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 involves 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 Apache Server (UI), Mongo dB server and the Node Server(backend) are all hosted in a single virtual box (Ubuntu 18.04 LTS) at **cbd.usask.ca** ports 80, 27017 and 8081 respectively. The mongodB port is accessible only locally within the server and is utilized by the Node Server for CRUD operations.

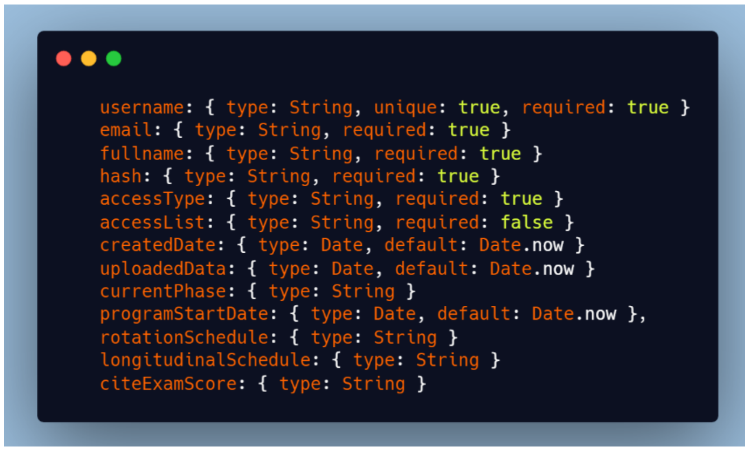
DATABASE

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 all the user level information and is used for authentication and providing additional information regarding residents. It is updated by admin users through the admin tab on the UI.



```
username: { type: String, unique: true, required: true }
email: { type: String, required: true }
fullname: { type: String, required: true }
hash: { type: String, required: true }
accessType: { type: String, required: true }
accessList: { type: String, required: false }
createdDate: { type: Date, default: Date.now }
uploadedData: { type: Date, default: Date.now }
currentPhase: { type: String }
programStartDate: { type: Date, default: Date.now },
rotationSchedule: { type: String }
longitudinalSchedule: { type: String }
citeExamScore: { type: String }
```

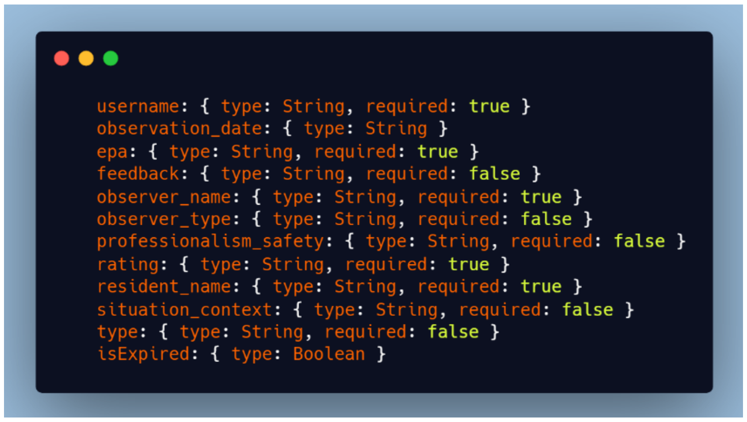
Passwords are encrypted using **bcrypt** a hashing library before they are stored.

Access Type can be of the following types – admin, program director, committee member, academic advisor and resident. The access type of a user determines what pages they will see on the UI.

AccessList stores the list of residents that can be accessed by academic advisors and is only used for users who are academic advisors. Similarly, the last 6 columns are only used for residents and their information.

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 some columns currently like 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.



```
username: { type: String, required: true }
observation_date: { type: String }
epa: { type: String, required: true }
feedback: { type: String, required: false }
observer_name: { type: String, required: true }
observer_type: { type: String, required: false }
professionalism_safety: { type: String, required: false }
rating: { type: String, required: true }
resident_name: { type: String, required: true }
situation_context: { type: String, required: false }
type: { type: String, required: false }
isExpired: { type: Boolean }
```

SERVER

The server is built using expressJS in NodeJS, the local server is http while the production server is secured as https using certificate keys from certbot (the keys need to be updated every 90 Days). 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 purposes. At some point we did think about writing some unit test cases but then again, we like to live life dangerously and so put it on the back burner again. 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 currently stored on the CS Departments Gitlab Server here (drop a mail to venkat.bandi@usask.ca for access)–

https://git.cs.usask.ca/em-cdb/em-cbd-ui
https://git.cs.usask.ca/em-cdb/em-cbd-server

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