MigrantHub

Team members

Name and Student id	GitHub id	Number of story points that member was an **author** on.	Role
Mira Marhaba (27497148) *miramarhaba@gmail.com*	miramarhaba	10	Backend, Frontend
Alexandre Masmoudi (27755473) *a-masmoudi@hotmail.com*	Smoudii	19	Backend, Frontend
Laxman Velauthapillai (40000111) *laxman_20@hotmail.com*	iamlax	16	Backend, Frontend, DevOps
Tusman Akhter (40003476) *tusman96@hotmail.com*	tusmanakhter	16	Backend, Frontend, DevOps
Tajbid Choudhury (40002177) *tajbidhussain@gmail.com*	CodeTaj	12	Backend, Frontend
Mazen Nahle (27003315) *nahlemazen@gmail.com*	mazennahle	16	Backend, Frontend
Rajeevan Vairamuthu (40000112) *rvairam10@gmail.com*	rajee10	14	Backend, Frontend

Project summary

A Social network application geared at international students, immigrants, refugees, and prospective refugees. This web app aims to help newcomers transition into their new life in Canada. The application aims to centralize services for newcomers in one place for easy access. It also aims to create a space where newcomers can interconnect and help one another. A user can create a profile and join communities that he or she is interested in. They can also register for services that will be integrated into the application. These services can be provided by both governmental and non-governmental organizations to help migrants. The application will gather user data about the services used by the migrants. This will include questions they need answers to or specific services that are most helpful to them. The data will be used to provide migrants with better choices.

Risk

In order for our application to provide accurate and relevant suggestions to our users, there is a high risk associated with the need to gather data at such an early stage of development. Without it, users will be unable to later get the guidance that is needed. To attack this most significant of risks, we made sure to begin with issues that cover the sign up and login phase (#4, #9, #16). By doing this, we can make sure to collect much of the later-needed data from the user about their status. We also ensured that they have the ability to modify their information (#2) to ensure that our data is not limited to a snapshot at one point of time, but that conclusions can also be drawn by analyzing the data over time. It is also important that the data obtained is widespread over many features, which is why we tried to implement early versions of as many features as we can (ex. Friends, Services/Reviews, Events), to maximize and diversify these aforementioned conclusions. Gradually, this information will be used to begin with the data analytics that will enhance the application's performance and user experience.

Another risk is that we must have actual users to use this application in order to get the data required. To solve this issue of getting critical mass, our stakeholder will provide us with the network of migrants and people that he has contact with. Another risk he will help us mitigate is providing us with data on multiple services for newcomers that he already knows about, this will help us centralize the services in one place without having to do too much research.

Finally, we will use Google Analytics once the application is being hosted to gather data on our users, how they use the application, and which pages are more visited. We will also use various tools from the Google Cloud Platform, such as TensorFlow and the Cloud Machine Learning Engine to craft models with the collected and already existing data. These models will allow us to learn insights on our users and create a recommendation system to provide them with a personalized experience on our application.

Legal and Ethical issues

- Data being acquired by unauthorized users and used in a way that can harm the migrants.
- Organizations creating services or events to take advantage of migrants
- Spam
- Security concerns: hacking
- Lack of moderation: i.e. someone creating a random event.
    - Not adequately moderating reviews causing misrepresentation of services due to the fact that these services could be crucial

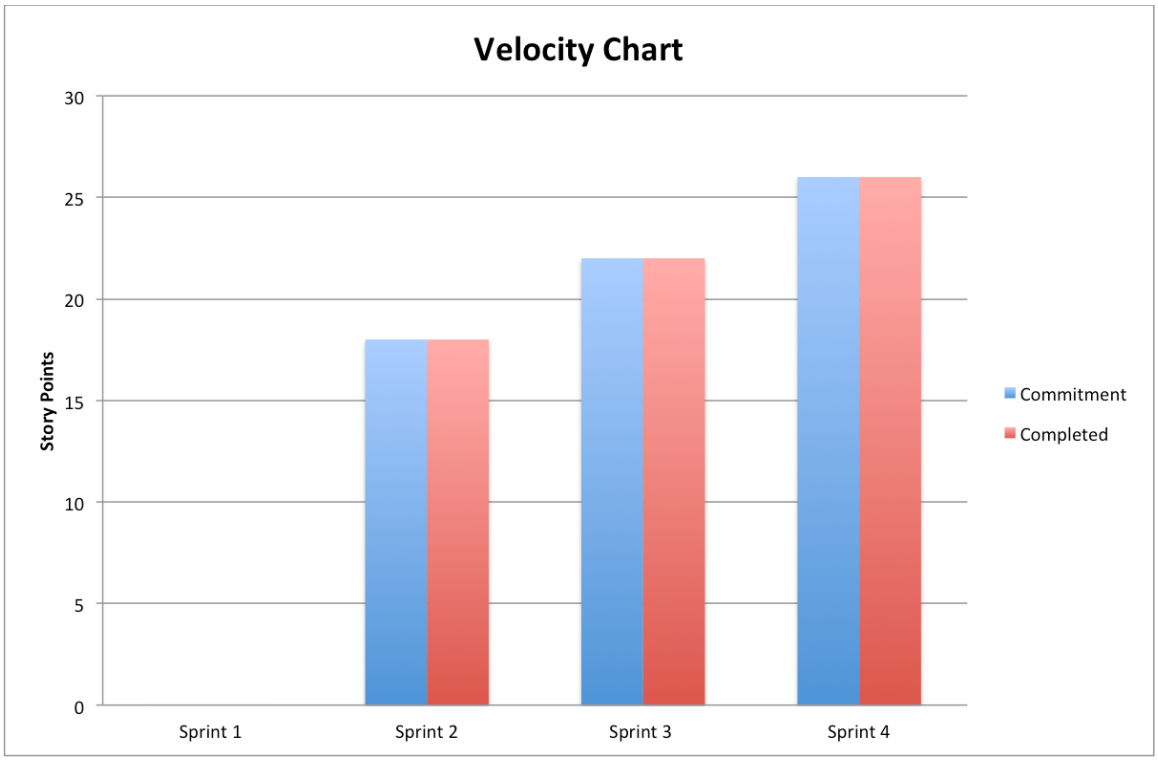
- Third Party Content/Copyright Infringement: i.e. someone posting a text, picture or video from another site.
    - We transfer the right of copyright to the users
- Criminal Activity:
    - Use of the application for illegal purposes are forbidden

Hardware/cloud/compute requirements

- NodeJS installed
- MongoDb installed
  or
- Docker installed

Velocity

Project Total: 19 stories, 66 points over 9 weeks



Iteration 1 (0 stories, 0 points)

Our main achievement for this iteration was getting the requirements from the stakeholder and coming up with user stories. Also, through discussion with the stakeholder, we decided on the tools and frameworks we are comfortable using and we setup a base architecture to use for the project.

Iteration 2 (3 stories, 18 points)

In this sprint, we setup the main signup functionality for both migrants and businesses as well as login. These forms are not normal signup forms since they include much more information than a normal website signup. We also created a homepage for users to be redirected to upon login. We also setup our testing infrastructure for both the backend and frontend.

Iteration 3 (7 stories, 22 points)

In this sprint, we added the following functionalities: the ability for an admin to sign up to an account, a migrant being able to add, accept/reject and see friends, creating events, creating and viewing services and editing your migrant or business profile. We also updated our testing infrastructure to have better tests.

Iteration 4 (9 stories, 26 points)

In this sprint, we added the following functionalities: the ability for a user to edit/delete an event, edit/delete a service while also being able to review other services and searching for services, view a service's reviews, remove a friend from friend list and finally managing admins. Furthermore, we ensured that all our system is properly storing users data without ever deleting them as it is a crucial step towards our goal of performing data analytics.

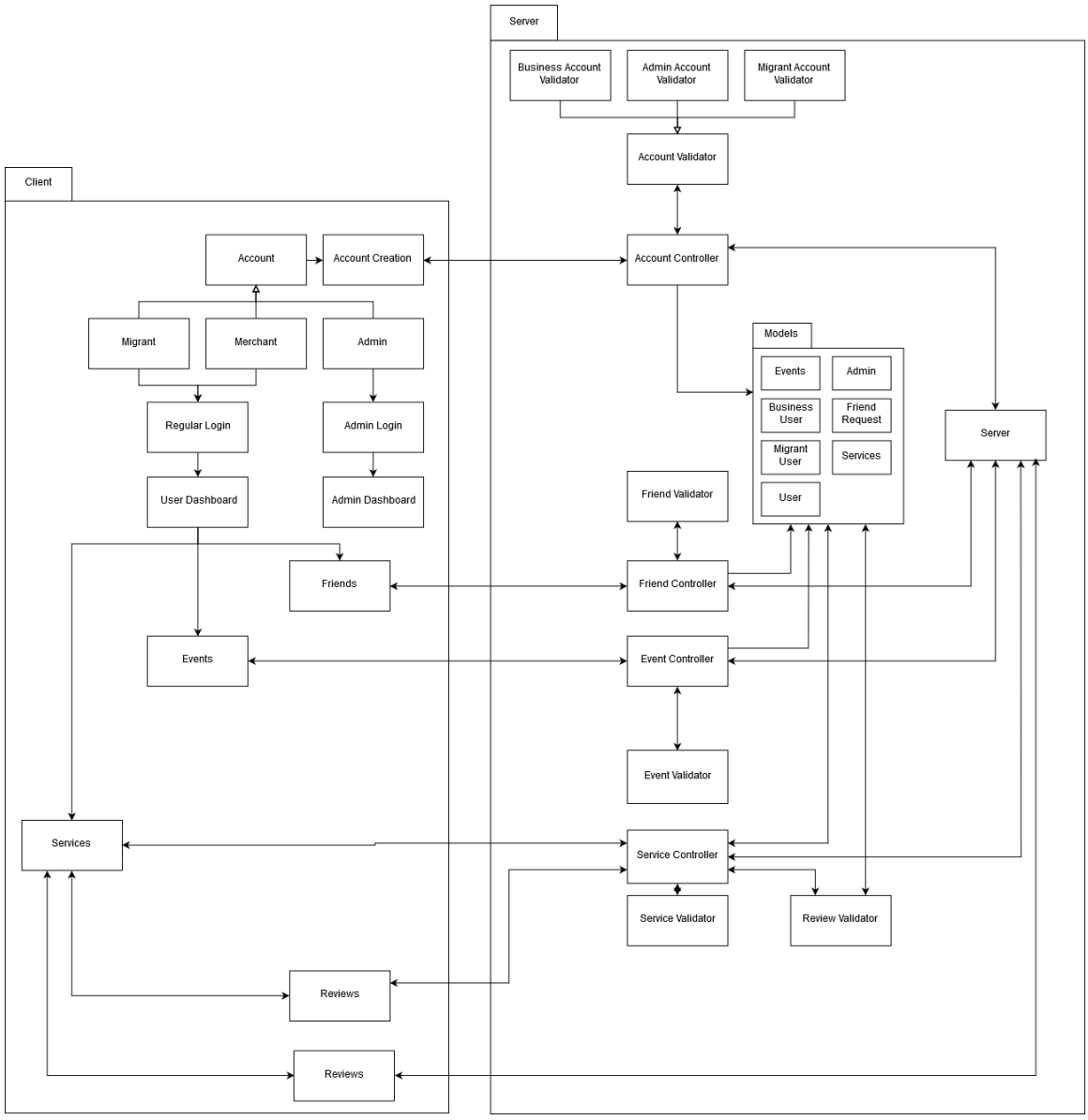
Iteration 5 (8 stories, 21 points) (Planned)

In this sprint, we will work on being able to delete a service in certain cases, delete a review, and referring a service to other migrants. We will ensure that a migrant/merchant can view their profile as opposed to just being able to edit it in the previous iteration. We will also be able to search for people, delete events, and create groups.

Iteration 6 (6 stories, 23 points) (Planned)

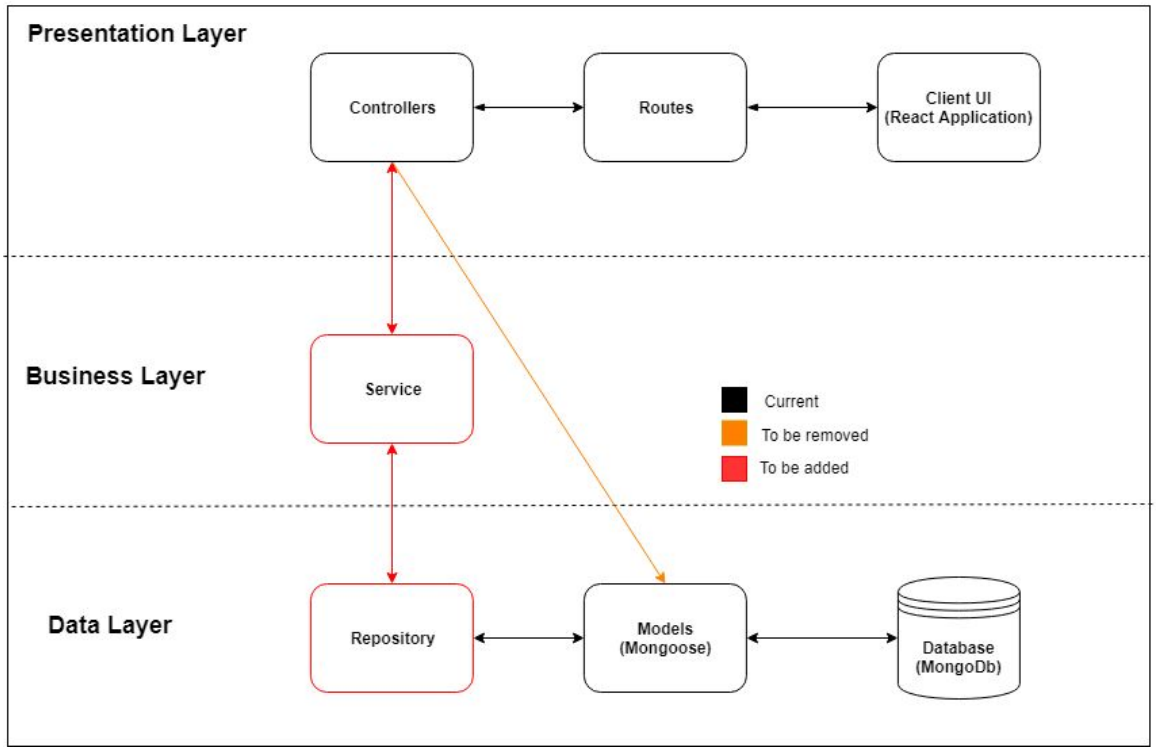
In this sprint, we will finally start taking a look at the Data Analysis portion of the project, which is extremely important as it plays one of the largest roles for our interface and helps achieve the ultimate goal. We will also be delving into the group page feature, by being able to edit a group's settings, deleting a group, and joining a group. We are also planning on giving a migrant the ability to edit their own privacy settings and to suggest a service to other migrants.

Overall Arch and Class diagram

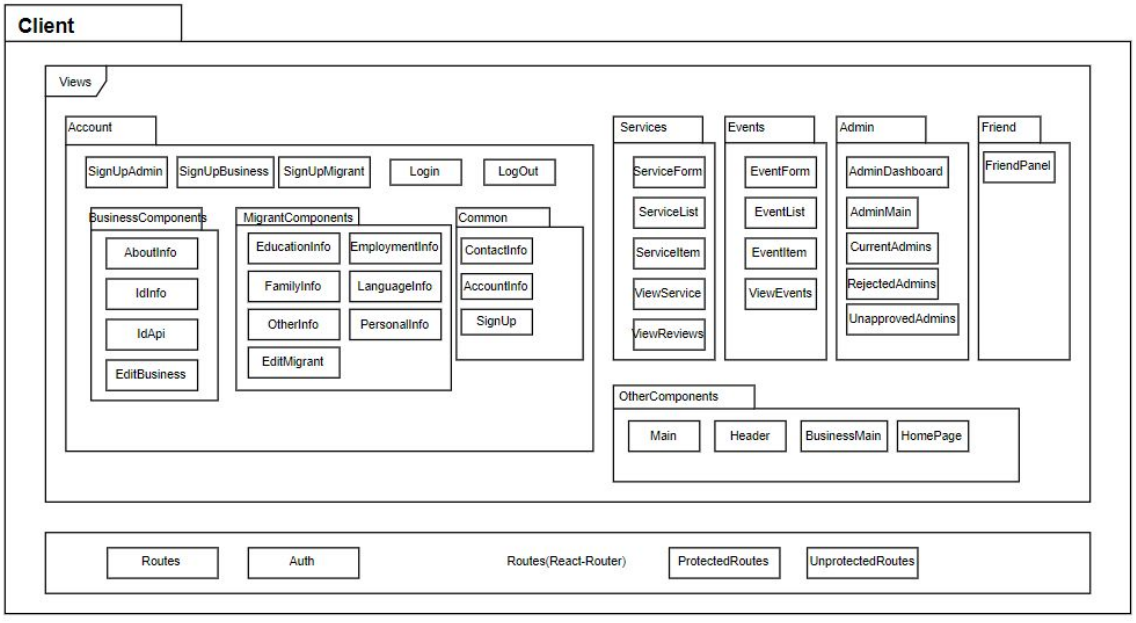


This is the overall architecture diagram showing the current general main components of the application and the links between them.

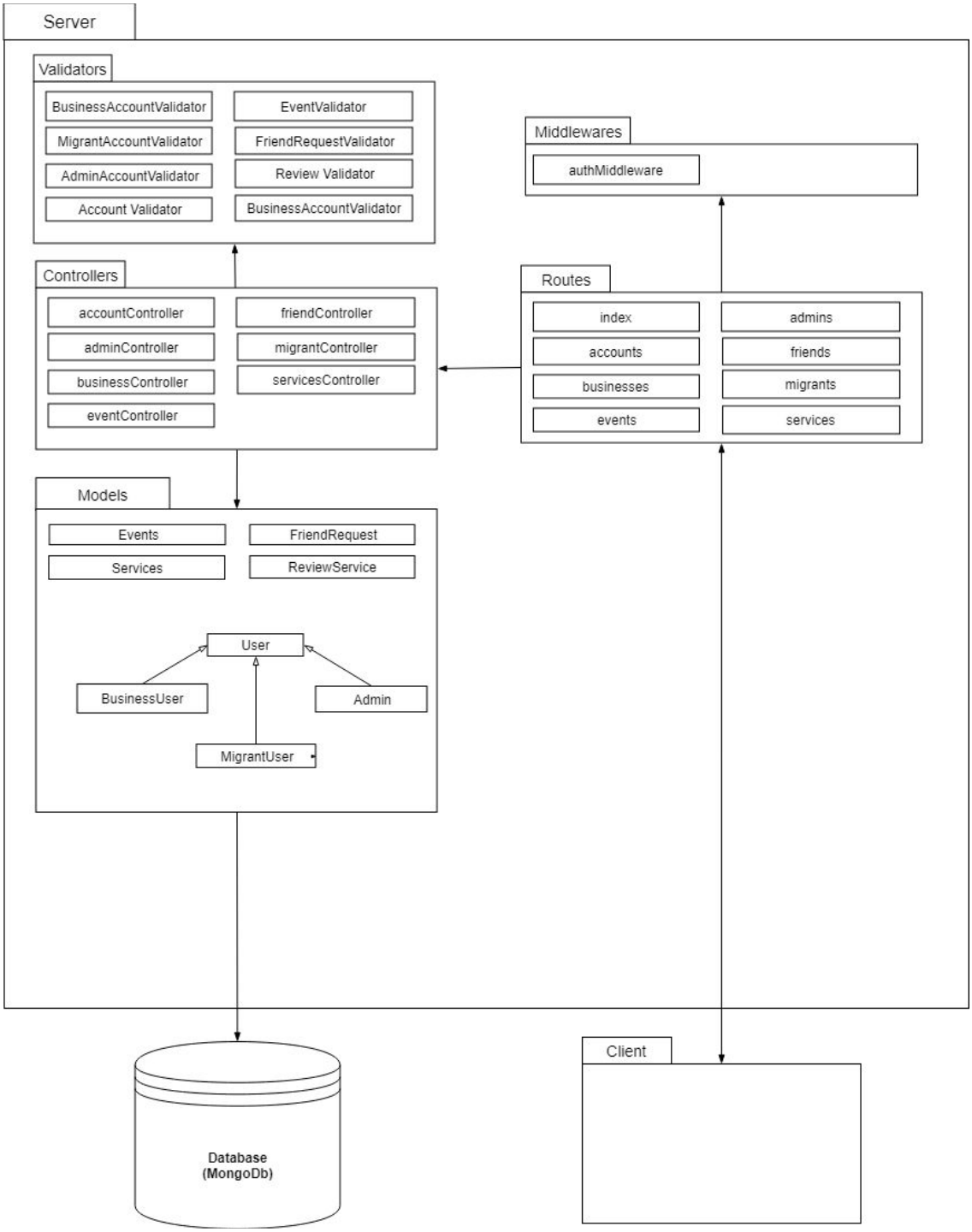
High Level Architecture Diagram



This is our architecture diagram, in the current architecture we do not include the patterns highlighted in red, however after discussion and recommendation from the TA, we will be refactoring our code to include this.



This is the overall client side class diagram.



Overall architecture and package diagram for server.

Infrastructure

Hosting

The application will be hosted using Google Cloud using the Google App Engine which will allow us to easily deploy our app as a docker container into the cloud without too much pre-configuration.

Frontend

Frameworks

React

Since our application is a web application, we will be using HTML, CSS and JavaScript. To make development faster and to build a good UI, we are deciding to use the React framework. This will allow us to encapsulate our application into components that we can use in many places, eliminated code duplication. This framework will also allow us to keep the DOM separate from our components logic and allow us to understand our code easier.

- Testing Tools
    - Jest: a javascript test runner that allows us to run javascript tests for react. Also generates coverage reports for the client.
    - Enzyme: allows us to mount single react components for unit testing and conduct unit tests on the user interface
- React.js Tools
    - react
    - react-autosuggest
    - react-dom
    - react-input-mask
    - react-number-format
    - react-router
    - react-router-dom
    - react-scripts
    - react-text-mask
    - react-transition-group
- UI Tools
    - Material-UI: The main framework for UI design.
    - MDBootstrap: Secondary framework for UI design.
    - animated
    - autosuggest-highlight
    - lodash
    - react-google-maps

    - recompose
- Validation
    - canada: library of canadian provinces and cities.
    - country-data: library of languages
    - validator: library of string validators
    - qs
- Server- Client communication
    - Axios: used to do http requests

Server

The back end of the application will be made using Node.js + Express.js.

The Node.js runtime environment will allow us to use JavaScript on our backend so that we can use one language across the stack and make our development easier. Our application will also need real-time functionality for messaging and Node.js will make this convenient with web sockets with a library such as socket.io.

Express.js framework will provide us with many functions that will speed up our development such as routing and middleware functionality. Express will also allow freedom for our choice of backend and project structure that frameworks such as Meteor or Sails would not allow. We will add libraries as we need them since Express has good package support.

- Login security
    - Bcryptjs: salt hash library that is used to store data securely and provide security against brute force attacks.
        - Used to hash a user's password
- Client-Server communication
    - Body-parser: parse incoming request bodies and get data available in the request body.
        - Used to communicate between client and server.
- Client request validation
    - validator: predefined rules for validating different types of strings are provided which save us time from writing our own regex.
- Testing Tools
    - Chai: assertion library used with Mocha to perform unit/integration tests. Provides numerous assertion capabilities to our tests.
    - Mocha: testing engine for express to run unit tests.
    - Sinon: JavaScript test spies, stubs and mocks
    - Sinon-test: Utility tool for Sinon, that allows cleaner tests.
    - Istanbul: Used to generate code coverage reports.
- Helper Libraries

    - Nodemon: allows for continuous restart of the web app when developing the app, this allows for changes to be seen instantly for the express server.
    - Concurrently: allows for us to run two node applications with one command, this allows us to run the express backend and react frontend at the same time.
- Node.js Tools
    - Express: provides features to build Node.js applications.
    - Morgan: HTTP request logger middleware.
    - Cookie-parser: allows usage of cookies
    - Debug: javascript debugger
- Store images on server
    - multer: Middleware to upload files(i.e. Images in our case)
    - fs-extra: Provides file system methods that are used to correctly upload images.
- Logging
    - Winston: logging library
    - App-root-path: specify path for log files

Database

Mongodb

MongoDB is a NoSQL database. We will be able to easily store JSON formats of data, that is typically used in a web application. It will also allow us to be more flexible with our schemas since they will be unstable in the beginning and using a relational database would create lots of overhead to change the schema.

- Mongoose: Object modelling tool that allows us to create a object format in mongoose with CRUD functionality.

Name Conventions

AirBnb JavaScript Style Guide: This style guide supports react.js as well as node.js. In addition, there is a eslint airbnb package that is available with/without react.js support that can be added to the project.

Code

File path with clickable GitHub link	Purpose (1 line description)
MigrantHub/server/controllers/accountController.js	All functions related to accounts (create, edit, login, logout) are in this class.
MigrantHub/client/src/account/business/SignUpBusiness.js	All functions related to create a business account.
MigrantHub/client/src/account/SignUpMigrant.js	Contains the main logic for a migrant to sign up for the service
MigrantHub/server/controllers/ServicesController.js	Controller for all functions related to services.
MigrantHub/MigrantHub/server/controllers/friendController.js	Main functionality for adding and managing a friend invitation.

Testing and Continuous Integration

Test File path with clickable GitHub link	What is it testing (1 line description)
MigrantHub/server/test/controllers/AccountControllerTests.js	That the account controller methods are working correctly.
MigrantHub/server/test/controllers/ServicesControllerTest.js	That the services controller methods are working correctly
MigrantHub/server/test/MigrantValidatorTest.js	Checks that a migrant account validator is correctly validating fields
MigrantHub/server/test/validators/ServiceValidatorTest.js	Checks that a service validator is correctly validating fields
MigrantHub/MigrantHub/server/test/controllers/FriendControllerTest.js	Check that an accepted friend request is properly making calls to the database

Our continuous integration environment is setup using TravisCI. The following is a link to our project: MigrantHub.

We have set this up to use the latest LTS version of node and npm, this then builds our project, in the client folder and in the server folder. After the builds are complete the test suites are run in the client folder and in the server folder. The client folder runs tests using Jest which conveniently is also is able to provide a coverage report. The server folder uses mocha to run tests and istanbul to create

code coverage reports. The code coverage reports are sent to codecov after each build to help us track the progress and difference. We have also added a linter to our application using ESlint, we fail builds that do not pass the linting for the server. This continuous integration pipeline is set to run after each commit as well as on each branch pull request. It is also run on master every time something is merged.