Build an incident management service using a Node.js framework on the backend, and

React on the frontend.

The service should have the following features:

● Raise an incident as an admin

● Assign the incident to a user

● Acknowledge the incident as a user

● Resolve the incident as a user

● Read details about a certain incident

● Index incidents (includes filtering, sorting by date created/updated and incident type

and paging)

● Delete an incident

Usage of following DBs is allowed:

● CouchDB (preferable)

● MongoDB

● Postgres

Admin Role

We can create an incident management dashboard which will be logged in by administrator of the application

The user list will be fetched from DB and new users could be added

Admin see’s a list of open incidents and selects the user from the drop down and presses the assign button to assign the incident to him

Admin can filter the incidents

Admin can delete the incidents

Admin can add new incidents

User Role

User can login into the application and see the incidents assigned to him

User can acknowledge the new incidents received

User can read the details of the incident ticket

User can mark the ticket as resolved from the incident detail view

User can filter the incident by a filtering criteria

Collections in mongo db

Admin Collection (contains the list of assigned and unassigned tickets)

User Collection (contains the list of incidents assigned to the user)

Auth Collection (for both admin and user)

Incidentdetails Collection (contains the details of incidents)

Admin Collection

[

{ \_id: 58fdbf5c0ef8a50b4cdd9a86, assignedTo: ‘<<UserID/can be blank if unassigned>>’, assignedDate: <<date>>, status:<<open/closed/inprogress/unassigned>>}

{ \_id: 58fdbf5c0ef8a50b4cdd9a87, assignedTo: ‘<<UserID/can be blank if unassigned>>’, assignedDate: <<date>>, status:<<open/closed/inprogress/unassigned>>}

]

Auth Collection

[  
  { \_id: 58fdbf5c0ef8a50b4cdd9a86, name: ‘Admin’, password: ‘<<Encrypted Hash>>’},

{ \_id: 58fdbf5c0ef8a50b4cdd9a91, name: 'Viola', password: ‘<<Encrypted Hash>>’},  
  { \_id: 58fdbf5c0ef8a50b4cdd9a8f, name: 'William', address: 'Central st 954'}  
]

Backend services

1) Authentication service

=>OAuth Service to authenticate the user using JWT token, will use the Auth DB, there will be a unique salt used for each user to convert the hashed password to store in mongo db

=> Add user (Only Admin) <POST API>

=> Delete user (Only Admin) <DELETE API>

2) Incident Management service for Admin

Incident Management service with crud operations for

=>adding incidents<Post API>

=>assigning incidents<PUT API>

=>deleting incidents<Del API>

=>fetching the list of incidents<Get API>

3) User Incidents service for Users

User Incident service with operations for

* Acknowledging the incident<PUT API>
* Resolving the Incident<PUT API>
* Fetching the list of incidents<Get API>
* API for admin to post incident to user <POST API/ Put API>

Incident details service

* Fetch the details of incident <Get API>

UI Layer

Node JS services

NoSQL DB

Mongoose for mongo db

Nodejs with Mongo integration