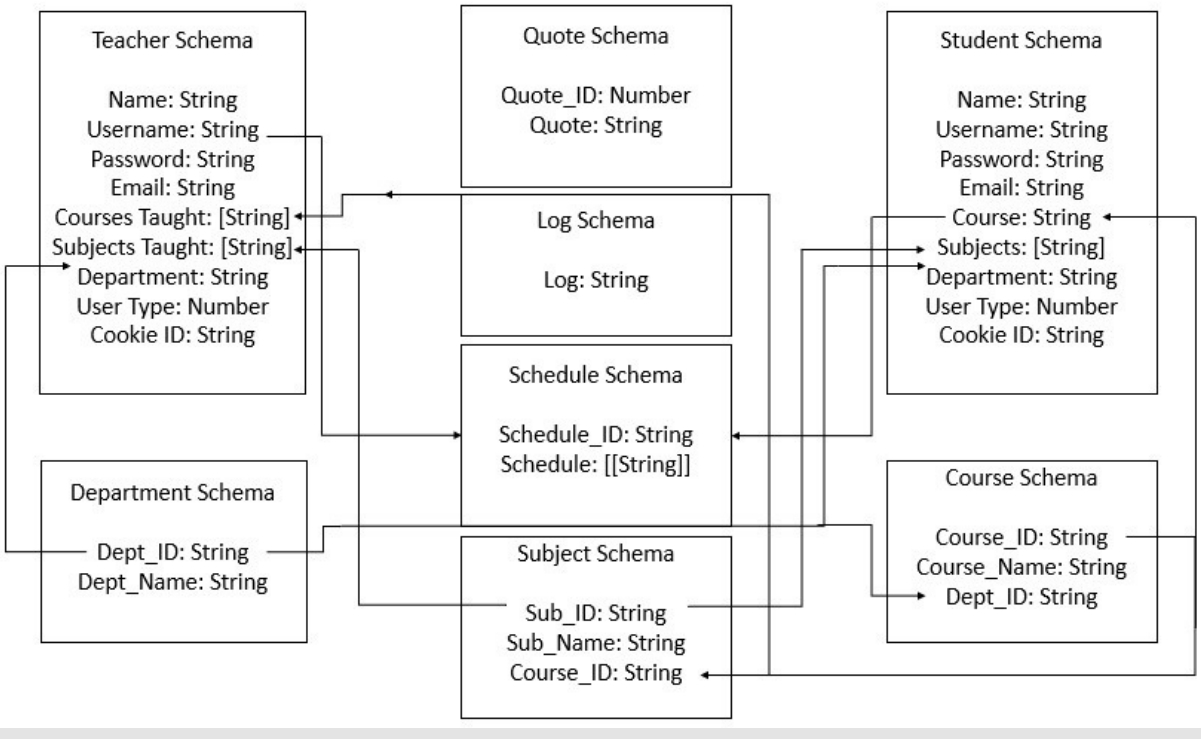

10:11:54
25-01-2023
Good Morning, Student 0
Student 0
Department
Qualifications
Edit User
Back
Email
Phone

Database Design

The MongoDB database can have models of each collection defined using Mongoose

The Collections present in the Database are:

1. Departments
2. Courses
3. Subjects
4. Teachers
5. Students
6. StudentSchedules
7. TeacherSchedules
8.Quotes
9. Logs



SAI Timetable Gen

The final MERN stack Web Application for scheduling timetables

With all the prerequisites ready for the development process, the development went smooth and steady. A few changes were made during the course of development of the App and a few more features were included after discussion with the project supervisor. All the functionalities mentioned in the flow chart diagram for all the users have been implemented and are functional. Admin using the final app can:

- Add, Edit and Delete Users (No Signup needed for individual users)
- Add, Edit and Delete Course Details
- Create, Edit and Delete Schedules

And apart from these main features, the app also lets the admin perform a Master Reset (wiping off the database) and also has permissions to check the Logs of the app. The Logging feature is ever running and it makes sure every action taken in the app by any user is logged and can be viewed by the admin.

Functionalities

Add Users

```
const reqPayload = {
  name: payload.get("teachername"),
  username: payload.get("teacherid"),
  password: "sairam",
  email: payload.get("emailid"),
  department: payload.get("deptid"),
  coursesTaught: payload.get("coursestaught").split(","),
  subjectsTaught: payload.get("subjectstaught").split(","),
  usertype: user,
  mastername: props.User.name,
};
fetch("http://192.168.34.129:8000/add-teacher", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

```
const reqPayload = {
  name: payload.get("studentname"),
  username: payload.get("regdno"),
  password: "sairam",
  email: payload.get("emailid"),
  department: payload.get("deptid"),
  course: payload.get("courseid"),
  usertype: 2,
  mastername: props.User.name,
};
fetch("http://192.168.34.129:8000/add-student", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

These are the two fetch methods in the frontend that takes input from the form for Adding Teachers and Students. The request is sent to the server that is always running in the backend with a JSON object holding the inputs from the form.
The "mastername" attribute sent is to accomodate the logging of this event. This similar method is used almost all the fetch methods.
Few values like the User Type and password are hardcoded as default values for each user. Admin's usertype is always 0, 1 for teacher and 2 for student. The password is by default "sairam" for all and can be later changed by the users themselves.

Add Course Details

```
const reqPayload = {
  sub_id: payload.get("subjectid"),
  sub_name: payload.get("subjectname"),
  course_id: props.courseClicked,
  username: props.User.name,
};
fetch("http://192.168.34.129:8000/add-subject", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

```
const reqPayload = {
  course_id: payload.get("courseid"),
  course_name: payload.get("coursename"),
  dept_id: props.buttonClicked,
  username: props.User.name,
};
fetch("http://192.168.34.129:8000/add-course", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

```
const reqPayload = {
  dept_id: payload.get("deptid"),
  dept_name: payload.get("deptname"),
  username: props.User.name,
};
fetch("http://192.168.34.129:8000/add-department", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

The fetch methods for adding course details such as Departments, Courses and Subjects. Handler functions in the backend will receive and parse the request body and do database querying based on the request sent and send back a success message. This applies for almost all the handler functions.

But it will get difficult for the admin to add all these course details and users individually. The "Add Cohort" feature was added with this problem in mind. A CSV file can be uploaded to add multiple course details or multiple users all at the same time.

```
async function handleCohortSubmit(flag, event) {
  let myPromise = new Promise(function (resolve) {
    Papa.parse(event.target.files[0], {
      header: true,
      skipEmptyLines: true,
      complete: function (results) {
        cohort = results.data;
        resolve(cohort);
      },
    });
  });
  // console.log(event.target.files[0]);
  let payload = await myPromise;
  const reqPayload = {
    payload: payload,
    flag: flag,
  };
  fetch("http://192.168.34.129:8000/add-cohort", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    mode: "cors",
    body: JSON.stringify(reqPayload),
  });
```

Create and Edit Schedules

```
const schedule = [
  payload.getAll("Monday"),
  payload.getAll("Tuesday"),
  payload.getAll("Wednesday"),
  payload.getAll("Thursday"),
  payload.getAll("Friday"),
  payload.getAll("Saturday"),
];
const reqPayload = {
  schedule_id: tuname,
  schedule: schedule,
  username: props.User.name,
  teachername: tname,
};

fetch("http://192.168.34.129:8000/create-schedule", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

```
const schedule = [
  payload.getAll("Monday"),
  payload.getAll("Tuesday"),
  payload.getAll("Wednesday"),
  payload.getAll("Thursday"),
  payload.getAll("Friday"),
  payload.getAll("Saturday"),
];
const reqPayload = {
  schedule_id: props.teacherid,
  schedule: schedule,
  username: props.User.name,
  teachername: tname,
};

fetch("http://192.168.34.129:8000/edit-schedule", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

The Create and Edit Schedules work in the same way except and get all the values on the 6*6 table entered by the user and makes it into a 6*6 array and sends it to the backend along with the information of who edited the schedule and whose schedule was edited for the backend handler to query and find the existing schedule and make changes. Incase of creating a schedule, a new schedule is created for that teacher and the array is sent as the schedule into the database.

Edit Users

Both teachers and students can only edit their password while admin can edit all the details of the users. This hierarchy is put in place to avoid any unwanted changes in details of the users.
This way, the Admin has full control over the creation and edition of the user.

```
if (
  props.User.usertype === 1 ||
  props.User.usertype === 9 ||
  props.User.usertype === 0
) {
  reqPayload = {
    old_username: user_id,
    new_username: payload.get("username"),
    new_password: payload.get("password"),
    new_email: payload.get("emailid"),
    new_course: payload.get("course").split(","),
    new_subject: payload.get("subjects").split(","),
    new_department: payload.get("department"),
    usertype: usertype,
  };
} else {
  reqPayload = {
    old_username: user_id,
    new_username: payload.get("username"),
    new_password: payload.get("password"),
    new_email: payload.get("emailid"),
    new_course: payload.get("course"),
    new_subject: payload.get("subjects").split(","),
    new_department: payload.get("department"),
  };
}

fetch("http://192.168.34.129:8000/edit-details", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

```
const reqPayload = {
  username: user_id,
  new_password: payload.get("new_password"),
};
fetch("http://192.168.34.129:8000/edit-user", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
})
```

Edit Course Details

```
const reqPayload = {
  old_dept_id: dept_id,
  new_dept_id: payload.get("dept_id"),
  new_dept_name: payload.get("deptname"),
};
fetch("http://192.168.34.129:8000/edit-department", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
});
```

```
const reqPayload = {
  old_course_id: course_id,
  new_course_id: payload.get("courseid"),
  new_course_name: payload.get("coursename"),
  new_dept_id: payload.get("dept_id"),
};
fetch("http://192.168.34.129:8000/edit-course", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
});
```

```
const reqPayload = {
  old_sub_id: sub_id,
  new_sub_id: payload.get("subid"),
  new_sub_name: payload.get("subname"),
  new_course_id: payload.get("course_id"),
};
fetch("http://192.168.34.129:8000/edit-subject", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(reqPayload),
});
```

Edit Course Details also work in the similar manner as Add Course Details. The fetch methods are displayed above for reference. The same methodology is applied for Deleting Course Details too.

Delete Course Details

```
fetch("http://192.168.34.129:8000/delete-department", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify({
    dept_id: dept_id,
    username: props.User.name,
  }),
})
```

```
fetch("http://192.168.34.129:8000/delete-course", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify({
    course_id: course_id,
    username: props.User.name,
  }),
})
```

```
fetch("http://192.168.34.129:8000/delete-subject", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify({
    sub_id: sub_id,
    username: props.User.name,
  }),
})
```

Delete Users

```
fetch("http://192.168.34.129:8000/delete-user", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify({
    user_id: user_id,
  }),
})
```

Delete Schedules

```
fetch("http://192.168.34.129:8000/delete-schedule", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify({
    schedule_id: schedule_id,
    teachername: schedule_id,
    username: props.User.name,
  }),
})
```

It should be observed that all the methods we have seen now have been repetitive and all work in the same way. That is the use of having React as the front-end as it offers us to use these fetch methods to send data in a specific tailored way and in-turn let's the back-end server just get the request JSON and apply only the back-end logic like calculation, mapping, querying, etc. This separation helps in keeping the back-end server code clean and compact. It is to be noted that this methodology has led to having only around 1.5k lines of code in the back-end avoiding redundancy of code in the back-end.

Backend Logic

Create Schedules

```
for (let i = 0; i < teacherFound.coursesTaught.length; i++) {
  console.log(teacherFound.coursesTaught[i]);
  let newschedule = [];
  let subjects = [];
  let finalschedule = [];
  let ultimateschedule = [];
```

```
for (let j = 0; j < teacherFound.subjectsTaught.length; j++) {
  console.log(teacherFound.subjectsTaught[j]);
  let myPromise = new Promise(function (resolve) {
    Subjects.findOne(
      {
        sub_id: tea (parameter) teacherFound: any
        course_id: teacherFound.coursesTaught[i],
      },
```

```
for (let a = 0; a < schedule.length; a++) {
  for (let b = 0; b < schedule.length; b++) {
    for (let c = 0; c < subjectsOfTeacher.length; c++) {
      console.log(subjectsOfTeacher[c]);
      if (schedule[a][b] === subjectsOfTeacher[c]) {
        console.log("in final if");
        finalschedule[a][b] = schedule[a][b];
        resolve(finalschedule);
      } else {
        resolve(finalschedule);
      }
    }
  }
}
```

The Create Schedule function runs in nested loops. The outermost loop runs for the total number of courses taught by the teacher and creates that many empty schedules Then, in the second loop, it runs for the total number of subjects taken by the teachers and checks which subject is part of which course and then finally the innermost loop runs under a double loop checking the schedule of the

Cont.

teacher and each subjects will be mapped to the new schedules of their respective course in the same exact index of the array as the original one. Edit Schedules is the same as Create Schedules.

Edit Schedules

```
for (let i = 0; i < teacherFound.coursesTaught.length; i++) {
  console.log(teacherFound.coursesTaught[i]);
  let newschedule = [];
  let subjects = [];
  let finalschedule = [];
  let ultimateschedule = [];
```

```
for (let j = 0; j < teacherFound.subjectsTaught.length; j++) {
  console.log(teacherFound.subjectsTaught[j]);
  let myPromise = new Promise(function (resolve) {
    Subjects.findOne(
      {
        sub_id: teacherFound.subjectsTaught[j],
        course_id: teacherFound.coursesTaught[i],
      },
```

```
for (let a = 0; a < schedule.length; a++) {
  for (let b = 0; b < schedule.length; b++) {
    for (let c = 0; c < subjectsOfTeacher.length; c++) {
      console.log(subjectsOfTeacher[c]);
      if (schedule[a][b] === subjectsOfTeacher[c]) {
        console.log("in final if");
        finalschedule[a][b] = schedule[a][b];
```

Here, it checks the new edited schedule sent from the front-end and compares the subjects in that with the subject taught by him and whenever that condition is satisfied, that index is mapped to the already existing schedule for students and modifies it in the database. The looping does not change from Create Schedule's looping structure.

Delete Schedules

The looping structure changes in deleting schedules. First the teacher's schedule is saved in an array before deleted and the same outer loop to run through all the courses taught and another loop under a array traversal to check the subjects taught and each subject is checked and deleted from its respective schedule based on what course it is part of.

Cont.

```
for (let i = 0; i < teacherFound[0].coursesTaught.length; i++) {
  let arr;
  studentSchedules.findOne(
    {
      schedule_id: teacherFound[0].coursesTaught[i],
    },
    (err, scheduleFound) => {
      if (scheduleFound) {
        arr = scheduleFound.schedule;
        for (let j = 0; j < arr.length; j++) {
          for (let k = 0; k < arr.length; k++) {
            for (
              let h = 0;
              h < teacherFound[0].subjectsTaught.length;
              h++
            ) {
              if (
                arr[j][k] === teacherFound[0].subjectsTaught[h]
              ) {
                arr[j][k] = "Free";
              }
            }
          }
        }
```

Add, Edit and Delete Course Details

Adding, Editing and Deleting Course Details are all the same and are very simple. Deleting although needed a little bit extra work since deleting a department means that all the courses under it and all the subjects under it need to be deleted. It also means that all the student schedules automatically created when creating a course should also be deleted. The snippet of the delete department back-end handler is attached for reference.

```
Departments.deleteOne(
  {
    dept_id: req.body.dept_id,
  },
  (err) => {
    if (err) throw err;
    else {
      const date = new Date().toLocaleTimeString();
      const day = new Date().toDateString();
      let logString =
        req.body.username +
        " " +
        "deleted department " +
        req.body.dept_id +
        " at " +
        date +
        " " +
        "on " +
        day +
        ".";
      Logs.create({ log: logString });
      res.send({ message: "success" });
      console.log("Deleted " + req.body.dept_id + " successfully");
    }
  }
);
```

Cont.

```
Courses.find(
  {
    dept_id: req.body.dept_id,
  },
  (err, coursesFound) => {
    if (err) throw err;
    else {
      coursesFound.map((course) => {
        Subjects.deleteMany({ course_id: course }, (err) => {
          if (err) throw err;
        });
        studentSchedules.findOneAndDelete(
          { schedule_id: course.course_id },
          (err) => {
            if (err) throw err;
          }
        );
      });
      Courses.deleteMany({ dept_id: req.body.dept_id }, (err) => {
        if (err) throw err;
      });
    }
  }
);
```

You, last month • Delete Done …

Add Cohort

The next feature that has a bit of more backend logic but still lesser than create, edit and delete schedule is Add Cohort. That too mainly when adding a cohort of Courses where creation of empty schedules for each course is required. The snippet of this part of the code is attached for reference.

```
Courses.insertMany(req.body.payload, (err) => {
  if (err) throw err;
  else {
    async function createEmptySchedules() {
      let myPromise = new Promise((resolve) => {
        let newschedule = [];
        for (let i = 0; i < 6; i++) {
          newschedule.push([]);
          for (let j = 0; j < 6; j++) {
            newschedule[i].push("Free");
          }
        }
        resolve(newschedule);
      });
      for (let i = 0; i < req.body.payload.length; i++) {
        studentSchedules.create({
          schedule_id: req.body.payload[i].course_id,
          schedule: await myPromise,
        });
      }
    }
    createEmptySchedules();
    res.send({ message: "success" });
  }
});
```

Miscellaneous Features

Retain Session

In-order to retain a user session, whenever the app is refreshed or opened after closing, the front-end sends a request to the back-end to check for the cookies for the current user and if the cookies of the user's database and the current cookie matches, then the session is retained. The front-end fetch method and the back-end handler logic is attached for reference.

```
const requestPayload = {
  username: cookie.username,
  cookieID: cookie.cookieID,
};
fetch("http://192.168.34.129:8000/retain-session", {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  mode: "cors",
  body: JSON.stringify(requestPayload),
})
```

```
app.post("/retain-session", (req, res) => {
  Teachers.findOne(
    {
      username: req.body.username,
    },
    (err, teacherFound) => {
      if (!teacherFound) {
        Students.findOne(
          {
            username: req.body.username,
          },
          (error, studentFound) => {
            if (error) throw error;
            else if (studentFound) {
              if (studentFound.cookieID === req.body.cookieID) {
                res.send({
                  message: "success",
                  user: studentFound,
                });
```

Next is the logic for Master Reset.

Master Reset

The master reset functionality is to clear or perform a full reset of the app; meaning that the entire database except for the logs get erased. Logs can be cleared by the admin separately if wanted.

Cont.

```
app.post("/flush-app", (req, res) => {
  //mongoose.connection.db.dropDatabase();
  Departments.deleteMany({}, (err) => {
    if (err) throw err;
    Courses.deleteMany({}, (err) => {
      if (err) throw err;
      studentSchedules.deleteMany({}, (err) => {
        if (err) throw err;
      });
    });
    Subjects.deleteMany({}, (err) => {
      if (err) throw err;
      Subjects.create({
        sub_id: "Free",
        sub_name: "Free",
      });
    });
  });
  Teachers.deleteMany({}, async (err) => {
    if (err) throw err;
    teacherSchedules.deleteMany({});
    const hashedPassword = await bcrypt.hash("2732", saltRounds);
    Teachers.create({
      name: "Master User",
      username: "master",
      password: hashedPassword,
      email: "master@mdh.edu.in",
      usertype: 9,
    });
  });
  Students.deleteMany({}, (err) => {
    if (err) throw err;
  });
  console.log("Master Reset complete");
});
```

This makes sure that the fallback account "master user" and the free subject never get deleted even after a master reset. This master user can be used to add an admin user later after the master reset. NOTE: A full wipe of the database can also happen due to database failure but that happens rarely.

Help (Documentation)

```
<button type="button" className="dropdown-item">
  <a
    style={{ textDecoration: "none", color: "black" }}
    href={require("../../src/Admin_Documentation.pdf")}
    rel="noreferrer"
    // download="Admin_Documentation"
    //eslint-disable-next-line
    target="_blank"
  >
    Help
  </a>
</button>
```

The front-end serves the documentation to user to learn how to use the app or in-case they are stuck. The documentation explains for each user their actions that can be performed and contains examples of how the app is used for help and troubleshooting purposes.

Email Service

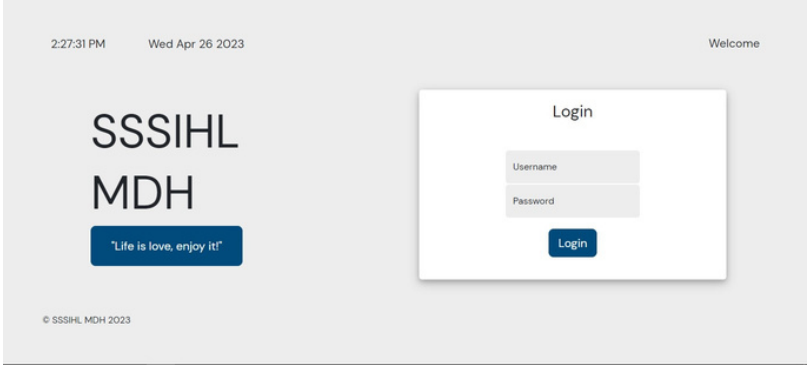
Whenever a user is added to the database, a mail is sent to them intimating their credentials via mail. When a teacher requests change, a mail is sent to the admin attached with it the changes required by the teacher. These email services are again done in the front-end using EmailJS. The snippet of the code for request change is attached for reference.

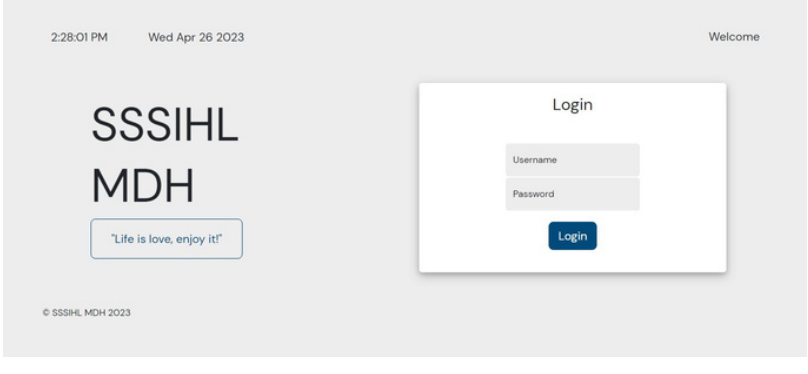
```
function handleRequest(event, name) {
  event.preventDefault();
  var templateParams = {
    to_name: "Admin",
    to_mail: "vidyasager162@gmail.com",
    from_name: name,
    message: event.target.issue.value,
  };
  emailjs
    .send("gmail", "req_change", templateParams, "1fuN55z3EW1BC5gy0")
    .then(
      (response) => {
        console.log("SUCCESS!", response.status, response.text);
      },
      (error) => {
        console.log("FAILED...", error);
      }
    );
}
```

Screenshots

The screenshots of all the pages of the final app are attached for reference

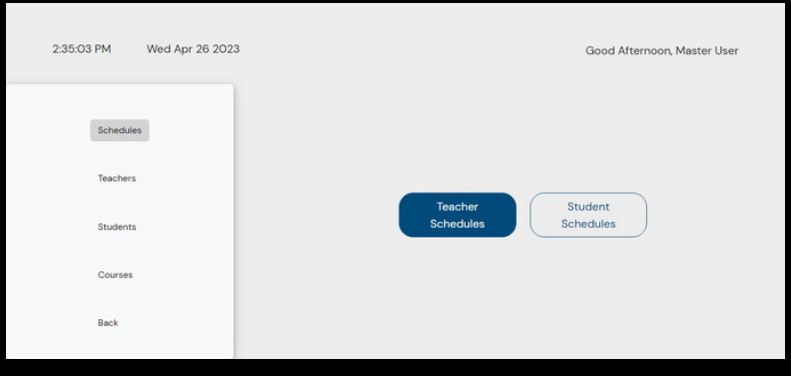
Home

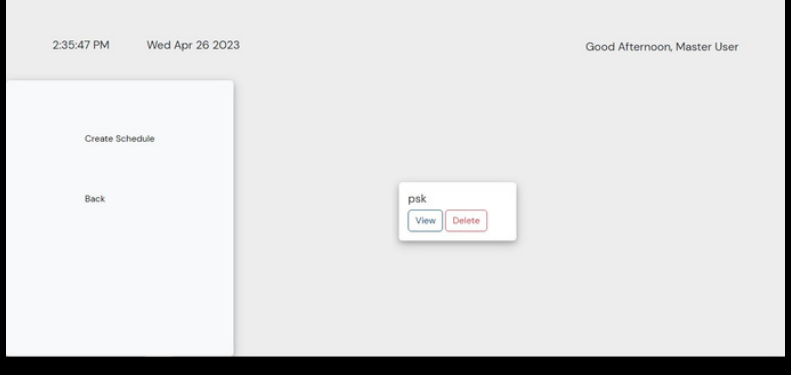




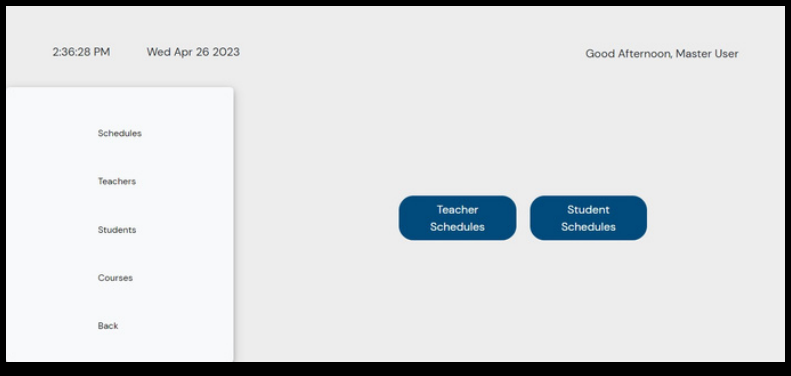
Admin

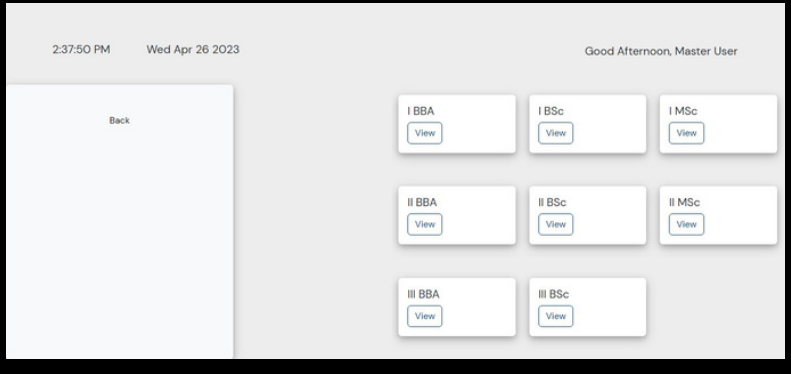

2:34:28 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen


2:35:03 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
Teacher Schedules
Student Schedules


2:35:47 PM
Wed Apr 26 2023
Good Afternoon, Master User
Create Schedule
Back
psk
View
Delete

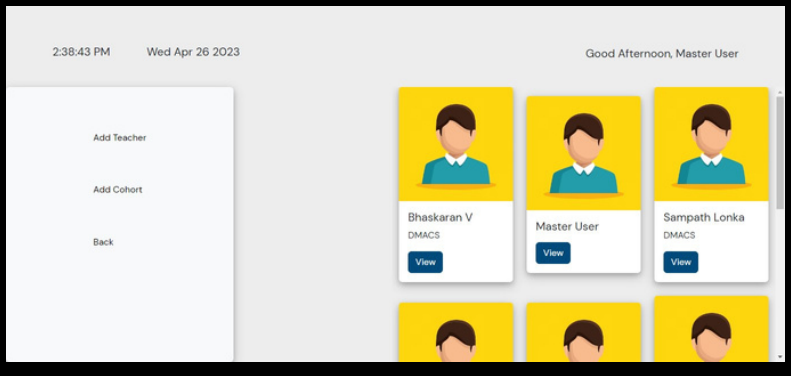
Cont.

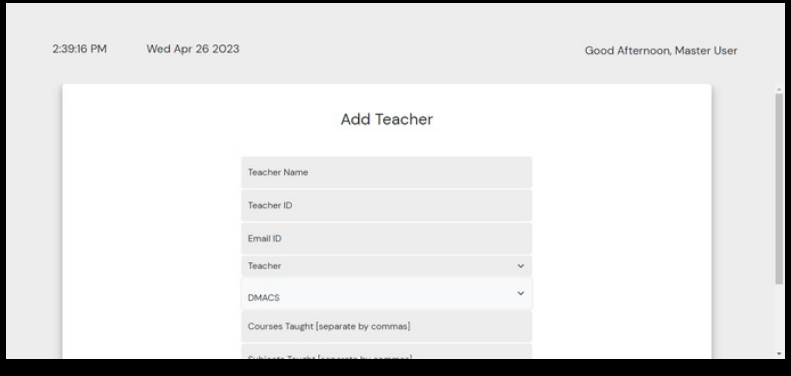

2:36:28 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
Teacher Schedules
Student Schedules

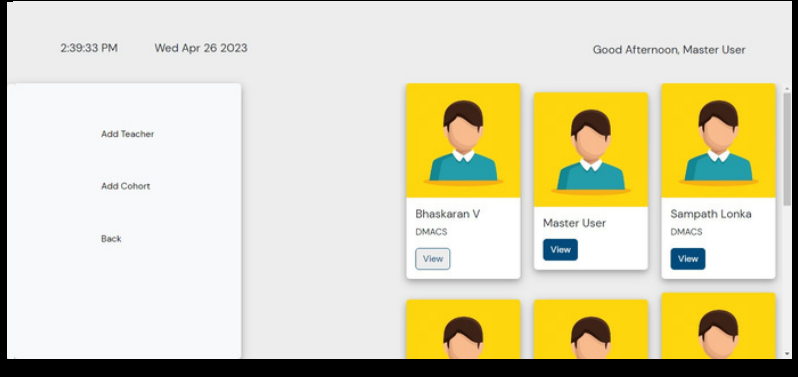

2:37:50 PM
Wed Apr 26 2023
Good Afternoon, Master User
Back
I BBA
View
I BSc
View
I MSc
View
II BBA
View
II BSc
View
II MSc
View
III BBA
View
III BSc
View


2:38:29 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen

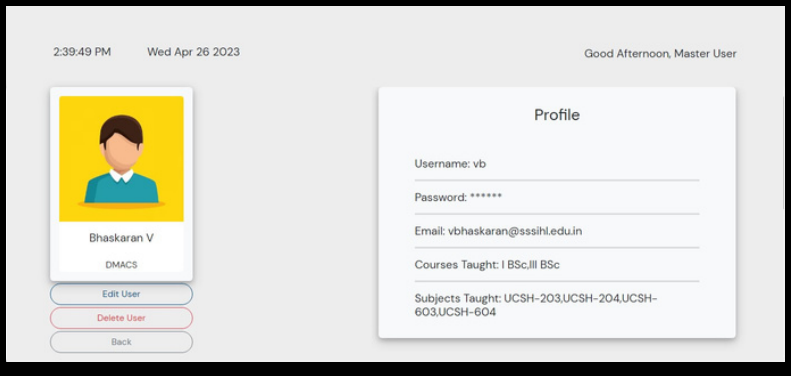
Cont.

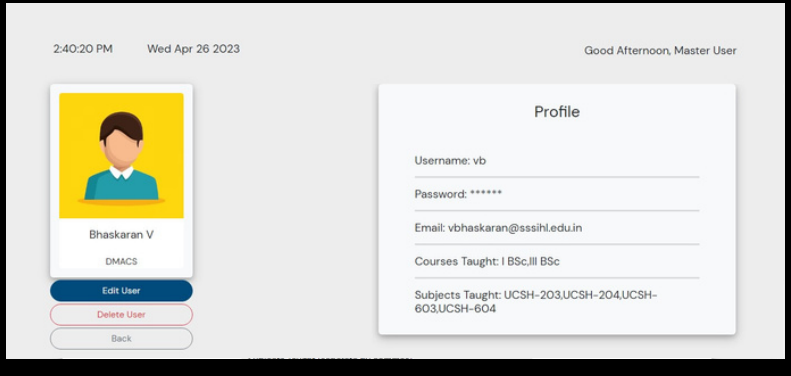

2:38:43 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Teacher
Add Cohort
Back
Bhaskaran V
DMACS
View
Master User
View
Sampath Lonka
DMACS
View

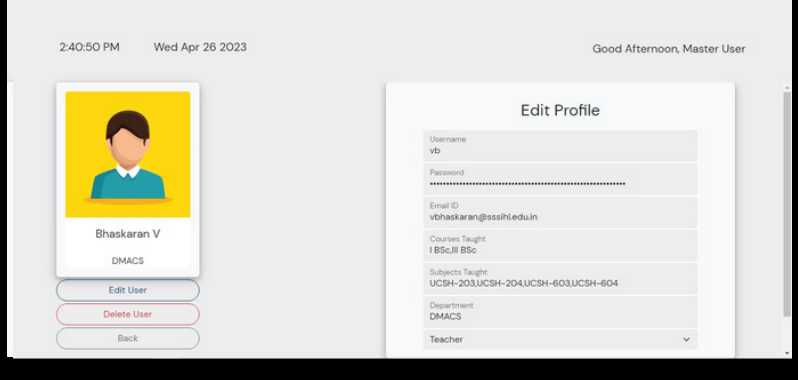

2:39:16 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Teacher
Teacher Name
Teacher ID
Email ID
Teacher
DMACS
Courses Taught [separate by commas]


2:39:33 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Teacher
Add Cohort
Back
Bhaskaran V
DMACS
View
Master User
View
Sampath Lonka
DMACS
View

Cont.

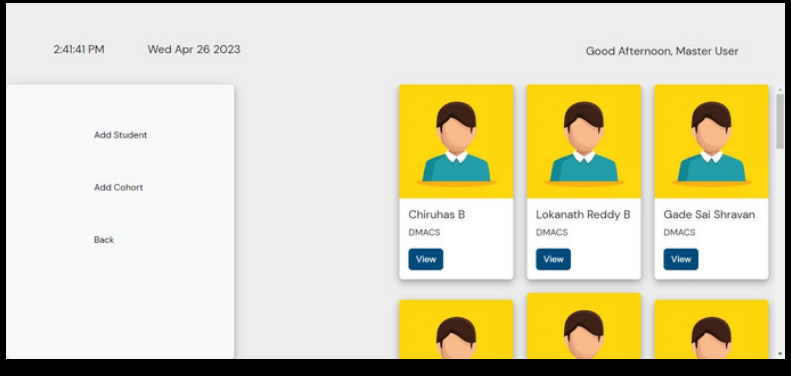

2:39:49 PM
Wed Apr 26 2023
Good Afternoon, Master User
Bhaskaran V
DMACS
Edit User
Delete User
Back
Profile
Username: vb
Password: ******
Email: vbhaskaran@sssihl.edu.in
Courses Taught: I BSc,III BSc
Subjects Taught: UCSH-203,UCSH-204,UCSH-603,UCSH-604

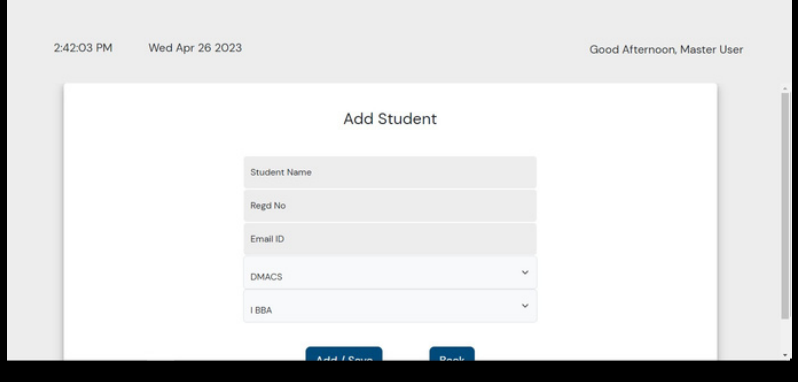

2:40:20 PM
Wed Apr 26 2023
Good Afternoon, Master User
Bhaskaran V
DMACS
Edit User
Delete User
Back
Profile
Username: vb
Password: ******
Email: vbhaskaran@sssihl.edu.in
Courses Taught: I BSc,III BSc
Subjects Taught: UCSH-203,UCSH-204,UCSH-603,UCSH-604


2:40:50 PM
Wed Apr 26 2023
Good Afternoon, Master User
Bhaskaran V
DMACS
Edit User
Delete User
Back
Edit Profile
Username
vb
Password
Email ID
vbhaskaran@sssihl.edu.in
Courses Taught
I BSc,III BSc
Subjects Taught
UCSH-203,UCSH-204,UCSH-603,UCSH-604
Department
DMACS
Teacher

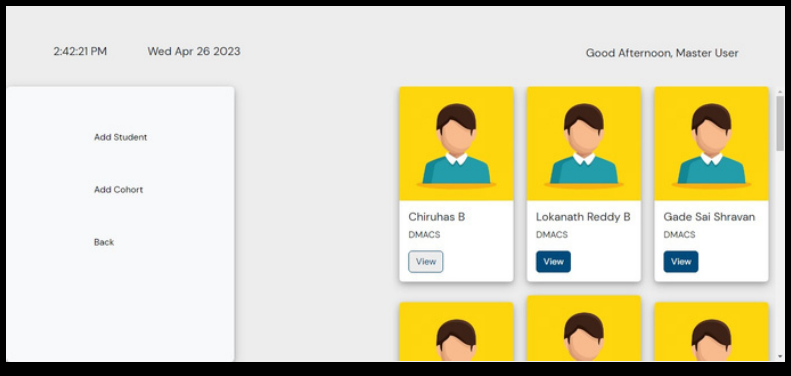
Cont.

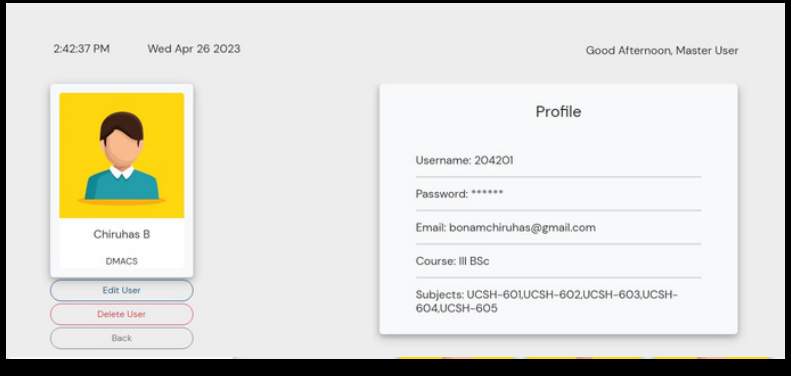

2:41:22 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen

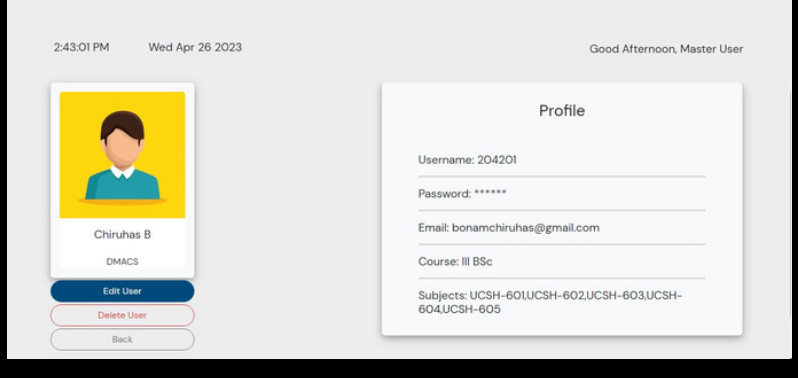

2:41:41 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Student
Add Cohort
Back
Chiruhas B
DMACS
View
Lokanath Reddy B
DMACS
View
Gade Sai Shravan
DMACS
View


2:42:03 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Student
Student Name
Regd No
Email ID
DMACS
I BBA

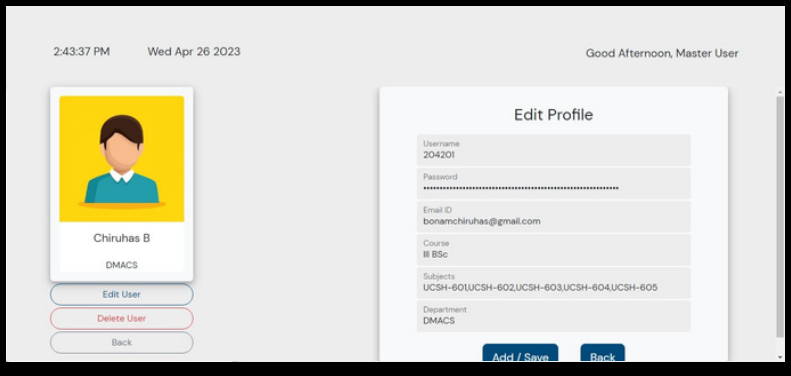
Cont.

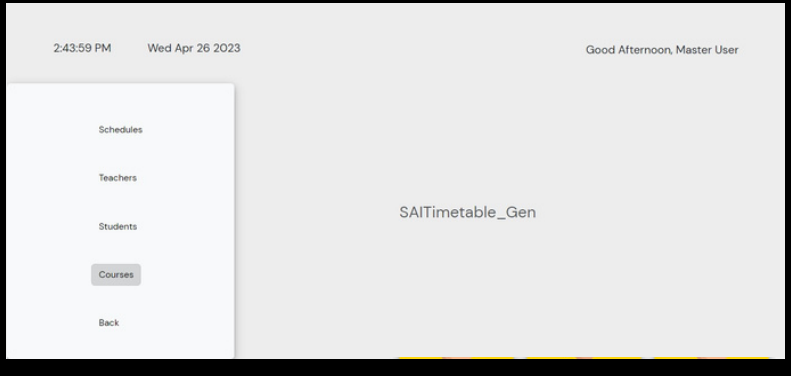

2:42:21 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Student
Add Cohort
Back
Chiruhas B
DMACS
View
Lokanath Reddy B
DMACS
View
Gade Sai Shravan
DMACS
View

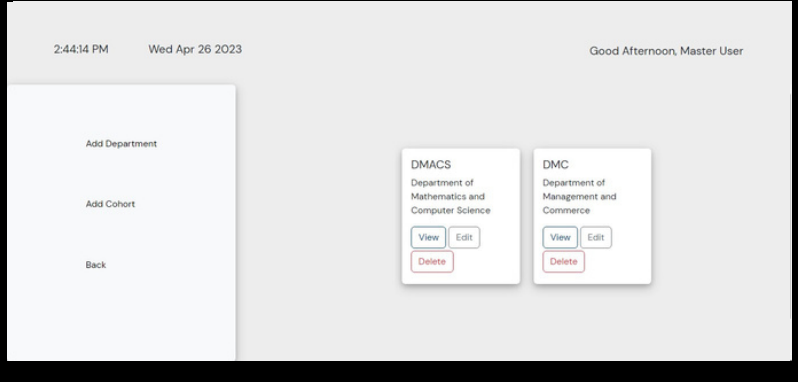

2:42:37 PM
Wed Apr 26 2023
Good Afternoon, Master User
Chiruhas B
DMACS
Edit User
Delete User
Back
Profile
Username: 204201
Password: ******
Email: bonamchiruhas@gmail.com
Course: III BSc
Subjects: UCSH-601,UCSH-602,UCSH-603,UCSH-604,UCSH-605


2:43:01 PM
Wed Apr 26 2023
Good Afternoon, Master User
Chiruhas B
DMACS
Edit User
Delete User
Back
Profile
Username: 204201
Password: ******
Email: bonamchiruhas@gmail.com
Course: III BSc
Subjects: UCSH-601,UCSH-602,UCSH-603,UCSH-604,UCSH-605

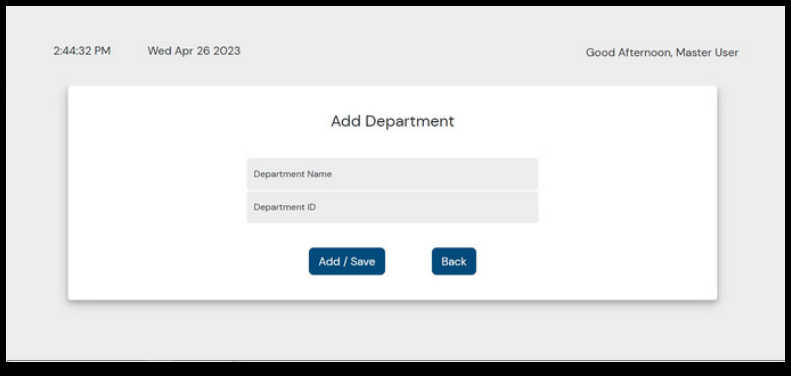
Cont.

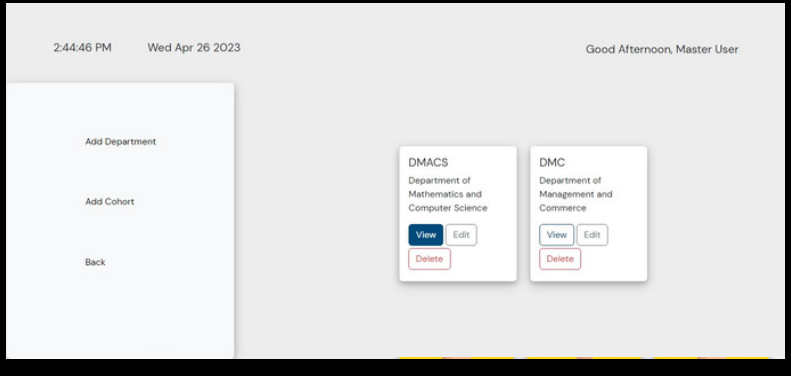

2:43:37 PM
Wed Apr 26 2023
Good Afternoon, Master User
Chiruhas B
DMACS
Edit User
Delete User
Back
Edit Profile
Username
204201
Password
Email ID
bonamchiruhas@gmail.com
Course
III BSc
Subjects
UCSH-601,UCSH-602,UCSH-603,UCSH-604,UCSH-605
Department
DMACS
Add / Save
Back

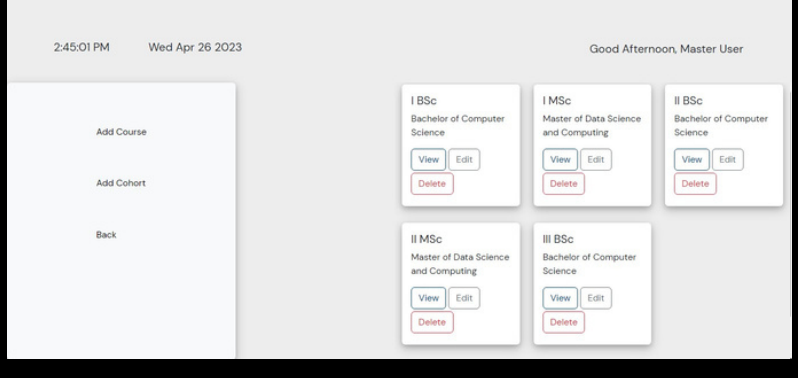

2:43:59 PM
Wed Apr 26 2023
Good Afternoon, Master User
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen


2:44:14 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Department
Add Cohort
Back
DMACS
Department of Mathematics and Computer Science
View
Edit
Delete
DMC
Department of Management and Commerce
View
Edit
Delete

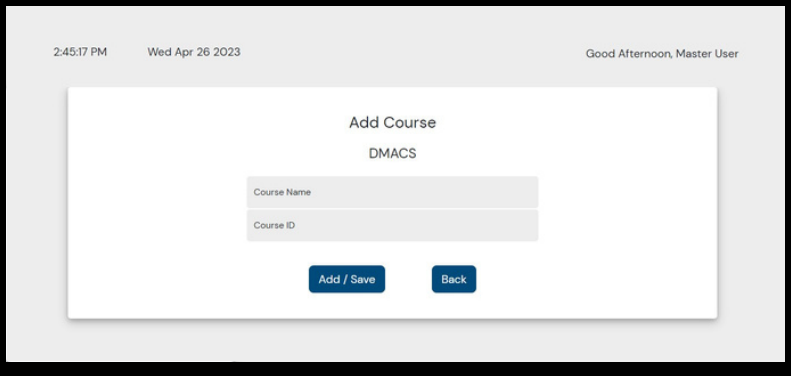
Cont.

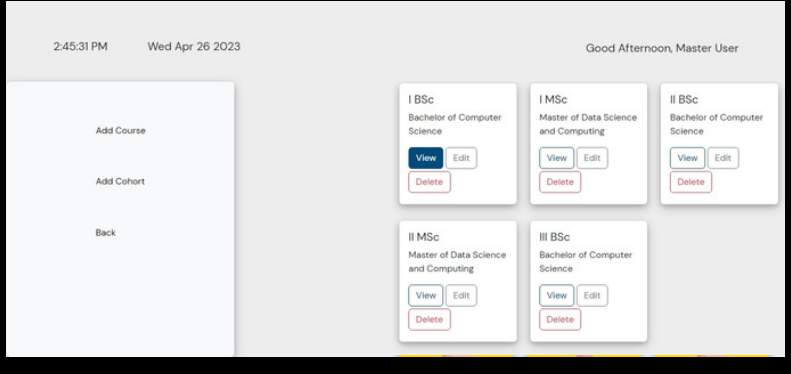

2:44:32 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Department
Department Name
Department ID
Add / Save
Back

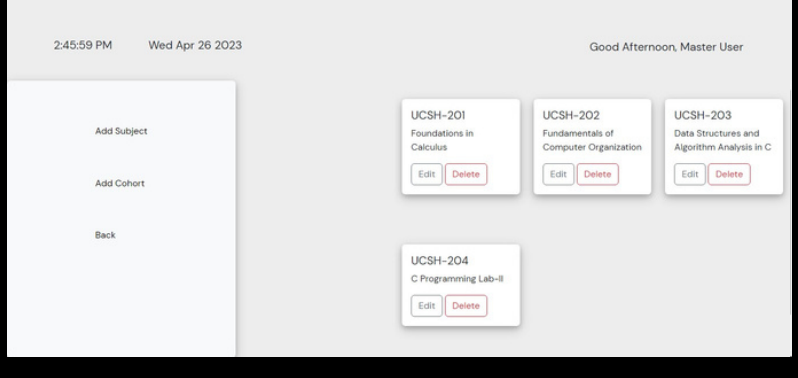

2:44:46 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Department
Add Cohort
Back
DMACS
Department of Mathematics and Computer Science
View
Edit
Delete
DMC
Department of Management and Commerce
View
Edit
Delete


2:45:01 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Course
Add Cohort
Back
I BSc
Bachelor of Computer Science
View
Edit
Delete
I MSc
Master of Data Science and Computing
View
Edit
Delete
II BSc
Bachelor of Computer Science
View
Edit
Delete
II MSc
Master of Data Science and Computing
View
Edit
Delete
III BSc
Bachelor of Computer Science
View
Edit
Delete

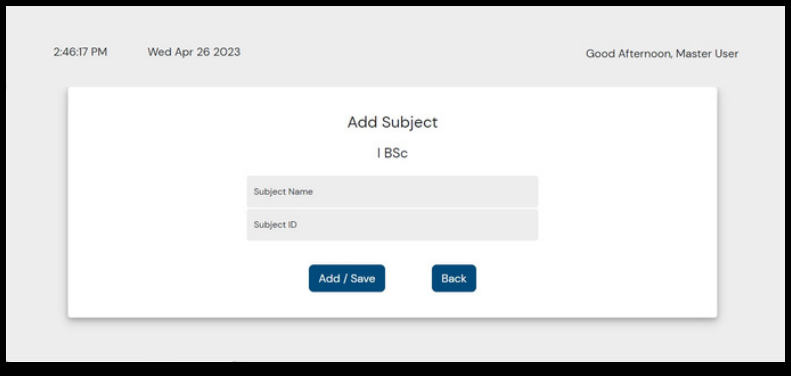
Cont.

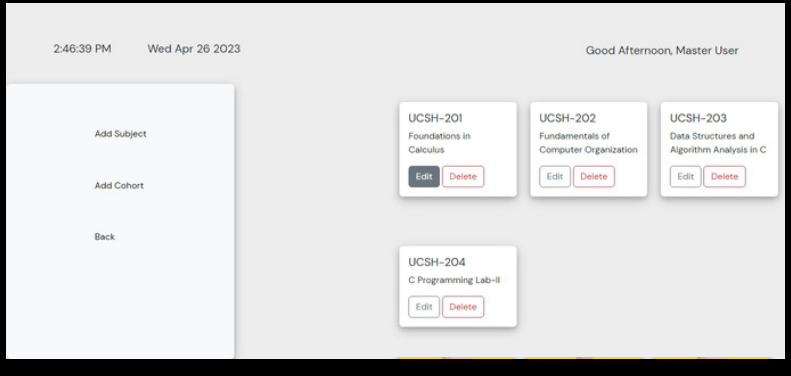

2:45:17 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Course
DMACS
Course Name
Course ID
Add / Save
Back

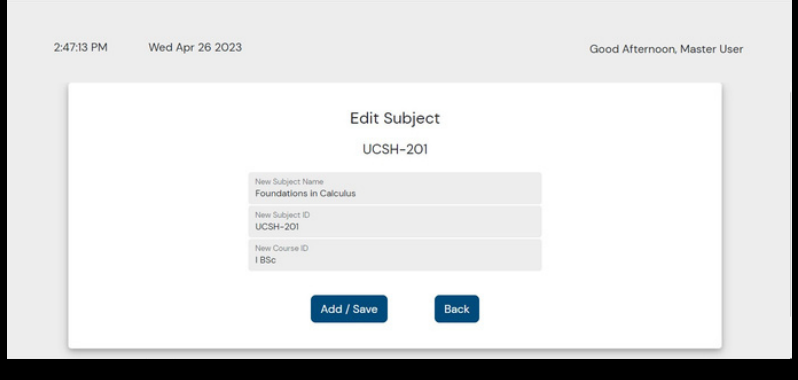

2:45:31 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Course
Add Cohort
Back
I BSc
Bachelor of Computer Science
View
Edit
Delete
I MSc
Master of Data Science and Computing
View
Edit
Delete
II BSc
Bachelor of Computer Science
View
Edit
Delete
II MSc
Master of Data Science and Computing
View
Edit
Delete
III BSc
Bachelor of Computer Science
View
Edit
Delete


2:45:59 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Subject
Add Cohort
Back
UCSH-201
Foundations in Calculus
Edit
Delete
UCSH-202
Fundamentals of Computer Organization
Edit
Delete
UCSH-203
Data Structures and Algorithm Analysis in C
Edit
Delete
UCSH-204
C Programming Lab-II
Edit
Delete

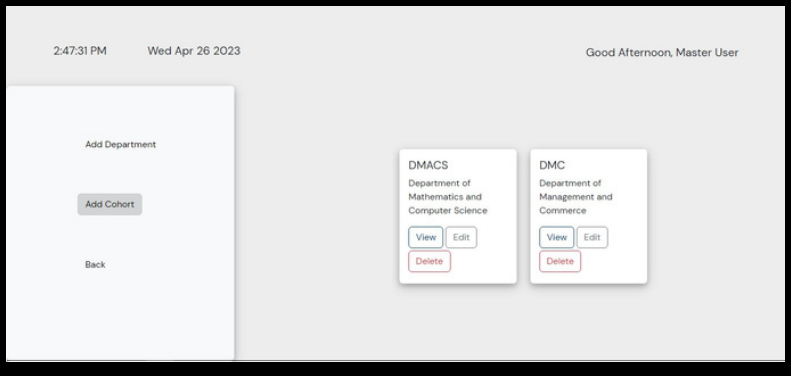
Cont.

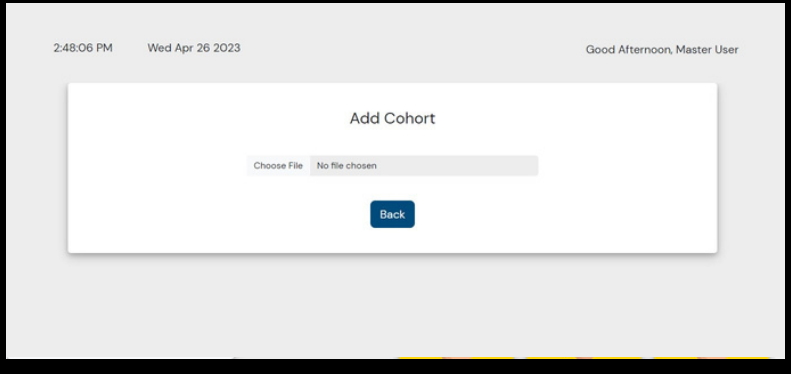

2:46:17 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Subject
I BSc
Subject Name
Subject ID
Add / Save
Back

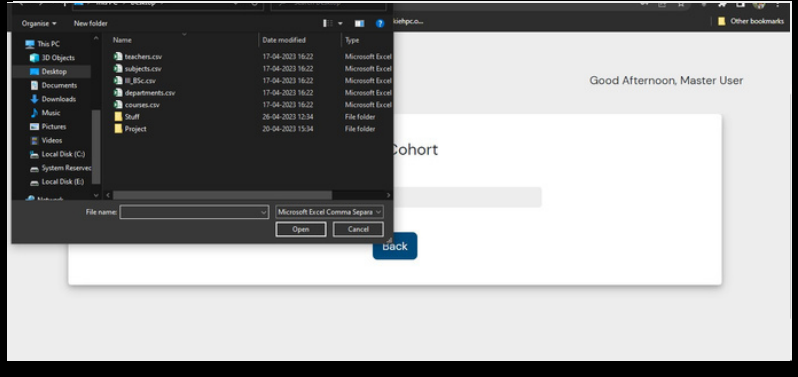

2:46:39 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Subject
Add Cohort
Back
UCSH-201
Foundations in Calculus
Edit
Delete
UCSH-202
Fundamentals of Computer Organization
Edit
Delete
UCSH-203
Data Structures and Algorithm Analysis in C
Edit
Delete
UCSH-204
C Programming Lab-II
Edit
Delete


2:47:13 PM
Wed Apr 26 2023
Good Afternoon, Master User
Edit Subject
UCSH-201
New Subject Name
Foundations in Calculus
New Subject ID
UCSH-201
New Course ID
I BSc
Add / Save
Back

Cont.

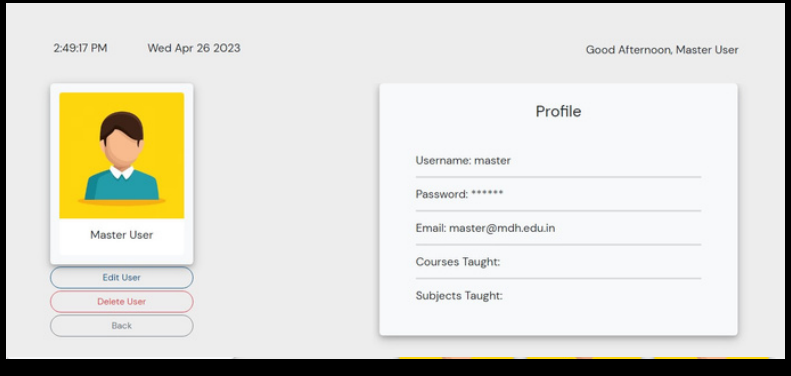

2:47:31 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Department
Add Cohort
Back
DMACS
Department of Mathematics and Computer Science
View
Edit
Delete
DMC
Department of Management and Commerce
View
Edit
Delete

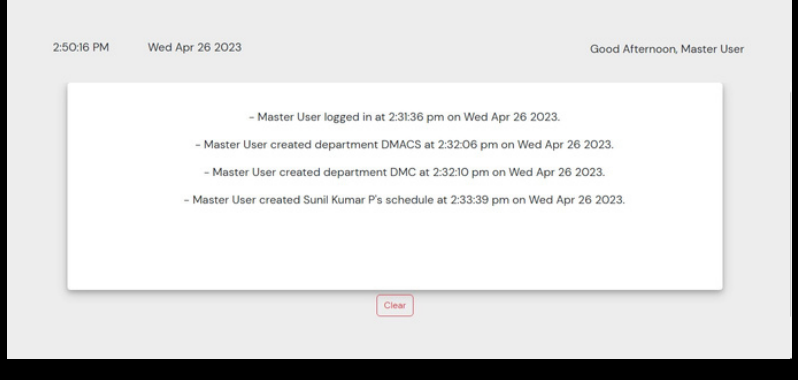

2:48:06 PM
Wed Apr 26 2023
Good Afternoon, Master User
Add Cohort
Choose File
No file chosen
Back


Organise
New folder
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
System Reserved
Local Disk (E:)
Name
Date modified
Type
teachers.csv
subjects.csv
III_BSc.csv
departments.csv
courses.csv
Stuff
Project
17-04-2023 16:22
26-04-2023 12:34
20-04-2023 15:34
Microsoft Excel
File folder
File name:
Microsoft Excel Comma Separa
Open
Cancel
Other bookmarks
Good Afternoon, Master User
Back

Cont.

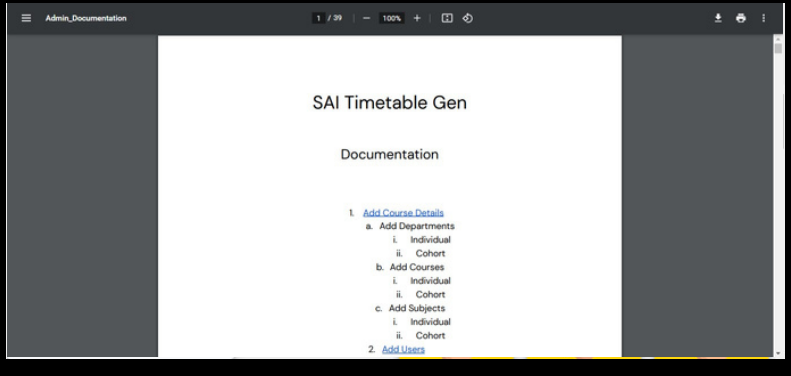

2:48:58 PM
Wed Apr 26 2023
Good Afternoon, Master User
Home
Profile
Master Reset
Logs
Help
Logout
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen


2:49:17 PM
Wed Apr 26 2023
Good Afternoon, Master User
Master User
Edit User
Delete User
Back
Profile
Username: master
Password: ******
Email: master@mdh.edu.in
Courses Taught:
Subjects Taught:


2:50:16 PM
Wed Apr 26 2023
Good Afternoon, Master User
- Master User logged in at 2:31:36 pm on Wed Apr 26 2023.
- Master User created department DMACS at 2:32:06 pm on Wed Apr 26 2023.
- Master User created department DMC at 2:32:10 pm on Wed Apr 26 2023.
- Master User created Sunil Kumar P's schedule at 2:33:39 pm on Wed Apr 26 2023.
Clear

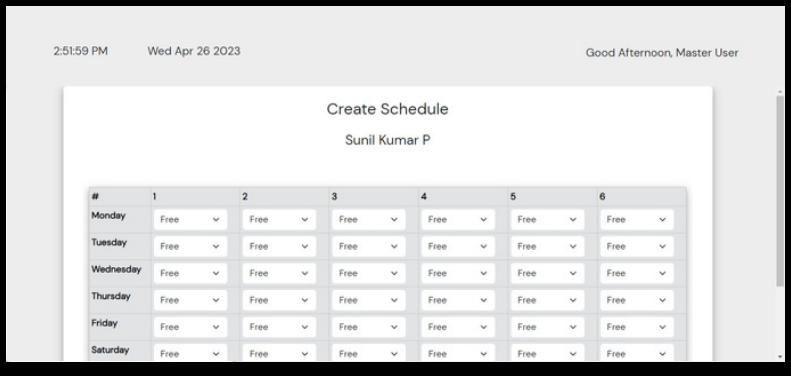
Cont.

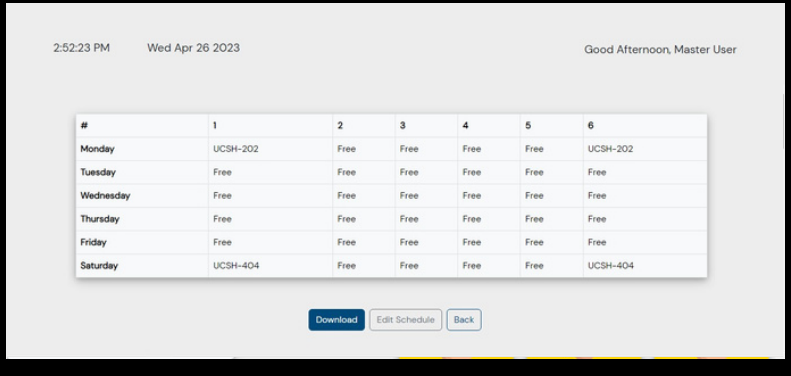

2:50:41 PM
Wed Apr 26 2023
Good Afternoon, Master User
Home
Profile
Master Reset
Logs
Help
Logout
Schedules
Teachers
Students
Courses
Back
SAITimetable_Gen

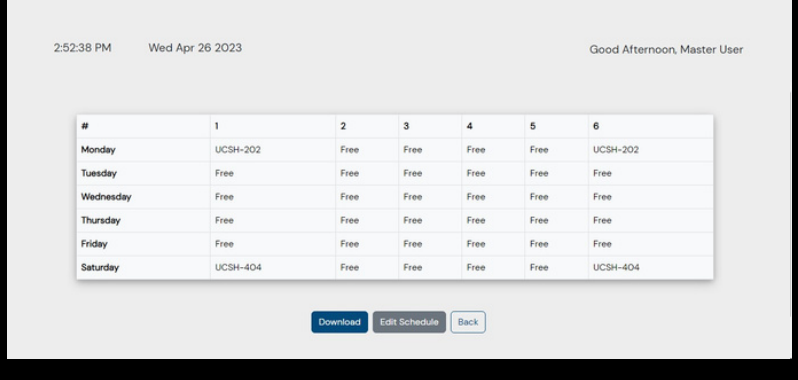

Admin_Documentation
1 / 39
100%
SAI Timetable Gen
Documentation
1. Add Course Details
a. Add Departments
i. Individual
ii. Cohort
b. Add Courses
i. Individual
ii. Cohort
c. Add Subjects
i. Individual
ii. Cohort
2. Add Users


2:51:32 PM
Wed Apr 26 2023
Good Afternoon, Master User
Create Schedule
Back
psk
View
Delete

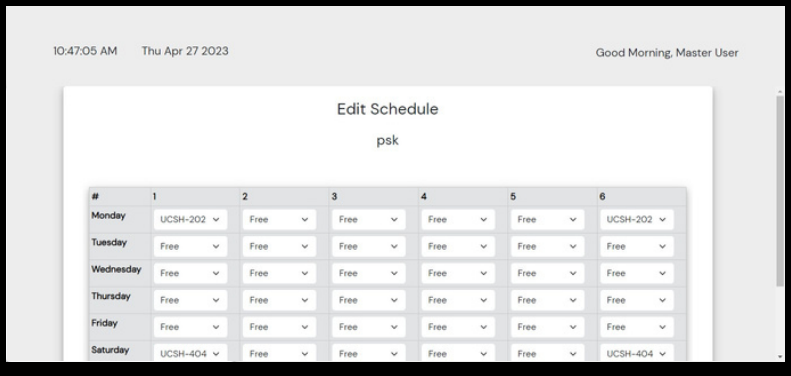
Cont.

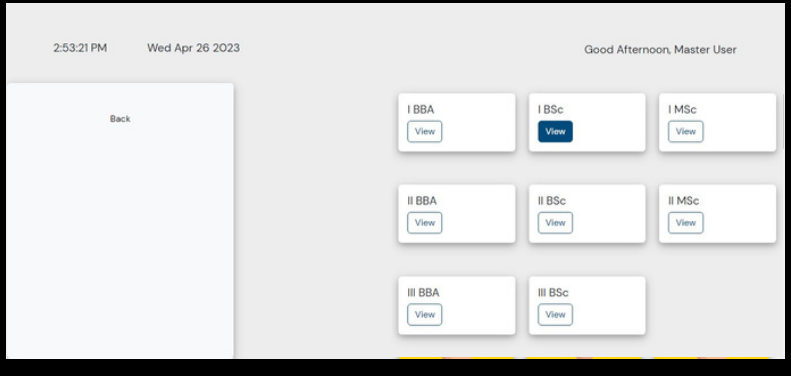

2:51:59 PM
Wed Apr 26 2023
Good Afternoon, Master User
Create Schedule
Sunil Kumar P
1 2 3 4 5 6
Monday Free Free Free Free Free Free
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday Free Free Free Free Free Free

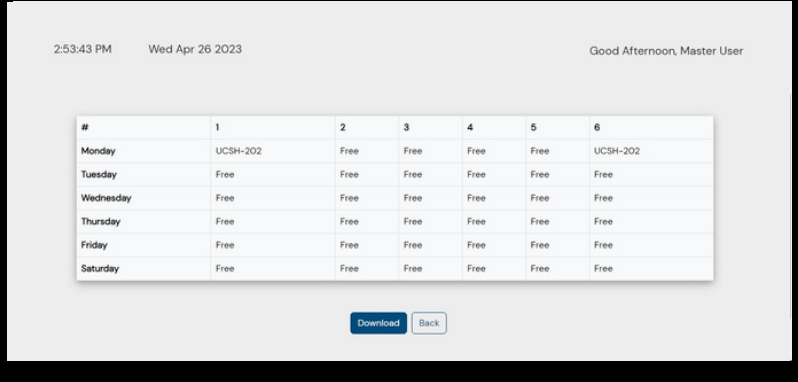

2:52:23 PM
Wed Apr 26 2023
Good Afternoon, Master User
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday UCSH-404 Free Free Free Free UCSH-404
Download
Edit Schedule
Back


2:52:38 PM
Wed Apr 26 2023
Good Afternoon, Master User
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday UCSH-404 Free Free Free Free UCSH-404
Download
Edit Schedule
Back

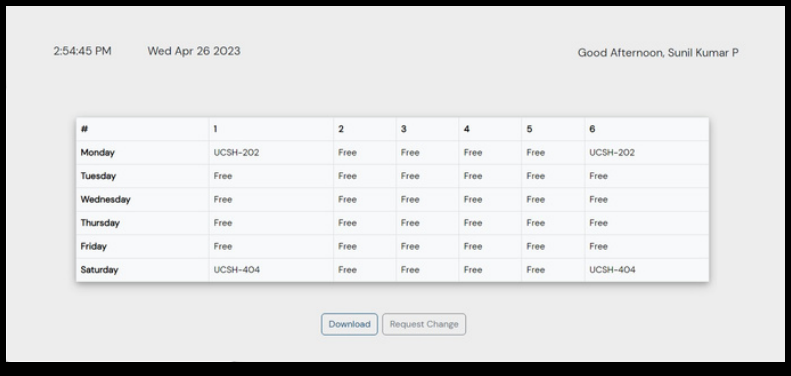
Cont.

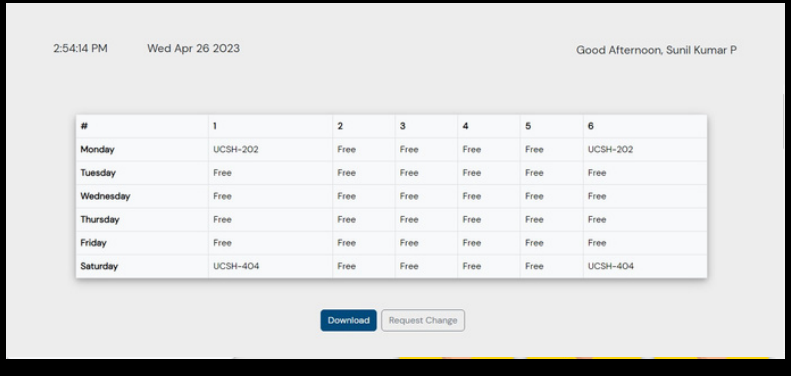

10:47:05 AM Thu Apr 27 2023
Good Morning, Master User
Edit Schedule
psk
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday UCSH-404 Free Free Free Free UCSH-404

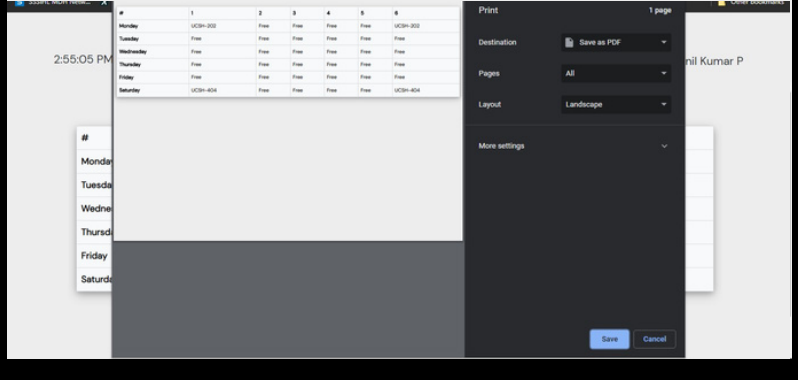

2:53:21 PM Wed Apr 26 2023
Good Afternoon, Master User
Back
I BBA View
I BSc View
I MSc View
II BBA View
II BSc View
II MSc View
III BBA View
III BSc View


2:53:43 PM Wed Apr 26 2023
Good Afternoon, Master User
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday Free Free Free Free Free Free
Download Back

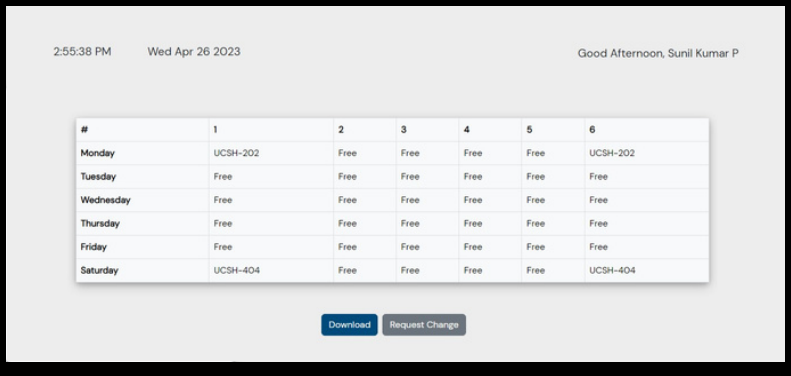
Teacher

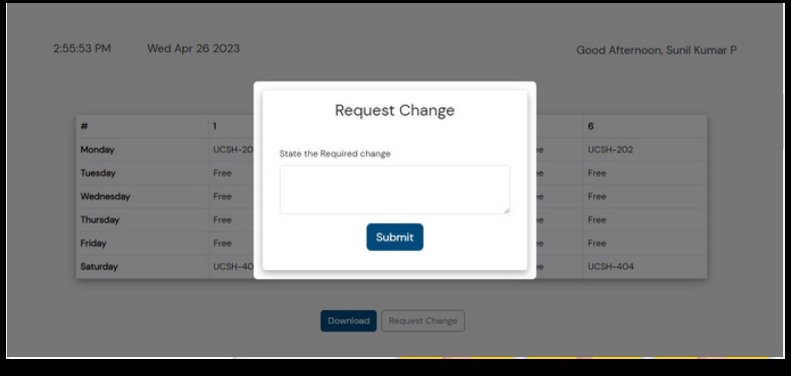

2:54:45 PM
Wed Apr 26 2023
Good Afternoon, Sunil Kumar P
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday UCSH-404 Free Free Free Free UCSH-404
Download
Request Change

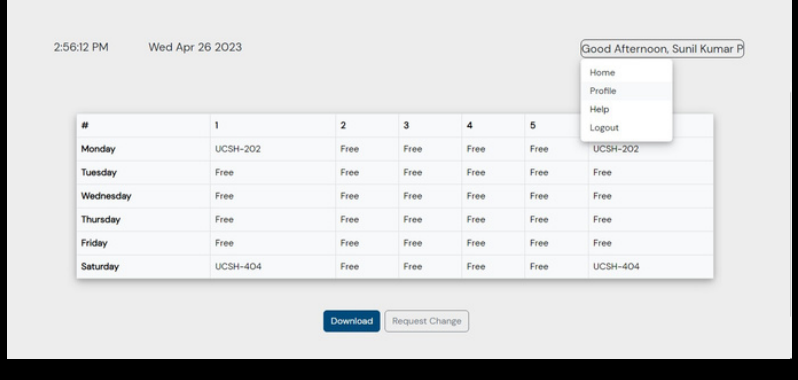

2:54:14 PM
Wed Apr 26 2023
Good Afternoon, Sunil Kumar P
1 2 3 4 5 6
Monday UCSH-202 Free Free Free Free UCSH-202
Tuesday Free Free Free Free Free Free
Wednesday Free Free Free Free Free Free
Thursday Free Free Free Free Free Free
Friday Free Free Free Free Free Free
Saturday UCSH-404 Free Free Free Free UCSH-404
Download
Request Change


2:55:05 PM
Print
1 page
Destination
Save as PDF
Pages
All
Layout
Landscape
More settings
Save
Cancel

Cont.

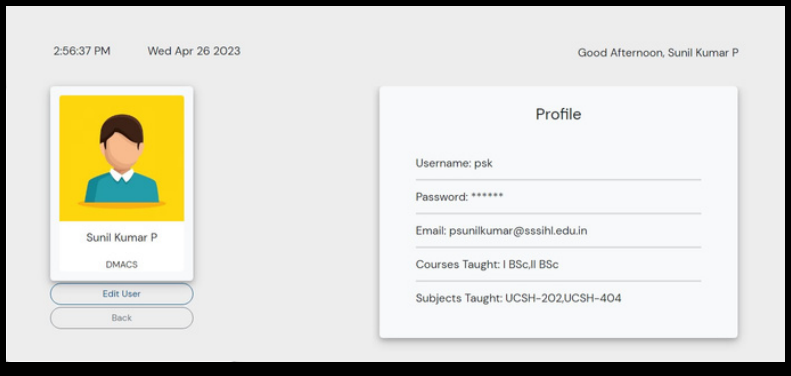


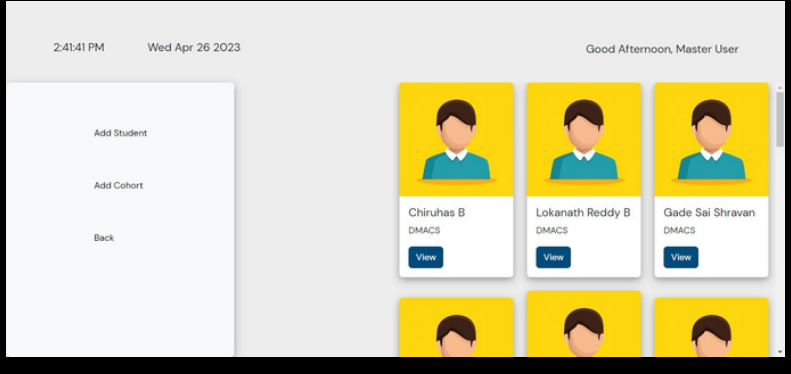




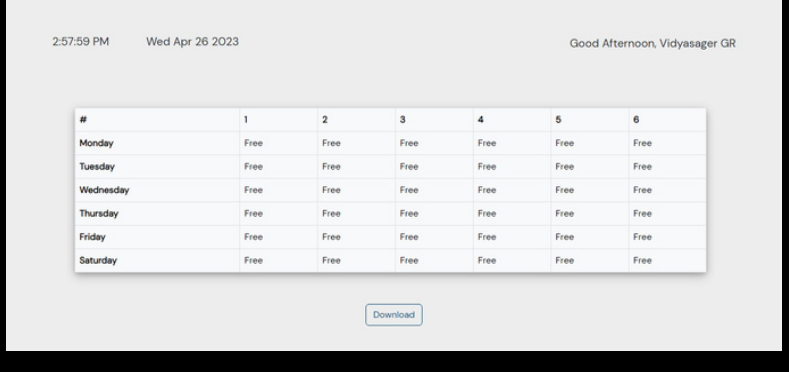
#	1	2	3	4	5	6
Monday	UCSH-202	Free	Free	Free	Free	UCSH-202
Tuesday	Free	Free	Free	Free	Free	Free
Wednesday	Free	Free	Free	Free	Free	Free
Thursday	Free	Free	Free	Free	Free	Free
Friday	Free	Free	Free	Free	Free	Free
Saturday	UCSH-404	Free	Free	Free	Free	UCSH-404

Cont.

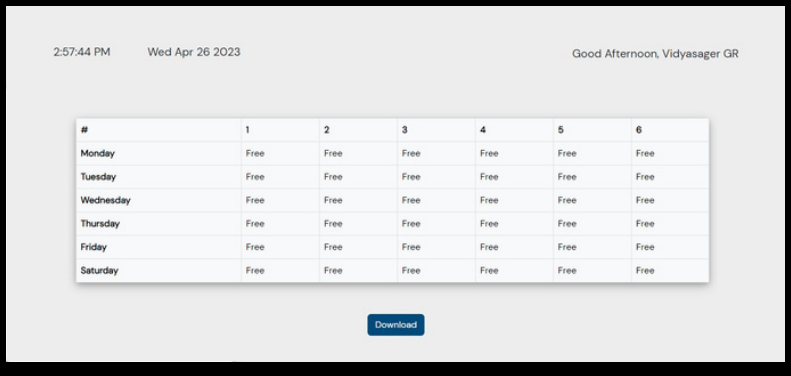


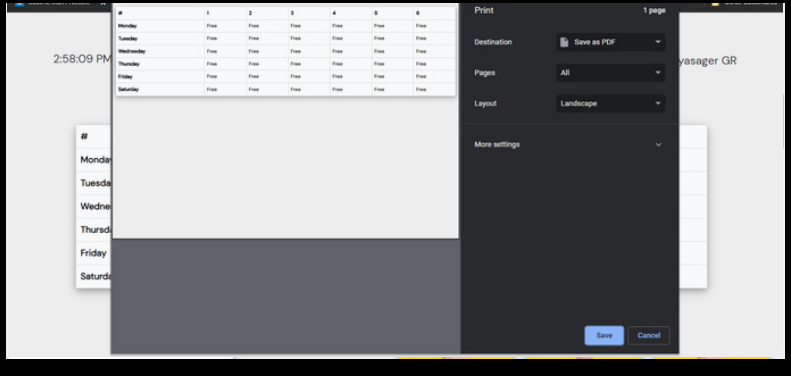


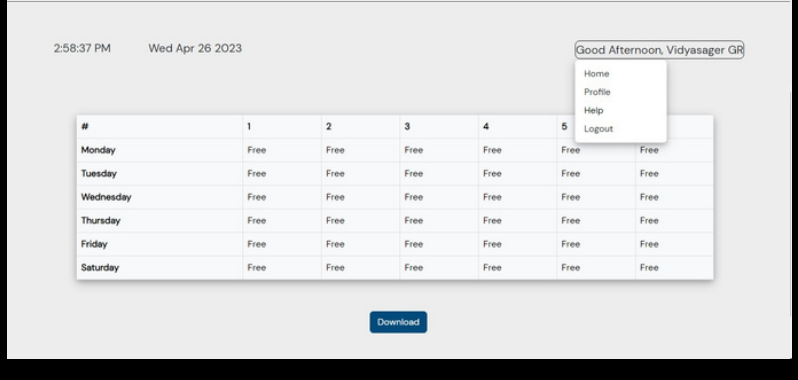
Student



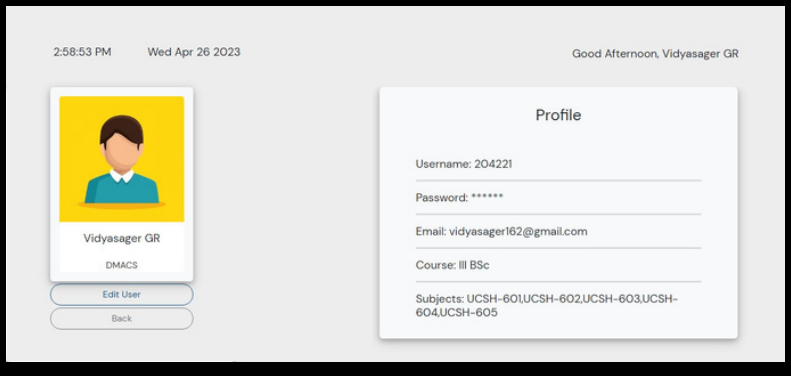
Cont.

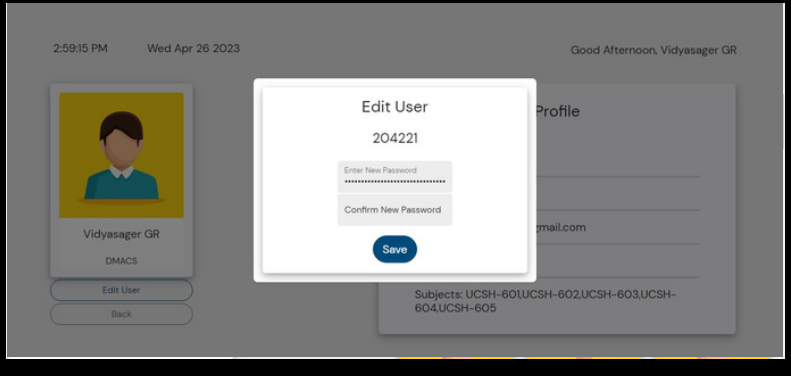

2:57:44 PM
Wed Apr 26 2023
Good Afternoon, Vidyasager GR
Download


2:58:09 PM
Print
1 page
Destination
Save as PDF
Pages
All
Layout
Landscape
More settings
Save
Cancel


2:58:37 PM
Wed Apr 26 2023
Good Afternoon, Vidyasager GR
Home
Profile
Help
Logout
Download

Cont.


2:58:53 PM
Wed Apr 26 2023
Good Afternoon, Vidyasager GR
Vidyasager GR
DMACS
Edit User
Back
Profile
Username: 204221
Password: ******
Email: vidyasager162@gmail.com
Course: III BSc
Subjects: UCSH-601,UCSH-602,UCSH-603,UCSH-604,UCSH-605


2:59:15 PM
Wed Apr 26 2023
Good Afternoon, Vidyasager GR
Vidyasager GR
DMACS
Edit User
Back
Edit User
204221
Enter New Password
Confirm New Password
Save
Profile
Subjects: UCSH-601,UCSH-602,UCSH-603,UCSH-604,UCSH-605

Future Scope

Although, the development process of this app has come to an end as all the functionalities envisioned and features envisioned in the development process and before have been implemented.
But now the deployment phase does not eradicate the development process. As the app is tested, a few features may have to be changes. As it is used, new features would be recommended by the user. Hence, we can conclude that the development phase does not have an end until the app reaches the end of its life cycle. On that note, I will take liberty to add a few things to be added and changed in the future.

- Modify the styles and bootstrap classes used to have more responsiveness in the app.
- Change the current Single Page Applicataion (SPA) architecture to MPA (Multi Page Application) using React Router.
- Implement Mail services using Google API rather than the currently used EmailJS.
- implement Form Validation for Login and all the forms for creating users.
- Implement OAuth for secure access of sensitive data replacing Bcrypt.
- Implement an attendance system to ensure if a teacher is absent, then any other teacher who is free can go as a substitute.
- Add Option to Add/Remove/Change profile pictures for the user.
- While Creating/Editing schedules for teachers, display all the other schedules of teachers who take the same course as the teacher for whom the schedule is being created.
- Add functionality for users to chat with each other (Convert this app into community type to add more usage for users other than admin).
- Add functionality to highlight the row of the current day in the timetable when being looked in the application.

Acknowledgements

I would like to first express my sincere gratitude to Bhagawan without whose consistent yet implicit support, I would not have been able to finish this project.

I also express my sincere gratitude to everyone who has supported me throughout this project. My parents for their encouraging words pushing me when I hit a snag in the project. My friends for helping me figure out the problem when I was stuck. My project supervisor P Sunil Kumar for constantly checking up on the status of the project and giving beautiful insights and suggestions and also pushing me towards the finish line by always being ever so sweet and supportive and providing me with all the resources needed for this project.

Lastly I would like to thank this institution, Sri Sathya Sai Institute of Higher Learning for providing with all the essential resources, time and support for this project. Our institution's Web Development course in the 5th semester has enabled me to pursue into the field of Web Development and learn about all the tools used and make meaningful contributions to our institution in this field.

It has been a stressful time building this Web Application, crunching deadlines and meeting the expectations of all parties involved in this project but it would be a lie if I said that I did not enjoy this. It was an amazing experience working with this project. I have learnt a lot of things regarding development. I have learnt from my mistakes that I have made in the development of this application and I intend to not make the same in future projects. I have also learnt the pros and cons of working solo in a project. I look forward to having more such experiences in this field and preferably with a team next time. I offer this project at Swami's lotus feet as an humble offering.

Sources

Many of the words and technical descriptions used in this report have been sourced from various sources

React: https://react.dev/ (All the definitions and information regarding React have been taken directly from their website and API documentation)

NPM: https://www.npmjs.com/ (All the information about the NPM packages used in this project have been taken from their respective API docs in NPMJs.com)

CORS: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing (The explanation and details about CORS have been taken from the Wikipedia page for CORS)

Bcrypt: https://en.wikipedia.org/wiki/Bcrypt (All the informations presented on the working of Bcrypt have been taken from the Wikipedia page for Bcrypt)

https://codahale.com/how-to-safely-store-a-password/ (This website has been proved very useful in knowing about all the encryption algorithms)

PapaParse: https://www.papaparse.com/ (The PapaParse official API documentation provided all the details)

EmailJS: https://www.emailjs.com/docs/examples/reactjs/