MSE907.2.pdf

作者为 User .

提交日期: 2025年07月03日 08:54上午 (UTC+0700)
提交作业代码: 2709487285
文档名称: MSE907.2.pdf (2.07M)
文字总数: 6024
字符总数: 35762

MSE907.2 – Work in Progress Report (WIP)

SafeRoads Navigator:

A System for Real-Time Road Hazards Monitoring Using Crowdsourced Data and Dual Vetting Approach

Arnold Aristotle Tayag

Student ID: 270559700

Introduction

Road safety remains a pressing public health and transportation concern worldwide. Traffic crashes are one of the leading causes of death and injury, accounting for approximately 1.19 million fatalities globally each year, of which, 92% occur in low– and middle–income countries (WHO Report, 2023). In New Zealand, the NZ Transport Agency reports 341 road fatalities and 2,442 serious injuries, averaging one death and seven serious injuries per day in 2023. This makes New Zealand's rate of road deaths amongst the worst in OECD countries (NZTA, 2024), with a rate of 6.5 deaths per 100,000 population (NZAA, 2025). Road hazards such as potholes, heavy rain & flooding, debris, poor signage, and other safety risks pose serious threats to motorists, cyclists, & pedestrians alike. Traditional approaches to identifying and fixing these hazards rely on periodic inspections or public complaints, which are often slow and reactive. Swift awareness of road hazards can significantly reduce secondary accidents and improve emergency response times (Young et al. 2019).

Crowdsourced data has emerged as a promising solution to provide real–time, widespread coverage of road conditions. Such community–driven reporting can fill critical information gaps, especially for hazards that might not trigger immediate official action from road authorities, i.e. a fallen tree branch or localised flooding. Despite their benefits, crowdsourced hazard systems face challenges regarding the reliability and accuracy of user–contributed data (Leal et al. 2017). False or outdated reports can occur, whether due to user error, changes in conditions, or even malicious misuse. Ensuring data integrity is essential for maintaining user trust and effectiveness of the system. There is a need for more robust verification to minimize false alarms while

preserving the speed and breadth of crowdsourced inputs. Thus, the need for a dual vetting solution is essential.

This capstone project, **SafeRoads Navigator**, proposes a novel approach to real–time road hazards monitoring by combining the strengths of crowdsourced data with a dual vetting approach. In this system, hazard reports from the public are subjected to two layers of vetting: first by road users through peer confirmations (aka Peer–vetting in the context of this study), thru upvotes or flags and second by road safety authorities (aka Authority–vetting) to ensure higher accuracy. The goal is to create a reliable, user–driven road hazard alert network.

Background and Problem Statement

There is a need for a real–time road hazard monitoring system that maximizes the advantages of crowdsourced data (speed and breadth of coverage) while minimizing misinformation through effective validation. Additionally, there is currently no comprehensive, real–time platform for road hazard management that connects the general public, road maintenance authorities, and city planners on a common information system. As a result, many road hazards either go unreported, unassessed, or are addressed too slowly, leading to avoidable accidents, infrastructure damage, and inefficiencies. Traditional reporting channels (phone hotlines, periodic surveys) are reactive and often fail to capture the full scope or urgency of issues. Furthermore, data that is collected (through isolated apps or studies) remains under–utilised for long–term safety improvements due to fragmentation and lack of analytical tools.

The core problem then can be summarised as: "How can we design and implement a real–time road hazard alert management system using crowdsourced reports enhanced with a dual–layer vetting process to deliver timely and trustworthy information to all stakeholders?" The absence of an integrated solution means that cities struggle to prioritize repairs effectively, road users lack awareness of known danger spots, and decision–makers do not have consolidated data on which to base infrastructure improvements. We need a system that not only gathers hazard reports from road users in real time, but also ensures those reports are reliable, promptly acted upon, and aggregated into meaningful insights for future road safety strategies.

SafeRoads Navigator directly addresses these problems by providing a unified platform for road hazards reporting, visualization, alerting, and analysis. By doing so, it bridges the communication gap between citizens experiencing road hazards and the authorities responsible for fixing them, while also creating a valuable data repository for planners and researchers for present and future use. In essence, the project seeks to transform the current reactive and fragmented approach into a proactive, data–driven, one–stop–shop, road hazard management process.

Objectives of the Project

The objectives of SafeRoads Navigator are defined to address the identified problems / research gaps stated. The successful completion of the capstone project will be measured against the three objectives outlined below:

1. To design and implement a real–time road hazards system that introduces a mechanism (1) for road users to report road hazards and verify (by upvoting or flagging) existing hazard reports, and (2) allow road safety stakeholders to filter out false or malicious submissions without introducing perceptible reporting delays, thus improving the accuracy and trustworthiness of crowdsourced road hazard data.
2. To develop a unified, end–to–end road hazard management platform prototype that integrates (a) crowdsourced road hazard reporting, (b) multi–layer verification workflows, (c) operational response coordination, and (d) strategic trend–analysis capabilities.
3. To embed interactive data analytics dashboards and open–data interfaces within the hazard management system—enabling stakeholders, i.e. city planners, maintenance teams, & researchers, to generate policy–relevant insights, export cleaned datasets, and to assess the platform's impact on decision–making efficiency and possibly, inter–agency collaboration.

Literature Review

We all want to make transportation greener, smarter and more sustainable. However, road authorities everywhere face increasing challenges in managing traffic and infrastructure to provide critical services to citizens. It is therefore vital that cities have an integrated solution that allows them to monitor and manage traffic infrastructure & transport operations in real time. Numerous studies and projects in

recent years have explored the use of crowdsourced data to improve road safety and traffic management. Early research recognised the potential of social and mobile platforms in capturing real–time traffic information. For example, in Christou et al. (2023), it highlighted that a government–backed, national–level crowdsourcing app can achieve high uptake and rapid response times, which could lay the groundwork for broader smart–city deployments. This work reported that citizen–generated data offered more frequent and geographically comprehensive coverage, particularly on secondary roads, than traditional inspections or vehicle–based surveys. Thier app, FixCyprus—a national mobile app in Cyprus that collects citizen reports of road infrastructure defects, demonstrated the impact of a government–adopted, country–wide crowdsourcing tool. In Olma et al. (2022), it identified that traditional road–safety work in Germany and the EU largely depends on collision black–spot analyses, which are inherently reactive, slow to identify hazards (since collisions are rare events), and *ethically problematic because human harm must occur before action is taken*. They introduced a web–based platform where road users can mark “danger spots” on an interactive map, providing details on the type of hazard and road users at risk, i.e. poor visibility affecting pedestrians. This approach treats the crowd as “underestimated experts”, asserting that daily road users experience conditions first–hand and can alert others and authorities to risks before accidents occur. Indeed, the study found optimism among stakeholders that active public participation could serve as an early warning system in road safety work, though they cautioned about the subjective nature of such reports and potential misuse. It concluded that systematically collected danger–spot submissions, especially those with higher user engagement, can serve as timely, cost–effective indicators for proactive road–safety

interventions. It goes further by saying that danger spots that garnered more user interactions (clicks, supports, comments) were significantly more likely to be confirmed as hazardous, indicating that collective attention reinforces report accuracy. In Telima et al. (2023), it mentioned crowdsourcing as a proactive tool, citing that systematic collection and analysis of user–generated incident reports provide an early–warning mechanism—capturing risks before collisions occur & complementing traditional crash records. It investigated the use of crowdsourced incident reports (including not only collisions but also near–misses and infrastructure issues) to analyze pedestrian safety in urban areas. One novel result of the project was the identification of a few pedestrian hazard locations in the area of study. A complementary study by Ibtissem et al. (2022) was proposed in Tunisia to leverage data from multiple crowdsourcing platforms (mobile+web) for urban road safety improvements. The R–Secure system features two main functionalities: first, collecting rich road anomaly data (including images and GPS location) from citizens, and second, integrating expert input and analyses to enhance road safety outcomes.

In parallel to manual user reports, technology is being used to automatically detect road hazards, which can complement crowdsourcing. An innovative approach by Cafiso et al. (2022) used bicycles and e–scooters as probe vehicles to monitor road pavement conditions via built–in smartphone sensors. By collecting accelerometer data as these micro–mobility devices traversed urban roads, the researchers could identify road surface anomalies. Notably, the e–scooter and bike data were able to detect various severities of cracks and even potholes that did not register in traditional car–based road condition surveys.

There were also more focused studies done on the subject of road safety. One in particular is an IoT–based pothole detection by Desai et al. (2024) where they developed an application that aimed at revolutionizing road safety by proactively identifying and preventing accidents caused by potholes. This research leverages the integration of Internet of Things (IoT) technology to provide real–time updates and alerts to drivers, significantly mitigating the risks associated with pothole–induced accidents. This was achieved by combining LiDAR, radar, GPS, & accelerometer data to produce reliable pothole detections across varying road conditions & speeds. A similar study was done in Bhoyar et al. (2023). In this study, they devised a pothole detection system that utilizes an ultrasonic sensor mounted on a vehicle to measure the depth of potholes. The sensor continuously measures distance to the road surface and readings exceeding an 8cm threshold are flagged as a potential pothole. One major limitation of this study, however, is that it solely focused on pothole detection. In Yang et al. (2021), the research employed technology that collects pothole data using smartphone accelerometers and cameras mounted on vehicles to obtain images of road surface anomalies. The images collected are then analysed with an image recognition–based convolutional neural network model to determine if the road surface scanned contains potholes. The final verified pothole images are then transmitted to a server along with the vehicle's GPS data. Then, the system generates road hazard information in the form of a four–level index indicating the road risk. This road hazard index can better assist road safety authorities in establishing road maintenance plans and planning their costs. In a similar study, Pena–Caballero et al. (2020) focused on a limited scope of road hazard classes, i.e. Manhole, Pothole, Blurred Crosswalk, & Blurred Street Line. It trained a segmentation model, using 1,700

annotated images from the over 4,000 images captured, that recognizes the four classes mentioned above and achieved great results with this model allowing the machine to effectively and correctly identify and classify the four classes in an image. Another recent study is the use of semantic web and ontologies to manage road hazard information. Kindo et al. (2024) introduced an ontology–based system to handle community–reported potholes, aiming to formalize how pothole data is described and shared between different systems. This ontology–based, community–assisted system bridges the gap between grassroots pothole detection and formal municipal maintenance, offering a scalable, cost–effective, and interoperable solution for improving road safety. Main limitation is that it focused primarily on pothole detection despite the many types of potential causes for road hazard incidents.

A body of literature has focused on evaluating the accuracy of crowdsourced road hazard data. In one study, danger zones were chosen from among four German cities. Thereafter, a quasi–randomised audit of 77 user–reported danger spots was done and over half were confirmed to have infrastructural deficits, i.e. true hazards, while about one quarter were classified as “uncertain” due to insufficient details. These findings (user reports and on-site inspections) were then compared with actual police collision data, Olma et al. (2022). In a related study, R–Safety, devised a mobile crowdsourcing platform that allows road users to report their “road safety feeling indexes” in real time. Their platform focuses on subjective safety perceptions and even lets users report traffic violations they witness, with the goal of informing citizens about risky areas through a vulnerability map (Khedher et al. 2022).

While the related literature presented thus far demonstrates clear benefits to using crowdsourced road hazard information, it also

consistently highlights challenges that SafeRoads Navigator aims to address—data quality control. First, an open contribution model, i.e. crowdsourced, raises some concerns about misinformation. Also, these community–driven validations can be slow or insufficient, i.e. areas with few users. The present study suggests that combining community validation with an additional layer—moderator oversight (aka dual vetting)—could improve reliability.

The literature shows a strong foundation for SafeRoads Navigator. Crowdsourced data is a powerful tool for real–time road hazard monitoring, offering speed and coverage advantages over traditional methods. Multiple studies have proven that user reports are generally reliable (with low false alarm rates) and can even save lives by enabling faster responses. It can also provide rich datasets for analysis. However, to fully realize this potential, a system must tackle data redundancy and trustworthiness as well. These insights directly inform the SafeRoads Navigator project. The proposed system builds on the success of crowdsourcing seen in prior research and explicitly addresses the identified gaps by introducing a structured dual vetting process for quality assurance. In the next section, I will define the specific problem this project will solve in light of the literature discussed and outline our research questions.

Identified Gaps

Several gaps in current solutions become evident. On one hand, crowdsourced reporting can dramatically improve the timeliness and coverage of road hazard information. On the other hand, ensuring the

accuracy and credibility of these reports remains an open challenge. Current navigation apps, for example Waze, & research prototypes largely rely on single–layer crowd validation—if enough users report or confirm an incident, it is considered credible. This approach, while useful, has its limitations. Minor hazards may not get multiple confirmations before they cause harm, and false reports can linger if no moderator intervenes. There is also a gap in seamlessly integrating crowdsourced hazard data with verification mechanisms in a way that is fast & reliable. Also, community engagement features such as crowdsourced verification, voting, or commenting are not universally present. The EDDA+ danger spot platform did implement support/upvote mechanisms and discussion threads (Olma et al., 2022), but many other systems do not leverage the crowd beyond the initial report. This can lead to issues with data quality and prioritization: for instance, without community feedback, authorities might not know which of the many reports are most urgent or if some reports are duplicates. Thus, there is a gap in using the crowd not just as reporters but as filters and enhancers of the data.

Second, many crowdsourced hazard reporting systems focus on one aspect of the problem. Some are focusing only on one particular type of road hazard, i.e. pothole detection. Some are primarily for reporting and notifying authorities, i.e. FixCyprus focuses on submitting reports to public works (Christou et al., 2023), while others emphasize collecting data for analysis or public awareness, i.e. the danger spots map in Germany for safety research (Olma et al., 2022). There is a lack of an integrated platform that seamlessly supports end–to–end hazards management—from immediate reporting and verification, through operational response, to strategic trend analysis—in a unified system.

Finally, the siloing of data is a concern. Data collected by one platform often isn't readily accessible for other purposes. The korean citizen science study had to perform significant data processing (text mining) on the complaints to evaluate maintenance efficiency (Kim et al., 2023), implying that the system used to collect complaints wasn't inherently providing those insights. Likewise, danger spot data needed to be combined with collision and kinematic data externally to derive hazard scores (Olma et al., 2022). This suggests a gap in MIS design: hazard reporting platforms could better incorporate data analytics and open data principles from the start. **SafeRoads Navigator** will attempt to close this gap by including dashboards and data export features that make analysis easier and allow stakeholders (planners, researchers) to directly use the information.

The gaps identified are:

1. Reliability of crowdsourced data – how to filter out incorrect or malicious reports without losing the benefits of real–time community inputs. The literature indicates trust mechanisms are needed, but an effective implementation in the road hazard domain is lacking;
2. Lack of a unified system covering reporting, response, and analysis;
3. Limited use of crowdsourced verification to improve road safety initiatives as data is not readily analysed or shared for policy use.

These gaps inform the problem statement and objectives of this capstone project.

Research Questions / Hypothesis

To tackle the stated research gaps above, the project will be guided by the following research and design questions:

1. How can real–time trust and reputation mechanisms be designed to automatically validate and filter crowdsourced road–hazard reports without significantly delaying the reporting process?
   Justification: This addresses the first gap identified which is about increasing the reliability of crowdsourced data by minimizing false or malicious submissions.

2. What architectural and workflow models enable the seamless integration of an end–to–end road hazards management within a single platform?
   Justification: This addresses the second gap identified by creating a unified solution covering the aspects of crowdsourced reporting, multi–layer verification, operational response, and strategic trend analysis.

3. In what ways can built–in data analytics and open–data interfaces be embedded into crowdsourced hazard reporting systems to facilitate immediate policy–relevant insights and stakeholder collaboration?
   Justification: This addresses the third gap enumerated above by providing data analytics and enabling the proposed solution to export datasets on road hazard incidents to be used by city planners, researchers, maintenance teams, and other concerned relevant parties.

Methodology / Approach

This capstone project will employ a design and development methodology typical of software engineering projects, enriched with elements of design science & action research. The approach can be outlined in several phases:

1. **Requirements Analysis** – based on the literature review and the defined use cases, I will detail the functional requirements (what the system should do) and non–functional requirements (performance, security, usability, etc.). This will involve creating use case diagrams and user stories, i.e. "As a driver, I want to quickly report a road hazard with a photo so that others are warned and it gets fixed". I will also identify the data requirements—what information each report will contain (hazard description and type, location, timestamps, reporter info, etc.) and how data flows between frontend, backend, and any external services.

2. **System Design and Architecture** – using the requirements identified above, I will design the system architecture. This will follow a client–server model: the frontend application will communicate with a backend Node.js Express server via RESTful API calls using JSON data. I will design API endpoints for operations such as submitting a new report (`POST /report-hazard`), retrieving hazard data (`GET /fetch/hazards`), voting on a report (`POST /upvote-hazard`), and admin actions (`POST /admin/approve-hazard` to approve a hazard report, `POST /admin/reject-hazard` to reject a reported hazard, etc.). A

relational database schema (RoadHazard.sqlite) will be designed to store users, hazard reports, votes, admin actions, geospatial information and any other related data. The Google Maps API will be incorporated for the map interface; this requires obtaining an API key and the Heatmap layer for density visualization. A particular focus in design will be on the dual vetting logic, i.e. designing how a hazard status transitions from "Received/UnderReview" to "Resolved or Rejected" and what triggers that (number of confirmations or admin approval).

3. **Implementation (Iterative Development)** – as an agile practitioner, I will adopt an agile iterative development methodology, aiming to produce incremental prototypes that can be tested and refined. For instance, in the first iteration, I will implement core reporting and mapping without all features, then gradually add verification, dual vetting, alerts, analytics, and reports in subsequent iterations. This approach allows for early feedback loops for prototyping, modifications, and fixes.

4. **Testing and Evaluation** – throughout development, I will test each component. This includes unit testing of backend logic and integration testing. I will also perform user acceptance testing: simulate a user reporting a hazard and ensure it appears on the map; simulate multiple users voting on a report and see that it escalates appropriately; test that an alert email is sent when it should, etc.

5. **Project Management and Milestones** – I will break the work into tasks with a rough timeline with expected deliverables for

each timeline (see Project Plan below). Version control (Git) will be used to manage code, and regular progress meetings with the Supervisor will ensure I stay on track. If certain aspects prove too complex, i.e. implementing a fully automated clustering algorithm for alerts, I will adapt by simplifying the approach.

Tools and Technologies Used

Following a modular architecture approach, the project will employ the specified technology stack below:

1. Frontend
    - HTML5 & CSS3 – the client–side user interface is built using modern HTML5 and CSS3. This ensures the development of a responsive design and accessibility.
    - Bootstrap 5 – this component was extensibly used throughout the project due to its pre–built UI components. This made the development of the responsive layouts and grid systems a breeze. The use of bootstrap icons provided enhanced visual interfaces.
    - JavaScript (Client–side) – Javascript was the logical choice for handling client–side logic, event handling, DOM manipulation, and AJAX calls to backend APIs, due to its tight integration with HTML.
    - Chart.js – this component was used to deliver the functionality of the Analytics Dashboard for delivering the interactive data visualizations you see there, such as the bar and pie charts.

- Google Maps API – this third–party component is used to provide the interactive mapping functionality of SafeRoads Navigator
- Leaflet.js a – this component powers the interactive heatmap in the Analytics Dashboard that allows road users to view hazard locations and concentrations using OpenStreetMap data.
- Socket.IO Client – enables integration with the backend which allows for real–time updates on the dashboard, such as live upvotes, flags, and moderation actions.

2. Backend
    - Node.js – the backend runtime environment was developed using Node.js.
    - Express.js – is a flexible Node.js web application framework that handles HTTP requests, routing, session management, and middleware integration for RESTful API web development.
    - SQLite3 – is used to store application data. This was chosen because it is very lightweight and very ideal for rapid development and ease of deployment. For the photo upload, instead of saving the uploaded image in the database, a directory path is created instead because of the inherent limitations in SQLite.
    - Other backend libraries:
        - Bcrypt – used for secure password hashing
        - Body–parser – used to parse incoming HTTP requests
        - Express–session – used for session management as well as user authentication

- Multer – used for handling file uploads
- Dotenv – used for managing application persistent environment variables
- Nodemailer – used for sending email notifications

3. Security, Privacy, and Best Practices
    - The application stack incorporates robust authentication and authorization using sessions and password hashing
    - Multer and body–parser handle the secure handling of file uploads and form submissions
    - Application variables are managed with dotenv to separate configuration settings from system code
    - Google Maps API key management will be handled securely (not exposing keys in public code)
    - User interfaces will be behind a login
    - All personally identifiable information, such as emails and contact numbers, will be encrypted while user passwords will be hashed using AES-256 crypthographic algorithm.

System Design / Architecture Overview

SafeRoads Navigator is proposed as a web–based application implementing the ideas discussed. This section describes the solution's key features, the technology stack and architecture, and the project plan including timeline and milestones for completion within 3½ months.

The SafeRoads Navigator system will be composed of several integrated components, forming a cohesive solution. On the client side, a React–based web application will serve as Hazard Reporting and

Viewer. On the server side, a Node.js application with an Express framework will act as the API Server and Processor (Figure 1), connected to a database (Figure 2) that serves as the Hazard Data Repository. Surrounding these core components, there will be external integrations: the Google Maps API for mapping, notification services, i.e. Gmail SMTP with Nodemailer, for sending out alerts, and OpenStreetMap Nominatim for geocoding services.

Figure 1: Architecture Diagram

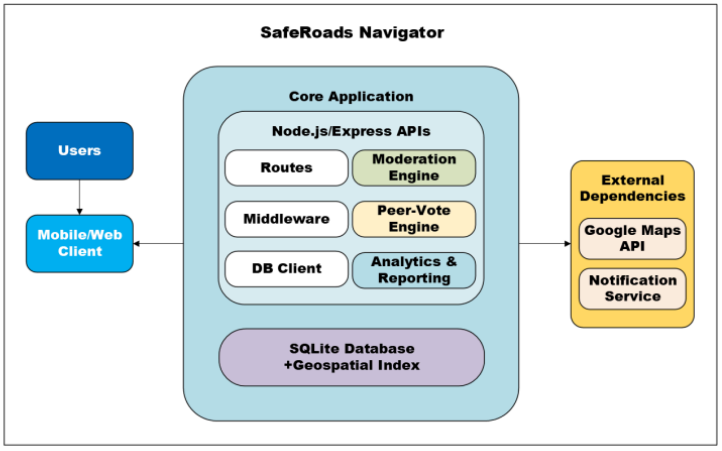
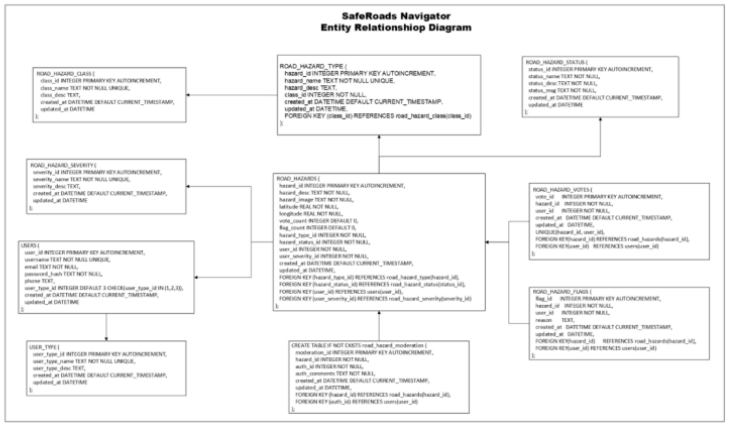


Figure 2: Database Design (ERD)



Road hazards reported will follow a defined process flow and lifecycle as below.

Figure 3: Road Hazard Identification and Treatment (process flow)

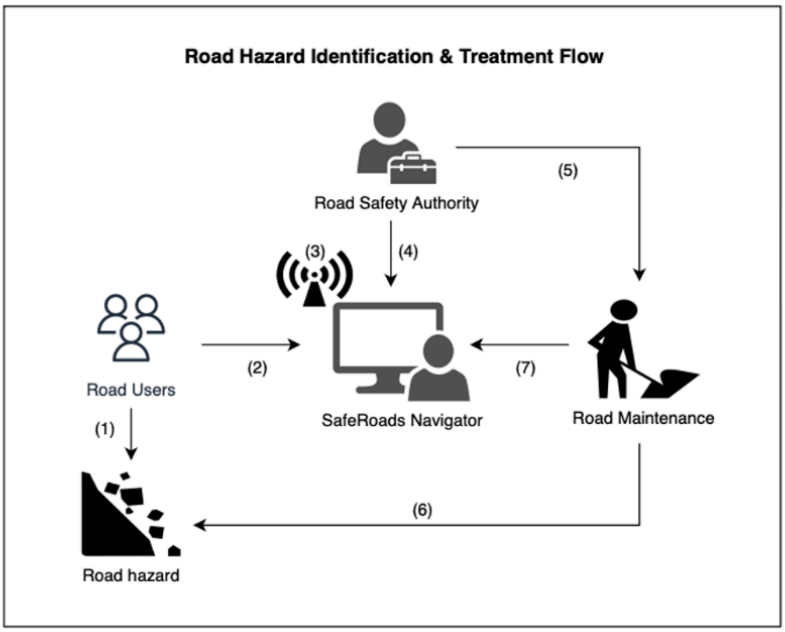
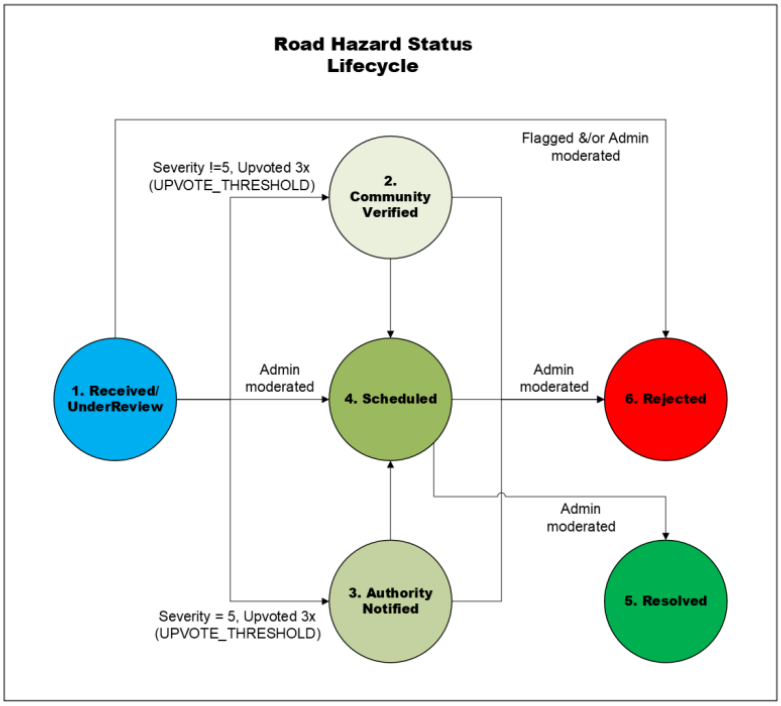


Figure 4: Road Hazard Status Lifecycle Diagram



Implementation Plan / Progress

The application will have the following components:

1. **User Login/Registration** – road users need to register an account to use the system

Figure 5: User Login

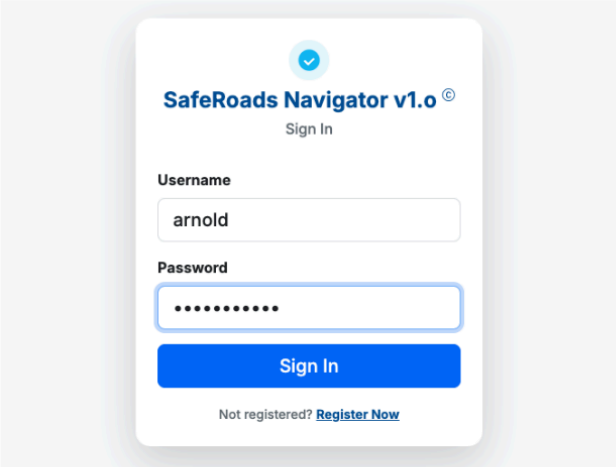
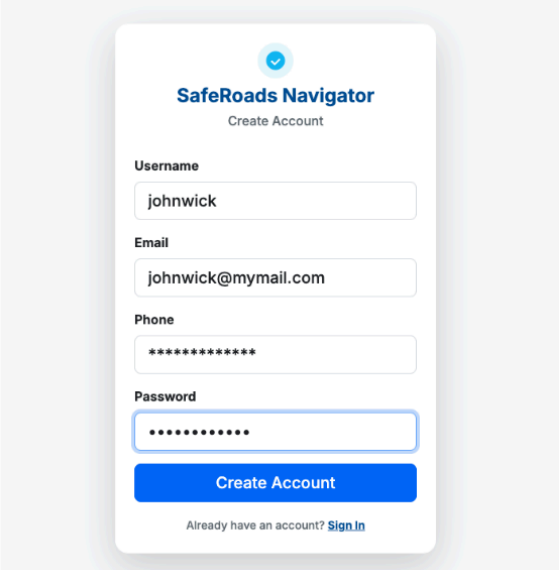
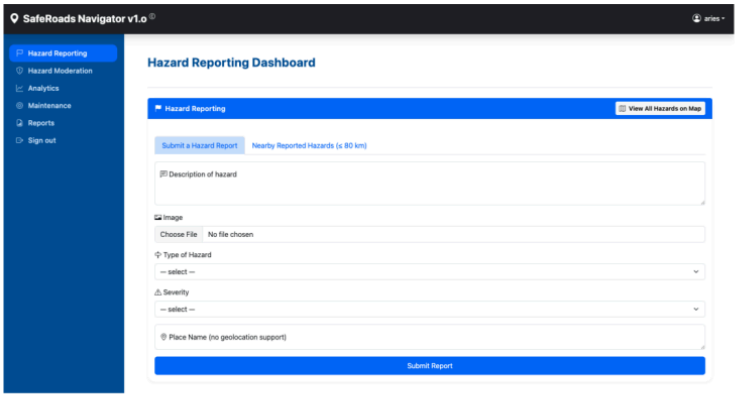


Figure 6: User Registration



2. **Hazard Reporting Module** – the module responsible for posting or reporting a hazard to the system, performing upvotes as well as flagging. This is the main interface for registered road users of the system. Users can submit reports of road hazards through a simple and intuitive web interface. A hazard report includes details such as hazard type, hazard description, the location of the hazard (captured automatically via GPS or entered manually), etc. In addition, they will also see other hazards reported to the system that is within a certain distance from their location.

Figure 7: Hazard Reporting Module



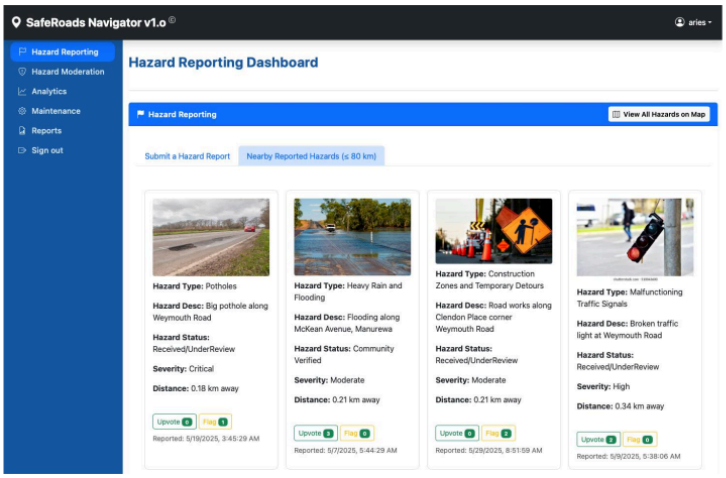
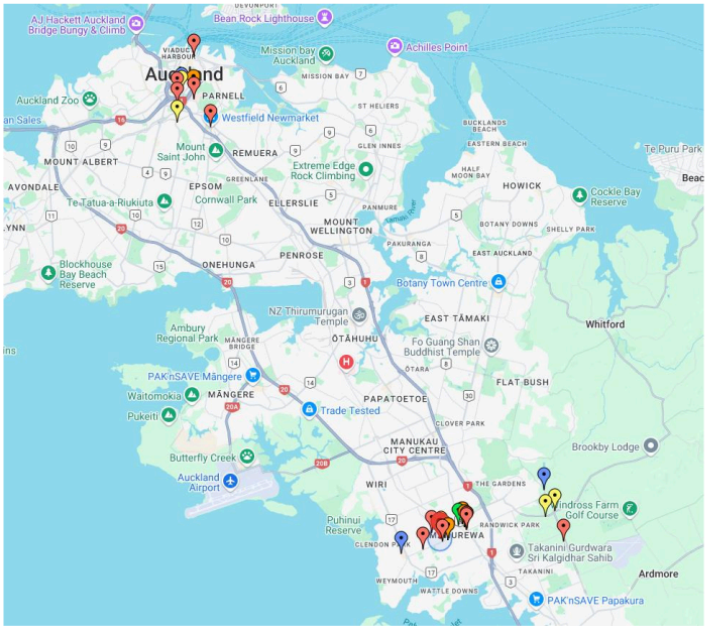

SafeRoads Navigator v1.o
aries
Hazard Reporting
Hazard Moderation
Analytics
Maintenance
Reports
Sign out
Hazard Reporting Dashboard
Hazard Reporting
View All Hazards on Map
Submit a Hazard Report
Nearby Reported Hazards (≤ 80 km)
Hazard Type: Potholes
Hazard Desc: Big pothole along Weymouth Road
Hazard Status: Received/UnderReview
Severity: Critical
Distance: 0.18 km away
Upvote 0
Flag 1
Reported: 5/19/2025, 3:45:29 AM
Hazard Type: Heavy Rain and Flooding
Hazard Desc: Flooding along McKean Avenue, Manurewa
Hazard Status: Community Verified
Severity: Moderate
Distance: 0.21 km away
Upvote 1
Flag 0
Reported: 5/7/2025, 5:44:29 AM
Hazard Type: Construction Zones and Temporary Detours
Hazard Desc: Road works along Clendon Place corner Weymouth Road
Hazard Status: Received/UnderReview
Severity: Moderate
Distance: 0.21 km away
Upvote 0
Flag 2
Reported: 5/29/2025, 8:51:59 AM
Hazard Type: Malfunctioning Traffic Signals
Hazard Desc: Broken traffic light at Weymouth Road
Hazard Status: Received/UnderReview
Severity: High
Distance: 0.34 km away
Upvote 2
Flag 0
Reported: 5/9/2025, 5:38:06 AM

Figure 8: The Interactive Map showing all hazards reported to the system, with emphasis on the ones that are near the road user



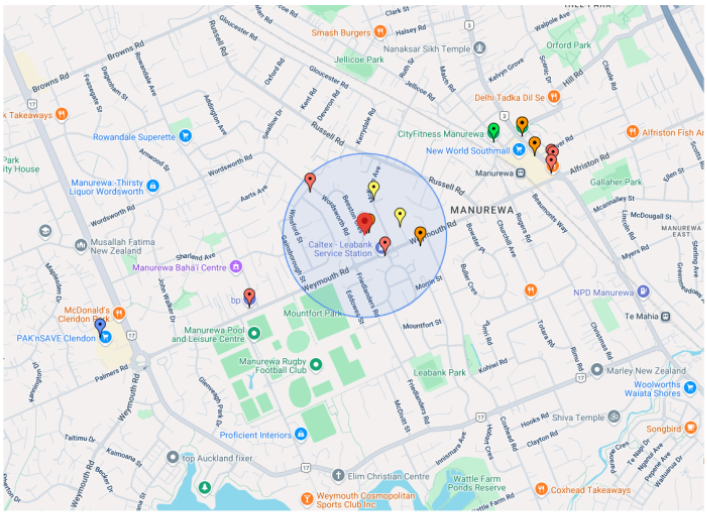
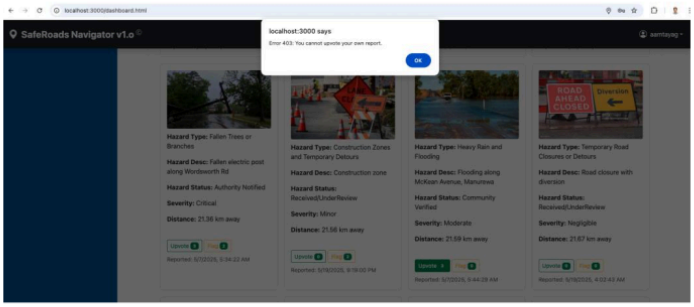
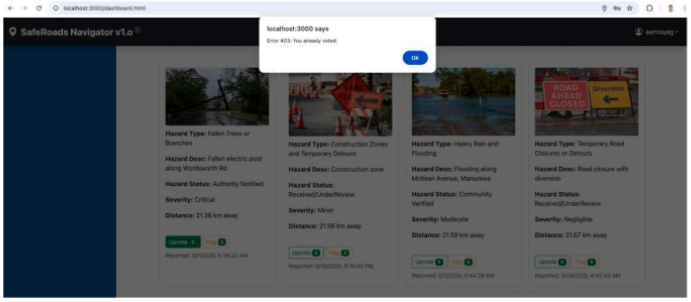


Figure 9: Upvote Restrictions

1. Road user cannot upvote his own report



2. Road user can upvote once per report



3. Road user can only upvote X number of times per day, specified by an environment variable, DAILY_UPVOTE_LIMIT

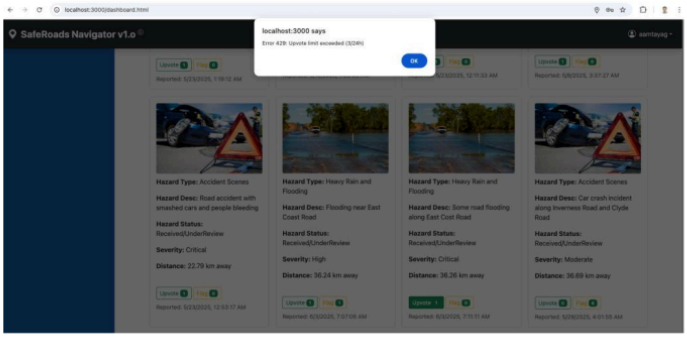
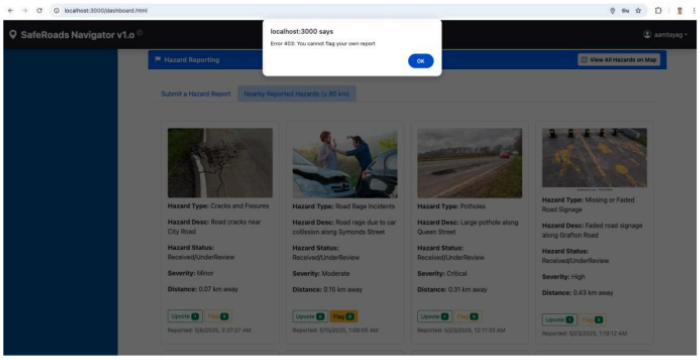
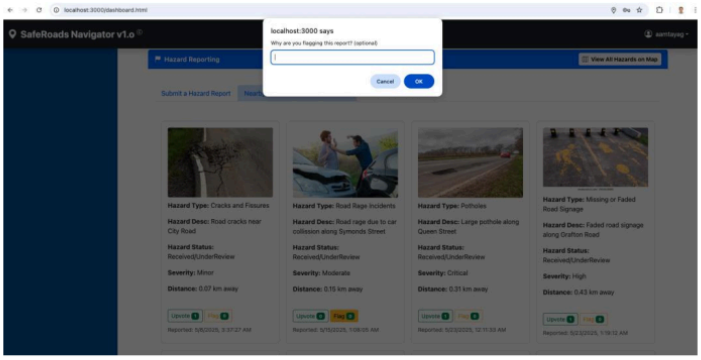


Figure 10: Flagging Restrictions

1. Road user cannot flag his own report

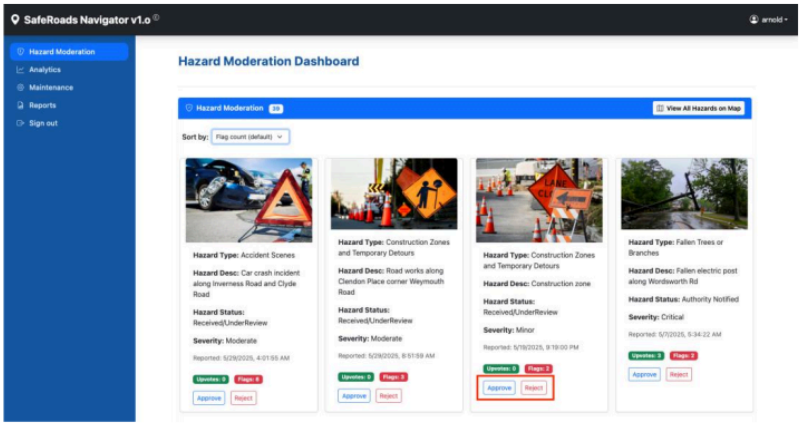


2. Road user must provide a reason for flagging a report



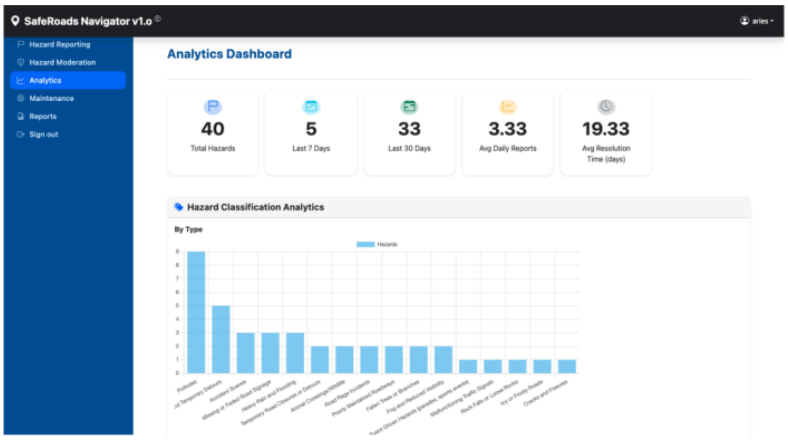
3. **Hazard Moderation Module** – an admin or moderator user (with elevated privileges) can verify or reject reports via a dashboard that lists new unverified reports. The admin can use external judgment or crosscheck, i.e. use local traffic cameras if available, to make a decision (i.e. either approve or reject the reported hazard). Once a report passes this second layer, it is marked as "Scheduled" in the system

Figure 11: Hazard Moderation Module



4. Analytics / Reports Module – the system provides hazard report statistics thru the Analytics dashboard. These stats include average daily reports, average resolution time, hazard classification analytics, location–based stats, user engagement analytics, and trends / insights. The reports dashboard allows moderators to download hazard reports on a weekly/monthly basis as well as resolution rate by hazard types as CSV files

Figure 12: Analytics Module



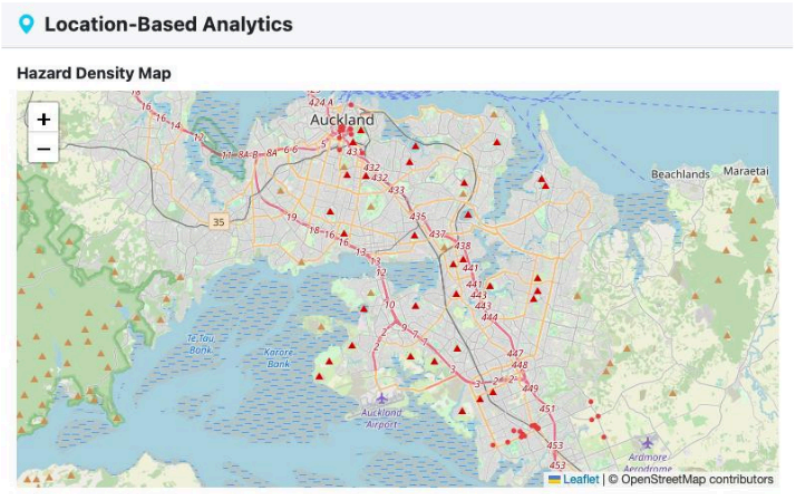
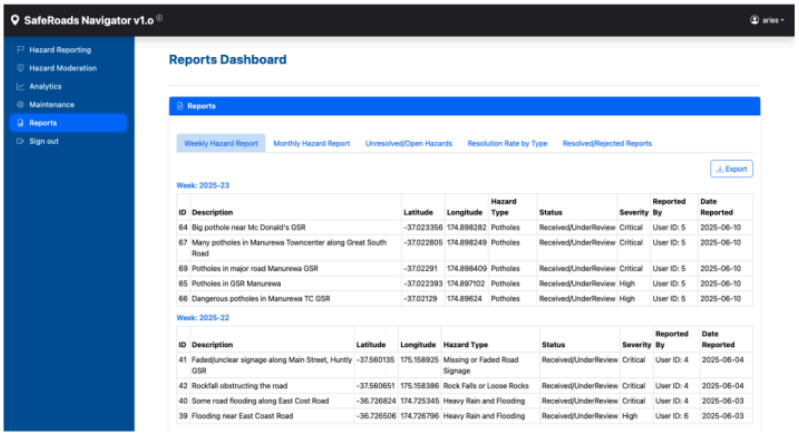
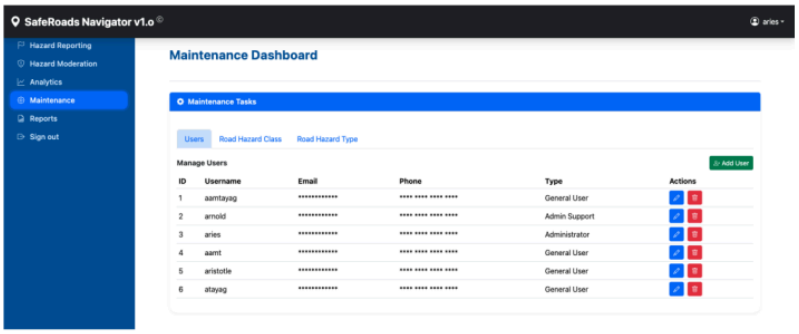


Figure 13: Reports Module



5. Maintenance Module – the maintenance dashboard provides for routine maintenance tasks necessary for the system like user maintenance, road hazard types maintenance, etc.

Figure 14: Maintenance Module



Evaluation / Testing Strategy

The following user acceptance test (UAT) cases have been executed and verified in the development environment:

1. User Login
    - wrong username – Pass
    - wrong password – Pass

2. User Registration
    - missing username – Pass
    - missing email – Pass
    - missing password – Pass
    - missing phone – Pass
    - duplicate username – Pass
    - email and phone are encrypted – Pass

3. Road Hazard Reporting
    - road hazard reporting without description – Pass
    - road hazard reporting without image – Pass
    - road hazard reporting without hazard type – Pass
    - road hazard reporting without severity – Pass
    - road hazard reported is visible in grid and map – Pass
    - manual entry of place name (geocoding feature) – Pass

4. Road Hazard Upvoting
    - road user cannot upvote his own report – Pass
    - road user can only upvote once per report – Pass
    - change RADIUS env variable and hazards report are adjusted – Pass

- road user exceed upvote limit per day (DAILY_UPVOTE_LIMIT) – Pass
- a hazard report with status=1 (Received/UnderReview), severity=5 (Critical), upvoted 3x (UPVOTE_THRESHOLD), status will change to 3 (Authority Notified) and will trigger email notification – Pass
- hazard report with status=1 (Received/UnderReview), severity!=5 (not Critical), upvoted 3x (UPVOTE_THRESHOLD), status will change to 2 (Community Verified) – Pass

5. Road Hazard Flagging
    - road user cannot flag his own report – Pass
    - road user can only flag once per session – Pass

6. Hazard Moderation
    - all sort features are working – Pass

7. Approve road hazard (sort by hazard status)
    - hazard status transition from 1/2/3 to 4 (Scheduled)
        - From 1->4 (Recieved/UnderReview -> Scheduled) – Pass
        - From 2->4 (Community Verified -> Scheduled) – Pass
        - From 3->4 (Authority Notified -> Scheduled) – Pass
    - hazard status transition from 4 (Scheduled) to 5 (Resolved) – Pass
    - road hazard is approved and record is inserted in road_hazard_moderation with comments – Pass

- road hazard is approved and record is inserted in road_hazard_moderation w/out comments – Pass

8. Reject road hazard (sort by hazard status)
    - hazard status transition from 1/2/3/4 to 6 (Rejected)
        - From 1->6 (Recieved/UnderReview -> Rejected) – Pass
        - From 2->6 (Community Verified -> Rejected) – Pass
        - From 3->6 (Authority Notified -> Rejected) – Pass
        - From 4->6 (Scheduled -> Rejected) – Pass
    - road hazard is rejected and record is inserted in road_hazard_moderation with comments – Pass

9. Analytics
    - average resolution computation is good – Pass

10. Maintenance
    - user maintenance tab load correctly and display the add user, edit, delete buttons – Pass
    - add user button is working properly – Pass
    - edit user button is working properly – Pass
    - delete user button is working properly – Pass
    - report hazard class maintenance tab load correctly and display the add user, edit, delete buttons – Pass
    - add class button is working properly – Pass
    - edit class button is working properly – Pass
    - delete class button is working properly – Pass
    - report hazard type maintenance tab load correctly and display the add user, edit, delete buttons – Pass

- add type button is working properly – Pass
- edit type button is working properly – Pass
- delete type button is working properly – Pass

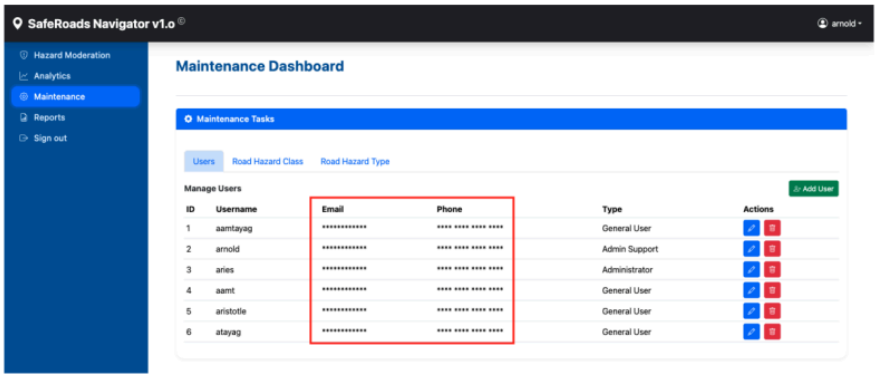
11. Reports
    - weekly report hazard load correctly and can export – Pass
    - monthly report hazard load correctly and can export – Pass
    - unresolved/open hazards load correctly and can export – Pass
    - resolution rate by type load correctly and can export – Pass
    - resolved/rejected reports load correctly and can export – Pass

Ethical, Legal, Cultural Considerations

In the design of the system, several features such as intuitive navigation, ease of use, and responsiveness were taken with careful consideration by making sure it displays correctly in all types of devices, i.e. smartphones, tablets, desktops, laptops. This ensures that users with varying needs can access the platform. In the context of security and legal considerations, all personally identifiable information (PIIs) stored in the system such as emails and phone numbers are encrypted using crypto, a built–in npm package for encrypting user data using AES-256-CBC algorithm. User passwords, on the other hand, are hashed using bcrypt, a javascript–based password–hashing library (see screenshot below).

Figure 15: PIIs in the system are encrypted/hashed

```
sqlite>
sqlite> select current_timestamp;
2025-06-29 21:37:57
sqlite>
sqlite> select username,email,phone,password_hash from users;
aamtayag|00c0c4cce5c35b5f00b05c6dbd3b6703:258fe40840f6257cf57fa1c6e12a3526f0e20980eaf741c857b63a0d4c785214|52a761c9456315726a0533cfab049989
:acf0cbf4f4f89fb77574438821a73c93|$2b$10$ZrR6DHA4PKUINNi4LIEZwOEq.064zhw5cUUN2MdFxd6m9bqSzEmfa
arnold|1855288c42094b33061bc634ac52931e:1b352827e8a1730fc5988428abb8012af199e72cfba82273fb314f59a3627c6e|b75fbaea44f042d7a5ec80e413480a1a:5
798416d81be760d266e7c599752abf2|$2b$10$CIHpCnBNZIFdOV/5bj0p9OmcgGHonKM0fdWVMKIRZTqeYErDrbbx6
aries|739fdf54533aa5147f90a75b7507b00f:f7f9dd81d1b4373d24b6527bc7c38d1ace3355676acdbf66a743a6d92541458e|e94611d4dcd9da73a2821a6a6f450864:e0
aebb3bde94b572fbc7dd1541d9281e|$2b$10$2h183cDhe50KcpzJPn9avu7pA6b6rUkHbmj7fNHG/ddgQfL4Wp9.m
aamt|73c60774f6bfa5a15c312125369dbeb3:098bc8b6829df97856b5574b7eb8199f82100d949d2f03ea2d72c07a03f90a99|2bea8c660af85321029e4874af6e9fb6:906
3a6259b3f379f12df4da63d753f71|$2b$10$s.QSDWMWQto.SvIaul2jFOfuY0BRJG3nkLUPBP0m0pGJo/kK7PzGq
aristotle|d36dbbc6bfab7ef00d862f2ffc8c4767:e9c59738918da10ce1d3e3888737032bc9350db0c6f4a7ecf2053dcc3a51d89e|ab597ffa4cda4a223ce34bd37af589c
8:5acc68521a32b590faaee2223fc794d1|$2b$10$/KIHu3iaLTXhId6unIrd0ebrINqpBPoGQ2AeRfzzFqSfXidaYZy26
atayag|390305020936bf36694059324e37477e:ea7f9ceafa09087af766c4b4a878b43d3c1760234adbae3bf25432e30116ed44|08fa76718cd50b28bf1cd6f608f8061b:7
6e73a28698c60cafa05141e1a0db5ac|$2b$10$pFGq8ZqjtrU9j75.pf8SSOMTdg1nIM2yxxFhmaAPfOx76PgB5tjcG
sqlite>
sqlite>
```



Timeline and Project Plan

The SafeRoads Navigator project is planned to be completed within a four–month period, which roughly translates to 15 weeks. To ensure timely completion, the following timeline with milestones have been proposed (aligned with the methodology steps):

1. Month 1 (Weeks 1–4) – Planning and Design (complete requirement analysis and finalize the system design. This includes the literature review (already in progress), drafting use cases, designing the database and API, and creating wireframes for the

UI. By the end of Week 4, a design review will be held with mentors to validate the approach.

- Milestone(s): Requirements & Design Document finalized, development environment set up (Node.js server skeleton, SQLite schema created), and use cases listed and identified.

2. Month 2 (Weeks 5–8) – Core Implementation (develop the core features (Iteration 1). By around Week 6, aim to have the hazard reporting and mapping functional: a user can submit a hazard and see it appear on the map (persisted in the DB). By Week 8, integrate user confirmation functionality (peer vetting) and have a basic dual vetting logic working in a rudimentary form.
    - Milestone(s): By end of Month 2, internal demo of the system showing multiple users reporting and confirming hazards, with the map updating live.
    - The following server endpoints (APIs) need to be developed:
        - `GET /fetch/hazards` to get hazards within a certain distance (RADIUS) that have reported to the system
        - `GET /fetch/allhazards` to get all hazards reported by road users
        - `POST /report-hazard` to report a road hazard
        - `POST /upvote-hazard` to upvote on a reported hazard
        - `POST /flag-hazard` to flag a reported hazard
        - `GET /admin/load-hazards` to load all hazards for moderation
        - `POST /admin/approve-hazard` to approve (moderate) upvoted hazards

- `POST /admin/reject-hazard` for rejecting hazards not meeting criterias
- `GET /api/analytics` for generating analytics/statistics
- `GET /api/reports/weekly` for generating statistics on a weekly basis
- `GET /api/reports/monthly` for generating statistics on a monthly basis
- `GET /api/reports/unresolved` for generating statistics of all unresolved/open hazards
- `GET /api/reports/resolution-rate` to get the latest resolution rate for all resolved hazards
- `GET /api/reports/resolved-rejected` for generating the list of all resolved and rejected hazards so far

3. Month 3 (Weeks 9–12) – Feature Completion and Refinement (Iteration II – implement remaining features and the full dual vetting rules. Build the admin dashboard in this phase and enforce the workflow for hazard status changes, i.e. unverified → verified or rejected). Also implement the heatmap view and any additional niceties (like user accounts or notifications) if time permits. Begin testing with sample data.
    - Milestone(s): Feature-complete beta version of SafeRoads Navigator by end of Week 12. At this point, all primary use cases can be executed on the system.

4. Month 4 (Weeks 13–14) – Final testing, Evaluation, & Finalization (Iteration III – rigorously test the system for bugs, fix any issues, and polish the UI/UX). Conduct the evaluation plan: simulate usage to gather data on how the dual vetting performs. Analyze

the results relative to research questions and incorporate findings into the final report. Concurrently, prepare the capstone presentation and documentation.

- Milestone(s): Week 15, project completion. Deliverables include the final research proposal document (with results from any experiments or tests), the working prototype deployed for demonstration, and a presentation for the academic panel.

Current Status and Completed Deliverables

As per the timeline mentioned in above timeline section, I have completed the deliverables identified in month 1, 2, and 3 (week 1–12) as follows:

- Month 1 milestones completed:
  - Review of related literature
  - Requirements analysis
  - Setup of development environment
  - Initial database schema design
  - Initial architecture design / API checklist
  - Drafting of use cases
  - Prepare initial list of test cases

- Month 2 milestones completed:
  - Completed review of related literature
  - Research and development plan completed
  - Finalized architecture design / API checklist
  - Finalized systems design

- Finalized database schema design
- Update test cases
- Implementation of core modules, i.e. 1) hazard reporting, hazard moderation
- Module testing

- Month 3 milestones completed:
  - Finalize test cases
  - Implementation of core module, i.e. 1) analytics
  - Development of core module, i.e. 1) maintenance, 2) reports
  - System integration
  - Module testing

Timeline for Pending Deliverables

Remaining activities for month 4 (week 13–15) are as follows:

- UI enhancements / final touches
- System integration completeness testing
- Integrated core modules testing
- Finalized research proposal document
- Demonstration of completed solution (MVP)

MSE907.2.pdf

原创性报告

66%	3%	3%	66%
相似指数	网际网络来源	出版物	学生文稿

主要来源

1	Submitted to North Shore International Academy 学生文稿	66%
2	Submitted to Multimedia University 学生文稿	<1%
3	Submitted to University of Leicester 学生文稿	<1%

不含引文	关闭	不含相符结果	关闭
排除参考书目	开		