

Chapter 1

Introduction

1.1 Introduction:

“Swift Assist” is an innovative and user-friendly application designed to provide efficient and seamless assistance during vehicle breakdown emergencies. It addresses the inefficiencies and delays associated with traditional roadside assistance methods by leveraging technology to deliver real-time solutions, comprehensive service options, and enhanced safety features. The app aims to ensure that drivers receive prompt and effective support during emergencies, offering peace of mind and confidence on the road. With a hassle-free registration process, users gain access to a curated list of approved mechanics, guaranteeing reliable and trustworthy assistance. The app’s advanced search feature enables users to locate mechanics based on their proximity and availability, ensuring timely resolutions for breakdowns. Furthermore, users can share valuable feedback after utilizing services, fostering continuous improvement and enhancing the overall user experience. The Swift Assist platform goes beyond basic breakdown assistance by providing additional features such as real-time location tracking for accurate service delivery, a directory of nearby gas stations and mechanic shops, and a unique bidding system that allows service providers to compete, ensuring competitive pricing. It also integrates options for purchasing vehicle spare parts and offers home service delivery, enhancing convenience and reliability. By integrating technology with practical solutions, Swift Assist revolutionizes the roadside assistance experience, offering drivers a dependable and comprehensive tool for managing vehicle emergencies. This initiative underscores the commitment to ensuring a seamless, stress-free driving experience while prioritizing safety and convenience for all users. With Swift Assist, drivers can navigate the roads confidently, knowing help is just a tap away.

In conclusion, the Swift Assist App is more than just a technological tool; it is a lifeline for drivers facing vehicle breakdowns and emergencies on the road. By combining the immediacy of real-time assistance features with the strategic advantage of user-friendly services and reliable connections, the app creates a comprehensive framework for roadside support. Its commitment to innovation and user empowerment ensures that it continually evolves to meet the changing needs of drivers. The Swift Assist App stands as a testament to the power of technology in providing safer, more convenient, and reliable driving experiences for everyone.

1.2 Motivation:

The Swift Assist App is an easy-to-use mobile platform designed to deliver swift assistance during vehicle breakdowns while enhancing road safety for users. Its main objective is to provide drivers and vehicle owners with a dependable solution for receiving immediate help during road emergencies or unforeseen situations. Key features of the app include real-time GPS location tracking, access to nearby roadside services such as towing, mechanics, and emergency fuel delivery, as well as information about gas stations, service centers, and spare parts shops. Users can quickly notify selected contacts or service providers about their situation and receive updates on the estimated time of arrival for assistance. The app addresses the increasing demand for a reliable and efficient breakdown support system in today’s busy world, empowering users to resolve vehicle issues effortlessly with the help of smartphone and GPS technology. Its straightforward and user-friendly interface ensures accessibility for people of all skill levels.

The app’s development focuses on integrating advanced location-tracking technology, robust communication systems, and a wide network of service providers. Its design emphasizes simplicity and efficiency, enabling smooth navigation and a seamless user experience. Additionally, it promotes community involvement by

allowing users to share feedback, reviews, and ratings for service providers, fostering trust and transparency within the platform.

The Swift Assist App is an innovative response to modern roadside assistance challenges. By blending cutting-edge technology with a user-focused approach, it equips drivers with a powerful tool to manage vehicle breakdowns effectively. With its all-inclusive features and intuitive interface, the Swift Assist App is poised to become an indispensable resource for ensuring a safer and more connected driving experience.

1.3 Problem Statements:

Developing the Swift Assist App tackles key challenges faced by drivers during vehicle breakdowns and roadside emergencies. A major concern is the inability to promptly connect with roadside assistance or trusted services, causing delays in getting help and resolving issues. Furthermore, many drivers find it difficult to share their precise location in real time, making it harder for service providers or helpers to locate them quickly and offer timely assistance. The Swift Assist App aims to address key challenges faced by drivers during vehicle breakdowns and emergencies. These challenges include delays in getting help, difficulty finding nearby services, and ensuring safety and convenience.

i). Lack of Immediate help during Vehicle Breakdowns:

Drivers often face delays in getting help when their vehicle breaks down because they can't quickly contact roadside assistance or inform someone for help. How can the Swift Assist App ensure drivers can instantly contact roadside assistance services like towing, mechanics, or emergency fuel delivery?

ii). Inaccurate or Missing Real-Time Location Tracking:

Finding stranded drivers quickly depends on reliable and accurate real-time location tracking, which many current solutions lack. Without precise location sharing, service providers and contacts face difficulties locating the driver.

iii). Challenges in Finding Nearby Essential Services:

In breakdown situations, drivers may need services like gas stations, repair shops, towing services, or spare parts stores. However, locating these essential services can be challenging, particularly in unfamiliar or remote areas. How can the Swift Assist App provide users with a comprehensive directory of nearby services based on their location?

iv). Complex Interfaces Leading to Delayed Responses:

Many existing apps have complicated interfaces that are difficult to use, especially during stressful situations like vehicle breakdowns. Drivers may struggle to navigate through multiple steps to access help.

v). Concerns About Privacy and Data Security:

Sharing real-time location and personal data is essential for timely help but also raises concerns about privacy and unauthorized access. Drivers need assurance that their information is secure.

How can the Swift Assist App keep user data safe from misuse while still providing quick and effective help?

vi). Enhancing Accessibility and Dependability through Technology:

Despite technological advancements, drivers often lack a dependable solution for handling breakdowns efficiently. How can the Swift Assist App empower drivers with a reliable tool that enhances their ability to manage vehicle emergencies, promoting confidence and road safety for all?

By addressing these challenges, the Swift Assist App aims to be a comprehensive and reliable solution for vehicle breakdown management, ensuring faster assistance, improved safety, and greater convenience for drivers.

1.4 Objectives

1.4.1 Specific Objectives:

- i. Quick Access to Assistance: Develop a platform that enables users to swiftly connect with dependable breakdown assistance services, reducing delays and minimizing downtime.
- ii. Location-Based Mechanic Discovery: Integrate a feature for users to easily find certified mechanics nearby, based on real-time location and availability.
- iii. Efficient Communication: Establish seamless channels for users to communicate directly with service providers, ensuring timely resolutions to breakdown incidents.
- iv. Data Privacy and Security: Implement strong security protocols to protect user data and maintain their privacy.
- v. User-Centric Improvement: Design mechanisms to gather and analyze user feedback, ensuring continuous improvement of services and responsiveness.

1.4.2 Overall Objectives:

- i. Improve Roadside Assistance: Offer a reliable and efficient solution to enhance personal safety for individuals dealing with vehicle breakdowns.
- ii. Enable Quick Response: Allow users to send rapid alerts to emergency services and contacts in critical roadside situations.
- iii. Precise Location Tracking: Provide accurate, location tracking to help locate users quickly in case of vehicle issues.
- iv. Access to Nearby Help: Enable users to easily find nearby tow services, repair shops, gas stations, and emergency providers.
- v. Intuitive User Interface: Design a straightforward interface that ensures users can quickly access key features during roadside emergencies.

By achieving these goals, the "Swift Assist" app aims to provide a practical and reliable tool that enhances user safety and convenience in the event of vehicle breakdowns, fostering a sense of security and community support.

1.5 Addressing PO, CO, Knowledge Profile (KP), Complex Engineering Problem (CP) and Complex Engineering Activities (CA)

Swift Assist Application: Comprehensive Roadside Vehicle Breakdown System.

"Swift Assist" is an innovative roadside vehicle breakdown system that provides drivers with immediate assistance during vehicle issues. Utilizing real-time data from traffic cameras, sensors, and GPS, Swift Assist identifies the exact location of a breakdown and offers a range of nearby services. These include gas stations for refueling, public toilets for convenience, utility shops for essential supplies, and mechanic shops or available mechanics for professional help. Swift Assist ensures swift and efficient support, minimizing downtime and enhancing driver safety and convenience.

1. Complex Engineering Problems (CP) with Swift Assist:

- i. **Predictive Maintenance:** Developing predictive maintenance algorithms to anticipate vehicle issues before they occur based on historical data and real time diagnostics.
- ii. **Integration with Smart City Infrastructure:** Collaborating with smart city initiatives to leverage existing infrastructure such as traffic management systems and IoT networks for enhanced service delivery.

- iii. **Real-time Fleet Management:** Implementing features to support fleet operators in managing multiple breakdown incidents simultaneously and optimizing resource allocation.
- iv. **Cross-border Compatibility:** Addressing challenges related to cross-border travel by ensuring Swift Assist can seamlessly transition between different regulatory environments and service standards.
- v. **Customer Feedback Mechanisms:** Implementing feedback loops to gather user insights and preferences, allowing continuous improvement of Swift Assist CO2: Utilize real-time data and apply modern tools and techniques to enhance Swift Assist's monitoring and response capabilities for roadside assistance features and services.
- vi. **Disaster Preparedness:** Designing Swift Assist to handle emergencies and natural disasters by integrating with emergency response protocols and disaster management systems.
- vii. **Environmental Impact Assessment:** Conducting assessments to understand and mitigate the environmental impact of Swift Assist operations, promoting sustainability in roadside assistance solutions
- viii. **Regulatory Compliance:** Ensuring compliance with local and international regulations governing data privacy, vehicle safety, and roadside assistance services.

2. Complex Engineering Activities (CA) with Swift Assist:

- i. **Project Planning:** Defining project scope, objectives, and deliverables for integrating Swift Assist into the roadside vehicle breakdown system.
- ii. **Team Collaboration:** Assigning roles and responsibilities to ensure effective coordination and communication throughout the project.
- iii. **Research and Requirements Gathering:** Conducting market research and gathering user requirements to inform the design and development of Swift Assist features.
- iv. **Data Collection and Analysis:** Gathering, processing, and analyzing traffic and vehicle breakdown data to develop predictive algorithms and service recommendations.
- v. **Algorithm Development:** Designing and refining algorithms to accurately predict traffic congestion, detect vehicle breakdowns, and recommend optimal routes and nearby services.
- vi. **System Architecture Design:** Designing the architecture for integrating sensors, IoT devices, software, and algorithms to support real-time traffic management and vehicle assistance.
- vii. **Testing and Validation:** Conducting rigorous testing in controlled environments to validate system performance, algorithm accuracy, and user satisfaction.
- viii. **System Integration:** Integrating hardware and software components into a cohesive system that enables seamless data flow and service delivery through Swift Assist.
- ix. **Deployment and Rollout:** Planning and executing the deployment strategy for Swift Assist, including user training, support infrastructure setup, and initial operational testing.
- x. **Documentation and Reporting:** Creating comprehensive documentation, including technical specifications, user manuals, and progress reports, to ensure transparency and facilitate future maintenance and updates.

3. Course Outcome (CO):

- i. **CO1:** Analyze roadside vehicle breakdown scenarios to minimize traffic disruptions and optimize response times, focusing on driver safety.
- ii. **CO2:** Utilize real-time data and apply modern tools and techniques to enhance Swift Assist's monitoring and response capabilities for roadside assistance.
- iii. **CO3:** Enhance Swift Assist's service delivery and driver convenience through real-time data, predictive analytics, and optimized routing algorithms.
- iv. **CO4:** Design and validate experiments to improve Swift Assist's response times and service recommendations, using performance metrics and user feedback.
- v. **CO5:** Collaborate service providers to integrate Swift Assist into a comprehensive roadside assistance solution, ensuring seamless coordination
- vi. **CO6:** Communicate Swift Assist's benefits and capabilities to stakeholders, budgeting and financial activities, focusing on improved roadside management and driver safety, while educating drivers and service providers.
- vii. **CO7:** Ensure Swift Assist complies with safety, legal, and ethical standards, protecting data privacy and sustainability, while innovating with technologies in a predictive maintenance.

PO, CO, Knowledge Profile (KP), Complex Engineering Problem (CP) and Complex Engineering Activities (CA) Mapping

Course Outcomes(COs)	Program Outcomes(POs)	Complex Engineering Problem (P1-P7)	Complex Engineering Activities (A1- A5)	Knowledge Profile (K1-K8)
CO1	PO1,PO2,PO4,PO6	P1,,P3,P4	A1,A4	K3,K4,K5,K6
CO2	PO1,PO2,PO5,PO12	P1,P4,P7	A1,A3,A4	K8
CO3	PO2,PO3,PO4, PO5	P2,P3,P5	A1,A3	
CO4	PO3,PO4,PO6, PO9	P3,P4,P6,P7	A2,A3,A4	
CO5	PO8,PO9,PO10	P5,P6,P7	A2,A4,A5	
CO6	PO7,PO10,PO11	P3,P6,P7	A2,A5	
CO7	PO6,PO7,PO8,PO12	P3,P4,P5	A1,A3,A4	

Table no.1 : CO, PO, KP, CP, CA table

Justification For Program Outcome PO), Complex Engineering Problem (CP) and Complex Engineering Activities(CA), Knowledge Profile (KP)

Below is justification for Program Outcomes (PO), Complex Engineering Problems (CP), Complex Engineering Activities (CA), and Knowledge Profile (KP) related to "Swift Assist Application" project.

CO1: Analyze roadside vehicle breakdown scenarios to minimize traffic disruptions and optimize response times, focusing on driver safety. Program Outcomes (POs):

Program Outcomes (POs):

- i. PO1: Engineering knowledge- Essential for understanding the technical aspects of vehicle breakdowns and devising effective solutions. PO2: Problem Analysis- Critical for identifying the root causes of breakdowns and potential traffic disruptions.

- ii. PO4: Investigation- Involves researching various scenarios and collecting data to develop optimal response strategies.
- iii. PO6: The Engineer and Society- Focuses on ensuring the safety and well-being of drivers, reflecting the societal impact of engineering solutions.

Complex Engineering Problems (P1-P7):

- i. P1: Depth of knowledge required- In-depth engineering knowledge is necessary to thoroughly understand and analyze breakdown scenarios. P3: Depth of analysis required- Requires detailed and thorough analysis to develop effective and efficient solutions.
- ii. P4: Familiarity with issues- Understanding common breakdown scenarios and their potential impact on traffic and driver safety.

Complex Engineering Activities (A1-A5):

- i. A1: Range of resources- Requires utilizing various tools and resources to analyze and respond to breakdowns.
- ii. A4: Consequences to society and environment - Considering how breakdowns and their management affect traffic flow and public safety.

Knowledge Profile (K1-K8):

- i. K3: A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline- Applying foundational engineering theories to analyze breakdown scenarios.
- ii. K4: Engineering specialist knowledge - Specialized knowledge in roadside assistance and traffic management.
- iii. K5: Knowledge that supports engineering design in a practice area- Designing solutions to mitigate the impact of breakdowns.
- iv. K6: Knowledge of engineering practice (technology) in the practice areas in the engineering discipline- Practical application of technologies to address and manage breakdowns.

CO2: Utilize real-time data and apply modern tools and techniques to enhance Swift Assist's monitoring and response capabilities for roadside assistance.

Program Outcomes (POs):

- i. PO1: Engineering knowledge- Necessary for understanding and applying advanced tools and technologies.
- ii. PO2: Problem Analysis- Analyzing real-time data to improve monitoring and response.
- iii. PO5: Modern Tools Usage- Employing the latest tools and techniques to enhance response capabilities.
- iv. PO12: Life-long learning - Staying updated with new technologies and continuously improving knowledge and skills.

Complex Engineering Problems (P1-P7):

- i. P1: Depth of knowledge required - Requires extensive understanding of data analysis and technology.

- ii. P4: Familiarity with issues- Understanding how real-time data impacts response times and effectiveness.
- iii. P7: Interdependence- Integration and coordination of various data sources and systems.

Complex Engineering Activities (A1-A5):

- i. A1: Range of resources- Utilizing diverse data sources and modern tools.
- ii. A3: Innovation - Implementing cutting-edge technologies and innovative solutions.
- iii. A4: Consequences to society and environment- Assessing how improved monitoring and response benefit public safety and traffic flow.

Knowledge Profile (K1-K8):

- i. K8: Engagement with selected knowledge in the research literature of the discipline Applying insights from current research to enhance monitoring and response systems.

CO3: Enhance Swift Assist's service delivery and driver convenience through real-time data, predictive analytics, and optimized routing algorithms.

Program Outcomes (POs):

- i. PO2: Problem Analysis - Analyzing data to identify and address service delivery issues.
- ii. PO3: Design/Development of Solutions - Designing systems that use real-time data and predictive analytics.
- iii. PO4: Investigation - Researching and developing advanced routing algorithms.
- iv. PO5: Modern Tools Usage - Utilizing state-of-the-art tools for data analysis and optimization.

Complex Engineering Problems (P1-P7):

- i. P2: Range of conflicting requirements - Balancing various requirements, such as speed, accuracy, and convenience.
- ii. P3: Depth of analysis required - Conducting detailed analysis to develop effective predictive models and algorithms.
- iii. P5: Extent of applicable codes - Ensuring compliance with relevant standards and **regulations**.

Complex Engineering Activities (A1-A5):

- i. A1: Range of resources - Leveraging multiple data sources and analytical tools.
- ii. A3: Innovation - Developing innovative algorithms and predictive models to improve service delivery.

CO4: Design and validate experiments to improve Swift Assist's response times and service recommendations, using performance metrics and user feedback.

Program Outcomes (POs):

- i. PO3: Design/Development of Solutions - Creating experiments to test and improve response strategies.
PO4: Investigation - Collecting and analyzing data to validate experimental results.
- ii. PO6: The Engineer and Society - Ensuring that improvements benefit society and enhance driver safety.
PO9: Individual and Teamwork - Collaborating with team members to design and conduct experiments.

Complex Engineering Problems (P1-P7):

- i. P3: Depth of analysis required - Conducting detailed analysis of performance metrics and feedback.

- ii. P4: Familiarity with issues - Understanding the factors that influence response times and service effectiveness.
- iii. P6: Extent of stakeholder involvement and conflicting requirements - Balancing feedback from various stakeholders to improve services.
- iv. P7: Interdependence - Coordinating efforts across different teams and systems.

Complex Engineering Activities (A1-A5):

- i. A2: Level of interactions - Engaging with various stakeholders, including users and team members. A3: Innovation - Developing new methods to enhance response times and service recommendations.
- ii. A4: Consequences to society and environment - Considering the broader impact of service improvements on society.

CO5: Collaborate with service providers to integrate Swift Assist into a comprehensive roadside assistance solution, ensuring seamless coordination.

Program Outcomes (POs):

- i. PO8: Ethics - Ensuring ethical collaboration and integration practices.
- ii. PO9: Individual and Teamwork - Working effectively in teams to achieve seamless coordination.
- iii. PO10: Communication - Communicating clearly with service providers to facilitate integration.

Complex Engineering Problems (P1-P7):

- i. P5: Extent of applicable codes - Ensuring compliance with industry standards and regulations during integration.
- ii. P6: Extent of stakeholder involvement and conflicting requirements - Managing different stakeholders' needs and requirements.
- iii. P7: Interdependence - Ensuring smooth coordination among multiple service providers.

Complex Engineering Activities (A1-A5):

- i. A2: Level of interactions - High level of interaction with service providers to ensure successful integration.
- ii. A4: Consequences to society and environment - Ensuring that the integrated solution benefits society.
- iii. A5: Familiarity with project management - Managing the integration project effectively, including planning and coordination.

CO6: Communicate Swift Assist's benefits and capabilities to stakeholders, budgeting and financial activities, focusing on improved roadside management and driver safety, while educating drivers and service providers.

Program Outcomes (POs):

- i. PO7: Environment and Sustainability - Highlighting the sustainable practices and benefits of Swift Assist.

- ii. PO10: Communication - Effectively communicating benefits, capabilities, and financial aspects to stakeholders.
- iii. PO11: Project Management and Finance - Managing budgeting and financial activities related to Swift Assist.

Complex Engineering Problems (P1-P7):

- i. P3: Depth of analysis required - Analyzing data to clearly communicate benefits and capabilities.
- ii. P6: Extent of stakeholder involvement and conflicting requirements - Balancing the needs and interests of various stakeholders.
- iii. P7: Interdependence - Coordinating with different stakeholders to ensure successful communication and education.

Complex Engineering Activities (A1-A5):

- i. A2: Level of interactions - Engaging with various stakeholders, including drivers and service providers.
- ii. A5: Familiarity with project management - Managing communication and financial aspects effectively.

CO7: Ensure Swift Assist complies with safety, legal, and ethical standards, protecting data privacy and sustainability, while innovating with technologies in predictive maintenance.

Program Outcomes (POs):

- i. PO6: The Engineer and Society - Ensuring that Swift Assist solutions are safe and ethical.
- ii. PO7: Environment and Sustainability - Promoting sustainable practices in predictive maintenance.
- iii. PO8: Ethics - Adhering to legal and ethical standards.
- iv. PO12: Life-long learning - Keeping up with evolving standards and technologies to ensure compliance.

Complex Engineering Problems (P1-P7):

- i. P3: Depth of analysis required - Detailed analysis to ensure compliance with safety, legal, and ethical standards.
- ii. P4: Familiarity with issues - Understanding the relevant safety, legal, and ethical issues.
- iii. P5: Extent of applicable codes - Ensuring compliance with relevant codes and standards.

Complex Engineering Activities (A1-A5):

- i. A1: Range of resources - Utilizing various resources to ensure compliance and innovation.
- ii. A3: Innovation - Innovating while adhering to standards.
- iii. A4: Consequences to society and environment - Considering the societal and environmental impact of innovations and compliance.

List of activities:

1. Create a Team - Teamwork
2. Idea sharing - Teamwork
3. Select project idea - Teamwork
4. Project proposal documentation

- i. Introduction to literature review - T
- ii. Existing system to methodology - M
- iii. Requirement to conclusion - S
- 5. Project Planning - Teamwork
- 6. Requirement gathering - Teamwork
- 7. Existing system analysis – S
- 8. Proposing New Methodology - Teamwork
- 9. Market analysis - T
- 10. Feasibility study
 - i. Technical feasibility - M
 - ii. Operation of feasibility - M
 - iii. Financial feasibility - S
 - iv. Legal and ethical feasibility - S
- 11. Risk analysis – S
- 12. Environmental Impact Assessment
- 13. Requirements specification documents - Teamwork
- 14. Communication to stakeholders to fix requirements - Teamwork
- 15. Budgeting - S
- 16. Cost benefit analysis - S
- 17. Project finalization with stakeholder - Teamwork
- 18. Business model selection – Teamwork
- 19. Software Model selection – Teamwork
- 20. SWOT analysis - M
- 21. Information gathering
 - i. Survey - T
 - ii. Interview - M
 - iii. Questionnaires - S
- 22. System design – M
- 23. Mapping the gathered information into data - T
- 24. Data flow diagram - S
- 25. Class diagram - S
- 26. Use case diagram - S
- 27. ER diagram - S
- 28. UI UX design
 - i. Login interface - T
 - ii. Sign up interface -T
 - iii. OTP getting interface-T
 - iv. User dashboard - T
 - v. Use a dashboard services list -M
 - vi. User profile - M
 - vii. User service request interface - M
 - viii. Service provider finding interface - M
 - ix. Service request details providing interface – M
 - x. Service Selection Details interface - M
 - xi. Location tracking interface - M
 - xii. Nearest gas station finding interface - M
 - xiii. Nearest public washroom finding interface - M
 - xiv. Nearest mechanic shop finding interface – M
 - xv. Nearest utility shop finding interface -
 - xvi. Mechanic dashboard interface - S
 - xvii. Mechanic profile interface - S
 - xviii. Mechanic services interface - S
 - xix. Service request accepting interface – S
 - xx. Select Service request interface – S
 - xxi. New Request checking interface -
 - xxii. User Location tracking interface - S

- xxiii. Admin dashboard - S
- xxiv. User and mechanic status checking interface - S
- xxv. Mechanic request approval interface - S
- xxvi. Active service provider interface – S
- xxvii. Payment Method Selection interface -
- xxviii. Payment interface – M
- xxix. Payment verification interface – M
- xxx. History Checking interface -S
- xxxi. Feedback interface - M

29. Project implementation (coding)

- i. Login design - T
- ii. Sign up design -T
- iii. OTP getting design -T
- iv. User dashboard - T
- v. Use a dashboard services list -M
- vi. User profile - M
- vii. User service request design - M
- viii. Service provider finding design - M
- ix. Service request details providing design – M
- x. Service Selection Details design - M
- xi. Location tracking design - M
- xii. Nearest gas station finding design - M
- xiii. Nearest public washroom finding design - M
- xiv. Nearest mechanic shop finding design – M
- xv. Nearest utility shop finding design
- xvi. Mechanic dashboard design - S
- xvii. Mechanic profile design - S
- xviii. Mechanic services design - S
- xix. Service request accepting design
- xx. Select Service request design – S
- xxi. New Request checking design -
- xxii. User Location tracking design - S
- xxiii. Admin dashboard - S
- xxiv. User and mechanic status checking design - S
- xxv. Mechanic request approval design - S
- xxvi. Active service provider design – S
- xxvii. Payment Method Selection design -
- xxviii. Payment design – M
- xxix. Payment verification design - M
- xxx. History Checking design – S
- xxxi. Feedback design

30. Google API purchase – Teamwork

31. API Management

32. Database creation - Teamwork

33. Entry of data to database - Teamwork

34. Frontend and Backend connection - Teamwork

35. Database connection - Teamwork

36. Merging all modules - Teamwork

37. Checking all modules – Teamwork

38. Performance Monitoring - Teamwork

39. Registration of nearby gas station - S

40. Registration of public washroom - T

41. Registration of mechanic shop – M

42. Registration of utility shop -

43. Shops listing - Teamwork

44. App testing - T

- i. Verification
 - ii. Validation
 - iii. Acceptance testing
 - iv. Functional testing
 - v. Nonfunctional testing
 - vi. System testing
45. Bug Fixing and Updates
46. Marketing and Advertising – Teamwork
47. Market Expansion Planning
48. Creating user manual - Teamwork
49. Training – Teamwork
50. Third-Party Verification
51. Final documentation – Teamwork
52. Stakeholder Reporting
53. Project delivery – Teamwork

Legend:

Kazi Mehnaz Nobi Supti (S)

Mahfuza Islam (M)

Tasnin Alam Jibon (T)

Teamwork (TW)

Colour indicator for Chart:

Kazi Mehnaz Nobi Supti (S) = Green

Mahfuza Islam (M) = Blue

Tasnin Alam Jibon (T) = Red

Teamwork (TW) = Purple

1.6 Report Layout:

This chapter lays the foundation for the Swift Assist App project, outlining its purpose, goals, expected benefits, and the essential planning and financial strategies required for its success. It provides an overview of the project's background, context, and rationale, introducing the Swift Assist App as a practical solution for efficiently managing vehicle breakdowns and delivering prompt assistance. The chapter highlights the app's key features, including On-Demand Towing Services, Live GPS Tracking, Mechanic Finder, and Emergency Assistance, emphasizing its aim to offer users accessible and dependable support during vehicle emergencies. The motivation section delves into the reasons driving the project's development, showcasing its significance in tackling common challenges associated with modern vehicle breakdowns. It stresses the need for a user-friendly app that combines real-time tracking, trustworthy service options, and smooth communication to alleviate stress during emergencies. The problem statement identifies specific issues the app seeks to address, such as delays in receiving roadside help, the absence of real-time service updates, difficulty finding nearby mechanics, and concerns over data privacy. The objectives are categorized into Overall

Objectives, such as enhancing the dependability of roadside services, and Specific Objectives, like developing a user-friendly interface, ensuring precise GPS tracking, and fostering collaborations with service providers. This chapter also explores the expected outcomes of the project, including faster response times and improved user satisfaction. By addressing these aspects, the chapter sets the stage for a successful and impactful implementation of the Swift Assist App.

1.7 Conclusion:

The creation of the "Swift Assist" app signifies a notable step forward in tackling the key challenges associated with vehicle breakdowns through cutting-edge and user-friendly mobile technology. This platform is designed to deliver an efficient and hassle-free solution by incorporating vital features such as location, on-demand roadside assistance, integration with service providers, and emergency notifications. By utilizing state-of-the-art technology, the app seeks to provide users with a reliable and intuitive tool that enhances ease, reduces downtime, and ensures safety during vehicle-related emergencies. A structured methodology was followed throughout the project from detailed planning and requirement gathering to precise design, development, and extensive testing to ensure the app achieves top-tier standards in dependability, user experience, and security. Both functional and non-functional requirements have been thoroughly addressed to create a solution that performs optimally while remaining easily accessible to a diverse range of users. Future advancements will prioritize incorporating user insights to continually refine the app's features and overall user experience. Enhancing data protection, optimizing scalability, and tailoring features to meet the varied needs of a diverse user base will further amplify the app's effectiveness and accessibility. In essence, "Swift Assist" signifies a groundbreaking innovation in the realm of vehicle breakdown management, providing users with a dependable and streamlined solution for addressing roadside issues. Its successful deployment will not only alleviate the frustration and challenges linked to breakdowns but also contribute to enhancing road safety, fostering user trust, and elevating overall transportation experience

Chapter 2

Background Study

2.1 Introduction:

In recent years, the rapid proliferation of mobile technology has revolutionized various aspects of everyday life, offering innovative solutions to long-standing challenges. One area that has seen significant attention is vehicle assistance, a critical issue that persists globally despite advancements in technology and infrastructure. Vehicle owners often face challenges such as breakdowns, fuel shortages, or flat tires, which can be stressful and inconvenient, especially in unfamiliar or remote locations. Traditional methods of seeking help, such as contacting roadside assistance services or relying on nearby help, often prove inadequate due to delays, lack of immediate access, or difficulty in communicating the issue effectively. In response to these challenges, the development of specialized mobile applications has emerged as a promising solution. Among these, the "Swift Assist" App stands out as a comprehensive platform designed to address vehicle assistance concerns in real time. This application leverages cutting-edge technology to provide users with critical features, including live GPS tracking, a panic button for emergencies, and seamless integration with nearby services such as mechanics, gas stations, and utility shops. By offering a centralized and user-friendly platform, the app aims to empower vehicle owners with a reliable tool that simplifies assistance during critical situations. However, the development of such an application requires a deep understanding of the technological, social, and practical contexts that shape its use. Technologically, the app must integrate advanced features like GPS tracking, secure communication channels, and data encryption to ensure both functionality and privacy. Socially, it must address the unique challenges faced by vehicle owners in diverse environments, considering geographical, cultural, and situational factors. Practically, the app must be intuitive and accessible, ensuring that users can activate its features quickly and easily in times of need. Beyond its technical capabilities, the "Swift Assist" App has the potential to contribute significantly to broader societal goals, particularly in enhancing road safety and convenience. By providing a reliable tool that streamlines assistance, the app not only helps vehicle owners feel more secure but also fosters trust and confidence on the road. In doing so, it addresses systemic inefficiencies in roadside assistance and promotes a more connected and supportive ecosystem for vehicle users. This background study delves into the multifaceted aspects of the "Swift Assist" App, exploring how it is informed by the latest developments in mobile technology, the ongoing discourse around vehicle assistance, and practical considerations for its implementation. Through this exploration, it aims to highlight the app's potential to be a transformative tool in improving road safety, convenience, and user confidence on a global scale.

2.2 Literature Review:

1. Zantrik

Zantrik provides vehicle health tracking, predictive maintenance, emergency roadside assistance, and online payments. However, it lacks features such as live location for nearest service points, free towing to the nearest garage, a comprehensive car breakdown service, and a rating system. [15]

2. Sheba.xyz

Sheba.xyz offers emergency roadside assistance, including services like towing, battery jump-starts, and tire changes. While it provides prompt support, specific details about live location tracking for service points, free towing policies, and user rating systems are not explicitly mentioned. [16]

3. Life Tracker BD

Life Tracker BD focuses on safety during travel, offering features like automated emergency alerts, real-time location sharing, and automated calls and SMS to preferred contacts in case of accidents. It also includes a women safety service. However, it does not provide direct roadside assistance services such as towing or repairs.[17]

4. RoadHop

RoadHop is an on-demand mobile platform offering transportation, logistics, and other lifestyle services. While it emphasizes convenient pick-and-drop services, safety, and advance booking, specific roadside assistance features like towing, vehicle health tracking, or a rating system are not detailed. [17]

5. Finder GPS Tracker

Finder provides a vehicle tracking system aimed at preventing theft, offering features like live tracking, geo-fencing, ignition alerts, and an SOS button. While it enhances vehicle security, it does not offer roadside assistance services such as towing or repairs. [18]

2.3 Comparative Study:

“Swift Assist” is a specialized safety and assistance app designed to cater to diverse user needs with a focus on user-friendliness, real-time support, and comprehensive safety solutions.

Here’s how it compares with other safety and assistance apps:

1. **Broader User Base:** “Swift Assist” serves diverse users by offering vehicle and utility services, unlike apps with limited audiences.
2. **Vehicle Breakdown Assistance:** Provides instant access to mechanics and towing services for quick resolution of breakdowns.
3. **Mechanic Service Integration:** Connects users with verified mechanics directly through the app.
4. **Live GPS Tracking:** Offers real-time location sharing for accurate navigation and assistance.
5. **Nearby Gas Stations and Utility Shops:** Lists essential resources with directions and operational details.
6. **User Feedback System:** Allows continuous improvement based on user experiences and suggestions.
7. **User-Friendly Design:** Ensures easy navigation and quick access during emergencies.

2.4 Challenges of Existing Systems:

Existing systems designed for vehicle assistance and user safety face several challenges that hinder their effectiveness, particularly during critical situations where timely help is required. One of the most significant issues is the lack of real-time communication and location tracking capabilities. In emergencies, such as vehicle

breakdowns or accidents, the ability to instantly share one's location with roadside assistance or emergency services can be crucial. However, many current solutions either do not offer this functionality or execute it poorly, leading to delays in response times. Such delays can leave users stranded, increasing stress and vulnerability. Another critical challenge is the insufficient integration of vehicle assistance apps with comprehensive databases of local services, such as mechanics, gas stations, or utility shops. For an assistance app to be effective, it needs to provide users with quick access to a range of nearby resources. Unfortunately, many existing systems lack this level of integration, leaving users to manually search for nearby services, which can be time-consuming and frustrating in an emergency. This lack of connectivity reduces the app's effectiveness and lowers users' trust in relying on these systems during critical times. The usability of existing vehicle assistance apps and services also presents a major barrier. Many of these solutions have complex or cluttered interfaces, requiring users to navigate through multiple screens or provide unnecessary information before receiving assistance. This can be especially problematic during high-stress moments, such as when users are stranded on a highway or facing an urgent situation. An app that requires detailed inputs or multiple steps to activate an alert may not be practical when immediate action is needed. The ability to quickly access essential features like roadside assistance, GPS tracking, or mechanic services is critical for the app's success, and failure to do so undermines its purpose. Privacy concerns are another issue, particularly related to location tracking and data sharing. Users may be reluctant to share their real-time location due to concerns about unauthorized access or misuse of personal data. The collection and storage of sensitive information, such as GPS data and vehicle details, raises questions about data security and user privacy. Without strong privacy protections and transparent policies, users may be hesitant to trust the system, despite its potential to provide critical assistance in emergencies. Cultural and societal factors also impact the adoption and effectiveness of these tools. In many communities, there may be a lack of awareness or understanding of the benefits of vehicle assistance apps, or users may feel embarrassed to use them in the event of a breakdown. Additionally, societal attitudes toward technology and dependence on traditional forms of help can further hinder adoption. Some users might prefer calling roadside assistance services directly rather than using an app, especially if they are unfamiliar with the technology or have concerns about its reliability. Addressing these challenges requires a user-centric approach that combines advanced technology with a focus on ease of use and trust-building. The "**Swift Assist**" project aims to address these issues by creating an intuitive, reliable, and secure app that integrates real-time location tracking, immediate access to nearby services, and seamless communication with roadside assistance providers. Additionally, ensuring robust data security measures and clear privacy policies will encourage users to trust and adopt the app. Furthermore, the app will feature easy-to-navigate interfaces and quick-response features to ensure that help is never more than a few taps away. To build user confidence, "**Swift Assist**" will also integrate a feedback system, allowing users to rate services and share their experiences, ensuring continuous improvement and better service delivery. By tackling these challenges, "**Swift Assist**" aims to provide a more reliable, accessible, and trusted solution for vehicle breakdowns and emergency assistance. This holistic approach will empower users to feel secure and confident in their ability to handle vehicle-related emergencies and access necessary services quickly and efficiently.

2.5. Conclusion:

In conclusion, the development of the "**Swift Assist**" app marks a significant step forward in addressing the challenges faced by individuals during vehicle breakdowns and emergencies. The app's design and functionality are driven by a comprehensive understanding of the everyday issues related to vehicle safety and roadside assistance, whether during travel, in remote areas, or in unexpected breakdown situations. By utilizing

mobile technology, “**Swift Assist**” aims to provide users with an immediate and reliable means of securing help when they need it most, ensuring that they are never alone in moments of distress. One of the core features of “**Swift Assist**” is its real-time communication and live GPS tracking capabilities. These features are essential during emergencies, where rapid sharing of accurate location data can make all the difference in reducing response times and preventing further complications. By enabling users to instantly share their exact location with roadside assistance providers, emergency services, or nearby service providers (like mechanics or gas stations), the app ensures that help arrives promptly, enhancing overall safety during breakdowns or accidents. This provides users with peace of mind, knowing they can rely on “**Swift Assist**” to help them in urgent situations. Additionally, the app's integration with nearby service providers—such as mechanics, gas stations, and utility shops—simplifies the process of finding assistance during a crisis. By eliminating the need to manually search for nearby services, “**Swift Assist**” ensures that users can quickly access the help they need without unnecessary delays, especially in high-stress environments. This integration allows users to avoid the frustration of navigating through complex systems or being left stranded without immediate options. As technology continues to advance, there are tremendous opportunities for refining and expanding the capabilities of “**Swift Assist**”. Future versions of the app could incorporate predictive analytics to anticipate breakdowns or traffic issues, offering proactive safety alerts to users. Integration of AI-driven features could provide personalized recommendations for nearby services based on user preferences or previous experiences. With continuous improvements in GPS accuracy, communication networks, and data security, “**Swift Assist**” will remain on the cutting edge, ensuring that it continues to provide users with optimal functionality and reliability. The usercentric design of “**Swift Assist**” ensures that the app remains simple and intuitive for individuals from all walks of life. By gathering continuous feedback from users, particularly drivers and vehicle owners, the app can evolve to meet the changing needs of its user base, ensuring accessibility and ease of use in every situation. Beyond the immediate safety and convenience benefits, “**Swift Assist**” also has the potential to contribute to broader social goals, such as improving road safety and promoting equal access to services. By empowering users to take control of their safety and access emergency services, the app fosters a sense of confidence, particularly for individuals in vulnerable situations. It provides a practical solution for everyday road safety concerns while helping to reduce the stress and anxiety that can accompany breakdowns and emergencies. As “**Swift Assist**” continues to gain adoption, its collective impact on road safety and service accessibility could be substantial. A safer environment for drivers improves not only their personal well-being but also contributes to a more inclusive and equitable approach to transportation and emergency assistance. Ultimately, “**Swift Assist**” is not just a tool for vehicle breakdowns; it serves as a catalyst for broader positive changes in road safety and accessibility, contributing to a more secure and wellconnected society. As the app evolves, it will play an increasingly important role in improving the safety and well-being of individuals on the road, ensuring a more just and reliable system for anyone in need of assistance. In this way, “**Swift Assist**” promises to pave the way for a safer, more efficient, and equitable future for all drivers and vehicle owners.

Chapter 3

Project Planning and Scheduling

3.1 Introduction:

Project Planning and Scheduling for the “**Swift Assist**” app is a critical and organized process essential for guiding the app from its initial idea to its continuous updates and maintenance. The planning process begins with defining clear project goals that address the core needs of individuals facing vehicle breakdowns and roadside emergencies, with a focus on delivering swift, accessible, and reliable assistance. Key components of the planning process involve identifying and engaging relevant stakeholders—drivers, service providers, emergency contacts, developers, and maintenance teams—to ensure that the app caters to diverse user needs. This includes outlining the scope of core features such as real-time location tracking, integration with emergency services, access to nearby service providers (e.g., mechanics, gas stations), and user feedback collection.

Scheduling plays a crucial role by establishing achievable timelines, allocating resources efficiently, and setting clear milestones to track the project’s progress and ensure it stays on schedule. The development follows an iterative approach, allowing for continuous refinement based on user feedback and real-world testing. Additionally, risk assessment and mitigation strategies are crucial to anticipate and address potential challenges, including technology integration, user interface usability, data privacy, and service provider coordination. By managing these components effectively, the project aims to deliver a reliable tool that empowers users to quickly secure assistance during vehicle breakdowns, contributing to their peace of mind and promoting road safety.

The scheduling process involves establishing realistic deadlines, managing resource allocation, and setting milestones to ensure timely delivery and efficient implementation. This process incorporates multiple development cycles to gather feedback and refine features based on user testing and the evolving needs of drivers. As the project progresses, risk management strategies will be integral to addressing challenges such as potential issues with GPS accuracy, integration with service providers, and ensuring the app’s functionality under varying network conditions.

Ultimately, effective Project Planning and Scheduling for “**Swift Assist**” will play a pivotal role in realizing its mission to provide a fast, efficient, and user-friendly solution for vehicle breakdowns, ensuring accessibility and convenience for all users. Through careful management of resources, timelines, and continuous evaluation, “**Swift Assist**” will become a vital tool for drivers, promoting road safety and offering a dependable emergency response system that serves to improve the overall driving experience.

3.2 Recognition of Need:

The recognition of the need for project planning and scheduling for **Swift Assist** arises from several key factors. First, effective planning ensures that every stage of the app’s development—from ideation to deployment and ongoing updates—is methodically organized and executed. This includes setting clear objectives, defining realistic timelines, and efficiently allocating resources, all of which are vital for meeting deadlines and staying within budget constraints. Secondly, scheduling allows for the prioritization of critical tasks, such as integrating real-time GPS tracking, coordinating with service providers (e.g., mechanics, gas stations), and developing the

user interface for seamless operation in emergency situations. By establishing milestones and checkpoints, project managers can monitor progress, identify potential bottlenecks early, and implement corrective measures when necessary. Additionally, detailed planning and scheduling promote collaboration among the diverse teams involved in the development of **Swift Assist**, such as software developers, user experience designers, and service partners. This collaboration ensures that the app meets technical specifications while addressing the specific needs of drivers in emergency situations. Moreover, it helps align all aspects of the project with the goal of delivering a user-friendly, responsive tool that can be relied upon during critical breakdown situations.

Through structured planning and scheduling, the **Swift Assist** project can effectively manage the complexities of integration with external services, real-time tracking accuracy, and app usability in emergency scenarios. It will also mitigate potential risks, ensuring that the app remains stable, secure, and operational at all times. Ultimately, the need for thorough planning and scheduling ensures that **Swift Assist** can be successfully developed and delivered on time, providing a dependable solution for drivers and improving road safety in a variety of emergency situations.

3.3 Information Gathering:

Building the Swift Assist App requires a thorough exploration of user expectations, challenges faced during vehicle breakdowns, and the capabilities of current technologies. This process involves understanding user preferences, studying comparable applications, and identifying essential features such as real-time location tracking, Gas stations and utility Shops, immediate access to mechanics, and emergency response services. Comprehensive and precise data collection is vital to ensure the app addresses user needs effectively and delivers a seamless experience.

3.3.1 Information Gathering Tools:

Developing the **Swift Assist App** involves leveraging various tools to gather valuable insights. Surveys and questionnaires help capture user requirements, while stakeholder interviews provide a deeper understanding of specific challenges. Focus groups offer detailed perspectives on user expectations, and market research examines existing solutions to identify strengths and weaknesses. Competitor analysis highlights opportunities for differentiation, and data analytics tools evaluate user behavior and preferences to refine features and ensure a user-centric approach.

3.3.1.1: Surveys and Interviews:

Engaging directly with potential users through surveys and interviews is essential to understanding their specific challenges during vehicle breakdowns, desired app features, and usability preferences. These interactions provide critical insights into the real needs of users, such as quick access to assistance, reliable navigation tools, and seamless functionality. By identifying common issues drivers face, developers can tailor the app's design to address these pain points effectively. The feedback collected also plays a key role in shaping the app's interface, features, and overall user experience, ensuring the Swift Assist App delivers practical and user-friendly solutions.

3.3.1.2 Focus Group:

Conducting focus groups involves bringing together drivers from different backgrounds to share their experiences with existing vehicle assistance services. Participants discuss what they find effective, what falls

short, and their expectations for an ideal solution. These sessions foster open conversations, enabling individuals to express their challenges, preferences, and desired features for the app. By carefully analyzing this feedback, developers can customize the **Swift Assist App** to cater to the diverse needs of its users, ensuring it provides practical and reliable solutions to real-world breakdown scenarios.

3.3.1.3 Case Study:

Reviewing case studies of successful vehicle assistance apps or related platforms provides valuable insights into practical and effective solutions. By analyzing these examples, developers can pinpoint features that enhance user experience, such as fast response times, seamless navigation, or user-friendly interfaces. Examining engagement strategies sheds light on how to maintain user interest and ensure continued app usage. Additionally, proven technological innovations from these case studies can be tailored and incorporated into the **Swift Assist App**, creating a comprehensive and dependable solution that aligns with the needs of its users.

3.3.1.4 Expert Consultation:

Engaging professionals from relevant fields, such as automotive services, roadside assistance, user experience design, and behavioral psychology, is essential for crafting an effective vehicle breakdown management app. Automotive and roadside assistance experts can provide insights into common issues faced by drivers and recommend features that address real-world challenges. Behavioral psychologists can guide the design of the app to ensure it remains intuitive and stress-free during high-pressure situations. Specialists in user experience and human-computer interaction can refine the app's interface, making it accessible and user-friendly for individuals with varying levels of tech proficiency. This multidisciplinary collaboration ensures that the app is practical, dependable, and designed with users' needs in mind.

3.3.2 Relevant Stakeholder:

- i. Vehicle Owners and Drivers:** The primary users of "Swift Assist," vehicle owners and drivers benefit from its breakdown management features. They rely on the app for instant access to mechanics, towing services, Utility Shops and nearby gas stations, ensuring safety and convenience during unexpected vehicle issues. The app helps drivers feel more secure, especially when traveling alone or in unfamiliar areas. Their feedback is crucial for refining the app's usability and effectiveness.
- ii. Emergency Contacts:** Designated friends, family members, or colleagues receive notifications when a user reports a breakdown. Real-time updates, including the user's location and issue details, allow emergency contacts to provide immediate assistance, coordinate towing services, or ensure the user's safety. Their involvement adds an extra layer of support during critical situations.
- iii. Mechanics and Towing Service Providers:** As key service providers, mechanics and towing professionals are connected directly with users through the app. "Swift Assist" streamlines their operations by matching them with nearby breakdown cases, reducing response times and increasing efficiency. Their timely support is integral to resolving user issues effectively.
- iv. Fuel Stations and Utility Shop Owners:** Local gas stations and utility shops listed in the app serve as essential resources for stranded drivers. By providing accurate information on location, operating hours, and available services, these businesses contribute to a reliable support network. Collaboration with "Swift Assist" enhances their visibility and customer reach.

- v. **App Developers and IT Specialists:** Developers and IT experts ensure "Swift Assist" runs smoothly, with user-friendly features and reliable performance. They manage app maintenance, resolve bugs, and implement enhancements like live GPS tracking and secure payment integration. Their work ensures the app meets the evolving needs of its users.
- vi. **Government and Policy Regulators:** Government agencies and regulators oversee the app's compliance with data privacy and safety standards. By setting guidelines for its functionality and security, they ensure "Swift Assist" operates in a manner that prioritizes user welfare. They may also promote such tools as part of broader public safety initiatives.
- vii. **Insurance Providers:** Insurance companies benefit from collaborating with "Swift Assist" by offering users tailored roadside assistance packages. The app's integration with insurance services simplifies claims processes and enhances customer experience.
- viii. **Fleet Management Companies:** Organizations managing vehicle fleets can use "Swift Assist" to monitor and respond to breakdowns in real-time. By integrating the app into their operations, fleet managers can optimize vehicle uptime and minimize disruptions caused by mechanical issues.
- ix. **Investors and Sponsors:** Investors and sponsors provide financial backing for the development and expansion of "Swift Assist." Their contributions support the addition of new features, scaling to new markets, and improving user engagement. By investing in the app, they help ensure its long-term success and impact.
- x. **Data Privacy and Security Experts:** These professionals ensure that "Swift Assist" protects user information and complies with privacy laws. They implement robust encryption and monitor the app's systems to prevent unauthorized data access, safeguarding user trust and adherence to legal standards.
- xi. **Public Transport and Highway Authorities:** These stakeholders support "Swift Assist" by providing updates on road conditions, construction work, or traffic disruptions that might affect users. Their input ensures accurate and up-to-date information is available, enhancing the app's reliability. This version emphasizes the vehicle breakdown management focus of "Swift Assist" while addressing the role of each stakeholder in its ecosystem.
- xii. **Non-Governmental Organizations:** NGOs advocating for road safety and driver welfare collaborate with "Swift Assist" to promote its adoption. They help raise awareness about the app's benefits and provide input to developers for improving features to cater to diverse user groups.

3.4 Project Scheduling

Scheduling Gantt Chart for Planning:

Sl no.	Module Name	Sub module	January				February				March			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
1	Create a team													
2	Idea sharing													
3	Select project idea													
4	Project proposal documentation	Introduction to literature review												
		Existing system methodology												

		Requirement to conclusion															
5	Project Planning																
6	Requirement gathering																
7	Existing system analysis																
8	Market analysis																

Scheduling Gantt Chart for Feasibility or Requirements:

Sl no.	Module Name	Sub module	April				May				June				July				August			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
27	UI UX design	Login interface																				
		Sign up interface																				
		OTP interface																				
		User dashboard																				
		User services list																				
	Service Request Management	User profile																				
		service request interface																				
		Service provider finding																				
		Service request details																				
		Location tracking																				
		Nearest gas station finding																				
		Nearest public washroom finding																				
		Nearest mechanic shop finding																				
		utility shop finding																				
		Mechanic dashboard																				
	Mechanic Services	Mechanic profile																				
		Mechanic services																				
		Service request accepting																				
		User Location tracking																				
		Admin dashboard																				
	User and Mechanic Status Checking	User and mechanic status checking																				

	Mechanic request approval																								
	Payment method selection																								
	Payment interface																								
	Payment verification																								
	History Checking																								
	Feedback																								

Scheduling Gantt Chart for Prototyping:

Sl n o.	Modul e Name	Sub module	January				February				March				April				May				June			
			W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4
9	Feasibil ity study	Technical																								
		Operational																								
		Financial																								
		Legal and ethical																								
10	Risk analysis																									
11	Environmental Impact Assessment																									
12	Requirement documents																									
13	Communication to stakeholders																									
14	Budgeting																									
15	Cost benefit analysis																									

Table no. 3 : Table for Feasibility or Requirements

	selection																								
17	Software model selection																								
19	SWOT Analysis																								
20	Information gathering	Survey																							
		Interview																							
		Questionnaires																							
21	System design																								
22	Mapping information																								
23	Data flow diagram																								
24	Class diagram																								
25	Use case diagram																								
26	ER diagram																								

Table no. 4 : Table for Prototyping

Scheduling Gantt Chart for Software Development:

Sl no .	Module Name	Sub module	June				July				August				September				October				
			W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	W 1	W 2	W 3	W 4	
28	Project implementation (coding)	Login page																					
		Sign up page																					
		OTP getting design																					
		User dashboard																					
		User dashboard service																					
		User profile																					
		User service																					
		Service provider finding																					
		Service request details																					
		Location tracking																					
		Nearest gas station																					
		Nearest public washroom																					
		Nearest mechanic shop																					
		Nearest utility shop																					
		Mechanic dashboard																					
		Mechanic profile																					
		Mechanic service																					
		Service request																					
		User Location																					
		Admin dashboard																					
		User and mechanic status checking design																					
		Mechanic request approval																					
		Active service provider																					
		Payment method																					
		Payment design																					
		Payment verification																					
		Feedback design																					
29	Google API purchase																						
30	API management																						
31	Database creation																						
32	Data Entry to Database																						

Table no. 5 : Table for development

Scheduling Gantt Chart for Operation & Maintenance:

Sl no.	Module Name	Sub module	October				November				December			
			W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
44	Marketing and Advertising													
45	Market Expansion Planning													
46	User manual creation													
47	Training													
48	Third-party verification													

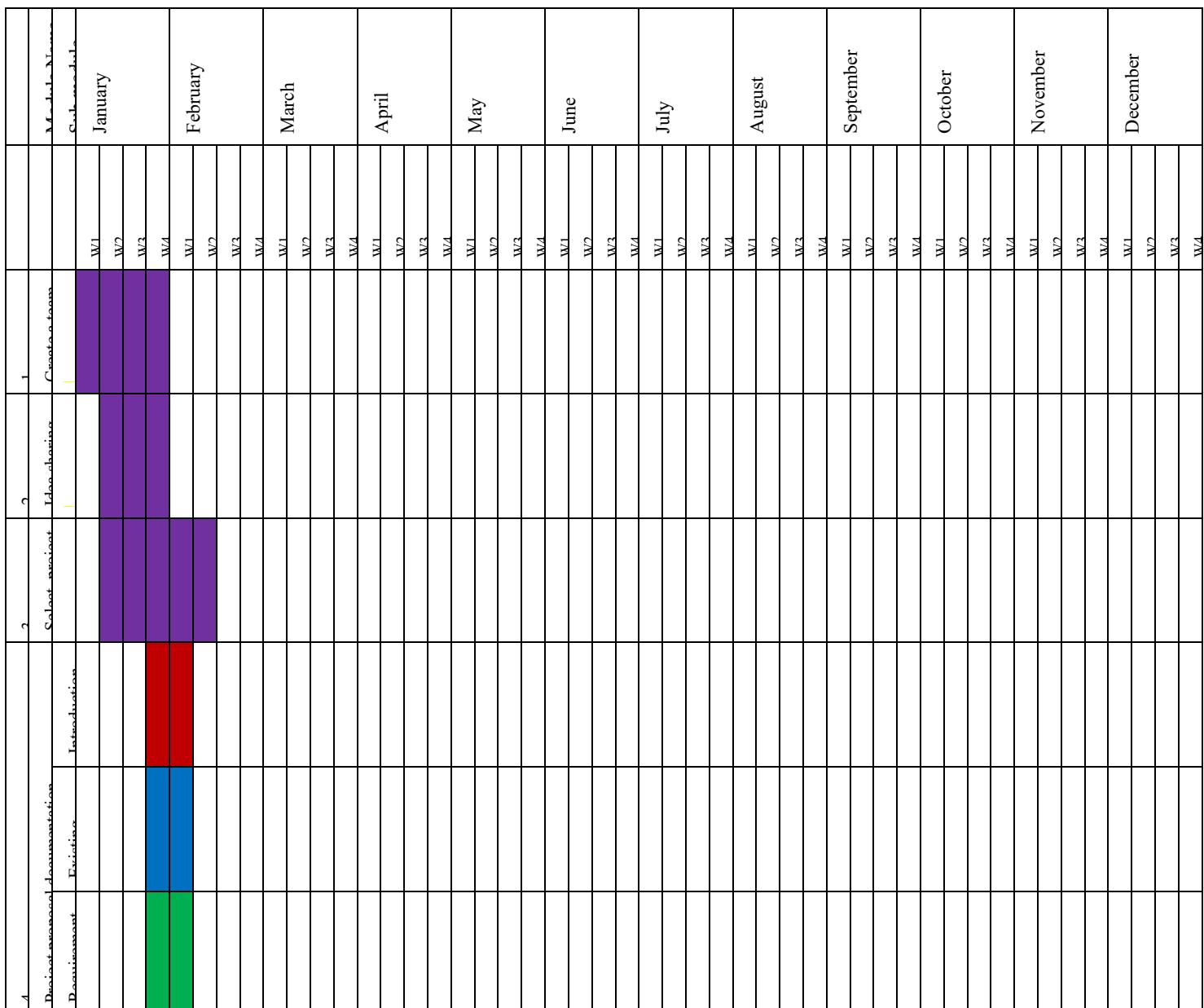
Table no. 6 : Table for Operation & Maintenance

Scheduling Gantt Chart for Deployment:

Sl no.	Module Name	Sub module	November				December				
			W1	W2	W3	W4	W1	W2	W3	W4	
49	Final documentation										
50	Stakeholder reporting										
51	Project delivery										

Table no. 7: Table for Deployment

Time Scheduling Gantt Chart for This Project (Yearly)



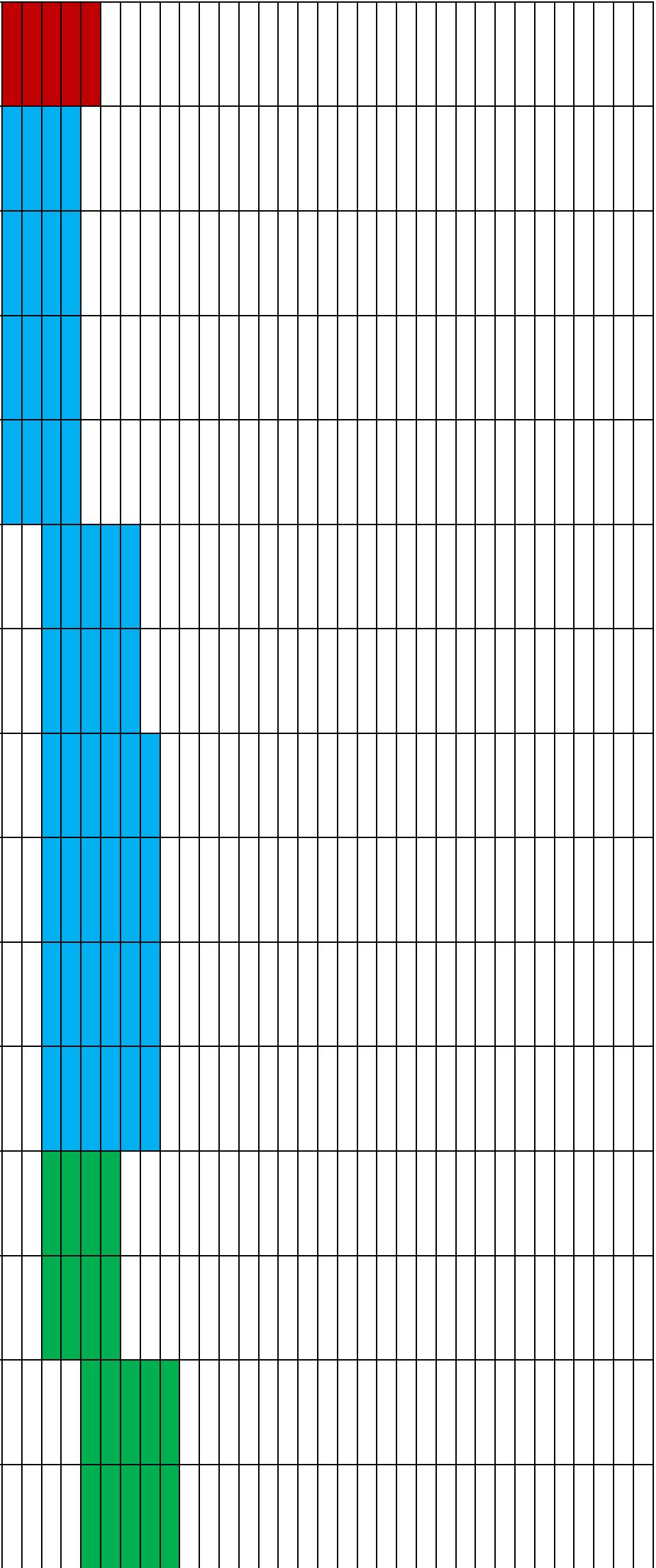
The Gantt chart illustrates the project timeline and task distribution for Project Alpha. The project begins at week 1 and concludes at week 16.

Legend:

- Project Planning: Purple
- Requirement gathering: Green
- Design & Architecture: Blue
- Development: Red
- Testing & Deployment: Grey

Task Details:

- Week 1:** Project Finalization (purple)
- Week 2:** Requirements Gathering (green)
- Week 3:** Requirements Gathering (green)
- Week 4:** Design & Architecture (blue)
- Week 5:** Design & Architecture (blue)
- Week 6:** Design & Architecture (blue)
- Week 7:** Design & Architecture (blue)
- Week 8:** Development (red)
- Week 9:** Development (red)
- Week 10:** Development (red)
- Week 11:** Development (red)
- Week 12:** Development (red)
- Week 13:** Development (red)
- Week 14:** Development (red)
- Week 15:** Development (red)
- Week 16:** Testing & Deployment (grey)



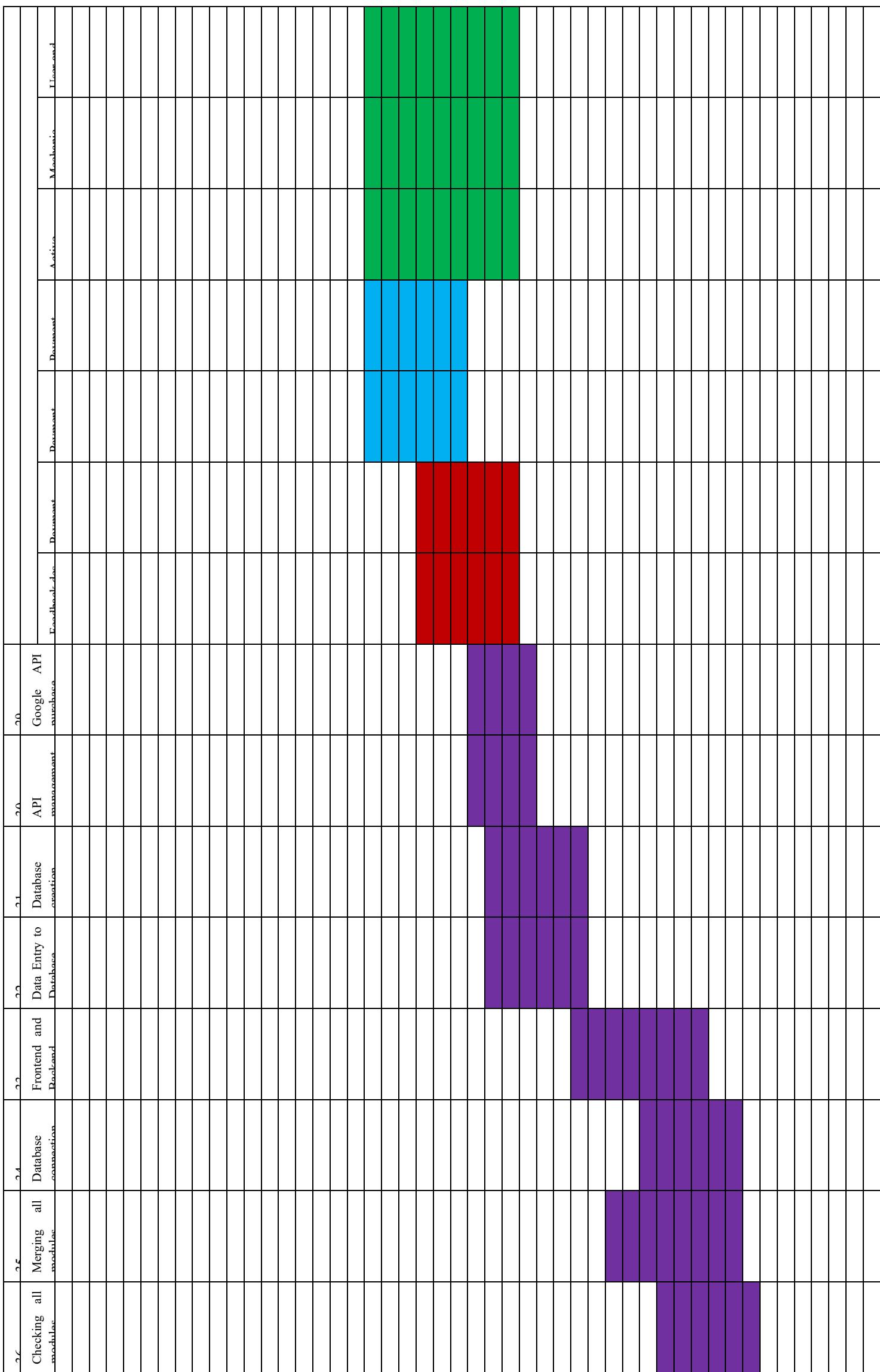


Table no. 8 : Table for Whole year working

Month Wise Time Scheduling for Swift Assist App

Sl no.	Module	Sub module	January			
			W1	W2	W3	W4
1	Create a team					
2	Idea sharing					
3	Select project					
4	Project proposal documentation	Introduction to literature review				
		Existing system methodology				
		Requirement to conclusion				
5	Project Planning					
6	Requirement gathering					

Table no. 9: Table for January

Sl no.	Module Name	Sub module	February			
			W1	W2	W3	W4
1. 1.	Select project idea					
2.	Project proposal documentation	Introduction to literature review	Red			
		Existing system methodology	Blue			
		Requirement to conclusion	Green			
3.	Project Planning					
4.	Requirement gathering					
5.	Existing system analysis					
6.	Market analysis					
	Feasibility study	Technical feasibility				
		Operation of feasibility				

Table no. 10: Table for February

Sl no.	Module Name	Sub module	March			
			W1	W2	W3	W4
1.	Project Planning					
2.	Requirement gathering					
3.	Existing system analysis					
4.	Market analysis					
5.	Feasibility study	Technical feasibility				
		Operation of feasibility				
		Financial feasibility				
		Legal and ethical feasibility				
6.	Risk analysis					Red
7.	Environmental Impact Assessment				Purple	Purple
8.	Requirement documents					Purple
9.	Communication to stakeholders					Purple
10.	Budgeting					Green

Table no. 11: Table for March

Sl no.	Module Name	Sub module	April			
			W1	W2	W3	W4
1.	Risk analysis					
2.	Environmental Impact Assessment					
3.	Requirement documents					
4.	Communication to stakeholders					
5.	Budgeting					
6.	Cost benefit analysis					
7.	Project finalization					
8.	Business model selection					
9.	Software model selection					
10.	SWOT Analysis					
11.	Information gathering	Survey				
		Interview				
		Questionnaires				
12.	System design					
13.	Mapping information					
14.	Data flow diagram					
15.	Class diagram					
16.	Use case diagram					
17.	ER diagram					
18.	UI UX design	Login interface				
		Sign up interface				
		OTP interface				
		User dashboard				
		User services list				
		User profile				
		service request interface				
		Service provider finding				

Table no. 12: Table for April

Sl no.	Module Name	Sub module	May			
			W1	W2	W3	W4
1	Requirement documents					
2						
3	Communication to stakeholders					
4	Budgeting					
5	Cost benefit analysis					
6	Project finalization					
7	Business model selection					
8	Software model selection					
9	SWOT Analysis					
1	Information gathering	Survey				
		Interview				
		Questionnaire				
1	System design					
1	Mapping information					
1	Data flow diagram					
1	Class diagram					
1	Use case diagram					
1	ER diagram					
	UI UX design and	Login interface				
		Sign up interface				
		OTP interface				
		User dashboard				
		User services list				
	User profile	User profile				
		service request interface				
		Service provider finding				
		Service request details				
		Location tracking				
		Nearest gas station finding				
		Nearest public washroom finding				
		Nearest mechanic shop finding				
		utility shop finding				
		Mechanic dashboard				
		Mechanic profile				
		Mechanic services				
		Service request accepting				

Table no. 13: Table for May

Sl no.	Module Name	Sub module	June			
			W1	W2	W3	W4
2	System design					
3	Mapping information					
4		OTP interface				
		Service request details				
		Location tracking				
		Nearest gas station finding				
		Nearest public washroom finding				
		Nearest mechanic shop finding				
		utility shop finding				
		Mechanic dashboard				
		Mechanic profile				
		Mechanic services				
		Service request accepting				
		User Location tracking				
		Admin dashboard				
		User and mechanic status checking				
5	Project implementation (coding)	Mechanic request approval				
		Payment method selection				
		Payment interface				
		Payment verification				
		History Checking				
		Feedback				
		Login page design				
		Sign up page design				
		OTP getting design				
		User dashboard				
		User dashboard service list				
		User profile				
		User service request				
		Service provider finding design				

	Nearest public washroom finding design				
	Nearest mechanic shop finding				
	Nearest utility shop finding				
	Mechanic dashboard design				
	Mechanic profile design				
	Mechanic services design				
	Service request accepting design				
	User Location tracking design				
	Admin dashboard				
	User and mechanic status checking design				
	Mechanic request approval design				
	Active service provided design				
	Payment method selection				
	Payment design				
	Payment verification				
	Feedback design				

Table no. 14: Table for June

Sl no.	Module Name	Sub module	July			
			W1	W2	W3	W4
6		User Location tracking				
		History Checking				
		Feedback				
		User dashboard service list				
		User profile				
		User service request				
		Service provider finding design				
		Service request details providing				
		Service selection details design				
		Location tracking				
		Nearest gas station finding design				
		Nearest public washroom finding design				
		Nearest mechanic shop finding				
		Nearest utility shop finding				
		User Location tracking design				
		Admin dashboard				
		User and mechanic status checking design				
		Mechanic request approval design				
		Active service provided design				
		Payment verification				

		Feedback design				
7	Google API purchase					
8	API management					
9	Database creation					
1	Data Entry to Database					

Table no. 15: Table for July

Sl no.	Module Name	Sub module	August			
			W1	W2	W3	W4
1.	Database creation					
2.	Data Entry to Database					
3.	Frontend and Backend creation connection					

Table no. 16: Table for August

Sl no.	Module Name	Sub module	September			
			W1	W2	W3	W4
1.	Frontend and Backend creation connection					
1	Database connection					
1	Merging all modules					
1	Checking all modules					
1	Registration of nearby gas station					
1	Registration of public washroom					
1	Registration of mechanic shop					
1	Registration of utility shop					
1	App testing	Verification				
		Validation				
		Acceptance testing				
		Functional tasting				
		Nonfunctional testing				
		System testing				

Table no. 17: Table for September

Sl no.	Module Name	Sub module	October			
			W1	W2	W3	W4
1.	Frontend and Backend creation connection					
1	Database connection					
2	Merging all modules					
2	Checking all modules					
2	Performance Monitoring					
2	Registration of nearby gas station		Green	Green	Green	
2	Registration of public washroom		Red	Red	Red	
2	Registration of mechanic shop		Green	Green	Green	
2	Registration of utility shop		Green	Green	Green	
2	App testing	Verification	Red	Red	Red	Red
		Validation	Red	Red	Red	Red
		Acceptance testing	Red	Red	Red	Red
		Functional tasting	Red	Red	Red	Red
		Nonfunctional testing	Red	Red	Red	Red
		System testing	Red	Red	Red	Red
2	Bug fixing					Blue
2	Marketing and Advertising					Blue
3	Market Expansion Planning					Blue
3	User manual creation					Blue
3	Training					Blue

Table no. 18: Table for October

Sl no.	Module Name	Sub module	November			
			W1	W2	W3	W4
1.	Checking all modules					
3	Performance Monitoring					
3	App testing	Verification	Red	Red		
		Validation	Red	Red		
		Acceptance testing	Red	Red		
		Functional tasting	Red	Red		
		Nonfunctional testing	Red	Red		
		System testing	Red	Red		
3	Bug fixing		Blue	Blue	Blue	Blue
3	Marketing and Advertising		Blue	Blue	Blue	

3	Market Expansion Planning					
3	User manual creation					
3	Training					
4	Third-party verification					
4	Final documentation					

Table no. 19: Table for November

Sl no.	Module Name	Sub module	December			
			W1	W2	W3	W4
1.	Bug fixing					
42.	Training					
43.	Third-party verification					
44.	Final documentation					
45.	Stakeholder reporting					
46.	Project delivery					

Table no. 20: Table for December

3.5 Conclusion:

In conclusion, effective project planning and scheduling are critical for the successful development of the "Swift Assist" app. This process starts with defining clear milestones and objectives, which provide a roadmap to guide each phase of development. These benchmarks help the project team manage progress effectively and ensure that every stage aligns with the app's overarching goals. Strategic allocation of resources, coupled with adherence to defined timelines, is essential for addressing the urgent needs of drivers experiencing vehicle breakdowns. This approach guarantees that key features—such as real-time GPS tracking, instant access to mechanics, towing services, and nearby fuel or utility shop locators—are seamlessly incorporated into the app, optimizing its functionality and user experience. Regular performance reviews and iterative improvements are vital components of this process.

Chapter 4

System Analysis

4.1 Introduction:

System analysis for the "Swift Assist" app is a vital stage that involves a detailed examination of user requirements, existing roadside assistance solutions, and technological capabilities. This thorough assessment aims to identify the functional and non-functional requirements necessary for developing an efficient vehicle breakdown management system. By understanding the needs of vehicle owners and evaluating the shortcomings of current solutions, we can design and integrate essential features such as real-time GPS tracking, instant access to mechanics, gas station, towing services, and nearby utility shop locators. The system analysis phase also involves identifying potential challenges that could impact the app's functionality. Key considerations include data security, ensuring that user information, such as location and payment details, is safeguarded against unauthorized access or breaches. Another critical factor is usability, requiring the app to have a straightforward and intuitive interface to encourage widespread adoption and seamless use during emergencies. Furthermore, integration with service providers, such as mechanics and towing companies, is crucial to ensure prompt and effective assistance. This foundational analysis ensures that the app is both user-centric and technically feasible. By addressing these aspects, the development process aims to produce a reliable and comprehensive tool that enhances roadside safety, minimizes inconvenience during breakdowns, and empowers users with quick, dependable solutions. Through meticulous system analysis, "Swift Assist" is positioned to deliver a significant improvement in vehicle breakdown management, providing drivers with peace of mind and contributing to safer and more efficient travel experiences.

4.2 Requirement Analysis:

The requirement analysis for the "Swift Assist" app necessitates a meticulous delineation and documentation of both functional and non-functional requisites integral to its development. Core functional imperatives encompass precise real-time GPS localization, enabling users to share their exact location with service providers; instantaneous connectivity to roadside assistance networks, including verified mechanics and towing services; and effortless access to proximal resources,

such as fuel stations, gas station and automotive supply shops, with detailed operational insights. Conversely, non-functional requisites emphasize the fortification of data integrity and user confidentiality, incorporating advanced encryption algorithms and secure communication protocols to shield sensitive information. The application's interface must exhibit intuitive navigability to ensure seamless utilization during exigent circumstances, while its design must support cross-platform compatibility across diverse mobile operating systems. Furthermore, multi-language support is imperative to accommodate a heterogeneous user demographic. The derivation of these requirements necessitates exhaustive stakeholder engagements, methodical surveys, and an incisive evaluation of extant roadside assistance solutions to identify existing deficiencies and avenues for enhancement. This exhaustive requirement analysis underscores the commitment to delivering a system that not only resolves vehicular breakdown scenarios with efficiency but also seamlessly integrates into the routines of users. By addressing these multifaceted dimensions, "Swift Assist" is poised to establish itself as a technologically advanced, user-centric solution that redefines the standards of roadside assistance.

4.2.1 Functional Requirement

- i. **Real-Time GPS Localization:** The app must provide accurate real-time location tracking to pinpoint the user's exact geographical position. This feature should allow users to share their location directly with service providers, ensuring swift and precise assistance. The system should integrate advanced mapping services to display the user's current location and nearby relevant services clearly. It must support route finding, enabling assistance teams (mechanics or tow trucks) to navigate efficiently to the user's location.
- ii. **Instantaneous Connectivity to Roadside Assistance Networks:** The app must include a network of verified roadside assistance providers, such as mechanics, tow truck operators, and emergency vehicle services. Users should have the ability to request help instantly, with the app automatically connecting them to the nearest and most suitable provider based on their needs. The system must include a service verification mechanism, ensuring only authenticated and reliable providers are listed. A feedback mechanism should be integrated, allowing users to rate and review service providers, fostering accountability and quality improvement.

iii. Effortless Access to Proximal Resources: The app must offer a comprehensive directory of nearby essential services, such as fuel stations, automotive utility supply shops, tire repair centers. Each resource listing should include detailed operational insights, such as:

- Location details: Exact address and distance from the user's position.
- Operating hours: Information on whether the facility is currently open or closed.
- Service details: A description of available services (types of fuel, available automotive parts, or specific repair capabilities).
- Contact information: Phone numbers or other direct communication options.

iv. User Authentication: Robust user authentication mechanisms are indispensable for the "Swift Assist" app, ensuring fortified protection of sensitive personal data and account credentials. The app must incorporate advanced identity verification protocols, including encrypted password-based login, biometric validation (such as fingerprint or facial recognition), or multi-factor authentication methods. These measures substantiate the legitimacy of user access, thereby thwarting unauthorized intrusions and mitigating the risk of data breaches. This multifaceted approach to authentication fortifies the app's security architecture, engendering user confidence in its reliability.

v. Privacy Controls: Privacy customization is a cornerstone of the "Swift Assist" app, enabling users to exercise granular control over their data-sharing preferences. The app must provide an array of configurable options to dictate the visibility of their real-time location, such as restricting access solely to verified roadside assistance providers during service requests or selectively sharing data with trusted contacts under specific conditions. These sophisticated privacy controls offer users a nuanced balance between safeguarding their personal information and availing themselves of essential services. By delivering an intuitive interface for managing these preferences, the app empowers users to make well-informed decisions regarding data dissemination.

vi. Additional Functional Features: The app must guide users to the selected service provider or resource using turn-by-turn navigation. Users should be able to filter service providers based on criteria such as proximity, user ratings, or service cost. The app must allow users to handle multiple requests, such as booking a tow truck while searching for a nearby gas station. A feature for quick signals in case of critical roadside emergencies, connecting users with priority assistance. These functional requirements collectively ensure that "Swift Assist"

provides a reliable, user-friendly, and comprehensive solution to vehicle breakdown scenarios.

Hardware and Software Requirements for Swift Assist :

Hardware Requirements

1. Client Devices (User and Service Providers)

- **Smartphones/Tablets:**
 - Processor: Quad-core or higher
 - RAM: 2GB or higher
 - Storage: 50MB (for app installation)
 - Operating System: Android 8.0 (Oreo) or higher
 - GPS module: Mandatory for real-time tracking
 - Internet connectivity: 4G/5G/Wi-Fi recommended

2. Server Infrastructure

- **Web Server:**
 - Processor: Quad-core or higher
 - RAM: 16GB or higher
 - Storage: SSD with 500GB or higher
 - Network: High-speed internet (100 Mbps or higher)
- **Database Server:**
 - Processor: Quad-core or higher
 - RAM: 32GB or higher
 - Storage: 1TB SSD for efficient data handling
- **Backup Server:**
 - Storage: 2TB HDD or higher for data backup

3. Other Hardware

- GPS-enabled tracking devices (optional for advanced service providers)
- Backup power supply or UPS for servers

Software Requirements

1. Client-side Software

- **Operating System:** Android (for app users and service providers)
- **Frameworks:** Android Studio for app development
- **APIs:** Google Maps API for location tracking
- **Browser Compatibility:** Latest versions of Chrome, Firefox, or Safari (for optional web interface)

2. Server-side Software

- **Operating System:** Windows
- **Database:** Firebase for data management
- **Backend Frameworks:** Node.js, Spring Boot, or Django
- **Cloud Services:** Firebase for authentication, push notifications, and real-time database

3. Development Tools

- **IDE:** Android Studio for mobile app development
- **Version Control:** GitHub or Git for collaboration and versioning
- **Design Tools:** Figma or Sketch for UI/UX design

4. Security Tools

- SSL Certificates for secure data transmission
- Firewall and Antivirus software for server protection

4.2.2 Non-Functional Requirements:

The non-functional requisites of the "Swift Assist" app prioritize the robust safeguarding of data integrity and user confidentiality, ensuring a secure, reliable, and trustworthy system. Here's a detailed breakdown:

- i. **Data Integrity:** The app must incorporate robust mechanisms to detect and prevent data corruption during transmission, storage, or processing. Input data from users, such as service requests and location details, must be validated to ensure accuracy and reliability before being processed or stored. The app should employ consistency algorithms to ensure that data remains uniform and unaltered across all components of the system.

- ii. **User Confidentiality:** The "Swift Assist" app ensures a secure and reliable environment for users. Security constitutes a paramount concern for the "Swift Assist App," necessitating the implementation of robust data encryption protocols and secure authentication mechanisms. These measures are indispensable for safeguarding user privacy by ensuring the confidentiality and integrity of personal data and communication channels.
- iii. **Usability:** The "Swift Assist App" must exhibit a high degree of usability, characterized by an intuitive and user-centric interface. The design should prioritize ease of use, enabling rapid and effortless activation of safety features, even in high-stress situations.
- iv. **Scalability:** Scalability represents a critical non-functional requirement for the "Swift Assist App," enabling it to effectively accommodate a growing user base and facilitate the expansion of its service offerings.
- v. **Regular Security Audits and Updates:** Conduct periodic security assessments, including vulnerability scans and penetration testing, to identify and address potential risks. Implement a proactive security posture through the regular execution of comprehensive security assessments, encompassing vulnerability scans and penetration testing.
- vi. **Resilience and Reliability:** In case of system failures, data must remain secure and consistent, with recovery processes in place to restore operations. Redundant storage and backup systems should be employed to prevent data loss and ensure availability. The app's infrastructure must support increasing traffic without compromising on data integrity or confidentiality.

4.2.3 Cost-Benefit Analysis:

Development Costs :

Software Development: ₦8,00,000

Data Acquisition and Training Costs :

Data Collection: ₦ 50,000

Data Preprocessing: ₦ 30,000

Operational Testing: ₦ 30,000

Infrastructure Costs :

Development and testing environment setup: ₦ 1,00,000 (utilizing existing resources)

Documentation Costs :

Documentation expenses: ₦ 40,000

Vehicle and transport Costs :

Vehicle: ₦ 10,000

Total Costs: ₦ 9,90,000

4.2.4 SWOT Analysis

The SWOT analysis of the **Swift Assist** application provides a comprehensive evaluation of its strengths, weaknesses, opportunities, and threats. **Strengths** include its user-friendly interface, real-time GPS tracking, efficient service coordination, and integration of essential services like towing, mechanic assistance, and feedback. These features ensure a seamless and reliable user experience, making the app a valuable tool for users facing vehicle breakdowns. **Weaknesses** lie in potential challenges such as dependency on internet connectivity, limited availability of service providers in remote areas, and concerns about data privacy and security. However, the app presents significant **opportunities** to expand its service offerings, partner with local mechanics and towing companies, and integrate advanced features like predictive maintenance alerts or offline support. Additionally, scaling the app to new regions or incorporating electric vehicle-specific services could broaden its market reach. **Threats** include competition from similar apps, technological advancements that could make existing features obsolete, and external factors like regulatory hurdles or operational delays. Addressing these factors

strategically can help ensure Swift Assist becomes a leading solution for vehicle breakdown assistance.

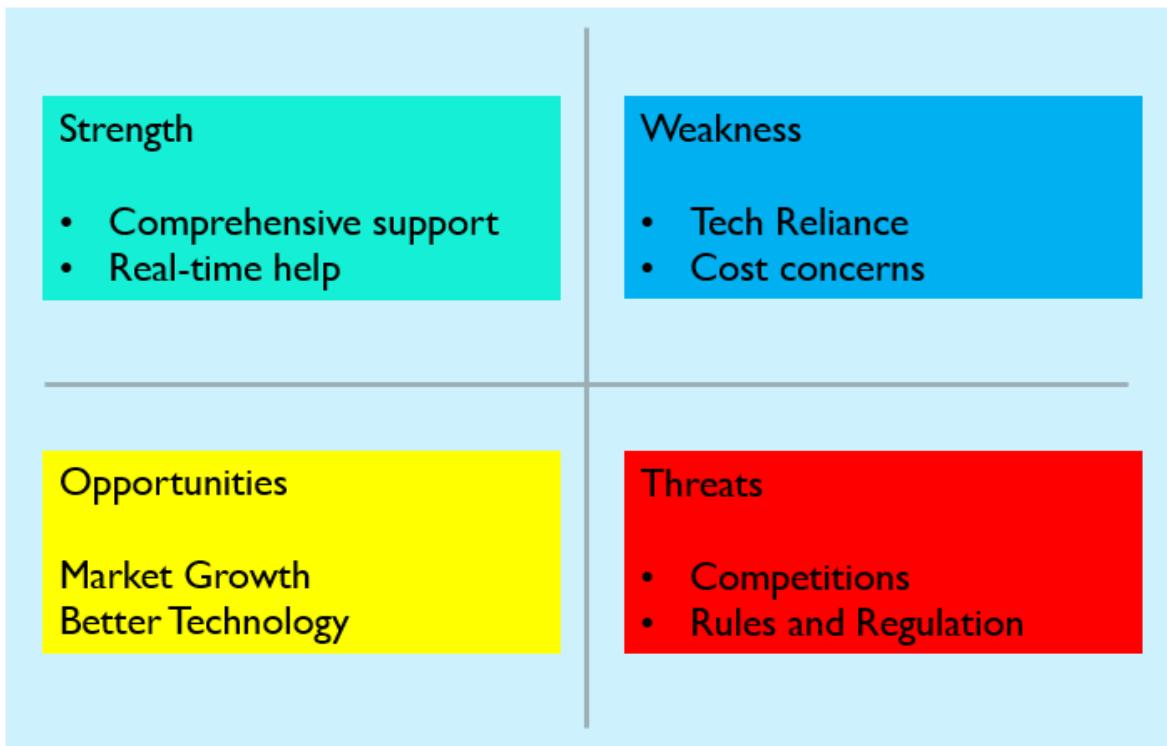


Fig No. 1 : SWOT Analysis

4.3 Conclusion

In summary, the system analysis for the "Swift Assist" app underscores the imperative for a sophisticated, dependable, and user-oriented mobile platform tailored to effectively mitigate vehicle breakdown challenges. This examination elucidates several pivotal non-functional requirements that are instrumental to the app's viability and widespread adoption. Critical considerations such as operational efficiency, system robustness, user adaptability, data fortification, and scalability are paramount to ensuring the application fulfills the nuanced demands of its user base. Core functionalities like GPS-based navigation and on-demand access to roadside assistance services must operate with exceptional precision and minimal latency to furnish users with timely support during unforeseen vehicular emergencies. The application must exhibit unwavering availability and a logically structured interface, designed to facilitate effortless usability even amidst high-stress scenarios, thereby fostering user reliance and promoting

Chapter 5

System Design

5.1 Introduction:

The system design phase is a critical milestone in the development of the "Swift Assist" app, focusing on crafting a robust and user-oriented framework tailored to address vehicle breakdown challenges. This stage converts insights from the system analysis into a structured blueprint, guiding the development process to ensure the app delivers dependable and efficient roadside assistance services. The design encompasses both the overarching architecture and detailed technical specifications, enabling smooth integration and functionality across all components. The core system architecture of the swift assis app integrates a user-friendly mobile interface, a secure and scalable backend system, and seamless connectivity with external resources like GPS, verified service providers, and real-time databases for local roadside assistance. The mobile interface is designed for simplicity and efficiency, allowing users to quickly access essential features such as live GPS tracking, instant roadside assistance requests, and a directory of nearby service providers, including fuel stations and towing services. The interface prioritizes ease of use to ensure that users can navigate it effectively, even in stressful situations. The backend infrastructure is engineered to manage high data volumes with minimal delays, facilitating real-time updates and secure data processing. It incorporates advanced security features, including encrypted data storage and robust authentication methods, to protect user information and prevent unauthorized access. This ensures both reliability and trustworthiness in the app's operations. By integrating these components, the system design provides a comprehensive solution for managing vehicle breakdowns, enabling users to handle roadside emergencies with confidence and convenience. Through meticulous design and technical precision, the swift assist app bridges the gap between modern technology and practical support, fostering a safer and more reliable driving experience for users.

5.2 System Design Tools and Techniques:

System design tools and techniques for the "Swift Assist" application include UML diagrams (Use Case, Class, Sequence) for modeling system architecture, Figma or Sketch for designing user interfaces, Google Maps API for real-time location tracking, Firebase for backend services, and Git for version control and collaboration among developers.

i. Unified Modelling Language (UML):

➤ Use Case Diagrams:

Use case diagrams are essential for capturing the functional requirements of the "Swift Assist" application. They visually represent the interactions between users (such as vehicle owners, tow truck operators, and mechanics) and the system, showcasing various scenarios in which the app will be used. These diagrams help developers understand user needs and system capabilities, ensuring all required features are included and effectively implemented.

➤ Class Diagrams:

Class diagrams are used to define the data model of the "Swift Assist" application by detailing the structure of the system. They illustrate the various classes, their attributes, and the relationships between them, providing a clear blueprint of the app's architecture. This representation aids developers in organizing the codebase, ensuring maintainability and seamless integration between components.

➤ Sequence Diagrams:

Sequence diagrams are used to visualize the flow of interactions between different objects or components in the "Swift Assist" app for specific functionalities. They detail the chronological sequence of messages exchanged, such as those involved in requesting roadside assistance or tracking a tow truck in real time. This step-by-step visualization helps developers accurately implement and optimize these processes.

ii. Wireframing and Prototyping Tools:

Figma or Sketch:

Figma or Sketch are essential tools for designing the user interface (UI) and user experience (UX) of the "Swift Assist" app. These platforms allow designers to create wireframes that outline the app's layout and structure and develop interactive prototypes to demonstrate its functionality and navigation flow. This helps stakeholders and developers refine the app's design and ensure it meets user expectations before implementation.

iii. Software Development Kits (SDKs) and APIs:

➤ Google Maps API:

The Google Maps API provides crucial services for the "Swift Assist" app, including real-time location tracking, route planning, and interactive mapping capabilities. This API enables developers to integrate accurate location data, allowing users to share their location with service providers and track assistance in real time. Features like proximity alerts and route visualization enhance the overall user experience, ensuring seamless navigation and safety during vehicle breakdowns.

➤ Firebase:

Firebase serves as the backbone for many essential backend functionalities in the "Swift Assist" app. Its real-time database ensures instant data synchronization, allowing users to receive up-to-date information about their service requests. Firebase Authentication provides secure login mechanisms, enhancing trust and protecting user data. Additionally, its push notification capabilities enable timely updates, such as service status alerts or reminders, ensuring effective communication with users.

iv. Version Control Systems:

5.2.1 Logical Design:

5.2.1.1 ER Diagram:

The Entity-Relationship (ER) Diagram for the "Swift Assist" application provides a visual representation of the database structure that supports the app's functionality. It outlines the key entities involved in the system, such as Users, Service Providers (Towing Services, Mechanics), Vehicles, Service Requests, and Location Data. Each entity is connected through relationships that define how data is stored, accessed, and interacted with within the system. For instance: Users are linked to their Vehicles and can initiate Service Requests. Service Providers are connected to Service Requests they handle, with real-time updates based on Location Data. Vehicles store important information such as registration details, type, and breakdown history.

The ER Diagram serves as a blueprint to illustrate how the application efficiently manages and links critical information. This structure ensures that users can report breakdowns, connect with nearby service providers, and receive assistance promptly, creating a seamless and reliable user experience. Given below is the diagram:

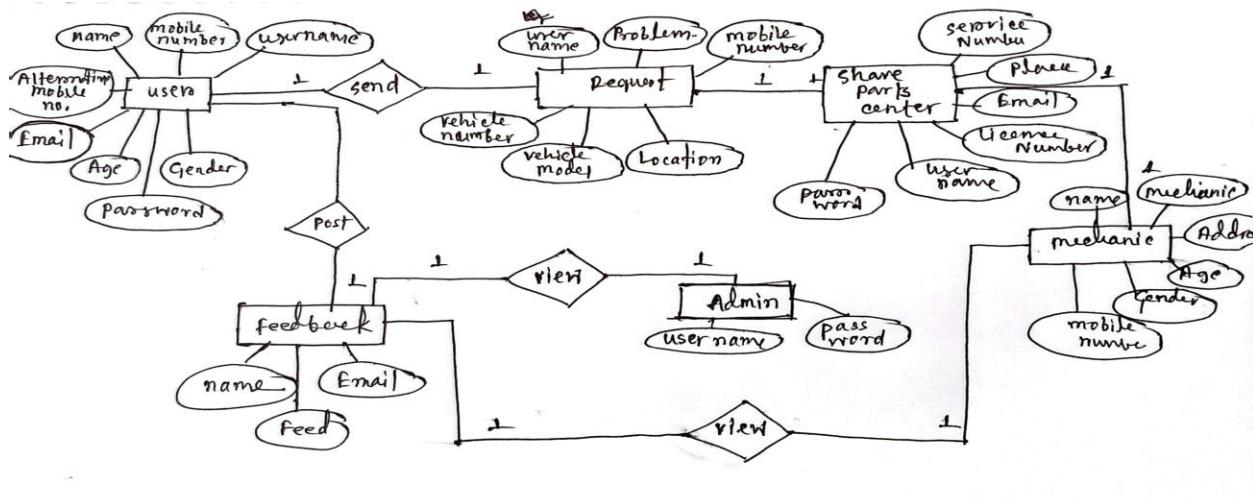


Fig No.2 : E-R Diagram

5.2.1.2 Class Diagram:

A class diagram is a vital component of object-oriented design, providing a visual blueprint of the structure of a software system. In the context of the "Swift Assist" application, the class diagram outlines the key classes that represent the application's core entities, such as User, Vehicle, Service Request, Service Provider, Location Data, and Feedback. Each class encapsulates attributes that define its properties and methods that dictate its behavior, reflecting the real-world relationships and interactions within the app. Given below is the class diagram for the Swift Assist application:

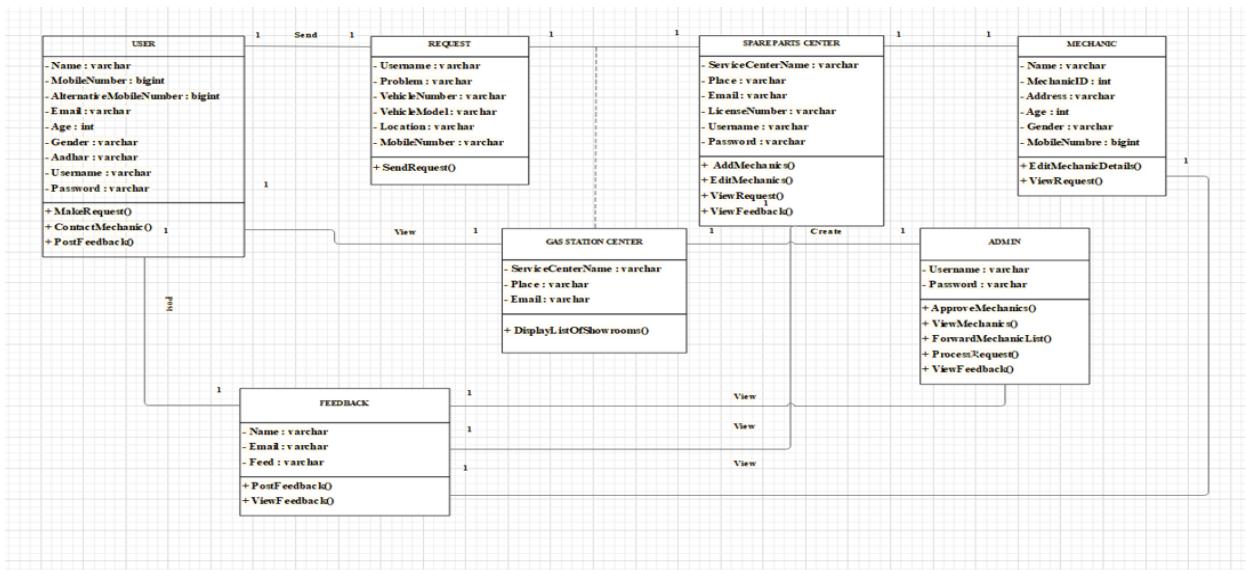


Fig No. 3: Class Diagram

5.2.1.3 The Data Flow Diagram (DFD):

The Data Flow Diagram (DFD) of the "Swift Assist" application visually represents how data moves through the system, illustrating the flow of information between processes, data stores, and external entities. It provides a clear overview of how the app manages vehicle breakdown requests, service coordination, and communication with service providers. The DFD captures high-level processes, such as the user submitting a service request, the system assigning the request to a service provider, and real-time tracking of the service provider's location. It also highlights how data is stored and updated, such as vehicle details and request status. The DFD helps analyze and optimize the system by identifying potential inefficiencies and ensuring smooth communication between all entities.

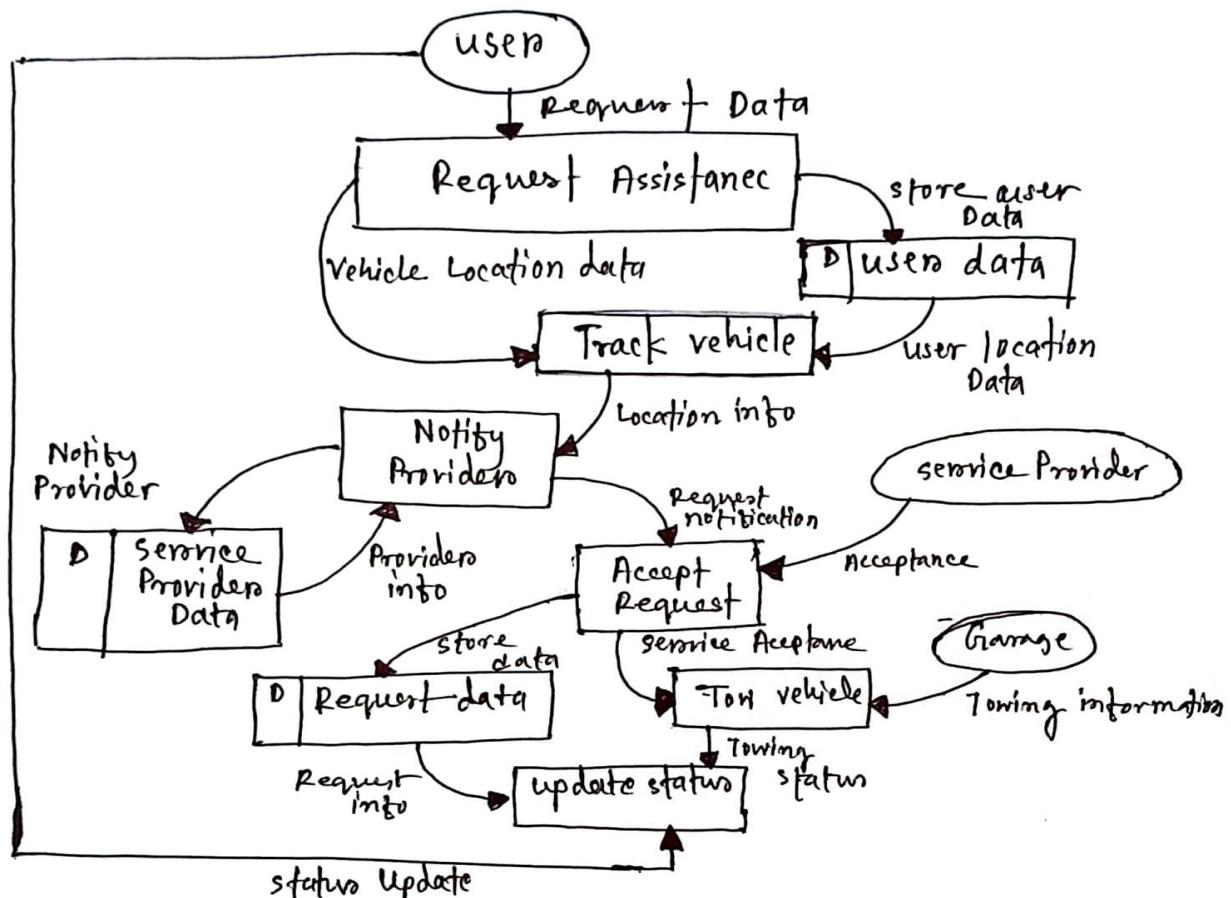


Fig No. 4: Data Flow Diagram

5. Implementation Code:

Main Source Code:

Main:

```
package com.brainybeam.roadsideassistance;
import androidx.appcompat.app.AppCompatActivity;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.os.Handler;
import android.view.animation.AlphaAnimation;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;

import com.brainybeam.roadsideassistance.Admin.DashBoard.AdminDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanEnableActivity;
import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
import com.brainybeam.roadsideassistance.User.Activity.UserEnableActivity;
import com.brainybeam.roadsideassistance.User.DashBoard.UserDashboardActivity;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData
import java.util.Objects;

import de.hdodenhof.circleimageview.CircleImageView;

public class MainActivity extends AppCompatActivity {

    private CircleImageView mainImage;
    private ImageView startImage, bikeImage;
```

```

SharedPreferences sp;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Objects.requireNonNull(getSupportActionBar()).hide();

    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    mainImage = findViewById(R.id.main_mainIcan);
    startImage = findViewById(R.id.main_starting_ican);
    bikeImage = findViewById(R.id.main_starting_ican2);

    AlphaAnimation alphaAnimation1 = new AlphaAnimation(10, 90);
    alphaAnimation1.setDuration(200);
    mainImage.setAnimation(alphaAnimation1);

    AlphaAnimation alphaAnimation = new AlphaAnimation(0, 50);
    startImage.setAnimation(alphaAnimation);

    Animation animation = AnimationUtils.loadAnimation(MainActivity.this, R.anim.translate);
    bikeImage.setAnimation(animation);

    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            if(sp.getString(SharedPreferencesData.UserID,"").isEmpty()||
sp.getString(SharedPreferencesData.UserID, "").equalsIgnoreCase("")){

                new CommonMethod(MainActivity.this, LoginActivity.class);
                finish();
            } else {

```

```

        if(sp.getString(SharedPreferencesData.UserType, "").equalsIgnoreCase("User")){
            new CommonMethod(MainActivity.this, UserDashboardActivity.class);
        } else if (sp.getString(SharedPreferencesData.UserType,
        "").equalsIgnoreCase("Foreman")){
            new CommonMethod(MainActivity.this, ForemanDashboardActivity.class);
        } else if(sp.getString(SharedPreferencesData.UserEnable,
        "").equalsIgnoreCase("UserEnable")){
            new CommonMethod(MainActivity.this, UserEnableActivity.class);
        } else if(sp.getString(SharedPreferencesData.ForemanEnable,
        "").equalsIgnoreCase("ForemanEnable")){
            new CommonMethod(MainActivity.this, ForemanEnableActivity.class);
        } else if(sp.getString(SharedPreferencesData.AdminLoginState,
        "").equalsIgnoreCase("AdminLogin")){
            new CommonMethod(MainActivity.this, AdminDashboardActivity.class);
        } else{
            new CommonMethod(MainActivity.this, LoginActivity.class);
        }
        finish();
    }
}
}, 500);
}
}

```

Signup:

```

package com.brainybeam.roadsideassistance.User.Signup;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

```

```
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import java.util.concurrent.TimeUnit;
```

```
public class UserSignupActivity extends AppCompatActivity {
```

```
    EditText FirstName, LastName, MobileNumber, Email, Password;
```

```
    Button GetOTP;
```

```
    private String sFirstName, sLastName, sMobileNumber, sEmail, sPassword;
```

```
    private final String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\.\+[a-z]+\+";
```

```
    private FirebaseAuth mAuth;
```

```

private String sPhone;
private SharedPreferences sp;
FirebaseApp firebaseApp;

Bundle bundle;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_signup);

    firebaseApp = FirebaseApp.initializeApp(getApplicationContext());
    mAuth = FirebaseAuth.getInstance();
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    FirstName = findViewById(R.id.userSignup_FirstName);
    LastName = findViewById(R.id.userSignup_LastName);
    MobileNumber = findViewById(R.id.userSignup_MobileNumber);
    Email = findViewById(R.id.userSignup_Email);
    Password = findViewById(R.id.userSignup_Password);
    GetOTP = findViewById(R.id.userSignup_GetOTPButton);

    GetOTP.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if(new ConnectionDetector(UserSignupActivity.this).isConnectingToInternet()){

                sFirstName = FirstName.getText().toString();
                sLastName = LastName.getText().toString();

```

```

sMobileNumber = MobileNumber.getText().toString().toLowerCase();
sEmail = Email.getText().toString().toLowerCase();
sPassword = Password.getText().toString();

if(sFirstName.isEmpty() || sFirstName.equalsIgnoreCase(" ")){
    FirstName.setError("FirstName is Required");
} else if(sLastName.isEmpty() || sLastName.equalsIgnoreCase(" ")){
    LastName.setError("LastName is Required");
} else if(sMobileNumber.isEmpty() || sMobileNumber.equalsIgnoreCase(" ")){
    MobileNumber.setError("Mobile number is Required");
} else if(sMobileNumber.length()!=10){
    MobileNumber.setError("Valid Mobile Number is Required");
} else if(sEmail.isEmpty() || sEmail.equalsIgnoreCase(" ")){
    Email.setError("Email is Required");
} else if(!sEmail.matches>EmailPattern){
    Email.setError("Valid Email is Required");
} else if(sPassword.isEmpty() || sPassword.equalsIgnoreCase(" ")){
    Password.setError("Password is Required");
} else if(sPassword.length()<8){
    Password.setError("Password must be 8 char long");
} else {

    sp.edit().putString(SharedPreferencesData.UserType, "User").apply();
    sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).apply();
    sp.edit().putString(SharedPreferencesData.LastName, sLastName).apply();
    sp.edit().putString(SharedPreferencesData.MobileNumber,
sMobileNumber).apply();
    sp.edit().putString(SharedPreferencesData.Email, sEmail).apply();
    sp.edit().putString(SharedPreferencesData.Password, sPassword).apply();

    bundle = new Bundle();
}

```

```

        sPhone = sMobileNumber;
        otpSendToMobile(sPhone);
    }

} else {
    new ConnectionDetector(UserSignupActivity.this).connectiondetect();
}

});

}

private void otpSendToMobile(String sPhone) {

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks =
        new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

@Override
    public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

    }

@Override
    public void onVerificationFailed(FirebaseException e) {
        new CommonMethod(UserSignupActivity.this, e.getLocalizedMessage());
    }

@Override
    public void onCodeSent(@NonNull String VerificationId,

```

```

        @NonNull PhoneAuthProvider.ForceResendingToken token) {
    new CommonMethod(UserSignupActivity.this, "OTP is successfully sent");

    bundle.putString("PhoneNumber", sPhone.trim());
    bundle.putString("Mobile_VerificationID", VerificationId);
    Intent intent = new Intent(UserSignupActivity.this, OTPVerificationActivity.class);
    intent.putExtras(bundle);
    startActivity(intent);
}

};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallbacks)
        .build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

```

OTP Verification Code:

```

package com.brainybeam.roadsideassistance.OTPVerification;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

```

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.AuthCredential;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.EmailAuthProvider;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
```

```
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebaseio.firestore.DocumentReference;
import com.google.firebaseio.firebaseio.FirebaseFirestore;

import java.util.HashMap;
import java.util.Map;
import java.util.Objects;

import org.jetbrains.annotations.NotNull;

import java.io.IOException;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class OTPVerificationActivity extends AppCompatActivity {

    //Mobile
    LinearLayout Mobile_linearlayout;
    TextView MobileNumberView;
    EditText Mobile OTP1, Mobile OTP2, Mobile OTP3, Mobile OTP4, Mobile OTP5,
    Mobile OTP6;
    TextView Mobile OTP_ResendButton;
    Button Mobile OTP Verify;

    //Email
    LinearLayout Email_linearlayout;
    TextView EmailView;
    TextView Email OTP_ResendButton;
```

```
Button Email OTP Verify;

FirebaseAuth mAuth;
FirebaseApp firebaseApp;
Firestore fStore;
private String Mobile_VerificationID;
private SharedPreferences sp;

String sMail, sPassword;
int count = 0;

double DriverLatitude, DriverLongitude;
String sForemanLatitude;
String sForemanLongitude;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_otpverification);

    firebaseApp = FirebaseApp.initializeApp(getApplicationContext());
    mAuth = FirebaseAuth.getInstance();
    fStore = Firestore.getInstance();
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    Bundle bundle = getIntent().getExtras();

    // TODO Mobile Number
    Mobile_linearlayout = findViewById(R.id.MobileOTP_linearlayout);
```

```

MobileNumberView = findViewById(R.id.MobileNumberView);
Mobile OTP1 = findViewById(R.id.Mobile_otp1);
Mobile OTP2 = findViewById(R.id.Mobile_otp2);
Mobile OTP3 = findViewById(R.id.Mobile_otp3);
Mobile OTP4 = findViewById(R.id.Mobile_otp4);
Mobile OTP5 = findViewById(R.id.Mobile_otp5);
Mobile OTP6 = findViewById(R.id.Mobile_otp6);
Mobile OTP_ResendButton = findViewById(R.id.Mobile_Resend OTP_TextButton);
Mobile OTP_Verify = findViewById(R.id.Mobile OTP Verify VerifyButton);

assert bundle != null;
String sMobileNumber = "+880" + bundle.getString("PhoneNumber");
MobileNumberView.setText(sMobileNumber);
Mobile_VerificationID = bundle.getString("Mobile_VerificationID");

// TODO Email
Email_linearlayout = findViewById(R.id.EmailOTP_linearlayout);
EmailView = findViewById(R.id.EmailView);
Email_linearlayout.setVisibility(View.GONE);

Email OTP_ResendButton = findViewById(R.id.Email_Resend OTP_TextButton);
Email OTP_Verify = findViewById(R.id.Email OTP Verify VerifyButton);

// TODO To Sending OTP in Mobile Number
Mobile_InputOTPInEditText();
}

Mobile OTP_ResendButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        new CommonMethod(OTPVerificationActivity.this, "OTP Send SuccessFully");
        otpSendToMobile(Objects.requireNonNull(bundle.getString("PhoneNumber")));
    }
});

```

```

        }

    });

Mobile OTP Verify.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        if (new ConnectionDetector(OTPVerificationActivity.this).isConnectingToInternet()) {
            // TODO Mobile OTP Verify
            MobileOTP_Validity_Check();
            if (count == 2) {
                FullyVerify();
            }
        } else {
            new ConnectionDetector(OTPVerificationActivity.this).connectiondetect();
        }
    }

    // TODO To Sending OTP in Email
    sMail = sp.getString(SharedPreferencesData.Email, "");
    EmailView.setText(sMail);
    sPassword = sp.getString(SharedPreferencesData.Password, "");

    // TODO Email OTP Send
    //otpSendToEmail(sMail);

Email OTP Verify.setOnClickListener(new View.OnClickListener() {
    @Override

```

```

public void onClick(View view) {

    if (new ConnectionDetector(OTPVerificationActivity.this).isConnectingToInternet()) {

        EmailVerify();

        if (count == 2) {
            FullyVerify();
        }

    } else {
        new ConnectionDetector(OTPVerificationActivity.this).connectiondetect();
    }
}

});

}

```

```

private void FullyVerify() {
    if (sp.getString(SharedPreferencesData.UserType, "").equalsIgnoreCase("User")) {
        String userID = Objects.requireNonNull(mAuth.getCurrentUser()).getUid();
        DocumentReference documentReference = fStore.collection("Users").document(userID);
        Map<String, Object> user = new HashMap<>();
        user.put("UserType", sp.getString(SharedPreferencesData.UserType, ""));
        user.put("FirstName", sp.getString(SharedPreferencesData.FirstName, ""));
        user.put("LastName", sp.getString(SharedPreferencesData.LastName, ""));
        user.put("MobileNumber", sp.getString(SharedPreferencesData.MobileNumber, ""));
        user.put("Email", sp.getString(SharedPreferencesData.Email, ""));
        user.put("Password", sp.getString(SharedPreferencesData.Password, ""));
        user.put("Active_Status", sp.getBoolean(SharedPreferencesData.Active_Status, true));
    }
}

```

```

        user.put("Account_Status", "Verified");
        documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                Toast.makeText(OTPVerificationActivity.this, "Sign Up Successful",
                        Toast.LENGTH_SHORT).show();
                Log.d("TAG", "onSuccess: User Profile is Created for " + userID);
            }
        });

    });
    new CommonMethod(OTPVerificationActivity.this, LoginActivity.class);
    finish();
} else if (sp.getString(SharedPreferencesData.UserType, "").equalsIgnoreCase("Foreman"))
{
    AddSPPProfileLocationData();
    String userID = Objects.requireNonNull(mAuth.getCurrentUser()).getUid();
    DocumentReference documentReference = fStore.collection("Users").document(userID);
    Map<String, Object> user = new HashMap<>();
    user.put("UserType", sp.getString(SharedPreferencesData.UserType, ""));
    user.put("FirstName", sp.getString(SharedPreferencesData.FirstName, ""));
    user.put("LastName", sp.getString(SharedPreferencesData.LastName, ""));
    user.put("MobileNumber", sp.getString(SharedPreferencesData.MobileNumber, ""));
    user.put("Email", sp.getString(SharedPreferencesData.Email, ""));
    user.put("Password", sp.getString(SharedPreferencesData.Password, ""));
    user.put("ForemanAddress", sp.getString(SharedPreferencesData.ForemanAddress, ""));
    user.put("ForemanArea", sp.getString(SharedPreferencesData.ForemanArea, ""));
    user.put("ForemanCity", sp.getString(SharedPreferencesData.ForemanCity, ""));
    user.put("ForemanState", sp.getString(SharedPreferencesData.ForemanState, ""));
    user.put("ForemanAccount_Status",
            sp.getBoolean(SharedPreferencesData.ForemanAccount_Status, false));
    user.put("sForemanLatitude", sForemanLatitude);
}

```

```

        user.put("sForemanLongitude", sForemanLongitude);
        user.put("Account_Status", "Verified");

    }

    documentReference.set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void aVoid) {
            Toast.makeText(OTPVerificationActivity.this, "Sign Up Successful",
                    Toast.LENGTH_SHORT).show();
            Log.d("TAG", "onSuccess: User Profile is Created for " + userID);
        }
    });

});

new CommonMethod(OTPVerificationActivity.this, LoginActivity.class);
finish();

} else {
    new CommonMethod(OTPVerificationActivity.this, "Not Valid Try Again");
}

}

private void AddSPPProfileLocationData() {

    String FullAddress = sp.getString(SharedPreferencesData.ForemanAddress, "") + "," +
            sp.getString(SharedPreferencesData.ForemanArea, "") + "," +
            sp.getString(SharedPreferencesData.ForemanCity, "") + "," +
            sp.getString(SharedPreferencesData.ForemanState, "");

    Geocoder coder = new Geocoder(OTPVerificationActivity.this);
    List<Address> address;
    try {

```

```

// May throw an IOException
address = coder.getFromLocationName(FullAddress, 5);
if (address == null) {
    DriverLatitude = 0.00;
    DriverLongitude = 0.00;
}
assert address != null;
Address location = address.get(0);

DriverLatitude = location.getLatitude();
DriverLongitude = location.getLongitude();
//p1 = new LatLng(location.getLatitude(), location.getLongitude() );

} catch (IOException ex) {
    ex.printStackTrace();
}

sForemanLatitude = String.valueOf(DriverLatitude);
sForemanLongitude = String.valueOf(DriverLongitude);
}

```

```

private void Mobile_InputOTPInEditTextField() {
    Mobile OTP1.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {

        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {

```

```
        Mobile OTP2.requestFocus();  
    }  
  
    @Override  
    public void afterTextChanged(Editable s) {  
  
    }  
});  
Mobile OTP2.addTextChangedListener(new TextWatcher() {  
    @Override  
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
    }  
  
    @Override  
    public void onTextChanged(CharSequence s, int start, int before, int count) {  
        Mobile OTP3.requestFocus();  
    }  
  
    @Override  
    public void afterTextChanged(Editable s) {  
  
    }  
});  
Mobile OTP3.addTextChangedListener(new TextWatcher() {  
    @Override  
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {  
  
    }  
  
    @Override  
    public void afterTextChanged(Editable s) {  
  
    }  
});
```

```
public void onTextChanged(CharSequence s, int start, int before, int count) {
    Mobile OTP4.requestFocus();
}

@Override
public void afterTextChanged(Editable s) {

};

Mobile OTP4.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            Mobile OTP5.requestFocus();
        }
    }

    @Override
    public void afterTextChanged(Editable s) {

};

Mobile OTP5.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        Mobile OTP6.requestFocus();
    }
}
```

```

    }

    @Override
    public void afterTextChanged(Editable s {

        }
    });

}

private void MobileOTP_Validity_Check() {

    if (Mobile OTP1.getText().toString().trim().isEmpty() ||
        Mobile OTP2.getText().toString().trim().isEmpty() ||
        Mobile OTP3.getText().toString().trim().isEmpty() ||
        Mobile OTP4.getText().toString().trim().isEmpty() ||
        Mobile OTP5.getText().toString().trim().isEmpty() ||
        Mobile OTP6.getText().toString().trim().isEmpty()) {

    new CommonMethod(OTPVerificationActivity.this, "Mobile OTP is not Valid!");

} else {

    if (Mobile_VerificationID != null) {

        String code = Mobile OTP1.getText().toString().trim() +
            Mobile OTP2.getText().toString().trim() +
            Mobile OTP3.getText().toString().trim() +
            Mobile OTP4.getText().toString().trim() +
            Mobile OTP5.getText().toString().trim() +
            Mobile OTP6.getText().toString().trim();

```

```
PhoneAuthCredential credential
PhoneAuthProvider.getCredential(Mobile_VerificationID, code);

FirebaseAuth
    .getInstance()
    .signInWithCredential(credential)
    .addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull @NotNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                new CommonMethod(OTPVerificationActivity.this, "Mobile Number Verified");
                count = count + 1;
            }
        }
    });

AuthCredential credential = EmailAuthProvider.getCredential(sMail, sPassword);
link_email(credential);
Mobile_linearlayout.setVisibility(View.GONE);
Email_linearlayout.setVisibility(View.VISIBLE);
if (count == 2) {
    FullyVerify();
}
} else {
    new CommonMethod(OTPVerificationActivity.this, "OTP is not Valid!");
}
});

}

}

}
```

```

private void link_email(AuthCredential credential) {
    mAuth.getCurrentUser().linkWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    new CommonMethod(OTPVerificationActivity.this,
                        "linkWithCredential:success");
                    FirebaseUser user = task.getResult().getUser();
                    otpSendToEmail(sMail);
                } else {
                    new CommonMethod(OTPVerificationActivity.this, "Authentication failed.");
                }
            }
        });
}

```

```

private void otpSendToEmail(String sMail) {
    // TODO Sign in success, update UI with the signed-in user's information
    FirebaseUser user = mAuth.getCurrentUser();
    if (user != null) {
        user.sendEmailVerification().addOnCompleteListener(new
            OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    if (task.isSuccessful()) {
                        AlertDialog.Builder alertDialogBuilder = new
                            AlertDialog.Builder(OTPVerificationActivity.this);
                        // set title

```

```

        alertDialogBuilder.setTitle("Please Verify Your EmailID");

        // set dialog message
        alertDialogBuilder.setMessage("A verification Email Is Sent To Your Registered
EmailID, please click on the link and Sign in again!")
        .setCancelable(false)
        .setPositiveButton("Sign In", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {

                }

            });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();
        // show it
        alertDialog.show();

    }

}

});

}

}

}

public void EmailVerify() {

    mAuth.signInWithEmailAndPassword(sp.getString(SharedPreferencesData.Email, ""),
sp.getString(SharedPreferencesData.Password, "")).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {

```

```

        if (Objects.requireNonNull(mAuth.getCurrentUser()).isEmailVerified()) {
            new CommonMethod(OTPVerificationActivity.this, "Email is Verified");
            count = count + 1;
            Email_linearlayout.setVisibility(View.GONE);
            if (count == 2) {
                FullyVerify();
            }
        } else {
            new CommonMethod(OTPVerificationActivity.this, "Email is Not Verified");
        }
    });
}

}

```

```

private void otpSendToMobile(String sPhone) {

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks =
        PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

        }

        @Override
        public void onVerificationFailed(FirebaseException e) {

            new CommonMethod(OTPVerificationActivity.this, e.getLocalizedMessage());
        }
    };
}

```

```

    }

    @Override
    public void onCodeSent(@NonNull String VerificationId,
                          @NonNull PhoneAuthProvider.ForceResendingToken token) {
        new CommonMethod(OTPVerificationActivity.this, "OTP is successFully Send");

    }
};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallbacks)
        .build();

PhoneAuthProvider.verifyPhoneNumber(options);

}
}

```

Login:

```
package com.brainybeam.roadsideassistance.Login;
```

```
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SwitchCompat;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Admin.AdminCredential.AdminCredential;
import com.brainybeam.roadsideassistance.Admin.DashBoard.AdminDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanForgotPasswordActivity;
import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.Signup.ForemanSignupActivity;
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.Notification.PushNotification.Config;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.UpdateFCMIDData;
import com.brainybeam.roadsideassistance.User.Activity.UserForgotPasswordActivity;
```

```
import com.brainybeam.roadsideassistance.User.DashBoard.UserDashboardActivity;
import com.brainybeam.roadsideassistance.User.Signup.UserSignupActivity;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebaseio.FirebaseException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebase.firestore.DocumentReference;
import com.google.firebase.firestore.DocumentSnapshot;
import com.google.firebase.firestore.EventListener;
import com.google.firebase.firestore.FirebaseFirestore;
import com.google.firebase.firestore.FirebaseFirestoreException;
import com.google.firebase.messaging.FirebaseMessaging;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import retrofit2.Call;
import retrofit2.Callback;
```

```
import retrofit2.Response;

public class LoginActivity extends AppCompatActivity {

    private EditText EmailORMobile, Password;
    private TextView ForgotPassword;
    SwitchCompat emailPhoneLoginSwitch;

    private ArrayList<String> AdminEmail;
    private ArrayList<String> AdminPassword;

    private final String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\".+[a-z]+";

    String sEmailORMobile, sPassword, Mobile_VerificationID = null;

    SharedPreferences sp;
    GPSTracker gpsTracker;
    //ApiInterface apiInterface;
    FirebaseFirestore fStore;
    private FirebaseAuth mAuth;

    // TODO FCMID For Notification
    BroadcastReceiver broadcastReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        int or = getResources().getConfiguration().orientation;
```

```

if (or == Configuration.ORIENTATION_LANDSCAPE) {
    Objects.requireNonNull(getSupportActionBar()).hide();
}

gpsTracker = new GPSTracker(LoginActivity.this);
//apiInterface = ApiClient.getClient().create(ApiInterface.class);
sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);
fStore = FirebaseFirestore.getInstance();
mAuth = FirebaseAuth.getInstance();

AdminEmail = new ArrayList<>();
AdminPassword = new ArrayList<>();

List<String> list1 = Arrays.asList(AdminCredential.AdminEmail);
List<String> list2 = Arrays.asList(AdminCredential.AdminPassword);
AdminEmail.addAll(list1);
AdminPassword.addAll(list2);

EmailORMobile = findViewById(R.id.login_EmailORMobile);
EmailORMobile.setHint(R.string.email);//value

Password = findViewById(R.id.login_Password);
Password.setHint(R.string.password);//value
ForgotPassword = findViewById(R.id.login_forgotPassword);
Button login = findViewById(R.id.login_loginButton);
Button getotp = findViewById(R.id.login_getotp);
getotp.setVisibility(View.GONE);
TextView signup = findViewById(R.id.login_signupButton);
emailPhoneLoginSwitch = findViewById(R.id.EmailPhoneLoginSwitch);

```

```

// Check if GPS enabled
if (!gpsTracker.canGetLocation()) {
    gpsTracker.showSettingsAlert();
}

login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sEmailORMobile = EmailORMobile.getText().toString();
        sPassword = Password.getText().toString();

        if (sEmailORMobile.isEmpty() || sEmailORMobile.equalsIgnoreCase(" ")) {
            EmailORMobile.setError("Email OR Mobile Number is Required");
        } else if (!sEmailORMobile.matches>EmailPattern) && sEmailORMobile.length() <
10) {
            EmailORMobile.setError("Valid Email OR Mobile Number is Required");
        } else if (sPassword.isEmpty() || sPassword.equalsIgnoreCase(" ")) {
            Password.setError("Password is Required");
        } else if (AdminEmail.contains(sEmailORMobile) &&
AdminPassword.contains(sPassword)) {
            AdminFCMIDStore();
            sp.edit().putString(SharedPreferencesData.AdminLoginState,
"AdminLogin").apply();
            new CommonMethod(LoginActivity.this, AdminDashboardActivity.class);
            finish();
        } else {

            if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
                if (emailPhoneLoginSwitch.isChecked()) {

```

```

        Email_loginMethod(sEmailORMobile, sPassword);
    } else {
        Mobile_loginMethod();
    }

} else {
    new ConnectionDetector(LoginActivity.this).connectiondetect();
}
}

});

signup.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(LoginActivity.this);

        alertDialog.setPositiveButton("Signup As User", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, UserSignupActivity.class);
            }
        });
        alertDialog.setNeutralButton("Signup As Foreman", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, ForemanSignupActivity.class);
            }
        });
    }
});

```

```
        }

    });

    alertDialog.show();

}

});

emailPhoneLoginSwitch.setOnCheckedChangeListener(new  
CompoundButton.OnCheckedChangeListener() {  
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
        if (isChecked) {  
            EmailORMobile.setHint(R.string.email);  
  
EmailORMobile.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);  
            Password.setHint(R.string.password);  
            Password.setInputType(InputType.TYPE_CLASS_TEXT  
|  
InputType.TYPE_TEXT_VARIATION_PASSWORD);  
            EmailORMobile.setText("");  
            Password.setText("");  
            ForgotPassword.setVisibility(View.VISIBLE);  
            getotp.setVisibility(View.GONE);  
        } else {  
            EmailORMobile.setHint(R.string.phone_number);  
            EmailORMobile.setInputType(InputType.TYPE_CLASS_PHONE);  
            Password.setHint(R.string.otp);  
            Password.setInputType(InputType.TYPE_CLASS_PHONE);  
            EmailORMobile.setText("");  
            Password.setText("");  
            ForgotPassword.setVisibility(View.GONE);  
            getotp.setVisibility(View.VISIBLE);  
    }
}
```

```

        }
    }
});

getotp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String sPhone = EmailORMobile.getText().toString();
        otpSendToMobile(sPhone);
    }
});

ForgotPassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(LoginActivity.this);

        alertDialog.setPositiveButton("User", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, UserForgotPasswordActivity.class);
            }
        });

        alertDialog.setNeutralButton("Foreman", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, ForemanForgotPasswordActivity.class);
            }
        });
    }
});

```

```

        alertDialog.setCancelable(true);
        alertDialog.show();
    }
});

}

private void AdminFCMIDStore() {

    getAdminToken();

}

private void otpSendToMobile(String sPhone) {

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks = new
    PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

        }

        @Override
        public void onVerificationFailed(FirebaseException e) {

            new CommonMethod(LoginActivity.this, e.getLocalizedMessage());
        }

        @Override

```

```

public void onCodeSent(@NonNull String VerificationId,
                      @NonNull PhoneAuthProvider.ForceResendingToken token) {
    Mobile_VerificationID = VerificationId;
    new CommonMethod(LoginActivity.this, "OTP is successFully Send");
}

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallbacks)
        .build();
PhoneAuthProvider.verifyPhoneNumber(options);

}

// TODO Login Method Start
public void Mobile_loginMethod() {
    if (Mobile_VerificationID == null) {
        new CommonMethod(LoginActivity.this, "OTP is not Send");
        return;
    }
    String code = sPassword.trim();
    PhoneAuthCredential credential
    mAuth.signInWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

```

```

    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()) {
            LoginMethod(Objects.requireNonNull(mAuth.getCurrentUser()).getUid());
            new CommonMethod(LoginActivity.this, "Successful Login ");
        } else {
            new CommonMethod(LoginActivity.this, "Login Error");
        }
    }
});

}

public void Email_loginMethod(String sEmailORMobile, String sPassword) {
    mAuth.signInWithEmailAndPassword(sEmailORMobile,
        sPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                LoginMethod(Objects.requireNonNull(mAuth.getCurrentUser()).getUid());
            } else {
                new CommonMethod(LoginActivity.this, "Login Error");
            }
        }
    });
}

public void LoginMethod(String userID) {
    sp.edit().putString(SharedPreferencesData.UserID, userID).apply();
    DocumentReference documentReference = fStore.collection("Users").document(userID);
}

```

```

documentReference.addSnapshotListener(new EventListener<DocumentSnapshot>() {
    @Override
    public void onEvent(@Nullable DocumentSnapshot value, @Nullable
FirebaseFirestoreException error) {
        assert value != null;
        sp.edit().putString(SharedPreferencesData.UserType,
value.getString("UserType")).apply();
        sp.edit().putString(SharedPreferencesData.FirstName,
value.getString("FirstName")).apply();
        sp.edit().putString(SharedPreferencesData.LastName,
value.getString("LastName")).apply();
        sp.edit().putString(SharedPreferencesData.MobileNumber,
value.getString("MobileNumber")).apply();
        sp.edit().putString(SharedPreferencesData.Email, value.getString("Email")).apply();
        sp.edit().putString(SharedPreferencesData.Password,
value.getString("Password")).apply();
        sp.edit().putString(SharedPreferencesData.Account_Status,
value.getString("Account_Status")).apply();
        sp.edit().putBoolean(SharedPreferencesData.Active_Status,
Boolean.TRUE.equals(value.getBoolean("Active_Status"))).apply();

        // TODO Notification FCMID
        broadcastReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                if (Objects.equals(intent.getAction(), Config.REGISTRATION_COMPLETE)) {

FirebaseMessaging.getInstance().subscribeToTopic(Config.TOPIC_GLOBAL);
                    getUserToken();
                } else if (Objects.equals(intent.getAction(), Config.PUSH_NOTIFICATION)) {
                    String s = intent.getStringExtra("message");

```

```

        } else {
        }
    }
};

if (Objects.requireNonNull(value.getString("UserType")).equalsIgnoreCase("User")) {
    getToken();
    new CommonMethod(LoginActivity.this, UserDashboardActivity.class);
    finish();
} else {
    sp.edit().putString(SharedPreferencesData.ForemanAddress,
value.getString("ForemanAddress")).apply();
    sp.edit().putString(SharedPreferencesData.ForemanArea,
value.getString("ForemanArea")).apply();
    sp.edit().putString(SharedPreferencesData.ForemanCity,
value.getString("ForemanCity")).apply();
    sp.edit().putString(SharedPreferencesData.ForemanState,
value.getString("ForemanState")).apply();
    sp.edit().putBoolean(SharedPreferencesData.ForemanAccount_Status,
Boolean.TRUE.equals(value.getBoolean("ForemanAccount_Status"))).apply();
    getToken();
    new CommonMethod(LoginActivity.this, ForemanDashboardActivity.class);
    finish();
}
});

// sp.edit().putString(SharedPreferencesData.ProfileImage,
data.response.get(i).profileImage).commit();
}

// TODO Login Method End

```

```

private void getAdminToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
OnSuccessListener<String>() {
    @Override
    public void onSuccess(String stoken) {
        Log.d("RESPONSE_TOKEN", stoken);
        sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).commit();

        if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
            if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).commit();
            }
        } else {
            new ConnectionDetector(LoginActivity.this).connectiondetect();
        }
    }
});

}

private void getUserToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
OnSuccessListener<String>() {
    @Override
    public void onSuccess(String stoken) {
        Log.d("RESPONSE_TOKEN", stoken);
        sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).apply();

        if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
            if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).apply();
            }
        }
    }
});
}

```

```

        } else {
            new ConnectionDetector(LoginActivity.this).connectiondetect();
        }
    }
});
```

```

private void getForemanToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
    OnSuccessListener<String>() {
        @Override
        public void onSuccess(String stoken) {
            Log.d("RESPONSE_TOKEN", stoken);
            sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).apply();

            if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
                if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                    sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).apply();
                }
            } else {
                new ConnectionDetector(LoginActivity.this).connectiondetect();
            }
        }
    });
}
```

```

private void addUserFcmIDData() {
//    sp.getString(SharedPreferencesData.UserID, ""),
//    sp.getString(SharedPreferencesData.UserFCMID, "")
```

```
        }

    private void addForemanFcmIDData() {
        //    sp.getString(SharedPreferencesData.UserID, ""),
        //    sp.getString(SharedPreferencesData.ForemanFCMID, "")
    }

}
```

Admin Dashboard:

```
package com.brainybeam.roadsideassistance.Admin.DashBoard;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.FragmentManager;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
```

```
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanHomeFragment;
import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.Notification.ForemanNotificationActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.ConstantData;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.material.navigation.NavigationView;

public class AdminDashboardActivity extends AppCompatActivity {

    TextView Title_menu;
    ImageView menu_image, calling_image, Notification_image;

    // TODO Navigation
    LinearLayout nav_header_home_layout, nav_header_NewForemanRequest_layout,
    nav_header_ActiveUsers_layout,
    nav_header_ActiveServiceProvider_layout, nav_header_Logout_layout;
```

```

ImageView nav_header_home_image_logo, nav_header_NewForemanRequest_image_logo,
nav_header_ActiveUsers_image_logo,
    nav_header_ActiveServiceProvider_image_logo, nav_header_Logout_image_logo;

TextView nav_header_home_TextButton, nav_header_NewForemanRequest_TextButton,
nav_header_ActiveUsers_TextButton,
    nav_header_ActiveServiceProvider_TextButton, nav_header_Logout_TextButton;

SharedPreferences sp;
ApiInterface apiInterface;
ProgressDialog pd;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_admin_dashboard);
//    getSupportActionBar().setTitle("Home");
//    ColorDrawable colorDrawable
//        = new ColorDrawable(Color.parseColor("#2196F3"));
//    getSupportActionBar().setBackgroundDrawable(colorDrawable);
//    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);
apiInterface = ApiClient.getClient().create(ApiInterface.class);

pd = new ProgressDialog(AdminDashboardActivity.this);
pd.setMessage("Network Connecting...");
pd.setCancelable(false);
pd.show();

```

```

if(new ConnectionDetector(AdminDashboardActivity.this).isConnectingToInternet()){
    pd.dismiss();
} else {

    if(!new ConnectionDetector(AdminDashboardActivity.this).isConnectingToInternet()){
        pd.setMessage("Please Check Your Network");
    }
    new ConnectionDetector(AdminDashboardActivity.this).connectiondetect();
    pd.dismiss();
}

// TODO Tab activity Start

Title_menu = findViewById(R.id.admin_content_main_dashboard_title);
Title_menu.setText("Home");

final DrawerLayout drawer = (DrawerLayout) findViewById(R.id.admin_drawer_layout);
menu_image = findViewById(R.id.admin_content_main_dashboard_menu_imageButton);
menu_image.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        drawer.openDrawer(GravityCompat.START);

    }
});

calling_image = findViewById(R.id.admin_content_main_dashboard_call_imageButton);
calling_image.setOnClickListener(new View.OnClickListener() {

```

```

@Override
public void onClick(View view) {

    AlertDialog.Builder builder = new AlertDialog.Builder(AdminDashboardActivity.this);
    builder.setTitle("Call Assistance Center?");
    builder.setPositiveButton("CALL", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            Intent intent = new Intent(Intent.ACTION_CALL);
            intent.setData(Uri.parse("tel:"+ ConstantData.Call_Assistance_MobileNumber));
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                if (checkSelfPermission(Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                    // TODO: Consider calling
                    // Activity#requestPermissions
                    // here to request the missing permissions, and then overriding
                    // public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                    //                                         int[] grantResults)
                    // to handle the case where the user grants the permission. See the
documentation
                    // for Activity#requestPermissions for more details.
                    return;
                }
            }
            startActivity(intent);
        }
    });
    builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
        @Override

```

```

        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    builder.show();

}

});

// TODO Tab activity finished

Notification_image =  

findViewById(R.id.admin_content_main_dashboard_Notification_image);  

Notification_image.setOnClickListener(new View.OnClickListener() {  

    @Override  

    public void onClick(View view) {  

        new CommonMethod(AdminDashboardActivity.this, AdminDashboardActivity.class);  

    }
});  

// TODO Default Fragment  

drawer.closeDrawer(GravityCompat.START);  

FragmentManager manager = getSupportFragmentManager();  

manager.beginTransaction()  

.replace(R.id.admin_content_main_dashboard_FragmentContainerView, new  

AdminHomeFragment(), null)  

.setReorderingAllowed(true)  

.addToBackStack("")
```

```
.commit();

// TODO -----
// TODO      NavigationView Start
// TODO -----



    NavigationView           navigationView      =      (NavigationView)
findViewById(R.id.admin_dashboard_nav_view);

    View header = navigationView.getHeaderView(0);





nav_header_home_layout = header.findViewById(R.id.admin_nav_header_home_layout);
nav_header_NewForemanRequest_layout      =
header.findViewById(R.id.admin_nav_header_NewForemanRequest_layout);
nav_header_ActiveUsers_layout      =
header.findViewById(R.id.admin_nav_header_activeUsers_layout);
nav_header_ActiveServiceProvider_layout =
header.findViewById(R.id.admin_nav_header_activeServiceProvider_layout);
nav_header_Logout_layout      =
header.findViewById(R.id.admin_nav_header_logout_layout);





nav_header_home_image_logo      =
header.findViewById(R.id.admin_nav_header_home_imageview_logo);
nav_header_NewForemanRequest_image_logo      =
header.findViewById(R.id.admin_nav_header_NewForemanRequest_imageview_logo);
nav_header_ActiveUsers_image_logo      =
header.findViewById(R.id.admin_nav_header_activeUsers_imageview_logo);
nav_header_ActiveServiceProvider_image_logo =
header.findViewById(R.id.admin_nav_header_activeServiceProvider_imageview_logo);
nav_header_ActiveServiceProvider_image_logo =
header.findViewById(R.id.admin_nav_header_activeServiceProvider_imageview_logo);
```

```

nav_header_Logout_image_logo =  

header.findViewById(R.id.admin_header_logout_imageview_logo);  
  

nav_header_home_TextButton =  

header.findViewById(R.id.admin_header_home_TextButton);  

nav_header_NewForemanRequest_TextButton =  

header.findViewById(R.id.admin_header_NewForemanRequest_TextButton);  

nav_header_ActiveUsers_TextButton =  

header.findViewById(R.id.admin_header_activeUsers_TextButton);  

nav_header_ActiveServiceProvider_TextButton =  

header.findViewById(R.id.admin_header_activeServiceProvider_TextButton);  

nav_header_Logout_TextButton =  

header.findViewById(R.id.admin_header_logout_TextButton);

```

```

// TODO Admin Home Fragment
nav_header_home_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Home");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminHomeFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
}

```

```
});

nav_header_home_image_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Home");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminHomeFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
nav_header_home_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Home");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminHomeFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```

    }

});

// TODO Admin New Foreman Request Fragment
nav_header_NewForemanRequest_TextButton.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("New Foreman Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminNewForemanRequestFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});

nav_header_NewForemanRequest_image_logo.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("New Foreman Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()

```

```
        .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminNewForemanRequestFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
    }
});
```

```
nav_header_NewForemanRequest_layout.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View view) {
        Title_menu.setText("New Foreman Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
        .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminNewForemanRequestFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
    }
});
```

```
// TODO Admin Active Users Fragment
nav_header_ActiveUsers_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Users");
        drawer.closeDrawer(GravityCompat.START);
```

```

FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveUsersFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_ActiveUsers_image_logo.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Users");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveUsersFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_ActiveUsers_layout.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Users");

```

```

        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveUsersFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

// TODO Admin Active Service Providers Fragment
nav_header_ActiveServiceProvider_TextButton.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Service Providers");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveServiceProviderFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

```

```
nav_header_ActiveServiceProvider_image_logo.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Service Providers");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveServiceProviderFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
nav_header_ActiveServiceProvider_layout.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Active Service Providers");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.admin_content_main_dashboard_FragmentContainerView, new
AdminActiveServiceProviderFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
```

```

        }
    });

// TODO Admin Logout Fragment
nav_header_Logout_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(AdminDashboardActivity.this,
LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

nav_header_Logout_image_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(AdminDashboardActivity.this,
LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

nav_header_Logout_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(AdminDashboardActivity.this,
LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

```

```

        }

    });

}

@Override
public void onBackPressed() {
    //    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.foreman_drawer_layout);
    //    if (drawer.isDrawerOpen(GravityCompat.START)) {
    //        drawer.closeDrawer(GravityCompat.START);
    //    } else {
    //        //super.onBackPressed();
    AlertDialog.Builder builder = new AlertDialog.Builder(AdminDashboardActivity.this);
    builder.setTitle(getResources().getString(R.string.app_name));
    builder.setMessage("Are You Sure Want To Exit!");
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            finishAffinity();
        }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    builder.show();
    // }
}

```

```
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_notification, menu);
    return true;
}
```

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        onBackPressed();
    }
    if(id == R.id.notification_menu){
        new CommonMethod(AdminDashboardActivity.this, AdminDashboardActivity.class);
    }
    return super.onOptionsItemSelected(item);
}
}
```

Admin Profile:

```
package com.brainybeam.roadsideassistance.Admin.DashBoard;
```

```
import android.app.AlertDialog;
import android.content.Context;
import android.content.SharedPreferences;
```

```
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;

public class AdminHomeFragment extends Fragment {

    TextView AppUser, NumberOfUser, NumberOfForeman;

    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;
    int sum;

    public AdminHomeFragment() {
        // Required empty public constructor
    }
}
```

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_admin_home, container, false);

    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp          = getSharedPreferences(SharedPreferencesData.PREF,
                                      Context.MODE_PRIVATE);

    AppUser = view.findViewById(R.id.frag_admin_home_activeAppUsers_TextView);
    NumberOfUser = view.findViewById(R.id.frag_admin_home_activeUsers_TextView);
    NumberOfForeman = view.findViewById(R.id.frag_admin_home_activeForeman_TextView);

    if(sp.getString(SharedPreferencesData.Admin_NumberOfUsers, "").equalsIgnoreCase("") ||
       sp.getString(SharedPreferencesData.Admin_NumberOfForeman,
                   "").equalsIgnoreCase("")){

        sum = 0;
    } else{

        sum = Integer.valueOf(sp.getString(SharedPreferencesData.Admin_NumberOfUsers, ""))
        +
        Integer.valueOf(sp.getString(SharedPreferencesData.Admin_NumberOfForeman,
                                    ""));
    }

    pd = new ProgressDialog(getActivity());
    pd.setMessage("Network Connecting...");
    pd.setCancelable(false);
    pd.show();
}

```

```

if(new ConnectionDetector(getActivity()).isConnectingToInternet()){
    pd.dismiss();
} else {

    if(!new ConnectionDetector(getActivity()).isConnectingToInternet()){
        pd.setMessage("Please Check Your Network");
    }
    new ConnectionDetector(getActivity()).connectiondetect();
    pd.dismiss();
}

sp.edit().putString(SharedPreferencesData.Admin_NumberOfTotleUsers,
String.valueOf(sum)).commit();

AppUser.setText(sp.getString(SharedPreferencesData.Admin_NumberOfTotleUsers,
""")+"+");
NumberOfUser.setText(sp.getString(SharedPreferencesData.Admin_NumberOfUsers,
""")+"+");

NumberOfForeman.setText(sp.getString(SharedPreferencesData.Admin_NumberOfForeman,
""")+"+");

return view;
}
}

```

New user request Recieve:

```
package com.brainybeam.roadsideassistance.Foreman.DashBoard;
```

```
import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.os.Bundle;

import androidx.core.app.ActivityCompat;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.provider.Settings;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.brainybeam.roadsideassistance.Foreman.CustomArrayList.ForemanUserSelectedServicesList;
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.GetForemanUserSelectedServicesData;
```

```
import com.brainybeam.roadsideassistance.User.Activity.UserForemanServicesActivity;
import com.brainybeam.roadsideassistance.User.Activity.UserVehicleSelectionActivity;
import com.brainybeam.roadsideassistance.User.CustomArrayList.UserForemanServiceList;
import com.brainybeam.roadsideassistance.User.DashBoard.UserVehicleDetailAdapter;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
```

```
public class NewUserRequestReceiveFragment extends Fragment {
```

```
    RecyclerView recyclerView;
    ArrayList<ForemanUserSelectedServicesList> arrayList;
    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;
    GPSTracker gpsTracker;
```

```

double CurrentLatitude, CurrentLongitude;

String sForemanAddress, sForemanLatitude, sForemanLongitude;

private static final int REQUEST_LOCATION = 1;
LocationManager locationManager;

public NewUserRequestReceiveFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_new_user_request_receive, container, false);

    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp          = getSharedPreferences(SharedPreferencesData.PREF,
    Context.MODE_PRIVATE);

    recyclerView
        =
    view.findViewById(R.id.frag_foreman_newUser_requestReceive_recyclerview);

    recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
    recyclerView.setItemAnimator(new DefaultItemAnimator());

    locationManager
        =
        (LocationManager)
    getActivity().getSystemService(Context.LOCATION_SERVICE);
    if (!locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {

```

```

        OnGPS();
    } else {
        getLocation();
    }
//    GPSTracker gpsTracker = new GPSTracker(getActivity());
//
Geocoder geocoder = new Geocoder(getActivity(), Locale.getDefault());
List<Address> addresses = null;
try {
    addresses = geocoder.getFromLocation(CurrentLatitude, CurrentLongitude, 1);
    String add = addresses.toString();
    //    String city = addresses.get(0).getLocality();
    //    String state = addresses.get(0).getAdminArea();
    //    String zip = addresses.get(0).getPostalCode();
    //    String country = addresses.get(0).getCountryName();

    //    CurrentLatitude = gpsTracker.getLatitude();
    //    CurrentLongitude = gpsTracker.getLongitude();
    sForemanAddress = add;
} catch (IOException e) {
    e.printStackTrace();
}

sp.edit().putString(SharedPreferencesData.Foreman_ForemanCurrLocation,
sForemanAddress).commit();

if(new ConnectionDetector(getActivity()).isConnectingToInternet()){
    pd = new ProgressDialog(getActivity());
    pd.setMessage("Please Wait... ");
    pd.setCancelable(false);
}

```

```

        pd.show();

        recyclerViewSetMethod();

    } else {
        new ConnectionDetector(getActivity()).connectiondetect();
    }

    return view;
}

private void OnGPS() {
    final AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setMessage("Enable GPS").setCancelable(false).setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
        }
    }).setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.cancel();
        }
    });
    final AlertDialog alertDialog = builder.create();
    alertDialog.show();
}

private void getLocation() {
    if (ActivityCompat.checkSelfPermission(

```

```

        getActivity(),           Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
            getActivity(),           Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(getActivity(), new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);
    } else {
        Location locationGPS = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
        if (locationGPS != null) {
            double lat = locationGPS.getLatitude();
            double longi = locationGPS.getLongitude();
            CurrentLatitude = lat;
            CurrentLongitude = longi;
        } else {
            new CommonMethod(getActivity(), "Not Found");
        }
    }
}

```

```

private void recyclerViewSetMethod() {

    Call<GetForemanUserSelectedServicesData> call = apiInterface.GetForemanUserSelectedServicesData(
        sp.getString(SharedPreferencesData.UserID, ""))
    );

    call.enqueue(new Callback<GetForemanUserSelectedServicesData>() {
        @Override
        public void onResponse(Call<GetForemanUserSelectedServicesData> call,
        Response<GetForemanUserSelectedServicesData> response) {

```

```

pd.dismiss();

if(response.code() == 200){

    if(response.body().status == true){

        new CommonMethod(getActivity(), response.body().message);

        arrayList = new ArrayList<>();
        GetForemanUserSelectedServicesData data = response.body();

        for(int i=0; i < data.response.size(); i++){

            ForemanUserSelectedServicesList list = new
            ForemanUserSelectedServicesList();

            list.setCartID(data.response.get(i).cartID);
            list.setServiceID(data.response.get(i).serviceID);
            list.setForemanID(data.response.get(i).foremanID);
            list.setUserID(data.response.get(i).userID);
            list.setVehicleID(data.response.get(i).vehicleID);

            list.setProblemDescriptionMessage(data.response.get(i).problemDescriptionMessage + "");
            list.setSPReqMoney(data.response.get(i).sPReqMoney);
            list.setUserLocation(data.response.get(i).userLocation);
            list.setPaymentStatus(data.response.get(i).paymentStatus);
            list.setTypeOfProblem(data.response.get(i).typeOfProblem);
            list.setProblemSubType(data.response.get(i).problemSubType);
            list.setServiceFixedCharge(data.response.get(i).serviceFixedCharge);
            list.setUserFirstName(data.response.get(i).firstName);
            list.setUserLastName(data.response.get(i).lastName);
            list.setUserProfileImage(data.response.get(i).profileImage);
        }
    }
}

```

```

        list.setUserMobileNumber(data.response.get(i).mobileNumber);
        list.setNumberPlate_number(data.response.get(i).numberPlateNumber);
        list.setTypeOfVehicle(data.response.get(i).typeOfVehicle);
        list.setVehicleModelName(data.response.get(i).vehiclemodelName);
        list.setVehicle_Colour(data.response.get(i).vehicleColour);

        arrayList.add(list);
    }

    ForemanUserSelectedServicesAdapter adapter = new
    ForemanUserSelectedServicesAdapter(getActivity(), arrayList);
    recyclerView.setAdapter(adapter);
    adapter.notifyDataSetChanged();

} else {
    new CommonMethod(getActivity(), response.body().message());
}

} else {
    new CommonMethod(getActivity(), "Server Error Code : "+response.code());
}

}

@Override
public void onFailure(Call<GetForemanUserSelectedServicesData> call, Throwable t) {

    pd.dismiss();
    new CommonMethod(getActivity(), t.getMessage());
}

}

```

```
});  
  
}  
}
```

User password Activity:

```
package com.brainybeam.roadsideassistance.User.Activity;  
  
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.app.AlertDialog;  
import android.content.SharedPreferences;  
import android.graphics.Color;  
import android.graphics.drawable.ColorDrawable;  
import android.os.Bundle;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
  
import com.brainybeam.roadsideassistance.Login.LoginActivity;  
import com.brainybeam.roadsideassistance.R;  
import com.brainybeam.roadsideassistance.RetrofitData.DeleteUserORForemanData;  
import com.brainybeam.roadsideassistance.Utils.ApiClient;  
import com.brainybeam.roadsideassistance.Utils.ApiInterface;  
import com.brainybeam.roadsideassistance.Utils.CommonMethod;  
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;  
  
import retrofit2.Call;  
import retrofit2.Callback;
```

```
import retrofit2.Response;

public class UserForgotPasswordActivity extends AppCompatActivity {

    EditText Password, ReEnterPassword;
    Button login;

    String sPassword, sPassword2;

    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_user_forgot_password);
        getSupportActionBar().setTitle("Forgot Password");
        ColorDrawable colorDrawable
                = new ColorDrawable(Color.parseColor("#2196F3"));
        getSupportActionBar().setBackgroundDrawable(colorDrawable);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        apiInterface = ApiClient.getClient().create(ApiInterface.class);
        sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

        Password = findViewById(R.id.user_forgot_password_PasswordEditText);
        ReEnterPassword = findViewById(R.id.user_forgot_password_ReEnterPasswordEditText);
        login = findViewById(R.id.user_forgot_password_LoginButton);
```

```

login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        sPassword = Password.getText().toString();
        sPassword2 = ReEnterPassword.getText().toString();

        if(sPassword.isEmpty() || sPassword.equalsIgnoreCase("")){
            Password.setError("Password is Required");
        } else if(sPassword2.isEmpty() || sPassword2.equalsIgnoreCase("")){
            ReEnterPassword.setError("Password is Required");
        } else if(sPassword.length()<8){
            Password.setError("Password Must be 8 char long");
        } else if(!sPassword.equals(sPassword2)){
            ReEnterPassword.setError("Password is Not Match");
        } else {

            pd = new ProgressDialog(UserForgotPasswordActivity.this);
            pd.setTitle("Password Resetting...");
            pd.setCancelable(false);
            pd.show();
            PasswordUpdateData();

        }
    }
});

}

private void PasswordUpdateData() {

    Call<DeleteUserORForemanData> call = apiInterface.UserForgotPassword(

```

```

        sp.getString(SharedPreferencesData.UserID, ""),
        sPassword
    );

    call.enqueue(new Callback<DeleteUserORForemanData>() {
        @Override
        public void onResponse(Call<DeleteUserORForemanData> call,
        Response<DeleteUserORForemanData> response) {

            pd.dismiss();
            if(response.code()==200){

                if(response.body().status==true){
                    sp.edit().putString(SharedPreferencesData.Password, sPassword).commit();
                    new CommonMethod(UserForgotPasswordActivity.this, LoginActivity.class);
                    finish();
                } else {
                    new CommonMethod(UserForgotPasswordActivity.this,
                    response.body().message);
                }
            } else {
                new CommonMethod(UserForgotPasswordActivity.this, "Server Error Code :
"+response.code());
            }
        }

        @Override
        public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {
            pd.dismiss();
        }
    });
}

```

```

        new CommonMethod(UserForgotPasswordActivity.this, t.getMessage()));

    }

});

}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}

}

```

User tracking Activity:

```
package com.brainybeam.roadsideassistance.User.Activity;
```

```

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.hardware.Sensor;
import android.hardware.SensorEvent;

```

```
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanTrackingActivity;
import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanUserPaymentActivity;
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.GetForemanLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.GetUserLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.LocationData;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
```

```
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.IOException;
import java.text.DecimalFormat;
import java.util.List;
import java.util.Locale;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class UserTrackingActivity extends AppCompatActivity implements
OnMapReadyCallback,
LocationListener, GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener, SensorEventListener {

    ImageView BackTabLogo, CallingTabLogo;
```

```
Button nextButton, LocationUpdateButton;  
TextView UserAddressView;  
  
private GoogleMap mMap;  
private static final int MY_PERMISSIONS_REQUEST_FINE_LOCATION = 111;  
private LocationListener locationListener;  
private LocationManager locationManager;  
// record the compass picture angle turned  
private float currentDegree = 0f;  
  
// device sensor manager  
private SensorManager mSensorManager;  
  
Location mLastLocation;  
Marker mCurrLocationMarker;  
GoogleApiClient mGoogleApiClient;  
LocationRequest mLocationRequest;  
  
private final long MIN_TIME = 1000; // 1 second  
private final long MIN_DIST = 5; // 5 Meters  
private LatLng latLng;  
  
GPSTracker gpsTracker;  
double latitude, longitude;  
  
ApiInterface apiInterface;  
SharedPreferences sp;  
ProgressDialog pd;  
  
String sUserAddress, sUserLatitude, sUserLongitude;  
String sForemanAddress, sForemanLatitude, sForemanLongitude;
```

```
double foremanLatitude, foremanLongitude;
double CurrentLatitude, CurrentLongitude;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_user_tracking);

    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    nextButton = findViewById(R.id.user_tracking_NextButton);
    BackTabLogo = findViewById(R.id.user_tracking_back_image_Logo);
    CallingTabLogo = findViewById(R.id.user_tracking_call_image_Logo);
    LocationUpdateButton = findViewById(R.id.user_tracking_updateLocationButton);
    UserAddressView = findViewById(R.id.user_tracking_UserAddress);

    nextButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new CommonMethod(UserTrackingActivity.this, UserEndActivity.class);
        }
    });

    BackTabLogo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            onBackPressed();
        }
    });
}
```

```

CallingTabLogo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        AlertDialog.Builder builder = new AlertDialog.Builder(UserTrackingActivity.this);
        builder.setTitle("Call to Foreman?");
        builder.setPositiveButton("CALL", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Intent intent = new Intent(Intent.ACTION_CALL);

                intent.setData(Uri.parse("tel:+880"+sp.getString(SharedPreferencesData.User_ForemanMobileNumber, "")));
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    if      (checkSelfPermission(Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                        // TODO: Consider calling
                        // Activity#requestPermissions
                        // here to request the missing permissions, and then overriding
                        // public void onRequestPermissionsResult(int requestCode, String[]
                        permissions,
                        // int[] grantResults)
                        // to handle the case where the user grants the permission. See the
                        documentation
                        // for Activity#requestPermissions for more details.
                        return;
                    }
                }
                startActivity(intent);
            }
        });
    }
});

```

```

    });

    builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    builder.show();

});

if (ContextCompat.checkSelfPermission(UserTrackingActivity.this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(
        this, // Activity
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        MY_PERMISSIONS_REQUEST_FINE_LOCATION);
}

// TODO Get Foreman Location Data
GetForemanCurrLocationData();
 GetUserLocationData();

gpsTracker = new GPSTracker(UserTrackingActivity.this);

SupportMapFragment mapFragment;

```

```

if (gpsTracker.canGetLocation()) {
    mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.user_tracking_mapFragment);
    mapFragment.getMapAsync(this);
    // mMap.setTrafficEnabled(true);
} else {
    gpsTracker.showSettingsAlert();
}

ActivityCompat.requestPermissions(this,
    new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
    PackageManager.PERMISSION_GRANTED);

ActivityCompat.requestPermissions(this,
    new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
    PackageManager.PERMISSION_GRANTED);

//fetch_GPS();
mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

// Check if GPS enabled
if(gpsTracker.canGetLocation()) {

    CurrentLatitude = gpsTracker.getLatitude();
    CurrentLongitude = gpsTracker.getLongitude();

    // getLocation();
}

} else {
    // Can't get location.
    // GPS or network is not enabled.
    // Ask user to enable GPS/network in settings.
}

```

```

gpsTracker.showSettingsAlert();
}

// TODO Distance

// TODO User Location
Location startPoint = new Location("locationA");
startPoint.setLatitude(CurrentLatitude);
startPoint.setLongitude(CurrentLongitude);

// TODO Foreman Location
Location endPoint = new Location("locationA");
endPoint.setLatitude(foremanLatitude);
endPoint.setLongitude(foremanLongitude);
DecimalFormat precision = new DecimalFormat("0.00");
double distance = Double.parseDouble(precision.format(startPoint.distanceTo(endPoint) /
1000));

if(distance<500){
    nextButton.setVisibility(View.VISIBLE);
    LocationUpdateButton.setVisibility(View.GONE);
}

LocationUpdateButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        GetForemanCurrLocationData();
        StoreUpdatedUserLocationData();
        new CommonMethod(UserTrackingActivity.this, UserTrackingActivity.class);
        finish();
    }
});

```

```

        }

    });

}

private void GetUserLocationData() {

    Call< GetUserLocationData> call = apiInterface.GetUserLocationData(
        sp.getString(SharedPreferencesData.UserID, ""),
        sp.getString(SharedPreferencesData.User_ForemanID, "")
    );

    call.enqueue(new Callback< GetUserLocationData>() {
        @Override
        public void onResponse(Call< GetUserLocationData> call,
        Response< GetUserLocationData> response) {

            if(response.code()==200){

                if(response.body().status==true){

                    new CommonMethod(UserTrackingActivity.this, "Get User");
                    GetUserLocationData data = response.body();
                    for(int i=0; i<data.response.size(); i++){

                        sp.edit().putString(SharedPreferencesData.User_UserLocationID,
                        data.response.get(i).userLocationID).commit();
                        sp.edit().putString(SharedPreferencesData.User_UserCurrLatitude,
                        data.response.get(i).userLatitude).commit();
                        sp.edit().putString(SharedPreferencesData.User_UserCurrLongitude,
                        data.response.get(i).userLongitude).commit();
                }
            }
        }
    });
}

```

```

        }

    } else {
        new CommonMethod(UserTrackingActivity.this, response.body().message());
    }

} else {
    new CommonMethod(UserTrackingActivity.this, "Server Error Code : "+response.code());
}

}

@Override
public void onFailure(Call< GetUserLocationData> call, Throwable t) {
    new CommonMethod(UserTrackingActivity.this, t.getMessage());
}

});

}

private void GetForemanCurrLocationData() {

Call<GetForemanLocationData> call = apiInterface.GetForemanLocationData(
    sp.getString(SharedPreferencesData.User_ForemanID, ""),
    sp.getString(SharedPreferencesData.UserID, "")
);

call.enqueue(new Callback<GetForemanLocationData>() {
    @Override

```

```

public void onResponse(Call<GetForemanLocationData> call,
Response<GetForemanLocationData> response) {

    if(response.code()==200){

        if(response.body().status==true){

            GetForemanLocationData data = response.body();
            for(int i=0; i<data.response.size(); i++){

                new CommonMethod(UserTrackingActivity.this, "GetForeman");

                foremanLatitude = Double.valueOf(data.response.get(i).foremanLatitude);
                foremanLongitude = Double.valueOf(data.response.get(i).foremanLongitude);

                sp.edit().putString(SharedPreferencesData.User_ForemanLocationID,
data.response.get(i).foremanLocationID).commit();
                sp.edit().putString(SharedPreferencesData.User_ForemanCurrLatitude,
data.response.get(i).foremanLatitude).commit();
                sp.edit().putString(SharedPreferencesData.User_ForemanCurrLongitude,
data.response.get(i).foremanLongitude).commit();
            }

        } else {

            new CommonMethod(UserTrackingActivity.this, response.body().message);
        }

    } else {

        new CommonMethod(UserTrackingActivity.this, "Server Error Code : "+response.code());
    }
}

```

```
    }

    @Override
    public void onFailure(Call<GetForemanLocationData> call, Throwable t) {
        new CommonMethod(UserTrackingActivity.this, t.getMessage());
    }
);

@Override
protected void onResume() {
    super.onResume();

    // for the system's orientation sensor registered listeners
    mSensorManager.registerListener(UserTrackingActivity.this,
        mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
        SensorManager.SENSOR_DELAY_GAME);
}

@Override
protected void onPause() {
    super.onPause();

    // to stop the listener and save battery
    mSensorManager.unregisterListener(this);
}

@Override
public void onSensorChanged(SensorEvent event) {

    // get the angle around the z-axis rotated
    float degree = Math.round(event.values[0]);
```

```
// create a rotation animation (reverse turn degree degrees)
RotateAnimation ra = new RotateAnimation(
    currentDegree,
    -degree,
    Animation.RELATIVE_TO_SELF, 0.5f,
    Animation.RELATIVE_TO_SELF,
    0.5f);
```

```
// how long the animation will take place
ra.setDuration(210);
```

```
// set the animation after the end of the reservation status
ra.setFillAfter(true);
```

```
// Start the animation
currentDegree = -degree;
```

```
}
```

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // not in use
}
```

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
```

```

if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }
} else {
    buildGoogleApiClient();
    //mMap.setMyLocationEnabled(true);
}

//    // Add a marker in Sydney and move the camera
//
//    gpsTracker = new GPSTracker(UserTrackingActivity.this);
//
//    latitude = gpsTracker.getLatitude();
//    longitude = gpsTracker.getLongitude();
//
//    LatLng sydney = new LatLng(latitude, longitude);
//    mMap.addMarker(new MarkerOptions().position(sydney).title("Sydney"));
//    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
//    locationListener = new LocationListener() {
//
//        @Override
//        public void onLocationChanged(Location location) {
//
//            try {
//                LatLng = new LatLng(23.0225, 72.5714);
//                mMap.addMarker(new MarkerOptions().position(LatLng).title("My Position"));
//                //
//                mMap.moveCamera(CameraUpdateFactory.newLatLng(LatLng));
//            }
//        }
//    }
}

```

```

//      }
//      catch (SecurityException e){
//          e.printStackTrace();
//      }
//
//  }
//
//  };
//
//  locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
//
//  try {
//      locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
// MIN_TIME, MIN_DIST, locationListener);
//  }
//  catch (SecurityException e){
//      e.printStackTrace();
//  }
}

}

```

```

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API).build();
    mGoogleApiClient.connect();
}

```

```

@Override
public void onPointerCaptureChanged(boolean hasCapture) {
    super.onPointerCaptureChanged(hasCapture);
}

```

```
    }

    @Override
    public void onConnected(@Nullable Bundle bundle) {

        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURA
CY);

        if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, UserTrackingActivity.this);
        }
    }

    @Override
    public void onConnectionSuspended(int i) {

    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {

    }
}
```

```
@Override
public void onLocationChanged(@NonNull Location location) {

    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }

    //Place current location marker
    final LatLng[] latLng = {new LatLng(location.getLatitude(), location.getLongitude())};
    final MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(latLng[0]);
    markerOptions.title("Your Current Position");
    markerOptions.draggable(true);

    markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.icprofile));

    mCurrLocationMarker = mMap.addMarker(markerOptions);
    mMap.setMapType(mMap.MAP_TYPE_SATELLITE);

    Geocoder geocoder = new Geocoder(UserTrackingActivity.this, Locale.getDefault());
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(),
1);
        String add = addresses.get(0).getAddressLine(0);
        String city = addresses.get(0).getLocality();
        String state = addresses.get(0).getAdminArea();
        String zip = addresses.get(0).getPostalCode();
        String country = addresses.get(0).getCountryName();
```

```

UserAddressView.setText(add);

sUserLatitude = String.valueOf(location.getLatitude());
sUserLongitude = String.valueOf(location.getLongitude());
sUserAddress = add;
// StoreUpdatedUserLocationData();

} catch (IOException e) {
    e.printStackTrace();
}

//move map camera
mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng[0]));
mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
//mMap.setTrafficEnabled(true);
//stop location updates
if (mGoogleApiClient != null) {
    LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

mMap.setOnMyLocationChangeListener(new GoogleMap.OnMyLocationChangeListener()
{
    @Override
    public void onMyLocationChange(Location location) {
        if (location == null)
            return;

        mCurrLocationMarker = mMap.addMarker(new MarkerOptions()
            .flat(true)
            .title("Foreman Location")
            .icon(BitmapDescriptorFactory

```

```

        .fromResource(R.drawable.icforeman))
        .anchor(0.5f, 1f)
        .position(new LatLng(foremanLatitude, foremanLongitude)));
    }

});

private void StoreUpdatedUserLocationData() {

    Call<LocationData> call = apiInterface.UpdateUserLocationData(
        sp.getString(SharedPreferencesData.User_UserLocationID, ""),
        sUserLatitude,
        sUserLongitude
    );

    call.enqueue(new Callback<LocationData>() {
        @Override
        public void onResponse(Call<LocationData> call, Response<LocationData> response) {
            if(response.code()==200){

                if(response.body().status==true){
                    new CommonMethod(UserTrackingActivity.this, "Update User");
                    // new CommonMethod(ForemanTrackingActivity.this,
                    response.body().message);
                } else {
                    new CommonMethod(UserTrackingActivity.this, response.body().message);
                }
            }
        }
    });
}

```

```

        } else {
            new CommonMethod(UserTrackingActivity.this, "Server Error Code : "
                    "+response.code());
        }

    }

@Override
public void onFailure(Call<LocationData> call, Throwable t) {
    new CommonMethod(UserTrackingActivity.this, t.getMessage());
}

});

}
}

```

User Map Fragment:

```
package com.brainybeam.roadsideassistance.User.Activity;
```

```

import android.Manifest;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Address;

```

```
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;

import androidx.cardview.widget.CardView;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.RecyclerView;
import androidx.recyclerview.widget.StaggeredGridLayoutManager;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanTrackingActivity;
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.R;
```

```
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
```

```
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;
```

```
public class UserMapFragment extends Fragment implements OnMapReadyCallback {

    private GoogleMap mMap;

    private LocationListener locationListener;
    private LocationManager locationManager;

    private final long MIN_TIME = 1000; // 1 second
```

```

private final long MIN_DIST = 5; // 5 Meters

private LatLng latLng;
GPSTracker gpsTracker;
double latitude, longitude;

public UserMapFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_user_map, container, false);

    SupportMapFragment mapFragment = (SupportMapFragment)
        getActivity().getSupportFragmentManager()
            .findFragmentById(R.id.frag_user_tracking_map);

    mapFragment.getMapAsync(this);

    ActivityCompat.requestPermissions(getActivity(),
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        PackageManager.PERMISSION_GRANTED);
    ActivityCompat.requestPermissions(getActivity(),
        new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
        PackageManager.PERMISSION_GRANTED);
}

return view;

```

```

    }

@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;

    // if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    //     if (ContextCompat.checkSelfPermission(this,
    //             Manifest.permission.ACCESS_FINE_LOCATION)
    //         == PackageManager.PERMISSION_GRANTED) {
    //         buildGoogleApiClient();
    //         //mMap.setMyLocationEnabled(true);
    //     }
    // } else {
    //     buildGoogleApiClient();
    //     //mMap.setMyLocationEnabled(true);
    // }

    // Add a marker in Sydney and move the camera

    gpsTracker = new GPSTracker(getActivity());

    latitude = gpsTracker.getLatitude();
    longitude = gpsTracker.getLongitude();

    LatLng sydney = new LatLng(latitude, longitude);
    mMap.addMarker(new MarkerOptions().position(sydney).title("Sydney"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    locationListener = new LocationListener() {

        @Override
        public void onLocationChanged(Location location) {

```

```

try {
    LatLng = new LatLng(23.0225, 72.5714);
    mMap.addMarker(new MarkerOptions().position(latLng).title("My Position"));

    mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
}
catch (SecurityException e){
    e.printStackTrace();
}

};

locationManager = (LocationManager)
getActivity().getSystemService(getActivity().LOCATION_SERVICE);

try {
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
MIN_TIME, MIN_DIST, locationListener);
}
catch (SecurityException e){
    e.printStackTrace();
}
}
}

```

User profile:

```
package com.brainybeam.roadsideassistance.User.DashBoard;
```

```
import android.Manifest;
import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.BitmapDrawable;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;

import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.contract.ActivityResultContracts;
import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;

import android.util.Base64;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
```

```
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebaseio.firestore.DocumentReference;
import com.google.firebaseio.firestore.FirebaseFirestore;
import com.google.firebaseio.storage.FileDownloadTask;
import com.google.firebaseio.storage.FirebaseStorage;
import com.google.firebaseio.storage.StorageReference;
import com.google.firebaseio.storage.UploadTask;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import de.hdodenhof.circleimageview.CircleImageView;
```

```
public class UserProfileFragment extends Fragment {  
  
    LinearLayout layout1, layout2;  
    CircleImageView ProfileImage1, ProfileImage2;  
    TextView FirstName, LastName, Contact, Email, AccountStatus;  
    EditText FirstName_EditText, LastName_EditText, Contact_EditText, Email_EditText,  
    Password_EditText;  
    Button EditTextButton, UpdateButton, DeactivateButton1, DeactivateButton2;  
  
    String sFirstName, sLastName, sContact, sEmail, sPassword;  
    SharedPreferences sp;  
    Bundle bundle;  
  
    Uri imageUri;  
    StorageReference storageReference;  
    FirebaseApp firebaseApp;  
    private FirebaseAuth mAuth;  
    FirebaseFirestore fStore;  
  
    private final String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\.[a-z]+";  
  
    String[] appPermission = {Manifest.permission.READ_EXTERNAL_STORAGE};  
    private static final int PERMISSION_REQUEST_CODE = 1240;  
  
    public UserProfileFragment() {  
        // Required empty public constructor  
    }  
}
```

```

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for getActivity() fragment
    View view = inflater.inflate(R.layout.fragment_user_profile, container, false);

    firebaseApp = FirebaseApp.initializeApp(requireActivity());
    mAuth = FirebaseAuth.getInstance();
    fStore = FirebaseFirestore.getInstance();

    sp      = requireActivity().getSharedPreferences(SharedPreferencesData.PREF,
Context.MODE_PRIVATE);

    layout1 = view.findViewById(R.id.frag_user_profile_layout1);
    layout2 = view.findViewById(R.id.frag_user_profile_layout2);
    ProfileImage1 = view.findViewById(R.id.frag_user_profile_userProfileImage);
    ProfileImage2 = view.findViewById(R.id.frag_user_profile_userProfileImage2);
    FirstName = view.findViewById(R.id.frag_user_profile_FirstName);
    LastName = view.findViewById(R.id.frag_user_profile_LastName);
    Contact = view.findViewById(R.id.frag_user_profile_MobileNumber);
    Email = view.findViewById(R.id.frag_user_profile_Email);
    AccountStatus = view.findViewById(R.id.frag_user_profile_AccountStatus);
    FirstName_EditText = view.findViewById(R.id.frag_user_profile_FirstNameEditText);
    LastName_EditText = view.findViewById(R.id.frag_user_profile_LastNameEditText);
    Contact_EditText = view.findViewById(R.id.frag_user_profile_MobileNumberEditText);
    Email_EditText = view.findViewById(R.id.frag_user_profile_EmailEditText);
    Password_EditText = view.findViewById(R.id.frag_user_profile_PasswordEditText);
    EditTextButton = view.findViewById(R.id.frag_user_profile_EditProfileButton);
    UpdateButton = view.findViewById(R.id.frag_user_profile_UpdateButton);
    DeactivateButton1 = view.findViewById(R.id.frag_user_profile_deactivateButton);
}

```

```

DeactivateButton2 = view.findViewById(R.id.frag_user_profile_deactivateButton2);

layout2.setVisibility(View.GONE);

// TODO Get Profile Image
GetImageSp();

FirstName.setText(sp.getString(SharedPreferencesData.FirstName, ""));
LastName.setText(sp.getString(SharedPreferencesData.LastName, ""));
Contact.setText(sp.getString(SharedPreferencesData.MobileNumber, ""));
Email.setText(sp.getString(SharedPreferencesData.Email, ""));
AccountStatus.setText(sp.getString(SharedPreferencesData.Account_Status, ""));

EditTextButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        layout2.setVisibility(View.VISIBLE);
        layout1.setVisibility(View.GONE);
    }
});

UpdateButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        sFirstName = FirstName_EditText.getText().toString();
        sLastName = LastName_EditText.getText().toString();
        sContact = Contact_EditText.getText().toString();
        sEmail = Email_EditText.getText().toString();
        sPassword = Password_EditText.getText().toString();

        if (sFirstName.isEmpty()) {

```

```

        FirstName_EditText.setError("FirstName is Required");
    } else if (sLastName.isEmpty()) {
        LastName_EditText.setError("LastName is Required");
    } else if (sContact.length() != 10) {
        Contact_EditText.setError("Mobile number is Required");
    } else if (sEmail.isEmpty()) {
        Email_EditText.setError("Email is Required");
    } else if (!sEmail.matches>EmailPattern)) {
        Email_EditText.setError("Valid Email is Required");
    } else if (sPassword.isEmpty()) {
        Password_EditText.setError("Password is Required");
    } else if (sPassword.length() < 8) {
        Password_EditText.setError("Password must be 8 char long");
    } else {
        UpdateUserPassword();
        UpdateUserProfileData();
        UpdateUserProfileImageData();
    }
}

});
```

```

ProfileImage2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (checkAndRequestPermission()) {
            selectImageMethod(view);
        }
    }
});
```

```

DeactivateButton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        DeactivateUserAccount();
    }
});

DeactivateButton2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        DeactivateUserAccount();
    }
);

return view;
}

// Todo Get Profile Image From Shared Preference
private void GetImageSp() {
    String encodedSaveImage = sp.getString(SharedPreferencesData.ProfileImage, "");
    byte[] decodedString = Base64.decode(encodedSaveImage, Base64.DEFAULT);
    Bitmap decodeBitmap = BitmapFactory.decodeByteArray(decodedString, 0,
decodedString.length);
    if (decodeBitmap != null) {
        ProfileImage1.setImageBitmap(decodeBitmap);
        new CommonMethod(getActivity(), "Image Got from Sp");
    } else {
        getUserImage();
    }
}

```

```

    }

// Todo Get Profile Image from Firebase

private void getUserImage() {
    storageReference = FirebaseStorage.getInstance().getReference("images/userprofile/" +
sp.getString(SharedPreferencesData.UserID, "") + ".jpg");

    try {
        File localfile = File.createTempFile("tempfile", ".jpg");
        storageReference.getFile(localfile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
            @Override
            public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {
                new CommonMethod(getActivity(), "Image Downloaded");
                Bitmap bitmap1 = BitmapFactory.decodeFile(localfile.getAbsolutePath());
                ProfileImage1.setImageBitmap(bitmap1);

                SaveImageSp();// Save Image to Sp
            }
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                new CommonMethod(getActivity(), "Image Not Downloaded");
                ProfileImage1.setImageResource(R.drawable.ic_profile);//default image
            }
        });

    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}

```

```

// Todo Save Image to Shared Preference
private void SaveImageSp() {
    BitmapDrawable bitmapDrawable = (BitmapDrawable) ProfileImage1.getDrawable();
    Bitmap bitmap = bitmapDrawable.getBitmap();

    ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
    bitmap.compress(Bitmap.CompressFormat.PNG, 100, outputStream);
    byte[] byteArray = outputStream.toByteArray();
    String encodedImage = Base64.encodeToString(byteArray, Base64.DEFAULT);

    sp.edit().putString(SharedPreferencesData.ProfileImage, encodedImage).apply();

    new CommonMethod(getActivity(), "Image saved to shared preference");
}

// Todo Update Profile Image
private void UpdateUserProfileImageData() {
    // Todo Delete Old Image
    storageReference = FirebaseStorage.getInstance().getReference("images/userprofile/" +
        sp.getString(SharedPreferencesData.UserID, "") + ".jpg");
    storageReference.delete().addOnSuccessListener(new OnSuccessListener<Void>() {
        @Override
        public void onSuccess(Void unused) {
            new CommonMethod(getActivity(), "Image Deleted");
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            new CommonMethod(getActivity(), "Image Not Deleted");
        }
    });
}

```

```

    });

    // Todo Upload New Image
    storageReference = FirebaseStorage.getInstance().getReference("images/userprofile/" +
sp.getString(SharedPreferencesData.UserID, "") + ".jpg");
    storageReference.putFile(imageUri).addOnCompleteListener(new
OnCompleteListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onComplete(@NonNull Task<UploadTask.TaskSnapshot> task) {
        new CommonMethod(getActivity(), "Image Uploaded");
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        new CommonMethod(getActivity(), "Image Not Uploaded");
    }
});

// Todo Deactivate User Account
private void DeactivateUserAccount() {

}

// Todo Update Password
private void UpdateUserPassword() {
    Objects.requireNonNull(mAuth.getCurrentUser()).updatePassword(sPassword);
    new CommonMethod(getActivity(), "Password Updated");
}

// Todo Update User Profile
private void UpdateUserProfileData() {
}

```

```

sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).apply();
sp.edit().putString(SharedPreferencesData.LastName, sLastName).apply();
sp.edit().putString(SharedPreferencesData.MobileNumber, sContact).apply();
sp.edit().putString(SharedPreferencesData.Email, sEmail).apply();
sp.edit().putString(SharedPreferencesData.Password, sPassword).apply();
String userID = Objects.requireNonNull(mAuth.getCurrentUser()).getUid();
DocumentReference documentReference = fStore.collection("Users").document(userID);
new CommonMethod(getActivity(), "User ID" + userID);
Map<String, Object> user = new HashMap<>();
user.put("FirstName", sFirstName);
user.put("LastName", sLastName);
user.put("MobileNumber", sContact);
user.put("Email", sEmail);
user.put("Password", sPassword);
Boolean value = true;
user.put("Active_Status", value);
documentReference.update(user).addOnSuccessListener(new OnSuccessListener<Void>() {
    @Override
    public void onSuccess(Void unused) {
        new CommonMethod(getActivity(), "Information Updated");
    }
}).addOnFailureListener(new OnFailureListener() {
    @Override
    public void onFailure(@NonNull Exception e) {
        new CommonMethod(getActivity(), "Information Not Updated");
    }
});
}

```

```

private void otpSendToMobile(String sPhone) {

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks = new
    PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

            }

        @Override
        public void onVerificationFailed(FirebaseException e) {
            new CommonMethod(getActivity(), e.getLocalizedMessage());
        }

        @Override
        public void onCodeSent(@NonNull String VerificationId,
                              @NonNull PhoneAuthProvider.ForceResendingToken token) {

            new CommonMethod(getActivity(), "OTP is successFully Send");
            bundle.putString("PhoneNumber", sPhone.trim());
            bundle.putString("Mobile_VerificationID", VerificationId);
            Intent intent = new Intent(getActivity(), OTPVerificationActivity.class);
            intent.putExtras(bundle);
            startActivity(intent);
        }
    };
}

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())

```

```

        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(requireActivity())
        .setCallbacks(mCallbacks)
        .build();

    PhoneAuthProvider.verifyPhoneNumber(options);

}

public void selectImageMethod(View view) {
    Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
    photoPickerIntent.setType("image/*");
    someActivityResultLauncher.launch(photoPickerIntent);
}

ActivityResultLauncher<Intent> someActivityResultLauncher = registerForActivityResult(
    new ActivityResultContracts.StartActivityForResult(),
    result -> {
        if (result.getResultCode() == Activity.RESULT_OK) {
            // There are no request codes
            // doSomeOperations();
            Intent data = result.getData();
            imageUri = Objects.requireNonNull(data).getData();
            Log.d("RESPONSE_URI", String.valueOf(imageUri));
            ProfileImage2.setImageURI(imageUri);
        }
    });
}

```

```

public boolean checkAndRequestPermission() {
    List<String> listPermission = new ArrayList<>();
    for (String perm : appPermission) {

```

```

        if      (ContextCompat.checkSelfPermission(requireActivity(),      perm)      !=
PackageManager.PERMISSION_GRANTED) {
            listPermission.add(perm);
        }
    }

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
        if      (ContextCompat.checkSelfPermission(requireActivity(),
Manifest.permission.READ_MEDIA_IMAGES)      !=
PackageManager.PERMISSION_GRANTED) {
            listPermission.add(Manifest.permission.READ_MEDIA_IMAGES);
        }
    }

    if (listPermission.size() >= 2) {
        ActivityCompat.requestPermissions(requireActivity(),      listPermission.toArray(new
String[0]), PERMISSION_REQUEST_CODE);
        return false;
    }
    return true;
}

```

User Home:

```

package com.brainybeam.roadsideassistance.User.DashBoard;

import android.app.AlertDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;

```

```
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.LinearLayout;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.User.Activity.UserForemanServicesActivity;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;

import java.util.ArrayList;

public class UserHomeFragment extends Fragment {

    private TextView continueTextButton;

    private LinearLayout accident_layout, battery_layout, clutch_break_problem_layout,
    fuel_problem_layout, lost_lock_Key_layout,
    Towing_layout, Tyre_Damage_layout, Other_layout;

    private LinearLayout fuel_problem_Sub_layout;
```

```

CheckBox outOfOil, outOfPetrol, outOfWrongFuel;
String sfuel_subProblem;

private LinearLayout Tyre_problem_Sub_layout;
RadioGroup TyreSubChoice_radioGroup;
String sTyre_subProblem;

SharedPreferences sp;
ProgressDialog pd;

public UserHomeFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_user_home, container, false);

    sp = getSharedPreferences(SharedPreferencesData.PREF,
        Context.MODE_PRIVATE);

    // TODO Layout
    accident_layout = view.findViewById(R.id.frag_user_home_accident_layout);
    battery_layout = view.findViewById(R.id.frag_user_home_battery_layout);
    clutch_break_problem_layout = view.findViewById(R.id.frag_user_home_clutch_layout);
    fuel_problem_layout = view.findViewById(R.id.frag_user_home_fuel_layout);
    lost_lock_Key_layout = view.findViewById(R.id.frag_user_home_lost_lock_key_layout);
}

```

```

Towing_layout = view.findViewById(R.id.frag_user_home_towing_layout);
Tyre_Damage_layout = view.findViewById(R.id.frag_user_home_tyre_layout);
Other_layout = view.findViewById(R.id.frag_user_home_other_layout);

// TODO Sub layout
fuel_problem_Sub_layout = view.findViewById(R.id.frag_user_home_fuel_sub_layout);
outOfOil = view.findViewById(R.id.frag_user_home_fuel_diesel);
outOfPetrol = view.findViewById(R.id.frag_user_home_fuel_petrol);
outOfWrongFuel = view.findViewById(R.id.frag_user_home_fuel_wrong);

fuel_problem_Sub_layout.setVisibility(View.GONE);

Tyre_problem_Sub_layout = view.findViewById(R.id.frag_user_home_tyre_sub_layout);
TyreSubChoice_radioGroup = view.findViewById(R.id.frag_user_home_tyre_radioGroup);

Tyre_problem_Sub_layout.setVisibility(View.GONE);

// TODO If User Any Issue Select after User can continue throw this Text Button
continueTextButton = view.findViewById(R.id.frag_user_home_continueTextButton);
continueTextButton.setVisibility(View.GONE);

// TODO Orientation of Device
int orientation = getResources().getConfiguration().orientation;
if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
    // In landscape
    RelativeLayout.LayoutParams layoutParams =
        (RelativeLayout.LayoutParams) continueTextButton.getLayoutParams();
    layoutParams.addRule(RelativeLayout.CENTER_IN_PARENT,
        RelativeLayout.TRUE);
}

```

```

        layoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT,
RelativeLayout.TRUE);
//    continueTextButton.setMaxHeight(200);
//    continueTextButton.setMaxWidth(200);
    continueTextButton.setLayoutParams(layoutParams);
}

// TODO Accident occur after vehicle issue
accident_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Accident
Situation").commit();
        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
        fuel_problem_Sub_layout.setVisibility(View.GONE);
        Tyre_problem_Sub_layout.setVisibility(View.GONE);

accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selecte
d_home_layout));

battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselect
ed_home_layout));

clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_b
order_notselected_home_layout));

```

```
fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no
tselected_home_layout));
```

```
lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n
otselected_home_layout));
```

```
Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselec
ted_home_layout));
```

```
Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n
otselected_home_layout));
```

```
Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselecte
d_home_layout));
```

```
        continueTextButton.setVisibility(View.VISIBLE);
    }
});

// TODO Battery Jump Start, Starter issue
battery_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Battery
Discharged/Not Working").commit();
        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
        fuel_problem_Sub_layout.setVisibility(View.GONE);
        Tyre_problem_Sub_layout.setVisibility(View.GONE);
    }
});
```

```
accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected_home_layout));

clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no tselected_home_layout));

lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n otselected_home_layout));

Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselec ted_home_layout));

Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n otselected_home_layout));

Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselecte d_home_layout));

    continueTextButton.setVisibility(View.VISIBLE);

}

});

// TODO Clutch/Break Problem
```

```
clutch_break_problem_layout.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
  
        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();  
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Clutch/Break  
Problem").commit();  
        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();  
        fuel_problem_Sub_layout.setVisibility(View.GONE);  
        Tyre_problem_Sub_layout.setVisibility(View.GONE);  
  
        accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));  
  
        battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));  
  
        clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected_home_layout));  
  
        fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no  
tselected_home_layout));  
  
        lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n  
otselected_home_layout));  
  
        Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselec  
ted_home_layout));
```

```
Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
    continueTextButton.setVisibility(View.VISIBLE);  
}  
});
```

```
// TODO Fuel Problem
```

```
fuel_problem_layout.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();  
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Fuel  
Problem").commit();
```

```
        fuel_problem_Sub_layout.setVisibility(View.VISIBLE);  
        StringBuilder stringBuilder = new StringBuilder();  
        ArrayList<String> arrayList = new ArrayList<>();  
        int x = 0;  
  
        if (outOfOil.isChecked()) {  
            arrayList.add(x, outOfOil.getText().toString());  
            x++;  
        }  
        //    else {  
        //        arrayList.remove(x);
```

```

//           x--;
//;      }

if (outOfPetrol.isChecked()) {
    arrayList.add(x, outOfPetrol.getText().toString());
    x++;
}

//      else {

//          arrayList.remove(x);
//          x--;
//      }

if (outOfWrongFuel.isChecked()) {
    arrayList.add(x, outOfWrongFuel.getText().toString());
    x++;
}

//      else {

//          arrayList.remove(x);
//          x--;
//      }

for (int i = 0; i < arrayList.size(); i++) {
    stringBuilder.append(arrayList.get(i));
}

sfuel_subProblem = stringBuilder.toString();
sp.edit().putString(SharedPreferencesData.User_ProblemSubType,
sfuel_subProblem).commit();

Tyre_problem_Sub_layout.setVisibility(View.GONE);

```

```
accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected_home_layout));

lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

    continueTextButton.setVisibility(View.VISIBLE);

}

});

// TODO Lost/Locked Key, Car issue
lost_lock_Key_layout.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
    public void onClick(View view) {

        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem,           "Lost/Lock
Key").commit();

        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
        fuel_problem_Sub_layout.setVisibility(View.GONE);
        Tyre_problem_Sub_layout.setVisibility(View.GONE);

        accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

        battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

        clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_b
order_notselected_home_layout));

        fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no
tselected_home_layout));

        lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_s
elected_home_layout));

        Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselec
ted_home_layout));

        Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n
otselected_home_layout));
```

```
Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

        continueTextButton.setVisibility(View.VISIBLE);
    }
});

// TODO Towing issue if Big issue
Towing_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem,
        "Towing").commit();
        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
        fuel_problem_Sub_layout.setVisibility(View.GONE);
        Tyre_problem_Sub_layout.setVisibility(View.GONE);

accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));

clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no
tselected_home_layout));

lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n
otselected_home_layout);

Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected
_home_layout));

Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_n
otselected_home_layout));

Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselecte
d_home_layout));

        continueTextButton.setVisibility(View.VISIBLE);
    }
});

// TODO Tyre Damage issue if Puncture tyre or destroyed tyre changes
Tyre_Damage_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Tyre
Problem").commit();
        fuel_problem_Sub_layout.setVisibility(View.GONE);
    }
});
```

```
TyreSubChoice_radioGroup.setOnCheckedChangeListener(new  
RadioGroup.OnCheckedChangeListener() {  
    @Override  
    public void onCheckedChanged(RadioGroup radioGroup, int i) {  
        RadioButton radioButton = view.findViewById(i);  
        sTyre_subProblem = radioButton.getText().toString();  
    }  
});
```

```
sp.edit().putString(SharedPreferencesData.User_ProblemSubType,  
sTyre_subProblem).commit();
```

```
Tyre_problem_Sub_layout.setVisibility(View.VISIBLE);
```

```
accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected_home_layout));
```

```
Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
    continueTextButton.setVisibility(View.VISIBLE);  
}  
});
```

```
// TODO Other issue
```

```
Other_layout.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        sp.edit().putString(SharedPreferencesData.UserSelectionCount, "Yes").commit();
```

```
        sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "Other  
issue").commit();
```

```
        sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
```

```
        fuel_problem_Sub_layout.setVisibility(View.GONE);
```

```
        Tyre_problem_Sub_layout.setVisibility(View.GONE);
```

```
accident_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
battery_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
clutch_break_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_notselected_home_layout));
```

```
fuel_problem_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no_tselected_home_layout));
```

```
lost_lock_Key_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no_ntselected_home_layout));
```

```
Towing_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no_tselected_home_layout));
```

```
Tyre_Damage_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_no_ntselected_home_layout));
```

```
Other_layout.setBackground(getResources().getDrawable(R.drawable.custom_border_selected_home_layout));
```

```
        continueTextButton.setVisibility(View.VISIBLE);  
    }  
});
```

```
continueTextButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        continueTextButton.setVisibility(View.GONE);  
  
        if (sp.getString(SharedPreferencesData.UserSelectionCount, "").equals("Yes")) {
```

```

        if (sp.getString(SharedPreferencesData.User_TypeOfProblem, "").equals("Fuel
Problem")) {

            if (sfuel_subProblem.isEmpty() || sfuel_subProblem.equalsIgnoreCase("")) {
                new CommonMethod(getActivity(), "Please Select Fuel Problem Type");
            } else {
                new CommonMethod(getApplicationContext(), UserForemanServicesActivity.class);
            }
        } else if (sp.getString(SharedPreferencesData.User_TypeOfProblem,
"").equals("Tyre Problem")) {
            if (TyreSubChoice_radioGroup.getCheckedRadioButtonId() == -1) {
                new CommonMethod(getActivity(), "");
                Toast.makeText(getActivity(), "Please Select Tyre Damage Type",
Toast.LENGTH_SHORT).show();
            } else {

//                sp.edit().putString(SharedPreferencesData.User_ServiceID, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_ForemanFirstName,
"").commit();
//                sp.edit().putString(SharedPreferencesData.User_ForemanLastName, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_ForemanProfileImage,
"").commit();

//                sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_FixCharge, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_ForemanID, "").commit();
//                sp.edit().putString(SharedPreferencesData.User_ForemanMobileNumber,
"").commit();
//                sp.edit().putString(SharedPreferencesData.User_SPReqMoney, "").commit();

```

```

        new CommonMethod(getContext(), UserForemanServicesActivity.class);
    }
} else {

    // sp.edit().putString(SharedPreferencesData.User_ServiceID, "").commit();
    // sp.edit().putString(SharedPreferencesData.User_ForemanFirstName, "").commit();
    // sp.edit().putString(SharedPreferencesData.User_ForemanLastName, "").commit();
    //         sp.edit().putString(SharedPreferencesData.User_ForemanProfileImage,
    // "").commit();

    //
    // sp.edit().putString(SharedPreferencesData.User_TypeOfProblem, "").commit();
    // sp.edit().putString(SharedPreferencesData.User_ProblemSubType, "").commit();
    // sp.edit().putString(SharedPreferencesData.User_FixCharge, "").commit();
    // sp.edit().putString(SharedPreferencesData.User_ForemanID, "").commit();
    //         sp.edit().putString(SharedPreferencesData.User_ForemanMobileNumber,
    // "").commit();

    //         sp.edit().putString(SharedPreferencesData.User_SPReqMoney, "").commit();

    new CommonMethod(getContext(), UserForemanServicesActivity.class);
}
} else {
    new CommonMethod(getActivity(), "Please Select Any One Option");
}

}

});

return view;
}
}

```

Login activity:

```
package com.brainybeam.roadsideassistance.Login;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SwitchCompat;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.Configuration;
import android.os.Bundle;
import android.text.InputType;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Admin.AdminCredential.AdminCredential;
import com.brainybeam.roadsideassistance.Admin.DashBoard.AdminDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanForgotPasswordActivity;
import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.Foreman.Signup.ForemanSignupActivity;
```

```
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.Notification.PushNotification.Config;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.UpdateFCMIDData;
import com.brainybeam.roadsideassistance.User.Activity.UserForgotPasswordActivity;
import com.brainybeam.roadsideassistance.User.DashBoard.UserDashboardActivity;
import com.brainybeam.roadsideassistance.User.Signup.UserSignupActivity;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.google.firebaseio.firestore.DocumentReference;
import com.google.firebaseio.firestore.DocumentSnapshot;
import com.google.firebaseio.firestore.EventListener;
import com.google.firebaseio.firestore.FirebaseFirestore;
import com.google.firebaseio.firestore.FirebaseFirestoreException;
import com.google.firebaseio.messaging.FirebaseMessaging;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
```

```
import java.util.Objects;
import java.util.concurrent.TimeUnit;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class LoginActivity extends AppCompatActivity {

    private EditText EmailORMobile, Password;
    private TextView ForgotPassword;
    SwitchCompat emailPhoneLoginSwitch;

    private ArrayList<String> AdminEmail;
    private ArrayList<String> AdminPassword;

    private final String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\\".+[a-z]+";

    String sEmailORMobile, sPassword, Mobile_VerificationID = null;

    Sharedpreferences sp;
    GPSTracker gpsTracker;
    //ApiInterface apiInterface;
    FirebaseFirestore fStore;
    private FirebaseAuth mAuth;

    // TODO FCMID For Notification
    BroadcastReceiver broadcastReceiver;

    @Override
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    int or = getResources().getConfiguration().orientation;
    if (or == Configuration.ORIENTATION_LANDSCAPE) {
        Objects.requireNonNull(getSupportActionBar()).hide();
    }

    gpsTracker = new GPSTracker(LoginActivity.this);
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);
    fStore = FirebaseFirestore.getInstance();
    mAuth = FirebaseAuth.getInstance();

    AdminEmail = new ArrayList<>();
    AdminPassword = new ArrayList<>();

    List<String> list1 = Arrays.asList(AdminCredential.AdminEmail);
    List<String> list2 = Arrays.asList(AdminCredential.AdminPassword);
    AdminEmail.addAll(list1);
    AdminPassword.addAll(list2);

    EmailORMobile = findViewById(R.id.login_EmailORMobile);
    EmailORMobile.setHint(R.string.email);//value

    Password = findViewById(R.id.login_Password);
    Password.setHint(R.string.password);//value
    ForgotPassword = findViewById(R.id.login_forgotPassword);
    Button login = findViewById(R.id.login_loginButton);
    Button getotp = findViewById(R.id.login_getotp);
```

```

getotp.setVisibility(View.GONE);
TextView signup = findViewById(R.id.login_signupButton);
emailPhoneLoginSwitch = findViewById(R.id.EmailPhoneLoginSwitch);

// Check if GPS enabled
if (!gpsTracker.canGetLocation()) {
    gpsTracker.showSettingsAlert();
}

login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        sEmailORMobile = EmailORMobile.getText().toString();
        sPassword = Password.getText().toString();

        if (sEmailORMobile.isEmpty() || sEmailORMobile.equalsIgnoreCase(" ")) {
            EmailORMobile.setError("Email OR Mobile Number is Required");
        } else if (!sEmailORMobile.matches>EmailPattern) && sEmailORMobile.length() <
10) {
            EmailORMobile.setError("Valid Email OR Mobile Number is Required");
        } else if (sPassword.isEmpty() || sPassword.equalsIgnoreCase(" ")) {
            Password.setError("Password is Required");
        } else if (AdminEmail.contains(sEmailORMobile) &&
AdminPassword.contains(sPassword)) {
            AdminFCMIDStore();
            sp.edit().putString(SharedPreferencesData.AdminLoginState,
"AdminLogin").apply();
            new CommonMethod(LoginActivity.this, AdminDashboardActivity.class);
            finish();
        }
    }
}

```

```

    } else {

        if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
            if (emailPhoneLoginSwitch.isChecked()) {
                Email_loginMethod(sEmailORMobile, sPassword);
            } else {
                Mobile_loginMethod();
            }
        } else {
            new ConnectionDetector(LoginActivity.this).connectiondetect();
        }
    }

});

signup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        AlertDialog.Builder alertDialog = new AlertDialog.Builder(LoginActivity.this);

        alertDialog.setPositiveButton("Signup As User", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, UserSignupActivity.class);
            }
        });
    }
});

```

```
        alertDialog.setNeutralButton("Signup") As Foreman", new
DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        new CommonMethod(LoginActivity.this, ForemanSignupActivity.class);
    }
});
alertDialog.show();

}
});
```

```
emailPhoneLoginSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked) {
            EmailORMobile.setHint(R.string.email);
```

```
EmailORMobile.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);
        Password.setHint(R.string.password);
        Password.setInputType(InputType.TYPE_CLASS_TEXT |
InputType.TYPE_TEXT_VARIATION_PASSWORD);
        EmailORMobile.setText("");
        Password.setText("");
        ForgotPassword.setVisibility(View.VISIBLE);
        getotp.setVisibility(View.GONE);
    } else {
        EmailORMobile.setHint(R.string.phone_number);
        EmailORMobile.setInputType(InputType.TYPE_CLASS_PHONE);
        Password.setHint(R.string.otp);
```

```

        Password.setInputType(InputType.TYPE_CLASS_PHONE);
        EmailORMobile.setText("");
        Password.setText("");
        ForgotPassword.setVisibility(View.GONE);
        getotp.setVisibility(View.VISIBLE);
    }
}

});

getotp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String sPhone = EmailORMobile.getText().toString();
        otpSendToMobile(sPhone);
    }
});

ForgotPassword.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        AlertDialog.Builder alertDialog = new AlertDialog.Builder(LoginActivity.this);

        alertDialog.setPositiveButton("User", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(LoginActivity.this, UserForgotPasswordActivity.class);
            }
        });

        alertDialog.setNeutralButton("Foreman", new DialogInterface.OnClickListener() {

```

```
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        new CommonMethod(LoginActivity.this, ForemanForgotPasswordActivity.class);
    }
});

alertDialog.setCancelable(true);
alertDialog.show();
}
});
}
```

```
private void AdminFCMIDStore() {
```

```
    getAdminToken();
```

```
}
```

```
private void otpSendToMobile(String sPhone) {
```

```
    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks = new
    PhoneAuthProvider.OnVerificationStateChangedCallbacks() {
```

```
        @Override
```

```
        public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {
```

```
}
```

```
        @Override
```

```
        public void onVerificationFailed(FirebaseException e) {
```

```

        new CommonMethod(LoginActivity.this, e.getLocalizedMessage());
    }

    @Override
    public void onCodeSent(@NonNull String VerificationId,
                          @NonNull PhoneAuthProvider.ForceResendingToken token) {
        Mobile_VerificationID = VerificationId;

        new CommonMethod(LoginActivity.this, "OTP is successFully Send");

    }
};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallbacks)
        .build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

// TODO Login Method Start
public void Mobile_loginMethod() {
    if (Mobile_VerificationID == null) {
        new CommonMethod(LoginActivity.this, "OTP is not Send");
        return;
    }
}

```

```

String code = sPassword.trim();
PhoneAuthCredential credential = PhoneAuthProvider.getCredential(Mobile_VerificationID, code);
mAuth.signInWithCredential(credential)
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                LoginMethod(Objects.requireNonNull(mAuth.getCurrentUser()).getUid());
                new CommonMethod(LoginActivity.this, "Successful Login ");
            } else {
                new CommonMethod(LoginActivity.this, "Login Error");
            }
        }
    });
}

public void Email_loginMethod(String sEmailORMobile, String sPassword) {
    mAuth.signInWithEmailAndPassword(sEmailORMobile,
        sPassword).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                LoginMethod(Objects.requireNonNull(mAuth.getCurrentUser()).getUid());
            } else {
                new CommonMethod(LoginActivity.this, "Login Error");
            }
        }
    });
}

```

```

    }

    public void LoginMethod(String userID) {
        sp.edit().putString(SharedPreferencesData.UserID, userID).apply();
        DocumentReference documentReference = fStore.collection("Users").document(userID);
        documentReference.addSnapshotListener(new EventListener<DocumentSnapshot>() {
            @Override
            public void onEvent(@Nullable DocumentSnapshot value, @Nullable FirebaseFirestoreException error) {
                assert value != null;
                sp.edit().putString(SharedPreferencesData.UserType,
                    value.getString("UserType")).apply();
                sp.edit().putString(SharedPreferencesData.FirstName,
                    value.getString("FirstName")).apply();
                sp.edit().putString(SharedPreferencesData.LastName,
                    value.getString("LastName")).apply();
                sp.edit().putString(SharedPreferencesData.MobileNumber,
                    value.getString("MobileNumber")).apply();
                sp.edit().putString(SharedPreferencesData.Email, value.getString("Email")).apply();
                sp.edit().putString(SharedPreferencesData.Password,
                    value.getString("Password")).apply();
                sp.edit().putString(SharedPreferencesData.Account_Status,
                    value.getString("Account_Status")).apply();
                sp.edit().putBoolean(SharedPreferencesData.Active_Status,
                    Boolean.TRUE.equals(value.getBoolean("Active_Status"))).apply();

                // TODO Notification FCMID
                broadcastReceiver = new BroadcastReceiver() {
                    @Override
                    public void onReceive(Context context, Intent intent) {
                        if (Objects.equals(intent.getAction(), Config.REGISTRATION_COMPLETE)) {

```

```

FirebaseMessaging.getInstance().subscribeToTopic(Config.TOPIC_GLOBAL);

        getToken();

    } else if (Objects.equals(intent.getAction(), Config.PUSH_NOTIFICATION)) {

        String s = intent.getStringExtra("message");

    } else {

    }

}

};

if (Objects.requireNonNull(value.getString("UserType")).equalsIgnoreCase("User")) {

    getToken();

    new CommonMethod(LoginActivity.this, UserDashboardActivity.class);

    finish();

} else {

    sp.edit().putString(SharedPreferencesData.ForemanAddress,
value.getString("ForemanAddress")).apply();

    sp.edit().putString(SharedPreferencesData.ForemanArea,
value.getString("ForemanArea")).apply();

    sp.edit().putString(SharedPreferencesData.ForemanCity,
value.getString("ForemanCity")).apply();

    sp.edit().putString(SharedPreferencesData.ForemanState,
value.getString("ForemanState")).apply();

    sp.edit().putBoolean(SharedPreferencesData.ForemanAccount_Status,
Boolean.TRUE.equals(value.getBoolean("ForemanAccount_Status"))).apply();

    getToken();

    new CommonMethod(LoginActivity.this, ForemanDashboardActivity.class);

    finish();

}

}

);

```

```

//                                         sp.edit().putString(SharedPreferencesData.ProfileImage,
data.response.get(i).profileImage).commit();
}

// TODO Login Method End

private void getAdminToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
OnSuccessListener<String>() {
    @Override
    public void onSuccess(String stoken) {
        Log.d("RESPONSE_TOKEN", stoken);
        sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).commit();

        if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
            if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).commit();
            }
        } else {
            new ConnectionDetector(LoginActivity.this).connectiondetect();
        }
    }
});

}

private void getUserToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
OnSuccessListener<String>() {
    @Override
    public void onSuccess(String stoken) {
        Log.d("RESPONSE_TOKEN", stoken);
        sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).apply();
    }
});
}

```

```

        if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
            if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).apply();
            }
        } else {
            new ConnectionDetector(LoginActivity.this).connectiondetect();
        }
    });
}

private void getForemanToken() {
    FirebaseMessaging.getInstance().getToken().addOnSuccessListener(new
    OnSuccessListener<String>() {
        @Override
        public void onSuccess(String stoken) {
            Log.d("RESPONSE_TOKEN", stoken);
            sp.edit().putString(SharedPreferencesData.UserFCMID, stoken).apply();

            if (new ConnectionDetector(LoginActivity.this).isConnectingToInternet()) {
                if (sp.getString(SharedPreferencesData.UserType, "").isEmpty()) {
                    sp.edit().putString(SharedPreferencesData.Admin_FCMID, stoken).apply();
                }
            } else {
                new ConnectionDetector(LoginActivity.this).connectiondetect();
            }
        }
    });
}

```

```
private void addUserFcmIDData() {  
    //    sp.getString(SharedPreferencesData.UserID, ""),  
    //    sp.getString(SharedPreferencesData.UserFCMID, "")  
}
```

```
private void addForemanFcmIDData() {  
    //    sp.getString(SharedPreferencesData.UserID, ""),  
    //    sp.getString(SharedPreferencesData.ForemanFCMID, "")  
}  
}
```

Foreman Tracking Activity:

```
package com.brainybeam.roadsideassistance.Foreman.Activity;
```

```
import android.Manifest;  
import android.app.AlertDialog;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.content.SharedPreferences;  
import android.content.pm.PackageManager;  
import android.hardware.Sensor;  
import android.hardware.SensorEvent;  
import android.hardware.SensorEventListener;  
import android.hardware.SensorManager;  
import android.location.Address;  
import android.location.Geocoder;  
import android.location.Location;
```

```
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.GetForemanLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.GetUserLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.LocationData;
import com.brainybeam.roadsideassistance.User.Activity.UserForemanServicesActivity;
import com.brainybeam.roadsideassistance.User.Activity.UserTrackingActivity;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
import com.google.android.gms.location.LocationListener;
```

```
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.IOException;
import java.text.DecimalFormat;
import java.util.List;
import java.util.Locale;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanTrackingActivity extends AppCompatActivity implements
OnMapReadyCallback,
    LocationListener, GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener, SensorEventListener {

    ImageView BackTabLogo, CallingTabLogo;
    Button nextButton, LocationUpdateButton;
    TextView UserAddressView;

    private GoogleMap mMap;
    private static final int MY_PERMISSIONS_REQUEST_FINE_LOCATION = 111;
```

```
private LocationListener locationListener;
private LocationManager locationManager;
// record the compass picture angle turned
private float currentDegree = 0f;

// device sensor manager
private SensorManager mSensorManager;

Location mLastLocation;
Marker mCurrLocationMarker;
GoogleApiClient mGoogleApiClient;
LocationRequest mLocationRequest;

private final long MIN_TIME = 1000; // 1 second
private final long MIN_DIST = 5; // 5 Meters
private LatLng latLng;

GPSTracker gpsTracker;
double latitude, longitude;

ApiInterface apiInterface;
SharedPreferences sp;
ProgressDialog pd;

String sForemanAddress, sForemanLatitude, sForemanLongitude;
String sUserAddress, sUserLatitude, sUserLongitude;

double userLatitude, userLongitude;

double CurrentLatitude, CurrentLongitude;
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_foreman_tracking);

    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    nextButton = findViewById(R.id.foreman_tracking_NextButton);
    BackTabLogo = findViewById(R.id.foreman_tracking_back_image_Logo);
    CallingTabLogo = findViewById(R.id.foreman_tracking_call_image_Logo);
    LocationUpdateButton = findViewById(R.id.foreman_tracking_updateLocationButton);
    UserAddressView = findViewById(R.id.foreman_tracking_UserAddress);

    // nextButton.setVisibility(View.GONE);
    nextButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            new CommonMethod(ForemanTrackingActivity.this,
                ForemanUserPaymentActivity.class);
        }
    });
}

BackTabLogo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        onBackPressed();
    }
});

```

```

CallingTabLogo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        AlertDialog.Builder builder = new AlertDialog.Builder(ForemanTrackingActivity.this);
        builder.setTitle("Call to User?" +
                "+sp.getString(SharedPreferencesData.Foreman_UserFirstName, ""));
        builder.setPositiveButton("CALL", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Intent intent = new Intent(Intent.ACTION_CALL);

                intent.setData(Uri.parse("tel:+880"+sp.getString(SharedPreferencesData.Foreman_UserMobileNumber, "")));
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    if (checkSelfPermission(Manifest.permission.CALL_PHONE) != PackageManager.PERMISSION_GRANTED) {
                        // TODO: Consider calling
                        // Activity#requestPermissions
                        // here to request the missing permissions, and then overriding
                        // public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                        //                                         int[] grantResults)
                        // to handle the case where the user grants the permission. See the
documentation
                        // for Activity#requestPermissions for more details.
                        return;
                    }
                }
                startActivity(intent);
            }
        });
    }
});

```

```

        }
    });

    builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    builder.show();

});

if (ContextCompat.checkSelfPermission(ForemanTrackingActivity.this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(
        this, // Activity
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        MY_PERMISSIONS_REQUEST_FINE_LOCATION);
}

gpsTracker = new GPSTracker(ForemanTrackingActivity.this);

// TODO Get User Location Data
 GetUserLocationData();
 GetForemanLocationData();

gpsTracker = new GPSTracker(ForemanTrackingActivity.this);

```

```

SupportMapFragment mapFragment;

if (gpsTracker.canGetLocation()) {
    mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.foreman_tracking_mapFragment);
    mapFragment.getMapAsync(this);
    // mMap.setTrafficEnabled(true);
} else {
    gpsTracker.showSettingsAlert();
}

ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
PackageManager.PERMISSION_GRANTED);

ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
PackageManager.PERMISSION_GRANTED);

//fetch_GPS();

mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

// Check if GPS enabled
if(gpsTracker.canGetLocation()) {

    CurrentLatitude = gpsTracker.getLatitude();
    CurrentLongitude = gpsTracker.getLongitude();

    // getLocation();
}

```

```

} else {
    // Can't get location.
    // GPS or network is not enabled.
    // Ask user to enable GPS/network in settings.
    gpsTracker.showSettingsAlert();
}

// TODO Distance

// TODO User Location
Location startPoint = new Location("locationA");
startPoint.setLatitude(CurrentLatitude);
startPoint.setLongitude(CurrentLongitude);

// TODO Foreman Location
Location endPoint = new Location("locationA");
endPoint.setLatitude(userLatitude);
endPoint.setLongitude(userLongitude);
DecimalFormat precision = new DecimalFormat("0.00");
double distance = Double.parseDouble(precision.format(startPoint.distanceTo(endPoint) /
1000));

if(distance<500){
    nextButton.setVisibility(View.VISIBLE);
    LocationUpdateButton.setVisibility(View.GONE);
}

```

```

        LocationUpdateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                GetUserLocationData();
                StoreUpdatedForemanLocationData();
                new CommonMethod(ForemanTrackingActivity.this, ForemanTrackingActivity.class);
                finish();
            }
        });
    }

private void GetForemanLocationData() {

    Call<GetForemanLocationData> call = apiInterface.GetForemanLocationData(
        sp.getString(SharedPreferencesData.UserID, ""),
        sp.getString(SharedPreferencesData.Foreman_UserID, "")
    );

    call.enqueue(new Callback<GetForemanLocationData>() {
        @Override
        public void onResponse(Call<GetForemanLocationData> call,
        Response<GetForemanLocationData> response) {
            if(response.code()==200){

                if(response.body().status==true){

                    new CommonMethod(ForemanTrackingActivity.this, "Get Foreman");
                }
            }
        }
    });
}

```

```

        GetForemanLocationData data = response.body();
        for(int i=0; i<data.response.size(); i++){

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanLocationID,
data.response.get(i).foremanLocationID).commit();

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanCurrLatitude,
data.response.get(i).foremanLatitude).commit();

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanCurrLongitude,
data.response.get(i).foremanLongitude).commit();

        }

    } else {

        new CommonMethod(ForemanTrackingActivity.this, response.body().message);

    }

} else {

    new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "+response.code());

}

}

@Override

public void onFailure(Call<GetForemanLocationData> call, Throwable t) {

    new CommonMethod(ForemanTrackingActivity.this, t.getMessage());

}

});

}

}

```

```

private void GetUserLocationData() {

    Call< GetUserLocationData> call = apiInterface.GetUserLocationData(
        sp.getString(SharedPreferencesData.Foreman_UserID, ""),
        sp.getString(SharedPreferencesData.UserID, "")
    );

    call.enqueue(new Callback< GetUserLocationData>() {
        @Override
        public void onResponse(Call< GetUserLocationData> call,
        Response< GetUserLocationData> response) {

            if(response.code()==200){

                if(response.body().status==true){

                    new CommonMethod(ForemanTrackingActivity.this, "Get User");
                    GetUserLocationData data = response.body();
                    for(int i=0; i<data.response.size(); i++){

                        userLatitude = Double.valueOf(data.response.get(i).userLatitude);
                        userLongitude = Double.valueOf(data.response.get(i).userLongitude);

                        Geocoder geocoder = new Geocoder(ForemanTrackingActivity.this,
                        Locale.getDefault());
                        List<Address> addresses = null;
                        try {
                            addresses =
                            geocoder.getFromLocation(Double.valueOf(data.response.get(i).userLatitude),
                            Double.valueOf(data.response.get(i).userLongitude), 1);
                            String addF = addresses.get(0).getAddressLine(0);
                        }
                    }
                }
            }
        }
    });
}

```

```

        String city = addresses.get(0).getLocality();
        String state = addresses.get(0).getAdminArea();
        String zip = addresses.get(0).getPostalCode();
        String country = addresses.get(0).getCountryName();

        // UserAddressView.setText(add);

        UserAddressView.setText(addF);
        // StoreUpdatedForemanLocationData();
    } catch (IOException e) {
        e.printStackTrace();
    }

    sp.edit().putString(SharedPreferencesData.Foreman_UserLocationID,
data.response.get(i).userLocationID).commit();
    sp.edit().putString(SharedPreferencesData.Foreman_UserCurrLatitude,
data.response.get(i).userLatitude).commit();
    sp.edit().putString(SharedPreferencesData.Foreman_UserCurrLongitude,
data.response.get(i).userLongitude).commit();
}

} else {
    new CommonMethod(ForemanTrackingActivity.this, response.body().message);
}

} else {
    new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "+response.code());
}

```

```
    }

    @Override
    public void onFailure(Call< GetUserLocationData> call, Throwable t) {
        new CommonMethod(ForemanTrackingActivity.this, t.getMessage());
    }
});
```

}

```
    @Override
    protected void onResume() {
        super.onResume();

        // for the system's orientation sensor registered listeners
        mSensorManager.registerListener(ForemanTrackingActivity.this,
            mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
            SensorManager.SENSOR_DELAY_GAME);
    }
}
```

```
    @Override
    protected void onPause() {
        super.onPause();

        // to stop the listener and save battery
        mSensorManager.unregisterListener(this);
    }
}

@Override
public void onSensorChanged(SensorEvent event) {
```

```
// get the angle around the z-axis rotated
float degree = Math.round(event.values[0]);  
  
// create a rotation animation (reverse turn degree degrees)
RotateAnimation ra = new RotateAnimation(
    currentDegree,
    -degree,
    Animation.RELATIVE_TO_SELF, 0.5f,
    Animation.RELATIVE_TO_SELF,
    0.5f);  
  
// how long the animation will take place
ra.setDuration(210);  
  
// set the animation after the end of the reservation status
ra.setFillAfter(true);  
  
// Start the animation
currentDegree = -degree;  
  
}  
  
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // not in use
}  
  
@Override
public void onMapReady(GoogleMap googleMap) {
```

```

mMap = googleMap;

if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        buildGoogleApiClient();
        mMap.setMyLocationEnabled(true);
    }
} else {
    buildGoogleApiClient();
    //mMap.setMyLocationEnabled(true);
}

//    // Add a marker in Sydney and move the camera
//
//    gpsTracker = new GPSTracker(ForemanTrackingActivity.this);
//
//    latitude = gpsTracker.getLatitude();
//    longitude = gpsTracker.getLongitude();
//
//    LatLng sydney = new LatLng(latitude, longitude);
//    mMap.addMarker(new MarkerOptions().position(sydney).title("Sydney"));
//    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
//    locationListener = new LocationListener() {
//
//        @Override
//        public void onLocationChanged(Location location) {
//
//        }
//
//        try {
//            LatLng = new LatLng(23.0225, 72.5714);
//            mMap.addMarker(new MarkerOptions().position(LatLng).title("My Position"));

```

```

// 
//         mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
//     }
//     catch (SecurityException e){
//         e.printStackTrace();
//     }
// 
// }
// 
// );
// 
// locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
// 
// try {
//         locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
// MIN_TIME, MIN_DIST, locationListener);
//     }
//     catch (SecurityException e){
//         e.printStackTrace();
//     }
// }

}

```

```

protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API).build();
    mGoogleApiClient.connect();
}

```

@Override

```
public void onPointerCaptureChanged(boolean hasCapture) {
    super.onPointerCaptureChanged(hasCapture);
}

@Override
public void onConnected(@Nullable Bundle bundle) {

    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(1000);
    mLocationRequest.setFastestInterval(1000);

    mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURA
CY);

    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
mLocationRequest, ForemanTrackingActivity.this);
    }
}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
```

```

}

@Override
public void onLocationChanged(@NonNull Location location) {

    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }

    //Place current location marker
    final LatLng[] latLng = {new LatLng(location.getLatitude(), location.getLongitude())};
    final MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(latLng[0]);
    markerOptions.title("Your Current Position");
    markerOptions.draggable(true);

    markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.icforeman));

    mCurrLocationMarker = mMap.addMarker(markerOptions);
    mMap.setMapType(mMap.MAP_TYPE_SATELLITE);

    Geocoder geocoder = new Geocoder(ForemanTrackingActivity.this, Locale.getDefault());
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(),
1);
        String add = addresses.get(0).getAddressLine(0);
        String city = addresses.get(0).getLocality();
        String state = addresses.get(0).getAdminArea();
        String zip = addresses.get(0).getPostalCode();
    }
}

```

```

String country = addresses.get(0).getCountryName();

// UserAddressView.setText(add);

sForemanLatitude = String.valueOf(location.getLatitude());
sForemanLongitude = String.valueOf(location.getLongitude());
sForemanAddress = add;
// StoreUpdatedForemanLocationData();
} catch (IOException e) {
    e.printStackTrace();
}

//move map camera
mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng[0]));
mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
//mMap.setTrafficEnabled(true);

//stop location updates
if (mGoogleApiClient != null) {
    LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

mMap.setOnMyLocationChangeListener(new GoogleMap.OnMyLocationChangeListener()
{
    @Override
    public void onMyLocationChange(Location location) {
        if (location == null)
            return;

        mCurrLocationMarker = mMap.addMarker(new MarkerOptions()
            .flat(true))
    }
})

```

```

        .title("Incident Location")
        .icon(BitmapDescriptorFactory
            .fromResource(R.drawable.iccar))
        .anchor(0.5f, 1f)
        .position(new LatLng(userLatitude, userLongitude)));
    }

});

}

private void StoreUpdatedForemanLocationData() {

    Call<LocationData> call = apiInterface.UpdateForemanLocationData(
        sp.getString(SharedPreferencesData.Foreman_ForemanLocationID, ""),
        sForemanLatitude,
        sForemanLongitude
    );

    call.enqueue(new Callback<LocationData>() {
        @Override
        public void onResponse(Call<LocationData> call, Response<LocationData> response) {
            if(response.code()==200){

                if(response.body().status==true){
                    new CommonMethod(ForemanTrackingActivity.this, "Update Foreman");
                    // new CommonMethod(ForemanTrackingActivity.this, response.body().message);
                } else {
                    new CommonMethod(ForemanTrackingActivity.this, response.body().message);
                }
            }
        }
    });
}

```

```
    } else {
        new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "+response.code());
    }

}

@Override
public void onFailure(Call<LocationData> call, Throwable t) {
    new CommonMethod(ForemanTrackingActivity.this, t.getMessage());
}

});

}

}

package com.brainybeam.roadsideassistance.Foreman.Activity;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.location.Address;
import android.location.Geocoder;
```

```
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.View;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.GetForemanLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.GetUserLocationData;
import com.brainybeam.roadsideassistance.RetrofitData.LocationData;
import com.brainybeam.roadsideassistance.User.Activity.UserForemanServicesActivity;
import com.brainybeam.roadsideassistance.User.Activity.UserTrackingActivity;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient;
```

```
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import java.io.IOException;
import java.text.DecimalFormat;
import java.util.List;
import java.util.Locale;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanTrackingActivity extends AppCompatActivity implements
OnMapReadyCallback,
LocationListener, GoogleApiClient.ConnectionCallbacks,
GoogleApiClient.OnConnectionFailedListener, SensorEventListener {

    ImageView BackTabLogo, CallingTabLogo;
    Button nextButton, LocationUpdateButton;
    TextView UserAddressView;

    private GoogleMap mMap;
```

```
private static final int MY_PERMISSIONS_REQUEST_FINE_LOCATION = 111;
private LocationListener locationListener;
private LocationManager locationManager;
// record the compass picture angle turned
private float currentDegree = 0f;

// device sensor manager
private SensorManager mSensorManager;

Location mLastLocation;
Marker mCurrLocationMarker;
GoogleApiClient mGoogleApiClient;
LocationRequest mLocationRequest;

private final long MIN_TIME = 1000; // 1 second
private final long MIN_DIST = 5; // 5 Meters
private LatLng latLng;

GPSTracker gpsTracker;
double latitude, longitude;

ApiInterface apiInterface;
SharedPreferences sp;
ProgressDialog pd;

String sForemanAddress, sForemanLatitude, sForemanLongitude;
String sUserAddress, sUserLatitude, sUserLongitude;

double userLatitude, userLongitude;
```

```

        double CurrentLatitude, CurrentLongitude;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_foreman_tracking);

        apiInterface = ApiClient.getClient().create(ApiInterface.class);
        sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

        nextButton = findViewById(R.id.foreman_tracking_NextButton);
        BackTabLogo = findViewById(R.id.foreman_tracking_back_image_Logo);
        CallingTabLogo = findViewById(R.id.foreman_tracking_call_image_Logo);
        LocationUpdateButton = findViewById(R.id.foreman_tracking_updateLocationButton);
        UserAddressView = findViewById(R.id.foreman_tracking_UserAddress);

        // nextButton.setVisibility(View.GONE);
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                new CommonMethod(ForemanTrackingActivity.this,
                    ForemanUserPaymentActivity.class);
            }
        });

        BackTabLogo.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onBackPressed();
            }
        });
    }

```

```

});

CallingTabLogo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        AlertDialog.Builder builder = new AlertDialog.Builder(ForemanTrackingActivity.this);
        builder.setTitle("Call           to           User?"+"
"+sp.getString(SharedPreferencesData.Foreman_UserFirstName, ""));
        builder.setPositiveButton("CALL", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                Intent intent = new Intent(Intent.ACTION_CALL);

                intent.setData(Uri.parse("tel:+880"+sp.getString(SharedPreferencesData.Foreman_UserMobileN
umber, "")));
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    if      (checkSelfPermission(Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
                        // TODO: Consider calling
                        // Activity#requestPermissions
                        // here to request the missing permissions, and then overriding
                        //     public void onRequestPermissionsResult(int requestCode, String[]
permissions,
                        //                                         int[] grantResults)
                        // to handle the case where the user grants the permission. See the
documentation
                        // for Activity#requestPermissions for more details.
                        return;
                    }
                }
            }
        });
    }
});

```

```

        startActivity(intent);
    }
});

builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.dismiss();
    }
});

builder.show();

});

if (ContextCompat.checkSelfPermission(ForemanTrackingActivity.this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(
        this, // Activity
        new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
        MY_PERMISSIONS_REQUEST_FINE_LOCATION);
}

gpsTracker = new GPSTracker(ForemanTrackingActivity.this);

// TODO Get User Location Data
 GetUserLocationData();
 GetForemanLocationData();

```

```

gpsTracker = new GPSTracker(ForemanTrackingActivity.this);

SupportMapFragment mapFragment;

if (gpsTracker.canGetLocation()) {
    mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.foreman_tracking_mapFragment);
    mapFragment.getMapAsync(this);
    // mMap.setTrafficEnabled(true);
} else {
    gpsTracker.showSettingsAlert();
}

ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},
PackageManager.PERMISSION_GRANTED);

ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
PackageManager.PERMISSION_GRANTED);

//fetch_GPS();

mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);

// Check if GPS enabled
if(gpsTracker.canGetLocation()) {

    CurrentLatitude = gpsTracker.getLatitude();
    CurrentLongitude = gpsTracker.getLongitude();
}

```

```

    // getLocation();

} else {
    // Can't get location.
    // GPS or network is not enabled.
    // Ask user to enable GPS/network in settings.
    gpsTracker.showSettingsAlert();
}

// TODO Distance

// TODO User Location
Location startPoint = new Location("locationA");
startPoint.setLatitude(CurrentLatitude);
startPoint.setLongitude(CurrentLongitude);

// TODO Foreman Location
Location endPoint = new Location("locationA");
endPoint.setLatitude(userLatitude);
endPoint.setLongitude(userLongitude);
DecimalFormat precision = new DecimalFormat("0.00");
double distance = Double.parseDouble(precision.format(startPoint.distanceTo(endPoint) /
1000));

```

```

if(distance<500){
    nextButton.setVisibility(View.VISIBLE);
    LocationUpdateButton.setVisibility(View.GONE);
}

```

```

        LocationUpdateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                GetUserLocationData();
                StoreUpdatedForemanLocationData();
                new CommonMethod(ForemanTrackingActivity.this, ForemanTrackingActivity.class);
                finish();
            }
        });
    }

private void GetForemanLocationData() {

    Call<GetForemanLocationData> call = apiInterface.GetForemanLocationData(
        sp.getString(SharedPreferencesData.UserID, ""),
        sp.getString(SharedPreferencesData.Foreman_UserID, "")
    );

    call.enqueue(new Callback<GetForemanLocationData>() {
        @Override
        public void onResponse(Call<GetForemanLocationData> call,
        Response<GetForemanLocationData> response) {
            if(response.code()==200){

                if(response.body().status==true){

```

```

        new CommonMethod(ForemanTrackingActivity.this, "Get Foreman");

        GetForemanLocationData data = response.body();
        for(int i=0; i<data.response.size(); i++){

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanLocationID,
data.response.get(i).foremanLocationID).commit();

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanCurrLatitude,
data.response.get(i).foremanLatitude).commit();

            sp.edit().putString(SharedPreferencesData.Foreman_ForemanCurrLongitude,
data.response.get(i).foremanLongitude).commit();
        }

    } else {

        new CommonMethod(ForemanTrackingActivity.this, response.body().message);
    }

} else {

    new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "
"+response.code());
}

}

@Override
public void onFailure(Call<GetForemanLocationData> call, Throwable t) {
    new CommonMethod(ForemanTrackingActivity.this, t.getMessage());
}

});

}

}

```

```

private void GetUserLocationData() {

    Call< GetUserLocationData> call = apiInterface.GetUserLocationData(
        sp.getString(SharedPreferencesData.Foreman_UserID, ""),
        sp.getString(SharedPreferencesData.UserID, "")

    );

    call.enqueue(new Callback< GetUserLocationData>() {
        @Override
        public void onResponse(Call< GetUserLocationData> call,
        Response< GetUserLocationData> response) {

            if(response.code()==200){

                if(response.body().status==true){

                    new CommonMethod(ForemanTrackingActivity.this, "Get User");
                    GetUserLocationData data = response.body();
                    for(int i=0; i<data.response.size(); i++){

                        userLatitude = Double.valueOf(data.response.get(i).userLatitude);
                        userLongitude = Double.valueOf(data.response.get(i).userLongitude);

                        Geocoder geocoder = new Geocoder(ForemanTrackingActivity.this,
                        Locale.getDefault());
                        List<Address> addresses = null;
                        try {
                            addresses
                            =
                            geocoder.getFromLocation(Double.valueOf(data.response.get(i).userLatitude),
                            Double.valueOf(data.response.get(i).userLongitude), 1);
                        }
                    }
                }
            }
        }
    });
}

```

```

        String addF = addresses.get(0).getAddressLine(0);
        String city = addresses.get(0).getLocality();
        String state = addresses.get(0).getAdminArea();
        String zip = addresses.get(0).getPostalCode();
        String country = addresses.get(0).getCountryName();

        // UserAddressView.setText(add);

        UserAddressView.setText(addF);
        // StoreUpdatedForemanLocationData();
    } catch (IOException e) {
        e.printStackTrace();
    }

    sp.edit().putString(SharedPreferencesData.Foreman_UserLocationID,
data.response.get(i).userLocationID).commit();
    sp.edit().putString(SharedPreferencesData.Foreman_UserCurrLatitude,
data.response.get(i).userLatitude).commit();
    sp.edit().putString(SharedPreferencesData.Foreman_UserCurrLongitude,
data.response.get(i).userLongitude).commit();
}

} else {
    new CommonMethod(ForemanTrackingActivity.this, response.body().message());
}

} else {
    new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "+response.code());
}

```

```
    }

    @Override
    public void onFailure(Call< GetUserLocationData> call, Throwable t) {
        new CommonMethod(ForemanTrackingActivity.this, t.getMessage());
    }
});
```

}

```
    @Override
    protected void onResume() {
        super.onResume();

        // for the system's orientation sensor registered listeners
        mSensorManager.registerListener(ForemanTrackingActivity.this,
            mSensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
            SensorManager.SENSOR_DELAY_GAME);
    }
}

@Override
protected void onPause() {
    super.onPause();

    // to stop the listener and save battery
    mSensorManager.unregisterListener(this);
}
```

```
    @Override
    public void onSensorChanged(SensorEvent event) {
```

```

// get the angle around the z-axis rotated
float degree = Math.round(event.values[0]);

// create a rotation animation (reverse turn degree degrees)
RotateAnimation ra = new RotateAnimation(
    currentDegree,
    -degree,
    Animation.RELATIVE_TO_SELF, 0.5f,
    Animation.RELATIVE_TO_SELF,
    0.5f);

// how long the animation will take place
ra.setDuration(210);

// set the animation after the end of the reservation status
ra.setFillAfter(true);

// Start the animation
currentDegree = -degree;

}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // not in use
}

}

@Override

```

```
public void onMapReady(GoogleMap googleMap) {  
    mMap = googleMap;  
  
    if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {  
        if (ContextCompat.checkSelfPermission(this,  
            Manifest.permission.ACCESS_FINE_LOCATION)  
            == PackageManager.PERMISSION_GRANTED) {  
            buildGoogleApiClient();  
            mMap.setMyLocationEnabled(true);  
        }  
    } else {  
        buildGoogleApiClient();  
        //mMap.setMyLocationEnabled(true);  
    }  
  
    //    // Add a marker in Sydney and move the camera  
    //  
    //    gpsTracker = new GPSTracker(ForemanTrackingActivity.this);  
    //  
    //    latitude = gpsTracker.getLatitude();  
    //    longitude = gpsTracker.getLongitude();  
    //  
    //    LatLng sydney = new LatLng(latitude, longitude);  
    //    mMap.addMarker(new MarkerOptions().position(sydney).title("Sydney"));  
    //    mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
    //    locationListener = new LocationListener() {  
    //        @Override  
    //        public void onLocationChanged(Location location) {  
    //  
    //            try {  
    //                latLng = new LatLng(23.0225, 72.5714);  
    //            }  
    //        }  
    //    }  
}
```

```

//           mMap.addMarker(new MarkerOptions().position(latLng).title("My Position"));

//
//           mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
//
//       }

//       catch (SecurityException e){
//           e.printStackTrace();
//
//       }

//
//   }

//
//   });

//
//   locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

//
//   try {

//       locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
// MIN_TIME, MIN_DIST, locationListener);

//
//   }

//   catch (SecurityException e){

//       e.printStackTrace();

//
//   }

}

```

```

protected synchronized void buildGoogleApiClient() {

mGoogleApiClient = new GoogleApiClient.Builder(this)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
    .addApi(LocationServices.API).build();

mGoogleApiClient.connect();

}

```

```
@Override
public void onPointerCaptureChanged(boolean hasCapture) {
    super.onPointerCaptureChanged(hasCapture);
}

@Override
public void onConnected(@Nullable Bundle bundle) {

    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(1000);
    mLocationRequest.setFastestInterval(1000);

    mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURA-
CY);
    if (ContextCompat.checkSelfPermission(this,
        Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {
        LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
        mLocationRequest, ForemanTrackingActivity.this);
    }
}

@Override
public void onConnectionSuspended(int i) {

}

@Override
public void onConnectionFailed(@NonNull ConnectionResult connectionResult) {
```

```
}

@Override
public void onLocationChanged(@NonNull Location location) {

    mLastLocation = location;
    if (mCurrLocationMarker != null) {
        mCurrLocationMarker.remove();
    }

    //Place current location marker
    final LatLng[] latLng = {new LatLng(location.getLatitude(), location.getLongitude())};
    final MarkerOptions markerOptions = new MarkerOptions();
    markerOptions.position(latLng[0]);
    markerOptions.title("Your Current Position");
    markerOptions.draggable(true);

    markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.icforeman));

    mCurrLocationMarker = mMap.addMarker(markerOptions);
    mMap.setMapType(mMap.MAP_TYPE_SATELLITE);

    Geocoder geocoder = new Geocoder(ForemanTrackingActivity.this, Locale.getDefault());
    List<Address> addresses = null;
    try {
        addresses = geocoder.getFromLocation(location.getLatitude(), location.getLongitude(),
        1);
        String add = addresses.get(0).getAddressLine(0);
        String city = addresses.get(0).getLocality();
        String state = addresses.get(0).getAdminArea();
```

```

String zip = addresses.get(0).getPostalCode();
String country = addresses.get(0).getCountryName();

// UserAddressView.setText(add);

sForemanLatitude = String.valueOf(location.getLatitude());
sForemanLongitude = String.valueOf(location.getLongitude());
sForemanAddress = add;
// StoreUpdatedForemanLocationData();
} catch (IOException e) {
    e.printStackTrace();
}

//move map camera
mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng[0]));
mMap.animateCamera(CameraUpdateFactory.zoomTo(15));
//mMap.setTrafficEnabled(true);

//stop location updates
if (mGoogleApiClient != null) {
    LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient, this);
}

mMap.setOnMyLocationChangeListener(new GoogleMap.OnMyLocationChangeListener()
{
    @Override
    public void onMyLocationChange(Location location) {
        if (location == null)
            return;

        mCurrLocationMarker = mMap.addMarker(new MarkerOptions())

```

```

        .flat(true)
        .title("Incident Location")
        .icon(BitmapDescriptorFactory
            .fromResource(R.drawable.iccar))
        .anchor(0.5f, 1f)
        .position(new LatLng(userLatitude, userLongitude)));
    }

});

}

private void StoreUpdatedForemanLocationData() {

    Call<LocationData> call = apiInterface.UpdateForemanLocationData(
        sp.getString(SharedPreferencesData.Foreman_ForemanLocationID, ""),
        sForemanLatitude,
        sForemanLongitude
    );

    call.enqueue(new Callback<LocationData>() {
        @Override
        public void onResponse(Call<LocationData> call, Response<LocationData> response) {
            if(response.code()==200){

                if(response.body().status==true){
                    new CommonMethod(ForemanTrackingActivity.this, "Update Foreman");
                    // new CommonMethod(ForemanTrackingActivity.this, response.body().message);
                } else {
                    new CommonMethod(ForemanTrackingActivity.this, response.body().message);
                }
            }
        }
    });
}

```

```

        }

    } else {
        new CommonMethod(ForemanTrackingActivity.this, "Server Error Code : "+response.code());
    }

}

@Override
public void onFailure(Call<LocationData> call, Throwable t) {
    new CommonMethod(ForemanTrackingActivity.this, t.getMessage());
}

});

}
}

```

Foreman Profile activity:

```
package com.brainybeam.roadsideassistance.Foreman.Activity;
```

```

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;

```

```
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.provider.Settings;
import android.util.Log;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
```

```
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.DeleteUserORForemanData;
import com.brainybeam.roadsideassistance.RetrofitData.GetProfileImageData;
import com.brainybeam.roadsideassistance.RetrofitData.LocationData;
import com.brainybeam.roadsideassistance.RetrofitData.UpdateForemanData;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConstantData;
import com.brainybeam.roadsideassistance.Utils.FilePath;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.squareup.picasso.Picasso;
//import com.zhihu.matisse.Matisse;
//import com.zhihu.matisse.MimeType;
//import com.zhihu.matisse.engine.impl.GlideEngine;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.TimeUnit;
```

```
import de.hdodenhof.circleimageview.CircleImageView;
import okhttp3.MediaType;
import okhttp3.MultipartBody;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanProfileActivity extends AppCompatActivity {

    LinearLayout layout1, layout2;
    CircleImageView ProfileImage1, ProfileImage2;
    TextView FirstName, LastName, Contact, Email, Address, Area, City, State, AccountStatus;
    EditText FirstName_EditText, LastName_EditText, Contact_EditText, Email_EditText,
    Password_EditText, Address_EditText, Area_EditText, City_EditText;
    Spinner Spinner_State;
    Button EditTextButton, UpdateButton, DeactivateButton1, DeactivateButton2;

    ArrayList<String> arrayList;
    String sFirstName, sLastName, sContact, sEmail, sPassword, sAddress, sArea, sCity, sState;

    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;
    Bundle bundle;

    FirebaseApp firebaseApp;
    private FirebaseAuth mAuth;
    private PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks;
    private String sPhone;
```

```
private String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\.\+[a-z]+";  
  
String[] appPermission = {Manifest.permission.READ_EXTERNAL_STORAGE};  
private static final int PERMISSION_REQUEST_CODE = 1240;  
private static final int PICK_IMAGE_REQUEST = 12;  
String sImagePath1 = "";  
String sImagePath2 = "";  
  
String getImageURL = "";  
double DriverLatitude, DriverLongitude;  
String sFullAddress, sForemanLatitude, sForemanLongitude;  
  
private static final int REQUEST_LOCATION = 1;  
LocationManager locationManager;  
  
GPSTracker gpsTracker;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_foreman_profile);  
    getSupportActionBar().setTitle("Profile");  
    ColorDrawable colorDrawable  
        = new ColorDrawable(Color.parseColor("#2196F3"));  
    getSupportActionBar().setBackgroundDrawable(colorDrawable);  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
  
    firebaseApp = FirebaseApp.initializeApp(ForemanProfileActivity.this);  
    mAuth = FirebaseAuth.getInstance();
```

```
apiInterface = ApiClient.getClient().create(ApiInterface.class);
sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

layout1 = findViewById(R.id.foreman_profile_layout1);
layout2 = findViewById(R.id.foreman_profile_layout2);
ProfileImage1 = findViewById(R.id.foreman_profile_foremanProfileImage);
ProfileImage2 = findViewById(R.id.foreman_profile_userProfileImage2);
FirstName = findViewById(R.id.foreman_profile_FirstName);
LastName = findViewById(R.id.foreman_profile_LastName);
Contact = findViewById(R.id.foreman_profile_MobileNumber);
Email = findViewById(R.id.foreman_profile_Email);
Address = findViewById(R.id.foreman_profile_Address);
Area = findViewById(R.id.foreman_profile_Area);
City = findViewById(R.id.foreman_profile_City);
State = findViewById(R.id.foreman_profile_State);
AccountStatus = findViewById(R.id.foreman_profile_AccountStatus);
FirstName_EditText = findViewById(R.id.foreman_profile_FirstNameEditText);
LastName_EditText = findViewById(R.id.foreman_profile_LastNameEditText);
Contact_EditText = findViewById(R.id.foreman_profile_MobileNumberEditText);
Email_EditText = findViewById(R.id.foreman_profile_EmailEditText);
Password_EditText = findViewById(R.id.foreman_profile_PasswordEditText);
Address_EditText = findViewById(R.id.foreman_profile_AddressEditText);
Area_EditText = findViewById(R.id.foreman_profile_AreaEditText);
City_EditText = findViewById(R.id.foreman_profile_CityEditText);
Spinner_State = findViewById(R.id.foreman_profile_SpinnerState);
EditTextButton = findViewById(R.id.foreman_profile_EditProfileButton);
UpdateButton = findViewById(R.id.foreman_profile_UpdateButton);
DeactivateButton1 = findViewById(R.id.foreman_profile_deactivateButton);
DeactivateButton2 = findViewById(R.id.foreman_profile_deactivateButton2);

layout2.setVisibility(View.GONE);
```

```

gpsTracker = new GPSTracker(ForemanProfileActivity.this);

// Check if GPS enabled
if(gpsTracker.canGetLocation()) {

    // CurrentLatitude = gpsTracker.getLatitude();
    // CurrentLongitude = gpsTracker.getLongitude();

    // getLocation();

} else {
    // Can't get location.
    // GPS or network is not enabled.
    // Ask user to enable GPS/network in settings.
    gpsTracker.showSettingsAlert();
}

// TODO Get Profile Image
getUserImage();
if(sp.getString(SharedPreferencesData.ProfileImage, "").isEmpty() ||
    sp.getString(SharedPreferencesData.ProfileImage, "").equalsIgnoreCase("") ||
    sp.getString(SharedPreferencesData.ProfileImage,
    "").equalsIgnoreCase("ProfileImage")){
    } else {

Picasso.with(ForemanProfileActivity.this).load(sp.getString(SharedPreferencesData.ProfileImag
e, "")).placeholder(R.drawable.ic_profile).into(ProfileImage1);
}

```

```

FirstName.setText(sp.getString(SharedPreferencesData.FirstName, ""));
LastName.setText(sp.getString(SharedPreferencesData.LastName, ""));
Contact.setText(sp.getString(SharedPreferencesData.MobileNumber, ""));
Email.setText(sp.getString(SharedPreferencesData.Email, ""));
Address.setText(sp.getString(SharedPreferencesData.ForemanAddress, ""));
Area.setText(sp.getString(SharedPreferencesData.ForemanArea, ""));
City.setText(sp.getString(SharedPreferencesData.ForemanCity, ""));
State.setText(sp.getString(SharedPreferencesData.ForemanState, ""));
AccountStatus.setText(sp.getString(SharedPreferencesData.Account_Status, ""));

EditTextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        layout2.setVisibility(View.VISIBLE);
        layout1.setVisibility(View.GONE);
    }
});

sState = "";
arrayList = new ArrayList<>();
arrayList.add("Select State");
String S_array[] = ConstantData.State;
List<String> list;
list = Arrays.asList(S_array);
arrayList.addAll(list);

Spinner_State.setSelection(arrayList.indexOf("Select State"));
ArrayAdapter adapter = new ArrayAdapter(ForemanProfileActivity.this,
    android.R.layout.simple_list_item_1, arrayList);
adapter.setDropDownViewResource(android.R.layout.simple_list_item_activated_1);
Spinner_State.setAdapter(adapter);

```

```

Spinner_State.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        sState = arrayList.get(i);
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {

    }
});
```

```

UpdateButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        sFirstName = FirstName_EditText.getText().toString();
        sLastName = LastName_EditText.getText().toString();
        sContact = Contact_EditText.getText().toString();
        sEmail = Email_EditText.getText().toString();
        sPassword = Password_EditText.getText().toString();
        sAddress = Address_EditText.getText().toString();
        sArea = Area_EditText.getText().toString();
        sCity = City_EditText.getText().toString();

        if(sFirstName.isEmpty() || sFirstName.equalsIgnoreCase("")){
            FirstName_EditText.setError("FirstName is Required");
        } else if(sLastName.isEmpty() || sLastName.equalsIgnoreCase("")){
            LastName_EditText.setError("LastName is Required");
        } else if(sContact.isEmpty() || sContact.equalsIgnoreCase("")){
```

```

        Contact_EditText.setError("Mobile number is Required");
    } else if(sContact.length()>10 || sContact.length()<10){
        Contact_EditText.setError("Mobile number is Not valid");
    }
    else if(sEmail.isEmpty() || sEmail.equalsIgnoreCase("")){
        Email_EditText.setError("Email is Required");
    } else if(!sEmail.matches>EmailPattern)){
        Email_EditText.setError("Valid Email is Required");
    } else if(sPassword.isEmpty() || sPassword.equalsIgnoreCase("")){
        Password_EditText.setError("Password is Required");
    } else if(sPassword.length()<8){
        Password_EditText.setError("Password must be 8 char long");
    } else if(sAddress.isEmpty() || sAddress.equalsIgnoreCase("")){
        Address_EditText.setError("Address is Required");
    } else if(sArea.isEmpty() || sArea.equalsIgnoreCase("")){
        Area_EditText.setError("Area is Required");
    } else if(sCity.isEmpty() || sCity.equalsIgnoreCase("")){
        City_EditText.setError("City is Required");
    } else if(sState.isEmpty() || sState.equalsIgnoreCase("")){
        new CommonMethod(ForemanProfileActivity.this, "State is Required");
    } else if(sState.equalsIgnoreCase("Select State")){
        new CommonMethod(ForemanProfileActivity.this, "Not Valid State");
    } else {

        pd = new ProgressDialog(ForemanProfileActivity.this);
        pd.setMessage("Please Wait... ");
        pd.setCancelable(false);
        pd.show();

        UpdateUserProfileImageData();
        sp.edit().putString(SharedPreferencesData.ProfileImage, sImagePath2).commit();
    }
}

```

```
        UpdateUserProfileData();
        UpdateSPPProfileLocationData();

    }

});

ProfileImage1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        selectImageMethod();
    }
});

ProfileImage2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        selectImageMethod();
    }
});

DeactivateButton1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        pd = new ProgressDialog(ForemanProfileActivity.this);
        pd.setMessage("Please Wait... ");
        pd.setCancelable(false);
        pd.show();

        DeactivateUserAccount();
    }
});
```

```

        }

    });

DeactivateButton2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        pd = new ProgressDialog(ForemanProfileActivity.this);
        pd.setMessage("Please Wait... ");
        pd.setCancelable(false);
        pd.show();

        DeactivateUserAccount();
    }
});

private void UpdateSPProfileLocationData() {

    String FullAddress = sp.getString(SharedPreferencesData.ForemanAddress,
""")+", "+sp.getString(SharedPreferencesData.ForemanArea,
""")+", "+sp.getString(SharedPreferencesData.ForemanCity,
""")+", "+sp.getString(SharedPreferencesData.ForemanState, "");

    Geocoder coder = new Geocoder(ForemanProfileActivity.this);
    List<android.location.Address> address;
    try {
        // May throw an IOException
        address = coder.getFromLocationName(FullAddress, 5);
        if (address == null) {
            DriverLatitude = 0.00;
            DriverLongitude = 0.00;

```

```

    }

    Address location = address.get(0);

    DriverLatitude = location.getLatitude();
    DriverLongitude = location.getLongitude();
    //p1 = new LatLng(location.getLatitude(), location.getLongitude() );

} catch (IOException ex) {

    ex.printStackTrace();
}

sForemanLatitude = String.valueOf(DriverLatitude);
sForemanLongitude = String.valueOf(DriverLongitude);

Call<LocationData> call = apiInterface.UpdateSPPProfileLocationData(
    sp.getString(SharedPreferencesData.UserID, ""),
    sForemanLatitude,
    sForemanLongitude
);

call.enqueue(new Callback<LocationData>() {
    @Override
    public void onResponse(Call<LocationData> call, Response<LocationData> response) {

        if(response.code()==200){

            if(response.body().status==true){

```

```

        } else {
            new CommonMethod(ForemanProfileActivity.this, response.body().message());
        }

    } else {
        new CommonMethod(ForemanProfileActivity.this, "Server Error Code
"+response.code());
    }

}

@Override
public void onFailure(Call<LocationData> call, Throwable t) {
    new CommonMethod(ForemanProfileActivity.this, t.getMessage());
}

});

}

private void getUserImage() {

    Call<GetProfileImageData> call = apiInterface.GetForemanProfileImage(
        sp.getString(SharedPreferencesData.UserID, ""))
);

    call.enqueue(new Callback<GetProfileImageData>() {
        @Override

```

```

public void onResponse(Call<GetProfileImageData> call,
Response<GetProfileImageData> response) {
    if(response.code()==200){
        if(response.body().status==true){
            // new CommonMethod(ForemanProfileActivity.this,response.body().message);

            GetProfileImageData data = response.body();
            for(int i=0; i<data.response.size(); i++){
                getImageURL = data.response.get(i).profileImage;
            }

            sp.edit().putString(SharedPreferencesData.ProfileImage,
getImageURL).commit();
        }
        else{
            new CommonMethod(ForemanProfileActivity.this,response.body().message);
        }
    }
    else{
        new CommonMethod(ForemanProfileActivity.this,"Server Error Code : "+response.code());
    }
}

@Override
public void onFailure(Call<GetProfileImageData> call, Throwable t) {
    new CommonMethod(ForemanProfileActivity.this,t.getMessage());
}
}

```

```

private void UpdateUserProfileImageData() {

    RequestBody namePart = RequestBody.create(MultipartBody.FORM,
sp.getString(SharedPreferencesData.FirstName, ""));

    File imageFile = new File(sImagePath2);
    RequestBody imageBody = RequestBody.create(MediaType.parse("image/*"), imageFile);
    MultipartBody.Part imagesParts = MultipartBody.Part.createFormData("file",
imageFile.getName(), imageBody);

    Call<DeleteUserORForemanData> call = apiInterface.AddForemanProfileImage(
        sp.getString(SharedPreferencesData.UserID, ""), namePart, imagesParts
    );
    call.enqueue(new Callback<DeleteUserORForemanData>() {
        @Override
        public void onResponse(Call<DeleteUserORForemanData> call,
Response<DeleteUserORForemanData> response) {
            pd.dismiss();
            if(response.code()==200){
                if(response.body().status==true){
                    new CommonMethod(ForemanProfileActivity.this,response.body().message);
                }
                else{
                    new CommonMethod(ForemanProfileActivity.this,response.body().message);
                }
            }
            else{
                new CommonMethod(ForemanProfileActivity.this,"Server Error Code : "+response.code());
            }
        }
    });
}

```

```

        }

    }

    @Override
    public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {
        pd.dismiss();
        new CommonMethod(ForemanProfileActivity.this,t.getMessage());
    }
);

}

private void DeactivateUserAccount() {

    Call<DeleteUserORForemanData> call = apiInterface.DeactivateForemanAccountData(
        sp.getString(SharedPreferencesData.UserID, "")
    );

    call.enqueue(new Callback<DeleteUserORForemanData>() {
        @Override
        public void onResponse(Call<DeleteUserORForemanData> call,
Response<DeleteUserORForemanData> response) {
            pd.dismiss();
            if(response.code()==200){

                if (response.body().status==true){

                    new CommonMethod(ForemanProfileActivity.this, "Your Account SuccessFully
Deactivate");
                    new CommonMethod(ForemanProfileActivity.this, LoginActivity.class);
                }
            }
        }
    });
}

```

```

        } else {
            new CommonMethod(ForemanProfileActivity.this, response.body().message);
        }

    } else {
        new CommonMethod(ForemanProfileActivity.this, "Server Error Code
"+response.code());
    }

}

@Override
public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {
    pd.dismiss();
    new CommonMethod(ForemanProfileActivity.this, t.getMessage());
}
});

}

private void UpdateUserProfileData() {

Call<UpdateForemanData> call = apiInterface.UpdateForemanData(
    sp.getString(SharedPreferencesData.UserID, ""),
    sFirstName, sLastName, sContact, sEmail, sPassword,
    sAddress, sArea, sCity, sState
);

call.enqueue(new Callback<UpdateForemanData>() {

```

```

@Override
public void onResponse(Call<UpdateForemanData> call,
Response<UpdateForemanData> response) {
    pd.dismiss();
    if(response.code()==200){

        if(response.body().status.equalsIgnoreCase("True")){
            new CommonMethod(ForemanProfileActivity.this, response.body().message);

            sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).commit();
            sp.edit().putString(SharedPreferencesData.LastName, sLastName).commit();
            sp.edit().putString(SharedPreferencesData.MobileNumber, sContact).commit();
            sp.edit().putString(SharedPreferencesData.Email, sEmail).commit();
            sp.edit().putString(SharedPreferencesData.Password, sPassword).commit();
            sp.edit().putString(SharedPreferencesData.ForemanAddress, sAddress).commit();
            sp.edit().putString(SharedPreferencesData.ForemanArea, sArea).commit();
            sp.edit().putString(SharedPreferencesData.ForemanCity, sCity).commit();
            sp.edit().putString(SharedPreferencesData.ForemanState, sState).commit();

            FirstName.setText(sp.getString(SharedPreferencesData.FirstName, ""));
            LastName.setText(sp.getString(SharedPreferencesData.LastName, ""));
            Contact.setText(sp.getString(SharedPreferencesData.MobileNumber, ""));
            Email.setText(sp.getString(SharedPreferencesData.Email, ""));
            Address.setText(sp.getString(SharedPreferencesData.ForemanAddress, ""));
            Area.setText(sp.getString(SharedPreferencesData.ForemanArea, ""));
            City.setText(sp.getString(SharedPreferencesData.ForemanCity, ""));
            State.setText(sp.getString(SharedPreferencesData.ForemanState, ""));
            AccountStatus.setText(sp.getString(SharedPreferencesData.Account_Status, ""));

            layout1.setVisibility(View.VISIBLE);
        }
    }
}

```

```

        layout2.setVisibility(View.GONE);

    } else if(response.body().status.equalsIgnoreCase("Pending")){
        new CommonMethod(ForemanProfileActivity.this, response.body().message());

        sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).commit();
        sp.edit().putString(SharedPreferencesData.LastName, sLastName).commit();
        sp.edit().putString(SharedPreferencesData.MobileNumber, sContact).commit();
        sp.edit().putString(SharedPreferencesData.Email, sEmail).commit();
        sp.edit().putString(SharedPreferencesData.Password, sPassword).commit();
        sp.edit().putString(SharedPreferencesData.ForemanAddress, sAddress).commit();
        sp.edit().putString(SharedPreferencesData.ForemanArea, sArea).commit();
        sp.edit().putString(SharedPreferencesData.ForemanCity, sCity).commit();
        sp.edit().putString(SharedPreferencesData.ForemanState, sState).commit();

        sPhone = sContact;
        otpSendToMobile(sPhone);

    } else {
        new CommonMethod(ForemanProfileActivity.this, response.body().message());
    }

} else {
    new CommonMethod(ForemanProfileActivity.this, "Server Error Code
"+response.code());
}
}

```

```

@Override
public void onFailure(Call<UpdateForemanData> call, Throwable t) {
    pd.dismiss();
    new CommonMethod(ForemanProfileActivity.this, t.getMessage());
}

});

}

private void otpSendToMobile(String sPhone) {

mCallbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

@Override
public void onVerificationCompleted(PhoneAuthCredential credential) {

}

@Override
public void onVerificationFailed(FirebaseException e) {
    pd.dismiss();
    new CommonMethod(ForemanProfileActivity.this, e.getLocalizedMessage());
}

@Override
public void onCodeSent(@NonNull String VerificationId,
                      @NonNull PhoneAuthProvider.ForceResendingToken token) {
    pd.dismiss();

    new CommonMethod(ForemanProfileActivity.this, "OTP is successFully Send");
}

```

```

        bundle.putString("PhoneNumber", sPhone.trim());
        bundle.putString("Mobile_VerificationID", VerificationId);
        Intent intent = new Intent(ForemanProfileActivity.this, OTPVerificationActivity.class);
        intent.putExtras(bundle);
        startActivity(intent);
    }
};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(ForemanProfileActivity.this)
        .setCallbacks(mCallbacks)
        .build();
PhoneAuthProvider.verifyPhoneNumber(options);

}

```

```

private void selectImageMethod() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent, "Select Pdf"),
        PICK_IMAGE_REQUEST);
}

```

```

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==PICK_IMAGE_REQUEST && resultCode==RESULT_OK && data
!=null){
        /*Uri uri = data.getData();
        Log.d("RESPONSE_URI", String.valueOf(uri));
        imageView.setImageURI(uri);*/
        sImagePath1 = FilePath.getPath(ForemanProfileActivity.this, data.getData());
        sImagePath2 = FilePath.getPath(ForemanProfileActivity.this, data.getData());
        // Log.d("RESPONSE_IMAGE_PATH",sImagePath);
        // fileText.setVisibility(View.VISIBLE);
        // uploadButton.setVisibility(View.VISIBLE);
    }
}

public boolean checkAndRequestPermission() {
    List<String> listPermission = new ArrayList<>();
    for (String perm : appPermission) {
        if (!ContextCompat.checkSelfPermission(ForemanProfileActivity.this, perm) != PackageManager.PERMISSION_GRANTED) {
            listPermission.add(perm);
        }
    }
    if (!listPermission.isEmpty()) {
        ActivityCompat.requestPermissions(ForemanProfileActivity.this,
listPermission.toArray(new String[listPermission.size()]), PERMISSION_REQUEST_CODE);
        return false;
    }
    return true;
}

```

```

}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == PERMISSION_REQUEST_CODE) {
        HashMap<String, Integer> permissionResult = new HashMap<>();
        int deniedCount = 0;
        for (int i = 0; i < grantResults.length; i++) {
            if (grantResults[i] == PackageManager.PERMISSION_DENIED) {
                permissionResult.put(permissions[i], grantResults[i]);
                deniedCount++;
            }
        }
        if (deniedCount == 0) {
            selectImageMethod();
        } else {
            for (Map.Entry<String, Integer> entry : permissionResult.entrySet()) {
                String permName = entry.getKey();
                int permResult = entry.getValue();
                if (ActivityCompat.shouldShowRequestPermissionRationale(
                        ForemanProfileActivity.this, permName)) {
                    showDialogPermission("", "This App needs Read External Storage permissions to
work without any problems.",
                            "Yes, Grant permissions", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialogInterface, int i) {
                                    dialogInterface.dismiss();
                                    checkAndRequestPermission();
                                }
                            })
                }
            }
        }
    }
}

```

```

    },
    "No, Exit app", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
            ForemanProfileActivity.this.finishAffinity();
        }
    }, false);
} else {
    showDialogPermission("", "You have denied some permissions. Allow all
permissions at [Setting] > [Permissions]",
    "Go to Settings", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
            Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
        Uri.fromParts("package",
ForemanProfileActivity.this.getPackageName(), null));
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            ForemanProfileActivity.this.finish();
        }
    }, "No, Exit app", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
            ForemanProfileActivity.this.finish();
        }
    }, false);
break;
}

```

```

        }
    }
}
}

public AlertDialog showDialogPermission(String title, String msg, String positiveLable,
DialogInterface.OnClickListener      positiveOnClickListener,      String      negativeLable,
DialogInterface.OnClickListener negativeOnClickListener, boolean isCancelable) {
    AlertDialog.Builder builder = new AlertDialog.Builder(ForemanProfileActivity.this);
    builder.setTitle(title);
    builder.setCancelable(isCancelable);
    builder.setMessage(msg);
    builder.setPositiveButton(positiveLable, positiveOnClickListener);
    builder.setNegativeButton(negativeLable, negativeOnClickListener);
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
    return alertDialog;
}

```

```

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}

```

Foreman Home pending payment activity:

```
package com.brainybeam.roadsideassistance.Foreman.Activity;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.app.AlertDialog;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;

import com.brainybeam.roadsideassistance.Foreman.CustomArrayList.PendingPaymentList;
import
com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanUserSelectedServicesAdapter;
import com.brainybeam.roadsideassistance.R;
```

```
import com.brainybeam.roadsideassistance.RetrofitData.CartPendingPaymentData;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;

import java.util.ArrayList;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanHomePendingPaymentActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    CardView cardView;
    ArrayList<PendingPaymentList> arrayList;

    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_foreman_home_pending_payment);
        getSupportActionBar().setTitle("Pending Payment");
        ColorDrawable colorDrawable
```

```

        = new ColorDrawable(Color.parseColor("#2196F3"));
getSupportActionBar().setBackgroundDrawable(colorDrawable);
getSupportActionBar().setDisplayHomeAsUpEnabled(true);

apiInterface = ApiClient.getClient().create(ApiInterface.class);
sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

recyclerView = findViewById(R.id.foremanHome_PendingPayment_RecyclerView);
cardView = findViewById(R.id.foremanHome_PendingPayment_CardViewNotHaveAnyPendingActivity);

cardView.setVisibility(View.GONE);

recyclerView.setLayoutManager(new
LinearLayoutManager(ForemanHomePendingPaymentActivity.this));
recyclerView.setItemAnimator(new DefaultItemAnimator());

if(new
ConnectionDetector(ForemanHomePendingPaymentActivity.this).isConnectingToInternet(){

    pd = new ProgressDialog(ForemanHomePendingPaymentActivity.this);
    pd.setMessage("Please Wait... ");
    pd.setCancelable(false);
    pd.show();

    recyclerViewDataSetMethod();

} else {
    new
ConnectionDetector(ForemanHomePendingPaymentActivity.this).connectiondetect();
}

```

```
}
```

```
private void recyclerViewSetMethod() {

    Call<CartPendingPaymentData> call = apiInterface.GetCartPendingPaymentData(
        sp.getString(SharedPreferencesData.UserID, ""))
    );

    call.enqueue(new Callback<CartPendingPaymentData>() {
        @Override
        public void onResponse(Call<CartPendingPaymentData> call,
        Response<CartPendingPaymentData> response) {

            pd.dismiss();
            if(response.code()==200){

                if(response.body().status==true){

                    arrayList = new ArrayList<>();
                    CartPendingPaymentData data = response.body();
                    for(int i=0; i<data.response.size(); i++){

                        PendingPaymentList list = new PendingPaymentList();

                        list.setCartID(data.response.get(i).cartID);
                        list.setServiceID(data.response.get(i).serviceID);
                        list.setForemanID(data.response.get(i).foremanID);
                        list.setUserID(data.response.get(i).userID);
                        list.setVehicleID(data.response.get(i).vehicleID);
                    }
                }
            }
        }
    });
}
```

```

list.setProblemDescriptionMessage(data.response.get(i).problemDescriptionMessage);

        list.setSPReqMoney(data.response.get(i).sPReqMoney);
        list.setPaymentMode(data.response.get(i).paymentMode);
        list.setPayment(data.response.get(i).payment);
        list.setPaymentStatus(data.response.get(i).paymentStatus);
        list.setTypeOfProblem(data.response.get(i).typeOfProblem);
        list.setProblemSubType(data.response.get(i).problemSubType);
        list.setServiceFixedCharge(data.response.get(i).serviceFixedCharge);
        list.setUserFirstName(data.response.get(i).firstName);
        list.setUserLastName(data.response.get(i).lastName);
        list.setUserProfileImage(data.response.get(i).profileImage);
        list.setUserMobileNumber(data.response.get(i).mobileNumber);
        list.setNumberPlate_number(data.response.get(i).numberPlateNumber);
        list.setTypeOfVehicle(data.response.get(i).typeOfVehicle);
        list.setVehicleModelName(data.response.get(i).vehicleModelName);
        list.setVehicle_Colour(data.response.get(i).vehicleColour);
        list.setForemanLocationID(data.response.get(i).foremanLocationID);
        list.setForemanLatitude(data.response.get(i).foremanLatitude);
        list.setForemanLongitude(data.response.get(i).foremanLongitude);
        list.setUserLocationID(data.response.get(i).userLocationID);
        list.setUserLatitude(data.response.get(i).userLatitude);
        list.setUserLongitude(data.response.get(i).userLongitude);

arrayList.add(list);
}

```

```

ForemanHomePendingPaymentAdapter adapter = new
ForemanHomePendingPaymentAdapter(ForemanHomePendingPaymentActivity.this, arrayList);
recyclerView.setAdapter(adapter);

```

```

        } else {
            new CommonMethod(ForemanHomePendingPaymentActivity.this,
response.body().message);
                recyclerView.setVisibility(View.GONE);
                cardView.setVisibility(View.VISIBLE);
            }
        }

    } else {
        new CommonMethod(ForemanHomePendingPaymentActivity.this, "Server Error
Code "+response.code());
    }
}

@Override
public void onFailure(Call<CartPendingPaymentData> call, Throwable t) {
    new CommonMethod(ForemanHomePendingPaymentActivity.this, t.getMessage());
}
});

}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}
}

```

Foreman user payment activity:

```
package com.brainybeam.roadsideassistance.Foreman.Activity;

import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import com.brainybeam.roadsideassistance.Foreman.DashBoard.ForemanDashboardActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.AddForemanServicesData;
import com.brainybeam.roadsideassistance.RetrofitData.CartAddServicesData;
import com.brainybeam.roadsideassistance.RetrofitData.SendNotificationData;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
```

```
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanUserPaymentActivity extends AppCompatActivity {

    EditText money;
    Button RequestButton, VerifyButton;

    LinearLayout main_layout, payment_layout;

    String paying;

    ApiInterface apiInterface;
    SharedPreferences sp;
    ProgressDialog pd;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_foreman_user_payment);
        getSupportActionBar().setTitle("Set Payment->Bill");
        ColorDrawable colorDrawable
                = new ColorDrawable(Color.parseColor("#2196F3"));
        getSupportActionBar().setBackgroundDrawable(colorDrawable);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        apiInterface = ApiClient.getClient().create(ApiInterface.class);
        sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

        money = findViewById(R.id.foreman_user_payment_MoneyTextView);
```

```

RequestButton = findViewById(R.id.foreman_user_payment_PayButton);
VerifyButton = findViewById(R.id.foreman_user_payment_VerifyButton);

main_layout = findViewById(R.id.foreman_user_payment_MainLinearLayout);
payment_layout = findViewById(R.id.foreman_user_payment_paymentLinearLayout);

main_layout.setVisibility(View.VISIBLE);
payment_layout.setVisibility(View.GONE);

RequestButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        paying = money.getText().toString();

        pd = new ProgressDialog(ForemanUserPaymentActivity.this);
        pd.setMessage("Payment Request Sending...");
        pd.setCancelable(false);
        pd.show();

        storeSPMoneyData();
        main_layout.setVisibility(View.GONE);
        payment_layout.setVisibility(View.VISIBLE);

    }
});

VerifyButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

```

```

        AlertDialog.Builder alertDialog = new
AlertDialog.Builder(ForemanUserPaymentActivity.this);
        alertDialog.setTitle("Payment Request => Waiting...");  

        alertDialog.setCancelable(false);

        alertDialog.setPositiveButton("Received", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                ToCheckPaymentStatus();
            }
        });

        alertDialog.setNeutralButton("Resend Request", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                new CommonMethod(ForemanUserPaymentActivity.this, "Please Try Again...");  

                new CommonMethod(ForemanUserPaymentActivity.this,
ForemanUserPaymentActivity.class);
            }
        });

        alertDialog.show();

    });
}

```

private void ToCheckPaymentStatus() {

```

Call<CartAddServicesData> call = apiInterface.GetPaymentStatusData(
    sp.getString(SharedPreferencesData.Foreman_CartID, ""))
);

call.enqueue(new Callback<CartAddServicesData>() {
    @Override
    public void onResponse(Call<CartAddServicesData> call,
    Response<CartAddServicesData> response) {
        if(response.code()==200){
            if(response.body().status==true){
                new CommonMethod(ForemanUserPaymentActivity.this, "Payment Received
SuccessFully");
                SendNotificationToUserPaymentSuccess();
                new CommonMethod(ForemanUserPaymentActivity.this,
ForemanDashboardActivity.class);
            } else {
                main_layout.setVisibility(View.VISIBLE);
                payment_layout.setVisibility(View.GONE);
                new CommonMethod(ForemanUserPaymentActivity.this, "Payment Received
UnSuccessFully");
            }
        } else {
            new CommonMethod(ForemanUserPaymentActivity.this, "Server Error Code :
"+response.code());
        }
    }
}

```

```

@Override
public void onFailure(Call<CartAddServicesData> call, Throwable t) {
    new CommonMethod(ForemanUserPaymentActivity.this, t.getMessage());
}
});

}

private void SendNotificationToUserPaymentSuccess() {

String Title = "UserID - "+sp.getString(SharedPreferencesData.UserID, "")+
"+sp.getString(SharedPreferencesData.Foreman_UserFirstName, "")+" +" Payment Received
SuccessFully";
String Message = "Your Payment Received SuccessFully Thank you for payment \n" +
"Please Rate Our Services \n"+
"Thank you!";
}

Call<SendNotificationData> call = apiInterface.SendToUserNotificationData(
    sp.getString(SharedPreferencesData.Foreman_UserID, ""),
    Title,
    Message
);

call.enqueue(new Callback<SendNotificationData>() {
    @Override
    public void onResponse(Call<SendNotificationData> call,
    Response<SendNotificationData> response) {

        // pd.dismiss();
        if(response.code()==200){

```

```

        if(response.body().status==true){
            //           new      CommonMethod(ForemanUserPaymentActivity.this,
            response.body().message);
        } else {
            new      CommonMethod(ForemanUserPaymentActivity.this,
            response.body().message);
        }

    } else {
        new  CommonMethod(ForemanUserPaymentActivity.this, "Server Error Code :
"+response.code());
    }

}

```

```

@Override
public void onFailure(Call<SendNotificationData> call, Throwable t) {
    // pd.dismiss();
    new CommonMethod(ForemanUserPaymentActivity.this, t.getMessage());
}

}

```

```

private void storeSPMoneyData() {

Call<CartAddServicesData> call = apiInterface.UpdateSPReqMoneyData(
    sp.getString(SharedPreferencesData.Foreman_ServiceID, ""),
    sp.getString(SharedPreferencesData.UserID, ""),
    sp.getString(SharedPreferencesData.Foreman.UserID, ""),
    paying
}

```

```

);

call.enqueue(new Callback<CartAddServicesData>() {
    @Override
    public void onResponse(Call<CartAddServicesData> call,
                          Response<CartAddServicesData> response) {
        pd.dismiss();
        if(response.code()==200){
            if(response.body().status==true){
                // new CommonMethod(ForemanUserPaymentActivity.this,
                response.body().message);
            } else {
                new CommonMethod(ForemanUserPaymentActivity.this,
                response.body().message);
            }
        } else {
            new CommonMethod(ForemanUserPaymentActivity.this, "Server Error Code :
"+response.code());
        }
    }

    @Override
    public void onFailure(Call<CartAddServicesData> call, Throwable t) {
        pd.dismiss();
        new CommonMethod(ForemanUserPaymentActivity.this, t.getMessage());
    }
});

```

```
}

@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home) {
        onBackPressed();
    }
    return super.onOptionsItemSelected(item);
}
}
```

Foreman signup activity:

```
package com.brainybeam.roadsideassistance.Foreman.Signup;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
```

```
import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.ConstantData;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.firebase.FirebaseApp;
import com.google.firebaseio.FirebaseException;
import com.google.firebaseio.auth.FirebaseAuth;
import com.google.firebaseio.auth.PhoneAuthCredential;
import com.google.firebaseio.auth.PhoneAuthOptions;
import com.google.firebaseio.auth.PhoneAuthProvider;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class ForemanSignupActivity extends AppCompatActivity {

    private EditText FirstName, LastName, MobileNumber, Email, Password, Address, Area, City;
    private TextView State;

    private ArrayList<String> arrayList;
    private final String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\.[a-z]+";

    private FirebaseAuth mAuth;

    private String sPhone;
    private SharedPreferences sp;
    FirebaseApp firebaseApp;
```

```
Bundle bundle;

private String sFirstName, sLastName, sMobileNumber, sEmail, sPassword, sAddress, sArea,
sCity, sState;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_foreman_signup);

    firebaseApp = FirebaseApp.initializeApp(getApplicationContext());
    mAuth = FirebaseAuth.getInstance();
    sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

    FirstName = findViewById(R.id.foremanSignup_FirstName);
    LastName = findViewById(R.id.foremanSignup_LastName);
    MobileNumber = findViewById(R.id.foremanSignup_MobileNumber);
    Email = findViewById(R.id.foremanSignup_Email);
    Password = findViewById(R.id.foremanSignup_Password);
    Address = findViewById(R.id.foremanSignup_Address);
    Area = findViewById(R.id.foremanSignup_Area);
    City = findViewById(R.id.foremanSignup_City);
    State = findViewById(R.id.foremanSignup_State);
    Spinner spinner_State = findViewById(R.id.foremanSignup_State_Spinner);
    Button getOTP = findViewById(R.id.foremanSignup_GetOTPButton);

    arrayList = new ArrayList<>();

    arrayList.add("-");
    String[] S_array = ConstantData.State;
```

```
List<String> list;
list = Arrays.asList(S_array);
arrayList.addAll(list);

ArrayAdapter adapter = new ArrayAdapter(ForemanSignupActivity.this,
    android.R.layout.simple_list_item_1, arrayList);
adapter.setDropDownViewResource(android.R.layout.simple_list_item_activated_1);
spinner_State.setAdapter(adapter);

spinner_State.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {

    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        State.setText(arrayList.get(i));
    }

    @Override
    public void onNothingSelected(AdapterView<?> adapterView) {
    }
});

getOTP.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (new ConnectionDetector(ForemanSignupActivity.this).isConnectingToInternet()) {

            sFirstName = FirstName.getText().toString();
        }
    }
});
```

```

sLastName = LastName.getText().toString();
sMobileNumber = MobileNumber.getText().toString().toLowerCase();
sEmail = Email.getText().toString().toLowerCase();
sPassword = Password.getText().toString();
sAddress = Address.getText().toString();
sArea = Area.getText().toString();
sCity = City.getText().toString();
sState = State.getText().toString();

if(sFirstName.isEmpty() || sFirstName.equalsIgnoreCase(" ")){
    FirstName.setError("FirstName is Required");
} else if(sLastName.isEmpty() || sLastName.equalsIgnoreCase(" ")){
    LastName.setError("LastName is Required");
} else if(sMobileNumber.isEmpty() || sMobileNumber.equalsIgnoreCase(" ")){
    MobileNumber.setError("Mobile number is Required");
} else if(sMobileNumber.length()!=10){
    MobileNumber.setError("Valid Mobile Number is Required");
} else if(sEmail.isEmpty() || sEmail.equalsIgnoreCase(" ")){
    Email.setError("Email is Required");
} else if(!sEmail.matches>EmailPattern){
    Email.setError("Valid Email is Required");
} else if(sPassword.isEmpty() || sPassword.equalsIgnoreCase(" ")){
    Password.setError("Password is Required");
} else if(sPassword.length()<8){
    Password.setError("Password must be 8 char long");
} else if(sAddress.isEmpty() || sAddress.equalsIgnoreCase(" ")){
    Address.setError("Address");
} else if(sArea.isEmpty() || sArea.equalsIgnoreCase(" ")){
    Area.setError("Area");
} else if(sCity.isEmpty() || sCity.equalsIgnoreCase(" ")){
    City.setError("City");
}

```

```

    } else if(sState.equalsIgnoreCase("Select State") || sState.equalsIgnoreCase(" ") ||  

sState.equalsIgnoreCase("-")){
        State.setError("Please Select State");
    } else {

        sp.edit().putString(SharedPreferencesData.UserType, "Foreman").apply();
        sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).apply();
        sp.edit().putString(SharedPreferencesData.LastName, sLastName).apply();
        sp.edit().putString(SharedPreferencesData.MobileNumber,
sMobileNumber).apply();
        sp.edit().putString(SharedPreferencesData.Email, sEmail).apply();
        sp.edit().putString(SharedPreferencesData.Password, sPassword).apply();
        sp.edit().putString(SharedPreferencesData.ForemanAddress, sAddress).apply();
        sp.edit().putString(SharedPreferencesData.ForemanArea, sArea).apply();
        sp.edit().putString(SharedPreferencesData.ForemanCity, sCity).apply();
        sp.edit().putString(SharedPreferencesData.ForemanState, sState).apply();

sp.edit().putBoolean(SharedPreferencesData.ForemanAccount_Status,false).apply();

        bundle = new Bundle();
        sPhone = sMobileNumber;
        // TODO OTP Send to Mobile Method
        otpSendToMobile(sPhone);
    }

} else {
    new ConnectionDetector(ForemanSignupActivity.this).connectiondetect();
}

```

```

    }
});

}

private void otpSendToMobile(String sPhone) {

    PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks = new
    PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onVerificationCompleted(@NonNull PhoneAuthCredential credential) {

        }

        @Override
        public void onVerificationFailed(FirebaseException e) {
            new CommonMethod(ForemanSignupActivity.this, e.getLocalizedMessage());
        }

        @Override
        public void onCodeSent(@NonNull String VerificationId,
                              @NonNull PhoneAuthProvider.ForceResendingToken token) {

            new CommonMethod(ForemanSignupActivity.this, "OTP is successFully Send");

            bundle.putString("PhoneNumber", sPhone.trim());
            bundle.putString("Mobile_VerificationID", VerificationId);
            Intent intent = new Intent(ForemanSignupActivity.this, OTPVerificationActivity.class);
            intent.putExtras(bundle);
        }
    };
}

```

```

        startActivity(intent);
    }
};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(this)
        .setCallbacks(mCallbacks)
        .build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

```

}

Dashboard Activity:

```

package com.brainybeam.roadsideassistance.Foreman.DashBoard;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.FragmentManager;

import android.Manifest;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;

```

```
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.Notification.ForemanNotificationActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.ConstantData;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.android.material.navigation.NavigationView;
import com.google.android.material.snackbar.Snackbar;

public class ForemanDashboardActivity extends AppCompatActivity {

    private static final int STORAGE_PERMISSION_CODE = 123;

    TextView Title_menu;
    ImageView menu_image, calling_image, Notification_image;

    // TODO Navigation
    ImageView nav_header_profileImage;
```

```
TextView nav_header_Name;

LinearLayout      nav_header_home_layout,      nav_header_foremanProfile_layout,
nav_header_Services_layout, nav_header_userLocationTracking_layout,
                           nav_header_newReceiveRequest_layout,      nav_header_rating_layout,
nav_header_history_layout, nav_header_setting_layout,
                           nav_header_logout_layout, nav_header_switchAccount_layout;
```

```
ImageView          nav_header_home_imageview_logo,
nav_header_foremanProfile_imageview_logo,      nav_header_Services_imageview_logo,
nav_header_userLocationTracking_imageview_logo,
                           nav_header_newReceiveRequest_imageview_logo, nav_header_rating_imageview_logo,
nav_header_history_imageview_logo, nav_header_setting_imageview_logo,
                           nav_header_logout_imageview_logo, nav_header_switchAccount_imageview_logo;
```

```
TextView      nav_header_home_TextButton,      nav_header_foremanProfile_TextButton,
nav_header_Services_TextButton, nav_header_userLocationTracking_TextButton,
                           nav_header_newReceiveRequest_TextButton,      nav_header_rating_TextButton,
nav_header_history_TextButton, nav_header_setting_TextButton,
                           nav_header_logout_TextButton, nav_header_switchAccount_TextButton;
```

```
SharedPreferences sp;
// ApiInterface apiInterface;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_foreman_dashboard);
```

```

// getSupportActionBar().hide();
//    getSupportActionBar().setTitle("Home");
//    ColorDrawable colorDrawable
//        = new ColorDrawable(Color.parseColor("#2196F3"));
//    getSupportActionBar().setBackgroundDrawable(colorDrawable);
//    getSupportActionBar().setDisplayHomeAsUpEnabled(true);

// Permission get method
requestStoragePermission();

sp = getSharedPreferences(SharedPreferencesData.PREF, MODE_PRIVATE);

if(new ConnectionDetector(ForemanDashboardActivity.this).isConnectingToInternet()){

} else {

    if(!new ConnectionDetector(ForemanDashboardActivity.this).isConnectingToInternet()){

    }

    new ConnectionDetector(ForemanDashboardActivity.this).connectiondetect();
}

// TODO Tab activity Start

Title_menu = findViewById(R.id.foreman_content_main_dashboard_title);
Title_menu.setText("Home");

```

```

DrawerLayout drawer = findViewById(R.id.foreman_drawer_layout);
menu_image =  

findViewById(R.id.foreman_content_main_dashboard_menu_imageButton);
menu_image.setOnClickListener(new View.OnClickListener() {  

    @Override  

    public void onClick(View view) {  

        drawer.openDrawer(GravityCompat.START);  

    }  

});  

calling_image = findViewById(R.id.foreman_content_main_dashboard_call_imageButton);
calling_image.setOnClickListener(new View.OnClickListener() {  

    @Override  

    public void onClick(View view) {  

        AlertDialog.Builder builder = new  

AlertDialog.Builder(ForemanDashboardActivity.this);  

        builder.setTitle("Call Assistance Center?");  

        builder.setPositiveButton("CALL", new DialogInterface.OnClickListener() {  

            @Override  

            public void onClick(DialogInterface dialogInterface, int i) {  

                Intent intent = new Intent(Intent.ACTION_CALL);  

                intent.setData(Uri.parse("tel:"+ ConstantData.Call_Assistance_MobileNumber));  

                if (checkSelfPermission(Manifest.permission.CALL_PHONE) !=  

PackageManager.PERMISSION_GRANTED) {  

                    // TODO: Consider calling  

                    // Activity#requestPermissions  

                    // here to request the missing permissions, and then overriding

```

```

        //     public void onRequestPermissionsResult(int requestCode, String[]
permissions,
        //                                         int[] grantResults)
        // to handle the case where the user grants the permission. See the documentation
        // for Activity#requestPermissions for more details.
        return;
    }
    startActivity(intent);
}
});

builder.setNegativeButton("CANCEL", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.dismiss();
    }
});
builder.show();

});
}

// TODO Tab activity finished

```

```

Notification_image =  

findViewById(R.id.foreman_content_main_dashboard_Notification_image);  

Notification_image.setOnClickListener(new View.OnClickListener() {  

    @Override  

    public void onClick(View view) {  

        new CommonMethod(ForemanDashboardActivity.this,  

ForemanNotificationActivity.class);

```

```

    }

});

// TODO Default Fragment
drawer.closeDrawer(GravityCompat.START);
FragmentManager manager = getSupportFragmentManager();
manager.beginTransaction()
    .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHomeFragment(), null)
    .setReorderingAllowed(true)
    .addToBackStack(null)
    .commit();

//// Call<DeleteUserORForemanData> call = apiInterface.GetForemanNotAcceptedData(
////     sp.getString(SharedPreferencesData.UserID, ""))
//// );
////
//// call.enqueue(new Callback<DeleteUserORForemanData>() {
////
////     @Override
////
////         public void onResponse(Call<DeleteUserORForemanData> call,
Response<DeleteUserORForemanData> response) {
////
////             if(response.code()==200){
////
////                 if(response.body().status==true){
////
////                     drawer.closeDrawer(GravityCompat.START);
////
////                     FragmentManager manager = getSupportFragmentManager();
////
////                     manager.beginTransaction()

```

```

//// .replace(R.id.foreman_content_main_dashboard_FragmentContainerView,
new ForemanHomeFragment(), null)
////
//// .setReorderingAllowed(true)
////
//// .addToBackStack("")
////
//// .commit();
////
////
//// @SuppressLint("ResourceType") View fragment =
findViewById(R.layout.fragment_foreman_home);
////
fragment.findViewById(R.id.frag_foreman_home_layout1).setVisibility(View.VISIBLE);
////
fragment.findViewById(R.id.frag_foreman_home_layout2).setVisibility(View.GONE);
////
////
} else {
new CommonMethod(ForemanDashboardActivity.this, response.body().message);
drawer.closeDrawer(GravityCompat.START);
FragmentManager manager = getSupportFragmentManager();
manager.beginTransaction()
.replace(R.id.foreman_content_main_dashboard_FragmentContainerView,
new ForemanHomeFragment(), null)
.setReorderingAllowed(true)
.addToBackStack("")
.commit();
////
////
//// @SuppressLint("ResourceType") View fragment =
findViewById(R.layout.fragment_foreman_home);
////
fragment.findViewById(R.id.frag_foreman_home_layout1).setVisibility(View.VISIBLE);
////
fragment.findViewById(R.id.frag_foreman_home_layout2).setVisibility(View.GONE);
////
// nav_header_home_layout.setVisibility(View.GONE);

```

```
////        }
////
////        } else{
////
////            new CommonMethod(ForemanDashboardActivity.this, "Server Error Code :
//
//"+response.code());
////
////        }
////
////
////        }
////
////
////        @Override
////
////        public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {
////
////            new CommonMethod(ForemanDashboardActivity.this, t.getMessage());
////
////        }
////
////    });

```

```
// TODO -----
// TODO      NavigationView Start
// TODO -----
```

NavigationView navigationView = (NavigationView)
findViewById(R.id.foreman_dashboard_nav_view);
View header = navigationView.getHeaderView(0);

nav_header_profileImage = header.findViewById(R.id.foreman_nav_header_profileImage);
nav_header_Name = header.findViewById(R.id.foreman_nav_header_Name);

String FirstName = sp.getString(SharedPreferencesData.FirstName, "");
String LastName = sp.getString(SharedPreferencesData.LastName, "");
nav_header_Name.setText(FirstName+" "+LastName);

nav_header_home_layout =
header.findViewById(R.id.foreman_nav_header_home_layout);
nav_header_foremanProfile_layout =
header.findViewById(R.id.foreman_nav_header_foremanProfile_layout);

```
nav_header_Services_layout =  
header.findViewById(R.id.foreman_nav_header_Service_layout);  
nav_header_userLocationTracking_layout =  
header.findViewById(R.id.foreman_nav_header_userLocationTracking_layout);  
nav_header_newReceiveRequest_layout =  
header.findViewById(R.id.foreman_nav_header_newReceiveRequest_layout);  
nav_header_rating_layout =  
header.findViewById(R.id.foreman_nav_header_rating_layout);  
nav_header_history_layout =  
header.findViewById(R.id.foreman_nav_header_history_layout);  
nav_header_setting_layout =  
header.findViewById(R.id.foreman_nav_header_setting_layout);  
nav_header_logout_layout =  
header.findViewById(R.id.foreman_nav_header_logout_layout);  
nav_header_switchAccount_layout =  
header.findViewById(R.id.foreman_nav_header_switchAccount_layout);  
  
nav_header_home_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_home_imageview_logo);  
nav_header_foremanProfile_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_foremanProfile_imageview_logo);  
nav_header_Services_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_Service_imageview_logo);  
nav_header_userLocationTracking_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_userLocationTracking_imageview_logo);  
nav_header_newReceiveRequest_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_newReceiveRequest_imageview_logo);  
nav_header_rating_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_rating_imageview_logo);  
nav_header_history_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_history_imageview_logo);
```

```
nav_header_setting_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_setting_imageview_logo);  
  
nav_header_logout_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_logout_imageview_logo);  
  
nav_header_switchAccount_imageview_logo =  
header.findViewById(R.id.foreman_nav_header_switchAccount_imageview_logo);  
  
nav_header_home_TextButton =  
header.findViewById(R.id.foreman_nav_header_home_TextButton);  
  
nav_header_foremanProfile_TextButton =  
header.findViewById(R.id.foreman_nav_header_foremanProfile_TextButton);  
  
nav_header_Services_TextButton =  
header.findViewById(R.id.foreman_nav_header_Service_TextButton);  
  
nav_header_userLocationTracking_TextButton =  
header.findViewById(R.id.foreman_nav_header_userLocationTracking_TextButton);  
  
nav_header_newReceiveRequest_TextButton =  
header.findViewById(R.id.foreman_nav_header_newReceiveRequest_TextButton);  
  
nav_header_rating_TextButton =  
header.findViewById(R.id.foreman_nav_header_rating_TextButton);  
  
nav_header_history_TextButton =  
header.findViewById(R.id.foreman_nav_header_history_TextButton);  
  
nav_header_setting_TextButton =  
header.findViewById(R.id.foreman_nav_header_setting_TextButton);  
  
nav_header_logout_TextButton =  
header.findViewById(R.id.foreman_nav_header_logout_TextButton);  
  
nav_header_switchAccount_TextButton =  
header.findViewById(R.id.foreman_nav_header_switchAccount_TextButton);  
  
// TODO Home Fragment  
nav_header_home_TextButton.setOnClickListener(new View.OnClickListener() {  
    @Override
```

```
public void onClick(View view) {
    Title_menu.setText("Home");
    drawer.closeDrawer(GravityCompat.START);
    FragmentManager manager = getSupportFragmentManager();

    manager.beginTransaction()
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHomeFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
}
```

```
nav_header_home_imageview_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Home");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHomeFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
nav_header_home_layout.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View view) {
    Title_menu.setText("Home");
    drawer.closeDrawer(GravityCompat.START);
    FragmentManager manager = getSupportFragmentManager();

    manager.beginTransaction()
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHomeFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
}

});
```

```
// TODO Foreman Services
nav_header_Services_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Services");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanServicesFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
});

nav_header_Services_imageview_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Services");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanServicesFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
nav_header_Services_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Services");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanServicesFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```

    }

});

// TODO ForemanProfile Fragment
nav_header_foremanProfile_TextButton.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Foreman Profile");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
        ForemanProfileFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});

nav_header_foremanProfile_imageview_logo.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Foreman Profile");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()

```

```

.replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanProfileFragment(), null)
.setReorderingAllowed(true)
.addToBackStack("")
.commit();
}

});

nav_header_foremanProfile_layout.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Title_menu.setText("Foreman Profile");
drawer.closeDrawer(GravityCompat.START);
FragmentManager manager = getSupportFragmentManager();

manager.beginTransaction()
.replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanProfileFragment(), null)
.setReorderingAllowed(true)
.addToBackStack("")
.commit();
}

});

// TODO UserLocationTracking Fragment
nav_header_userLocationTracking_TextButton.setOnClickListener(new
View.OnClickListener() {
@Override
public void onClick(View view) {
Title_menu.setText("Location Tracking");

```

```

        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanUsersLocationTrackingFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_userLocationTracking_imageview_logo.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("Location Tracking");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanUsersLocationTrackingFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_userLocationTracking_layout.setOnClickListener(new View.OnClickListener()
{

```

```
@Override
public void onClick(View view) {
    Title_menu.setText("Location Tracking");
    drawer.closeDrawer(GravityCompat.START);
    FragmentManager manager = getSupportFragmentManager();

    manager.beginTransaction()
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanUsersLocationTrackingFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
}

});
```

```
// TODO NewUserRequestReceive Fragment
nav_header_newReceiveRequest_TextButton.setOnClickListener(new
View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("User's Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
NewUserRequestReceiveFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```

    }
});

nav_header_newReceiveRequest_imageview_logo.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("User's Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
NewUserRequestReceiveFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});

```



```

nav_header_newReceiveRequest_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("User's Request");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
NewUserRequestReceiveFragment(), null)
            .setReorderingAllowed(true)

```

```

        .addToBackStack("")
        .commit();
    }

});

// TODO Rating Fragment
nav_header_rating_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Rating");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
RatingFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});

nav_header_rating_imageview_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Rating");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()

```

```
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
RatingFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
    }
});
```

```
nav_header_rating_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Rating");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
RatingFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }
});
```

```
// TODO History Fragment
nav_header_history_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("History");
        drawer.closeDrawer(GravityCompat.START);
```

```

FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHistoryFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_history_imageview_logo.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("History");
        drawer.closeDrawer(GravityCompat.START);
        FragmentManager manager = getSupportFragmentManager();

        manager.beginTransaction()
            .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHistoryFragment(), null)
            .setReorderingAllowed(true)
            .addToBackStack("")
            .commit();
    }

});

nav_header_history_layout.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View view) {
        Title_menu.setText("History");

```

```
drawer.closeDrawer(GravityCompat.START);
FragmentManager manager = getSupportFragmentManager();

manager.beginTransaction()
.replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanHistoryFragment(), null)
.setReorderingAllowed(true)
.addToBackStack("")
.commit();
}

});
```

```
// TODO Settings Fragment
nav_header_setting_TextButton.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
Title_menu.setText("Settings");
drawer.closeDrawer(GravityCompat.START);
FragmentManager manager = getSupportFragmentManager();

manager.beginTransaction()
.replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanSettingsFragment(), null)
.setReorderingAllowed(true)
.addToBackStack("")
.commit();
}

});
```

```
nav_header_setting_imageview_logo.setOnClickListener(new View.OnClickListener() {
```

```

@Override
public void onClick(View view) {
    Title_menu.setText("Settings");
    drawer.closeDrawer(GravityCompat.START);
    FragmentManager manager = getSupportFragmentManager();

    manager.beginTransaction()
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanSettingsFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
}

});

nav_header_setting_layout.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View view) {
    Title_menu.setText("Settings");
    drawer.closeDrawer(GravityCompat.START);
    FragmentManager manager = getSupportFragmentManager();

    manager.beginTransaction()
        .replace(R.id.foreman_content_main_dashboard_FragmentContainerView, new
ForemanSettingsFragment(), null)
        .setReorderingAllowed(true)
        .addToBackStack("")
        .commit();
}

});

```

```

// TODO Logout Fragment
nav_header_logout_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(ForemanDashboardActivity.this,
>LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

nav_header_logout_imageview_logo.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(ForemanDashboardActivity.this,
>LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

nav_header_logout_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Logout");
        sp.edit().clear().commit();
        startActivity(new Intent(ForemanDashboardActivity.this,
>LoginActivity.class).setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP));
    }
});

```

```
// TODO SwitchAccount Fragment
nav_header_switchAccount_TextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Switch Account");
        sp.edit().clear().commit();
        new CommonMethod(ForemanDashboardActivity.this, LoginActivity.class);
    }
});

nav_header_switchAccount_imageview_logo.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Switch Account");
        sp.edit().clear().commit();
        new CommonMethod(ForemanDashboardActivity.this, LoginActivity.class);
    }
});

nav_header_switchAccount_layout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Title_menu.setText("Switch Account");
        sp.edit().clear().commit();
        new CommonMethod(ForemanDashboardActivity.this, LoginActivity.class);
    }
});
```

```
// TODO If User Have Foreman Account  
nav_header_switchAccount_layout.setVisibility(View.GONE);  
String sUserEmail = sp.getString(SharedPreferencesData.Email, "");  
CheckUserISForeman(sUserEmail);
```

```
// TODO -----  
// TODO NavigationView Finished  
// TODO -----
```

```
}
```

```
private void CheckUserISForeman(String sUserEmail) {  
  
    // if(response.body().message.equalsIgnoreCase("Switch")){  
    //     nav_header_switchAccount_layout.setVisibility(View.VISIBLE);  
    // } else if (response.body().message.equalsIgnoreCase("NotSwitch")) {  
    //     nav_header_switchAccount_layout.setVisibility(View.GONE);  
    // }
```

```
}
```

```
@Override  
public void onBackPressed() {  
    super.onBackPressed();
```

```

DrawerLayout drawer = (DrawerLayout) findViewById(R.id.foreman_drawer_layout);
if (drawer.isDrawerOpen(GravityCompat.START)) {
    drawer.closeDrawer(GravityCompat.START);
} else {
    //super.onBackPressed();
    AlertDialog.Builder builder = new AlertDialog.Builder(ForemanDashboardActivity.this);
    builder.setTitle(getResources().getString(R.string.app_name));
    builder.setMessage("Are You Sure Want To Exit!");
    builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            finishAffinity();
        }
    });
    builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
        }
    });
    builder.show();
}
}

```

```

private void requestStoragePermission() {
    if (ContextCompat.checkSelfPermission(ForemanDashboardActivity.this,
        Manifest.permission.CALL_PHONE)
        != PackageManager.PERMISSION_GRANTED) {

```

```

        if (ActivityCompat.shouldShowRequestPermissionRationale

```

```

        (ForemanDashboardActivity.this, Manifest.permission.CALL_PHONE)) {

    Snackbar.make(ForemanDashboardActivity.this.findViewById(android.R.id.content),
        "Please Grant Permissions to Download photo",
        Snackbar.LENGTH_INDEFINITE).setAction("ENABLE",
        new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                requestPermissions(
                    new String[]{Manifest.permission
                        .CALL_PHONE},
                    STORAGE_PERMISSION_CODE);
            }
        }).show();
    } else {
        requestPermissions(
            new String[]{Manifest.permission
                .CALL_PHONE},
            STORAGE_PERMISSION_CODE);
    }
} else {
    // write your logic code if permission already granted
}
}

```

```

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_notification, menu);
    return true;
}

```

```
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        int id = item.getItemId();
        if (id == android.R.id.home) {
            onBackPressed();
        }
        if(id == R.id.notification_menu){
            new CommonMethod(ForemanDashboardActivity.this,
                ForemanNotificationActivity.class);
        }
        return super.onOptionsItemSelected(item);
    }
}
```

```
package com.brainybeam.roadsideassistance.Foreman.DashBoard;
```

```
import android.app.AlertDialog;
import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;

import androidx.cardview.widget.CardView;
import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.DefaultItemAnimator;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;

import
com.brainybeam.roadsideassistance.Foreman.Activity.ForemanHomePendingPaymentActivity;
import
com.brainybeam.roadsideassistance.Foreman.Activity.ForemanHomePendingPaymentAdapter;
import com.brainybeam.roadsideassistance.Foreman.CustomArrayList.PendingPaymentList;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.CartPendingPaymentData;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConnectionDetector;
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;

import java.util.ArrayList;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanUsersLocationTrackingFragment extends Fragment {

    RecyclerView recyclerView;
    CardView cardView;
    ArrayList<PendingPaymentList> arrayList;
    ApiInterface apiInterface;
```

```

SharedPreferences sp;
ProgressDialog pd;

public ForemanUsersLocationTrackingFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_foreman_users_location_tracking, container,
        false);

    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp          = getSharedPreferences(SharedPreferencesData.PREF,
        Context.MODE_PRIVATE);

    recyclerView      =
    view.findViewById(R.id.frag_foreman_user_location_tracking_RecyclerView);
    cardView          =
    view.findViewById(R.id.frag_foreman_user_location_tracking_CardViewNotHaveAnyPending
        Activity);

    cardView.setVisibility(View.GONE);

    recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
    recyclerView.setItemAnimator(new DefaultItemAnimator());

    if(new ConnectionDetector(getActivity()).isConnectingToInternet()){

```

```

pd = new ProgressDialog(getActivity());
pd.setMessage("Please Wait... ");
pd.setCancelable(false);
pd.show();

recyclerViewDataSetMethod();

} else {
    new ConnectionDetector(getActivity()).connectiondetect();
}

return view;
}

private void recyclerViewDataSetMethod() {

Call<CartPendingPaymentData> call = apiInterface.GetCartPendingPaymentData(
    sp.getString(SharedPreferencesData.UserID, ""))
);

call.enqueue(new Callback<CartPendingPaymentData>() {
    @Override
    public void onResponse(Call<CartPendingPaymentData> call,
    Response<CartPendingPaymentData> response) {

        pd.dismiss();
        if(response.code()==200){

            if(response.body().status==true){


```

```
arrayList = new ArrayList<>();
CartPendingPaymentData data = response.body();
for(int i=0; i<data.response.size(); i++){

    PendingPaymentList list = new PendingPaymentList();

    list.setCartID(data.response.get(i).cartID);
    list.setServiceID(data.response.get(i).serviceID);
    list.setForemanID(data.response.get(i).foremanID);
    list.setUserID(data.response.get(i).userID);
    list.setVehicleID(data.response.get(i).vehicleID);

    list.setProblemDescriptionMessage(data.response.get(i).problemDescriptionMessage);
    list.setSPReqMoney(data.response.get(i).sPReqMoney);
    list.setPaymentMode(data.response.get(i).paymentMode);
    list.setPayment(data.response.get(i).payment);
    list.setPaymentStatus(data.response.get(i).paymentStatus);
    list.setTypeOfProblem(data.response.get(i).typeOfProblem);
    list.setProblemSubType(data.response.get(i).problemSubType);
    list.setServiceFixedCharge(data.response.get(i).serviceFixedCharge);
    list.setUserFirstName(data.response.get(i).firstName);
    list.setUserLastName(data.response.get(i).lastName);
    list.setUserProfileImage(data.response.get(i).profileImage);
    list.setUserMobileNumber(data.response.get(i).mobileNumber);
    list.setNumberPlate_number(data.response.get(i).numberPlateNumber);
    list.setTypeOfVehicle(data.response.get(i).typeOfVehicle);
    list.setVehicleModelName(data.response.get(i).vehicleModelName);
    list.setVehicle_Colour(data.response.get(i).vehicleColour);
    list.setForemanLocationID(data.response.get(i).foremanLocationID);
    list.setForemanLatitude(data.response.get(i).foremanLatitude);
    list.setForemanLongitude(data.response.get(i).foremanLongitude);
```

```

        list.setUserLocationID(data.response.get(i).userLocationID);
        list.setUserLatitude(data.response.get(i).userLatitude);
        list.setUserLongitude(data.response.get(i).userLongitude);

        arrayList.add(list);
    }

    ForemanLocationTrackingAdapter adapter = new
    ForemanLocationTrackingAdapter(getActivity(), arrayList);
    recyclerView.setAdapter(adapter);
    adapter.notifyDataSetChanged();

} else {
    new CommonMethod(getActivity(), response.body().message());
    recyclerView.setVisibility(View.GONE);
    cardView.setVisibility(View.VISIBLE);
}

} else {
    new CommonMethod(getActivity(), "Server Error Code "+response.code());
}

}

@Override
public void onFailure(Call<CartPendingPaymentData> call, Throwable t) {
    new CommonMethod(getActivity(), t.getMessage());
}

});

}

}

```

```
}
```

Foreman profile Fragement:

```
package com.brainybeam.roadsideassistance.Foreman.DashBoard;

import static android.app.Activity.RESULT_OK;

import android.Manifest;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ActivityInfo;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationManager;
import android.net.Uri;
import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.fragment.app.Fragment;

import android.provider.MediaStore;
```

```
import android.provider.Settings;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.TextView;

import com.brainybeam.roadsideassistance.Foreman.Activity.ForemanProfileActivity;
import com.brainybeam.roadsideassistance.Foreman.Signup.ForemanSignupActivity;
import com.brainybeam.roadsideassistance.GPS.GPSTracker;
import com.brainybeam.roadsideassistance.Login.LoginActivity;
import com.brainybeam.roadsideassistance.OTPVerification.OTPVerificationActivity;
import com.brainybeam.roadsideassistance.R;
import com.brainybeam.roadsideassistance.RetrofitData.AddForemanServicesData;
import com.brainybeam.roadsideassistance.RetrofitData.DeleteUserORForemanData;
import com.brainybeam.roadsideassistance.RetrofitData.GetProfileImageData;
import com.brainybeam.roadsideassistance.RetrofitData.LocationData;
import com.brainybeam.roadsideassistance.RetrofitData.UpdateForemanData;
import com.brainybeam.roadsideassistance.RetrofitData.UpdateUserData;
import com.brainybeam.roadsideassistance.User.Activity.UserForemanServicesActivity;
import com.brainybeam.roadsideassistance.Utils.ApiClient;
import com.brainybeam.roadsideassistance.Utils.ApiInterface;
import com.brainybeam.roadsideassistance.Utils.CommonMethod;
import com.brainybeam.roadsideassistance.Utils.ConstantData;
import com.brainybeam.roadsideassistance.Utils.FilePath;
```

```
import com.brainybeam.roadsideassistance.Utils.SharedPreferencesData;
import com.google.firebase.FirebaseApp;
import com.google.firebase.FirebaseException;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.PhoneAuthCredential;
import com.google.firebase.auth.PhoneAuthOptions;
import com.google.firebase.auth.PhoneAuthProvider;
import com.squareup.picasso.Picasso;
//import com.zhihu.matisse.Matisse;
//import com.zhihu.matisse.MimeType;
//import com.zhihu.matisse.engine.impl.GlideEngine;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.concurrent.TimeUnit;

import de.hdodenhof.circleimageview.CircleImageView;
import okhttp3.MediaType;
import okhttp3.MultipartBody;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class ForemanProfileFragment extends Fragment {
```

```
LinearLayout layout1, layout2;
CircleImageView ProfileImage1, ProfileImage2;
TextView FirstName, LastName, Contact, Email, Address, Area, City, State, AccountStatus;
EditText FirstName_EditText, LastName_EditText, Contact_EditText, Email_EditText,
Password_EditText, Address_EditText, Area_EditText, City_EditText;
Spinner Spinner_State;
Button EditTextButton, UpdateButton, DeactivateButton1, DeactivateButton2;

ArrayList<String> arrayList;
String sFirstName, sLastName, sContact, sEmail, sPassword, sAddress, sArea, sCity, sState;

ApiClient apiInterface;
SharedPreferences sp;
ProgressDialog pd;
Bundle bundle;

FirebaseApp firebaseApp;
private FirebaseAuth mAuth;
private PhoneAuthProvider.OnVerificationStateChangedCallbacks mCallbacks;
private String sPhone;

private String EmailPattern = "[a-zA-Z0-9._-]+@[a-z]+\.\+[a-z]+";

String[] appPermission = {Manifest.permission.READ_EXTERNAL_STORAGE};
private static final int PERMISSION_REQUEST_CODE = 1240;
private static final int PICK_IMAGE_REQUEST = 12;
String sImagePath1 = "";
String sImagePath2 = "";
```

```

String getImageURL = "";
double DriverLatitude, DriverLongitude;
String sFullAddress, sForemanLatitude, sForemanLongitude;

private static final int REQUEST_LOCATION = 1;
LocationManager locationManager;

GPSTracker gpsTracker;

public ForemanProfileFragment() {
    // Required empty public constructor
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_foreman_profile, container, false);

    firebaseApp = FirebaseApp.initializeApp(getActivity());
    mAuth = FirebaseAuth.getInstance();

    // onRequestPermissionsResult(PERMISSION_REQUEST_CODE, appPermission, );
    apiInterface = ApiClient.getClient().create(ApiInterface.class);
    sp          = getActivity().getSharedPreferences(SharedPreferencesData.PREF,
Context.MODE_PRIVATE);

    layout1 = view.findViewById(R.id.frag_foreman_profile_layout1);
    layout2 = view.findViewById(R.id.frag_foreman_profile_layout2);
    ProfileImage1 = view.findViewById(R.id.frag_foreman_profile_foremanProfileImage);

```

```

ProfileImage2 = view.findViewById(R.id.frag_foreman_profile_userProfileImage2);
FirstName = view.findViewById(R.id.frag_foreman_profile_FirstName);
LastName = view.findViewById(R.id.frag_foreman_profile_LastName);
Contact = view.findViewById(R.id.frag_foreman_profile_MobileNumber);
Email = view.findViewById(R.id.frag_foreman_profile_Email);
Address = view.findViewById(R.id.frag_foreman_profile_Address);
Area = view.findViewById(R.id.frag_foreman_profile_Area);
City = view.findViewById(R.id.frag_foreman_profile_City);
State = view.findViewById(R.id.frag_foreman_profile_State);
AccountStatus = view.findViewById(R.id.frag_foreman_profile_AccountStatus);
FirstName_EditText = view.findViewById(R.id.frag_foreman_profile_FirstNameEditText);
LastName_EditText = view.findViewById(R.id.frag_foreman_profile_LastNameEditText);
Contact_EditText = view.findViewById(R.id.frag_foreman_profile_MobileNumberEditText);
Email_EditText = view.findViewById(R.id.frag_foreman_profile_EmailEditText);
Password_EditText = view.findViewById(R.id.frag_foreman_profile_PasswordEditText);
Address_EditText = view.findViewById(R.id.frag_foreman_profile_AddressEditText);
Area_EditText = view.findViewById(R.id.frag_foreman_profile_AreaEditText);
City_EditText = view.findViewById(R.id.frag_foreman_profile_CityEditText);
Spinner_State = view.findViewById(R.id.frag_foreman_profile_SpinnerState);
EditTextButton = view.findViewById(R.id.frag_foreman_profile_EditProfileButton);
UpdateButton = view.findViewById(R.id.frag_foreman_profile_UpdateButton);
DeactivateButton1 = view.findViewById(R.id.frag_foreman_profile_deactivateButton);
DeactivateButton2 = view.findViewById(R.id.frag_foreman_profile_deactivateButton2);

layout2.setVisibility(View.GONE);

```

```
gpsTracker = new GPSTracker(getActivity());
```

```
// Check if GPS enabled
if(gpsTracker.canGetLocation()) {
```

```

//      CurrentLatitude = gpsTracker.getLatitude();
//      CurrentLongitude = gpsTracker.getLongitude();

//getLocation();

} else {
    // Can't get location.
    // GPS or network is not enabled.
    // Ask user to enable GPS/network in settings.
    gpsTracker.showSettingsAlert();
}

// TODO Get Profile Image
getUserImage();
if(sp.getString(SharedPreferencesData.ProfileImage, "").isEmpty() ||
    sp.getString(SharedPreferencesData.ProfileImage, "").equalsIgnoreCase("") ||
    sp.getString(SharedPreferencesData.ProfileImage,
    "").equalsIgnoreCase("ProfileImage")){
}

} else {
    Picasso.with(getActivity()).load(sp.getString(SharedPreferencesData.ProfileImage,
    "")).placeholder(R.drawable.ic_profile).into(ProfileImage1);
}

FirstName.setText(sp.getString(SharedPreferencesData.FirstName, ""));
LastName.setText(sp.getString(SharedPreferencesData.LastName, ""));
Contact.setText(sp.getString(SharedPreferencesData.MobileNumber, ""));
Email.setText(sp.getString(SharedPreferencesData.Email, ""));
Address.setText(sp.getString(SharedPreferencesData.ForemanAddress, ""));
Area.setText(sp.getString(SharedPreferencesData.ForemanArea, ""));

```

```

City.setText(sp.getString(SharedPreferencesData.ForemanCity, ""));
State.setText(sp.getString(SharedPreferencesData.ForemanState, ""));
AccountStatus.setText(sp.getString(SharedPreferencesData.Account_Status, ""));

EditTextButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        layout2.setVisibility(View.VISIBLE);
        layout1.setVisibility(View.GONE);
    }
});

sState = "";
arrayList = new ArrayList<>();
arrayList.add("Select State");
String S_array[] = ConstantData.State;
List<String> list;
list = Arrays.asList(S_array);
arrayList.addAll(list);

Spinner_State.setSelection(arrayList.indexOf("Select State"));
ArrayAdapter adapter = new ArrayAdapter(getActivity(),
        android.R.layout.simple_list_item_1, arrayList);
adapter.setDropDownViewResource(android.R.layout.simple_list_item_activated_1);
Spinner_State.setAdapter(adapter);

Spinner_State.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
        sState = arrayList.get(i);
    }
})

```

```
@Override  
public void onNothingSelected(AdapterView<?> adapterView) {  
  
}  
});
```

```
UpdateButton.setOnClickListener(new View.OnClickListener() {  
  
    @Override  
    public void onClick(View view) {  
        sFirstName = FirstName_EditText.getText().toString();  
        sLastName = LastName_EditText.getText().toString();  
        sContact = Contact_EditText.getText().toString();  
        sEmail = Email_EditText.getText().toString();  
        sPassword = Password_EditText.getText().toString();  
        sAddress = Address_EditText.getText().toString();  
        sArea = Area_EditText.getText().toString();  
        sCity = City_EditText.getText().toString();  
  
        if(sFirstName.isEmpty() || sFirstName.equalsIgnoreCase("")){  
            FirstName_EditText.setError("FirstName is Required");  
        } else if(sLastName.isEmpty() || sLastName.equalsIgnoreCase("")){  
            LastName_EditText.setError("LastName is Required");  
        } else if(sContact.isEmpty() || sContact.equalsIgnoreCase("")){  
            Contact_EditText.setError("Mobile number is Required");  
        } else if(sContact.length()>10 || sContact.length()<10){  
            Contact_EditText.setError("Mobile number is Not valid");  
        }  
        else if(sEmail.isEmpty() || sEmail.equalsIgnoreCase("")){  
            Email_EditText.setError("Email is Required");  
        }
```

```

} else if(!sEmail.matches>EmailPattern){
    Email_EditText.setError("Valid Email is Required");
} else if(sPassword.isEmpty() || sPassword.equalsIgnoreCase("")){
    Password_EditText.setError("Password is Required");
} else if(sPassword.length()<8){
    Password_EditText.setError("Password must be 8 char long");
} else if(sAddress.isEmpty() || sAddress.equalsIgnoreCase("")){
    Address_EditText.setError("Address is Required");
} else if(sArea.isEmpty() || sArea.equalsIgnoreCase("")){
    Area_EditText.setError("Area is Required");
} else if(sCity.isEmpty() || sCity.equalsIgnoreCase("")){
    City_EditText.setError("City is Required");
} else if(sState.isEmpty() || sState.equalsIgnoreCase("")){
    new CommonMethod(getActivity(), "State is Required");
} else if(sState.equalsIgnoreCase("Select State")){
    new CommonMethod(getActivity(), "Not Valid State");
} else {

    pd = new ProgressDialog(getActivity());
    pd.setMessage("Please Wait...");
    pd.setCancelable(false);
    pd.show();

    UpdateUserProfileImageData();
    sp.edit().putString(SharedPreferencesData.ProfileImage, sImagePath2).commit();
    UpdateUserProfileData();
    UpdateSPPProfileLocationData();

}

```

```
});
```

```
ProfileImage1.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        selectImageMethod();
```

```
    }
```

```
});
```

```
ProfileImage2.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        selectImageMethod();
```

```
    }
```

```
});
```

```
DeactivateButton1.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```
        pd = new ProgressDialog(getActivity());
```

```
        pd.setMessage("Please Wait...");
```

```
        pd.setCancelable(false);
```

```
        pd.show();
```

```
        DeactivateUserAccount();
```

```
    }
```

```
});
```

```
DeactivateButton2.setOnClickListener(new View.OnClickListener() {
```

```
    @Override
```

```
    public void onClick(View view) {
```

```

        pd = new ProgressDialog(getActivity());
        pd.setMessage("Please Wait...");
        pd.setCancelable(false);
        pd.show();

        DeactivateUserAccount();
    }

});

return view;
}

private void UpdateSPPProfileLocationData() {

    String FullAddress = sp.getString(SharedPreferencesData.ForemanAddress,
    "")+","+sp.getString(SharedPreferencesData.ForemanArea,
    "")+","+sp.getString(SharedPreferencesData.ForemanCity,
    "")+","+sp.getString(SharedPreferencesData.ForemanState, "");

    Geocoder coder = new Geocoder(getActivity());
    List<Address> address;
    try {
        // May throw an IOException
        address = coder.getFromLocationName(FullAddress, 5);
        if (address == null) {
            DriverLatitude = 0.00;
            DriverLongitude = 0.00;
        }
        Address location = address.get(0);

        DriverLatitude = location.getLatitude();
    }
}

```

```

DriverLongitude = location.getLongitude();
//p1 = new LatLng(location.getLatitude(), location.getLongitude() );

} catch (IOException ex) {

    ex.printStackTrace();
}

sForemanLatitude = String.valueOf(DriverLatitude);
sForemanLongitude = String.valueOf(DriverLongitude);

Call<LocationData> call = apiInterface.UpdateSPPProfileLocationData(
    sp.getString(SharedPreferencesData.UserID, ""),
    sForemanLatitude,
    sForemanLongitude
);

call.enqueue(new Callback<LocationData>() {
    @Override
    public void onResponse(Call<LocationData> call, Response<LocationData> response) {

        if(response.code()==200){

            if(response.body().status==true){

                } else {
                    new CommonMethod(getActivity(), response.body().message);
                }
            } else {
        }
    }
});

```

```

        new CommonMethod(getActivity(), "Server Error Code "+response.code());
    }

}

@Override
public void onFailure(Call<LocationData> call, Throwable t) {
    new CommonMethod(getActivity(), t.getMessage());
}
});

}

private void getUserImage() {

Call<GetProfileImageData> call = apiInterface.GetForemanProfileImage(
    sp.getString(SharedPreferencesData.UserID, ""))
);

call.enqueue(new Callback<GetProfileImageData>() {
    @Override
    public void onResponse(Call<GetProfileImageData> call,
    Response<GetProfileImageData> response) {
        if(response.code()==200){
            if(response.body().status==true){
                // new CommonMethod(getActivity(),response.body().message);

                GetProfileImageData data = response.body();
                for(int i=0; i<data.response.size(); i++){
                    imageURL = data.response.get(i).profileImage;
                }
            }
        }
    }
});

```

```

        sp.edit().putString(SharedPreferencesData.ProfileImage,
getImageURL).commit();
    }
    else{
        new CommonMethod(getActivity(),response.body().message);
    }
}
else{
    new CommonMethod(getActivity(),"Server Error Code : "+response.code());
}
}

@Override
public void onFailure(Call<GetProfileImageData> call, Throwable t) {
    new CommonMethod(getActivity(),t.getMessage());
}
});

}

private void getLocation() {
    if (ActivityCompat.checkSelfPermission(
        getActivity(), Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(
        getActivity(), Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(getActivity(),
new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_LOCATION);
    } else {

```

```

Location locationGPS = null;
locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

if (locationGPS != null) {
    double lat = locationGPS.getLatitude();
    double longi = locationGPS.getLongitude();
    DriverLatitude = lat;
    DriverLongitude = longi;
} else {
    new CommonMethod(getActivity(), "Not Found");
}
}
}
}

```

```

private void UpdateUserProfileImageData() {

    RequestBody namePart = RequestBody.create(MultipartBody.FORM,
sp.getString(SharedPreferencesData.FirstName, ""));

    File imageFile = new File(sImagePath2);
    RequestBody imageBody = RequestBody.create(MediaType.parse("image/*"), imageFile);
    MultipartBody.Part imagesParts = MultipartBody.Part.createFormData("file",
imageFile.getName(), imageBody);

    Call<DeleteUserORForemanData> call = apiInterface.AddForemanProfileImage(
sp.getString(SharedPreferencesData.UserID, ""), namePart, imagesParts
);
    call.enqueue(new Callback<DeleteUserORForemanData>() {
        @Override
        public void onResponse(Call<DeleteUserORForemanData> call,
Response<DeleteUserORForemanData> response) {

```

```

pd.dismiss();

if(response.code()==200){

    if(response.body().status==true){

        new CommonMethod(getActivity(),response.body().message);

    }

    else{

        new CommonMethod(getActivity(),response.body().message);

    }

}

else{

    new CommonMethod(getActivity(),"Server Error Code : "+response.code());

}

}

@Override

public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {

    pd.dismiss();

    new CommonMethod(getActivity(),t.getMessage());

}

});

}

private void DeactivateUserAccount() {

Call<DeleteUserORForemanData> call = apiInterface.DeactivateForemanAccountData(
    sp.getString(SharedPreferencesData.UserID, "")

);

call.enqueue(new Callback<DeleteUserORForemanData>() {

    @Override

```

```

public void onResponse(Call<DeleteUserORForemanData> call,
Response<DeleteUserORForemanData> response) {
    pd.dismiss();
    if(response.code()==200){

        if (response.body().status==true){

            new CommonMethod(getActivity(), "Your Account SuccessFully Deactivate");
            new CommonMethod(getActivity(), LoginActivity.class);

        } else {
            new CommonMethod(getActivity(), response.body().message);
        }

    } else {
        new CommonMethod(getActivity(), "Server Error Code "+response.code());
    }

}

@Override
public void onFailure(Call<DeleteUserORForemanData> call, Throwable t) {
    pd.dismiss();
    new CommonMethod(getActivity(), t.getMessage());
}

private void UpdateUserProfileData() {

```

```

Call<UpdateForemanData> call = apiInterface.UpdateForemanData(
    sp.getString(SharedPreferencesData.UserID, ""),
    sFirstName, sLastName, sContact, sEmail, sPassword,
    sAddress, sArea, sCity, sState
);

call.enqueue(new Callback<UpdateForemanData>() {
    @Override
    public void onResponse(Call<UpdateForemanData> call,
    Response<UpdateForemanData> response) {
        pd.dismiss();
        if(response.code()==200){

            if(response.body().status.equalsIgnoreCase("True")){
                new CommonMethod(getActivity(), response.body().message);

                sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).commit();
                sp.edit().putString(SharedPreferencesData.LastName, sLastName).commit();
                sp.edit().putString(SharedPreferencesData.MobileNumber, sContact).commit();
                sp.edit().putString(SharedPreferencesData.Email, sEmail).commit();
                sp.edit().putString(SharedPreferencesData.Password, sPassword).commit();
                sp.edit().putString(SharedPreferencesData.ForemanAddress, sAddress).commit();
                sp.edit().putString(SharedPreferencesData.ForemanArea, sArea).commit();
                sp.edit().putString(SharedPreferencesData.ForemanCity, sCity).commit();
                sp.edit().putString(SharedPreferencesData.ForemanState, sState).commit();

                FirstName.setText(sp.getString(SharedPreferencesData.FirstName, ""));
                LastName.setText(sp.getString(SharedPreferencesData.LastName, ""));
                Contact.setText(sp.getString(SharedPreferencesData.MobileNumber, ""));
                Email.setText(sp.getString(SharedPreferencesData.Email, ""));
            }
        }
    }
});

```

```

Address.setText(sp.getString(SharedPreferencesData.ForemanAddress, ""));
Area.setText(sp.getString(SharedPreferencesData.ForemanArea, ""));
City.setText(sp.getString(SharedPreferencesData.ForemanCity, ""));
State.setText(sp.getString(SharedPreferencesData.ForemanState, ""));
AccountStatus.setText(sp.getString(SharedPreferencesData.Account_Status, ""));

layout1.setVisibility(View.VISIBLE);
layout2.setVisibility(View.GONE);

} else if(response.body().status.equalsIgnoreCase("Pending")){
    new CommonMethod(getActivity(), response.body().message);

    sp.edit().putString(SharedPreferencesData.FirstName, sFirstName).commit();
    sp.edit().putString(SharedPreferencesData.LastName, sLastName).commit();
    sp.edit().putString(SharedPreferencesData.MobileNumber, sContact).commit();
    sp.edit().putString(SharedPreferencesData.Email, sEmail).commit();
    sp.edit().putString(SharedPreferencesData.Password, sPassword).commit();
    sp.edit().putString(SharedPreferencesData.ForemanAddress, sAddress).commit();
    sp.edit().putString(SharedPreferencesData.ForemanArea, sArea).commit();
    sp.edit().putString(SharedPreferencesData.ForemanCity, sCity).commit();
    sp.edit().putString(SharedPreferencesData.ForemanState, sState).commit();

    sPhone = sContact;
    otpSendToMobile(sPhone);

} else {
    new CommonMethod(getActivity(), response.body().message);
}

```

```
        } else {
            new CommonMethod(getActivity(), "Server Error Code "+response.code());
        }

    }

@Override
public void onFailure(Call<UpdateForemanData> call, Throwable t) {
    pd.dismiss();
    new CommonMethod(getActivity(), t.getMessage());
}
});
```

```
}

private void otpSendToMobile(String sPhone) {

    mCallbacks = new PhoneAuthProvider.OnVerificationStateChangedCallbacks() {

        @Override
        public void onVerificationCompleted(PhoneAuthCredential credential) {

        }

        @Override
        public void onVerificationFailed(FirebaseException e) {
            pd.dismiss();
            new CommonMethod(getActivity(), e.getLocalizedMessage());
        }
    }
}
```

```

@Override
public void onCodeSent(@NonNull String VerificationId,
                      @NonNull PhoneAuthProvider.ForceResendingToken token) {
    pd.dismiss();

    new CommonMethod(getActivity(), "OTP is successFully Send");

    bundle.putString("PhoneNumber", sPhone.trim());
    bundle.putString("Mobile_VerificationID", VerificationId);
    Intent intent = new Intent(getActivity(), OTPVerificationActivity.class);
    intent.putExtras(bundle);
    startActivity(intent);
}

};

PhoneAuthOptions options =
    PhoneAuthOptions.newBuilder(mAuth)
        .setPhoneNumber("+880" + sPhone.trim())
        .setTimeout(60L, TimeUnit.SECONDS)
        .setActivity(getActivity())
        .setCallbacks(mCallbacks)
        .build();

PhoneAuthProvider.verifyPhoneNumber(options);

}

private void selectImageMethod() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
}

```

```

        startActivityForResult(Intent.createChooser(intent, "Select Pdf"),
PICK_IMAGE_REQUEST);
    }

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode==PICK_IMAGE_REQUEST && resultCode==RESULT_OK && data
!=null){
        /*Uri uri = data.getData();
        Log.d("RESPONSE_URI", String.valueOf(uri));
        imageView.setImageURI(uri);*/
        sImagePath1 = filePath.getPath(getActivity(), data.getData());
        sImagePath2 = filePath.getPath(getActivity(), data.getData());
        // Log.d("RESPONSE_IMAGE_PATH",sImagePath);
        // fileText.setVisibility(View.VISIBLE);
        // uploadButton.setVisibility(View.VISIBLE);
    }
}

public boolean checkAndRequestPermission() {
    List<String> listPermission = new ArrayList<>();
    for (String perm : appPermission) {
        if (ContextCompat.checkSelfPermission(getActivity(), perm) != PackageManager.PERMISSION_GRANTED) {
            listPermission.add(perm);
        }
    }
    if (!listPermission.isEmpty()) {
        ActivityCompat.requestPermissions(getActivity(), listPermission.toArray(new
String[listPermission.size()]), PERMISSION_REQUEST_CODE);
        return false;
    }
}

```

```

    }

    return true;
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == PERMISSION_REQUEST_CODE) {
        HashMap<String, Integer> permissionResult = new HashMap<>();
        int deniedCount = 0;
        for (int i = 0; i < grantResults.length; i++) {
            if (grantResults[i] == PackageManager.PERMISSION_DENIED) {
                permissionResult.put(permissions[i], grantResults[i]);
                deniedCount++;
            }
        }
        if (deniedCount == 0) {
            selectImageMethod();
        } else {
            for (Map.Entry<String, Integer> entry : permissionResult.entrySet()) {
                String permName = entry.getKey();
                int permResult = entry.getValue();
                if (ActivityCompat.shouldShowRequestPermissionRationale(
                    getActivity(),
                    permName)) {
                    showDialogPermission("", "This App needs Read External Storage permissions to
work without any problems.",
                    "Yes, Grant permissions", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {
                            dialogInterface.dismiss();

```

```

        checkAndRequestPermission());
    }
},
"No, Exit app", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.dismiss();
        getActivity().finishAffinity();
    }
}, false);
} else {
    showDialogPermission("", "You have denied some permissions. Allow all
permissions at [Setting] > [Permissions]",
    "Go to Settings", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialogInterface, int i) {
            dialogInterface.dismiss();
            Intent intent = new
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS,
        Uri.fromParts("package", getActivity().getPackageName(), null));
            intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(intent);
            getActivity().finish();
        }
}, "No, Exit app", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        dialogInterface.dismiss();
        getActivity().finish();
    }
}, false);
}

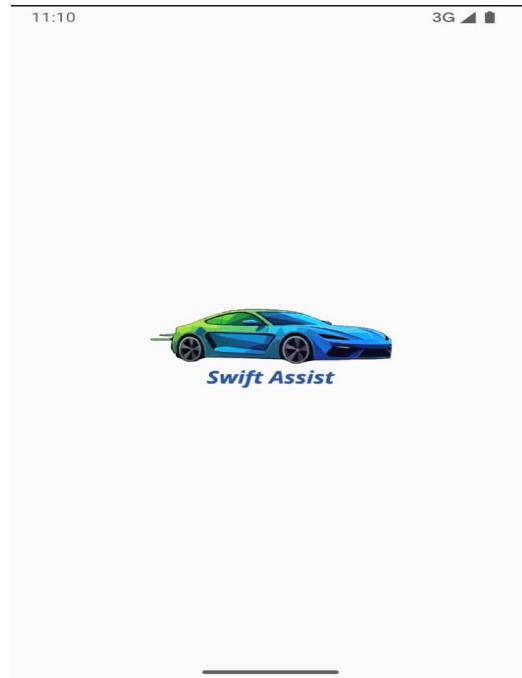
```

```

        break;
    }
}

public AlertDialog showDialogPermission(String title, String msg, String positiveLable,
DialogInterface.OnClickListener positiveOnClickListener, String negativeLable,
DialogInterface.OnClickListener negativeOnClickListener, boolean isCancelable) {
    AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());
    builder.setTitle(title);
    builder.setCancelable(isCancelable);
    builder.setMessage(msg);
    builder.setPositiveButton(positiveLable, positiveOnClickListener);
    builder.setNegativeButton(negativeLable, negativeOnClickListener);
    AlertDialog alertDialog = builder.create();
    alertDialog.show();
    return alertDialog;
}

```



1.22.1Physical design

UI/UX:

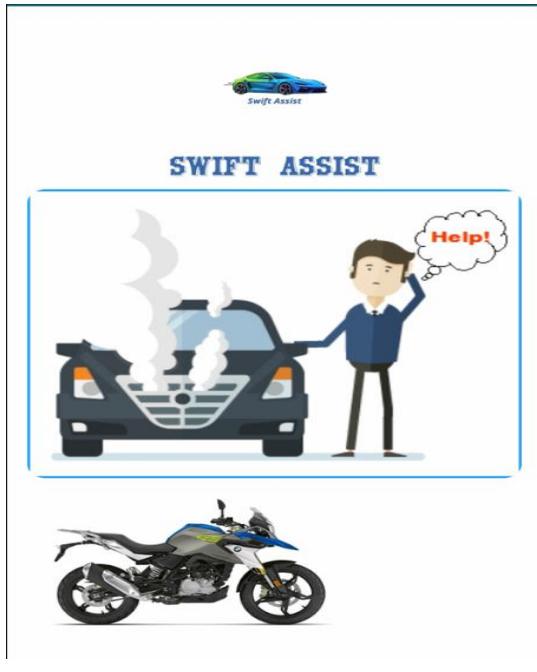


Fig no. 6: Introduction2

The image shows a login/signup form. At the top, there is a header "Login/SignIn" with icons for user and device. Below the header are two input fields: "Email" and "Password", each with a clear button. There is also a "Forgot Password" link. A large blue "Login" button is at the bottom. At the very bottom, there is a link "Don't have an account yet? [SignUp](#)".

Figno. 7 : login/signup

The image shows a "SignUp" form. It consists of five input fields: "First Name", "Last Name", "Mobile Number", "Email", and "Password". The "Password" field includes an "eye" icon for password visibility. Below the input fields is a large blue "Get OTP" button.

Fig no.8 : Signup page

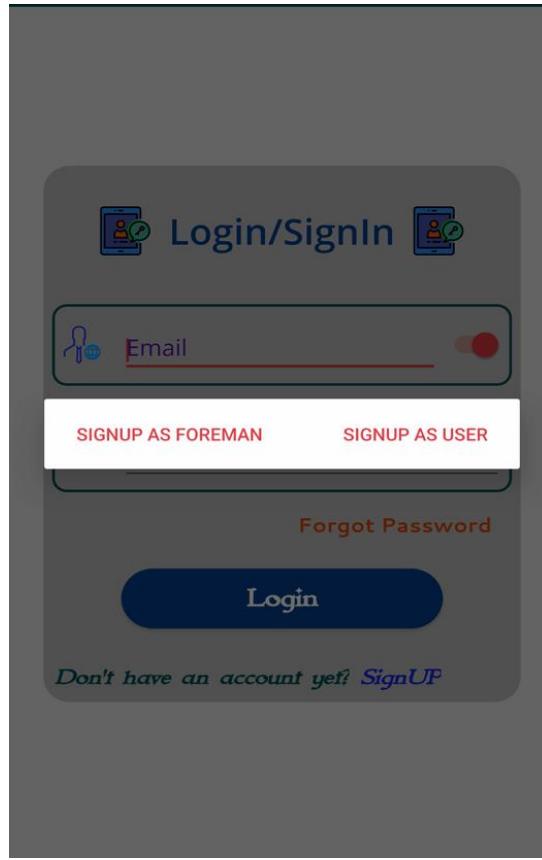


Fig no. 9: SignUp Option Select

The image shows a "SignUp" form. It consists of several input fields: "First Name", "Last Name", "Mobile Number", "Email", "Password" (with an eye icon), "Address", "Area", "City", and a dropdown menu. At the bottom is a large blue "Login" button.

Fig no. 10: SignUp Information

The image shows an "OTP" verification screen. At the top, there is a field labeled "Phone Number" with a placeholder icon of a person and a globe. To the right of the field is a toggle switch. Below the phone number field is a field labeled "OTP" with a placeholder icon of a smartphone. At the bottom are two blue buttons: "Get OTP" on the left and "Login" on the right. Below these buttons is a numeric keypad. The keypad includes digits 1 through 9, 0, *, #, and special characters . and -. The number "8888888888" is entered into the OTP field. The text "Supti1800" is displayed above the keypad.

Fig no. 11: OTP Verification

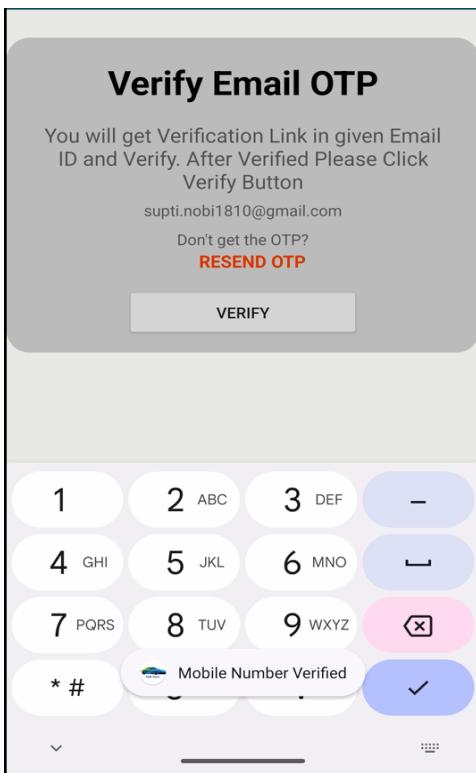


Fig no. 12: OTP Verification 2

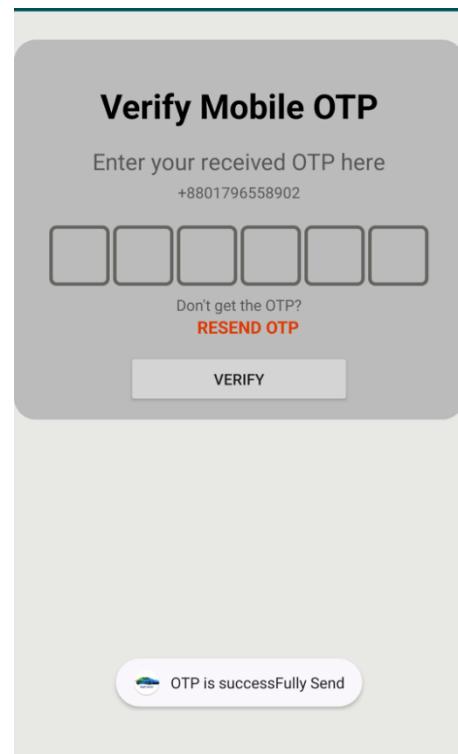


Fig no. 13: OTP Verification 3

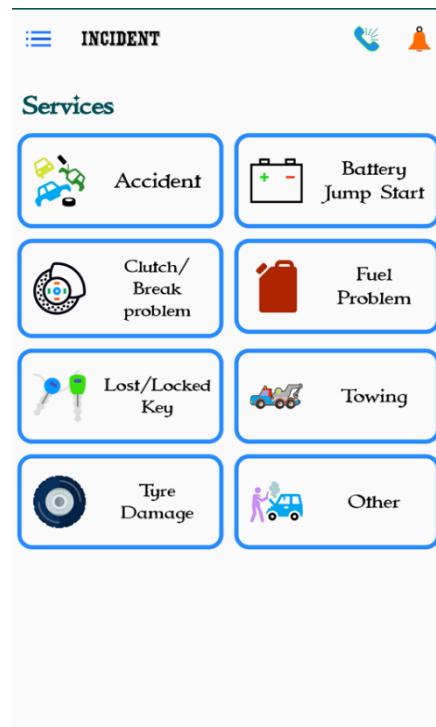


Fig no. 14: User Dashboard

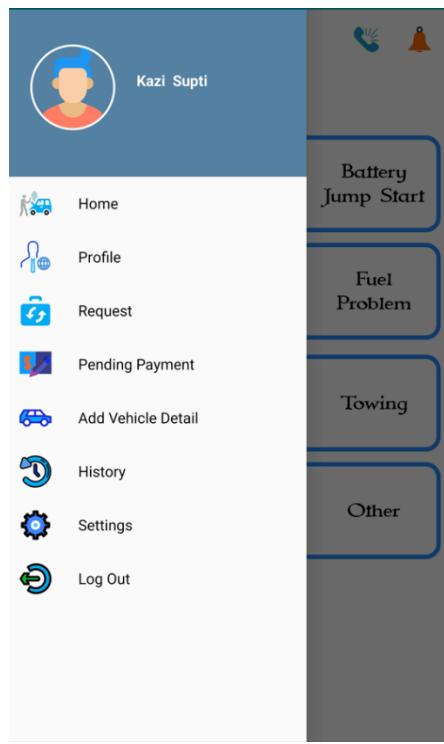


Fig no. 15: User Mapbar

A user profile screen titled "PROFILE". It features a large circular profile picture of a person with blue hair and an orange shirt. Below the picture is the title "User Profile". Underneath are five text input fields: "FirstName Kazi", "LastName Supti", "Contact 1796558902", "Email supti.nobil810@gmail.com", and "Account Status Verified". At the bottom is a button labeled "Edit Profile" with a blue gradient background and white text, accompanied by a small circular icon.

Fig no. 16: User Profile

A screen titled "Add Vehicle" with a back arrow. It has a large blue circular placeholder at the top. Below it is a title "Add Your Vehicle". There are four text input fields: "Vehicle Plate Number Ex.GJ01AB1234", "Select Type Of Vehicle" (with a dropdown arrow), "Model Name", and "Vehicle Color". At the bottom are two blue rounded rectangular buttons labeled "Add" and "Back".

Fig no. 17: Add User Vehicle Details

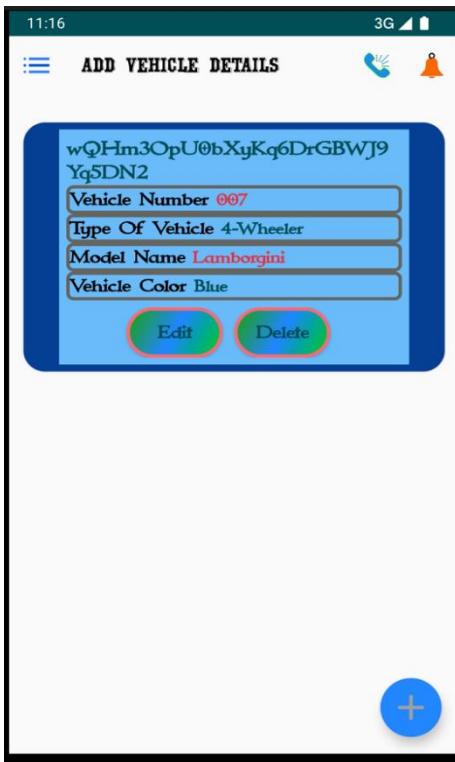


Fig no. 18: User Vehicle Details

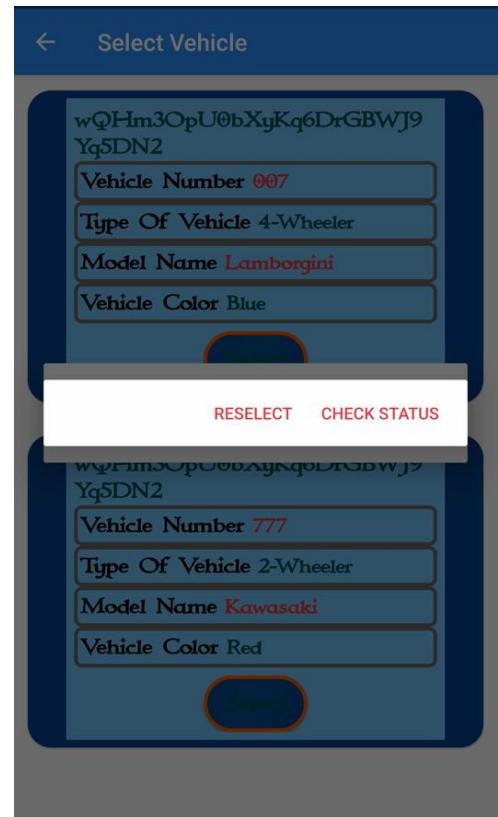


Fig no. 19: Select Vehicle Details

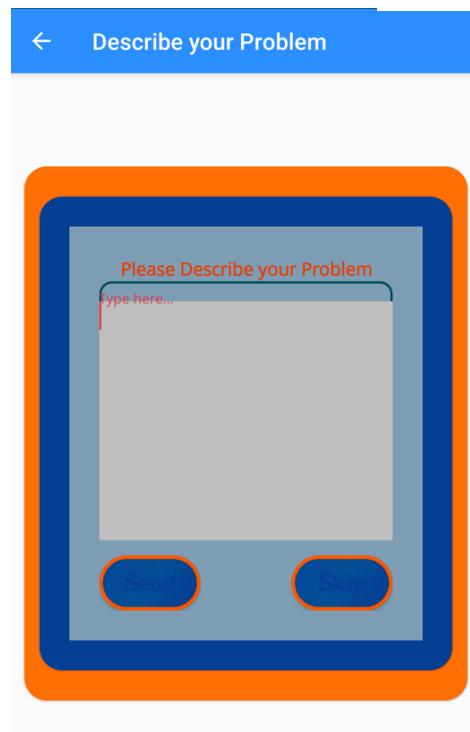


Fig no. 20: Vehicle Problem Details

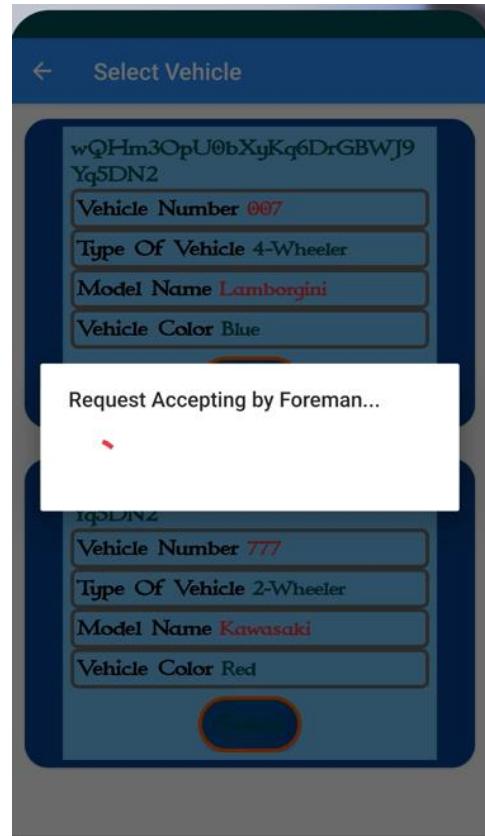


Fig no. 21: Requesting Foreman Service

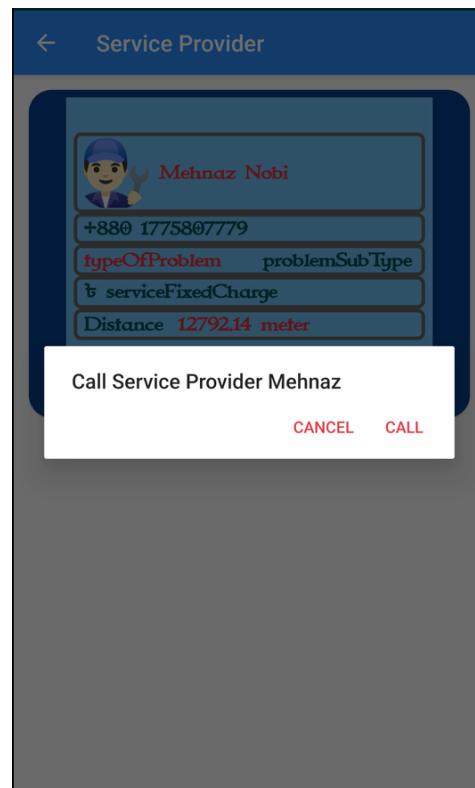


Fig no. 22: Call Foreman

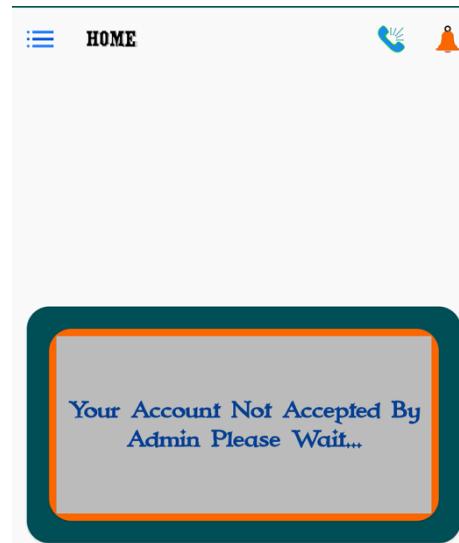


Fig no. 23: Foreman homepage before Admin accepts request

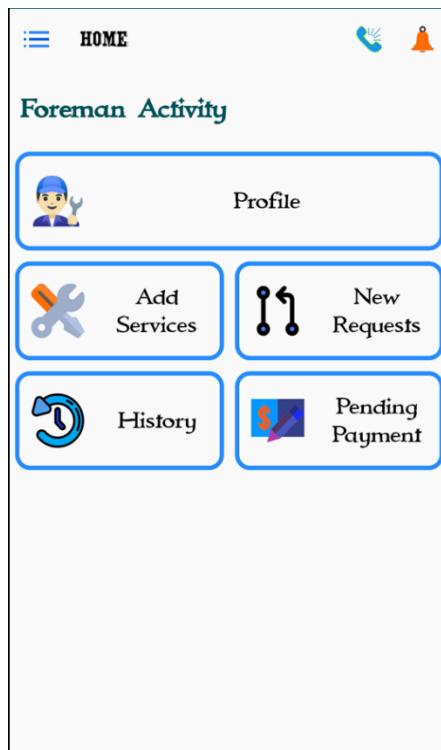


Fig no. 23: Foreman homepage After Admin accepts request

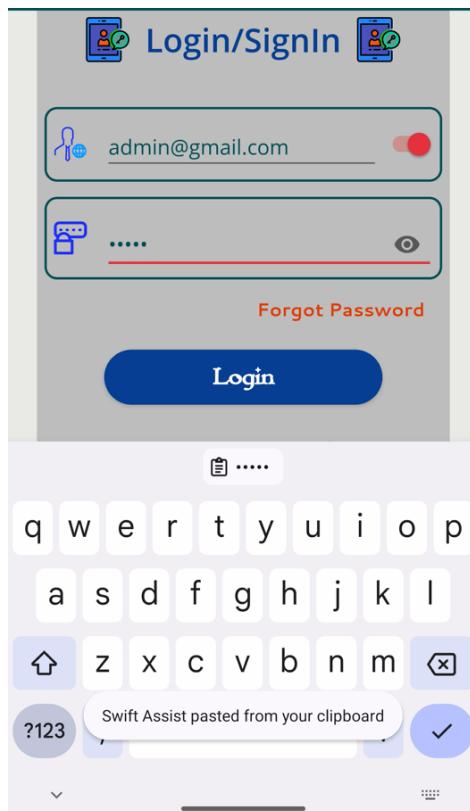


Fig no. 24: Admin Login Page

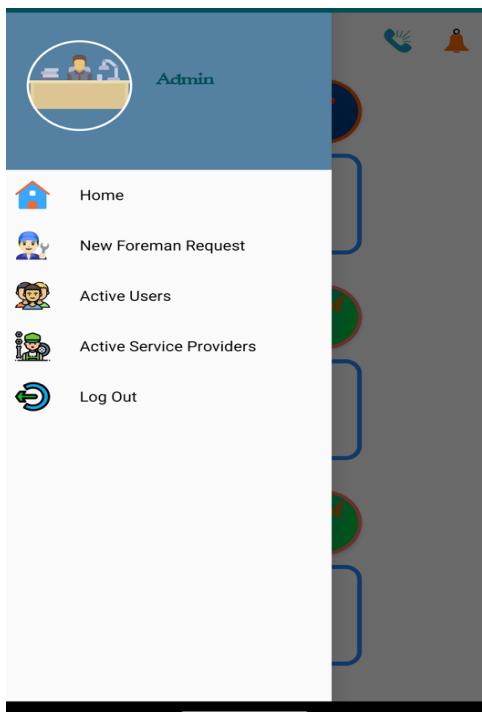


Fig no. 25: Admin Mapbar

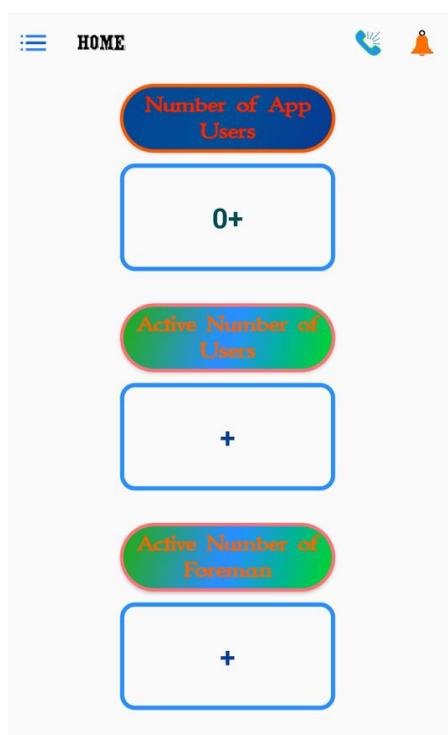


Fig no. 26: Admin home page

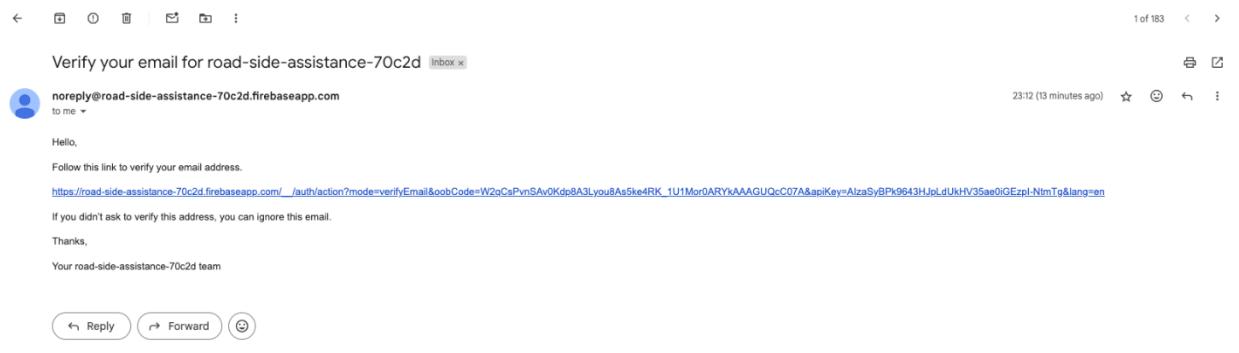


Fig no. 27: E-mail varification

Sign-in providers		Add new provider
Provider	Status	
Email/Password	Enabled	
Phone	Enabled	

Fig no. 28: Verified Vrovidiers

Search by email address, phone number or user UID					Add user	C	⋮
Identifier	Providers	Created	Signed in	User UID			
kazimehnaznobisupti@... +8801775807779	📞 ✉️	7 Jan 2025	7 Jan 2025	bkpn2mBmBbQLBAjAWBXWT...			
supti.nobi1810@gmail.... +8801796558902	📞 ✉️	7 Jan 2025	7 Jan 2025	wQHm3OpU0bXyKq6DrGBWJ...			
		Rows per page	50	▼	1 - 2 of 2	<	>

Fig no. 29: Verified Users

The screenshot shows the Google Cloud Firestore interface. On the left, there's a sidebar with a 'default' section containing a '+ Start collection' button and a 'Users' section. Under 'Users', there's a document with the ID 'bkpn2mBmBbQLBajAWBXWT1YJiIC3'. On the right, the document details are displayed:

```

{
  "Email": "kazimehnaznobisupti@gmail.com",
  "FirstName": "Mehnaz",
  "ForemanAccount_Status": true,
  "ForemanAddress": "Tangail",
  "ForemanArea": "Tangail",
  "ForemanCity": "Tangail",
  "ForemanState": "Mohammadpur",
  "LastName": "Nobi",
  "MobileNumber": "1775807779",
  "Password": "Supti1800",
  "UserType": "Foreman",
  "sForemanLatitude": "24.2514853",
  "sForemanLongitude": "89.9198043"
}

```

Fig no. 30: Database

The screenshot shows the 'Phone' provider setup in the Firebase console. It includes an 'Enable' switch, a note about requiring configuration steps for different platforms (Apple, Android, Web), and a section for testing with phone numbers +1 650-555-1234 and +880 1775-807779.

Fig no. 31: Verified OTP

The screenshot shows the 'gs://road-side-assistance-70c2d.appspot.com/images/userprofile' folder in Google Cloud Storage. It lists three files: '4NmRdoasL3PoJvy32N7DCl6xhkm2.jpg' (size 8.2 KB, type image/jpeg, last modified 30 Dec 2024) and 'wQHm3OpU0bXyKq6DrGBWJ9Yq5DN2.jpg' (size 8.2 KB, type image/jpeg, last modified 7 Jan 2025). There is also an 'Upload file' button.

Fig no. 32: User profile ID

Chapter 6

Implementation and Testing

The implementation and testing phase of the "Swift Assist App" epitomizes a cardinal juncture in its developmental odyssey, transitioning the app from an abstract conceptualization to a tangible, efficacious, and resilient mobile application. A revolutionary vehicle breakdown management system, signify a crucial stage in its software development life cycle. This app amalgamates a plethora of features including GPS tracking, real-time navigation to gas stations, utility shops, and mechanics, ensuring users receive swift and effective support during vehicular emergencies. The primary goal of this phase is to actualize conceptual designs into a robust, reliable, and user-centric application. The process integrates advanced coding techniques, modular system designs, and the deployment of sophisticated algorithms to meet both functional and non-functional requirements effectively. Rigorous testing procedures ensure adherence to industry benchmarks, elevating performance, security, and user satisfaction to unparalleled levels. The implementation process is characterized by an intricate tapestry of undertakings, encompassing the intricate codification of the application and the seamless synthesis of quintessential functionalities such as geospatial telemetry for real-time vehicular positioning, exigency communiqué systems, and a meticulously architected databank management schema. These functionalities are scrupulously devised to furnish users with expeditious and efficacious succor during exigent automotive vicissitudes. For instance, the geospatial telemetry apparatus empowers users to relay their precise coordinates to pertinent service entities, thereby expediting operational remediation. Testing constitutes the sine qua non of this phase, serving as the linchpin for corroborating the app's operational veracity, ergonomic dexterity, and robustness across a plethora of contingencies. This poly faceted process entails the scrupulous assay of modular constituents to preclude and rectify anomalies, the concatenative validation of integrated systems to ascertain seamless interoperability across heterogeneous platforms (Android) and user acceptance testing to assimilate invaluable perspicacity derived from the app's end-users. This iterative recalibration ensures that the app's functionalities coalesce with user desiderata while concurrently optimizing its interface for consummate intuitive engagement. Moreover, the implementation and testing phase encompasses an assiduous emphasis on non-functional exigencies, including computational celerity, systemic dependability, cryptographic sanctity, and operational extensibility. Computational celerity testing

affirms the app's aptitude to sustain unimpeachable efficacy amidst prodigious user loads. Systemic dependability testing ensures unremitting functionality devoid of perturbations. By adhering scrupulously to exegetical paradigms and preeminent methodologies within the ambit of software engineering, the development consortium aspires to bequeath a holistic, perspicuous, and tenacious application. The "Swift Assist App" aspires to transcend the paradigms of vehicular breakdown remediation, establishing itself as a paragon of user-centric design and unimpeachable dependability.

6.2 Process Model:

Swift Assist employs an Agile methodology to ensure its safety features are constantly evolving. This dynamic approach includes frequent evaluations, robust testing, and continuous refinement of safety protocols and procedures. By iteratively improving, Swift Assist guarantees its safety measures remain cutting-edge and highly effective in proactively addressing potential risks.

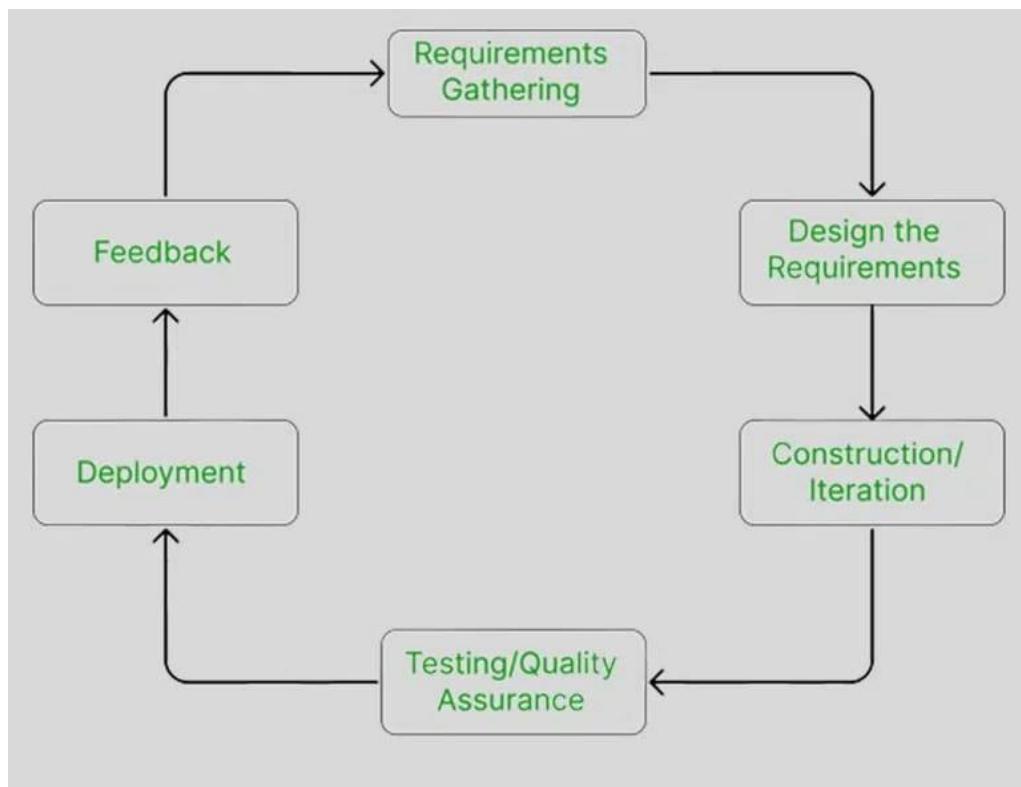


Fig No. 6 : Software process model

6.3 Testing

Testing represents the cornerstone of quality assurance, encapsulating a gamut of methodologies designed to validate the app's functionality, reliability, and scalability.

6.3.1 Validation Testing: Validation testing for the "Swift Assist App" emphasizes ascertaining that the developed application adheres to predefined specifications and satisfies the exigencies of its users. This evaluative process encompasses a sequence of meticulous assessments and scrupulous verifications, designed to ensure that the final product harmonizes with the app's overarching objective: revolutionizing vehicular breakdown assistance and emergency management. It encompasses:

- i. **Requirement Adherence:** Conducting a thorough examination to validate the accurate implementation of core functionalities such as geolocation tracking, emergency support mechanisms, and real-time navigational assistance. This step ensures these components align with the app's foundational objectives, offering seamless user support during vehicular emergencies.
- ii. **End-User Validation:** Collaborating with target users to assess the app's practicality, aesthetic coherence, and functional efficiency. This includes gauging the intuitiveness of the interface, the ease of navigation, and the app's capacity to deliver prompt and reliable solutions under diverse scenarios.
- iii. **Regulatory and Ethical Compliance Assessment:** Ensuring the app adheres to prevailing legal mandates and ethical directives, particularly those concerning data security, user privacy, and safety standards. This involves rigorous scrutiny to uphold user trust and mitigate potential liabilities.
- iv. **Environmental Simulation Testing:** Emulating a variety of real-world circumstances, such as intermittent connectivity, heavy user traffic, or extreme operational demands, to ascertain the app's resilience and dependability. This ensures that the app performs optimally across a spectrum of unpredictable and challenging conditions.

Validation testing of the 'Swift Assist App' is crucial to refine and optimize its functionality, guaranteeing its adherence to stated objectives and its provision of a dependable resource for users to augment their personal security.

6.3.2 Verification Testing:

Verification testing for the Swift Assist App constitutes a methodical evaluation phase aimed at substantiating the precision of technical implementations and ensuring strict conformity to design paradigms. This phase is indispensable for certifying that the application's core functionalities are meticulously constructed to align with architectural blueprints and performance benchmarks. The following aspects exemplify the intricacies of this process:

- i. **Code Audit and Scrutiny:** A granular dissection of the underlying source code is performed to ascertain adherence to rigorous coding protocols and development standards. This entails identifying latent anomalies, eliminating inefficiencies, and ensuring that the syntax and structure resonate with best practices.
- ii. **Isolated Module Appraisal:** Each discrete functional component of the app—such as geospatial tracking systems, emergency response alerts, and service locators—is subjected to exhaustive standalone testing. This ensures that individual modules exhibit operational coherence and impeccability in isolation from the broader system.
- iii. **Integrated System Examination:** Following the validation of individual components, an integrative analysis is conducted to verify the synergistic interaction among interconnected modules. This step ascertains that the app functions holistically, maintaining unblemished interoperability across diverse features and frameworks.
- iv. **Documentation Vetting and Reconciliation:** A comprehensive review of all accompanying technical documentation, including system architecture diagrams, design specifications, and operational guidelines, is undertaken. This ensures that the documented schema aligns with the actualized design and facilitates seamless comprehension for future development or maintenance.

6.3.3 Acceptance Testing

This pivotal terminal phase in the software development lifecycle ensures the Swift Assist App is ready for deployment by meticulously assessing its functionality, usability, and reliability through multifaceted evaluations:

- i. **User Acceptance Testing (UAT):** This segment involves the simulation of real-world scenarios to validate the app against user expectations and operational requirements. The process includes: Executing test cases that mimic actual user workflows to ascertain the app's capability to meet specified business needs. End-users and stakeholders provide critical feedback, ensuring the app aligns with intended utility and usability standards.
- ii. **Operational Testing:** This subsection examines the app's performance and reliability under pseudo-production conditions, focusing on infrastructure and systemic robustness.
 - **Environmental Simulation:** This means creating a realistic test environment that closely resembles the real world where the app will actually work. This includes things like simulating the network it will use, the databases it will store information in, and the computers it will run on.
 - **Redundancy Checks:** This is about testing the app's backup plans. We want to see how well it can recover from unexpected problems, such as equipment failure or power outages. This helps to minimize downtime and ensure the app stays available to users even in difficult situations.
- iii. **Feedback Loop and Iterative Refinement:** An iterative cycle for continuous improvement based on insights gleaned from testers and stakeholders. The feedback loop involves gathering both qualitative and quantitative insights to identify areas for improvement. Targeted refinements and optimizations are then implemented to address these issues. Finally, modified components are retested to ensure they meet established benchmarks.
- iv. **Feedback and Approval:** The testing team collects feedback and addresses any issues raised during testing. Once all tests are passed and user satisfaction is confirmed, the app receives approval for release.

6.3.4 Load Testing

The load testing phase rigorously evaluates the Swift Assist App's resilience and performance under peak user demand, leveraging advanced methodologies to expose potential bottlenecks.

- i. **Simulating High Traffic:** Artificially inducing elevated levels of concurrent user interactions to evaluate system robustness. Robustness evaluation necessitates the artificial induction of elevated levels of concurrent user interactions. This involves the deployment of virtual users to emulate real-world interaction patterns at scale, encompassing a spectrum of usage scenarios, from moderate activity to extreme surges.
- ii. **Comprehensive Performance Metrics Evaluation:** Comprehensive scrutiny of key indicators to assess app behavior under stress. "Performance evaluation encompasses critical metrics such as Latency Measurement, monitoring response times to ensure timely delivery of app functions under load. Throughput Calculation analyzes the volume of data processed within a given timeframe to ensure operational efficiency. Resource Utilization tracks CPU, memory, and bandwidth consumption to identify potential inefficiencies or bottlenecks.
- iii. **Optimization for Scalability and Responsiveness:** Leveraging data-driven insights to enhance app performance under demanding conditions. Optimization for Scalability and Responsiveness involves leveraging data-driven insights to enhance app performance under demanding conditions. Key activities include Configuration Refinement, such as tweaking server, database, and application parameters for optimal performance. Code Optimization focuses on streamlining algorithms and processes to reduce computational overhead. Capacity Planning proactively prepares infrastructure to accommodate anticipated future growth.

6.4 Conclusion:

The implementation and testing of the Swift Assist app not only ensure technical excellence but also underscore the commitment to delivering a seamless user experience. By adhering to a rigorous, multi-faceted testing framework, the app is poised to redefine vehicle breakdown management, fostering reliability, user trust and operational efficiency. The continuous refinement process paves the way for future enhancements, ensuring the app remains a pinnacle of

technological innovation and user-centric design. The development process involves bringing conceptual frameworks to life by embedding essential functionalities, such as live navigation assistance, quick access to help services, and a well-organized repository of service providers and resources. The primary goal is to ensure smooth operation and user-focused design for an intuitive and efficient experience. Thorough testing is the foundation of this phase, comprising different levels like component validation, integrated system checks, and end-user validation. Component-level testing confirms the accuracy of individual features, while system integration ensures cohesive functionality across devices, including iOS and Android platforms. End-user validation is critical for evaluating real-world performance and collecting actionable insights to enhance user satisfaction and app reliability. An efficient feedback mechanism is crucial during this stage, enabling continuous enhancements based on user input. Swift resolution of identified issues builds user confidence while fine-tuning the app's interface and overall performance. Deployment strategies focus on ensuring adaptability to accommodate increasing user demands and implementing robust data privacy protocols to protect sensitive information. The app's ability to scale effectively and safeguard user data will remain a top priority to maintain high performance and security as its audience grows. After launch, ongoing monitoring will be essential to address unforeseen issues promptly, ensuring consistent functionality across platforms and delivering a seamless user experience. In summary, the successful completion of the development and validation phase for the Swift Assist App will empower users with a reliable tool for assistance and navigation. This initiative highlights the transformative potential of technology in solving everyday challenges and improving quality of life.

Chapter 7

Conclusion and Future Work

7.1 Introduction:

As the development of the "Swift Assist" application approaches its final stages, it is essential to reflect on the app's profound significance in addressing critical concerns related to vehicle breakdowns and on-road assistance. This app represents a pivotal advancement in leveraging mobile technology to provide users with reliable and practical support during stressful vehicle-related emergencies. One of the core functionalities of the app is real-time GPS tracking. This feature allows users to share their precise location with service providers or trusted contacts, ensuring timely assistance when required. By offering a continuous, accurate view of both the user's and service provider's location, the app significantly improves the efficiency of help during breakdown situations. Another essential component is the on-demand service request system, enabling users to quickly access towing services, mechanic support, or other roadside assistance. With just a few taps, users can instantly connect with reliable professionals, ensuring that help is mobilized swiftly to minimize disruptions. The app's intuitive and user-friendly interface ensures that these features are easily accessible, even under stressful conditions. This design prioritizes simplicity, allowing users to navigate the app effortlessly and seek assistance without delays. Additionally, user feedback integration plays a crucial role in enhancing service quality by enabling users to rate their experience and provide valuable input for continuous improvement. Overall, "Swift Assist" represents a significant step forward in empowering users to handle vehicle breakdowns with greater confidence and ease. By harnessing the power of modern mobile technology, the app not only addresses immediate assistance needs but also fosters a sense of security for users during challenging situations. As the development process nears completion, the focus remains on ensuring that the app delivers on its promise of providing reliable, efficient, and user-centric roadside support services.

7.2 Impact on Society, Environment, and Sustainability:

The "Swift Assist" application holds significant potential to positively impact society, the environment, and sustainability. By providing reliable on-road assistance, the app contributes directly to societal well-being by ensuring users' safety and minimizing the stress and risks

associated with vehicle breakdowns. This empowerment can lead to broader societal benefits, such as improving road safety and fostering confidence in transportation systems. From an environmental perspective, the app's reliance on mobile technology reduces the need for physical infrastructure or manual intervention, thereby decreasing the carbon footprint associated with traditional roadside assistance services. By facilitating quicker responses and preventing prolonged idling of vehicles during breakdowns, the app also indirectly reduces fuel wastage and emissions. Looking forward, future work could focus on expanding the app's capabilities to address a wider range of vehicle-related challenges, including services in rural and remote areas where access to assistance is limited. Enhancing the app's usability for diverse user demographics and integrating additional features, such as electric vehicle (EV) charging support or eco-friendly towing options, could further amplify its impact. Partnerships with local governments, roadside assistance providers, and automotive organizations could help scale the app's deployment and make it more accessible to a broader audience. In conclusion, "Swift Assist" represents a technological advancement that not only enhances personal safety and convenience but also aligns with sustainability goals by reducing environmental impacts and promoting efficient resource utilization. Continued innovation and collaboration will be essential to maximize its positive contributions to society, the environment, and sustainable development.

7.3 Ethical Issues:

The development and deployment of the "Swift Assist" app raise several ethical considerations that must be addressed to ensure its responsible use and effectiveness. Privacy concerns regarding the collection, storage, and utilization of user data, such as location and personal details, are paramount. Robust measures, including data encryption, secure servers, and transparent privacy policies, are essential to safeguard sensitive information and maintain user trust. Ensuring inclusivity and accessibility is critical to avoid marginalizing users based on technological proficiency or socioeconomic status. The app must remain user-friendly and affordable, promoting equitable access to its services. Continuous user feedback and iterative improvements are vital to enhance the app's usability and effectiveness. This involves conducting real-world user testing to refine the interface and functionalities, ensuring the app meets diverse user needs. Collaboration with local service providers, automotive organizations, and insurance companies can strengthen the app's integration into existing support systems, enabling seamless emergency responses.

Additionally, addressing regional disparities in service availability and affordability is important to ensure the app is equally effective in urban and rural settings.

7.4 Future Work (Including Limitations of the Research):

Looking ahead, several avenues for future work can enhance the effectiveness and adoption of the "Swift Assist" application. Firstly, integrating advanced machine learning algorithms could improve the app's predictive capabilities, enabling it to anticipate vehicle breakdown risks based on user driving patterns, weather conditions, and vehicle diagnostics. Enhancements in real-time data analytics could further improve the accuracy of GPS tracking and optimize response times for towing and roadside assistance services. Expanding the app's functionality to include offline mode capabilities and integration with wearable devices (such as smartwatches) would cater to diverse user needs, ensuring accessibility in areas with limited connectivity. Additionally, incorporating features like electric vehicle (EV) charging station locators and sustainable towing options would align the app with emerging trends in the automotive sector. Conducting usability studies across different user demographics and geographic regions could provide valuable insights into optimizing the app's interface, ensuring it remains intuitive and user-friendly for a broad audience. However, the research does come with limitations. Privacy concerns related to location tracking and data security require careful mitigation through robust encryption and transparent data management policies. Additionally, the scalability of the app in managing large user bases and the availability of reliable service providers in rural and remote areas remain challenges that need further exploration. Cultural and regional differences in user behavior and expectations could also impact the app's adoption and effectiveness, necessitating localization efforts and user education initiatives.

7.5 Conclusion:

In conclusion, the development of "Swift Assist" represents a significant advancement in addressing critical concerns related to vehicle breakdowns and on-road assistance. By harnessing the power of mobile technology, the app integrates features such as real-time GPS tracking, on-demand service requests, and user feedback mechanisms. These functionalities are designed to provide users with a reliable and efficient tool for resolving vehicle-related emergencies, offering

immediate assistance and peace of mind. Meticulous project planning and scheduling have laid a strong foundation for developing essential features that prioritize user-centric design and functionality. Moving forward, future work will focus on several key areas to enhance the app's effectiveness and impact. Continuous user feedback will play a central role in driving iterative updates and refinements, ensuring that the app remains responsive to user needs and emerging trends in transportation. Strengthening data security protocols will be a priority to protect user privacy, ensuring the confidentiality of sensitive information such as location data and service history. Localization efforts and partnerships with towing companies, mechanics, and insurance providers will help expand the app's reach, making it accessible to diverse communities across urban, rural, and remote areas. Furthermore, ongoing research and development will explore advanced technologies, such as AI-driven predictive analytics and real-time diagnostic tools, to proactively identify and address vehicle issues before breakdowns occur. These innovations will enable "Swift Assist" to transition from a reactive tool to a proactive system that prevents incidents and enhances user convenience.

References:

- [1] Hernandez, J., & Singh, R. (2020). Dynamic Assistance Scoring in Urban Environments: A Study on GeoAssistance. *Journal of Urban Assistance and Data Analytics*, 12(3)
- [2] Sharma, P., Kumar, S., & Mehta, R. (2019). Community-Driven Mapping for Roadside Assistance: The Role of Crowdsourcing in Emergency Support. *International Journal of Public Safety and Geoinformatics*, 14(3).
- [3] Mendez, D., Rios, C., & Garcia, A. (2020). Enhancing Roadside Assistance Through Crowdsourced Data and GIS in Urban Areas. *Journal of Geospatial Assistance Solutions*, 9(2).
- [4] Johnson, K., & Liu, M. (2020). Community Networks in Vehicle Assistance: A Peer Support Approach. *Journal of Social Computing and Assistance Technologies*, 11(4).
- [5] Rao, A., & Banerjee, P. (2019). Evaluating the Integration of Government-Endorsed Roadside Assistance Apps in Public Safety Systems. *International Journal of Mobile Assistance Technologies*, 14(1).
- [6] Zhang, L., Wu, H., & Li, J. (2020). Implementing REST APIs for Communication in Vehicle Assistance Applications. *Journal of Software Engineering and Assistance Solutions*, 22(4).
- [7] Senaratne, H., Mobasher, A., & Zipf, A. (2017). The Role of Digital Storytelling in Vehicle Assistance Services. *International Journal of Location-Based Services*, 11(3).
- [8] Nakamura, H., & Patel, V. (2021). Voice-Activated SOS Systems for Vehicle Assistance: Enhancing Safety with Machine Learning. *Journal of Artificial Intelligence and Assistance Technologies*, 18(3).

- [9] Alshammari, F., Sarhan, Q., & Albarak, M. (2019). Predictive Analytics for Location-Based Roadside Assistance Applications. *Journal of Mobile Computing and Assistance Solutions*, 8(2).
- [10] Hernandez, J., & Singh, R. (2018). Evaluating Gesture-Based Emergency Alerts in Vehicle Assistance Applications. *Journal of Cybersecurity and Assistance Technology*, 12(2).
- [11] Reardon, S., & Kalms, N. (2019). Voice Recognition Technology for Roadside Assistance: A Focused Study. *Journal of Automotive Assistance Technologies*, 15(1).
- [12] Sharma, P., Kumar, S., & Mehta, R. (2021). The Impact of Peer Networks on Vehicle Assistance Applications. *Journal of Urban Assistance and Geoinformatics*, 14(2).
- [13] Mendez, D., Rios, C., & Garcia, A. (2020). Timer-Based Emergency Alerts in Vehicle Assistance Applications. *Journal of Geospatial Assistance Solutions*, 9(3).
- [14] Zhang, L., Wu, H., & Li, J. (2020). Enhancing Family Assistance with Integrated Location Sharing Systems for Roadside Emergencies. *Journal of Software Engineering and Assistance Solutions*, 22(4).
- [15]<https://zantrik.com>
- [16]<https://www.sheba.xyz>
- [17]<https://lifetrackerbd.com>
- [18]<https://roadhop.com>
- [19]<https://www.finder.com.bd>