Final Report

Team D

Yongsi Liu

Arturo Salazar

Karen Weng Liang

Guadalupe Gonzalez Alvarez

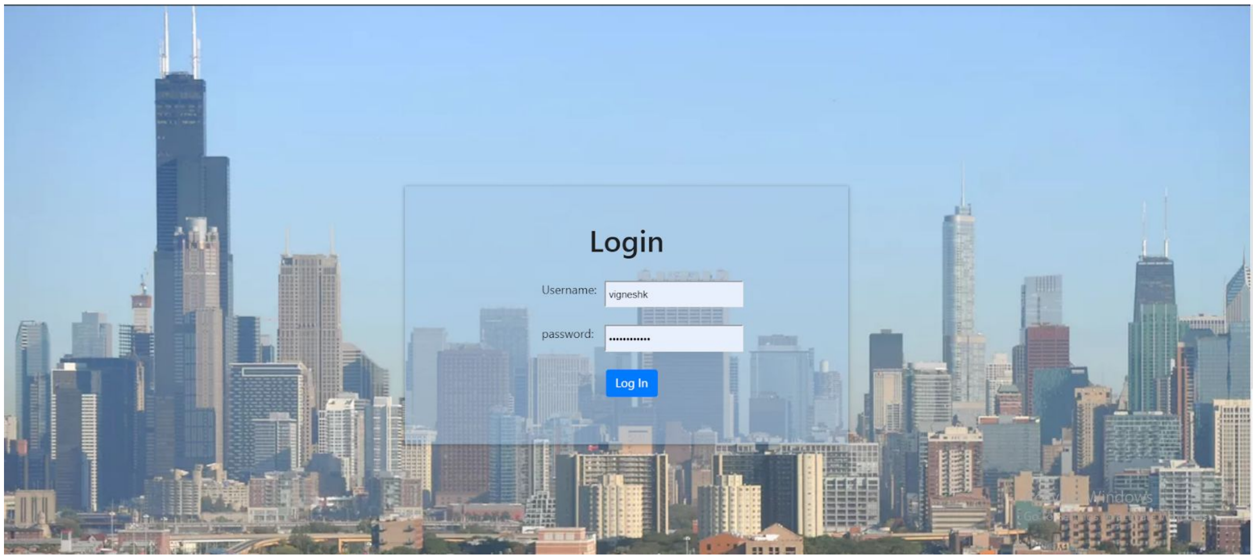
Vignesh Kumar Karthikeyan Rajalakshmi

CS-487-01

Software Engineering

November 30, 2019

1. Screenshots of prototype UI

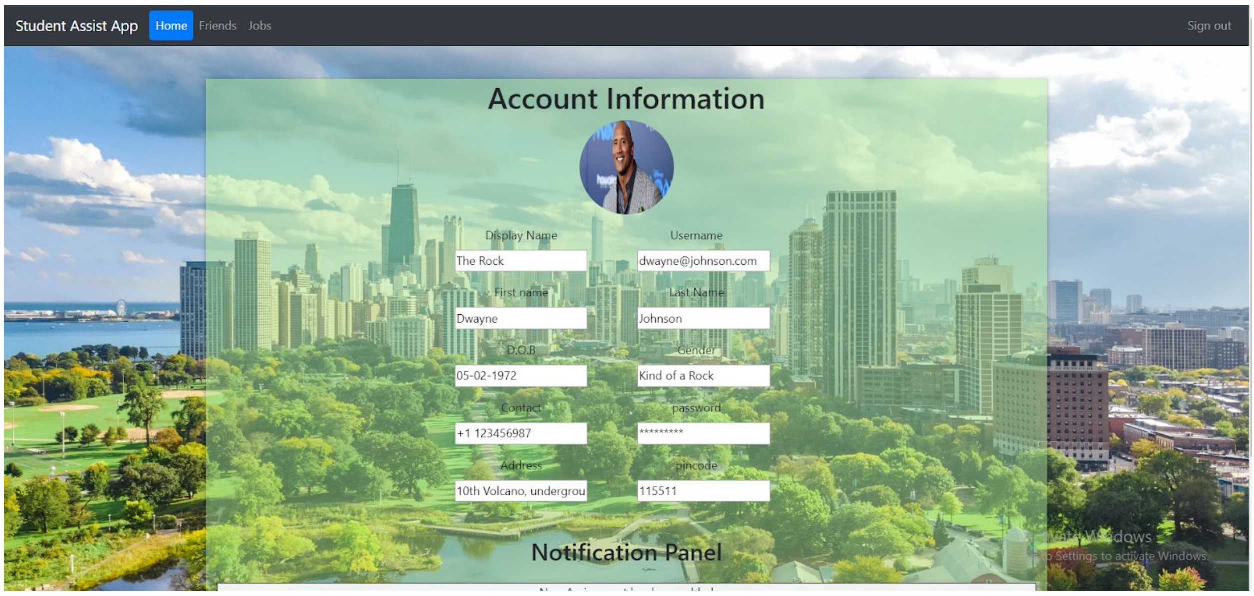


Login

Username: vigneshk

password: ••••••••••••

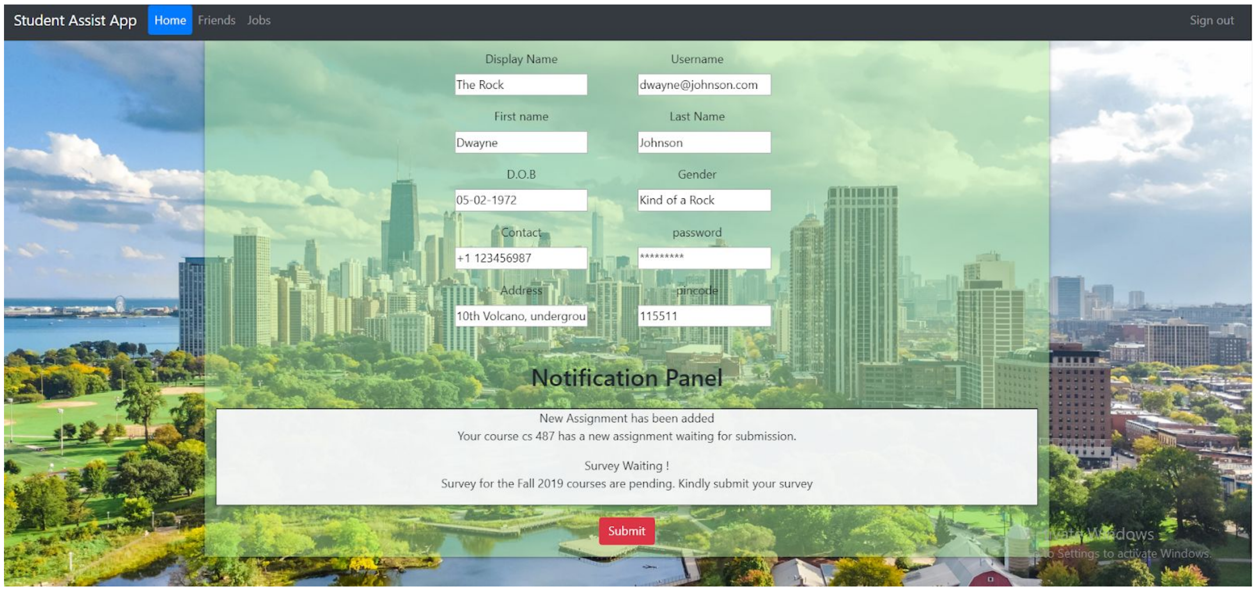
Login



Account Information

Display Name	Username
The Rock	dwayne@johnson.com
First name	Last Name
Dwayne	Johnson
D.O.B	Gender
05-02-1972	Kind of a Rock
Contact	password
+1 123456987	*********
Address	pincode
10th Volcano, undergrou	115511

Notification Panel



Student Assist App | Home | Friends | Jobs | Sign out

Display Name	Username
The Rock	dwayne@johnson.com
First name	Last Name
Dwayne	Johnson
D.O.B	Gender
05-02-1972	Kind of a Rock
Contact	password
+1 123456987	*********
Address	pincode
10th Volcano, undergrou	115511

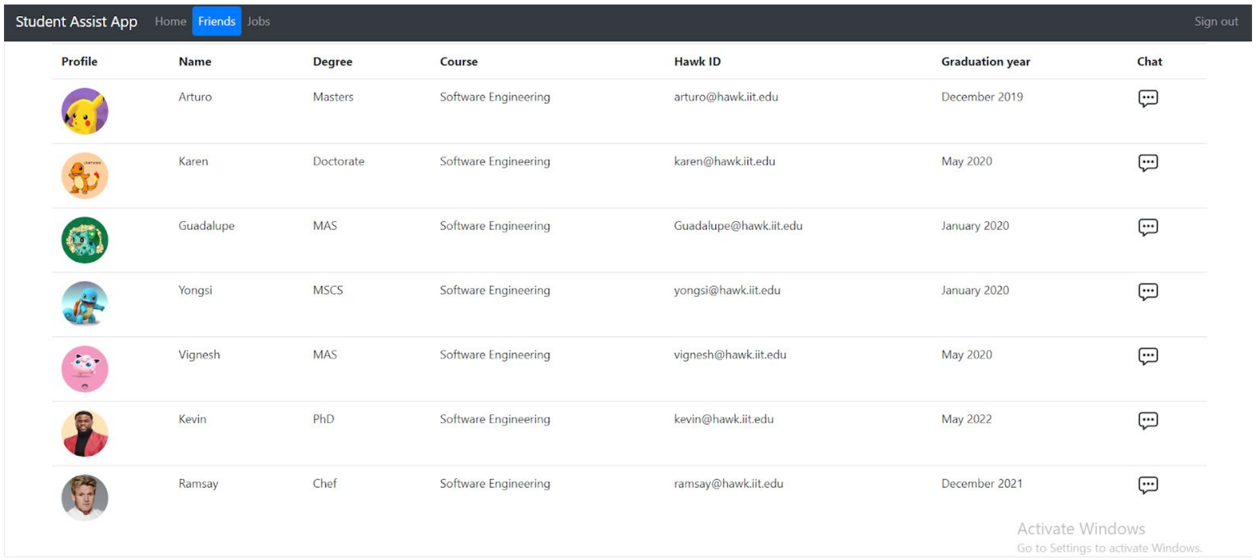
Notification Panel

New Assignment has been added
Your course cs 487 has a new assignment waiting for submission.

Survey Waiting !
Survey for the Fall 2019 courses are pending. Kindly submit your survey

Submit

Home / Sign up

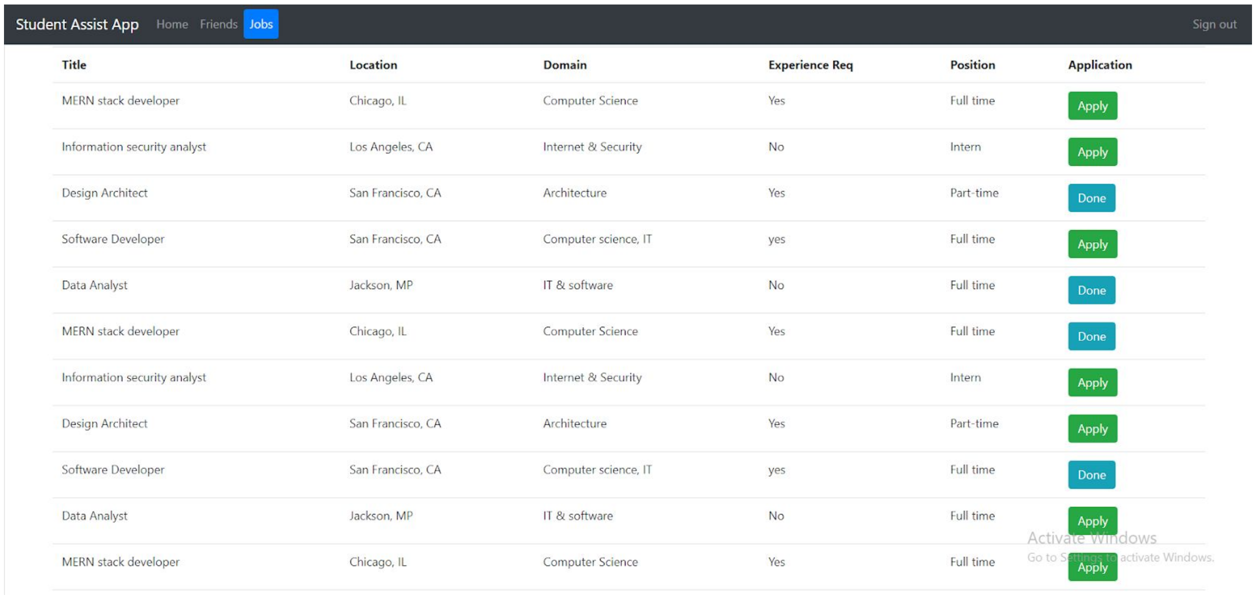


Student Assist App | Home | Friends | Jobs | Sign out

Profile	Name	Degree	Course	Hawk ID	Graduation year	Chat
	Arturo	Masters	Software Engineering	arturo@hawk.iit.edu	December 2019	
	Karen	Doctorate	Software Engineering	karen@hawk.iit.edu	May 2020	
	Guadalupe	MAS	Software Engineering	Guadalupe@hawk.iit.edu	January 2020	
	Yongsi	MSCS	Software Engineering	yongsi@hawk.iit.edu	January 2020	
	Vignesh	MAS	Software Engineering	vignesh@hawk.iit.edu	May 2020	
	Kevin	PhD	Software Engineering	kevin@hawk.iit.edu	May 2022	
	Ramsay	Chef	Software Engineering	ramsay@hawk.iit.edu	December 2021	

Activate Windows
Go to Settings to activate Windows.

Friends / Contact List



Title	Location	Domain	Experience Req	Position	Application
MERN stack developer	Chicago, IL	Computer Science	Yes	Full time	Apply
Information security analyst	Los Angeles, CA	Internet & Security	No	Intern	Apply
Design Architect	San Francisco, CA	Architecture	Yes	Part-time	Done
Software Developer	San Francisco, CA	Computer science, IT	yes	Full time	Apply
Data Analyst	Jackson, MP	IT & software	No	Full time	Done
MERN stack developer	Chicago, IL	Computer Science	Yes	Full time	Done
Information security analyst	Los Angeles, CA	Internet & Security	No	Intern	Apply
Design Architect	San Francisco, CA	Architecture	Yes	Part-time	Apply
Software Developer	San Francisco, CA	Computer science, IT	yes	Full time	Done
Data Analyst	Jackson, MP	IT & software	No	Full time	Apply
MERN stack developer	Chicago, IL	Computer Science	Yes	Full time	Apply

Jobs

2. Prototype source code

App.js

```
import React, { Component } from 'react';
import { BrowserRouter, Route, Switch } from 'react-router-dom';
import { Navbar, Nav} from 'react-bootstrap';
import Login from './components/Login'
import Home from './components/Home';
import Friends from './components/Friends';
import Jobs from './components/Jobs';
import Error from './components/Error';
import 'bootstrap/dist/css/bootstrap.css';
import { withRouter } from "react-router";
```

```
const Header = props => {
  const { location } = props;
  return (
    <Navbar className="bg-dark " variant="dark" expand="lg" fixed="top">
    <Navbar.Brand href="">Student Assist App</Navbar.Brand>
    <Navbar.Toggle aria-controls="basic-navbar-nav" />
    <Navbar.Collapse id="basic-navbar-nav">
      <Nav justify variant="pills" activeKey={location.pathname}
className="mr-auto">
        <Nav.Link href="/home">Home</Nav.Link>
        <Nav.Link href="/friends">Friends</Nav.Link>
        <Nav.Link href="/job">Jobs</Nav.Link>
      </Nav>
      <Nav>
        <Nav.Link href="/login">Sign out</Nav.Link>
      </Nav>
    </Navbar.Collapse>
  </Navbar>
  );
};
const HeaderWithRouter = withRouter(Header);

class App extends Component {
  render() {
    return (
       <BrowserRouter>
        <div>
        <HeaderWithRouter />
            <Switch>
             <Route path="/login" component={Login} exact/>
             <Route path="/home" component={Home} />
             <Route path="/friends" component={Friends}/>
             <Route path="/job" component={Jobs}/>
            <Route component={Error}/>
           </Switch>
        </div>
      </BrowserRouter>
    );
  }
}

export default App;
```

login.js

```
import React from 'react';
import { Link } from 'react-router-dom';
import {Button, Row,Col} from 'react-bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';

const login = () => {
    return (

       <div className="Login-component">
           <img
src={"https://cdn.vox-cdn.com/thumbor/AhMT8thvxo5vMferw_wf8Hg4NMM=/0x0:3912x2388/1200x800/filters:focal(1644x882:2268x1506)/cdn.vox-cdn.com/uploads/chorus_image/image/65516135/COMED_10XX19_2.0.jpg"} alt="bg"
class="bg"></img>
          <div className="log1 container align-items-center">
            <Row className="justify-content-md-center
pt-5"><h1>Login</h1></Row>
            <div style={{marginRight:90}}>
                <Row className="justify-content-md-center mt-4">
                <Col lg='2'><label className>Username:</label></Col>
                <Col lg='2'><input type="text" className="p-1"/></Col>
                </Row>
                <Row className="justify-content-md-center mt-4">
                <Col lg='2'><label>password:</label></Col>
                <Col lg='2'><input type="password"className="p-1"/></Col>
                </Row>
                <Row className="justify-content-md-center mt-4"
style={{marginLeft:90}}>
```

```
                <Col sm lg='3'><Link to="/home"><Button
variant="primary">Log In</Button></Link></Col>
                </Row>
            </div>
        </div>
        </div>
    );
}

export default login;
```

Error.js

```
import React from 'react';

const Error = () => {
    return (
        <div>
            <p>Error: Page does not exist!</p>
        </div>
    );
}

export default Error;
```

Home.js

```

import React, { Component } from 'react';
import { Grid, Header, Form,  Message, Input, Segment, Select } from
'semantic-ui-react';
import { Button, Row,Col} from 'react-bootstrap';
import { Auth } from 'aws-amplify';
import Amplify from 'aws-amplify';

const loadImageError = (e)=> {
    e.target.src = 'https://via.placeholder.com/40';
}

const options = [
  { key: 'm', text: 'Male', value: 'male' },
  { key: 'f', text: 'Female', value: 'female' },
  { key: 'o', text: 'Other', value: 'other' },
]

class MyAccount extends Component {

    constructor(props){
        super(props);
        this.state = {
            authState: this.props.authState,
            nickname:'',
            email:'',
            given_name:'',
            middle_name:'',
            family_name:'',
            birthdate:'',
            gender:'',
            phone_number:'',
            address:'',
            website:'',
        }
    }


    handleChange = (e, { name, value }) => {
        this.setState({ [name]: value });
    }


    render(){

        let loading = true;
        const
{nickname,email,given_name,middle_name,family_name,birthdate,gender,phone_
number,address,website} = this.state;

        return(

               <div className="log2 container">
             <img
src={"https://s3-prod.chicagobusiness.com/GettyImages-1057157166.jpg"}
alt="bg" class="bg"></img>

               <div style={{textAlign:"center"}}>
                <Header as="h1">Account Information</Header>
                <div className="nav-profile2"><img
src={"https://www.maxim.com/.image/t_share/MTU3ODY0MzkxNDY1NzcyMzYx/dwayne
-the-rock-johnson-promo.jpg"} alt="profile"
onError={loadImageError}/></div>

              <Form onSubmit={this.handleSubmit} loading={loading} >
            <Row className="mt-3">

                <Col md={{ span: 3, offset: 3 }}>  <Form.Input
name='nickname' value={"The Rock"} label='Display Name'
placeholder='Display name' width={6} onChange={this.handleChange}
error={false} /></Col>
                <Col lg="2"> <Form.Input label='Username'
value={"dwayne@johnson.com"} width={10}  /></Col>

            </Row>
                <Row className="mt-3">
                  <Col md={{ span: 3, offset: 3 }}><Form.Input
name='given_name' value={"Dwayne"} label='First name' placeholder='First
Name' width={6} onChange={this.handleChange} error={false} /></Col>
```

```
                <Col lg="2"><Form.Input name='family_name'
value={"Johnson"} label='Last Name' placeholder='Last Name' width={6}
onChange={this.handleChange} error={false} /></Col>
                </Row>

                <Row className="mt-3">

                <Col md={{ span: 3, offset: 3 }}>  <Form.Input
name='nickname' value={"05-02-1972"} label='D.O.B'
placeholder='02/11/1992' width={6} onChange={this.handleChange}
error={false} /></Col>
                <Col lg="2"> <Form.Input label='Gender' value={"Kind of a
Rock"} width={10}  /></Col>

                 </Row>

                 <Row className="mt-3">

               <Col md={{ span: 3, offset: 3 }}>  <Form.Input
name='Contact' value={"+1 123456987"} label='Contact' placeholder='+1
5584323248' width={6} onChange={this.handleChange} error={false} /></Col>
               <Col lg="2"> <Form.Input label='password'
placeholder='*********' value={"*********"} width={10}  /></Col>

                </Row>

                <Row className="mt-3">
                <Col md={{ span: 3, offset: 3 }}> <Form.Input
name='address' value={"10th Volcano, underground gym"} label='Address'
placeholder='2/77 New Street, Newport 3015 Melbourne, Victoria, Australia
' width={20} height={30} onChange={this.handleChange} error={false}
/></Col >
                <Col lg="2"> <Form.Input label='pincode' value={"115511"}
width={10}  /></Col>
                </Row>
                <h2 className="justify-content-center mt-5">Notification
Panel</h2>
```

```
                <div className="log3 mt-4">
                <Message
                  success
                  header='New Assignment has been added'
                  content="Your course cs 487 has a  new assignment
waiting for submission."
                />

                <Message
                  error
                  header='Survey Waiting ! '
                  content='Survey for the Fall 2019 courses are pending.
Kindly submit your survey '
                />
                </div>
                <Button variant="danger" className="mt-3 mb-3"
type='submit'>Submit</Button>
              </Form>
              </div>
              </div>



        );
    }

}

export default MyAccount;
```

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
```

```
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed
on a
      user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
    -->
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <!--
      Notice the use of %PUBLIC_URL% in the tags above.
      It will be replaced with the URL of the `public` folder during the
build.
      Only files inside the `public` folder can be referenced from the
HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico"
will
      work correctly both with client-side routing and a non-root public
URL.
      Learn how to configure a non-root public URL by running `npm run
build`.
    -->
    <title>Student Assist App</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.
      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.
      To create a production bundle, use `npm run build` or `yarn build`.
    -->
  </body>
</html>
```

Index.css

```
body {
  margin: auto;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto",
"Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
body, html {
  height: 100%;
  margin: 0;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
    monospace;
}

.log1 {
  margin-top: 300px;
  background: rgba(162, 199, 231, 0.473);
  max-width: 600px !important;
  box-shadow: 0 1px 6px rgba(0, 0, 0, 0.12), 0 1px 4px rgba(0, 0, 0,
0.24);
  height:350px;
}

.log2 {
  margin-top: 100px;
  margin-bottom: 50px;
  background: rgba(174, 231, 151, 0.473);
  box-shadow: 0 1px 6px rgba(0, 0, 0, 0.12), 0 1px 4px rgba(0, 0, 0,
0.24);
}

.log3 {
  margin-top: 100px;
  border: 1px solid ;
  background: rgba(246, 247, 248, 0.959);
  box-shadow: 0 1px 6px rgba(0, 0, 0, 0.12), 0 1px 4px rgba(0, 0, 0,
0.24);
}

img.bg {
  /* Set rules to fill background */
  min-height: 100%;
  min-width: 1024px;

  /* Set up proportionate scaling */
  width: 100%;
  height: auto;

  /* Set up positioning */
  position: fixed;
  top: 0;
  left: 0;
  z-index: -1;
}
```

```css
.nav-profile img {
  width:4rem;
  height: 4rem;
  border-radius: 50%;
}

.nav-profile2 img {
  width:8rem;
  height: 8rem;
  border-radius: 50%;
}
```

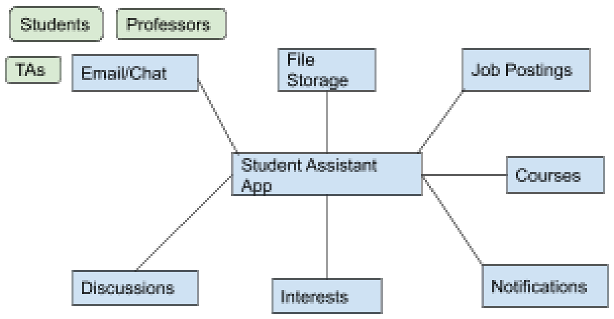
3. Final analysis

One of the main goals of creating an analysis report was to be able to understand as much as possible the needs and experiences that our potential users could have with our app before blindly jumping ahead on developing the application. The early planning and sketching of our application played a fundamental role throughout the process. Utilizing our own experience as students as well as creating a survey on what students experience when trying to study and collaborate helped us determine the types of functionality that we could provide. This also helped us identify useful categories that we might have missed (namely, remote versus on campus students) that allowed us to keep in mind the way we would need to implement this system to make it easy for all students to use. This analysis also allowed us to focus on the main functionalities that would provide the most use for our users without adding additional features that may not be feasible. Through this exercise, we also determined possible exceptions we could end up encountering and how to detect and handle them.

In this deliverable, the application also takes into consideration nonfunctional requirements like password protection for user login and signup, ease of use, performance time of 10 seconds at most, portability, maintainability for technical support and also reliability.

4. Design information

As promised, our application strives to deliver a user-friendly interface that allows the user to easily access and navigate around the app. We have designed the application with professors, teaching assistants (TAs), and especially students as our target audience.



Based on the UI Sketches and context model description that we have designed, we decided on achieving this by making our application come to life using React. We also utilized an object oriented design in order to implement our thoughts and different stages of the application.

Our application currently allows users to register an account, sign in, display job postings and friends in the contact list. There are still some proposed functionalities that need to be created and worked on, but for the purposes of this report and complexity they will not be included with this deliverable. Such functionalities are: chat between friends users, notification center and pairing based on course information. These will be later considered if our team decides on moving on with the Student Assist App idea in the future.

In the first version of the application, we have successfully achieved our main purpose of creating an application that allows students to meet other students and find tutors to assist them with either assignments or class related questions. The application accepts user input and will know how to proceed based on the input received. If input is valid, the application will allow the user to continue to the main page, but if it does not, it will console out an error message so the user can understand what has occurred without crashing the program. Another important required function present in our application is the ability to post new jobs so that our Student Assist App not only helps students academically, but also offers an extension of career growth and opportunities which is something that as students, we think it adds more value to the app.

5. Discussion of testing

a. **Testing strategy:** We focus on testing different situations when students use the application

b. **Testing cases**

   i. **Account Information**

      1. **Initial:** Two sets of data, first are all the possible valid user data input and second is all the possible invalid user data input.
      2. **Execution:** Try on all possible invalid user set up and see whether the system would let it go through or not, and try out all possible valid user to see whether all of them will set up successful.
      3. **Final:** Look through all the output results and see whether there are any bugs or exceptions that are not handled by the system yet.

ii. **General Functionality**

1. **Initial:** Use two test user accounts to run test cases. These test user accounts will have a series of text messages, and search options that will cover expected and unexpected input by users.
2. **Execution:** Run users through simulations of finding/connecting with other test users, view the test profiles from the user, send messages to other users (and view the result), search for users and also search with incorrect input, check that users can become members of student organizations and create/view/locate events.
3. **Final:** Test that the expected user input updates the system correctly and provides expected output in terms of other user profiles, messages, events, and searches. Confirm that unexpected output is handled with appropriate error messages and suggested input (if any), and do not cause the system to crash.

iii. **Dynamic Data**

1. **Initial:** Use two test users' accounts, and make them be friends on the system. Create test messages to send through the system and set up alerts for classes and messages.
2. **Execution:** Send inbox information to the two test users from the system to see whether they will get a real-time notification, also send messages within each other as friends to make sure the inbox react the exact way we wanted for the system. Second of all, let the

two test users in completely different classes or organization to see whether they received the updates in real-time correctly as well.

3. **Final:** If any of the two cases failed to meet the real-time notification purpose, then the tester needs to see what might be the issues and let the developer noticed the issues or bugs.

c. **Results:** Different test cases and three different testers previously mentioned to target each of the cases will help test out the application well before its launches to any platform.