



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

### PROJECT REPORT

#### CSE1901 - TECHNICAL ANSWERS FOR REAL WORLD PROBLEMS

Summer Semester Special 2021-2022

#### App Based Road Safety Violation Recording System

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

## **DECLARATION BY THE CANDIDATES**

We hereby declare that the project report entitled “App Based Road Safety Violation Recording System” submitted by us to Vellore Institute of Technology, Vellore in partial fulfillment of the required for the award of the degree of BACHELOR OF TECHNOLOGY in SCHOOL OF COMPUTER SCIENCE DEPARTMENT is a record of the bonafide project work carried out by us under the guidance of Prof. Deepa K. We further declare that the work reported in this project has not been submitted and will not be submitted, either in partial or full, for the award of any other degree or diploma in this institute or any other institute or university.

**Signature of the candidates**

**K.Singh**

**Kulvir Singh  
(19BCE0833)**

**V.Khara**

**Vardhan Khara  
(19BCE0833)**

**A.Srivastava  
Anitej Srivastava  
(19BCE0835)**

## INDEX

Serial Number	Topic
1.	Abstract
2.	Introduction to the identified work
3.	Problem Statement
4.	Prevalence and occurrence of the problem
5.	Proposed Methodology
6.	Architecture
7.	Output Screenshots
8.	Server Snapshots
9.	Comparison Charts
10.	Results and Discussions
11.	References

## **Abstract**

During our daily lives, road travel is something that is constant and the most widely performed activity by people of every class. Due to the large population of people that travel on the roads, road safety is an increasing concern among people. But there is a very less amount of people that address this concern technically/scientifically. While travelling on the roads of India, if an individual faces an issue or within the campus of an organisation, they generally have no avenues available to them to report the issue immediately, effectively and without hassle. In this project we aim to address this issue and find a solution that is effective not only in India but anywhere, where road travel is involved. We aim to develop a system which users can instantly access to register their issues and customize it according to the issue they are facing. The system would be flexible to a plethora of issues with equal ease. The admins or the people concerned with addressing the issue would also have a seamless interface to deal with all the complaints that have been lodged in order to increase productivity.

## **Introduction**

The project work deals with the implementation of a complaint registering system as a web application. The aim of the project is to develop a user friendly and easily interactable web based portal to log a complaint against any vehicle/transporter who is not abiding by the rules and guidelines of the company. It also includes the feature of viewing a complaint and its respective details along with its resolved status. The portal is supported by an admin feature which enables certain authorized users to interact and edit the status of the logged complaint. The main objective of the project was to develop a functioning website which has all the aforementioned features to make the complaint logging system digitalized and hassle free. This project also speeds up the complaint logging and complaint resolving process and also ensures a certain level of data and user security.

## **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.

## Literature Survey

Research Paper	Proposed Work	Gaps Identified
<p>Analysis of Road Accidents in India and Prediction of Accident Severity By Sajal Jain, Shrivatsa Krishna, Saksham Pruthi, Rachna Jain &amp; Preeti Nagrath, 2021</p>	<p>In this paper, an attempt has been made to study the various factors associated with a road accident and its effect on the cause and severity of the accident by analyzing the road accidents occurring in the nation of India from 2000 onwards. In this paper, decision tree classifier has been implemented for the prediction of the severity of a road accident.</p>	<p>This paper mainly aims at providing the reasons and causes for major road accidents in India. It does not aim to provide any solutions for the same. Also, the model used here to predict the road accidents severity is based on decision tree classifier model and is only 80% accurate.</p>
<p>A Framework for Analysis of Road Accidents By Shristi Sonal, Saumya Suman, International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR), 2018</p>	<p>In this paper, the road accident data analysis uses data mining and machine learning techniques, focusing on identifying factors that affect the severity of an accident. It is expected that the findings from this paper would help civic authorities to take proactive actions on likely crash prone weather and traffic condition</p>	<p>Although this paper provides comprehensive machine learning based techniques to analyze the road accidents, it does not take into account the characteristics of humanity and behavior which are also important in road accidents. This paper fails in providing a better solution to the road safety issue or even discussing some new approaches that could be used to tackle the issues identified.</p>
<p>Francesc Soriguera, Enric Miralles, Driver Feedback Mobile App, Transportation Research Procedia, Volume 18, 2016</p>	<p>In this paper the human factor in driving and its consequences for road safety have been discussed. The paper provides relevant information on the driver in terms of a grade depending on the aggressiveness and risks taken while driving. Different driving contexts have been explored and smartphone sensors have been identified applying cluster analysis to the measurements, and treated independently giving us an idea on</p>	<p>The solution the paper provides is a bit too dependent on the sensors available on the smartphone rather than on cars. The availability of these sensors could vary depending on the regions the driver is located in. The smartphone is a battery constrained device and the sensor usage in a long journey could affect feasibility of the solution.</p>

	how to approach the road safety problem.	
Apostolos Ziakopoulos, Dimitrios Tselentis, Armira Kontaxi, George Yannis, A critical overview of driver recording tools, Journal of Safety Research, Volume 72, 2020	This paper explores traditional survey methods like questionnaires, police reports, and direct observer methods initially, followed by investigating issues pertinent to the use of driving simulators. The paper provides an extensive section for naturalistic driving data tools, including the utilization of in-car diagnostics (OBD) and in-vehicle data recorders. The in-depth incident analysis and the usage of smartphone data is useful in approaching a solution.	The paper acts as more of a review paper than providing a better solution to the road safety issue or even discussing some new approaches that could be used to tackle the issues identified. The paper does not provide a method for reducing the high costs identified in the study for using new technologies in order to tackle road safety issues.
Milan Tešić, Elke Hermans, Krsto Lipovac, Dalibor Pešić, Identifying the most significant indicators of the total road safety performance index, Accident Analysis & Prevention Volume 113 2018	A wide range of road safety indicators have been explored in this paper. Road safety performance index (RSPI) obtained on the basis of a larger number of safety performance indicators (SPIs) have been found to enable decision makers to more precisely define the earlier goal-oriented actions. We learn about the need for calculating a road safety performance index before coming up with a better approach.	The paper does not address the variation in safety performance index or other performance indicators based on different regions. For a country like India, a lot more factors need to be considered for determining a general safety performance index for the entire nation. Similarly, the paper fails to address the adjustment of various factors for various countries.
Guo, D., & Onstein, E. (2020). State-of-the-art geospatial information processing in NoSQL databases. ISPRS International Journal of Geo-Information, 9(5), 331.	This paper has worked on the geospatial storage of data in a NoSQL based language. Geospatial information has been previously stored in a relational database such however this paper has discussed the value and efficiency of processing the data by storing it on a NoSQL database. The paper discusses the implementation and correspondence of applications	The paper has given insight and has motivated us to work with NoSQL databases to build our app and store the location data while recording any sort of road safety violation. However the use case model which we want to incorporate cannot be sufficed by geospatial storage itself. We will be including a part of the storage feature along with our own data supplemented with the spatial

	that use NoSQL DBs and its difficulty of using a different relational DB for geospatial data	data
Anand, D., Khemchandani, V., Sabharawal, M., Cheikhrouhou, O., & Ben Fredj, O. (2021). Lightweight technical implementation of single sign-on authentication and key agreement mechanism for multiserver architecture-based systems. Security and Communication Networks, 2021.	The authors have stated the importance of authentication in any application. The paper discusses in detail about single sign on authentication which is based on the key agreement or signature based agreement. The use of a single authentication verified with a key across various devices hosted on multiple servers is discussed. The authors have also thrown light about the applications which are hosted on multiple servers and communicating between them is also authenticated using SSO	The paper speaks in depth about SSO however for our application which tends to solve high traffic and high latency issues across an app server, will make it very tedious and time consuming. Instead of an SSO if we can apply a middleware which will help us communicate with all the servers in the multi server architecture and validate each request individually. This will help in balancing the load on the frontend servers which will allow a good user experience and the backend can use middleware based token authentication to provide security.
Mehra, M., Kumar, M., Maurya, A., & Sharma, C. (2021). MERN stack Web Development. Annals of the Romanian Society for Cell Biology, 25(6), 11756-11761.	This paper states that the MERN stack is excellent for building a complete web system. This paper looks at four components of the MERN stack (MongoDb, Expresses, ReactJs & NodeJs) and how well they work together, their beauty as a complete stack in web design. This paper focuses exclusively on the functions of these four MERN stack technologies and how they are applied to current popularity.	This paper has only talked about MERN Stack based applications. In our project, we will be using MERN however for generating a globally located image storage, a Firebase based FileStore will be used. Also to provide a free, easy and interactive UI/UX Tailwind CSS will be used for the frontend. Express and React servers will be hosted on 2 different servers.

## Prevalence and occurrence of the problem

Road safety happens to be a primary concern all over the world considering the fact that it affects every individual nation, people within the nation and consequent impact on the economy and public health of the nations across the globe. Major reasons for accidents are unruly traffic and bad condition of roads, absence of sign board, unauthorized median cuts etc. Thus, preventive measures should be taken and modern technology can be used to address the issue. As per the

data published by the Ministry of Road Transport and Highway, India, the total number of accidents increased by 2.5 percent within the span from 2014 to 2015 and the rate of deaths because of accidents is 400 deaths every day. Another report on Road Accidents in India 2016, published by Transport Research wing under Ministry of Road Transport & Highways, Government of India, has revealed that more people died on roads accidents in India last year, as compared to the number of deaths in 2015. The data has further revealed that the states of Uttar Pradesh and Tamil Nadu have accounted for maximum number of deaths.

## **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.

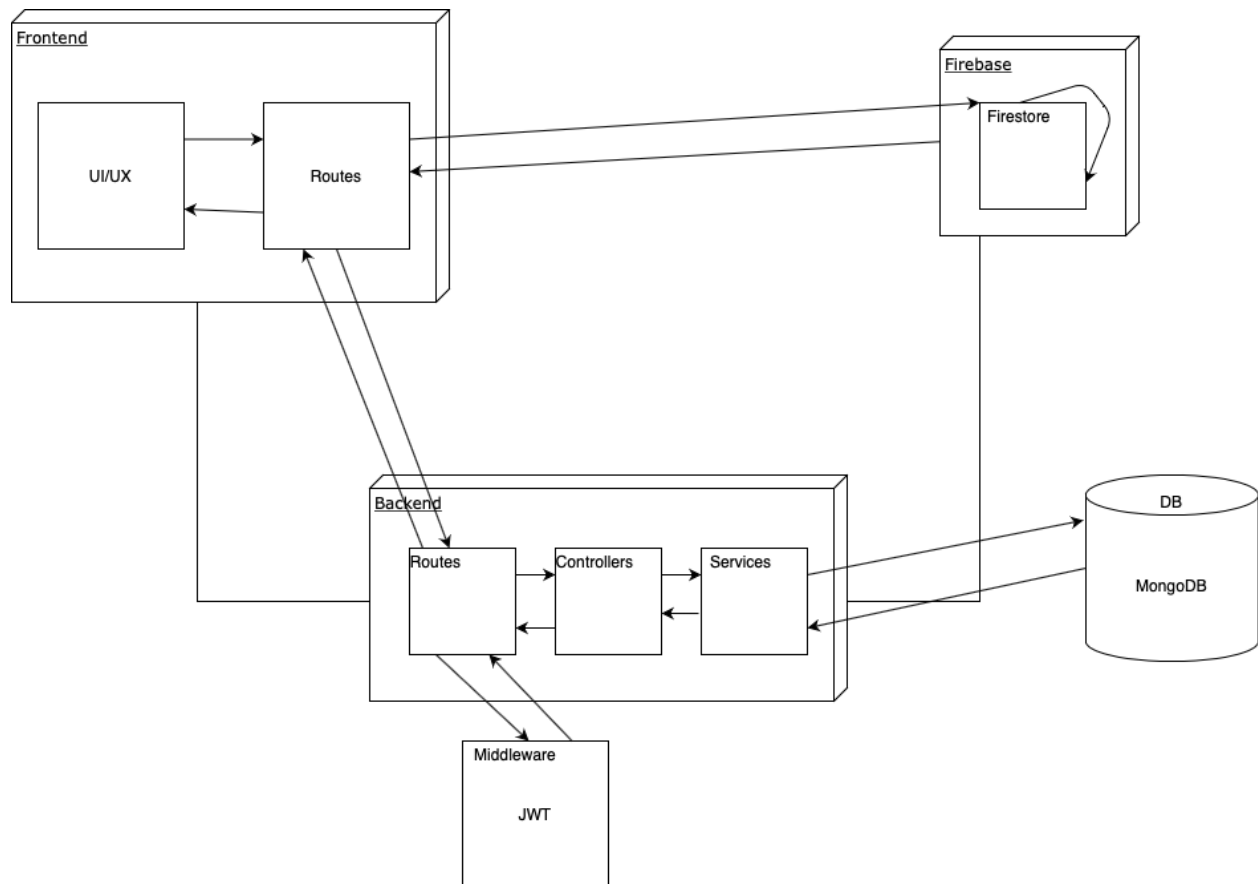


Fig 1: The Process Flow Diagram

## 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.

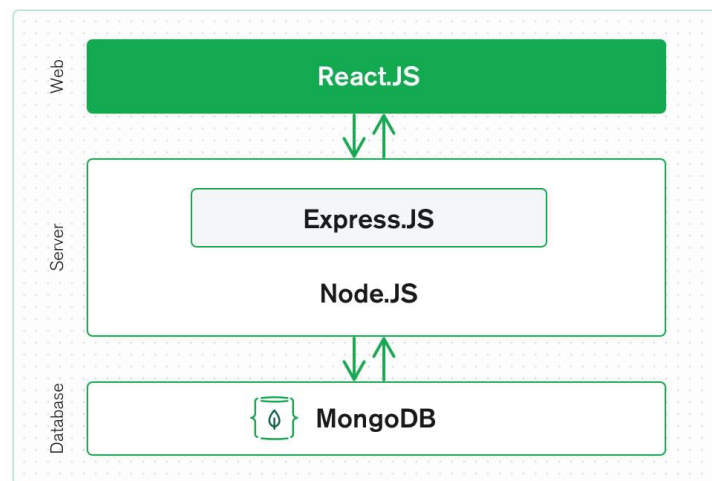


Fig 2 : The 3 Tier Architecture

## Modules

### 1. 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 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 the features of the component will be rendered.

The following pages fall under the Admin Component –

- Admin page

This page asks the admin type user to enter search parameters to look for a particular complaint to be edited

- Edit Complaint page

This page asks the admin type user to enter new status and new remarks for the selected complaint and also displays the existing details of the selected complaint.

## 2. 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

## 3. Middleware and Authorization

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.

## 4. Backend Routes

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 if 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.

#### API Specifications

- **‘/login\_user’** This POST method checks if the user id and password present in json data sent as a body of the request is valid or not and returns message accordingly
- **‘/user\_dashboard’** This POST method checks if the access-token is present as a header of the request and returns a json accordingly
- **‘/upload\_complaint’** This POST method creates a new document in the complaints collection and fills the fields of the document by extracting information from the json sent as body of the request.
- **‘/get\_location\_data’** This POST method returns the documents present in the locations collection as an array of objects
- **‘/get\_transporter\_data’** This POST method returns the documents present in the transporters collection as an array of objects
- **‘/get\_complaint\_data’** This POST method returns the documents present in the complaints collection as an array of objects of the logged in user only
- **‘/get\_complaint\_data\_admin’** This POST method searches for the documents according to the search parameters given in the body as JSON object and returns those documents present in the complaints collection as an array of objects
- **‘/update\_complaint\_data’** This POST method updates the specified document of the complaints collection according to the data sent as the body of the request.
- **‘/report’** This POST method returns the documents present in the complaints collection as an array of objects according to the date range specified in the body of the request.

#### 5. Database Design

The entire database has been hosted on the mongoDB cloud service – Atlas. A total of 4 tables or collections have been made to interact with. The schema for each collection has been written in 4 separate files kept in the models folder of the server directory.

##### Collection 1 – locations

This collection has been made to enter all the possible locations inside the works where a fault or a complaint can be registered. The schema comprises of :

1. location – type: object

This object has a sub property called coordinates which is an array of type Point. It stores the longitude and latitude of the location. It also checks if the longitude and latitude stored is a valid one or not

2. name – type: string

This stores the name of the location

#### Collection 2 – transporters

This collection has been made to enter all the transporters that are registered to the company. The schema comprises of :

1. name – type: string

This stores the name of the transporter

2. id – type: string

This stores the unique id that has been given to the transporter in the master database of the company

#### Collection 3 – users

This collection has been made to store all the users that are permitted to access specified or all features of the application. The schema comprises of :

1. id – type: string

This stores the unique id given to every employee of the company which can be used to login and identify the user of the application.

2. password – type: string

This stores the password associated to the respective user id in encrypted form.

3. type – type: string [“VIEWER”, “ADMIN”]

This stores the accessibility or role of the user. It enables the website to figure out if a user has ADMIN privileges or VIEWER privileges.

4. lastLogin – type: date

This stores the date and time of the user’s most recent login to the application.

5. createdOn – type: date

this stores the date and time when the document was first added to the collection

#### Collection 4 – complaints

This collection has been made to store any complaint that is uploaded by the user.

The schema comprises of :

1. imageUrl – type: array

This stores all the URLs of the images uploaded by the user while logging a complaint. The file is stored in the firestore cloud storage service of Google.

2. Status – type: string [“OPEN”, “CLOSED”]

This stores the status of the complaint. OPEN is the complaint has not been resolved and CLOSED if the complaint has been resolved.



3. Remarks – type: string

This stores the remarks added by the admin about the complaint.

4. reportedBy – type: string

This stores the id of the user who has reported the complaint.

5. reportedLocation – type: string

This stores the name of the location where the fault has occurred.

6. description – type: string

this stores the description of the fault and describes the complaint raised.

7. vehicleNumber – type: string

This stores the vehicle number of the vehicle which has not followed the protocol of the company

8. transporterName – type: string , optional

This stores the name of the transporter to which the vehicle belongs

9. transporterCode – type: string , optional

This stores the id of the transporter to which the vehicle belongs

10. comments – type: string

This stores the comments of the user while registering the complaint

11. createdOn – type: date

This stores the date and time when the complaint was created by the user.

```
const complaintSchema = new mongoose.Schema ({
  // id: { type: String, required: true },
  reportedBy: { type: String, required: true },
  reportedLocation: { type: String, required: true },
  description: { type: String },
  imageUrl: [String],
  vehicleNumber: { type: String, required: true },
  transporterName: { type: String },
  transporterCode: { type: String },
  comments: { type: String },
  createdOn: { type: Date, default: Date.now },
  status: { type: String, enum: ["OPEN", "CLOSED"], default: "OPEN" },
  remarks: { type: String, default: "None" }
});
```

**Fig 3 : Complaint Schema**

```
const transporterSchema = new mongoose.Schema({
  name: { type: String, required: true },
  id: { type: String, required: true }
});
```

Fig 4 : Transport Schema

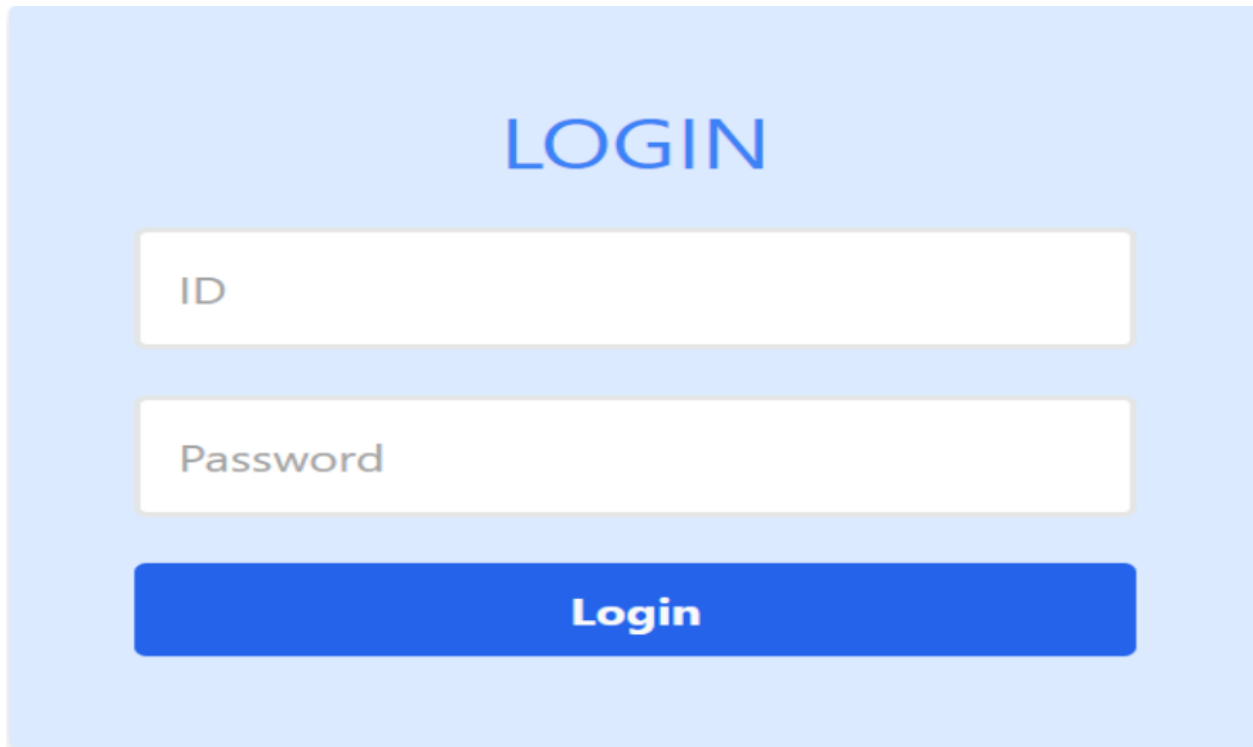
```
const locationSchema = new mongoose.Schema({
  name: { type: String, required: true },
  location: {
    type: {
      type: String,
      enum: ["Point"],
      required: true
    },
    coordinates: {
      type: [Number],
      required: true
    }
  }
});
```

Fig 5 : Location Schema

```
const userSchema = new mongoose.Schema ({
  id: { type: String, required: true } ,
  password: { type: String, required: true } ,
  type: { type: String, enum: ["ADMIN", "VIEWER"], required: true } ,
  lastLogin: { type: Date, default: Date.now },
  createdOn: { type: Date, default: Date.now }
  // currentToken: String,
  // currentTokenTimestamp: Date
  // modifiedOn: Date
});
```

Fig 6 : User Schema

## Output Screenshots



A screenshot of a login page with a light blue background. The word "LOGIN" is centered at the top in a large, blue, sans-serif font. Below it are two white input fields with thin grey borders. The first field is labeled "ID" in a grey font, and the second field is labeled "Password" in a grey font. At the bottom is a solid blue button with the word "Login" in white, bold, sans-serif font.

LOGIN

ID

Password

Login

**Fig 7. Login Page**

WELCOME

Dashboard

View Complaints

Report

Admin

Logout

REGISTER COMPLAINT :

Reported Location

Description

Vehicle Number

Transporter name-id

comments

Choose Files

No file chosen

Upload

**Fig 8. Dashboard page**

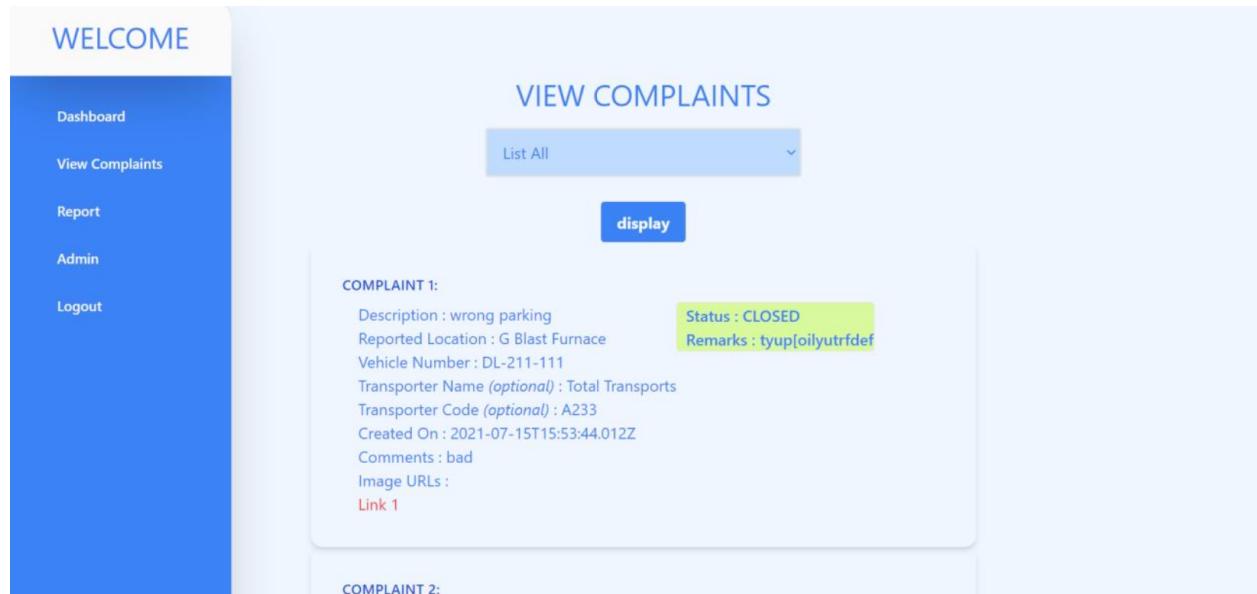


Fig 9. View Complaints Page

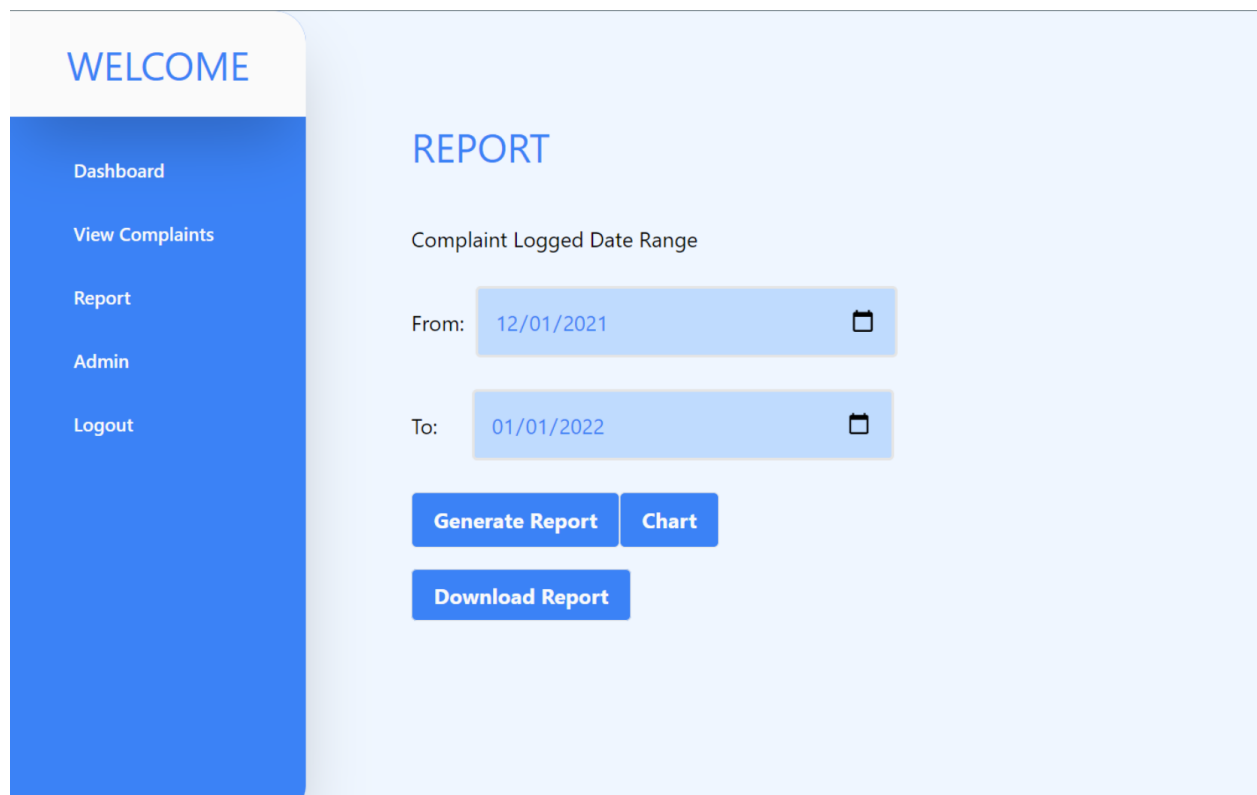


Fig 10. View Report Page

## EDIT COMPLAINT

**COMPLAINT ID = 60F97A4771F1AD329CE3BF6E**

Reported By : abc123  
Description : jjj  
Reported Location : CRM  
Vehicle Number : kkk  
Transporter Name *(optional)* : MAZDA & YADAV ASSOCIATES  
Transporter Code *(optional)* : K033  
Created On : 2021-07-22T14:01:43.757Z  
Comments : ooo  
Image URLs :  
Link 1

Status : OPEN

Remarks : None

ADD REMARKS

Choose a status ▼

Edit Complaint

**Fig 11. Edit Complaint Page**

## SERVER SNAPSHOTS

```
> nodemon --exec babel-node app.js

[nodemon] 2.0.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `babel-node app.js`
Server listening on Port - 3001
http://localhost:3001
connected to db
```

**Fig 12. Backend Server**

```

> client@0.1.0 start C:\Users\kulvir\Desktop\internship\TataSteel\PROJECT\client
> craco start

[wds] : Project is running at http://192.168.80.1/
[wds] : webpack output is served from
[wds] : Content not from webpack is served from C:\Users\kulvir\Desktop\interns
hip\TataSteel\PROJECT\client\public
[wds] : 404s will fallback to /
Starting the development server...

Compiled with warnings.

src\App.js
  Line 3:34:  'Link' is defined but never used      no-unused-vars
  Line 3:40:  'NavLink' is defined but never used      no-unused-vars
  Line 3:64:  'Redirect' is defined but never used      no-unused-vars

src\pages\Complaint-View-Dashboard.jsx
  Line 4:8:  'ReactDOM' is defined but never used      no-unused-vars

src\pages\Edit-Complaint.jsx
  Line 17:7:  React Hook React.useEffect has missing dependencies: 'complaintId'
and 'getComplaint'. Either include them or remove the dependency array      react-h
ooks/exhaustive-deps

src\pages\Login.jsx
  Line 1:27:  'Component' is defined but never used      no-unused-vars
  Line 2:26:  'Router' is defined but never used      no-unused-vars
  Line 2:34:  'Link' is defined but never used      no-unused-vars
  Line 2:40:  'NavLink' is defined but never used      no-unused-vars
  Line 2:49:  'Switch' is defined but never used      no-unused-vars
  Line 2:57:  'Route' is defined but never used      no-unused-vars
  Line 2:64:  'Redirect' is defined but never used      no-unused-vars

src\pages\Register.jsx
  Line 1:27:  'Component' is defined but never used      no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

```

Fig 13. Frontend Server

## COMPARISON CHARTS

	A	B	C	D	E	F	G	H	I	J	K
1	Created On	Status	Remarks	Reported	Reported	Description	Vehicle No	Transport	Transport	Comments	
2	2021-07-1	OPEN	ghasdsdg	asd	G Blast Fu	wrong par	DL-211-11	Total Tran	A233	bad	
3	2021-07-1	OPEN	asdqewra	asd	H Blast Fu	fff	sss			qqq	
4	2021-07-2	OPEN	None	abc123	CRM	jjj	kkk	MAZDA &	K033	ooo	
5	2021-07-2	OPEN	hgfgsdadf	asd	CRM	oooo	pppp	MAZDA &	K033	qqqq	
6	2021-07-2	OPEN	remark ne	asd	TSPDL	zzzz	xxxxx	Kailash Ro	34112	cccc	
7	2021-07-2	OPEN	None	abc123	CDC	wfghjgf	sgfhfdghj	Magadh A	12322	ghmgjhgj	
8	2021-07-2	OPEN	None	abc123	Safety Dep	jhfdsdrfv	yugjdfgda	Konkan Er	W042	ghfdfgasdrf	
9											
10											

Fig 14. Report

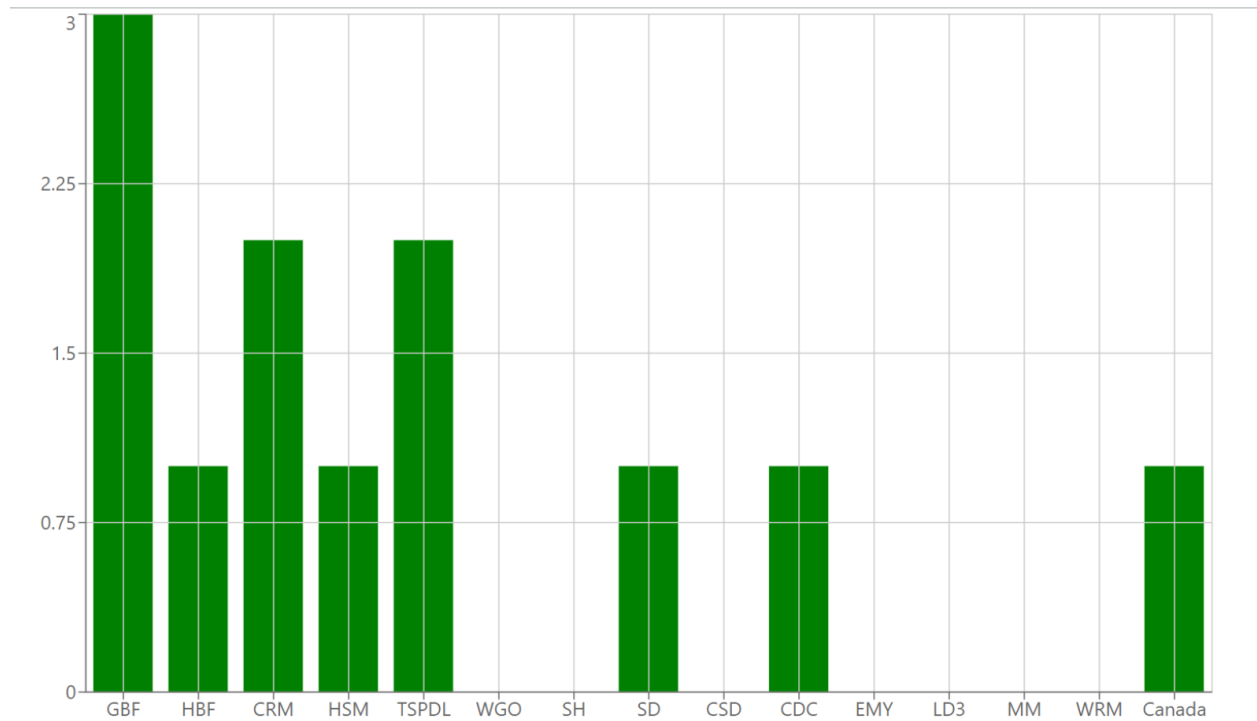


Fig 15. Bar Chart Analysis

#### Overall Analysis

1. The green ring suggests the number of complaints distributed across various departments.
2. The inner ring gives the count for the number of open complaints to the closed one.



Fig 16. Pie Chart Analysis



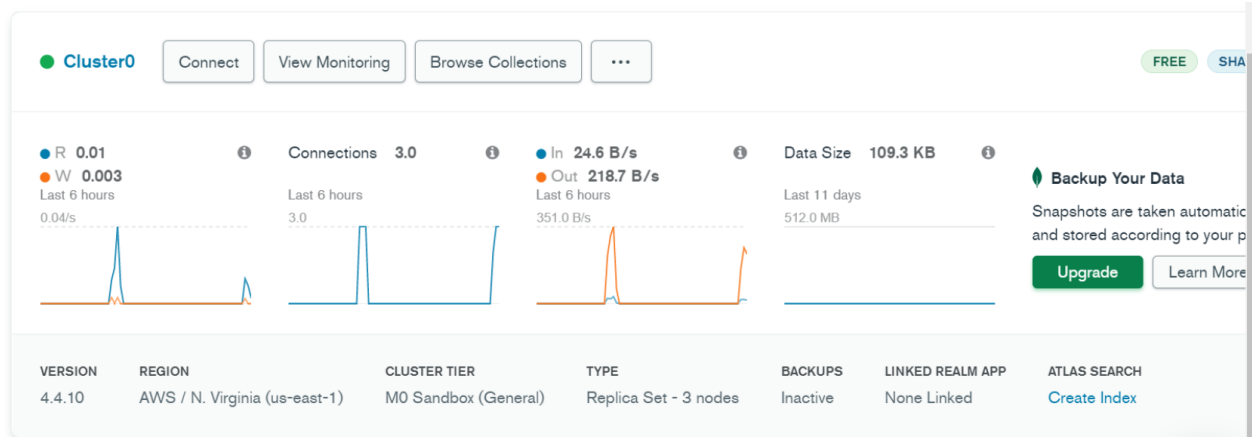


Fig 17. Atlas Utility Chart

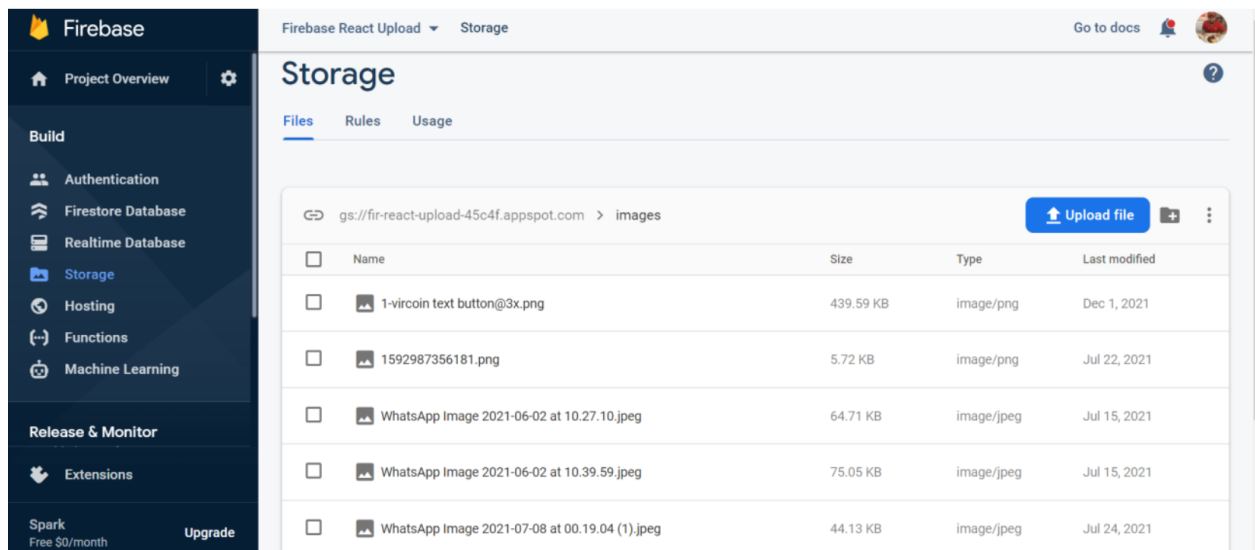


Fig 18. FireStore Storage

## Results and Discussions

The App Based Road Safety Violation Recording System uses secure and efficient servers through Firebase to facilitate fast and reliable storage/retrieval of files. It uses multiple servers to serve up the data to the user on demand and is active 24/7. Load balancing is provided due to the usage of multiple servers as well as backup and redundancy. Hence the failure of one server would not result in a system crash. These properties result in the system being highly scalable for various users.

The usage of Java Web Tokens (JWT) ensures data confidentiality, integrity and authentication. Availability is ensured by the servers. These four properties make the system feasible for industry usage.

The application comprises various state-of-the-art tech stacks due to which it is highly versatile. The UI/UX enable the user to get acquainted with the system in a minimum amount of time and makes the application highly intuitive to use. This property would make the cost of building and running this application for a small number of users (50-100) is NIL since the entire tech stack is based on open software. The future cost of running and maintaining the application would depend on how the application will require scaling and the costs pertaining to the web hosting platform which are minimal. These costs will not grow exponentially with the increase in the number of users. Hence the development and maintenance of the application is economically feasible for a small as well as a large organisation.

The major chunk of costs would arise from extra storage that would be required i.e. expanding the MongoDB Atlas capability for the application while scaling.

The multiple servers being used are bound to increase costs but their benefits to various organisations will cover those costs and beyond. The revenue model for the application would charge organisations depending on the amount of resources they use rather than the time for which they use.

At a broad level, such frameworks to assess transport policies, plans and programmes tend to focus on key environmental issues, while the assessment of social impacts is in many cases indirect, qualitative or simply reflected in the appraisal or planning assumptions.

Such an application will affect the social realm of the society in a beneficial manner as the road complaint system will provide an open platform for users to notify the authorities. Also, the responsiveness and accountability will increase as the public will be able to keep a check of the actions taken for improvement, if any.

In terms of public convenience, since it is an online system, it is socially feasible. It is much better than the existing offline complaint registry system. If the user comes across any road safety violation, they can immediately inform the concerned authorities through this app, rather than waiting in lines or doing the same thing through exhausting phone calls. Moreover, not only will it be socially feasible, it will also be desirable, as the pros outweigh the cons. It will be convenient, user friendly, authoritative, reliable and versatile.

An Environmental Feasibility Study assesses the viability of a proposed development from an environmental and social perspective, identifying potential issues and threats to the successful completion of the proposed development. Solutions and mitigative measures are investigated.

In terms of the environmental impact, it by far, does not have any implications on a large scale that will negatively affect the environment. As it is an online system, no physical stress is involved. Additionally, it saves the environment as it compels the stoppage/reduction of pollution, deforestation and harm to animals. It provides a platform where all the negative elements hurting nature can be kept under check. It will help diminish the negatively impactful activities and those not directly involved will also be benefited from the app. When it comes to an actual environmental risk, its job is to eliminate the environmental risks by securely ensuring the safety and compliance of rules which are laid out for the road safety and its violations. Finally, it provides “Environmental Sanitation” and “Environmental Sustainability”, which means activities aimed at improving and maintaining the standard basic environmental conditions affecting the well being of people.

The application solves a social problem of road safety recording using computer engineering paradigms and is not affiliated to any particular political, social or religious groups. The application is suitable for any political climate since it solves a common problem for the people of India and does not discriminate based on caste, creed, class and religious beliefs. The documentation and the UI is clear in communicating its requirements and constraints to the user which make it very difficult to be misrepresented. Hence this application is unlikely to be influenced politically or have a biased influence on an individual or any organisation.

This section comprises the various demographic factors involved in the usage and the implications of the same. It includes age, race, ethnicity, gender, marital status, income, education, and employment. To start off with describing the demographics of the user market, we can say that the majority of the users of the app will be vendors, production factory owners, logistics service providers, truck owners and drivers. The platform is most suited for them as they will be provided with a platform to deliver their grievances. There are no race and ethnic particularities for the app. However, the primary users are generally males in their 30s through early 50s, with low to medium average monthly income. They are usually school pass outs and some of them are college graduates. Since it will be a free application, it will be easily accessible for almost all demographics of users. It requires a smartphone which is no longer a luxury. The

factory owners and logistics services providers comprise the economically elite class of users of the app and thus, it is feasible for them as well. Also, the app provides a straightforward platform to add and solve complaints, and hence, there are no backlashes or demographic complications involved since it has no bias whatsoever.

## References

- [1] Imran Ashraf, Soojung Hur, Muhammad Shafiq Catastrophic factors involved in road accidents: Underlying causes and descriptive analysis, Yongwan Park, 2019
- [2] Sajal Jain, Shrivatsa Krishna, Saksham Pruthi, Rachna Jain & Preeti Nagrath, Analysis of Road Accidents in India and Prediction of Accident Severity, 2021
- [3] Shruti Sonal, Saumya Suman, A Framework for Analysis of Road Accidents, International Conference on Emerging Trends and Innovations In Engineering And Technological Research (ICETIETR), 2018
- [4] Francesc Soriguera, Enric Miralles, Driver Feedback Mobile App, Transportation Research Procedia, Volume 18, 2016
- [5] Apostolos Ziakopoulos, Dimitrios Tselentis, Armira Kontaxi, George Yannis, A critical overview of driver recording tools, Journal of Safety Research, Volume 72, 2020
- [6] Milan Tešić, Elke Hermans, Krsto Lipovac, Dalibor Pešić, Identifying the most significant indicators of the total road safety performance index, Accident Analysis & Prevention Volume 113 2018
- [7] Guo, D., & Onstein, E. (2020). State-of-the-art geospatial information processing in NoSQL databases. ISPRS International Journal of Geo-Information, 9(5), 331.
- [8] Anand, D., Khemchandani, V., Sabharawal, M., Cheikhrouhou, O., & Ben Fredj, O. (2021). Lightweight technical implementation of single sign-on authentication and key agreement mechanism for multiserver architecture-based systems. Security and Communication Networks, 2021.
- [9] Mehra, M., Kumar, M., Maurya, A., & Sharma, C. (2021). MERN stack Web Development. Annals of the Romanian Society for Cell Biology, 25(6), 11756-11761.

