**ARCHITECTURE**The CBD Design project is built using a RESTful Architecture (MERN Stack) with the Front end being a Single Page Application developed using React and Redux that connects to a REST API in the backend built using NodeJS. The data for the project is stored in a NOSQL MongoDB Database.

We adopted a stateless approach by using a REST API instead of the more traditional Dynamic Approach with pages being rendered on the server for two main reasons: loose coupling and easier scalability. Since the project involved rapid prototyping with the design of the dashboard changing constantly, we wanted to keep the server code separate and untouched as we played around with different designs on the UI and deployed several versions parallelly. The second reason takes into consideration the possible increase on the load of the server as the project gets wider adoption within the university. Since the server doesn’t hold any state at a given point of time, multiple instances can be spawned behind a common gateway. Also, if a client (department or program) wanted the data of their residents to remain independent we can always spawn a different server for them while still reusing the UI.

The Nginx Server (UI), Mongo dB server and the Node Server (backend) are all hosted in a single virtual box (Ubuntu 18.04 LTS) at **cbme.usask.ca**. Access to the mongodB port is restricted locally within the server and is utilized by the NodeJS backend for CRUD operations.

**DATABASE**

Because of our rapid prototype design, we did not have a very clear-cut understanding of the data that we would be storing and so as the project evolved our document model evolved too which is why we adopted a NoSQL approach for our database. Also, mongodB was our preferred choice because we were using a NodeJS server for our backend and so we could access data directly as JSON objects without having to parse them.

Our database **RCM-CBD** has two collections **USERS** and **RECORDS**

**USER DOCUMENT MODEL**

This table is for storing user information and for authentication and providing additional information regarding residents. Entries in this table can be updated by admin users through the admin tab from the UI. We use an access type entry in the user table to distinguish users, It can be: admin, program director, competence committee member, academic advisor or resident. The access type of a user determines what pages they are able see on the UI. Access to the dashboard is controlled through the CAS (central authentication system) interface maintained by the University of Saskatchewan. Users are redirected to our dashboard after they login successfully and the CAS system provides us with their NSID (unique six character code). We check for a corresponding entry for that NSID in our user database and then issue a JWT (JSON web token) for access to our dashboard.

**RECORD DOCUMENT MODEL**

This table holds all the EPA observation records and is updated when new data files are uploaded for residents. While we don’t use information in some columns currently such as “professionalism safety” and “observer type”, this ****data is stored in pretty much the same format that the royal college portal exports. Also, when a user is deleted on the user table, we automatically drop all records stored for him in this table.

**SERVER**

The server is built using expressJS on NodeJS, the local server is http while the production server is secured as https using certificate keys from certbot. All API Requests to the server are logged in a file **server-combined.log** and a curated list of calls are logged in a file **server-activity.log**. The second file is meant for research purposes to study how the tool is being used while the first file is for more generic debugging. The server thread once launched is maintained by the pm2 process manager.

**WEB GUI**

The UI is built as a single page application using React and Redux. We use webpack a module bundler for running our development servers and also for creating a production build with all our web files bundled together as static assets.

React is used for all the DOM manipulations and routing is done by react-router. Each component on the page is built as an individual react component with its styling residing in an individual corresponding style sheet. The application logic is centralized using Redux with the logic for smaller components sitting in the component file itself. The styling is done in sass and the several sass files are compiled by webpack into a single css file. Apart from the main dependencies we also use a couple of other libraries such as moment (data manipulation), lodash (array handling), axios (http requests) and d3 (visualization).

We use a custom shell script to currently deploy our code when an update is made, but this will eventually be refactored so we can have continuous integration directly through the master branch from our codebase.

All our source code is version controlled using git and is available on GitHub under an open-access license

<https://github.com/kiranbandi/cbd-dashboard-ui>  
<https://github.com/kiranbandi/cbd-dashboard-server>

We track our current development process using Trello - <https://trello.com/b/SuLCQr1u/cbd-design-board>