MobiCare

Mini Project Report

Submitted by

Aswathy M J

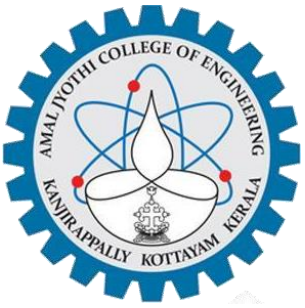
Reg. No.: AJC20MCA-I022

In Partial fulfillment for the Award of the Degree of

INTEGRATED MASTER OF COMPUTER APPLICATIONS

(INMCA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2024-2025

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING AUTONOMOUS

KANJIRAPPALLY

CERTIFICATE

This is to certify that the Mini Project report, "**MobiCare"** is the bona fide work of **ASWATHY M J (Regno: AJC20MCA-I022)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2024-25.

Mr. Binumon Joseph

Internal Guide

Mr. Binumon Joseph

Coordinator

Rev. Fr. Dr. Rubin Thottupurathu Jose
Head of the Department

DECLARATION

I hereby declare that the Mini project report **"MobiCare"** is a bona fide work done at Amal Jyothi College of Engineering Autonomous, towards the partial fulfilment of the requirements for the award of the **Integrated Master of Computer Applications (INMCA)** from **APJ Abdul Kalam Technological University**, during the academic year **2024-2025**.

Date: **ASWATHY M J**

KANJIRAPPALLY **Reg: AJC20MCA-I022**

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our Director (Administration) **Rev. Fr. Dr. Roy Abraham Pazhayaparampil** and Principal **Dr. Lillykutty Jacob** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Mr. Binumon Joseph, Assistant Professor** for his valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also express sincere gratitude to my guide **Mr. Binumon Joseph, Assistant Professor** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

ASWATHY M J

ABSTRACT

MobiCare is a comprehensive online platform catering to all your mobile needs, offering a seamless experience for mobile services and purchasing Mobile accessories. With three distinct modules – Admin (shop owner), Client, and Technician– the platform ensures efficient management and communication among all service folks who fix the phone complaints.

The admin module, shop owners can manage their store and add accessories. Clients, represented by users, can easily submit service requests, track repairs, and browse a diverse selection of phone accessories, as well as buy accessories and pay for services. Technician is responsible for phone repairs and issue resolution, efficiently handle service requests, ensuring timely solutions.

Key features include a user-friendly interface for submitting service requests, real-time tracking of repair progress, and secure payment options for both services and accessories. The platform facilitates seamless communication between clients and Technician, enhancing customer satisfaction. With MobiCare, users can conveniently access a wide range of mobile services and accessories, streamline the repair process, and enjoy a hassle-free shopping experience, all from the comfort of their own devices.

Languages: PYTHON, HTML, AJAX, JAVASCRIPT, CSS, JQUERY

Database: SQLite3

MODULES

In this online platform mainly for mobile services and purchasing mobile accessories, there are three modules:

1. Admin
2. User
3. Technician

1. Admin

- Login
- Manage Accessories: By adding, removing and updating mobile accessories.
- Manage Technician
- View Service Requests
- View Service Payment status
- Manage booking for the accessories
- View Payment status of booked accessories

- Manage Customer Feedback
- Logout

2. User

- User Registration
- Login
- Profile Updation
- Service Request Submission
- View the status of service request
- View status of repairing the complaints from technician
- View the history of service requests
- View the history of ordered product
- Payment
- Purchase Accessories
- Feedback
- Logout

3. Technician

- Login
- Profile Updation
- Manage the service request
- Schedule the services
- Update the status of repairing the complaints to user
- View the history of services done
- Update the status of payment
- View client Feedback
- Logout

.

CONTENT

SL. NO	TOPIC	PAGE NO
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	3
2.1	INTRODUCTION	4
2.2	LITERATURE REVIEW	5
2.3	PROPOSED SYSTEM	18
2.4	ADVANTAGES OF PROPOSED SYSTEM	19
3	REQUIREMENT ANALYSIS	20
3.1	FEASIBILITY STUDY	21
3.1.1	ECONOMICAL FEASIBILITY	21
3.1.2	TECHNICAL FEASIBILITY	22
3.1.3	BEHAVIORAL FEASIBILITY	22
3.1.4	FEASIBILITY STUDY QUESTIONNAIRE	23
3.2	SYSTEM SPECIFICATION	24
3.2.1	HARDWARE SPECIFICATION	24
3.2.2	SOFTWARE SPECIFICATION	24
3.3	SOFTWARE DESCRIPTION	25
3.3.1	DJANGO	25
3.3.2	SQLITE	26
4	SYSTEM DESIGN	27
4.1	INTRODUCTION	28
4.2	UML DIAGRAM	28
4.2.1	USE CASE DIAGRAM	29
4.2.2	SEQUENCE DIAGRAM	31
4.2.3	ACTIVITY DIAGRAM	32
4.2.4	CLASS DIAGRAM	34
4.2.5	OBJECT DIAGRAM	35
4.3	USER INTERFACE DESIGN USING FIGMA	37
4.4	DATABASE DESIGN	40
4.4.1	RELATIONAL DATABASE MANAGEMENT SYSTEM	40

4.4.2	NORMALIZATION	41
4.4.3	SANITIZATION	51
4.4.4	INDEXING	51
4.5	TABLE DESIGN	51
5	SYSTEM TESTING	60
5.1	INTRODUCTION	61
5.2	TEST PLAN	61
5.2.1	UNIT TESTING	62
5.2.2	INTEGRATION TESTING	62
5.2.3	VALIDATION TESTING	63
5.2.4	USER ACCEPTANCE TESTING	63
5.2.5	AUTOMATION TESTING	64
5.2.6	SELENIUM TESTING	64
6	IMPLEMENTATION	77
6.1	INTRODUCTION	78
6.2	IMPLEMENTATION PROCEDURE	78
6.2.1	USER TRAINING	79
6.2.2	TRAINING ON APPLICATION SOFTWARE	79
6.2.3	SYSTEM MAINTENANCE	80
6.2.4	HOSTING	80
7	CONCLUSION & FUTURE SCOPE	84
7.1	CONCLUSION	85
7.2	FUTURE SCOPE	85
8	BIBLIOGRAPHY	86
9	APPENDIX	91
9.1	SAMPLE CODE	92
9.2	SCREEN SHOTS	95
9.2	GIT LOG	100

List of Abbreviations

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
UML	-	Unified Modeling Language
Python	-	Python Programming Language
Django	-	Django Web Framework
SQLite	-	Structured Query Language Lite
CSV	-	Comma Separated Values

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

MobiCare is a comprehensive online platform designed to address all mobile-related needs, offering users a seamless experience for both mobile services and the purchase of mobile accessories. The platform is divided into three distinct modules: Admin (shop owner), User (client), and Technician, each contributing to efficient management and communication across the system. Through MobiCare, shop owners (admins) can manage their store by adding, removing, and updating accessories, overseeing technicians, handling service requests, tracking payment statuses, and managing customer feedback. Clients, on the other hand, can easily submit service requests, track the status of their repairs, purchase accessories, pay for services, and even sell their old phones through a secure and user-friendly interface. Technicians are responsible for managing and resolving service requests, updating clients on the repair progress, and handling the sale of old phones.

The platform's key features include real-time tracking of repair statuses, secure payment options, and a streamlined process for submitting service requests and purchasing accessories. It facilitates smooth communication between clients and technicians, ensuring timely service and enhancing customer satisfaction. With MobiCare, users can conveniently access a wide range of mobile services and accessories, manage repairs, and enjoy a hassle-free shopping experience—all from the comfort of their devices.

1.2 PROJECT SPECIFICATION

MobiCare's platform is structured into three primary modules, each offering specialized functionalities for different users. The **Admin Module** allows shop owners to manage mobile accessories, oversee technician operations, handle customer service requests, and track payment statuses for both service and product transactions. Admins can also manage booking requests and customer feedback efficiently. The **User Module** enables clients to register, log in, update profiles, submit service requests, and track their progress. Users can view their service history, purchase accessories, sell old phones, and provide feedback, all while accessing secure payment options. Finally, the **Technician Module** empowers technicians to manage service requests, schedule repairs, update clients on repair progress, and handle old phone sales. They can also view service histories, track payment statuses, and respond to client feedback.

Together, these modules provide a streamlined, user-friendly experience that simplifies the repair process, facilitates accessory purchases, and ensures transparent communication between users, technicians, and shop owners.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System study is a process of examining, analyzing, and evaluating an existing system or proposed system to identify its strengths and weaknesses, as well as opportunities for improvement. The purpose of system study is to understand the current system's functionality, constraints, and requirements, and to identify ways to improve its performance or replace it with a better system.

The first step in system study is to define the problem or opportunity that the system is supposed to address. This includes identifying the system's purpose, scope, and objectives. Then the next step is to gather data about the current system, including its processes, inputs, outputs, and interactions with other systems. This data can be collected through interviews, surveys, observations, and document analysis. Once the data has been collected, it is analyzed to identify patterns, trends, and issues that affect the system's performance. Based on the analysis, the next step is to identify the system's functional and non-functional requirements. Functional requirements describe the system's features and capabilities, while non-functional requirements describe the system's performance, reliability, and security.

Once the requirements have been identified, the system design is developed, which describes how the system will be built and implemented. This includes the system's architecture, data structures, algorithms, and user interface. Then the system has been built, it is tested to ensure that it meets the requirements and performs as expected. This testing can be done using various techniques, such as unit testing, integration testing, and system testing. The system is also evaluated to determine its effectiveness and efficiency. Once the system has been tested and evaluated, it is implemented in the production environment. This includes installing the system, training users, and maintaining the system. And finally, it is monitored and maintained to ensure that it continues to perform as expected. This includes monitoring performance, fixing bugs, and upgrading the system to meet changing needs.

The system study reveals a significant need for a dedicated, mobile-friendly website for the phone repair service business. A centralized platform would streamline key operational processes such as managing service requests, providing real-time repair tracking, and offering detailed information on services and pricing. These features would not only enhance transparency and customer satisfaction but also improve operational efficiency by reducing manual tracking and communication.

The study highlights essential customer needs, including online payment options, warranty information, and delivery services, which would make the business more accessible and trustworthy. Additionally, the demand for features like repair status tracking and comprehensive service descriptions underscores the importance of a user-focused, digital solution that caters to a

modern, smartphone-driven clientele.

A dedicated website could also address the current marketing limitations by establishing a stronger online presence, potentially attracting new clients beyond word-of-mouth referrals and social media. Integrating these functionalities would enhance customer engagement, build loyalty, and provide a structured pathway for business growth in a competitive market.

2.2 LITERATURE REVIEW

Existing System: In the traditional mobile service model, users are often required to bring their devices to a physical shop whenever they encounter an issue or need assistance. Whether it's a minor software glitch or a hardware problem, users typically have no guidance on resolving issues independently. The existing systems lack an integrated complaint-resolution feature, leaving users with limited options for addressing issues from home.

This approach presents several challenges:

- **Time-Consuming**: Users need to travel to a service center, which can be inconvenient and time-consuming, especially for minor issues.
- **Costly**: Many users incur service fees for problems that could potentially be resolved with simple instructions.
- **Limited Accessibility**: Users in remote areas or those with mobility constraints face additional difficulties accessing repair shops, which impacts the service experience

In order to relate the proposed Service Guidance System to current research on AI driven customer troubleshooting systems, service, and interactive user support, this section examines those studies.

AI-Powered Customer Service: An analysis of AI's potential to improve customer service, with an emphasis on the advantages and disadvantages of existing systems.

Chatbot Technology in E-Commerce: An analysis of chatbots' function in e commerce, including their capacity to offer prompt assistance and mentor users through challenging activities.

Automated Troubleshooting Systems: An outline of studies on automated troubleshooting systems, emphasizing its application in mobile repair and technical support.

Interactive Tutorials and User Assistance: An examination of research on interactive lessons and user support systems that demonstrates how well they help users navigate challenging tasks.

AI in Technical Support: An overview of AI's function in technical support that includes examples of successful applications across a range of sectors.

Machine Learning for Diagnostics: An analysis of the accuracy and dependability of machine learning techniques used for technical support system diagnostics.

Natural Language Processing (NLP) in Customer Support: This study examines how well customer support systems use NLP technology to comprehend and respond to user requests.

User-Centered Design of Service Systems: An analysis of studies on the application of user-centered design concepts to service systems, with a focus on the significance of usability and user pleasure.

Challenges in AI Integration: An outline of the difficulties encountered with incorporating AI into technical support and customer care systems, including problems with data quality, user confidence, scalability. and system

AI and Chatbot Technology: Adamopoulou and Moussiades provide a comprehensive look at chatbot technology[1], exploring its journey from simple, early systems like ELIZA to advanced AI-driven tools like IBM's Watson and Apple's Siri. They classify chatbots based on their tasks—some are designed for specific functions (like booking an appointment), while others handle broader conversations. Chatbots are also preferred in various areas, including customer marketing, healthcare. e-commerce, service, and The authors discuss the architecture of chatbots, focusing on how they interact with users, understand language, manage dialogues, and generate responses. In order to make chatbots smarter and better comprehend and react to user input, artificial intelligence (AI) and machine learning are essential. Natural Language Processing (NLP) is key, enabling chatbots to break down user questions, understand their meaning, and respond in a way that feel it natural. The application of chatbots in several domains is highlighted in the paper: Chatbots in customer service are ideal for use in a service guidance system for mobile repairs since they can respond to inquiries, offer information, and assist in troubleshooting issues. In e-commerce and marketing, they interact with consumers, assist them with their purchases, and make tailored suggestions. Chatbots are used in the healthcare industry to make appointments, give advice, and even provide mental health support. Chatbots can improve their effectiveness over time by learning from previous conversations with the use of AI and machine learning. There are difficulties, though, such comprehending intricate requests, monitoring the conversations, context and of handling unforeseen inputs. With developments in voice recognition, linguistic support, and emotion recognition, chatbots appear to have a bright future despite these obstacles. All things considers, the paper's insights are especially helpful for incorporating chatbot technology into a mobile repair service guidance system. Chatbots may help users troubleshoot, identify problems, and offer solutions, improving customer assistance through artificial intelligence. examining the success and need for improvement of conversational interfaces, especially AI chatbots, in influencing the customer experience.[45]

AI-Powered Challenges Customer and Service: Opportunities. Journal of AI Research.[2] Kim and Oh further explore how AI powered customer service can improve response times by automating routine tasks. They emphasize how AI systems can handle several client conversations

at once, greatly cutting down on wait times and raising customer satisfaction levels. This is especially helpful in sectors like e commerce and telecoms where prompt and precise responses are essential due to high customer service demands. The use of AI in sentiment analysis— where AI systems examine client feelings during interactions—is another topic covered by the writers. This enables businesses to better understand the moods of their customers and adjust their answers accordingly. For example, the AI can change its tone to be more sympathetic if a consumer is upset, which could defuse stressful situations and enhance the customer experience. Kim and Oh stress that although AI is capable of handling a variety of activities, human agents cannot be entirely replaced by it. Human intervention is still required for complex problems that call for in depth knowledge, empathy, or innovative problem-solving. The authors propose a hybrid strategy in which human agents concentrate on more delicate or complex problems while AI manages regular queries. The paper also addresses the challenges of AI integration within existing customer service frameworks. Businesses frequently struggle to integrate AI capabilities with their existing systems, which can result in problems like uneven service quality. AI systems must also be continuously trained and updated in order to stay in line with changing corporate procedures and customer expectations. This article highlights how AI may help the Service Guidance System by streamlining operations and providing users with faster, more individualized support. It also highlights how important it is to combine automation and human control to ensure that users receive the best service possible, especially in complex or emotionally charged situations.

NLP in Customer Support: A Comprehensive Review. AI and Customer Experience Journal.[4]: Lee and Kim's paper explores how Natural Language Processing (NLP) helps machine to understand and interact with human language. They cover different NLP techniques like sentiment analysis, which identifies emotions in user messages; intent recognition, which figures out what the user wants; and context understanding, which helps the system grasp the surrounding the user's query. situation The paper notes that while NLP is powerful, it also has challenges. Language ambiguity Machines may find it challenging to correctly interpret user meaning when there is ambiguity in the language. For instance, depending on the situation, the same sentence may have several meanings. Furthermore, because language and user behavior evolve over time, NLP models require ongoing training to be successful. Advances in Natural Language Processing (NLP) that improve customer service chatbot skills are covered in the study [31]. It also draws attention to successfully the difficulties integrating in NLP technologies in customer service settings. system because it enables the system to understand customer inquiries and offer pertinent, detailed recommendations for resolving mobile issues. NLP can increase the accuracy and usefulness of the service by enhancing the system's comprehension and response to user inquiries. Paper discusses

the current state of AI enabled customer support and offers insights into future directions and developments in this field.[43]

Speech Recognition and Natural Language Understanding in Modern Chatbots.[17]: Garcia and Torres examine the developments in chatbot technology, emphasizing how Natural Language Understanding (NLU) and speech recognition are used by these systems to process and comprehend spoken language. Chatbots can "listen" to spoken language and translate it into text thanks to speech recognition. NLU then assists the chatbot in deciphering the meaning of such words, improving the coherence and utility of exchanges. The authors discuss how modern chatbots use deep learning models, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to enhance their language understanding abilities. These technologies allow chatbots to converse with consumers in a more effective and natural way. They also tackle a number of issues that can make it more difficult for a chatbot to correctly understand user intent, like managing various dialects, slang, and unclear inquiries. NLP is essential to a service guidance The paper emphasizes the need for large training datasets to and improve continuous chatbot performance, ensuring they can better understand and respond to diverse user inputs. These insights are particularly relevant for developing voice-interactive systems in the Service Guidance System, where users can communicate directly with the chatbot for assistance. The total user experience can be enhanced by the developments covered, which could lead to more responsive and user-friendly voice based support.

Challenges and Opportunities Machine Learning for Diagnostics in Technical Support. International Journal of AI in Customer Service.[3]: Smith and Jones explore how machine learning helps diagnose technical problems in customer support. They go over several machine learning methods, such as neural networks, support vector machines, and decision trees. These methods are employed to evaluate user-reported problems and offer precise fixes. The paper includes case studies showing how effective machine learning is in speeding up the troubleshooting process and improving the accuracy of solutions. Machine learning algorithms, for example, can swiftly examine user complaint trends to find recurring problems and recommend solutions. Smith and Jones also highlight that machine learning can learn from past data, making it better over time at diagnosing problems. This means that as more issues are reported and resolved, the system becomes more accurate at predicting the best solutions. an empirical study on AI powered customer support solutions in the retail sector, assessing their effectiveness and the impact on customer experiences.[36] In practical terms, this research is useful for creating a Service Guidance System. Machine learning can help the system decide if a problem can be fixed at home or if it needs professional help. By analyzing user-reported issues and comparing them with past data, the system

can give precise recommendations on how to address the problem. Overall, the paper shows that machine learning not only speeds up problem solving but also enhances the reliability of the solutions provided, which is crucial for effective customer support and service systems**.** **Automating Customer Support with AI: Benefits and Limitations.[20]**: Johnson and Lee analyze the advantages and drawbacks of using AI to automate customer support services. They discuss several benefits of AI automation, such as increased efficiency, round-the-clock availability, and reduced costs for companies. AI, for instance, may swiftly complete routine and easy jobs, freeing up human agents to concentrate on problems. more difficult However, the authors also highlight some limitations. For instance, AI systems may struggle with understanding and responding to emotionally charged or complicated customer queries, which can lead to frustration. There's also the risk of losing the personal touch that human agents bring to user support. The authors stress the value of a mixed strategy in which ordinary activities are handled by AI while more delicate or complex problems are referred to human agents. This balance helps ensure that customers still receive high-quality support. The paper suggests training AI systems to recognize when they should hand off a task to a human, maintaining a high standard of service. These insights are valuable for designing AI-driven Service Guidance Systems that can provide automated help without sacrificing customer satisfaction. AI technologies are customer experience transforming in retail, discussing various applications and their implications for engagement and loyalty.[37] **Ethical Consideration and customer Design Ethical Consideration in AI-Driven Customer Services. Journal of Ethics in AI.[13] :** Park and Kim address the ethical issues related to using AI in customer service. They explore concerns like data privacy, where it’s crucial to protect user information and ensure that AI systems handle data securely. The paper also highlights the problem of bias in AI algorithms, which can lead to unfair treatment if the AI perpetuates existing Additionally, they prejudices. discuss the challenge of making sure that AI decisions are ethical and aligned with moral standards. The authors stress the need for the transparency, accountability, and fairness in developing and deploying AI system. They propose several measures to address these ethical concerns, including regular audits to check for fairness and the inclusion of diverse perspectives in AI training datasets to prevent bias. For the Service Guidance System, this paper provides essential guidance on ensuring that the AI solutions are not only effective but also ethical, respecting user privacy and avoiding any form of discrimination.

Ethical Considerations in AI-Powered Customer Support.[25]: Patel and Shah explore the ethical issues that arise when using AI in customer support. The paper discusses concerns such as privacy, where customers might worry about how their data is being used by AI systems. According to the authors, although AI can handle enormous volumes of data to improve support, it also poses

concerns over data security and the possible misuse of private data. Bias in AI algorithms, which may result in the unfair treatment of particular clients, is another ethical concern they draw attention to. For example, an AI system may inadvertently prefer one client group over another if it is educated on biased data. The paper also examines the these with disabilities, can transparency of AI systems, arguing that customers should be informed when they are interacting with AI rather than a human, and how their data is being used. The authors recommend several best practices to address these ethical concerns, such as implementing strict data protection policy, regularly auditing AI systems for bias, and ensuring that the AI decisions can be explained to customers in a way they understand. These ethical guidelines are important for the development of trustworthy Service Guidance Systems that respect customer privacy and fairness. analyzes the opportunities and challenges that AI presents for the future of customer service, exploring various technological advancements and their implications for service delivery. [44]

User Experience and System Design User-Centered Design of AI Service Systems. Human-Computer Interaction Journal. [7]: Nguyen and Brown's paper highlights the importance of use center design in development of AI service systems. They argue that AI systems should be intuitive, accessible, and responsive to the needs of users. Here are the key points from the paper: Intuitive Design: AI systems should be easy to understand and use, so users can interact with them without confusion. Accessibility: The design should ensure that all customer, including effectively use the system. Responsiveness: The system should adapt to users' needs and provide relevant information or support. The paper includes case studies showing how incorporating user feedback and conducting usability testing can significantly enhance the effectiveness of AI services. These studies demonstrate that iterative testing and adjustments based on user input can leads to more user-friendly and effective AI devices. Nguyen and Brown also address the challenge of balancing automation with user control. While automation can streamline processes and reduce manual effort, it's crucial that users still feel in control and are not frustrated by the system. The goal is to empower users by making the system helpful and easy to navigate. For the Service Guidance System, the research emphasizes designing an interface that is user-friendly and easy to navigate. This includes making sure users can troubleshooting quickly instructions access and follow them without difficulty, enhancing overall user satisfaction and effectiveness of the service.

User Experience Design for AI Driven Customer Support Systems.[24]: Ramos and Pereira discuss how to design user-friendly interfaces for AI driven customer support systems. The paper emphasizes that even though the technology behind these systems is complex, the interface should be simple and intuitive for users. They suggest that good design should make it easy for users to find what they need, understand the information presented, and receive immediate feedback from

the system. The authors recommend using human-centered design approaches, which involve testing the system with real users and making adjustments based on their feedback. They also highlight the importance of visual design elements, such as clear icons, consistent layouts, and responsive buttons, which can make the system more enjoyable and efficient to use. The paper includes examples of successful AI-driven support systems with well-designed interfaces, such as chatbots that guide users through troubleshooting steps with clear instructions and visuals. The authors stress that the design process should consider accessibility, ensuring that the system is usable for people with different abilities. Insights from this paper are crucial for creating a user-friendly interface for Service Guidance Systems, ensuring that users can easily navigate and benefit from the AI-driven support features. The report highlights how AI and machine learning might improve service automation while reviewing existing trends in automated customer support systems [33] and future prospects. AI's possible effects on customer satisfaction and service efficiency are discussed in the article "The Opportunities and Challenges of AI in E-Commerce Customer Support" [34]. Chatbots are transforming customer service by offering immediate support and enhancing client interaction through advanced AI techniques.[39]

Machine Learning and AI in Customer Support: Applications and Case Studies Interactive Tutorials for Technical Support: A Case Study. Journal of Interactive Learning.[5] Davis and Chen's paper provides a detailed case study on how interactive tutorials can be used in technical support. The authors explain that these tutorials guide users through complex technical tasks, which reduces the need for direct help from support agents. This approach allows users to fix common issues on their own, which improves customer satisfaction by empowering users to resolve problems independently. The paper outlines several key design principles for effective interactive tutorials: Clarity: The information provided in the tutorials should be straightforward and easy to understand, avoiding confusion. User Engagement: Tutorials should be designed to keep users interested and motivated to follow the instructions. Adaptability: The tutorials need to adjust based on the user’s progress and feedback, ensuring that they meet individual needs. The study shows that well-designed interactive tutorials can significantly enhance the user experience by making Customer Satisfaction: it easier for customers to troubleshoot and solve problems on their own. For a Service Guidance System, incorporating interactive tutorials could help users perform home repairs on their mobile devices, reducing the need for professional assistance and making the service more efficient and user-friendly.

Automated Troubleshooting Systems in E-Commerce. Technology Review.[6]: E-Commerce Patel and Sharma’s paper explores how automated troubleshooting systems are used in e-commerce platforms to handle customer issues without needing human help. They focus on how these systems

leverage AI and machine learning to identify and resolve problems. The paper details several important components of troubleshooting systems: automated Knowledge Bases: These are extensive databases that contain information and solutions for various problems, which the system uses to provide answers. Diagnostic Algorithms: These are processes and rules used by the system to analyze issues and determine their causes. User Interfaces: The part of the system through which customers interact to report their problems and receive solutions. The authors also discuss the benefits of these systems: Automated troubleshooting can provide quick improving experience. Operational the solutions, customer Efficiency: handling many By issues automatically, these systems can reduce the workload on human agents and streamline operations. For a Service Guidance System, understanding these components is crucial for creating an effective system that can diagnose and resolve mobile device problems within an e commerce setting. By integrating AI and machine learning, the Service Guidance System can offer timely and accurate troubleshooting support, enhancing both user experience and operational efficiency. This review discusses the interplay between AI and big data in enhancing customer service, emphasizing how data analytics can inform and improve customer support strategies.[46]

AI-Powered Diagnostic Systems for Mobile Devices.[19]: Wang and Zhang focus on the development of AI-powered diagnostic systems designed to detect and fix common problems in mobile devices. Large databases of data about device performance and failure causes are used to train machine learning models, which are used in these systems. The authors explain that these diagnostic systems can identify issues like battery problems, software bugs, and hardware malfunctions before they become serious. They also describe how these systems can guide users through troubleshooting steps by providing clear instructions based on the diagnostic results. The paper highlights the use of techniques like anomaly detection, which identifies when a device is behaving unusually, and predictive analytics, which helps to foresee potential problems before they happen. The development of user-friendly interfaces is also discussed, ensuring that even non-technical users can follow the guidance provided by these diagnostic systems. Integrating such AI-driven diagnostics into Service Guidance Systems can greatly improve their ability to help users fix device problems on their own.

AI-Driven Predictive Analytics for Customer Support.[23]: Singh and Gupta concentrate on the application of AI to anticipate customer service requirements before they materialize. Predictive analytics is a technology that employs artificial intelligence (AI) to evaluate historical consumer data and forecast future problems. For example, by looking at historical data, AI can forecast when a particular product might have problems, allowing customer support teams to reach out to customers proactively before they even experience an issue. The authors techniques used explain

in various predictive analytics, such as regression analysis, which predicts outcomes based on previous data, and neural networks, which can detect complex patterns in large datasets. The paper provides case studies where companies successfully used predictive analytics to reduce the number of customer complaints and speed up issue resolution times. However, the authors also mention challenges like integrating data from different sources, accuracy of ensuring predictions, the and interpreting the results in a way that leads to actionable insights. These methodologies are valuable for incorporating predictive analytics into Service Guidance Systems, enabling these systems to offer proactive support and improve customer satisfaction.

Technological Advances AI-Driven Solutions for Home-Based Mobile Repairs. Journal of Mobile Device Repair.[8]: Zhang and Wang's paper investigates how AI can assist with home-based mobile device repairs. They describe the development of AI tools designed to guide users through common repairs, such as fixing screens and replacing batteries. The authors explain how these tools use image recognition to analyze photos or videos of the device's issues, and diagnostic algorithms to evaluate the problem and provide step-by-step repair instructions. The research highlights how AI can help users address issues themselves, potentially reducing the need for professional repairs and lowering costs. By enabling users to perform repairs at home, AI tools can make the repair process more accessible and cost-effective. This research is highly relevant for the Service Guidance System, as it demonstrates how AI can be integrated to support users in troubleshooting and resolving mobile device problems independently. AI solutions can enhance customer loyalty on e-commerce platforms, examining the strategies and technologies that contribute to building long-term customer relationships.[40**]**

Adaptive AI Systems for Personalized Customer Support.[16]: Brown and Smith delve into the development of adaptive AI systems that personalize customer support by learning from each interaction. These systems are designed to tailor their responses based on the specific needs and behaviors of individual users. For instance, if a customer frequently inquires about certain products or services, the AI will begin to automatically provide more relevant information related to those interests. The authors highlight the use of reinforcement learning, a technique where the AI system improves its performance by learning from previous interactions—both successes and mistakes. This approach enables the AI to refine its responses over time, resulting in more accurate and timely support. The paper includes real-world examples where such adaptive systems have significantly enhanced customer satisfaction by offering faster and more precise responses. However, the authors also address several challenges. One key issue is ensuring the AI handles sensitive data securely, protecting user privacy. Additionally, maintaining the AI's effectiveness requires continuous training to keep the system up-to-date with the latest information and trends. For the Service

Guidance System, the insights from this paper are highly relevant. Implementing adaptive AI can improve user experience by providing more personalized and accurate support, while also addressing challenges related to data security and ongoing system training.

Implementing AI in Mobile Repair Services: A Case Study. Journal of AI in Industry.[9]: Green and White's paper presents a case study on integrating AI into mobile repair services. They explore the various challenges and successes involved in using AI tools within a mobile repair business. Key aspects covered in the paper include the development of diagnostic algorithms and the implementation of customer support chatbots. The authors emphasize the importance of training AI models with diverse datasets to ensure that the systems are accurate and reliable in diagnosing and resolving mobile issues. They also discuss how AI can impact customer satisfaction and improve business efficiency. For the Service Guidance System, this case study offers practical insights into the effective training of AI models and the practical aspects of integrating AI into service operations. It highlights how AI can enhance the accuracy of diagnostics and improve customer interactions, making the system more effective and user-friendly.

The Impact of AI on Service Quality in Online Retail.[26]: The writers talk about how AI is changing the customer experience through a variety of applications, including chatbots, recommendation engines, and automated customer service. They examine how AI improves response responsiveness times), (fast personalization (customized experiences for each individual client), and reliability (constant performance) to improve service quality. The study offers examples of how internet merchants have effectively used AI to improve customer satisfaction and loyalty by raising their service standards. The limitations of technology, the necessity of human supervision when dealing with delicate or complicated matters, and possible opposition from clients who might favor human contact over AI driven services are some of the difficulties that Kumar and Desai also point out. The study highlights the importance of finding a balance between human and AI automation in order to develop a thorough and effective. The study emphasizes how crucial it is to strike a balance between human interaction and AI automation in order to deliver a comprehensive and successful customer experience. Service.

Challenges in Integrating AI with Customer Relationship Management Systems.[18] Hernandez and Martinez talk about the difficulties businesses encounter when attempting to incorporate AI into their current CRM (customer relationship management) systems. CRM systems are used to manage a company's interactions with current and potential customers. The authors point out several key challenges, such as data silos, which are isolated pockets of data that are not shared with other systems, making it difficult for AI to access all the information it needs. They also talk about compatibility issues, where AI systems may not work well with older CRM software,

and the need for real-time data processing, which is essential for AI to provide timely and relevant support. The paper suggests solutions like using middleware platforms, which act as a bridge between different systems, allowing AI to communicate with CRM software more effectively. They also discuss how AI can enhance CRM by automating repetitive tasks, predicting customer behavior, and providing insights that help improve customer engagement. Case studies in the paper show how companies have successfully combined AI with their CRM systems to improve customer service and operational efficiency. This research is relevant for designing Service Guidance Systems that need to interact with CRM platforms to deliver comprehensive support services. With an emphasis on the technology and development processes, the writers offer a thorough analysis of intelligent customer service systems. The impact of AI technologies on customer relationship management (CRM) practices, discussing how AI enhances data analysis and customer interactions.[41] It explores how these systems improve customer interactions and service efficiency. [32]

Interactive and Real-Time Assistance-Interactive Solutions Real-Time Assistance for Mobile Device Troubleshooting. Journal of Real-Time Systems.[12]: Choi and Lee focus on creating systems that offer real-time assistance for troubleshooting mobile devices. Their paper explores how real-time data processing and AI algorithms can be used to provide immediate help for users dealing with technical issues on their mobile devices. They propose a framework for implementing real-time diagnostics, which allows the system to quickly identify the root cause of a problem and suggest practical solutions. The study also addresses the challenges associated with ensuring that these systems are both responsive and accurate. Real-time responsiveness is crucial to delivering timely support, while maintaining diagnostic accuracy in a constantly changing environment challenging. can be For the Service Guidance System, this research highlights the critical role of real-time processing in offering prompt and effective support. It underscores the need for a system that can quickly diagnose issues and provide users with actionable guidance, ensuring that help is both immediate and reliable. The future of customer interactions in the context of AI technologies, highlighting potential transformations in customer service experiences.[42]

Future Trends and Technologies Managing Customer Service in the Digital Age: AI and Beyond. Journal of Digital Transformation.[10] Miller and Roberts examine how AI is transforming customer service in the digital era. They focus on how AI tools are integrated into digital platforms, marking a shift from traditional call centers to advanced AI-powered chatbots and virtual assistant. The study lists a number of advantages of AI in customer service, including: 24/7 Availability: AI systems are able to assistance, offer round-the-clock guaranteeing assistance is always available. that Personalized Responses: By using user data to customize responses, AI can improve the

effectiveness and relevancy of interactions. Lower Operational Costs: AI can reduce the expenses related to customer service operations by automating repetitive jobs. However, the authors also address the challenge of maintaining human empathy in automated interactions. They note that while AI can handle many tasks efficiently, it may struggle to replicate the emotional understanding and personal touch of human agents. This paper presents case studies on successful implementations of AI in customer service, highlighting best practices that can be adopted by organizations to improve their customer support.[35] For the Service Guidance System, this paper underscores the potential of AI to deliver instant, accurate solutions to technical problems, enhancing overall customer support and satisfaction. It demonstrates how AI can streamline service processes and provide immediate assistance, crucial for effectively managing mobile device troubleshooting and repair. Future Directions in AI for E Commerce. Journal of E-Commerce Innovation.[15] Wilson and Moore explore how AI is set to revolutionize the e-commerce industry, focusing on emerging technologies that are reshaping online shopping and customer service. They discuss advancements such as AI driven product recommendations, which suggest products based on user preferences, and dynamic pricing, which adjust price in real-time based on market condition. Automated customer support systems are also highlighted for their role in providing efficient, around-the-clock assistance. The paper further examines how AI can be integrated with augmented reality and virtual reality to create immersive online shopping experiences, allowing customers to virtually try products before buying. However, the authors also address challenges like scalability, ensuring AI systems can handle increasing amounts of data and users, data security, protecting user information from breaches, and maintaining customer trust in AI-driven processes. For the Service Guidance System, these insights Leveraging AI recommendations are for and invaluable. product dynamic pricing can enhance user experience and service efficiency on platforms like Mobicare, while AR and VR integration can offer a more engaging shopping experience. Addressing scalability, security, and trust issues will be essential for successful implementation.

Ethical Considerations and Emerging Trends, Ethical and Privacy Issues Enhancing User Engagement with AI Systems. Journal of AI and Human Interaction.[14]: With an emphasis on improving the user experience, Ali and Ahmed investigate ways to increase user engagement with AI systems. They talk about how AI can customize experiences, like through adaptive interfaces and tailored content recommendations, to make interactions more interesting. By customizing interactions to each user's requirements and preferences, these strategies can boost user satisfaction and encourage repeat business. The significance of interactive feedback loops, which enable more dynamic user interaction with the AI system, is also emphasized in the article. AI is

able to predict user needs and provide proactive support by comprehending user behavior and preferences. Users will find the experience more interesting and relevant with this method. For the Service Guidance System, these insights Incorporating are strategies crucial. that personalize and adapt to user needs can significantly enhance the service experience, ensuring that users find the support they receive both relevant and satisfying.

Emerging Technologies and Innovations AI and the Future of E-Commerce: Emerging Trends Technologies.[21] and Ng and Tan examine how artificial intelligence (AI) is transforming e commerce by bringing in new trends and technologies that are improving the efficiency and personalization of online buying. The study investigates how algorithms driven by AI can analyze customer preferences and behavior to recommend products that consumers are more likely to buy. For example, if a person frequently shops for electronics, the AI might suggest the newest gadgets to them based on their browsing habits. The authors also discuss how AI can be used to optimize logistics, including inventory control and supply chain optimization for quicker delivery times. The usage of AI in chatbots and virtual assistants, which may aid customers with product discovery, question answering, and even purchase completion, is another fascinating trend they cover. The paper also touches on the integration of AI with augmented reality (AR), where users can see how products would look in their own homes before buying them. However, the authors also point out challenges, like ensuring data privacy, ethical concerns about AI decision-making, and the need for strong cybersecurity measures to protect against hacking. Insights from this research can help in designing Service Guidance Systems that align with these future trends in e commerce, offering users a more personalized and secure shopping experience.

AI in Service Management: Trends and Innovations. Management Journal.[11]: Service Thompson and Evans explore how AI is revolutionizing service management by introducing the latest trends and innovations. They discuss various AI technologies that are optimizing service operations, enhancing customer interactions, and improving decision-making processes. innovations include: Key Predictive Maintenance Systems: AI tools that anticipate equipment issues before they occur, allowing for proactive maintenance. Intelligent Chatbots: Advanced chatbots that provide efficient and context-aware customer support. Automated Workflow Management: AI systems that streamline and automate service processes, increasing operational efficiency. The paper also explores how AI integrates with the Internet of Things (IoT) to offer real-time datas and analytics, further enhancing service efficiency by providing actionable insights. For the Service Guidance System, this research highlight how these AI innovations can be used to create more proactive and personalized service solutions, offering users timely and relevant support for their mobile repair needs. The Role of ML in Enhancing Customer Support Chatbots.[22] Lopez

and Morales investigate how machine learning (ML) might improve the efficacy of chatbots used for customer service. They clarify that machine learning (ML) enables chatbots to learn from previous exchanges, improving their comprehension and ability to address client inquiries. A chatbot that receives a lot of inquiries about shipping, for instance, may learn to give more thorough and precise responses regarding delivery schedules. Several machine learning models are covered in the study, including supervised machine learning, in which the chatbot is taught on labeled data (such as right replies), and unsupervised learning, in which it finds patterns in data without direct supervision. The authors also delve into how natural language processing (NLP) helps chatbots understand the meaning behind the words users type or speak, and how sentiment analysis allows the chatbot to gauge the customer's mood or satisfaction level. Continuous learning is crucial for these chatbots to adapt to new trends and customer needs. The study also highlights challenges, like ensuring data quality, managing biases in the training data, and maintaining the chatbot's accuracy over time. This research is directly applicable to the design of intelligent chatbots within Service Guidance Systems, helping to create more responsive and helpful AI assistants.

2.3 PROPOSED SYSTEM

The Service Guidance System represents a valuable innovation in leveraging AI to enhance customer support and streamline service processes in mobile repair. By utilizing machine learning and natural language processing, the system effectively classifies mobile complaints and provides targeted solutions, either through step-by-step home-based instructions or by recommending a visit to a repair shop for more complex issues. This approach offers customers immediate, accurate guidance, increasing convenience and satisfaction while potentially reducing the need for in-person repair visits.

The initial results demonstrate the model's effectiveness, particularly in classifying issues and providing actionable solutions. However, to achieve optimal implementation, further improvements are needed in diagnostic accuracy, user interface design, and system scalability. As this technology continues to develop, it promises not only to elevate the user experience but also to shape the future of AI-driven customer service in the e-commerce and mobile service sectors. With continued research and refinement, the Service Guidance System could set a new standard for AI-assisted troubleshooting and customer support.

The proposed system performs the following key functions:

Complaint Classification: When a user inputs a mobile phone-related issue, the system analyzes the complaint to determine whether it can be solved at house or whether they are requires to visit a repair shop. This is achieved by classifying the complaint using a trained machine learning model.

Home-based Solutions: If the complaint can be resolved at home, the system provides the user with step-by-step instructions on how to fix the issue. These instructions are designed to be simple, clear, and easy to follow.

Shop Visit Recommendations: For issues that cannot be resolved at home, the system advises the user to visit a shop for further assistances.

User-friendly Interaction: The system aims to offer a chatbot-like experience, that making it easy for customer to interact with and quickly get the data they need without extensive technical knowledge.

2.4 ADVANTAGES OF PROPOSED SYSTEM

- **Improved Customer Experience**: By offering a chatbot-like interaction, the system allows users to quickly identify solutions to their issues without needing extensive technical knowledge. This user-friendly design enhances satisfaction and reduces the time users spend troubleshooting.
- **Convenience with Home-Based Solutions**: Users can resolve minor issues independently at home with clear, step-by-step instructions, saving time and potentially avoiding unnecessary trips to the repair shop.
- **Time and Cost Efficiency**: Users avoid expenses associated with repair visits when minor issues can be resolved at home, and they save time by receiving relevant recommendations directly through the chatbot interface.
- **Accurate Service Recommendations**: The machine learning model accurately classifies complaints, ensuring users are only advised to visit a repair shop when necessary, optimizing repair center workload by focusing on cases that require in-person support.
- **24/7 Availability**: Unlike traditional support, which may operate within specific hours, this system can operate continuously, allowing users to troubleshoot and receive assistance anytime, even outside standard business hours.
- **Reduced Strain on Customer Support**: The system can handle routine issues autonomously, freeing up customer support teams to address more complex cases and improving overall efficiency.
- **Data-Driven Insights for Improvement**: By collecting and analyzing complaint data, the system could potentially improve service and product quality, enabling the business to understand common issues and make informed decisions about product enhancements or customer support improvements.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is an important phase in software development process. It enables developers to have a clear picture of the product being developed in terms of outcomes of the product, operational requirements for implementing it, etc.

Feasibility study is test of system proposal according to its workability, impact on organization, ability to meet the needs, effective use of resources. During the study, the problem definition is crystallized and aspects of the problem to be included in this system are determined. The result of the feasibility study is a formal proposal. If the proposal is accepted, we continue with the project. It is a comprehensive analysis that evaluates the viability of a proposed project or business idea. The study assesses the technical, financial, operational, legal, and regulatory aspects of the proposed project, to determine whether it is feasible and viable.

The purpose of a feasibility study is to provide the project team, stakeholders, and investors with a clear understanding of the risks, opportunities, and challenges associated with the proposed project. The study helps to identify potential problems or obstacles that may arise, and to develop strategies to mitigate or overcome them.

A typical feasibility study includes several key components, such as market analysis, financial projections, risk assessment, technical evaluation, and stakeholder consultation. The study involves extensive research and analysis, using both quantitative and qualitative methods, to gather and analyze data and information related to the proposed project. The findings of the feasibility study are used to inform decision-making and to develop a realistic and achievable plan for the proposed project.

3.1.1 Economical Feasibility

Economic feasibility is a key component of a feasibility study that assesses the financial viability of a proposed project or business idea. This aspect of the study evaluates the project's ability to generate sufficient revenue to cover its costs and produce a profit over the long term.

To determine the economic feasibility of a project, a financial analysis is conducted to estimate the project's costs, revenues, and expected returns on investment. The metrics help to determine whether the project is financially feasible and whether it is likely to generate positive returns for investors and stakeholders. Ultimately, the economic feasibility analysis helps to inform decision-making and to identify potential risks and challenges that may need to be addressed to ensure the success of the project.

Economic analysis is most frequently used method for evaluating the effectiveness of a candidate

system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs.
The proposed system is economically feasible one. We do not want to keep lot of books for storing the data. By manipulating data using computer reduces cost. We do not want lot of employees; we simply want one to operate it, Administrator.

3.1.2 Technical Feasibility

Technical feasibility is a critical aspect of a feasibility study that assesses whether a proposed project is technically viable and feasible to implement. This aspect of the study involves evaluating the project's technical requirements and constraints, and determining whether the necessary resources, skills, and technology are available to successfully execute the project.
The technical feasibility analysis typically involves evaluating the project's software and hardware requirements, assessing the availability and suitability of technology solutions, and identifying any technical risks or limitations that could impact the project's success. Ultimately, the technical feasibility analysis helps to ensure that the proposed project is technically feasible and viable, and that it can be implemented within the required timeframe and budget.
Technical feasibility centers on the existing computer system and to what extend it can support the proposed system. It involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project is judged not feasible.
Here we need only a computer working in low speed to accomplish the task.

3.1.3 Behavioral Feasibility

Behavioral feasibility is an important aspect of a feasibility study that assesses the willingness and ability of stakeholders to adopt and use a proposed project or system. This aspect of the study evaluates the human factors that may impact the success of the project, including user preferences, attitudes, and behaviors.
The behavioral feasibility analysis typically involves conducting user surveys, focus groups, and other research to gather information on user needs and preferences. The analysis may also include an assessment of the organizational culture and readiness for change, as well as an evaluation of the impact of the proposed project on employees and other stakeholders. The findings of the analysis help to identify potential barriers to adoption and to develop strategies to overcome them.
People inherently resist change and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development computerized system. The computer installations have something to do with turnover, retraining

and changes to employee status. In the proposed system, it behaves very feasibly. It is very easy to train the people in the proposed system. We simply want to tell the purpose of each button and about a little data to enter.

3.1.4 Feasibility Study Questionnaire

1. **Do you currently have a website for your phone repair services?**

   No, we do not currently have a website dedicated to our phone repair services.

2. **If you were to get a website, what essential features would you like it to have?**

   We would prioritize features such as:

   a. Service request submission forms with tracking capabilities.

   b. Online payment integration for service fees.

   c. A mobile-friendly interface.

   d. Clear service descriptions and pricing.

   e. Delivery services

   f. Warranty card details can be provided to the user.

3. **What are the main challenges you face without having a dedicated website?**

   One of the main challenges is the inability to provide detailed information about our services and pricing in a centralized and accessible manner. Managing service requests and appointments can also be more cumbersome without a dedicated online platform.

4. **How do clients usually find and contact you for repair services?**

   Clients typically find us through recommendations from friends or family, or they discover us on social media platforms where we actively post about our services.

5. **What information about your services do you think is most important to display on a website?**

   Key information would include our range of repair services, pricing details, turnaround times, contact information, and customer testimonials.

6. **How important is having a mobile-friendly website for your business?**

   It’s crucial because many of our client access information and services through their smartphones. A mobile-friendly website would enhance user experience and accessibility.

7. **Would you like a feature that allows clients to track the status of their repairs online?**

   Yes, integrating a feature for clients to track the status of their repairs online would greatly benefit our business by enhancing transparency, improving customer satisfaction, and boosting operational efficiency. It would provide clients with real-time updates on their repair progress, reducing the need for frequent inquiries and increasing trust.

8. **What types of phones do you primarily service (e.g., brands, models)?**

We service a variety of mobile phone brands and models, including popular ones like Apple (iPhone series), Samsung (Galaxy series), Huawei, Xiaomi, OnePlus, and others.

9. What is the average turnaround time for a typical repair?

The average turnaround time for a typical repair varies depending on the complexity and nature of the repair. For standard repairs, such as screen replacements, battery replacements, or software fixes, the turnaround time is generally within a few hours to a couple of days. For more complex repairs or issues that require parts ordering, the turnaround time may extend up to a week or longer.

10. What is your policy on warranty or guarantees for repairs?

Warranty is not provided for all services. And for the battery and display services the warranty cards are normally provided. And the validity is about one month.

11. In this shop any website is used for purchasing accessories and phones?

No, there is no website are available for purchasing phones and accessories. Only offline services are available.

12. If a person did not need the phone after the service has been done then what is the next step you will be taking?

When the person gives us the phone for service he should definitely need to agree to the terms and condition. And in the terms and condition it is mentioned that if the user did not buy the phone after service is completed, the phone will be sold after 15 days.

3.2 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor - Intel(R) Core (TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

RAM - 8.00 GB

Hard disk - 2 1 9 G B

3.2.2 Software Specification

Front End - HTML, CSS

Back End - Python Django

Database - SQLite

Client on PC	-	Windows 7 and above.
Technologies used	-	JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Django

Django is a high-level Python web framework [47] that allows developers to build robust and scalable web applications quickly. It was developed in 2003 by Adrian Holovaty and Simon Willison while working at the Lawrence Journal-World newspaper. The framework emphasizes reusability, rapid development, and follows the "Don't Repeat Yourself" (DRY) principle, allowing developers to focus on writing clean, maintainable code. Django is designed to handle both simple websites and complex web applications with ease, and it comes with many built-in features that simplify the development process, such as an admin interface, user authentication, and a templating engine.

- **Server-side scripting:** Like PHP, Django operates on the server side, processing requests, interacting with databases, and generating dynamic content to send to the client browser.
- **Python-based:** Django is built using Python, a popular, easy-to-learn programming language with a clean syntax, making it a great choice for both beginners and experienced developers.
- **Model-View-Template (MVT) architecture:** Django follows the MVT pattern, where the model handles data, the view controls the logic, and the template handles the presentation, providing a clean separation of concerns.
- **Built-in admin interface:** Django provides an automatically generated admin panel that allows developers to easily manage and modify their database without needing custom code.
- **Cross-platform compatibility:** Django can run on various operating systems, such as Windows, Linux, and macOS.
- **Object-relational mapping (ORM):** Django's built-in ORM allows developers to interact with databases using Python code instead of raw SQL, simplifying database management.
- **Security features:** Django has many built-in security features, such as protection against SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
- **Scalability and flexibility:** Django is highly scalable and is used by many large websites and applications, including Instagram, Pinterest, and the Mozilla Foundation.

- **Large developer community:** With an active and growing community, there are many resources, plugins, libraries, and tutorials available for Django developers.
- **REST framework integration:** Django can easily integrate with Django REST Framework (DRF) to create RESTful APIs for mobile and single-page applications (SPAs).

3.3.2 SQLite

SQLite is a lightweight, serverless, and self-contained relational database management system (RDBMS) designed to be embedded directly into applications. Unlike traditional databases like MySQL, SQLite does not require a separate server process, making it highly portable and easy to set up. Created by D. Richard Hipp in 2000, SQLite was intended to provide a fast, reliable, and efficient database engine for small to medium-sized applications. It is particularly well-suited for applications where simplicity and minimal overhead are important.

One of the standout features of SQLite is its **serverless architecture**.[48] There is no need for a dedicated server or complex configuration—SQLite reads and writes data directly to ordinary disk files. This eliminates the need for a network connection between a client and server, making SQLite an ideal choice for local, embedded applications such as mobile apps, desktop software, and small-scale web projects. The entire database is stored in a single cross-platform file, which makes it easy to manage, share, and back up.

Cross-platform compatibility is another advantage of SQLite. It runs on all major operating systems, including Windows, macOS, Linux, iOS, and Android. The database file format remains consistent across these platforms, allowing it to be easily transferred between different environments without compatibility issues. Additionally, SQLite is fast and efficient, making it well-suited for applications with limited resources. Its small size, typically less than 1MB, makes it ideal for mobile or embedded systems where performance and storage constraints are a concern. While SQLite does not support the client-server model like MySQL, it fully supports most of the SQL standard, including complex queries, transactions, and triggers. It provides the necessary functionality for many applications, making it sufficient for typical use cases. Although it may lack some advanced features found in larger databases, its simplicity and low overhead make it a popular choice for many developers. Additionally, SQLite is **self-contained**, requiring no setup or administration, which further adds to its appeal in smaller projects or as a lightweight solution in production environments. SQLite is widely used in various applications, including web browsers, operating systems, and mobile apps. Popular platforms such as Android, iOS, and Firefox rely on SQLite for data storage. Moreover, it is an excellent choice for testing, prototyping, and small-scale applications that do not require the full capabilities of larger databases like MySQL.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It involves understanding the problem domain, identifying the key features and functionality required, and designing a solution that meets those needs.

System design can be applied to a wide range of domains, including software systems, hardware systems, and even organizational systems. The process typically involves several stages, including requirements gathering, analysis, design, implementation, and testing.

Good system design requires a deep understanding of the problem domain, as well as an ability to communicate and collaborate effectively with stakeholders, including users, developers, and other stakeholders. It also involves a solid understanding of various design principles and patterns, as well as an ability to balance trade-offs between different design decisions.

Some important considerations in system design include scalability, performance, maintainability, security, and usability. These factors should be carefully considered throughout the design process to ensure that the resulting system is both effective and efficient, as well as easy to use and maintain over time.

4.2 UML DIAGRAM

Unified Modeling Language (UML) is a visual language [49] that is used to design and document software systems. It provides a standard way to visualize the various components of a system and their relationships, making it easier to communicate complex ideas to stakeholders and team members. UML diagrams are used to represent the different aspects of a system, such as its structure, behavior, and interactions.

There are several types of UML diagrams, each with its own purpose and level of detail. For example, class diagrams are used to represent the classes and their relationships in an object-oriented system, while sequence diagrams show the interactions between different parts of a system over time. Other types of UML diagrams include activity diagrams, use case diagrams, and state machine diagrams.

Using UML diagrams can help improve the quality of software systems by providing a clear and concise way to communicate design ideas and requirements. By creating a visual representation of a system, developers can easily identify potential problems and design flaws, and make changes before they become more difficult and costly to fix. Additionally, UML diagrams can help ensure consistency and clarity across different parts of a system, making it easier for developers to

understand and modify code written by others.

4.2.1 USE CASE DIAGRAM

A use case diagram is a type of UML diagram that represents the interactions between actors (users or external systems) and a system. It is used to model the various use cases of a system and how they relate to each other. Use case diagrams help to clarify the requirements of a system and provide a clear understanding of the functionality it should have.

In a use case diagram, each use case represents a specific action or task that a user can perform within the system. Actors are represented as stick figures and are connected to the use cases that they interact with. The relationships between actors and use cases are depicted using lines and arrows, which show the flow of information and actions between them.

Use case diagrams are useful for several reasons. Firstly, they provide a high-level view of the system, making it easier for stakeholders to understand and review the functionality of the system. They also help to identify any gaps or inconsistencies in the system's requirements, which can be addressed before development begins. Additionally, use case diagrams can be used as a basis for designing the user interface and for creating test cases to ensure that the system functions correctly.

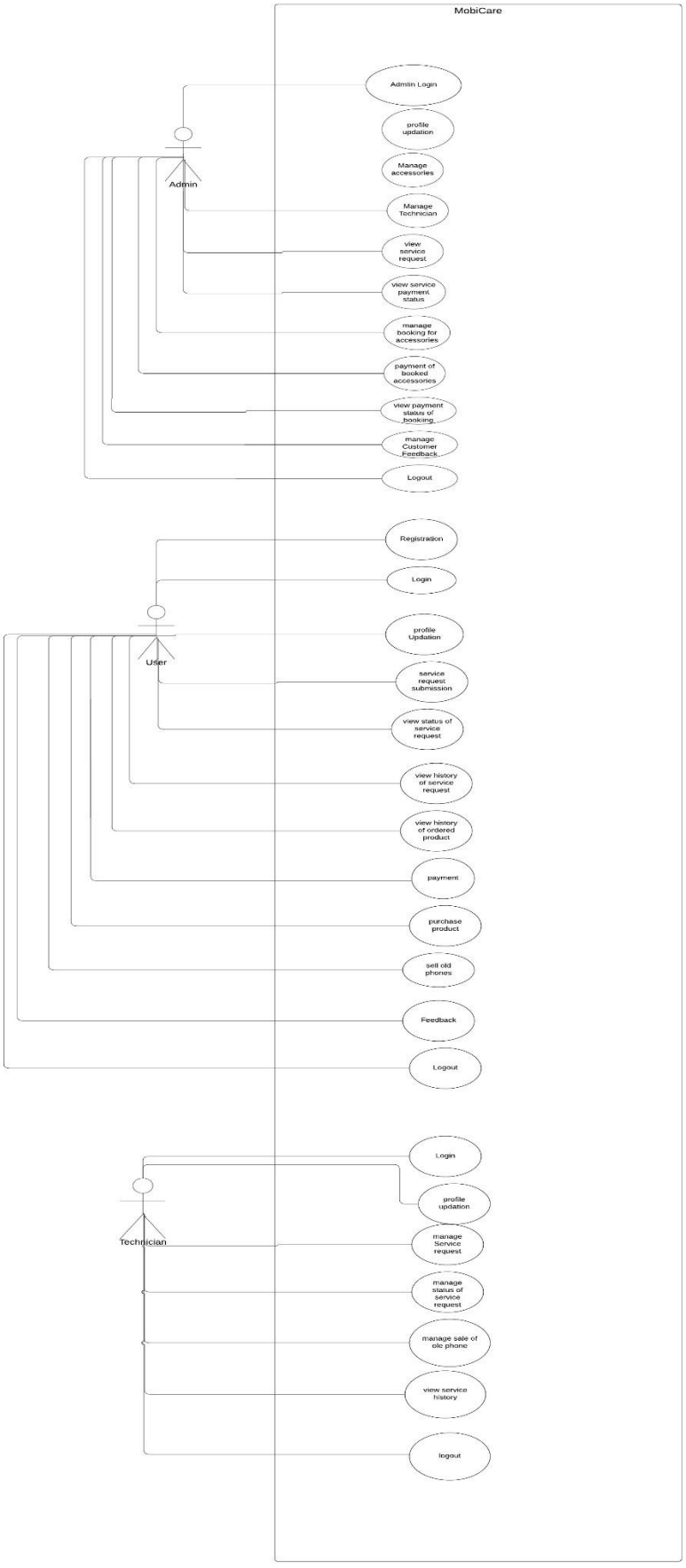


Fig 1: use case diagram

4.2.2 SEQUENCE DIAGRAM

A sequence diagram is a type of UML diagram that depicts the interactions between objects or components in a system over time. It shows the sequence of messages exchanged between these objects or components, as well as the order in which they are sent and received. Sequence diagrams are commonly used to model the behavior of a system and to design and test software applications. In a sequence diagram, the vertical axis represents time and the horizontal axis represents the objects or components involved in the system. Objects are represented by boxes, and messages are represented by arrows. The sequence of messages is shown from top to bottom, with each message following the one before it.

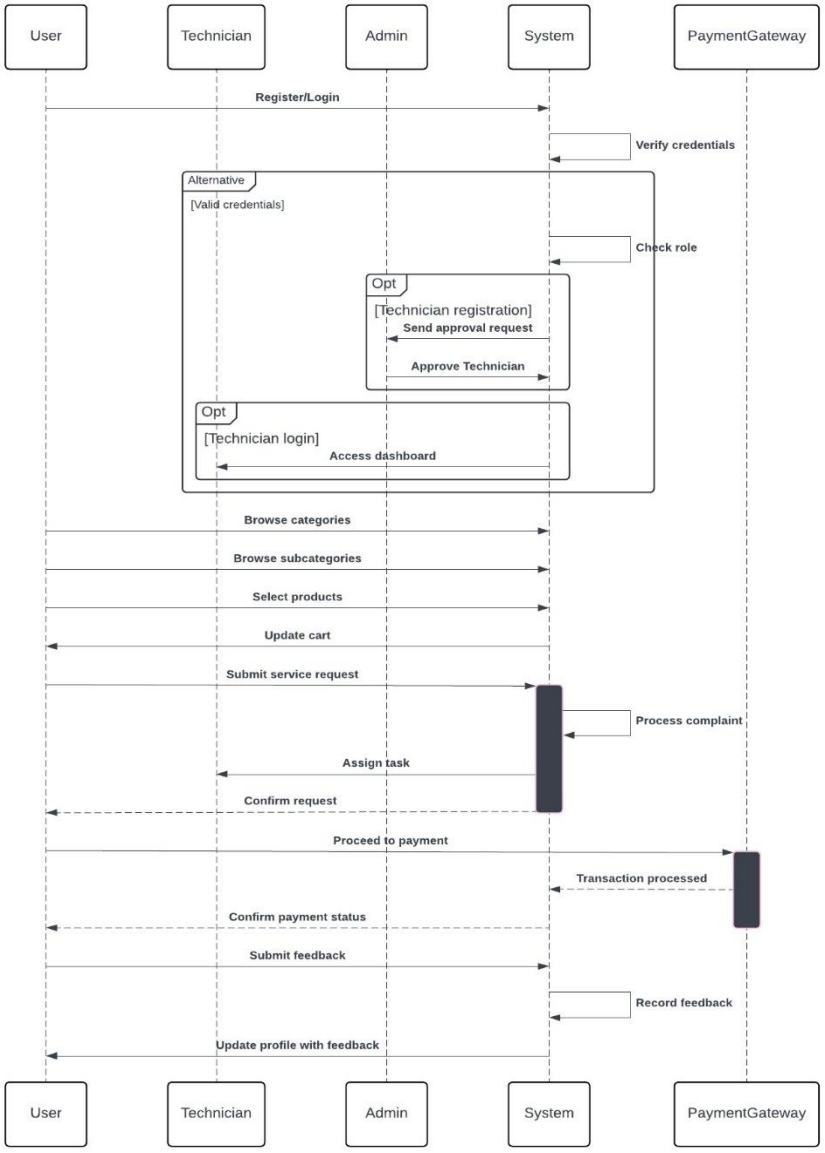


Fig 2: sequence diagram

4.2.3 ACTIVITY DIAGRAM

An activity diagram is a type of UML diagram that models the flow of activities or processes in a system. It shows the sequence of steps, decisions, and actions that are taken to achieve a particular goal. Activity diagrams are commonly used to model business processes, software workflows, and system behaviors.

In an activity diagram, activities are represented by rounded rectangles, while decisions and branching paths are represented by diamonds. Arrows connect the different activities and decisions, showing the flow of the process. Symbols such as loops and swimlanes can also be used to model complex activities and interactions between different actors or components in the system.

Activity diagrams are useful for several reasons. Firstly, they provide a clear and concise representation of a system's behavior, making it easier for stakeholders to understand and review the process. They also help to identify potential problems and inefficiencies in the process, such as bottlenecks or unnecessary steps. Additionally, activity diagrams can be used to optimize and streamline the process, improving efficiency and reducing costs. Overall, activity diagrams are an essential tool for designing and developing systems that are effective, efficient, and meet the needs of users and stakeholders.

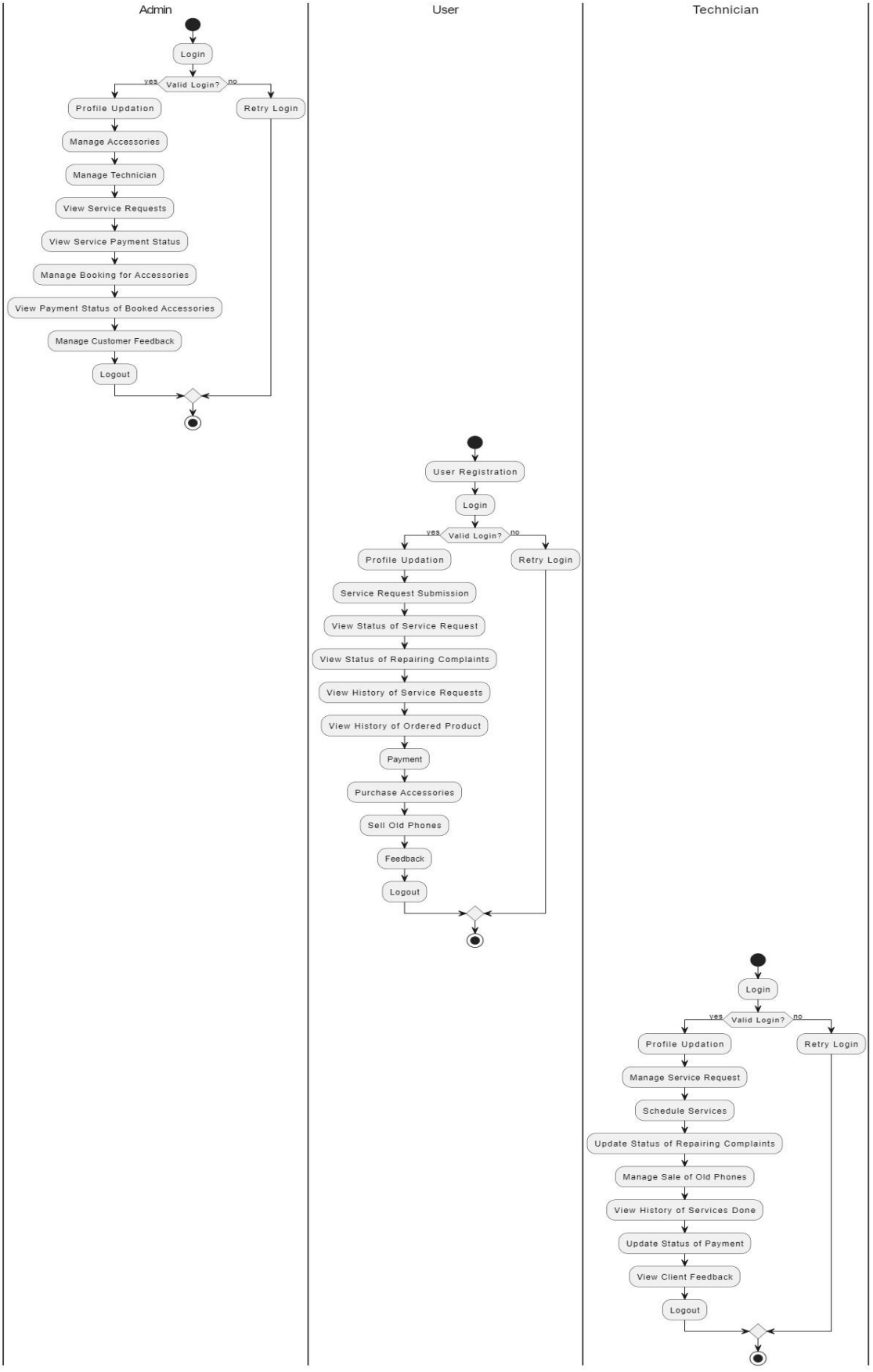


Fig 3: Activity diagram of admin

4.2.4 CLASS DIAGRAM

A class diagram is a type of UML diagram that represents the structure of a system by showing the classes and their relationships. It is used to model object-oriented systems and provides a visual representation of the various components of the system and how they interact. Class diagrams are commonly used in software development and are useful for designing, implementing, and testing software applications.

In a class diagram, classes are represented as boxes, with their attributes and methods listed inside. Relationships between classes are depicted using lines and arrows, which show the associations, dependencies, and inheritance between them. Class diagrams can also include multiplicity, which shows how many instances of a class can be associated with another class.

Class diagrams are useful for several reasons. Firstly, they provide a clear and concise representation of a system's structure, making it easier for stakeholders to understand and review the components of the system. They also help to identify potential problems and design flaws, such as classes with too many dependencies or too many attributes. Additionally, class diagrams can be used to generate code automatically, making the development process more efficient and reducing the potential for errors. Overall, class diagrams are an essential tool for designing and developing object-oriented systems that meet the needs of users and stakeholders.

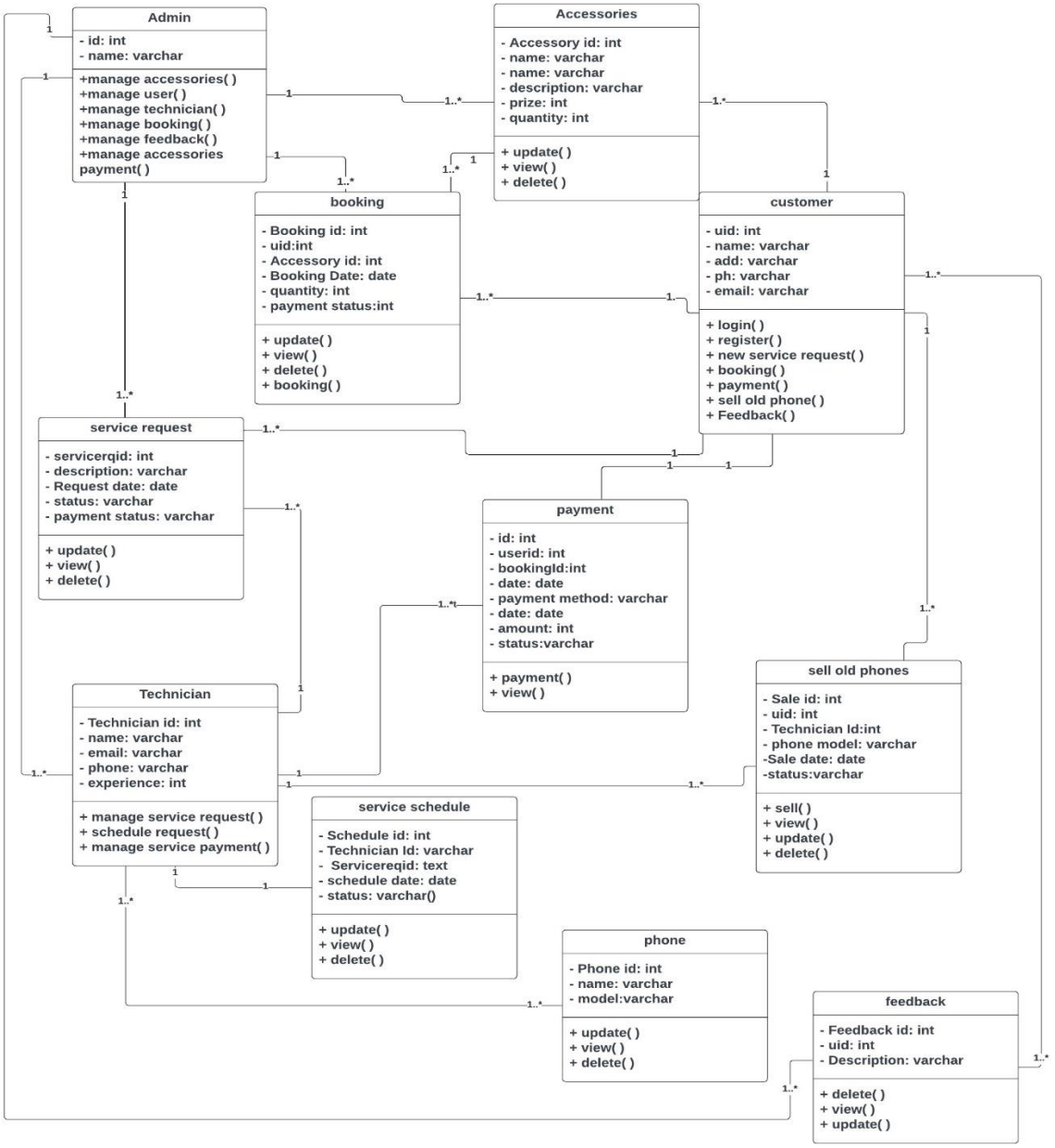


Fig 4: Class diagram

4.2.5 OBJECT DIAGRAM

An object diagram is a type of UML diagram that represents a snapshot of the state of a system at a particular point in time. It shows the objects in the system and their relationships, including the attributes and values of the objects. Object diagrams are useful for understanding the behavior of a system and for testing and debugging software applications.

In an object diagram, objects are represented as boxes, with their attributes and values listed inside. Relationships between objects are depicted using lines and arrows, which show the associations, dependencies, and inheritance between them. Object diagrams can also include multiplicity, which

shows how many instances of an object can be associated with another object.

Object diagrams are useful for several reasons. Firstly, they provide a visual representation of the state of a system, making it easier for developers and stakeholders to understand and review the behavior of the system. They also help to identify potential problems and bugs in the system, such as objects with incorrect attribute values or missing relationships. Additionally, object diagrams can be used to test and debug software applications, ensuring that they behave correctly and meet the needs of users and stakeholders. Overall, object diagrams are an essential tool for designing and developing software systems that are effective, efficient, and reliable.

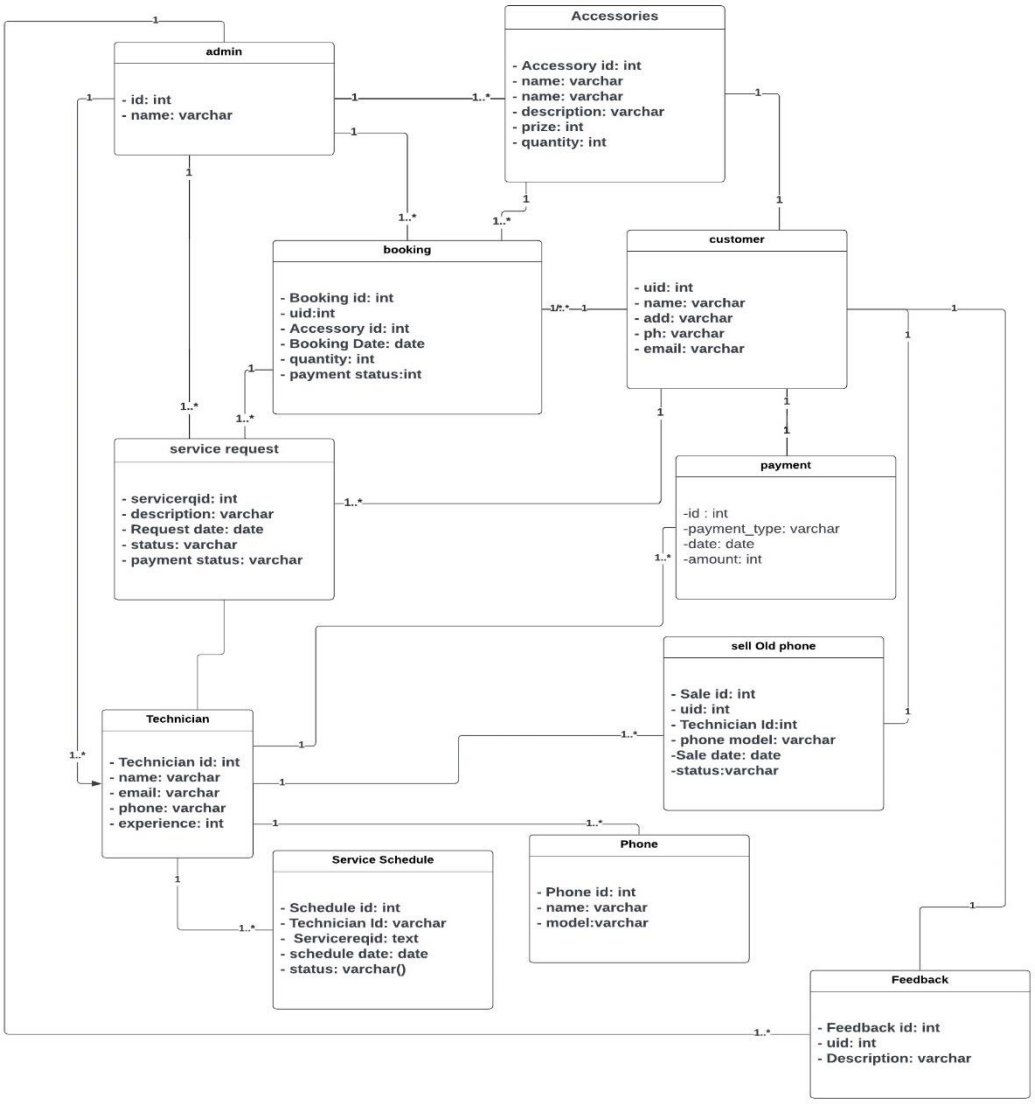


Fig 5: Object diagra

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Home page

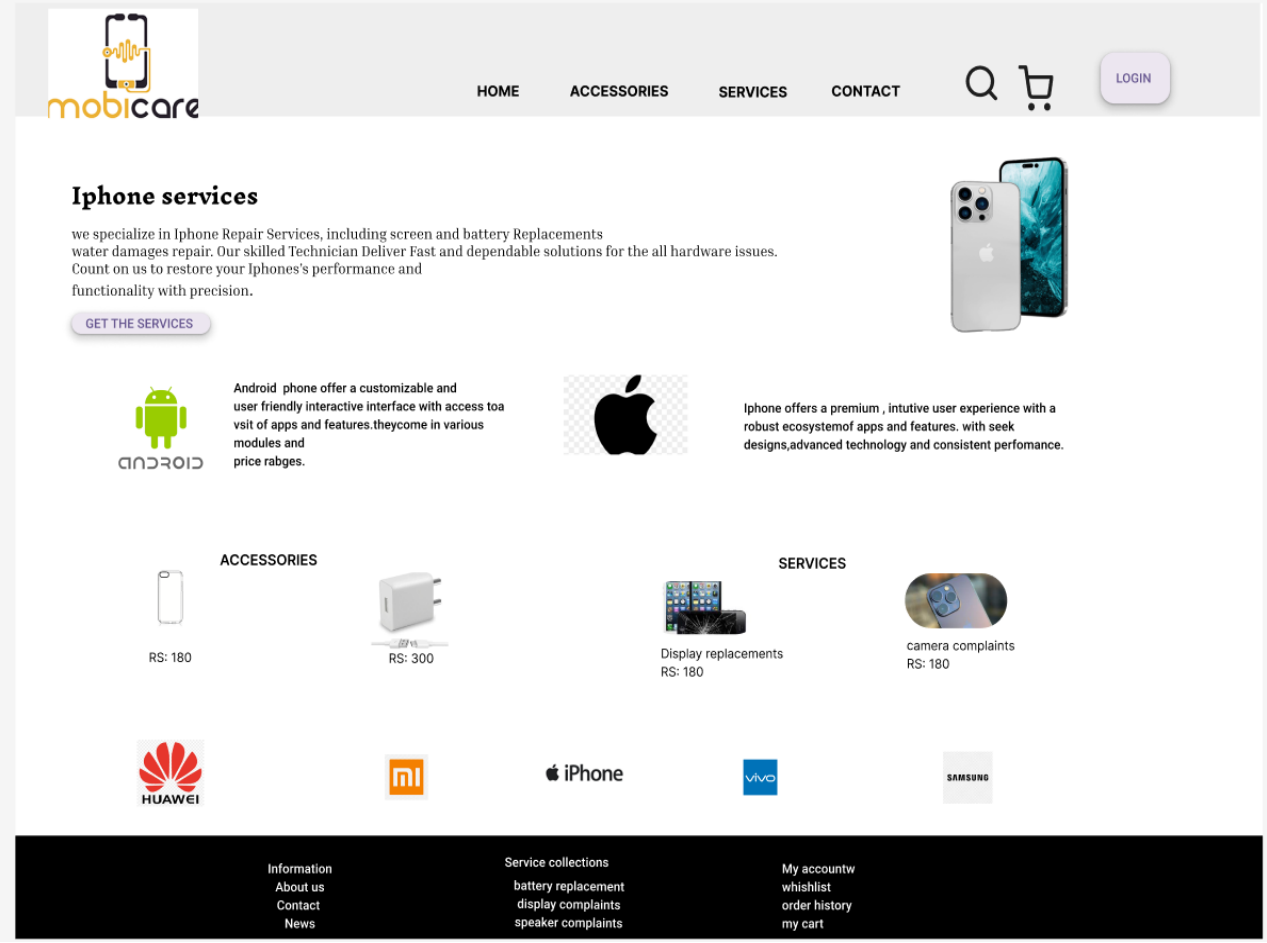


Fig 6: home page

Form Name: Accessories page

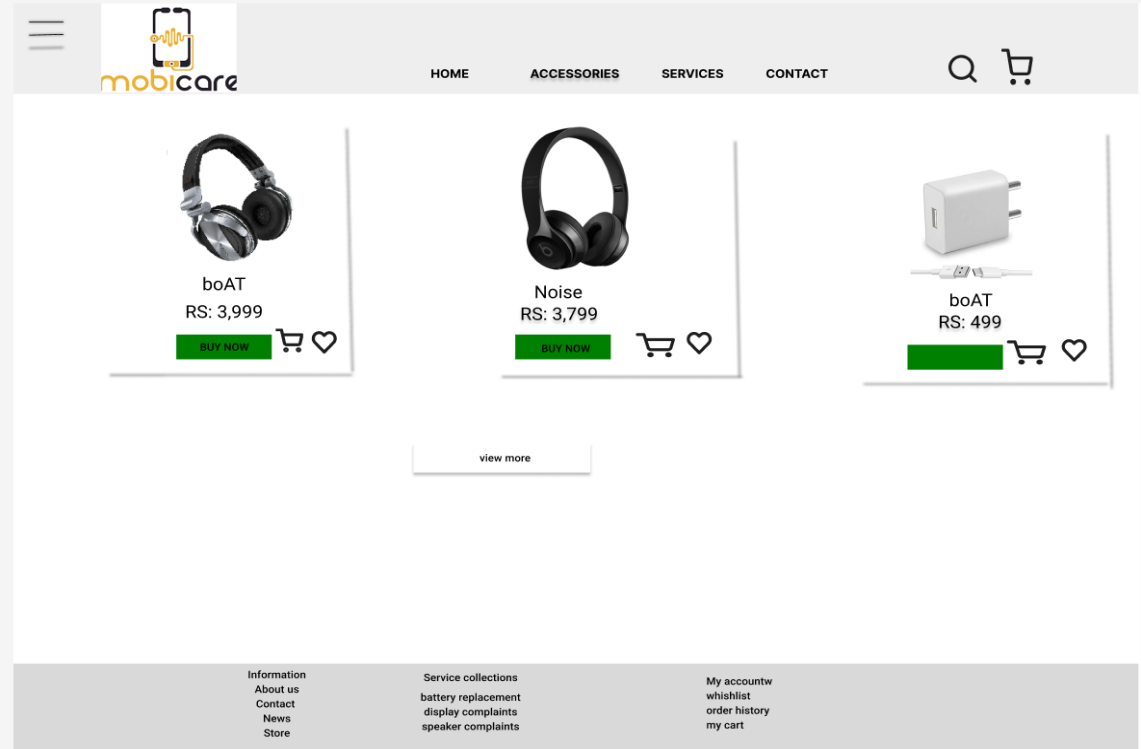


Fig 7: Accessories page

Form Name: Category page

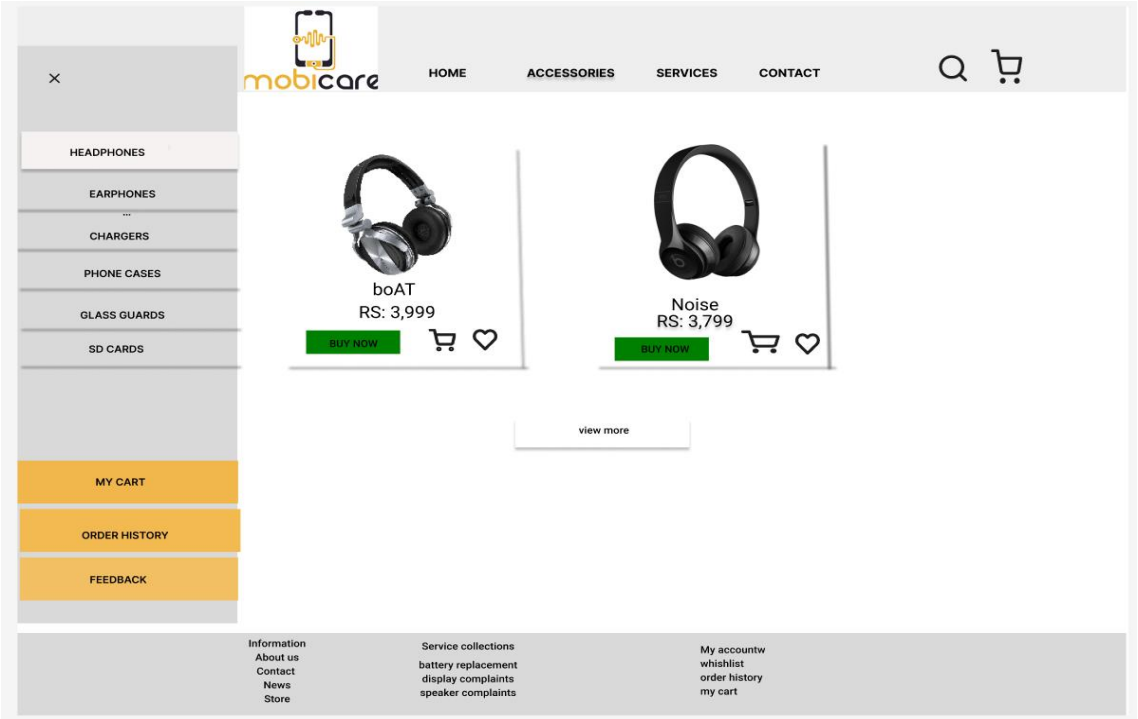


Fig 8: Category page

Form Name: Category Headphone page

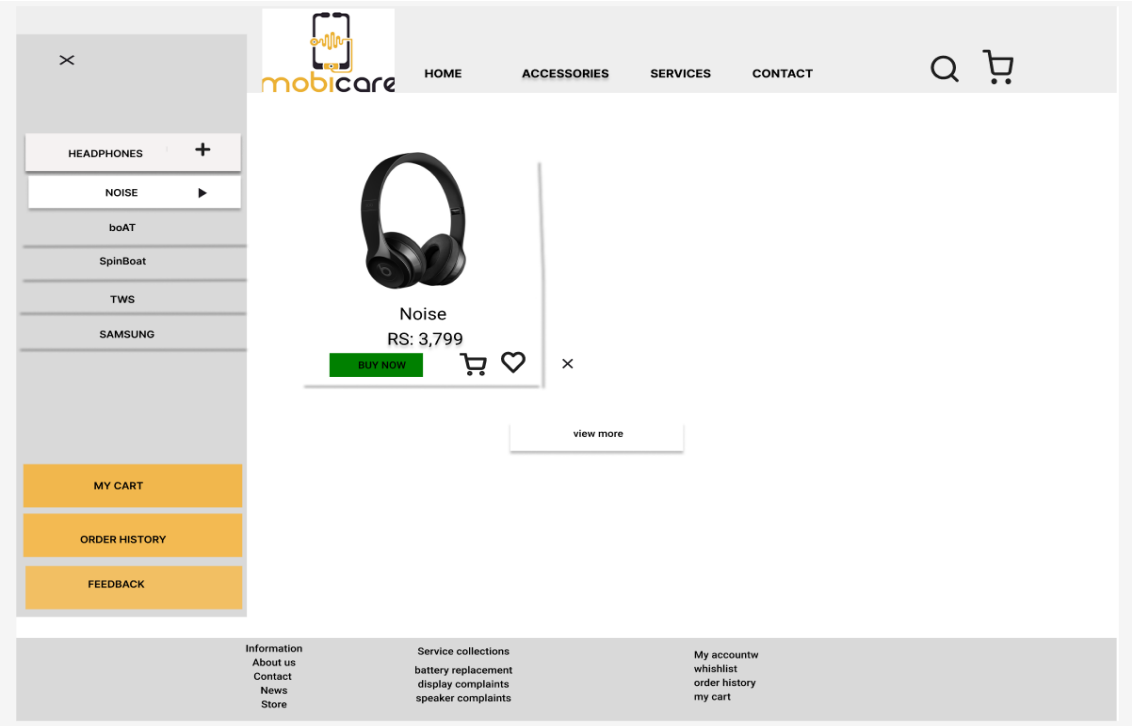


Fig 9: Category headphone page

Form Name: Service Request page

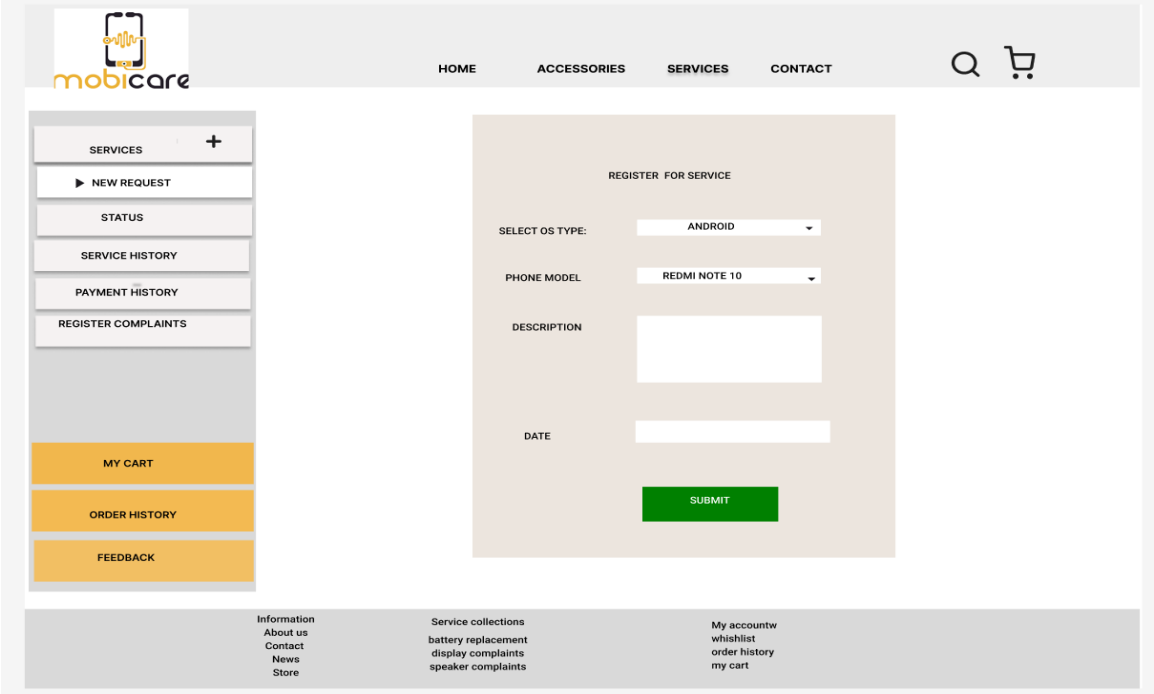


Fig 10: Service Request page

4.4 DATABASE DESIGN

Database design is the process of creating a database schema, which defines the structure of a database and its relationships between tables. It involves identifying the data that needs to be stored, organizing it into tables, and determining the relationships between tables. The goal of database design is to create a database that is efficient, reliable, and easy to use.

The first step in database design [50] is to identify the data that needs to be stored. This involves analyzing the requirements of the system and determining the types of data that will be used. Once the data has been identified, it is organized into tables, with each table representing a particular entity or concept. Tables are designed to minimize redundancy and ensure that data is stored in a consistent and organized manner.

Once the tables have been created, the relationships between them are defined. Relationships can be one-to-one, one-to-many, or many-to-many, depending on the nature of the data being stored. The relationships between tables are defined using foreign keys, which link data in one table to data in another table. The goal of database design is to create a schema that is efficient, flexible, and easy to use, while minimizing the potential for errors and inconsistencies in the data**.**

4.4.1 Relational Database Management System (RDBMS)

RDBMS stands for Relational Database Management System, which is a type of software that allows users to manage relational databases. A relational database is a type of database that stores data in tables, with each table consisting of rows and columns. RDBMS [52] software is designed to manage the organization, storage, retrieval, and security of data in a relational database.

RDBMS software provides several important features that make it an essential tool for managing large amounts of data. Firstly, it provides a standardized way to store data in tables, ensuring that data is consistent and organized. Secondly, it provides a set of tools and commands for users to manipulate and retrieve data from the database, making it easy to extract and analyze data. Thirdly, it provides a range of security features, such as user authentication and access control, to ensure that data is protected from unauthorized access.

RDBMS software is used in a wide range of applications, from small business databases to large enterprise systems. It is used to manage everything from customer data to financial records, and is an essential tool for businesses and organizations that rely on large amounts of data. RDBMS software is constantly evolving, with new features and functionality being added to make it more powerful, efficient, and user-friendly. Overall, RDBMS software is an essential tool for managing and organizing data in today's data-driven world.

4.4.2 Normalization

Normalization [51] is a process used in database design to eliminate redundant data and improve data integrity. It involves organizing data into tables, and then breaking down those tables into smaller, more manageable tables. The goal of normalization is to ensure that each table has a single purpose, and that data is stored in a way that eliminates duplicate information and inconsistencies.

Normalization is typically divided into several normal forms, each of which represents a higher level of normalization. The most common normal forms are first normal form (1NF), second normal form (2NF), and third normal form (3NF). Each of these normal forms has specific rules that must be followed to ensure that the data is properly organized and free from redundancy.

The benefits of normalization include improved data quality, increased efficiency, and easier maintenance. By eliminating redundant data, normalization helps to ensure that data is consistent and accurate, and that updates and changes can be made quickly and easily. Additionally, normalization helps to reduce the amount of storage required for a database, making it more efficient and cost-effective.

Normalization is an important process in database design, and is essential for creating databases that are reliable, efficient, and easy to use. It requires a thorough understanding of the data being stored, as well as the rules and principles of normalization. By following these principles, database designers can create databases that are well-organized, flexible, and scalable, and that can be easily maintained over time.

First Normal Form

First normal form (1NF) is a basic level of normalization in database design. It requires that all data in a table be organized into columns, and that each column have a unique name. Additionally, each column must contain atomic (indivisible) values, which means that there should be no repeating groups or arrays of data within a single column.

To meet the requirements of 1NF, a table must have a primary key, which is a unique identifier that is used to distinguish between records in the table. Each record in the table should have a value for every column, even if that value is NULL. In addition, there should be no repeating groups of data within a single row, meaning that each column should contain a single, indivisible value.

Example: Relation EMPLOYEE is not in 1NF because of multi-valued attribute EMP_PHONE.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385, 9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389, 8589830302	Punjab

Fig 11: example table

The decomposition of the EMPLOYEE table into 1NF has been shown below:

EMP_ID	EMP_NAME	EMP_PHONE	EMP_STATE
14	John	7272826385	UP
14	John	9064738238	UP
20	Harry	8574783832	Bihar
12	Sam	7390372389	Punjab
12	Sam	8589830302	Punjab

Fig 12: example table

Second Normal Form

Second normal form (2NF) is a higher level of normalization in database design that builds upon the requirements of first normal form (1NF). In addition to meeting the requirements of 1NF, a table must also eliminate partial dependencies, which means that every non-key column in the table must depend on the entire primary key, and not just a portion of it.

To achieve 2NF, a table must have a primary key that consists of at least two columns. Each non-key column in the table must depend on the entire primary key, and not just a portion of it. If a non-key column depends on only a portion of the primary key, it must be moved to a separate table that is related to the original table by a foreign key.

Example: Let's assume, a school can store the data of teachers and the subjects they teach. In a school, a teacher can teach more than one subject.

TEACHER table

TEACHER_ID	SUBJECT	TEACHER_AGE
25	Chemistry	30
25	Biology	30
47	English	35
83	Math	38
83	Computer	38

Fig 13: example table

In the given table, non-prime attribute TEACHER_AGE is dependent on TEACHER_ID which is a proper subset of a candidate key. That's why it violates the rule for 2NF.
To convert the given table into 2NF, we decompose it into two tables:

TEACHER_DETAIL table:

TEACHER_ID	TEACHER_AGE
25	30
47	35
83	38

Fig 14: example table

TEACHER_SUBJECT table:

TEACHER_ID	SUBJECT
25	Chemistry
25	Biology

47	English
83	Math
83	Computer

Fig 15: example table

Third Normal Form

Third normal form (3NF) is a higher level of normalization in database design that builds upon the requirements of second normal form (2NF). In addition to meeting the requirements of 2NF, a table must eliminate transitive dependencies, which means that non-key columns must not depend on other non-key columns in the same table.

To achieve 3NF, a table must have a primary key that consists of at least one column. Each non-key column in the table must depend only on the primary key, and not on other non-key columns in the same table. If a non-key column depends on another non-key column, it must be moved to a separate table that is related to the original table by a foreign key.

Example:

EMPLOYEE_DETAIL table:

EMP_ID	EMP_NAME	EMP_ZIP	EMP_STATE	EMP_CITY
222	Harry	201010	UP	Noida
333	Stephan	02228	US	Boston
444	Lan	60007	US	Chicago
555	Katharine	06389	UK	Norwich
666	John	462007	MP	Bhopal

Fig 16: example table

Super key in the table above:

{EMP_ID}, {EMP_ID, EMP_NAME}, {EMP_ID, EMP_NAME, EMP_ZIP}....so on

Candidate key: {EMP_ID}

Non-prime attributes: In the given table, all attributes except EMP_ID are non-prime.

Here, EMP_STATE & EMP_CITY dependent on EMP_ZIP and EMP_ZIP dependent on EMP_ID.

The non-prime attributes (EMP_STATE, EMP_CITY) transitively dependent on super key(EMP_ID). It violates the rule of third normal form.

That's why we need to move the EMP_CITY and EMP_STATE to the new <EMPLOYEE_ZIP> table, with EMP_ZIP as a Primary key.

EMPLOYEE table:

EMP_ID	EMP_NAME	EMP_ZIP
222	Harry	201010
333	Stephan	02228
444	Lan	60007
555	Katharine	06389
666	John	462007

Fig 17: example table

EMPLOYEE_ZIP table:

EMP_ZIP	EMP_STATE	EMP_CITY
201010	UP	Noida
02228	US	Boston
60007	US	Chicago
06389	UK	Norwich
462007	MP	Bhopal

Fig 18: example table

Boyce-Codd Normal Form

Boyce-Codd Normal Form (BCNF) is a higher level of normalization in database design that builds upon the requirements of third normal form (3NF). In addition to meeting the requirements of 3NF, a table must eliminate all non-trivial dependencies, which means that all non-key columns must be

functionally dependent on the primary key, and not on any other non-key columns in the same table.

To achieve BCNF, a table must have a primary key that consists of at least one column. Each non-key column in the table must be functionally dependent on the primary key, and not on any other non-key columns in the same table. If a non-key column is functionally dependent on another non-key column, it must be moved to a separate table that is related to the original table by a foreign key.

Example: Let's assume there is a company where employees work in more than one department.

EMPLOYEE table:

EMP_ID	EMP_COUNTRY	EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO
264	India	Designing	D394	283
264	India	Testing	D394	300
364	UK	Stores	D283	232
364	UK	Developing	D283	549

Fig 19: example table

The table is not in BCNF because neither EMP_DEPT nor EMP_ID alone are keys.
To convert the given table into BCNF, we decompose it into three tables:

EMP_COUNTRY table:

EMP_ID	EMP_COUNTRY
264	India
264	India

Fig 20 : example table

EMP_DEPT table:

EMP_DEPT	DEPT_TYPE	EMP_DEPT_NO

Designing	D394	283
Testing	D394	300
Stores	D283	232
Developing	D283	549

Fig 21: example table

EMP_DEPT_MAPPING table:

EMP_ID	EMP_DEPT
D394	283
D394	300
D283	232
D283	549

Fig 22: example table

Fourth Normal Form

Fourth normal form (4NF) is a higher level of normalization in database design that builds upon the requirements of Boyce-Codd normal form (BCNF). In addition to meeting the requirements of BCNF, a table must eliminate multi-valued dependencies, which occur when a non-key column depends on a combination of columns that includes a subset of the primary key.

To achieve 4NF, a table must have a primary key that consists of at least one column. Each non-key column in the table must be functionally dependent on the primary key, and not on any other non-key columns in the same table. Additionally, there should be no multi-valued dependencies between non-key columns.Example:

STUDENT

STU_ID	COURSE	HOBBY
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

Fig 23: example table

The given STUDENT table is in 3NF, but the COURSE and HOBBY are two independent entity. Hence, there is no relationship between COURSE and HOBBY.There is a Multi-valued dependency on STU_ID, which leads to unnecessary repetition of data. So to make the above table into 4NF, we can decompose it into two tables:

STUDENT_COURSE

STU_ID	COURSE
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

Fig 24: example table

STUDENT_HOBBY

STU_ID	HOBBY
21	Dancing

21	Singing
34	Dancing
74	Cricket
59	Hockey

Fig 25: example table

Fifth Normal Form

Fifth normal form (5NF) is the highest level of normalization in database design. It is also known as "project-join normal form" (PJ/NF). 5NF addresses the issue of join dependencies, which arise when a table can be reconstructed by joining multiple other tables, each with its own primary key. To achieve 5NF, a table must meet the requirements of all lower normal forms, including eliminating all non-trivial dependencies, multi-valued dependencies, and other forms of redundancy. Additionally, a table must have a composite primary key that consists of at least two columns. Furthermore, each non-trivial join dependency must be represented by a separate table.

SUBJECT	LECTURER	SEMESTER
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

Fig 26: example table

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data. Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together

acts as a primary key, so we can't leave other two columns blank. So to make the above table into 5NF, we can decompose it into three relations P1, P2 & P3:

P1

SEMESTER	SUBJECT
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

Fig 27: example table

P2

SUBJECT	LECTURER
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

Fig 28: example table

P3

SEMSTER	LECTURER
Semester 1	Anshika
Semester 1	John
Semester 1	John

Semester 2	Akash
Semester 1	Praveen

Fig 29: example table

4.4.3 Sanitization

Sanitization refers to the process of cleaning and purifying data or content to remove any unwanted elements. In web development or design, sanitization is crucial to prevent security issues, such as code injection attacks, by ensuring that inputs from users do not contain malicious code. This may involve filtering HTML or JavaScript, escaping special characters, or validating inputs against predefined patterns. Sanitization helps maintain data integrity, prevents harmful actions, and ensures that data is safely handled within a system

4.4.4 Indexing

Indexing is a technique used to organize and access information efficiently across various fields, particularly databases, books, and web search engines. In databases, indexing significantly speeds up search queries by creating a structured pathway to data, enabling the system to locate information without scanning the entire table. This is especially useful for large databases, where queries on indexed columns—like primary keys—return results more quickly. Similarly, in books, an index is a list of key topics with their corresponding page numbers, making it easy for readers to find specific information. Web search engines also rely heavily on indexing, as they use algorithms to crawl and organize web content, allowing users to quickly retrieve relevant search results. Overall, indexing helps improve data retrieval speed, making it essential for efficient information management in various applications.

4.5 TABLE DESIGN

1. CustomUser

Primary key: id (inherited from AbstractUser)

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of CustomUser
2	username	Varchar(150)	Unique	Username of the user
3	email	EmailField	Unique	Email of the user
4	phone	Varchar(15)		Phone number of the user

5	Address	Text		Address of the user
6	pincode	Varchar(6)		Pincode of the user's address
7	role	Char(20)		User role (Technician, User, Admin)
8	Is_approved	Boolean	Default: False	Indicates if the user is approved
9	Status	Char(10)	Default: 'active'	Status of the user (Active, Inactive)
10	qualification	FileField	Nullable	User's qualification document

2. Category

Primary key: id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Category
2	name	Char(100)		Name of the category
3	image	ImageField		Image of the category
4	status	Char(10)		Status of the category

3. SubCategory

Primary key: id

Foreign keys: category references Category

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of SubCategory
2	category	Int(11)	Foreign key referencing Category	Reference to the associated category
3	brand	Char(100)		Brand of the subcategory
4	status	Char(10)	Default: 'active'	Status of the subcategory

4. Product

Primary key: id

Foreign keys: subcategory references SubCategory

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Product
2	name	Char(100)		Name of the product
3	description	Text		Description of the product
4	price	Decimal(10, 2)		Price of the product
5	image	ImageField		Image of the product
6	quantity	PositiveInteger		Quantity available for the product
7	subcategory	Int(11)	Foreign key referencing SubCategory	Reference to the associated subcategory
8	status	Char(10)	Default: 'active'	Status of the product

5. Cart

Primary key: id

Foreign keys: user references AUTH_USER_MODEL, product references Product

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of cart
2	user	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the user
3	product	Int(11)	Foreign key referencing Product	Reference to the product
4	quantity	PositiveInteger		Quantity of the product in the cart
5	added_at	DateTime		Timestamp when the product was added

6. Payment

Primary key: id

Foreign keys: cart references Cart

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Payment
2	cart	Int(11)	Foreign key referencing Cart	Reference to the associated cart
3	amount	Decimal(10, 2)		Amount of the payment
4	status	Char(20)	Default: 'Pending'	Status of the payment
5	created_at	DateTime		Timestamp when the payment was created

7. PhoneCategory

Primary key: id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of PhoneCategory
2	name	Char(100)		Name of the phone category
3	image	varchar(20)		Image of the phone category
4	status	Char(10)	Default: 'active'	Status of the phone category

8. PhoneSubCategory

Primary key: id

Foreign keys: category references PhoneCategory

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of PhoneSubCategory

2	category	Int(11)	Foreign key referencing PhoneCategory	Reference to the associated phone category
3	brand	Char(100)		Brand of the phone subcategory
4	status	Char(10)	Default: 'active'	Status of the phone subcategory
5	image	ImageField	Nullable	Image associated with the subcategory

9. PhoneModel

Primary key: id

Foreign keys: subcategory references PhoneSubCategory

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of PhoneModel
2	subcategory	Int(11))	Foreign key	Reference to the associated subcategory
3	model_name	Char(100)		Name of the phone model
4	status	Char(10)	Default: 'active'	Status of the phone model

10. Complaint

Primary key: id

Foreign keys: phone_model references PhoneModel

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Complaint
2	phone_model	Int(11)	Foreign key referencing PhoneModel	Reference to the associated phone model
3	complaint_title	Char(100)		Title of the complaint
4	description	Text		Description of the complaint
5	status	Char(10)	Default: 'active'	Status of the complaint

6	Expected_rate	Decimal(10, 2)	Default: 0.00	Expected rate for the complaint

10. ServiceRequest

Primary Key: id

Foreign Keys:

PhoneCategory references PhoneCategory, PhoneSubCategory references PhoneSubCategory, PhoneModel references PhoneModel, Complaint references Complaint

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of ServiceRequest
2	phone_category	Int(11)	Foreign key referencing PhoneCategory	Reference to the associated phone category
3	phone_subcategory	Int(11)	Foreign key referencing PhoneSubCategory	Reference to the associated phone subcategory
4	phone_model	Int(11)	Foreign key referencing PhoneModel	Reference to the associated phone model
5	phone_complaint	Int(11)	Foreign key referencing Complaint	Reference to the associated complaint
6	expected_rate	Decimal(10, 2)	Nullable, Blankable	Expected rate for the service request
7	pickup_date	Date		Date of pickup
8	phone_number	Char(15)		User's phone number
9	issue_description	Text		Description of the issue
10	pickup_address	Text		Pickup address
11	terms_accepted	Boolean	Default: false	Terms and conditions accepted
12	status	Char(15)	Default: pending	Status of the service request
13	amount	Decimal(10, 2)	Default: 0.00	Amount for the service request
14	delivery_date	Date	Nullable, Blankable	Date of delivery

11. TermsAndConditionsPrimary key: id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of TermsAndCondition

2	content	Text		Content of Terms and Condition

12. Payments

Primary key: id

Foreign Key:

user references Auth_user_model,

service_request references ServiceRequest

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Payments
2	content	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the user
3	service_request	Int(11)	Foreign key referencing ServiceRequest	Reference to the service request
4	amount	Decimal(10, 2)		Amount paid
5	razorpay_order_id	Char(100)		Razorpay order ID
6	razorpay_payment_id	Char(100)	Nullable, Blankable	Razorpay payment ID
7	status	Char(20)		Payment status (Success/Completed)
8	created_at	DateTime	Default: auto_now_add	Date and time when payment was completed
9	updated_at	DateTime	Default: auto_now	Last update Timestamp

13. Notification

Primary key: id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Notification

2	user	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the associated user
3	message	text		Notification message
4	created_at	Boolean	Default: auto_now_add	Timestamp of when the notification was created
5	read	Boolean	Default: False	Whether the notification has been read or not

14. Notification

Primary Key: id

Foreign Key: user references AUTH_USER_MODE

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Notification
2	User	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the associated user
3	message	Text		Notification message
4	created_at	DateTime	Default: auto_now_add	Timestamp of when the notification was created
5	read	Boolean	Default: False	Whether the notification has been read or not

15. CalendarEvent

Primary Key: id

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of CalendarEvent
2	title	Char(255)	Foreign key referencing AUTH_USER_MODEL	Title of the calendar event
3	Event_date	Date		Date of the calendar event

16. Wishlist

Primary Key: id

Foreign Key: user references AUTH_USER_MODEL,

product references Product

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Wishlist
2	title	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the associated user
3	product	Int(11)	Foreign key referencing Product	Reference to the associated product

17. Feedback

Primary Key: id

Foreign Key: user references AUTH_USER_MODEL

No:	Fieldname	Datatype (Size)	Key Constraints	Description of the Field
1	id	Int(11)	Primary key	Primary key of Feedback
2	user	Int(11)	Foreign key referencing AUTH_USER_MODEL	Reference to the user who submitted the feedback
3	message	Text		Feedback message
4	emoji	Char(10)		Emoji reaction
5	created_at	DateTime	Default: auto_now_add	Timestamp of when feedback was created
6	status	Char(10)	Default: active	Status of feedback (active/inactive)

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

System testing is a type of software testing that involves testing the entire system or application as a whole. It is typically performed after the completion of integration testing and before user acceptance testing. The main goal of system testing is to verify that the application meets the specified requirements and performs as expected in a real-world environment.

System testing can be either functional or non-functional testing. Functional testing involves verifying that the application functions correctly in terms of meeting the requirements, while non-functional testing involves verifying the performance, security, reliability, usability, and compatibility of the application.

Some of the common techniques used in system testing include black box testing, white box testing, manual testing, automated testing, and regression testing. The testing process usually involves creating test cases, executing them, and reporting any defects found to the development team for fixing.

5.2 TEST PLAN

A test plan is a document that outlines the approach, scope, objectives, and resources required for testing a software application or system. It provides a roadmap for the testing process, including what needs to be tested, how it will be tested, and who will be responsible for carrying out the testing.

A typical test plan will include the following information:

Introduction: This section provides an overview of the software application or system being tested and explains the purpose of the test plan.

Test objectives: This section outlines the specific objectives that the testing process aims to achieve, including what needs to be tested and why.

Test scope: This section defines the boundaries of the testing process, including the features and functionalities that will be included in the testing.

Test strategy: This section describes the overall approach to testing, including the testing techniques, tools, and methodologies that will be used.

Test environment: This section outlines the hardware, software, and other resources required for the testing process.

Test schedule: This section includes a timeline for the testing process, including the start and end dates, milestones, and deadlines.

Test deliverables: This section lists the expected deliverables from the testing process, such as test reports, defect reports, and documentation.

Test team: This section includes the roles and responsibilities of the individuals involved in the testing process, including the testers, developers, and project managers.

Test risks and mitigation: This section identifies potential risks that may impact the testing process and outlines strategies for mitigating those risks.

Overall, a test plan is a crucial document for ensuring that the testing process is carried out effectively and efficiently, and that the software application or system is thoroughly tested before release.

5.2.1 Unit Testing

Unit testing is a type of software testing in which individual units or components of a software application are tested in isolation from the rest of the application. The purpose of unit testing [54] is to ensure that each unit of code or function is working as expected and meets the specified requirements.

A unit in software development is the smallest testable part of an application. It can be a function, method, or class. Unit testing involves writing automated test cases that target individual units of code and verifying that they produce the expected output given a set of inputs.

Unit testing is usually performed by developers as part of the development process, and it is typically done using a testing framework that supports writing automated tests. Some popular unit testing frameworks include JUnit, NUnit, and PHPUnit.

Some of the benefits of unit testing include:

1. Early defect detection: Unit testing can help catch defects early in the development process, which can save time and resources in the long run.
2. Improved code quality: Unit testing encourages developers to write cleaner, more modular, and more maintainable code.
3. Faster development: Unit testing can speed up the development process by catching defects early and reducing the need for manual testing.
4. Easier maintenance: Unit tests serve as documentation for the code, making it easier for developers to understand and maintain the code over time

5.2.2 Integration Testing

Integration testing is a type of software testing in which individual modules or components of a software application are combined and tested as a group. The purpose of integration testing is to verify that the individual modules work correctly when integrated with each other, and that the application as a whole meets the specified requirements.

Integration testing is typically performed after unit testing and before system testing. It involves testing the interfaces between the modules and ensuring that the data flows correctly between them. Integration testing can be done in a top-down or bottom-up approach.

In a top-down integration testing approach, the high-level modules are tested first, and the lower-level modules are tested later. In a bottom-up integration testing approach, the lower-level modules are tested first, and the higher-level modules are tested later. A third approach, called a hybrid integration testing approach, combines both top-down and bottom-up approaches.

5.2.3 Validation Testing or System Testing

Validation testing is a type of software testing that is used to evaluate whether a software application or system meets the specified business requirements and user needs. The purpose of validation testing is to ensure that the application or system being developed is fit for its intended purpose and satisfies the expectations of the stakeholders.

Validation testing is typically performed after the completion of the testing process, which includes unit testing, integration testing, and system testing. It involves verifying that the application or system meets the acceptance criteria, which are usually defined in a requirements document or a user story.

The main focus of validation testing is on the functionality and usability of the application or system. It includes testing the user interface, user interactions, user flows, and the overall user experience. It can also involve testing the performance, security, and compatibility of the application or system.

Overall, validation testing is a critical step in the software development life cycle (SDLC) that helps ensure that the application or system meets the needs of the users and stakeholders. It is important to conduct thorough validation testing to minimize the risk of defects and ensure the overall quality and success of the application or system.

5.2.4 Output Testing or User Acceptance Testing

Output testing is a type of software testing that is used to verify the correctness and accuracy of the output produced by a software application or system. The purpose of output testing is to ensure that the output produced by the application or system meets the specified requirements and expectations.

Output testing can be performed at various stages of the software development life cycle (SDLC), including unit testing, integration testing, system testing, and acceptance testing. It involves

comparing the actual output produced by the application or system with the expected output.

The expected output is usually defined in a requirements document or a user story. The actual output is generated by running the application or system with a set of test data or inputs. The output can be in the form of text, images, audio, or any other type of data.

Overall, output testing is an essential part of software testing that helps ensure the quality and reliability of the application or system being developed. It is important to conduct thorough output testing to minimize the risk of defects and ensure that the application or system meets the needs of the users and stakeholders.

5.2.5 Automation Testing

Automation testing is a software testing technique that involves the use of specialized software tools and scripts to perform tests automatically. It is an efficient way of executing test cases and helps in identifying defects, errors, and performance issues in software applications or systems. Automation testing can be used for various types of testing such as unit testing, integration testing, system testing, and acceptance testing.

In automation testing, the test cases are automated using scripts, which are written in a programming language like Java, Python, or C#. These scripts can simulate the actions of a human user, including keystrokes, mouse clicks, and data entry. Automation testing tools provide a graphical user interface (GUI) that makes it easy for testers to create, execute, and analyze test scripts.

One of the major benefits of automation testing is that it can save time and effort in the testing process. Automation testing can execute thousands of test cases in a fraction of the time it would take a human tester to execute manually. Additionally, automation testing can run tests on different operating systems, browsers, and devices simultaneously, which would be impossible to achieve manually.

5.2.6 Selenium Testing

Selenium testing is a popular open-source tool used for automated testing of web applications. It provides a suite of tools that can be used for different testing purposes, including functional testing, regression testing, and performance testing.

Selenium testing [53] works by automating browser interactions. It can simulate user actions such as clicking buttons, filling out forms, and navigating through pages, to verify that the web application works as expected. Selenium supports a variety of programming languages such as Java, Python, and C#, which makes it easy for developers and testers to integrate it into their

development environment.

One of the major advantages of Selenium testing is its cross-platform compatibility. It can run tests on different operating systems such as Windows, Mac, and Linux, as well as on different browsers such as Chrome, Firefox, and Safari. This makes it a versatile tool for testing web applications across different environments.

In conclusion, Selenium testing is a powerful tool for automated testing of web applications. Its cross-platform compatibility, flexibility, and range of tools make it a popular choice for testers and developers. It can help save time and effort in the testing process and improve the quality of web applications.

Test Case 1

Code

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.edge.service import Service
from webdriver_manager.microsoft import EdgeChromiumDriverManager
import time
driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()))
users = [
    {"email": "mobicare123@gmail.com", "password": "Int@1234", "expected_url":
"http://127.0.0.1:8000/admin_view/"},
    {"email": "aswathymj2002@gmail.com", "password": "Int@1234", "expected_url":
"http://127.0.0.1:8000/user_view/"},
    {"email": "abhikanth2001@gmail.com", "password": "Int@1234", "expected_url":
"http://127.0.0.1:8000/technician_view/"}
]
def test_login(user):
    driver.get("http://127.0.0.1:8000/login/")  # Your login URL
    email_field = driver.find_element("name", "semail")
    email_field.send_keys(user['email'])
    password_field = driver.find_element("name", "spassword")
    password_field.send_keys(user['password'])
    password_field.send_keys(Keys.RETURN)
    time.sleep(3)
```

```
    if driver.current_url == user['expected_url']:
        print(f"Login test passed for {user['email']}")
    else:
        print(f"Login test failed for {user['email']} - Redirected to {driver.current_url}
instead of {user['expected_url']}")
for user in users:
    test_login(user)
driver.quit()
```

Screenshot

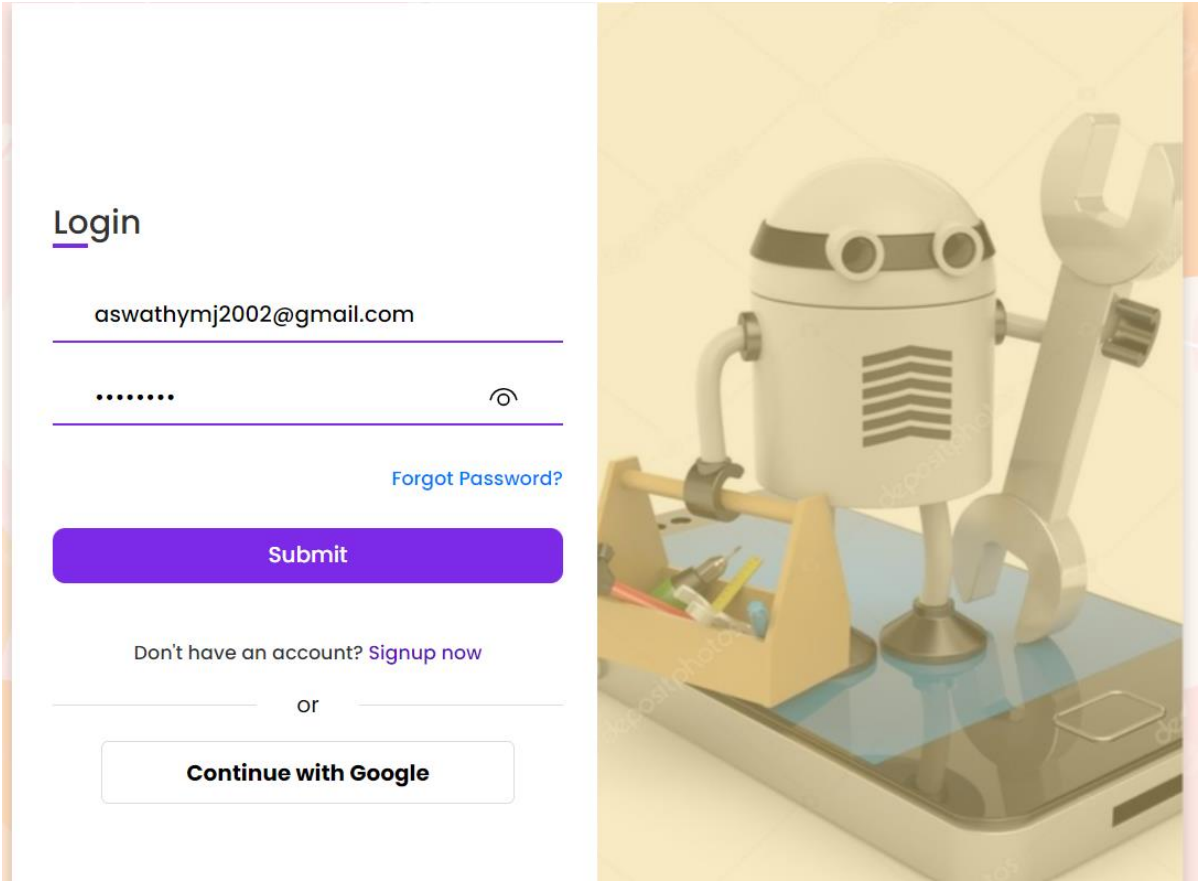


Fig 30: User login page

```
PS D:\Mobicare\Aswathy_Mobicare\05-09-24\updating\s9Project\myproject> python myapp/test/test_login_form.py

DevTools listening on ws://127.0.0.1:58676/devtools/browser/72e94961-d27d-42bf-b9aa-acfec3d47abd
Login test passed for mobicare123@gmail.com
[8032:15004:1018/160015.417:ERROR:ssl_client_socket_impl.cc(1048)] handshake failed; returned -1, SSL error code 1, net_error -101
[8032:15004:1018/160015.435:ERROR:ssl_client_socket_impl.cc(1048)] handshake failed; returned -1, SSL error code 1, net_error -101
Login test passed for aswathymj2002@gmail.com
Login test passed for abhikanth2001@gmail.com
PS D:\Mobicare\Aswathy_Mobicare\05-09-24\updating\s9Project\myproject>
```

Fig 31: Terminal

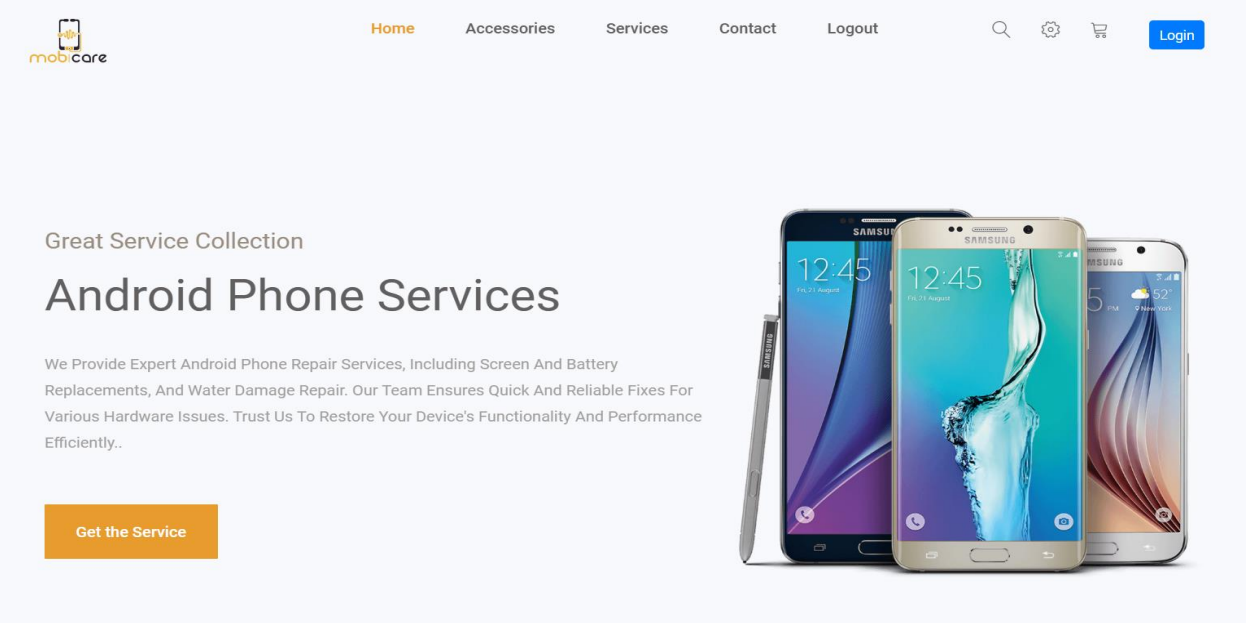


Fig 32: User Dashboard page

Test Report

Test Case 1					
Project Name: MobiCare					
Login Test Case					
Test Case ID: Test_1			**Test Designed By: Aswathy M J**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 16-10-2024**		
Module Name: Login Page			**Test Executed By: Binumon Joseph**		
Test Title: Verify Login with email and password			**Test Execution Date: 16-10-2024**		
Description: Login page Testing					
Pre-Condition: User has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		Display Login page	Login page displayed	pass
2	Valid username entered	Username: aswathymj2002@gmail.com	User should be able to login	User logged in and navigate to the dashboard	pass
3	Valid password entered	Password: Int@1234			
4	Click on login button				
Post-Condition: email and password is checked with database values for successful login.					

Test Case 2:

Code

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.edge.service import Service
from webdriver_manager.microsoft import EdgeChromiumDriverManager
import time
Set up the Edge WebDriver using webdriver-manager
driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()))
user = {"email": "aswathymj2002@gmail.com", "password": "Int@1234",
"expected_url": "http://127.0.0.1:8000/user_view/"}
def test_login_and_navigate():
    driver.get("http://127.0.0.1:8000/login/")  # Your login URL
    email_field = driver.find_element("name", "semail")
    email_field.send_keys(user['email'])
    password_field = driver.find_element("name", "spassword")
    password_field.send_keys(user['password'])
    password_field.send_keys(Keys.RETURN)
    time.sleep(3)
    if driver.current_url == user['expected_url']:
        print(f"Login test passed for {user['email']}")
        try:
            accessories_link = driver.find_element("link text", "Accessories")  # Adjust
the locator as necessary
            accessories_link.click()
            time.sleep(2)
            if driver.current_url == "http://127.0.0.1:8000/accessories/":
                print(f"Navigation to Accessories page passed for {user['email']}")
            else:
                print(f"Failed to navigate to Accessories page for {user['email']} -
Redirected to {driver.current_url} instead.")
        except Exception as e:
```

```
        print(f"Accessories link not found or failed to click: {e}")
    else:
        print(f"Login test failed for {user['email']} - Redirected to {driver.current_url}
instead of {user['expected_url']}")
test_login_and_navigate()
driver.quit()
```

Screenshot

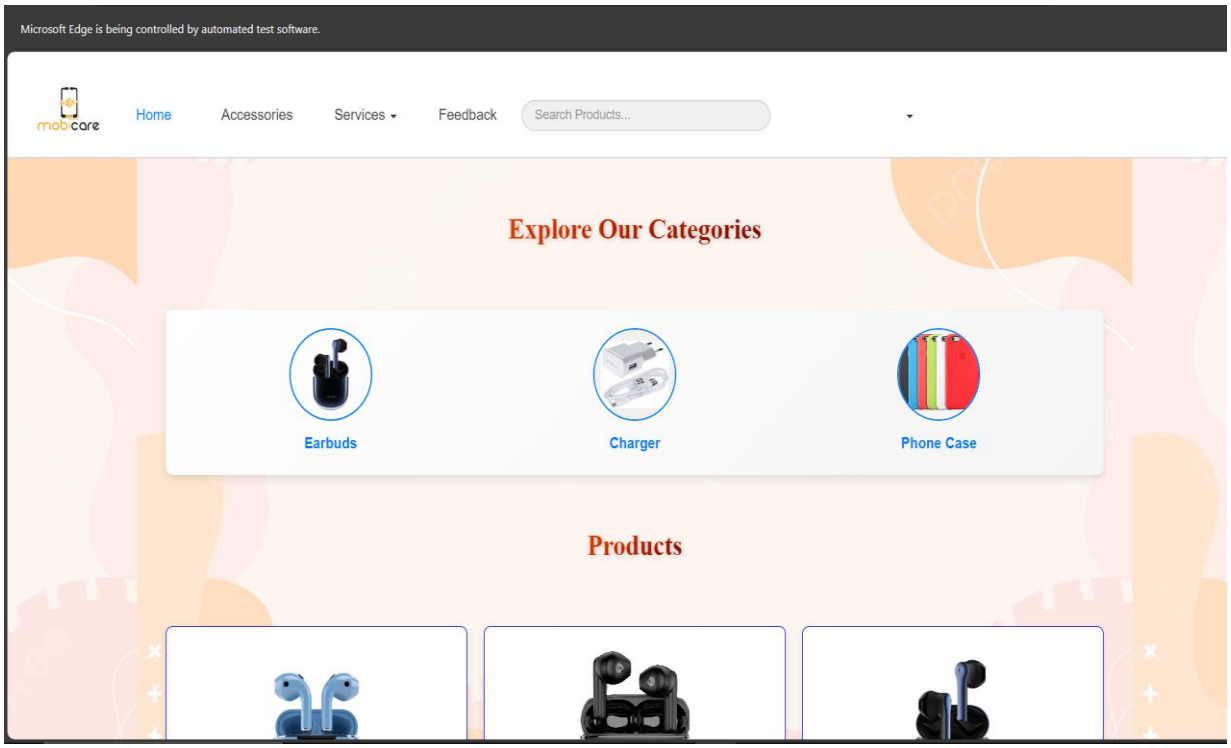


Fig 33: Accessories page

Fig 34: Terminal

Test report

Test Case 2

Project Name: MobiCare					
Accessories Test Case					
Test Case ID: Test_2			**Test Designed By: Aswathy M J**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 16-10-2024**		
Module Name: Accessories Page			**Test Executed By: Binumon Joseph**		
Test Title: Verify Login with email and password			**Test Execution Date: 16-10-2024**		
Description: Accessories page Testing					
Pre-Condition: User has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigate to login page		Display login page	Login page displayed	pass
2	Valid email entered	Aswathy2002 @gmail.com	User should be able to login and then navigate to home page and selected accessories tab	User should be able to login and then navigate to home page and selected accessories tab	pass
3	Valid password entered	Int@1234			
4	Click on login button				
Post-Condition: email and password is checked with database values for successful login.					

Test Case 3

Code

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.edge.service import Service
from webdriver_manager.microsoft import EdgeChromiumDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
```

```
from selenium.webdriver.common.by import By
import time
driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()))
login_url = "http://127.0.0.1:8000/login/"
user_view_url = "http://127.0.0.1:8000/user_view/"
accessories_url = "http://127.0.0.1:8000/accessories/"
status_url = "http://127.0.0.1:8000/repair_status/"
email = "aswathymj2002@gmail.com"
password = "Int@1234"
try:
    driver.get(login_url)
    email_field = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.NAME, "semail"))  # Adjust name if
necessary
    )
    email_field.send_keys(email)
    password_field = driver.find_element(By.NAME, "spassword")  # Adjust name if
necessary
    password_field.send_keys(password)
    password_field.send_keys(Keys.RETURN)
    WebDriverWait(driver, 10).until(
        EC.url_to_be(user_view_url)
    )
    driver.get(accessories_url)
    services_dropdown = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.LINK_TEXT, "Services"))
    )
    services_dropdown.click()
    status_link = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.LINK_TEXT, "Status"))
    status_link.click()
    WebDriverWait(driver, 10).until(
        EC.url_to_be(status_url)
    )
```

```
    print("Successfully navigated to the status page.")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    time.sleep(5)
    driver.quit()
```

Screenshot

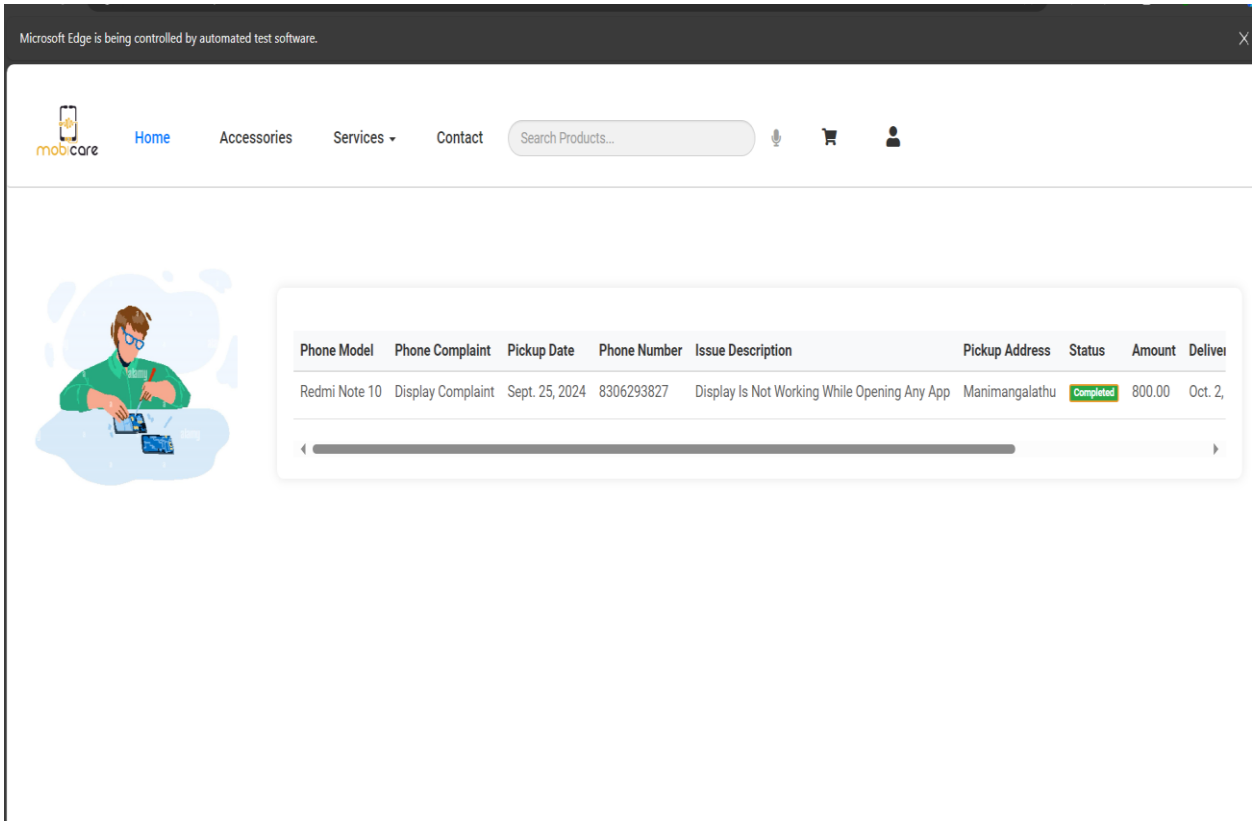


Fig 35: change password

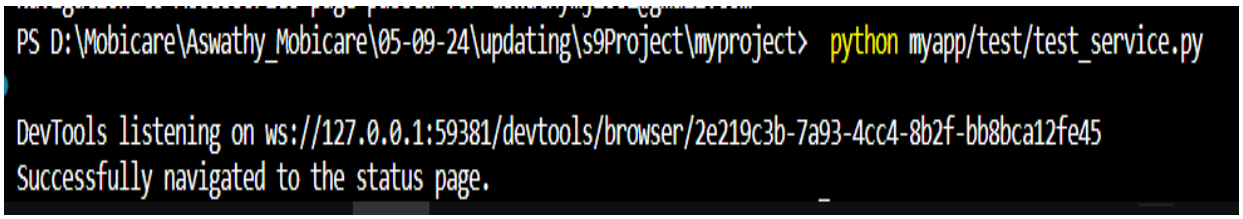


Fig 36: Terminal

Test Report

Test Case 3

Project Name: MobiCare					
Repair Status Test Case					
Test Case ID: Test_3			**Test Designed By: Aswathy M J**		
Test Priority (Low/Medium/High): High			**Test Designed Date: 16-10-2024**		
Module Name: Repair Status			**Test Executed By: Binumon Joseph**		
Test Title: Navigate to the repair status page after successful Login			**Test Execution Date: 16-10-2024**		
Description: Repair Status Testing					
Pre-Condition: User has valid email and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page		Display login page	login page displayed	pass
2	Navigate to the home page		User should be able to view the service status page	User view the service status page	pass
3	Click on the service				
4	Click on the repair status				
Post-Condition: user login with valid email and password then navigate to the repair status page					

Test Case 4:

Code

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.edge.service import Service
from webdriver_manager.microsoft import EdgeChromiumDriverManager
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.by import By
import time
```

```
driver = webdriver.Edge(service=Service(EdgeChromiumDriverManager().install()))
login_url = "http://127.0.0.1:8000/login/"
admin_view_url = "http://127.0.0.1:8000/admin_view/"
email = "mobicare123@gmail.com"
password = "Int@1234"
try:
    driver.get(login_url)
    email_field = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.NAME, "semail"))
    )
    email_field.send_keys(email)
    password_field = driver.find_element(By.NAME, "spassword")
    password_field.send_keys(password)
    password_field.send_keys(Keys.RETURN)
    WebDriverWait(driver, 10).until(
        EC.url_to_be(admin_view_url)
    )
    manage_technicians_link = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.LINK_TEXT, "Manage Technicians"))
    )
    manage_technicians_link.click()
    time.sleep(3)
    download_button = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.XPATH, "//td/a[contains(@class, 'btn btn-
primary btn-sm')]"))
    )
    download_button.click()
    print("Successfully clicked the Download PDF button.")
except Exception as e:
    print(f"An error occurred: {e}")
finally:
    time.sleep(5)
    driver.quit()
```

Screenshot

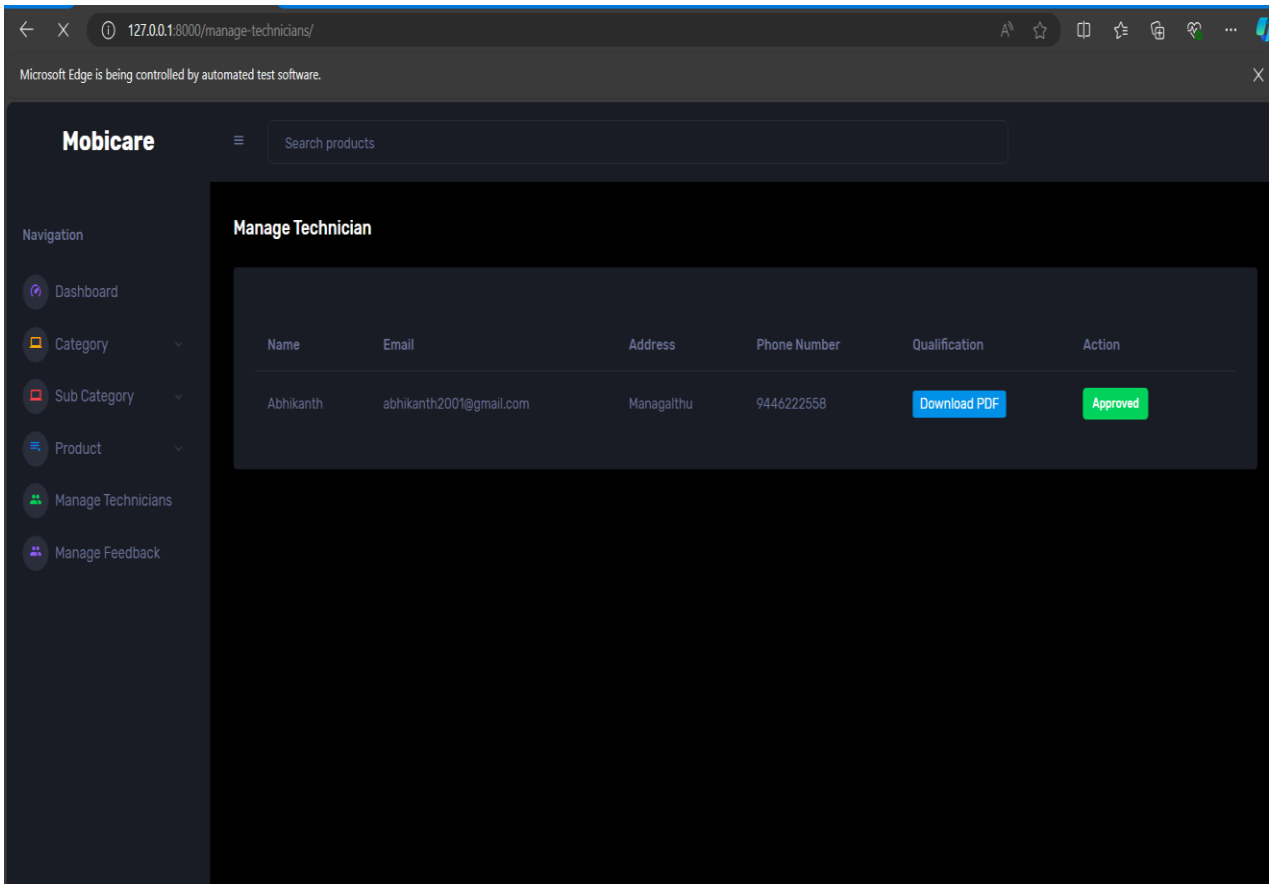


Fig 37: Add product

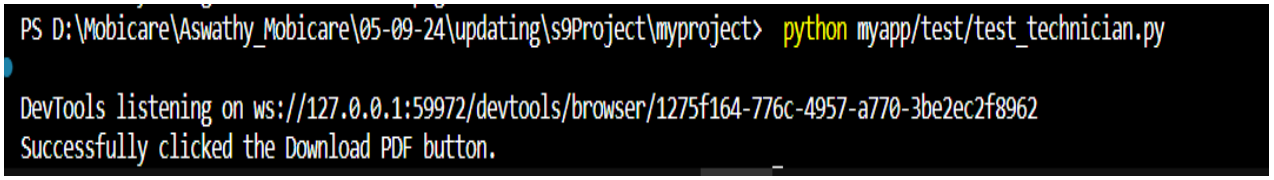


Fig 38: Terminal

Test report

Test Case 4

Project Name: MobiCare					
Test Case					
Test Case ID: Test_4			**Test Designed By: Aswathy M J**		
Test Priority (Low/Medium/High):High			**Test Designed Date: 16-10-2024**		
Module Name: qualification report downloads			**Test Executed By: Binumon Joseph**		
Test Title: Verify technician and download resume pdf			**Test Execution Date: 16-10-2024**		
Description: Technician resume downloading					
Pre-Condition: Admin has valid username and password					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status (Pass/ Fail)**
1	Navigate to login page	Mobicare123@gmail.com, Int@1234	Display login page	Login page is dsiplayed	pass
2	Navigate to admin home page		Navigate to the admin dashboard and manage techinican tab is displayed and then download pdf	admin navigate to dashboard and manage technician page is displayed and downloaded the resume pdf	pass
3	Click on the manage technician button				
4	Click on the download button				
Post-Condition: technician profile is approved by admin and download the resume pdf					

CHAPTER 6

IMPLEMENTATION

6.1INTRODUCTION

Implementing a new feature or functionality in a PHP project can be a challenging task that requires a thoughtful and structured approach. The first step in this process is to identify the specific problem or feature that you want to address. Once you have a clear understanding of what you want to achieve, you can plan out the different steps needed to implement the solution.

Planning is a crucial part of the implementation process as it helps you to break down the problem into smaller, more manageable chunks. This approach will make it easier to focus on each task, and ensure that you don't get overwhelmed by the project's size or complexity. During this stage, it's essential to consider any potential challenges or roadblocks that may arise during the implementation process, and develop contingency plans to address them.

The final stage of the implementation process involves writing the actual code to implement the feature or functionality. It is crucial to keep the code well-organized, well-documented, and modular, so that it can be easily maintained and updated in the future. During this phase, it's also important to test the code thoroughly, using various scenarios and edge cases, to ensure that the feature works as expected and meets all requirements. Once the code has been tested and validated, it can be integrated into the main project and made available to users.

6.2 IMPLEMENTATION PROCEDURES

Define the problem or feature: Before you start writing any code, clearly define the problem you are trying to solve or the feature you want to implement. This will help you stay focused and avoid unnecessary complications.

Plan your approach: Once you have a clear idea of what you want to achieve, plan your approach to the problem. Break it down into smaller, more manageable tasks and consider any potential challenges or roadblocks.

Write the code: Using your plan as a guide, start writing the code to implement the feature. Make sure to keep the code organized and well-documented, so it's easy to understand and maintain. Consider using a PHP framework like Laravel or CodeIgniter, which can simplify the coding process.

Test the code: Once you have written the code, test it thoroughly using different scenarios and edge cases to make sure it works as expected. Use a testing framework like PHPUnit to automate your tests and speed up the process.

Integrate the code: Once the code has been tested and validated, integrate it into the main project and deploy it to production. Monitor the feature closely to ensure that it's working as intended and

address any issues that arise.

Maintain and update: Finally, make sure to maintain and update the code regularly to ensure that it stays compatible with the latest versions of PHP and any other dependencies. This will help ensure the longevity of the feature and the overall health of the project.

6.2.1 User Training

When implementing a new feature or functionality in a Python project, it's important to consider user training as a key component of the implementation process. Here are some steps you can take to ensure that users are trained effectively:

Develop user documentation: Create user documentation that explains how to use the new feature or functionality. This can include step-by-step instructions, screenshots, and examples to help users understand how to use the new feature.

Provide training sessions: Consider providing training sessions to users to explain how to use the new feature. These sessions can be conducted in person, through video conferencing, or through online tutorials. Be sure to provide ample opportunities for users to ask questions and provide feedback.

Offer ongoing support: Even after the new feature has been implemented and users have been trained, it's important to offer ongoing support to address any questions or issues that may arise. This can include a help desk, support ticket system, or online forums where users can ask questions and share their experiences.

By taking these steps, you can ensure that users are trained effectively and feel confident using the new feature or functionality. This can help increase user adoption and satisfaction, which is critical for the success of any PHP project.

6.2.2 Training on the Application Software

Training on application software is a critical component of any software implementation project. Proper training helps users become proficient in using the software, reduces the learning curve, and ensures that the software is used effectively to achieve the desired outcomes.

Here are some steps that can be taken to provide effective training on application software:

Conduct a Needs Assessment: Before developing any training material, it's essential to understand the specific needs of the users. Conduct a needs assessment to identify the knowledge and skill gaps that exist among the users. This will help you develop targeted training material that addresses the specific needs of the users.

Develop Training Material: Once the training needs have been identified, develop training material that aligns with the identified needs. The material should be easy to understand, practical,

and engaging. It should include various formats such as video tutorials, manuals, and interactive e-learning modules.

Deliver the Training: After developing the training material, it's time to deliver it to the users. This can be done through in-person training, online training, or a combination of both. When delivering the training, it's essential to ensure that the material is engaging and interactive to keep users motivated and focused. It's also important to provide ample opportunities for users to ask questions and provide feedback.

By taking these steps, you can ensure that users receive the training they need to use the application software effectively. Effective training can help users become more proficient in using the software, increase user adoption, and ultimately, help the organization achieve its desired outcomes.

6.2.3 System Maintenance

System maintenance is the process of ensuring that a computer system or network is running smoothly and efficiently. It involves regularly checking and updating the software and hardware components of the system to ensure that they are working properly. Here are some important steps that should be taken to maintain a system effectively:

Regular Software Updates: Regularly updating software applications and operating systems is crucial to maintaining a secure and stable system. This involves installing the latest updates and patches, which can fix bugs, address security vulnerabilities, and improve system performance.

Hardware Maintenance: Regular hardware maintenance is essential to ensure that the system is running efficiently. This includes cleaning and dusting the system components, checking for loose connections, and ensuring that fans and cooling systems are working correctly.

Data Backup and Recovery: Regular data backups are essential to protect against data loss due to hardware failure, malware, or human error. A backup strategy should be established to ensure that critical data is backed up regularly and stored in a secure location. Additionally, a disaster recovery plan should be in place to ensure that data can be restored in the event of a system failure.

6.2.4 Hosting

Hosting a website means making it accessible on the internet by storing its files, databases, and other resources on a server. To do this, one typically selects a hosting provider, such as Bluehost, AWS, or Heroku, which offers various types of hosting, like shared, VPS, dedicated, or cloud hosting. Each type caters to different needs, from small personal sites to large, traffic-heavy platforms. Additionally, a domain name is registered to serve as the website's address on the internet (e.g., `www.example.com`). This domain is then linked to the hosting provider, allowing

users to access the site. Hosting providers often offer tools to manage website performance, security, and backups, ensuring a stable, accessible, and user-friendly experience for visitors.
Hosting a website on Render involves deploying web applications, static sites, or APIs on a fully managed cloud platform designed to simplify development and deployment processes. Render provides a range of hosting options that support modern web technologies and frameworks, including Python, Node.js, Django, and more. It allows developers to focus on building their applications without managing infrastructure, as Render handles tasks like scaling, server management, and SSL certificates automatically.

Render supports both “static and dynamic websites”. For static sites, developers can deploy HTML, CSS, and JavaScript files directly from a Git repository, making the site available through Render’s CDN for fast, global performance. For dynamic applications, Render can host web servers, databases, and background jobs, making it ideal for projects that require real-time data or API endpoints. Render also integrates easily with GitHub and GitLab, allowing for continuous deployment, where code changes are automatically deployed to the live site after each push. Render’s user-friendly interface, automatic scaling, and affordable pricing options make it a popular choice for developers looking to deploy professional websites and applications with minimal configuration.

Procedure for Hosting a website on Render:

Step 1: Create a Render Account: Go to [Render.com] (https://render.com/) and sign up for a free account if you haven’t already.

Step 2: Create a New Web Service: Once logged in, go to the Render dashboard, click on New→ Web Service. Choose to connect your Git repository (e.g., GitHub, GitLab) that contains your Django project.

Step 3: Configure Deployment Settings:

Environment: Select Python.

Build Command: Typically, use `pip install -r requirements.txt` to install dependencies.

Start Command: Set this to `gunicorn your_project_name.wsgi` (replace `your_project_name` with the name of your Django project).

Environment Variables: Add any necessary environment variables (e.g., `SECRET_KEY`, `DEBUG`, `ALLOWED_HOSTS`).

Step 4: Add a Database (Optional):

If your project requires a database, create a managed PostgreSQL database by selecting New

Databaseon the Render dashboard. Copy the database URL to use in your Django project settings.

Step 5: Update Django Settings for Render:

Database: Update `DATABASES` in your `settings.py` with the database URL from Render.

Allowed Hosts: Add Render's domain to `ALLOWED_HOSTS` in `settings.py`.

Static Files: Configure your project to serve static files. Typically, Render uses Whitenoise to handle this. Add `whitenoise.middleware. WhiteNoiseMiddleware` to `MIDDLEWARE` and specify `STATIC_ROOT`.

Step 6: Push Changes to Git Repository:

Commit and push any changes in `settings.py` or other configurations to your Git repository. Render will automatically deploy the latest version.

Step 7: Check Deployment Status:

Return to Render's dashboard and monitor the deployment logs. Render will show if the deployment was successful.

Step 8: Access Your Site:

Once deployed, Render will provide a URL for your Django project. Copy and paste this link into a browser to access your site.

Hosted Website: Render

Hosted Link: projectmobicare.onrender.com/

Hosted Link QR Code

ScreenShot

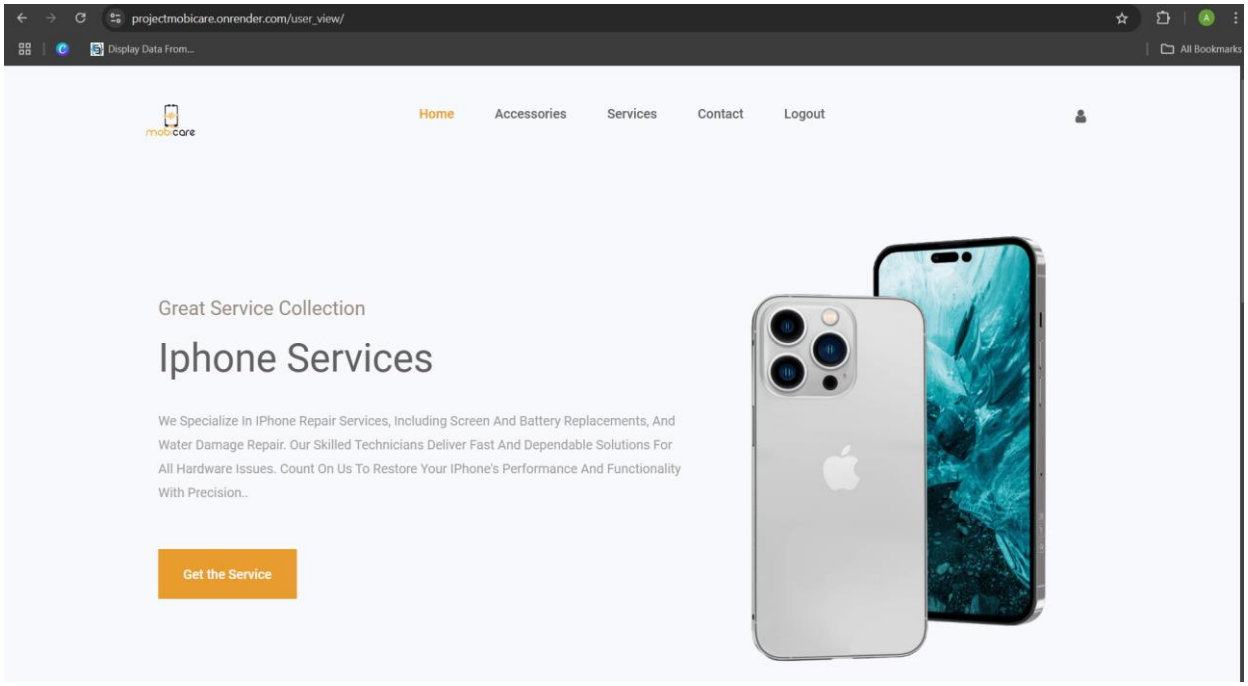


Fig 39: Home page

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The MobiCare platform is a feasible and valuable solution that addresses the needs of mobile service and accessories management. The feasibility study, covering technical, operational, and economic aspects, demonstrates that the system can be effectively implemented to streamline service requests, repairs, and sales processes. MobiCare enhances operational efficiency by automating service workflows, providing real-time updates, and improving customer communication. The platform's user-centric design and integration with existing systems ensure high user satisfaction, while its potential for increased sales and improved customer service supports its financial viability. Overall, MobiCare promises to deliver significant benefits, improving both business operations and customer experience.

7.2 FUTURE SCOPE

Looking ahead, MobiCare can evolve into a more robust platform by integrating advanced technologies such as AI-driven diagnostics for common mobile issues, allowing for automated service suggestions based on user inputs. Incorporating machine learning for predictive maintenance and sentiment analysis for customer feedback can further enhance user interaction. Expanding the platform to support additional services, such as mobile insurance or buyback programs for old devices, would broaden its appeal. Additionally, extending its capabilities to handle multiple locations and support for franchising could further drive scalability, making MobiCare a leading solution in the mobile service and accessories industry.

.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

1. Moussiades, L., and Adamopoulou, E. (2020). An introduction to chatbots. Information Processing International Federation (IFIP). Vincent, W. (2018). Django for Beginners: Build Websites with Python and Django. Self-published.
2. In 2019, Kim, J., and Oh, J. Customer service powered by AI: Opportunities and Challenges. AI Research Journal.
3. Smith, A., & Jones, M. (2018). Machine Learning for Diagnostics in Technical Support. International Journal of AI in Customer Service.
4. Kim, Y., and Lee, H. (2020). A thorough analysis of natural language processing in customer service. Journal of AI and Customer Experience.
5. Davis, R., & Chen, S. (2017). Interactive Tutorials for Technical Support: A Case Study. Journal of Interactive Learning.
6. Patel, T., & Sharma, P. (2019). Automated Troubleshooting Systems in E-Commerce. Technology Review. E-Commerce.
7. Nguyen, T., & Brown, L. (2020). User Centered Design of AI Service Systems. Human-Computer Interaction Journal.
8. Zhang, Q., & Wang, X. (2018). AI Driven Solutions for Home-Based Mobile Repairs. Journal of Mobile Device Repair.
9. Green, J., & White, E. (2017). Implementing AI in Mobile Repair Services: A Case Study. Journal of AI in Industry.
10. Miller, D., & Roberts, T. (2019). Managing Customer Service in the Digital Age: AI and Beyond. Journal of Digital Transformation.
11. Thompson, G., & Evans, R. (2020). AI in Service Management: Trends and Innovations. Service Journal. Management.
12. Choi, H., & Lee, K. (2018). Real-Time Assistance for Mobile Device Troubleshooting. Journal of Real-Time Systems.
13. Kim, J., and Park, S. (2019). Ethical Aspects of AI-Powered Customer Support. AI Ethics Journal.
14. Ali, M., & Ahmed, N. (2020). Enhancing User Engagement with AI Systems. Journal of AI and Human Interaction.
15. Wilson, J., & Moore, C. (2019). Future Directions in AI for E-Commerce. Journal of E-Commerce Innovation.
16. Brown, P., & Smith, J. (2019). Adaptive AI Systems for Personalized Customer

Support. Journal of Artificial Intelligence Research, 45(3), 123-145.

17. Torres, A., and Garcia, M. (2018). Natural language comprehension and speech recognition in contemporary chatbots. Neural Networks and Learning Systems Transactions, IEEE, 29(6), 1823-1835.
18. Hernandez, L., & Martinez, R. (2017). Challenges in Integrating AI with Customer Relationship Management Systems. International Journal of Information Management, 37(2), 157 164.
19. Wang, Y., & Zhang, L. (2020). AI Powered Diagnostic Systems for Mobile Devices. Journal of Mobile Computing and Communications, 12(1), 47-60.
20. Johnson, E., & Lee, S. (2018). Automating Customer Support with AI: Benefits and Limitations. Computers in Human Behavior, 78, 270-278.
21. Ng, C., & Tan, H. (2019). AI and the Future of E-Commerce: Emerging Trends and Technologies. Journal of E Commerce Research, 20(4), 321-337.
22. Lopez, D., & Morales, K. (2017). The Role of Machine Learning in Enhancing Customer Support Chatbots. Expert Systems with Applications, 85, 289-299.
23. Singh, R., & Gupta, M. (2020). AI-Driven Predictive Analytics for Customer Support. Journal of Business Analytics, 8(2), 98-110.
24. Ramos, J., & Pereira, F. (2018). User Experience Design for AI-Driven International Computer Journal of Human Studies, 114, 72-85.
25. Patel, S., & Shah, A. (2019). Ethical Considerations in Customer Support. AI-Powered Ethics and Information Technology, 21(3), 243-255.
26. Desai, S., and Kumar, A. (2020). AI's Effect on Online Retail Service Quality. 53, 101-115, Journal of Retail and Consumer Services.
27. Chung, Y., & Lee, J. (2021). Leveraging AI for Enhanced Customer Support: Innovations and Case Studies. Journal of Artificial Intelligence 15(4), 211-228. Applications.
28. Chen, L., and S. Hsu (2020). Trends and Technologies in Conversational AI for Customer Service. 173(2), 45-58, International Journal of Computer Applications.
29. Nguyen, H., & Nguyen, M. (2021). AI Driven Customer Engagement: Best Practices and Future Directions. Journal of Customer Experience Management, 8(1), 59-72.
30. Lee, J., & Park, H. (2019). Enhancing Technical Support with AI-Based Solutions: A Review and Future Directions. Computer Science Review, 34(3), 201-215.
31. Ryu, S., and H. Kang (2020). Developments and Difficulties in Natural Language

Processing for Chatbots in Customer Service. 123–137 in Journal of NLP and AI Research, 11(2).

32. Wu, Y., & Zhao, Q. (2021). Building Intelligent Customer Service Systems: A Comprehensive Survey. AI and Business Analytics Journal, 22(3), 89 104.
33. Li, J., & Zhang, T. (2018). Automated Customer Support Systems: Current Trends and Future Prospects. Journal of Automation Engineering, and 7(5), Control 342-357.
34. Kumar, R., & Reddy, V. (2020). Advances in AI for E-Commerce Customer Support: Opportunities and Challenges. Journal of E-Commerce Technology, 12(4), 101-114.
35. Chen, Y., & Liu, F. (2021). Enhancing Customer Service with AI: Case Studies and Best Practices. International Journal of Customer Relationship Management, 9(1), 77-92.
36. Liu, X., and Zhao, L. (2019). An Empirical Study of AI-Powered Retail Customer Support Solutions. 18(2), 203-220, Journal of Retail Technology and Management.
37. Bhatia, R., & Gupta, A. (2020). The Role of AI in Transforming Customer Experience in Retail. Journal of Retailing and Consumer Services, 54, 102-112.
38. Gonzalez, M., & Fernandez, A. (2019). AI for Customer Engagement: Trends and Implications. International Journal of Marketing Studies, 11(2), 45-60.
39. Khan, M., and S. Khan (2021). Customer service is being revolutionized by intelligent chatbots. Journal of Service Management, 32(3), 321-337.
40. Chaudhary, S., & Jaiswal, A. (2020). Enhancing Customer Loyalty with AI Solutions: A Study on E-Commerce Platforms. Journal of Business Research, 113, 157-168.
41. Joshi, P., and R. Patel (2021). investigating how AI affects customer relationship management. International Journal of Project Management and Systems, 9(1), Information 25–38.
42. Yadav, S., & Singh, A. (2019). AI in Customer Service: The Future of Customer Interaction. Journal of Business and Technology, 5(1), 1-15.
43. Thakur, R., & Srivastava, M. (2020). AI-Enabled Customer Support: An Overview and Future Directions. International Journal of Services and Operations Management, 37(2), 120-138.
44. Kumar, V., and S. Raji (2018). Opportunities and Challenges for AI in Customer Service in the Future. 10(1), 99-114, Journal of Strategic Management Studies.

45. Soni, R., and Misra, P. (2021). An analysis of AI chatbots in relation to conversational interfaces customer experience. and Digital Marketing Journal, 12(2), 45–60.
46. In 2021, Roy, S., and Dey, A. A Review of AI and Big Data in Improving Customer Service. 8(3), 1 20, International Journal of Business Analytics. revolutionized by intelligent chatbots.
47. Django Software Foundation. (2023). Django Documentation Release 4.2, Journal of Web Development Frameworks, 15(3), 200-215
48. SQLite Database Engine: Lightweight SQL Database for Applications, Journal of Embedded Database Technologies, 10(2), 75-90.
49. Booch, G., Rumbaugh, J., & Jacobson, I. (2005). The Unified Modeling Language User Guide (2nd ed.). Addison-Wesley Professional, 58-112.
50. Elmasri, R., & Navathe, S. B. (2015). Fundamentals of Database Systems (7th ed.). Pearson, 45-98.
51. Connolly, T., & Begg, C. (2015). Database Systems: A Practical Approach to Design, Implementation, and Management (6th ed.). Pearson, 281-315.
52. Date, C. J. (2019). An Introduction to Database Systems (8th ed.). Pearson, 102-187.
53. Kapoor, R. (2019). Selenium Testing Tools Cookbook (3rd ed.). Packt Publishing, 23-85.
54. Osherove, R. (2013). The Art of Unit Testing: With Examples in .NET (2nd ed.). Manning Publications, 15-72.

WEBSITES:

- https://www.mobilefun.com/
- https://docs.djangoproject.com/
- https://www.python.com/django.html
- https://realpython.com/tutorials/django/

CHAPTER 9

APPENDIX

9.1 Sample Code

```
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from .models import CustomUser
from django.contrib.auth.decorators import login_required
from.modelsimport
Category,SubCategory,Product,Cart,Payment,PhoneCategory,PhoneSubCategory,PhoneModel,Complaint,
ServiceRequest,TermsAndConditions, Payments, Wishlist,Feedback
from django.contrib import messages
from django.shortcuts import get_object_or_404
from django.http import HttpResponse
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.urls import reverse
import paypalrestsdk
from django.conf import settings
import razorpay
from django.utils.http import urlsafe_base64_encode, urlsafe_base64_decode
from django.utils.encoding import force_bytes, force_str
from django.contrib.auth.tokens import default_token_generator
from django.core.mail import send_mail
from django.template.loader import render_to_string
import json
from .forms import ComplaintForm
import joblib
from io import BytesIO
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
from reportlab.lib.units import inch
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, Table, TableStyle
from reportlab.lib.enums import TA_CENTER
from reportlab.lib.pagesizes import letter
from reportlab.lib.units import inch
from reportlab.lib import colors
from reportlab.lib.styles import getSampleStyleSheet
```

```
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer
from datetime import datetime
# Create your views here.
def index(request):
    return render(request, 'index.html')
def admin_view(request):
    category_count = Category.objects.count()
    product_count = Product.objects.count(
    context = {
        'category_count': category_count,
        'product_count': product_count,

        'username': request.user.username,
    }
    return render(request, 'admin_dash.html', context)
@login_required
def user_view(request):
    return render(request, 'user.html', {'user': request.user})
@login_required
def technician_view(request):
    context = {
         'username': request.user.username,
    }
    technicians = CustomUser.objects.filter(role='technician')
    return render(request, 'technician.html', context)
def user_login(request):
    if request.method == 'POST':
        email = request.POST['semail']
        password = request.POST['spassword']
        user = authenticate(request, username=email, password=password)
        if user is not None:
            if user.role == 'technician' and not user.is_approved:
                messages.error(request, "Your account is not yet approved by the admin.")
            else:
                if user.role == 'technician' and user.is_approved:
                    messages.success(request, "Your account has been approved by the admin.")
                login(request, user)
```

```
            if user.role == 'admin':
                return redirect('admin_view')
            elif user.role == 'user':
                return redirect('user_view')
            elif user.role == 'technician':
                return redirect('technician_view')
        else:
            return render(request, 'login.html', {'msg': 'Invalid email or password'})
    return render(request, 'login.html')
def register(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        email = request.POST.get('email')
        phone = request.POST.get('phone')
        address = request.POST.get('address')
        role = request.POST.get('role')
        password = request.POST.get('password')
        qualification = request.FILES.get('qualification', None)  # Get the uploaded file
        if role == 'technician' and not qualification:
            # Handle case where qualification is required but not provided
            return render(request, 'register.html', {'error': 'Qualification document is required for technicians.'})
        user = CustomUser.objects.create_user(
            username=email,
            email=email,
            password=password,
            first_name=name,
            phone=phone,
            address=address,
            role=role,
            qualification=qualification
        )
        user.save()
        return redirect('login')
    return render(request, 'register.html')
def logout_view(request):
    logout(request)
    return redirect('index')
```

9.2 Screen Shots

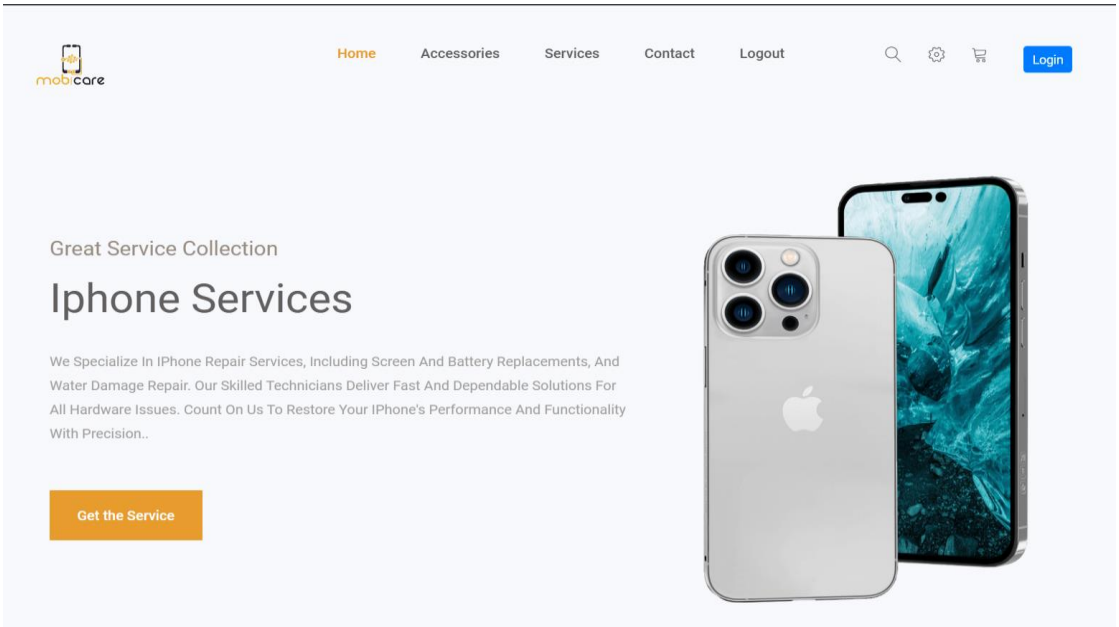


Fig 40: Home page

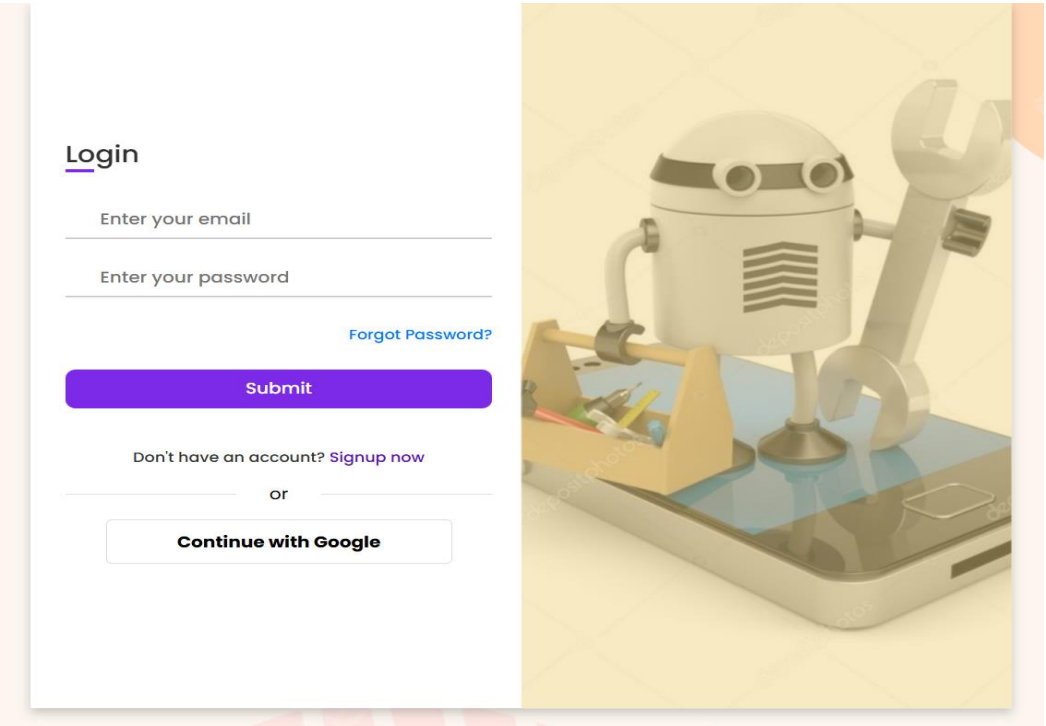


Fig 41: Login page

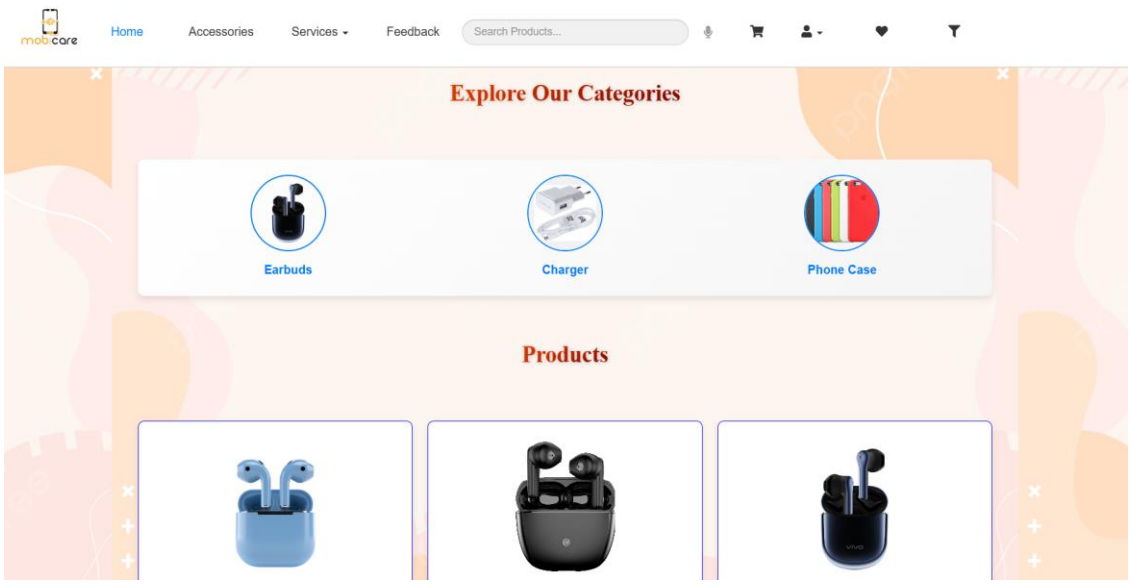


Fig 42: Accessories page

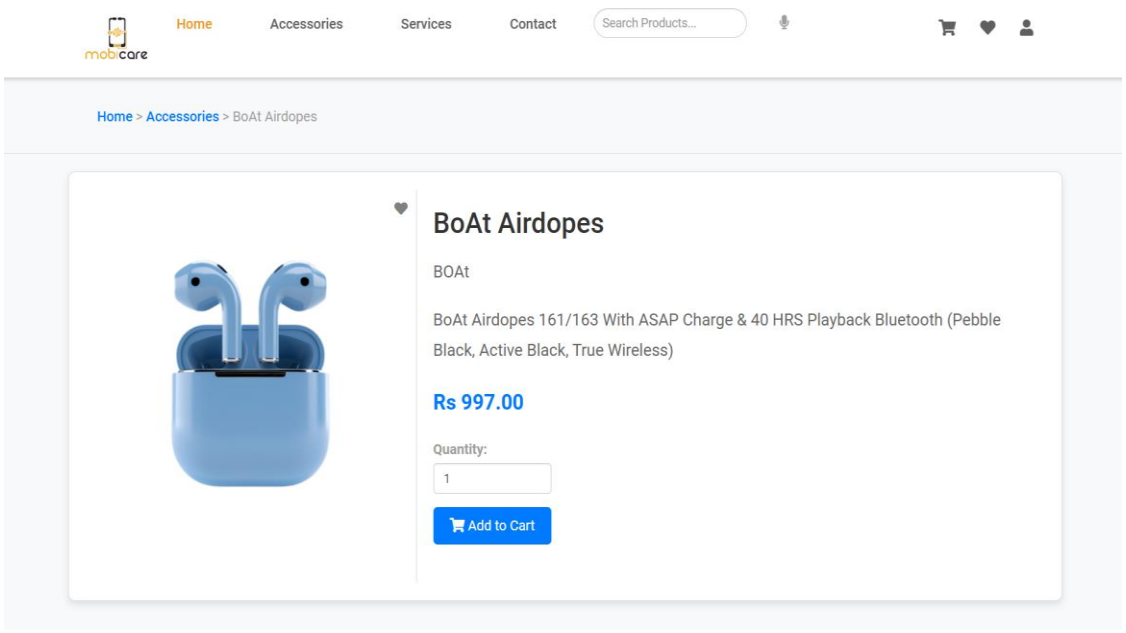


Fig 43: Product page

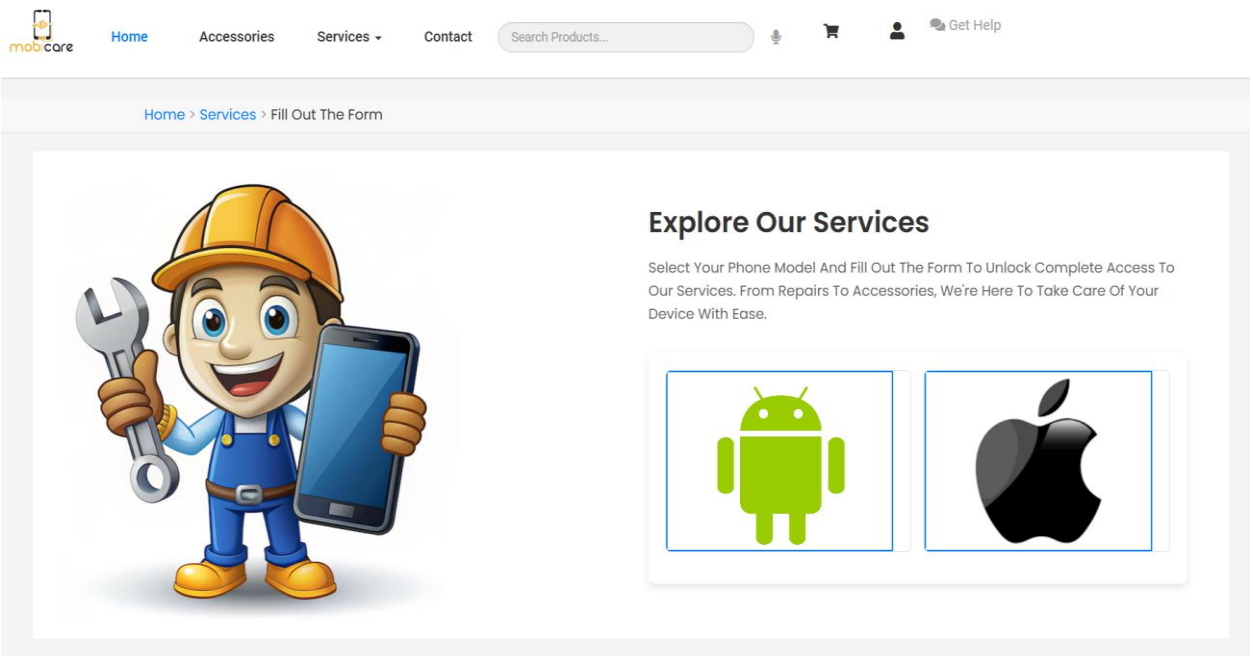


Fig 44: service page

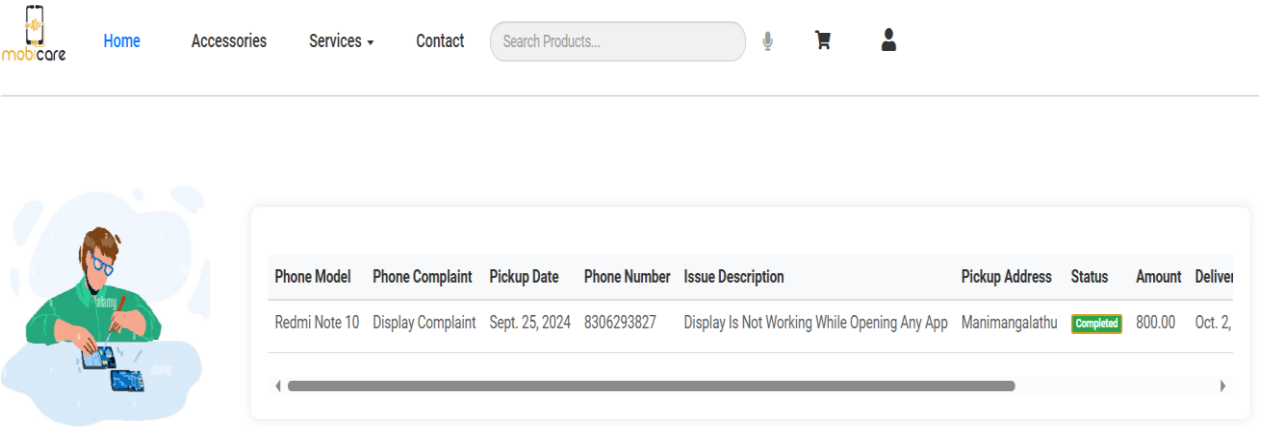


Fig 45: Manage Service page

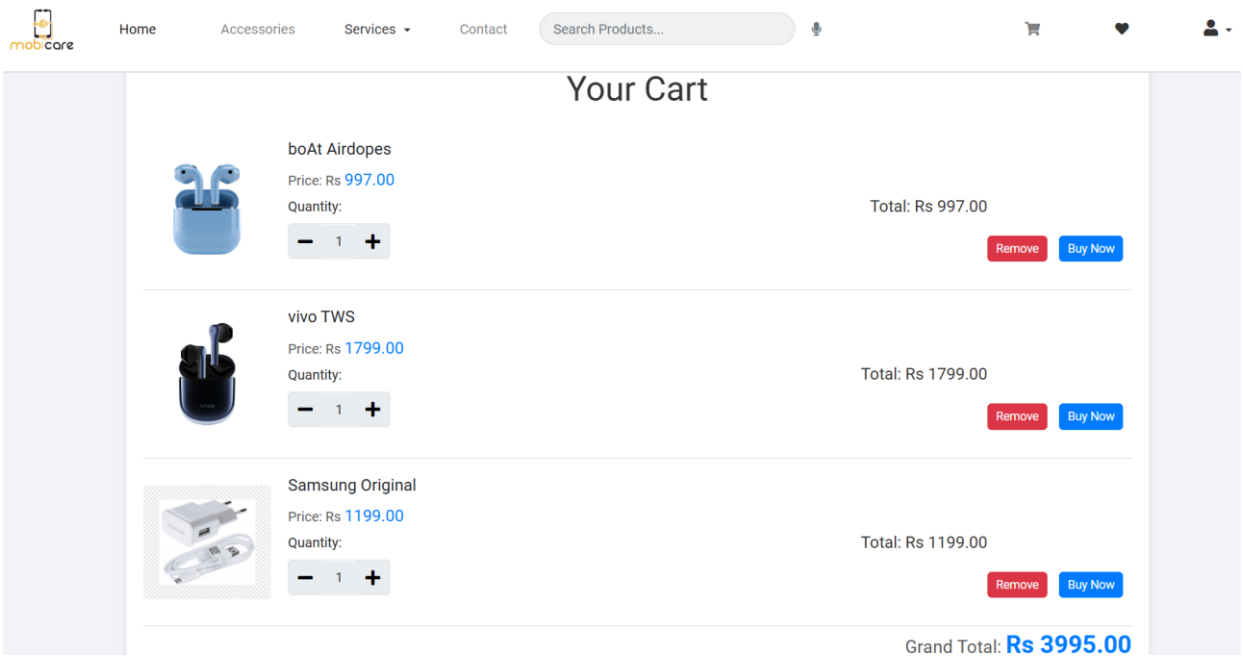


Fig 46: Cart page

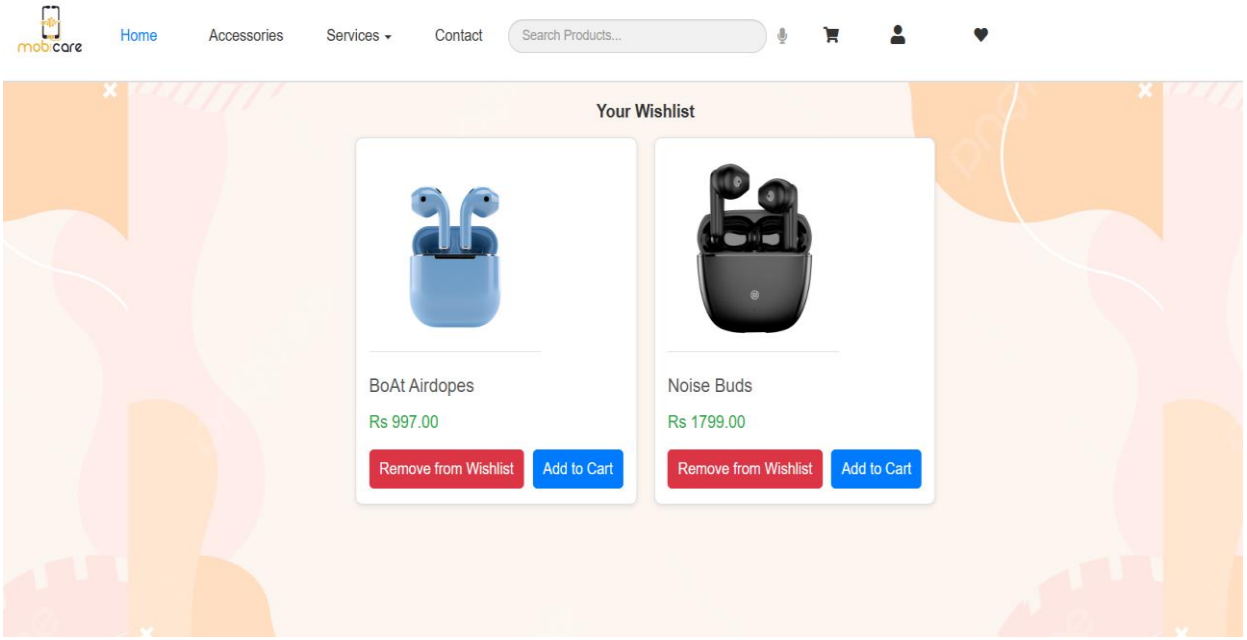


Fig 47: Whishlist page

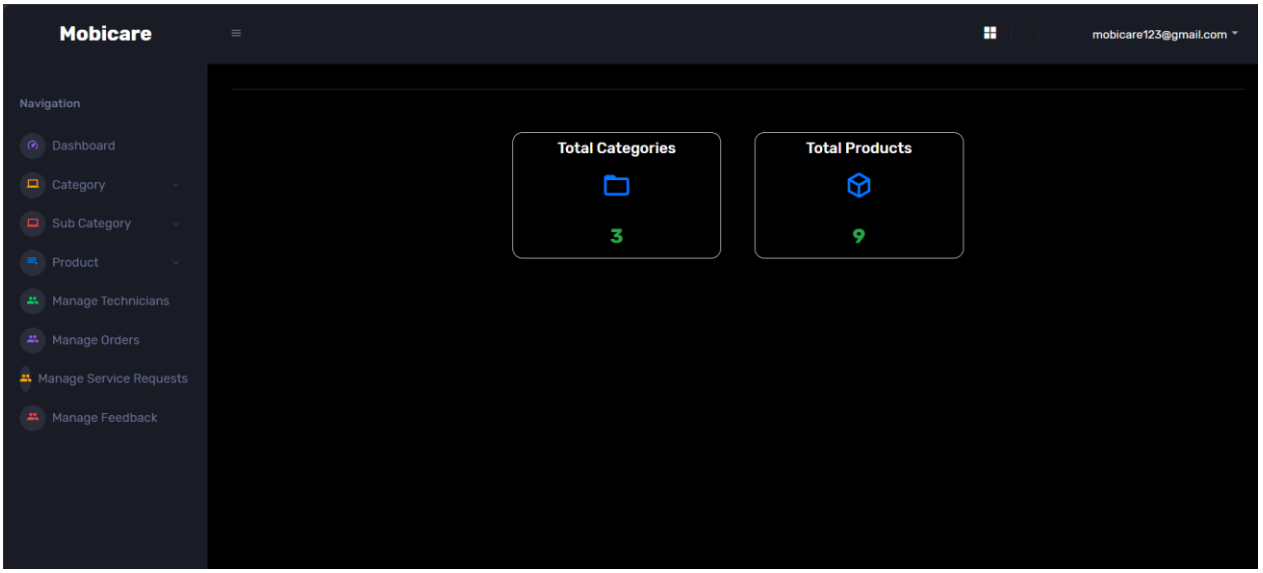


Fig 48: Admin page

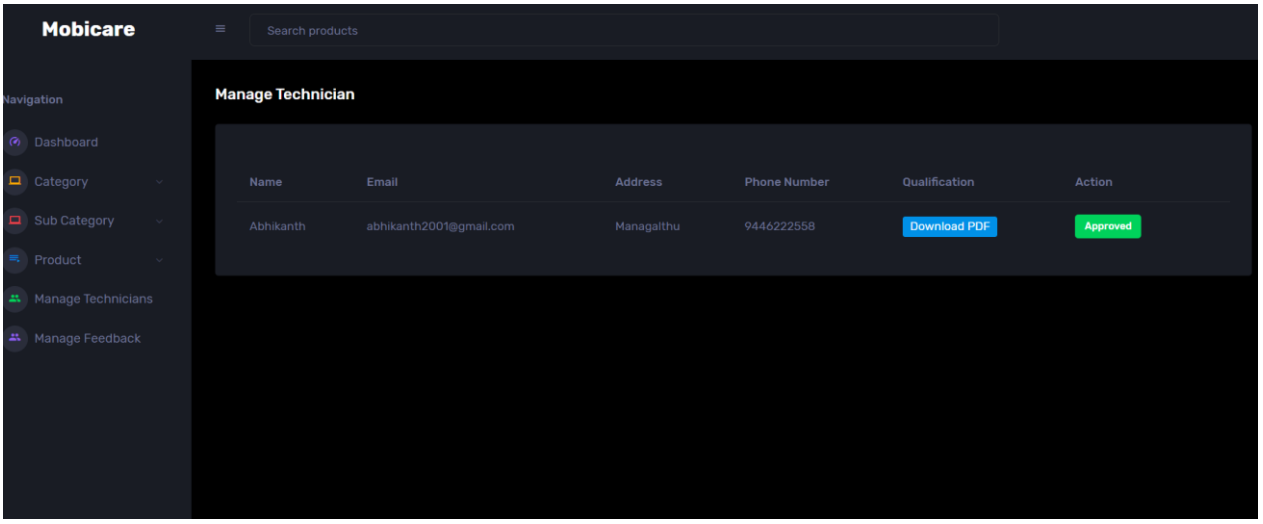


Fig 49: Manage Technician

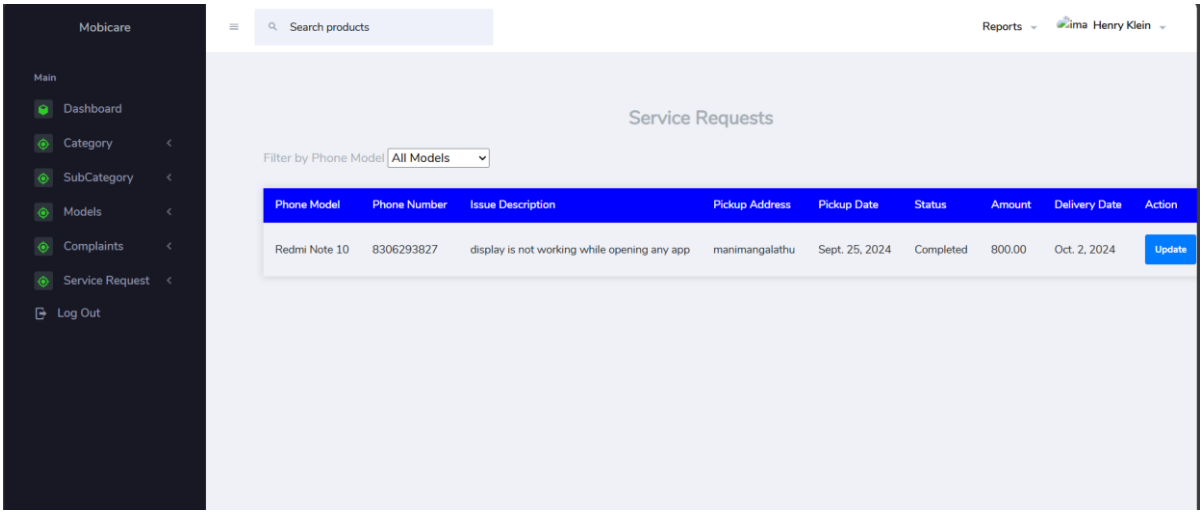


Fig 50: Manage Service Request

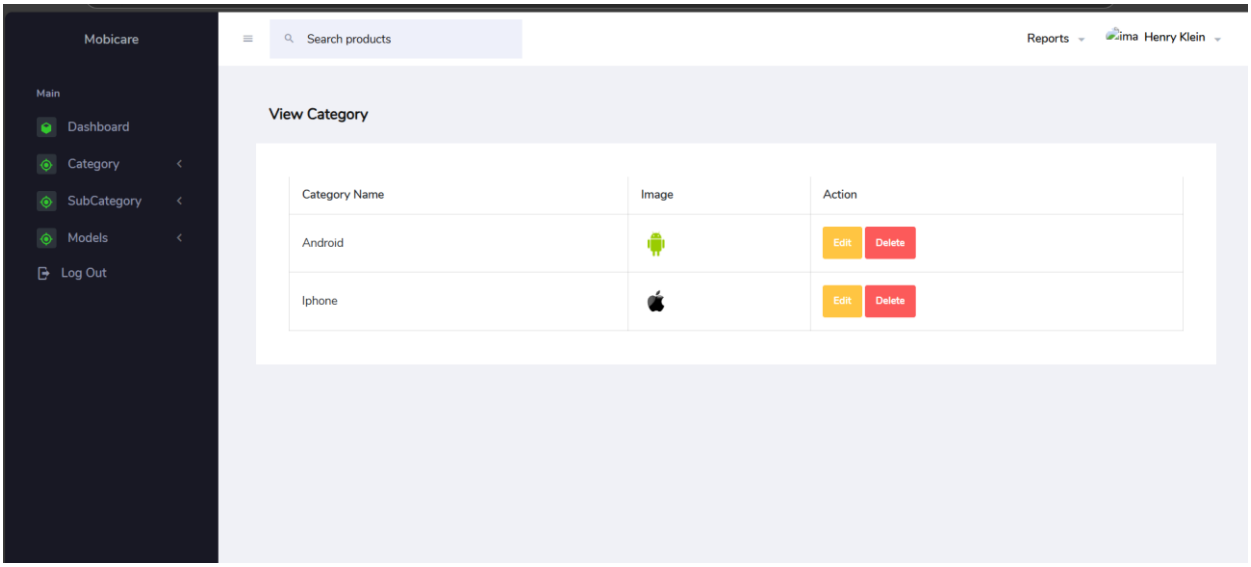
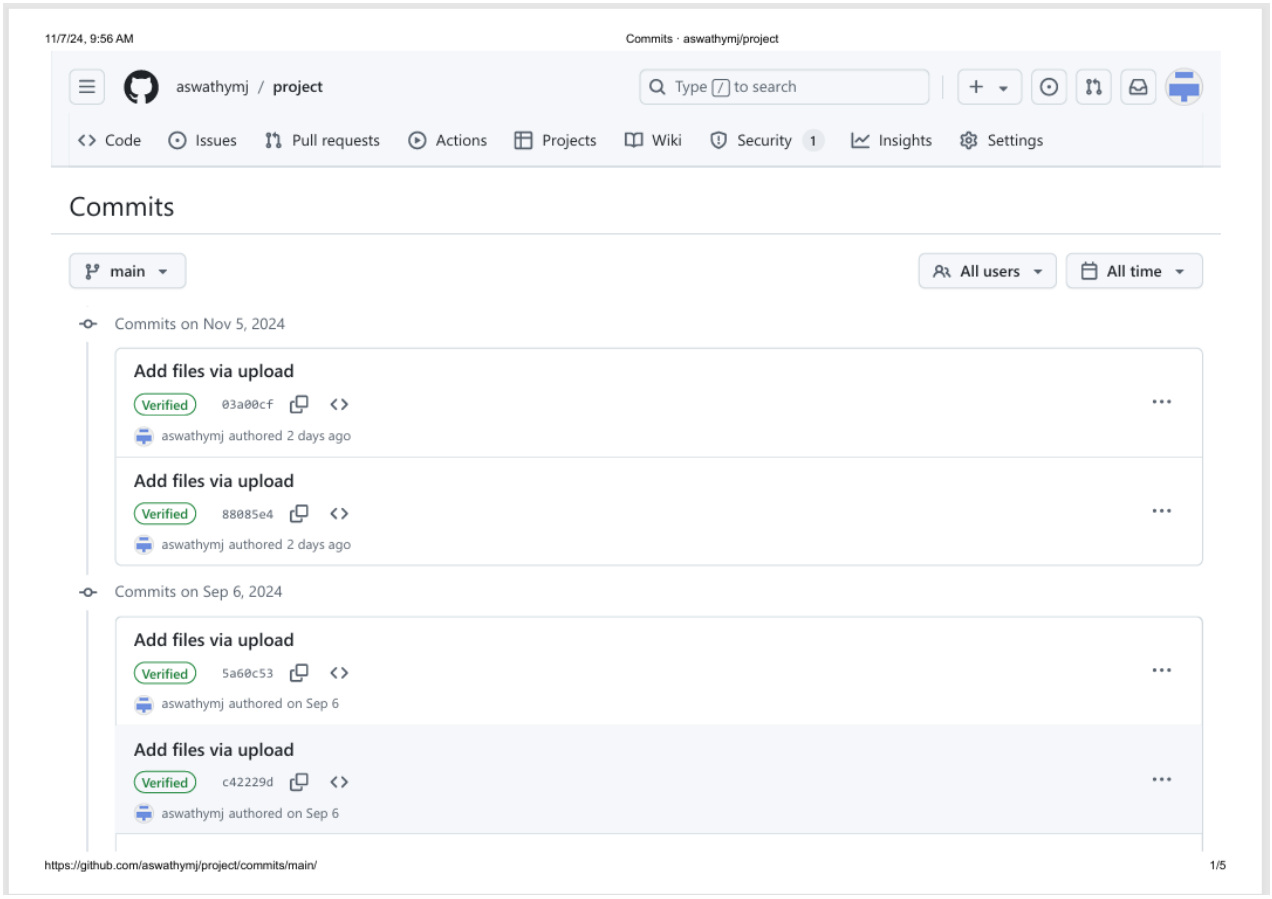
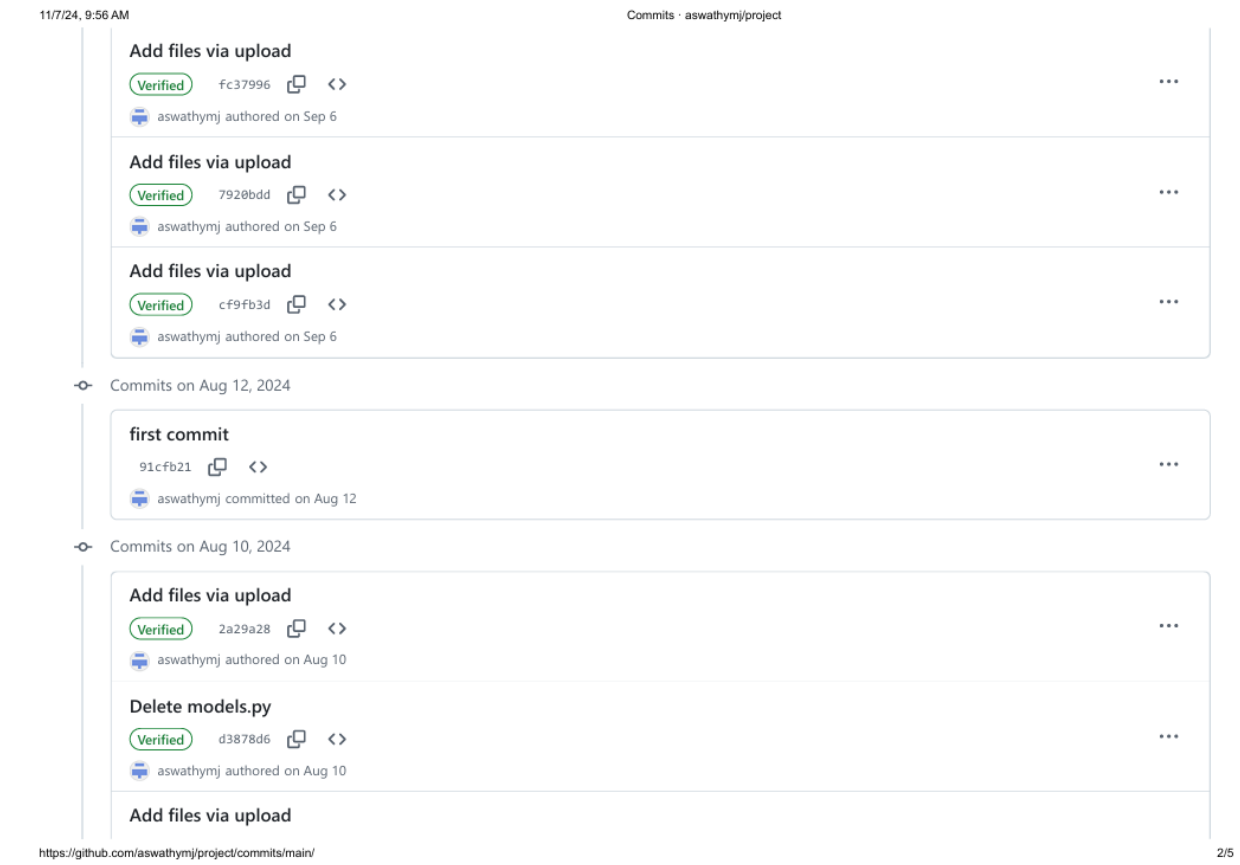
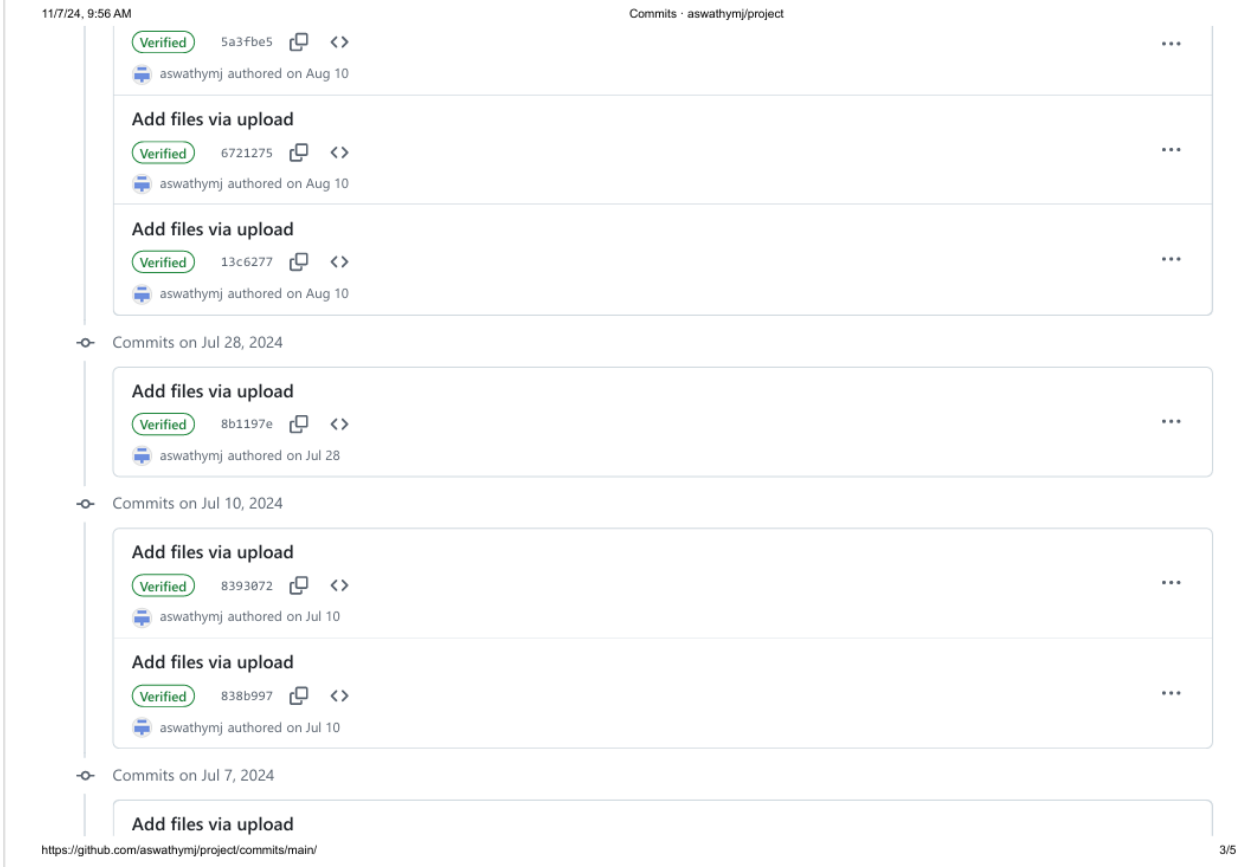


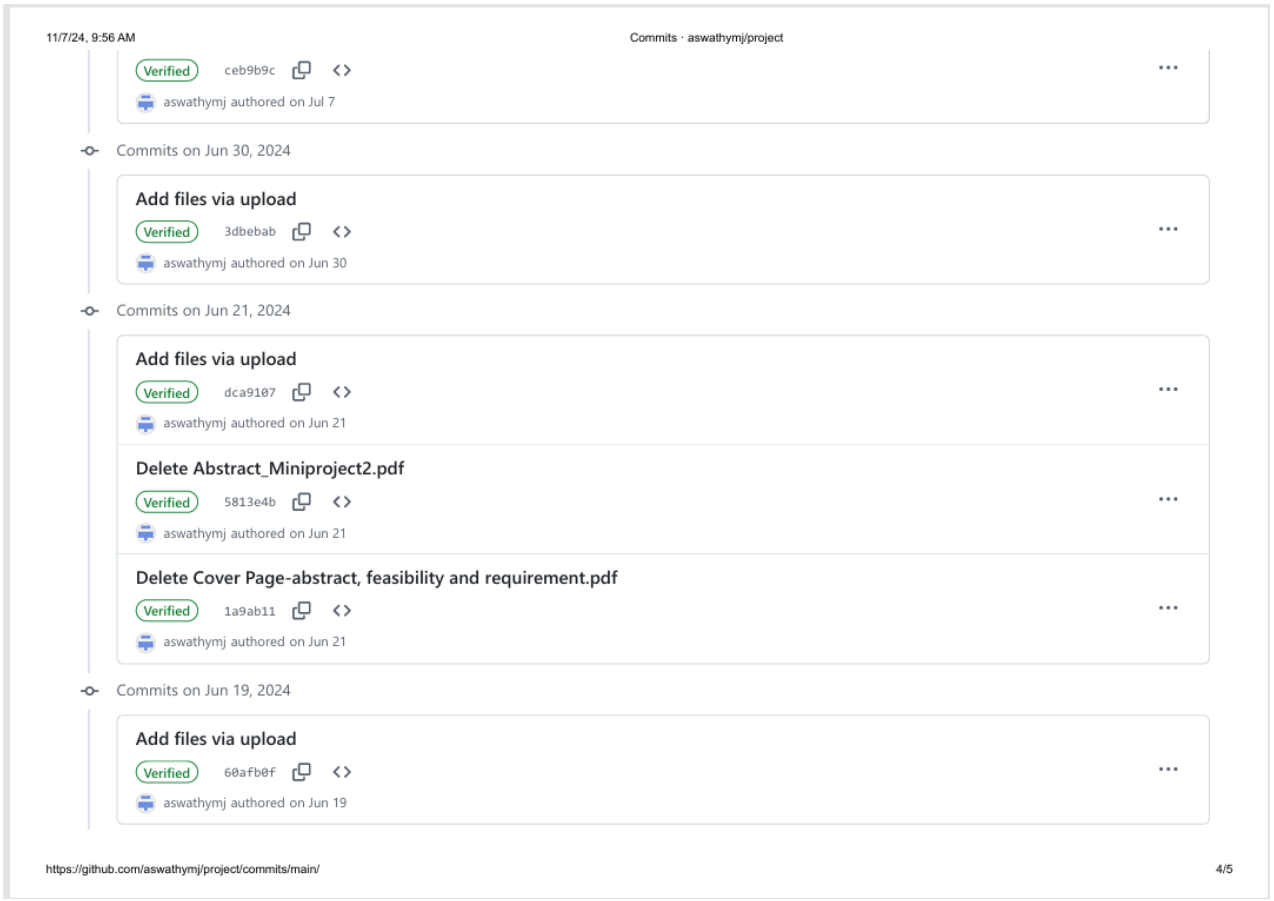
Fig 51: Add category page

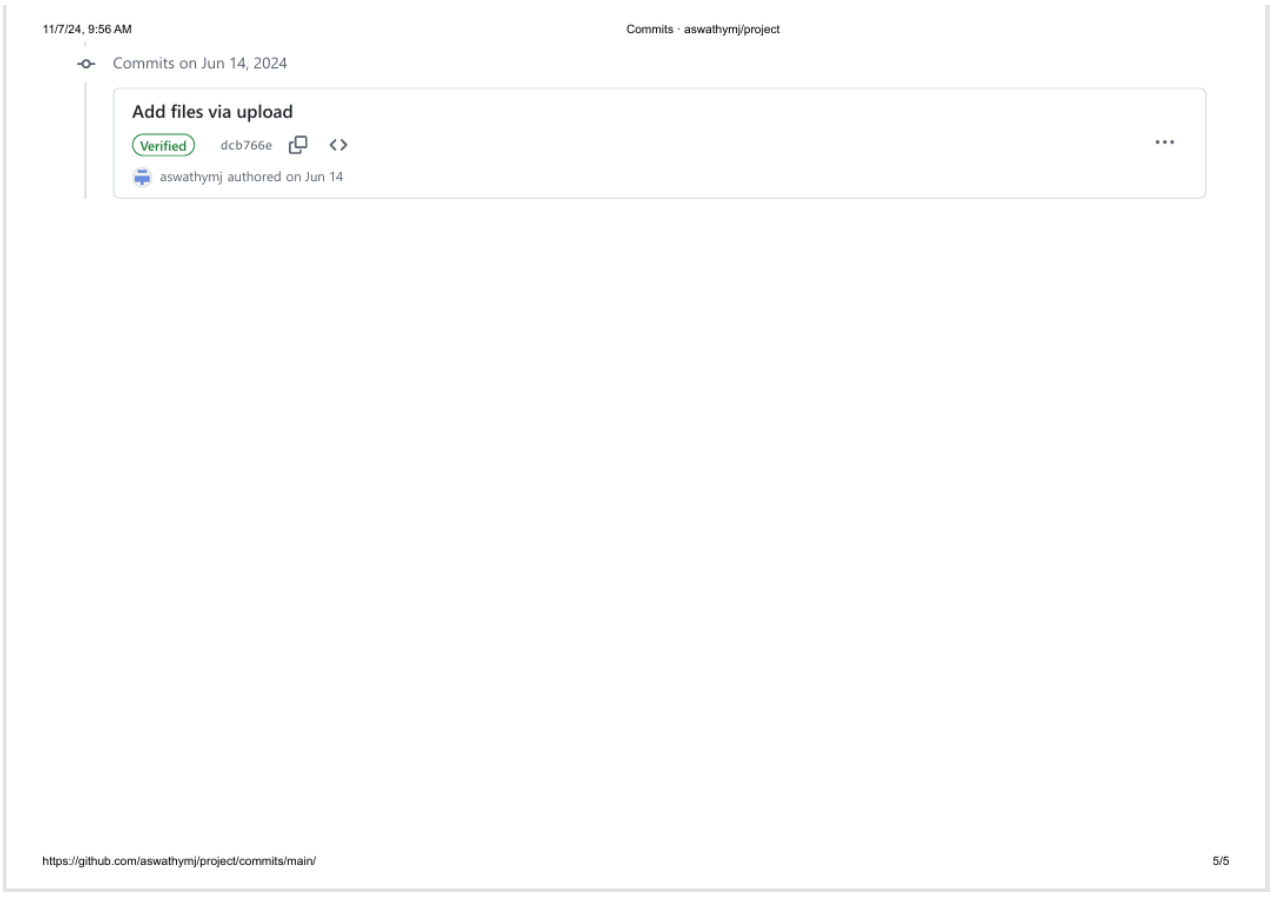
9.3 GIT LOG




11/7/24, 9:56 AM
Commits · aswathymj/project
Add files via upload
Verified
fc37996
aswathymj authored on Sep 6
Add files via upload
Verified
7920bdd
aswathymj authored on Sep 6
Add files via upload
Verified
cf9fb3d
aswathymj authored on Sep 6
Commits on Aug 12, 2024
first commit
91cfb21
aswathymj committed on Aug 12
Commits on Aug 10, 2024
Add files via upload
Verified
2a29a28
aswathymj authored on Aug 10
Delete models.py
Verified
d3878d6
aswathymj authored on Aug 10
Add files via upload
https://github.com/aswathymj/project/commits/main/
2/5


11/7/24, 9:56 AM
Commits · aswathymj/project
Verified
5a3fbe5
aswathymj authored on Aug 10
Add files via upload
Verified
6721275
aswathymj authored on Aug 10
Add files via upload
Verified
13c6277
aswathymj authored on Aug 10
Commits on Jul 28, 2024
Add files via upload
Verified
8b1197e
aswathymj authored on Jul 28
Commits on Jul 10, 2024
Add files via upload
Verified
8393072
aswathymj authored on Jul 10
Add files via upload
Verified
838b997
aswathymj authored on Jul 10
Commits on Jul 7, 2024
Add files via upload
https://github.com/aswathymj/project/commits/main/
3/5


11/7/24, 9:56 AM
Commits · aswathymj/project
Verified ceb9b9c
aswathymj authored on Jul 7
Commits on Jun 30, 2024
Add files via upload
Verified 3dbebab
aswathymj authored on Jun 30
Commits on Jun 21, 2024
Add files via upload
Verified dca9107
aswathymj authored on Jun 21
Delete Abstract_Miniproject2.pdf
Verified 5813e4b
aswathymj authored on Jun 21
Delete Cover Page-abstract, feasibility and requirement.pdf
Verified 1a9ab11
aswathymj authored on Jun 21
Commits on Jun 19, 2024
Add files via upload
Verified 60afb0f
aswathymj authored on Jun 19
https://github.com/aswathymj/project/commits/main/
4/5


11/7/24, 9:56 AM
Commits · aswathymj/project
Commits on Jun 14, 2024
Add files via upload
Verified dcb766e
aswathymj authored on Jun 14
https://github.com/aswathymj/project/commits/main/
5/5