SLUber Architecture

Updated 5/11/18

**General Overview**

The app consists of three main parts.

1. **MongoDB server** – Backend is configured to point to localhost:27017/sluber (default port for Mongo on localhost, using database ‘sluber’). The backend uses collection ‘rides’ that is created in the backend and referenced.
2. **Express backend** – Default runs on port 4200, easily changeable in config. Processes ride requests with format specified in ‘Ride.js’ in the models folder of backend, and routes to add/update/delete rides found in ‘rideRouter.js’ in the routes folder of backend.
3. **React frontend –** Runs in Node environment. When main URL of app is accessed, points to client to add a ride. Adding ‘/dispatcher’ after the main URL brings you to dispatcher side. Dispatcher components are found in the dispatcher folder under components in the client. Ignore the dispatcher folder found in the top-level directory, only use the one within ‘client’.

**Backend**

The backend uses Mongoose to create models for ride requests (found in /backend/models/Ride.js) Each request can have any of the attributes listed in this file. Ride additions, updates, or deletions are handled using /backend/routes/rideRouter.js. Actual database lookups are done there. Data is passed back as JSON. New routes performing new functionality may be added here, using appropriate Mongoose/MongoDB commands.

**Frontend**

All files are found within the client folder.

App.css, App.js, and App.test.js can all be ignored – they are left over from ‘create-react-app’ template and can eventually be deleted.

Config.js contains the route to the backend server referenced in the rest of the frontend, making it easy to port this to other systems.

Index.css contains all styling for the frontend, accessible using the className attribute in HTML code.

Index.js contains all routes for the front-end, more can be easily added. This utilizes react-router, and extensive documentation on this can be found online.

Locations.js contains all approved locations from SLU Ride. This should be updated when new buildings are constructed or when pickup/dropoff points are added/removed. Addition/deletion could possibly be implemented on the dispatcher side.

/components/AddRide.js is the main ride ordering screen where users fill out personal information and it is sent to the backend and thus the dispatcher. Google Maps implementation is found here. After adding a ride, this file receives back the Mongo ID for the ride from RideService.js and passes it as a parameter to Complete.js (the url is something like /complete/abcd1234, where abcd1234 is the Mongo ID)

/components/Complete.js is the screen that users are directed to after requesting a ride, with status updates given to them. It queries the database every second for updates to the status tag for that particular ride.

/components/EditRide.js is used for the client to edit a ride after it is added. The client can only do this if the ride is marked as ‘Active’ and not dispatched yet.

/components/IndexItem.js is the main screen users are directed to with hours of operation, a count of current rides, and the request ride button. Eventually, this shouldn’t allow users to request a ride if it is not during SLU Ride’s operating hours.

/components/RideService.js provides helper functions for the front-end to communicate with the backend (ride addition, deletion, update). It provides the Mongo ID of the newly created ride back to the application so that the complete page can use this ID to show ride updates.

/components/dispatcher/AddRide.js is the contents of the modal overlay that appears when ‘Add Ride’ is clicked on the dispatcher side.

/components/dispatcher/DispatcherIndex.js is the primary dispatcher side that pulls together various components to show active rides and provide ride addition/viewing/deleting

/components/dispatcher/EditRide.js is no longer in use – ride editing is achieved in TableRow.js

/components/dispatcher/LoginPage.js is our login page – not functional yet.

/components/dispatcher/LoginService.js will contain helper functions to login – not functional yet.

/components/dispatcher/RideService.js provides helper functions for the dispatcher side to communicate with the backend (addition, deletion, update)

/components/dispatcher/TableRow.js is used by the primary page to map all rides from the database and show their respective information. It provides the modal popup with full info, as well as the ability to edit each ride inline.