Hassan Khushhal

01-135201-024

Muhammad Hasnain Ashraf

01-135201-054

An E-Complaint Registration Management System for Housing Society

Bachelor of Science in Information Technology

Supervisor : Ms. Mehroz Sadiq

Department of Computer Science

Bahria University, Islamabad

December, 2023

Certificate

We accept the work contained in the report titled **"Fixome"**, written by **Mr. Hassan Khushhal** and **Mr. Muhammad Hasnain Ashraf** as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

. . . .
HOD
Department of Computer Sciences,
Bahria University Islamabad.

. . . .
Supervisor
Department of Computer Sciences,
Bahria University Islamabad.

Acknowledgements

All gratitude is due to Almighty Allah, who gave us a little fraction of His vast wisdom, as well as for the good health and wellbeing required to finish this report. We wish to express our sincere thanks to Ms. Mehroz Sadiq, for guiding us at various stages of the project and providing us with all the necessary facilities for the research. We are extremely thankful and indebted to her for sharing expertise, sincere and valuable guidance, and encouragement. This project would not have been completed without their supervision, advice, and invaluable guidance. We are grateful to them for their support and encouragement throughout this project. We would want to take this opportunity to thank everyone of the department's faculty for their assistance and support. We also like to thank our parents for their constant love, care, and support.

Hassan Khushhal and Muhammad Hasnain Ashraf

Islamabad, Pakistan

Abstract

The ongoing household maintenance complaint management system faces a few difficulties that frustrate its effectiveness. Normally, citizens need to physically visit municipal offices or call assigned helplines to report issues, bringing about time-consuming and lumbering cycles. Absence of real-time updates leaves citizens ignorant about the advancement of their complaints, prompting disappointment and diminished trust in the system. Moreover, the manual treatment of complaints frequently prompts shortcomings in issue prioritization, classification, and goal, creating setbacks for resolving basic issues. To defeat the restrictions of the ongoing system, our versatile app, "Fixome," offers a user-friendly and proficient answer for household maintenance complaint management. The app empowers citizens to report issues straightforwardly through their smartphones, wiping out the requirement for actual visits or calls. With most recent computerized abilities, users can give exact subtleties, like location and images, guaranteeing far reaching complaint system. Real-time updates and notifications keep citizens educated about the advancement regarding their complaints, imparting a feeling of confidence and trust in the system. The app's prioritization and classification module order complaints in view of urgency and effect, empowering authorities to zero in on settling basic issues speedily. By digitizing the complaint management process, Fixome limits manual mediation, lessening reaction times and further developing generally speaking help productivity. Fixome up-sets household maintenance complaint management by placing power in the possession of citizens. Users can download the app from app stores, create accounts, and effectively submit complaints with only a couple of taps. The app's instinctive connection point guarantees that even non-educated users can navigate and utilize it really. "Fixome," overcomes any barrier among citizens and local authorities, giving a productive, straightforward, and user-centric approach to household maintenance complaint management. By utilizing the comfort of smartphones and trend setting innovation, the app improves administration proficiency, guaranteeing a cleaner, more secure, and better living climate for all.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

GUI	Graphical User Interface
RFQ	Request For Quotation
TC	Test Case
UC	Use Case
UI	User Interface
UX	User Experience
UML	Unified Modified Language

Chapter 1

Introduction

1.1 Background

Household maintenance [1] is a basic aspect of metropolitan residing, ensuring the well-being of residents and the upkeep of essential infrastructure. However, the current, manual, complaint management systems in this space often suffer from inefficiencies, leading to delayed issue resolution and reduced citizen satisfaction. Manual processes and the absence of real-time updates contribute to these challenges, necessitating a more efficient and transparent approach to complaint management. To address these constraints, this project proposes the development of "Fixome," an E-Complaint and Management System for Household Maintenance. Leveraging mobile technology and automated issue classification, the app means to streamline complaint reporting and resolution, empowering citizens to actively engage in the improvement of their neighborhoods and fostering a seamless correspondence channel between residents and local authorities. By enhancing service efficiency and citizen support, Fixome seeks to revolutionize how household maintenance complaints are handled, ensuring a cleaner, safer, and better residing environments for all.

1.2 Problem Description

The domain of household maintenance faces several challenges in its complaint management system, leading to inefficiencies and diminished service quality. The current manual processes for registering complaints often result in time-consuming and cumbersome procedures for citizens. Absence of real-time updates and correspondence channels leaves residents uninformed about the situation with their complaints, leading to dissatisfaction and reduced trust in the system. Furthermore, the absence of an automated prioritization and classification mechanism hampers the efficient allocation of resources, resulting in delayed resolution of basic issues. The absence of a user-friendly and accessible platform restricts citizen engagement, hindering the active support of residents in working on their communities. To address these restrictions and enhance the overall effectiveness of household maintenance complaint management, there is a pressing need for an innovative solution that leverages modern technology to foster transparency, efficiency, and citizen-centric service delivery. This project intends to develop Fixome, an E-Complaint and Management System for Household Maintenance, to overcome these challenges and provide a seamless and user-friendly platform for citizens to report and monitor complaints, thereby enhancing the quality-of-service delivery and advancing greater local area engagement.

1.3 Objectives

Fixome is a mobile application which provides a platform for the citizen to register and track their household maintenance whines by means of a smartphone. The objectives of this application are:

Streamline Complaint Reporting:

The essential objective of the app is to provide a user-friendly and efficient platform for citizens to report household maintenance complaints. By enabling users to submit complaints directly through their smartphones, the app intends to eliminate manual processes and streamline the complaint reporting procedure.

Real-time Updates and Notifications:

The app seeks to enhance communication between citizens and local authorities by giving real-time updates and notifications on the progress of reported complaints. Timely updates keep users informed about the situation with their complaints, fostering transparency and imparting confidence in the complaint resolution process.

Automated Prioritization and Classification:

Fixome plans to automate the prioritization and classification of complaints based on urgency and effect. By employing an intelligent system, authorities can efficiently allocate resources and address basic issues expeditiously, resulting in more effective issue resolution.

Multimedia Capabilities:

The app permit users to provide precise details about the location and nature of their complaints through geotagging and multimedia capabilities. Users can append images or videos to their complaints, giving comprehensive description to better understanding and resolution.

1.4 Project Scope

The Fixome app has been deployed and used within a specific geographic region, like a city or municipality, for managing household maintenance complaints and improving citizen engagement. The initial scope of the project plans to cater to the needs of residents in the selected area, allowing them to report and screen complaints related to household services and infrastructure.

Upon successful implementation and approval of the app's functionality and usability, there are vital opportunities for expanding the project at a larger scale:

City-wide Implementation:

After starting deployment and positive feedback from users, the Fixome app can be stretched out to cover the whole city or municipality. By teaming up with local authorities, the app's scope can be extended to help a bigger populace, making a more complete and comprehensive complaint management system.

Multi-City Deployment:

If the app ends up finding success in its pilot city, it tends to be considered for deployment in different urban communities or locales with comparative household maintenance challenges. This would require collaboration with multiple municipalities to adjust the app to suit the extraordinary necessities of every location.

Regional and National Adoption:

As the app gets some decent momentum and recognition for its effectiveness, it might actually be taken on at a regional or even national level. Increasing the app would include participation with more significant level governmental bodies and vital organizations to guarantee consistent combination across various authoritative areas.

Collaboration with Service Providers:

The Fixome app can likewise be extended to incorporate collaboration with different service providers, like service organizations or waste management offices. By incorporating their complaint management systems into the app, users can report issues connected with these services, further upgrading the stage's utility and effect.

Continuous Improvement and Feature Expansion:

As the app gains notoriety and is utilized by a bigger user base, continuous improvement and feature expansion become fundamental. In view of user feedback and developing necessities, the app can be upgraded with new functionalities and improvements to additionally raise the nature of service gave.

The project scope at first spotlights on a particular locale, yet the potential for growing the Fixome app to a bigger scope is promising. By exhibiting its viability and worth in the underlying implementation, the app can prepare for a more far reaching and widespread answer for household maintenance complaint management, helping networks on a more extensive scale.

1.5 Organization of Report

The document is organized as follows. In the second Chapter 2, literature review, we have discussed the comparative study and similar work. Chapter 3 presents the requirement specifications while Chapter 4 details the system design. System Implementation is discussed in Chapter 5 while Chapter 6 presents the results of the experimental evaluations. Finally, we conclude the report in Chapter 7.

Chapter 2

Literature Review

The literature review provides a clear perception of the standards being used in the past and the present while making and agreement or investment. It gives us an insight of the business models that currently exist and changes with the passage of time. Due to the evolution of the Android and Web-based Applications the many developers have faced many discontinuities which may include features which application offered with respect to demand of the time but cannot satisfy today's demands. Now it is time saving and very easy to use. There is a lot of competition, mobile and web-related technologies effect in such a way that today almost everyone uses a mobile phone which is also used as a schedule reminder and communicator. People find it more convenient to use such applications. This application connects both clients and team/officer in such a way that both can communicate (when necessary) with each other get answers to the queries.

2.1 Citizen Complain Management System (CCMS)

The point of CCMS is to provide a comprehensive web-based platform for citizens in Pakistan to register and screen complaints related to public services and infrastructure. By offering a user-friendly interface, it promotes transparency, responsibility, and efficient complaint resolution processes. CCMS [2] enables citizens to report different issues, going from defective streetlights to garbage removal problems, ensuring their concerns are heard and addressed expeditiously. The web-based nature of CCMS permits users to access the platform from any internet-connected device, ensuring widespread accessibility and investment. Tools and Technologies: The particular apparatuses and innovations used to construct CCMS might fluctuate relying upon the association or government body that created it. For the most part, such frameworks are electronic applications that utilization a mix of innovations. Normally involved advances for online applications like CCMS include:

- **Front-end:**

  HTML, CSS, JavaScript, Bootstrap, jQuery

- **Back-end:**

  PHP, Python, Java, ASP.NET, Node.js

- **Database:**

  MySQL, PostgreSQL, Oracle

- **Web Frameworks:**

  Laravel, Django, Spring, Express.js

- **Geographic Information System (GIS):**

  Integration for location tracking and mapping.

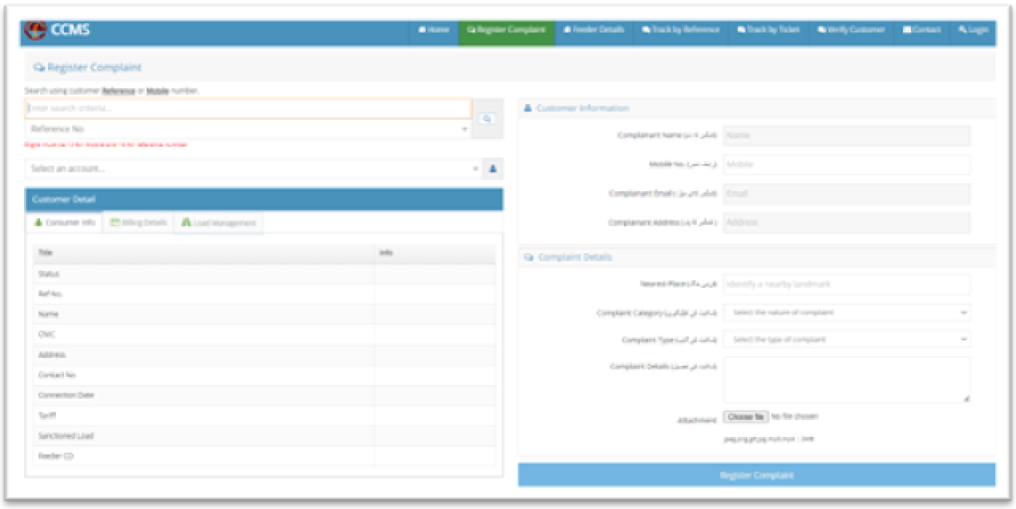


FIGURE 2.1: CCMS

2.2 FixMyStreet

FixMyStreet (both Mobile and Web app) aims to empower users to actively participate in improving their local communities by reporting and tracking local issues such as potholes, graffiti, or street lighting problems. The web and mobile app platforms provide a convenient and user-friendly interface for users to submit detailed reports, including location, description, and photos, enabling authorities to quickly identify and address the reported issues. FixMyStreet [3] encourages community engagement, collaboration, and accountability by facilitating communication between citizens and local government bodies.

Tools and Technologies:

The explicit apparatuses and advances used to construct FixMyStreet, or comparative applications can differ in light of the improvement group's inclinations, project prerequisites, and innovation patterns at the hour of advancement. Here are a few normal devices and innovations that may be utilized:

- **Front-end Technologies:**

  HTML, CSS, and JavaScript for building the user interface and enabling interactivity on the client-side.

- **Front-end Frameworks:**

  Libraries like React, Angular, or Vue.js for faster and more organized development.

- **Mapping and Geolocation:**

  APIs like Google Maps API or Mapbox API to integrate mapping and geolocation features.

Back-end Technologies:

- **Programming Languages:**

Commonly used languages like Python, Ruby, PHP, or Node.js for server-side development.

- **Web Frameworks:**

For example, Django (Python), Ruby on Rails (Ruby), Laravel (PHP), or Express.js (Node.js) to provide a structured back-end environment.

- **Database:**

A database management system like PostgreSQL or MySQL to store and manage user data and reported issues.

Mobile App Development (Optional, if applicable):

For mobile app development, different technologies can be used based on the platform (iOS or Android)

Native Development:

Swift or Objective-C for iOS and Java or Kotlin for Android.

Cross-platform Development:

Frameworks like React Native or Flutter for building apps that work on both iOS and Android.

User Authentication and Security:

OAuth or Token-based authentication for secure user login and authorization. SSL certificates for secure data transmission over the internet.

Version Control:

Tools like Git for managing code versions and facilitating collaboration among developers.

Testing and Quality Assurance:

Testing Frameworks:

Tools like Jest (for JavaScript), pytest (for Python), or PHP-Unit (for PHP) for automated testing.

Continuous Integration (CI) and Continuous Deployment (CD) tools for automated testing and deployment.

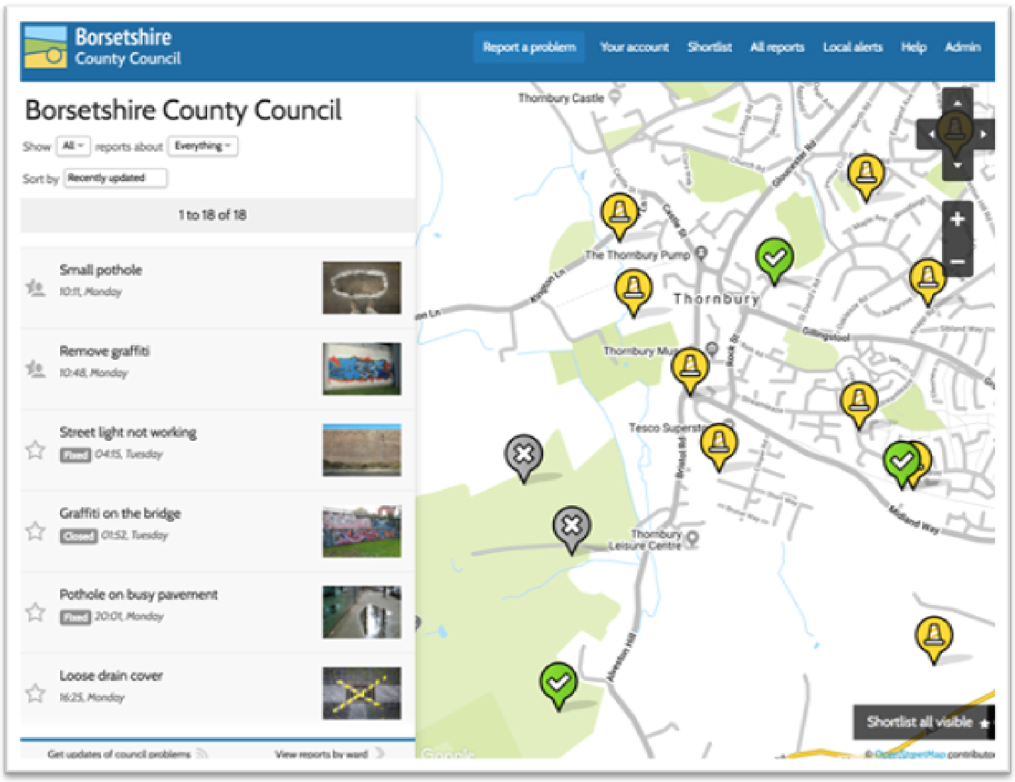


FIGURE 2.2: FixMyStreet

The genuine tools and technologies utilized for FixMyStreet, or comparative applications might vary in light of the improvement group's mastery and the particular necessities of the application. Furthermore, new technologies and advances might have arisen since the improvement of FixMyStreet, impacting the decisions made for resulting applications in a similar space.

2.3 SnapFix

SnapFix [4] (Mobile app) aims to empower individuals to actively contribute to community improvement by capturing and reporting issues in their surroundings, such as litter, broken infrastructure, or environmental concerns. The mobile app provides an intuitive and user-friendly interface, allowing users to take photos, add descriptions, and geotag the reported issues. SnapFix promotes a cleaner and more sustainable environment by facilitating prompt reporting and resolution of community-related problems.

There is no specific information related about the tools and technologies through which the SnapFix have developed. But there is a rough idea on what kind of tools and technologies could be used if such an app must be built:

Front-end Development:

- **Mobile App (iOS and Android):**

  For building native mobile apps, technologies like Swift or Objective-C for iOS, and Java or Kotlin for Android might be used.

- **Cross-platform Development:**

  Alternatively, frameworks like React Native or Flutter could be used to develop a single codebase that works on both iOS and Android platforms.

Back-end Development:

- **Server-side Programming Language:**

  Languages like Python, Ruby, or Node.js might be used for the back-end development.

- **Web Framework:**

  Web frameworks like Django (Python), Ruby on Rails (Ruby), or Express.js (Node.js) could be used to provide a structured back-end environment.

Database:

A database management system like PostgreSQL, MySQL, or MongoDB might be used to store and manage user data, reported issues, and other relevant information.

Geolocation and Mapping:

APIs like Google Maps or Mapbox might be integrated to enable users to pinpoint the location of reported issues on a map.

User Authentication and Security:

Techniques like OAuth or token-based authentication could be implemented for secure user login and data protection.

Testing and Quality Assurance:

Automated testing frameworks like Jest (for JavaScript), XCTest (for iOS), or Espresso (for Android) might be used to perform testing and ensure app functionality and performance.

Deployment and Distribution:

The app could be deployed on app stores (e.g., Apple App Store and Google Play Store) for public use.

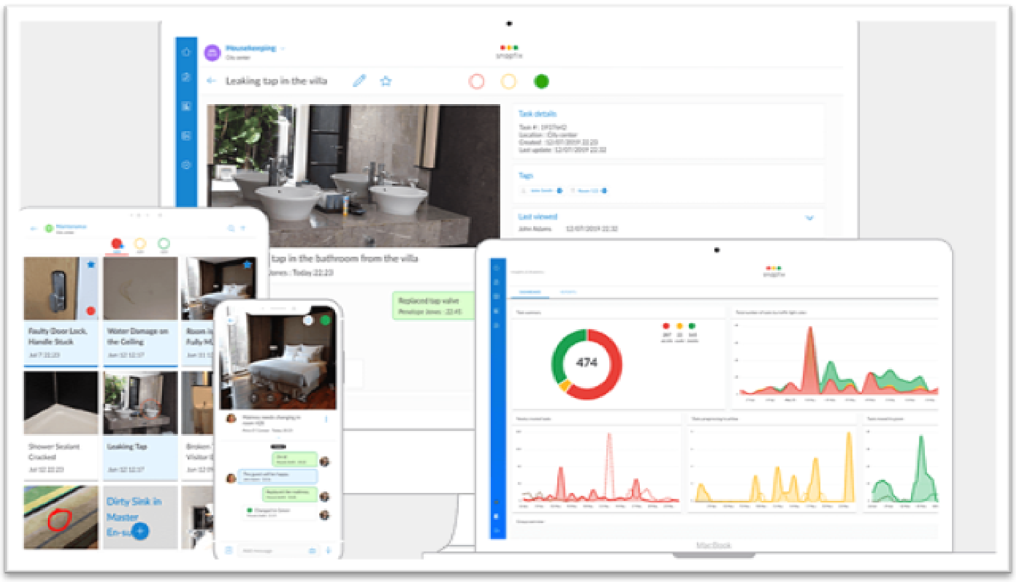

Leaking tap in the villa
Faulty Door Lock, Handle Stuck
Water Damage on the Ceiling
Shower Sealant Cracked
Leaking Tap
Dirty Sink in Master En-su
474

FIGURE 2.3: SnapFix with complaints list, dashboard

2.4 StreetBump

The aim of StreetBump [5] is to improve road infrastructure and maintenance by utilizing smartphone sensors to help users identify and report potholes and road conditions. The mobile app utilizes the device's accelerometer and GPS to detect and record road irregularities as users drive or walk. By collecting and analyzing data from multiple users, StreetBump provides valuable insights to authorities for prioritizing road maintenance efforts. The mobile app platform allows users to conveniently report road issues on-the-go, contributing to safer and smoother roads for the community.

Tools and Technologies:

The development of StreetBump or similar apps might involve using various tools and technologies, including:

Mobile App Development:

For building native mobile apps on iOS and Android platforms, technologies like Swift or Objective-C (for iOS) and Java or Kotlin (for Android) might be used.

Cross-platform Development:

Alternatively, frameworks like React Native or Flutter may be used to develop a single codebase that works on both iOS and Android platforms.

Sensors and Data Collection:

The app could utilize smartphone sensors like GPS, accelerometer, and gyroscope to collect data while driving and detect road conditions.

Front-end Development:

HTML, CSS, and JavaScript (for web views in the app) might be used to create the user interface.

Back-end Development:

Server-side programming languages like Python, Ruby, or Node.js, along with web frameworks like Django, Ruby on Rails, or Express.js, could be used for the back-end development.

Database:

A database management system like PostgreSQL, MySQL, or MongoDB might be used to store and manage user data and reported road conditions.

Geolocation and Mapping:

APIs like Google Maps or Mapbox might be integrated to display road conditions on a map and allow users to visualize the data.

User Authentication and Security:

Techniques like OAuth or token-based authentication could be implemented for secure user login and data protection.

Testing and Quality Assurance:

Automated testing frameworks like Jest or XCTest could be used to perform testing and ensure app functionality and performance.

Deployment and Distribution: The app could be deployed on app stores (e.g., Apple App Store and Google Play Store) for public use.

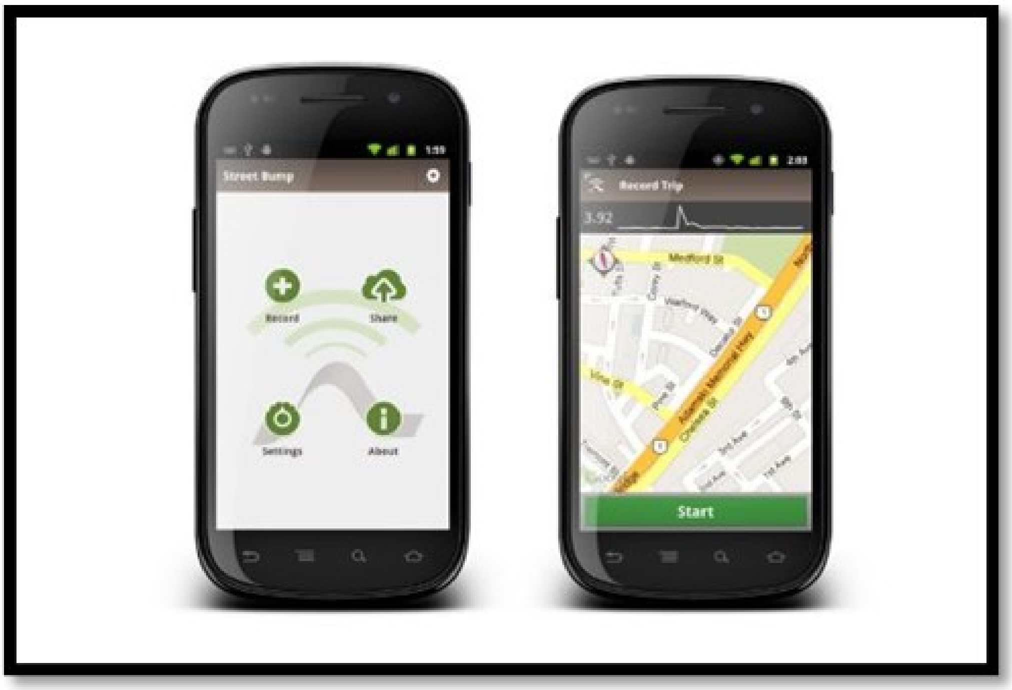


Figure 2.4: StreetBump

2.5 Development Procedure:

There is no particular data on how these applications were made. Yet, there is a structure which reflects how such applications can be made. A portion of those systems are:

Requirements Gathering:

The development team gathers requirements from stakeholders, including citizens, local government authorities, and administrators. They identify the key features needed for the app.

Design and Planning:

The app's architecture and design are planned, including the user interface (UI) design, database design, and overall system flow. The team chooses a technology stack suitable for web or mobile app development.

Front-end Development:

The front-end of the app is developed using web technologies (for web apps) or native technologies (for mobile apps). The UI is designed to be user-friendly and intuitive.

Back-end Development:

The backend of the app is developed to handle user requests, process data, and interact with the database. The team chooses a programming language and a web framework for back-end development.

Database Integration:

The app integrates with a database system to store and manage user data, reported issues, and other relevant information.

Geolocation and Mapping:

Geolocation features are implemented to allow users to pinpoint the location of the reported issues on a map.

User Authentication and Authorization:

User authentication and authorization mechanisms are put in place to ensure secure access and protect user data.

Testing and Quality Assurance:

The app undergoes testing to identify and fix bugs, ensure smooth functionality, and validate its performance.

Deployment and Launch:

Once the app is thoroughly tested, it is deployed on servers or app stores for public use.

User Engagement and Feedback:

After launch, users are encouraged to provide feedback on the app's performance, usability, and reported issues.

Continuous Improvement:

After the app's launch, feedback from users and administrators is collected to make continuous improvements and updates to enhance its functionality and address any emerging needs.

2.6 Comparative Study

The articles we covered for the comparative study are shown in the Table 2.1 as under:

Ref	**Title**	**Developer's Name**	**Year**	**Feature**
[1]	CCMS	PITB	2013	Allows citizens to register.
[2]	FixMyStreet	mySociety	2007	Enables users to report and track.
[3]	SnapFix	Love Clean Streets	2013	Easily capture and report issues.
[4]	StreetBump	New Urban Mechanics	2012	Identify and report potholes.

TABLE 2.1: Comparative study

Chapter 3

Requirement Specification

System requirements are configurations that a system needs in order to function properly and effectively. Failure to comply with these requirements may cause installation issues or performance issues. System requirements are documents or sets of documents that outline the features and behavior of a system or software application.

3.1 Existing System

The current, manual, household maintenance complaint management system suffers from several downsides and impediments that hinder its effectiveness. The system relies heavily on manual processes, requiring citizens to physically visit municipal offices or call designated helplines to report issues. This cumbersome approach consumes valuable time as well as results in an absence of real-time updates for citizens regarding the situation with their complaints. As a consequence, residents often feel disconnected from the resolution process, leading to reduced trust and satisfaction with the system.

Moreover, the manual handling of complaints leads to inefficiencies in issue prioritization and classification. Without an automated system, basic issues may not receive immediate attention, causing delays in resolution and impacting the overall nature of service delivery. Moreover, the absence of a centralized stage for complaint management makes it challenging for local authorities to allocate resources and track the progress of reported complaints efficiently.

Overall, the constraints of the existing system underscore the need for an innovative arrangement that streamlines the complaint reporting process, ensures real-time updates for citizens, automates issue prioritization, and fosters greater local area engagement in household maintenance management. The proposed, Fixome, app means to address these downsides and revolutionize how household maintenance complaints are handled, providing a more transparent, efficient, and citizen-centric approach.

3.2 Proposed System

Our proposed system, Fixome, offers a comprehensive answer for covering the holes and limitations of the existing household maintenance complaint management system. The app has introduced several key features to enhance the overall efficiency and effectiveness of the complaint reporting and resolution process.

Right off the bat, "Fixome" replaces manual processes with a user-friendly mobile-based stage, enabling citizens to report complaints conveniently from their smartphones. This computerized approach eliminates the need for actual visits or phone calls, streamlining the complaint reporting process and saving valuable time for the two citizens and authorities.

Secondly, the app provides real-time updates and notifications to users, keeping them informed about the progress of their complaints. This feature fosters transparency and instills confidence in the system, ensuring that citizens are actively engaged all through the resolution process.

Automated prioritization and classification are another key aspect of the app. By leveraging an intelligent system, Fixome categorizes complaints based on urgency and effect, allowing local authorities to address basic issues quickly and allocate resources efficiently.

Moreover, the app upholds geotagging and multimedia capabilities, allowing users to provide precise location details and connect images or videos to their complaints. This comprehensive documentation ensures better understanding and faster resolution of reported issues.

By covering these holes and impediments of the existing system, Fixome means to change household maintenance complaint management, promoting a cleaner, safer, and more responsive living environment for all residents.

3.3 Requirement Specifications

A requirements specification's goal is to give all parties involved—including users and developers—a shared understanding of what the system should look like. A requirements definition is also crucial for the implementation phase and the testing phase that follows (where the functionality is compared to the predefined requirements). It's crucial to remember that a specification can never take the place of effective, ongoing communication between the parties.

3.3.1 Functional Requirements

Following is the list of functional requirements of each of the stakeholders.

FR-01: Admin

- **Login**

  Admin has been able to login to his/her account.

- **Add user**

  Admin has been able to Add new user.

- **Delete User**

  Admin has been able to Delete registered users.

- **View Feedback**

  Admin has been able to View responses or feedback given by customers.

- **Urgent Complaints**

  Admin has been able to approve and view Urgent basis complaints.

- **Normal Complaints**

  Admin has been able to approve and view Normal basis complaints.

- **Completion Notification**

  Admin has been able to receive a notification after the complaint is dealt with.

- **Complaints Fulfilled**

  Admin has been able to mark the complaint as Fixed or Fulfilled.

- **Complaints Pending**

  Admin has been able to keep the complaints in Pending state.

- **Complaints Rejected**

  Admin has been able to reject the complaints due to certain reasons.

- **Update Team Information**

  Admin has been able to update the team information.

- **Logout**

  Admin has been able to Logout from his/her account.

FR-02: User

- **Register**

  User has been able to Register to the application.

- **Login**

  User has been able to Login to his/her account.

- **Register Complaint**

  Use has been able to Register or File complaint via Pictures, Videos, and brief Description of the issue.

- **View Team**

  User has been able to view the Head of the Team and total number of team members.

- **Track Complaint Status**

  User has been able to Track his/her registered complaint's status (i.e., Confirmed, On the way, Complete) via a Tracking ID.

- **Edit Personal Information**

  User has been able to update his/her personal information.

- **Delete Account**

  User has been able to delete his/her account.

- **Give Feedback**

  User has been able to give feedback after the complaint is fixed.

- **Contact Admin**

  User has been able to contact Admin via a phone number.

- **Forgot Password**

  The user has been able to recover his/her password in case if the user forgets it.

- **Logout** User has been able to logout from his/her account.

FR-03: Team

- **Login**

  Team has been able to login to the profile.

- **View Complaint Location**

  Vendor has been able to view the location where the complaint needs to be fixed at.

- **Click Proof Picture**

  Vendor has been able to click the picture after the complaint is fixed, for proof.

- **Mark as Fixed or Pending**

  Vendor has been able to mark the complaint as Fixed or Pending.

- **Submit**

  Vendor has been able to send the status to the Admin by clicking Submit button.

- **Logout**

  Vendor has been able to logout from the profile.

3.3.2 Non-Functional Requirements

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability.

3.3.3 Performance Requirement

The Fixome app should be loaded within 10 seconds. Initially, the app should allow up to 1000 users, easily. The app has been updated consistently to maintain a strategic distance from issues. In addition to the above, the mobile app should have the following abilities and capabilities.

- **NFR-1:**

  The responsiveness of the app should be high, and the app should respond according to the user's action.

- **NFR-2:**

  The complaints should be dealt with priority basis after meeting a certain criterion.

- **NFR-3:**

  The response time on the app shall be minimal.

- **NFR-4:**

  Consistency on the mobile app shall be maintained across all the pages/screens.

- **NFR-5:**

  The layout of the mobile app shall be kept simple and must be self-explanatory.

3.3.4 Security Requirements

Our app also considers safety requirements as a crucial part of fulfilling non-functional requirements. We register users on the basis of their CNIC number, which has been kept confidential, to register only authentic users. This decreases the load of the app and enhance performance. 2-way Authentication has been used for logging in to our app.

3.3.5 Software Quality Attributes

- Usability:

  User has been able to provide his/her information while registering.

- Performance:

  User has been able to login to his/her account.

- Security:

  User has been verified by the Admin.

3.4 Use Case Diagram

Use case describes the actions that can be performed by the actor. In our system we have three types of actors Admin, User and Team. It tells the interaction between actor and the system to accomplish the goal.

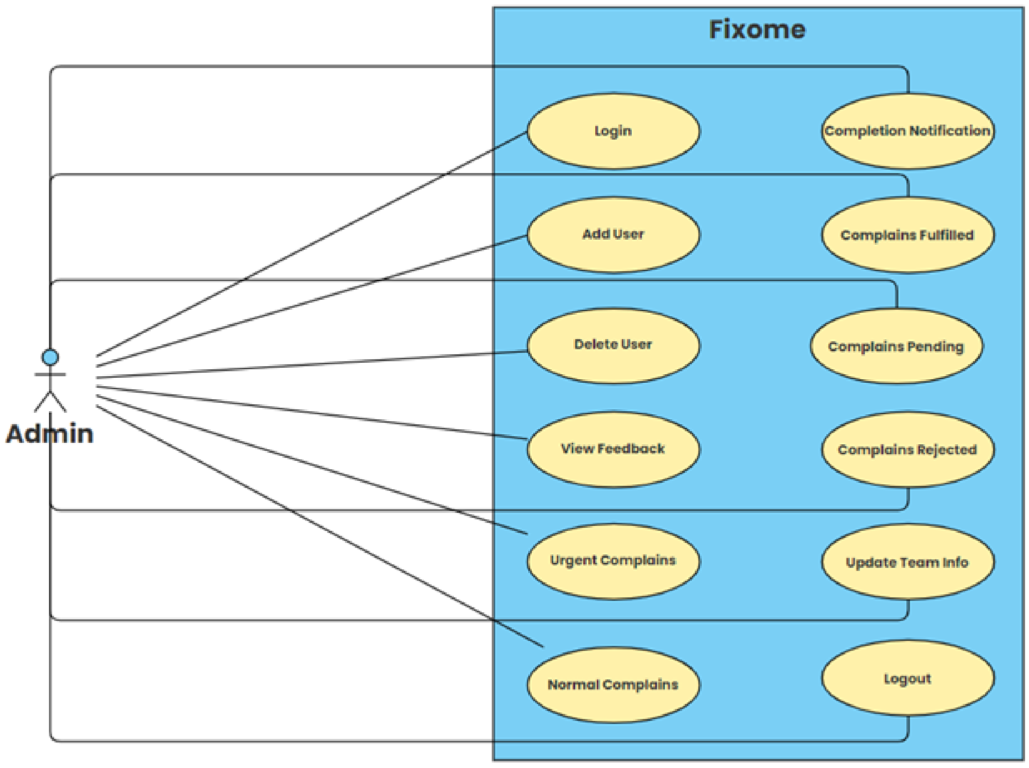


FIGURE 3.1: Shows the interaction of Admin with the application

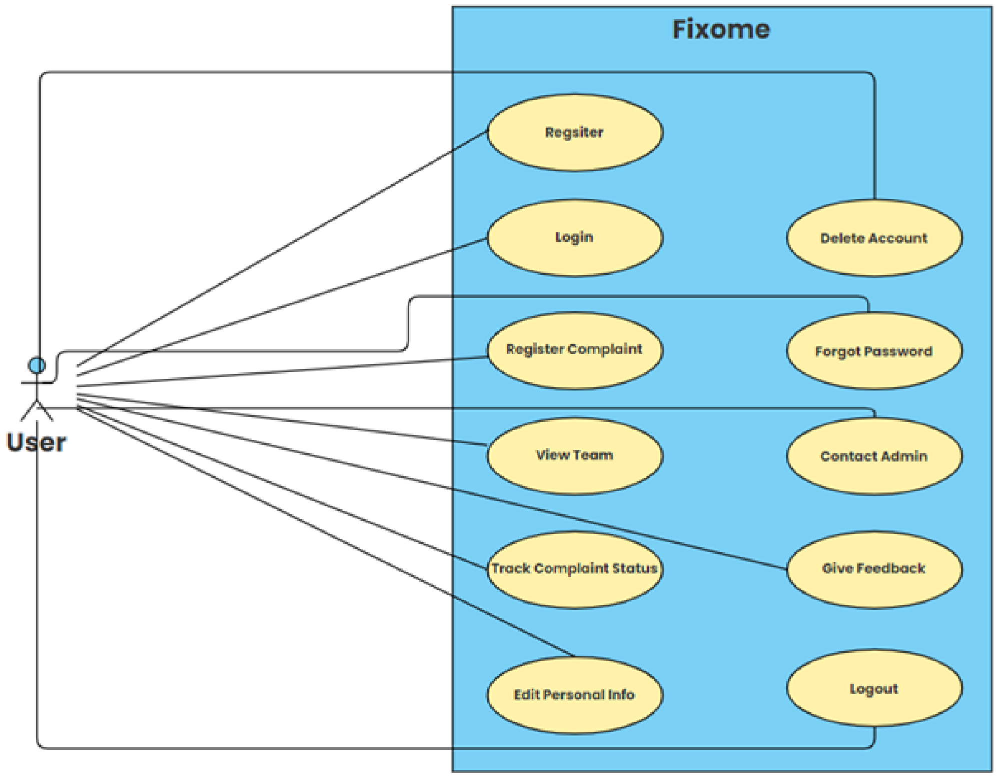


FIGURE 3.2: Shows the interaction of User with the application

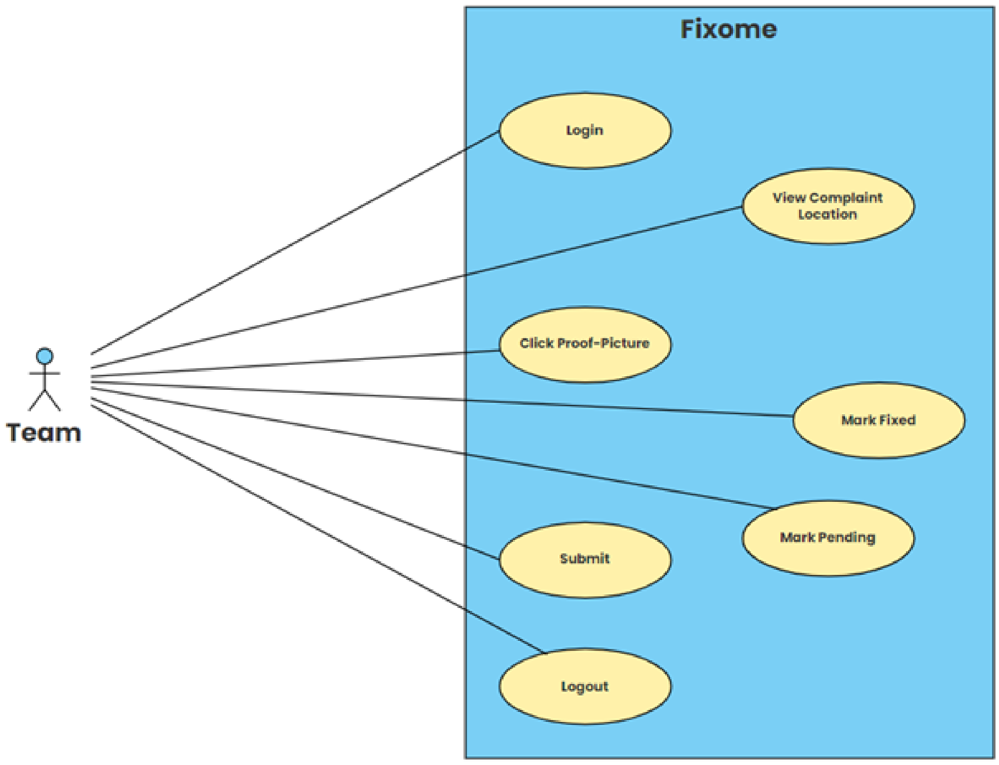


FIGURE 3.3: Shows the interaction of Team with the application

3.5 Descriptive Use Case

This section gives the explanation in tabular form for the use cases we made. In these explanatory tables, we provided a brief description of the use cases, and which actor has the ability to achieve it. The pre-and post-condition to accomplish the task and the steps defining how the actor conducts the activity are shown in the following tables

3.5.1 Admin Management

Extended use case of the functionality for login performed by the actor admin shown in table 3.1.

Use Case ID	UC-1
Use Case Name	Login
Actor(s)	Admin
Pre-Conditions	Admin has added information to login
Priority	High
Basic Flow/Inputs	
1.	Opens the mobile app
2.	Enters Email, Password and Key
3.	Click Login. Logged into mobile app
Actor Actions	
Admin inputs email address, password, and key	
System Response	
System checks admin's provided credentials.	
Alternative Course of Action (if any)	
N/A	

Table 3.1: Admin Login

Use Case ID	UC-2
Use Case Name	Add New User
Actor(s)	Admin
Pre-Conditions	Admin is logged in as admin
Priority	High
Basic Flow/Inputs	
1.	Admin is at Dashboard
2.	Admin clicks on Users
3.	List of Users has been displayed
4.	Admin clicks on Requests
5.	Admin checks the necessary requirements for approval
6.	If authentic, clicks Approve
7.	If not authentic, clicks Discard and gives reason
Actor Actions	
1. Admin clicks the Approve button after if the requirements are fulfilled and authentic.	
2. clicks the Discard button and gives reason if the requirements are not authentic	
System Response	
System checks admin's provided credentials.	
Alternative Course of Action (if any)	
N/A	

Table 3.2: Add a new user

Extended use case of the functionality for Delete User performed by the actor admin shown in table 3.3.

Use Case ID	UC-3
Use Case Name	Delete User
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow/input	
1.	Admin is at the Dashboard
2.	Admin clicks on Users
3.	List of Users displays
4.	Admin clicks on Delete Users
5.	Admin selects the User (s) to delete from the database
6.	Admin clicks on Delete Button
7.	Selected Users has been deleted
Actor action	
1. Admin deletes the user	
System Response	
System display message “User Deleted Successfully’	
Alternative Course of Action (if any)	
N/A	

Table 3.3: Delete user

Extended use case of the functionality for View Feedback performed by the actor admin shown in table 3.4.

Use Case ID	UC-4
Use Case Name	View Feedback
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic flow / inputs	
1.	Admin is at the Dashboard
2.	Admin clicks on Feedback
3. Admin sees the feedback given by Users along with their Name and Date.	
Actor Action	
1. Admin selects Feedback	
System Response	
1. The system displays all the Feedback of the Users successfully	
Alternative Course of Action (if any)	
N/A	

Table 3.4: Admin: view feedback

Extended use case of the functionality for Urgent Complaints performed by the actor admin shown in table 3.5.

Use Case ID	UC-5
Use Case Name	Urgent Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic flow/input	
1.	Admin is on dashboard
2.	Admin clicks on Complaints
3.	From dropdown, admin selects Urgent
4.	Admin sees the certain complaint details
5. If belongs to Urgent, admin clicks Approve button	
Actor Action	
1. Admin marks the complaint as Urgent	
System Response	
1. System shifts the complaint 'Urgent' Category by displaying a message 'shifted to urgent	
Alternative Course of Action (if any)	
N/A	

Table 3.5: Admin 'Urgent Complaints'

Extended use case of the functionality for Normal Complaints performed by the actor admin shown in table 3.6.

Use Case ID	UC-6
Use Case Name	Normal Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic flow/input	
1.	Admin is on dashboard
2.	Admin clicks on Complaints
3.	From dropdown, admin selects Normal
4.	Admin sees the Normal basis complaint list
Actor Action	
1. Admin clicks to see the Normal basis complaints	
System Response	
1. System displays the Normal basis complaints	
Alternative Course of Action (if any)	
N/A	

TABLE 3.6: Admin 'Urgent Complaints'

Extended use case of the functionality for Completion Notification performed by the actor admin shown in table 3.7.

Use Case ID	UC-7
Use Case Name	Completion Notification
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
1.	Admin is on dashboard
2.	Admin sees a number bubble on Notification icon
3.	Admin clicks to open it
4.	Admin sees the Complaint number, marked as Fixed or Pending
5.	Admin clicks Manage
6.	Admin clicks Fixed or Pending according to the message
Actor Action	
1. Admin receives a pop up on Notification icon	
2. Admin clicks the Fixed button	
3. Admin clicks the Pending button	
System Response	
1. System displays the Complaint number, marked as Fixed or Pending.	
2. System shows ‘Complaint is Fixed’	
3. System show ‘Complaint in Pending’	
Alternative Course of Action (if any)	
N/A	

Table 3.7: Admin:Completion Notification

Extended use case of the functionality for Complaints Fulfilled performed by the actor admin shown in table 3.8.

Use Case ID	UC-8
Use Case Name	Complaints Fulfilled
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow/input	
1.	Admin is on dashboard
2.	Admin clicks on View
3.	Admin clicks on Complaints Fulfilled
4.	Admin sees all the complaints marked as fixed in the list
Actor Action	
1. Admin clicks the View button > Complaints Pending	
System Response	
1. System shows all the complaints which have been fixed along with date and time	
Alternative Course of Action (if any)	
N/A	

Table 3.8: Admin 'Complaints Fulfilled'

Extended use case of the functionality for Complaints Pending performed by the actor admin shown in table 3.9.

Use Case ID	UC-9
Use Case Name	Complaints Pending
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow/input	
1.	Admin is at dashboard
2.	Admin clicks on View
3.	Admin clicks Complaint Pending
Actor Action	
1. Admin clicks the View button > Complaints Pending	
System Response	
1. System shows all the complaints which have been fixed along with date and time	
Alternative Course of Action (if any)	
N/A	

Table 3.9: Admin 'Complaint Pending'

Extended use case of the functionality for Complaints Rejected performed by the actor admin shown in table 3.10.

Use Case ID	UC-10
Use Case Name	Complaints Rejected
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow/input	
1.	Admin is at dashboard
2.	Admin clicks on View
3.	Admin clicks Complaint Rejected
Actor Action	
1. Admin clicks the View button > Complaints Rejected	
System Response	
1. System shows all the complaints which have been Rejected due to certain reasons	
Alternative Course of Action (if any)	
N/A	

Table 3.10: Admin 'Complaint Rejected'

Extended use case of the functionality for Update Team Info performed by the actor admin shown in table 3.11.

Use Case ID	UC-11
Use Case Name	Update Team Info
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Low
Basic flow/input	
1	Admin is at dashboard
2	Admin clicks on Team button
3	Admin clicks Edit on top right corner
4	Admin clicks the Edit icon against the member's info
5	Admin makes the changes the clicks Save button
Actor Action	
1.Admin edits the member info	
System Response	
1. System shows,'Updated Successfully'	
Alternative Course of Action (if any)	
N/A	

TABLE 3.11: Admin 'Update Team Info'

Extended use case of the functionality for Logout performed by the actor admin shown in table 3.12.

Use Case ID	UC-12
Use Case Name	Logout
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Low
Basic flow/input	
1.	The admin is done using the mobile application
2.	The admin clicks on the profile icon
3.	The admin clicks on logout option at the end
4.	The system logs the admin out
Actor Action	
1. Admin clicks on the log out button	
System Response	
1. System successfully displayed the log in screen	
Alternative Course of Action (if any)	
N/A	

TABLE 3.12: Admin 'Logout'

3.5.2 User Management

Extended use case of the functionality for Registration performed by the User shown in table 3.13.

Use Case ID	UC-13
Use Case Name	Registration
Actor(s)	User
Pre-Conditions	User adds personal information for registration
Priority	High
Basic flow /input	
1.	User opens application
2.	User can choose to register as a User
3.	The user enters Email and Password
4.	The user enters Name and CNIC
5.	The user clicks Register button
6.	The system verifies entered email and CNIC and registers the user
7.	System displays a message that "Your account has been registered successfully"
Action Actor	
1. The user enters personal details in the relevant field. The user clicks on the "Register" button	
System Response	
1. System checks user's provided credentials if the user's credentials	
Alternative Course of Action (if any)	
N/A	

TABLE 3.13: Registration Table

Extended use case of the functionality for Login performed by the User shown in table 3.14

Use Case ID	UC-14
Use Case Name	Login
Actor(s)	User
Pre-Conditions	User adds personal information to login
Priority	High
Basic flow/input	
1.	User opens mobile application
2.	Enters Email/CNIC and password
3.	Click Login. Logged into mobile application
Actor Action	
1. User inputs Email/CNIC and Password	
System Response	
1. System checks user's provided credentials are valid, home screen or dashboard is displayed	
Alternative Course of Action (if any)	
N/A	

Table 3.14: Registration Table

Extended use case of the functionality for Complaint Registration performed by the User shown in table 3.15

Use Case ID	UC-15
Use Case Name	Complaint Registration
Actor(s)	User
Pre-Conditions	Actor logged in as User Must see the criteria for Urgent basis complaints
Priority	High
Basic flow/input	
1.	User is on dashboard
2.	User clicks Register Complaint button
3.	User selects the Category i.e., Water, Gas etc.
4.	User Clicks pictures or video and click on upload button
5.	Selected media is uploaded
6.	Users write a brief description explaining the issue clearly
7.	User Enters his/her Name, Address, Nearby place (optional), Phone number
8.	User clicks on type of complaint i.e., Normal or Urgent.
9.	User submit and must wait for the approval and Complaint-ID from Admin
Actor Action	
1. User files or registers a complaint	
System Response	
1. System displays a notification and give complain ID-1234	
Alternative Course of Action (if any)	
N/A	

Table 3.15: Complaint Registration

Extended use case of the functionality for Track Complaint performed by the User shown in table 3.16

Use Case ID	UC-16
Use Case Name	Track Complaint
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow/input	
1.	User is on the dashboard
2.	Complaint has been registered
3.	User has the Complaint-ID
4.	User clicks on Complaints
5.	User sees track complaints option
6.	User enters the Complaint-ID
7.	App loads for 3-4 seconds
8.	User can now see the status of the registered complaint.
Actor Action	
1. User tracks complaint	
System Response	
1.App loads for 3-4 seconds and then the app prompts a message.	
Alternative Course of Action (if any)	
N/A	

Table 3.16: Track Complaint

Extended use case of the functionality for View Team performed by the User shown in table 3.17

Use Case ID	UC-17
Use Case Name	View Team
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow/input	
1.	User is on dashboard or home screen
2.	User clicks on Team icon
3.	User has been able to see the category of the complaints
4.	User selects any one category
5.	User able to see the total team member or workers of that category
Actor Action	
1. User views Team	
System Response	
1. System displays the most expert of team along with total team members of that category	
Alternative Course of Action (if any)	
N/A	

Table 3.17: View Team

Extended use case of the functionality for Edit personal Info performed by the User shown in table 3.18

Use Case ID	UC-18
Use Case Name	Edit Personal Info
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow/input	
1.	User is on the dashboard
2.	User clicks on Profile icon at bottom right corner
3.	User clicks Personal Information
4.	User clicks Edit
5.	User able to see the last updated info (Name, Email,Address,etc.)
6.	User can make changes in his/her personal info
7.	App asks for the password to make changes
8.	User enters the correct password
9.	App verifies the password
10.	Password is correct
11.	User clicks Save button
12.	App loads for 2-3 seconds
13.	User sees a message, "Information Updated Successfully"
Actor Action	
1. User make changes in his/her personal information	
System Response	
Information Updated Successfully"	
Alternative Course of Action (if any)	
N/A	

Table 3.18: Edit Personal Info

Extended use case of the functionality for Delete Account performed by the User shown in table 3.19

Use Case ID	UC-19
Use Case Name	Delete Account
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow/input	
1.	User is on dashboard
2.	User clicks Personal Info
3.	User clicks Account information
4.	User clicks Delete Account
5.	User sees app demanding for Email, CNIC and Password
6.	User enters the correct details
7.	User clicks Delete button
8.	App asks for confirmation
9.	User clicks Yes
10.	App loads for 3-5 seconds
11.	User gets a message “Your account has been deleted successfully”
Actor Action	
1. User deletes his/her account	
System Response	
1.System deletes the account successfully by saying, “Your account has been deleted successfully”	
Alternative Course of Action (if any)	
N/A	

Table 3.19: Delete Account

Extended use case of the functionality for Forgot Password performed by the User shown in table 3.20

Use Case ID	UC-20
Use Case Name	Forgot Password
Actor(s)	User
Pre-Conditions	The app is open, and user is on Login page
Priority	Medium
Basic flow/input	
1.	Enters correct Email or CNC
2.	Enters wrong password
3.	Click login
4.	Login gets failed
5.	The user clicks on forgot password
6.	A new page opens in which user enter email and then click on the Go button
7.	User receives the mail through which they can recover their password
Actor Action	
1. The user enters the correct email or CNIC but wrong password	
2. The user forgot the password	
3. Users logs in with new and correct details	
System Response	
1. Displays “Either Email/CNIC or Password is invalid”	
2. The password is recovered	
3. The Home screen/Dashboard is displayed	
Alternative Course of Action (if any)	
N/A	

Table 3.20: Forgot Password

Extended use case of the functionality for Give Feedback performed by the User shown in table 3.21

Use Case ID	UC-21
Use Case Name	Give Feedback
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow/input	
1.	The user is on dashboard
2.	The user selects give feedback
3.	User gives his/her thoughts after the complaint has been fixed
4.	User clicks Submit button
5.	System says, “Feedback sent successfully!”
Actor Action	
1. The user selects Feedback and enters feedback	
System Response	
1. A message has been displayed “Feedback sent successfully!”	
Alternative Course of Action (if any)	
N/A	

Table 3.21: Give Feedback

Extended use caseof the functionality for Contact Us performed by the User shown in table 3.22

Use Case ID	UC-22
Use Case Name	Contact Us
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow/input	
1.	User is on dashboard
2.	User click Contact icon
3.	User sees three options i.e., Help center, Email, Call Helpline
4.	User selects one of them and proceed accordingly
Actor Action	
1. User contacts for the help	
System Response	
1. System displays the ways to contact or seek help via three options Help Center, Email, Call.	
Alternative Course of Action (if any)	
N/A	

Table 3.22: Contact Us

Extended use case of the functionality for Logout performed by the User shown in table 3.23

Use Case ID	UC-23
Use Case Name	Logout
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow/input	
1.	User clicks Profile icon
2.	At the end, user sees logout button
3.	User clicks the button, checks the Yes button, and log out after 2-3 seconds
Actor Action	
1. User clicks on the logout button	
System Response	
1. System successfully displayed the login screen	
Alternative Course of Action (if any)	
N/A	

Table 3.23: Logout

3.5.3 Team Management

Extended use case of the functionality for Login performed by the Team shown in table 3.24

Use Case ID	UC-24
Use Case Name	Login
Actor(s)	Team
Pre-Conditions	Member adds personal information to login
Priority	High
Basic flow/input	
1.	Team member selects the Complaint Cell
2.	Team member enters the Team Code
3.	Clicks Login
4.	Member can now see the dashboard
Actor Action	
1. Member login with correct credentials	
System Response	
1. Member can now see the dashboard screen of the app	
Alternative Course of Action (if any)	
N/A	

Table 3.24: Login

Extended use case of the functionality for Complaint Details performed by the Team shown in table 3.25

Use Case ID	UC-25
Use Case Name	Complaint Details
Actor(s)	Team
Pre-Conditions	Member is logged in
Priority	High
Basic flow/input	
1.	Member is on dashboard
2.	Member clicks Complaint
3.	Member sees the Pending complaints
4.	Member selects the complaint
5.	Member can now see the Name, Address and Issue explained
6.	Member now clicks “On the way” status to notify User and Admin
Actor Action	
1. Member checks the complaint details	
System Response	
1. Member can see the Name, Address and Issue explained of the registered complaint	
Alternative Course of Action (if any)	
N/A	

Table 3.25: Complaint Details

Extended use case of the functionality for Proof-Picture performed by the Team shown in table 3.26

Use Case ID	UC-26
Use Case Name	Proof-Picture
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	High
Basic flow/input	
1.	Member arrived at the location
2.	Member fixed the complaint
3.	Member clicks picture of object after it has been fixed the User is in picture.
4.	Member writes Remarks (Fixed or Pending) beneath the picture
5.	Member clicks Submit
6.	The submitted data reaches the Admin's Notification module
Actor Action	
1. Member clicks proof picture	
System Response	
1. The submitted data (proof-picture and remarks) reaches the Admin's side	
Alternative Course of Action (if any)	
N/A	

Table 3.26: Logout

Extended use case of the functionality for Logout performed by the Team shown in table 3.27

Use Case ID	UC-27
Use Case Name	Logout
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	Medium
Basic flow/input	
1.	Member clicks on the Profile icon
2.	On the bottom, Member could see the Logout button
3.	Member clicks that button
4.	Member performs confirmation i.e., clicks Yes
5.	App loads for 3-4 seconds
6.	Member can see the Login page successfully
Actor Action	
1. Member logs out from the app	
System Response	
1. Member could see the Login Page successfully	
Alternative Course of Action (if any)	
N/A	

Table 3.27: Logout

Chapter 4

System Design

Systems design is the procedure of delineating a system's structure, components, modules, connections, and data with the intention of fulfilling pre-established specifications. We shall define this part of our project here. The process through which a system's components, including its architecture, modules, and parts, were correctly recognized, and categorized is known as system design. It is the transformation of user requirements into some appropriate shape. Designing very rarely uses a systematic approach, sometimes top-down approach is preferred and sometimes the bottom-up approach. In this chapter, we look at the Design of our system including its user interface either low or high-fidelity designs. The design approaches and constraints are discussed in detail. The Systems Class, Sequence, Data flow and ER Diagrams are shown along with Projects logical and functional flows. This Chapter has prodigious importance as in the Software Development Life Cycle the designing phase is one of the highly important phases. While designing the project User Interface it is very important for the designer to consider its use and should know their technical and cultural approach. While designing our Fixome app we tried our best to use all the performance standards of the software industry and fulfil the needs of its user.

4.1 System Architecture

Here, the system is in narrative form using non-technical terms. Figure 4.1 shows the three main Users of our application are User, Admin and Team. We have used Firebase for storing and manipulating the data. Both interacts with the application using mobile interface. The information has been stored within the Firebase database. There's a mere chance that at the implementation time, our database could change from Firebase to some other potential database.

It can be seen from a high-level system architecture diagram.

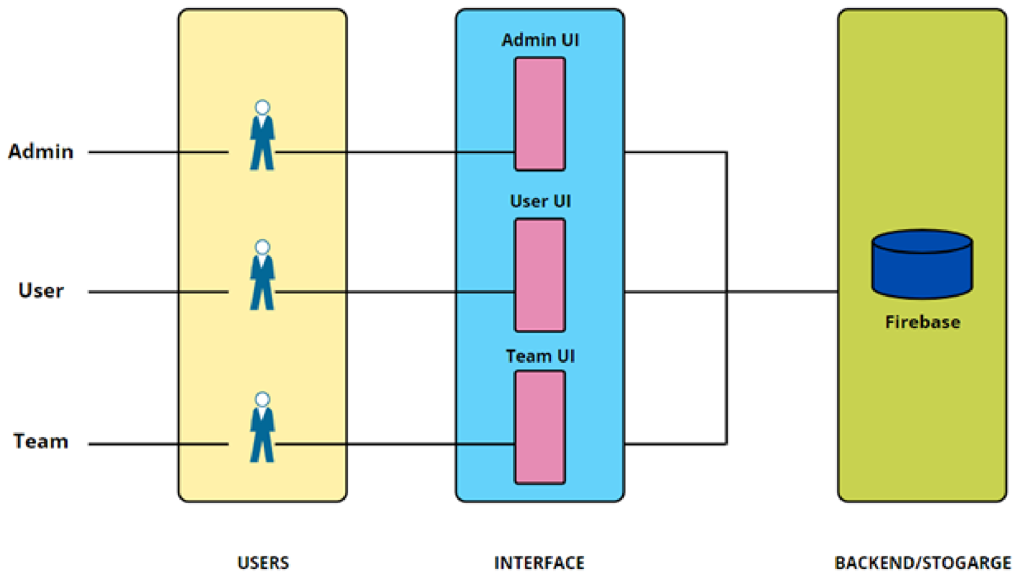


FIGURE 4.1: High Level System Architecture

Now the High-level context diagram is shown in Figure 4.2 with primary functions that can be performed by the Admin, User and Team. The User can file a complaint and request to the Admin for the particular solution he wants to avail. The requests have been sent to the Admin he can accept or discard the request. Then the status has been displaying to the User. Once the problem has been fixed successfully by the Team Worker, the application stores the response given by Team Worker. The details have been stored and fetched from the Firebase.

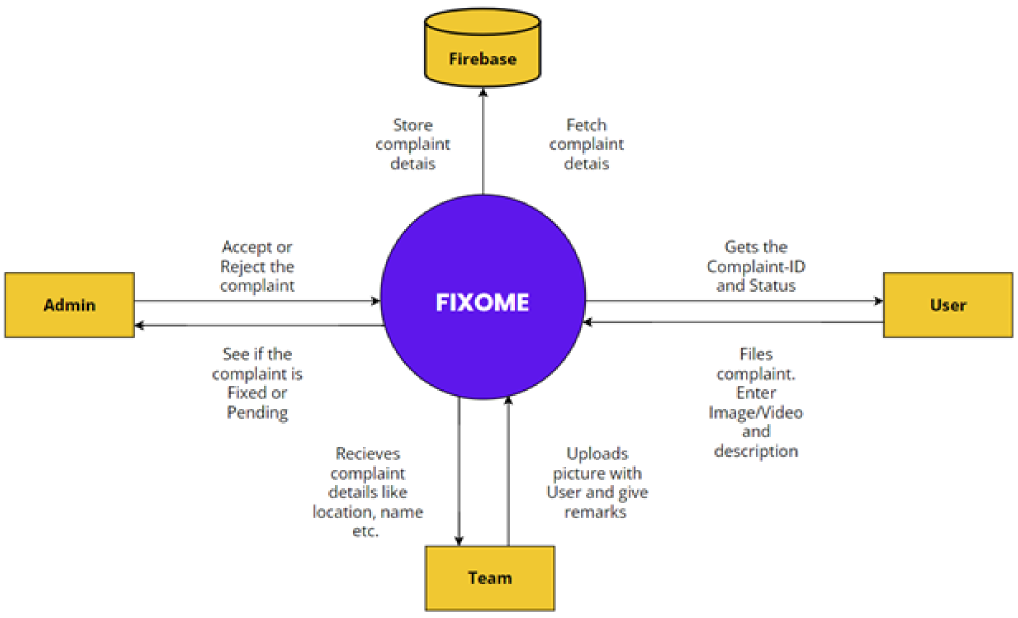

Firebase
Store complaint detais
Fetch complaint detais
Accept or Reject the complaint
Admin
FIXOME
Gets the Complaint-ID and Status
User
See if the complaint is Fixed or Pending
Files complaint. Enter Image/Video and description
Recieves complaint details like location, name etc.
Uploads picture with User and give remarks
Team

Figure 4.2: High Level Context Diagram

4.2 Design Constraints

The software design process can be perceived as a series of well-defined steps. The Fixome app has three modules that were designed separately keeping in mind the ease of their users. Constraints are shown in the Table 4.1.

Reliability	There should be reliable connection between app and th database.
Criticality of the Application	The app should respond and communicate the data immediately.
Security and Safety	Connection between the database,application should be secured.
Internet Connection	There must be a good internet connection.

Table 4.1: Design Constraints

4.3 Design Methodology

The Agile model is a project management methodology specifically chosen for the development of complex software. This framework enables iterative processes, significantly reducing the occurrence of common mistakes and errors.

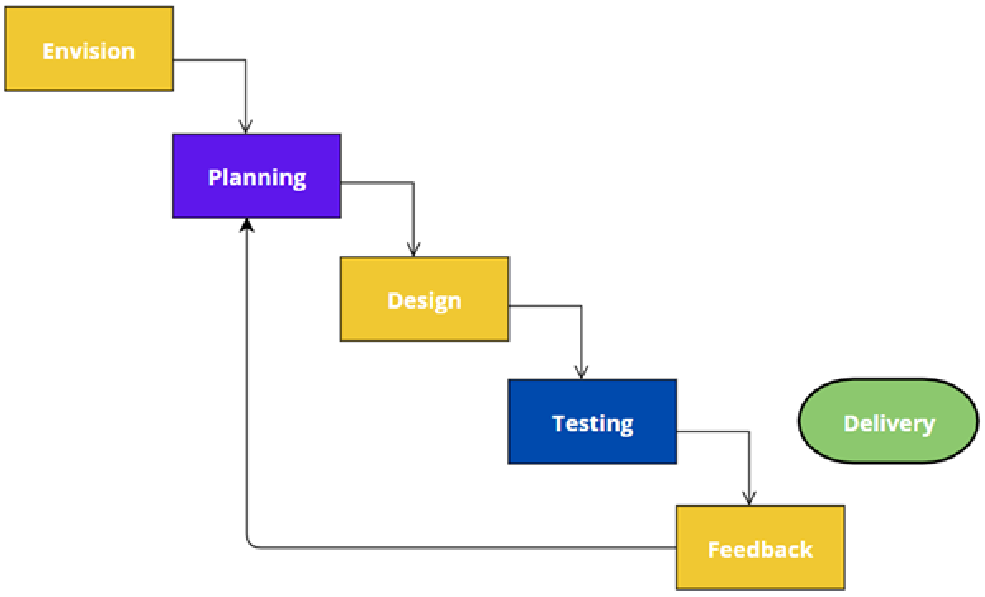


FIGURE 4.3: Agile Method for Development

The model as shown in Figure 4.3 divides the project into a series of development cycles or short time boxes, which are assigned to each professional on the project team. It's a collaborative approach that enables quick responses to rapid changes. It is flexible enough to accommodate changes in project requirements throughout the mobile app development life cycle.

4.4 High Level Design

This section shows the high-level design of our system. The best high-level designs seek to model groups of system components from a variety of viewpoints. Common views of view are:

4.4.1 Component Diagram

Our application has the models of Admin, Users, Team Worker, and complaints details that are stored in the Firebase. These modules can interact with each other. The component diagram of our application is shown in Figure 4.4.

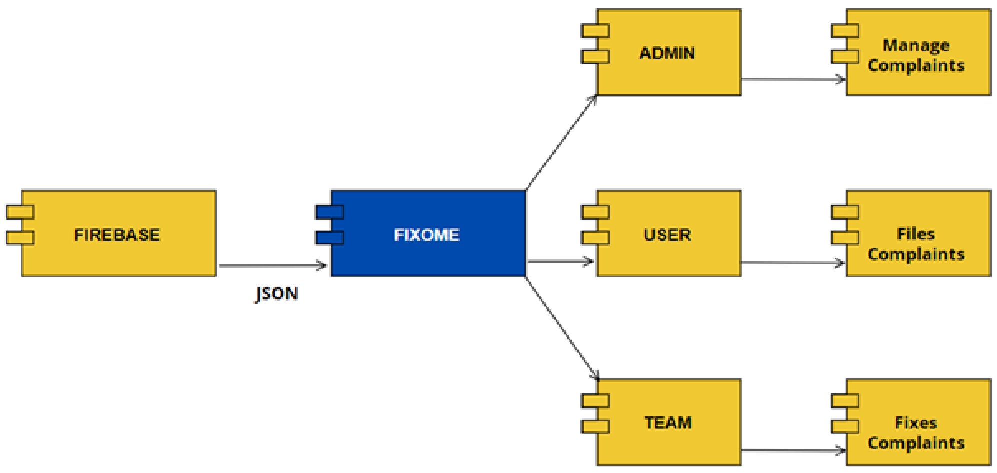


FIGURE 4.4: High Level Component Diagram

4.4.2 Sequence Diagram

A sequence diagram visually represents the interaction between objects in a sequential manner, capturing the order in which these interactions occur. Different types if interactions have been performed by using our application. The user interacts with the system to perform various tasks. This section provides the detail view if how the user interacts with the Fixome and the response by our application.

Figure 4.5 shows the interaction between user and the system when the user registers his/her account. To register an account user needs to provide certain information. The system adds the user if he/she is not already registered.

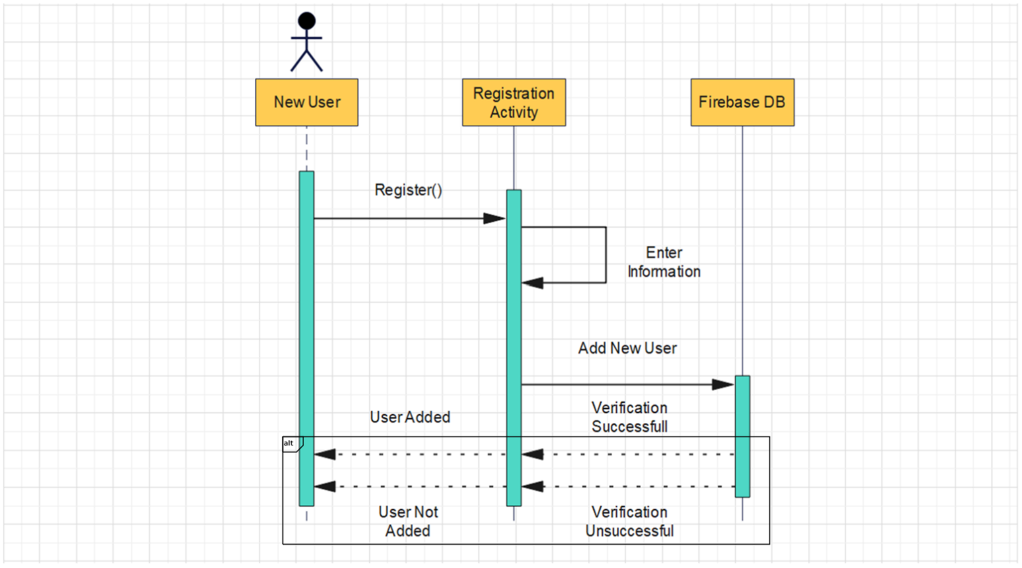


FIGURE 4.5: Sequence Diagram for Registration

Figure 4.6 shows the interaction between user and the system when user want to login his/her account. To login the user needs to enter credentials. The credentials are validated. If the credentials are correct the user successfully logs-in to the account but if they are not correct the error message has been displayed.

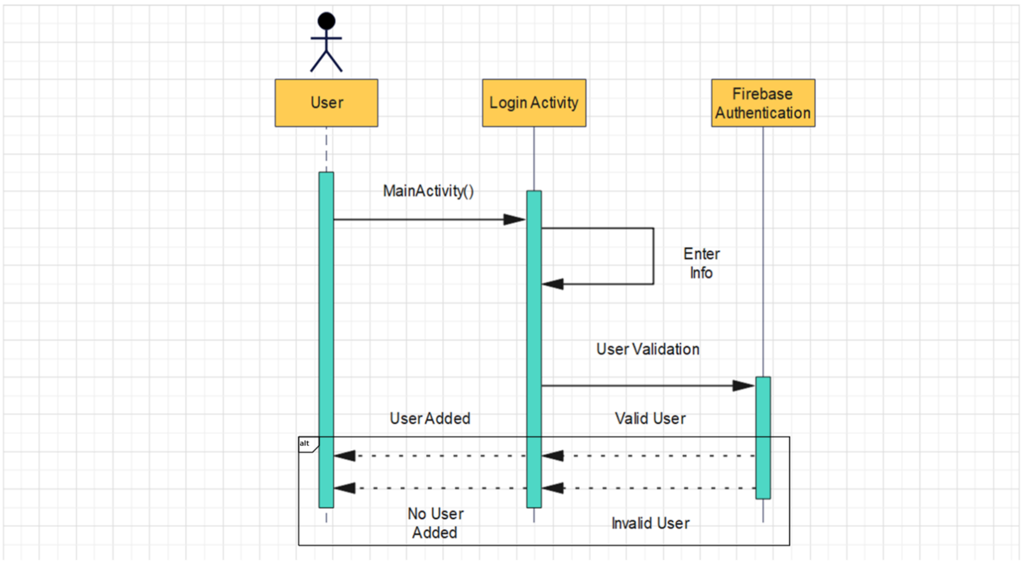


FIGURE 4.6: Sequence Diagram for Login

Figure 4.7 show the interaction between the User and the system when they want to file complaint against an issue. If the complaint is registered by the system, then it has been displayed to the User.

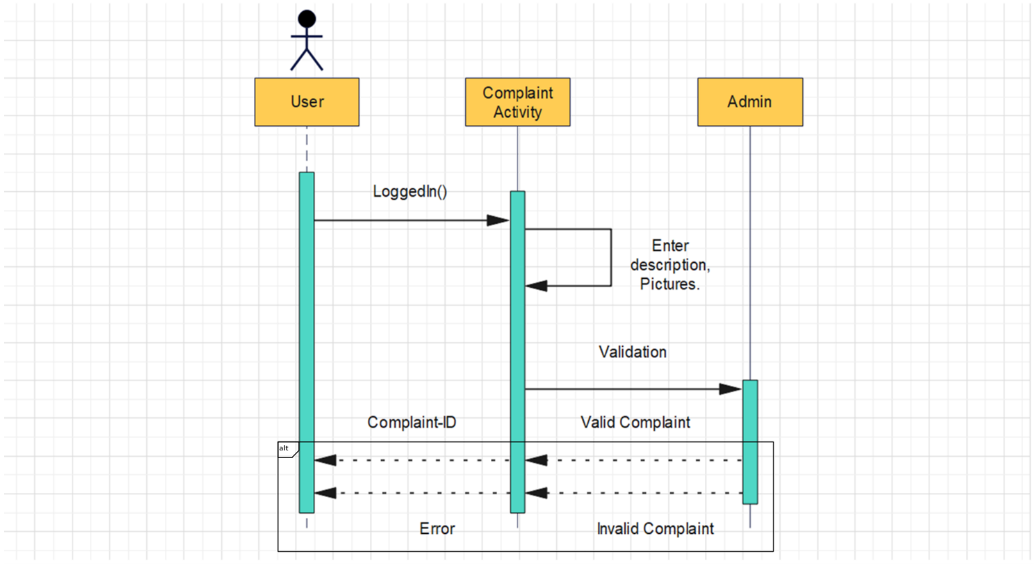


FIGURE 4.7: Sequence Diagram for Registering Complaint

Figure 4.8 show the interaction between the User and the system when they want to view the list of Team Members. If the server is not facing downtime, it displays the list of Team Members to the User (s).

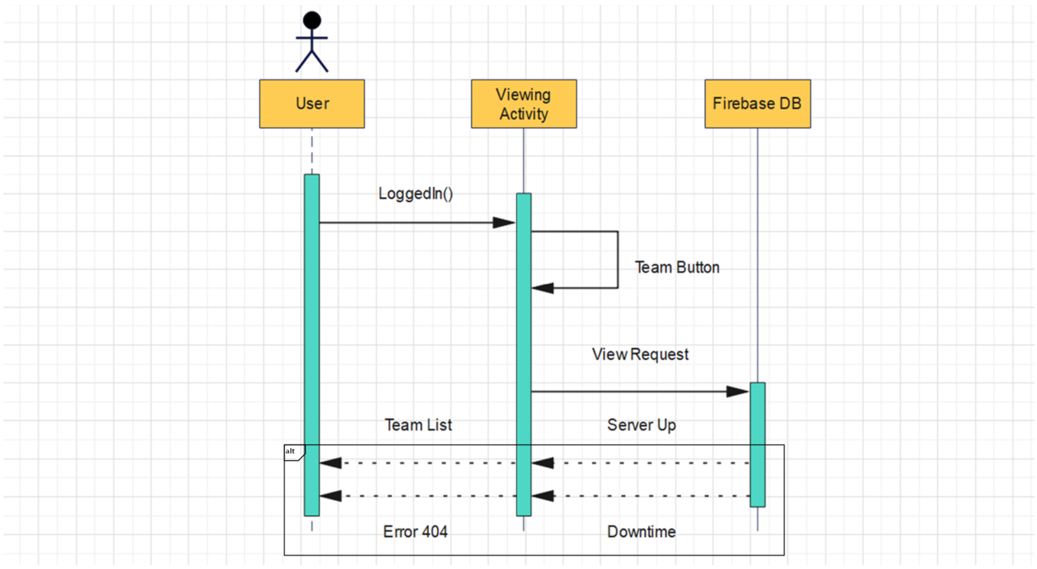


FIGURE 4.8: Sequence Diagram for Viewing Team Member List

Figure 4.9 show the interaction between the User and the system when they want to Track the Status (i.e., Confirmed, On the way, Complete) of their complaints. User can track their complaint status by entering Complaint-ID.

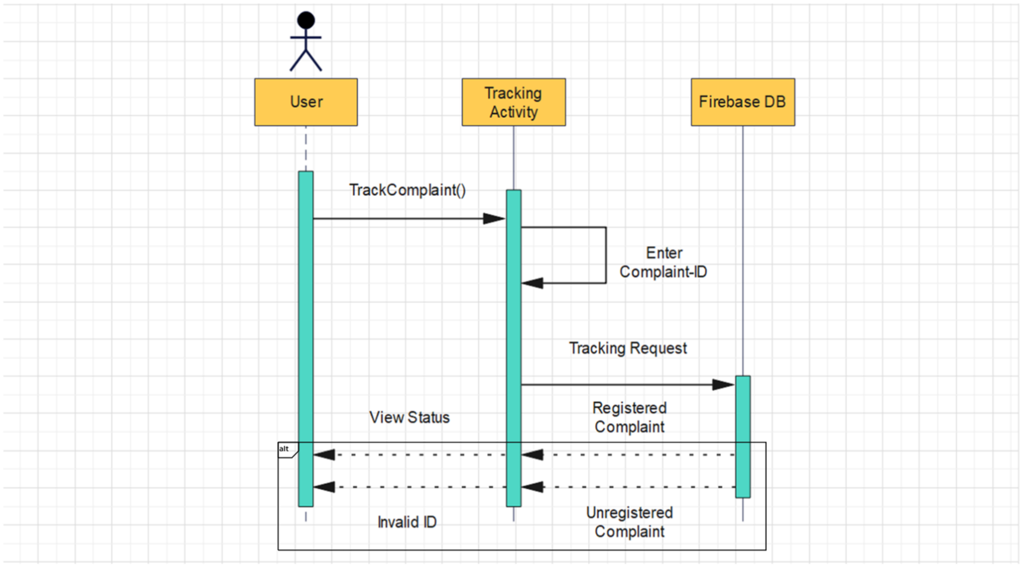


FIGURE 4.9: Sequence Diagram for Tracking Complaint Status

Figure 4.10 shows the interaction between the User and the system when they want to Update their profile. The user goes to the profile section, make changes, and click Save to update the information successfully.

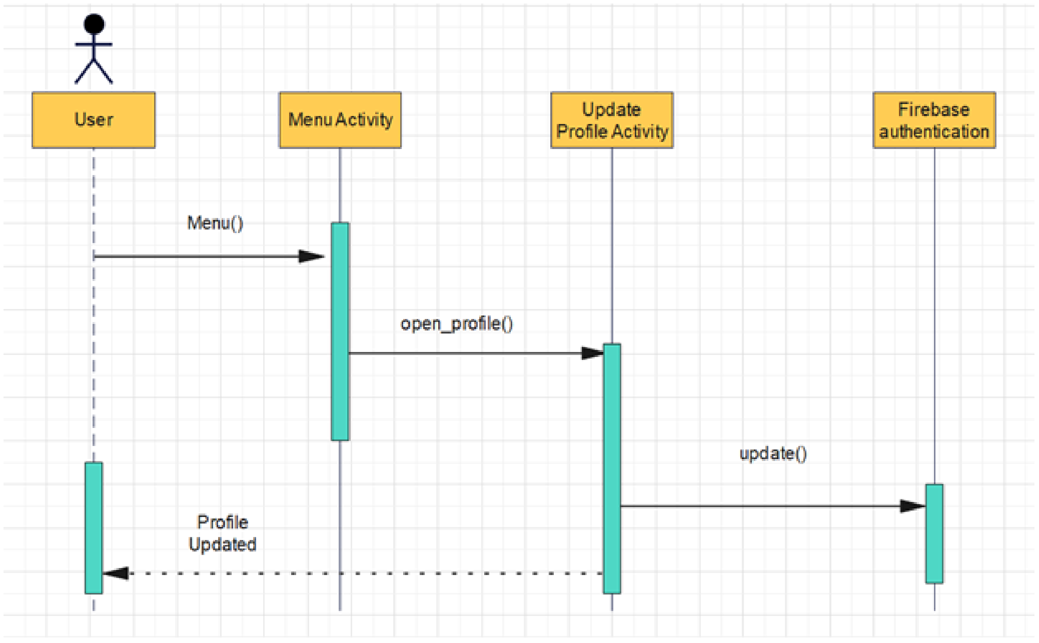


FIGURE 4.10: Sequence Diagram for Updating Profile

Figure 4.11 shows the interaction between the User and the system when they want to Delete their account. The user goes to the profile section, request for deletion, and click Yes to delete the Profile successfully.

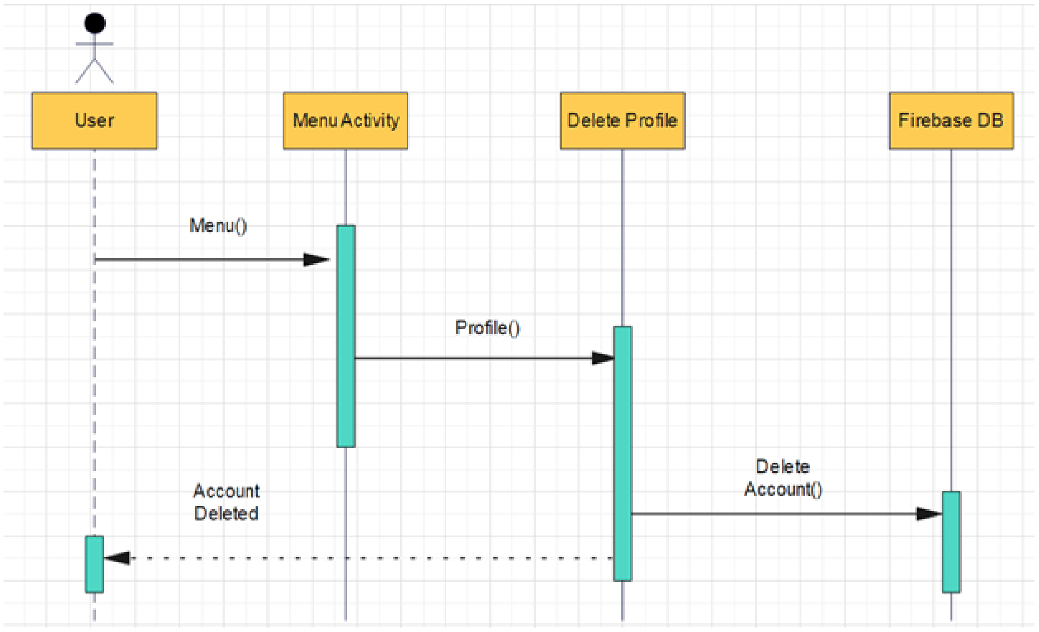


FIGURE 4.11: Sequence Diagram for Account Deletion

Figure 4.12 shows the interaction between the User and the system when they want to Give Feedback against their complaint after being fixed. The feedback has been stored in the Firebase DB and user gets a message like ‘Thanks for your feedback’.

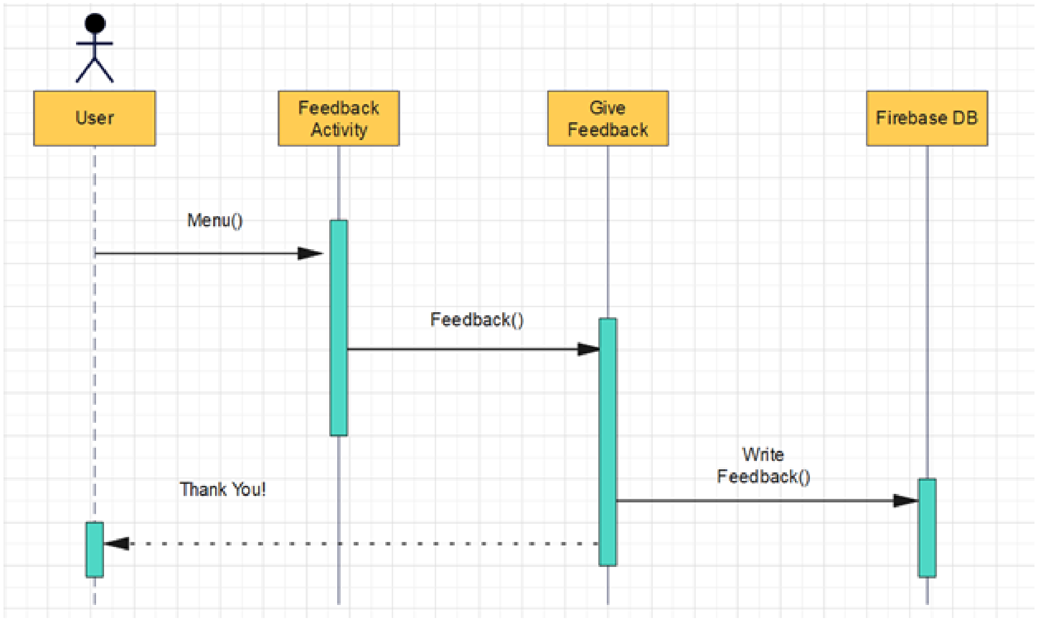


FIGURE 4.12: Sequence Diagram for Giving Feedback

Figure 4.13 shows the interaction between the User and the system when they want to Contact Admin. User clicks on Contact Us button on the Home screen. This gives him/her ways to contact the Admin.

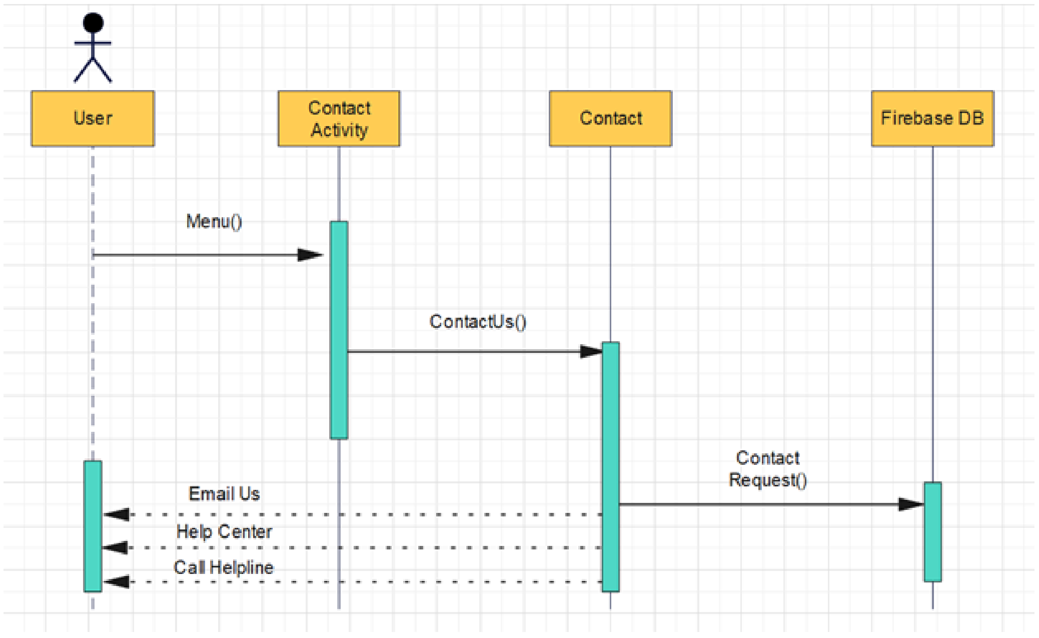


FIGURE 4.13: Sequence Diagram for Contacting Admin

Figure 4.14 shows the interaction between the User and the system when they want to recover the password which they have forgotten. User have to enter his/her email on which they can receive OTP via which they can update password.

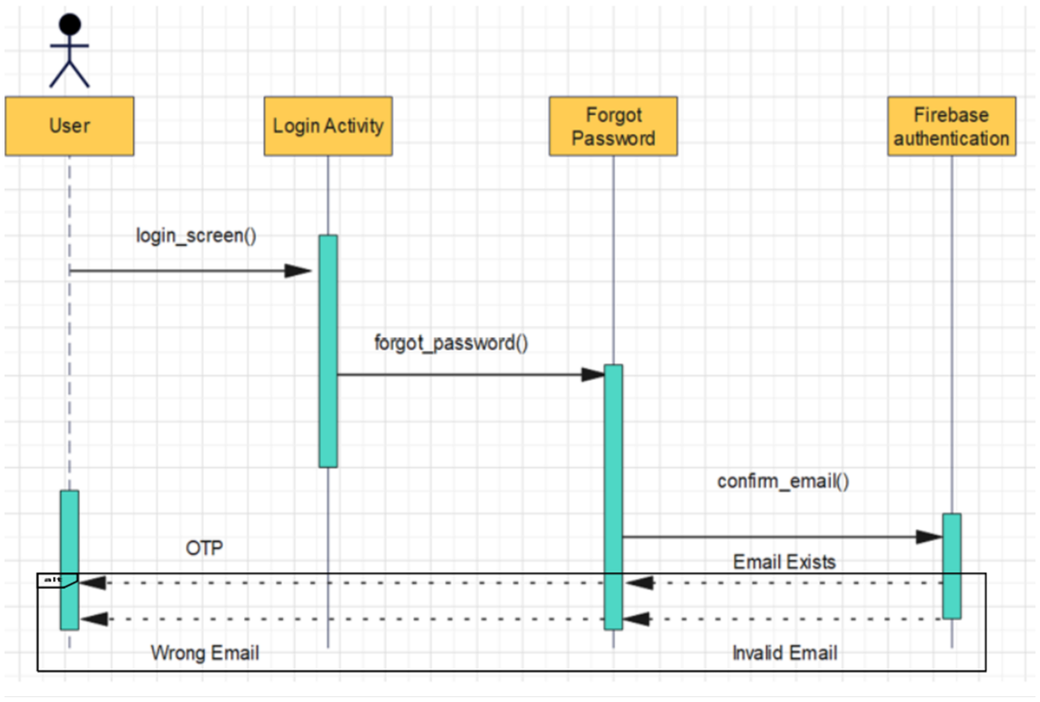


FIGURE 4.14: Sequence Diagram for Password Recovery

Figure 4.15 shows the interaction between the User and the system when they want to logout from the application.

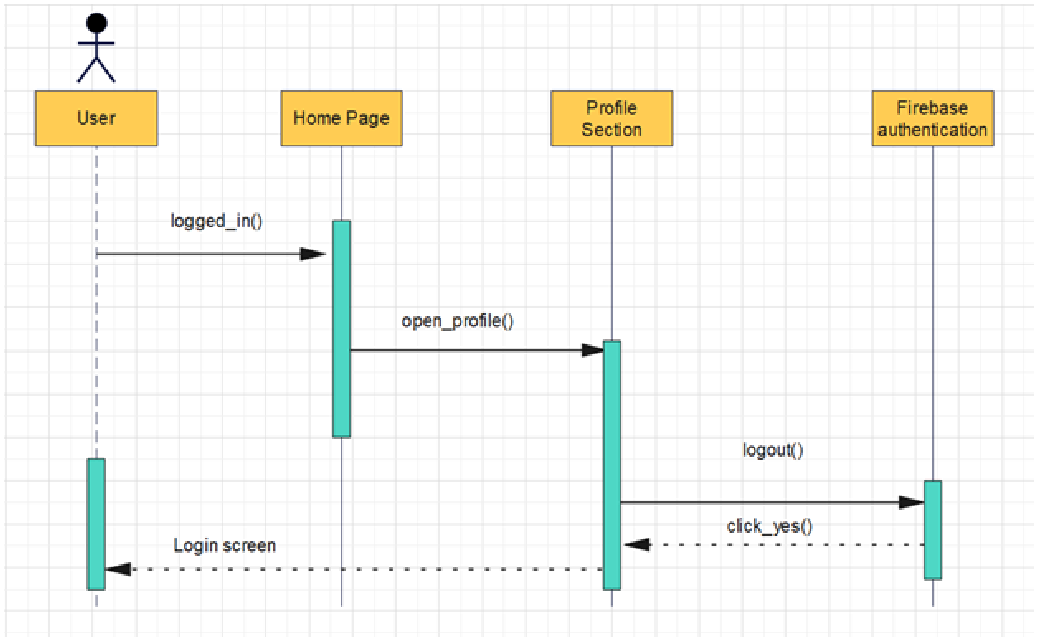


FIGURE 4.15: Sequence Diagram for Logout

4.5 Low Level Design

This section provides a low-level design our application Fixome. Low-level design descriptions that directly enable the creation of modules are provided in this section. Now, we dissect each component into its underlying pieces or modules.

4.5.1 Class Diagram

An UML class diagram is a visual portrayal that shows the classes, attributes, and methods for an application or framework. It helps in figuring out how various parts (classes) of the application are associated and collaborate with one another. With regards to our application, an UML class diagram would show the fundamental pieces of the application, as Admin, User, and Team, alongside their exceptional highlights (attributes) and activities (methods). It permits us to perceive how these parts are connected and the way that they cooperate to accomplish the general usefulness of the application.

Figure 4.16 visually explains what are the major functions that a User can perform using our application, from Filing a Complaint to Tracking Complaint.

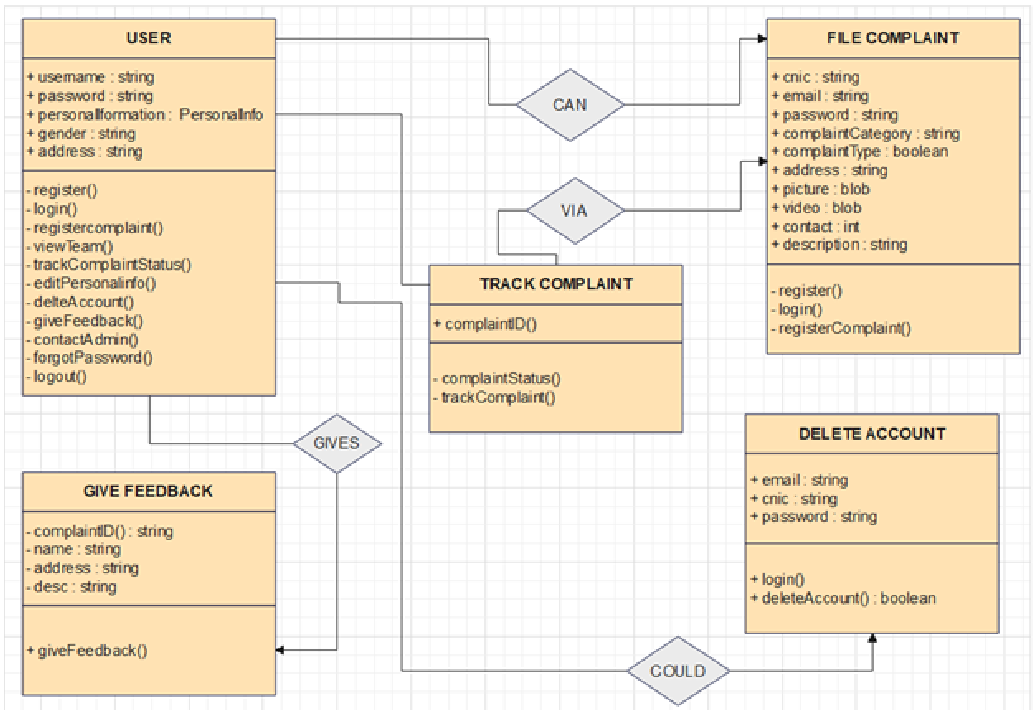


FIGURE 4.16: UML Class Diagram for User module

Figure 4.17 shows in a graphical manner, how the Admin of our application manages the complaints submitted by the User, and how the Team Worker submits the Complaint and reports to the Admin.

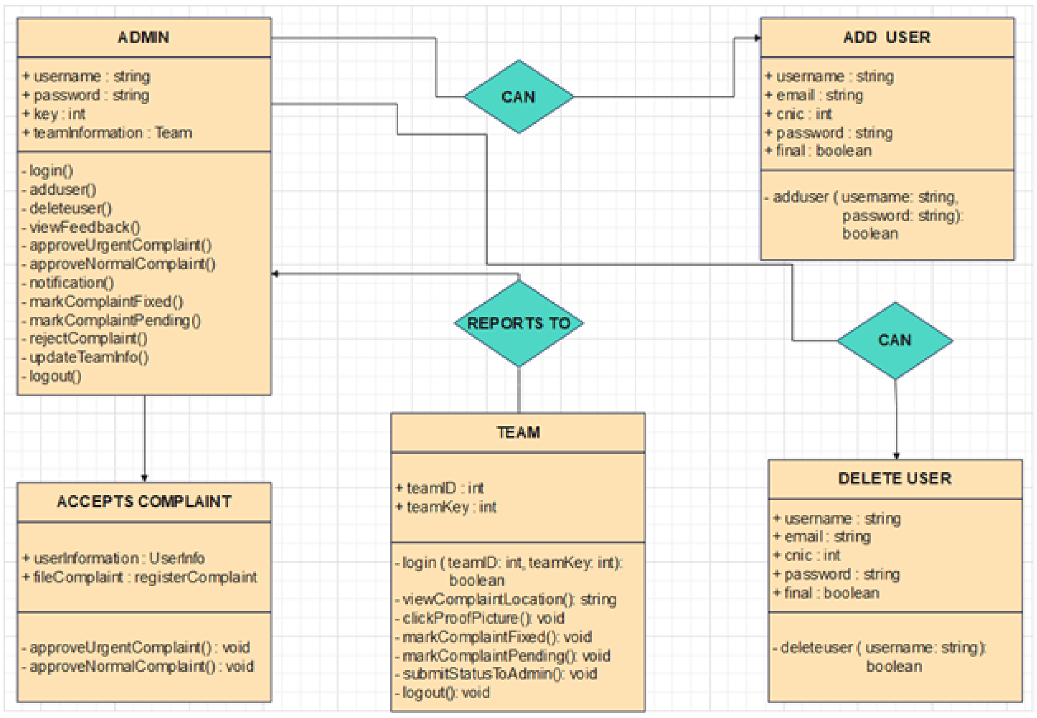


FIGURE 4.17: UML Class Diagram for Admin-Team module

4.6 GUI Design

We have designed an interface keeping in mind all the usability factors. Also, we tried to make it look pleasing and aesthetic. The purpose of creating such interface is that the user should feel good while using our application. As much importance is now given to the look and feel of interface now a days. In this section we are going to show our GUI (Graphical User Interface).

4.6.1 User Login

User can enter his/her Email and Password and then click Login to successfully see the application interface, as shown in figure 4.18.

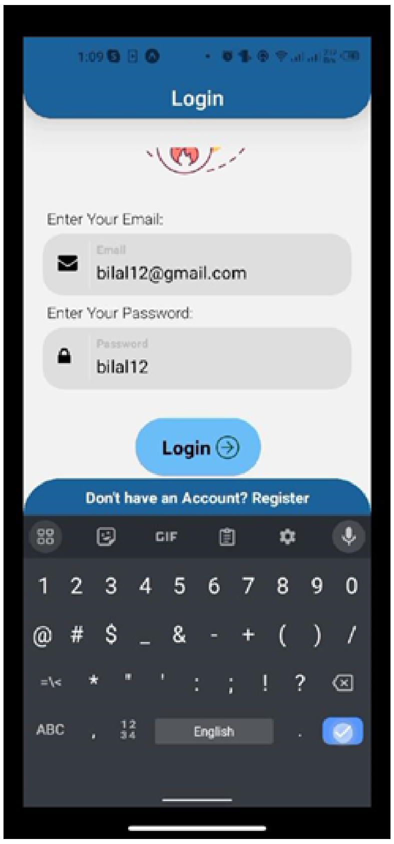
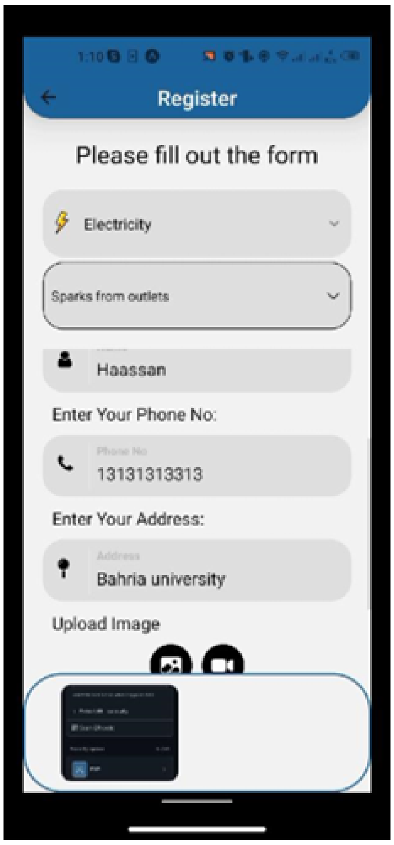


FIGURE 4.18: User Login Screen

4.6.2 User Complaint Registration

In the below figure 4.19, user can register the complaint by selecting the complaint type from the dropdown and then proceeding accordingly to the next steps. For demonstration purposes, we have chosen electricity complaint.



Electricity

Sparks from outlets

Haassan

Enter Your Phone No:

Phone No
13131313313

Enter Your Address:

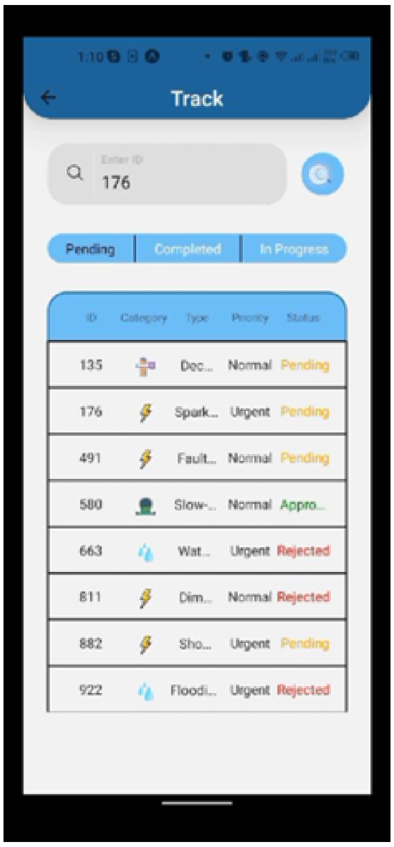
Address
Bahria university

Upload Image

FIGURE 4.19: User Complaint Registration

4.6.3 User Complaint Tracking

After registering the complaint, the user gets a Tracking ID through which he/she can track the complaint. For the above electricity complaint, our tracking ID is 176. Below figure 4.20 can show us how does the interface looks like with the complaint status.



176

ID	Category	Type	Priority	Status
135		Dec...	Normal	Pending
176	⚡	Spark...	Urgent	Pending
491	⚡	Fault...	Normal	Pending
580		Slow-...	Normal	Appro...
663	💧	Wat...	Urgent	Rejected
811	⚡	Dim...	Normal	Rejected
882	⚡	Sho...	Urgent	Pending
922	💧	Floodi...	Urgent	Rejected

FIGURE 4.20: User Complaint Tracking

4.6.4 User Feedback

Here the user can give the feedback against his/her fixed or un-fixed complaint. Figure 4.21 shows us the feedback given by a user.

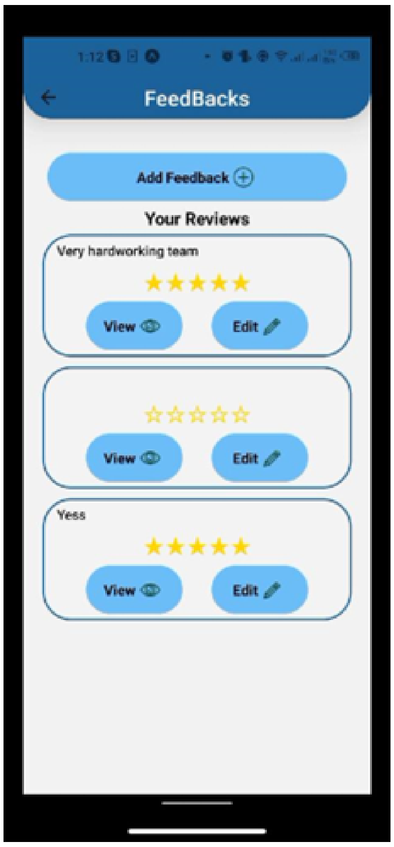


Figure 4.21: User Feedback

4.6.5 Admin Login Screen

Below figure 4.22 shows us the login screen of the admin where he/she can enter the Email and Password and hit the Login button.

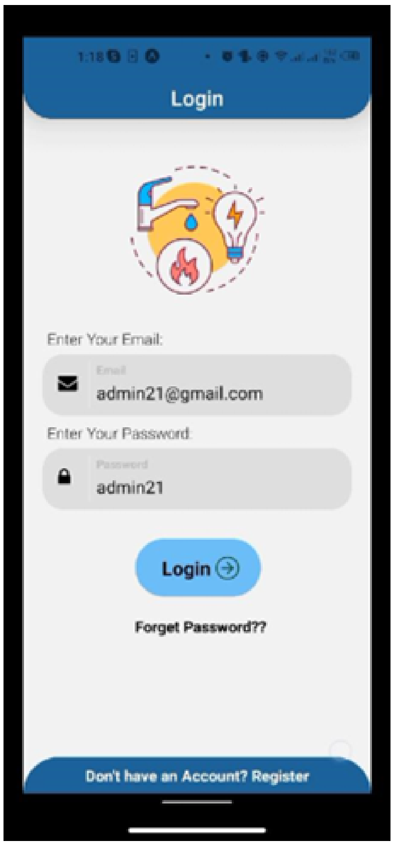


FIGURE 4.22: Admin Login Screen

4.6.6 Manage User's Complaints

Admin can Approve, Reject, or set the complaint as Pending, depending upon the queue. For our previous complaint number 176, admin can perform these three actions and then hit the Save button so that the complaint's updated status reaches the User. Below figure 4.23 shows us the respective screen.

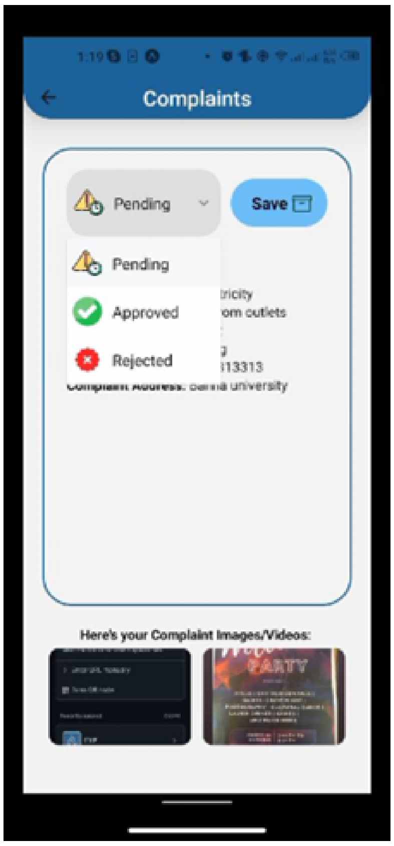
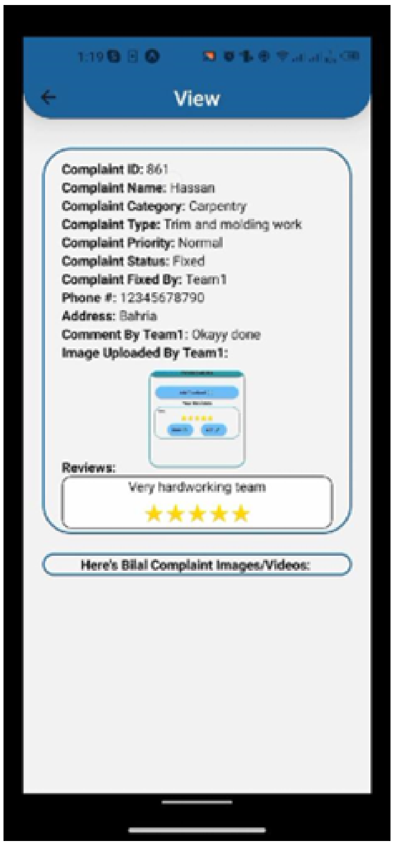


FIGURE 4.23: Manage User's Complaints

4.6.7 View Feedback

Admin can view the feedback given against the respective complaints by multiple users. An easy example snapshot of a complaint shows us the concept in below figure 4.24.



Complaint ID: 861
Complaint Name: Hassan
Complaint Category: Carpentry
Complaint Type: Trim and molding work
Complaint Priority: Normal
Complaint Status: Fixed
Complaint Fixed By: Team1
Phone #: 12345678790
Address: Bahria
Comment By Team1: Okayy done
Image Uploaded By Team1:

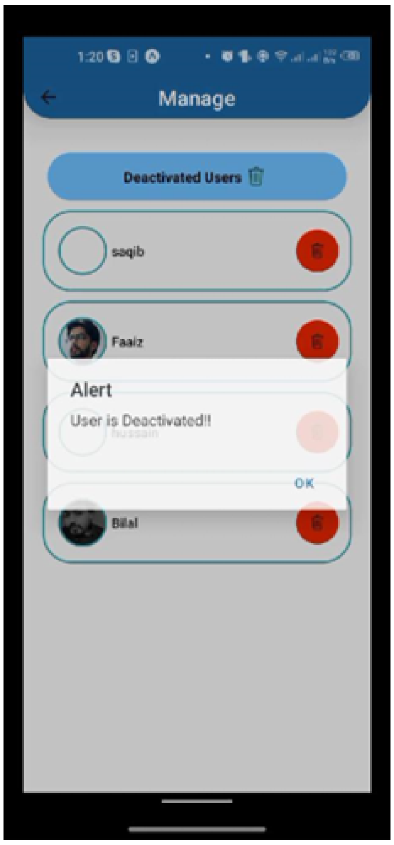
Reviews:
Very hardworking team
★★★★★

Here's Bilal Complaint Images/Videos:

FIGURE 4.24: View Feedback

4.6.8 Delete Users

Admin can delete the users from the database. Below figure 4.25, merged snapshot, shows that the admin deleted the first user named Saqib.



Alert

User is Deactivated!!

FIGURE 4.25: Delete User

4.6.9 Team Login

The team member/worker can login to his/her account using their Email and Password. After entering the correct credentials, and being verified from the database, the member has been successfully able to see the Home screen of the Fixome application, shown below in figure 4.26.

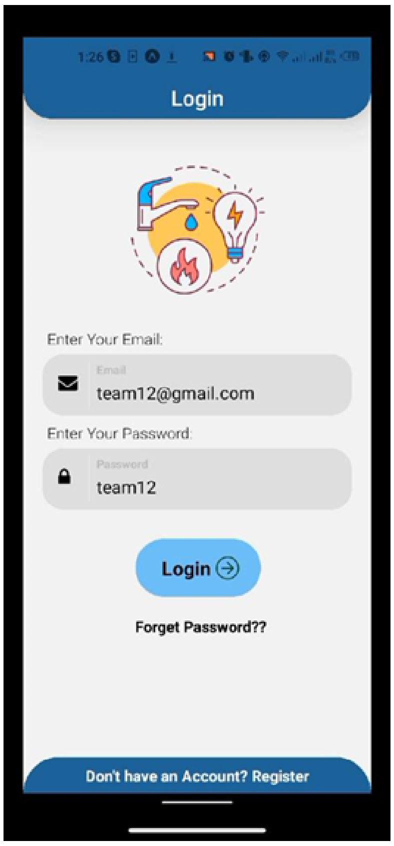


FIGURE 4.26: Team Login

4.6.10 Update Status

The worker can update the status of the complaint and hit the Save button after it has been registered by the User, and approved by the Admin. The updation must be done accordingly and keeping in mind the already to-be-fixed complaints. Figure 4.27 elaborates this concept for the above-mentioned complaint 176.

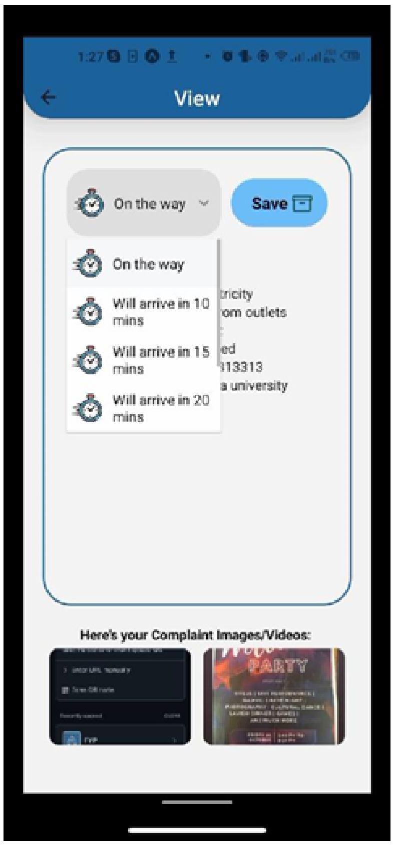


FIGURE 4.27: Update Status

Chapter 5

Implementation

5.1 System implentation

The process of turning an idea from a concept into reality is called implementation. A technical specification, algorithm, or other computer system is realised as a program, software component, or other computer system through programming and deployment.

5.2 System Architecture

Our system is based on 2-tier architecture. The front end of the application has been made by using React-Native and Firebase as the backend to store the data of the user. You can see the system architecture in Figure 5.1.

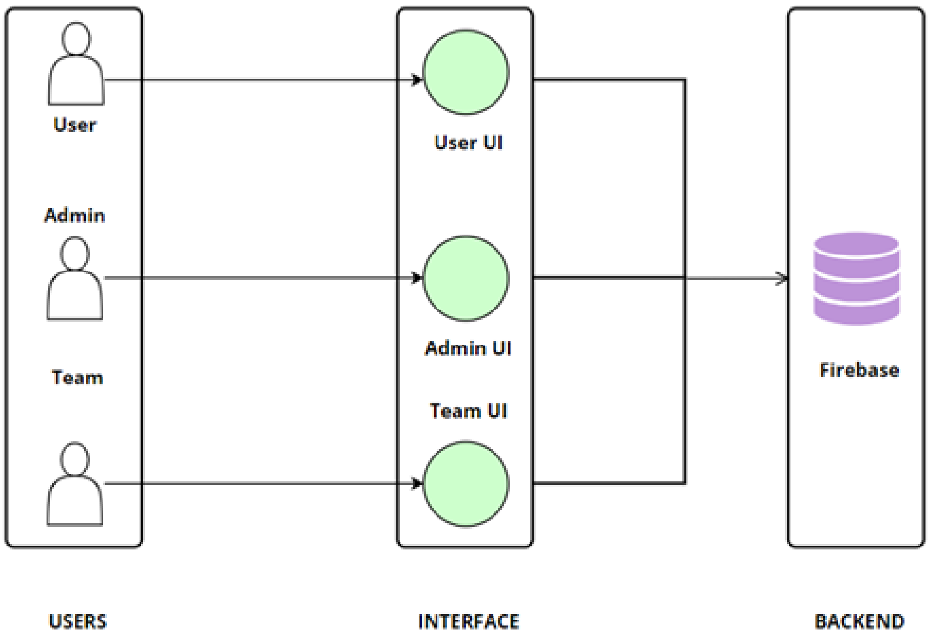


FIGURE 5.1: System Architecture

5.3 Tools and Technologies

Tools and Technologies which are used in our application are shown in table 5.1.

Tools	**Version**	**Rationale**
Latex	LaTeX2e	Documentation
MS PowerPoint	2021	Presentation
Visual Studio Code	1.80.1.0	IDE
Visual Paradigm	17.1	UML, Use cases, Sequence Diagrams
Wondershare EdrawMax	12.5.0	UML, Use cases, Sequence Diagrams
Technology	**Version**	**Rationale**
React-Native	0.72	Android and iOS App
Node.js	18.17.0x64	Backend Language
Expo Go	1017565	Sandbox
Firebase	12.7.0	Database

Table 5.1: Tools and Technologies

5.3.1 Explanation

- **Latex:** We have used the latest version of Latex i.e., v(LaTeX2e) for the documentation purpose.
- **MS PowerPoint:** We have used the Microsoft PowerPoint, version 2021, for creating presentations.
- **Visual Studio Code:** We have used Visual Studio Code, for debugging, code editing and source version control.
- **Visual Paradigm:** We have used Visual Paradigm for creating UML diagrams with the version v (17.1)
- **Wondershare EdrawMax:** We have used Wondershare EdrawMax for creating UML (Unified Modelling Language) diagrams with the version v (12.5.0)

- **React-Native:** We have used the latest technology named React-Native for our application because it is a cross- platform technology. The first stable version of React-Native has been version 0.5. React-Native provides custom, animated UIs of any complexity available.
- **Firebase:** We have used Firebase, version 12.7.0, as a database because it is a real-time database. Firebase provides swift and secured hosting and optimized app performances.
- **Node.js:** We have used Node.js as a backend language with React-Native.
- **Expo Go:** Open-source sandbox for building and testing apps being developed with React-Native.

5.4 Security

User Authentication

5.5 Conclusion

The system requires the study of Mobile Application Development in Android and React-Native. The mobile application interface and main features require a grip in React-Native language because all the implementation of interfaces are coded. The implementation of android and iOS applications requires ample knowledge of React-Native language because React-Native is a platform- independent tool, to some extent.

Chapter 6

System Testing

System testing of hardware and software is testing conducted on an integrated and complete system to evaluate the system's compliance with its specified requirements. Every software needs to be tested before deployment to ensure whether the software platform is as intended. It also ensures the reliability and usability of the system by unearthing the bugs. Testing ensures that maximum bugs are diagnosed in the system. To test our system, we followed the code, implement and test strategy, initially we code a module and perform a test on it to ensure that the module work as per intention. We then implement the module in the overall system and perform more strenuous tests so that its functionality is assured to be optional when used with the overall system. To carry out testing, we performed different types of tests depending on the nature of our application. We conduct following testing:

- Admin Module Testing
- User Module Testing
- Team Module Testing
- Compatibility Testing

- Exception Testing
- Load Testing
- Installation Testing

6.1 Admin Module Testing

In Admin Module Testing, each Admin Module Use Case is tested two times to check the success and failure of each core functionality in admin module of Fixome.

Extended test case 1 of the functionality for login performed by the admin shown in table 6.1.

Test Case ID	TC-1
Test Case Name	Login
Actor(s)	Admin
Pre-Conditions	Admin adds information to login
Priority	High
Flow/Inputs	Opens the mobile app Enters Email, Password and Key Click Login
Actor Actions	Admin inputs email address, password, and key
System Response	System checks admin's provided credentials. If valid, authenticates admin and displays home screen.
Expected Result	Login Screen
Actual Result	Login Screen
Failure Reason (if failed)	N/A

Table 6.1: Admin login

Extended test case 2 of the functionality for login performed by the admin shown in table 6.2.

Test Case ID	TC-2
Test Case Name	Login
Actor(s)	Admin
Pre-Conditions	Admin adds information to login
Priority	High
Basic Flow/Inputs	Opens the mobile app Enters Email, Password, and Key Clicks Login
Actor Actions	Admin inputs email address, password, and key
Expected System Response	System checks admin's provided credentials. If valid, authenticates admin and displays home screen.
Expected Result	Login Screen
Actual Result	Login Failed Error
Failure Reason (if failed)	Missing @ symbol in the Email entry

Table 6.2: Admin login

Extended test case 1 of the functionality for add new user performed by the admin shown in table 6.3.

Table 6.3: Admin Adding a New User

Test Case ID	TC-3
Test Case Name	Add New User
Actor(s)	Admin
Pre-Conditions	Admin is logged in as admin and at the Dashboard
Priority	Medium

Basic Flow/Inputs	1. Admin clicks on Users 2. List of Users displays 3. Admin clicks on Requests 4. Admin checks the necessary requirements for approval 5. If authentic, clicks Approve 6. If not authentic, clicks Discard and gives reason
Actor Actions	Admin clicks the Approve button after if the requirements are fulfilled and authentic Admin clicks the Discard button and gives reason if the requirements are not authentic or complete
System Response	A message pops up, saying 'New User Added' A message has been displayed, saying 'Request Denied'
Expected Result	User Added
Actual Result	User Added
Failure Reason (if failed)	N/A

Extended test case 2 of the functionality for add new user performed by the admin shown in table 6.4.

Test Case ID	TC-4
Test Case Name	Add New User
Actor(s)	Admin
Pre-Conditions	Admin is logged in as admin
Priority	Medium

Basic Flow/Inputs	1. Admin clicks on Users 2. List of Users displays 3. Admin clicks on Requests 4. Admin checks the necessary requirements for approval 5. If authentic, clicks Approve 6. If not authentic, clicks Discard and gives reason
Actor Actions	**Expected System Response**
Admin clicks the Approve button	A message pops up, saying 'New User Added'
Admin clicks the Discard button and gives reason	A message has been displayed, saying 'Request Denied'
Expected Result	User Added
Actual Result	User Not Added
Failure Reason (if failed)	Internet Connection Lost

Table 6.4: Admin : Add New User

Extended test case 1 of the functionality for Delete User performed by the actor admin shown in table 6.5.

Test Case ID	TC-5
Test Case Name	Delete User
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic Flow/Inputs	1. Admin is at the Dashboard 2. Admin clicks on Users 3. List of Users displays 4. Admin clicks on Delete Users 5. Admin selects the User(s) to delete from the database 6. Admin clicks on Delete Button 7. Selected Users has been deleted
Actor Actions	**System Response**
Admin deletes the user	System displays message "User Deleted Successfully"
Expected Result	User Deleted
Actual Result	User Deleted
Failure Reason (if failed)	N/A

Table 6.5: Admin: Delete User

Extended test case 2 of the functionality for Delete User performed by the actor admin shown in table 6.6.

Test Case ID	TC-6
Test Case Name	Delete User
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic Flow/Inputs	1. Admin is at the Dashboard 2. Admin clicks on Users 3. List of Users displays 4. Admin clicks on Delete Users 5. Admin selects the User(s) to delete from the database 6. Admin clicks on Delete Button 7. Selected Users has been deleted
Actor Actions	**System Response**
Admin deletes the user	System displays message "User Deleted Successfully"
Expected Result	User Deleted
Actual Result	User not Deleted
Failure Reason (if failed)	Connection lost

Table 6.6: Admin: Delete User

Extended test case 1 of the functionality for View Feedback performed by the admin shown in table 6.7.

Test Case ID	TC-7
Test Case Name	View Feedback
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is at the Dashboard Admin clicks on Feedback Admin sees feedback of user their Name and Date.
Actor Actions	**System Response**
Admin selects Feedback	The system displays all the Feedback of the Users successfully
Expected Result	Feedback displayed
Actual Result	Feedback displayed
Failure Reason (if failed)	N/A

Table 6.7: Admin: View Feedback

Extended test case 2 of the functionality for View Feedback performed by the admin shown in table 6.8.

Test Case ID	TC-8
Test Case Name	View Feedback
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is at the Dashboard Admin clicks on Feedback Admin sees feedback of user their Name and Date.
Actor Actions	**System Response**
Admin selects Feedback	The system displays all the Feedback of the Users successfully
Expected Result	Feedback displayed
Actual Result	Feedback did not display
Failure Reason (if failed)	User Deleted his/her feedback

Table 6.8: Admin: View Feedback

Extended test case 1 of the functionality for Urgent Complaints performed by the admin shown in table 6.9.

Test Case ID	TC-9
Test Case Name	Urgent Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is on dashboard Admin clicks on Complaints Admin selects Urgent from dropdown Admin sees the certain complaint details If belongs to Urgent, admin clicks Approve button
Actor Actions	**System Response**
Admin mark complaint Urgent	Displaying a message 'shifted to urgent'
Expected Result	Urgent Complaints displayed
Actual Result	Urgent Complaints displayed
Failure Reason (if failed)	N/A

Table 6.9: Admin: Urgent Complaints

Extended test case 2 of the functionality for Urgent Complaints performed by the admin shown in table 6.10.

Test Case ID	TC-10
Test Case Name	Urgent Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is on dashboard Admin clicks on Complaints Admin selects Urgent from dropdown Admin sees the certain complaint details If belongs to Urgent, admin clicks Approve button
Actor Actions	**System Response**
Admin mark complaint Urgent	Displaying a message 'shifted to urgent'
Expected Result	Urgent Complaints displayed
Actual Result	Urgent Complaints did not display
Failure Reason (if failed)	Complaint belonged to undefined category

Table 6.10: Admin: Urgent Complaints

Extended test case 1 of the functionality for Normal Complaints performed by the actor admin shown in table 6.11.

Test Case ID	TC-11
Test Case Name	Normal Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is on dashboard Admin clicks on Complaints Admin selects Normal from dropdown Admin sees the certain complaint details
Actor Actions	**System Response**
Admin clicks to see Normal basis complaints	System displays the Normal basis complaints
Expected Result	Normal Complaints displayed
Actual Result	Normal Complaints displayed
Failure Reason (if failed)	N/A

Table 6.11: Admin: Normal Complaints

Extended test case 2 of the functionality for Normal Complaints performed by the actor admin shown in table 6.12.

Test Case ID	TC-12
Test Case Name	Normal Complaints
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic Flow/Inputs	Admin is on dashboard Admin clicks on Complaints Admin selects Normal from dropdown Admin sees the certain complaint details
Actor Actions	**System Response**
Admin clicks to see the Normal basis complaints	System displays the Normal basis complaints
Expected Result	Normal Complaints displayed
Actual Result	Normal Complaints did not display
Failure Reason (if failed)	No complaint has been registered as normal

Table 6.12: Admin: Normal Complaints

Extended test case 1 of the functionality for Completion Notification performed by the actor admin shown in table 6.13.

Test Case ID	TC-13
Test Case Name	Completion Notification
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic flow /inputs	Admin is on dashboard Admin sees a number bubble on Notification icon Admin clicks to open it Admin sees the Complaint number, marked as Fixed or Pending Admin clicks Manage Admin clicks Fixed or Pending according to the message
Actor Actions	Admin receives a pop up on Notification icon.
System Response	System shows ‘Complaint is Fixed’
Expected Result	Status Changed
Actual Result	Status Changed
Failure Reason (if failed)	N/A

Table 6.13: Admin: Completion Notification

Extended test case 2 of the functionality for Completion Notification performed by the actor admin shown in table 6.14.

Test Case ID	TC-14
Test Case Name	Completion Notification
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	High
Basic flow /inputs	Admin is on dashboard Admin sees a number bubble on Notification icon Admin clicks to open it Admin sees the Complaint number, marked as Fixed or Pending Admin clicks Manage Admin clicks Fixed or Pending according to the message
Actor Actions	Admin receives a pop up on Notification icon.
System Response	System shows 'Complaint is Fixed'
Expected Result	Status Changed
Actual Result	Status did not Changed
Failure Reason (if failed)	N/A

Table 6.14: Admin: Completion Notification

Extended test case 1 of the functionality for Complaints Fulfilled performed by the actor admin shown in table 6.15.

Test Case ID	TC-15
Test Case Name	Complaints Fulfilled
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	1. Admin clicks on View 2. Admin clicks on Complaints Fulfilled 3. Admin sees all the complaints marked as fixed in the list
Actor Actions	Admin clicks the View button > Complaints > Pending
System Response	shows all the complaints which have been fixed along with date
Expected Result	Fixed Complaints List shown
Actual Result	Fixed Complaints List shown
Failure Reason (if failed)	N/A

Table 6.15: Admin: Complaints Fulfilled

Extended test case 2 of the functionality for Complaints Fulfilled performed by the actor admin shown in table 6.16.

Test Case ID	TC-16
Test Case Name	Complaints Fulfilled
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	1. Admin clicks on View 2. Admin clicks on Complaints Fulfilled 3. Admin sees all the complaints marked as fixed in the list
Actor Actions	Admin clicks the View button > Complaints > Pending
System Response	shows all the complaints which have been fixed along with date
Expected Result	Fixed Complaints List shown
Actual Result	No 'Fixed Complaints List' shown
Failure Reason (if failed)	Team did not report any Fixed Complaint yet

Table 6.16: Admin: Complaints Fulfilled

Extended test case 1 of the functionality for Complaints Pending performed by the actor admin shown in table 6.17.

Test Case ID	TC-17
Test Case Name	Complaints Pending
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	Admin is at dashboard Admin clicks on View Admin clicks Complaint Pending
Actor Actions	Admin clicks the View button > Complaints Pending
System Response	System shows all the complaints which are pending along with date
Expected Result	Pending Complaints List
Actual Result	Pending Complaints List shown
Failure Reason (if failed)	N/A

TABLE 6.17: Admin: Complaints pending

Extended test case 2 of the functionality for Complaints Pending performed by the actor admin shown in table 6.18.

Test Case ID	TC-18
Test Case Name	Complaints Pending
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	Admin is at dashboard Admin clicks on View Admin clicks Complaint Pending
Actor Actions	Admin clicks the View button > Complaints Pending
System Response	System shows all the complaints which are pending along with date
Expected Result	Pending Complaints List
Actual Result	No Pending Complaints List shown
Failure Reason (if failed)	No new complaint has been registered by the user

Table 6.18: Admin: Complaints pending

Extended test case 1 of the functionality for Complaints Rejected performed by the actor admin shown in table 6.19.

Test Case ID	TC-19
Test Case Name	Complaints Rejected
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	Admin is at dashboard Admin clicks on View Admin clicks Complaint Rejected
Actor Actions	Admin clicks the View button > Complaints
System Response	System shows all the complaints which are Rejected along with date
Expected Result	Rejected Complaints List shown
Actual Result	Rejected Complaints list shown
Failure Reason (if failed)	N/A

Table 6.19: Admin: Complaints Rejected

Extended test case 2 of the functionality for Complaints Rejected performed by the actor admin shown in table 6.20.

Test Case ID	TC-20
Test Case Name	Complaints Rejected
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	Medium
Basic flow /inputs	Admin is at dashboard Admin clicks on View Admin clicks Complaint Rejected
Actor Actions	Admin clicks the View button > Complaints Rejected
System Response	System shows all the complaints which are Rejected along with date
Expected Result	Rejected Complaints List shown
Actual Result	No 'Rejected Complaints list' shown
Failure Reason (if failed)	No faulty complaint has been made

Table 6.20: Admin: Complaints Rejected

Extended test case 1 of the functionality for Update Team Info performed by the actor admin shown in table 6.21.

Test Case ID	TC-21
Test Case Name	Update Team Info
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	low
Basic flow /inputs	Admin is at dashboard Admin clicks on Team button Admin clicks Edit on top right corner
Actor Actions	Admin edits the member info
System Response	System shows, 'Updated Successfully'
Expected Result	Updated Successfully
Actual Result	Updated Successfully
Failure Reason (if failed)	N/A

Table 6.21: Admin: Update Team Info

Extended test case 2 of the functionality for Update Team Info performed by the actor admin shown in table 6.22.

Test Case ID	TC-22
Test Case Name	Update Team Info
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin
Priority	low
Basic flow /inputs	Admin is at dashboard Admin clicks on Team button Admin clicks Edit on top right corner
Actor Actions	Admin edits the member info
System Response	System shows, 'Updated Successfully'
Expected Result	Updated Successfully
Actual Result	Did not Updated Successfully
Failure Reason (if failed)	Username has special characters

Table 6.22: Admin: Update Team Info

Extended test case 1 of the functionality for Logout performed by the actor admin shown in table 6.23.

Test Case ID	TC-23
Test Case Name	Logout
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin .The admin no longer wants to be logged in
Priority	low
Basic flow /inputs	The admin is done using the mobile application The admin clicks on the profile icon The admin clicks on logout option at the end
Actor Actions	Admin clicks on the log out button
System Response	System successfully displayed the log in screen
Expected Result	Login Screen
Actual Result	Login Screen
Failure Reason (if failed)	N/A

Table 6.23: Admin: Logout

Extended test case 2 of the functionality for Logout performed by the actor admin shown in table 6.24.

Test Case ID	TC-24
Test Case Name	Logout
Actor(s)	Admin
Pre-Conditions	Admin logged in as admin .The admin no longer wants to be logged in
Priority	low
Basic flow /inputs	The admin is done using the mobile application The admin clicks on the profile icon The admin clicks on logout option at the end
Actor Actions	Admin clicks on the log out button
System Response	System successfully displayed the log in screen
Expected Result	Login Screen
Actual Result	Logout Loading Screen
Failure Reason (if failed)	Internet Connection has been disturbed

Table 6.24: Admin: Logout

6.2 User Module Testing

Extended test case 1 of the functionality for Registration performed by the User shown in table 6.25.

Test Case ID	TC-25
Test Case Name	Registration
Actor(s)	User
Pre-Conditions	User adds personal information for registration
Priority	High
Basic flow /inputs	User opens application User can choose to register as a User The user enters Email and Password The user enters Name and CNIC The user clicks Register button The system verifies entered email and CNIC and registers the user System displays a msg “Your account has been registered successfully”
Actor Actions	User enters personal details. clicks on the “Register” button
System Response	User’s provide credentials are valid dashboard is displayed
Expected Result	Home Screen
Actual Result	Home Screen
Failure Reason (if failed)	N/A

Table 6.25: Registration

Test Case ID	TC-26
Test Case Name	Registration
Actor(s)	User
Pre-Conditions	User adds personal information for registration
Priority	High
Basic flow /inputs	User opens application User can choose to register as a User The user enters Email and Password The user enters Name and CNIC The user clicks Register button The system verifies entered email and CNIC and registers the user System displays a msg “Your account has been registered successfully”
Actor Actions	User enters personal details. clicks on the “Register” button
System Response	User’s provide credentials are valid dashboard is displayed
Expected Result	Home Screen
Actual Result	Registration Failed
Failure Reason (if failed)	User entered – sign in CNIC Entry

Table 6.26: Registration

Extended test case 1 of the functionality for Login performed by the User shown in table 6.27

Test Case ID	TC-27
Test Case Name	Login
Actor(s)	User
Pre-Conditions	User adds personal information to login
Priority	High
Basic flow /inputs	User opens mobile application Enters Email/CNIC and password Click Login. Logged into mobile application
Actor Actions	User inputs Email/CNIC and Password
System Response	User's provide credentials are valid dashboard is displayed
Expected Result	Home Screen
Actual Result	Home Screen
Failure Reason (if failed)	N/A

Table 6.27: Registration

Extended test case 2 of the functionality for Login performed by the User shown in table 6.28

Test Case ID	TC-28
Test Case Name	Login
Actor(s)	User
Pre-Conditions	User adds personal information to login
Priority	High
Basic flow /inputs	User opens mobile application Enters Email/CNIC and password Click Login. Logged into mobile application
Actor Actions	User inputs Email/CNIC and Password
System Response	User's provide credentials are valid dashboard is displayed
Expected Result	Home Screen
Actual Result	Login Failed
Failure Reason (if failed)	Wrong Password

Table 6.28: Registration

Extended test case 1 of the functionality for Complaint Registration performed by the User shown in table 6.29

Test Case ID	TC-29
Test Case Name	Complaint Registration
Actor(s)	User
Pre-Conditions	Actor logged User Must see the list of Urgent complaints
Priority	High
Basic flow /inputs	User is on dashboard User clicks Register Complaint button User selects the Category i.e., Water, Gas etc. User Clicks pictures or video and click on upload button Selected media is uploaded Users write a brief description explaining the issue clearly User Enters Name, Address, Nearby place (optional), No
Actor Actions	User files or registers a complaint
System Response	System displays a msg and ID
Expected Result	Tracking ID
Actual Result	Tracking ID
Failure Reason (if failed)	N/A

Table 6.29: Complaint Registration

Extended test case 2 of the functionality for Complaint Registration performed by the User shown in table 6.30

Test Case ID	TC-30
Test Case Name	Complaint Registration
Actor(s)	User
Pre-Conditions	Actor logged User Must see the list of Urgent complaints
Priority	High
Basic flow /inputs	User is on dashboard User clicks Register Complaint button User selects the Category i.e., Water, Gas etc. User Clicks pictures or video and click on upload button Selected media is uploaded Users write a brief description explaining the issue clearly User Enters Name, Address, Nearby place (optional), No
Actor Actions	User files or registers a complaint
System Response	System displays a msg and ID
Expected Result	Tracking ID
Actual Result	Complaint Registration Failed
Failure Reason (if failed)	User entered only 1 complaint picture

Table 6.30: Complaint Registration

Extended test case 1 of the functionality for Track Complaint performed by the User shown in table 6.31

Test Case ID	TC-31
Test Case Name	Track Complaint
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow /inputs	User is on the dashboard Complaint has been registered User has the Complaint-ID User clicks on Complaints User sees track complaints option User enters the Complaint-ID
Actor Actions	User tracks complaint
System Response	App loads for 3-4 seconds "Status: [any of above three statuses]”
Expected Result	Complaint Status
Actual Result	Complaint Status
Failure Reason (if failed)	N/A

Table 6.31: Track Complaint

Extended test case 2 of the functionality for Track Complaint performed by the User shown in table 6.32

Test Case ID	TC-32
Test Case Name	Track Complaint
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow /inputs	User is on the dashboard Complaint has been registered User has the Complaint-ID User clicks on Complaints User sees track complaints option User enters the Complaint-ID
Actor Actions	User tracks complaint
System Response	App loads for 3-4 seconds "Status: [any of above three statuses]"
Expected Result	Complaint Status
Actual Result	No Complaint Shown
Failure Reason (if failed)	User Entered Wrong Tracking ID

Table 6.32: Track Complaint

Extended test case 1 of the functionality for View Team performed by the User shown in table 6.33

Test Case ID	TC-33
Test Case Name	View Team
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard or home screen User clicks on Team icon User has been able to see the category of the complaints User selects any one category Member can see the Login page successfully
Actor Actions	User views Team
System Response	System displays experts with total no of team members category
Expected Result	Team Info
Actual Result	Team Info
Failure Reason (if failed)	N/A

Table 6.33: View Team

Extended test case 2 of the functionality for View Team performed by the User shown in table 6.34

Test Case ID	TC-34
Test Case Name	View Team
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard or home screen User clicks on Team icon User has been able to see the category of the complaints User selects any one category Member can see the Login page successfully
Actor Actions	User views Team
System Response	System displays experts with total no of team members category
Expected Result	Team Info
Actual Result	Connection Timed Out Error
Failure Reason (if failed)	Internet Connection Disturbed

Table 6.34: View Team

Extended test case 1 of the functionality for Edit personal Info performed by the User shown in table 6.35

Test Case ID	TC-35
Test Case Name	Edit Personal Info
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on the dashboard User clicks on Profile icon at bottom right corner User clicks Personal Information User clicks Edit User see the last updated personal info (Name, CNIC, Email etc.) User can make changes in his/her personal info App asks for the password to make changes User enters the correct password App verifies the password Password is correct User clicks Save button App loads for 2-3 seconds User sees a message, "Information Updated Successfully"
Actor Actions	App loads for 2-3 seconds User make changes in personal info
System Response	"Information Updated Successfully"
Expected Result	Updation Successful
Actual Result	Updation Successful
Failure Reason (if failed)	N/A

Table 6.35: Edit Personal Info

Extended test case 2 of the functionality for Edit personal Info performed by the User shown in table 6.36

Test Case ID	TC-36
Test Case Name	Edit Personal Info
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on the dashboard User clicks on Profile icon at bottom right corner User clicks Personal Information User clicks Edit User see the last updated personal info (Name, CNIC, Email etc.) User can make changes in his/her personal info App asks for the password to make changes User enters the correct password App verifies the password Password is correct User clicks Save button App loads for 2-3 seconds User sees a message, "Information Updated Successfully"
Actor Actions	App loads for 2-3 seconds User make changes in personal info
System Response	"Information Updated Successfully"
Expected Result	Updation Successful
Actual Result	Changes Failed
Failure Reason (if failed)	CNIC cannot be updated

Table 6.36: Edit Personal Info

Extended test case 1 of the functionality for Delete Account performed by the User shown in table 6.37

Test Case ID	TC-37
Test Case Name	Delete Account
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard User clicks Personal Info User clicks Account information User clicks Delete Account User sees app demanding for Email, CNIC and Password User enters the correct details User clicks Delete button App asks for confirmation User clicks Yes App loads for 3-5 seconds User gets a message “Your account has been deleted successfully” App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User deletes his/her account
System Response	Your account has been deleted successfully”
Expected Result	Login Screen / Deletion Successful
Actual Result N/A	Login Screen / Deletion Successful heightFailure Reason (if failed)

Table 6.37: Delete Account

Extended test case 2 of the functionality for Delete Account performed by the User shown in table 6.38

Test Case ID	TC-38
Test Case Name	Delete Account
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard User clicks Personal Info User clicks Account information User clicks Delete Account User sees app demanding for Email, CNIC and Password User enters the correct details User clicks Delete button App asks for confirmation User clicks Yes App loads for 3-5 seconds User gets a message "Your account has been deleted successfully" App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User deletes his/her account
System Response	Your account has been deleted successfully"
Expected Result	Login Screen / Deletion Successful
Actual Result	Deletion Failed
Failure Reason (if failed)	Wrong password

Table 6.38: Delete Account

Extended test case 1 of the functionality for Forgot Password performed by the User shown in table 6.39

Test Case ID	TC-39
Test Case Name	Forgot Password
Actor(s)	User
Pre-Conditions	The app is open, and user is on Login page
Priority	Medium
Basic flow /inputs	Enters wrong password Click login Login gets failed The user clicks on forgot password Enter email address and then click on the Go button User receives the mail through which they can recover their password App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User enters Email
System Response	An error message has been displayed that this account does not exist
Expected Result	User Receives Link via Email
Actual Result	User Receives Link via Email
Failure Reason (if failed)	N/A

Table 6.39: Forgot Password

Extended test case 2 of the functionality for Forgot Password performed by the User shown in table 6.40

Test Case ID	TC-40
Test Case Name	Forgot Password
Actor(s)	User
Pre-Conditions	The app is open, and user is on Login page
Priority	Medium
Basic flow /inputs	Enters wrong password Click login Login gets failed The user clicks on forgot password Enter email address and then click on the Go button User receives the mail through which they can recover their password App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User enters Email
System Response	An error message has been displayed that this account does not exist
Expected Result	User Receives Link via Email
Actual Result	User does not Receives any Link via Email
Failure Reason (if failed)	Account does not exist / Wrong Email

Table 6.40: Forgot Password

Extended test case 1 of the functionality for Give Feedback performed by the User shown in table 6.41

Test Case ID	TC-41
Test Case Name	Give Feedback
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow /inputs	The user is on dashboard The user selects give feedback User gives his/her thoughts after the complaint has been fixed User clicks Submit button System says, "Feedback sent successfully!"
Actor Actions	selects Feedback and enters feedback
System Response	Feedback sent successfully
Expected Result	Feedback sent successfully
Actual Result	Feedback send successfully
Failure Reason (if failed)	N/A

Table 6.41: Give Feedback

Extended test case 2 of the functionality for Give Feedback performed by the User shown in table 6.42

Test Case ID	TC-42
Test Case Name	Give Feedback
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Medium
Basic flow /inputs	The user is on dashboard The user selects give feedback User gives his/her thoughts after the complaint has been fixed User clicks Submit button System says, "Feedback sent successfully!"
Actor Actions	selects Feedback and enters feedback
System Response	Feedback sent successfully
Expected Result	Feedback sent successfully
Actual Result	Feedback did not send successfully
Failure Reason (if failed)	User cannot submit empty feedback

Table 6.42: Give Feedback

Extended test case 1 of the functionality for Contact Us performed by the User shown in table 6.43

Test Case ID	TC-43
Test Case Name	Contact Us
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard User click Contact icon User sees three options i.e., Help center, Email, Call Helpline User selects one of them and proceed accordingly Member performs confirmation i.e., clicks Yes App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User contacts for the help
System Response	to contact or seek help via three options i.e., Help Center, Email, Call
Expected Result	Admin Responded Back
Actual Result	Admin Respond Back
Failure Reason (if failed)	N/A

Table 6.43: Contact Us

Extended test case 2 of the functionality for Contact Us performed by the User shown in table 6.44

Test Case ID	TC-44
Test Case Name	Contact Us
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User is on dashboard User click Contact icon User sees three options i.e., Help center, Email, Call Helpline User selects one of them and proceed accordingly Member performs confirmation i.e., clicks Yes App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User contacts for the help
System Response	to contact or seek help via three options i.e., Help Center, Email, Call
Expected Result	Admin Responded Back
Actual Result	Admin Could Not Respond Back
Failure Reason (if failed)	Admin has been busy with other users / teams

Table 6.44: Contact Us

Extended test case 1 of the functionality for Logout performed by the User shown in table 6.45

Test Case ID	TC-45
Test Case Name	Logout
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User clicks Profile icon At the end, user sees logout button User clicks the button, and log out after 2-3 seconds App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User clicks on the logout button
System Response	System successfully displayed the login screen
Expected Result	Login Screen
Actual Result	Login Screen
Failure Reason (if failed)	N/A

Table 6.45: Contact Us

Extended test case 2 of the functionality for Logout performed by the User shown in table 6.46

Test Case ID	TC-46
Test Case Name	Logout
Actor(s)	User
Pre-Conditions	User is logged into app
Priority	Low
Basic flow /inputs	User clicks Profile icon At the end, user sees logout button User clicks the button, and log out after 2-3 seconds App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	User clicks on the logout button
System Response	System successfully displayed the login screen
Expected Result	Login Screen
Actual Result	Logout Screen
Failure Reason (if failed)	Slow Internet Connection

Table 6.46: Contact Us

6.3 Team Module Testing

Extended test case 1 of the functionality for Login performed by the Team shown in table 6.47

Test Case ID	TC-47
Test Case Name	Login
Actor(s)	Team
Pre-Conditions	Member adds personal information to login
Priority	High
Basic flow /inputs	Team member selects the Complaint Cell Team member enters the Team Code Clicks Login Member can now see the dashboard
Actor Actions	Member login with correct credentials
System Response	Member can now see the dashboard screen of the app
Expected Result	Home Screen
Actual Result	Home Screen
Failure Reason (if failed)	N/A

Table 6.47: Login

Extended test case 2 of the functionality for Login performed by the Team shown in table 6.48

Test Case ID	TC-48
Test Case Name	Login
Actor(s)	Team
Pre-Conditions	Member adds personal information to login
Priority	High
Basic flow /inputs	Team member selects the Complaint Cell Team member enters the Team Code Clicks Login Member can now see the dashboard
Actor Actions	Member login with correct credentials
System Response	Member can now see the dashboard screen of the app
Expected Result	Home Screen
Actual Result	Login Screen
Failure Reason (if failed)	Team Member entered wrong digit(s) of unique key

Table 6.48: Login

Extended test case 1 of the functionality for Complaint Details performed by the Team shown in table 6.49

Test Case ID	TC-49
Test Case Name	Complaint Details
Actor(s)	Team
Pre-Conditions	Member is logged in
Priority	High
Basic flow /inputs	Member is on dashboard Member clicks Complaint Member sees the Pending complaints Member selects the complaint Member can now see the Name, Address and Issue explained Member now clicks "On the way" status to notify User and Admin
Actor Actions	Member checks the complaint details
System Response	The submitted data reaches the Admin's side
Expected Result	Details of a registered complaint
Actual Result	Details shown to Team member
Failure Reason (if failed)	N/A

Table 6.49: Complaint Details

Extended test case 2 of the functionality for Complaint Details performed by the Team shown in table 6.50

Test Case ID	TC-50
Test Case Name	Complaint Details
Actor(s)	Team
Pre-Conditions	Member is logged in
Priority	High
Basic flow /inputs	Member is on dashboard Member clicks Complaint Member sees the Pending complaints Member selects the complaint Member can now see the Name, Address and Issue explained Member now clicks "On the way" status to notify User and Admin
Actor Actions	Member checks the complaint details
System Response	The submitted data reaches the Admin's side
Expected Result	Details of a registered complaint
Actual Result	No details shown to Team member
Failure Reason (if failed)	Complaint has been removed by admin or user

Table 6.50: Complaint Details

Extended test case 1 of the functionality for Proof-Picture performed by the Team shown in table 6.51

Test Case ID	TC-51
Test Case Name	Proof-Picture
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	High
Basic flow /inputs	Member arrived at the location Member fixed the complaint For proof, clicks the picture after it is fixed the User is in picture. Member writes Remarks beneath the picture Member clicks Submit The submitted data reaches the Admin's Notification module
Actor Actions	Member clicks proof picture
System Response	The submitted data reaches the Admin's side
Expected Result	Picture uploaded
Actual Result	Picture uploaded
Failure Reason (if failed)	N/A

Table 6.51: Proof-Picture

Extended test case 2 of the functionality for Proof-Picture performed by the Team shown in table 6.52

Test Case ID	TC-52
Test Case Name	Proof-Picture
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	High
Basic flow /inputs	Member arrived at the location Member fixed the complaint For proof, clicks the picture after it is fixed the User is in picture. Member writes Remarks beneath the picture Member clicks Submit The submitted data reaches the Admin's Notification module
Actor Actions	Member clicks proof picture
System Response	The submitted data reaches the Admin's side
Expected Result	Picture uploaded
Actual Result	Picture did not upload
Failure Reason (if failed)	You selected video instead of a picture

TABLE 6.52: Proof-Picture

Extended test case 1 of the functionality for Logout performed by the Team shown in table 6.53

Test Case ID	TC-53
Test Case Name	Logout
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	Medium
Basic flow /inputs	Member clicks on the Profile icon On the bottom, Member could see the Logout button Member clicks that button Member performs confirmation i.e., clicks Yes App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	Member logs out from the app
System Response	Member could see the Login Page successfully
Expected Result	Login Screen
Actual Result	Logout Screen
Failure Reason (if failed)	N/A

Table 6.53: Logout

Extended test case 2 of the functionality for Logout performed by the Team shown in table 6.54

Test Case ID	TC-54
Test Case Name	Logout
Actor(s)	Team
Pre-Conditions	Member logged in as Team Member
Priority	Medium
Basic flow /inputs	Member clicks on the Profile icon On the bottom, Member could see the Logout button Member clicks that button Member performs confirmation i.e., clicks Yes App loads for 3-4 seconds Member can see the Login page successfully
Actor Actions	Member logs out from the app
System Response	Member could see the Login Page successfully
Expected Result	Login Screen
Actual Result	Logout Screen
Failure Reason (if failed)	Internet Connection Lost

Table 6.54: Logout

6.4 Compatibility Testing

We conduct compatibility testing to test that our mobile application is working perfectly on different mobiles or not. When we run our application of different mobiles our application, we find that our application is compatible with them.

6.5 Exception Handling

During development we come across many exceptions, but we resolve them by displaying certain messages and instructions to the user. Some exceptions were:

- When the user is trying to register himself while he is already registered in the application.
- To resolve it we display the user with error message to let them know that they are already registered.
- While selecting location when there is lack of permission.To resolve it we ask the user to give permission to the application in order to proceed.

6.6 Load Testing

During load testing, a non-functional software testing process, we examine the performance of our application under a predefined predicted load. So, in this test we check how the application works when numerous people are using it simultaneously. We run our application on different mobiles at the same time and it has been working fine.

6.7 Security Testing

In this test we check whether our application is secure to use or not. The tasks that were performed to check security and their results are shown in Table 6.55.

Test Case ID	**Test case SEC-01**	
Description	Test the security of the app	
Initial Condition	Equipment is set as per requirement	
Steps	Task	Result
1.	Open the application	PASS
2.	Verify after logging out the application does not show any user data	PASS
3.	Verify that the password is in encrypted form	PASS
4.	Verify that the application does not allow any invalid users	PASS
5.	Verify that the application.	PASS

Table 6.55: Security Test Case

6.8 Installation Testing

The process of testing the processes required to install a functional software system is known as installation testing. To ensure that the software application has been correctly installed we conduct installation testing and check whether all the features are installed and working properly or not. The application passes this test as all the features were installed and they were also working properly.

Chapter 7

Conclusions

Fixome represents a transformative solution in the field of household maintenance, promising to streamline and modernize the way we address everyday issues. With a focus on user convenience and efficient service delivery, Fixome empowers users to report and track complaints related to water, gas, electricity, sewerage, and carpentry, all from the comfort of their homes. Our journey began with a vision to bridge the gap between traditional maintenance systems and the fast-paced, tech-driven world we live in. The Fixome app is not just a service; it's a commitment to making life easier for individuals and households, ensuring that no maintenance issue goes unresolved. The development process of Fixome adhered to the software development lifecycle, encompassing rigorous requirements analysis, thoughtful design, meticulous implementation, and rigorous testing. Our dedicated team of professionals, along with cutting-edge technology, ensures that every reported complaint, whether urgent or routine, receives the attention it deserves. Fixome isn't just an app; it's a comprehensive solution to your household maintenance needs. Users have the power to report issues, track progress, and provide feedback to maintain service quality. Our qualified team is poised to address problems efficiently, and we're always ready to respond to your inquiries. As we look ahead, Fixome promises to redefine the way we approach household maintenance. We envision a future where our services become even more convenient and efficient, making life more comfortable for our users.

7.1 Contributions

All the requirements related to technical and non-technical constraints are covered in this project report. This mobile application tries to solve the problems related to interaction between User/Complain-Filer, Admin/Officer, Team Worker/Vendor.

7.2 Disciplined Project Management

The key to the successful completion of a project and most importantly making the correct product is the conduction of proper and disciplined project management, and through the course of the project, we learned the importance of managing the time and the resources simultaneously to meet the schedule and the plan of the respective project and what effect these factors can have if they are not dealt with the cautious effort.

7.3 Team Communication

Another aspect of the importance we got familiar with is the role the communication between the team members which is the primary goal of all projects. The communication should be open, honest, and fair so that everyone in the team can contribute his/her best and never feel like their ideas and contributions are minimized or marginalized. To get the best result, it is important to maintain a good relationship with all the team members

7.4 Future Work

Still certain things can be enhanced in this project. Some of which are:

7.4.1 Restricted Registration

Restricting registration to only society residents within the app serves as a pivotal measure to ensure the platform's exclusivity and relevance to the community it serves. By limiting registration to society residents, we uphold the integrity of the platform, fostering a sense of community ownership and trust among users. It's imperative to ensure that the services provided within the app are tailored and accessible to those directly impacted by household maintenance issues in the society. This approach guarantees that the app remains a dedicated space for addressing local concerns efficiently and effectively, fostering a sense of belonging and active participation among the residents. Moreover, restricting registration helps maintain data accuracy, ensuring that the app's functionalities align with the specific needs and requirements of the society, thereby enhancing its overall utility and impact.

In cases where some residents might prefer external vendors for household maintenance services, despite the existence of the Fixome app, it's important to accommodate these preferences within the system. This highlights the necessity of ensuring flexibility in the platform to cater to varying user preferences and requirements. While the app aims to facilitate streamlined complaint reporting and resolution within the community, it should also recognize and respect the choices of residents who opt for alternatives.

For property owners renting out their spaces, the distinction between owners and tenants becomes crucial. The app's registration process should include an "owner or tenant" categorization to accurately record the status of users. Owners would need to provide proof of ownership, ensuring that the platform maintains a clear distinction between owners and tenants. The assignment of unique apartment numbers known only to owners and the society management team aids in maintaining confidentiality and security within the system.

Regarding the tenant registration process, collaboration between the tenant and owner with the society's administration office is proposed. This collaboration ensures that tenant details are properly documented, authenticated, and then updated within the society's database. This step guarantees accurate information storage and provides a mechanism for seamless communication between the app and the society's management system. The integration of Fixome' s database with the society's database enhances the platform's functionality, creating a comprehensive and centralized system for managing resident information.

This proposed solution aligns the Fixome app with the operational mechanisms of the residential society, ensuring efficient management of user data and seamless coordination between the app's functionalities and the society's administrative processes.

7.4.2 Other Platforms

Fixome application has been developed for android and iOS-based mobile devices due to the large market of consumers present in our country, however, our product can be developed for web platforms too, in the future.

7.4.3 Scalability

Right now, Fixome is planned to be deployed in a specific municipality. But according to the feedback from our users and community and innovation, we made the changes accordingly and make this app available to a greater number of users and also, we expanded the usage of the app to more regions, areas and municipalities and societies.

7.4.4 Additional Features

Cognizant of our commitment to continuous improvement, we acknowledge that user feedback and app usage data are invaluable. Based on these insights, we are open to exploring the integration of many additional features for the user, admin and team ease and satisfaction, for instance, AI models into Fixome, with the aim of enhancing operational efficiency and further elevating the user experience, all in the pursuit of greater customer satisfaction.

References

1. What is Maintenance
2. POTC, CCMS, Ministry of Energy, Power Division, 2013
3. mySociety, FixMyStreet, 2013
4. Love Clean Streets, SnapFix, 2013
5. New Urban Mechanics, StreetBump, 2012. Cited on page 9
6. Descriptive use case diagrams. Reference Template
7. Descriptive test cases. Reference template