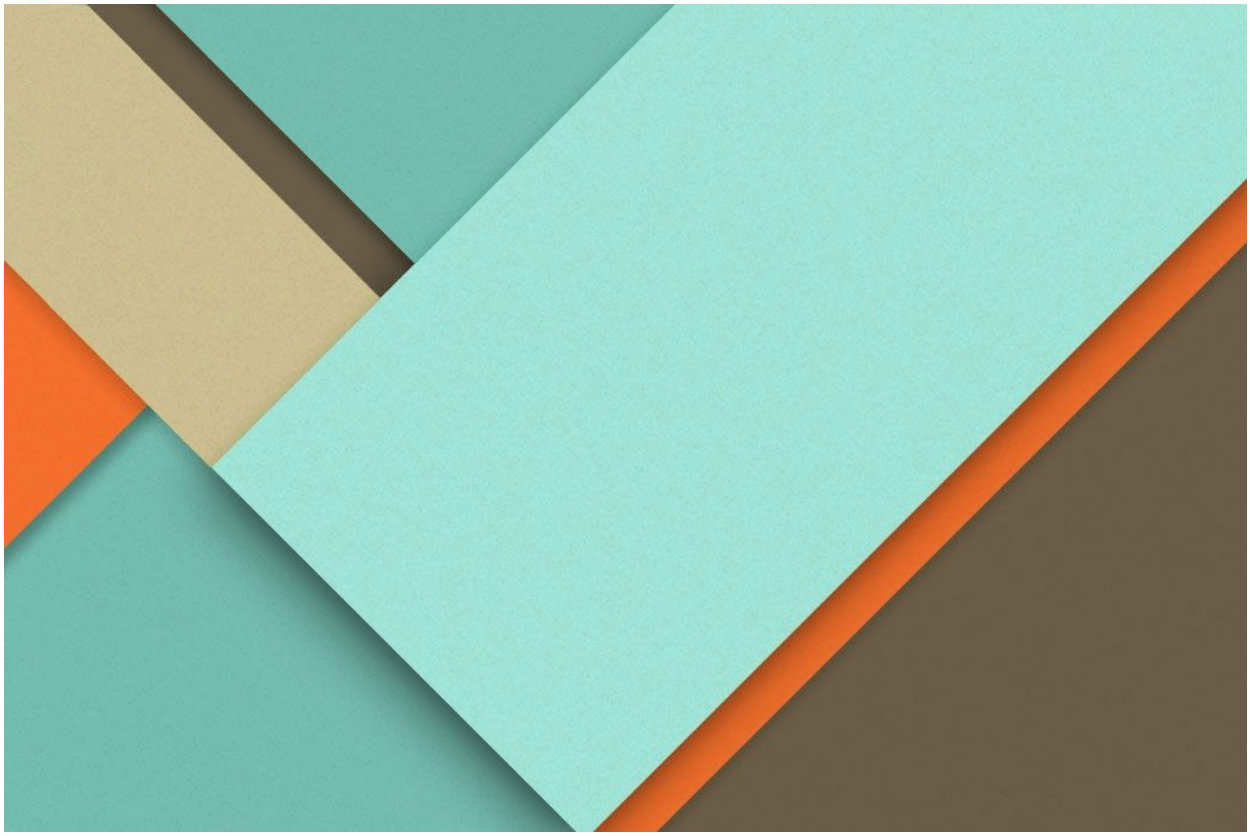


System Design

19/10/2020

Mohammad Amr Khan

Tony Xu

Sofia Rahul

Rui Wu

Winston Ge

Mohammad Sajjad

Sahil Hakimi

Team TBD
CSCC01

Table of Contents

CRC Cards

Front-End

navbar(consultant)	
-Implements a sidebar/navbar that allows users to navigate to other relevant pages	navbarItems

navbarItems	
-Includes the content for all the links that users may need to navigate to with a navbar	

ConsultantDashboard	
-Has a sidebar tailored to consultants to let them navigate the website -Has a topbar to let users know it is the dashboard -Contains a calendar with all upcoming events and relevant dates for them -Displays all the courses they are teaching	navbar(consultant) Calendar

SocialInitiativeDashboard	
- Includes a side navigation bar that helps a social organization user navigate the site - Includes profile information about the organization as it would be seen by other users - Includes the ability to edit profile info, about us, mission statement - Includes mini navbar with items:	Navbar (Social Initiative) *see below*

about, current campaigns, what you can do	

Navbar (Social Initiative)	
- Navigation bar component for the Social Initiative user to use to navigate to other pages	navbarItems

navbarItems (Social Initiative)	
- Know the components of the items on the social initiative user's side navigation bar	

Editable	
- Allows text on the page to be editable (i.e. a simple div when displaying, but a textbox when editing)	

Profile	
- Displays all the information about the user (education, skills, languages, firstname, lastname, email, phonenumber, etc...) - Maintains information of the user and updates the database when information is edited - Maintains a profile picture and updates it accordingly when the user uploads a new one - Has a side menu bar to navigate	Editable ProfileBar Footer UserRouter

ProfileBar	
- A side menu bar that allows navigation to other areas of the	learnerNavItems navbarItems (consultant)

dashboard (home, my class, opportunities, logout, etc...)	

learnerNavItems	
- Know the different headings that will be displayed on the learner sidebar - Know the paths to redirect to once one of those headings are clicked - Know the icons for the pictures	

learnerNav	
- Create the topbar - Create the heading and subheadings for the dashboard view - Create the sidebar - Implement the logo for the company	learnerNavItems

LearnerDashboard	
- The main page that implements learnerNav and footer together in one view	learnerNav footer Calendar

PrivateRoute	
- Keeps track of if the user is logged in - if not redirects them to the homepage	

Auth	
- Creates the context that holds the state of the user	Firebase

About

- Used to render the about page	

Footer	
- Creates the footer component so that all the other classes can call it.	

Homepage	
- Used to render the homepage (the first page that the user see)	Firebase

Login	
- Used to render the login page - Uses the firebase.js file to validate the users	Firebase

Signup	
- Used to render the signup - Pass the data to firebase to create used	Firebase

NavBarHome	
- Used to render the navigation bar for before the user is logged	About Homepage Login Signup

Calendar	
- Display upcoming class sessions for the user	LearnerRouter ConsultantRouter CourseRouter

LeanerCourse	
- Render a learner's view of a course - E.g. course info, sessions, drop a course - Also allows the learner to switch to another course	LearnerRouter CourseRouter

ConsultantCourse	
- Render a consultant's view of a course - Also allows the consultant to switch to another course	ConsultantRouter CourseRouter

AllCourses	
- Displays all the courses available - Allows the user to join or save courses	LeanerRouter CourseRouter

SavedCourses	
- Display all courses saved by a user - Allows the user to join or unsave courses	LeanerRouter CourseRouter

AllInitiatives	
- Display all existing social initiatives with options to search/filter	SocialInitiativeRouter

SocialInitiativeInfo	
- Display information for a particular social initiative (e.g. contact info, description, available positions)	SocialInitiativeRouter

CreateCourse

- Allow consultant to create a course	CourseRouter ConsultantRouter

SessionManager	
- Allow consultant to manage sessions for a course (e.g. time, instructions for joining, documents)	CourseRouter

InitiativePositions	
- Allow social initiative to view their currently posted jobs/volunteer opportunities - Allow them to create, edit, and delete	SocialInitiativeRouter PositionRouter Application (front-end)

Application	
- Pop up that displays an application to a position (e.g. who applied, skills, resume, cover letter etc) - Can accept or reject	PositionRouter ApplicationRouter

Back-End

UserModel	
- Stores data about users (id, userType, firstname, lastname, etc...)	

LearnerModel	
- Stores data about the learner - Inherits data fields present in User	UserModel (parent)

ConsultantModel	
- Stores data about consultant - Inherits data fields present in User	UserModel (parent)

SocialInitiativeModel	
- Stores data about a social initiative - Inherits data fields present in User	UserModel (parent)

Course	
- Knows info about a course (name of course, course outline, dates, announcements etc.) - Knows what users are taking it - Knows what user have bookmarked it - Know which instructor is teaching it	- LearnerModel - ConsultantModel

Position	
- Has information about jobs/volunteer opportunities such as skills required, which users applied, posted by etc.	- SocialInitiativeModel

Application	
- Has information about applications to a position (e.g. which learner applied, what position, resume, cover letter)	- Position

Firebase	
- Stores all the information required to connect to firebase	

UserRouter	
- Collect the data in API and send the data to mongoose - Extract data from Mongoose database and display on API	UserModel

CourseRouter	
- Collects the JSON data received by an API call and interacts with Mongo to store or get course information	Course

PositionRouter	
- Collects the JSON data received by an API call and interacts with MongoDB to store or get a position (job/volunteer) information	Position

LearnerRouter	
- Collects the JSON data received by an API call and interacts with MongoDB to store or get learner-specific information (e.g. courses taken)	LearnerModel

ConsultantRouter	
- Collects the JSON data received by an API call and interacts with MongoDB to store or get Consultant-specific information (e.g. courses taught)	ConsultantModel

SocialInitiativeRouter	
- Collects the JSON data received by	SocialInitiativeModel

an API call and interacts with MongoDB to store or get social initiative-specific information (e.g. posted positions)	

ApplicationRouter	
- Collects the JSON data received by an API call and interacts with MongoDB to store or get application information	Application

System Architecture

For this project as a group, we chose to use the MERN stack as the technology stack and the MVC model as the architectural model.

System Interaction with Environment

The system is a web application that is built using the MERN stack. MongoDB Atlas is the NoSQL database that is going to be used to store data. We are using the Atlas version which is a cloud-based deployment, offered on three major cloud providers from Mongo. Express.js is being used to help build APIs to the database. Firebase is used for user authentication.

Accounts have already been set up for Firebase and MongoDB Atlas and are being used to send and get data from those sources.

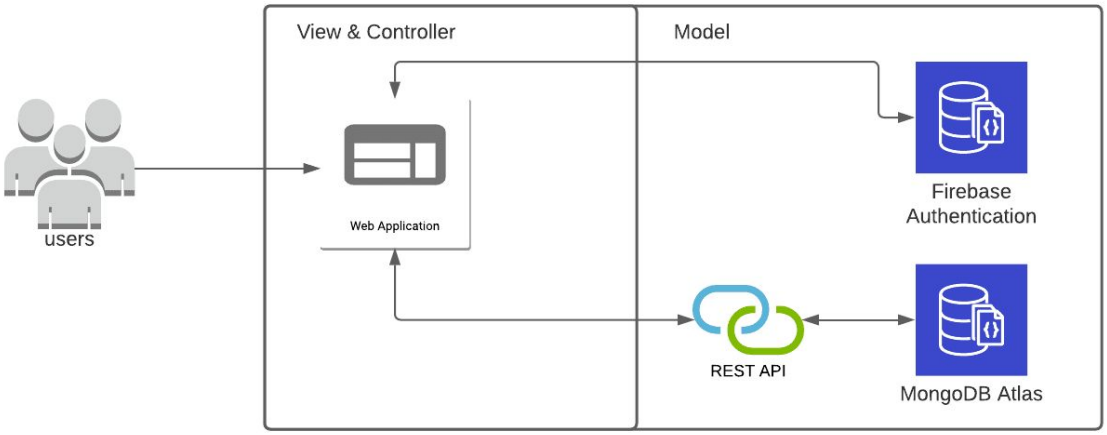
All Node.js dependencies have been added to the package.json, so running **npm install** will install them.

How to Run Application

1. Ensure that React installed on your system before proceeding
2. Clone the project from the Github repository
3. Using CLI launch the project
    a. To launch the back-end:
        i. Navigate to <Repo_Home>/u-impactify/backend
        ii. Run the command - **npm install** (only do this if on the first time)
        iii. Run the command - **nodemon server**
        iv. Backend is available at localhost:500 for API calls
    b. To launch the front-end:

i. Navigate to <Repo_Home>/u-impactify
ii. Run the command - **npm install** (only do this if on the first time)
iii. Run the command - **npm start**
iv. The web app will be available using a browser at localhost:3000

System Architecture



The diagram above shows system interaction and flows with the application. An explanation of the components below.

Component	Description
User	The user is the person who will be interacting with the application
Web App	The web application is what the user will be interacting with and where they will see information based on what they select
Firebase Authentication	Authentication utility of firebase is leveraged, and the credentials are stored in Firebase
REST API	Used to send data from the front-end to MongoDB using API calls
MongoDB Atlas	A cloud version of MongoDB is being used. This will be used to store data.

System Decomposition

MongoDB

MongoDB will be used to store information about the users (social initiatives, learners, consultants) that will be displayed on their profiles. MongoDB will also be used to store information about courses and jobs that are available. This will be part of the model in the MVC architecture.

Web Application

The web application that is being developed will serve as the viewer and controller in the MVC architecture. Uses will view everything using the website and interact with it and any changes will be reflected on the web pages. Express.js and Axios will be used to get or put information from MongoDB (model). In the case of an error what went wrong will be displayed on the page.

Firebase Authentication

Firebase Authentication will be used to authenticate users for the website. Firebase will allow users to log in or signup with their email, or use Facebook or LinkedIn to authenticate with. If the users are unable to authenticate they will not be able to access all the features that require an account. They will only be able to navigate to what is available on the landing page. This is part of the model in the MVC architecture.