

BIG-SHOPE E-COMMERCE SYSTEM

Project Report Submitted By

ARAVIND V V

Reg. No.: AJC20MCA-2024

In Partial fulfillment for the Award of the Degree Of

**MASTER OF COMPUTER APPLICATIONS
(MCA)**

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY



**AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY**

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

2020-2022

DEPARTMENT OF COMPUTER APPLICATIONS
AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY



CERTIFICATE

This is to certify that the Project report, “**BIG-SHOPE E-COMMERCE SYSTEM**” is the bonafide work of **ARAVIND V V** (Reg.No:AJC20MCA-2024) in partial fulfillment of the requirements for the award of the Degree of Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Mr. T J Jobin
Internal Guide

Ms Nimmy Francis
Coordinator

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

External Examiner

DECLARATION

I hereby declare that the project report “**BIG_SHOPE E-COMMERCE SYSTEM**” is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date: 12/07/2022

KANJIRAPPALLY

ARAVIND V V

Reg. No: AJC20MCA-2024

ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillikutty Jacob** for providing good faculty for guidance.

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

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

ARAVIND V V

ABSTRACT

Shopping online and home delivery has become a stable in the current market, its easier for people to order products online and convenient to get the product delivered at their home. Electronic products, sports items and even food items are currently shopped online more than shopped manually. The ecommerce software we provide act as a storage and website thus, it provides the customers a platform to buy a particular products online, to deliver it to the particular address and pay online using credit, debit and other cards. Through this software our effort is to expand the current system followed by shopping stores into a website and storage model , So that the system becomes more efficient, faster, and secure and error free in terms of data. This will satisfy almost all the system requirements specification.

Big-Shope acts as an online store, shopping and stock management system that permits a customer to register and login. Customers can then place orders for items. Items will be added to a virtual cart which will then be purchased. Money is paid online. The ordered items will be delivered to customers address. Based on prior purchases, the customer will receive recommendations. Customers may rate and review items to express their thoughts on them.

The online store is likewise covered by the system. The system includes complete information on items, stocks, vendors, orders, sales, etc. for the owner or administrator to access. The system keeps track of the current stock. A proper warning and reorder requests are generated in the case of stock depletion. Products are purchased from vendors who are registered by admin. Vendors restock products according to reorder requests and latest prices. Sales reports are also generated. These can be used to look at a day's/week's/month's or year's worth of sales and earnings. Sales reports are also used to identify high selling products. The customer purchase, stock, sales and purchase from vendors work together as a chain in the BigShope system and once customers order products, stock check, updation, sales report generation and reorder of items from vendors are all automatically done based on criteria's .

CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	5
2.1	INTRODUCTION	6
2.2	EXISTING SYSTEM	7
2.3	DRAWBACKS OF EXISTING SYSTEM	7
2.4	PROPOSED SYSTEM	7
2.5	ADVANTAGES OF PROPOSED SYSTEM	10
3	REQUIREMENT ANALYSIS	11
3.1	FEASIBILITY STUDY	12
3.1.1	ECONOMICAL FEASIBILITY	12
3.1.2	TECHNICAL FEASIBILITY	13
3.1.3	BEHAVIORAL FEASIBILITY	13
3.2	SYSTEM SPECIFICATION	14
3.2.1	HARDWARE SPECIFICATION	14
3.2.2	SOFTWARE SPECIFICATION	14
3.3	SOFTWARE DESCRIPTION	14
3.3.1	DJANGO	14
3.3.2	GOOGLE FIRESTORE	15
4	SYSTEM DESIGN	17
4.1	INTRODUCTION	18
4.2	UML DIAGRAM	18
4.2.1	USE CASE DIAGRAM	20
4.2.2	SEQUENCE DIAGRAM	22
4.2.3	STATE CHART DIAGRAM	24
4.2.4	ACTIVITY DIAGRAM	25
4.2.5	CLASS DIAGRAM	26
4.2.6	OBJECT DIAGRAM	27
4.2.7	COMPONENT DIAGRAM	28
4.2.8	DEPLOYMENT DIAGRAM	29

4.3	USER INTERFACE DESIGN	30
4.4	DATA BASE DESIGN	33
5	SYSTEM TESTING	41
5.1	INTRODUCTION	42
5.2	TEST PLAN	43
5.2.1	UNIT TESTING	44
5.2.2	INTEGRATION TESTING	52
5.2.3	VALIDATION TESTING	52
5.2.4	USER ACCEPTANCE TASTING	53
6	IMPLEMENTATION	54
6.1	INTRODUCTION	55
6.2	IMPLEMENTATION PROCEDURE	55
6.2.1	USER TRAINING	56
6.2.2	TRAINING ON APPLICATION SOFTWARE	56
6.2.3	SYSTEM MAINTENANCE	56
6.24	HOSTING	57
7	CONCLUSION & FUTURE SCOPE	58
7.1	CONCLUSION	59
7.2	FUTURE SCOPE	59
8	BIBLIOGRAPHY	60
9	APPENDIX	62
9.1	SAMPLE CODE	63
9.2	SCREEN SHOTS	68
9.3	PLAGIARISM REPORT	71

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

‘Big Shope E-Commerce’ acts as an online store, shopping and stock management system that permits a customer to register and login. Customers can then place orders for products available in the site. Product will be added to a virtual cart which then can be purchased by customer. Money is paid online. The ordered items will be packaged and delivered to location customer provides. The online store is likewise covered by the system. The system includes complete information on items, stocks, vendors, orders, sales, etc. for the owner or administrator to access. Admin registers vendor into the system. Vendors add/update products with the latest prices.

Available products are displayed in the website along with appropriate categorization, so that customers can find and order the products easily. The orders, stock, sales and purchase from vendors work together as a chain in the BigShope system and once customers order products, stock check, updation, sales report generation etc, are all automatically done based on given criteria's . The project is done using django framework and google firestore as database.

1.2 PROJECT SPECIFICATION

A superstore's e-commerce website is the proposed system. The overall system is divided into in five modules.

Modules:

Customer

- Customers refer to the registered users who can login to the online shopping portal using email and password.
- Each customer has a profile of their own on the site with information such as name, location, contact details, etc. They can also edit these information in their profile, change password or email as their need.
- Customers can go through the entire product list. They can also find products through search option, categories and suggestions. Customers can also compare and sort out products based on price range, ratings etc.
- Customers add products to the virtual cart and place an order. Payment for the purchase is done through online payment portal. The product will be delivered to customers

location.

- Customers can cancel the order if necessary.
- Products can be added to wish lists for later purchase by customers.
- Customers will get product recommendations based on their purchase behavior and ratings they give to products.
- Admin can access details of registered customers along with the orders they placed and sales report. Admin can disable customers.

Order

- Customers can add any number of items into the cart.
- Successive items selected for purchase are placed into the cart until a customer completes their shopping. Products of cart can be edited and deleted.
- The total amount is calculated. Items added will remain in the cart for a specific time and will be removed if it's not purchased.
- Products in the cart can then be purchased. Both location of delivery and online payment follows. Customers can choose particular locations for delivery. After making their purchase, the consumer can print the order information to get a physical bill.
- The order details include unique order id, payment id customer details, product details, price, date, location etc. Status of purchased order is given in the profile of customers (status: cancelled, not delivered, delivered, etc).
- Customers can check order history to view details of previous orders. Current stock is updated according to purchases. Admin can monitor each order.
- Customers can pay purchase amount through the payment portal razorpay.

Stock

- Online shopping require large number of products that belong to different categories. These products make up the stock. Stock contain details of all the products in the store. Details include product name, categories and subcategories, quantity, cost price and selling price, purchase and expiry dates, vendor details etc.
- Stocks are stored in warehouses on different locations. Available stock is displayed for customers to order. Customers purchase from available stocks and they are reduced accordingly. Once the stocks are reduced below a certain threshold, they need to be

restocked.

- Each product is differentiated and identified by a category, sub category, brand, etc. Users may find what they're looking for more quickly based on categories and brands. Products can also be filtered based on price.

Vendors

- For stock procurement, vendors are contacted. The system maintains track of the information of a large number of vendors. Admin/Owners register vendors into the system.
- Vendors add products into the system along with its features and prices.
- A stock reorder report is generated if any product runs out of stock, and this report is used to reorder new stock. Admin requests corresponding vendors for new stocks based on this reorder report. Admin can make modifications to the generated reorder report.
- Vendors receive the request and, after it is fulfilled, respond to the system. The vendor restock the products according to the request with available quantity and latest prices.
- Admin can disable vendors or vendor's products. Admin can directly request for new stocks for any product.

Sales

- After each purchase, the information is recorded in a database. The information includes purchase-related details such as the current status, date, amount, customer details etc. Sales reports can be generated using this data.
- The sales reports contain information about the sales. Sales for a specific day, month etc, can be viewed by the owner/admin. To determine the level of profit or loss the shop gets, sales records are essential.
- Sales data can be used to determine which items/products sell quicker and more frequently. This information is utilized to give such goods additional attention and ensure that they are always in stock. When a stocks are depleted, a reorder report is generated. The admin checks and approves the reorder of these items to replenish stock. The reorder request reach corresponding vendors and the vendors restock the products.

CHAPTER 2

SYSTEM STUDY

2.1 INTRODUCTION

System analysis is the procedure of getting and analyzing data, identifying problems, and using this data to propose changes in the system. The users and system developers must communicate well and thoroughly during this process. Any system development process should start with a system study and analysis. The system is entirely examined. The system analyst acts in the role of an interrogator and delves deeply into how the current system functions. The input to the system is identified, and the system as a whole is examined. Different procedures of the system may be directly linked to the output of the organizations. Being aware of the problem, identifying the important factors, evaluating and synthesizing the many elements, and selecting the adequate course of action are all part of system analysis.

The process must be thoroughly studied using different methods, like surveys and interviews. To reach a conclusion, the information gathered by various sources must be thoroughly studied. Understanding how the system works is the conclusion. The current system is what it's called. Now, the current system is examined, and problem areas are found. The designer now acts as a problem-solver and works to solve the issues. Proposals are made as the solutions to problems. The proposal is then compared against the current system, and the best one is chosen. The proposal is examined in response to user suggestions or decisions, and necessary revisions are made. As soon as the user is content with the proposal, this loop breaks.

Preliminary research is the process of gathering and evaluating data in order to use it for upcoming system investigations. Initial research requires strong collaboration between system users and developers since it involves problem-solving. It carries out several feasibility studies. These studies offer a rough idea of the system activities, which may be utilised to choose the methods to employ for effective system research and analysis..

2.2 EXISTING SYSTEM

The main function of existing system is storing details of all the products owned by the supermarket and keeping the report of current stocks and sales. This system is used to manually store and retrieve information. The system is purely storage based. It stores:

Stock details: Stores the information about all their available products including the product name, brand name, category, price, quantity and purchase date.

Purchase Details : Details about purchase of products from the vendors.

Sales Details: Includes the info about all the sales.

2.3 DISADVANTAGES OF PRESENT SYSTEM

- 1) Only owner uses the system to store, retrieve and manage information's.
- 2) The system doesn't act as a website.
- 3) A customer doesn't have any part in the current system.
- 4) Customer need to get to the store manually to check available items and purchase.
- 5) Not flexible enough to include home delivery. Payment methods are basic.
- 6) The user interface can be improved. It's a little complicated for the admin to learn the system.
- 7) The current system doesn't provide any facility for marketing the company.

2.4 PROPOSED SYSTEM

The proposed system is intended to address drawbacks of the current system. It is essential to have a system that is more user-friendly and appealing to users. The system that is offered takes this into account. Customers, admin and vendors are the users of this system. Users don't need any special training to utilize the system and it overall reduces the number of manual hours needed to complete routine tasks, thus saving time and improving performance. The database is scalable according to usage.

The system is relatively easy to install and develop and works in practically all settings and uses very little resources. It eliminates manual errors and ensure correct data. Data won't be lost as it is stored in the cloud with adequate backups.

Our proposed system primarily acts as a website, it stores data, which can be modified if required and users can access it online to check needed information. The system has a database to store details of stock, sales, customers etc. The information can be retrieved easily to do appropriate editing and modification by the authorized users. Every product is displayed in the listing page along with its details which can be viewed by the customer and from there the customers can purchase items and pay money online from the website. The purchased item will be delivered to the location provided by the customer.

Customer Features

Through the registration page, any user may sign up. Customers who have successfully registered may log in using an email and password. Following a successful login, a client may accomplish the following:

1. Visit website anytime.
2. Orders items.
3. Pay online.
4. View all available items.
5. Adding items to the cart for multiple purchase in the same time.
6. Query needed item.
7. View/Update profile.
8. Orders and Order history.
9. Add products to wishlist.
10. Login/Logout.

Admin Features

1. Check current stock details and availability.
2. Manage currents items/products(Edit,Disable).
3. Check all orders and order history.

4. Check current sales done through website. Details such as no of products sold, total sale amount, total profit sale of each product etc can be analyzed.
5. Manage users.
6. Admin adds vendor into BigShope system.
7. Admin can confirm/cancel reorder request generated upon stock depletion, can also send reorder requests for products himself.
8. Login/Logout.

Vendors

1. Vendors are registered by admin into BigShope.
2. Vendors add products to the system, along with their corresponding categories, brands and product features. Vendor can manage their own products.
3. Vendor modifies the cost price and selling prices of each products according to price changes.
4. Vendors receives reorder request of his products. Vendor can modify prices or quantity of requested product and confirm reorders. If confirmed, new stock of specified quantity is updated for the requested products into the system along with modified prices.
5. Vendor can view detailed reorder reports.

2.5 ADVANTAGE OF PROPOSED SYSTEM

1. The main advantage is that the proposed system can act as a website.
2. Users can check products availability and make purchases online anytime. Users can manage profile and keep track of purchases.
3. It provides better advertisement for the company and its offering.
4. Admin can easily manage orders and check sales/previous reports.
5. Admin can keep track of customers and manage products and stock.
6. Admin register vendors into the system.
7. Vendors can add products and manage product prices in the system itself.
8. Vendors can also restock the products according to requests with one click, stock details and purchase report are generated.
9. Chances of getting errors are very low.
10. Easy to use and reduce manual efforts.
11. Faster data access is possible.
12. Security, which prevents unauthorized access.

-
13. Online purchase is easier and push digital world.
 14. Home delivery is available, customers doesn't need to visit the store.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical and Economic feasibilities. The features are: -

3.1.1 Economical Feasibility

Analyses of costs and benefits are necessary to support the developing system to get best results. Cost associated with establishing a new system is an important factor.

The following are some of the important financial questions asked during preliminary investigation:

- The expenses carry out a comprehensive system analysis.
- The price of the hardware and software.
- The benefits in the form of reduced costs.

The proposed system was created as part of a project, thus there are no manual expenses associated with it. The system may also be put into place at a reasonable cost given the availability of the required resources..

The project's costs were broken down into three categories: system costs, development costs, and hosting costs. All estimates indicate that the project was created at a modest cost.

3.1.2 Technical Feasibility

First, technical feasibility study is necessary. The assessment of it's viability is done on an overview design of the system. After identifying this outline system, the investigation must next recommend the type of equipment, essential construction methods, and ways to operate the system once it has been established.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be created in such a way that the required performance and functionality are met within the limitations. The project uses cryptographic methods and calls for a high resolution scanning equipment. Since a newer version of the same software still functions with an older version, the system may still be used even if the technology becomes outmoded over time. Consequently, there aren't many restrictions on this project. The system was created with the Django framework and Google Firebase, and it should be possible to finish the project. The computer used has a potent Intel i3 core CPU, 4GB of RAM, and a 1TB hard drive.

3.1.3 Behavioral Feasibility

Because it would accomplish the objectives after being developed and put into action, the project would be considered advantageous. After thoroughly examining all behavioral parameters, such as how easy it is to use the system or will it cause any harm, it is determined that the project is behaviorally feasible.

3.2 SYSTEM SPECIFICATIONS

3.2.1 Hardware Specifications

Processor	- Intel core i3
RAM	- 4 GB
Hard disk	- 1 TB

3.2.2 Software Specifications

Front End	- HTML, CSS
Backend	- Django, Google Firetsore
Client on PC	- Windows 7 and above.
Technologies used	- JavaScript, HTML, J Query, Django Framework, CSS

3.3 SOFTWARE DESCRIPTION

3.3.1 Django Framework

A Python-based web framework called Django enables you to easily build effective online apps. Because Django has built-in functionality for everything, including the Django Admin Interface and the default database, SQLite3, it is often known as a batteries included framework. Establishing a website often requires a similar set of components: A way to manage user authentication (signing up, logging in, and logging out), a control panel for your website, forms, and a way to post files are all included. Django provides you with pre-made components that may be used for quick development.

Why Django Framework ?

- Excellent documentation and excellent scalability.
- Top MNCs and Companies, like Instagram, Disqus, Spotify, Youtube, Bitbucket, Dropbox, etc use Django.
- The simplest Framework to understand, quick development, and completely integrated Batteries

- Python, which has a large library and functions like web scraping, machine learning, image processing, scientific computing, etc., is the final but not least incentive to learn Django. All of this may be integrated with a web application to do several advanced tasks.

3.3.2 Google Firestore

- Firestore is a NoSQL document database built for automatic scaling, high performance, and ease of application development. While the Firestore interface has many of the same features as traditional databases, as a NoSQL database it differs from them in the way it describes relationships between data objects.
- Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, you store data in *documents*, which are organized into collections.
- Serverless document database that effortlessly scales to meet any demand, with no maintenance.
- Accelerate development of mobile, web, and IoT apps with direct connectivity to the database.
- Built-in live synchronization and offline mode makes it easy to develop real-time applications.
- Fully customizable security and data validation rules to ensure the data is always protected.
- Seamless integration with Firebase and Google Cloud services like Cloud Functions and BigQuery.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step in the creation of any technical system or product. Design is a creative process. A good design is the key to a system that works effectively. "Design" is the process of employing many approaches and concepts to thoroughly outline a process or a system so that it may be physically implemented. The process of employing several approaches and concepts to specify a tool, a procedure, or a system in sufficient detail to enable its physical actuality is one way to put it. Software design serves as the technical foundation of the software engineering process, regardless of the development paradigm used. The system design generates the architectural detail required to build a system or product. As with any systematic approach, this software underwent the best design phase possible, fine-tuning all efficiency, performance, and accuracy levels. A user-oriented document is converted into a document for programmers or database staff throughout the design process. There are two stages to the creation of a system design: the physical and logical design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for Unified Modeling Language. UML is different from other well-known programming languages like C++, Java, COBOL, etc. Software designs are created using a visual language referred to as UML. UML is a general-purpose visual modelling language used for the definition, design, development, and documentation of software systems. Although representing software systems is the most popular use of UML, it is not the only one. Simulating non-software systems is another use for it. the process flow in the manufacturing plant, etc. Despite not being a programming language, UML may be used with tools to generate code in a number of other languages. UML is intimately connected to the analysis and design of objects-oriented systems. It is completed by using all the other components. The following nine diagrams are included in UML.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Activity diagram
- Statechart diagram
- Deployment diagram
- Component diagram

4.2.1 USE CASE DIAGRAM

A use case diagram is a visual representation of the interactions between system components. A approach for identifying, outlining, and organizing system requirements is called a use case. The word "system" here refers to a thing that is being created or run, like a website for mail-order goods sales and services. UML (Unified Modeling Language), a standard language for the modelling of real-world objects and systems, uses use case diagrams.

Planning general requirements, validating hardware designs, testing and debugging software products while they are still in development, creating online help resources, and finishing customer support-focused tasks are a few examples of system objectives. For instance, customer support, item ordering, catalogue updating, and payment processing are examples of use cases in a setting of product sales. A use case diagram has four components.

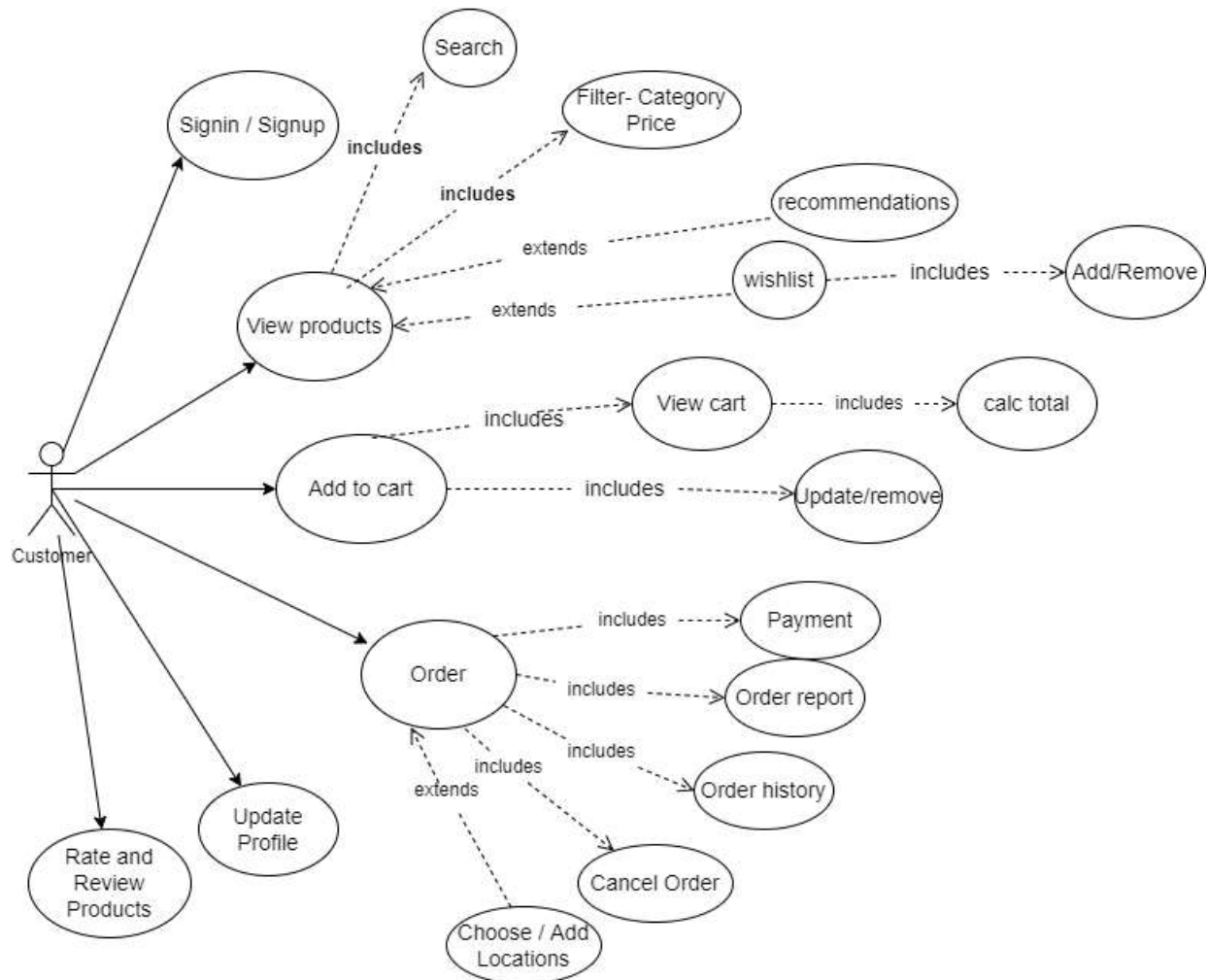
- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are created to depict a system's functional needs. To create an effective use case diagram after identifying the aforementioned things, we must adhere to the following rules.

- A use case's naming is highly significant. The name should be chosen in a way that makes it clear what functions are being carried out.
- Give the actors names that fit them.
- Clearly depict links and dependencies in the diagram.
- Keep in mind that the diagram's primary function is to indicate the needs, do not attempt to include all possible links.
- When necessary, take notes to help you remember some crucial details.

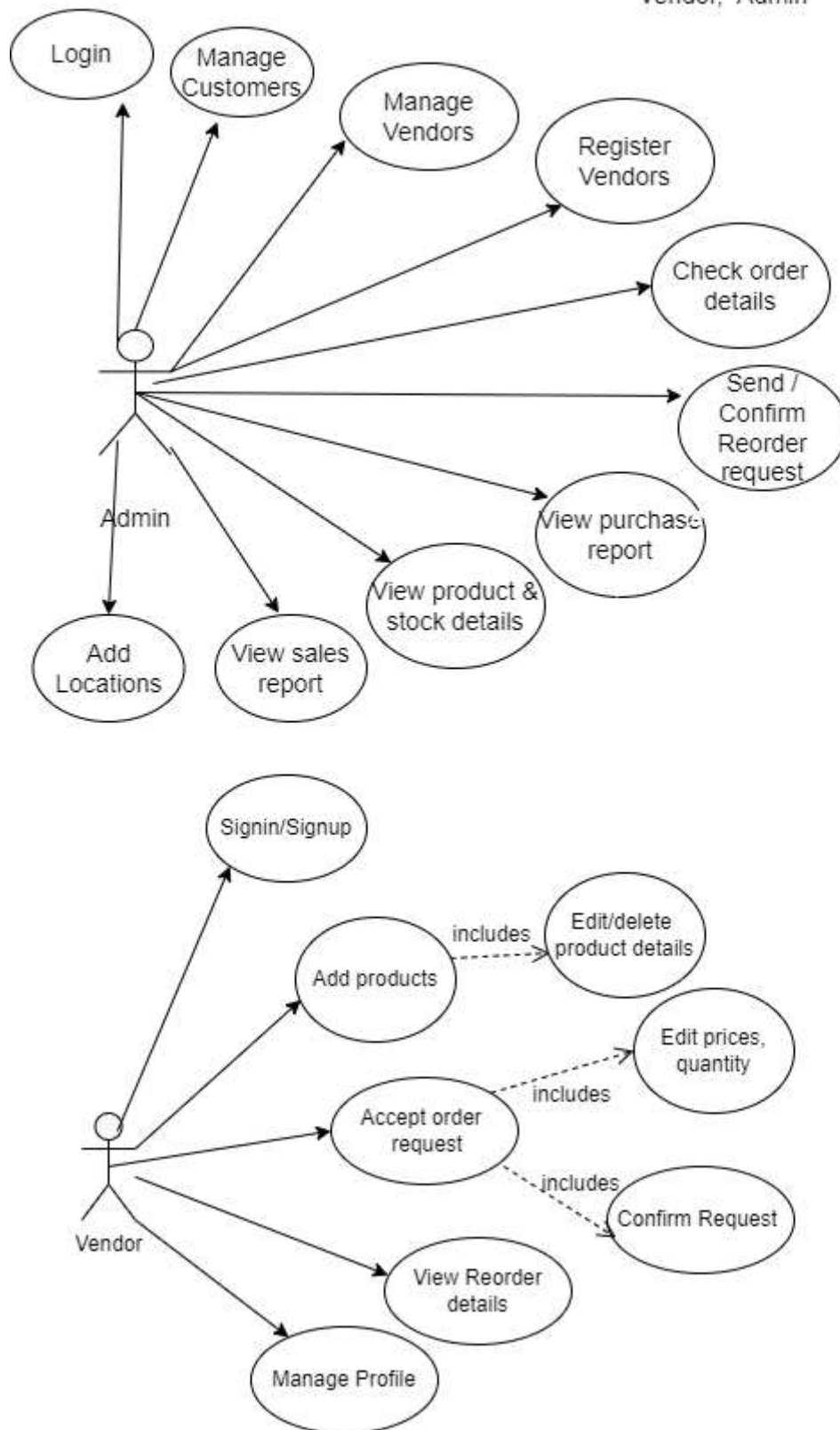
UseCase Diagram

Customer



UseCase Diagram

Vendor, Admin



4.2.2 SEQUENCE DIAGRAM

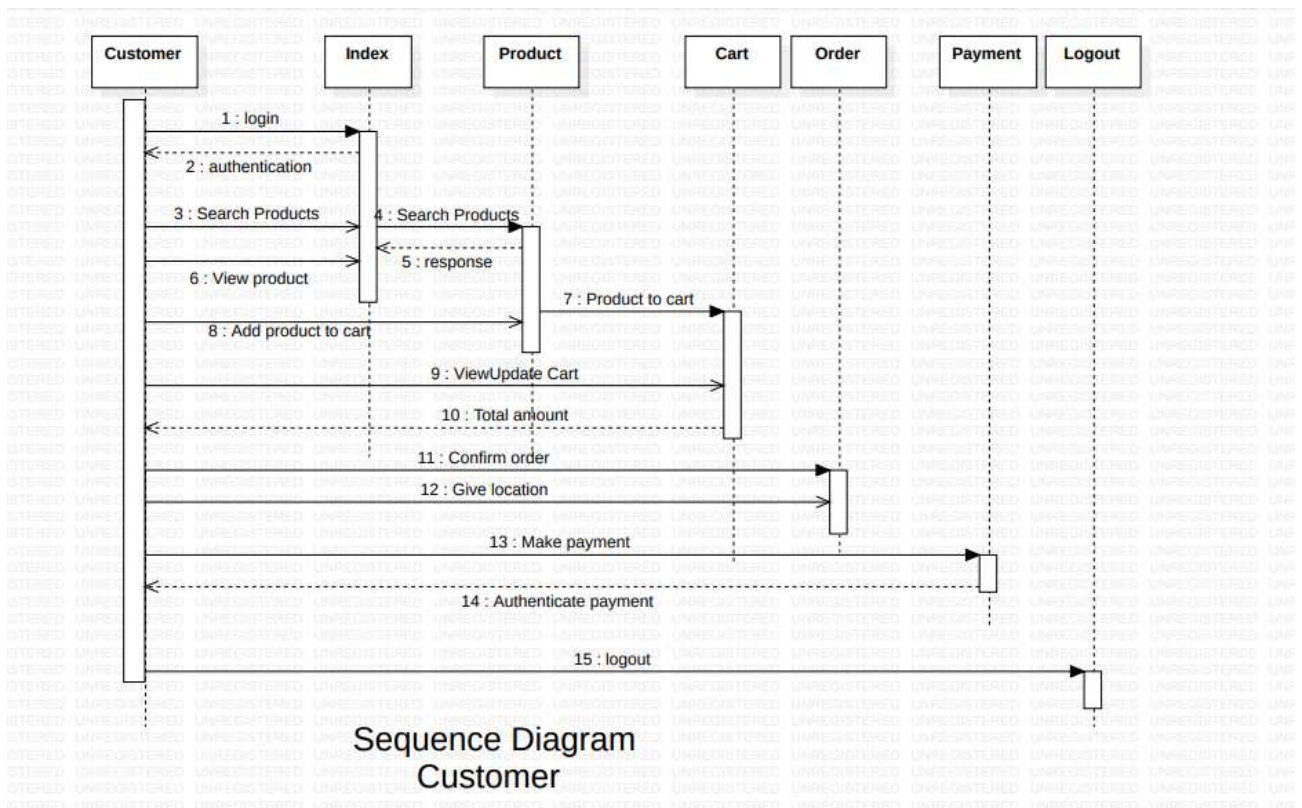
A sequence diagram fundamentally depicts the sequential order in which events occur or how they interact with one another. Event diagrams and event scenarios are other names for sequence diagrams. Sequence diagrams display the activities performed by a system's parts in time order. These diagrams are widely used by businesspeople and software engineers to document and explain the requirements for new and existing systems.

Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram symbolises a certain sort of role in which it interacts with the system's elements. An actor is never inside the scope of the system that we want to describe using the UML diagram. For a range of roles, including those of human users and other external topics, we use actors. An actor is shown using the stick person notation in a UML diagram. A sequence diagram could have several actors.
- ii. **Lifelines** – A named piece that shows a specific participant in a sequence diagram is called a lifeline. In essence, a lifeline represents each incident in a sequence diagram. The lifeline components in a sequence diagram are at the top.
- iii. **Messages** – It is shown how things may communicate with one another through messages. The lifeline displays the messages in reverse chronological order. Messages are represented by arrows. The two major elements of a sequence diagram are lifelines and messages.
- iv. **Guards** – We use guards in the UML to model scenarios. We employ them when we need to restrict communication under the pretext that a criterion has been met. Guards are relied upon by software developers to notify them of the limitations imposed by a system or particular technique..

Uses of sequence diagrams –

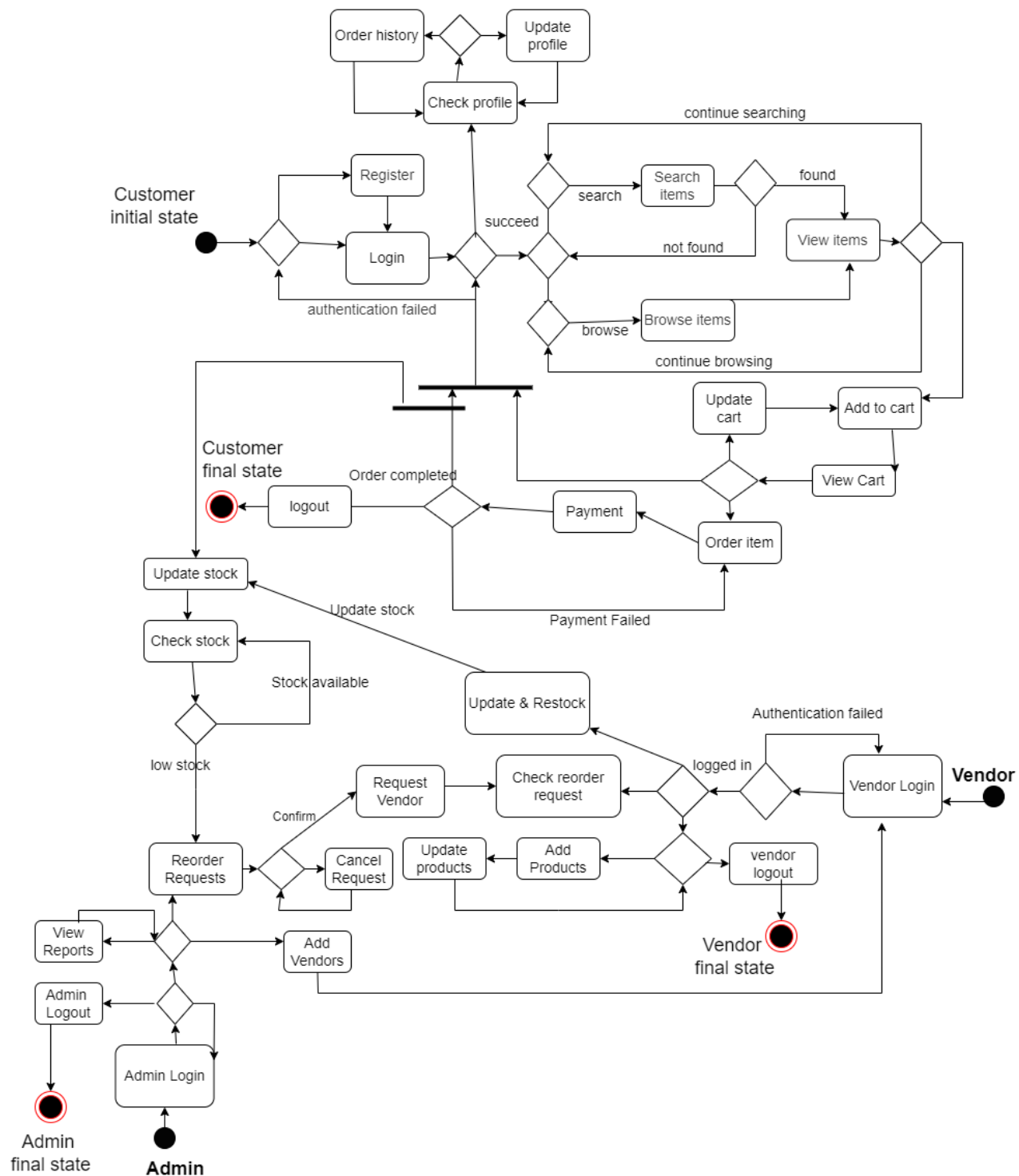
- They are also used to display specifics of UML use case diagrams, as well as to describe and depict the logic underlying complex functions, operations, or procedures.
- Used to comprehend the precise operation of the existing system
- Recognize the flow of tasks and communications among the various system's elements.



4.2.4 ACTIVITY DIAGRAM

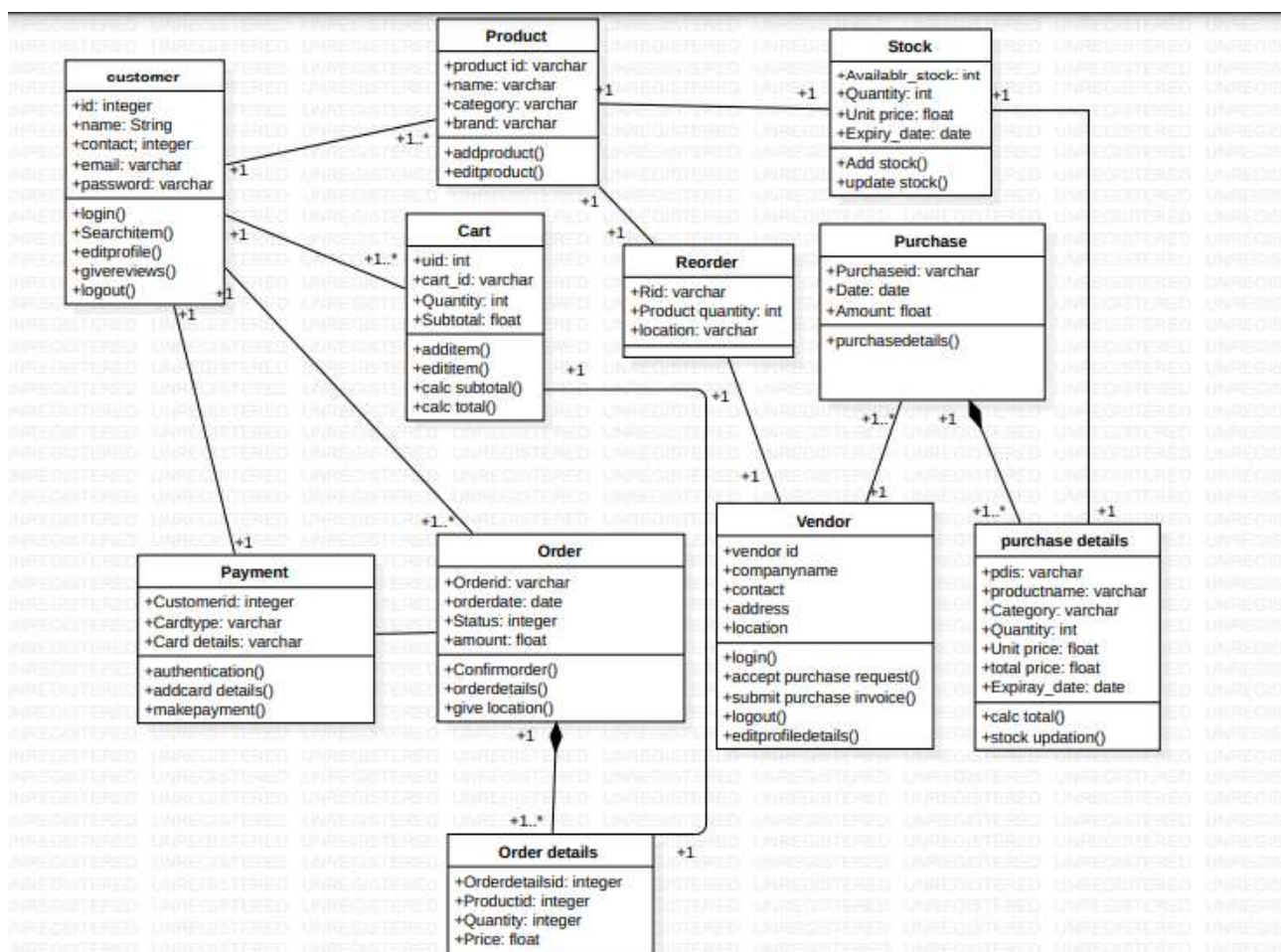
Activity diagrams show how multiple levels of abstraction of activities are coordinated to produce a service. It may also be used to illustrate how a set of related use cases interact together to reflect business operations.

Activity Diagram



4.2.5 CLASS DIAGRAM

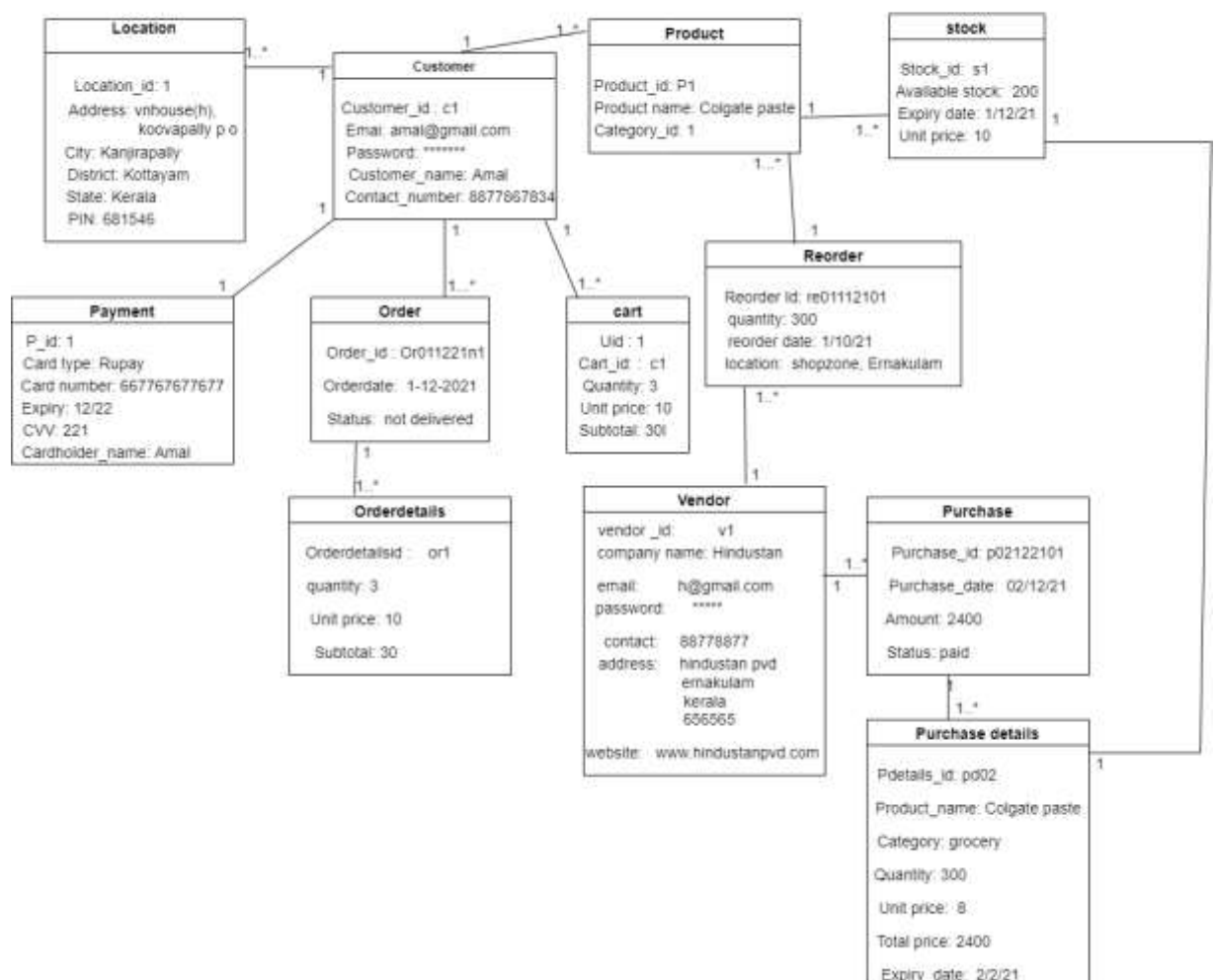
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.



4.2.6 OBJECT DIAGRAM

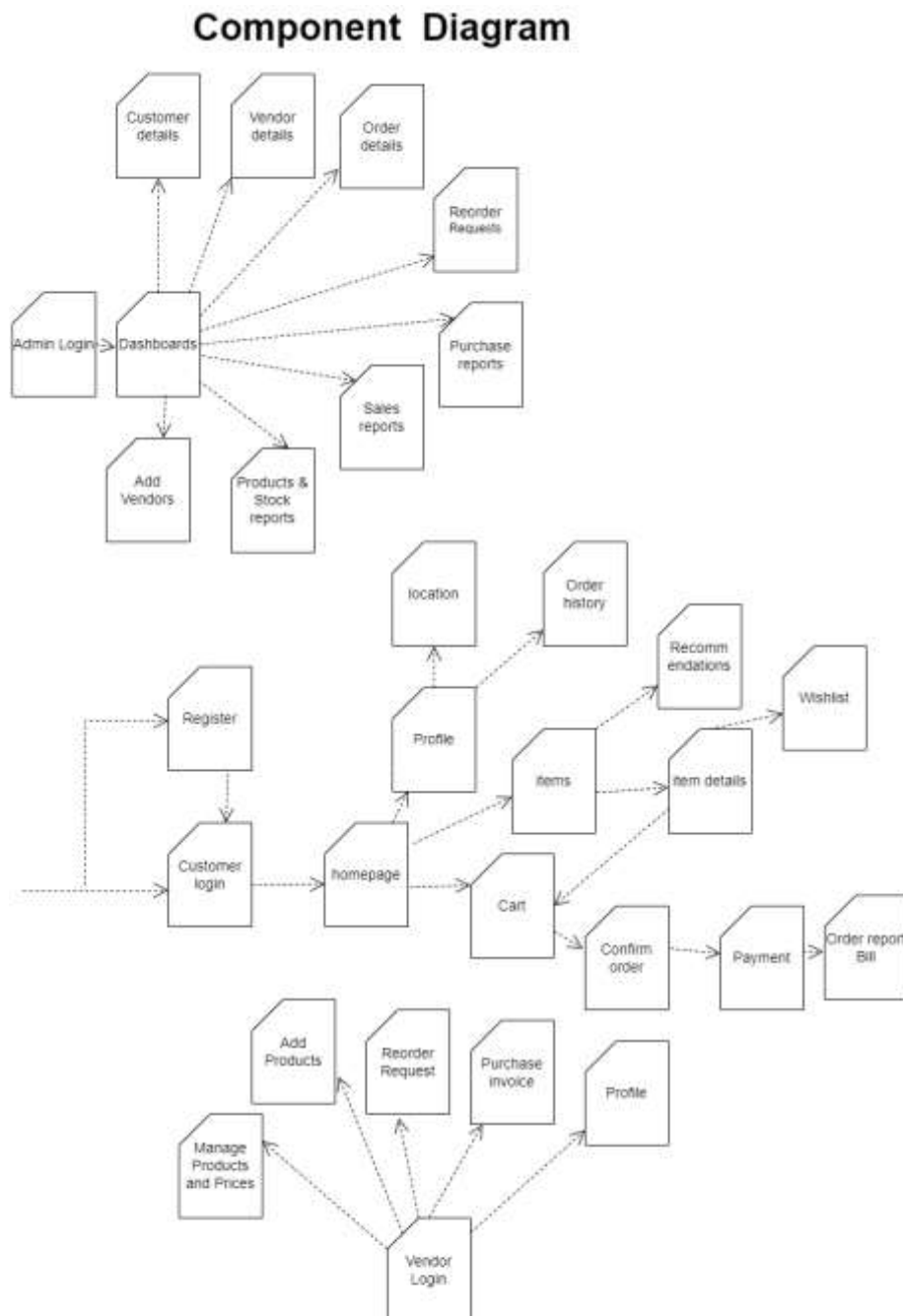
Object diagrams are derived from class diagrams so object diagrams are dependent upon class diagrams. Object diagrams represent an instance of a class diagram. The basic concepts are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system but this static view is a snapshot of the system at a particular moment. Object diagrams are used to render a set of objects and their relationships as an instance.

Object Diagram



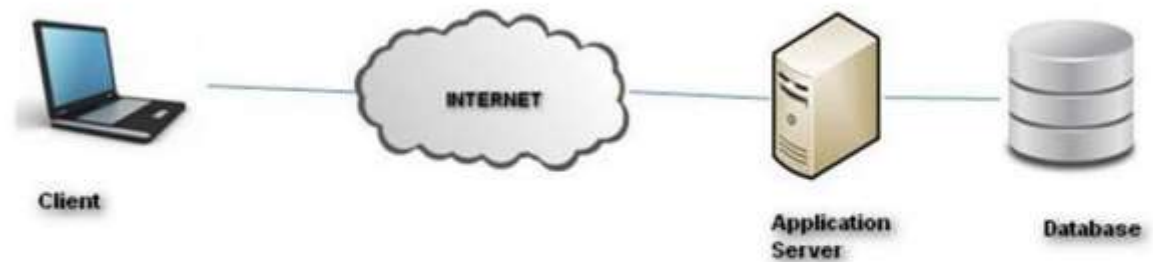
4.2.7 COMPONENT DIAGRAM

Component diagrams have different behaviors and personalities. The physical parts of the system are represented using component diagrams. Executables, libraries, files, documents, and other items that are physically present in a node are just a few examples. Component diagrams are used to show how a system's components are connected and arranged.



4.2.8 DEPLOYMENT DIAGRAM

Deployment diagrams are used to depict the topology of a system's physical components, as well as the locations of software components. Deployment diagrams are used to describe a system's static deployment view. Nodes and their relationships are shown in deployment diagrams.



4.3 USER INTERFACE DESIGN

4.3.1-INPUT DESIGN

Form Name : User Registration

Registration

Name

Email

Contact Number

Password

Confirm Password

Form Name : User Login

Login

Email

Password

4.3.2 OUTPUT DESIGN

User Login

LOGIN IN

EMAIL

PASSWORD

[New User?](#)

SIGN IN

User Registration

User Registration

NAME

EMAIL

PHNO

PASSWORD

CONFIRM PASSWORD

REGISTER[Login](#)

4.4 DATABASE DESIGN

A database is a structured system with the capacity to store information and allows users to access stored information quickly and effectively. Any database's primary goal is its data, which need protection.

There are two stages to the database design process. The user needs are obtained in the first phase, and a database is created to as clearly as possible satisfy these criteria. This process, known as information level design, is carried out independently of all DBMSs.

The design for the specific DBMS that will be used to construct the system in issue is converted from an information level design to a design in the second stage. Physical Level Design is the stage where the characteristics of the particular DBMS that will be utilized are discussed. Parallel to the system design is a database design. The database's data arrangement aims to accomplish the following two main goals.

- Data Integrity
- Data independence

4.4.1 NoSQL database

In contrast to relational databases, NoSQL databases store data in documents. Pure document databases, key-value stores, wide-column databases, and graph databases are some examples of NoSQL database types. NoSQL databases are created from the bottom up to accommodate an increasing number of modern organisations and store and analyse massive volumes of data at scale.

Instead of the columns and rows that relational databases employ to store data, NoSQL database technology stores data as JSON documents. To be precise, NoSQL does not mean "no SQL," but rather "not only SQL." This indicates that a NoSQL JSON database may essentially "use no SQL" to store and retrieve data. Or, for the best of both worlds, you may combine the adaptability of JSON with the strength of SQL. Because of this, NoSQL databases are designed to be adaptable, scalable, and quick to meet the data management needs of contemporary enterprises. Here, Google Firestore, a NoSQL document database, is utilised.

I

We are able to store data in documents that include field mappings for values thanks to Cloud Firestore's NoSQL data architecture. Collections is a container used to hold the documents. Our data is organised using these containers, and queries are made using them. Documents may handle a variety of data kinds, from straightforward strings and integers to intricate nested structures. Additionally, we may build sub-collections inside of documents and design a hierarchical data structure that expands with our database. Whatever data structure works best for our programme is supported by the Firestore data architecture. The query in Cloud Firestore is also eloquent, effective, and versatile.

DATABASE DESIGN**CUSTOMER -- Collection Name**

Fields	Data Type	Description
Email_Id	string	Customers email address
password	string	Password
name	string	Customer name
Contact_No	number	Customers contact number
Register_Date	timestamp	Registration date of customer
Active	boolean	Currently active or not

LOCATION

Fields	Datatype	Description
Customer	path	Customer document
Address	string	Address for delivery
pin	number	Pin number
city	string	Customers city
district	string	Customers district
state	string	Customers state
active	boolean	Currently active or not

STOCK

Stocknumber	string	Stock number
Product	path	which product
A_stock	integer	available stock
Expiry	date	Expiry date of this stock of product
Cost price	number	cost price of this stock of product
Selling Price	number	selling price of this stock of product
active	boolean	Active or not

PRODUCT

Fields	Data Type	Description
Category	path	To identify category
Brand	path	To identify brand
Product _Name	string	Name of product
Description	string	Describes the product
Product details	string	Details about the product
image1	string	Product image
active	boolean	Active or not
vendor	path	Info of vendor who added product

CATEGORY

Fields	Data Type	Description
Category_name	string	Name of category
description	string	describes the category
Image	string	image
active	boolean	Active or not

CART

Field	Datatype	Description
customer	path	Path to customer
product	path	Path to product
quantity	number	Added quantity
unit_price	number	Price of single item
total_price	number	Quantity * unit price

ORDER

Order_id	string	Id of order
Payment_id	string	To get payment info
Customer	path	Which Customer
Location	path	Customers location
Amount	number	Total amount of order
Order_date	timestamp	Date of order
Delivery_date	timestamp	Date of delivery
status	string	Current status of order
active	boolean	Active or not

ORDER DETAILS

Order	path	To identify which order
Product	path	Id of stock from which item is ordered
Quantity	number	Quantity of item ordered
Unit_price	number	Price of single item
Total_price	number	Total price

VENDOR DETAILS

Vname	string	Company name
Contact	string	Contact info of vendor
Address	string	Address of company
Location	string	Specific location
Email	string	Vendors email
url	string	url of company
logo	string	Logo of company
active	boolean	Active or not

VENDOR STOCK

Vendor	path	Vendor details
Product	path	which product
Expiry	date	Expiry date of this stock of product
C_price	number	Cost price of product
S_Price	number	Unit price of this stock of product
active	boolean	Active or not

REORDER

Product	path	Which product to reorder
vendor	path	request to which vendor
Quantity	number	How much to reorder
Reorder_date	timestamp	Date of reorder
status	string	Status of request
cprice	number	Current Cost price of product
Sprice	number	Current selling price of product

PURCHASE TABLE

Purchase id	string	Id of purchase
Vendor	path	From which vendor purchased
Date	date	Date of purchase
Amount	number	Total Amount of purchase
Status	string	Current status

PURCHASE DETAILS

purchase_id	path	Details of this purchase id is given below
product	path	Product id
quantity	number	quantity
stock	path	Path to new Stock
cost_price	number	Cost price of product
selling price	number	Selling price of product
totalprice	number	quantity * cost price

RATING

customer	path	Customers who posted review
product	path	Product which is reviewed
rating	number	Rating given to product out of 5.
review	string	Product review
r_date	timestamp	Reviewed Date
active	boolean	Active or not

WISHLIST

customer	path	Customers who added to wishlist
product	path	Product which is added to wishlist
added_date	timestamp	Added Date
active	boolean	Active or not

BRAND

brandname	string	Name of brand
brandlogo	string	Logo of Brand
added_date	timestamp	Added Date
active	boolean	Active or not

CHAPTER 5

SYSTEM TESTING

5.1 INTRODUCTION

Software testing is the practice of carefully controlling the execution of software in order to determine if it behaves as intended. The words verification and validation are frequently used in conjunction with software testing. Validation is the process of examining or evaluating a product, including software, to determine if it complies with all relevant specifications. One type of verification, software testing, employs methods including reviews, analyses, inspections, and walkthroughs as well. Verifying that what has been specified matches what the user truly want is the process of validation.

The processes of static analysis and dynamic analysis are additional ones that are frequently related to software testing. Static analysis examines the software's source code, searching for issues and obtaining statistics without actually running the code. Dynamic analysis examines how software behaves while it is running in order to offer data like execution traces, timing profiles, and test coverage details.

Testing is a collection of activities that may be planned ahead of time and carried out in a methodical manner. Testing starts with individual modules and progresses to the integration of the full computer-based system. There are many rules that may be used as testing objectives, and testing is necessary for the system testing objectives to be successful. As follows:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a test is successfully carried out in accordance with the aforementioned aims, it will reveal software bugs. Additionally, testing shows that the software functions seem to be functioning in accordance with the specifications and that the performance requirements seem to have been satisfied.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Testing for correctness is used to ensure that a programme performs precisely as it was intended to. This is considerably harder than it would initially seem, especially for big projects.

5.2 TEST PLAN

A test plan suggests a number of required steps that need be taken in order to complete various testing methodologies. The activity that is to be taken is outlined in the test plan. A computer programme, its documentation, and associated data structures are all created by software developers. It is always the responsibility of the software developers to test each of the program's separate components to make sure it fulfils the purpose for which it was intended. In order to solve the inherent issues with allowing the builder evaluate what they have developed, there is an independent test group (ITG). Testing's precise goals should be laid forth in quantifiable language. Therefore, the test plan should include information on the mean time to failure, the cost to locate and correct the flaws, the residual defect density or frequency of occurrence, and the number of test labour hours required for each regression test.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

5.2.1 Unit Testing

Unit testing concentrates verification efforts on the software component or module, which is the smallest unit of software design. The component level design description is used as a reference for testing crucial control pathways to find faults inside the module's perimeter. the level of test complexity and the untested area determined for unit testing. Unit testing is white-box focused, and numerous components may be tested simultaneously. To guarantee that data enters and exits the software unit under test appropriately, the modular interface is checked. To make sure that data temporarily stored retains its integrity during each step of an algorithm's execution, the local data structure is inspected. To confirm that each statement in a module has been performed at least once, boundary conditions are evaluated. All error handling pathways are then evaluated.

Before starting any other test, tests of data flow over a module interface are necessary. All other tests are irrelevant if data cannot enter and depart the system properly. An important duty during the unit test is the selective examination of execution pathways. Error circumstances must be foreseen in good design, and error handling pathways must be put up to cleanly redirect or halt work when an error does arise. The final phase of unit testing is boundary testing. Software frequently fails at its limits.

In the Sell-Soft System, unit testing was carried out by considering each module as a distinct entity and subjecting them to a variety of test inputs. The internal logic of the modules had certain issues, which were fixed. Each module is tested and run separately after development. To guarantee that every module functions properly and produces the desired outcome, all extraneous code was deleted.

5.2.1.1 Test Case

Test Case 1					
Project Name: BigShope					
Registration Test Case					
Test Case ID: Test_1			Test Designed By: Aravind V V		
Test Priority(Low/Medium/High):High			Test Designed Date: 18-07-2022		
Module Name: Customer			Test Executed By : T J Jobin		
Test Title : Register the user with email and password			Test Execution Date: 18-07-2022		
Description: Test the Registration Page					
Pre-Condition :User has valid email id and contact number and password					
Step	Test Step	Test Data	ExpectedResult	Actual Result	Status(Pass/Fail)
1	Navigation to Register Page	http://127.0.0.1:8000/regs	Register Page should be displayed	Register page displayed	Pass
2	Provide Valid Email Id	Email: aravindvinod@protonmail.com	User should be able to Register and go to	User Registered and went to login page	Pass
3	Provide Valid Password	Password: 12345 Confirm:12345			

4	Provide 10 digit phone number	Number:767 8989876	login page		
5	Provide Name	Name:Arav			
6	Click on Sign Up button				
5	Provide Invalid Email Id or password or contact number	Email Id: aravindvinod@protonmail.com Password: 123456 Confirm password: 123456 Number:78 78787878	User shouldn't register And redirect back to register page.	User didn't register as email already exists from previous registration. Invalid email,number etc can also cause error messages	Pass
6	Provide Null Email, Password, Name,Number	Email Id: null Password: null			
7	Click on Sign In button				
Post-Condition: User is validated and successfully registerd and can now login.					

Code

```

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import unittest

class TestExample(unittest.TestCase):

    @classmethod
    def setUpClass(cls):

        cls.selenium = webdriver.Chrome()

    def test_testform(self):

        self.selenium.get('http://127.0.0.1:8000/regs')

```

```
player_email = self.selenium.find_element('id','email')
player_name = self.selenium.find_element('id','name')
player_phno= self.selenium.find_element('id','phno')
player_password= self.selenium.find_element('id','pswd')
player_cpassword= self.selenium.find_element('id','cpswd')
submit = self.selenium.find_element('id','reg')
player_email.send_keys('aravindvinod@protonmail.com')
player_password.send_keys('123456')
player_cpassword.send_keys('123456')
player_phno.send_keys('7678987898')
player_name.send_keys('Arav')
submit.send_keys(Keys.RETURN)
```

```
def tearDown(self) -> None:
```

```
    self.selenium.close()
    self.selenium.quit()
    return super().tearDown()
```

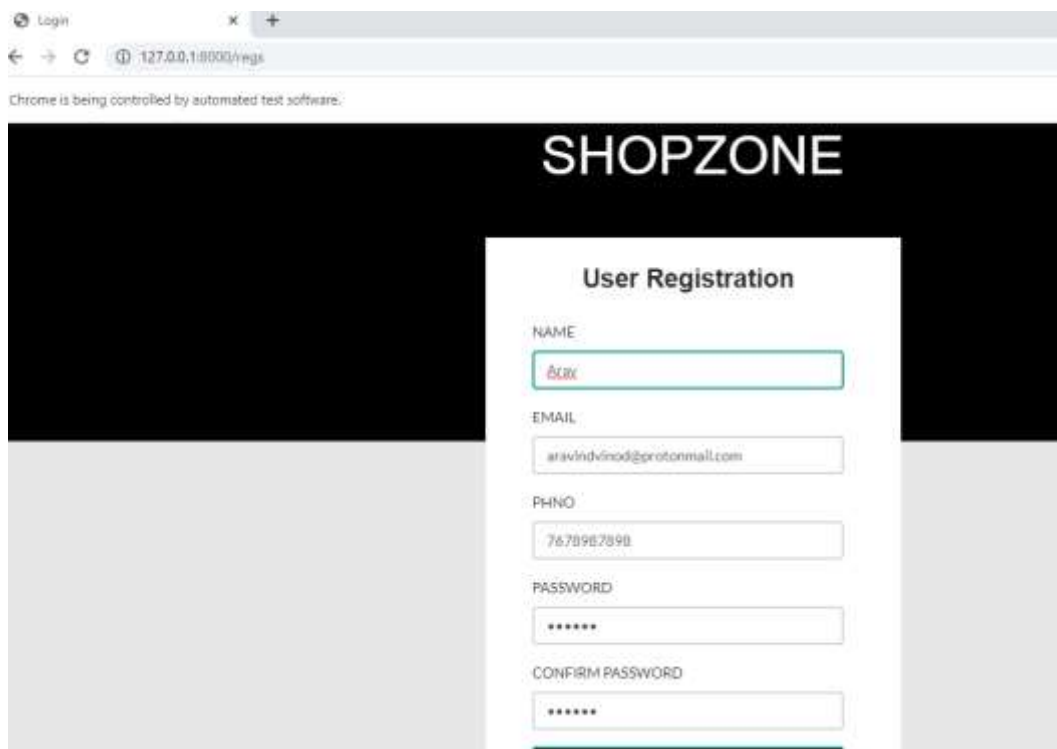
```

tests.py 6, M x  views.py 9+, M  <> page-login.html U  <> reg.html U
ecom > tests.py > TestExample > test_testform
9
10 class TestExample(unittest.TestCase):
11     @classmethod
12     def setUpClass(cls):
13         cls.selenium = webdriver.Chrome()
14     def test_testform(self):
15         self.selenium.get('http://127.0.0.1:8000/regs')
16         player_email = self.selenium.find_element('id','email')
17         player_name = self.selenium.find_element('id','name')
18         player_phno= self.selenium.find_element('id','phno')
19         player_password= self.selenium.find_element('id','pswd')
20         player_cpassword= self.selenium.find_element('id','cpswd')
21         submit = self.selenium.find_element('id','reg')
22         player_email.send_keys('aravindvinod@protonmail.com')
23         player_password.send_keys('123456')
24         player_cpassword.send_keys('123456')
25         player_phno.send_keys(7678987898)
26         player_name.send_keys('Arav')
27         submit.send_keys(Keys.RETURN)
28
PROBLEMS 26  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

DevTools listening on ws://127.0.0.1:56883/devtools/browser/348e4e49-22f4-447b-83a3-30aad41d7ca
[12032:2720:0718/105413.625:ERROR:device_event_log_impl.cc(214)] [10:54:13.616] USB: usb_device_handle_win.cc:16
r from node connection: A device attached to the system is not functioning. (0x1F)
[12032:2720:0718/105413.927:ERROR:device_event_log_impl.cc(214)] [10:54:13.919] USB: usb_device_handle_win.cc:16
r from node connection: A device attached to the system is not functioning. (0x1F)
-----
Ran 1 test in 41.904s

OK

```



The screenshot shows a web browser window with the URL `http://127.0.0.1:8000/regs` and a search bar containing the text "register". Below the browser, a code editor displays a Python script named `test_testform` using Selenium. The script defines a `test_testform` method that interacts with various form elements (email, name, phone, password, confirm password, and register button) and sends keys to them. The script is executed in a Jupyter environment, and the terminal output shows an error: `ERROR: test_testform (ecom.tests.TestExample)`. The traceback indicates an `UnexpectedAlertPresentException` with the message "Alert Text: Email Already Exists".

```

14
15 def test_testform(self):
16     self.selenium.get('http://127.0.0.1:8000/regs')
17     player_email = self.selenium.find_element('id','email')
18     player_name = self.selenium.find_element('id','name')
19     player_phno= self.selenium.find_element('id','phno')
20     player_password= self.selenium.find_element('id','pswd')
21     player_cpassword= self.selenium.find_element('id','cpswd')
22     submit = self.selenium.find_element('id','reg')
23     player_email.send_keys('aravindvinod@protonmail.com')
24     player_password.send_keys('123456')
25     player_cpassword.send_keys('123456')
26     player_phno.send_keys('8899776677')
27     player_name.send_keys('Arav')
28     submit.send_keys(Keys.RETURN)

```

PROBLEMS 26 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

DevTools listening on ws://127.0.0.1:56656/devtools/browser/7629b4e1-1575-4699-b4c4-5f49b47fba0a

E

ERROR: test_testform (ecom.tests.TestExample)

Traceback (most recent call last):

File "E:\P\ecommerce\ecom\tests.py", line 30, in tearDown

self.selenium.close()

File "C:\Python\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 552, in close

self.execute(Command.CLOSE)

File "C:\Python\lib\site-packages\selenium\webdriver\remote\webdriver.py", line 435, in execute

self.error_handler.check_response(response)

File "C:\Python\lib\site-packages\selenium\webdriver\remote\errorhandler.py", line 246, in check_response

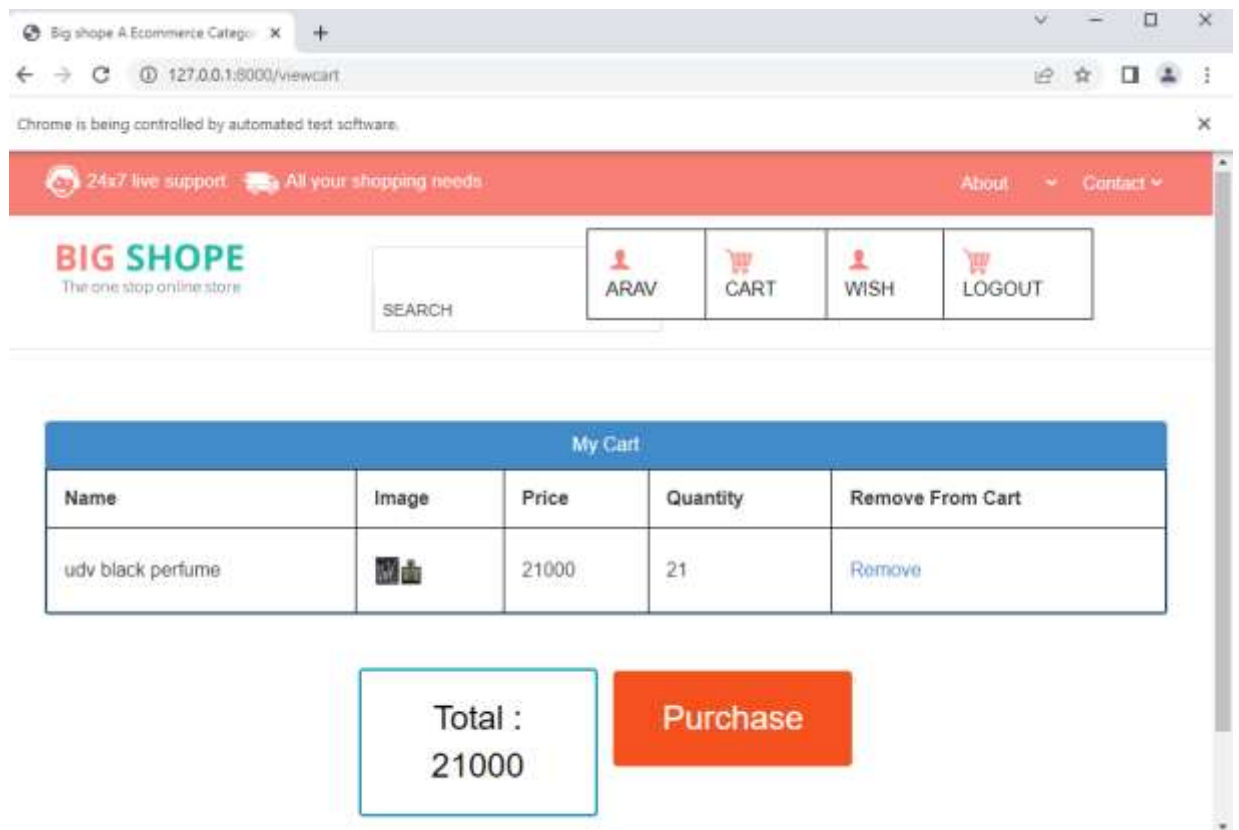
raise exception_class(message, screen, stacktrace, alert_text) # type: ignore[call-arg] # mypy is not smart enough here

selenium.common.exceptions.UnexpectedAlertPresentException: Alert Text: Email Already Exists

Test Case 2

Project Name: BigShope					
Cart Test Case					
Test Case ID: Test_2			Test Designed By: Aravind V V		
Test Priority(Low/Medium/High):High			Test Designed Date: 18-07-2022		
Module Name: Customer			Test Executed By : T J Jobin		
Test Title : Add to Cart			Test Execution Date: 18-07-2022		
Description: Test the Adding product to cart					
Pre-Condition :Login, choose product, add product to cart with valid quantity.					
Step	Test Step	Test Data	ExpectedResult	Actual Result	Status(Pass/Fail)
1	Navigation to Login Page	http://127.0.0.1:8000/logs	Login Page should be displayed	Login page displayed	Pass

2	Provide Valid Email Id	Email: aravindvinod@protonmail.com	User should login to index page	User Logged in to index page	Pass
3	Provide Valid Password	Password: 123456			
4	Click on Sign In button				
5	Provide Invalid Email Id or password	Email Id: aravindvinod@p.com Password: 123456	User shouldn't login	User didn't login. Message - Invalid Details	Pass
7	Click on Sign In button				
8	Navigate to product page.	Product is: http://127.0.0.1:8000/single/productid	A product should be chosen	Product is chosen from product page	Pass
9	Choose a product.				
10	Add Quantity	Quantity:2	Successfully add product to cart	Product added to cart	Pass
	Click Add to Cart				
11	Add Quantity	Quantity: a	Should not add	Cannot add product as quantity is invalid	Pass
	Click Add to Cart		Product to cart		
Post-Condition: User add a product and valid quantity into cart.					



Code

```
from selenium import webdriver

from selenium.webdriver.common.keys import Keys

import unittest

class TestExample(unittest.TestCase):

    @classmethod
    def setUpClass(cls):

        cls.selenium = webdriver.Chrome()

    def test_testform2(self):

        self.selenium.get('http://127.0.0.1:8000/logs')

        player_email = self.selenium.find_element('id','id_email')

        player_password= self.selenium.find_element('id','id_pass')

        submit = self.selenium.find_element('id','submits')

        player_email.send_keys('aravindvinod@protonmail.com')
```

```

player_password.send_keys('123456')

submit.send_keys(Keys.RETURN)

products = self.selenium.find_element('id','id_pro')

products.click()

self.selenium.get('http://127.0.0.1:8000/single/LmR6JyHzwEYptKtp4CxJ/9Q0c3YU
X1iUea1aiL2z7')

q = self.selenium.find_element('id','q')

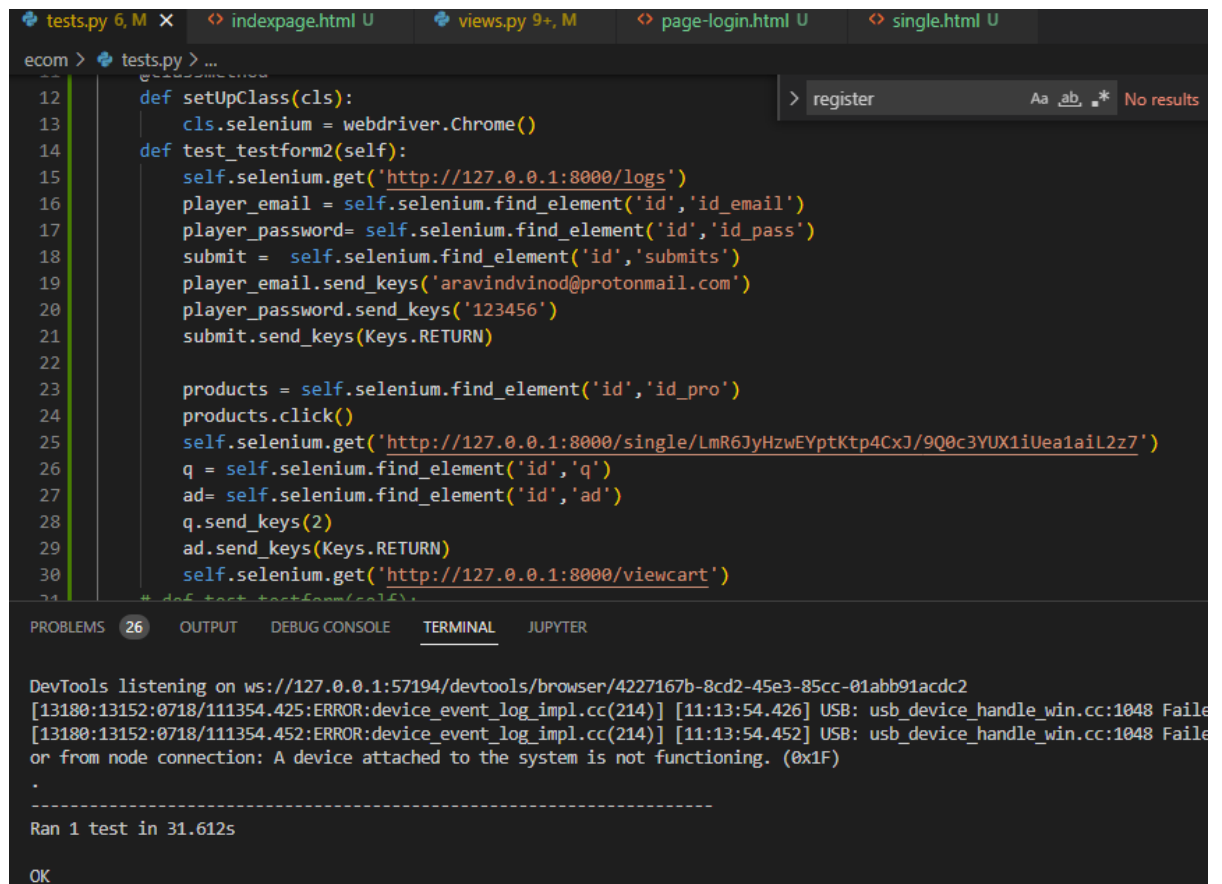
ad= self.selenium.find_element('id','ad')

q.send_keys(2)

ad.send_keys(Keys.RETURN)

self.selenium.get('http://127.0.0.1:8000/viewcart')

```



The screenshot shows a web browser window at the top with the URL `http://127.0.0.1:8000/single/LmR6JyHzwEYptKtp4CxJ/9Q0c3YUX1iUea1aiL2z7`. Below the browser is a code editor with a Python file named `tests.py`. The code defines a `setUpClass` method and a `test_testform2` method. The `test_testform2` method performs the following actions: it gets the page, finds the email and password fields, enters the email `aravindvinod@protonmail.com` and password `123456`, clicks the submit button, finds the product element, clicks it, finds the quantity and address fields, enters `2` in the quantity field and `ad` in the address field, and finally gets the viewcart page.

Below the code editor is a terminal window showing the output of the test. It indicates that DevTools is listening on `ws://127.0.0.1:57194/devtools/browser/4227167b-8cd2-45e3-85cc-01abb91acdc2`. There are two error messages from the USB device, both stating: `[13180:13152:0718/111354.425:ERROR:device_event_log_impl.cc(214)] [11:13:54.426] USB: usb_device_handle_win.cc:1048 Failed or from node connection: A device attached to the system is not functioning. (0x1F)`. The terminal also shows that the test ran successfully in 31.612s.

5.2.2 Integration Testing

Integration testing is a methodical approach for creating the program's structure while also carrying out tests to find interface issues. The goal is to construct a programme structure that has been determined by design using unit tested components. The software as a whole is tested. Correction is challenging since the size of the overall programme makes it hard to isolate the reasons. As soon as these mistakes are fixed, new ones arise, and the process repeats itself in an apparently unending cycle. All of the modules were merged once unit testing was completed in the system to check for any interface inconsistencies. A distinctive programme structure also developed when discrepancies in programme structures were eliminated.

5.2.3 Validation Testing or System Testing

This is the final step in testing. This involved testing the complete system in its entirety, including all forms, code, modules, and class modules. Popular names for this type of testing include system tests and black box testing.

The functional requirements of the programme are the main emphasis of the black box testing approach. That example, using Black Box testing, a software engineer may create sets of input circumstances that will thoroughly test every programme requirement.

Problems in data structures or external data access, erroneous or missing functions, interface faults, performance issues, initialization issues, and termination issues are all types of errors that black box testing looks for.

5.2.4 Output Testing or User Acceptance Testing

User approval of the system under consideration is tested, in this case, it must meet the needs of the company. When developing, the programme should stay in touch with the user and perspective system to make modifications as needed. With regard to the following points, this was done:

- Input Screen Designs,
- Output Screen Designs,

The aforementioned testing is carried out using a variety of test data. The preparation of test data is essential to the system testing process. The system under investigation is then put to the test using the prepared test data. When testing the system, test data issues are found again and fixed using the testing procedures described above. The fixes are also logged for use in the future.

CHAPTER 6

IMPLEMENTATION

6.1 INTRODUCTION

The project's implementation phase is where the conceptual design is transformed into a functional system. It can be regarded as the most important stage in creating a successful new system since it gives users assurance that the system will operate as intended and be reliable and accurate. User documentation and training are its main concerns. Usually, conversion happens either during or after the user's training. Implementation is the process of turning a newly updated system design into an operational one, and it simply refers to placing a new system design into operation.

The user department now bears the most of the workload, faces the most disruption, and has the biggest influence on the current system. Uncontrolled implementation can create confusion.

Implementation encompasses all of the steps used to switch from the old system to the new one. The new system might be entirely different, take the place of an existing manual or automated system, or it could be modified to work better. A dependable system that satisfies organisational needs must be implemented properly. System implementation refers to the process of actually using the built system. This comprises all the processes involved in switching from the old to the new system. Only after extensive testing and if it is determined that the system is operating in accordance with the standards can it be put into use. The system employees assess the system's viability. The work necessary for system analysis and design to implement the three key components of education and training, system testing, and changeover will increase in proportion to how complicated the system being implemented is.

The implementation state involves the following tasks:

- Good planning.
- Checking of system and constraints.
- Design of methods for the changeover.

6.2 IMPLEMENTATION PROCEDURES

Software implementation refers to the complete installation of the package in its intended environment, as well as to the system's functionality and fulfilment of its intended applications. The software development project is frequently commissioned by someone who will not be using it. People first have doubts about the software, but it's important to

make sure that resistance doesn't grow since one needs to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before examining the system, the user must be aware that the server software has to be running on the server in order to access the results. The real procedure won't happen if the server object is not active and functioning on the server.

6.2.1 User Training

The purpose of user training is to get the user ready to test and modify the system. The individuals who will be participating must have faith in their ability to contribute to the goal and advantages anticipated from the computer-based system. Training is more necessary as systems get more complicated. The user learns how to input data, handle error warnings, query the database, call up routines to generate reports, and execute other important tasks through user training.

6.2.2 Training on the Application Software

The user will need to receive the essential basic training on computer awareness after which the new application software will need to be taught to them. This will explain the fundamental principles of how to use the new system, including how the screens work, what kind of help is displayed on them, what kinds of errors are made while entering data, how each entry is validated, and how to change the date that was entered. Then, while imparting the program's training on the application, it should cover the knowledge required by the particular user or group to operate the system or a certain component of the system. It's possible that this training will vary depending on the user group.

6.2.3 System Maintenance

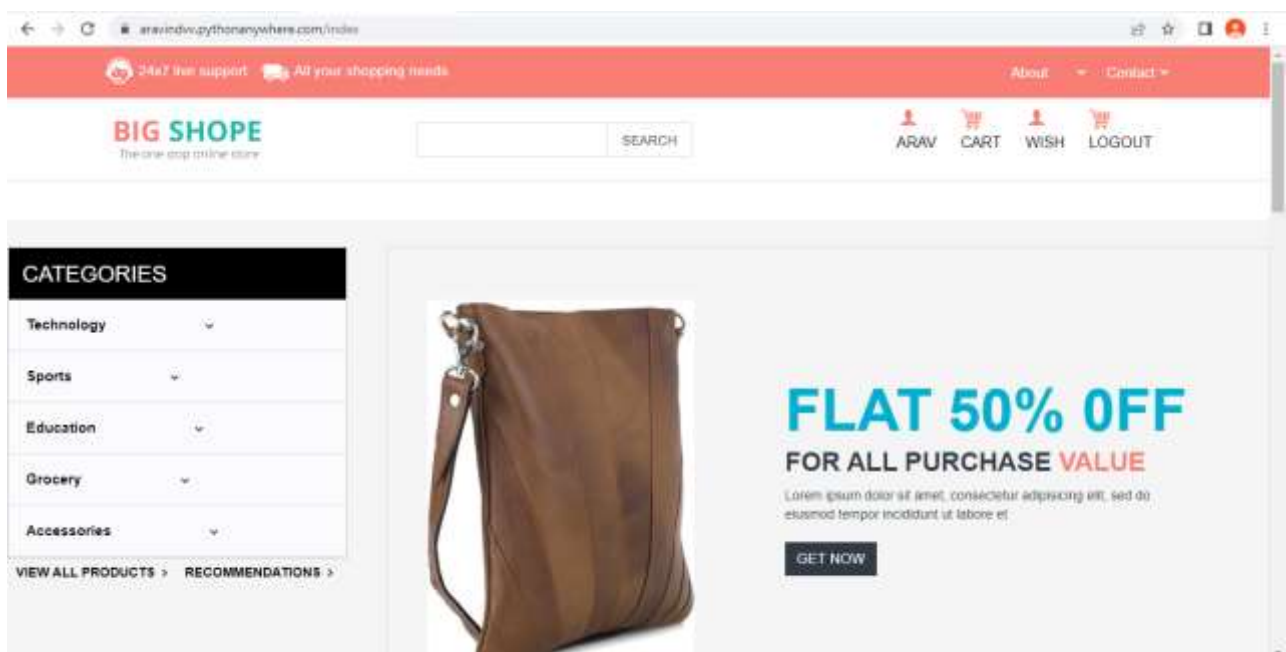
The mystery of system development is maintenance. When a software product is in the maintenance stage of its lifecycle, it is actively working. A system should be properly maintained after it has been effectively installed. An essential part of the software development life cycle is system maintenance. In order for a system to be flexible to changes in the system environment, maintenance is required. Of course, software maintenance involves much more than just "Finding Mistakes".

6.3 Hosting

A web hosting service is a type of internet hosting service that hosts websites for clients, ie it offers the facilities required for them to create and maintain a site and makes it accessible on the World Wide Web. The system is hosted in PythonAnywhere. PythonAnywhere provide free and easy hosting of python projects upto 500 MB in project size. The database is Google Firestore which is already hosted by Google. PythonAnywhere provide a free domain with our registered username. The url of the hosted website is: [https:// aravindvv.pythonanywhere.com](https://aravindvv.pythonanywhere.com).

Steps of hosting process:

- Register a free account in pythonanywhere with valid username and password.
- Clone you project from your github or upload project folder directly.
- Create a virtual environment with your python version.
- Install required dependencies in the virtual environment (Django, Firebase, etc).
- Create a web app with your python version. Connect the virtual environment with created web app.
- Configure the wsgi file for Django with your project details, also configure static files.
- Reload your web app and use the given url to access your hosted website online.



CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The proposed system offers the ability for customers to examine all of the shop's items and make orders, for the owner to keep track of store activity, and for vendors to add their products directly into the system in accordance with requirements. This system has been designed to be flexible and simple to use. It has been created in a way that makes it simple to edit and that allows for accurate and effective updating. Because of the system's simplicity and adequate instructions, the user won't be uncertain about how it works. The system was built with Django Framework and tested properly. Data is guaranteed to be safe, backed up, and constantly accessible in the cloud with Google Firebase acting as the backend. As whole, the system was well planned and designed. The system's performance is assured to be reliable. Through the system, all consumers benefit in the long run. The system provides flexibility for incorporating new features, which may be necessary in future.

CHAPTER 8

BIBLIOGRAPHY

REFERENCES:

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- Pankaj Jalote, “*Software engineering: a precise approach*”, 2010.
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES:

- www.w3schools.com, www.geeksfor geeks
- www.django.com
- Big Shope templates
- Youtube.com
- Draw.io, figma.com
- Google firebase documentations

CHAPTER 9

APPENDIX

9.1 Sample Code

Views.py

LOGIN

```
def login2(request):  
  
    if request.method=='POST':  
        email = request.POST['email']  
        password = request.POST['password']  
        if email=="admin@admin.com" and password == "123456":  
            try:  
                login = authe.sign_in_with_email_and_password(email,password)  
                uid = login['localId']  
                request.session['adminid']=uid  
                return HttpResponseRedirect('/adminindex')  
            except:  
                messages.error(request,"Invalid Credentials")  
                return HttpResponseRedirect('/logs')  
        try:  
            login = authe.sign_in_with_email_and_password(email,password)  
            uid = login['localId']  
            d=db.collection('Customer').where('userid','==',uid).where('active','==',"true").get()  
            if len(d)==0:  
                messages.error(request,"Account Disabled");  
                return redirect(logs)  
        else:  
            pass  
    except:  
        messages.error(request,"Invalid Credentials")  
        return HttpResponseRedirect('/logs')  
  
    return redirect('/logs')
```

REGISTER

```
def register2(request):  
    if request.method=='POST':  
  
        email = request.POST['email']  
        password = request.POST['pswd']  
        name = request.POST['name']  
        mobile = request.POST['phno']  
        today = datetime.now()  
        try:  
            user=authe.create_user_with_email_and_password(email,password)  
            #link=authe.generate_email_verification_link(email, action_code_settings=None)  
            generate_random = random.randint(111111,999999)  
            generate_random = str(generate_random)  
            email_from = settings.EMAIL_HOST_USER
```

```

        res = send_mail("Big Shope Registration OTP", generate_random, email_from, [email])
        generate_random = int(generate_random)
        uid = user['localId']
    except:
        messages.error(request,("Email Already Exists"))
        return render(request,'ecom/reg.html')

    today = datetime.now()
    db.collection('Customer').add(
        {
            'customer_name': name,
            'creationDate': today,
            'userid':uid,
            'contact': mobile,
            'email':email,
            'password':password,
            'active':"false",
            'role':'user'
        }
    )
    return render(request,'ecom/verifymail.html',
    {'email':email,'generate_random':generate_random})

```

PRODUCT DETAILS

```

def single2(request,pid,sid):
    if request.session.is_empty():
        return redirect(logs)
    docs = db.collection('Product').document(pid).get()
    #context = {'datas': [doc.id,doc.to_dict() for doc in docs]}

    docs = docs.to_dict()
    sub = docs["subcategory"].get().to_dict()

    cat = db.collection('persons').document(aa).
    collection('subcategory').document(docs["sub"]).get()

    stock = db.collection('Stock').document(sid).get()
    stocks = stock.to_dict()
    stocks["id"]=stock.id

    category = db.collection('persons').get()
    cats = []
    subcats = []

    for cat in category:
        catdict = cat.to_dict()
        catdict["id"]=cat.id
        cats.append(catdict)

    subcat = db.collection('persons').document(cat.id).collection('subcategory').get()
    for sub in subcat:
        subdict = sub.to_dict()
        subdict["sub_id"]=sub.id

```



```

        subdict["cat_id"]=cat.id
        subcats.append(subdict)

    cus_rate = request.session['id']
    rateid = db.document('Customer/'+cus_rate)
    proid = db.document('Product/'+pid)
    orderstatus = 0
    orderid=db.collection('Orders').where('cus_order','==',rateid).get()
    for order in orderid:
        orderef = db.document('Orders/'+order.id)
        a=db.collection('Orderdetails').
    where('orderid','==',orderef).where('orderproduct','==',proid).get()
        if a != None:
            orderstatus = 1

    print(orderstatus)
    rating = db.collection('Rating').where('cus_rating','==',rateid).
    where('pro_rating','==',proid).get()
    ratdict = { }
    ratingid=0
    for rat in rating:
        ratingid = rat.id
        ratdict = rat.to_dict()
    res = not ratdict
    ratelist=[]
    allratings = db.collection('Rating').where('pro_rating','==',proid).get()
    for allrating in allratings:
        allratingdict = allrating.to_dict()
        customerd=allratingdict['cus_rating'].get().to_dict()
        result=allratingdict | customerd
        ratelist.append(result)

    wish = db.collection('Wishlist').where('cref','==',rateid).where('pref','==',proid).get()
    mywish = 0
    for w in wish:
        wd = w.to_dict()
        mywish=1
    context = {'orderstatus':orderstatus,'products':docs,'sub':sub,'cats':
    cats,'subcats':subcats,'stock':stocks,'pid':pid,'sid':sid,'rating':ratdict,'res':res,'ratingid':ratingid,
    'rlist':ratelist,'mywish':mywish}
    return render(request,'ecom/single.html',context)

```

ADD TO CART

```

def addtocart2(request):
    if request.session.is_empty():
        return redirect(logs)
    if request.method=='POST':
        stockid = request.POST['stockid']
        quantity = request.POST['quantity']
        stocked = db.collection('Stock').document(stockid).get()
        stockdetails=stocked.to_dict()
        productdetails = stockdetails["pro_refer"].get()
        productid = productdetails.id
        customerid = request.session["id"]

```

```

stockdetails=stocked.to_dict()
quantity = int(quantity)
currentquantity = stockdetails["quantity"]
currentquantity=int(currentquantity)
if quantity > currentquantity:
    messages.success(request, ("Out of Stock"))
    return redirect(single2, pid = productid, sid= stockid)
unitprice = stockdetails["price"]
totalprice = unitprice * quantity
# if totalprice > 20000:
#     messages.success(request, ("20000 is transaction limit"))
#     return redirect(single2, pid = productid, sid= stockid)
cus_ref=db.document('Customer/'+customerid)
stock_ref=db.document('Stock/'+stockid)

alreadyadded=db.collection('Cart').where('cus_ref','==',cus_ref).where('pro
_ref','==',stockdetails["pro_refer"]).get()
print(alreadyadded)
if alreadyadded != []:
    for added in alreadyadded:
        adddict = added.to_dict()
        currentquant=adddict["quantity"]
        addquantity = currentquant + quantity
        addtotalprice = addquantity * adddict['unitprice']

db.collection('Cart').document(added.id).update({'quantity':addquantity,'tot
alprice':addtotalprice})
else:

db.collection('Cart').add({'product_id':productid,'stocknumber':stock_ref,'q
uantity':quantity,'cus_ref':cus_ref,'stock_ref':stock_ref,'pro_ref':stockdetails
["pro_refer"],'totalprice':totalprice,'unitprice':unitprice})

newq = currentquantity-quantity
db.collection('Stock').document(stockid).update({'quantity':newq})
newstock = db.collection('Stock').document(stockid).get().to_dict()
# if newstock["quantity"]==0:
#     db.collection('Stock').document(stockid).update({'active':'false'})
messages.success(request,"Added to Cart")
return redirect(single2, pid = productid, sid = stockid)

```

ORDER

```

def pay2(request):
    if request.session.is_empty():
        return render(request,'ecom/page-login.html')

    status2="Ordered"
    customerid = request.session["id"]
    cusid=db.document("Customer/"+customerid)

    cartitems=db.collection('Cart').where("cus_ref","==",cusid).get()
    cartlist=[]

```

```
amount=0
for cart in cartitems:
    cartdict = cart.to_dict()
    amount=amount+cartdict["totalprice"]
amount=int(amount)
orderdate = datetime.today()
location=request.COOKIES['location']
print(location)
loc_id=db.document("Location/"+location)
roid=request.COOKIES['razorpay_order_id']
print(roid)
poid=request.COOKIES['razorpay_payment_id']
print(poid)

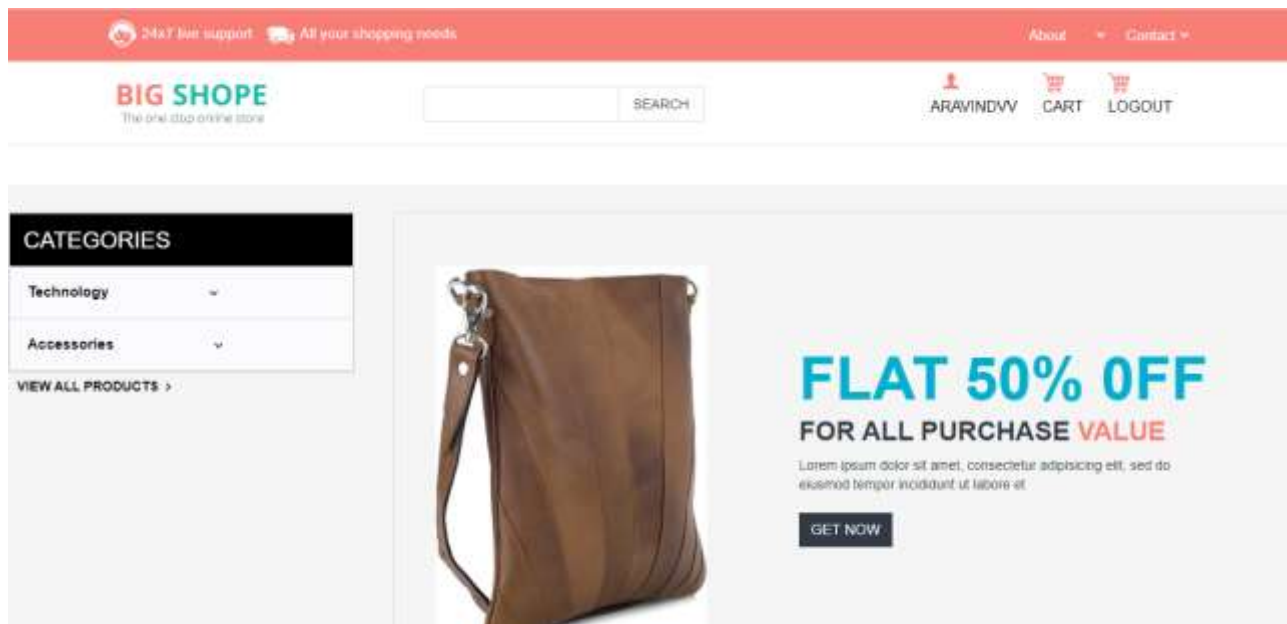
orders=db.collection('Orders').add({'cus_order':cusid,'loc_order':loc_id,'amount':amount,'orderdate':orderdate,'deliver_date':orderdate,'order_status':status2,'pay_orderid':roid,'payment_id':poid,'active':'true'})

orderid = db.document("Orders/"+orders[1].id)
a=redirect(ordereport,orderid=orders[1].id)
a.delete_cookie('location')
a.delete_cookie('razorpay_order_id')
a.delete_cookie('razorpay_payment_id')
for cart in cartitems:
    cartdict = cart.to_dict()
    ordersdetails=
db.collection('Orderdetails').add({'orderid':orderid,'orderproduct':cartdict["pro_ref"],'orderstock':cartdict["stocknumber"],'quantity':cartdict["quantity"],'unitprice':cartdict["unitprice"],'totalprice':cartdict["totalprice"]})

for cart in cartitems:
    db.collection('Cart').document(cart.id).delete()
reordercheck(orders[1].id)
return a
```

9.2 Screen Shots

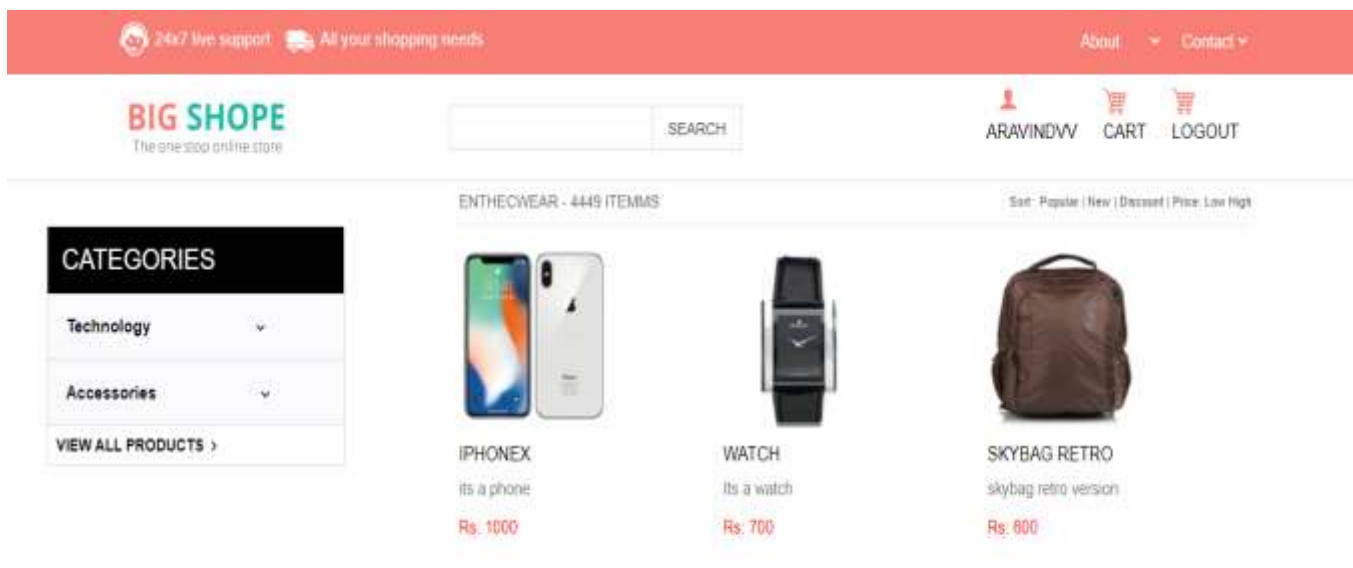
Home page



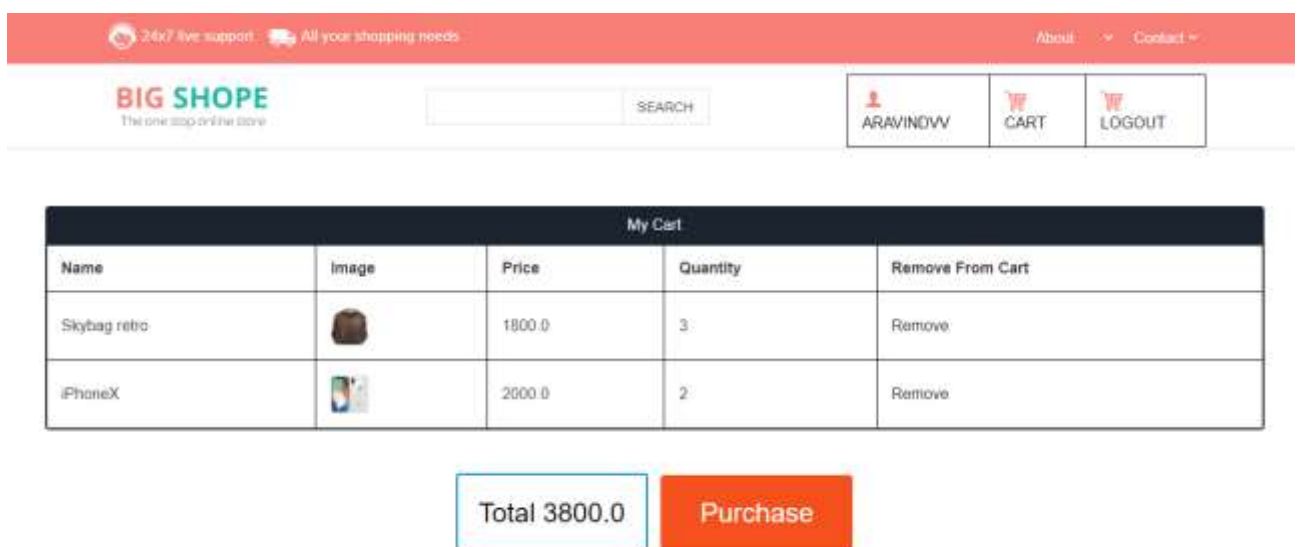
Register page

The screenshot shows a 'User Registration' form. It includes input fields for 'NAME', 'EMAIL', 'PHNO' (Phone number), 'PASSWORD', and 'CONFIRM PASSWORD'. A red 'REGISTER' button is at the bottom. The form is set against a dark background with a light gray border.

Product display



Cart



Order details

[print](#)

ORDER INFORMATION

Order date :	July 12, 2022, 3:58 p.m.
Payment id :	pay JsQCsbns3ocqwU
Order id :	order JsQBakCwzGnTKd
Total Amount :	70
Status :	Ordered
Customer Name :	Aravind
Email :	ara@gmail.com
Location :	Home, Manatavady, 667744

Products



HP 1 70

PAPER NAME

mainbody.pdf

AUTHOR

Aravind V

WORD COUNT

10389 Words

CHARACTER COUNT

58843 Characters

PAGE COUNT

69 Pages

FILE SIZE

1.3MB

SUBMISSION DATE

Jul 21, 2022 3:33 PM GMT+5:30

REPORT DATE

Jul 21, 2022 3:33 PM GMT+5:30

● 16% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 16% Internet database
- 0% Publications database

● Excluded from Similarity Report

- Crossref database
- Bibliographic material
- Small Matches (Less than 10 words)

● 16% Overall Similarity

Top sources found in the following databases:

- 16% Internet database
- 0% Publications database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	issuu.com Internet	4%
2	seminarprojects.com Internet	1%
3	slideshare.net Internet	1%
4	docshare.tips Internet	1%
5	cloud.google.com Internet	1%
6	tutorialspoint.com Internet	1%
7	whatis.techtarget.com Internet	<1%
8	origin.geeksforgeeks.org Internet	<1%
9	geeksforgeeks.org Internet	<1%

10	coursehero.com	Internet	<1%
11	seminartopicsforcomputerscience.com	Internet	<1%
12	guru99.com	Internet	<1%
13	slides.com	Internet	<1%
14	docplayer.net	Internet	<1%
15	django project.com	Internet	<1%
16	easystudy.info	Internet	<1%
17	shrvenkataraman.github.io	Internet	<1%
18	couchbase.com	Internet	<1%
19	192.192.246.204	Internet	<1%
20	firebase.google.com	Internet	<1%
21	dzone.com	Internet	<1%

22	gitlab.sliit.lk Internet	<1%
23	studentsrepo.um.edu.my Internet	<1%
24	liiamra.com Internet	<1%
25	trickideas.com Internet	<1%
26	tudr.thapar.edu:8080 Internet	<1%
27	pdfcoffee.com Internet	<1%
28	repository.president.ac.id Internet	<1%
29	seminarprojects.org Internet	<1%