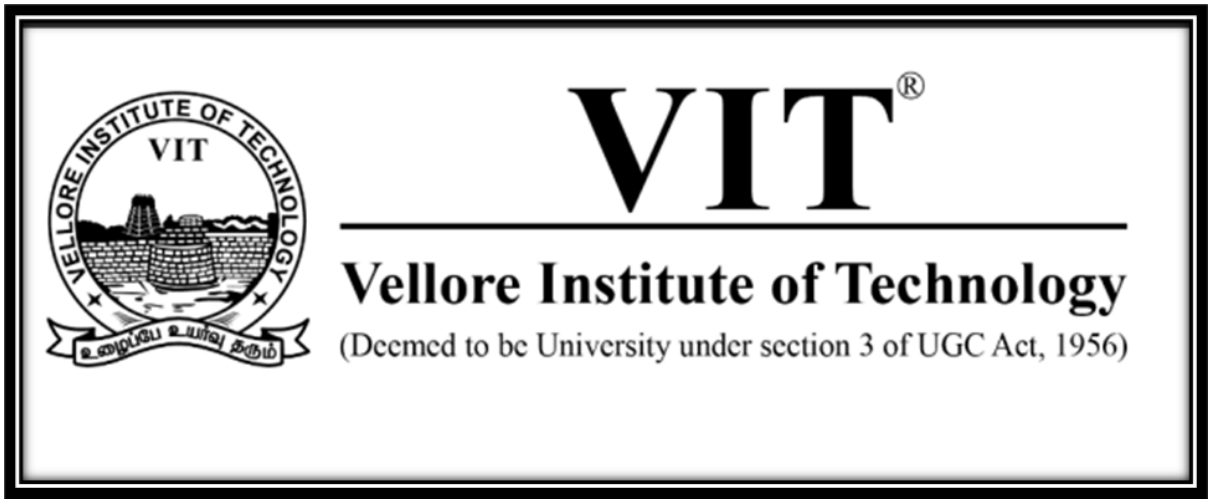

Digital Assignment - 2
CSE1901
Summer Semester Special 2021-2022

App Based Road Safety Violation Recording System

Name	Reg No.
Kulvir Singh	19BCE2074
Anitej Srivastava	19BCE0835
Vardhan Khara	19BCE0833

Problem Statement:

In any industry where logistics division is present, a network of transportation is imperative, especially road transport.
If we take the case of steel producing industry like Tata Steel, there is a need to monitor and evaluate constantly the proper mechanism of parking and scheduling the road transport vehicles in order to reduce and nullify the issues that may arise due to uncoordinated transportation.

Proposed Methodology:

The entire project has been split into the following modules:

Login and Authentication

The user needs to login to the web portal to access the features. Each user has an id and a respective password. The user enters the id and password to the login form and is then sent to the backend for authentication. The user id is first validated and then the password is checked for the corresponding id. Since the password stored in the database is encrypted using bcrypt, the hash has to be decrypted to check for validity of password. On successful match of user id and password, a JSON web token is generated for the user and stored in the local storage. The token has a certain validity time and hence creates a session. Once the token is past its validity time, it gets corrupted and the user's session will expire and will be logged out.

Frontend Routes

The frontend routes are divided across three major components – the public component whose accessibility is public and anyone can view the content encapsulated within it, the dashboard component and the admin component. The public and encapsulated routes to interact with the system.

The Public Component

The public component only consists of the login page. This route consists of a form which asks for the user id and password which is used for logging in and authentication. On entering the details required and clicking the login button, the frontend requests for the authentication route from the backend which validates and authenticates the user as discussed in the previous point. On successful authentication, the user enters the home page and is now able to interact with the dashboard components

The Dashboard Component

This is a custom component which is created to check if the user making request is authenticated or not. This is necessary to ensure security of the application and reduces the vulnerability of the application.

The dashboard component comprises of the following routes and pages on the frontend –

- Dashboard page or Home page

This page consists of a form which is used to register or log a complaint. It is here where firestore is implemented. The user can select and write various details of the complaint and also upload multiple images of the complaint.

- Complaint View page

This page displays all the complaints according to the search query and search parameters entered by the user. It displays the complaint that are filed by the user that is logged in. No other complaints are visible on this page.

- Report page

This page asks for the user to enter a specific date range and accordingly an excel file is generated which contains the dataset of complaints that are registered within the date range specified by the user. A download option is also available after the report has been generated

- Admin page

This page is secured and only allows those users whose type is of ADMIN kind to enter and view this page. it is safe guarded by the admin component discussed below.

Apart from this a logout button option is also present

The Admin Component

This custom component is created to check if the user making the request is authenticated as well as authorized. The authentication is similar to that of the dashboard component. The authorization check is dealt in the following way. The JSON Web Token associated to the user has Payload associated to it. The Payload data comprises of the user id as well as its type. The type property

defines the role of the user. On decoding the Payload, if the type property is “ADMIN” only then

- **Firebase Implementation**

The Filestore feature of Firebase is only implemented at the frontend. The firebase folder consists of the configuration of the Firebase console and the app related to the project. Similarly the config is exported and is used in the Services part of the frontend application

Services (Client)

The services at the client side comprise mainly of –

· Authentication

The file exports various functions to login, logout, send API requests all related to authentication and authorization. The functions are stored in an object which is then exported.

· Upload Service is used to interact with firestore and upload images to the cloud

· Edit Complaint service is used make API request to bring about changes in the complaint

· Various Get Data services are used to make API calls to request and fetch data from the backend regarding transporters, locations and complaints to be displayed on the webpage

- **Middlewares and Authorisation.**
  Each request that is made from the frontend to the backend is first authorized and checked before the entering the controllers part of the backend route. Authorization is carried out using the help JSON Web Token middleware. The token which is generated only on login is associated for a particular time period. If the token is corrupted or not valid, the middleware catches the error and sends the message to the frontend part. This disallows the flow of control to reach the data modification or database interaction part of the application and prevents any unauthenticated or unauthorized user to access functionalities beyond his/her privileges
- **Backend Routes**
  The controllers and services of the app.

Every backend route is associated with a middleware for verification and then the control shifts to a particular controller which in turn accesses a required service to performs the task. The controller then returns the result received from the service to the frontend for completion of request.

All routes in the backend are “POST” method calls.

The following routes and its functions are discussed below –

· User Dashboard Route

This route checks the validity of the token and returns an appropriate message to the frontend of the user token is corrupted or not.

· Upload Complaint

This route is used to upload the complaint details entered by the user. It creates a new document in the complaint database and creates a new complaint in the database.

· Get Location Data

This route is used to get the names of all the locations that are present in the locations collection and send them as an object array to the frontend

· Get Transporter Data

This route is used to send the documents present in the transporters collection as an array of objects to the frontend.

· Get Complaint Data

This route is used to send the documents present in the complaints collection as an array of objects to the frontend where the ‘reportedBy ‘property is equal to id stored as a payload of the JSON Web Token associated to the request.

· Get Admin Complaint Data

This route is used to send the documents present in the complaints collection as an array of objects to the frontend

· Report

This route is used to send the documents present in the complaints collection as an array of objects to the frontend where the createdOn property is within the date range specified as the query parameter.

· Update Complaint Data

This route is used to update any change made by the admin user to a particular complaint. The changes are directly reflected in the database.

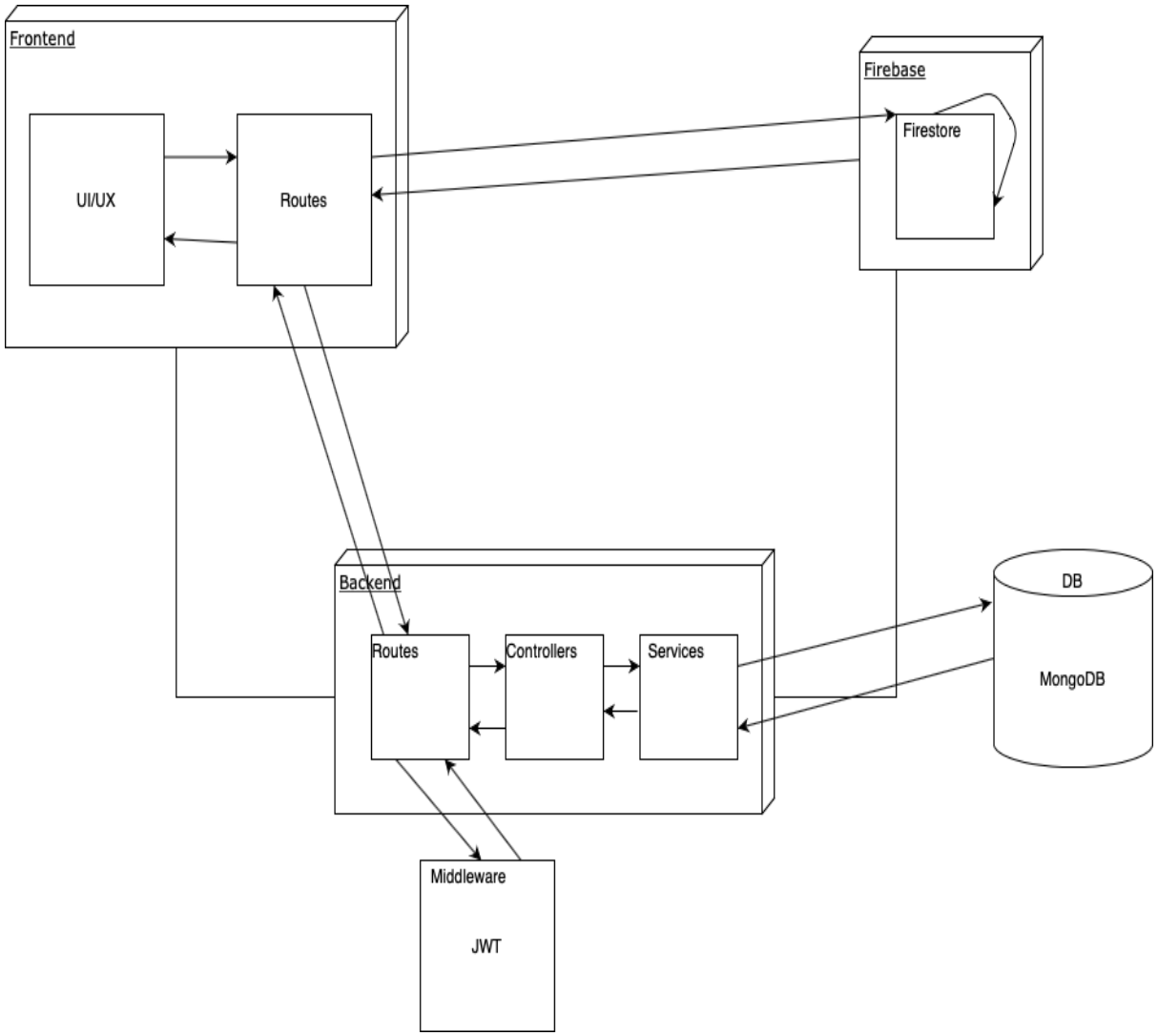

Fig 1: The Process Flow Diagram

Architecture

The project is implemented across two dedicated servers. The frontend server and the backend server to be precise. This helps us to achieve a very stable and secure architecture which is load balanced. The two different servers can accept a large number of requests parallelly and interact without much latency. Hence this provides smooth functioning of the entire complaint registration portal.
The web application architecture can be termed a server to client and client to server architecture type and follows the basic principles of a 3 - tier client server architecture.
The basic path followed by any request to the system is discussed below. A request is directed towards the frontend server. It is handled by the various components and is checked for the user type using the custom-built routes and decoding the JWT token. Once a valid token is checked for, only then that request can trigger a particular functional component on the frontend. Depending on the trigger, a fetch request is sent to the backend where all the routes are present. These routes have a middleware layer to provide double security and check for malware etc. Once the request is validated at the backend, the control is passed to the controllers and services which make the final updates and a response in the form of a JSON object is sent back to the frontend server and the output page is rendered.

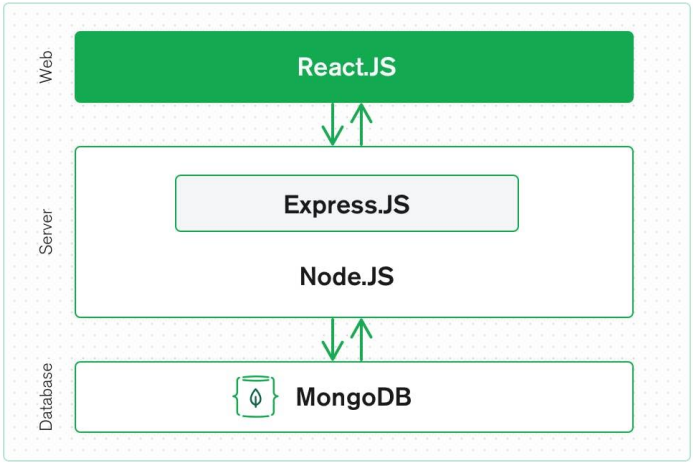

Fig 2 : The 3 Tier Architecture