1. INTRODUCTION

Congestion and oversaturation in traffic are worldwide issues. Because of the rapid increase in the number of vehicles, this problem is more acute in metropolitan areas. During peak business hours, city traffic frequently causes traffic jams. The same thing may be seen on freeways, especially at city entrances and exits. Most drivers are completely uninformed of road traffic situations ahead of time. Intelligent transportation systems have a wide range of uses for improving road traffic. Travelers can benefit significantly from fast and reliable traffic information in this way. Vehicles can use real-time trip time information as a great tool to lessen the effects of ongoing traffic growth on urban roadways.

For commuters and transport planners, traffic management is a crucial tool. Traditional traffic management systems employ an algorithm that has been prepared in advance and uses traffic data from years in the past to direct vehicles through possibly gridlocked traffic. Although the concept generally applies to traffic situations, there are always unusual situations in which the presumptive measures might not be the best.

Bring in the emergency vehicles (EVs.) These vehicles offer swift reactions to protect people and property. EVs (police cars, fire engines, and ambulances) get on the site quickly to offer initial aid and treatment for both minor accidents and severe calamities. Early intervention is essential in some emergency situations and can mean the difference between life and death. The chances of survival for patients with Sudden Cardiac Arrest (SCA) decrease by 7–10% per minute, and they are extremely poor after 8–10 minutes. Another emergency situation when prompt action lowers the risk of fatality and impairment is a stroke. After a flashover, the extent and heat of a fire intensifies over time and may spiral out of control (when all combustibles in the space have been heated to their ignition temperature and spontaneous combustion occurs.) By making sure that people receive early medical care, one-third of deaths caused by auto accidents can be avoided.

Traffic control systems ought to give preference to EVs over other vehicles so that they can react more quickly. To put it another way, intelligent traffic management systems require context.

1.1. PROJECT DESCRIPTION

A project consisting of a traffic simulator and an application that records data of emergency vehicles can be a powerful tool for understanding the impact of emergency vehicles on traffic flow and improving emergency response times. This project would involve two primary components: a traffic simulator and an application for recording and analyzing data from emergency vehicles.

The traffic simulator would be designed to simulate various traffic scenarios, including the presence of emergency vehicles. It would be programmed to simulate the behavior of different types of vehicles, including cars, trucks, buses, and emergency vehicles, and to replicate the conditions of real-world traffic flow. The simulator would also include an algorithm for traffic signal preemption, allowing emergency vehicles to receive priority at intersections.

The traffic simulator would be used to test various traffic scenarios and to evaluate the impact of emergency vehicles on traffic flow. By analyzing the data generated by the simulator, researchers and traffic engineers could identify potential bottlenecks and areas where emergency vehicles are likely to experience delays. This information could then be used to optimize traffic flow and improve response times for emergency services.

The application for recording data from emergency vehicles would be designed to collect information on the location, speed, and direction of emergency vehicles. This data would be transmitted in real-time to a central database, where it could be analyzed and used to inform traffic management decisions.

The application would also include a user interface for emergency responders, allowing them to input data on the nature of the emergency and the urgency of the situation. This information would be used to prioritize emergency response and to provide traffic signals with the necessary information to give emergency vehicles priority.

One potential application of this project could be in the design of traffic management systems for smart cities. By collecting data on emergency vehicles and traffic flow, cities could design traffic systems that are optimized for emergency response times and that minimize disruption to other road users. This could lead to more efficient and effective emergency response, as well as reduced congestion and improved safety for all road users.

Overall, a project consisting of a traffic simulator and an application for recording data from emergency vehicles could have significant benefits for traffic management and emergency response. By providing a better understanding of the impact of emergency vehicles on traffic flow and developing systems for prioritizing emergency response, this project could lead to more efficient and effective emergency services, reduced congestion, and improved safety for all road users.

1.2. TOOL REVIEW

1.2.1. FIREBASE

Firebase is a cloud-based mobile and web application development platform owned by Google. It was launched in 2011 and has since become one of the most popular platforms for building apps, offering a wide range of features and services that developers can use to build high-quality applications quickly and easily.

Firebase offers a comprehensive suite of tools and services that make it easy for developers to build, test, and deploy their applications. These tools include a real-time database, cloud messaging, hosting, authentication, analytics, and more. Firebase is designed to help developers create apps that are scalable, secure, and easy to maintain, while also providing the flexibility to customize and extend the platform as needed.

One of the key features of Firebase is its real-time database. This database is built on top of Google Cloud Platform and allows developers to store and sync data in real-time between clients and servers. The database is also fully managed, meaning that developers do not have to worry about infrastructure or maintenance.

Firebase also includes a cloud messaging service that allows developers to send notifications and messages to their users. This service supports both Android and iOS platforms and can be integrated with other Firebase services, such as the real-time database, to create powerful and personalized messaging experiences for users.

Another important feature of Firebase is its hosting service, which allows developers to quickly and easily deploy their applications to the web. Firebase hosting is built on top of a global content delivery network, which ensures that applications load quickly and reliably for users around the world. Developers can also use Firebase hosting to deploy static content, such as HTML, CSS, and JavaScript files, making it an ideal solution for building single-page applications and other web-based projects.

Firebase also includes a range of authentication services, which allow developers to add secure user authentication and authorization to their applications. These services include support for email and password authentication, social media sign-in, and third-party identity providers such as Google and Facebook. Firebase authentication is also fully customizable, allowing developers to tailor the user authentication experience to their specific needs.

In addition to these core features, Firebase also includes a range of other services that can be used to enhance the functionality and performance of applications. These include a cloud functions service, which allows developers to write and deploy serverless functions that can be triggered by events in their applications, and an analytics service, which provides real-time insights into user behavior and application performance.

Firebase also provides a range of tools and resources to help developers get started with the platform. These include extensive documentation, sample code, and a range of pre-built integrations with popular development tools and frameworks. Firebase also has an active developer community, with regular updates, tutorials, and support available from both Google and other developers.

In terms of pricing, Firebase offers a range of plans to suit different types of applications and development needs. The platform offers a free tier, which includes a range of core

features and services, as well as paid plans that offer additional features and functionality, such as increased storage and more advanced analytics.

In summary, Firebase is a comprehensive mobile and web application development platform that offers a wide range of features and services to help developers build high-quality applications quickly and easily. With its real-time database, cloud messaging, hosting, authentication, analytics, and more, Firebase is a powerful and flexible platform that is ideal for developers looking to create scalable, secure, and easy-to-maintain applications. Whether you're building a simple mobile app or a complex web-based project, Firebase offers the tools, resources, and support you need to bring your ideas to life.

1.2.2. JAVA

Java is a widely used object-oriented programming language that was developed in the mid-1990s by James Gosling and his team at Sun Microsystems. It was created to be a simple, secure, and portable language that could run on any platform, from small devices to large servers. Today, Java is used by millions of developers and is one of the most popular programming languages in the world.

One of the key features of Java is its portability. Java code can run on any platform that has a Java Virtual Machine (JVM) installed, which translates the Java code into machine language. This means that developers can write code once and run it on multiple platforms without having to make any modifications. This makes Java a popular choice for developing cross-platform applications, such as mobile apps, desktop applications, and web applications.

Another key feature of Java is its security. Java was designed with security in mind, and it has several built-in features that help to prevent common security issues, such as buffer overflows and memory leaks. The JVM also provides a secure sandbox environment for executing untrusted code, which helps to prevent malicious code from damaging the system.

Java is also an object-oriented language, which means that it is designed around objects and classes. Objects are instances of classes, which define the attributes and behaviors of the object. This makes it easy to create complex applications by breaking them down into smaller, more manageable objects.

Java also has a strong community of developers and users, which has led to the creation of many libraries, frameworks, and tools that make it easier to develop Java applications. Some popular libraries include Apache Commons, Google Guava, and Spring Framework. These libraries provide developers with pre-built components that they can use in their applications, saving them time and effort.

Java is also known for its performance. The JVM has a just-in-time (JIT) compiler, which compiles Java code into machine code at runtime, improving performance. This means that Java applications can be fast and efficient, even on low-end hardware.

Java is used in a wide range of applications, from mobile apps to large-scale enterprise systems. It is particularly popular in the enterprise space, where it is used to develop mission-critical systems such as banking applications, airline reservation systems, and inventory management systems. Java's ability to handle large-scale, complex systems makes it a popular choice for enterprise developers.

Java is also widely used in the development of Android mobile applications. Android Studio, the official integrated development environment (IDE) for Android, uses Java as its primary programming language. This makes it easy for developers to create high-quality Android apps using Java.

Java has a large and active community of developers, who contribute to the language's development and share their knowledge and expertise with others. There are many online resources available for learning Java, including tutorials, forums, and documentation. This makes it easy for developers to get started with Java and to continue learning and improving their skills.

In conclusion, Java is a powerful, versatile, and widely used programming language that is well-suited to a wide range of applications. Its portability, security, and performance

make it a popular choice for developers, particularly in the enterprise space. With a strong community of developers and a wealth of online resources available, Java is a great choice for anyone looking to learn a new programming language or to develop complex applications.

1.3. EXISTING SYSTEMS

Transit Signal Priority (TSP) is a traffic management system that provides priority to public transportation vehicles, such as buses and trains, by adjusting traffic signal timing. The goal of TSP is to reduce delays and improve the reliability of public transportation, while minimizing the impact on other road users.

TSP systems typically use a combination of sensors and communication technology to detect the presence of public transportation vehicles and to communicate with traffic signals. When a public transportation vehicle approaches a signalized intersection, the TSP system can request that the traffic signal provide a priority green phase, allowing the vehicle to proceed through the intersection more quickly.

There are two main types of TSP systems: passive and active. Passive TSP systems rely on sensors embedded in the roadway that detect the presence of public transportation vehicles. Active TSP systems use communication technology, such as radio or GPS, to communicate with the transit vehicle and to request priority.

Passive TSP systems are typically less expensive than active systems, but they are also less flexible. They require physical infrastructure, such as sensors embedded in the roadway, and are limited to providing priority only at specific locations where the infrastructure is installed. Active TSP systems, on the other hand, are more flexible and can provide priority at any location where the transit vehicle can be detected by the communication technology.

TSP systems can also be classified based on the level of priority provided. There are three main levels of priority: preemption, conditional priority, and absolute priority.

Preemption is the highest level of priority and involves the traffic signal switching immediately to a green phase for the public transportation vehicle, regardless of the signal phase for other road users. This is typically used in emergency situations, such as when a fire truck or ambulance is responding to a call.

Conditional priority provides priority to public transportation vehicles only if certain conditions are met, such as if the vehicle is running behind schedule or if there are a certain number of passengers on board. This allows TSP systems to be more flexible and to adapt to changing conditions.

Absolute priority provides priority to public transportation vehicles at all times, regardless of other traffic conditions. This is typically used on dedicated transit lanes, such as bus rapid transit (BRT) systems.

TSP systems can provide many benefits to public transportation systems and the communities they serve. By reducing delays and improving reliability, TSP can make public transportation a more attractive option for commuters, leading to increased ridership and reduced congestion on roads. TSP can also reduce greenhouse gas emissions and improve air quality by reducing the amount of time public transportation vehicles spend idling in traffic.

However, TSP systems also have potential drawbacks. Providing priority to public transportation vehicles can create delays for other road users, particularly at busy intersections where traffic congestion is already high. TSP systems must be carefully designed to minimize the impact on other road users and to ensure that they do not create additional safety hazards.

TSP systems must also be designed to ensure that they are effective and reliable. The systems must be able to accurately detect the presence of public transportation vehicles and to communicate with traffic signals in a timely manner. TSP systems must also be tested and validated to ensure that they are effective in reducing delays and improving the reliability of public transportation.

In conclusion, Transit Signal Priority is a traffic management system that provides priority to public transportation vehicles by adjusting traffic signal timing. TSP can provide many benefits to public transportation systems and the communities they serve, but it must be carefully designed and tested to ensure that it is effective and reliable, and that it minimizes the impact on other road users. With proper design and implementation, TSP can help to improve the reliability and attractiveness of public transportation, leading to reduced congestion and improved air quality.

1.4. PROPOSED SYSTEM

The system that registers users with emergency vehicles using an Android application and deploys green corridors with the help of timed phasing of traffic signals.

The system consists of two main components: an Android application that allows users to register their vehicles with emergency services and a traffic signal coordination system that can be activated in emergency situations. When a user registers their vehicle with the emergency services, they provide information about their emergency vehicle, including its type, model, and license plate number. This information is stored in a central database that emergency services can access when they need to respond to an emergency. When an emergency occurs, the emergency services can activate the traffic signal coordination system, which will create a green corridor that allows emergency vehicles to move through traffic quickly and safely.

The Android application is the primary interface through which users can register their vehicles with emergency services. The application should be easy to use and accessible to a wide range of users, including those who may not be tech-savvy.

The application should prompt users to enter information about their vehicles, including the email, emergency type, license plate number, etc. Users should also be able to specify the type of emergency service they are registering with, such as ambulance, fire department, or police department. Once the user has entered this information, they should be able to submit it to the central database, where it will be stored securely.

To ensure that only authorized users can register their vehicles with emergency services, the application should require users to verify their identity before they can submit their vehicle information. This could be done through email verification.

The traffic signal coordination system is the backbone of the system and enables the deployment of green corridors in emergency situations. The system should be designed to work seamlessly with existing traffic signal infrastructure, and it should be able to coordinate traffic signals across multiple intersections to create a clear path for emergency vehicles.

When an emergency occurs, the emergency services should be able to activate the traffic signal coordination system through a centralized control center. Once activated, the system will use data from the central database to identify the location of the emergency vehicle and create a green corridor that enables the vehicle to move through traffic quickly and safely.

The green corridor should be created by coordinating the timing of traffic signals at multiple intersections. The system should prioritize the movement of emergency vehicles over other vehicles, and it should be able to adjust the timing of traffic signals dynamically to respond to changing traffic conditions. The system should also be able to communicate with other transportation systems, such as public transit and railways, to ensure that emergency vehicles can move through the city quickly and safely.

The traffic signal coordination system should be designed to be flexible and adaptable to different types of emergencies. For example, the system could be configured to create different types of green corridors depending on the type of emergency. In the case of a medical emergency, the system could prioritize the movement of ambulances to hospitals, while in the case of a fire emergency, the system could prioritize the movement of fire trucks to the scene of the fire.

To ensure that the traffic signal coordination system is deployed safely and effectively, it is important to have buy-in from the Regional Transport Office.

1.5. OBJECTIVES

i) To improve traffic conditions for city traffic.

Traffic in cities is abysmal due to their sheer volume and the lack of infrastructure to support them. As a result, we aim to find efficient systems and methods to improve traffic systems. The improvement is done with the intention of giving requisite attention to Emergency Vehicles.

ii) To ensure the swift arrival of Emergency Vehicles (EVs) to provide essential services.

EVs offer swift reactions to protect people and property. EVs (police cars, fire engines, and ambulances) get on the site quickly to offer initial aid and treatment for both minor accidents and severe calamities. Early intervention is essential in some emergency situations and can mean the difference between life and death. The chances of survival for patients with Sudden Cardiac Arrest (SCA) decrease by 7–10% per minute, and they are extremely poor after 8–10 minutes. Another emergency situation when prompt action lowers the risk of fatality and impairment is a stroke. After a flashover, the extent and heat of a fire intensifies over time and may spiral out of control (when all combustibles in the space have been heated to their ignition temperature and spontaneous combustion occurs.) By making sure that people receive early medical care, one-third of deaths caused by auto accidents can be avoided.

iii) To reduce manpower and infrastructure to remove uncertainty and risk elements in traffic scenarios.

Outstanding traffic scenarios are often dealt with by manual officers. We aim to create a system to avoid spending these resources. Additionally, we aim to minimize resources while creating the smart traffic system.

1.6. PURPOSE, SCOPE AND APPLICABILITY

1.6.1. PURPOSE

Emergency vehicles, such as ambulances, fire trucks, and police cars, play a critical role in responding to emergencies and providing lifesaving services. However, navigating

through traffic can be challenging, particularly during rush hour or in congested urban areas. To address this challenge, many cities and municipalities have implemented traffic systems that prioritize emergency vehicles, allowing them to reach their destinations more quickly and safely.

The purpose of a traffic system that prioritizes emergency vehicles is to reduce response times and improve the overall effectiveness of emergency services. When emergency vehicles are delayed in traffic, it can result in serious consequences, including loss of life and property damage. By providing a clear path for emergency vehicles to navigate through traffic, response times can be significantly reduced, allowing emergency services to arrive on the scene more quickly and provide lifesaving services.

There are several ways that traffic systems can prioritize emergency vehicles. One approach is to use traffic signal preemption, which involves changing the timing of traffic signals to allow emergency vehicles to move through intersections more quickly. In a preemption system, an emergency vehicle approaching an intersection sends a signal to the traffic control system, which adjusts the signal timing to provide a green light for the emergency vehicle.

The benefits of a traffic system that prioritizes emergency vehicles are significant. By reducing response times, emergency services can be more effective in responding to emergencies and providing lifesaving services. This can result in reduced property damage, fewer injuries, and more lives saved.

In addition, prioritizing emergency vehicles can also have a positive impact on the overall safety of the community. When emergency responders are able to arrive on the scene more quickly, they can help to prevent accidents and other emergencies from escalating into more serious situations.

There are also economic benefits to prioritizing emergency vehicles. When emergency responders are able to arrive on the scene more quickly, it can reduce the amount of time that businesses and other organizations are disrupted by emergencies. This can result in reduced economic losses and improved productivity.

However, there are also challenges and considerations that must be taken into account when implementing a traffic system that prioritizes emergency vehicles. One challenge is the potential impact on other road users. When traffic signals are preempted for emergency vehicles, it can create delays and disruptions for other road users, particularly during rush hour or in congested areas.

To address this challenge, traffic systems must be designed to minimize the impact on other road users while still providing priority to emergency vehicles. This can be achieved through careful planning and coordination, as well as the use of advanced technologies that can help to minimize delays and disruptions.

Another consideration is the potential cost of implementing a traffic system that prioritizes emergency vehicles. While the benefits of such a system are significant, the costs can be high, particularly if physical infrastructure, such as dedicated emergency vehicle lanes, is required.

To address this challenge, cities and municipalities can explore alternative approaches, such as automated traffic management systems, that may be more cost-effective and easier to implement.

In conclusion, a traffic system that prioritizes emergency vehicles can play a critical role in improving the effectiveness of emergency services and reducing response times. By providing a clear path for emergency vehicles to navigate through traffic, response times can be significantly reduced, allowing emergency services to arrive on the scene more quickly and provide lifesaving services. While there are challenges and considerations that must be taken into account

1.6.2. SCOPE

The project is easily scalable to apply to multiple traffic signals; however, there are certain assumptions made for its application:

i) Accuracy of the radar/ tracking system is so high that error is minimal.

For the traffic system to detect and execute the algorithm and provide the most efficient action, we have to assume that the range of error is negligible. For example, if two EVs approach the traffic intersection, the distance plays the role of one of the determining factors in deciding which is the best course of action.

ii) The vehicles are incapable of giving way to EVs.

If the vehicles are capable of giving way to the EVs, there is no purpose for the Smart Traffic System as they can move ahead without being stuck in traffic. The idea behind the project had arisen from the observations made from observing traffic through the narrow roads in Bengaluru city which are not wide enough for the vehicles to make way for EVs.

iii) The traffic signals are connected to each other.

There are many traffic lights that run on timed systems i.e. they run independent of the other lights. For the smart system to perform accordingly, the traffic lights have to work together: when one light turns green, the others have to match the green with red to avoid collisions.

In addition to these assumptions, we also state a limitation: the implementation of the system to all traffic light intersections. As stated prior, the interruption caused by EVs are disruptive to the existing traffic system. In other words, there should be a method to make up for the disruption caused.

The scope of a traffic system that prioritizes emergency vehicles is broad, encompassing various aspects of emergency services and traffic management. The system's primary objective is to ensure that emergency vehicles can reach their destination quickly and efficiently, reducing response times and saving lives.

One of the significant scopes of the system is the integration of advanced technologies such as GPS, real-time traffic information, and communication systems. These technologies can provide critical information to emergency services, such as the location of the emergency vehicle, the fastest route to the destination, and potential roadblocks or obstacles on the way.

Another scope of the system is the development of algorithms that can optimize traffic flow and minimize delays for emergency vehicles. This can be achieved by dynamically adjusting traffic signals and rerouting non-emergency traffic to alternate routes, thereby creating a clear path for emergency vehicles.

The system can also provide real-time data on emergency vehicle status, such as speed, fuel level, and maintenance requirements. This information can be used to optimize the maintenance schedule, ensure that vehicles are in good working condition, and reduce the risk of breakdowns during emergency situations.

Additionally, the system's scope includes the ability to simulate various traffic scenarios and emergency situations to improve emergency response planning and evaluate the effectiveness of the system. Simulation models can help identify potential bottlenecks, optimize routing and scheduling, and test the system's ability to respond to various types of emergencies.

Finally, the scope of the system also includes the ability to provide real-time information to the public about emergency situations, traffic conditions, and alternate routes. This information can be made available through various channels, such as mobile applications, websites, and social media platforms, enabling citizens to make informed decisions and avoid areas affected by emergencies or traffic disruptions.

In summary, the scope of a traffic system that prioritizes emergency vehicles is vast, encompassing the integration of advanced technologies, the development of traffic optimization algorithms, real-time vehicle status monitoring, emergency response planning through simulations, and the provision of real-time information to the public. With continued investment in smart city infrastructure and emerging technologies, the scope of such a system will only continue to expand, improving emergency response times and saving lives.

1.6.3. APPLICABILITY

A traffic system that prioritizes emergency vehicles is highly applicable in urban areas where there is a high volume of traffic and congestion. Emergency services such as

ambulances, fire trucks, and police cars play a critical role in responding to emergencies and saving lives. However, navigating through traffic can be challenging, and delays can result in serious consequences, including loss of life and property damage. A traffic system that prioritizes emergency vehicles can reduce response times and improve the overall effectiveness of emergency services.

One of the primary benefits of a traffic system that prioritizes emergency vehicles is that it reduces response times. When emergency vehicles are delayed in traffic, it can result in serious consequences, including loss of life and property damage. By providing a clear path for emergency vehicles to navigate through traffic, response times can be significantly reduced, allowing emergency services to arrive on the scene more quickly and provide lifesaving services. In some cases, the difference of just a few minutes can be the difference between life and death.

Another benefit of a traffic system that prioritizes emergency vehicles is that it can improve the overall safety of the community. When emergency responders are able to arrive on the scene more quickly, they can help to prevent accidents and other emergencies from escalating into more serious situations. For example, when a car accident occurs, emergency responders can quickly clear the scene, reducing the risk of additional accidents and injuries.

A traffic system that prioritizes emergency vehicles can also have economic benefits. When emergency responders can arrive on the scene more quickly, it can reduce the amount of time that businesses and other organizations are disrupted by emergencies. This can result in reduced economic losses and improved productivity. In addition, when emergency responders can save lives and prevent property damage, it can reduce the costs associated with emergency response and disaster recovery.

There are several ways that a traffic system can prioritize emergency vehicles. One approach is to use traffic signal preemption, which involves changing the timing of traffic signals to allow emergency vehicles to move through intersections more quickly. In a preemption system, an emergency vehicle approaching an intersection sends a signal to the traffic control system, which adjusts the signal timing to provide a green light for the

emergency vehicle. This approach is highly effective in reducing response times and minimizing delays for emergency vehicles.

Another approach is to use dedicated emergency vehicle lanes or corridors, which provide a clear path for emergency vehicles to navigate through traffic. These lanes can be physically separated from other lanes, or they can be shared with other high-occupancy vehicles, such as buses. Dedicated emergency vehicle lanes are highly effective in reducing response times and ensuring that emergency vehicles can quickly and safely reach their destinations.

Some cities also use automated traffic management systems that use sensors and cameras to detect the presence of emergency vehicles and adjust traffic flow accordingly. These systems can also provide real-time information to emergency responders, such as traffic conditions and estimated travel times. Automated traffic management systems are highly effective in reducing response times and minimizing delays for emergency vehicles.

There are also challenges and considerations that must be taken into account when implementing a traffic system that prioritizes emergency vehicles. One challenge is the potential impact on other road users. When traffic signals are preempted for emergency vehicles, it can create delays and disruptions for other road users, particularly during rush hour or in congested areas. To address this challenge, traffic systems must be designed to minimize the impact on other road users while still providing priority to emergency vehicles. This can be achieved through careful planning and coordination, as well as the use of advanced technologies that can help to minimize delays and disruptions.

Another consideration is the potential cost of implementing a traffic system that prioritizes emergency vehicles. While the benefits of such a system are significant, the costs can be high, particularly if physical infrastructure, such as dedicated emergency vehicle lanes, is required.

1.7. OVERVIEW OF THE REPORT

System Analysis and Requirements:

This section will outline the initial analysis of the current traffic system and the requirements for the new system. It will include an overview of the current system, the problems that need to be addressed, and the goals of the new system. It will also cover the various stakeholders involved in the project and their respective requirements.

Problem Definition:

This section will provide a more detailed definition of the problems with the current traffic system, including bottlenecks, delays, and safety concerns. It will also outline the potential consequences of these problems and the benefits of implementing a new system.

Diagrams:

This section will include the various diagrams used in the project, including flowcharts, data flow diagrams, and entity-relationship diagrams. These diagrams will help to visualize the system architecture and data flow.

System Requirements and Specification:

This section will provide a detailed overview of the system requirements and specifications. It will cover the functional and non-functional requirements of the system, as well as the system architecture, hardware and software requirements, and data storage requirements.

Constraints:

This section will outline the various constraints that must be considered in the development of the system. This will include technical constraints, such as hardware and software limitations, as well as regulatory and legal constraints.

Conceptual Models:

This section will include the conceptual models used in the development of the system. It will cover the various use cases, scenarios, and workflows involved in the system.

Data Flow Diagrams:

This section will include the data flow diagrams used in the development of the system. These diagrams will show how data moves through the system and between different components.

System Design:

This section will provide a detailed overview of the system design. It will cover the system architecture, module and database designs, system configuration, UI design, and application flow.

System Architecture:

This section will outline the overall architecture of the system, including the various components and their interactions.

Module and Database Designs:

This section will cover the individual modules and databases used in the system. It will provide an overview of their functions and interactions.

System Configuration:

This section will outline the system configuration, including the hardware and software requirements and the network configuration.

UI Design:

This section will provide an overview of the user interface design, including the various screens and their functions.

Application Flow:

This section will cover the application flow, including the various use cases and scenarios involved in the system.

Report Design:

This section will outline the design of the various reports generated by the system. It will cover the data and metrics included in the reports and their layout and formatting.

Implementation:

This section will cover the implementation of the system, including the approaches used, coding standards, coding details, and screenshots.

Implementation Approaches:

This section will cover the various approaches used in the implementation of the system, including the choice of programming languages and development methodologies.

Coding Standard:

This section will outline the coding standard used in the development of the system. It will cover the coding conventions, documentation requirements, and testing procedures.

Coding Details and Screenshots:

This section will provide a detailed overview of the coding details and include screenshots of the various components of the system.

Testing:

This section will cover the testing of the system, including the test cases, testing approaches, and test reports.

Test Cases:This section will cover the various test cases used in the testing of the system. It will include the expected outcomes

2. SYSTEM ANALYSIS AND REQUIREMENTS

2.1. PROBLEM DEFINITION

Emergency vehicles such as ambulances, police cars, and fire trucks play a crucial role in saving lives and minimizing damage during emergencies. However, traffic congestion can significantly delay their response times, putting lives at risk. Therefore, a smart traffic system that prioritizes emergency vehicles can significantly improve emergency response times.

2.2. REQUIREMENTS SPECIFICATION

Traffic Simulator Requirements Specification:

1. Simulation Environment: The simulator should provide an accurate simulation environment for the traffic conditions in the target area, including road types, traffic signals, and vehicle behavior.

2. Traffic Behavior: The simulator should simulate the behavior of all types of vehicles, including passenger cars, trucks, buses, and emergency vehicles.

3. Traffic Flow Analysis: The simulator should be able to analyze the traffic flow in real-time and detect bottlenecks, congestion, and other issues.

4. Emergency Vehicle Prioritization: The simulator should be able to prioritize emergency vehicles in traffic flow, including providing them with the right-of-way at intersections and other traffic scenarios.

5. Scenario Creation: The simulator should allow for the creation of custom traffic scenarios for testing and analysis purposes.

6. Reporting: The simulator should generate reports detailing the traffic flow analysis and performance metrics for each scenario.

7. User Interface: The simulator should have a user-friendly interface that allows for easy navigation and control of the simulation.

8. Compatibility: The simulator should be compatible with a wide range of operating systems and devices.

Emergency Vehicle Tracking Application Requirements Specification:

1. Vehicle Information: The application should store and display relevant information about each emergency vehicle, including its location, status, and destination.

2. Prioritization: The application should prioritize emergency vehicles in traffic flow, including providing them with the right-of-way at intersections and other traffic scenarios.

3. Route Optimization: The application should provide route optimization for emergency vehicles based on traffic conditions, including avoiding congestion and other delays.

4. Reporting: The application should generate reports detailing the performance of emergency vehicles in response to incidents.

5. User Interface: The application should have a user-friendly interface that allows for easy navigation and control of the emergency vehicle tracking system.

6. Compatibility: The application should be compatible with a wide range of operating systems and devices.

7. Security: The application should have strong security measures in place to protect sensitive information, including user data and emergency response protocols.

8. Integration: The application should be able to integrate with other emergency response systems, including dispatch centers and medical facilities.

9. Scalability: The application should be scalable to accommodate an increasing number of emergency vehicles and incidents in the target area.

The requirements for a traffic simulator and an emergency vehicle tracking application are critical to the effective management of traffic and emergency response in a given area. A traffic simulator can help to optimize traffic flow and prioritize emergency vehicles, while an emergency vehicle tracking application can provide real-time tracking and management of emergency incidents. The requirements specification for each system

must be carefully considered to ensure that they meet the needs of all stakeholders and are compatible with the target area's unique traffic and emergency response conditions.

2.3. BLOCK DIAGRAM

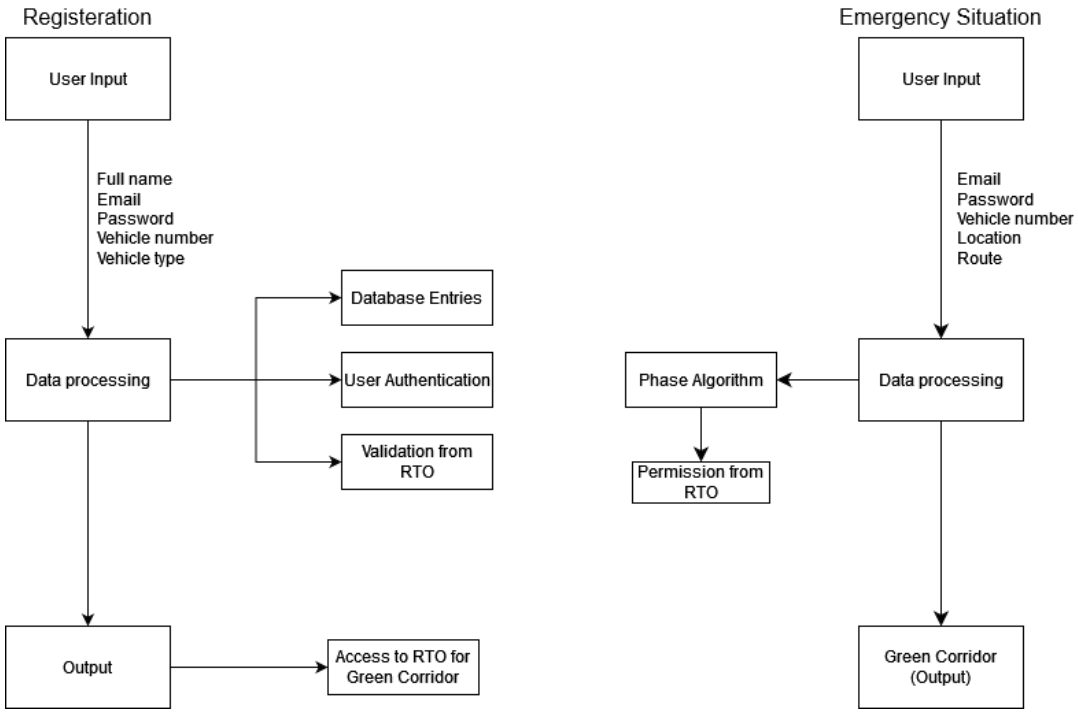

Fig 1.1: Block diagram

2.4. SYSTEM REQUIREMENTS

2.4.1. USER CHARACTERISTICS

The target demographic for the front-end application and website are those who are both novices and professionals at handling software. The registration process is simplified to reduce the tediousness that comes along with most registrations. The web application should have a help section or user manual that provides information on how to use the different features of the application.

A novice in handling application software is someone who is new to using a particular application or software program. They may have little to no experience using technology

or may have experience with other applications but are unfamiliar with the one they are currently working with. Here are some characteristics of a novice at handling application software:

1. Lack of Familiarity: Novices are not familiar with the software they are using, including its interface, tools, features, and functions. They may find it difficult to navigate the application and locate the features they need.

2. Limited Technical Knowledge: Novices may have limited technical knowledge about computers and software applications. They may not be familiar with common technical terms and jargon used in the software.

3. Slow Learning Curve: Novices often have a slow learning curve when it comes to learning how to use new software. They may require more time and patience to understand the application's features and functionality.

4. Tendency to Rely on Help Documentation: Novices tend to rely heavily on help documentation, such as user manuals or online tutorials, to learn how to use the software. They may not feel comfortable experimenting with the application on their own.

5. Mistakes and Errors: Novices may make mistakes and errors while using the software, such as clicking the wrong button or selecting the wrong option. These mistakes can lead to frustration and further confusion.

6. Fear of Breaking the Software: Novices may fear that they will break the software or cause it to malfunction if they use it incorrectly. This fear can make them hesitant to explore the application and try new things.

7. Need for Assistance: Novices often require assistance from more experienced users or support staff to learn how to use the software effectively. They may need help troubleshooting issues or understanding how to perform specific tasks.

8. Lack of Customization: Novices may not know how to customize the software to their needs. They may not be aware of the different settings and options available to them, or they may not know how to change them.

9. Inefficiency: Novices may be inefficient at using the software, as they may take longer to perform tasks or may not know how to perform certain tasks at all. This inefficiency can lead to frustration and a lack of productivity.

10. Limited Understanding of Advanced Features: Novices may not be aware of or understand advanced features of the software that could enhance their experience or improve their productivity. They may not be ready to explore these features until they become more comfortable with the application

In conclusion, novices at handling application software are characterized by their lack of familiarity with the software, limited technical knowledge, slow learning curve, tendency to rely on help documentation, mistakes and errors, fear of breaking the software, need for assistance, lack of customization, inefficiency, and limited understanding of advanced features. As novices become more experienced with the software, they will develop greater confidence and proficiency, allowing them to take advantage of the software's full range of features and benefits.

In addition, the mobile application simplifies the tediousness of searching for the computer. Anyone who is adept at handling cell phones can also easily traverse through the application provided.

2.4.2. SOFTWARE AND HARDWARE REQUIREMENTS

2.4.2.1 HARDWARE REQUIREMENTS

Hardware Requirements for Anylogic Simulation:

- CPU: AnyLogic Simulation can run on most modern CPUs, including Intel and AMD processors. However, multi-core processors are recommended for faster performance.

- RAM: AnyLogic Simulation requires a minimum of 4GB of RAM, but 8GB or more is recommended for optimal performance.

- Hard Drive Space: AnyLogic Simulation requires at least 1GB of free hard drive space for installation and operation.

- Display: AnyLogic Simulation is designed to work with a standard display resolution of 1024x768 or higher.

Hardware Requirements for Firebase:

- CPU: Firebase can run on most modern CPUs, including Intel and AMD processors. However, multi-core processors are recommended for faster performance.

- RAM: Firebase does not require a lot of memory to operate, but it is recommended to have at least 2GB of RAM for optimal performance.

- Hard Drive Space: Firebase does not require a lot of hard drive space to operate, but it is recommended to have at least 1GB of free space for installation and operation.

2.4.2.2 SOFTWARE REQUIREMENTS

AnyLogic

AnyLogic Simulation is a software tool for creating complex simulation models that can be used to analyze and optimize business processes, systems, and operations. To run AnyLogic Simulation, certain hardware and software requirements must be met. Here is a detailed explanation of the software and hardware requirements for AnyLogic Simulation.

- Operating System: AnyLogic Simulation is compatible with Windows, Mac OS X, and Linux operating systems.

- Java: AnyLogic Simulation requires Java to be installed on the computer. The recommended version is Java 8, although newer versions of Java may also work.

- Web Browser: AnyLogic Simulation requires a web browser for accessing online resources and help documentation. Any modern web browser should work.

Graphics Card:

- AnyLogic Simulation requires a graphics card that supports OpenGL 2.0 or later for 3D graphics rendering. Most modern graphics cards meet this requirement.

Networking:

- AnyLogic Simulation does not require a network connection to operate, but may need access to the internet to download updates or access online resources.

Software Licensing:

- AnyLogic Simulation is licensed software and requires a valid license to operate. Licenses can be purchased from AnyLogic and activated on the computer where the software will be installed.

In summary, AnyLogic Simulation requires a modern CPU with multi-core processing, at least 4GB of RAM, 1GB of free hard drive space, and a graphics card that supports OpenGL 2.0 or later. It is compatible with Windows, Mac OS X, and Linux operating systems, and requires Java to be installed. Any modern web browser can be used to access online resources and help documentation. A valid license is required to operate the software. Meeting these requirements will ensure that AnyLogic Simulation operates smoothly and efficiently, allowing users to create complex simulation models to optimize business processes, systems, and operations.

FIREBASE

Firebase is a popular mobile and web application development platform, developed by Google. It provides developers with a range of features and services to develop and manage applications, including real-time database, authentication, hosting, storage, messaging, and more. To use Firebase effectively, certain hardware and software requirements must be met. Here is a detailed explanation of the software and hardware requirements for Firebase.

SOFTWARE REQUIREMENTS:

- Operating System: Firebase is compatible with Windows, Mac OS X, and Linux operating systems.

- Browser: Firebase requires a web browser for accessing the Firebase console, which is used to manage Firebase projects. Any modern web browser, such as Google Chrome or Mozilla Firefox, should work.

- Node.js: Firebase uses Node.js for its command-line interface (CLI) tools, so it is recommended to have Node.js installed on your computer. The latest version of Node.js can be downloaded from the Node.js website.

- Firebase CLI: To use Firebase from the command line, you must install the Firebase CLI tool. This can be installed globally using Node.js and the npm package manager.

MOBILE DEVELOPMENT:

- Android: To develop Android applications using Firebase, you will need to have Android Studio installed on your computer. Android Studio is the official IDE for Android development and can be downloaded from the Android Studio website.

- iOS: To develop iOS applications using Firebase, you will need to have Xcode installed on your computer. Xcode is the official IDE for iOS development and can be downloaded from the Apple Developer website.

In summary, Firebase can run on most modern CPUs with at least 2GB of RAM and 1GB of free hard drive space. It is compatible with Windows, Mac OS X, and Linux operating systems and requires a modern web browser, Node.js, and the Firebase CLI tool to operate. To develop mobile applications using Firebase, Android Studio is required for Android development and Xcode is required for iOS development. Meeting these requirements will ensure that developers can effectively use Firebase to develop and manage mobile and web applications.

2.4.3. CONSTRAINTS

Developing a priority-based smart traffic system, even on a small scale, can be a complex and challenging task. Here are some constraints that must be considered when developing a priority-based smart traffic system on a small scale:

1. Limited Resources: Developing a priority-based smart traffic system on a small scale can be limited by resources such as funding, technical expertise, and access to technology. Without adequate resources, it may be difficult to develop a system that is efficient and effective.

2. Limited Data: A priority-based smart traffic system relies heavily on data to make informed decisions. However, on a small scale, there may be limited data available to use in the system. This can make it challenging to accurately predict traffic patterns and make informed decisions regarding emergency vehicle prioritization.

3. Limited Infrastructure: On a small scale, the infrastructure required to support a priority-based smart traffic system may be limited. For example, there may not be enough cameras or sensors installed to accurately monitor traffic, or the road network may not be large enough to make it feasible to implement the system.

4. Legal and Regulatory Constraints: Developing a priority-based smart traffic system on a small scale may be limited by legal and regulatory constraints. For example, there may be restrictions on the use of emergency vehicle sirens, or there may be restrictions on the types of data that can be collected and used.

5. Implementation Challenges: Implementing a priority-based smart traffic system on a small scale can be challenging due to the complexity of the system. It requires the integration of multiple technologies, such as sensors, cameras, and software, which can be difficult to coordinate and maintain.

6. Unforeseen Situations: Even with the best planning and design, a priority-based smart traffic system on a small scale may face unforeseen situations that can impact its effectiveness. For example, unexpected weather conditions or road closures can affect traffic patterns and the ability of emergency vehicles to navigate through traffic.

7. Emergency Vehicles (EVs) are broadly categorized into three: police vehicles, ambulances and fire trucks. There are no distinctions provided based on whether the vehicles are of governmental or privatized origin. In addition, no other vehicles are to be registered as there is a fourth category.

8. The traffic phase manipulation performed is passive. As a result, only the intersections in the direction of the Emergency Vehicle (EV) are adjusted. Doing so avoids the phase manipulation of adjoint intersections. In turn, the intersections that pose traffic that is heading in all directions run the risk of becoming clogged with vehicles.

9. The database used for the project is Firebase. For the sake of being cost-effective, only the free version of Firebase is utilized and, as a result, has limited storage for the registered Emergency Vehicles. For a fully up-scale model, a stronger version of Firebase would be required.

10. There is no email verification for the registering process. As a result, there is the possibility of a person misusing the application by binding the wrong email address with the Emergency Vehicle (EV.) However, since the supervisors have access to vehicle and driver details, there is always the possibility of remedying the error.

11. The database works over a strong and reliable internet connection. Failing to do so would result in faulty or failed services. For example, when there is a weak internet connection, the routing and ETA calculation would yield inappropriate results.

2.5. CONCEPTUAL MODELS

2.5.1. DATA MODELS

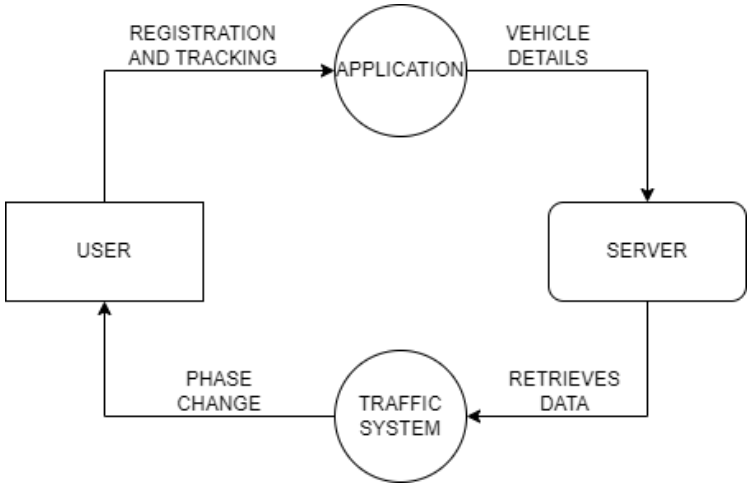

Fig 2.1. DFD Level 0

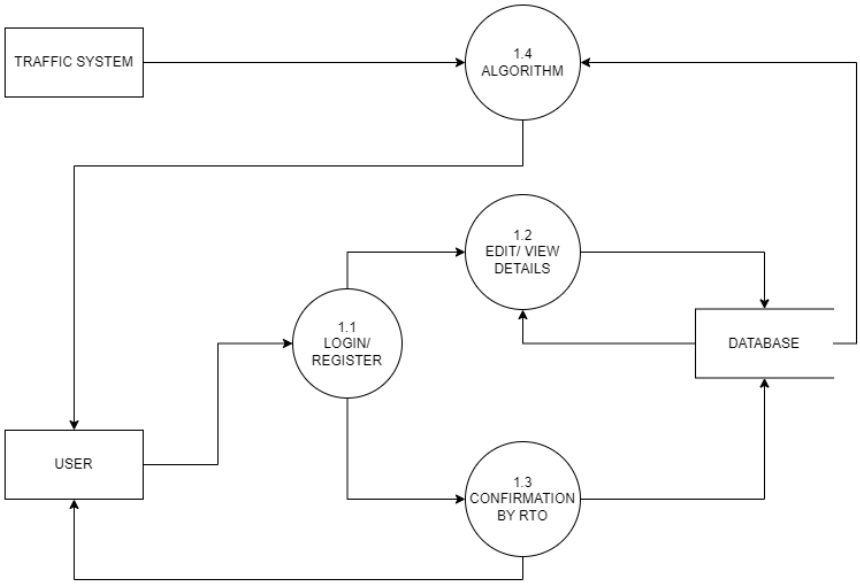

Fig 2.2. DFD Level 1

3. System Design

3.1 SYSTEM ARCHITECTURE

The system architecture of the project can be divided into four different major parts:

3.1.1. PRESENTATION LAYER

The user interface elements that end users utilize to communicate with the system are contained in this layer. Java-based code for Kotlin (Android Studio) will be used to create the user interface. The rendering of the user interface and management of user input will fall under the purview of this layer.

3.1.2. APPLICATION LAYER

This layer contains the logic of the application, including data validation, processing, and communication with other layers. It includes the main application code that handles user input and generates output. The application layer will be developed using firebase as the programming platform. It will include the following components:

a. Login Component

This component will be responsible for the validation of data presented to the rightful user. It will provide the required data protection.

b. Registration Component

This component will be responsible for providing access to create an account into the database. It will provide functionality of adding details into the database.

c. View Details

This component will be responsible for providing access to view the respective account from the database. It will provide functionality of accessing details from the database.

3.1.3. DATA LAYER

Data from the database is stored and retrieved by this layer. The database management system used to construct the data layer will be Firebase. Data from the database is stored and retrieved by this layer. It includes the tables and stored procedures used to store and retrieve data from the database.

3.1.4. SIMULATION LAYER

The simulation required to run the system must be provided by this layer. It has APIs, and other integration tools that let the programme communicate with other systems and understand the interaction between agents, hence extracting data. It offers data understanding, modelling, and validation checks as features for the algorithm applied. It has components for making and checking the decisions taken based on user information, preventing worst and edge case scenarios, and making sure the system, in play, is safe. It consists of the necessary software and hardware to operate the application. It consists of the software to run the simulation, (here AnyLogic) to support the application.

3.2 MODULE DESIGN

In this section, we will describe each process of the System Architecture.

User Interface

The User Interface module is responsible for providing an intuitive and user-friendly interface for users to interact with the system. This module includes components such as the registration page that allows the user to submit details about the Emergency Vehicle, a view feature to check vehicle details once logged in and an editable functionality that allows the user to update details as the need arises.

Backend API

The Backend API module is responsible for handling the communication between the User Interface module and the back-end system. This module includes components such as API endpoints that define the interface between the front-end and back-end, request

handling that manages user requests and data, and response generation that provides the appropriate response back to the user. The backend API can be developed using popular web frameworks such as Django, Flask and Java.

The backend of AnyLogic Simulation is the engine that powers the simulation. It includes a variety of components, such as the simulation model, the simulation engine, and the data analysis tools.

The simulation model is created by users using the UI. It defines the structure of the simulation system, including the agents, the flowcharts, the charts, and any other simulation elements.

The simulation engine is responsible for running the simulation based on the configuration provided by the user. It uses a variety of algorithms and techniques to simulate the behavior of the system over time, based on the rules defined in the simulation model.

The data analysis tools in AnyLogic Simulation enable users to analyze the results of the simulation. Users can view the results in a variety of formats, such as charts, graphs, and tables. They can also export the results to other software tools for further analysis.

In addition to these key components, the backend of AnyLogic Simulation also includes a variety of supporting technologies, such as database management, data visualization, and optimization algorithms. These technologies work together to enable users to create and analyze complex simulations with ease.

In our case, Backend API is used to run simulations to exhibit how our algorithm works.

3.3. SYSTEM CONFIGURATION

The system configuration for training PBSTS model for traffic simulations and reduction of response times is minimal. Here are some guidelines for the system configuration:

● CPU: A decently powerful CPU is also required for traversing the dataset, loading data into memory, and running training scripts. A multi-core CPU such as Intel Core i3 or i5 is amply sufficient.

●RAM: The amount of RAM required depends on the size of the dataset and the complexity of the model. A minimum of 4GB RAM is recommended, but for larger datasets, 8GB is more than enough.

●Storage: A small amount of storage is required to store the dataset for the Emergency Vehicle data. A Hard-Disk Drive (HDD) is recommended for faster data access.

●Software: The simulations can be implemented using Java.

●Networking: An Internet connection of 1Mbps is needed for keeping track of the locations of the vehicles as per need.

3.3.1 INTERFACE DESIGN AND PROCEDURAL DESIGN

Interface design and procedural design are two important aspects of developing any software application, and an emergency vehicle registration application is no exception. In this section, we will discuss these two design concepts and their application to an emergency vehicle registration application.

Interface design refers to the design of the graphical user interface (GUI) of an application. It includes the layout, color scheme, font size, and other visual elements that make up the user interface. The primary goal of interface design is to create a user-friendly and visually appealing interface that allows users to easily interact with the application.

For an emergency vehicle registration application, the interface design should be intuitive and easy to use. The application should have a simple and clean layout with clear and concise instructions on how to use it. The application should also have a color scheme that is easy on the eyes and makes it easy to distinguish different elements of the interface. The font size should be large enough to be easily readable, and the buttons and other interactive elements should be easy to click on.

Procedural design, on the other hand, refers to the design of the application's underlying logic and processes. It includes the flow of the application, the algorithms used to perform certain functions, and the sequence of steps required to complete certain tasks. The primary goal of procedural design is to create an efficient and effective system that meets the needs of the users.

For an emergency vehicle registration application, the procedural design should be well thought out and logical. The application should have a clear flow of actions, from registering a new vehicle to updating existing information. The system should be designed to be efficient, with minimal steps required to complete tasks. For example, when a user is registering a new vehicle, the application should automatically validate the information entered and alert the user if any information is missing or incorrect.

In addition, the application should have clear error messages and instructions for users in case something goes wrong. This will ensure that users are able to quickly resolve any issues they encounter while using the application.

Overall, the interface and procedural design of an emergency vehicle registration application are critical to its success. The application should be designed with the user in mind, making it easy to use and navigate, while also providing efficient and effective processes to complete tasks. By considering these design principles, developers can create an application that is intuitive, user-friendly, and meets the needs of its users.

3.3.2 USER INTERFACE DESIGN

AnyLogic Simulation is a powerful simulation software that provides users with the ability to simulate complex systems in a variety of industries, including transportation, manufacturing, and healthcare. In order to provide users with a comprehensive simulation experience, AnyLogic Simulation features both a user interface and a backend that work together seamlessly to enable simulation creation, configuration, and analysis.

User Interface

The user interface (UI) of AnyLogic Simulation is designed to be intuitive and easy to use, even for users who are not experienced in simulation software. The UI is organized into several key areas, including the main window, the toolbars, and the palettes.

The main window is where users create and configure simulations. It includes a variety of views, such as the simulation model, the simulation results, and the simulation parameters. Users can navigate between these views using tabs at the top of the window.

The toolbars in AnyLogic Simulation provide access to a variety of functions, such as creating new simulation elements, modifying simulation parameters, and running simulations. The toolbars are organized based on their functionality, with related functions grouped together.

The palettes in AnyLogic Simulation provide access to simulation elements, such as agents, flowcharts, and charts. Users can drag and drop these elements onto the simulation model to create the desired system.

In addition to these key areas, the UI of AnyLogic Simulation also includes a variety of menus, dialogs, and other interface elements that enable users to configure and analyze simulations.

The integration between the UI and the backend in AnyLogic Simulation is seamless, allowing users to create and modify simulations with ease. Users can drag and drop simulation elements from the palettes onto the simulation model in the UI, and the backend will automatically update to reflect the changes. Users can also modify simulation parameters and run simulations using the toolbars in the UI, and the backend will update to reflect the new configuration.

Overall, the user interface and backend of AnyLogic Simulation work together to provide users with a comprehensive simulation experience. The UI is designed to be intuitive and easy to use, while the backend is powerful and flexible, enabling users to create and analyze complex simulations with ease. The integration between the UI and the backend is seamless, providing users with a fluid and efficient simulation experience.

3.3.3 APPLICATION FLOW

The UI for our application has been designed in a way that people with even a basic smartphone can use this facility being offered by us. The application opens up to a login page where the user needs to input their unique User ID and Password. Upon the details being entered, these are verified and matched with the details present in our database(MySQL) and the user gets logged in. The user after getting logged in can view their personal details entered upon the time of registration. They can also view how the traffic priority system works and what is the precedence in which the vehicles will be offered priority. They will also be able to update their personal details like vehicle no and vehicle type, so that they don't have to do a new registration every time they switch from their existing vehicle to a new vehicle. In addition to this they will also have the option to add more vehicles to their existing User ID. In case of any help required they can access the help option from the menu and reach out to us.

The login page also has a registration option in the lower half of the page. A new user, not having an existing User ID can click on this option and register to this system. They will be needed to enter their name, email address, password , vehicle number and vehicle type as part of this one time registration process. Once they are registered, they can click on the login option in the lower half of the page and enter their credentials for logging into their profile.

4. iMPLEMENTATION

4.1. CODING STANDARD

Coding standards, also known as coding conventions or programming style, refer to a set of guidelines and best practices that software developers follow when writing code. These guidelines help to ensure that the code is consistent, readable, maintainable, and easy to understand by other developers. Coding standards can be specific to a particular programming language or can be more general guidelines that apply to any language.

Coding standards can cover a wide range of topics, including naming conventions, indentation, comments, formatting, variable declarations, function definitions, and error handling. In general, coding standards help to ensure that code is consistent, well-organized, and easy to read.

Naming Conventions

Naming conventions are one of the most important aspects of coding standards. They refer to guidelines for naming variables, functions, classes, and other program elements. Naming conventions help to ensure that code is easy to understand and that other developers can quickly identify the purpose of different program elements.

For example, in Java, the convention is to use camelCase notation for variable and method names. This means that the first word is lowercase, and each subsequent word is capitalized. So, a variable name might be "firstName" instead of "first_name". In Python, the convention is to use underscores to separate words, so a variable name might be "first_name" instead of "firstName".

Indentation and Formatting

Indentation and formatting are other important aspects of coding standards. They refer to the use of white space and other formatting elements to make code more readable and easier to understand. Good indentation and formatting can make it easier to identify different parts of the code and to understand the overall structure of the program.

For example, it is common to use four spaces for indentation in Python. This makes it easier to see the different levels of nesting in the code. In Java, it is common to use curly braces to define blocks of code, and to put the opening curly brace on the same line as the declaration, while the closing brace goes on its own line.

Comments

Comments are another important aspect of coding standards. Comments are used to explain what the code is doing, why it is doing it, and how it works. Good comments can help other developers to understand the code and to identify potential issues or bugs.

It is important to use comments sparingly and only when necessary. Comments should be clear and concise, and they should not duplicate information that is already apparent from the code itself. Comments should also be kept up-to-date, so that they accurately reflect the code as it evolves.

Variable Declarations and Function Definitions

Coding standards can also cover guidelines for variable declarations and function definitions. For example, it is common to declare all variables at the beginning of a function in C or C++, while in Java, variables are typically declared as close to their first use as possible.

Function definitions can also follow coding conventions. For example, in Python, functions are typically defined with a docstring that describes what the function does, its parameters, and its return value. In C++, it is common to use comments to describe the purpose of the function, its parameters, and its return value.

Error Handling

Coding standards can also include guidelines for error handling. Error handling is an important aspect of software development, and it is important to handle errors in a consistent and effective manner.

For example, in Java, it is common to use exception handling to handle errors. Exceptions are used to signal when something unexpected happens, and they can be caught and handled by the program. In Python, it is common to use the try-except block to handle errors.

Benefits of Coding Standards

Following coding standards has several benefits for software development. First, it helps to ensure that code is consistent and easy to read. This can make it easier to maintain and modify the code in the future.

APPLICATION

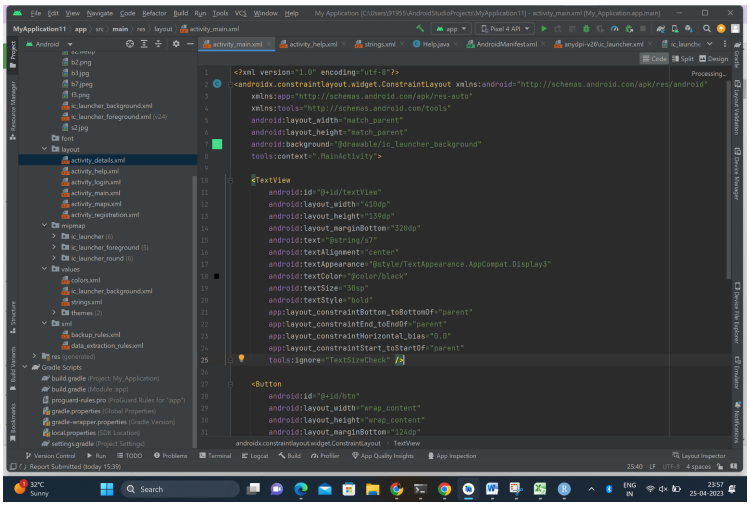

Fig 4.1. Application Code 1

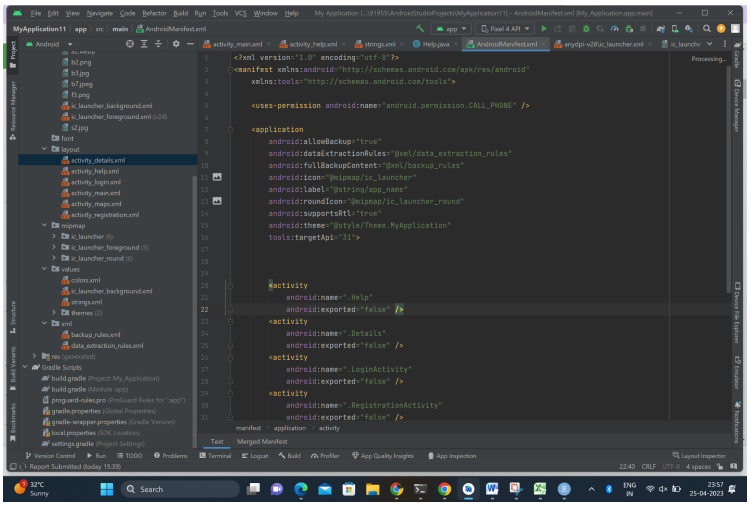

Fig 4.2. Application Code 2

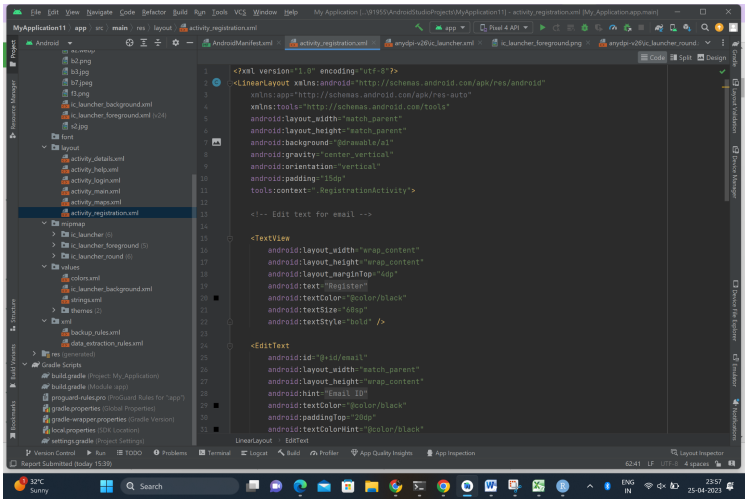

Fig 4.3: Application Code 3

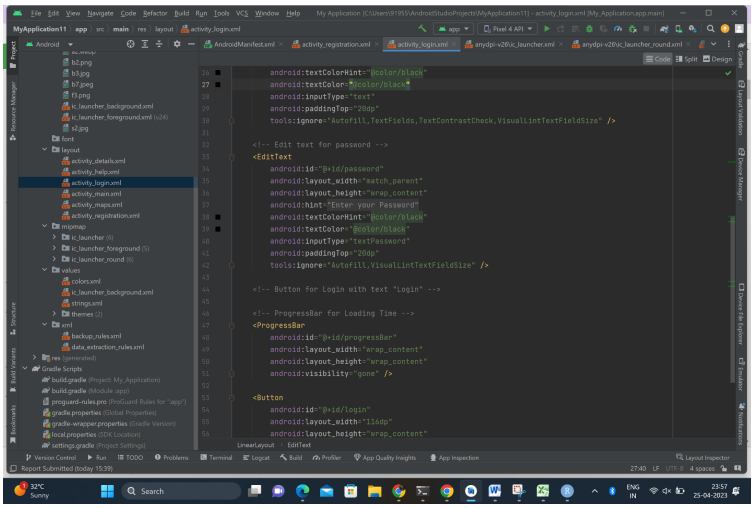

Fig 4.4: Application Code 4

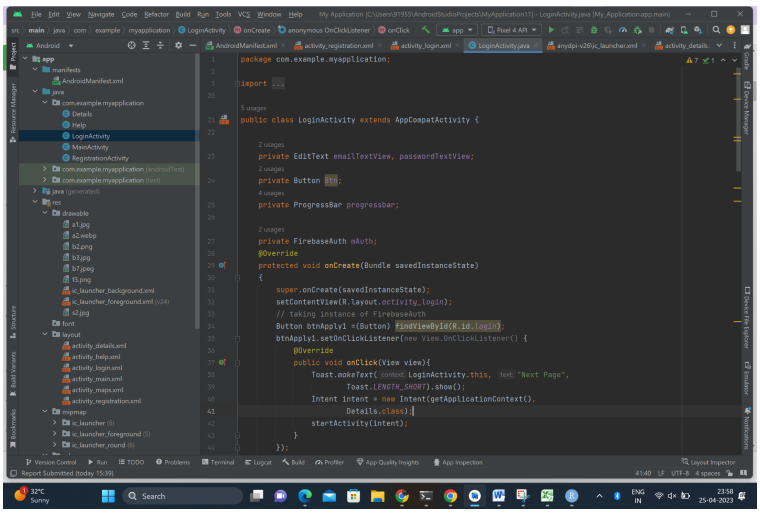

Fig 4.5: Application Code 5

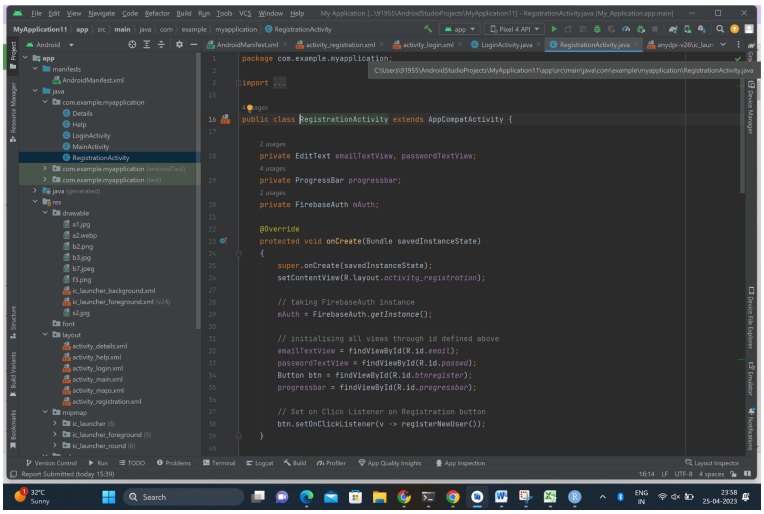

Fig 4.6: Application Code 6

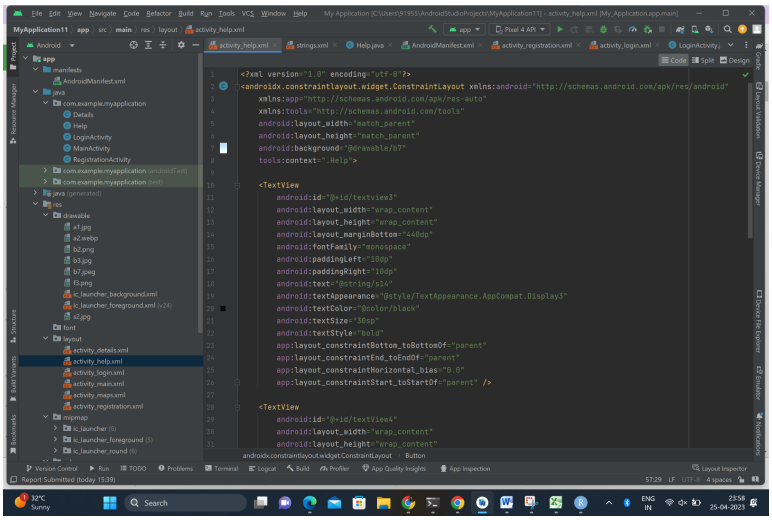

Fig 4.7. Application Code 7

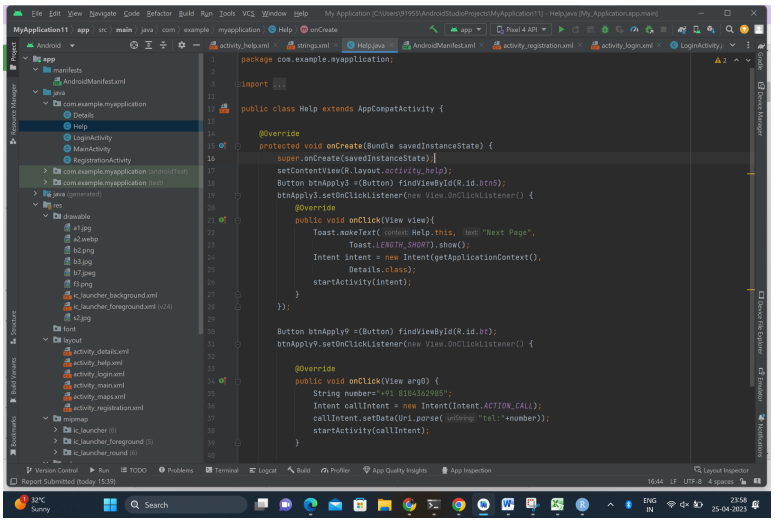

Fig 4.8: Application Code 8

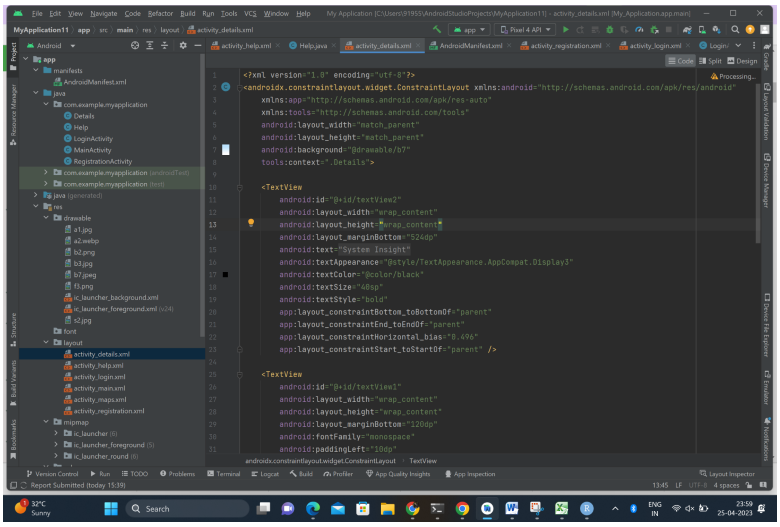

Fig 4.9: Application Code 9

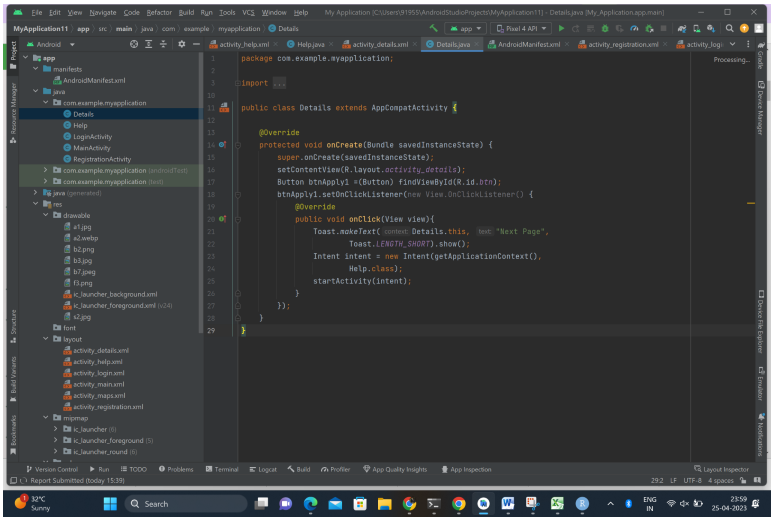

Fig 4.10: Application Code 10

```
roadNetworkDescriptor.size() ;
  }

  @AnyLogicInternalCodegenAPI
  public DataSet _plot1_expression0_dataSet_xjal = new DataSet( 240, new DataUpdater_xjal() {
    double _lastUpdateX = Double.NaN;
    @Override
    public void update( DataSet _d ) {
      if ( time() == _lastUpdateX ) { return; }
      _d.add( time(), _plot1_expression0_dataSet_xjal_YValue() );
      _lastUpdateX = time();
    }
  } );
```

Fig 4.11: Simulation Code Snippet 1

```
public DataSet timeInSystemDS = new DataSet( 100, new DataUpdater_xjal() {
  double _lastUpdateX = Double.NaN;
  @Override
  public void update( DataSet _d ) {
    if ( time() == _lastUpdateX ) { return; }
    _d.add( time(), 0 );
    _lastUpdateX = time();
  }
```

Fig 4.12: Simulation Code Snippet 2

```
private void _initialize_roadNetwork_xjal() {
    roadNetwork.addAll(road4, stopLine9, stopLine8, road, intersection1, intersection, road5, stopLine2, road6, stopLine5, intersection2, road7, stopLine4, road3, road8
    trafficLight = _trafficLight_xjal();
    roadNetwork.add(trafficLight);
    trafficLight1 = _trafficLight1_xjal();
    roadNetwork.add(trafficLight1);
}
```

Fig 4.13: Simulation Code Snippet 3

```
intersection = new Intersection( this, SHAPE_DRAW_2D3D, true, true, new PathEnd[] {
        new PathEnd( road5, PathEndType.END ),
        new PathEnd( road6, PathEndType.END ),
        new PathEnd( road8, PathEndType.BEGIN ),
        new PathEnd( road1, PathEndType.BEGIN ),
        new PathEnd( road2, PathEndType.END ),
    }, _intersection_laneConnectors_xjal()
```

Fig 4.14: Simulation Code Snippet 4

```
@AnyLogicInternalCodegenAPI
public static LinkToAgentAnimationSettings _connections_commonAnimationSettings_xjal = new LinkToAgentAnimationSettingsImpl( false, black, 1.0, LINE_STYLE_SOLID, ARROW_

public LinkToAgentCollection<Agent, Agent> connections = new LinkToAgentStandardImpl<Agent, Agent>(this, _connections_commonAnimationSettings_xjal);
@Override
public LinkToAgentCollection<? extends Agent, ? extends Agent> getLinkToAgentStandard_xjal() {
  return connections;
}


@AnyLogicInternalCodegenAPI
public void drawLinksToAgents(boolean _underAgents_xjal, LinkToAgentAnimator _animator_xjal) {
  super.drawLinksToAgents(_underAgents_xjal, _animator_xjal);
  if ( _underAgents_xjal ) {
    _animator_xjal.drawLink( this, connections, true, true );
  }
}

public List<Object> getEmbeddedObjects() {
  List<Object> list = super.getEmbeddedObjects();
  if (list == null) {
    list = new LinkedList<>();



public AgentList<? extends Main> getPopulation() {
  return (AgentList<? extends Main>) super.getPopulation();
}

public List<? extends Main> agentsInRange( double distance ) {
  return (List<? extends Main>) super.agentsInRange( distance );
}
```

Fig 4.15: Simulation Code Snippet 5

```
  @Override
  @AnyLogicInternalCodegenAPI
  public void doCreate() {
    super.doCreate();
    // Creating embedded object instances
    instantiatePopulations_xjal();
    // Assigning initial values for plain variables
    setupPlainVariables_Main_xjal();
    // Dynamic initialization of persistent elements
    _createPersistentElementsAP0_xjal();
    _initialize_level_xjal();
    _initialize_roadNetwork_xjal();
    level.initialize();
    presentation = new ShapeTopLevelPresentationGroup( Main.this, true, 0, 0, 0, 0 , level );
    presentation.getConfiguration3D().setBackgroundColor( oliveDrab );
    window3d.setCamera( camera, false );
// Creating replicated embedded objects
    setupInitialConditions_xjal( Main.class );
    // Dynamic initialization of persistent elements
    _createPersistentElementsBS0_xjal();
    checkbox1.setValueToDefault();
    cbxCameraFollowToCar.setValueToDefault();
    checkbox7.setValueToDefault();
  }
```

Fig 4.16: Simulation Code Snippet 6

4.2. SCREENSHOTS

Simulation

Fig 4.17: Traffic Simulation 1

Fig 4.18: Traffic Simulation 2

Fig 4.19: Traffic Simulation 3

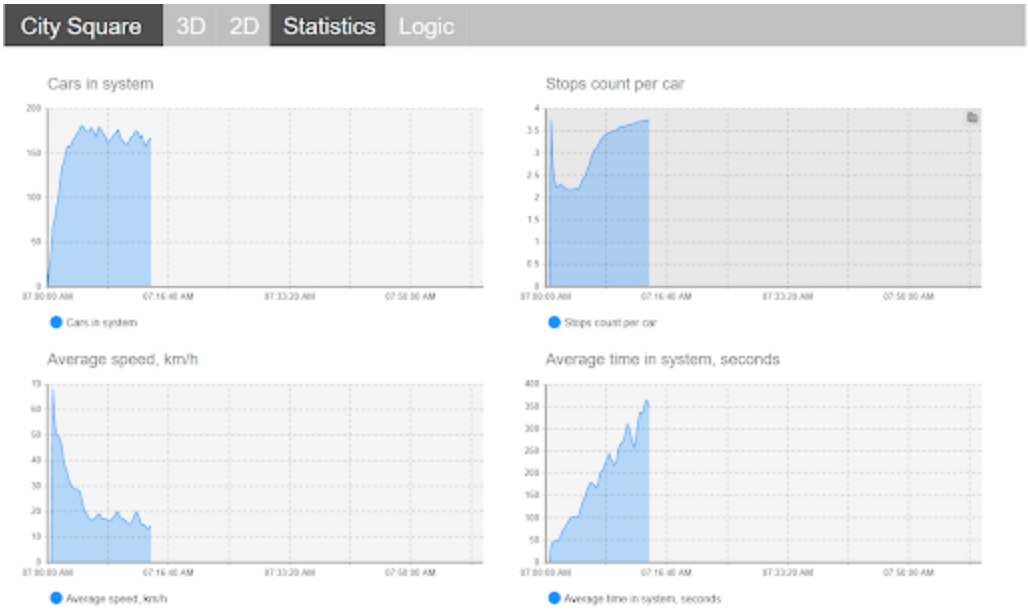

Fig 4.20: Traffic Simulation Statistics

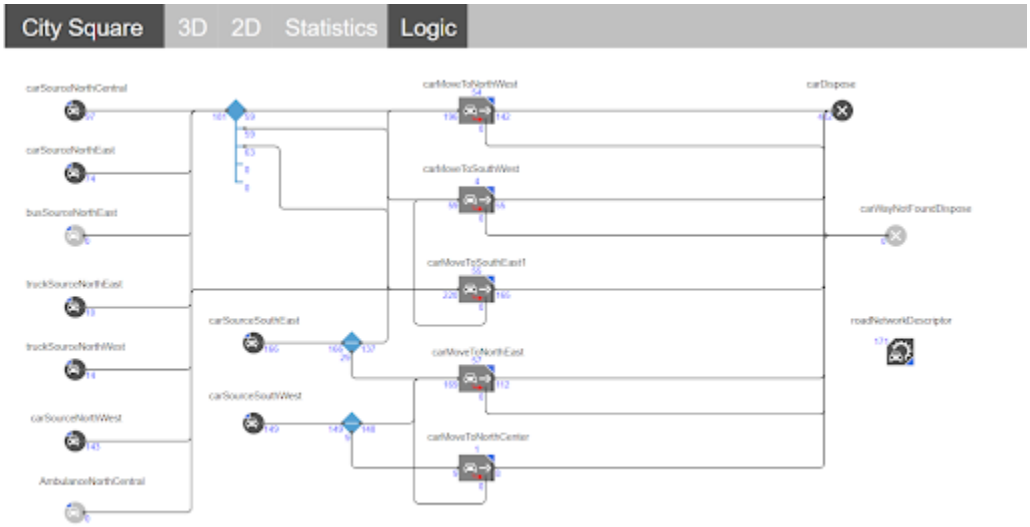

Fig 4.21: Traffic Simulation Logic

Application

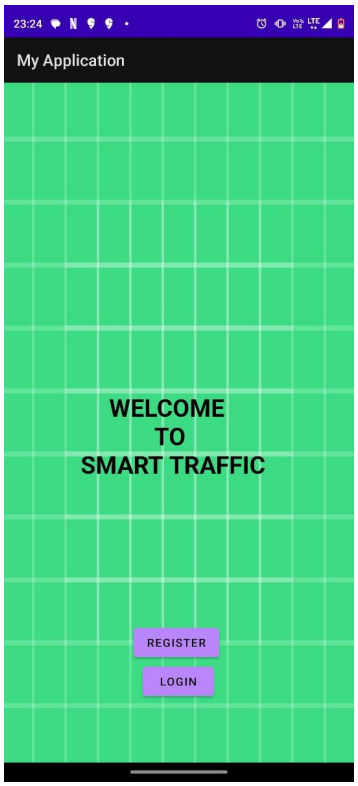

Fig 4.22: Application Window 1

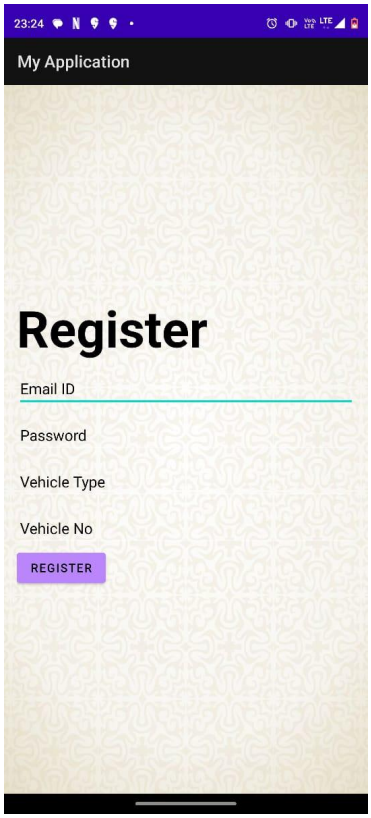

Fig 4.23: Application Window 2

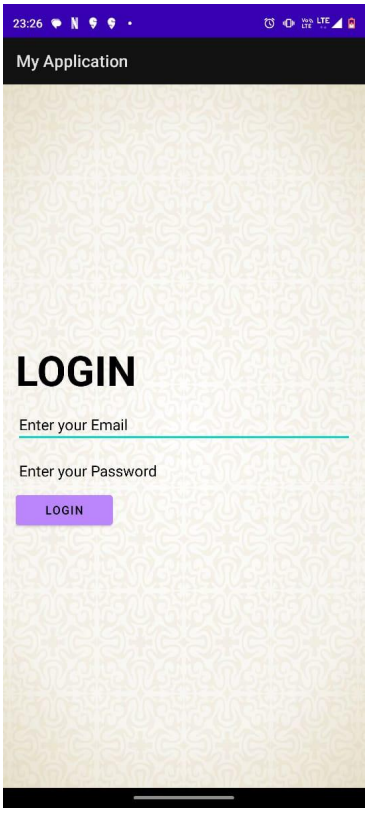

Fig 4.24: Application Window 3

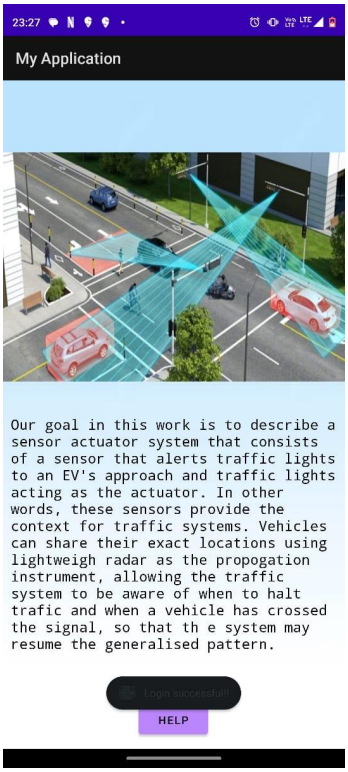

Fig 4.25: Application Window 4

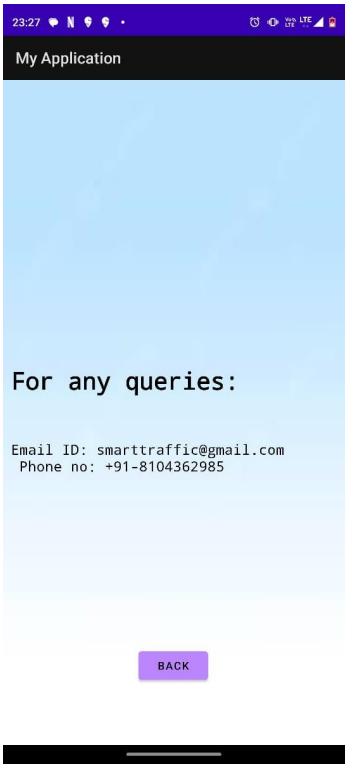

Fig 4.26: Application Window 5

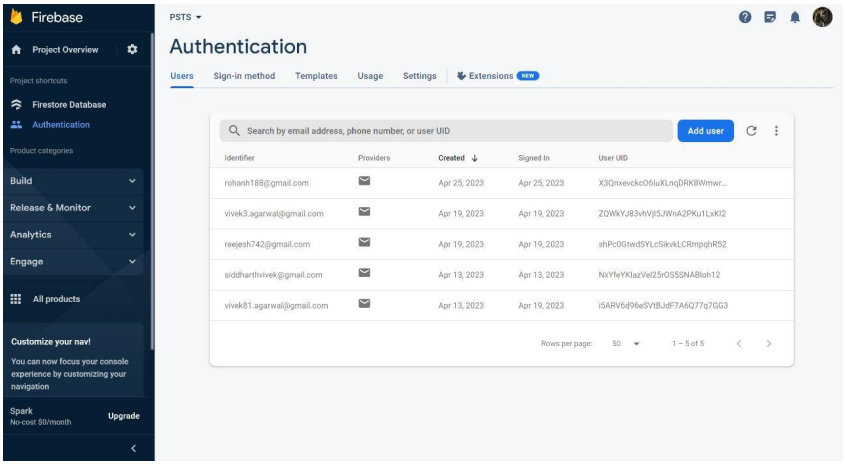

Fig 4.27: Application Database

5. TESTING

Firebase Authentication is a service provided by Google's Firebase platform that allows developers to easily add user authentication to their applications.

Firebase Authentication stores user data such as email addresses and passwords securely in a Firebase Realtime Database or Cloud Firestore. These databases are NoSQL databases that can store and sync data in real time between multiple clients and platforms.

Firebase Authentication supports several authentication methods, including email and password, phone number, Google, Facebook, Twitter, GitHub, and more. Developers can choose the authentication methods that work best for their applications and users.

Using Firebase Authentication, developers can easily manage user authentication, including account creation, email verification, password reset, and more. Firebase Authentication also provides a secure way to store and manage user data, ensuring that user data is safe and accessible only to authorized users.

In summary, Firebase Authentication uses a Firebase Realtime Database or Cloud Firestore to securely store and manage user authentication data, making it easy for developers to add user authentication to their applications.

Firebase Authentication uses industry-standard security measures like OAuth 2.0 and OpenID Connect to authenticate users and protect their data.

In addition to email and password authentication, Firebase Authentication also supports several other methods, including anonymous authentication, custom authentication, and federated identity providers like Apple, Microsoft, and Yahoo.

Firebase Authentication can be integrated with other Firebase services, such as Firebase Cloud Messaging, Firebase Analytics, and Firebase Remote Config, to create a complete end-to-end solution for app development.

Firebase Realtime Database is a NoSQL database that can store and sync data in real time between multiple clients and platforms, including mobile devices, web browsers, and backend servers.

Cloud Firestore is a NoSQL document database that can also store and sync data in real time, but offers additional features like flexible querying and indexing, document-level security, and automatic scaling.

Firebase Authentication stores user data securely in either the Firebase Realtime Database or Cloud Firestore, depending on which database you choose to use for your app.

To integrate Firebase Authentication with your app, you'll need to set up a Firebase project, configure the authentication providers you want to use, and add the necessary code to your app to handle user authentication and data management. Firebase provides detailed documentation and code samples to help you get started.

6. CONCLUSION

In conclusion, the development of a traffic system that prioritizes emergency vehicles is an essential step towards improving emergency response times and reducing the number of fatalities. The simulation and application we have proposed will aid in achieving this goal by providing real-time information on the location and status of emergency vehicles and simulating traffic scenarios to ensure their smooth passage.

We started with a system analysis that helped us identify the problem and the requirements of the project. We then developed a conceptual model and data flow diagrams to better understand the flow of information within the system. Next, we designed the system architecture, including the module and database designs, and configured the system accordingly.

To ensure a user-friendly experience, we also designed the user interface of the application and the flow of the application. The application will display critical information about the emergency vehicle and its location on a map. This will help emergency services to manage their operations more efficiently and respond quickly to emergencies.

The report also discussed the constraints that are likely to arise when making a priority-based smart traffic system on a small scale, such as the availability of resources, limited data, and the need for accurate data collection. It is essential to address these constraints before implementing the system to ensure its success.

During the implementation phase, we followed coding standards to ensure that the software is reliable, efficient, and easy to maintain. We also tested the system extensively to ensure its functionality and to identify any potential issues. Our testing approach included the creation of test cases and test reports that helped us verify the system's performance.

In conclusion, the proposed traffic system that prioritizes emergency vehicles has the potential to improve emergency response times, save lives, and reduce the number of fatalities. The simulation and application we have developed will help emergency

services manage their operations more efficiently, provide real-time information on the location and status of emergency vehicles, and simulate traffic scenarios to ensure their smooth passage. We recommend that future research focuses on the scalability and sustainability of the system, as well as its integration with other smart city technologies.

With investment in smart city infrastructure and the use of emerging technologies, we can improve the quality of life for citizens and make our cities safer, more efficient, and more resilient.

7. REFERENCES

[1] M. Asaduzzaman and K. Vidyasankar, "*A Priority Algorithm to Control the Traffic Signal for Emergency Vehicles*," 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, ON, Canada, 2017, pp. 1-7, doi: 10.1109/VTCFall.2017.8288364.

[2] Wenwen Kang, Gang Xiong, Yisheng Lv, Xisong Dong, Fenghua Zhu and Qingjie Kong, "*Traffic signal coordination for emergency vehicles*," 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, 2014, pp. 157-161, doi: 10.1109/ITSC.2014.6957683.

[3] C. -W. Tan, S. Park, H. Liu, Q. Xu and P. Lau, "*Prediction of Transit Vehicle Arrival Time for Signal Priority Control: Algorithm and Performance*," in IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 4, pp. 688-696, Dec. 2008, doi: 10.1109/TITS.2008.2006799.

[4] D. Fagan and R. Meier, "*Intelligent time of arrival estimation*," 2011 IEEE Forum on Integrated and Sustainable Transportation Systems, Vienna, Austria, 2011, pp. 60-66, doi: 10.1109/FISTS.2011.5973595.

[5] Wanjing Ma, Yue Liu & Xiaoguang Yang (2013) "*A Dynamic Programming Approach for Optimal Signal Priority Control Upon Multiple High-Frequency Bus Requests"*, Journal of Intelligent Transportation Systems, 17:4, 282-293, DOI: 10.1080/15472450.2012.729380