Profile Builder

Submitted By -

1. Ms. Ashwini Gore
2. Ms. Nandita Dixit
3. Mr. Viplava Bharadwaj
4. Mr. Yugeshwar Devtare

Guided By –

1. Mr. Ashutosh Mishra
2. Mr. Sachin Pande

Index

1. Introduction

a. Overview

This is a project document for the **'Profile Builder'** a web application. Profile builder is a web application for **Skillship Foundation** for assigning tasks to different roles. In this portal there are roles such as Super Admin, Region Head, City Head, and Campus CEO who can assign tasks to each other and keep an eye on completion status. Also there is a leaderboard where the users are ranked according to their tasks completion. This document describes the objectives and goal of the system. This document models the functional requirements with process flow, interaction diagrams, designs, implementation, output and installation.

b. Objectives

- Four different users –
    - Superadmin
    - Region Heads
    - City Heads
    - Campus CEOs
- Superadmin can track every Region Heads, City Heads and Campus CEOs for tasks and their respective users from the dashboard.
- Each users have their own dashboard to manage tasks and their respective users.
- Superadmin can assign tasks and create different users.
- Region Heads can assign tasks and create City Heads.
- City Heads can assign tasks and create Campus CEO.
- Maintained Leader Boards according to tasks completion.
- Maintained profile page for each users.

c. Roles & Responsibility

- Frontend
    1. Ashwini Gore
    2. Yugeshwar Devtare

- Backend
    1. Nandita Dixit
    2. Viplava Bhardwaj

2. Implementation Methodology

a. Technologies

Frontend

- ReactJs
- react-router-dom
- axios
- react-validation
- Bootstrap
- validator

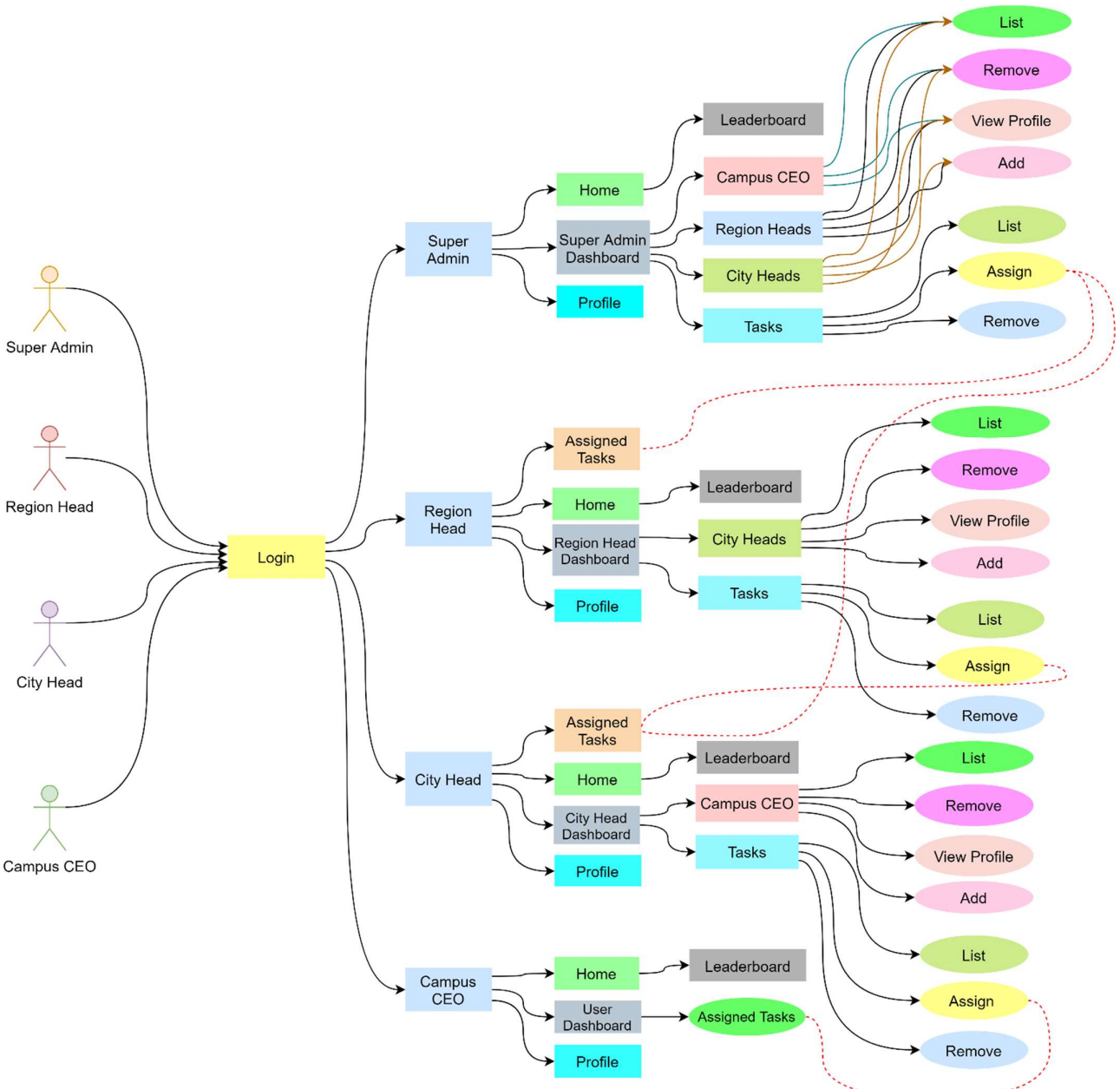
Backend

- NodeJs
- Express
- bcryptjs
- jsonwebtoken
- Sequelize

Database

- MySQL

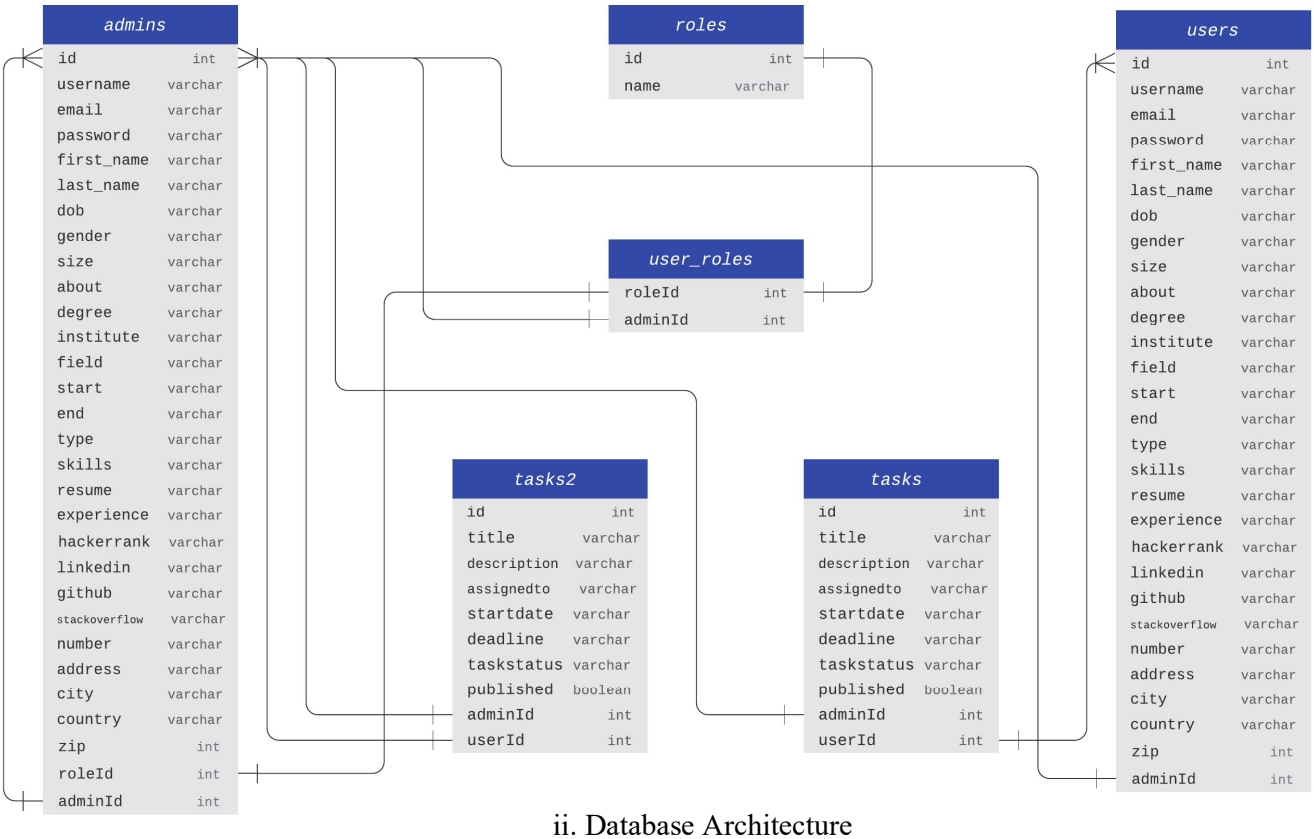
b. Process Flow



i. Process Flow

c. Database Architecture

2

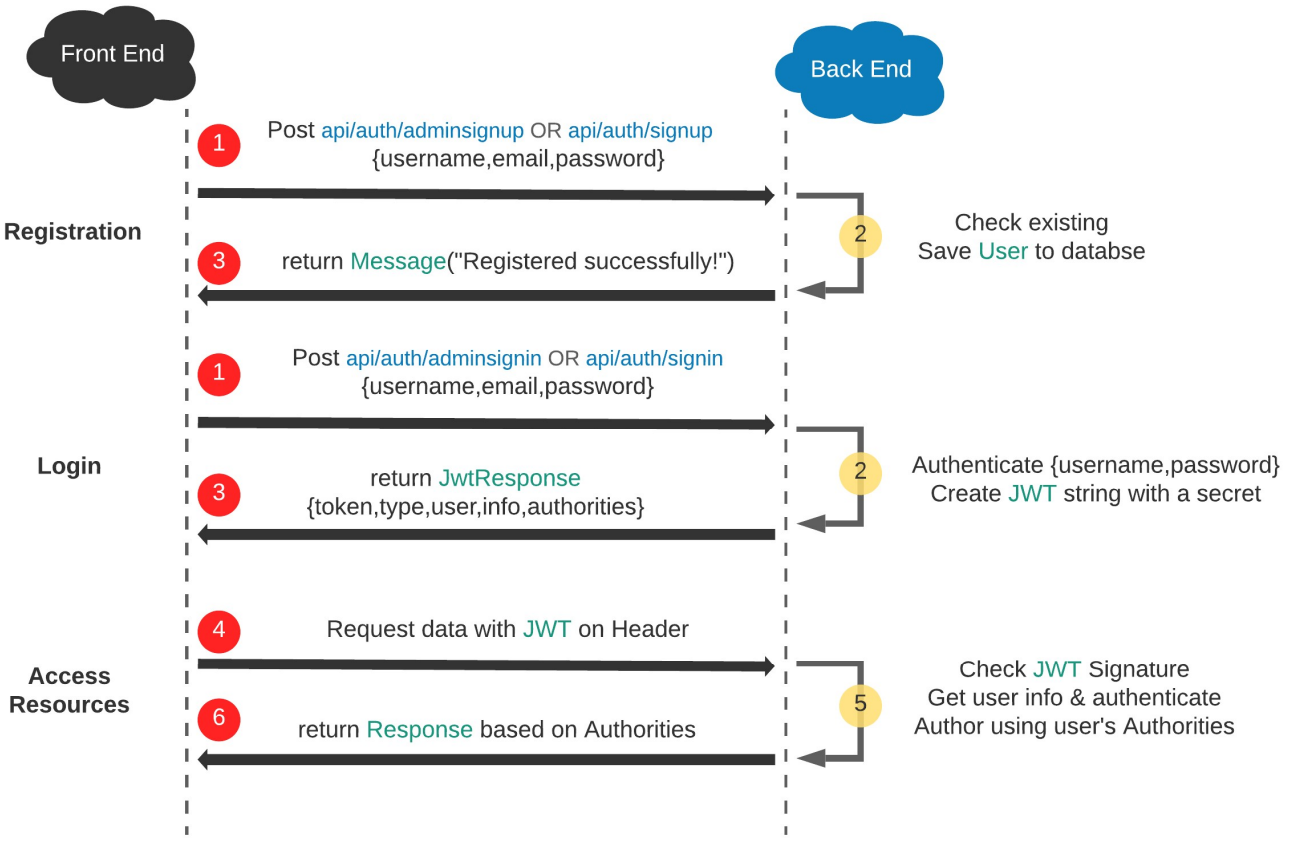


ii. Database Architecture

Created six databases in MySQL –

1. ***admins*** – Used for storing three admins
    - Superadmin
    - Region Heads
    - City Heads
2. ***users*** – Used for storing Campus CEOs.
3. ***roles*** – Used for storing roles with respective ids.
    - superadmin
    - admin (Region Heads)
    - admin2 (City Heads)
4. ***user_roles*** – Used for storing the roleId and adminId.
5. ***tasks*** – Used for storing tasks given to Campus CEOs.
6. ***tasks2*** – Used for storing tasks given to Region Heads & City Heads.

d. Flow between NodeJs and ReactJs



iii. NodeJs and ReactJs flow

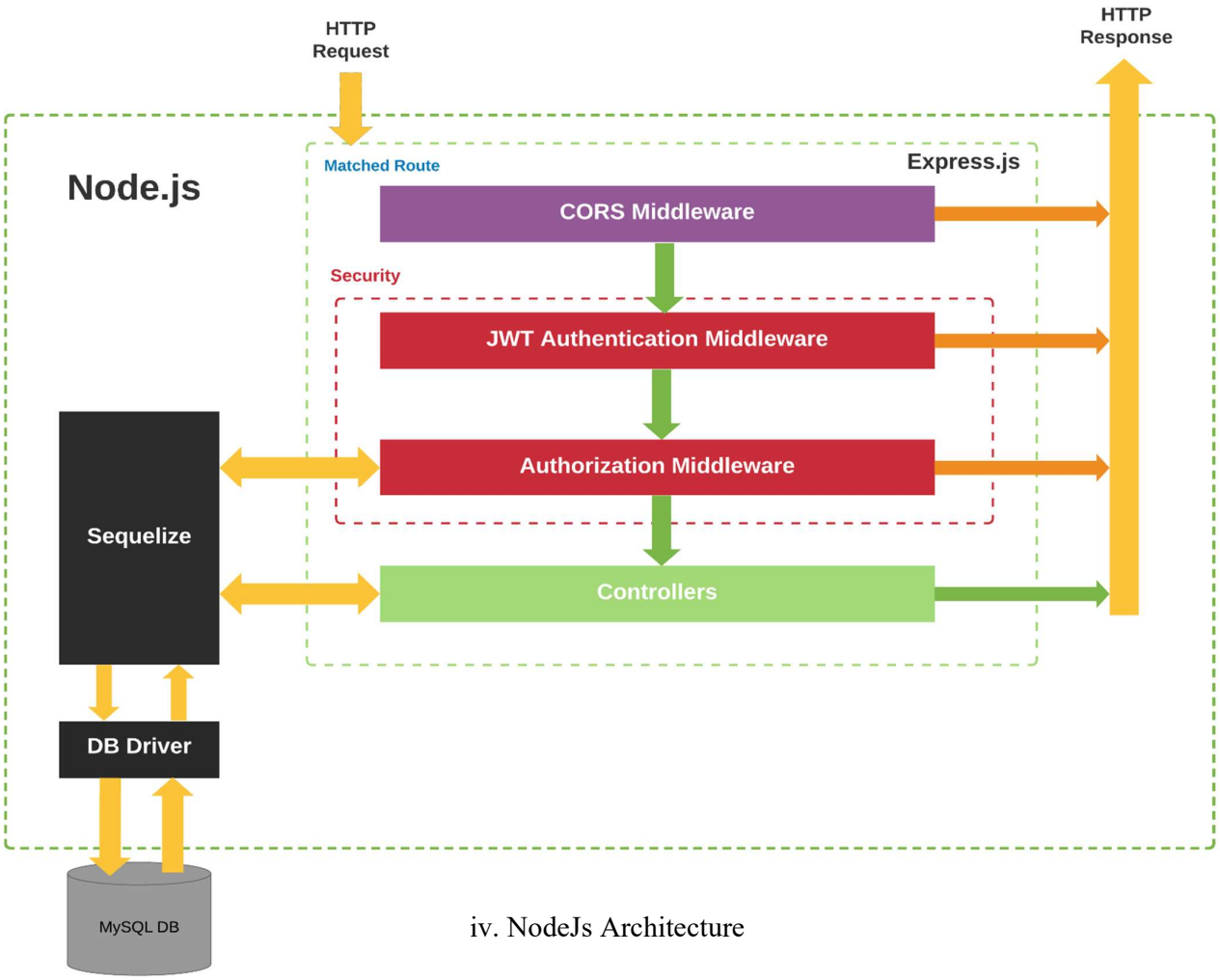
The diagram shows the flow of Registration, Login and Authorization process.

There are 2-2 endpoints for authentication admins & users:

1. Admins
   - api/auth/adminsignup for Registration
   - api/auth/adminsignin for Login
2. Users
   - api/auth/signup for Registration
   - api/auth/signin for Login

When request is send from frontend to protected data/endpoints, it will add legal JWT to HTTP **x-access-token** Header.

e. NodeJs Architecture



iv. NodeJs Architecture

With the help of Express routes, ***HTTP request*** that matches a route will be checked by ***CORS Middleware*** before coming to ***Security layer***.

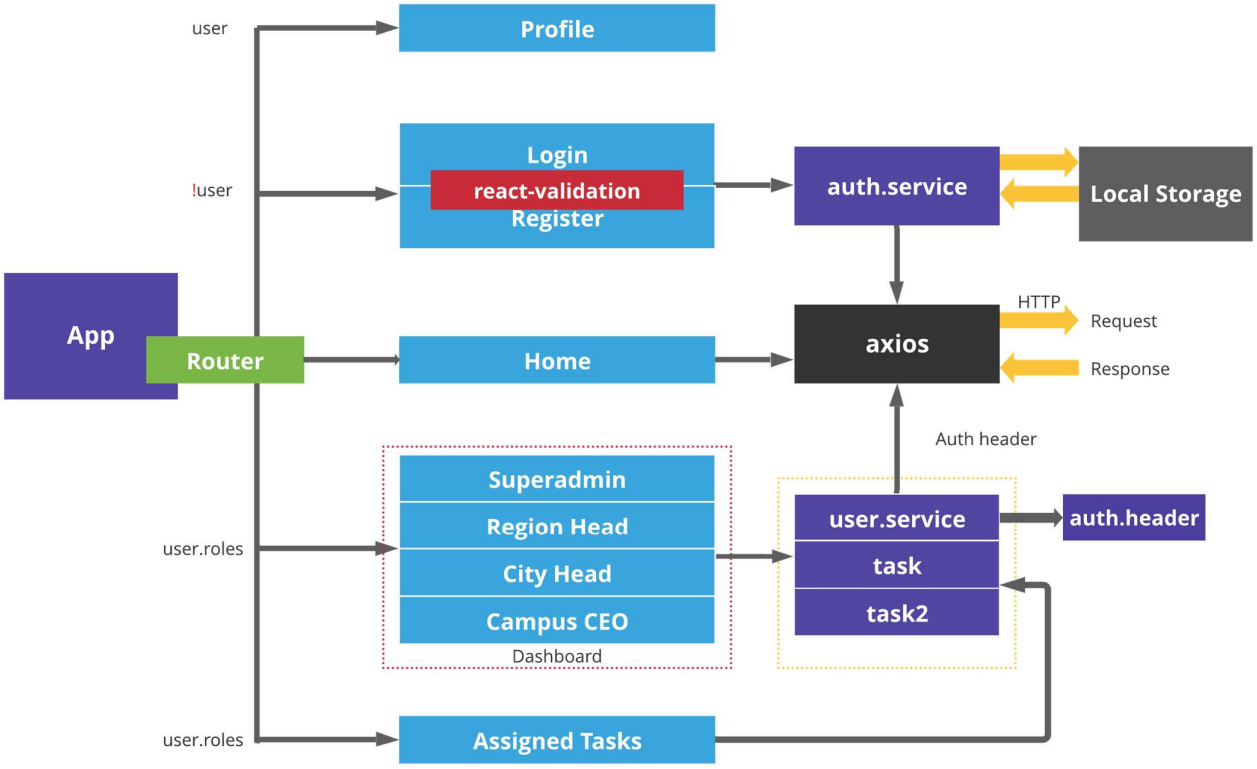
Security layer includes:

- JWT Authentication Middleware: verify SignUp, verify token
- Authorization Middleware: check Admin's roles & Users with record in database

If these ***middlewares*** throw any error, a message will be sent as ***HTTP response***.

Controllers interact with MySQL Database via ***Sequelize*** and send ***HTTP response*** (token, user information, tasks, data based on roles…) to client.

f. ReactJs Architecture



v. ReactJs Architecture

– The ***App*** component is a container with React Router (***BrowserRouter***). Based on the state, the navbar can display its items.

– ***Login*** & ***Register*** components have form for data submission (with support of ***react-validation*** library). They call methods from ***auth.service*** to make login/register request.

– ***auth.service*** methods use ***axios*** to make HTTP requests. It also store or get **JWT** from Browser Local Storage inside these methods.

– ***Home*** component is public for all visitor.

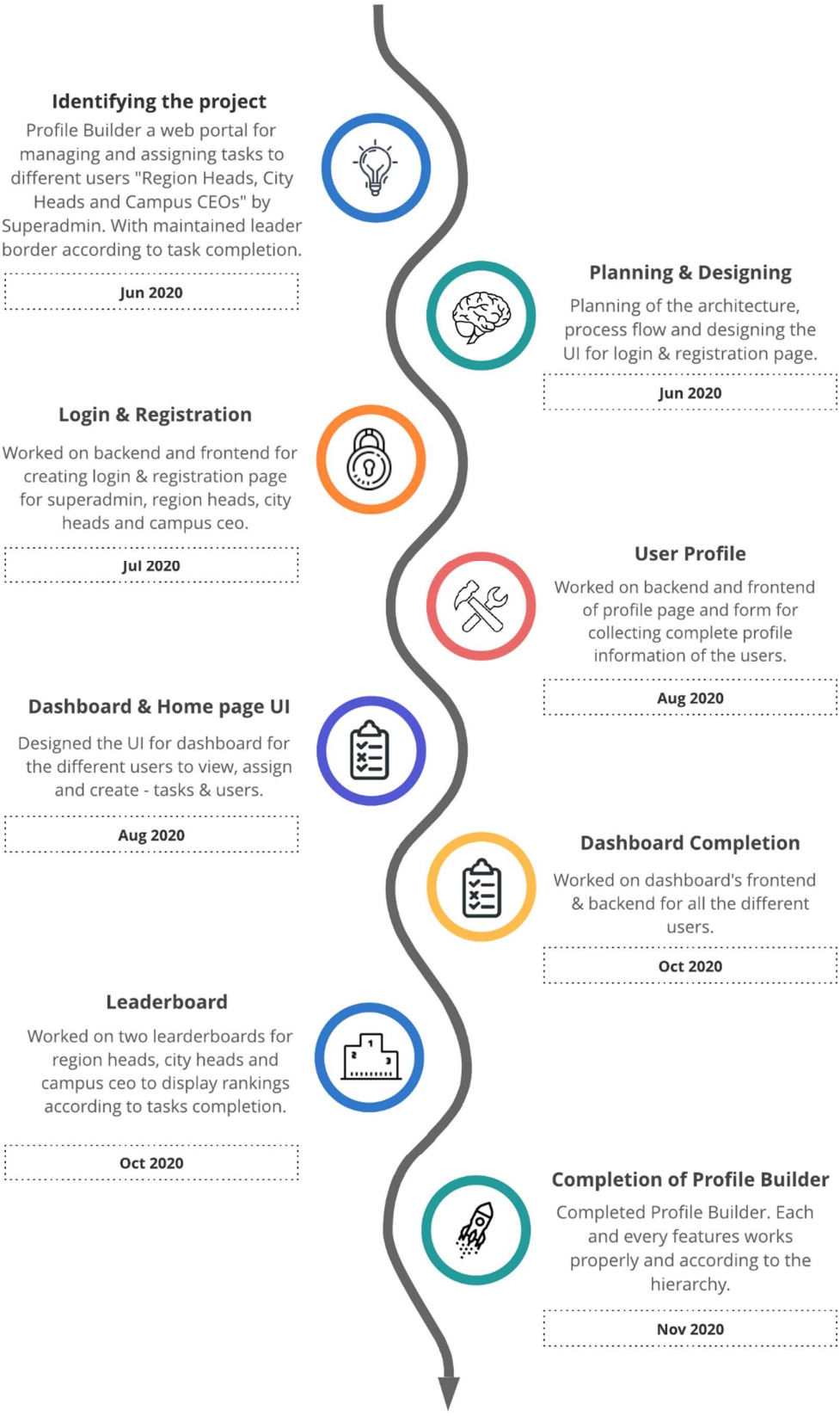
– ***Profile*** component displays user information after the login action is successful.

– ***Superadmin, Region Head, City Head & Campus CEO*** components will be displayed by state ***user.roles***. In these components, we use ***user.service, task & task2*** to access protected resources from Web API.

– ***user.service, task & task2*** uses ***auth-header()*** helper function to add JWT to HTTP header. ***auth-header()*** returns an object containing the JWT of the currently logged in user from Local Storage.

– ***Assigned Tasks*** component will be displayed by state ***user.roles***. In these components, we use ***user.service, task & task2*** to access protected resources from Web API to show assigned tasks.

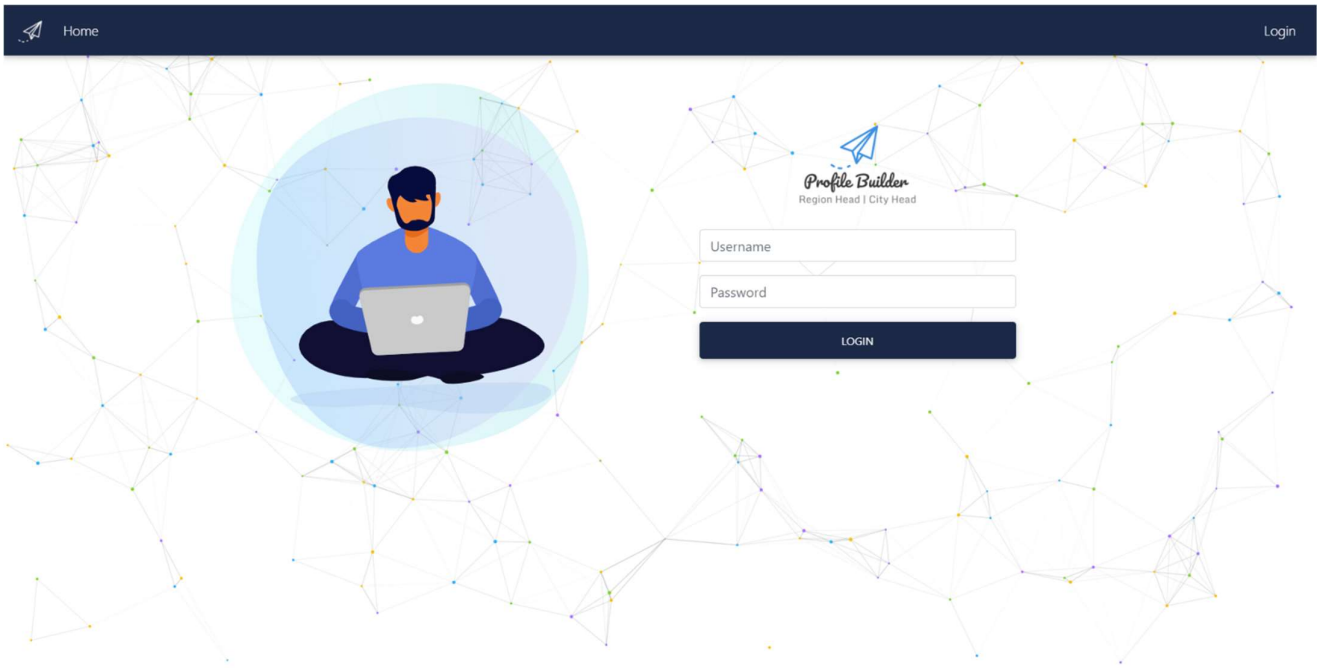
3. Timeline



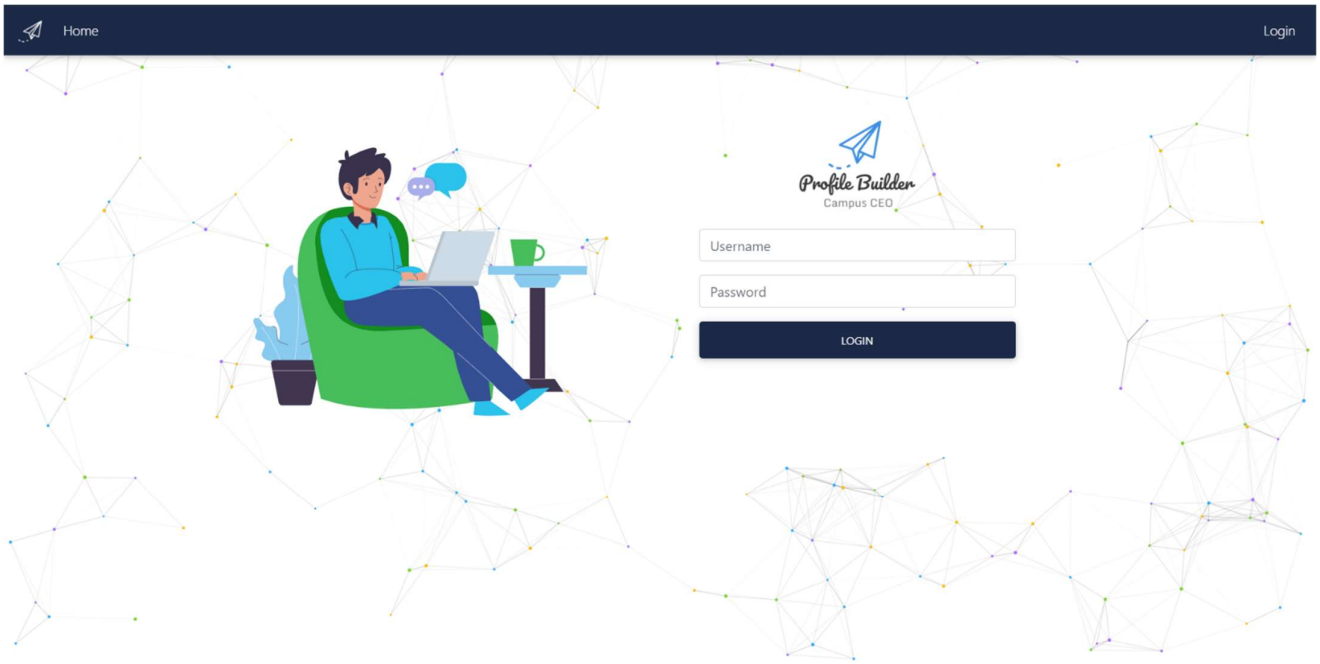
vi. Timeline

4. Output

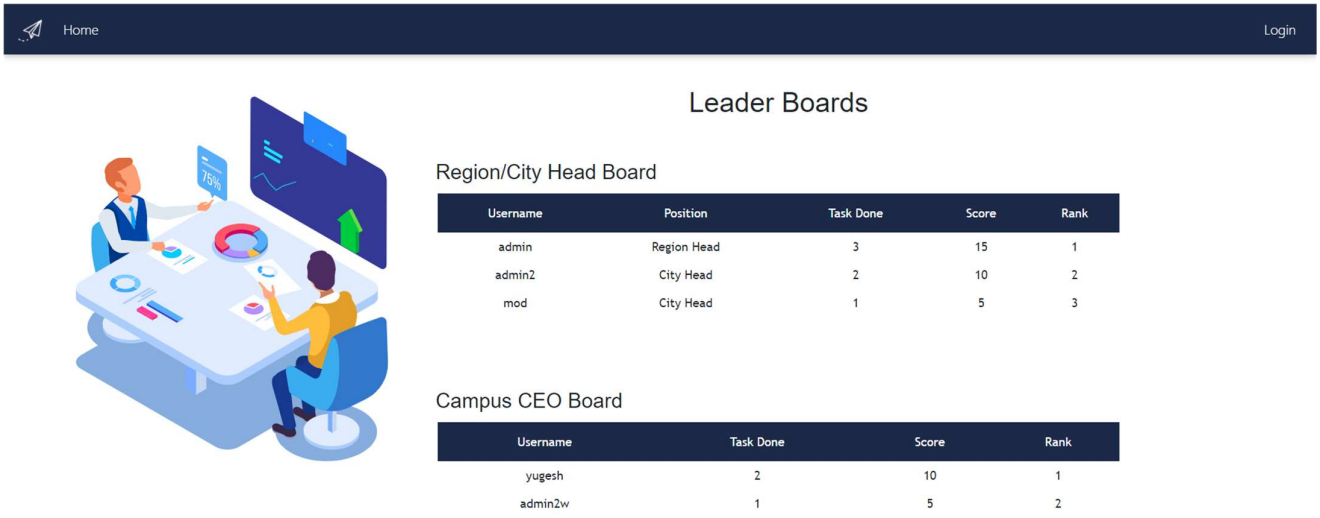
Below are some screenshots of Profile Builder.



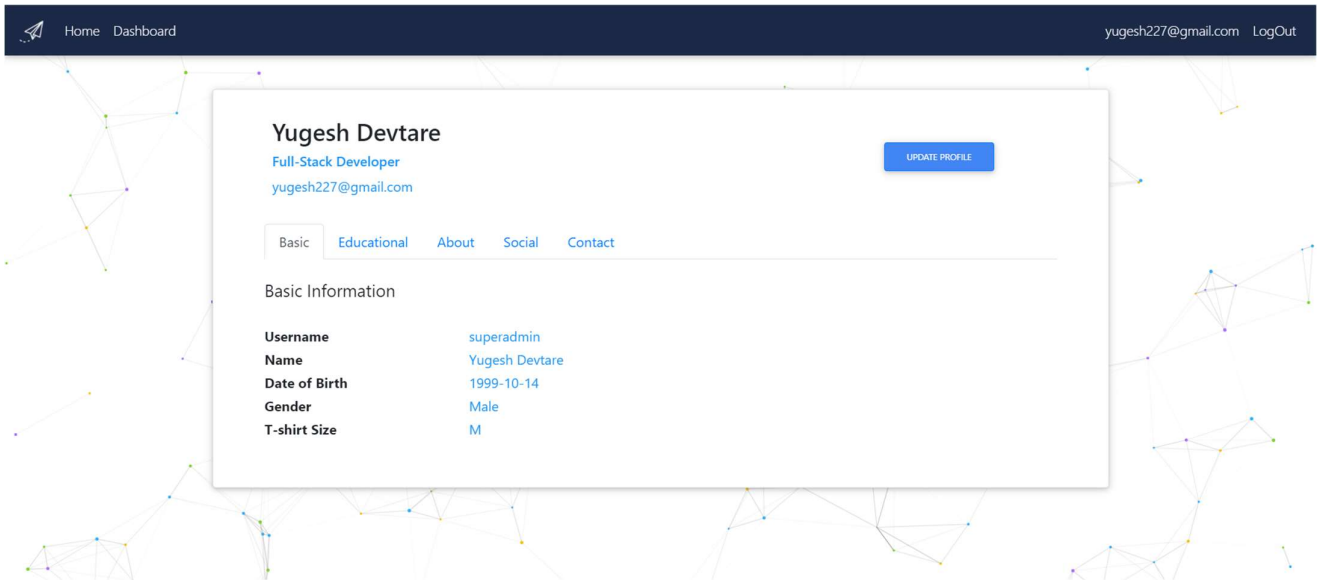
vii. Admin Login



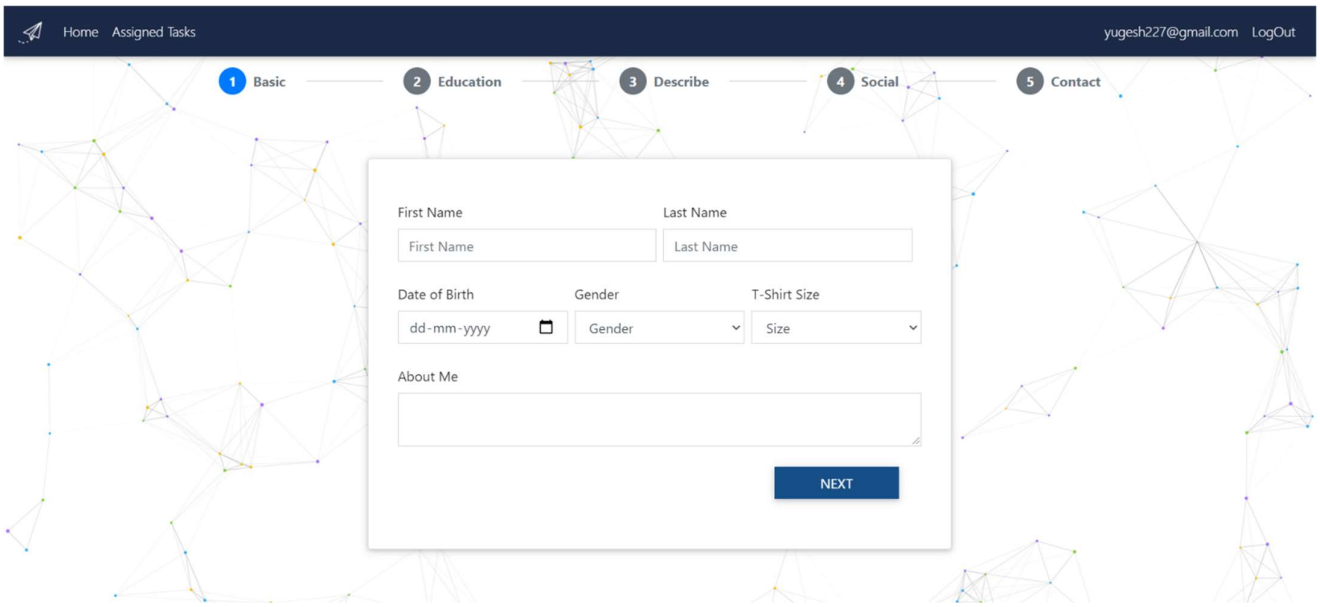
vii. Campus CEO Login



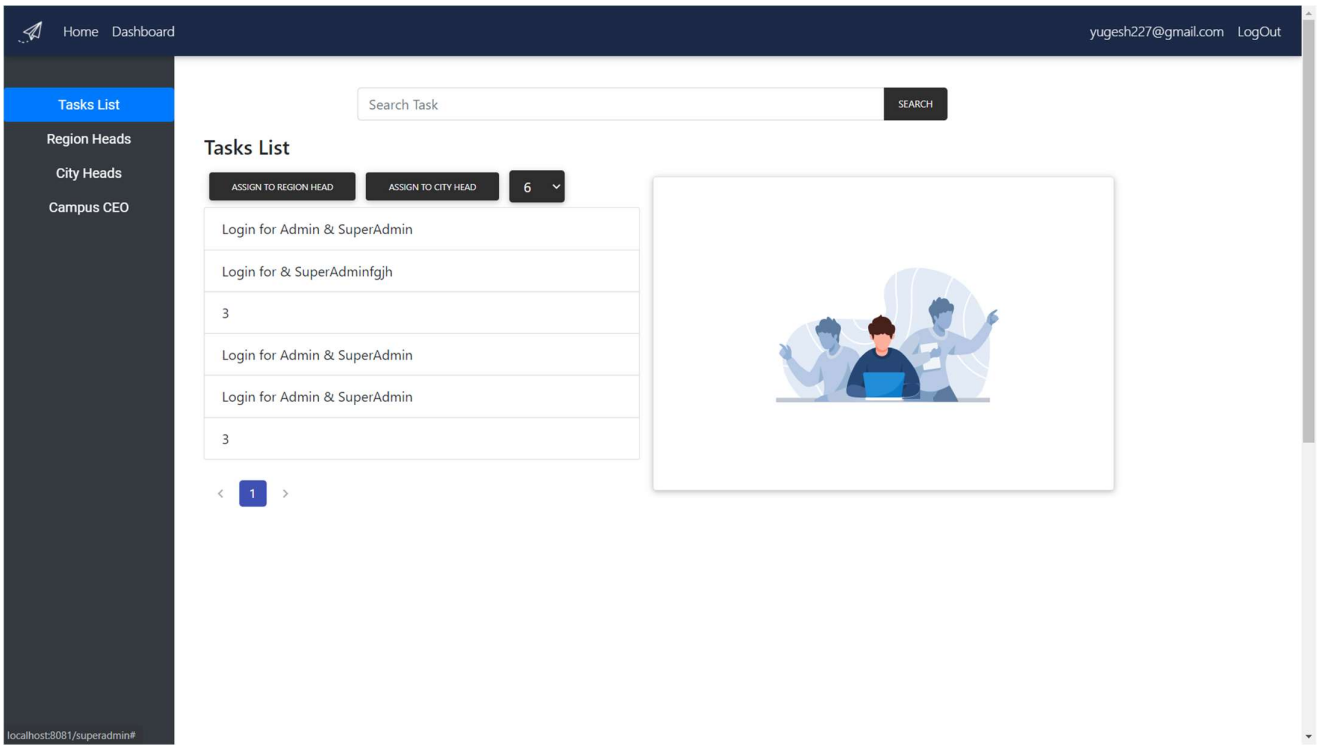
vii. Home Page



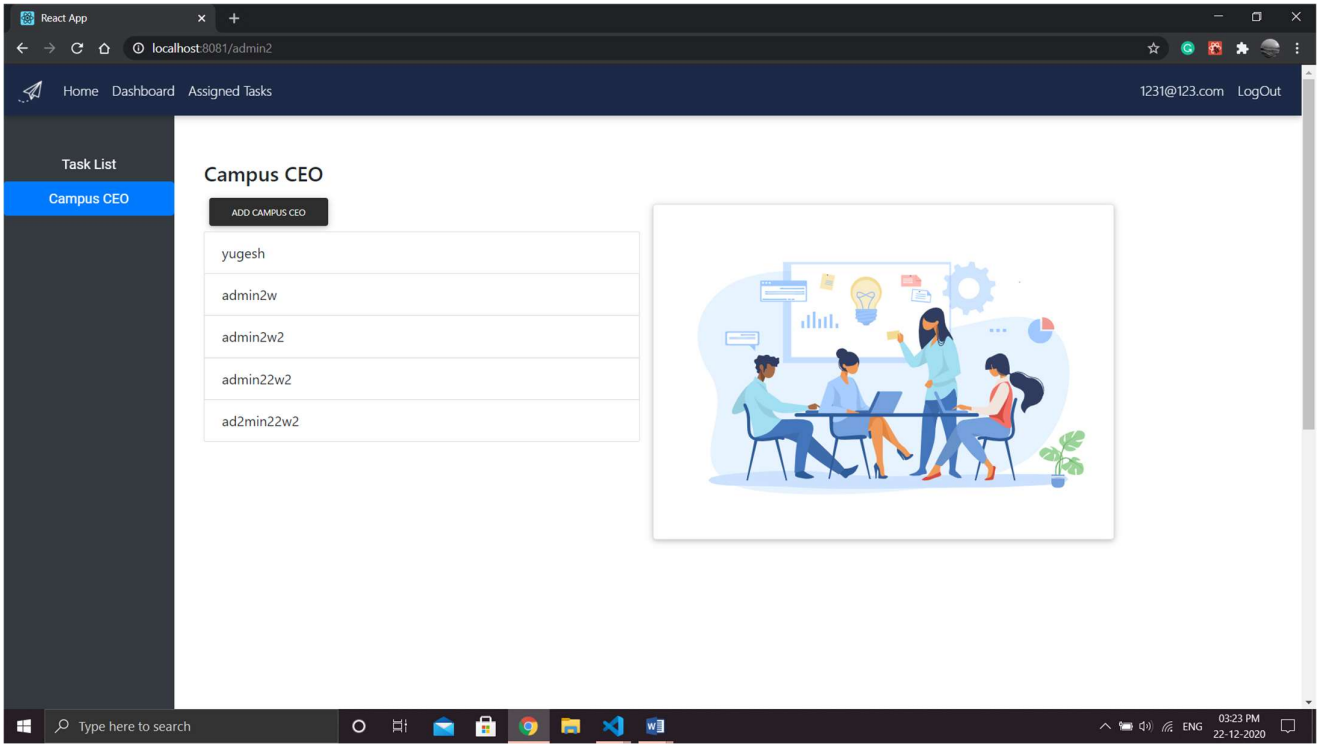
vii. Profile Page



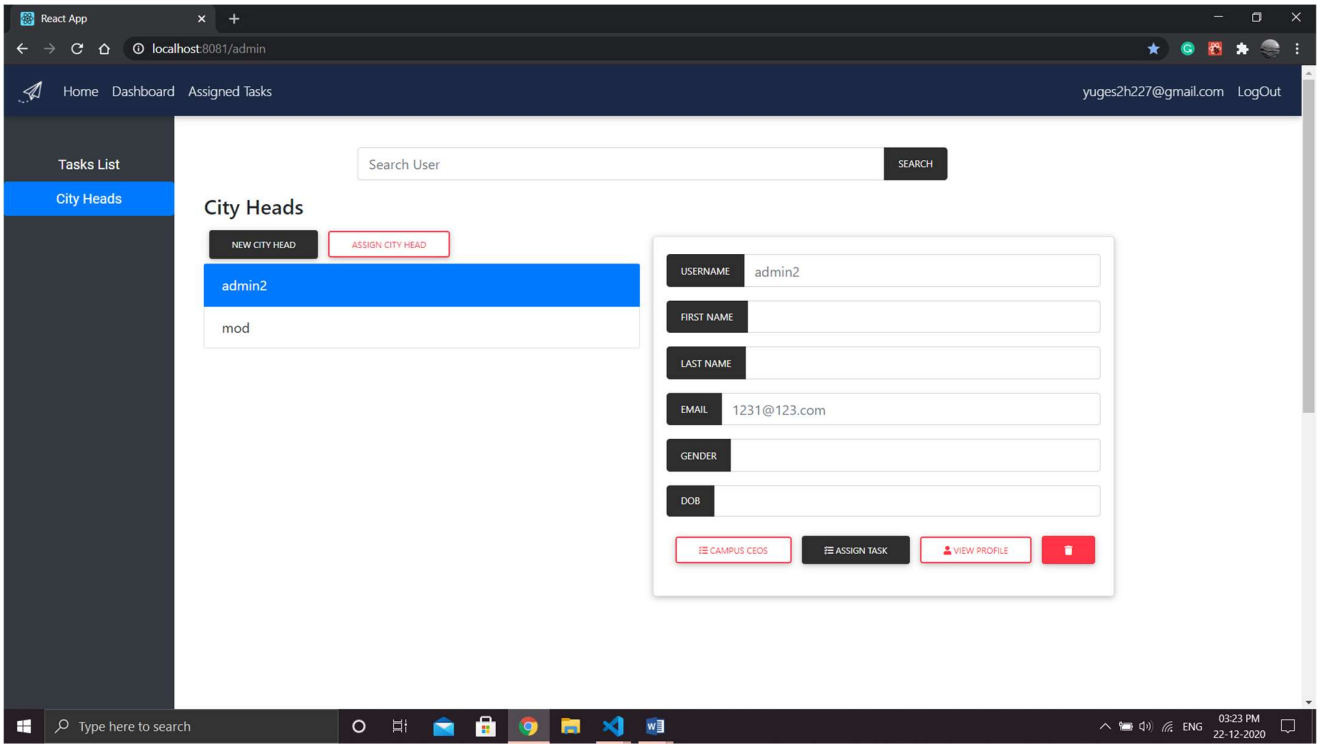
viii. Profile Form



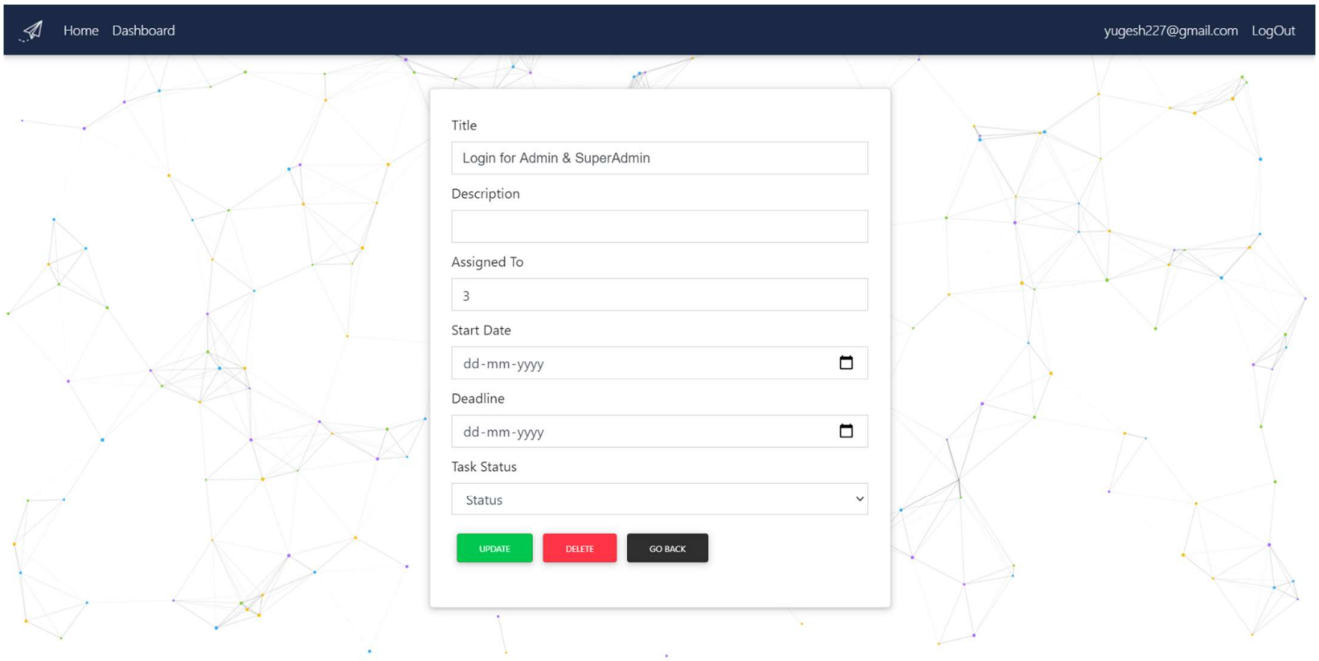
ix. Superadmin Dashboard



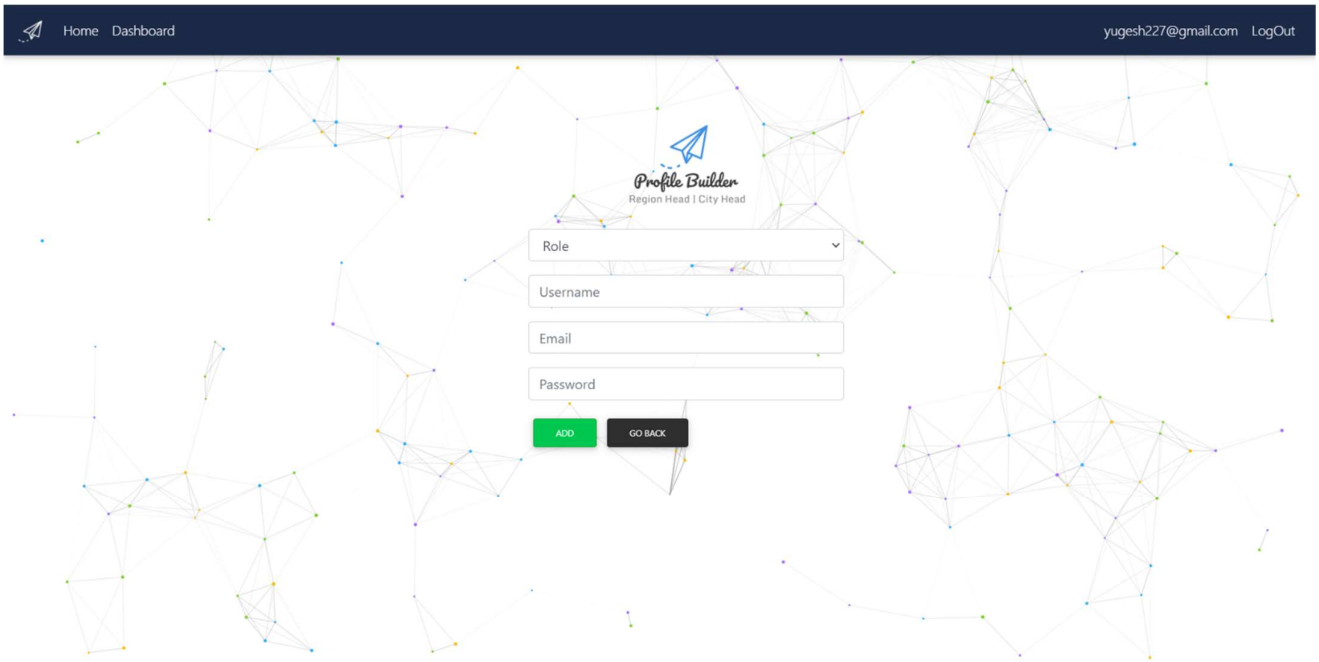
x. Region Head Dashboard



xi. City Head Dashboard



xii. Assign Tasks



xii. Create Users

5. Installation

Below is the installation process for profile builder.

1. Clone the repository and download to your system.
2. In the project directory –
   - For client (Frontend)
     - run ***npm install*** or ***yarn install*** to install client, then
     - run ***npm start*** or ***yarn start*** to start client.

     Open http://localhost:8081 to view it in the browser.

   - For server (Backend)
     - run ***npm install*** or ***yarn install*** to install client, then
     - create a database in MySQL with the name **testdb**, then
     - run ***node server.js*** to start client.
3. After successfully running the client & server, open the browser and navigate to http://localhost:8081/addsuperadmin for creating a **superadmin.** Fill up the form with a role, username, email, and a password.
4. After successfully creating a **superadmin**, you will be redirected to admin login page.
5. Log in with the username and password.
6. Now you can access all features available for **superadmin**.
7. Superadmin can now create new region heads, city heads and assign them tasks.

Some important ***urls= http://localhost:8081*** *are*

- *url+/ addsuperadmin* for creating a superadmin.
- *url+/ adminlogin* for **superadmin, region heads, city heads** login**.**
- *url+/ login* for campus ceo's login.
- *url+/ home* for home page.
- *url+/ profile* for profile page.

6. Conclusion

Now we have an overview of Profile Builder along with flow of the project. The purpose of this project was to develop a web application to assign and track tasks. Manage different set of users which are going to use the web application. The project Profile Builder has been developed as per the requirement specification.
It has been developed using ReactJs, NodeJs and MySQL. The complete system is thoroughly tested, every design procedure and diagrams are present in this project report. Designs and diagrams are easy to understand that any new modules can be incorporated easily.

7. References

i. ReactJs - https://reactjs.org/
ii. react-router-dom - https://www.npmjs.com/package/react-router-dom
iii. axios - https://www.npmjs.com/package/axios
iv. react-validation - https://www.npmjs.com/package/react-validation
v. bootstrap - https://getbootstrap.com/
vi. validator - https://validator.w3.org/
vii. nodejs - https://nodejs.org/en/
viii. sequelize - https://sequelize.org/
ix. mysql - https://www.mysql.com/
x. expressjs - https://expressjs.com/