Z Store
Used Products Selling Platform

Project Report

Submitted by

Muhamed Ashiq

Reg. No.: AJC22MCA-2063

In Partial Fulfillment for the Award of the Degree of

MASTER OF COMPUTER APPLICATIONS
(MCA TWO YEAR)
(Accredited by NBA)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

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

2022-2024

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

<u>CERTIFICATE</u>

This is to certify that the Project report, "**Z Store**" is the bona fide work of **MUHAMED ASHIQ (Regno: AJC22MCA-2063)** in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2023-24.

Ms. Ankitha Philip   **Ms. Meera Rose Mathew**

Internal Guide   **Coordinator**

Rev. Fr. Dr. Rubin Thottupurathu Jose

Head of the Department   **External Examiner**

DECLARATION

I hereby declare that the project report **"Z Store"** is a bona fide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2023-2024.

Date: **MUHAMED ASHIQ**

KANJIRAPPALLY   **Reg: AJC22MCA-2063**

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 **Ms. Meera Rose Mathew** for her 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 **Ms. Ankitha Philip** for her 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.

MUHAMED ASHIQ

ABSTRACT

.

The "Z Store" project is a groundbreaking initiative set to redefine the online marketplace landscape. With a focus on diversity, sustainability, and technological innovation, this platform transcends the conventional boundaries of e-commerce. At its core, the e-store aims to create a virtual space that caters to a wide spectrum of products, ranging from old vintage treasures to high-priced commodities, high-end vintage electronics, and contemporary short-term used items.

One of the key pillars of my project is sustainability. We recognize the value in extending the lifecycle of products, and my platform encourages users to engage in a circular economy by buying and selling pre-loved items. This not only contributes to a more sustainable and eco-friendly consumer culture but also fosters a sense of community among users who share a passion for unique, well-crafted products.

Here, particularly my platform utilizes something called the Bidding System which will be like giving the value to that product that a buyer is able to give to that particular product. similarly, there will be many users with having many bids and so, the user who sold the product select or automate the system to provide or do the rest of the thing, if manually then the user can select the favorable bid or if it is automated bidding then the user can set up the exit strategy and give it to the system.

CONTENT

List of Abbreviation

1. UML - Unified Modelling Language.

2. ORM - Object-Relational Mapping.

3. MVT - Model-View-Template.

4. MVC - Model-View-Controller.

5. RDBMS - Relational Database Management System.

6. 1NF - First Normal Form.

7. 2NF - Second Normal Form.

8. 3NF - Third Normal Form.

9. IDE - Integrated Development Environment.

10. HTML - Hypertext Markup Language.

11. JS - JavaScript.

12. CSS - Cascading Style Sheets

13. AJAX - Asynchronous JavaScript and XML

14. JSON - JavaScript Object Notation

15. API - Application Programming Interface

16. UI - User Interface

17. HTTP - Hypertext Transfer Protocol

18. URL - Uniform Resource Locator

19. PK - Primary Key

20. FK - Foreign Key

21. SQL - Structured Query Language

22. CRUD - Create, Read, Update, Delete

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

The "Z Store" Main project is like a small but powerful online marketplace adventure. It's all about selling and buying cool stuff – from vintage treasures to fancy electronics and even slightly used modern items. We care a lot about the planet, so my project promotes reusing things and being kind to the environment. We also love handmade stuff, so we're giving local artists a special place to show off their awesome creations. Plus, we've made it super easy and safe for people to shop and sell online with my user-friendly technology.

Imagine a tiny world where you can find unique items, support local artists, and help the Earth – that's my Main project. Whether you're a collector, an artist, or just someone who wants cool things, my "Z Store" is here to make online buying and selling simple, fun, and good for the planet.

1.2 PROJECT SPECIFICATION

<u>User View</u>

1. **Home Page:**
    - The platform welcomes users with an aesthetically pleasing and intuitive homepage interface upon entry.
    - Users can seamlessly browse and filter a diverse array of listed products, with options to make immediate purchases or add items to their shopping cart for future transactions.
    - Beyond buying, the platform empowers users to engage actively by providing functionalities for selling their products and editing personal details, enhancing the overall user experience and platform engagement.

2. **User Settings Page**
    - The page serves as a centralized hub for users to manage their current settings, facilitating seamless updates to their profile information.
    - Users have the flexibility to enhance their profile by updating or removing their profile photo, ensuring a personalized and dynamic online presence.
    - Password management is streamlined on this page, enabling users to securely update their password by confirming their existing one, contributing to a robust and user-centric security framework.

3. **Product Selling Page**
    - The product selling page provides a dedicated space for users to showcase and manage their listed items, streamlining the selling process.

- Users can effortlessly insert product details, including images, to present their items in the best possible light, enhancing visibility and appeal.

4. **User Cart Page**

- The user's cart page serves as a convenient hub for reviewing and managing selected items before finalizing a purchase.

- Users can effortlessly remove items, or explore related products, ensuring a flexible and user-centric shopping experience.

5. **Check Out Page**

- The user checkout page offers a streamlined and secure process for users to complete their purchases with confidence.

- Users can easily review and confirm their selected items, quantities, and total costs before proceeding to payment, ensuring accuracy in their orders.

- The checkout page integrates multiple payment options and facilitating a smooth and hassle-free transaction experience.

6. **User's Product's Page**

- This page in particular will be handling the information that is obtained by the system of their product and also the information of the products that they have bidded on.

- Also, here in this particular page the user can sell the product for particular bid where the user will have the information on the profit they may get, all available bids for their product etc.

- Also the user will also be shown to buy the product if that particular user's bid is selected by the seller user.

7. **My Orders Page**

- This Page will be having the information on the products that we purchase also the delivery tracking is also in it.

- Also this page provides the products we bought which can be filtered and download by the user as in PDF Format.

Admin View

1. **Admin Dashboard**

- The admin dashboard page serves as a centralized control center, presenting a comprehensive overview of platform activities and performance.

- The admin dashboard includes the functionality to suspend user accounts when necessary, providing a crucial tool for enforcing platform policies and maintaining a

secure community.

- A comprehensive list of users and testers is readily available on the dashboard, facilitating quick access to user profiles and testing data for efficient management and support purposes.

2. **Add Tester Page**

- The "Add Tester" page offers a straightforward interface for administrators to onboard new testers, contributing to the expansion and diversification of the testing community.

- Admins can input essential tester details, including profile information and testing preferences, streamlining the process of integrating new members into the testing.

- The page incorporates validation checks and user-friendly error handling, ensuring accurate and complete information entry for a smooth and error-free tester registration process.

3. **Add Delivery Boy Page**

- After the delivery boy sends a request for the job the request will accepted by the admin and further update will be added.

- The admin will be adding how and where the delivery boy is located and the salary of the delivery boy will be decided and imputed by the admin.

- This will be then send as the mail to the delivery boy with his username and password for the platform, also the salary he will be receiving will also be mentioned.

4. **Add Category Page.**

- This page is particularly designed for adding more categories, sub-categories or brand names for that particular category.

- Here the admin can also see the all available brands, sub-category and category and how many products each of these has and how much worth of these are present will be known.

Tester View

1. **Tester Panel**

- The tester panel provides a dedicated space for testers to access and manage their testing assignments efficiently.

- Testers within the panel have the responsibility to verify products listed by users, ensuring adherence to quality standards and platform guidelines.

- The panel allows testers to modify values if users input incorrect or invalid data when listing products, maintaining data accuracy and consistency.

- Testers can utilize the panel's functionality to and upload pictures, providing visual confirmation and enhancing the authenticity of product verification.

2. **View Page**

- The Tester View page presents a comprehensive overview for testers, showcasing products that they have successfully verified, providing a sense of accomplishment and contribution.

- In addition to verified products, the page displays a section dedicated to unverified products within the tester's designated category and locality, enabling focused attention on items that require validation.

<u>**Delivery Man's View**</u>

1. **Delivery Man Dashboard**

- This particular page will be having the data of all the delivery requests they can handle with the information and the delivery by date by which the Delivery boy has to deliver it to the buyer.

- From this page the delivery boy can select the request that is received to the system or platform, this will be sorted and identify the delivery boy based on the location they are in also the least distance (Euclidean Distance) between the product location and the delivery boy

- Also each delivery boy will be having a weightage and according to their weightage the system will identify the nearest free delivery boy and send the job request to that delivery boy, which will be shown in this page.

CHAPTER 2

SYSTEM STUDY

2.1INTRODUCTION

A critical stage in the creation of any system is system analysis. Its main objective is to collect and examine data in order to identify issues and provide solutions. The key to this phase is effective communication between system users and developers. In fact, a system analysis should always be the first step in every system development project.

Here, the system analyst assumes the role of an investigator, carefully evaluating the effectiveness of the current system. This requires determining the system's inputs and outputs as well as the relationship between its activities and the outcomes of the organization.

Information is gathered through a variety of methods, including surveys and interviews. The broad objectives include learning how the system works, identifying problem areas, and suggesting solutions to deal with the problems facing the company.

2.2EXISTING SYSTEM

2.2.1 NATURAL SYSTEM STUDIED

Natural System Study for the "Z Store" offline system provides a comprehensive understanding of the traditional practices surrounding the exchange of unique and crafted items. Through an exploration of buying and selling practices, community dynamics, sustainability efforts, technological integrations, and identified challenges, we have gained valuable insights into the historical context of these transactions.

This approach ensures that the online platform aligns with and enhances the strengths of the existing offline system, creating a dynamic and inclusive marketplace for unique and crafted products.

2.2.2 DESIGNED SYSTEM STUDIED

Designed System Study for the current online platform of the "Z Store" involves a meticulous examination of its various components. Starting with the user interface and experience, the study assesses the visual appeal and usability of the platform, focusing on how users navigate, search for products, and complete transactions. The aim is to ensure that the design fosters an intuitive and engaging experience for both buyers and sellers.

In the realm of product listings and management, the study delves into the process of how users present their products. This includes evaluating the comprehensiveness of information provided, the quality of images uploaded, and the effectiveness of categorization. Additionally, attention is given to the platform's support for inventory management, ensuring that sellers can efficiently update, add, or remove products as needed.

Security and privacy are paramount considerations, and the study rigorously assesses the implemented measures to safeguard user data and financial transactions. This includes an examination of privacy controls and an evaluation of how the platform complies with relevant regulations, ensuring a secure and trustworthy environment for users.

Finally, the study investigates customer support and conflict resolution mechanisms. This includes an evaluation of responsiveness, available channels for support, and the platform's approach to addressing conflicts. User feedback mechanisms are also analyzed to understand their integration into system improvements, ensuring continuous enhancement based on user insights. The Designed System Study aims to provide a holistic understanding of the existing online system, guiding refinements and innovations to align with user expectations and industry standards.

2.3 DRAWBACKS OF EXISTING SYSTEM

1. Limited User Interface: The existing system might have a dated or complicated user interface, making it challenging for users to navigate and engage with the platform effectively.

2. Ineffective Product Verification Processes: If the system doesn't have robust mechanisms for verifying product authenticity, it may be susceptible to fraudulent listings, undermining user trust.

3. Invalid Price Tags: Pricing discrepancies, whether due to human error or intentional manipulation, can create confusion among users. Impact on the fairness of transactions and potential financial disputes.

4. Product Authenticity: Verification challenges may lead to discrepancies between product listings and actual item conditions. Difficulty in ensuring accurate representation of products, impacting user trust and satisfaction.

5. Lack of Quality Control: Inconsistent product quality due to limited mechanisms for assessing and ensuring the standard of items listed on the platform.

6. Insufficient Security Measures: Gaps in security protocols, leaving user data vulnerable to unauthorized access and potential breaches.

7. Complex Navigation: Challenges in navigating the platform, particularly for new users, leading to difficulties in finding products or completing transactions.

8. Insufficient Transparency: Lack of transparent communication on platform policies, updates, and changes, contributing to user confusion and uncertainty.

9. Geographical Limitations: Restrictions in the reach of the platform, limiting access for users in certain geographical areas and potentially hampering market expansion.

10. Scams and Fraudulent Activities: Online nature of the platform exposes vulnerabilities,

allowing for potential scams and fraudulent transactions. Possibility of malicious users exploiting loopholes, resulting in financial losses and undermining platform credibility.

2.4 PROPOSED SYSTEM

"Z Store" meticulously examines the various components that constitute the proposed online platform. Beginning with the user interface and experience, the study places a strong emphasis on the visual appeal and intuitiveness of the homepage, ensuring that users are greeted with a seamless and engaging entry point. Additionally, it evaluates the navigation system, prioritizing a user-friendly design that facilitates effortless browsing, filtering, purchasing, and cart management for an optimal overall experience.

The functionality related to selling and listing products is subjected to detailed scrutiny, with a focus on the user-friendly features that empower sellers to efficiently manage their inventory, update product details, and set pricing. The study also delves into the intricacies of user profile management, examining the ease with which users can update their information, including profile photos and passwords, while adhering to robust security standards to safeguard sensitive data.

The administrative dashboard's functionality is evaluated comprehensively, focusing on its role as a centralized control center for overseeing user activities, sales analytics, and product listings. This includes features that empower administrators to suspend users and efficiently manage user and tester lists. The tester panel and verification process are examined in detail, emphasizing the tools available to testers for modifying product details, capturing images, and providing ratings during the verification process.

The Designed System Study lays the foundation for refining and optimizing the "Z Store" to not only meet but surpass user expectations. Through a thorough examination of each component, the study aims to ensure that the online platform aligns seamlessly with the project's overarching goals of sustainability, craftsmanship, and technological sophistication.

2.5 ADVANTAGES OF PROPOSED SYSTEM

1. Diverse Product Range: The proposed system aims to cater to a wide range of products, including old vintage items, high-priced commodities, high-end vintage electronics, and modern short-term used products, offering users a diverse and comprehensive marketplace.

2. Sustainability Focus: By encouraging a circular economy and promoting the buying and selling of pre-loved items, the proposed system aligns with sustainability goals, contributing to eco-friendly consumer practices.

3. Craftsmanship Celebration: The platform provides a dedicated space for artisans and creators to showcase their work, celebrating craftsmanship and offering users a unique selection of handcrafted and artistic products.

4. Technological Sophistication: The proposed system integrates advanced technological features to provide users with a seamless and secure shopping experience, demonstrating a commitment to staying at the forefront of e-commerce technology.

5. User-Friendly Interface: Users are greeted with a visually appealing and user-friendly homepage, emphasizing the importance of a positive and intuitive user experience from the moment they access the platform.

6. Tailored Solutions for Users: The platform offers tailored solutions for different user needs, whether they are passionate collectors, artisans, or individuals looking for unique items, demonstrating a commitment to meeting distinct user requirements.

7. Unified Buying and Selling Environment: Users can both sell and buy products within a unified environment, streamlining the overall user experience and providing a one-stop platform for all their buying and selling needs.

8. Privacy and Security Measures: The proposed system places a strong emphasis on security and privacy, ensuring users can engage confidently in transactions within the platform while maintaining the confidentiality of their personal information.

9. Community Building: By fostering a sense of community among users who share a passion for unique items, the platform goes beyond a traditional e-store, creating a vibrant online community around shared interests.

10. Integration of Sustainability, Craftsmanship, and Technology: The unique fusion of sustainability, craftsmanship, and technological sophistication sets the proposed system apart, offering users a platform that combines environmental consciousness, artistic expression, and cutting-edge technology in a cohesive manner.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The primary purpose of conducting a feasibility study is to comprehensively assess whether the proposed project can successfully meet the organization's objectives in terms of available resources, labor, and time. This crucial study allows the developers and decision-makers to gain valuable insights into the project's potential viability and prospects. By carefully examining various aspects of the proposed system, such as its impact on the organization, its ability to fulfill user requirements, and the optimal utilization of resources, a feasibility study helps in determining the project's feasibility and potential success.

The assessment of the proposed project's feasibility involves multiple dimensions, each playing a critical role in the decision-making process.

- Technical Feasibility
- Operational Feasibility
- Economic Feasibility

A well-conducted feasibility study provides valuable insights to decision-makers, allowing them to make informed judgments about the project's potential success. It assists in identifying potential risks, challenges, and opportunities associated with the proposed endeavor, enabling stakeholders to devise effective mitigation strategies.

3.1.1 Economical Feasibility

As a student project, the economic feasibility study for the "Z Store" acknowledges the constraints of limited resources and a primary focus on learning outcomes. The market analysis is conducted with a keen understanding of the target audience's preferences, aligning with the project's defined scope and objectives. Cost estimation takes a pragmatic approach, considering realistic assumptions and accounting for available funding designated for software development, infrastructure, and marketing endeavors.

Financial projections are crafted with the recognition of the project's finite duration and scope, aligning closely with the confines of a student project. In essence, the economic feasibility study aims to equip the student team with valuable insights to make informed decisions, giving precedence to the educational aspects of the project. The emphasis is on nurturing skills, gaining practical experience, and fostering a deep understanding of the project lifecycle, acknowledging that the primary goal is educational growth rather than immediate financial gains.

3.1.2 Technical Feasibility

The technical feasibility study for the "Z Store," being a student project, focuses on evaluating the practicality of developing and implementing the proposed platform with the available technical resources and skills. The assessment encompasses an evaluation of the student team's technical expertise, the accessibility of required technologies, and the infrastructure needs essential for project realization.

The study acknowledges the learning-oriented nature of the project, prioritizing the development of technical skills and understanding. It emphasizes the importance of practical experience gained through addressing technical challenges and making informed decisions. In doing so, the technical feasibility study aims to equip the team with a realistic assessment of their ability to bring the proposed online marketplace to fruition within the confines of their technical capabilities and resources.

3.1.3 Behavioral Feasibility

Evaluating the behavioral feasibility of the "Z Store" is a pivotal aspect of my feasibility study, focusing on the alignment of the system with users' needs and behaviors. To conduct this assessment effectively, we establish a robust outline of system requirements that encompasses inputs, outputs, programs, and procedures. This meticulous approach serves as the foundational framework for my evaluation, ensuring a comprehensive understanding of user expectations and system functionalities.

In essence, the behavioral feasibility study aims to provide a thorough understanding of how well the "Z Store" aligns with the behaviors and needs of its intended users. By establishing clear requirements and considering the necessary resources, we aim to create a platform that not only meets but exceeds user expectations, fostering positive engagement and satisfaction within the limitations of a project.

3.1.4 Feasibility Study Questionnaire

1) **Project Overview?**

The project aims to create an innovative online marketplace, named "Z Store," providing a platform for the buying and selling of unique and crafted products. This includes old vintage items, high-priced commodities, high-end vintage electronics, and modern short-term used products. Z Store is designed to integrate sustainability, craftsmanship, and technological sophistication, offering tailored solutions for passionate collectors, artisans, and individuals seeking distinctive items.

2) **To what extent is the system proposed for?**

Z Store aims to be a comprehensive online marketplace catering to a diverse range of users involved in the buying and selling of unique products. While initiated as a project, it has the potential for future scalability and expansion.

3) **Specify the Viewers/Public which is to be involved in the System?**

Artisans, Collectors, Individuals Seeking Unique Products, Second Product Buyers Market.

4) **List the modules in your system:**

- User Management Module: Overview: This module facilitates user registration, login, and account management functionalities. Users can customize their profiles and preferences to enhance their overall experience on the platform.

- Product Selling & Listing Module: These modules form the core of the platform, enabling users to sell and browse unique products.

- Tester Panel Module: This module is dedicated to testers responsible for verifying product authenticity and quality.

- Cart and Checkout Module: These modules streamline the user's purchasing process, from selecting items to completing the transaction.

- Admin Dashboard Module: This module serves as a centralized control center for administrators, offering tools to manage the platform efficiently.

- User Cart Page Module: This module provides users with a dedicated page to view and manage the items in their shopping cart.

- User Checkout Page Module: This module is focused on guiding users through the final steps of the purchasing process.

5) **Identify the users in your project?**

- Users: Individuals listing unique products on Z Store also if Users interested in purchasing distinctive items

- Administrators: Platform overseers managing listings, users, and testing

- Testers: Individuals responsible for product verification and quality control

6) Who owns the system?

Z-Store.

7) System is related to which firm/industry/organization?

The system is related to the online marketplace industry, specifically targeting users interested in unique and crafted products.

8) Details of the person contacted for data collection?

OLX Support, Craftsman

9) Questionnaire to collect details about the project?

- **Are there any significant costs associated with developing Z Store as part of the project work?**

    No, Z Store is developed as part of the project work, incurring minimal costs.

- **What is the cost of hardware and software required for Z Store?**

    All necessary resources, including hardware and software, are already available, making it a cost-effective project.

- **Are there any additional costs for operational expenses, such as maintenance or server hosting?**

    No, operational expenses are minimal, designed to operate within the scope of the project.

- **Is the project feasible within the limits of current technology?**

    Yes, Z Store is designed within the limits of current technology, ensuring a straightforward and feasible development process.

- **Technical issues raised during the investigation are:**

    The investigation did not uncover any major technical issues hindering the development of Z Store.

- **Can the technology be easily applied to current problems?**

    Yes, the technology can be easily applied to current problems, aligning with the goal of providing an accessible platform for unique product enthusiasts.

- **Does the technology have the capacity to handle the solution?**

    Yes, the technology has the capacity to handle the solution, considering the small-scale nature of the project and its focus on limited user interactions.

- **Is the required technology readily available and accessible to the individual for developing Z Store?**

    Yes, the required technology is readily available and accessible, with the team having

access to necessary development tools and resources.

- **Are the infrastructure requirements for Z Store, such as servers and hosting services, feasible and within the project's scope?**

  Yes, the infrastructure requirements for Z Store are feasible and within the project's scope, considering the small-scale nature of the platform.

- **Will users receive adequate support while using Z Store?**

  Absolutely, Z Store is committed to providing users with comprehensive support, ensuring a seamless and positive user experience.

- **Will users be exposed to any harmful elements or content while using Z Store?**

  Z Store has been meticulously designed to maintain a safe and secure environment, free from harmful elements, providing users with a worry-free space to explore and purchase unique products.

- **Does Z Store offer user-friendly features and an intuitive interface?**

  Yes, Z Store takes pride in its user-centric approach, offering a user-friendly interface with intuitive features, ensuring that users can effortlessly navigate and explore unique products.

- **Is there a mechanism in place for users to share their thoughts and suggestions?**

  Absolutely, Z Store actively encourages user feedback and suggestions, aiming to foster a vibrant community where users can readily share their thoughts and responses.

3.1 SYSTEM SPECIFICATION

3.2.1 Hardware Specification

Processor   - Intel Core i3

RAM   - 4GB

Hard disk   - 256GB

3.2.2 Software Specification

Front End   - HTML, CSS, JS, Bootstrap, Ajax

Back End   - Python-Django

Database   - MySQL

Client on PC   - Windows 7 and above.

Technologies used   - JS, HTML5, AJAX, J Query, PHP, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Python-Django

Django is a popular and powerful open-source web framework written in Python, designed to facilitate rapid development and maintainable web applications. It follows the Model-View-Template (MVT) architectural pattern, which is similar to the Model-View-Controller (MVC) pattern. Django provides a structured and efficient way to build web applications, offering several key components and features.

At its core, Django includes a robust Object-Relational Mapping (ORM) system that simplifies database interactions, allowing developers to work with Python objects instead of raw SQL queries. It also includes a URL dispatcher for mapping URLs to view functions, an automatic admin interface for managing application data, and a templating engine for creating dynamic and reusable user interfaces.

Django places a strong emphasis on security, with built-in features to protect against common web

vulnerabilities. It offers authentication and authorization systems, middleware support for global request and response processing, and compatibility with various databases.

The framework's scalability, extensibility, and a vibrant community of developers make it a popular choice for building web applications, from simple websites to complex, high-traffic platforms. Django's extensive documentation and ecosystem of third-party packages further enhance its appeal for web developers.

3.3.2 MySQL

SQLite is a self-contained and serverless relational database management system (RDBMS) known for its simplicity and efficiency. It stores the entire database in a single file, eliminating the need for a separate server process, which simplifies deployment. This makes SQLite an ideal choice for embedded systems, mobile applications, and small to medium-sized projects. Developers interact with the database directly through function calls, making it an embedded database well-suited for various applications, including mobile apps, desktop software, and web applications.

Despite its lightweight nature, SQLite is ACID-compliant, ensuring data integrity with support for transactions. It offers a wide range of data types, and its compatibility with multiple programming languages, including Python, C/C++, and Java, makes it accessible to a diverse developer community. SQLite is open-source, free, and known for its speed and efficiency, particularly for read-heavy workloads. While not intended for extremely high-concurrency or large-scale applications, SQLite excels in scenarios where simplicity, portability, and low resource consumption are key requirements.

It is commonly used as a local data store, cache, or embedded database within larger applications, contributing to its versatility and widespread adoption in software development.

CHAPTER 4
SYSTEM DESIGN

4.1 INTRODUCTION

The initial stage of developing any engineered product or system is the design phase, which involves a creative approach. A well-crafted design plays a critical role in ensuring the successful functioning of a system. Design is defined as the process of employing various techniques and principles to define a process or system in enough detail to enable its physical realization. This involves using different methods to describe a machine or system, explaining how it operates, in sufficient detail for its creation. In software development, design is a crucial step that is always present, regardless of the development approach. System design involves creating a blueprint for building a machine or product. Careful software design is essential to ensure optimal performance and accuracy. During the design phase, the focus shifts from the user to the programmers or those working with the database. The process of creating a system typically involves two key steps: Logical Design and Physical Design.

4.2 UML DIAGRAM

UML, which stands for Unified Modeling Language, is a standardized language used for specifying, visualizing, constructing, and documenting the elements of software systems. The Object Management Group (OMG) is responsible for the creation of UML, with the initial draft of the UML 1.0 specification presented to OMG in January 1997. Unlike common programming languages such as C++, Java, or COBOL, UML is not a programming language itself. Instead, it is a graphical language that serves as a tool for creating software blueprints.

UML is a versatile and general-purpose visual modeling language that facilitates the visualization, specification, construction, and documentation of software systems. While its primary use is in modeling software systems, UML's applications are not limited to this domain. It can also be employed to represent and understand processes in various contexts, including non-software scenarios like manufacturing unit processes.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case is a tool for understanding a system's requirements and organizing them, especially in the context of creating or using something like a product delivery website. These tools are represented using "use case" diagrams within the Unified Modeling Language, a standardized way of creating models for real-world things and systems.

A use case diagram consists of these main elements:

- The boundary, which delineates the system and distinguishes it from its surroundings.
- Actors, representing individuals or entities playing specific roles within the system.
- The interactions between different people or elements in specific scenarios or problems.
- The primary purpose of use case diagrams is to document a system's functional specifications. To create an effective use case diagram, certain guidelines must be followed:
- Providing clear and meaningful names for use cases and actors.
- Ensuring that the relationships and dependencies are well-defined.
- Including only necessary relationships for the diagram's clarity.
- Utilizing explanatory notes when needed to clarify essential details.

Fig 1: Use Case Diagram

4.2.2   SEQUENCE DIAGRAM

A sequence diagram illustrates the specific order in which objects interact with each other, showcasing the sequential flow of events. This type of diagram is also known as event diagrams or event scenarios. Sequence diagrams serve the purpose of elucidating how various components of a system collaborate and the precise sequence in which these actions occur. These diagrams find frequent application among business professionals and software developers, aiding in the understanding and depiction of requirements for both new and existing systems.

Sequence Diagram Notations:

I.   Actors - Within a UML diagram, actors represent individuals who utilize the system and its components. Actors are not depicted within the UML diagram as they exist outside the system being modeled. They serve as role-players in a story, encompassing people and external entities. In a UML diagram, actors are represented by simple stick figures. It's possible to depict multiple individuals in a diagram that portrays the sequential progression of events.

II.   Lifelines - In a sequence diagram, each element is presented as a lifeline, with lifeline components positioned at the top of the diagram.

III.   Messages - Communication between objects is achieved through the exchange of messages, with messages being arranged sequentially on the lifeline. Arrows are employed to represent messages, forming the core structure of a sequence diagram.

IV.   Guards - Within the UML, guards are employed to denote various conditions. They are used to restrict messages in the event that specific conditions are met, providing software developers with insights into the rules governing a system or process.
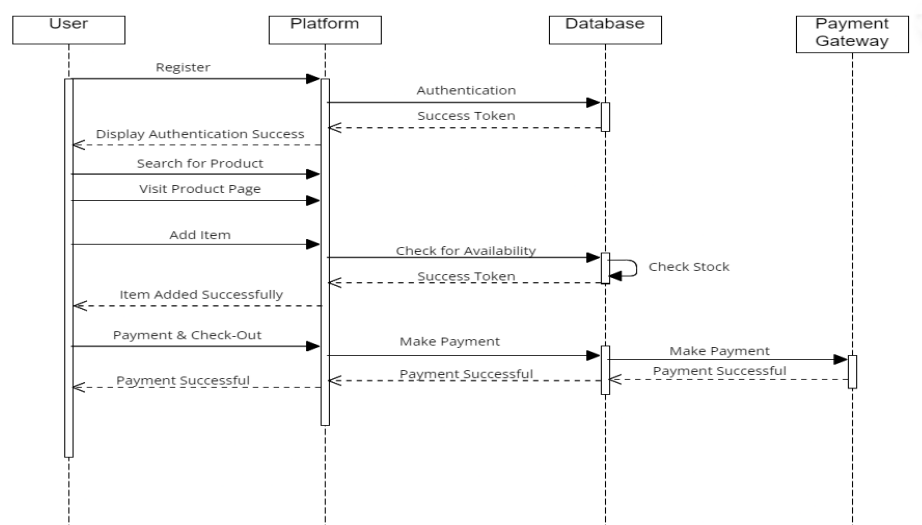
Fig 2: Sequence Diagram

4.2.3 State Chart Diagram

A state machine diagram, also known as a state chart, visually represents the various states an object undergoes within a system and the sequence in which these states are traversed. It serves as a record of the system's behavior, illustrating the collaborative functioning of a group of entities, whether it's a team, a collection of students, a large assembly, or an entire organization. State machine diagrams are a valuable method for depicting the interactions of diverse components within a system, outlining how objects evolve in response to events and elucidating the diverse conditions that each entity or component can inhabit.

Notations within a state machine diagram encompass:

- Initial state: Symbolized by a black circle, this signifies the commencement of a process.
- Final state: Representing the conclusion of a process, this is denoted by a filled circle within another circle.
- Decision box: Shaped like a diamond, it aids in decision-making by considering the evaluation of a guard.
- Transition: Whenever a change in authority or state occurs due to an event, it is termed a transition. Transitions are depicted as arrows with labels indicating the triggering event.
- State box: It portrays the condition or state of an element within a group at a specific moment. These are typically represented by rectangles with rounded corners.
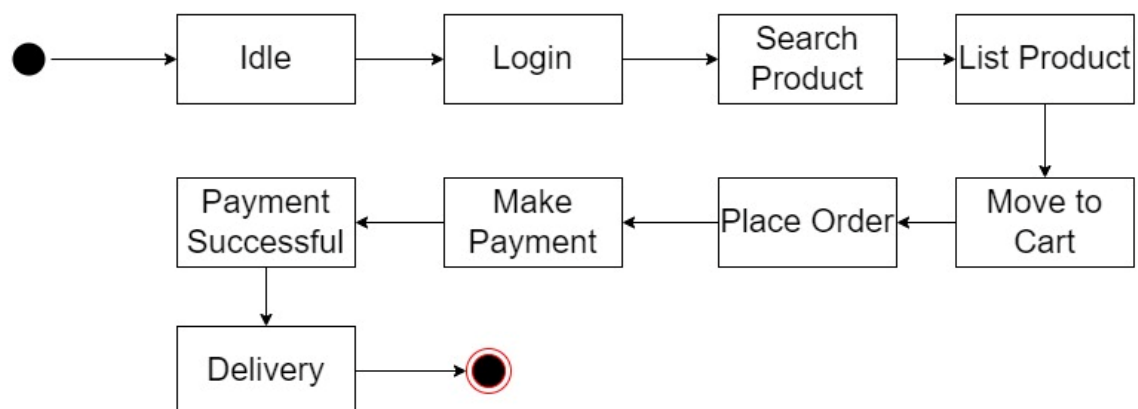
Fig 3: State Chart Diagram

4.2.4 Activity Diagram

An activity diagram is a visual representation of how events unfold simultaneously or sequentially. It aids in understanding the flow of activities, emphasizing the progression from one task to the next. Activity diagrams focus on the order in which tasks occur and can depict various types of flows, including sequential, parallel, and alternative paths. To facilitate these flows, activity diagrams incorporate elements such as forks and join nodes, aligning with the concept of illustrating the functioning of a system in a specific manner.

Key Components of an Activity Diagram include:

a) Activities: Activities group behaviors into one or more actions, forming a network of interconnected steps. Lines between these actions outline the step-by-step sequence of events. Actions encompass tasks, controlling elements, and resources utilized in the process.

b) Activity Partition/Swim Lane: Swim lanes are used to categorize similar tasks into rows or columns, enhancing modularity in the activity diagram. They can be arranged vertically or horizontally, though they are not mandatory for every activity diagram.

c) Forks: Fork nodes enable the simultaneous execution of different segments of a task. They represent a point where one input transforms into multiple outputs, resembling the diverse factors influencing a decision.

d) Join Nodes: Join nodes are distinct from fork nodes and employ a Logical AND operation to ensure that all incoming data flows converge to a single point, promoting synchronization.

Notations in an Activity Diagram include:

i.   Notations in an Activity Diagram include:

ii.   Initial State: Depicting the beginning or first step in the process.

iii. Final State: Signifying the completion of all actions with no further progress.

iv.   Decision Box: Ensuring that activities follow a specific path.

v.   Action Box: Representing the specific tasks or actions to be performed in the process.
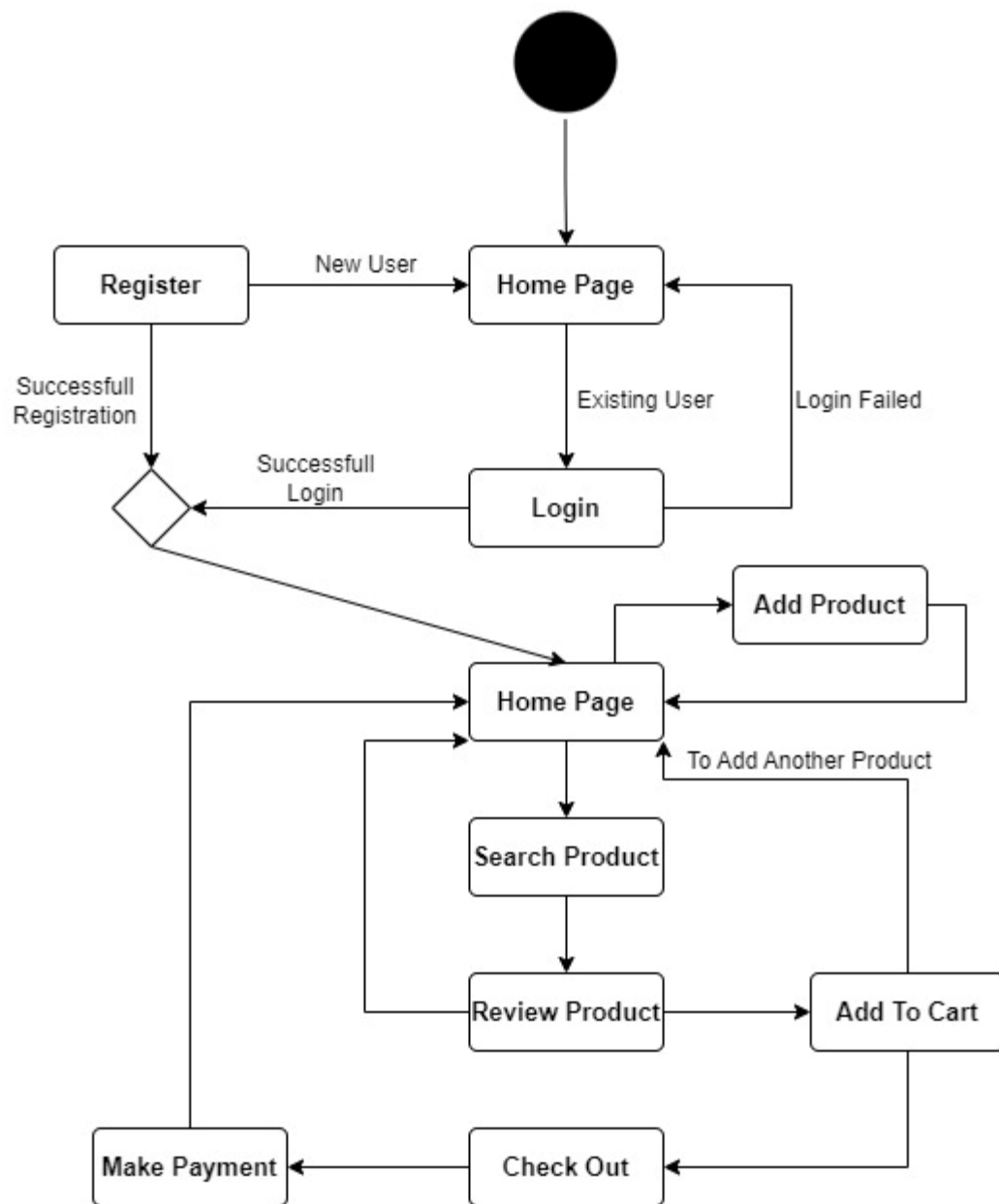
Fig 4: Activity Diagram

4.2.5 Class Diagram

A class diagram serves as a static blueprint for an application, illustrating its components and their relationships when the system is in a dormant state. It provides insights into the system's structure, showcasing the various elements it comprises and how they interact. In essence, a class diagram acts as a visual guide for software development, aiding in the creation of functional applications.

Key Aspects of a Class Diagram:

- System Overview: A class diagram offers a high-level representation of a software system, presenting its constituent parts, associations, and collaborative dynamics. It serves as an organizational framework for elements such as names, attributes, and methods, simplifying the software development process.

- Structural Visualization: The class diagram is a structural diagram that combines classes, associations, and constraints to define the system's architecture.

Components of a Class Diagram:

The class diagram comprises three primary sections:

- Upper Section: This top segment features the class name, representing a group of objects that share common attributes, behaviors, and roles. Guidelines for displaying groups of objects include capitalizing the initial letter of the class name, positioning it in the center, using bold lettering, and employing slanted writing style for abstract class titles.

- Middle Section: In this part, the class's attributes are detailed, including their visibility indicators, denoted as public (+), private (-), protected (#), or package (~).

- Lower Section: The lower section elaborates on the class's methods or operations, presented in a list format with each method on a separate line. It outlines how the class interacts with data.

In UML, relationships within a class diagram fall into three categories:

- Dependency: Signifying the influence of one element's changes on another.

- Generalization: Representing a hierarchical relationship where one class acts as a parent, and another serves as its child.

- Association: Indicating connections between elements.

- Multiplicity: Defining constraints on the number of instances allowed to possess specific characteristics, with one being the default value when not specified.

- Aggregation: An aggregation is a group that is a part of a relationship called association.

- Composition: "Composition" is like a smaller part of "aggregation.". This describes how a parent and child need each other, so if one is taken away, the other won't work anymore.
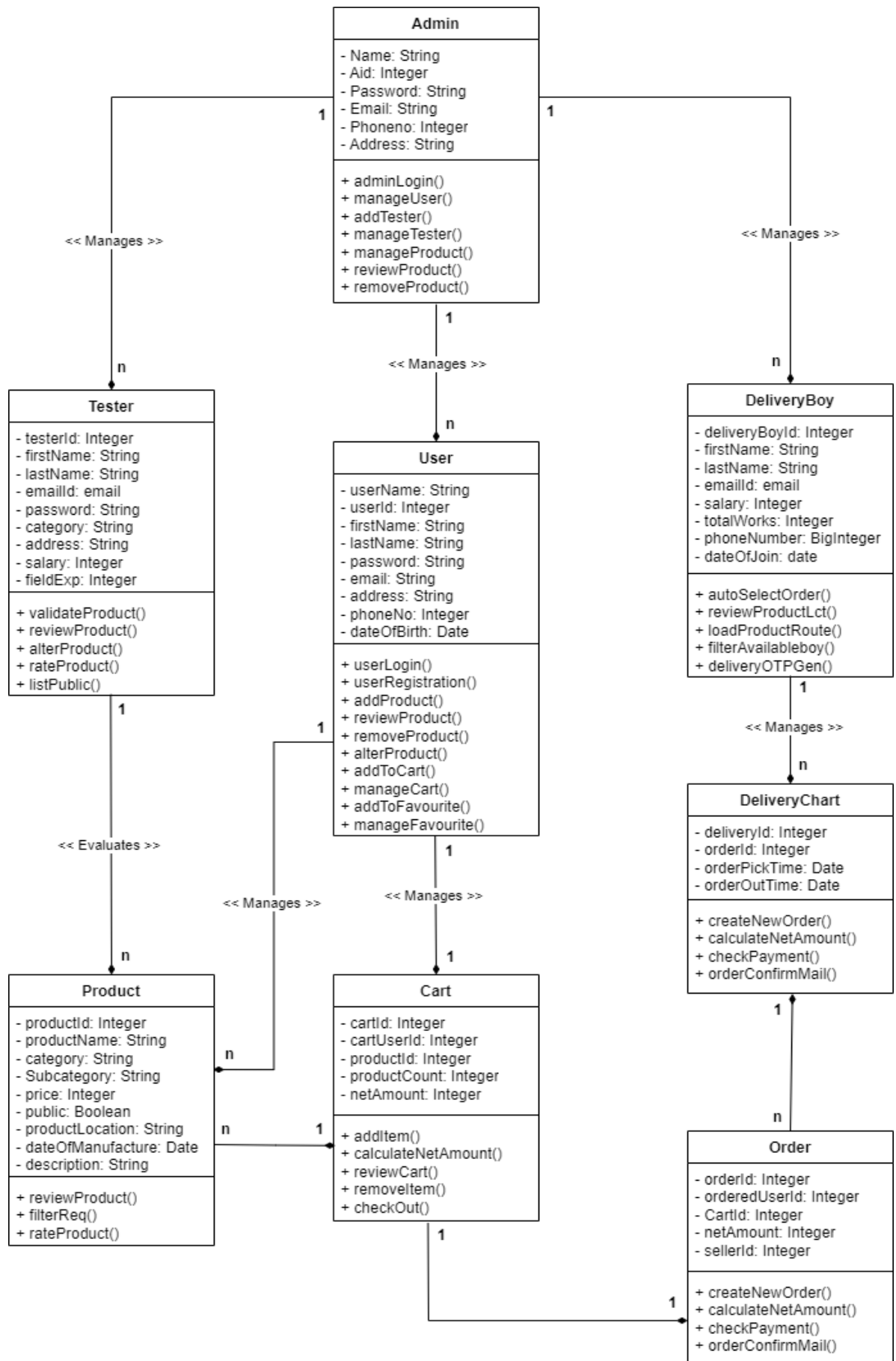
Fig 5: Class Diagram

4.5.6 Object Diagram

Object diagrams are derived from class diagrams and rely on them to provide a visual representation. They offer an illustration of a collection of objects related to a particular class. Object diagrams provide a snapshot of objects in an object-oriented system at a specific point in time.

Object diagrams and class diagrams share similarities, but they also have distinctions. Class diagrams are more generalized and do not portray specific objects. This abstraction in class diagrams simplifies the comprehension of a system's functionality and structure.
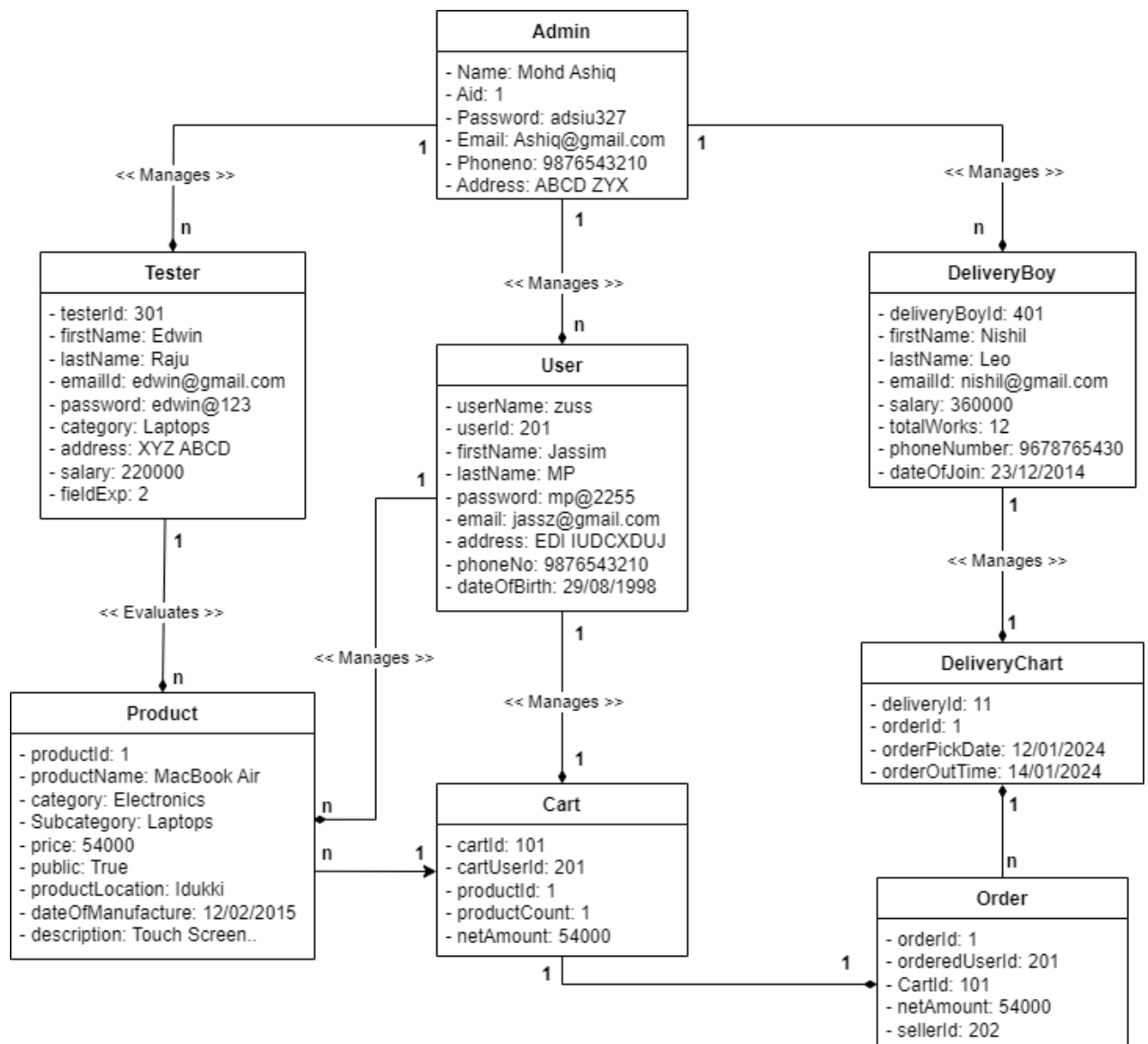
Fig 6: Object Diagram

4.5.7 Component Diagram

A component diagram serves the purpose of breaking down a complex system that utilizes objects into more manageable segments. It offers a visual representation of the system, showcasing its internal components such as programs, documents, and tools within the nodes.

This diagram elucidates the connections and organization of elements within a system, resulting in the creation of a usable system.

In the context of a component diagram, a component refers to a system part that can be modified and operates independently. It retains the secrecy of its internal operations and requires a specific method to execute a task, resembling a concealed box that functions only when operated correctly.

Notation for a Component Diagram includes:

  - A component

  - A node
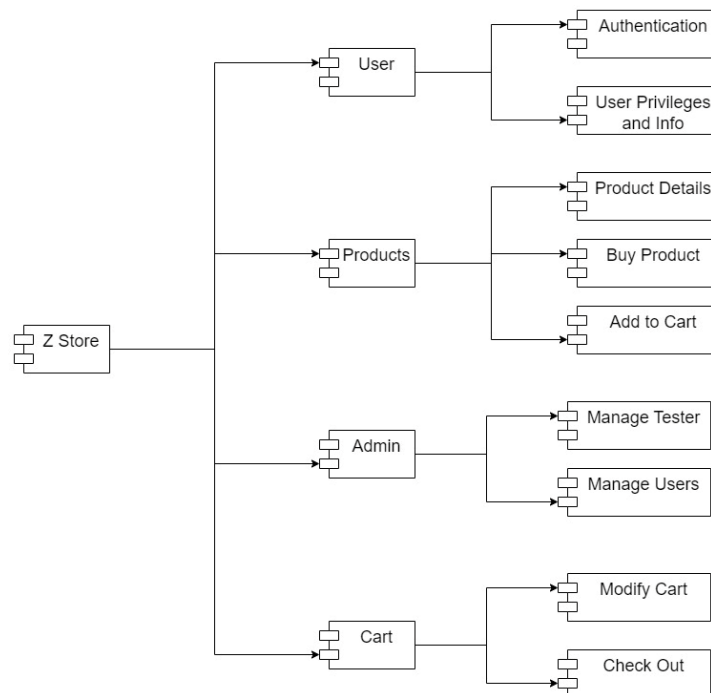
Fig 7: Component Diagram

4.2.8 Deployment Diagram

A deployment diagram provides a visual representation of how software is positioned on physical computers or servers. It depicts the static view of the system, emphasizing the arrangement of nodes and their connections.

This type of diagram delves into the process of placing programs on computers, elucidating how software is constructed to align with the physical computer system.

Notations in a Deployment Diagram include:

- A component
- An artifact
- An interface
- A node
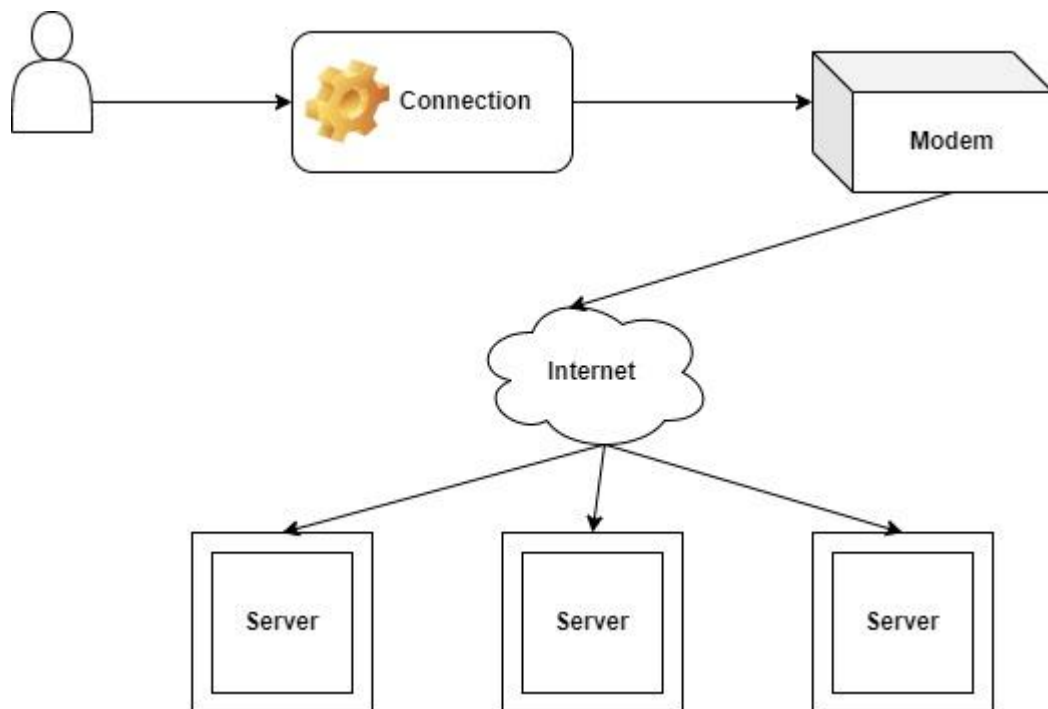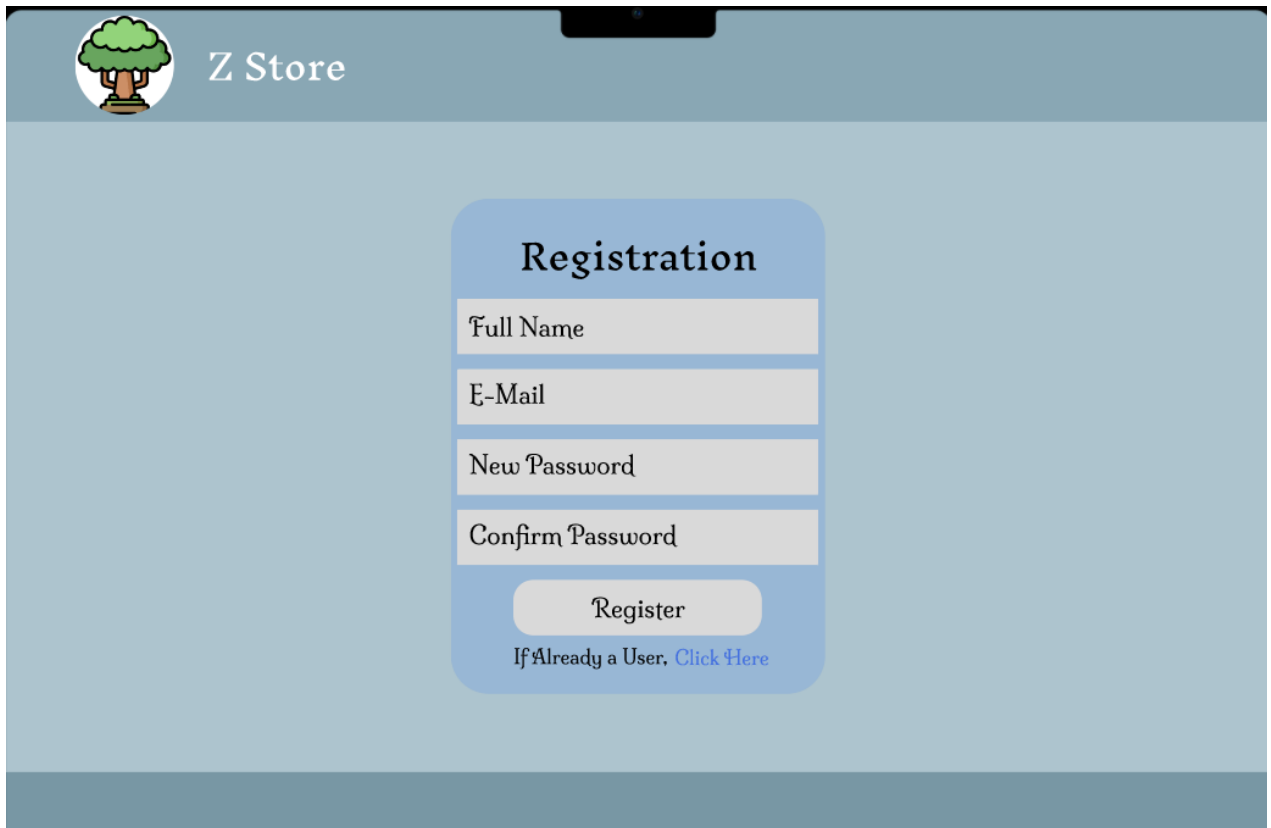
Fig 8: Deployment Diagram

4.3 USER INTERFACE DESIGN USING FIGMA

Form Name: Registration Page

Form Name: Login Page

Form Name: Index Page

Form Name: User Settings Page

4.4 DATABASE DESIGN

4.4.1 Relational Database Management System (RDBMS)

A database is a system designed to store and facilitate the retrieval and utilization of information. The integrity and security of the data within a database are of paramount importance. Creating a database involves two essential steps. Initially, it is crucial to understand the user's requirements and establish a database structure that aligns with their needs. The first step entails devising an organizational plan for the information, which is not reliant on any particular computer program.

Subsequently, this plan is employed to craft a design tailored to the specific computer program that will be employed to construct the database system. This phase is centered on determining how data will be stored within the chosen computer program.

Key aspects of database development include:

- Data Integrity: Ensuring the accuracy and consistency of data stored within the database.
- Data Independence: The ability to modify the data storage structure without affecting the application's access to the data.

4.4.2 Normalization

A way of showing a database as a group of connections is called a relational model. Every relationship is like a chart of information or a group of data. In the formal way of talking about tables, a row is called a tuple, a column header is called an attribute, and the table is called a relation. A bunch of tables with special names make up a relational database. A row in a story shows a group of things that are connected.

Relations, Domains & Attributes

Tables serve as containers for organized and related information. Within a table, the rows are referred to as tuples, and a tuple is essentially an ordered list containing n items. Columns in a table are synonymous with characteristics, each representing a specific aspect of the data.

In a relational database, tables are interconnected, ensuring coherence and meaningful associations between different sets of data. This relational structure forms the foundation of a well-structured database.

A domain can be thought of as a collection of fundamental values with a shared source or data type. Naming domains is a helpful practice as it aids in comprehending the significance of the values they

encompass. It's essential to recognize that values within a domain are indivisible.

- Table relationships are established using keys, with the primary key and foreign key being the primary types. Entity Integrity and Referential Integrity are fundamental principles that guide these relationships.
- Entity Integrity mandates that no Primary Key should contain null values, reinforcing the unique and integral nature of primary keys in database relationships.

Normalization is the process of organizing data to ensure its efficient storage and to minimize redundancy while maintaining accuracy and reliability. This approach involves eliminating unnecessary columns and breaking down large tables into smaller, more manageable parts. Normalization helps prevent errors when adding, removing, or modifying data. In data modeling, two key concepts are integral to its normal form: keys and relationships.

Keys serve as unique identifiers to distinguish one row from all others in a table. There are two main types of keys: the primary key, which groups similar records within a table, and the foreign key, which acts as a special code to identify specific records in another table.

1. **First Normal Form (1NF):** First Normal Form (1NF): This rule dictates that an attribute can only contain a single value that cannot be further divided. Each value in a tuple must match the attribute's type. In 1NF, tables cannot contain other tables or have tables as part of their data. This form ensures that values are indivisible and represent a single entity. Achieving 1NF often involves organizing data into separate tables, with each table having a designated key based on project requirements. New relationships are established for various data categories or related groups, eliminating redundant information. A relationship adhering to primary key constraints alone is termed the first normal form.

rollno	name	course	age
1	Rahul	c/c++	22
2	Harsh	java	18
3	Sahil	c/c++	23
4	Adam	c/c++	22
5	Lisa	java	24
6	James	c/c++	19
NULL	NULL	NULL	NULL

rollno	name	course	age
1	Rahul	c	22
1	Rahul	c++	22
2	Harsh	java	18
3	Sahil	c	23
3	Sahil	c++	23
4	Adam	c	22
4	Adam	c++	22
5	Lisa	java	24

2.Second Normal Form (2NF): In simpler terms, 2NF stipulates that no additional information should be associated with only part of the primary information used for data organization. This

involves breaking down the information and creating separate groupings for each part along with its related details. It's essential to maintain a connection between the original primary key and any data dependent on it. This step helps remove data that relies on only a portion of the key. When a set of information has a primary means of identifying each item, and all other details depend solely on that primary means, it is considered to be in the second normal form.

	cust_id	storeid	store_location
▶	1	D1	Toronto
	2	D3	Miami
	3	T1	California
	4	F2	Florida
	5	H3	Texas

	cust_id	storeid
▶	1	D1
	2	D3
	3	T1
	4	F2
	5	H3

	storeid	store_location
▶	D1	Toronto
	D3	Miami
	T1	California
	F2	Florida
	H3	Texas

3.Third Normal Form (3NF): is a critical concept in database normalization, aiming to achieve table independence and control over attributes. In 3NF, a table should not contain columns that depend on other non-key columns, ensuring that each column is functionally dependent only on the primary key. It breaks down relationships involving non-key attributes to eliminate dependencies on attributes not part of the primary key. To meet the criteria for 3NF, data must already be in the Second Normal Form (2NF), and non-key attributes should not rely on other non-key attributes within the same table, avoiding transitive dependencies. This process enhances data integrity, reduces redundancy, and optimizes database structure and organization.

	stu_id	name	subid	sub	address
▶	1	Arun	11	SQL	Delhi
	2	Varun	12	Java	Bangalore
	3	Harsh	13	C++	Delhi
	4	Keshav	12	Java	Kochi

	stu_id	name	subid	address
▶	1	Arun	11	Delhi
	2	Varun	12	Bangalore
	3	Harsh	13	Delhi
	4	Keshav	12	Kochi

	subid	subject
▶	11	SQL
	12	java
	13	C++
	12	Java

4.4.3 Sanitization

Django incorporates various in-built mechanisms for data sanitization. To begin with, it provides a diverse range of field types that come with automatic validation and sanitization capabilities. For instance, the CharField performs automatic validation and cleaning of text input, while the EmailField ensures that email addresses adhere to the correct format. These field types serve as protective measures to prevent the storage of malicious or invalid data in the database.

Moreover, Django strongly advocates for the utilization of form validation to sanitize user input. Django's forms are equipped with predefined validation methods and validators that can be applied to form fields. These validators execute a range of checks and cleansing operations, such as confirming that numerical values fall within specified ranges or verifying that uploaded files adhere to specific formats. These combined features in Django promote data integrity and security, making

it a reliable choice for web application development.

4.4.4 Indexing

The index keeps track of a certain piece of information or group of information, arranged in order of its value. Arranging the index entries helps find things quickly and easily that match exactly or are within a certain range. Indexes make it easy to find information in a database without having to search through every record every time the database is used. An index is like a roadmap for finding information in a database. It helps you look up data quickly and also makes it easy to find records that are in a certain order. It can be based on one or more columns in the table.

4.5 TABLE DESIGN

1.Tbl_userAuth

Primary key: Email.

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Email	VARCHAR(100)	PRIMARY KEY	Registered User Email
2	FirstName	VARCHAR(100)	NOT NULL	User's First Name
3	LastName	VARCHAR(100)	NOT NULL	User's Last Name
4	Username	VARCHAR(100)	NOT NULL	User's Username
5	UserRole	VARCHAR(100)	NOT NULL	User's Role
6	Password	STRING	NOT NULL	Hash Value of User's Password
7	Is_Active	BOOLEAN	NOT NULL	User is Currently Active or Not
8	Last_Login	DATE AND TIME	NOT NULL	Date/Time of Last user login

2. Tbl_userLoca

Primary key: Location_Id

Foreign key: Email references table **Tbl_userAuth**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Location_Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	Country	VARCHAR(100)	NOT NULL	User's Country
4	State	VARCHAR(100)	NOT NULL	User's State
5	City	VARCHAR(100)	NOT NULL	User's City
6	Street	TEXT	NOT NULL	User's Street Name
7	Address	TEXT	NOT NULL	User's Address
8	Is_Default	BOOLEAN	NOT NULL	User's Default Location or Not

3. Tbl_userData

Primary key: Data_Id

Foreign key: Email references table **Tbl_userAuth**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Data_Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	Gender	VARCHAR(100)	NOT NULL	User's Gender
4	Phone_No	VARCHAR(100)	NOT NULL	User's Phone Number
5	DateOfBirth	DATE FIELD	NOT NULL	User's Date of Birth

4. Tbl_Tester_Data

Primary key: Tester_Id

Foreign key: Email references table **Tbl_userAuth**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Tester_Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	TesterSalary	VARCHAR(100)	NOT NULL	Tester's Salary
4	TesterRating	VARCHAR(100)	NOT NULL	Tester's Rating
5	TesterCategory	VARCHAR(100)	NOT NULL	Tester's Working category
6	DateOfJoin	DATE AND TIME	NOT NULL	Tester's Date of Join

5. Tbl_Products

Primary key: Product_Id

Foreign key: Email references table **Tbl_userAuth,** Tester_Id references table **Tbl_Tester_Data**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Product_Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	ProductName	VARCHAR(100)	NOT NULL	Name of the Product
4	Price	INTEGER	NOT NULL	Price of the Product
5	Quantity	INTEGER	NOT NULL	Quantity of the Product
6	Availability	TEXT	NOT NULL	Available Stock of the Product
7	Tester_Id	INTEGER	FOREIGN KEY	Foreign Key to Tester_Data table
8	Description	TEXT	NOT NULL	Description on the Product

6. Tbl_UserCart

Primary key: Cart_Id

Foreign key: Email references table **Tbl_userAuth,** Product_Id references table **Tbl_Products.**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Cart_Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	Product_Id	INTEGER	FOREIGN KEY	Foreign Key to table Products
4	Quantity	INTEGER	NOT NULL	Total Quantities of the Product
5	SumOfPrice	INTEGER	NOT NULL	Total Price of the Product

7. Tbl_CartItems

Primary Key: Id.

Foreign Key: UserCart reference to **Tbl_UserCart**, Product_Id Reference to

Tbl_Products.

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	UserCart	VARCHAR(100)	FOREIGN KEY	Foreign Key to table UserAuth
3	Product_Id	INTEGER	FOREIGN KEY	Foreign Key to table Products
4	Quantity	INTEGER	NOT NULL	Total Quantities of the Product
5	SumOfPrice	INTEGER	NOT NULL	Total Price of the Product

8. Tbl_UserOrders

Primary Key: Id

Foreign Key: UserAuth reference to **Tbl_userAuth**.

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	UserAuth	INTEGER	FOREIGN KEY	Foreign Key to table UserAuth
3	NetAmount	INTEGER	NOT NULL	Net Amount of User's Order

9. Tbl_OrderedItems

Primary Key: Id.

Foreign Key: UserOrders Reference to **Tbl_UserOrders**, Tbl_Product Reference to **Tbl_Products**,
          Email Reference to **Tbl_DeliveryAgent.**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	UserOrders	INTEGER	FOREIGN KEY	Foreign Key to table Tbl_UserOrders
3	Tbl_Product	INTEGER	FOREIGN KEY	Foreign Key to Tbl_Product
4	Quantity	INTEGER	NOT NULL	Net Quantity of a Product
5	CurrentLocation	VARCHAR(255)	NOT NULL	Gives current location of product
6	DeliveryByDate	DATE	NOT NULL	The date of delivery by
7	DeliveryByTime	TIME	NOT NULL	The time of delivery by

8	Email	INTEGER	FOREIGN KEY	Foreign key reference to Tbl_DeliveryAgent

11. Tbl_DeliverAgent

Primary Key: Id

Foreign Key: Email references **Tbl_UserAuth.**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	INTEGER	FOREIGN KEY	Foreign Key to table UserAuth
3	CurrentLocation	INTEGER	NOT NULL	Delivery Agents Current Location

12. Tbl_DeliveryData

Primary Key: Id

Foreign Key: Email references **Tbl_UserAuth,** Product_Id references **Tbl_Products.**

No:	Field name	Datatype (Size)	Key Constraints	Description of the field
1	Id	INTEGER	PRIMARY KEY	Primary Key of this table
2	Email	INTEGER	FOREIGN KEY	Foreign Key to table UserAuth
3	Product_Id	INTEGER	FOREIGN KEY	Foreign Key to Tbl_Products
4	DeliverySuccess	BOOLEAN	NOT NULL	Boolean Field for Success

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software testing is an essential procedure employed to verify if a computer program functions as intended. It is conducted to ensure that the software performs its designated tasks accurately and complies with the prescribed requirements and standards. Validation is the process of inspecting and assessing software to confirm its adherence to the specified criteria. Software testing is a method for assessing the performance of a program, often in conjunction with techniques like code inspection and program walkthroughs. Validation ensures that the software aligns with the user's expectations and requirements.

Several principles and objectives guide the process of software testing, including:

1. Testing is the practice of executing a program with the primary aim of identifying errors.
2. An effective test case is one that has a high likelihood of uncovering previously undiscovered errors.
3. A successful test is one that exposes previously undiscovered errors.

When a test case operates effectively and accomplishes its objectives, it can detect flaws within the software. This demonstrates that the computer program is functioning as intended and is performing well. The process of evaluating a computer program encompasses three primary aspects:

1. Correctness assessment
2. Evaluation of implementation efficiency
3. Examination of computational complexity

5.2 TEST PLAN

A test plan serves as a comprehensive set of instructions for conducting various types of tests. It can be likened to a map that outlines the steps to follow when evaluating a computer program. Software developers create instructions for both using and organizing the necessary information for the program's proper functionality. They ensure that each component of the program performs its intended functions. To ensure the thorough testing of the software, a group known as the ITG (Information Technology Group) is responsible for verifying its functionality, instead of relying solely on the software's creators for testing.

The objectives of testing should be clearly defined and measurable. A well-structured test plan should encompass details about the frequency of failures, the associated repair costs.

- Unit testing
- Integration testing
- Data validation testing
- Output testing

5.2.1   Unit Testing

Unit testing checks the smallest part of a software design - the software component or module. Testing important control paths within a module using the design guide to find errors. This means how difficult the tests are for each small part of a program and what parts of the program haven't been tested yet. Unit testing is a type of testing that looks at how the code works inside and can be done at the same time for different parts of the program.

Before starting any other test, we need to check if the data flows correctly between different parts of the computer program. If the information doesn't move in and out correctly, all other checks are pointless. When designing something, it's important to think about what could go wrong and make a plan for how to deal with those problems. This can mean redirecting the process or stopping it completely.

The Sell-Soft System was tested by looking at each part by itself and trying different tests on it. Some mistakes in the design of the modules were discovered and then fixed. After writing the instructions for different parts, each part is checked and tried out separately. We got rid of extra code and made sure everything works the way it should.

5.2.2 Integration Testing

Integration testing is a critical process in software development that involves constructing a program while simultaneously identifying errors in the interaction between different program components. The primary objective is to utilize tested individual parts and assemble them into a program according to the initial plan. This comprehensive testing approach assesses the entire program to ensure its proper and correct functionality.

As issues are identified and resolved during integration testing, it's not uncommon for new problems to surface, leading to an ongoing cycle of testing and refinement. After each individual component of the system is thoroughly examined, these components are integrated to ensure they function harmoniously. Additionally, efforts are made to standardize all programs to ensure uniformity rather than having disparate versions.

5.2.3 Validation Testing or System Testing

The final phase of testing involves a comprehensive examination of the entire system to ensure the correct interaction of various components, including different types of instructions and building blocks. This testing approach is referred to as Black Box testing or System testing.

Black Box testing is a method employed to determine if the software functions as intended. It assists software engineers in identifying all program issues by employing diverse input types. Black Box testing encompasses the assessment of errors in functions, interfaces, data access, performance, as well as initialization and termination processes. It is a vital technique to verify that the software meets its intended requirements and performs its functions correctly.

5.2.4   Output Testing or User Acceptance Testing

System testing is conducted to assess user satisfaction and alignment with the company's requirements. During the development or update of a computer program, it's essential to maintain a connection with the end-users. This connection is established through the following elements:

- Input Screen Designs.

- Output Screen Designs.

To perform system testing, various types of data are utilized. The preparation of test data plays a crucial role in this phase. Once the test data is gathered, it is used to evaluate the system under investigation. When issues are identified during this testing, they are addressed by following established procedures. Records of these corrections are maintained for future reference and improvement. This process ensures that the system functions effectively and meets the needs of both users and the organization.

5.2.5 Automation Testing

Automated testing is a method employed to verify that software functions correctly and complies with established standards before it is put into official use. This type of testing relies on written instructions that are executed by testing tools. Specifically, UI automation testing involves the use of specialized tools to automate the testing process. Instead of relying on manual interactions where individuals click through the application to ensure its proper functioning, scripts are created to automate these tests for various scenarios. Automating testing is particularly valuable when it is necessary to conduct the same test across multiple computers simultaneously, streamlining the testing process and ensuring consistency.

5.2.6   Selenium Testing

Selenium is a valuable and free tool designed for automating website testing. It plays a crucial role for web developers as it simplifies the testing process. Selenium automation testing refers to the practice of using Selenium for this purpose. Selenium isn't just a single tool; it's a collection of tools, each serving distinct functions in the realm of automation testing. Manual

testing is a necessary aspect of application development, but it can be monotonous and repetitive. To alleviate these challenges, Jason Huggins, an employee at ThoughtWorks, devised a method for automating testing procedures, replacing manual tasks. He initially created a tool named the JavaScriptTestRunner to facilitate automated website testing, and in 2004, it was rebranded as Selenium.

Test Case 1 – User Login

Code

```java
package testdefinitions;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class login {
        WebDriver driver;
        @Given("browser is open")
        public void browser_is_open() {
                driver = CommonDriver.getDriver();
                driver.manage().window().maximize();
        }

        @And("user is on login page")
        public void user_is_on_login_page() {
                driver.get("http://127.0.0.1:8000/signin");
        }

        @When("user enters username and password")
        public void user_enters_username_and_password() {
                WebElement username_input =
driver.findElement(By.id("exampleInputEmail1"));
                WebElement passwordInput =
driver.findElement(By.id("exampleInputPassword1"));

                username_input.sendKeys("ashiq");
                passwordInput.sendKeys("abcd@123");
        }

        @And("User clicks on login")
        public void user_clicks_on_login() {
                WebElement loginButton =
driver.findElement(By.xpath("//input[@type='submit']"));
            loginButton.click();
        }

        @Then("user is navigated to the home page")
        public void user_is_navigated_to_the_home_page() {
        }
}
```

Screenshot

```
Scenario: Check login is succesfull with valid credentials # src/test/resources/Features/login.feature:3
  Given browser is open                                    # testdefinitions.login.browser_is_open()
  And user is on login page                                # testdefinitions.login.user_is_on_login_page(
  When user enters username and password                   # testdefinitions.login.user_enters_username_a
  And User clicks on login                                 # testdefinitions.login.user_clicks_on_login()
  Then user is navigated to the home page                  # testdefinitions.login.user_is_navigated_to_t

1 Scenarios (1 passed)
5 Steps (5 passed)
0m6.422s
```

Test Report - 1

Test Case 1

Project Name: Z Store					
Login Test Case					
Test Case ID: Test_1			Test Designed By: Muhamed Ashiq		
Test Priority(Low/Medium/High):			Test Designed Date: 02/ 04/ 2024		
Module Name: Login Module			Test Executed By: Ms. Ankitha Philip		
Test Title : User Login with Valid data			Test Execution Date: 02 / 04/ 2024		
Description: User is able to Login using valid username and password					
Pre-Condition :User has valid username and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Navigate to login page		Login form should be displayed	Login form is displayed	Pass
2	Provide valid username	Username: ashiq	User Should be able to login	User Logs in	Pass
3	Provide valid password	Password: abcd@123			
4	Click on login button				
Post-Condition: User is validated with database and successfully logs into account. The Account session details are logged in database.					

Test Case 2: User Filtering Products

Code

```java
package testdefinitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.ui.Select;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class function1 {

        WebDriver driver = CommonDriver.getDriver();

        @Given("user is on the homepage")
        public void user_is_on_the_homepage() {

        }

        @When("user selects a main category from the filter options")
        public void user_selects_a_main_category_from_the_filter_options() {
                WebElement selectCat = driver.findElement(By.id("mainCategory"));
                Select dropdown = new Select(selectCat);
                dropdown.selectByVisibleText("Electronics");
                WebElement selectSubCat = driver.findElement(By.id("category"));
                Select dropdown2 = new Select(selectSubCat);
                dropdown2.selectByVisibleText("Laptops");
        }

        @And("user submits the filter form")
        public void user_submits_the_filter_form() {
                WebElement filterForm = driver.findElement(By.id("filterForm"));
                filterForm.submit();
        }

        @Then("the displayed products should belong to the selected main category")
        public void
the_displayed_products_should_belong_to_the_selected_main_category() {
                driver.quit();
        }
}
```

Screenshot

```
Scenario: User filters products by main category                              # src/test/resources/Features/f
  Given browser is open                                                       # testdefinitions.login.browser
  And user is on login page                                                   # testdefinitions.login.user_is
  When user enters username and password                                      # testdefinitions.login.user_en
  And User clicks on login                                                    # testdefinitions.login.user_cl
  Then user is navigated to the home page                                     # testdefinitions.login.user_is
  Given user is on the homepage                                               # testdefinitions.function1.use
  When user selects a main category from the filter options                   # testdefinitions.function1.use
  And user submits the filter form                                            # testdefinitions.function1.use
  Then the displayed products should belong to the selected main category     # testdefinitions.function1.the

1 Scenarios (1 passed)
9 Steps (9 passed)
0m9.695s
```

Test report - 2

Test Case 2					
Project Name: Z Store					
User Products Filtering					
Test Case ID: Test_2			**Test Designed By: Muhamed Ashiq**		
Test Priority(Low/Medium/High):			**Test Designed Date: 02/ 04/ 2024**		
Module Name: **Products Module**			**Test Executed By: Ms. Ankitha Philip**		
Test Title: Products Filtering			**Test Execution Date: 02 / 04/ 2024**		
Description: User is able to filter Products					
Pre-Condition :User is able to filter Products					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	Navigate to Index page		User is Navigated to Index Page	Index Page displayed	Pass
2	Provide Main Category Filtering Selection	Main Category: Electronics	Products in the Index Page will be Filtered	Products are filtered	Pass
3	Provide Sub Category Filtering Selection	Sub Category: Laptops			
4	Click on Filter button				
Post-Condition: User is able to filter the Products. Hence the Functionality Works					

Test Case 3: User Updating Profile Settings

Code

```java
package testdefinitions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

import io.cucumber.java.en.And;
import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class function2 {

    WebDriver driver = CommonDriver.getDriver();

    @Given("user is on the settings page")
    public void user_is_on_the_settings_page() {
        driver.get("http://127.0.0.1:8000/settings");
    }

    @When("user enters a new full name {string}")
    public void user_enters_a_new_full_name(String string) {
        WebElement fullname_input = driver.findElement(By.id("fullname"));
        fullname_input.clear();
        fullname_input.sendKeys("Jeevarag");
    }

    @And("user submits the update form")
    public void user_submits_the_update_form() throws InterruptedException {
        WebElement submitButton =
driver.findElement(By.id("updateCuccu"));
        TimeUnit.SECONDS.sleep(2);
      submitButton.click();
    }

    @Then("the full name should be updated to {string}")
    public void the_full_name_should_be_updated_to(String string) {

    }
}
```

Screenshot

```
Scenario: User updates their full name                      # src/test/resources/Features/function2.feature
  Given browser is open                                     # testdefinitions.login.browser_is_open()
  And user is on login page                                 # testdefinitions.login.user_is_on_login_page()
  When user enters username and password                    # testdefinitions.login.user_enters_username_an
  And User clicks on login                                  # testdefinitions.login.user_clicks_on_login()
  Then user is navigated to the home page                   # testdefinitions.login.user_is_navigated_to_th
  Given user is on the settings page                        # testdefinitions.function2.user_is_on_the_sett
  When user enters a new full name "New Full Name"          # testdefinitions.function2.user_enters_a_new_f
  And user submits the update form                          # testdefinitions.function2.user_submits_the_up
  Then the full name should be updated to "New Full Name" # testdefinitions.function2.the_full_name_shoul

1 Scenarios (1 passed)
9 Steps (9 passed)
0m9.148s
```

Test report - 3

Test Case 3

Project Name: Z Store					
User Updating Profile Settings					
Test Case ID: Test_3			**Test Designed By: Muhamed Ashiq**		
Test Priority(Low/Medium/High):			**Test Designed Date: 02/ 04/ 2024**		
Module Name: **Products Module**			**Test Executed By: Ms. Ankitha Philip**		
Test Title: User Profile Update			**Test Execution Date: 02 / 04/ 2024**		
Description: Users Profile can be Updated					
Pre-Condition : User is able to update values					
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/ Fail)**
1	User Logs in and navigate to index page		User is logged in and navigated to settings page	Settings Page Reached	Pass
2	Clear already existing Fullname		User's Full name should be updated	User's Fullname is Updated	Pass
3	Provide new Fullname Value	Fullname: Jeevarag			
4	Click on Submit button				
Post-Condition: User is able to filter the Products. Hence the Functionality Works					

Test Case 4: Admin Suspends the User

Code

```
package testdefinitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class function3 {

    WebDriver driver = CommonDriver.getDriver();

    @Given("User is logged in with admin privileges")
    public void user_is_logged_in_with_admin_privileges() {

    }

    @Given("User is on the user management page")
    public void user_is_on_the_user_management_page() {
        driver.get("http://127.0.0.1:8000/adminPy");
    }

    @When("Admin suspends the user with username {string}")
    public void admin_suspends_the_user_with_username(String string) {
        WebElement suspendButton =
driver.findElement(By.id("disableUser"));
        suspendButton.click();
    }

    @Then("The user with username {string} should be marked as suspended")
    public void the_user_with_username_should_be_marked_as_suspended(String
string) {
        driver.quit();
    }
}
```

Screenshot

```
Scenario: Suspend an active user                                    # src/test/resources/Features/func
  Given browser is open                                            # testdefinitions.login.browser_is
  And user is on login page                                        # testdefinitions.login.user_is_on
  When user enters username and password                           # testdefinitions.login.user_enter
  And User clicks on login                                         # testdefinitions.login.user_click
  Given User is logged in with admin privileges                    # testdefinitions.function3.user_i
  And User is on the user management page                          # testdefinitions.function3.user_i
  When Admin suspends the user with username "testUser"            # testdefinitions.function3.admin_
  Then The user with username "testUser" should be marked as suspended # testdefinitions.function3.the_us

1 Scenarios (1 passed)
8 Steps (8 passed)
0m14.497s
```

Test report - 4

Test Case 4	

Project Name: Z Store	
Admin Suspends the User	
Test Case ID: Test_4	**Test Designed By: Muhamed Ashiq**
Test Priority(Low/Medium/High):	**Test Designed Date: 02/ 04/ 2024**
Module Name: Admin Module	**Test Executed By: Ms. Ankitha Philip**
Test Title: Admin Suspends User	**Test Execution Date: 02 / 04/2024**
Description: Admin can Suspend **Users**	

Pre-Condition : User is able to update values

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Logged in as admin and navigated to AdminPanel		Should be logged in and navigated to admin panel	Navigated to Admin Panel	Pass
2	Admin Selects the User to be Suspended		Seleted user will be suspended	User is Suspended	Pass
3	Click on the Suspend Button	Fullname: Jeevarag			

Post-Condition: Admin is able to Suspend the User. Hence the Functionality Works.

Test Case 5: Admin add More Sub-Categories

Code

```
package testdefinitions;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;

import io.cucumber.java.en.Given;
import io.cucumber.java.en.Then;
import io.cucumber.java.en.When;

public class function3 {

        WebDriver driver = CommonDriver.getDriver();

        @Given("User is logged in with admin privileges")
        public void user_is_logged_in_with_admin_privileges() {

        }

        @Given("User is on the user management page")
        public void user_is_on_the_user_management_page() {
                driver.get("http://127.0.0.1:8000/adminPy");
        }

        @When("Admin suspends the user with username {string}")
        public void admin_suspends_the_user_with_username(String string) {
                WebElement suspendButton =
driver.findElement(By.id("disableUser"));
                suspendButton.click();
        }

        @Then("The user with username {string} should be marked as suspended")
        public void the_user_with_username_should_be_marked_as_suspended(String
string) {
                driver.quit();
        }
}
```

Screenshot

```
Apr 17, 2024 3:36:16 PM cucumber.api.cli.Main run
WARNING: You are using deprecated Main class. Please use io.cucumber.core.cli.Main

Scenario: Adding a sub-category                            # src/test/resources/Features
  Given browser is open                                    # testdefinitions.login.brows
  And user is on login page                                # testdefinitions.login.user_
  When user enters username and password                   # testdefinitions.login.user_
  And User clicks on login                                 # testdefinitions.login.user_
  Given I am on the category form page                     # testdefinitions.function4.i
  And I select "Sub Category" option                       # testdefinitions.function4.i
  And I select a main category                             # testdefinitions.function4.i
  And I enter "Sample Sub Category" in the sub category field # testdefinitions.function4.i
  When I submit the form                                   # testdefinitions.function4.i
  Then the sub category should be added successfully        # testdefinitions.function4.t

1 Scenarios (1 passed)
10 Steps (10 passed)
0m7.104s
```

Test report - 5

Test Case 5

Project Name: Z Store	
Admin add More Sub-Categories	
Test Case ID: Test_5	**Test Designed By: Muhamed Ashiq**
Test Priority(Low/Medium/High):	**Test Designed Date: 02/ 04/ 2024**
Module Name: Admin Module	**Test Executed By: Ms. Ankitha Philip**
Test Title: Admin Suspends User	**Test Execution Date: 02 / 04/2024**
Description: Admin can Suspend **Users**	

Pre-Condition : User is able to update values

Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/ Fail)
1	Logged in as admin and navigated to AdminPanel		Should be logged in and navigated to admin panel	Navigated to Admin Panel	Pass
2	Admin Selects the User to be Suspended		Seleted user will be suspended	User is Suspended	Pass
3	Click on the Suspend Button	Fullname: Jeevarag			

Post-Condition: Admin is able to Suspend the User. Hence the Functionality Works.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage where a planned system transforms into a tangible and operational entity. Building trust and instilling confidence in users is of paramount importance for the success of the system. This is a critical phase that places a strong emphasis on user training and the creation of informative materials. The actual transition often occurs during or after user training. Implementation signifies the act of putting a new system into action after its design phase, turning a conceptual design into a functional one.

Implementation involves the process of transitioning from the old method of operation to the new one, whether it replaces the old system entirely or makes incremental changes. Ensuring that the process is executed accurately is vital to establish a system that aligns well with the organization's requirements. System implementation encompasses the following tasks:

- Meticulous planning.

- Assessment of the existing system and its constraints.

- Designing methods to facilitate the transition.

6.2 IMPLEMENTATION PROCEDURES

Software implementation involves the installation of software in its intended location and ensuring that it functions as intended. In some organizations, an individual who is not directly involved in using the software may oversee and approve the project's development. Initially, there may be some skepticism regarding the software, and it's essential to address these concerns to prevent strong resistance.

To ensure a successful transition, several key steps are important:

- Communicating Benefits: Users need to understand why the new software is an improvement over the old one, instilling trust in its capabilities.
- User Training: Providing training to users is crucial to make them feel confident and comfortable using the application.

To evaluate the outcome, it is essential to verify that the server program is running on the server. If the server is not operational, the expected outcomes will not be achieved.

6.2.1 User Training

User training is a critical component of ensuring that individuals know how to use and adapt to a new system. It plays a vital role in fostering user comfort and confidence with the system. Even when a system becomes more complex, the need for training becomes even more apparent. User training covers various aspects, including inputting information, error handling, querying databases, and utilizing tools for generating reports and performing essential tasks. It aims to empower individuals with the knowledge and skills needed to effectively interact with the system.

6.2.2 Training on the Application Software

Once the fundamental computer skills have been covered, the next step is to instruct individuals on using a new software program. This training should provide a comprehensive understanding of the new system, including navigation through screens, accessing help resources, error management, and resolution procedures. The training aims to equip users or groups with the knowledge and skills necessary to effectively utilize the system or its components. It's important to note that training may be tailored differently for various groups of users and individuals in different roles within the organization to cater to their specific needs and requirements.

6.2.3 System Maintenance

Maintaining a system's functionality is a complex challenge in software development. In the maintenance phase of the software life cycle, the software performs critical functions and operates smoothly. Once a system is operational, it requires ongoing care and maintenance to ensure its continued optimal performance. Software maintenance is a crucial aspect of the development process as it ensures that the system can adapt to changes in its environment. Maintenance involves more than just identifying and correcting errors in the code. It encompasses various tasks that contribute to the system's stability and effectiveness.

6.2.4 Hosting

Hosting a website entails making it accessible to users on the internet by storing its files and data on a server. This process involves several steps, beginning with choosing a suitable hosting provider that meets your website's requirements for storage, bandwidth, uptime, and support. Once a hosting provider is selected, you typically register a domain name for your website, which serves as its unique address.

After domain registration, you set up a hosting account by selecting a plan, providing payment information, and creating an account with the hosting provider. Subsequently, you upload your website files to the server using FTP or a web-based file manager. Organizing these files properly, including HTML, CSS, JavaScript, images, and other assets, ensures smooth functioning.

Configuring domain settings to point to the hosting server via DNS records is the next step. Thorough testing of the website for issues such as broken links or formatting problems is crucial before making it live. Once everything is set, you launch the website by updating hosting account settings or additional configurations as needed.

Post-launch, regular monitoring of performance, security, and uptime is essential. Many hosting providers offer tools and analytics to help track website metrics and address any arising issues. Consistently updating website content and software ensures security and relevance to the audience, ensuring a seamless online presence.

6.2.4.1 AMAZON ELASTIC COMPUTE CLOUD (EC2)

Amazon Elastic Compute Cloud (EC2) is a core component of Amazon Web Services (AWS), offering users the ability to rent virtual servers, or instances, in the cloud. It allows for flexible scaling of computing resources based on demand, with a variety of instance types optimized for different workloads. EC2 provides full control over server instances, supports various operating systems, and offers features for security, scalability, and cost-effectiveness. Users can easily manage their instances through the AWS Management Console or APIs. Overall, AWS EC2 enables businesses to run applications and services in a reliable, scalable, and cost-efficient manner in the cloud.

Procedure for hosting a website on Amazon EC2:

Step1: Create an AWS Account: Start by registering for an AWS account if you haven't already done so. Once logged in to the AWS Management Console, proceed to the EC2 dashboard.

Step2: Launch an EC2 Instance: Click on the "Launch Instance" button to initiate a new EC2 instance. Select the Ubuntu AMI (Amazon Machine Image) and choose an appropriate instance type according to your project's specifications.

Step3: Configure Instance Settings: Customize instance parameters such as quantity, networking configurations, and storage preferences. Include storage volumes to securely store

your project files and data.

Step4: Set Up Security Group: Establish a new security group or utilize an existing one to define firewall regulations. Ensure that ports 22 (SSH) for remote access and 8000 (or any other required port for your application) for web traffic are accessible.

Step5: Launch Instance: Review the instance setup and commence the EC2 instance launch process. Opt for or generate a key pair for SSH access.

Step6: Connect to Your Instance: Upon instance launch, establish an SSH connection. Utilize the public IP address or DNS name of your instance along with the key pair to establish a secure connection.

Step7: Clone Project Repository: Install Git on the EC2 instance and clone the EduSphere Fusion project repository from your Git repository using the git clone command.

Step8: Install Python and Django: Deploy Python and Django on the EC2 instance. Utilize the apt package manager for Python installation and pip to install Django along with any other necessary Python packages.

Step9: Install Dependencies: Install additional packages and dependencies essential for running the website, including database drivers and Django extensions.

Step10: Configure Django Settings: Update the Django settings file (settings.py) with the required database configuration, static file settings, and other project-specific configurations.

Step11: Run Django Server: Initiate the Django development server by executing the command python manage.py runserver 0.0.0.0:8000. This command operates the server on port 8000, accessible on all network interfaces.

Step12: Test Your Website: Access a web browser and navigate to the public IP address or DNS name of your EC2 instance followed by the designated port number (<public_ip>:8000). Confirm that your website is accessible and operates correctly.

Step13: Domain Name Configuration (Optional): If you own a domain name, configure the DNS settings to point to the public IP address of your EC2 instance.

Step14: SSL/TLS Certificate Setup (Optional): Enable HTTPS for your website by setting up an SSL/TLS certificate using AWS Certificate Manager or a third-party provider.

Step15: Monitor Your EC2 Instance: Employ AWS CloudWatch or other monitoring utilities to oversee the performance, security, and uptime of your EC2 instance. Regularly update your instance and software to ensure security and reliability.

Hosted Website: Amazon EC2

Hosted Link: http://13.201.70.229:1254

1. AWS Instance Dashboard
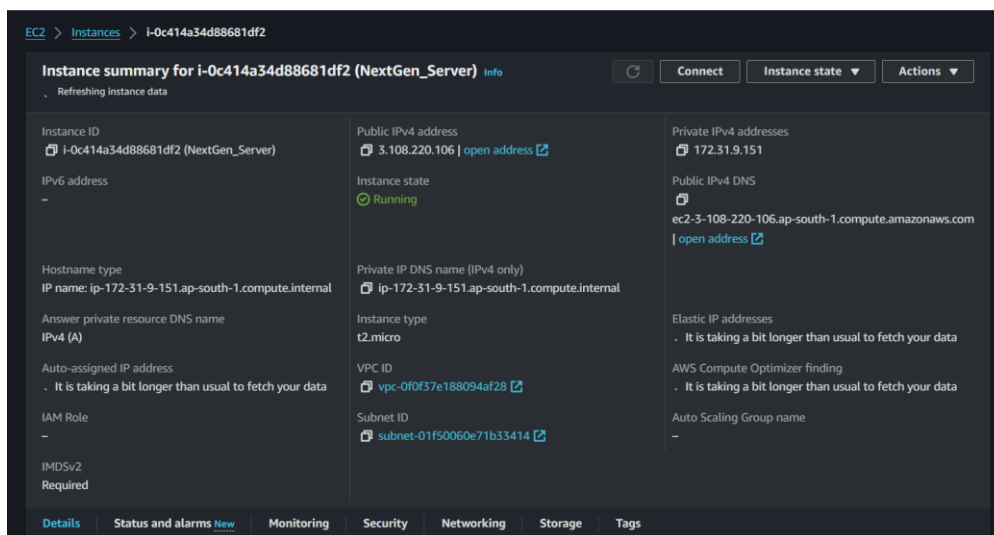
2. Using Public IP Address
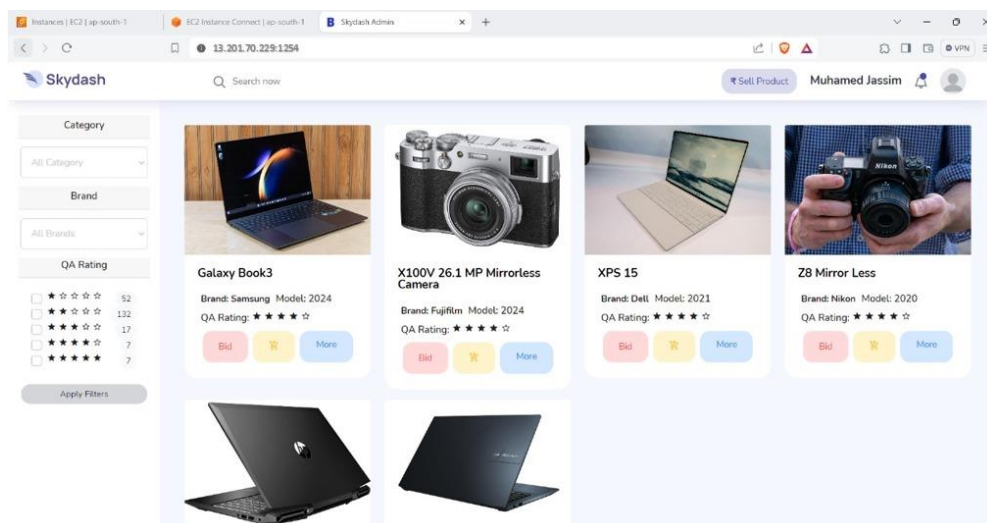
CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

In conclusion, the "Z Store" project stands as a beacon of innovation, uniting sustainability, craftsmanship, and technological advancement in a unique online marketplace that caters to both crafted and used products. The platform places a premium on user experience, boasting a visually appealing and intuitive interface that seamlessly integrates browsing, searching, and personalized account features for users. The vibrant community engagement adds an extra layer of richness to the platform, fostering connections among enthusiasts.

The "Z Store" project prioritizes a user-centric approach, offering an interface that is visually appealing and intuitive. Users can seamlessly browse and search for unique products, personalize their accounts, and engage with a vibrant community of like-minded enthusiasts. From the administrative standpoint, the system provides a robust dashboard for overseeing product inventory, user accounts, and community engagement features. Administrators can efficiently manage product listings, user interactions, and overall site settings to enhance the platform's appeal.

The "Z Store" system aspires to deliver a seamless and enriching experience for both users and administrators, utilizing technology to create a comprehensive online marketplace for crafted products. The successful implementation of this system holds the potential to significantly benefit its target audience, fostering a community of artisans, collectors, and individuals passionate about unique and handcrafted items. The project embodies a harmonious fusion of sustainability, craftsmanship, and technological sophistication, promising a vibrant and inclusive online marketplace.

7.1   FUTURE SCOPE

The future scope for the "Z Store" project envisions exciting possibilities with the integration of Machine Learning (ML) to enhance user engagement and expand its offerings. Introducing a new module that allows users to actively participate in the creation and customization of both crafted and used products could significantly enrich the platform, with ML algorithms providing personalized recommendations based on user preferences.

To elevate user experiences, the implementation of advanced features such as refined filtering options, extended multilingual support, and a comprehensive notification system can be explored. These enhancements would ensure that users have access to a diverse and personalized range of crafted products while staying informed about the latest additions and updates.

The vision for the "Z Store" project is to create a dynamic and thriving ecosystem that goes beyond traditional online marketplaces. By encouraging collaboration, customization, and community engagement, the project can position itself as a vibrant space where users actively contribute to the creation and appreciation of crafted products. The project is poised for an exciting and dynamic future, adapting to the evolving needs and aspirations of its expanding user community.

CHAPTER 8
BIBLIOGRAPHY

REFERENCES:

- "System Analysis and Design" by Gary B. Shelly and Harry J. Rosenblatt, 2009.

- "The Pragmatic Programmer: From Journeyman to Master" by Andrew Hunt and David Thomas, Pearson India, 1st Edition (2008).

- "Agile Software Development with Scrum" by Ken Schwaber and Mike Beedle, Pearson (2008).

- "Agile Testing: A Practical Guide for Testers and Agile Teams" by Lisa Crispin and Janet Gregory, Addison Wesley Professional, 1st Edition (2008).

WEBSITES:

- https://docs.djangoproject.com/en

- www.GeeksForg.com

CHAPTER 9
APPENDIX

9.1   Sample Code

Login Page

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>
 <!-- Required meta tags -->
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
 <title>Skydash Admin</title>
 <!-- plugins:css -->
 <link rel="stylesheet" href="{% static 'vendors/feather/feather.css' %}">
 <link rel="stylesheet" href="{% static 'vendors/ti-icons/css/themify-icons.css' %}">
 <link rel="stylesheet" href="{% static 'vendors/css/vendor.bundle.base.css' %}">
 <!-- endinject -->
 <!-- Plugin css for this page -->
 <!-- End plugin css for this page -->
 <!-- inject:css -->
 <link rel="stylesheet" href="{% static 'css/vertical-layout-light/style.css' %}">
 <!-- endinject -->
 <link rel="shortcut icon" href="{% static 'images/Custom Logo/i2.png' %}" />
</head>

<body>
 <div class="container-scroller">
  <div class="container-fluid page-body-wrapper full-page-wrapper">
   <div class="content-wrapper d-flex align-items-center auth px-0">
    <div class="row w-100 mx-0">
     <div class="col-lg-4 mx-auto">
      <div class="auth-form-light text-left py-5 px-4 px-sm-5">
       <div class="brand-logo">
        <img src="{% static 'images/Custom Logo/p2.png' %}" alt="logo">
       </div>

```
<h4>Hello! let's get started</h4>
<h6 class="font-weight-light">Sign in to continue.</h6>
<form class="pt-3" action="{% url 'signin' %}" method="post" id="logForm">
  {% csrf_token %}
  {% if invalid_credential %}
  <div class="alert alert-danger">
    Invalid Username or Password
  </div>
  {% endif %}
  <div class="form-group">
    <input type="text" class="form-control form-control-lg" name="username"
      id="exampleInputEmail1" placeholder="Username">
  </div>
  <div class="form-group">
    <input type="password" class="form-control form-control-lg" name="password"
      id="exampleInputPassword1" placeholder="Password">
  </div>
  <div class="mt-3">
    <input type="submit" class="btn btn-block btn-primary btn-lg font-weight-medium
      auth-form-btn" value="SIGN IN">
  </div>
  <div class="my-2 d-flex justify-content-between align-items-center">
    <a class="auth-link text-black" href="">Forgot password?</a>
  </div>
  <!-- <div class="mb-2">
    <button type="button" class="btn btn-block btn-facebook auth-form-btn">
      <i class="ti-facebook mr-2"></i>Connect using facebook
    </button>
  </div> -->
  <div class="text-center mt-4 font-weight-light">
    Don't have an account? <a href="{% url 'signup' %}" class="text-primary">Create</a>
  </div>
</form>
</div>
```

```
      </div>
     </div>
    </div>
    <!-- content-wrapper ends -->
   </div>
   <!-- page-body-wrapper ends -->
  </div>
  <!-- container-scroller -->
  <!-- plugins:js -->
  <script src="{% static 'vendors/js/vendor.bundle.base.js' %}"></script>
  <!-- endinject -->
  <!-- Plugin js for this page -->
  <!-- End plugin js for this page -->
  <!-- inject:js -->
  <script src="{% static 'js/off-canvas.js' %}"></script>
  <script src="{% static 'js/hoverable-collapse.js' %}"></script>
  <script src="{% static 'js/template.js' %}"></script>
  <script src="{% static 'js/settings.js' %}"></script>
  <script src="{% static 'js/todolist.js' %}"></script>
  <script>
   // window.onload = function (){
   //   location.replace('/userLoginCheck/');
   // };


   document.getElementById('logForm').addEventListener("submit", function(event){
    if (document.getElementById('exampleInputEmail1').value.trim().length == 0){
     event.preventDefault();
    }
   });
  </script>
  <!-- endinject -->
</body>

</html>
```

View Function

```
@never_cache
def showSignIn(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        userauth = authenticate(request, username=username, password=password)

        if userauth is not None:
            if TesterInfo.objects.filter(tester_id=userauth.email).exists():
                tester = TesterInfo.objects.get(tester_id=userauth.email)
                if not tester.is_active:
                    login(request, userauth)
                    return redirect('regTester')
                else:
                    login(request, userauth)
                    return redirect('testerPanel')
            else:
                request.session['user_email'] = userauth.email
                login(request, userauth)
                return HttpResponseRedirect(reverse('index'))
        else:
            context = {
                'invalid_credential':'wrong',
            }
            return render(request, 'login.html', context)
    else:
        return render(request, 'login.html')
```

Admin Panel

```
{% load static %}
<!DOCTYPE html>
<html lang="en">

  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <title>Z Store</title>
    <!-- plugins:css -->
    <link rel="stylesheet" href="{% static 'vendors/feather/feather.css' %}">
    <link rel="stylesheet" href="{% static 'vendors/ti-icons/css/themify-icons.css' %}">
    <link rel="stylesheet" href="{% static 'vendors/css/vendor.bundle.base.css' %}">
    <link rel="stylesheet" href="{% static 'vendors/mdi/css/materialdesignicons.min.css' %}">
    <!-- endinject -->
    <!-- Plugin css for this page -->
    <!-- End plugin css for this page -->
    <!-- inject:css -->
    <link rel="stylesheet" href="{% static 'css/vertical-layout-light/style.css' %}">
    <!-- endinject -->
    <link rel="shortcut icon" href="{% static 'images/Custom Logo/i2.png' %}" />
  </head>
```

```
<body class="sidebar-dark">
  <div class="container-scroller">
    <!-- partial:partials/_navbar.html -->
    <nav class="navbar col-lg-12 col-12 p-0 fixed-top d-flex flex-row navbar-dark">
      <div class="text-center navbar-brand-wrapper d-flex align-items-center justify-content-center">
        <a class="navbar-brand brand-logo mr-5" href="index.html"><img
            src="{% static 'images/Custom Logo/p2.png' %}" class="mr-2" alt="logo" /></a>
        <a class="navbar-brand brand-logo-mini" href="index.html"><img
            src="{% static 'images/logo-mini.svg' %}" alt="logo" /></a>
      </div>
      <div class="navbar-menu-wrapper d-flex align-items-center justify-content-end">
        <ul class="navbar-nav mr-lg-2">
          <li class="nav-item nav-search d-none d-lg-block">
            <div class="input-group">
              <div class="input-group-prepend hover-cursor" id="navbar-search-icon">
                <span class="input-group-text" id="search">
                  <i class="icon-search"></i>
                </span>
              </div>
              <input type="text" class="form-control" id="navbar-search-input"
                placeholder="Search now" aria-label="search" aria-describedby="search">
            </div>
          </li>
        </ul>
        <ul class="navbar-nav navbar-nav-right">
          <li class="nav-item dropdown">
            <a class="nav-link count-indicator dropdown-toggle" id="notificationDropdown"
href="#"
              data-toggle="dropdown">
              <i class="icon-bell mx-0"></i>
              <span class="count"></span>
            </a>
            <div class="dropdown-menu dropdown-menu-right navbar-dropdown preview-list"
              aria-labelledby="notificationDropdown">
              <p class="mb-0 font-weight-normal float-left dropdown-header">Notifications</p>
              <a class="dropdown-item preview-item">
                <div class="preview-thumbnail">
                  <div class="preview-icon bg-success">
                    <i class="ti-info-alt mx-0"></i>
                  </div>
                </div>
                <div class="preview-item-content">
                  <h6 class="preview-subject font-weight-normal">Application Error</h6>
                  <p class="font-weight-light small-text mb-0 text-muted">
                    Just now
                  </p>
                </div>
              </a>
              <a class="dropdown-item preview-item">
                <div class="preview-thumbnail">
                  <div class="preview-icon bg-warning">
                    <i class="ti-settings mx-0"></i>
                  </div>
                </div>
                <div class="preview-item-content">
```

```
                              <h6 class="preview-subject font-weight-normal">Settings</h6>
                              <p class="font-weight-light small-text mb-0 text-muted">
                                 Private message
                              </p>
                           </div>
                        </a>
                        <a class="dropdown-item preview-item">
                           <div class="preview-thumbnail">
                              <div class="preview-icon bg-info">
                                 <i class="ti-user mx-0"></i>
                              </div>
                           </div>
                           <div class="preview-item-content">
                              <h6 class="preview-subject font-weight-normal">New user registration</h6>
                              <p class="font-weight-light small-text mb-0 text-muted">
                                 2 days ago
                              </p>
                           </div>
                        </a>
                     </div>
                  </li>
                  <li class="nav-item nav-profile dropdown">
                     <a class="nav-link dropdown-toggle" href="#" data-toggle="dropdown"
id="profileDropdown">
                        <h4 style="display: inline-block; font-weight: 900; justify-content: center;" class="mr-
2">{{context.userData.fullname}}</h4>
                        {% if context.userData.profileImage %}
                        <img src="{{ context.userData.profileImage.url }}" alt="profile" />
                        {% else %}
                        <img src="{% static 'images/faces/avatar.jpeg' %}" alt="profile" />
                        {% endif %}
                     </a>
                     <div class="dropdown-menu dropdown-menu-right navbar-dropdown"
                        aria-labelledby="profileDropdown">
                        <a class="dropdown-item" href="{% url 'settings' %}">
                           <i class="ti-settings text-primary"></i>
                           Settings
                        </a>
                        <a class="dropdown-item" href="{% url 'userLogout' %}">
                           <i class="ti-power-off text-primary"></i>
                           Logout
                        </a>
                     </div>
                  </li>
               </ul>
               <button class="navbar-toggler navbar-toggler-right d-lg-none align-self-center" type="button"
                  data-toggle="offcanvas">
                  <span class="icon-menu"></span>
               </button>
            </div>
         </nav>
         <!-- partial -->
         <div class="container-fluid page-body-wrapper">
            <!-- partial:partials/_settings-panel.html -->
            <div class="theme-setting-wrapper">
```

```
<div id="settings-trigger"><i class="ti-settings"></i></div>
<div id="theme-settings" class="settings-panel">
  <i class="settings-close ti-close"></i>
  <p class="settings-heading">SIDEBAR SKINS</p>
  <div class="sidebar-bg-options selected" id="sidebar-light-theme">
    <div class="img-ss rounded-circle bg-light border mr-3"></div>Light
  </div>
  <div class="sidebar-bg-options" id="sidebar-dark-theme">
    <div class="img-ss rounded-circle bg-dark border mr-3"></div>Dark
  </div>
  <p class="settings-heading mt-2">HEADER SKINS</p>
  <div class="color-tiles mx-0 px-4">
    <div class="tiles success"></div>
    <div class="tiles warning"></div>
    <div class="tiles danger"></div>
    <div class="tiles info"></div>
    <div class="tiles dark"></div>
    <div class="tiles default"></div>
  </div>
</div>
</div>
<div id="right-sidebar" class="settings-panel">
  <i class="settings-close ti-close"></i>
  <ul class="nav nav-tabs border-top" id="setting-panel" role="tablist">
    <li class="nav-item">
      <a class="nav-link active" id="todo-tab" data-toggle="tab" href="#todo-section"
role="tab"
          aria-controls="todo-section" aria-expanded="true">TO DO LIST</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" id="chats-tab" data-toggle="tab" href="#chats-section" role="tab"
          aria-controls="chats-section">CHATS</a>
    </li>
  </ul>
  <div class="tab-content" id="setting-content">
    <!-- To do section tab ends -->
    <!-- chat tab ends -->
  </div>
</div>
<!-- partial -->
<!-- partial:partials/_sidebar.html -->
<nav class="sidebar sidebar-offcanvas" id="sidebar">
  <ul class="nav">
    <li class="nav-item active">
      <a class="nav-link" href="#">
        <i class="icon-grid menu-icon"></i>
        <span class="menu-title">Admin Dashboard</span>
      </a>
    </li>
    <li class="nav-item">
      <a class="nav-link" data-toggle="collapse" href="#ui-basic" aria-expanded="false"
        aria-controls="ui-basic">
        <i class="icon-head menu-icon"></i>
        <span class="menu-title">Authority</span>
        <i class="menu-arrow"></i>
```

```
            </a>
            <div class="collapse" id="ui-basic">
              <ul class="nav flex-column sub-menu">
                <li class="nav-item"> <a class="nav-link" href="{% url 'addTester' %}">Add
Tester</a></li>
                <li class="nav-item"> <a class="nav-link" href="#">Add Admin</a></li>
                <li class="nav-item"> <a class="nav-link" href="#">Manage T & A</a></li>
              </ul>
            </div>
          </li>
          <li class="nav-item">
            <a class="nav-link" data-toggle="collapse" href="#form-elements" aria-expanded="false"
              aria-controls="form-elements">
              <i class="mdi mdi-cart-outline menu-icon"></i>
              <span class="menu-title">Products</span>
              <i class="menu-arrow"></i>
            </a>
            <div class="collapse" id="form-elements">
              <ul class="nav flex-column sub-menu">
                <li class="nav-item"><a class="nav-link" href="{% url 'addCategory' %}">Add
Category</a></li>
                <li class="nav-item"><a class="nav-link" href="#"><del>Review
Products</del></a>
                </li>
                <li class="nav-item"><a class="nav-link" href="#"><del>Manage</del></a></li>
              </ul>
            </div>
          </li>
          <li class="nav-item">
            <a class="nav-link" data-toggle="collapse" href="#charts" aria-expanded="false"
              aria-controls="charts">
              <i class="icon-bar-graph menu-icon"></i>
              <span class="menu-title">Statistics</span>
              <i class="menu-arrow"></i>
            </a>
            <div class="collapse" id="charts">
              <ul class="nav flex-column sub-menu">
                <li class="nav-item"> <a class="nav-link" href="#"><del>Sales
Data</del></a></li>
                <li class="nav-item"> <a class="nav-link" href="#"><del>Commission
Data</del></a>
                </li>
                <li class="nav-item"> <a class="nav-link" href="#"><del>User
Activity</del></a></li>
              </ul>
            </div>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="{% static 'pages/documentation/documentation.html '%}">
              <i class="icon-paper menu-icon"></i>
              <span class="menu-title"><del>Documentation</del></span>
            </a>
          </li>
        </ul>
      </nav>
```

```
<!-- partial -->
<div class="main-panel">
  <div class="content-wrapper">
    <div class="row">
      <div class="col-12 col-xl-8 mb-4 mb-xl-0">
        <h3 class="font-weight-bold">Welcome {{ context.userData.fullname }}</h3>
        <h6 class="font-weight-normal mb-0">All systems are running smoothly! You have
<span
          class="text-primary">3 unread alerts!</span></h6><br>
      </div>
      <div class="col-lg-12 grid-margin stretch-card">
        <div class="card">
          <div class="card-body">
            <h4 class="card-title">Registered Users</h4>
            <p class="card-description">
              Registered User's both <code>active and inactive</code> users
            </p>
            <div class="table-responsive">
              <table class="table table-striped">
                <thead>
                  <tr>
                    <th>User</th>
                    <th>Full name</th>
                    <th>Email</th>
                    <th>Phone No:</th>
                    <th>Address Location</th>
                    <th>User Status</th>
                    <th>Action</th>
                  </tr>
                </thead>
                <tbody>
                  {% if combined_data %}
                  {% for userA, userD, userL in combined_data %}
                  {% if userA.userRole == 2 %}
                  {% if not userA.is_active %}
                  <tr style="background-color: black; color: aliceblue;">
                    {% else %}
                  <tr>
                    {% endif %}
                    {% if userD.profileImage %}
                    <td><img src="{{userD.profileImage.url}}" alt=""></td>
                    {% else %}
                    <td><img src="{% static 'images/faces/avatar.jpeg' %}" alt="">
                    </td>
                    {% endif %}
                    <td>{{ userD.fullname }}</td>
                    <td>{{ userA.email }}</td>
                    {% if userD.phoneNumber %}
                    <td>{{ userD.phoneNumber }}</td>
                    {% else %}
                    <td style="color: red;">Contact Number Unavailable</td>
                    {% endif %}
                    {% if userL.country %}
                    <td>
                        {{ userL.country}}, {{userL.state}},{{ userL.city }}
```

```
                                                </td>
                                                {% else %}
                                                <td style="color: red;">Location Unavailable</td>
                                                {% endif %}
                                                <td class="pr-1" style="font-weight: 900;">
                                                   {% if userA.is_active %}
                                                   <span style="color: azure;"
                                                      class="badge bg-success">Active</span>
                                                   {% else %}
                                                   <span style="color: red;"
                                                      class="badge bg-dark">Suspended</span>
                                                   {% endif %}
                                                </td>
                                                <td>
                                                   {% if userA.is_active %}
                                                   <a href="{% url 'suspendView' %}?email={{userA.email}}"
id="disableUser"

                                                      class="btn btn-inverse-warning btn-sm mb-2 ml-
2">Suspend</a><br>

                                                   {% else %}
                                                   <a href="{% url 'suspendView' %}?email={{userA.email}}"
                                                      class="btn btn-inverse-warning btn-sm mb-2 ml-2">Un-
Suspend</a><br>

                                                   {% endif %}
                                                   <button
                                                      class="btn btn-inverse-info btn-sm ml-2 accordion-button"
                                                      onclick="toggleAccordion(this)">Full Info</button>
                                                </td>
                                             </tr>
                                             <tr class="accordion-content" style="display: none;">
                                                <td colspan="7">
                                                   <div class="container">
                                                      <div class="row">
                                                         <div class="col-md-2">
                                                            {% if userD.profileImage %}
                                                            <img class="rounded float-start"
src="{{userD.profileImage.url}}" style="width: 100%; height: 100%;" alt="">
                                                            {% else %}
                                                            <img class="rounded float-start" src="{% static
'images/faces/avatar.jpeg' %}" style="width: 100%; height: 100%;" alt="">
                                                            {% endif %}
                                                         </div>
                                                         <div class="col-md-4">
                                                            <p style="display: inline-block;"><strong>Full
Name:</strong> {{userD.fullname}}</p><br>
                                                            <p style="display: inline-block;"><strong>User
Name:</strong> {{userA.username}}</p><br>
                                                            <p style="display: inline-block;"><strong>Email:</strong>
{{userA.email}}</p><br>
                                                            {% if userD.phoneNumber %}
                                                            <p style="display: inline-block;"><strong>Contact
Number:</strong> {{userD.phoneNumber}}</p><br>
                                                            {% else %}
                                                            <p style="display: inline-block;"><strong>Contact
Number:</strong> <span style="color: red;">Unavailable</span></p><br>
```

```
                                                    {% endif %}
                                                    <p style="display: inline-block;"><strong>Signed In
On:</strong> {{userA.date_joined}}</p><br>
                                                </div>
                                                <div class="col-md-4">
                                                    {% if userL.country %}
                                                    <p style="display: inline-block;"><strong>Country:</strong>
{{userL.country}}</p><br>
                                                    <p style="display: inline-block;"><strong>State:</strong>
{{userL.state}}</p><br>
                                                    <p style="display: inline-block;"><strong>City:</strong>
{{userL.city}}</p><br>
                                                    {% else %}
                                                    <p style="display: inline-block;"><strong>Country:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                                    <p style="display: inline-block;"><strong>State:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                                    <p style="display: inline-block;"><strong>City:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                                    {% endif %}
                                                    <p style="display: inline-block;"><strong>Last
Login:</strong> {{userA.last_login}}</p><br>
                                                    <p style="display: inline-block;"><strong>User
Status:</strong>
                                                        {% if userA.is_active %}
                                                        <span style="color: azure;"
                                                            class="badge bg-success">Active</span>
                                                        {% else %}
                                                        <span style="color: red;"
                                                            class="badge bg-dark">Suspended</span>
                                                        {% endif %}</p><br>
                                                </div>
                                                <div class="col-md-2">
                                                    <button class="btn btn-primary mt-5">User
Activity</button><br>
                                                </div>
                                            </div>
                                        </div>
                                    </td>
                                </tr>
                                {% endif %}
                                {% endfor %}
                                {% endif %}
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
        <div class="col-lg-12 grid-margin stretch-card">
            <div class="card">
                <div class="card-body">
                    <h4 class="card-title">Quality Testers</h4>
                    <p class="card-description">
                        Something About Quality Tester
```

```
                    </p>
                    <div class="table-responsive">
                      <table class="table table-striped">
                        <thead>
                          <tr>
                            <th>User</th>
                            <th>Name</th>
                            <th>Category</th>
                            <th>Phone no</th>
                            <th>Work Location</th>
                            <th>Action</th>
                          </tr>
                        </thead>
                        <tbody>
                          {% if combined_data %}
                          {% for userA, userD, userL in combined_data %}
                          {% if userA.userRole == 3 %}
                          {% if not userA.is_active %}
                          <tr style="background-color: black; color: aliceblue;">
                            {% else %}
                          <tr>
                            {% endif %}
                            {% if userD.profileImage %}
                          <td><img src="{{userD.profileImage.url}}" alt=""></td>
                            {% else %}
                          <td><img src="{% static 'images/faces/avatar.jpeg' %}" alt="">
                          </td>
                            {% endif %}
                          <td>{{ userA.username }}</td>
                          <td>{{userA.tester_set.all.first.fullname}}</td>
                            {% if userD.phoneNumber %}
                          <td>{{ userD.phoneNumber }}</td>
                            {% else %}
                          <td style="color: red;">Contact Number Unavailable</td>
                            {% endif %}
                            {% if userL.country %}
                          <td>
                            {{ userL.country}}, {{userL.state}},{{ userL.city }}
                          </td>
                            {% else %}
                          <td style="color: red;">Location Unavailable</td>
                            {% endif %}
                          <td class="pr-1" style="font-weight: 900;">
                            {% if userA.is_active %}
                            <span style="color: azure;"
                              class="badge bg-success">Active</span>
                            {% else %}
                            <span style="color: red;"
                              class="badge bg-dark">Suspended</span>
                            {% endif %}
                          </td>
                          <td>
                            {% if userA.is_active %}
                            <a href="{% url 'suspendView' %}?email={{userA.email}}"
class="btn btn-inverse-warning btn-sm mb-2 ml-2">Suspend</a>
```

```
                                    <button class="btn btn-inverse-danger btn-sm mb-2 ml-
2">Send<br>Termination Letter</button><br>
                                    <br>
                                    {% else %}
                                    <a href="{% url 'suspendView' %}?email={{userA.email}}"
class="btn btn-inverse-warning btn-sm mb-2 ml-2">Un-Suspend</a>
                                    <button class="btn btn-inverse-danger btn-sm mb-2 ml-
2">Send<br>Termination Letter</button><br>
                                    {% endif %}
                                    <button class="btn btn-inverse-info btn-sm ml-2 accordion-button"
onclick="toggleAccordion(this)">Full Info</button>
                                    <button class="btn btn-inverse-success btn-sm ml-
3">Portfolio</button>

                                    <!-- <td>
                                      <button
                                        class="btn btn-inverse-warning btn-sm mb-2 ml-
2">Suspend</button>

                                      <button class="btn btn-inverse-info btn-sm ml-2">Full
                                        Info</button>

                                    </td> -->
                                  </td>
                                </tr>
                                <tr class="accordion-content" style="display: none;">
                                  <td colspan="7">
                                    <div class="container">
                                      <div class="row">
                                        <div class="col-md-2">
                                          {% if userD.profileImage %}
                                          <img class="rounded float-start"
src="{{userD.profileImage.url}}" style="width: 100%; height: 100%;" alt="">
                                          {% else %}
                                          <img class="rounded float-start" src="{% static
'images/faces/avatar.jpeg' %}" style="width: 100%; height: 100%;" alt="">
                                          {% endif %}
                                        </div>
                                        <div class="col-md-4">
                                          <p style="display: inline-block;"><strong>Tester
Name:</strong> {{userA.username}}</p><br>
                                          <p style="display: inline-block;"><strong>Email:</strong>
{{userA.email}}</p><br>
                                          {% if userD.phoneNumber %}
                                          <p style="display: inline-block;"><strong>Contact
Number:</strong> {{userD.phoneNumber}}</p><br>
                                          {% else %}
                                          <p style="display: inline-block;"><strong>Contact
Number:</strong> <span style="color: red;">Unavailable</span></p><br>
                                          {% endif %}
                                          <p style="display: inline-block;"><strong>Signed In
On:</strong> {{userA.date_joined}}</p><br>
                                        </div>
                                        <div class="col-md-4">
                                          {% if userL.country %}
                                          <p style="display: inline-block;"><strong>Country:</strong>
```

```
{{userL.country}}</p><br>
                                        <p style="display: inline-block;"><strong>State:</strong>
{{userL.state}}</p><br>
                                        <p style="display: inline-block;"><strong>City:</strong>
{{userL.city}}</p><br>
                                        {% else %}
                                        <p style="display: inline-block;"><strong>Country:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                        <p style="display: inline-block;"><strong>State:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                        <p style="display: inline-block;"><strong>City:</strong>
<span style="color: red;">Unavailable</span></p><br>
                                        {% endif %}
                                        <p style="display: inline-block;"><strong>Last
Login:</strong> {{userA.last_login}}</p><br>
                                        <p style="display: inline-block;"><strong>User
Status:</strong>
                                            {% if userA.is_active %}
                                            <span style="color: azure;"
                                               class="badge bg-success">Active</span>
                                            {% else %}
                                            <span style="color: red;"
                                               class="badge bg-dark">Suspended</span>
                                            {% endif %}</p><br>
                                     </div>
                                     <div class="col-md-2">
                                        <button class="btn btn-primary mt-5">User
Activity</button><br>
                                     </div>
                                  </div>
                               </div>
                            </td>
                         </tr>
                         {% endif %}
                         {% endfor %}
                         {% endif %}
                      </tbody>
                   </table>
                </div>
             </div>
           </div>
         </div>
       </div>
       <!-- content-wrapper ends -->
       <!-- partial:partials/_footer.html -->
       <!-- partial -->
     </div>
     <!-- main-panel ends -->
   </div>
   <!-- page-body-wrapper ends -->
 </div>
 <!-- container-scroller -->
 <!-- plugins:js -->
 <script>
```
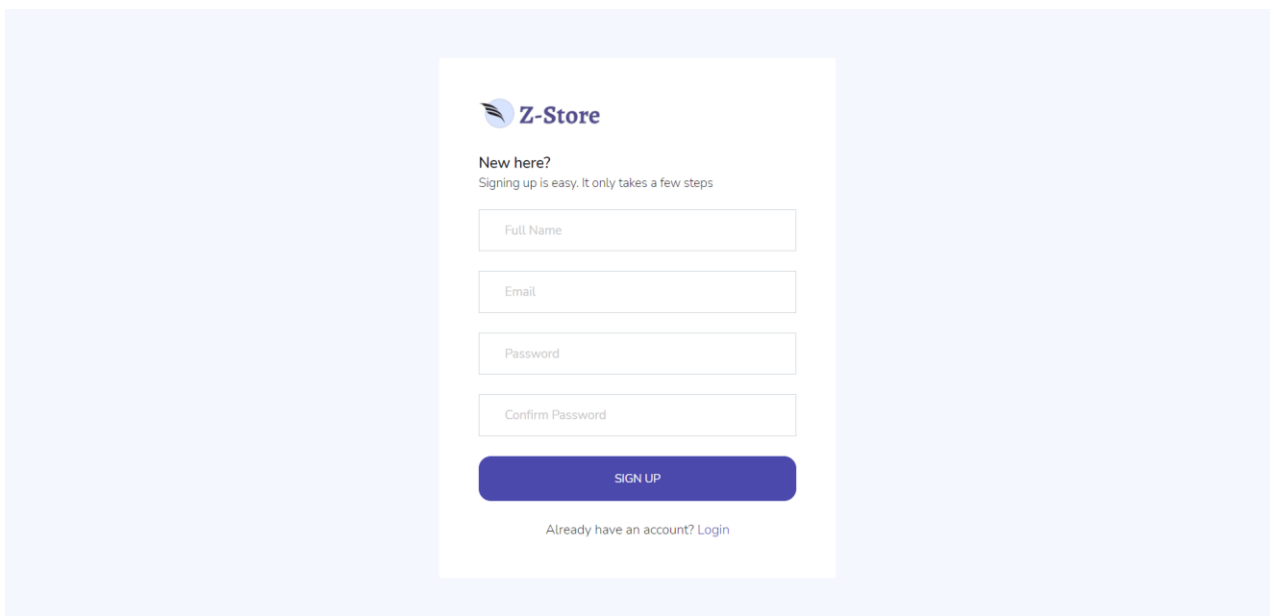
```
        function toggleAccordion(button) {
            var content = button.parentElement.parentElement.nextElementSibling;
            if (content.style.display === 'none' || content.style.display === '') {
                content.style.display = 'table-row';
                button.textContent = "Less Info"
            } else {
                content.style.display = 'none';
                button.textContent = "Full Info"
            }
        }
    </script>
    <script src="{% static 'vendors/js/vendor.bundle.base.js' %}"></script>
    <!-- endinject -->
    <!-- Plugin js for this page -->
    <!-- End plugin js for this page -->
    <!-- inject:js -->
    <script src="{% static 'js/off-canvas.js' %}"></script>
    <script src="{% static 'js/hoverable-collapse.js' %}"></script>
    <script src="{% static 'js/template.js' %}"></script>
    <script src="{% static 'js/settings.js' %}"></script>
    <script src="{% static 'js/todolist.js' %}"></script>
    <!-- endinject -->
    <!-- Custom js for this page-->
    <!-- End custom js for this page-->
  </body>
</html>
```

View Function

```python
@login_required
@never_cache
def showAdminPage(request):
    if request.user.userRole != 1:
        return redirect('index')
    users_authData = UserAuth.objects.all()
    usersAuthList= []
    usersDataList= []
    usersLocaList= []
    for user in users_authData:
        usersAuthList.append(user)
        users_Data = UserData.objects.filter(email_id=user.email).first()
        users_Loca = UserLoca.objects.filter(email_id=user.email).first()
        usersDataList.append(users_Data)
        usersLocaList.append(users_Loca)
    combined_data = list(zip(usersAuthList, usersDataList, usersLocaList))
    user_data = UserData.objects.get(email=request.user.email)
    user_loca = UserLoca.objects.get(email=request.user.email)
    context = {
        "userData": user_data,
        "UserLoca": user_loca
    }
    return render(request, 'admin_Page.html', {"combined_data": combined_data, "context": context})

def suspendView(request):
    respond_User = request.GET['email']
```
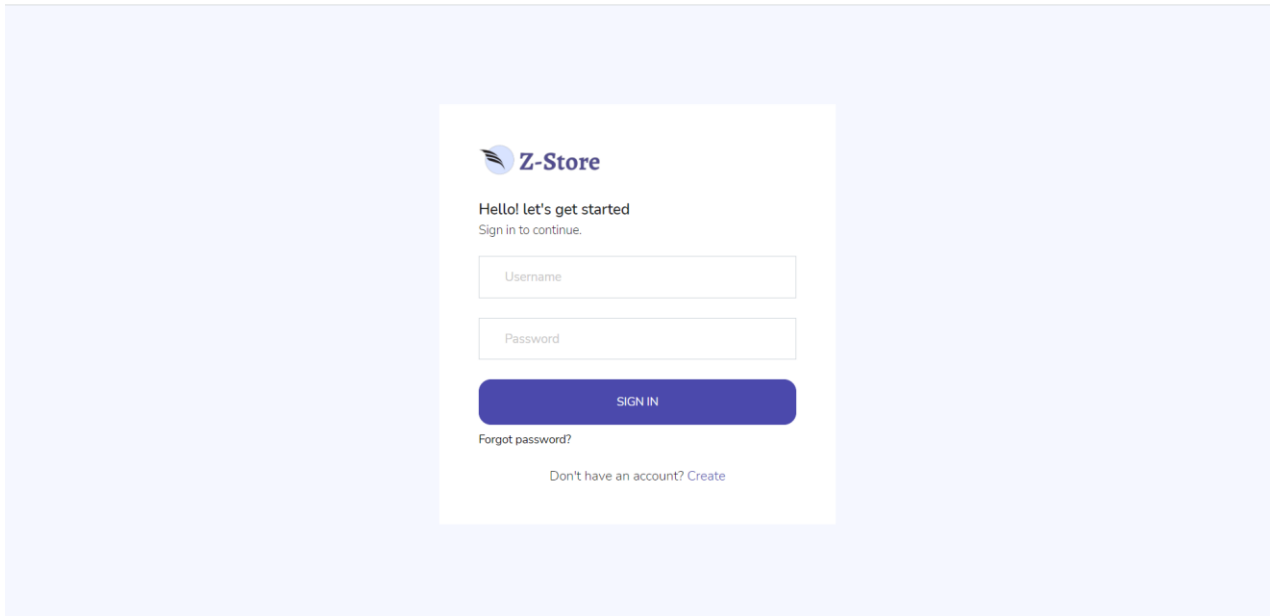
```
    user = request.user  # Example user data
    userauth = UserAuth.objects.get(email=respond_User)
    userdata = UserData.objects.get(email=user)
    if userauth.is_active:
        subject = 'Hello, '+userdata.fullname
        message = ""
        from_email = formataddr(('Z-Store', EMAIL_HOST_USER))
        recipient_list = [respond_User]
        userauth.is_active = 0
        data = {
            'user': user,
            'userauth': userdata,
            'message': True,
        }
        html_message = render_to_string('email_template.html', data)
        send_mail(subject, message, from_email, recipient_list, fail_silently=False,
html_message=html_message)
    else:
        subject = 'Hello, '+userdata.fullname
        message = "This is the Message from Admin\nYour Account has been released"
        from_email = formataddr(('Z-Store', EMAIL_HOST_USER))
        recipient_list = [respond_User]
        userauth.is_active = 1
        data = {
            'user': user,
            'userauth': userdata,
            'message': False,
        }
        html_message = render_to_string('email_template.html', data)
        send_mail(subject, message, from_email, recipient_list, fail_silently=False,
html_message=html_message)
    userauth.save()
    return redirect('adminPy')
```

9.1   Screen Shots

Login Page

Registration Page

User Settings Page

User Product Listing

Admin Panel

Tester Adding Page

User Product Management

User Cart Page