# AYURVEDIC MRD SYSTEM



***Submitted by***

|  |  |
| --- | --- |
| BOBY P VARUGHESE : | 33222825000 |
| JAITH J : | 33222825000 |
| MANU SHAJI : | 33222825000 |
| MIDHUN M : | 33222825000 |
| STEVE SAJAN JACOB : | 33222825056 |

***Guided by***

Ms. LEKSHMI V

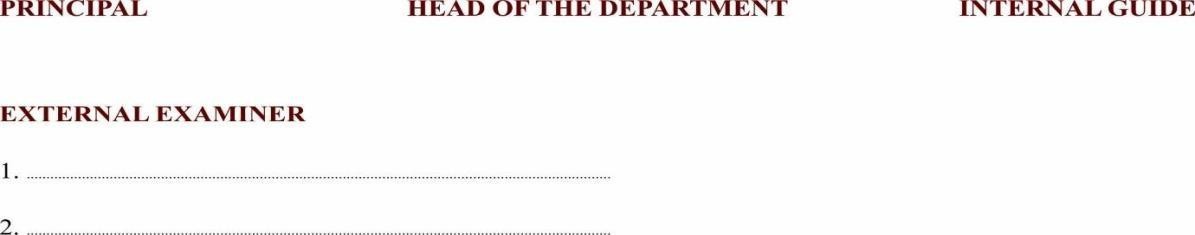
Assistant Professor



### 2024



CERTIFIED THAT THIS REPORT TITLED **AYURVEDIC MRD SYSTEM** IS A BONAFIDE RECORD OF THE MAJOR PROJECT WORK DONE BY **BOBY P VARUGHESE (33222825000), JAITH J (33222825000), MANU SHAJI (33222825000), MIDHUN M (33222825000)** and **STEVE SAJAN JACOB (33222825056)** UNDER OUR SUPERVISION AND GUIDANCE, AT THE DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS, CHRIST NAGAR COLLEGE TOWARDS THE PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF **BACHELOR OF COMPUTER APPLICATIONS** OF THE UNIVERSITY OF KERALA.



# ACKNOWLEDGEMENTS

We are using this opportunity to express our sincere gratitude to everyone who supported us throughout the course of this BCA project. We first like to thank the almighty for showering his blessings upon us.

We would like to thank our Manager, **Rev.Fr. Cyriac Madathil, CMI** for the encouragement provided.

Then we would like to thank our Principal, **Dr. Jolly Jacob** for the support and encouragement provided.

We express our sincere gratitude towards our Head of the Department of Computer Science and Applications, **Ms. Sunitha S Nair** for providing an opportunity for undertaking this project.

We would also like to thank our project internal guide, **Ms**. **Lekshmi V,** Assistant Professor, for all the support given by her throughout the completion of the project.

Last but not least we would like to express our profound thanks to all our friends for their support. We thank all the above-mentioned people for their aspiring guidance, invaluably constructive criticism, and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a few issues related to the project.

**WITH GRATITUDE**

BOBY P VARUGHESE

JAITH J

MANU SHAJI

MIDHUN M

STEVE SAJAN JACOB

# ABSTRACT

This abstract presents an overview of our project, the Ayurvedic Medical Record and Documentation (MRD) System, designed to streamline patient record management and treatment tracking in Ayurvedic hospitals. The system is divided into three primary modules, each catering to distinct user roles: Administrator Module, Doctor Module, and Nurse Module. The Administrator Module acts as the core of the system, offering functionalities to manage staff and patient registrations. Administrators can create and update records for staff members and patients, ensuring that the hospital's operations run smoothly and efficiently. The Doctor Module enables doctors to perform initial patient assessments, prescribe medications, and order treatment procedures. Doctors can also monitor patient vitals and update medical records, facilitating a thorough and continuous care approach. The Nurse Module is designed for nurses to conduct initial patient assessments, evaluate fall risks, and develop nursing care plans. Nurses can also administer medications as prescribed and update the medication administration chart, ensuring that patient care is carried out accurately and safely. Together, these modules ensure a comprehensive and integrated approach to patient care, allowing for seamless communication and data sharing across different hospital departments. By leveraging technologies such as React for the frontend, Node.js with Express.js for the backend, and MySQL for database management, the system provides a robust and scalable solution for managing patient records and enhancing healthcare delivery. The Ayurvedic MRD System not only improves operational efficiency but also enhances patient care by integrating holistic wellness programs, supporting personalized treatment plans, and providing advanced analytics for data-driven decision-making. This project aims to modernize Ayurvedic healthcare practices by combining traditional wisdom with contemporary medical technology, ultimately promoting a higher standard of care in Ayurvedic hospitals.

# CONTENTS

1. INTRODUCTION 1
   1. [ABOUT THE PROJECT 2](#_bookmark0)
   2. [ORGANISATIONAL OVERVIEW 3](#_TOC_250025)
2. SYSTEM ANALYSIS 4
   1. [EXISTING SYSTEM… 6](#_TOC_250024)
      1. [Limitations of Existing System… 6](#_TOC_250023)
   2. [PROPOSED SYSTEM 8](#_TOC_250022)
      1. [Advantages Of Proposed System… 9](#_TOC_250021)
      2. [Features Of Proposed System… 9](#_TOC_250020)
   3. [REQUIREMENT SPECIFICATIONS… 10](#_TOC_250019)
      1. [Hardware Requirements 10](#_TOC_250018)
      2. [Software Requirements 10](#_TOC_250017)
      3. [Functional Requirements 11](#_TOC_250016)
      4. [Non-Functional Requirements 11](#_TOC_250015)
   4. [FEASIBILITY STUDY 15](#_TOC_250014)
      1. [Economic Feasibility… 15](#_TOC_250013)
      2. [Operational Feasibility… 16](#_TOC_250012)
      3. [Technical Feasibility… 16](#_TOC_250011)
3. SYSTEM DESIGN 17
   1. [DESIGN OF SUBSYSTEM… 18](#_TOC_250010)
   2. [USER INTERFACE DESIGN 19](#_TOC_250009)
      1. [Input Design… 19](#_TOC_250008)
      2. [Output](#_bookmark1) Design 20
   3. [DATABASE DESIGN 21](#_TOC_250007)
      1. [Table Design… 23](#_TOC_250006)
      2. [Data Dictionary… 25](#_TOC_250005)
   4. [MODELLING 27](#_TOC_250004)
      1. [Data flow diagram (DFD) 27](#_bookmark2)
      2. [Entity relationship model… 32](#_TOC_250003)
   5. [ARCHITECTURAL DESIGN 34](#_TOC_250002)
      1. [Structure](#_bookmark3) Chart 34
   6. [PROCEDURAL](#_bookmark4) DESIGN 35
      1. [Flow Chart… 35](#_TOC_250001)
4. PROCESSING ENVIRONMENT 40
   1. [HARDWARE](#_bookmark5) SPECIFICATIONS 41
   2. [SOFTWARE](#_bookmark6) SPECIFICATIONS 43
5. CODING AND IMPLEMENTATION 46
   1. [CODING 4](#_bookmark7)7
   2. [TESTING](#_bookmark8) 63
      1. [Unit Testing 63](#_bookmark9)
      2. [Integration Testing…..…, 64](#_bookmark10)
      3. [System Testing… 64](#_TOC_250000)
6. SECURITY, BACKUP AND RECOVERY MECHANISM 66
7. FUTURE ENHANCEMENT 68
8. SOFTWARE MAINTENANCE 70
9. CONCLUSION 73
10. APPENDIX 74

[GANTT CHART 75](#_bookmark11)

[SCREENSHOTS 76](#_bookmark12)

[MENU](#_bookmark13) TREE 81

MEETING MINUTES 82

1. BIBLIOGRAPHY 87

1

**INTRODUCTION**

## ABOUT THE PROJECT

The Ayurvedic Medical Record and Documentation (MRD) System epitomizes a significant advancement in the healthcare management landscape, particularly within the realm of Ayurvedic hospitals. Our innovative approach encompasses three essential modules: the Administrator Module, Doctor Module, and Nurse Module, each meticulously crafted to address the distinct needs of hospital administrators, doctors, and nurses. The Administrator Module serves as the backbone of our platform, furnishing administrators with comprehensive tools for managing staff and patient registrations. Administrators can create, update, and maintain detailed records, ensuring that hospital operations run smoothly and efficiently. This module includes robust features for overseeing hospital activities, thereby enhancing operational security and coordination. The Doctor Module empowers physicians by providing intuitive tools for managing patient assessments and treatment plans. Doctors can conduct initial assessments, prescribe medications, order treatment procedures, and monitor patient vitals. This module facilitates a holistic approach to patient care, enabling doctors to provide continuous and personalized treatment, integrating traditional Ayurvedic principles with modern medical practices. At the heart of our platform lies the Nurse Module, which offers a seamless interface for nurses to manage patient care effectively. Nurses can perform initial assessments, evaluate fall risks, develop nursing care plans, and administer medications. This module ensures that patient care is carried out accurately and safely, promoting a coordinated and compassionate approach to healthcare. Our commitment to excellence drives us to continuously enhance our platform, incorporating user feedback and embracing innovation to meet the evolving needs of Ayurvedic hospitals. By fostering a collaborative and user-centric environment, we strive to create a vibrant healthcare community where traditional wisdom meets contemporary technology. In essence, the Ayurvedic MRD System transcends traditional patient record management platforms, offering a holistic and integrated healthcare experience for both providers and patients. Whether you're a hospital administrator seeking to optimize operations, a doctor looking to provide comprehensive care, or a nurse dedicated to ensuring patient safety, our system is designed to meet your needs and exceed your expectations. Join us in revolutionizing Ayurvedic healthcare management and experience the unparalleled efficiency and quality of our platform.

## ORGANISATIONAL OVERVIEW

# SYSTEM ANALYSIS

## INTRODUCTION TO SYSTEM ANALYSIS

System analysis involves a comprehensive examination of a system's components, processes, and interactions to understand its functionality, identify requirements, and pinpoint areas for improvement. This iterative process begins with gathering and documenting stakeholder needs and requirements, encompassing both functional aspects (what the system should do) and non- functional aspects (how it should perform). System analysts delve into existing system operations, scrutinizing data, user feedback, and industry trends to uncover inefficiencies, limitations, and potential risks. Feasibility assessments are conducted to evaluate proposed solutions in terms of technical, economic, and operational viability. Collaborating with designers and developers, system analysts translate requirements into detailed system designs, considering architecture, functionalities, and user interfaces. Risk management strategies are devised to mitigate potential threats and ensure the system's reliability and security. Throughout the analysis process, continuous evaluation and optimization efforts are undertaken to enhance the system's performance, usability, and alignment with stakeholder needs. Ultimately, system analysis enables organizations to make informed decisions, allocate resources effectively, and develop solutions that meet the evolving demands of users, stakeholders, and the broader organizational objectives.

## EXISTING SYSTEM

The existing system in reference to our project comprises various conventional e-commerce platforms available on the internet. These platforms typically offer a user-facing interface for browsing and purchasing products, along with basic functionalities for sellers to list their products. However, the existing systems often lack comprehensive tools for efficient inventory management and seller engagement. Sellers may find it challenging to manage their product listings effectively, leading to inefficiencies and suboptimal user experiences. Additionally, administrators may encounter difficulties in monitoring and moderating user activities, potentially compromising the security and integrity of the platform. Furthermore, existing systems may not provide a seamless and intuitive shopping experience for users, with limited features for product exploration, secure payment processing, and order tracking. As a result, users may face obstacles in finding and purchasing products, leading to dissatisfaction and decreased user engagement. In contrast, our project aims to address these shortcomings by introducing a multi-module online store website with dedicated modules for administrators, sellers, and users. By leveraging advanced technologies and innovative design principles, we seek to create a comprehensive and user-centric platform that enhances the overall e-commerce experience for all stakeholders.

### Limitations of Existing System

* + - * **Limited Seller**: Many existing platforms offer basic functionalities for sellers to list their products but lack comprehensive tools for efficient inventory management, pricing optimization, and customer engagement. This limitation can hinder sellers' ability to effectively showcase their products and maximize their online presence.
      * **Scalability Challenges**: Some existing systems may struggle to accommodate a growing number of users and products, leading to performance issues such as slow loading times, system crashes, and downtime. This scalability limitation can negatively impact user experience and hinder the platform's ability to support a large and diverse user base.
      * **Security Concerns**: Security vulnerabilities such as data breaches, unauthorized access, and fraudulent activities pose significant risks to both users and administrators of existing

e-commerce systems. Without robust security measures in place, sensitive user information and financial transactions may be compromised, leading to loss of trust and credibility.

* + - * **Limited User Experience**: Existing platforms may offer a suboptimal user experience characterized by cluttered interfaces, confusing navigation, and lack of personalized recommendations. This limitation can frustrate users and deter them from engaging with the platform, resulting in decreased user retention and satisfaction.
      * **Inefficient Search and Discovery**: Many existing systems rely on basic search algorithms and filtering options, making it challenging for users to discover relevant products efficiently. This limitation can impede users' ability to find desired items quickly and may result in abandoned shopping carts and lost sales opportunities.
      * **Poor Mobile Optimization**: Some existing platforms may lack proper optimization for mobile devices, leading to a disjointed and cumbersome user experience for mobile users. With the increasing prevalence of mobile shopping, this limitation can significantly impact the platform's com
      * **Limited Customer Support**: Inadequate customer support channels and response times can frustrate users who encounter issues or have questions about their purchases. Without timely and effective support, users may perceive the platform as unresponsive and unreliable, leading to negative reviews and diminished trust positiveness and user engagement.

## PROPOSED SYSTEM

Our proposed system is an innovative online store website with a multimodule approach tailored for administrators, sellers, and users. The Admin Module empowers administrators with tools for managing user accounts, product categories, and system settings, ensuring platform security and smooth operation. The Shopkeeper Module provides sellers with efficient inventory management tools to optimize offerings and maximize revenue. Meanwhile, the User Module offers a seamless shopping experience with intuitive browsing, secure payments, and personalized recommendations. Key features include advanced search options, secure payment processing, responsive design, analytics, and comprehensive user support. Our system prioritizes usability, security, and scalability to deliver an unparalleled e-commerce platform that meets the evolving needs of stakeholders.

### Advantages of Proposed System:

* + - * **Enhanced User Experience**: Our platform will offer a seamless and intuitive shopping experience, fostering user engagement and satisfaction.
      * **Improved Seller Tools**: Sellers will benefit from efficient inventory management tools and enhanced visibility for their products, leading to increased sales and revenue.
      * **Payment Approval**: Admins have the responsibility to monitor and approve the payment status of users. They can verify payments, process refunds when necessary, and maintain financial integrity on the platform.
      * **Advanced Search and Filtering**: Our platform will feature advanced search and filtering options, allowing users to discover products more efficiently based on their preferences, price range, and other criteria.

### Features of Proposed System:

* + - * **Streamlined Administration:** Administrators will have access to powerful tools for managing the platform, ensuring smooth operation and security.
      * **Payment Approval:** Admins have the responsibility to monitor and approve the payment status of users. They can verify payments, process refunds when necessary, and maintain financial integrity on the platform.
      * **E-commerce Platform**: Online marketplace for buying and selling goods and services.
      * **User Authentication**: Verification process ensuring authorized access to the system.
      * **Product Catalogue**: Database containing information on available products for sale.
      * **Order Management:** System for processing and tracking customer orders.
      * **Payment Gateway**: Secure service enabling online transactions between buyers and sellers.
      * **Feedback Mechanism:** Feature allowing users to provide reviews and ratings for products/services.
      * **Inventory Control**: Management of stock levels and product availability.
      * **Responsive Design**: Website layout adjusts for optimal viewing across various devices.
      * **Customer Support**: Assistance provided to users for inquiries and issue resolution.

## REQUIREMENT SPECIFICATIONS

### Hardware Requirements

Processor : Intel core i7

Processor speed : 3GHz or above

RAM : 3GB or above

Hard Disk Capacity : 1TB

Keyboard : Multimedia Keyboard

Mouse : Standard

USB : 2.0 & 3.0

### Software Requirements

Operating System : Windows 11

Front End : HTML 5, CSS, Angular 14

Language : Python, Node JS

Tools used : Visual Studio Code

Database : SQLite 3

### Functional Requirements

"Merch Mingle" revolutionizes shopping by blending social networking with e-commerce. It offers a dynamic platform where users can explore a diverse range of products from fashion to gadgets, all while engaging with friends and fellow shoppers. With personalized recommendations and advanced algorithms, each browsing experience is tailored to individual preferences. Brands can connect directly with their target audience, showcasing products through interactive content and exclusive offers. Cutting-edge technology like virtual try-on and augmented reality enhances the shopping experience, allowing users to visualize products and make confident purchasing decisions. A streamlined checkout process ensures convenience, while community-driven feedback fosters transparency and trust. Whether seeking inspiration or connecting with like-minded individuals, Merch Mingle provides a vibrant digital marketplace where shopping becomes an immersive and interactive experience.

**ADMIN MODULE**

**FN1: Login:** Admin can login with the given user id and password. Input: Username and Password.

Output: Login Successfully and redirects to profile of admin.

**FN2: View Shops:** The admin should be able to view shops. Input: shop\_id

Output: shop details viewed successfully.

**FN3: Approve/Reject Shops:** The admin should be able to Approve or reject Shops Input: shopid, status

Output: Shop Approved

**FN4: Remove Shops:** Admin can remove shops based on the feedback given by the customer.

Input: shopid

Output: Shop Rejected

**FN5: View Feedback**: Admin can view the feedback submitted by the customer. Input: Shop id

Output: Feedback viewed successfully

## SHOP KEEPER MODULE

**FN1: Registration:** Shopkeeper can register an account which is approved or rejected by the admin. Input: Store Name, Owner Name, Email

Output: Signed Up Successfully.

**FN2: Add Product:** The Shopkeeper should be able to add product. Input: Product Name, Product Rate, Product Stock, choose Product Image. Output: Product Added.

**FN3: View Product:** The shopkeeper should be able to view the status of the products. Input: Product id.

Output: Product Status.

**FN4: Add Vehicle:** The shopkeeper can add Vehicle for product transportation.

Input: Vehicle Driver Name, Vehicle Registration Number. Output: Vehicle Added successfully.

**FN5: View Vehicle**: The shopkeeper can View Vehicle Status.

Input: Driver name, Registration number Output: Vehicle Status

**FN6: View Orders:** The shopkeeper can approve the packages for users.

Input: Order id, Date

Output: Action Invoked Correctly

**FN7: View Placed Orders:** The shopkeeper can view the placed orders. Input: Date

Output: Action Invoked Correctly

**FN8: View Shipped Orders:** The shopkeeper can view the shipped orders.

Input: Date, order id

Output: Action invoked correctly.

**FN9: View Feedback:** The shopkeeper can view the feedback from users. Input: Product name, Date

Output: Feedback Submitted.

## CUSTOMER MODULE

**FN1: Registration:** A user can register into the software by giving valid credentials**.** Input: email, phone no, Password, Confirm password.

Output: New user has been added.

**FN2: Login:** It allows registered applicants to login with the given user id and password**.** Input : Email and Password.

Output: Login Successfully and redirects to profile of patient.

**FN3: View Product**: Customer can view the details of available Product. Input: Product name

Output: View details of available product.

**FN4: Make Order:** Customer can make the order**.** Input: Product id, Product name, Quantity.

Output: Order Placed.

**FN5: Payment:** Once approved, the customer can make payment for further processing. Input: card no, CVV, Expiry.

Output: Payment details completed successfully.

**FN6: Feedback:** Customer can post feedback. Input: Product name, feedback

Output: Feedback Posted

### Non-Functional Requirements

* + - * Specifies how well the system should perform under various conditions, Performance including response times, throughput, and scalability.
      * Security: Defines the security measures the system must adhere to, including access controls, data encryption, authentication, and authorization.
      * Usability and User Experience: Outlines requirements related to user interface design, accessibility, and user satisfaction.
      * Reliability and Availability: Specifies the system's expected uptime, reliability, and mechanisms for backup and recovery.
      * Scalability: Describes how the system should handle increased loads and demands as the user base grows.
      * Maintainability: Specifies how easily the system can be maintained, updated, and extended over time.
      * Compatibility: Outlines the system's ability to work with different hardware, software, and configurations.

## FEASIBILITY STUDY

Feasibility study is the test of the system proposal made to identify whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be cost effective from a business point of view and whether it can be developed with the given budgetary constraints. A feasibility study should be relatively cheap and done at the earliest possible time. Depending on the study, the decision is made whether to go ahead with a more detailed analysis. When a new project is proposed, it normally goes through feasibility assessment. A Feasibility study is carried out to determine whether the proposed system is possible to develop with available resources and what should be the cost consideration. Facts considered in the feasibility analysis were:

* Economic Feasibility
* Operational Feasibility
* Technical Feasibility

### Economic Feasibility

This feasibility study presents tangible and intangible benefits from the prefect by comparing the development and operational cost. This system needs some more initial investment than the existing system, but it can be justifiable that it will improve the quality of service. Thus, feasibility study should centre along the following points:

* + - * Improvement resulting over the existing method in terms of accuracy, timeliness.
      * Cost comparison
      * Estimate the life expectancy of the hardware.
      * Overall objective
      * Our project is economically feasible. It does not require much cost to be involved in the overall process. It is essential because the main goal of the proposed system is to have economically better results with the increase.

### Operational Feasibility

Operational analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis, the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. An entrepreneur must accurately weigh the cost versus benefits before taking an action. Cost-based study: It is important to identify cost and benefit factors, which can be categorized as follows:

1. Development costs.
2. Operating costs.

This is an analysis of the costs to be incurred in the system and benefits derivable out of the system. Time-based study: This is an analysis of the time required to achieve a return on investments The future value of a project is also a factor. This application can be operated by all the users across the world.

### Technical Feasibility

Technical Feasibility deals with the hardware as well as software requirements. Technology Is not a constraint to type system development. We have to find out whether the necessary technology, the proposed equipment has the capacity to hold the data, which is used in the project, should be checked to carry out this technical feasibility. The technical feasibility issues usually raised during the feasibility stage of investigation include.

* + - * This software is running Windows 11.
      * The minimum hardware required is 3GB and maximum hardware required is 400GB.
      * The system can be expanded.

# SYSTEM DESIGN

## DESIGN OF SUBSYSTEM

The Design of Subsystem encapsulates behaviour, providing explicit and formal interfaces, and does not expose its internal contents. This provides the ability to completely encapsulate the interactions of a few classes and/or subsystems. This project has three major modules. These three modules are the core functionalities of the system. The project is divided into three major modules according to their functionalities.

1. ADMIN
2. USER
3. SHOPKEEPER

## ADMIN

* + **Login**: Admin can login with the given user id and password
  + **View Shops**: The admin should be able to view shop**s.**
  + **Approve/Reject Shops**: The admin should be able to Approve or reject Shops.

## CUSTOMER

* + **Registration**: A user can register into the software by giving valid credentials.
  + **View Product:** User can view the details of available Product.
  + **Make Order**: User can make the order.
  + **Payment:** Once approved, user can make payment for further process
  + **Feedback**: Customer can post feedback

## SHOPKEEPER

* + **Registration:** Shopkeeper can register an account which is approved or rejected by the admin.
  + **Add Product**: The Shopkeeper should be able to add product**.**
  + **View Product:** The admin should be able to view the status of the product.
  + **View Orders**: The admin can approve the packages for users.
  + **View Placed Orders**: The admin can view the placed orders.
  + **View Feedback**: The admin can view the feedback from users.

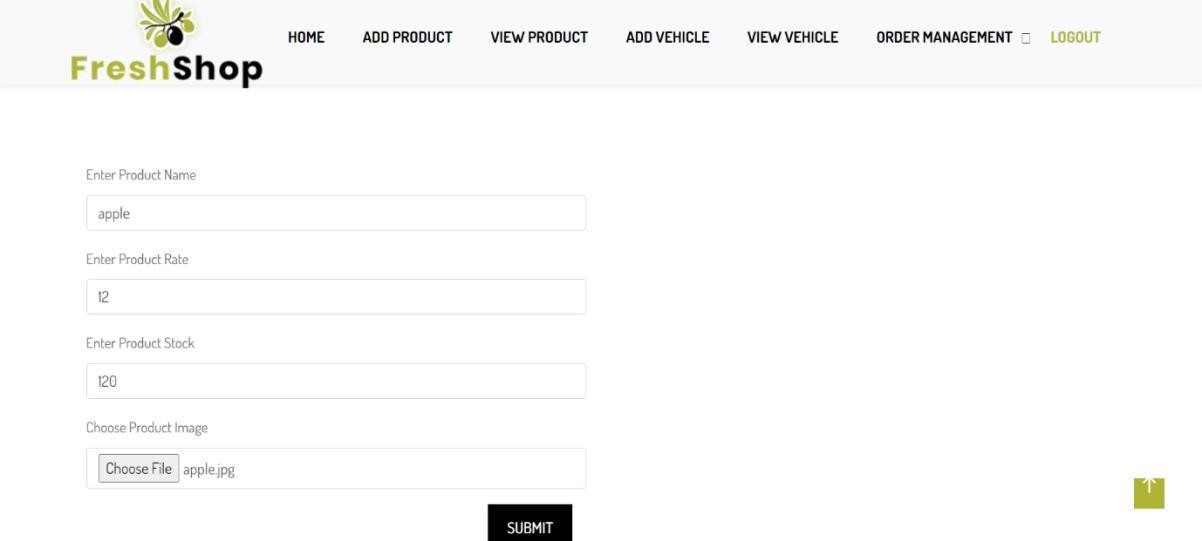
## USER INTERFACE DESIGN

### Input Design

Input designing is the basic theory to be considered during system study. The input media used in the system is the keyboard. Details are entered in the system through different data entry screens. The system is designed in a user-friendly manner. Appropriate error messages are displayed when false data is entered. The user interface design is very important for any application. The interface design defines how the software communicates within itself, to system that interpreted with it and with human who use it. The input design is the process of converting the user-oriented description of inputs into a programmer-oriented specification. It is the link that ties the system into the world of its users. Input design involves determining the record media, method of input, speed of capture and entry to the system. The analyst should consider the following points when designing the input:

* + - * Nature of the input processing.
      * Flexibility and thoroughness validation rules.
      * Handling of priorities with the input procedures.
      * Use of composite input documents to reduce the number of different ones.

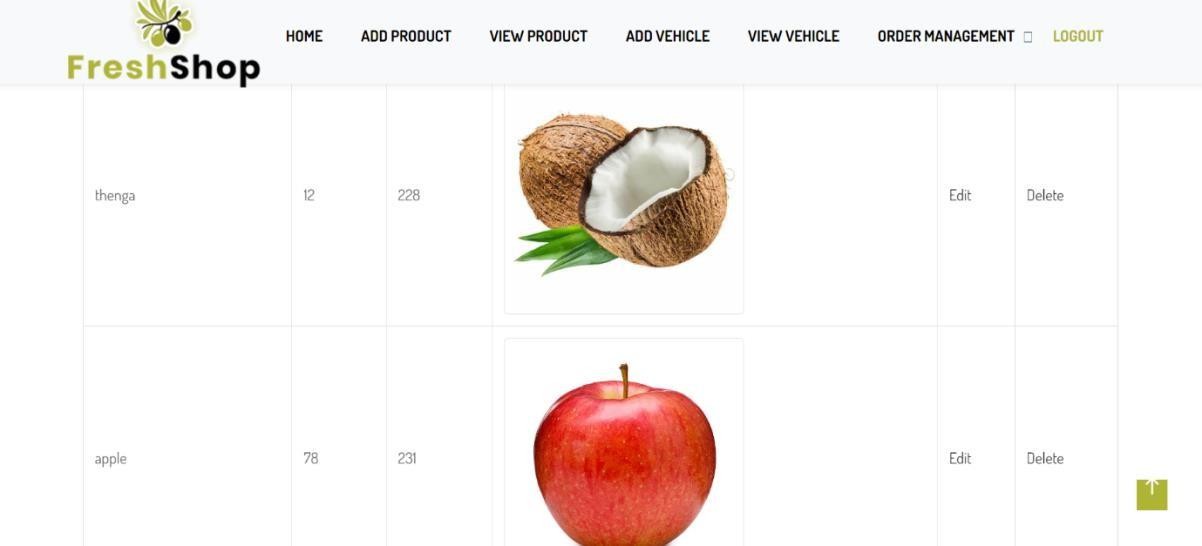
***Add Product***



### Output Design

An inevitable activity in the system design is the proper design of output in a form acceptable to the user. Outputs from the system are required primarily to communicate the result of processing to users. Outputs also provide a permanent copy of the results for later consultation. An intelligible output will improve system relationship with the user and help in the decision- making process. The output design must be specified and documented, data items must be accurately defined and arranged for clarity and easy comprehension. An inevitable activity in the system design is the proper design of output in a form acceptable to the user. Outputs from the system are required primarily to communicate the result of processing to users. Outputs also provide a permanent copy of the results for later consultation. An intelligible output will improve the system relationship with the user and help in the decision-making process. The output design must be specified and documented, data items must be accurately defined and arranged for clarity and easy comprehension.

***View Product Page***



## DATABASE DESIGN

The overall objective in the development of database technology has been to treat data as an organizational resource and as an integrated whole. DBMS allows data to be protected and organized separately from other resources. Database is an integrated collection of data. The most significant form of data as seen by the programmers is data stored on direct access storage devices. This is the difference between logical and physical data. Database files are the key source of information in the system. It is the process of designing database files, which are the key source of information to the system. The files should be properly designed and planned for collection, accumulation, editing and retrieving the required information. The organization of data in database aims to achieve three major objectives:

* Data integration.
* Data integrity.
* Data independence.

In the context of "Merch Mingle," SQLite 3 serves as an ideal database solution for managing various aspects of the project. As a lightweight, serverless, and self-contained database engine, SQLite 3 offers several advantages tailored to the needs of your project. Firstly, SQLite 3's simplicity and ease of setup make it an excellent choice for a project like "Merch Mingle." With minimal configuration requirements, developers can quickly integrate SQLite into the project without the need for a separate database server. This streamlined setup process saves time and resources, allowing the team to focus on building the application's core functionality. Additionally, SQLite 3's small footprint is well-suited for applications with limited resources or deployment environments. Since SQLite operates as a single-file database, it eliminates the complexities associated with managing multiple database files or server installations.

### Normalization

Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion and updating anomalies. The table has been normalized up to the third normal form. It is the process of structuring a relational database in accordance with a series of so-called normal forms in order to reduce data.

redundancy and improve data integrity. Normalization entails organizing the columns (attributes) and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design). In short, the rules for each of the five normal forms are as follows:

-

* **First normal form**- A relation is said to be in 1NF if all the under lying domain of attributes contain simple individual values.
* **Second normal form**- The 2NF is based on the concept of full functional dependency. A relation said to be in 2NF if and only if it is in 1NF and every non-key attribute is fully functionally dependent on candidate key of the table.
* **Third normal form**- The 3NF is based on the concept of transitive dependency. A relation in 2NF is said to be in 3NF if every non-key attribute is non-transitively dependent on candidate key of the table. Our project is in the second normal form because every non key attribute is fully functionally dependent on the primary key. A relation is in third normal form, if there is no transitive dependency for non-prime attributes as well as it is in second normal form. o Boyce Codd normal form (BCNF) BCNF is the advance version of 3NF. It is stricter than 3NF. A table is in BCNF if every functional dependency X → Y, X is the super key of the table. For BCNF, the table should be in 3NF, and for every FD, LHS is super key.
* **Fourth normal form**- A relation will be in 4NF if it is in Boyce Codd normal form and has no multi-valued dependency. For a dependency A → B, if for a single value of A, multiple values of B exist, then the relation will be a multi-valued dependency.
* **Fifth normal form**- A relation is in 5NF if it is in 4NF and not contains any join dependency and joining should be lossless. 5NF is satisfied when all the tables are broken into as many tables as possible in order to avoid redundancy. 5NF is also known as Project-join normal form (PJ/NF).

The project Merch Mingle is second normal form (2NF).

## TABLE DESIGN

***Feedback***

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type(Size)** | **Constraints** |
| id | int(50) | Primary key |
| User id | int(50) | Foreign Key |
| ordered | int(50) | Foreign key |
| feedback | int(50) | Not Null |

***Order***

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type (Size)** | **Constraints** |
| id | int(50) | Primary key |
| product id | int(50) | Foreign Key |
| user id | int(50) | Foreign Key |
| vehicle id | int(50) | Foreign Key |
| qty | int(50) | Not Null |
| rate | decimal(50) | Not Null |
| amount | decimal(50) | Not Null |
| date | date(50) | Not Null |
| status | int(50) | Not Null |

***Products***

|  |  |  |
| --- | --- | --- |
| **Field name** | **Data Type (Size)** | **Constraints** |
| id | int(50) | Primary Key |
| shop id | int(50) | Foreign Key |
| rate | int(50) | Not Null |
| stock | int(50) | Not Null |
| photo | varchar | Not Null |
| product name | Varchar(50) | Not Null |

***User***

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type (Size)** | **Constraints** |
| id | Int(50) | Primary Key |
| first\_name | Int(50) | Not Null |
| last\_name | Int(50) | Not Null |
| email | Int(50) | Not Null |
| phone | Int(50) | Not Null |
| address | Varchar(50) | Not Null |
| license\_no | Varchar(50) | Not Null |
| pin\_no | Varchar(50) | Not Null |
| username | varchar (150) | Not Null |
| password | varchar (128) | Not Null |

***Vehicle***

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Constraints** |
| Id | Int(50) | Primary key |
| shop\_id | Int(50) | Foreign key |
| driver name | Varchar(50) | Not Null |
| reg\_num | Varchar(20) | Not Null |
| license | Image(-) | Not Null |
| rc\_book | Image(-) | Not Null |

### Data Dictionary

Data Dictionary is the major component in the structured analysis model of the system. It lists all the data items appearing in DFD. A data dictionary is a file or a set of files that

includes a database’s metadata (hold records about other objects in the database), like data ownership, relationships of the data to another object, and some other data**.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type (Size)** | **Constraints** |
| user\_id | Id of User | int(50) | Primary key |
| first\_name | First name of User | int(50) | Not Null |
| last\_name | Last name of User | int(50) | Not Null |
| email | Email of user | int(50) | Not Null |
| phone | Contact details of  user | int(50) | Not Null |
| address | Address of User | varchar(50 | Not Null |
| license\_no | License Number of  Shop | varchar(50) | Not Null |
| pin\_no | Pin code of User | varchar(50) | Not Null |
| username | Username of User | varchar(150) | Nor Null |
| password | Password of User | varchar(128) | Not Null |
| product\_id | Id of Product | int(50) | Primary key |
| shop\_id | Id of Shop | int(50) | Foreign Key |
| rate | Rate of Product | int(50) | Not Null |
| stock | Stock of Product | int(50) | Not Null |
| photo | Image of the product | varchar(50) | Not Null |
| product\_name | Name of the product | varchar(-) | Not Null |
| ordered | Ordered product | varchar(50) | Not Null |
| id | Id of product | int(50) | Not Null |
| date | Order of date | int(50) | Not Null |
| driver name | Name of driver | varchar(50) | Not Null |
| reg num | Reg num of vehicle | int(50) | Not Null |
| id | Id of vehicle | int(50) | Primary key |

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Description** | **Data Type (Size)** | **Constraints** |
| id | Id of order | int(50) | Primary Key |
| product\_id | Id of product | int(50) | Foreign key |
| user\_id | Id of user | int(50) | Foreign key |
| vehicle\_id | Id of vehicle  assigned | int(50) | Not Null |
| qty | Quantity of ordered  product | int(50) | Not Null |
| rate | Total amount of the  product | decimal(10,2) | Not Null |
| status | Status of ordered  product | int(50) | Not Null |
| feedback | Feedback of the  ordered product | varchar(100) | Not Null |
| license | Image of license | image(-) | Not Null |
| rc book | Image of rc | image(-) | Not Null |
| Order id | Id of order | Int(50) | Not Null |
| amount | Amount of order | Decimal(50) | Not Null |
| id | Order id | Int(50) | Primary Key |

## MODELLING

### Data flow diagram (DFD)

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of

process or information about whether processes will operate in sequence or in parallel.

### Data Flow Diagram Notations

Function:

File/Database:

Input/Output:

Flow of direction:



**DATA FLOW DIAGRAM**

**LEVEL-0/CONTEXT LEVEL**

Request

Response

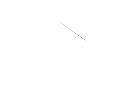
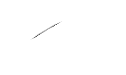
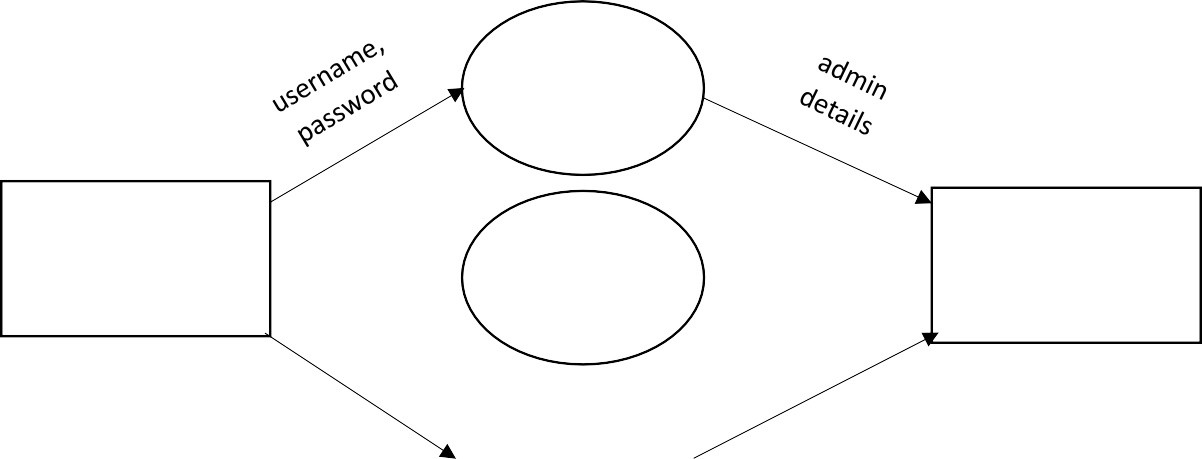
Merch

Mingle

Admin/Customer/ Shopkeeper

Admin/Customer/ Shopkeeper

**LEVEL-1**



1.1

Admin

Admin/Customer/

Shopkeeper

1.2

Admin/Customer/

Shopkeeper

password

Customer

Customer

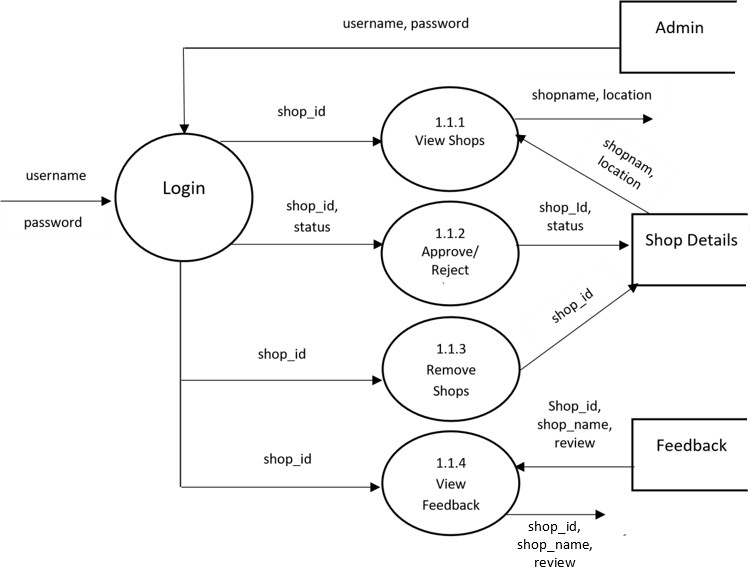
details

1.3

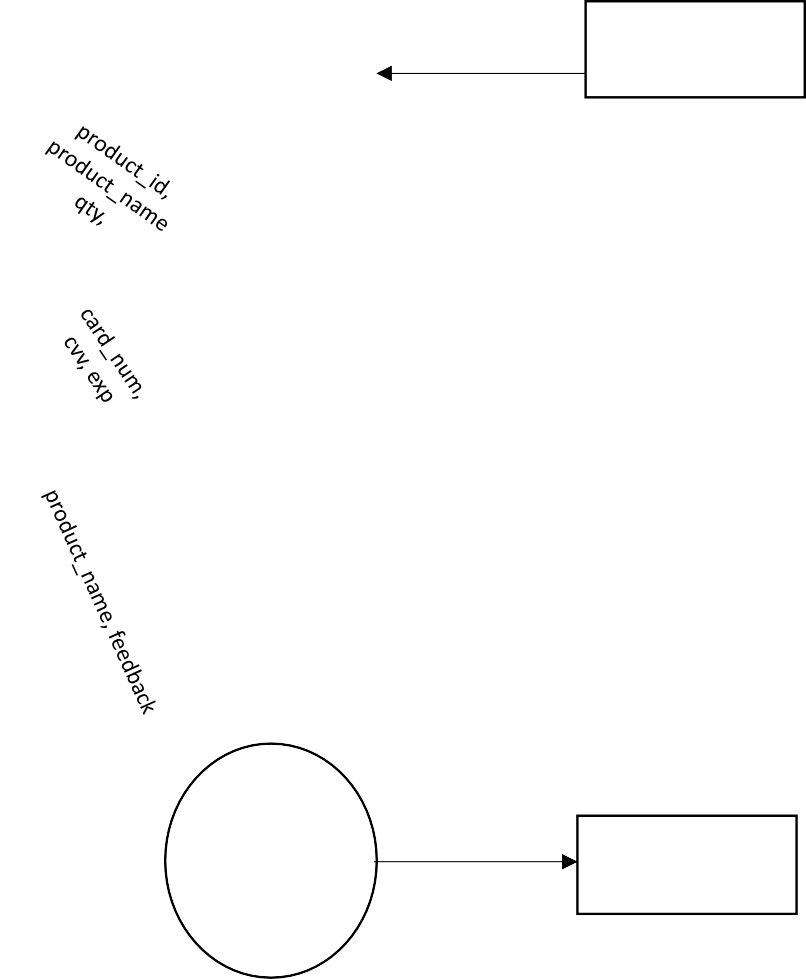
Shopkeeper

username

**LEVEL 1.1 OF ADMIN**



**LEVEL 1.2 OF CUSTOMER**



username, password

username

product\_name

product\_id,

product\_name qty,

Login

1.2.1

View

Product

Cart

password

1.2.2

Make Order

product\_id,

product\_name, qty

1.2.3

Make Payment

card\_num,

cvv, exp

1.2.4

Feedback

Feedback

product\_name

feedback

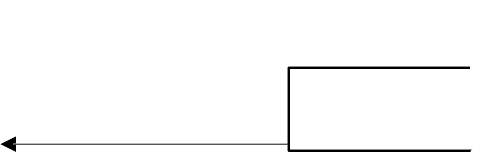
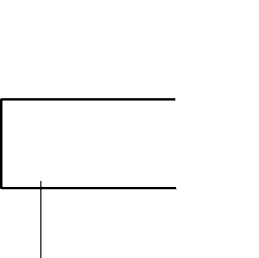
Payment

Customer

product\_id, product\_name, qty

**LEVEL-1.3 OF SHOPKEEPER**

username, password



Shopkeeper

username password

Login

product\_name rate, stock, image

Product\_name

driver\_name, reg\_num, license, rc\_book

1.3.1

Add Product

1.3.2

View Product

1.3.3

Add vehicle

product\_name rate, stock, image

product\_name rate, stock, image

product\_name rate, stock, image

driver\_name, reg\_num,

license, rc\_book

Product

Vehicle

reg\_num

1.3.4

View Vehicle

driver\_name, reg\_num, license, rc\_book

driver\_name, reg\_num, license, rc\_book

order\_id, date

date

1.3.5

View Orders

1.3.6

View Placed Orders

product\_name, date, price, qty

product\_name, date, qty

product\_name, date, price, qty, total

Orders

product\_name, date, price, qty, total

date, order\_id 1.3.7

View Shipped Orders

username, product\_name, date, price, qty, total

username, product\_name, date, qty qty, total

product\_name, date 1.3.8

View Feedback

username, product\_name, feedback

username, product\_name, feedback

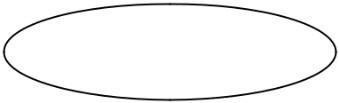
Feedback

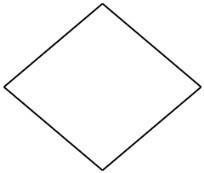
### Entity Relationship Model

An entity relationship diagram is a specialized graphic that illustrates the inter-relationship between entities in a database. ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. An entity– relationship model is the result of using a systematic process to describe and define a subject area of business data. It does not define business process; only visualize business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into zero or more apartments, but one apartment can only be located in one building. Entities may have various properties (attributes) that characterize them. Diagrams created to represent these entities, attributes, and relationships graphically are called entity–relationship diagrams.

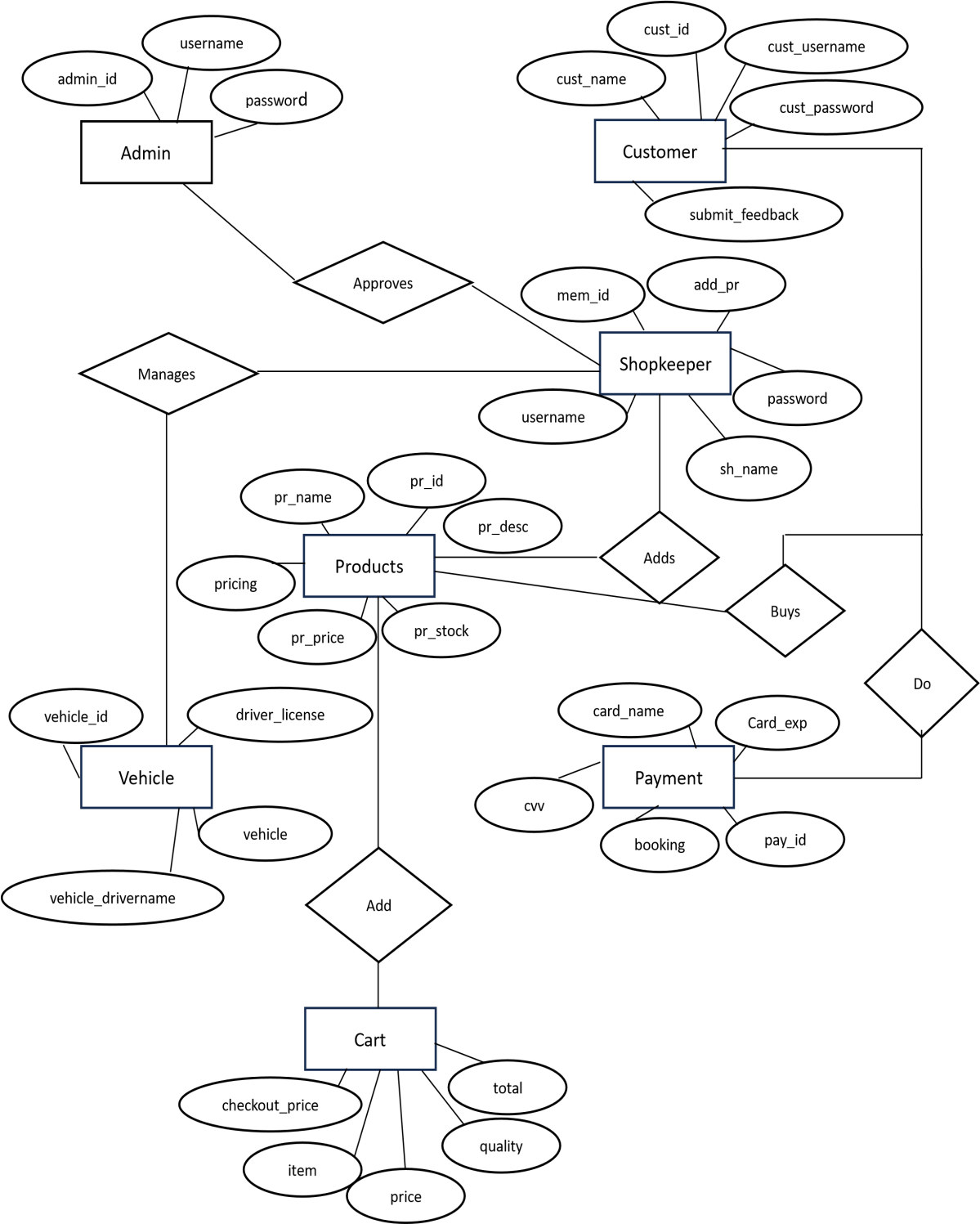
Symbols used in ER diagram: -

Entity:

Attributes:

Relationship: 

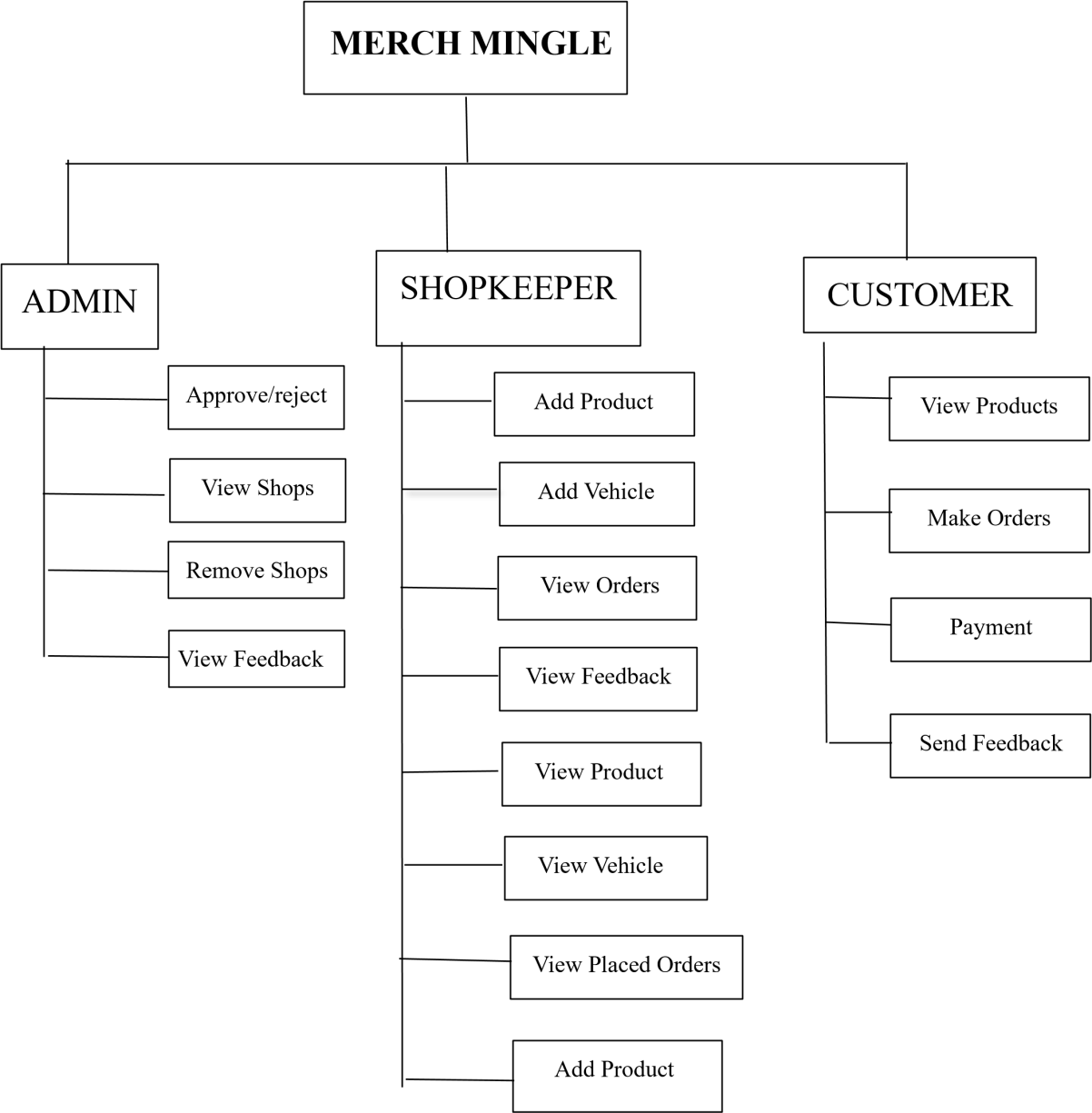
## ENTITY RELATIONSHIP DIAGRAM



## ARCHITECTURAL DESIGN

### Structure Chart

A structure chart (SC) in software engineering and organizational theory, is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structure programs to arrange modules into a tree. Each module is represented by a box, which contains the module’s name. The tree structure visualizes the relationships between the Modules.



## PROCEDURAL DESIGN

### Flow Chart

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analysing, designing, documenting or managing a process or program in various fields.

Symbols used: Flow line:



Terminal:

Process:

Decision:

Input/Output:

Connector:

**FLOW CHART OF MERCH MINGLE**



Start

Enter

username & password

Yes

If Admin

No

If Customer

No

If

shopkeeper

No

Enter the user details.

Yes

Yes

Yes

View Admin Home page

View Customer

Homepage

View Shopkeeper Home page

Registration

A

B

C

Signup

Successful

Login

**FLOW CHART OF ADMIN**



A

No

No

No

Is

Approve/

Reject?

Is View

Shop?

Is Remove

Shop?

Yes

Yes

Yes

Enter Shop

Id

Enter Shop

Id

Enter

Shop Id

Shop

Approved Successfully

Shop

Viewed Successfully

Shop

Removed Successfully

Stop

View Shop

Approve/ Reject

Remove Shop

**FLOWCHART OF SHOPKEEPER**



B

No

No

No

If Add

Product?

If Add

Vehicle?

If View

Feedback

?

Yes

Yes

Yes

Enter

product Details

Enter

Vehicle Details

Enter

Product name,date

Product

Added Successfully

Vehicle

Added Successfully

Viewed

Feedback Successfully

Stop

Add Vehicles

Add product.

View Feedback

**FLOWCHART OF CUSTOMER**



c

No

No

No

If View

Product?

If Make order.

If Made

Payment?

Yes

Yes

Yes

Enter

Product name

Enter

Product name,qty

Enter card

no,cvv

Product

Added Successfully

Product

Purchased Successfully

Viewed

Status Successfully

Stop

Confirm Purchase

View Product

View Order Status

# PROCESSING ENVIRONMENT

## HARDWARE SPECIFICATIONS

### Intel core i7 Processor

An Intel Corei7 is the fastest version of the Intel processor for consumer-end computers and devices. Like the Intel Corei5, the Corei7 is embedded with Intel Turbo Boost Technology. The Intel Corei7 is available in two- to six-core varieties and can support up to 12 different threads simultaneously. Its processor clock speed ranges from 1.70 GHz to up to 3.90 GHz, with cache memory from 4 to 12 MB. Intel Corei7 thermal design power (TDP) range goes from 130 watts TDP to as low as 15 watts TDP. A Core i7 will typically be better for multitasking, media- editing and media-creation tasks, high-end gaming, and similar demanding workloads.

### Processor speed

With technology, increased productivity goals, faster internet, and more devices, we’ve created a need for speed wherever we go. We’re used to getting results instantaneously and expect our devices to keep up with our requests as we multi-task our way through life. Computer processors and their clock speed are two features we most commonly associate with high performing, fast technology. Computer processor speed (CPU speed) is one of the most important elements to consider when comparing computers. The CPU is often referred to as “the brain” of your computer, so ensuring it’s working properly is very important to the longevity and functionality of your computer. Understanding what makes a good processor speed starts with understanding what exactly a processor does - and what its components do to improve the functionality of your computer.

## RAM

RAM (Random Access Memory) is the hardware in a computing device where the operating system (OS), application programs and data in current use are kept so they can be quickly reached by the device's processor. RAM is the main memory in a computer. This memory stores a temporary data. Temporary data means as long as your computer system is running, your data is stored in RAM and as soon as your computer is shut down, the data stored in RAM gets deleted. RAM is volatile in nature, which means, the data is lost when the device is switched off. RAM is known as the Primary memory of the computer. RAM is known to be expensive since the memory can be accessed directly. RAM is the fastest memory; therefore, it is an internal memory for the computer.

### Hard Disk Capacity

A computer hard disk drive (HDD) is a non-volatile data storage device. Non-volatile refers to storage devices that maintain stored data when turned off. All computers need a storage device, and Hard Disk Drive are just one example of a type of storage device. HDD are usually installed inside desktop computers, mobile devices, consumer electronics and enterprise storage arrays in data centres. They can store operating systems, software programs and other files using magnetic disks.

### Keyboard

A multimedia keyboard is one with media keys — additional buttons, typically along the top, for controlling audio playback, for starting common applications (e.g., e-mail client and Web browser) and other auxiliary functionality. Many such keyboards also contain a volume knob, implemented as a rotary encoder so as to be able to provide relative volume changes regardless of the volume level set by the user in the operating system and application

software.

### Mouse

A mouse is a small device that a computer user pushes across a desk surface in order to point to a place on a display screen and to select one or more actions to take from that position. The mouse first became a widely-used computer tool when Apple Computer made it a standard part of the Apple Macintosh.

## USB

USB 2.0, also known as hi-speed USB was introduced in 2000. It is an updated version of USB 1.1, which provides improved functionalities and better speed. It is capable to deliver the maximum transfer speed of 480 Megabits per second. However, practically it is approximately 280 Mbps. USB 3.0, also known as SuperSpeed USB was first made available in November 2009. It is a much-improved version of USB 2.0. It supports the data transfer rate of 5 Gigabits per second, which is much faster than the speed provided by USB 2.0.

## SOFTWARE SPECIFICATIONS

### Windows 11

Windows 11 is the latest major release of Microsoft's Windows NT operating system, released in October 2021. It is a free upgrade to its predecessor, Windows 10 (2015), available for any Windows 10 devices that meet the new Windows 11 system requirements. Windows 11. A version of the Windows NT operating system.

## HTML 5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and final major HTML version that is a World Wide Web Consortium recommendation. The current specification is known as the HTML Living Standard.

## CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

### Angular 14

Angular, a typescript-based web application framework is Google’s one of the brilliant creations. Angular 14 is one of the most systematic pre-planned upgrades of Angular. Angular 14 comes with typed reactive forms, CLI auto compiled, directives, and a developer preview of standalone components. The latest version of Angular 14 comes with better template diagnostics, allowing Angular developers to be protected from common errors by

the compiler, similar to how TypeScript code is protected.

### SQLite 3

SQLite 3 is a lightweight, serverless, self-contained SQL database engine that excels in simplicity, reliability, and efficiency. It's perfect for projects like "Merch Mingle" due to its easy setup, minimal configuration, and seamless integration with various programming languages, including Python.

### JavaScript

JavaScript, often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for web page behaviour, often incorporating third- party libraries. JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behaviour. It is used to enhance HTML pages and is commonly found embedded in HTML code. JavaScript is an interpreted language. Thus, it doesn't need to be compiled. JavaScript renders web pages in an interactive and dynamic fashion.

### Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, Mac-OS and Linux. It comes with built-in support for JavaScript, TypeScript and Node.

### Python

Python serves as a versatile backbone for Merch Mingle, contributing to various facets of the project. Leveraging Python's frameworks like Django or Flask facilitates robust backend development, enabling efficient handling of web requests and database management. Moreover, Python's extensive libraries such as pandas and NumPy empower data processing and analysis, facilitating insightful decision-making based on user behaviours and trends. Additionally, Python excels in web scraping tasks through libraries like Beautiful Soup and Scrapy, ensuring seamless data collection from diverse online sources.

**Node JS**

In Merch Mingle, Node.js functions as the fundamental framework, ensuring the platform's dynamism and efficiency. Its asynchronous capabilities enable seamless handling of concurrent requests, guaranteeing a smooth user experience. Node.js's event-driven architecture supports scalability in the backend, effortlessly managing data processing and database interactions. Leveraging its extensive library ecosystem, notably Express.js, facilitates streamlined development and implementation of essential features like routing and middleware. Additionally, Node.js facilitates real-time communication functionalities crucial for Merch Mingle, including live updates, chat, and notifications. Overall, Node.js empowers the creation of a robust, responsive, and feature-rich web application, enhancing user engagement and satisfaction.

# CODING AND IMPLEMENTATION

## CODING

***Admin Login (login.py)***

def logins(request):

if request.method=='POST':

usern = request.POST.get('username') passw = request.POST.get('password')

user = authenticate(request,username=usern,password=passw) if user is not None:

request.session['forValidation'] = 's'

if user is not None and user.is\_staff == 0 and user.is\_superuser == 0: login(request,user)

request.session['userId'] = user.id print(' ')

return redirect('userhome')

elseif user is not None and user.is\_superuser == 1: login(request,user)

return redirect('adminhome')

elseif user is not None and user.is\_superuser == 0 and user.is\_staff==1: login(request,user)

request.session['shopId'] = user.id return redirect('shophome')

else:

return HttpResponse('Sorry Invalid details/ User is not Varified') else:

return HttpResponse('Sorry Invalid details')

# return render\_to\_response('template\_name', message='Save complete') else:

return render(request, 'main/login.html') ## for user Home

@login\_required(login\_url='loginpage') def userhome(request):

if request.session.has\_key('forValidation'): return render(request,'main/userhome.html')

else:

return redirect('logouts') ## for Admin Home

@login\_required(login\_url='loginpage') def adminhome(request):

if request.session.has\_key('forValidation'): toActiveShop=User.objects.filter(is\_active=False).count() print(toActiveShop,' ')

ActiveShop=User.objects.filter(is\_active=True, is\_staff=True, is\_superuser=False).count()

ActiveShop=ActiveShop print(ActiveShop,' ')

return render(request,'main/adminhome.html',{"toActiveShop":toActiveShop,"ActiveShop":ActiveS hop})

else:

return redirect('logouts') @login\_required(login\_url='loginpage') def shophome(request):

if request.session.has\_key('forValidation'): if 'shopId' in request.session:

shopid = request.session['shopId'] sid=User.objects.get(id=shopid)

order\_count = Order.objects.filter(product\_id shop\_id\_id=sid, status=1).count() print("order\_count:", order\_count)

Placed\_count = Order.objects.filter(product\_id shop\_id\_id=sid, status=2).count() print("Placed\_count:", Placed\_count)

Maid\_Payment = Order.objects.filter(product\_id shop\_id\_id=sid, status=3).count() print("Placed\_count:", Maid\_Payment)

Shipped = Order.objects.filter(product\_id shop\_id\_id=sid, status=4).count() print("Placed\_count:", Shipped)

Delivered = Order.objects.filter(product\_id shop\_id\_id=sid, status=5).count() print("Placed\_count:", Delivered)

return render(request,'main/shophome.html',{'order\_count':order\_count,'Placed\_count':Placed\_count

'Maid\_Payment':Maid\_Payment,'Shipped':Shipped,'Delivered':Delivered})

else:

return redirect('logouts')

***Payment (payment.py)***

def UserMakePayment(request): if request.method=="POST":

connection = mysql.connector.connect(host='localhost',

database='dreamzbank', user='root', password='root', port='3306')

cursor = connection.cursor()

cursor.execute("select Name,accountNo from tbl\_accountdetails")

myresult = cursor.fetchone() print(myresult) paymentToName = myresult[0]

paymentToAccountNo = myresult[1] userId = request.session['userId']

amount= Order.objects.filter(user\_id=userId).filter(status=2).aggregate(Sum('amount')) amountToPay=str(amount['amount sum'])

print('////////////////////////////////////////////////',amountToPay) if amountToPay!="None":

request.session['amount']=amountToPay

alert\_list= Order.objects.filter(user\_id=userId).filter(status=2) print('||||||||||| is not none||||||||')

tmpJson = serializers.serialize("json",alert\_list) tmpObj = json.loads(tmpJson)

orderid=[]

for i in tmpObj: orderid.append( i['pk'])

request.session['purcahcedItems']=orderid

# print(' ',paymentToName,paymentToAccountNo,amount['amount sum'])

# Session["amount"] = lblTotal.Text response = "~/Customer/test2.aspx"

# Response.Redirect("../GateWay/paymentGateWay.aspx?amount=" + lblTotal.Text + "&paymentToName=" + paymentToName + "&paymentToaccountNo=" + paymentToAccountNo + "&response=" + response + "");

# return

render(request,'main/paymentGateWay.html',{'amount':amountToPay,'paymentToName':pay mentToName,'paymentToaccountNo':paymentToAccountNo})

return redirect('OnlineBanking\_bank\_paymentGateWay',amount=amountToPay,paymentToName=p aymentToName,paymentToaccountNo=paymentToAccountNo)

# return redirect('OnlineBanking\_bank',pk=amountToPay,c=paymentToName) else:

messages.success(request, 'No Items In Cart for Payment') return redirect('userhome')

else:

order={} try:

if 'userId' in request.session:

userId = request.session['userId'] data=Order.objects.filter(user\_id=userId).filter(status=2) # print(' ',data)

# sum = Sale.objects.filter(type='Flour').aggregate(Sum('column'))['column sum']

# sum = Order.objects.filter(user\_id=userId).filter(status=2) sum =

Order.objects.filter(user\_id=userId).filter(status=2).aggregate(Sum('amount')) # print(' ',sum)

if data.exists():

order={'order':data ,'totalAmount':sum } else:

messages.success(request, 'No Items In Cart For Payment')

except Exception as e: print(e)

pass return render(request,'main/userCartPayment.html',order) return render(request,'main/userCartPayment.html',order)

***FeedBack (FeedBack.py)***

const {config} = require("dotenv"); const mongoose = require("mongoose"); const logger = require("../utils/logger")

config();

const username = process.env.DATABASE\_USERNAME; const password = process.env.DATABASE\_PASSWORD; const url =

`mongodb+srv://rajalex881:alexraj2002@cluster0.4xnga7i.mongodb.net/login1`; const db = mongoose.connection; exports.connect = () =>

mongoose.set("strictQuery", false); mongoose.connect(url, { useNewUrlParser: true, useUnifiedTopology: true, });

};

def userFeedBack(request):

if 'userId' in request.session:

userId = request.session['userId'] user=User.objects.get(id=userId) if request.method == "GET":

order = Order.objects.select\_related('product\_id').filter(status=5, user\_id\_id=userId) print(order,'////////////')

return render(request,'main/userFeedBack.html',{'order':order}) else:

o\_id = request.POST.get('order\_id') feedback = request.POST.get('feedback') order=Order.objects.get(id=o\_id) fb=FeedBack()

fb.user\_id=user fb.order\_id=order fb.feedback=feedback fb.save()

messages.success(request, 'Successfully Feedback Added!')

order.status=6 order.save()

return redirect('userFeedBack') def dictfetchallFeedback(shopid):

"Return all rows from a cursor as a dict" with connection.cursor() as cursor:

cursor.execute(""" select \*, Home\_order.id as order\_id from Home\_feedback join Home\_order on Home\_order.id=Home\_feedback.order\_id\_id

join Home\_products on Home\_products.id=Home\_order.product\_id\_id join Home\_user on Home\_feedback.user\_id\_id=Home\_user.id

where Home\_products.shop\_id\_id = %s; """, (shopid,))

columns = [col[0] for col in cursor.description] return [

dict(zip(columns, row)) for row in cursor.fetchall()

]

def shopViewFeedBack(request):

if request.session.has\_key('forValidation'): if request.method=="GET":

if 'shopId' in request.session:

shopid = request.session['shopId'] feedbacks=dictfetchallFeedback(shopid) print(feedbacks)

return render(request,'main/shopViewFeedBack.html',{'f':feedbacks})

***Add Vehicle(Add Vehicle.py)***

@login\_required(login\_url='loginpage') def addvehicle(request):

if request.session.has\_key('forValidation'): if request.method=="GET":

return render(request,'main/vehicle.html') else:

vehicle=Vehicle() vehicle.Drivername=request.POST.get('Drivername') vehicle.reg\_num=request.POST.get('reg\_num') vehicle.license=request.FILES['license'] vehicle.Rc=request.FILES['Rc']

if 'shopId' in request.session:

shopid = request.session['shopId'] print(' ')

vehicle.shop\_id=User.objects.get(id=shopid) # print(' ',product.shop\_id)

vehicle.save()

messages.success(request, 'Successfully Vehicle Added!') return redirect('addvehicle')

else:

return redirect('logouts')

***Add To Cart(Add To Cart.py)***

# add to cart @login\_required(login\_url='loginpage') def product\_tocart(request,id):

if request.session.has\_key('forValidation'): try:

if 'userId' in request.session:

userId = request.session['userId'] user=User.objects.get(id=userId) product = Products.objects.get(id=id) order=Order()

order.user\_id=user order.product\_id=product order.qty=1 order.rate=product.rate order.amount=product.rate order.save()

messages.success(request, 'Successfully Added to the Cart !') except Exception as e:

print(e) pass

return redirect('order') else:

return redirect('logouts')

***Shop View Shipped Orders (shipped orders.py)***

def ShopViewUserShippedOrder(request):

if request.session.has\_key('forValidation'): if request.method=="GET":

if 'shopId' in request.session:

shopid = request.session['shopId']

order=dictfetchallShippedOrders(shopid) # print(order)

data=Vehicle.objects.filter(shop\_id=shopid) vehicle={

'vehicle':data

}

print(data,'////////////')

return render(request,'main/ShopViewUserShippedOrder.html',{'f':order,'vehicle':data})

***User Feedback (User Feedback.py)***

def userFeedBack(request):

if 'userId' in request.session:

userId = request.session['userId'] user=User.objects.get(id=userId) if request.method == "GET":

order = Order.objects.select\_related('product\_id').filter(status=5, user\_id\_id=userId) print(order,'////////////')

return render(request,'main/userFeedBack.html',{'order':order}) else:

o\_id = request.POST.get('order\_id') feedback = request.POST.get('feedback') order=Order.objects.get(id=o\_id) fb=FeedBack()

fb.user\_id=user fb.order\_id=order fb.feedback=feedback fb.save()

messages.success(request, 'Successfully Feedback Added!') order.status=6

order.save()

return redirect('userFeedBack')

***Admin Home(Admin Home.py)***

def adminhome(request):

if request.session.has\_key('forValidation'): toActiveShop=User.objects.filter(is\_active=False).count() print(toActiveShop,' ')

ActiveShop=User.objects.filter(is\_active=True, is\_staff=True,

is\_superuser=False).count() ActiveShop=ActiveShop print(ActiveShop,' ')

return render(request,'main/adminhome.html',{"toActiveShop":toActiveShop,"ActiveShop":ActiveS hop})

else:

return redirect('logouts')

***Shop Home(Shop Home.py)*** @login\_required(login\_url='loginpage') def shophome(request):

if request.session.has\_key('forValidation'): if 'shopId' in request.session:

shopid = request.session['shopId'] sid=User.objects.get(id=shopid)

order\_count = Order.objects.filter(product\_id shop\_id\_id=sid, status=1).count()

print("order\_count:", order\_count)

Placed\_count = Order.objects.filter(product\_id shop\_id\_id=sid, status=2).count()

print("Placed\_count:", Placed\_count)

Maid\_Payment = Order.objects.filter(product\_id shop\_id\_id=sid, status=3).count()

print("Placed\_count:", Maid\_Payment)

Shipped = Order.objects.filter(product\_id shop\_id\_id=sid, status=4).count()

print("Placed\_count:", Shipped)

Delivered = Order.objects.filter(product\_id shop\_id\_id=sid, status=5).count()

print("Placed\_count:", Delivered) return

render(request,'main/shophome.html',{'order\_count':order\_count,'Placed\_count':Placed\_count, 'Maid\_Payment':Maid\_Payment,'Shipped':Shipped,'Delivered':Delivered})

else:

return redirect('logouts') ***Store register (Store Register.py)*** def Storeregister(request):

if request.method=='POST':

store\_name = request.POST.get('store\_name')

first\_name = request.POST.get('first\_name') last\_name = request.POST.get('last\_name') # name=str(first\_name)+ ' '+str(last\_name) email = request.POST.get('email')

phone = request.POST.get('phone') address = request.POST.get('address') passw = request.POST.get('password') licenceNo = request.POST.get('licenceNo')

User.objects.create\_user(first\_name=first\_name,last\_name=last\_name, email= email,name=store\_name,phone=phone,address=address, username=email,password=passw, is\_staff=1, is\_active=0,licenceNo=licenceNo)

# messages.success(request, 'Successfully Registered!')

# return render(request,'main/login.html', {'some\_flag': True}) return redirect(logins)

else:

return render(request,'main/storeAdd.html')

***Logout (Logout.py)***

def logouts(request):

if request.session.has\_key('forValidation'): del request.session['forValidation']

logout(request)

return redirect(logins) else:

return redirect(logins)

***Shop Approve (Shop Approve.py)***

def shop\_approve(request,id):

if request.session.has\_key('forValidation'): user = User.objects.get(id=id) user.is\_active=1

user.save()

messages.success(request, 'Shop Successfully Approved!') return redirect('loadShops')

else:

return redirect('logouts')

***Shop Reject (Shop Reject.py)***

def shop\_reject(request,id):

if request.session.has\_key('forValidation'): user = User.objects.get(id=id) user.delete()

messages.success(request, 'Shop Successfully Rejected!') return redirect('loadShops')

else:

return redirect('logouts')

***Add Product to Cart (Product To Cart.py)***

def product\_tocart(request,id): if

request.session.has\_key('forValidation'): try:

if 'userId' in request.session:

userId = request.session['userId'] user=User.objects.get(id=userId) product = Products.objects.get(id=id) order=Order()

order.user\_id=user order.product\_id=product order.qty=1 order.rate=product.rate order.amount=product.rate order.save()

messages.success(request, 'Successfully Added to the Cart !') except Exception as e:

print(e) pass

return redirect('order') else:

return redirect('logouts') ***Delete Cart (Delete from Cart.py)*** def delete\_fromcart(request):

if request.session.has\_key('forValidation'):

if request.is\_ajax and request.method == "GET":

# get the nick name from the client side cartId = request.GET.get("id", None) print(cartId)

order = Order.objects.get(id=cartId) order.delete()

return redirect(placeorder) else:

return redirect('logouts')

***Place Order (PlacingOrderbyshop.py)***

def placingOrderByShop(orderDetails): "Return all rows from a cursor as a dict" with connection.cursor() as cursor:

for i in orderDetails: print('//////////////////')

cursor.execute(""" update Home\_order set status=2 where id= %s; """, (i['order\_id'],) )

cursor.execute(""" select product\_id\_id, qty from Home\_order where id= %s; """, (i['order\_id'],) )

details=cursor.fetchone()

cursor.execute(""" update Home\_products set stock = (select stock from Home\_products where id=%s)- %s where id= %s;

""", (details[0],details[1],details[0],) ) # cursor.commit()

# print(i['order\_id']);

## TESTING

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also. The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of the testing phase is to detect the errors that may be present in the program. Hence one should not start testing with the intent of showing that a program works, but the intent should be to show that a program doesn’t work. Testing is the process of executing a program with the intent of finding errors.

### Unit Testing

Unit testing focuses verification effort on the smallest unit of software i.e., the module. Using the detailed design and the process specifications, unit testing is done to uncover errors within the boundary of the module. All modules must be successful in the unit test before the integration testing starts. In this project, each service can be thought of as a module. There are 3 modules like Admin, Customer, and Shopkeeper. While developing the modules, unit testing is done to ensure that each module works without any errors. The inputs are validated after accepting from the user. In this project, the developer tests the programs as system. Software units in a system are the modules and routines that are assembled and integrated to form a specific function. Unit testing is first done on modules, independent of one another to locate errors. This enables us to detect errors. Through these, errors resulting from interaction between modules are initially avoided.

**Black Box Testing:** In the black box testing, test cases are designed from the examination of the input/output values only and no knowledge of design or code is required. The main approaches are:

* + - * Equivalent class partitioning
      * Boundary value analysis

**White Box Testing:** White box testing is an important type of unit testing. Many white box testing strategies exist. White box testing strategies are:

* + - * Fault based testing.
      * Coverage Based testing.

### Integration Testing

Integration testing is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers. The aim of integration testing is to test the interfaces between the modules and expose any defects that may arise when these components are integrated and need to interact with each other.

### Types of test cases

**Integration test case**: Here we execute test cases which just tell about the connectivity from one module to another module and integrating one application to another application and how the application moves from parent node to child no demand vice versa.

**Functional Test case**: Here we execute test cases which tell about the functionality of the application and talk about the desired output to be seen. Internally we have different types of test cases when we write here like range, output values, BVA, ECP and so on. We give input and expect some output according to the SRS. Here we check individual module completely by checking each tab, text box, and buttons and so on.

**Non-Functional Test case**: Test cases related to user friendliness like font, image, colour, how easy to navigate etc., performance related, security related comes under here.

**User Acceptance test case**: These test cases are crucial and very important to client-side people because these test case talks about these business and approach of the application to complete a particular client task, which is also called as End to End Business scenario test case. Here we won’t be doing testing related to UI, Functional or Non-Functional, we talk about business and scenario for which the application is made for.

### System Testing

Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. The entire software has been tested against the requirements of project and it is checked whether all requirements of project have been satisfied or not.

**Alpha Test:** The first test of newly developed, when the first round of bugs has been fixed, the product goes with actual users for testing. For custom software the customer may be invited into the vendor's facilities for an alpha test to ensure the client's vision has been interpreted properly by the developer.

**Beta Test:** A test of new or revised software application that is performed by users at their facilities under normal operating conditions. Beta testing follows alpha testing. Vendors of packaged software often offer their customers the opportunity of beta testing new releases or versions and the beta testing of elaborate products such as operating systems can take months. **Acceptance testing:** Testing is performed by the Client of the application to determine whether the application is developed as per the requirements specified by him/her. It is performed within the development of the organization or at the client site.

# SECURITY, BACKUP AND RECOVERY MECHANISMS

### Security

This application deals with a simple authorization mechanism i.e., username and password. The user just must register into the application with his username and password. The application can only be operated by a registered user and with his account only. Admin can accept or decline the user request based on the authenticity of the data entered. Security in "Merch Mingle" includes user authentication, data encryption, firewalls, intrusion detection, and regular audits to protect user information. Compliance with PCI DSS ensures secure payment processing. These measures safeguard data and maintain the website's integrity against potential threats and breaches.

### Backup

In any case where the user encounters a problem with the application, support will be provided to the user for fixing the problem. Backup source code for our application is always kept. “Merch Mingle” employs data backups, redundancy strategies, and a disaster recovery plan. Automated backups preserve data integrity. Redundancy ensures high availability, while the disaster recovery plan outlines steps for system restoration in unforeseen events, ensuring service continuity.

### Recovery

In any case the application crashes the application just needs to be refreshed to be back online. In the worst case refreshing the system might prove helpful and solves the problem. If the application has any problems, there are other recovery options as well. “Merch Mingle” utilizes automated backups, storing critical data offsite in secure locations to mitigate the risk of data loss. Version control tracks code changes, enabling easy rollbacks. A data retention policy ensures efficient storage management, enhancing data security and system resilience.

# FUTURE ENHANCEMENT

In the future, the Merch Mingle project could undergo several enhancements aimed at improving user experience, expanding market reach, and incorporating cutting-edge technologies. One significant upgrade could involve implementing advanced recommendation algorithms to provide personalized product suggestions to customers. By analysing browsing history, purchase patterns, and user preferences, the platform could offer tailored recommendations, increasing the likelihood of customer engagement and satisfaction. Integration of AI-powered chatbots represents another promising enhancement. These chatbots could handle customer inquiries, provide real-time support, and help throughout the shopping journey. By leveraging natural language processing and machine learning, chatbots could streamline communication, reduce response times, and enhance overall customer service efficiency. expanding the platform's capabilities to support international transactions and multiple currencies could also be a valuable addition. This enhancement would enable merchants to reach a broader audience and facilitate cross-border sales, thereby driving business growth and increasing revenue opportunities. Implementing robust payment gateways and complying with international regulations would be crucial aspects of this expansion. Furthermore, incorporating virtual try-on features for products such as apparel and accessories could revolutionize the online shopping experience. Utilizing augmented reality (AR) technology, customers could virtually visualize products in real-world settings, enabling them to make more informed purchasing decisions and reducing the likelihood of returns. Lastly, leveraging AR technology for product visualization could further enhance the platform's offerings. By allowing customers to visualize products in their physical environment before making a purchase, AR could increase confidence in product selection and enhance overall satisfaction.

# SOFTWARE MAINTENANCE

Software maintenance is a widely accepted part of SDLC now days. It stands for all the modifications and updates done after the delivery of software product. Software Maintenance planning includes ten activities:

* **Preparation** – Describe software preparation and transition activities including the conception and creation of the maintenance plan; describe how to handle problems identified during development and configuration.

management.

* **Modification** – After the application has become the responsibility of the maintenance team, explain how to analyse each request; confirm and check validity; investigate and propose solutions; document the proposal and get the required authorizations to apply the modifications.
* **Implementation** – Describe the process for considering the implementation of the modification itself.
* **Acceptance** – Describe how the modification is accepted by the maintenance team.
* **Migration** – Describe any migration tasks that need to be executed. If the software needs to be moved to another system, outline the steps to do so

without impacting its functionality.

* **Transition** – Document the sequence of activities to transition the system from Development to Maintenance.
* **Service Level Agreements** – Document SLAs and maintenance contracts negotiated by Maintenance.
* **Change Request** – Outline the problem-handling process to prioritize, documents and route change and maintenance requests.
* **Modification Request acceptance**/**reject**– Describe the request including details of the size/effort/complexity. If this is too complex to resolve, outline the steps to route the issue back to the software team.
* **Retirement** – This is the final stage in the lifecycle. Describe how to retire the software and the steps to archive any data that may be a by-product of the system.

### Types of Maintenance

Traditionally, 5 types of maintenance have been distinguished, which are differentiated by the nature of the tasks that they include:

**Corrective maintenance:** The set of tasks is destined to correct the defects to be found in the different equipment and that are communicated to the maintenance department by users of the same equipment.

**Preventive Maintenance:** Its mission is to maintain a level of certain service on equipment, programming the interventions of their vulnerabilities in the most opportune time. It is used to be a systematic character, that is, the equipment is inspected even if it has not given any symptoms of having a problem.

**Predictive Maintenance:** It constantly pursues know and report the status and operational capacity of the installations by knowing the values of certain variables, which represent such state and operational ability. To apply this maintenance, it is necessary to identify physical variables (temperature, vibration, power consumption, etc.) and to understand which type of variation is indicative of problems that may be appearing on the equipment. This maintenance is the most technical, since it requires advanced technical resources, and at times of strong mathematical, physical and / or technical knowledge.

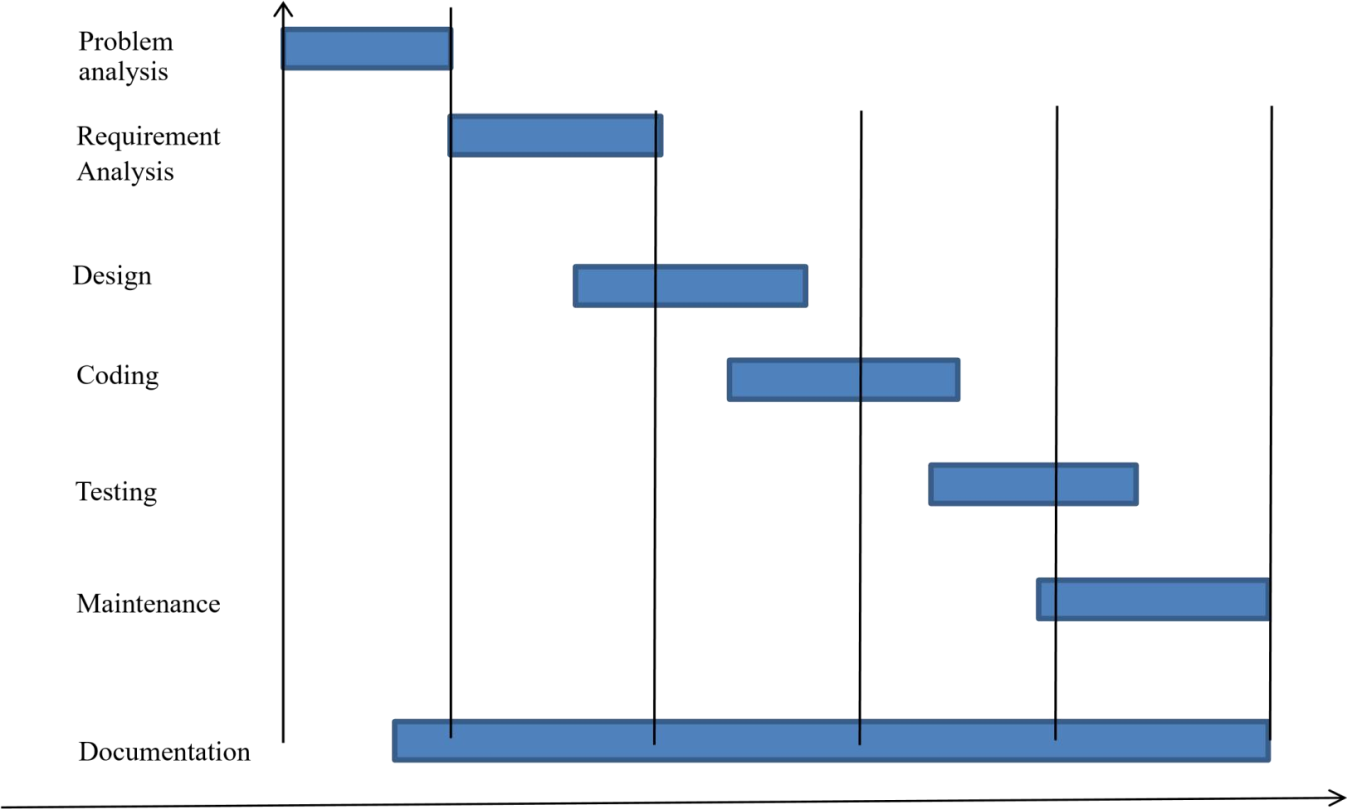
**Zero Hours Maintenance (Overhaul):** The set of tasks whose goal is to review the equipment at scheduled intervals before appearing any failure, when the reliability of the equipment has decreased considerably so it is risky to make forecasts of production capacity. This review is based on leaving the equipment to zero hours of operation, that is, as if the equipment were new. These reviews will replace or repair all items subject to wear.

# CONCLUSION

In conclusion, the development of Merch Mingle marks a significant milestone in revolutionizing the online shopping experience. Through meticulous planning, innovative design, and collaborative effort, we have successfully created a comprehensive platform that connects merchants with customers in a seamless and engaging manner. The implementation of features such as user-friendly interfaces, robust security measures, and efficient transaction processing underscores our commitment to delivering a reliable and user-centric e-commerce solution. Moreover, the incorporation of advanced technologies like AI-driven recommendation systems and AR-enabled product visualization showcases our dedication to staying at the forefront of industry trends and meeting evolving customer needs. As we reflect on our journey, it's evident that the success of Merch Mingle is not merely measured by its functionality but by the positive impact it has on both merchants and customers alike. By fostering a dynamic marketplace environment that fosters growth, fosters innovation, and fosters community, we have laid the foundation for a thriving online ecosystem where businesses can thrive, and customers can find the products they love. Looking ahead, we remain committed to continual improvement and evolution, embracing new challenges and opportunities as they arise. Together, we can continue to shape the future of e-commerce, empowering merchants to succeed and delighting customers at every step of their shopping journey. With gratitude for the collective effort invested in this project, we celebrate our achievements and eagerly anticipate the exciting journey ahead.

# APPENDIX

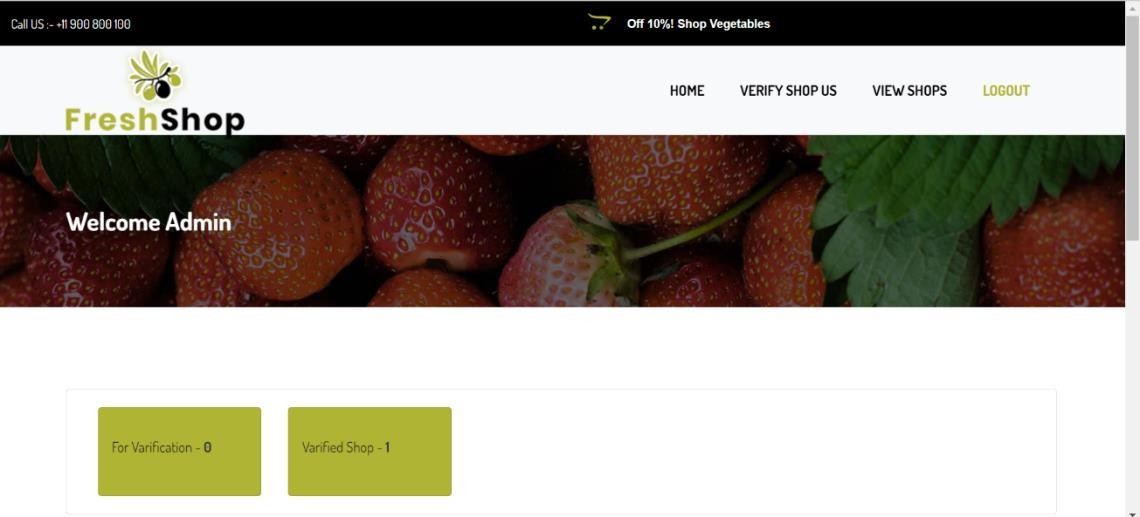
## GANTT CHART



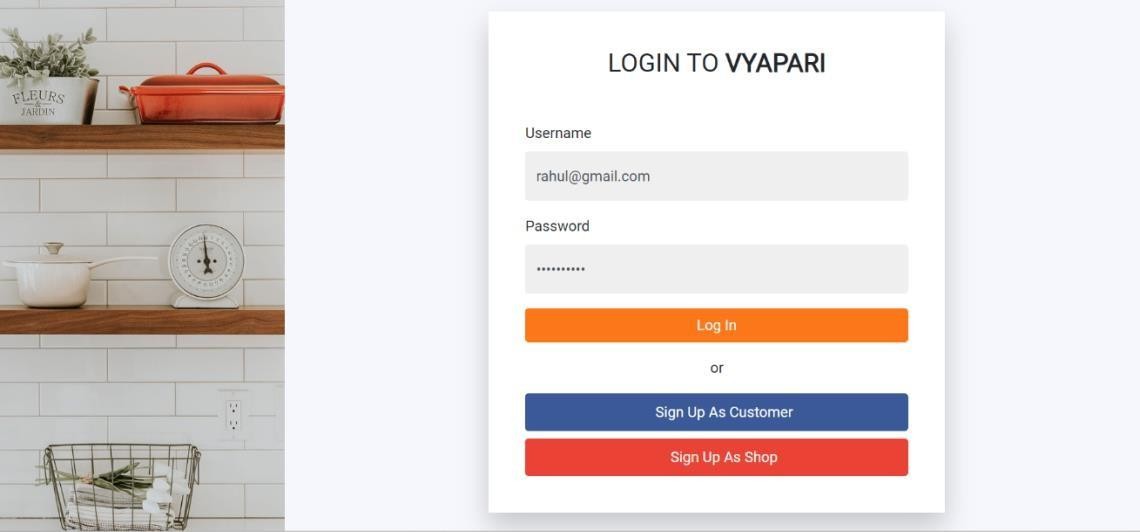
06/11/2023 15/12/2023 29/12/2023 19/01/2024 16/02/2024 15/03/2024

## SCREENSHOTS

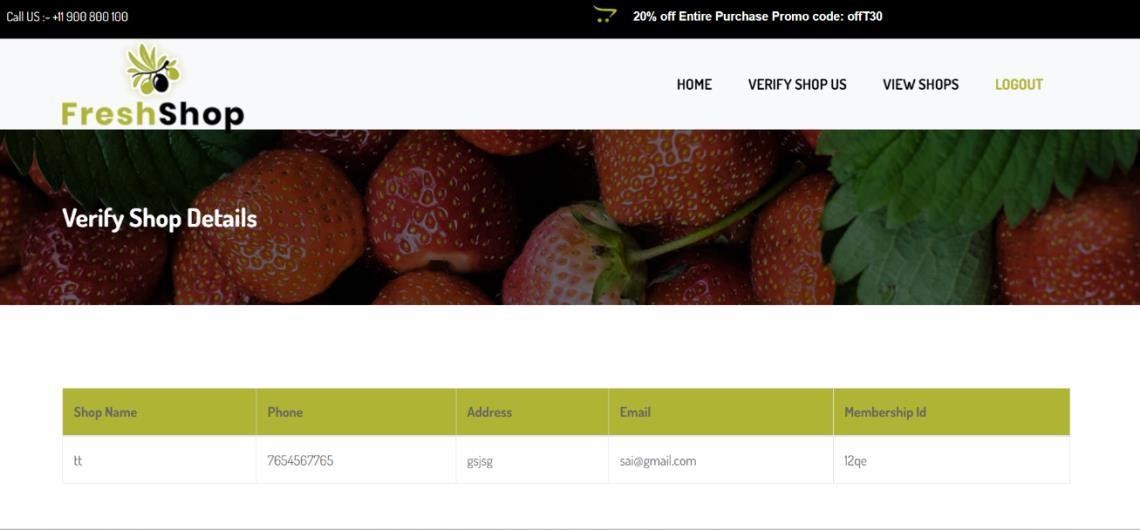
***Admin Homepage***



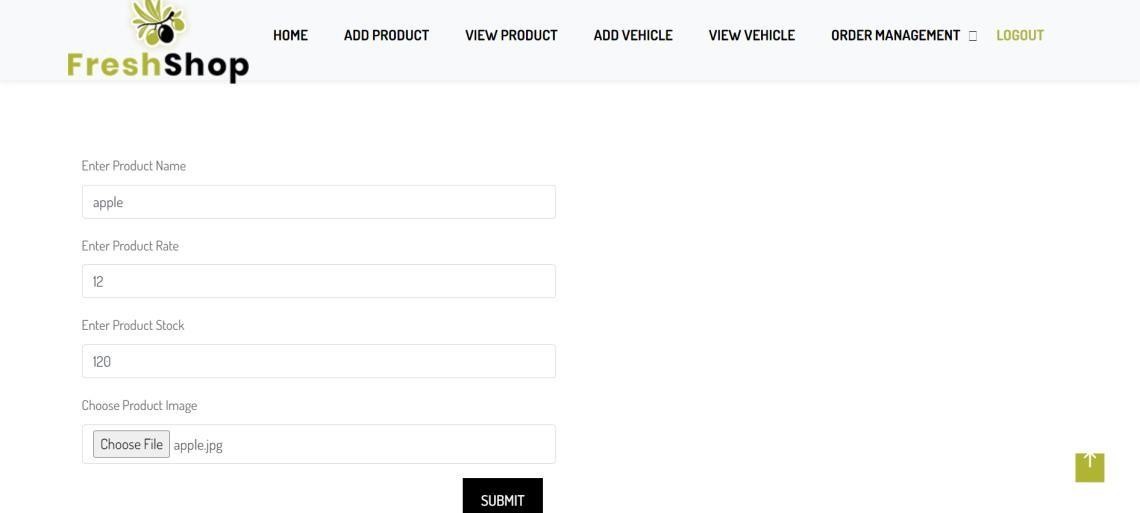
***Customer/Admin Login***



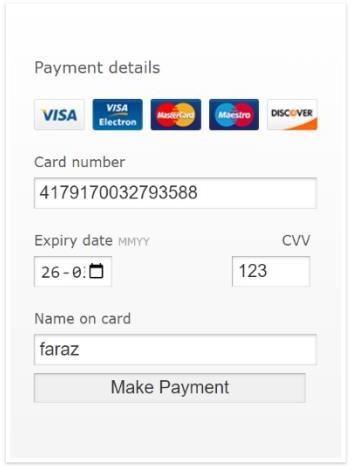
***Admin View(View Shops)***



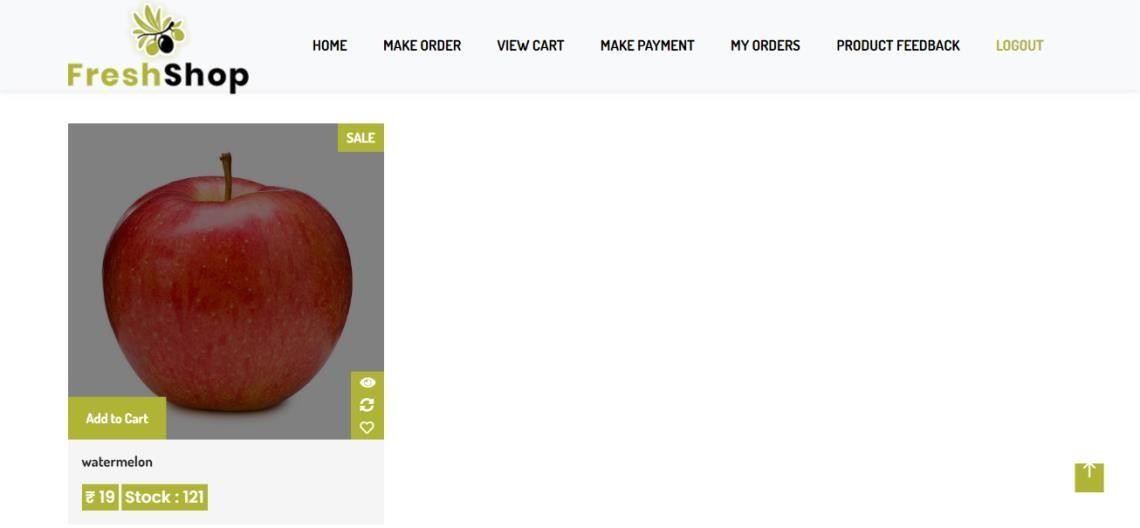
***Shopkeeper (Add Product)***



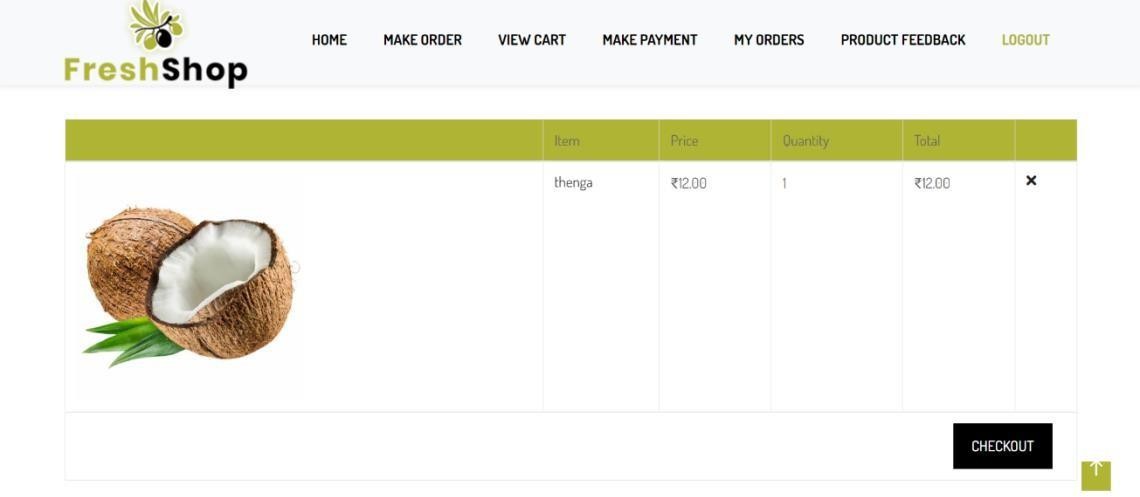
***Payment***



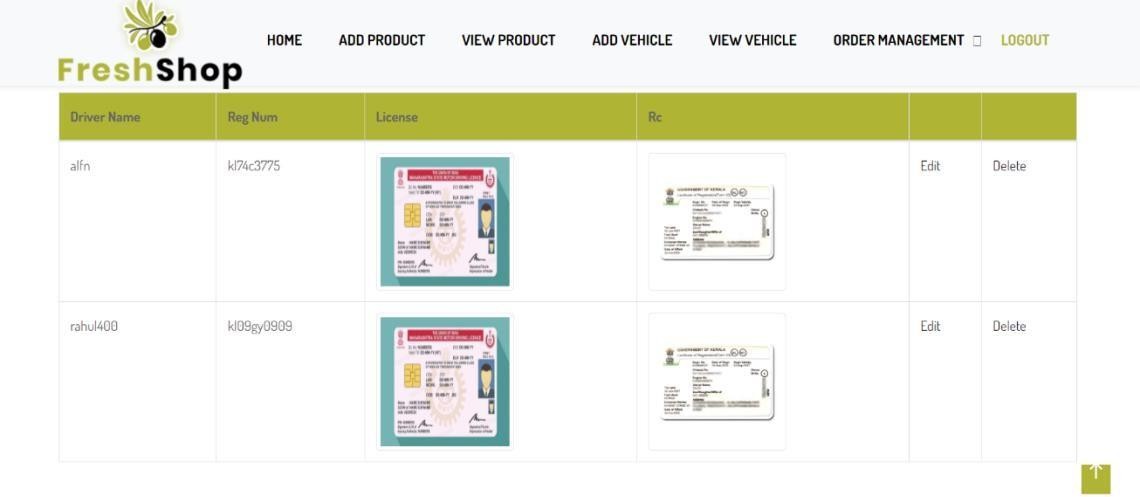
***Make Order Page***



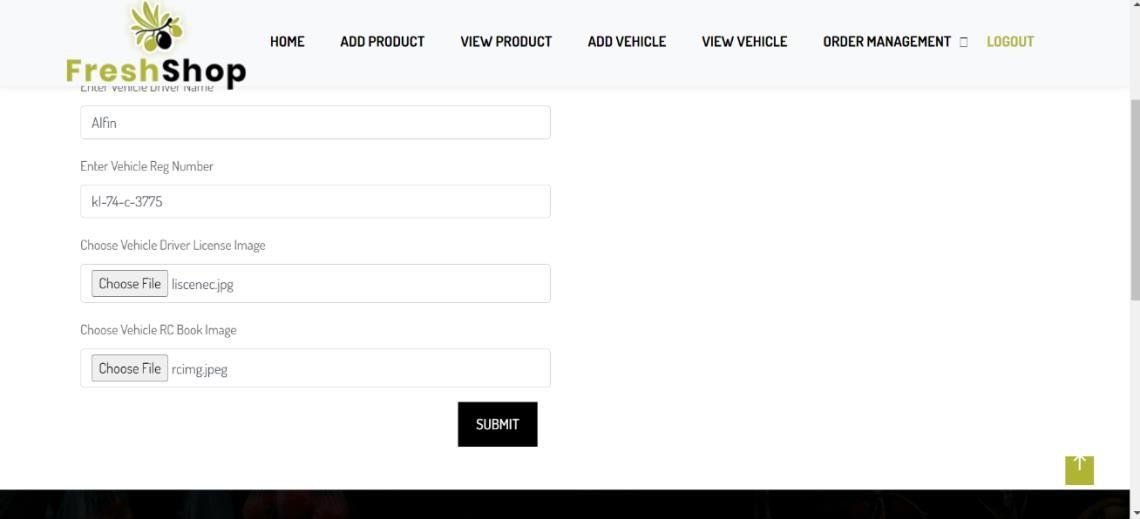
***Customer Add To Cart Page***



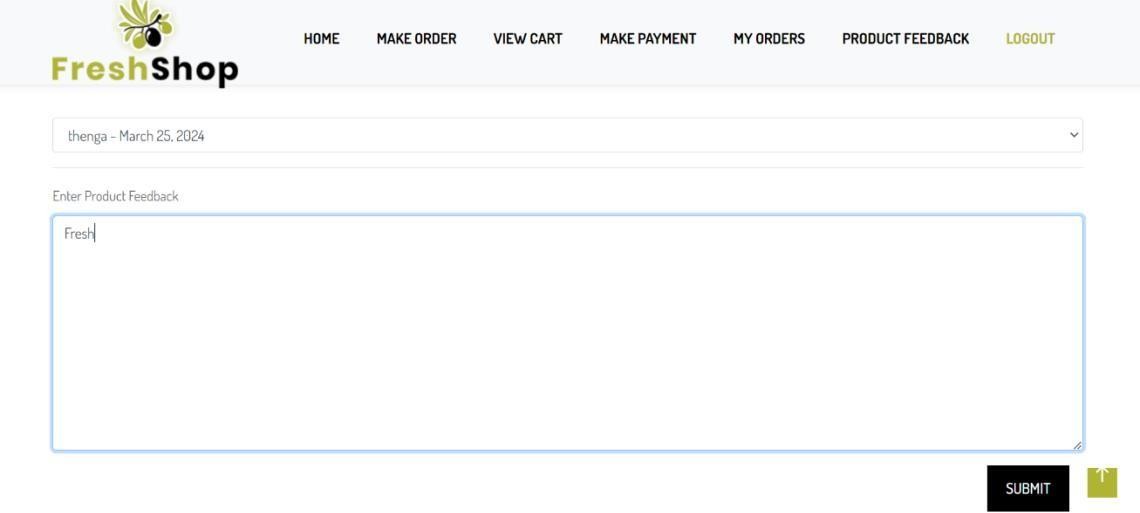
***Shopkeeper View Vehicle Page***



***Shopkeeper Add Vehicle***



***Customer Feedback Page***



## MENU TREE

**LOGIN**

**MERCH MINGLE**



Add vehicle

Add product

View product

View shipped orders

View feedback

SHOPKEEPER

View placed orders

View order

View vehicle

Product feedback

CUSTOMER

My orders

User make payment

Cart

Order

ADMIN

Verify shops

View shops

|  |  |
| --- | --- |
| **MEETING MINUTES**  **Date :** | **06/11/2023** |
| Place : | Christ Nagar College. |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna  ML. |
| Discussion : | Discussed about the topic. |
| **Date :** | **10/11/2023** |
| Place : | Christ Nagar College. |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |
| Discussion : | Topic Merch Mingle Package finalized. |
| **Date :** | **17/11/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |
| Discussion : | Conducted research on existing system and its drawbacks. |
| **Date :** | **24/11/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |
| Discussion : | Discussed about the various modules and functionalities to be included in the project. |
| **Date :** | **01/12/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

|  |  |
| --- | --- |
| Discussion : | Categorized the project into two modules. |
| **Date :** | **08/12/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Documentation started.

|  |  |
| --- | --- |
| **Date :** | **15/12/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |
| Discussion : | Started design of webpages. |
| **Date :** | **22/12/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |
| Discussion : | Documentation Discussion. |
| **Date :** | **29/12/2023** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Design of project was completed.

|  |  |
| --- | --- |
| **Date :** | **05/01/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Coding for Admin module started.

|  |  |
| --- | --- |
| **Date :** | **12/01/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Coding for User module started.

|  |  |
| --- | --- |
| **Date :** | **19/01/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Testing started for completed modules.

|  |  |
| --- | --- |
| **Date :** | **26/01/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Module wise functionalities completed

|  |  |
| --- | --- |
| **Date :** | **02/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Database created.

|  |  |
| --- | --- |
| **Date :** | **03/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Backend started.

|  |  |
| --- | --- |
| **Date :** | **09/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Documentation Discussion.

|  |  |
| --- | --- |
| **Date :** | **10/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Front End coding completed.

|  |  |
| --- | --- |
| **Date :** | **15/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Backend coding completed.

|  |  |
| --- | --- |
| **Date :** | **16/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Linking of backend and database started.

|  |  |
| --- | --- |
| **Date :** | **23/02/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Coding completed.

|  |  |
| --- | --- |
| **Date :** | **01/03/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Backend and database linking completed.

|  |  |
| --- | --- |
| **Date :** | **07/03/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Testing Completed.

|  |  |
| --- | --- |
| **Date :** | **14/03/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Verification and validation process done and completed project.

|  |  |
| --- | --- |
| **Date :** | **15/03/2024** |
| Place : | Quest Innovative Solution |
| Members : | Alex P Raj, Roy Mathew, Alfin Vincent, Rahul John, Nandhu Krishna ML. |

Discussion : Documentation completed.

# BIBLIOGRAPHY

***Book References***

1. Eric Ries Shari Lawrence Pfleeger, "*The Lean Startup* ".
2. Jennifer Niederst Robbins “*A Beginner’s Guide to CSS, JavaScript and Web Graphics, 2012, Shroff*.”
3. Nir Yayasan Djirdeh "*Hooked: How to Build Habit-Forming Products*".
4. David Karlins and Doug Sahlin, “*Building Websites All-in-One For Dummies*"
5. Dorothy Graham and Erik van Veenendaal, *“Foundation of Software Testing*”.
6. Grant Allen and Mike Owens,*"The Definitive Guide to SQLite"*.

***Web References***

* 1. https://[www.amazon.in,](http://www.amazon.in/)
  2. https://[www.shopify.com.](http://www.shopify.com/)
  3. https://[www.zoho.com.](http://www.zoho.com/)
  4. https:// [www.Ecwid.com.](http://www.ecwid.com/)
  5. https://[www.sqlite.org/docs.html.](http://www.sqlite.org/docs.html)
  6. https://[www.tutorialspoint.com/sqlite/index.html.](http://www.tutorialspoint.com/sqlite/index.html)
  7. https://realpython.com/.