Problem

Create a website for the university which can be used by the members of the university students, TAS, Lecturers and so on.
It needed to include many many functionalities such as:

-Login and logout of the system.

-**View and update Profile.**

-**Sign in and Sign out when entering and leaving the campus.**

-View attendance records, missing/extra hours and missing/extra days.

IDEA

1- **MODEL**-Model class for each role in the university, **Student, HR, TA, ETC…..**

2- **ROUTER**- Router class for each model. The router has 3 main functionalities.

3- **SERVER**- Server class to start sending and receiving data to/from the user.

4- **FETCHER**- Fetcher classes which are responsible for the CRUD operations.

5- **MONGODB**- Connect the project to mongoDB to store all the data in.

6- **UI**- Create the UI of the website.

7- **CONNECTION**- Finally connect between the back-end and the front-end.

What I did in this project was create all the models, routers and fetchers.

Implementation

For the model Implementation, I Created a class for each model, each class contained a certain number of schemas, each schema was an object that contained variables that defined it. So for example the **Attendance Record Schema** had a date, sign In, sign Out and so on… as variables, some Schemas had classes in them and that was the implementation in a nutshell, however it was not as easy as it sounds.

For the router implementation, I created a router class for each model, the router was responsible for mainly 3 things, making sure that the user has the authority of the action he/she is committing by checking the **JAVA WEB TOKEN.** For example if the person needs to view student grades he needs to be a TA or a Lecturer. Make sure that the action the person is taking is a valid action, for example if the HR wants to delete a certain room from the system of the university but this location is currently occupied, the router will not allow this action to happen. And finally make sure that the input from the user is a valid one, let's say the lecturer wants to put more students in a lecture room than the maximum capacity, the router will deny the request and send and error to the user.

For the fetchers, I created a fetcher class for all the entities of the project which required CRUD operations, for example **Log In Fetcher** was responsible for comparing the log

In information with the data in the server, it receives the encoded password and username from the sever and compares them with the input after encoding it, and it is also responsible for updating the log In time of the user. The **HR Fetcher** had an **update staff member** functionality, the fetcher updates the server data accordingly.

.

Result

The project worked very well but it had an issue with calculating the correct salary deduction for staff that had missed working hours, that was the only issue and I still consider myself proud of this project because it was on a very very large scale and to only come out with one mistake is a big accomplishment but of course I always aim to deliver a flawless project every time.

REACT TECH

Challenges

Creating the models, it was very challenging to create the models, because I had no guidance while creating them, I needed to come up with the number of models, all the variables and classes inside them and all the data types of all the variables to make the project work properly and to deliver results that will fit in the required functionalities of each model.

Different

-I created a model for the TAs and another for the lecturers which led to a lot of issues while implementing the other classes, the right thing to do was to create one model called **ACADEMIC** and inside it we add a variable called **Job-title** or **Role**

- also I would have put more effort into finding an algorithm that returns the correct salary of all the staff of the university