Swift Buy Hub

Final Year Project

Session 2020-2024

A project submitted in partial fulfillment of the degree of

BS in Computer Science

Department of Computer Science

Faculty of Computer Science & Information Technology

The Superior University, Lahore

Spring 2024

Type (Nature of project)	[ ] **D**evelopment [ ] **R**esearch [✓] **R&D**
Area of specialization	Web Development
FYP ID	FYP-BCSM-F23032

Project Group Members

Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)		Hussnain Sarwar	Bcsm-f20-111@superior.edu.pk	
(ii)		Rana Waleed	Bcsm-f20-188@superior.edu.pk	
(iii)		Samra Tariq	Bcsm-f20-422@superior.edu.pk	

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to the work of others

Plagiarism Free Certificate

This is to certify that, I Hussnain Sarwar S/D of Muhammad Sarwar, group leader of FYP under registration no FYP-BCSM-F23032 at Software Engineering Department, The Superior College, Lahore. I declare that my FYP report is checked by my supervisor.

Date:

Name of Group Leader: Hussnain Sarwar Signature: _____

Name of Supervisor: Dr. Humayun Khan
Designation: Associate Professor
Signature: _____

Co-Supervisor: Mr. Rizwan
Designation: Lecturer
Signature: _____

HoD: Dr. Irfan Ud Din Signature: _____

Swift Buy Hub

Change Record

Author(s)	Version	Date	Notes	Supervisor's Signature
Samra Tariq Hussnain Waleed	1.0	2-08-2023	started working on data gathering	
Samra Tariq Hussnain Waleed	1.0	15-08-2023	Research on selected project	
Samra Tariq Hussnain Waleed	1.0	30-08-2023	Supervisor approval to start the project and advised to initiate the project.	
Samra Tariq	1.0	15-09-2023	Worked on the approved project has been started.	
Samra Tariq Rana Waleed	1.0	30-09-2023	Initial Phase, Working on project documentation.	
Samra Tariq Hussnain	1.0	20-10-2023	Adding and Removal of content	
Samra Rana Waleed Hussnain	1.0	01-11-2023	Documentation completed.	
Rana Waleed	2.0	15-11-2023	Designing The Product	

APPROVAL

PROJECT SUPERVISOR

Comments: _____

_____

Name:_____

Date:_____ Signature:_____

PROJECT MANAGER

Comments: _____

_____

Date:_____ Signature:_____

HEAD OF THE DEPARTMENT

Comments: _____

_____

Date:_____ Signature:_____

Dedication

This project is dedicated to the All-Powerful God for his love, direction, and protection during my academic career.

Acknowledgements

I give thanks to the Almighty God for his protection, wisdom, knowledge, and insight during my academic career. My supervisor, Mr. Humayun Khan, has my sincere gratitude for his unwavering support and meticulously coaching me throughout the time. Despite his busy schedule and obligations, he accommodated all of my requests for his time. I appreciate your diligence and skill.

Executive Summary

The project aims to develop an online shopping platform with integrated artificial intelligence capabilities for small and medium retailers. The solution called Swift Buy Hub provides an easy-to-use webstore to enable transactions along with smart price predictions using machine learning to suggest optimal pricing. Additionally, it incorporates a conversational chatbot to offer voice-based shopping assistance. Built on cloud infrastructure leveraging NodeJS, MongoDB and Angular, Swift Buy Hub hopes to make advanced technologies accessible for niche ecommerce segments. The automated and intelligent features distinguish it from typical catalog-driven platforms to solve pain points around inventory planning, demand forecasting and customer engagement for virtual storefronts. After launch, the next phase involves extending its recommendations engine for personalized promotions and loyalty programs to further boost customer experience.

This summarizes the objective, key capabilities, technology stack, target user base, overall vision and future roadmap for the Swift Buy Hub ecommerce web application. Please let me know if you need any modifications or have additional guidelines on structuring the executive summary. I'm happy to refine it further.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1: Introduction

Nowadays, everyone is aware with the term "online shopping," which can apply to anything from electronics to cars. We decided to innovate or provide new solutions to the current websites in order to concentrate on business produced from online resources for our final year project. Due to its convenience, everyone prefers to buy and sell online these days. We thus worked on it, but because cars and laptops are such a broad topic, we just chose to focus on it. We do many studies on the primary issue that consumers of laptops and cars encounter, which is that they are unaware of the precise cost of the item based on its specifications.. There's a point to focus on here is that the existing solutions such as OLX and PakWheels are working on both categories no doubt they are working well but there's a need for innovation in this type of classified website. The consumer wants to get the exact price by comparing the specs and also there should be a more convenient way to search for the product which is the best match according to the information we provided. By considering these problems we worked on this and innovated into the existing solutions but we created a new classified website Named "Swift Buy Hub". In this, we consider the consumer's concerns and introduce a price prediction model which is to predict the price of the product according to the specs the seller provides.
To support the purpose of easy buying and selling we are proposed the solution "Swift Buy Hub" which is a classified website where sellers can publish ads and the buyers can get a predicted price as well as enhanced searching in the form of a chatbot for more precise results.

1.1. Background

The E-commerce industry has boomed in recent years. In Today's world, everyone tries to do anything that is convenient for them. In this era everyone go for a thing which is more time convenient to them. One of the main thing people loves to do is online buying but people are also much concerned about what they see online and what they will get after placing the order. Two categories which are mainly focused by the youth is Cars and gadgets and more so. For our Final Year Project, we choose to focus on business generated from online resources by innovating or introducing new solutions to the existing websites. Today Everyone likes to Buy and sell online because it is more convenient for them. So we worked on this but we selected a limited category of Cars and Laptops as it is a vast category in itself. We do different Research on the main problem which laptop and car buyers face is that they don't know the exact price of the product according to the specs. There's a point to focus on here is that the existing solutions such as OLX and

PakWheels are working on both categories no doubt they are working well but there's a need for innovation in this type of classified website. The consumer wants to get the exact price by comparing the specs and also there should be a more convenient way to search for the product which is the best match according to the information we provided. By considering these problems we worked on this and innovated into the existing solutions but we created a new classified website Named "Swift Buy Hub". In this, we consider the consumer's concerns and introduce a price prediction model which is to predict the price of the product according to the specs the seller provides.
To support the purpose of easy buying and selling we are proposed the solution "Swift Buy Hub" which is a classified ads website where sellers can publish ads and the buyers can get a predicted price as well as enhanced searching in the form of a chatbot for more precise results and market accurate price.

1.2. Motivations and Challenges

The thing that motivates us to work on this project is the existing e-commerce websites which are complex for first-time sellers and also there was no innovation in the existing solutions. Existing classified websites are working on just simple search models and there's no precise filtration of search models used to make it easy for consumers to buy and sell products easily. Our customized site will simplify processes for merchants as well as for people who have zero knowledge of the products. Challenges we can face are how we can predict the price of the products and how to train the AI models to give the predicted price and a precise search algorithm.

1.3. Goals and Objectives

- Develop scalable and reliable online applications using ExpressJS, AngularJS, NodeJS, and MongoDB.
- Enhance UI/UX by incorporating features like product search, filters, reviews, and ratings.
- Simplify the process for retailers to list items and manage inventory.
- Simplify ad publishing for the sellers
- Sellers and buyers direct chat
- Physical inspection option

- A price prediction tool leveraging machine learning techniques and statistical models.
- Develop a smart chatbot to provide immediate responses to customer inquiries.

1.4. Literature Review/Existing Solutions

MEAN stack is the priority choice of developers who want to build large-scale apps and e-commerce projects as it is more scalable and more flexible in any terms [4].
In literature Review we will cover importance of MEAN stack to build a classified website and how AI models integration is sustainable in MEAN technology.

2.1. MEAN Stack Overview

MEAN stack is a set of Javascript technologies that allows to create large scale applications. MEAN stands for MongoDB, Express.js, AngularJS, and Node.
These technologies used to build a robust and scalable web application.

2.1.1. MongoDB

MongoDB is a noSQL database which provides flexible and scalable solution to store large amount of data. noSQL means it is not a relational database. It can store large amount of data in minimal number of document[5].

2.1.2. Express JS

Express is a server-side frame build on Node.js runtime. It is designed to handle client-server interaction. The primary function of Express is to manage routing and support HTTP methods. To achieve these tasks express works on middleware architecture [6].

2.1.3. Angular

Angular is and open source JavaScript framework maintained by google. It is designed for the creation of client-side application.it is known for developing Single Page application (SPA's), it loads the entire website with and initial request which works more efficiently on client side and also reduces the server load. AngularJS operates on an MVW (Model View Whatever) architecture, providing developers with the flexibility to implement the framework in various ways to suit their project need [7].

2.1.4. Node.js

Node.js is a runtime environment for JavaScript build on Google V8 engine. Node.js utilizes an asynchronous I/O event-driven model, making it ideal for developing scalable network applications [8].

2.2. AI Integration with NodeJS

Scalability being the core feature of this web project to deliver its users a fully responsive and scalable experience. AI also being one of the unique core features of this web project helps stands out this project among other relative competitors relatively.
Integration of AI with the frameworks of NodsJS or simply MEAN framework is a highly advantageous option. Running AI technologies on other language stacks can be quite challenging in times and cause the desired output to be none of the standard type. On the other hand NodeJS and AI implementation together is quite a different ride.
NodeJS has an access to a very large number of potential libraries for such a scale of development ultimately resulting in an excellent high standard driven product with AI capabilities. NodeJS with AI implementation allows access to libraries such sci-kit learn, DialogBox & Twilio for chatbots predictive models recognition system and even so.
In conclusion implementing of AI technologies with NodeJS framework is an ultimate step for creating such a high scalable project with MEAN Stack implemented with AI capabilities like chatbot, predictive models datasets analysis or even recognition system of specific products or person.[9]

1.5. Gap Analysis

While base ecommerce capability exists, there is gap in providing market-specific specialized functionality, smart price predictions and conversational commerce.

1.6. Proposed Solution

After considering all the issues consumers facing while using other classified websites that they are not innovating or introducing new features. Swift Buy Hub utilizes the MEAN stack for its robust and scalable architecture. The platform includes:

User-Friendly Interface:

Swift Buy Hub is working on to introduce a user-friendly interface for both web and mobile ensure easy navigation and a seamless shopping experience.

Product Catalog:

In this website all products categories will be managed in a proper catalogue. Users can easily access whatever category they want.
it will make convenient for the user to use our platform

AI-Powered Personalization:

for better experience we introduce price prediction model for the users who want to compare the price of the product according to the specs they mentioned. This model will fetch the information from the product specific dataset or data model and they will give a predicted price for the product.

Security Measures:

Implement robust security protocols to protect user data, transactions, and ensure a secure online environment.

Search and Filtering:

Develop advanced search and filtering options to help users quickly find products based on various criteria, such as category, price range, and brand giving out the specified needs. For this we created a chatbot which will enhance the search option and user can see exact product for the category with same features they want.

User Registration and Profiles:

User will only register one profile to do both tasks. They can buy and sell products from the same profile.

Messaging System: - Implement a messaging system that facilitates communication between buyers and sellers easily within the platform.

Product Listings and Management: [1] [2]

Users can easily list their products from their profile and can run classified ads on one. Users can manage the listing and ads from the same panel.

Physical Inspection:

There will be a additional feature of physical inspection but this feature will be introduced on premium mode of the website. Users will have to pay for the physical inspection of the product they want to buy.

Scalability and Performance:

Ensure the platform is scalable to accommodate a growing user base and optimized for performance.

1.7. Project Plan

We will follow an agile approach to build the website incrementally in sprints. Each sprint duration will be 2 weeks. Daily standup meetings will track progress and impediments.

1.7.1. Work Breakdown Structure

Project Initiation (Week 1-2)

- Define project scope, objectives, and stakeholders.
- Set up project management tools and communication channels.
- Assemble project team and assign roles and responsibilities.

Requirements Gathering (Week 3-4)

- Conduct user interviews and surveys to gather requirements.
- Document functional and non-functional requirements.
- Define project milestones and deliverables.

System Design (Week 5-8)

- Architectural design of the platform.
- Design database schema.
- UI/UX design for web interfaces.
- Define AI recommendation algorithms and personalization features.

Development (Week 9-20)

- **Frontend development:** Implement responsive web interfaces.
- **Backend development:** Set up server, API development, and database integration.
- Implement AI recommendation engine.
- Implement Price prediction
- Payment integration, messaging system, and community features.
- Implement security measures and testing.

Testing and Quality Assurance (Week 21-24)

- Conduct unit, integration, and end-to-end testing.
- Address bugs, glitches, and security vulnerabilities.
- Ensure cross-browser and cross-device compatibility.
- Performance testing and optimization.

Documentation (Week 25-26)

- Create comprehensive user and developer documentation.
- Prepare API documentation using Swagger or similar tools.

Deployment (Week 27-28)

- Deploy the platform on AWS or Heroku for production.
- Set up continuous integration and deployment (CI/CD) pipelines.
- Ensure scalability and high availability.

User Testing and Feedback (Week 29-32)

- Invite users to test the platform.
- Gather user feedback for iterative improvements.
- Conduct usability testing and address user suggestions.

Marketing and User Acquisition (Week 33-34)

- Develop a marketing strategy to promote Swift Buy Hub.
- Launch promotional campaigns and user acquisition initiatives.

Project Evaluation and Review (Week 35-36)

- Evaluate project outcomes against initial objectives and requirements.
- Conduct a project review with the team and stakeholders.
- Identify lessons learned and areas for future improvements.

|----------|----------|----------|----------|

Initiation Requirements System Design Development

|----------|----------|----------|----------|

Testing & QA Documentation Deployment User Testing

|----------|----------|----------|----------|

Marketing Evaluation & Review

1.7.2. Roles & Responsibility Matrix

In a team of three members working collaboratively on the entire project:

Hussnain: Frontend and Back-End (Leader)

Waleed: Front end and Testing

Samra: Documentation

1. Team Collaboration:

- All team members work together on project tasks and activities.
- Collaboratively decide on project goals, priorities, and deadlines.
- Regular communication and coordination among team members are essential for project success.

2. Technical Responsibilities:

- Waleed and Hussnain will contribute to technical development tasks, including:
    - Frontend development (HTML, CSS, JavaScript).
    - Backend development (Node.js, Express.js).
    - Database management (MongoDB).
    - AI recommendation engine integration.
    - Payment gateway integration.
    - Messaging system implementation.
    - Community features development.
    - Server configuration and deployment.

3. Documentation Responsibilities:

- Samra and Waleed collaborate on project documentation, including:
    - User and developer documentation.
    - Technical documentation for the backend.
    - Ensuring that all documentation is complete, accurate, and up-to-date.

4. Project Coordination:

- The team collectively manages project coordination and communication.
- Regular meetings to discuss progress, address issues, and make decisions together.
- Final decision-making is a collaborative effort among team members.

5. Testing and Quality Assurance:

- Waleed will do testing and quality assurance activities, including:
- Unit and integration testing.

- Identifying and reporting bugs and issues.

This collaborative approach allows all team members to have a holistic understanding of the project and contribute their skills and expertise to various aspects of development and documentation as needed. Effective communication and cooperation among team members will be key to the project's success.

1.7.3. Gantt Chart

1.8. Empathy Map

- **User Persona**: Online shoppers and sellers looking for a user-friendly e-commerce platform.

  Says:

  - "I want to find products easily."
  - "I need a secure platform for transactions."
  - "I like personalized product recommendations."
  - "I want to communicate with sellers and buyers in real-time."
  - "I value user reviews and ratings."

  Thinks:

  - "Will this platform have the items I'm looking for?"
  - "Is my payment information secure?"
  - "I appreciate when the platform understands my preferences."
  - "How can I trust the people I'm buying from?"
  - "I rely on reviews to make informed decisions."

  Feels:

  - Excited about discovering new products.
  - Anxious about online security.
  - Appreciates when the platform feels personalized.
  - Frustrated when communication is slow or difficult.
  - Confident when reading positive reviews.

  Does:

  - Searches for products using keywords.
  - Checks product details and seller ratings.
  - Interacts with the messaging system for inquiries.
  - Leaves reviews and ratings for products and sellers.
  - Shares great deals with friends.

  Pain Points:

  - Difficulty finding specific products.

- Concerns about payment security.
- Irrelevant product recommendations.
- Slow or unresponsive communication.
- Lack of trust in unknown sellers.

Gains:

- Quick and easy product discovery.
- Confidence in secure transactions.
- Personalized product recommendations.
- Seamless communication with others.
- Trustworthy and reliable sellers.

Chapter 2

Software Requirement Specifications

Chapter 2: Software Requirements Specifications

2.1. Introduction

2.1.1. Purpose

The software requirement specifications or simply SRS is initially the most important foundation of the life cycle of a software development. SRS defines the quality and determines caliber of the software development. SRS stage/phase clearly defines the requirements needed for the development of a life cycle of software product. The higher the quality of a SRS the finer the document it will. There will be lesser chance of mistakes and errors made during the life cycle of the product. Lesser error and mistakes ultimately lead to a superior and very finer outcome which is in this case is a software product known as Swift Buy Hub. The purpose of this SRS for Swift Buy Hub summarizes as to finalize an elegant and error free report free of any unusual mistakes and error bindings.

2.1.2. Document Conventions

Document Conventions are crucial part of a SRS documentation. Without the use of document conventions in SRS the document is to lose its integrity and the complete use of conventions. [1]The document conventions of Swift Buy Hub are as:

Convention For Heading and Sub Heading:

Heading and Sub Headings – Calibri 20, Calibri 16, Calibri 14

Convention For Body:

Text and Documentation – Calibri 12

Font Face – Calibri 12

Convention For Page Title

Title Face Font – Calibri 48 Position Right

[1] [2] “Usability Requirements of Web Software’s”, Available: https://www.sciencedirect.com/science/article/pii/S1110016814000568

2.1.3. Intended Audience and Reading Suggestions

The indented audience of a project is the core part of a project as it defines the stream where and how a project will be turned and for what people the project is made and how. The intended audience for Swift Buy Hub are as follows:

Seller – The person who will be using the platform to sell its products using the platform features like chatting with client etc.

Buyer – The person who will be using the platform to discover the required products and buying them using the platform features like smart inspection, chatting with seller, smart search etc.

2.1.4. Product Scope

1. Overview

Today Everyone likes to Buy and sell online because it is more convenient for them. So we worked on this but we selected a limited category of Cars and Laptops as it is a vast category in itself. We do different Research on the main problem which laptop and car buyers face is that they don't know the exact price of the product according to the specs.

2. Objectives

- Design a friendly e-commerce web platform
- Design a website with easy and smooth navigation
- Optimize website for buyer searches
- Integrate website with smart AI Search capability for buyers
- Integrate websites with smart inspection for products
- Integrate website with security and performance capabilities
- Keeping records of buyer and seller profiles in DB
- Integrate a chat system for buyer and seller
- Integrate website with suitable and secure payment system functionality.
- Create a system for buyers to rate the products and seller.

2.2. Overall Descriptions

2.2.1. Product Perspective

The product Swift Buy Hub is an ecommerce classified type website implemented with ai capabilities such as smart inspection of products, smart searches using NLP. And chat system for buyer and seller. The product Swift Buy Hub is in replacement of classified websites like OLX and Pakwheels buy with improved system functionalities for users easing up their experience.

2.2.2. User Classes and Characteristics

Swift Buy Hub is comprised of multiple class of users and user characteristics each with itw own role and importance needed for the building of a classified website.

1. **Buyer – The Customer**
    - **Frequency of use** – regular to frequent
    - **Technical Expertise** – novice to well informed
    - **Key Requirements** – products navigations, chat system, easy and smart navigations, smart inspection of products
2. **Seller**
    - **Frequency of use** – regular to frequent
    - **Technical Expertise** – novice to well informed
    - **Key Requirements** – products inspection, chat with clients etc
3. **Admin – The Developer**
    - **Frequency of use** – daily
    - **Technical Expertise** – moderate to highly skilled
    - **Key Requirements** – dashboard, user management, products managements, database etc.

2.2.3. Operating Environment

Swift Buy Hub is made up with smart operating capabilities which are easy to run on systems smoothly and without any hurdle

Hardware Platform:

- Server Infrastructure – Cloud Host
- Hardware Configuration – Dual Core Minimum and 4 GB of RAM at least

Software Platform:

- Database Server: MongoDB v4.2 or Higher

- Application Runtime – NodeJS v12+/npm
- Operating System – Windows 7 or higher, Android 7 or higher, IOS 8 or Higher

External Services:

- Smart Inspection of Products
- Free Number of Physical Inspection Trials

2.2.4. Design and Implementation Constraints

The product Swift Buy Hub must follow some design and implementation constraints as these are what shows us what limitations are needed.

- The project must be based on Angular Frontend
- The project must have backend of node.js with express.js or Django
- The project must have a MongoDB database
- The project interface must be easy to navigate
- The project must perform all the functionalities mentioned and Implemented
- The project features like smart inspection must work on all products
- The project feature like smart search must work for all products given the need
- The project design must be responsive for different types of devices and sizes
- The project must operate in legal terms and rules
- The project must be completed and running within the specified duration of time

2.2.5. Assumptions and Dependencies

Assumptions:

- Users must know how to operate a website
- Users must have an internet connections
- Users must know the type of product they needed
- Availability of hosting connection
- Good performance of website
- Working of search feature accurately

- Working of smart inspection correctly

Dependencies:

- Successful connection between website and hosting platform
- Seller to upload its product for selling
- Buyer to buy products
- Dependency of successful database connections

2.3. External Interface Requirements

2.3.1. User Interface

The website of Swift Buy Hub will be responsive and easy to navigate for users giving a smooth and excellent web experience. The website will show all the products listed by the buyers with its respective details and data. The smart search feature will be operational for all users including the buyer and the seller.

2.3.2. Hardware Interface

The hardware needed for Swift Buy Hub to be operational includes a mobile phone with internet connection, a computer system with a touch screen, mouse or some input device. All needed to be connected to n internet connection to establish link

2.3.3. Software Interface

Android or IOS for mobile or tablet devices. Windows, Linux or some other OS for computers

2.3.4 Communication Interface

HTTP/HTTPS for communications via the internet along with the JSON/XML file for data formats.

2.4. System Features

System features defines the capabilities and scope of how and what your website should be able to do

2.4.1 Smart Inspection

Smart inspection is one of the core features of Swift Buy Hub. It is one of the extreme points that defines the uniqueness of our web project from other competitors. The product is compared against list of data from the data model of the respective product either a car or mobile or laptop. It compares the features and use and gives a relatively market honest price.

2.4.2. Smart Search

Allows the user to freely search out its need. The user can explain its need of product as in like typing or chatting casually. The NLP feature filters out the required keywords and brings out a list of products that matches the client`s descriptive need

2.4.3. Chat Box

Swift Buy Hub understands the very need of its users and allows a chatbox between the buyer and seller. The buyer and seller can freely discuss the product in talks and can easily share information between them

2.5 Nonfunctional Requirements

2.5.1 Performance Requirements

Swift Buy Hub has excellent project features extending the user experience,

- Homepage loads under 5 sec
- Pages are loaded under 3 sec
- Products are loaded under 2 sec
- Products are added under 3 sec
- Chat are send under 1 sec
- Searches are done in under 2 sec
- Searches results are shown under 4 sec
- Inspection results are loaded in under 4 sec

2.5.2. Safety Requirements

Authentications of users are handled carefully. In case of wrong login and password the web page does not allow the user to login. Also the data of users are completely secured and stored.

2.5.3. Security Requirements

The data is encrypted using HTTPS/SSL Protocols keeping the users data safe

2.5.4. Usability Requirements

- Smart navigations
- Fast and smooth user interface
- Responsive UI
- Easy access to data
- Fast Searches
- Fast loading time
- Easy management of products
- Excellent Feedback system

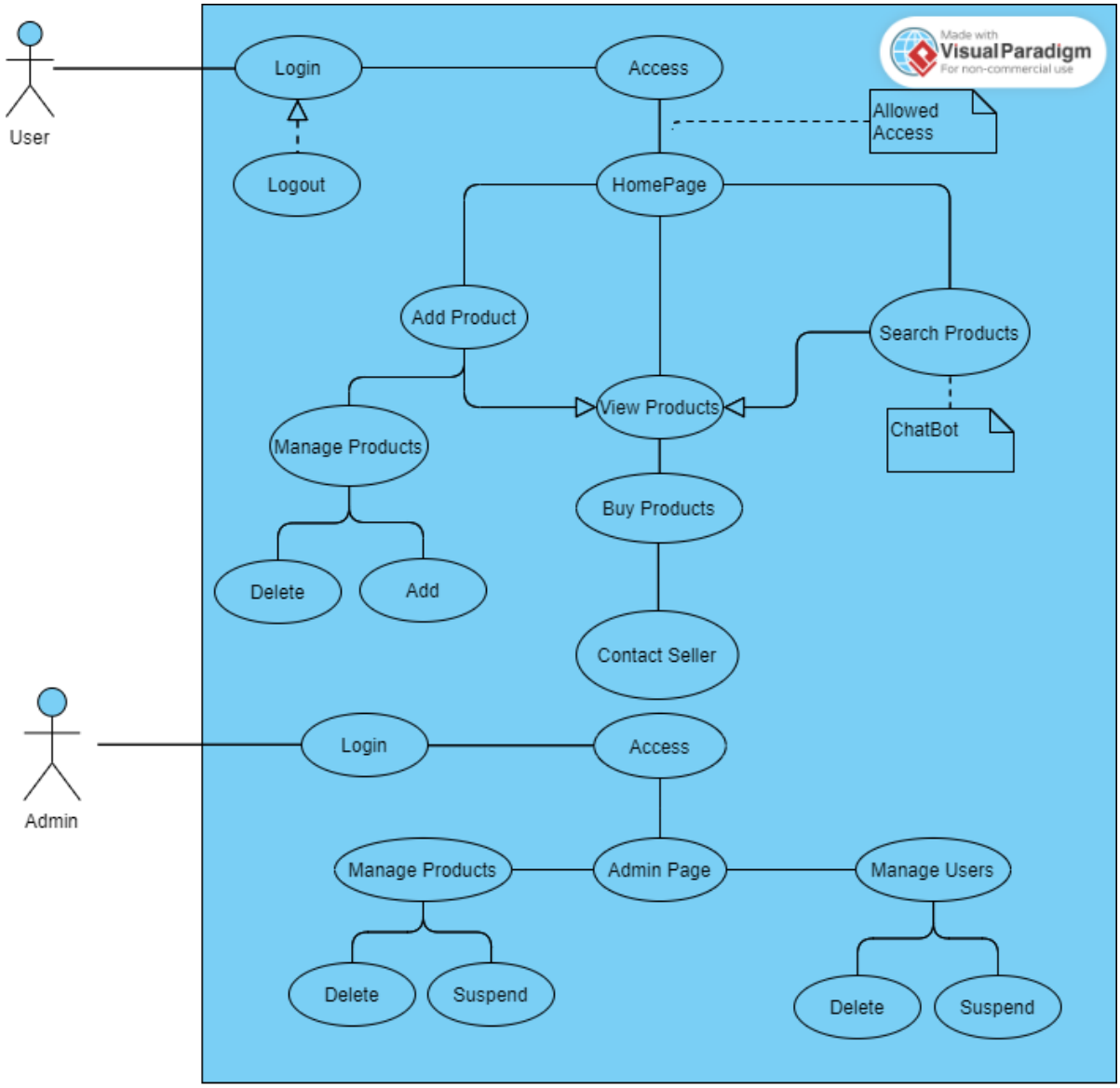
Chapter 3

Use Case Analysis

Chapter 3: Use Case Analysis

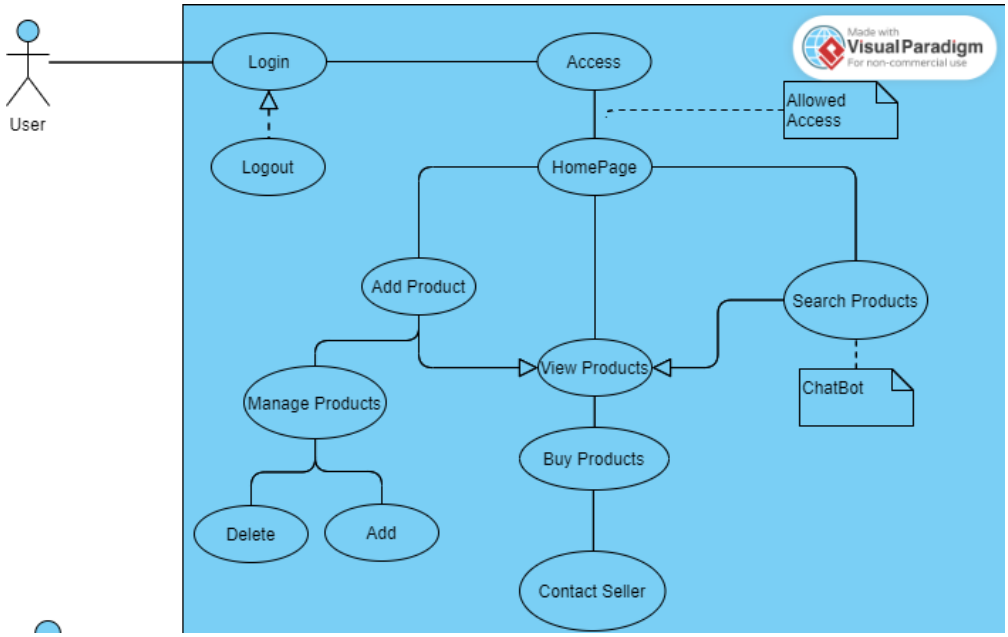
It involves comprehending user interactions, identifying use cases, and detailing their descriptions. The process includes actor identification, prioritizing use cases based on business importance, managing relationships between use cases, and validating their alignment with system requirements. This iterative refinement ensures that use cases evolve in response to feedback and changing requirements. The documentation of this analysis, comprising diagrams and descriptions, serves as a foundational reference for stakeholders throughout the software development life cycle.

3.1. Use Case Model

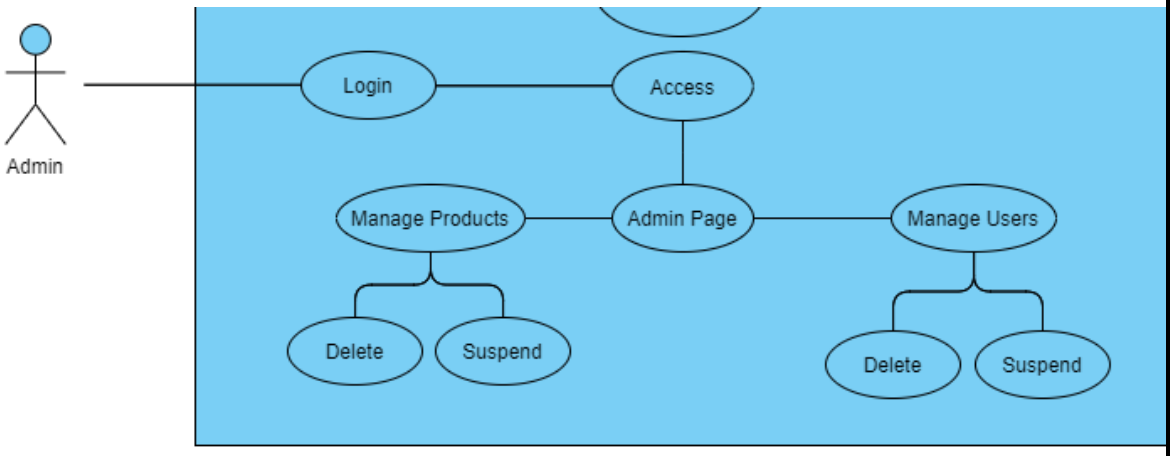


3.1. Use Case Model

User Panel:



Admin Panel:



3.2. Use Case Descriptions

The Login Use Case Description:

Use Case Name	Login	
Scope	Login For Manage Products	
Primary Actor(s)	Client	
Stakeholders and Interests	Client: Client uses it to add or delete products	
Pre-Conditions	Login Must	
Post-Conditions	After Login Client Must Do Its work	
Main Scenario	➢ Client Open Website ➢ Client can view, add or remove products	
Failure Case/ Alternative case	Failure Failure due to internet connection lost or server error	Alternative Case Try again later when service is available
Frequency Of Occurrence	Client can view, add, or delete products infinite times	

The View Products Use Case Description:

Use Case Name	View Products	
Scope	View For Products	
Primary Actor(s)	Manager, Client & Customers	
Stakeholders and Interests	Client: Client can view products as many times Manager: manager can view products as many times Customer: Customer can view products as many times	
Pre-Conditions	Login Must	
Post-Conditions	After Login user Must Do Its work	
Main Scenario	➢ User Open Website ➢ Client can view as many products as possible	
Failure Case/ Alternative case	Failure Failure due to internet connection lost or server error	Alternative Case Try again later when service is available
Frequency Of Occurrence	Users can view products infinite times	

The Buy Products Use Case Description:

Use Case Name	Buy Products	
Scope	Buy Products	
Primary Actor(s)	Customers	
Stakeholders and Interests	Customer: Customer can view products and buy as many products	
Pre-Conditions	Login Must	
Post-Conditions	After Login Customer Must select product	
Main Scenario	➢ User Open Website ➢ Customer can view as many products as possible	
Failure Case/ Alternative case	Failure Failure due to internet connection lost or server error	Alternative Case Try again later when service is available
Frequency Of Occurrence	Customer can buy as many products as possible	

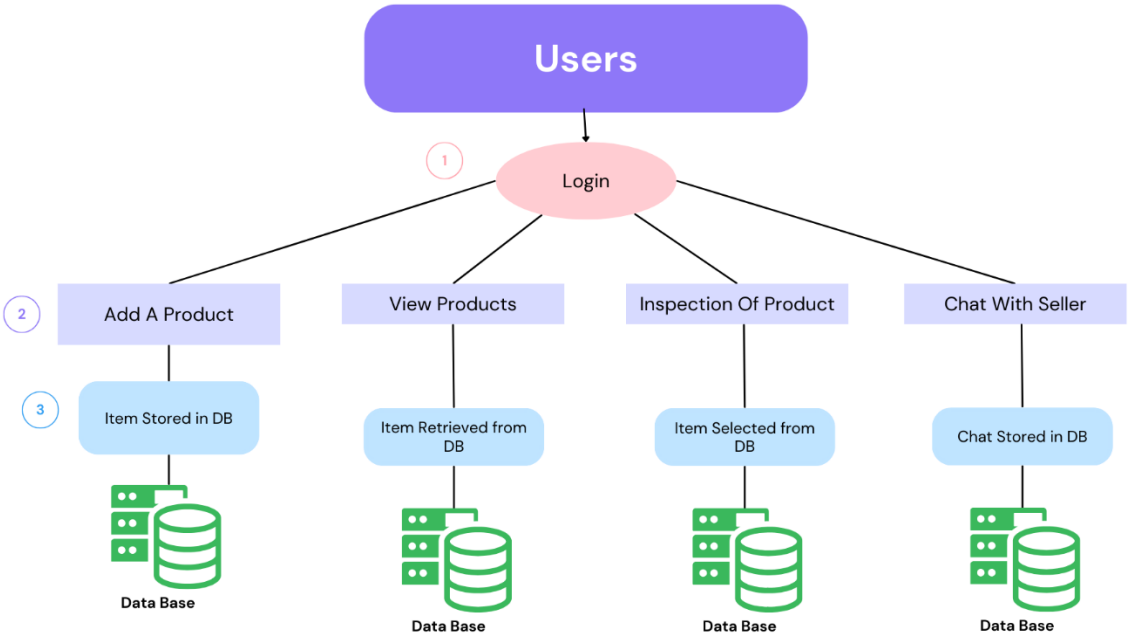
Chapter 4

System Design

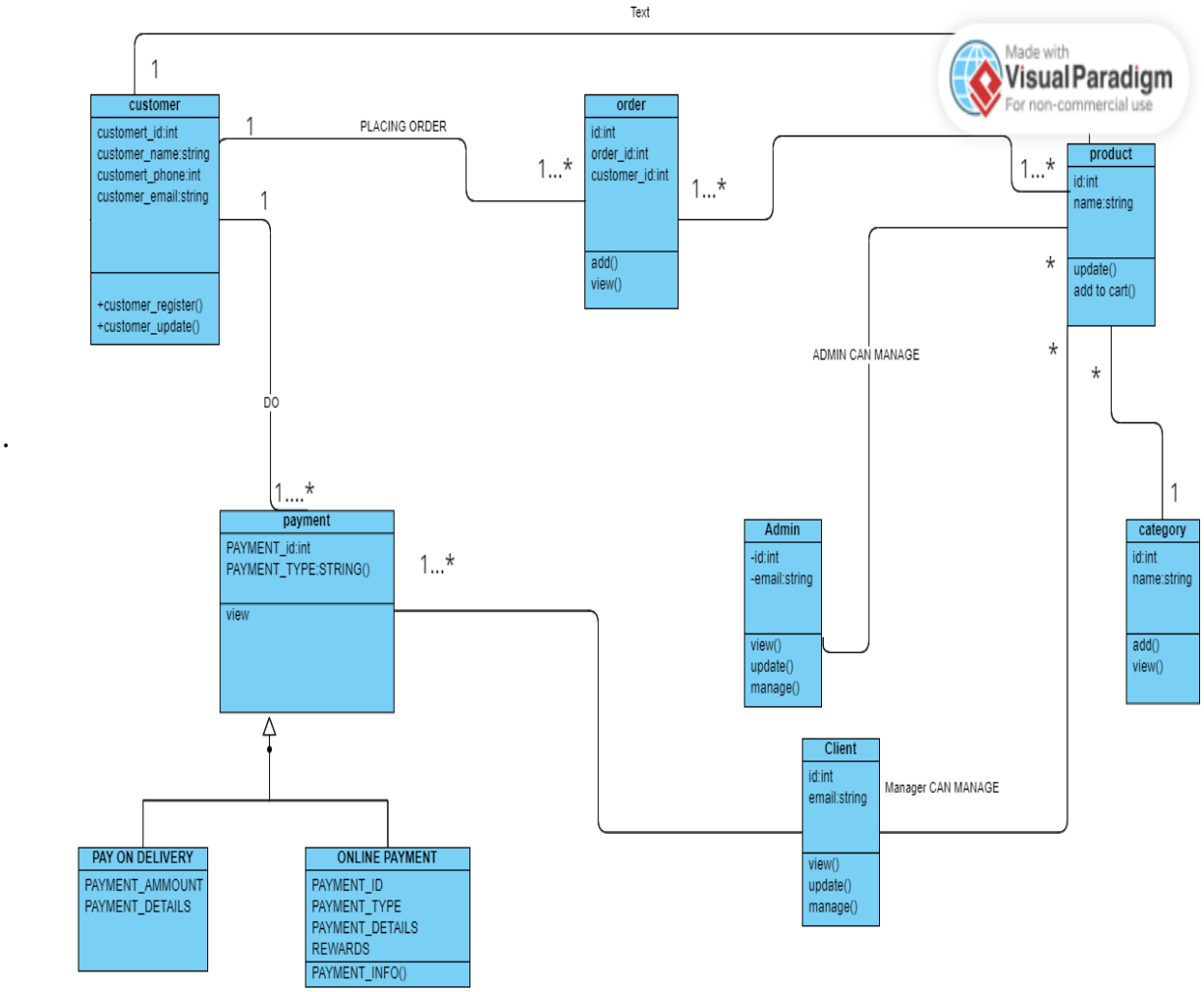
Chapter 4: System Design

The design should prioritize scalability, ensuring efficient handling of a growing product database and increasing user interactions. An architectural diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. First, we login to the system then perform POS tasks. Data store in database MySQL.

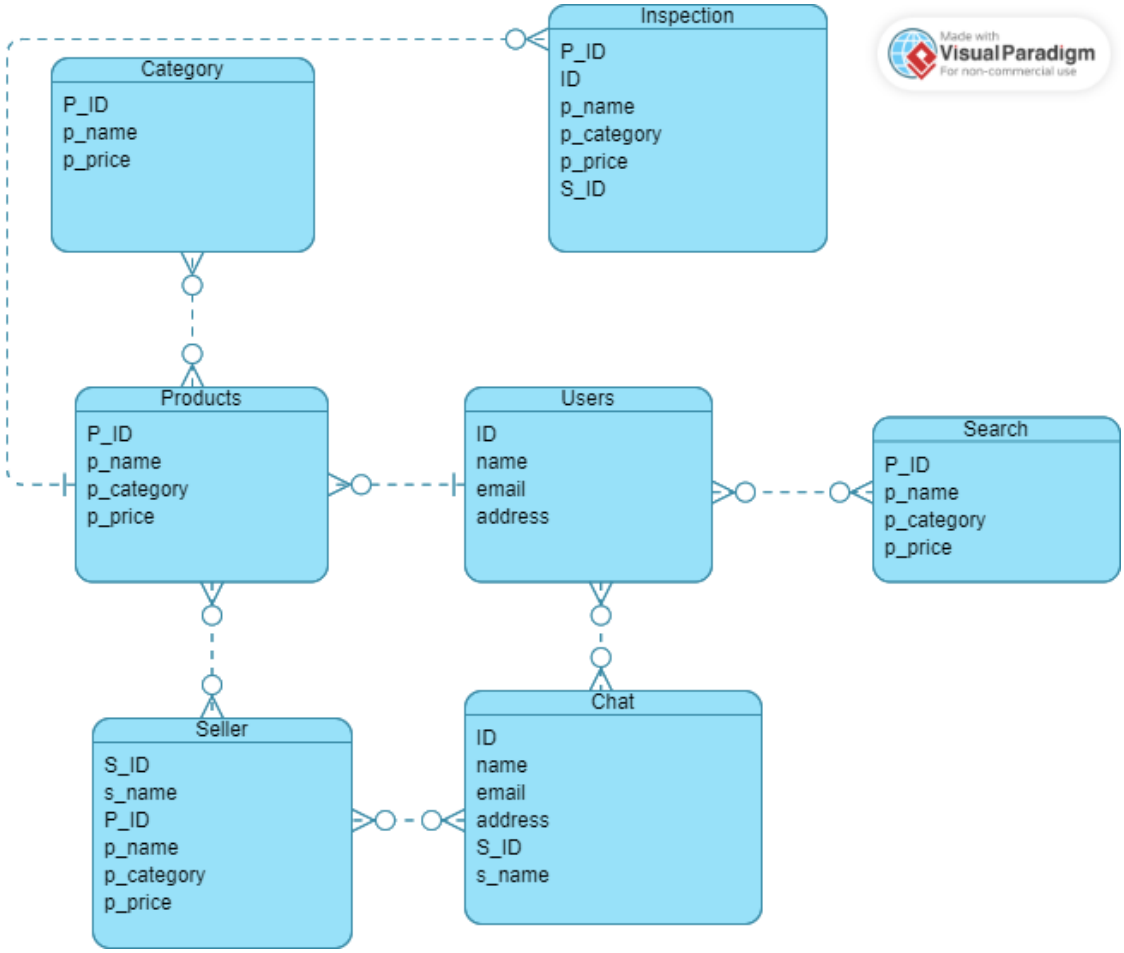
4.1. Architecture Diagram



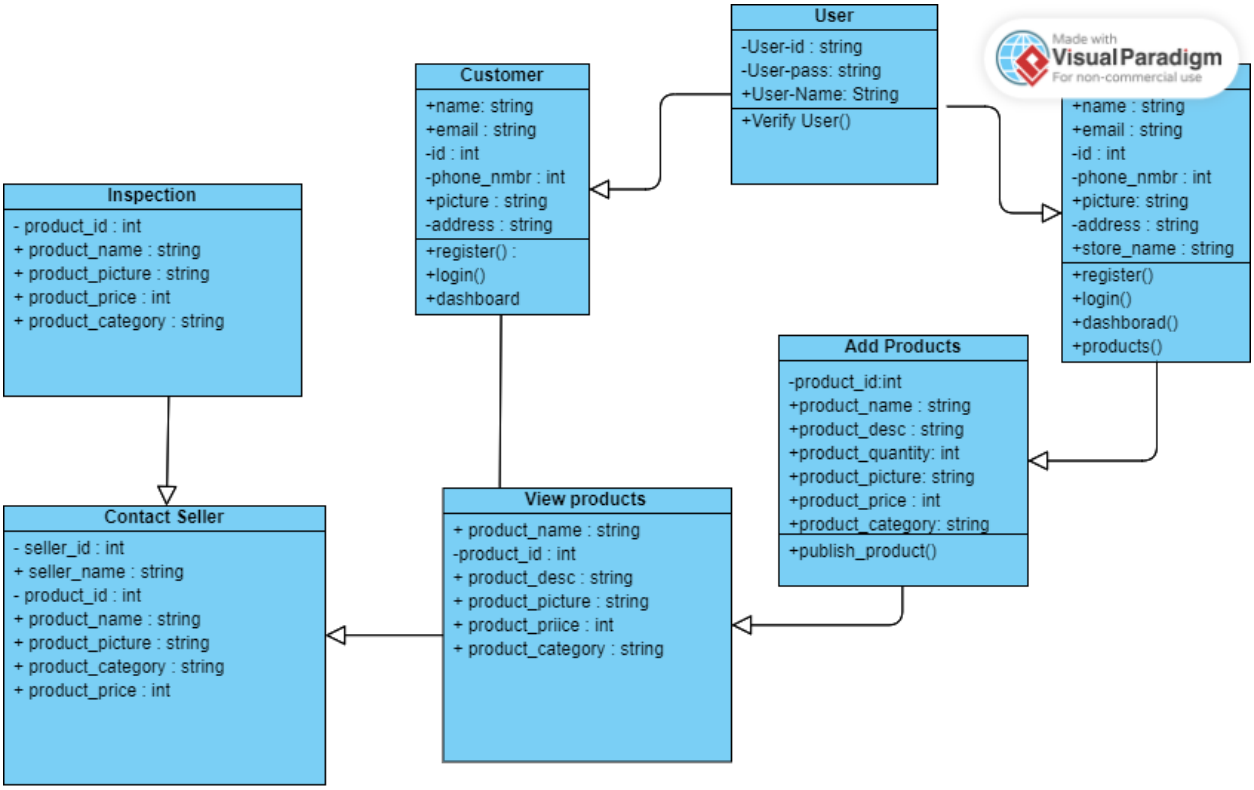
4.2. Domain Model



4.3. Entity Relationship Diagram with data dictionary

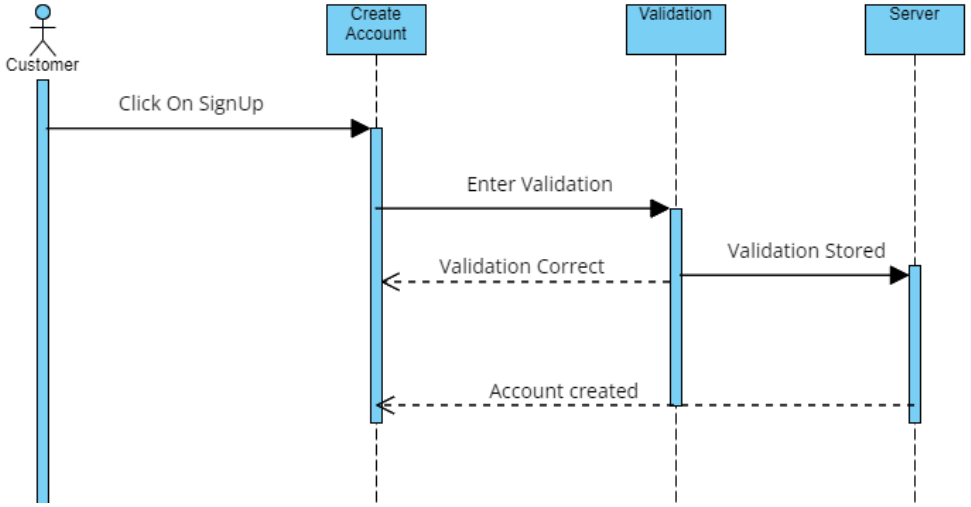


4.4. Class Diagram

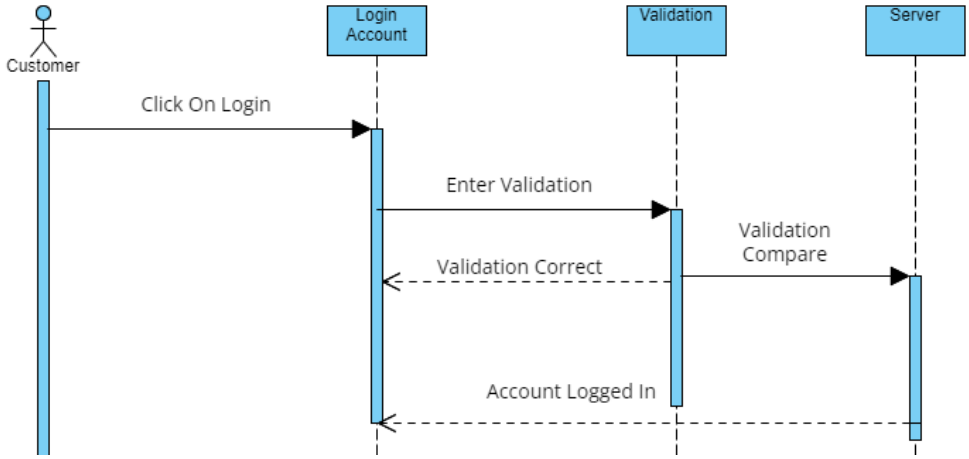


4.5. Sequence / Collaboration Diagram

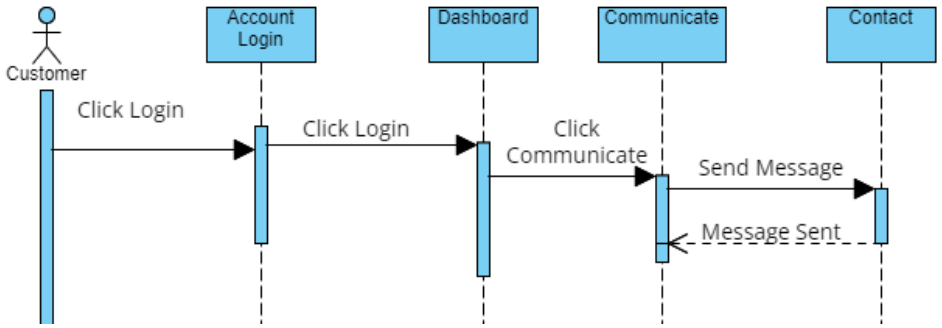
1 User Sign-Up:



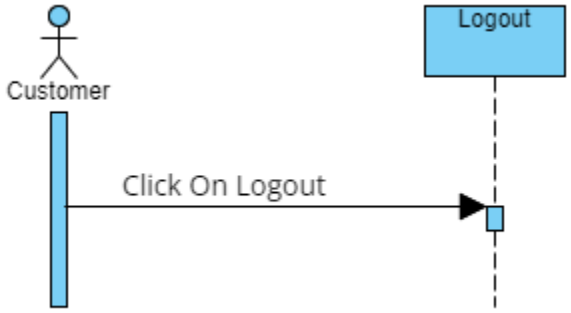
2) User Sign-In:



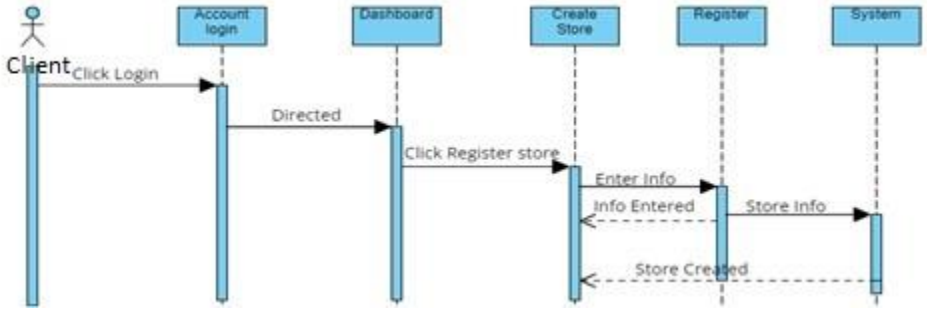
3) Communicate With Seller:



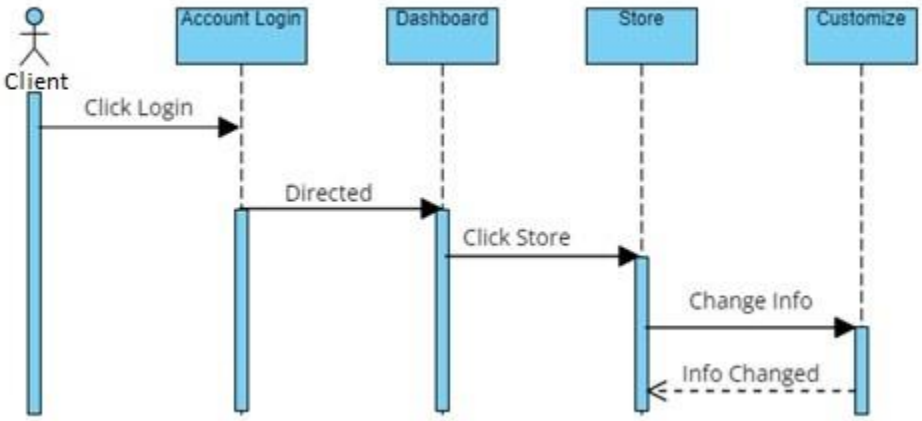
4) Logout:



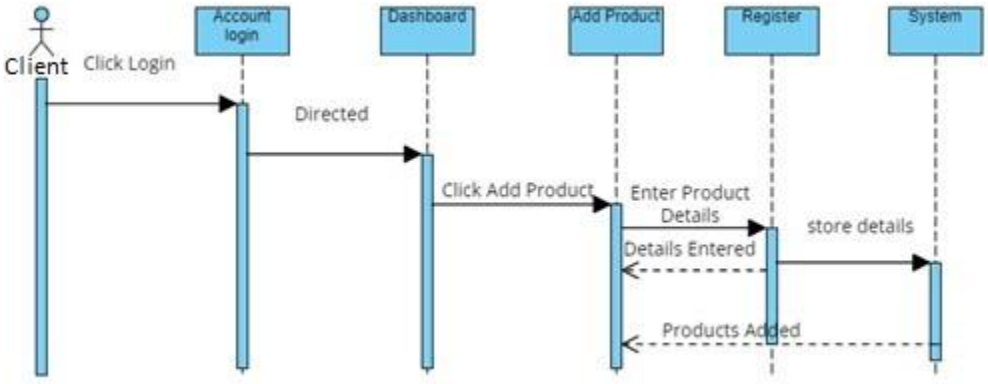
5) User Create Store:



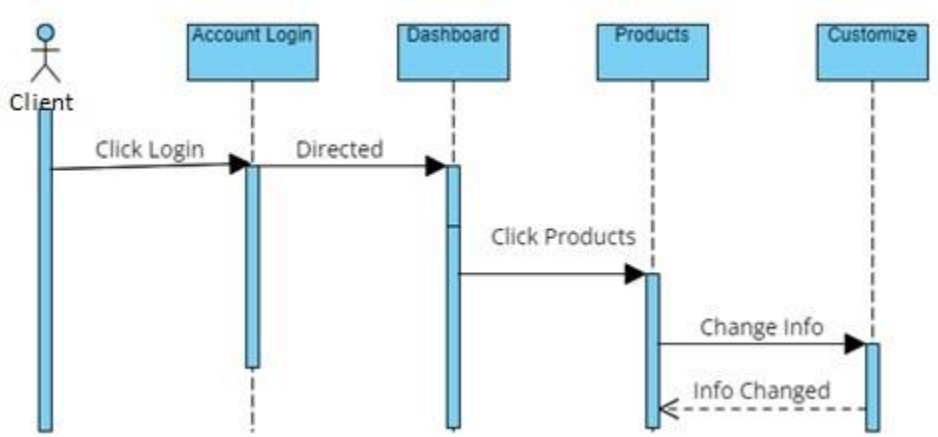
6) User Change Store Info:



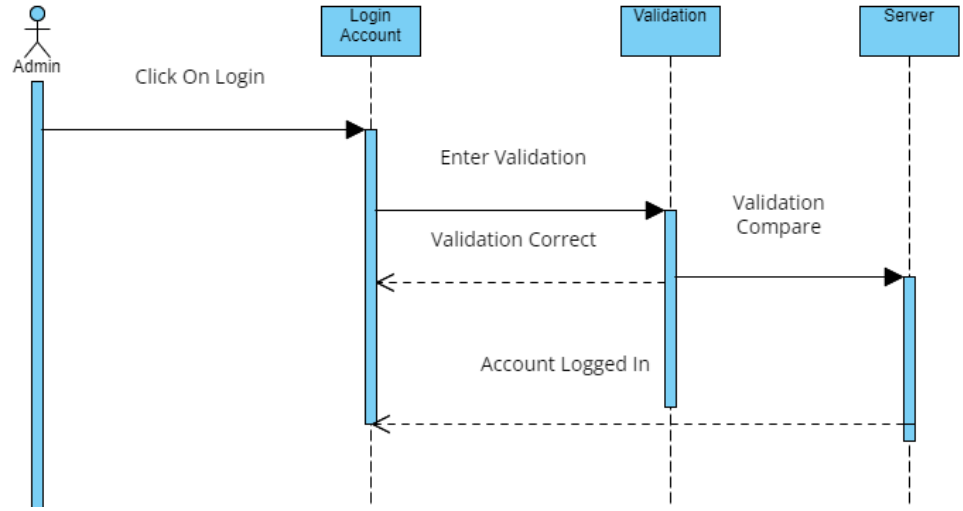
7) User Add Product:



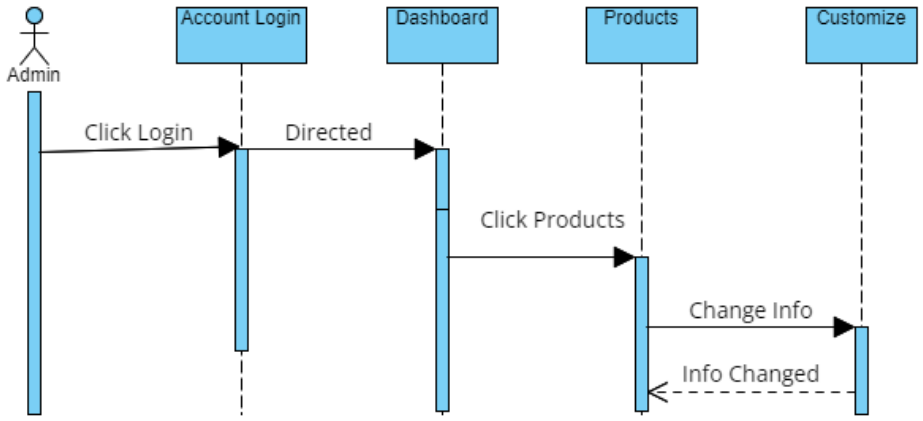
8) User Change Product:



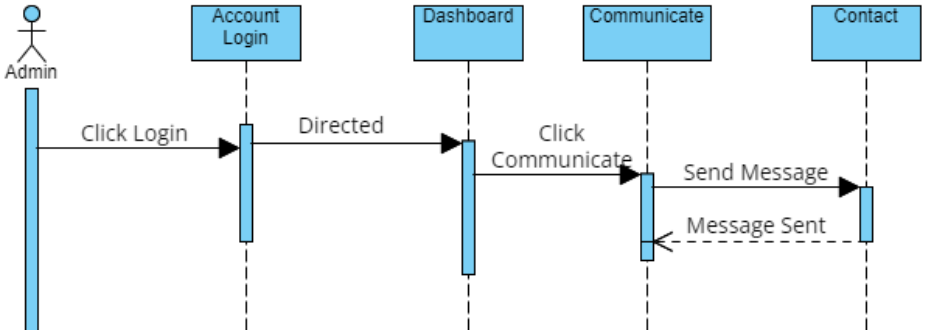
9) Admin Sign in:



10) Admin change product



11) Admin communicate:



4.6. Operation contracts

Contract : Select Product

Operation: Select Product(Product ID, Product Name, Product Quantity)

Cross Reference : Use Case: Checkout

Pre-Conditions : Products are already entered in the app

Post-Conditions:

- Product instance pro was created
- Pro was associated with selected products
- Pro.quantity was set to quantity
- Pro was associated with the specifications based on ID, Name and quantity

Contract : Place Order

Operation: Place Order(Product_Name, Product_Quantity)

Cross Reference : Use Case: Checkout

Pre-Conditions : Products are already selected and entered in cart

Post-Conditions:

- An Order instance ord was created
- ord was associated with current order placed
- Attributes of ord were initialized

Contract : End Sale

Operation: End Sale()

Cross Reference : Use Case: Checkout

Pre-Conditions : Products orders are already entered and in process

Post-Conditions:

- Sale was confirmed
- Sale wascompleted

Contract : Make Payment

Operation: Make Payment(Product_Name, Product_Quantity)

Cross Reference : Use Case: Checkout

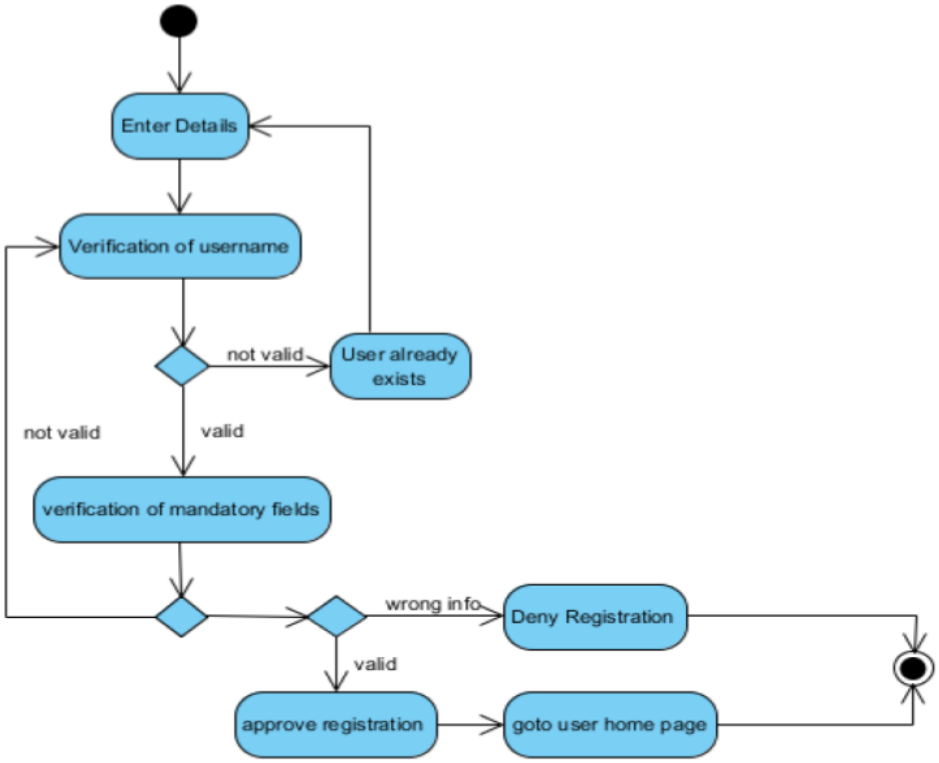
Pre-Conditions : Products orders are already entered and in process

Post-Conditions:

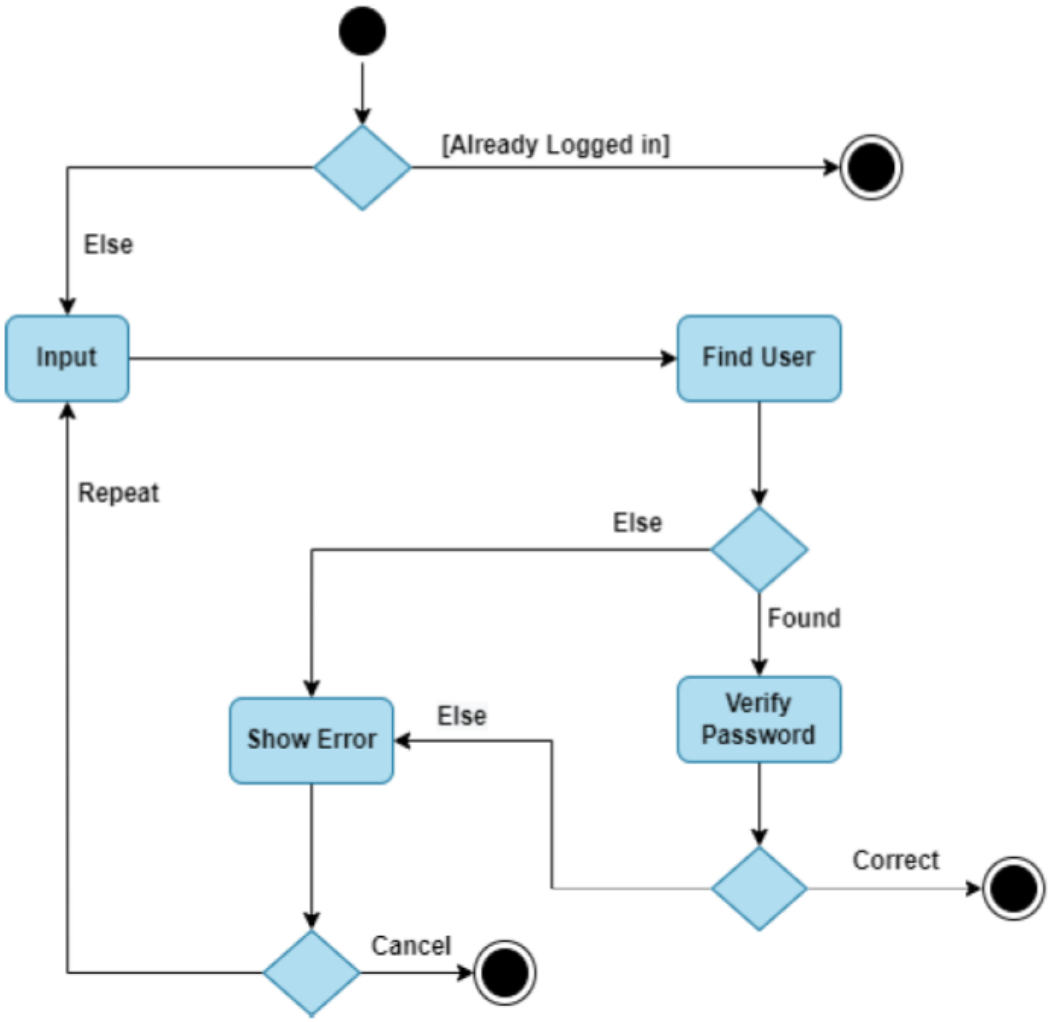
- A payment instance pay was created
- Pay.amount was set to amount
- Pay was associated with current sale

4.7. Activity Diagram

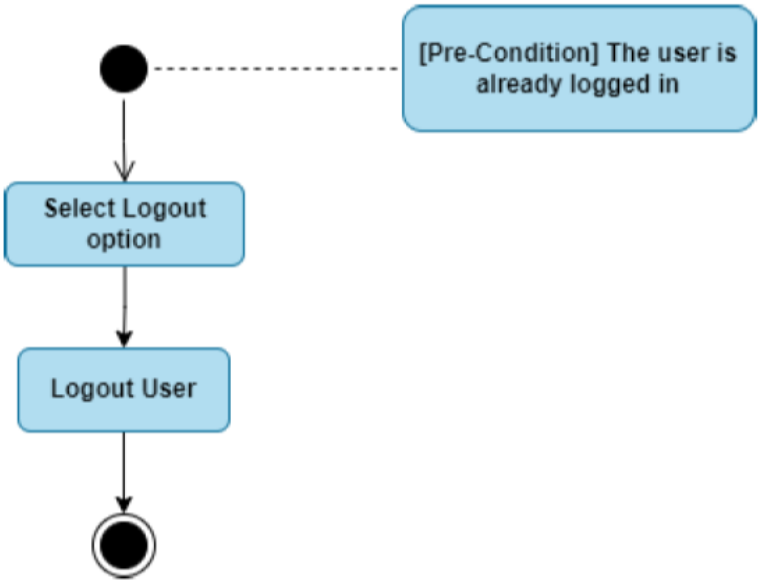
User – SignUp:



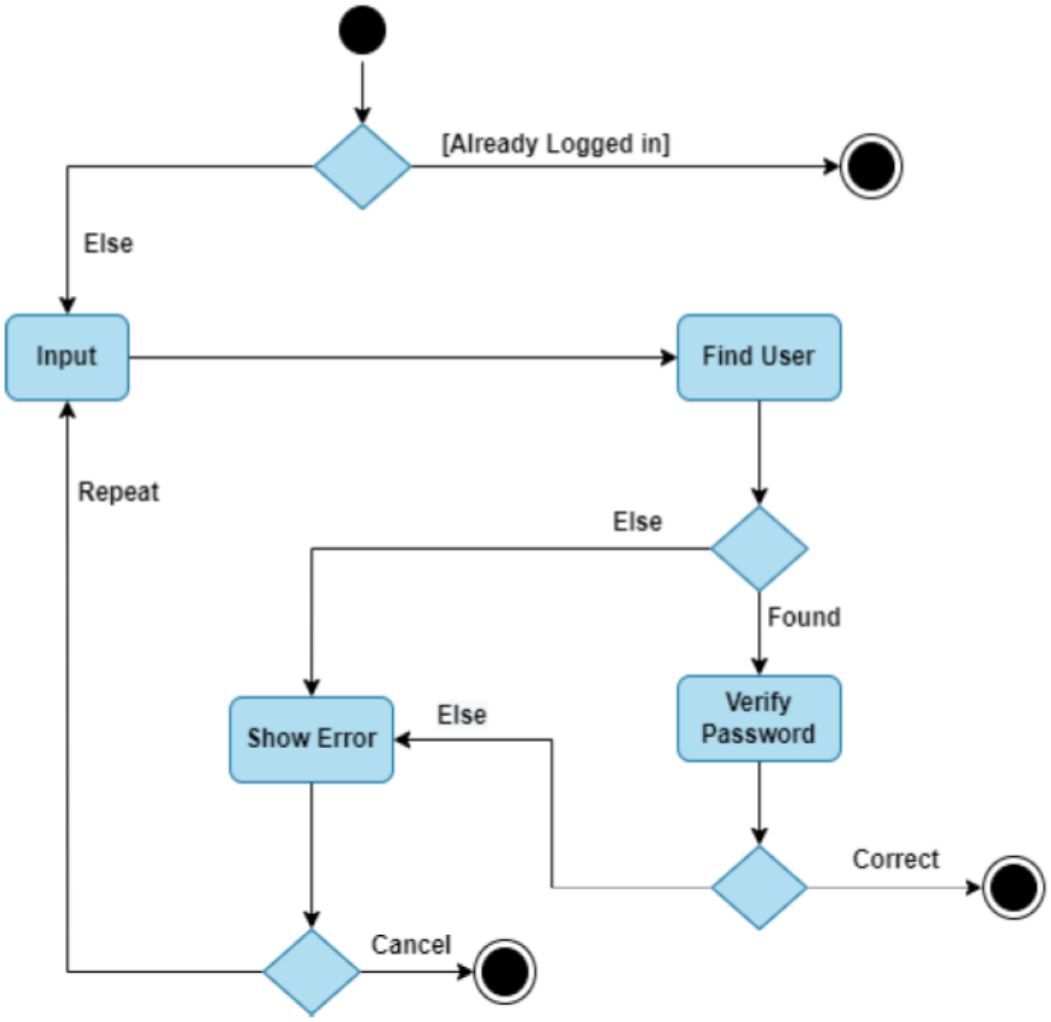
User – Login:


[Already Logged in]
Else
Input
Find User
Repeat
Else
Found
Show Error
Else
Verify Password
Correct
Cancel

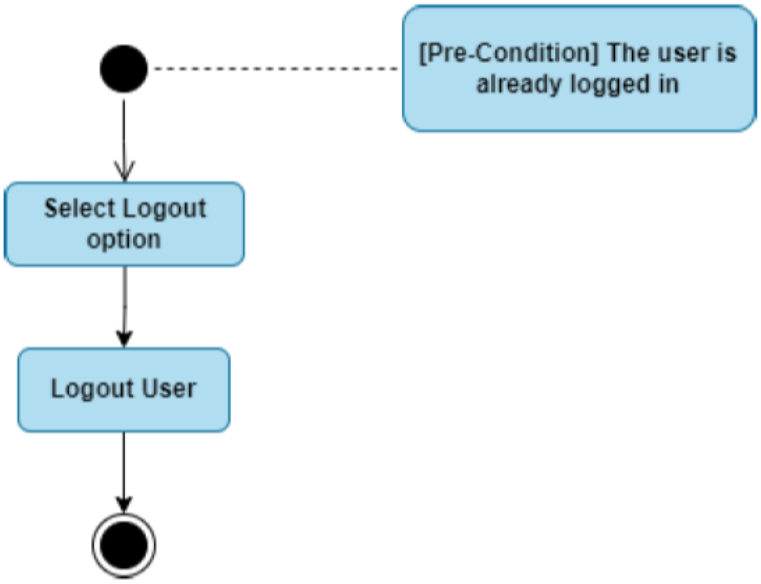
User – Logout:


[Pre-Condition] The user is already logged in
Select Logout option
Logout User

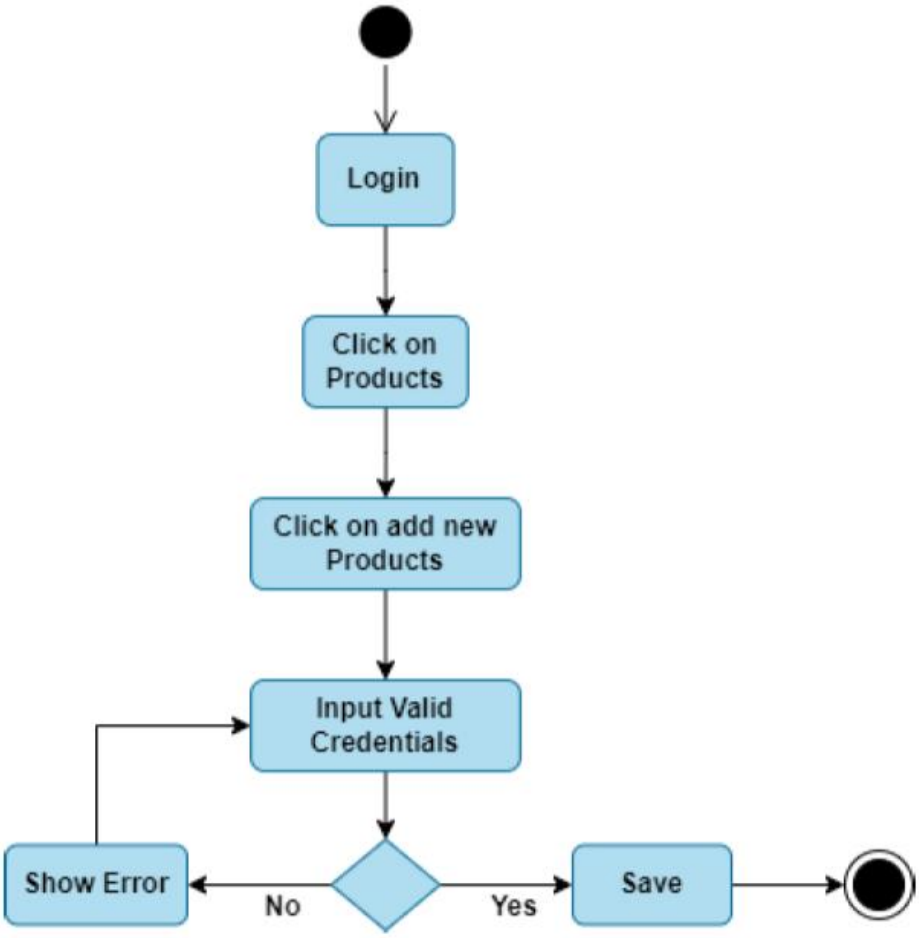
User(seller) – Login:



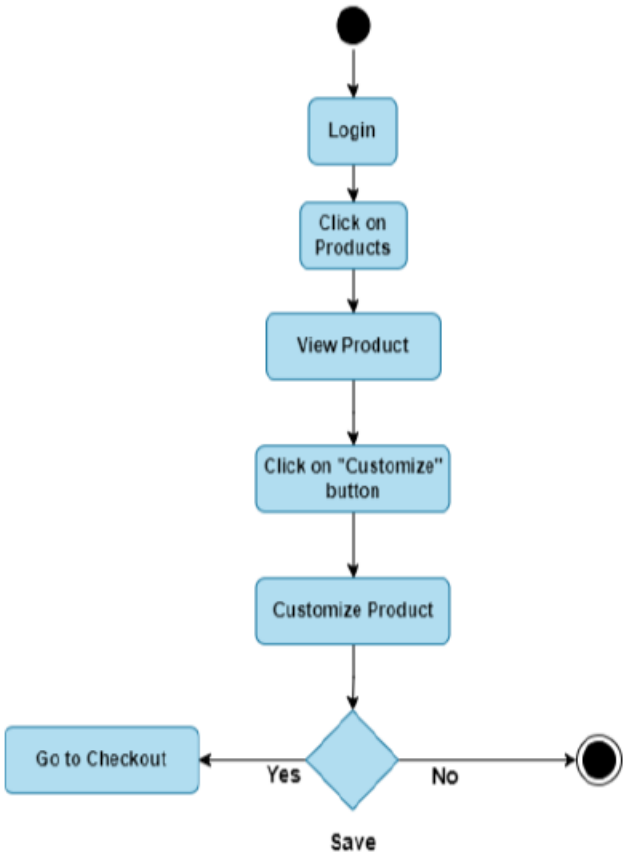
User(Seller) – Logout:



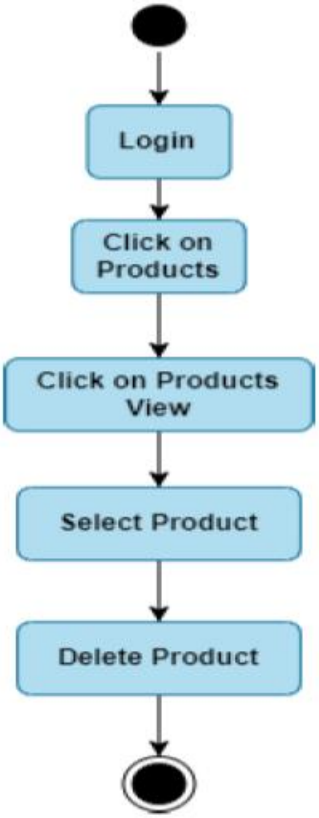
User – New Products:



User – Customize Products:

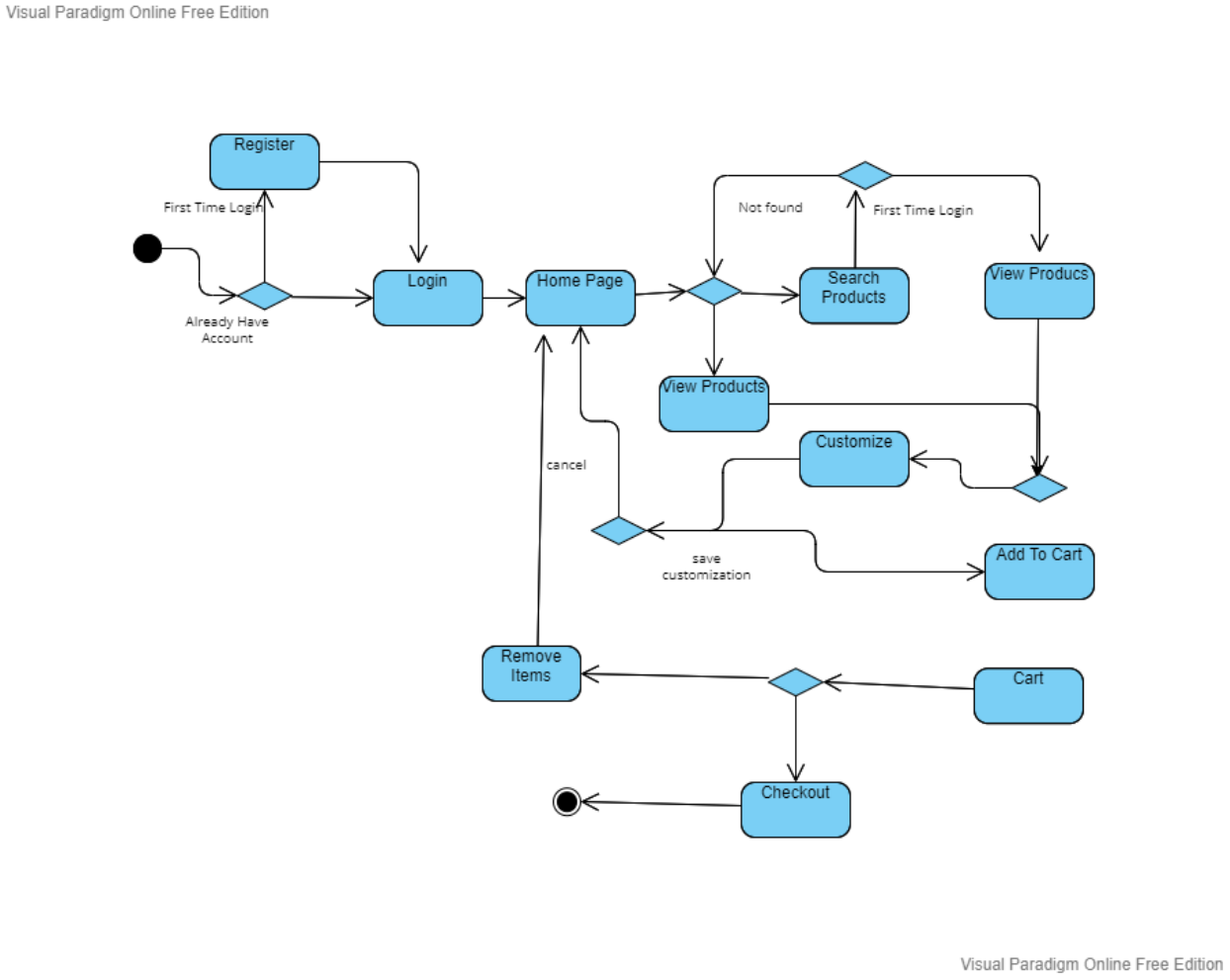


Client – Delete Products:

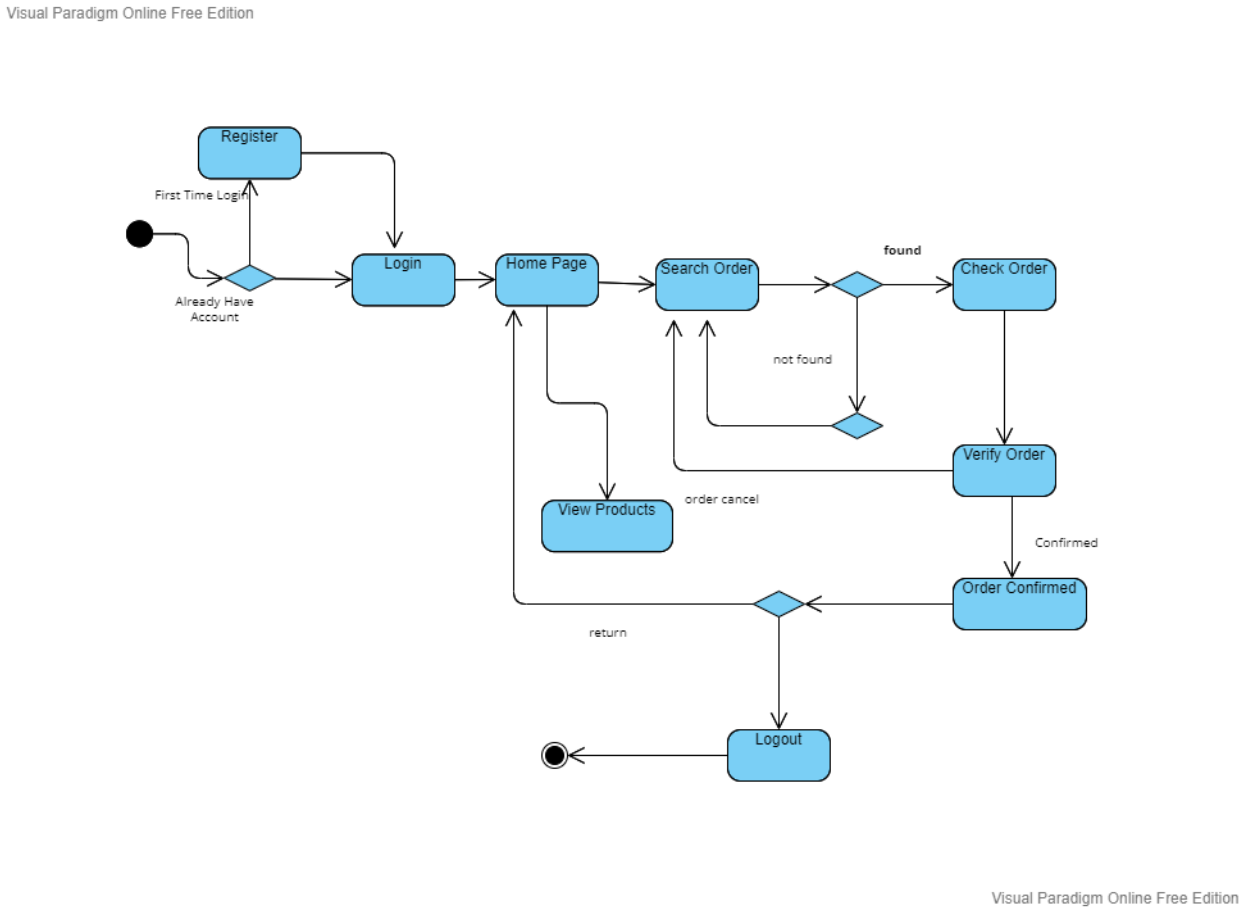


4.8. State Transition Diagram

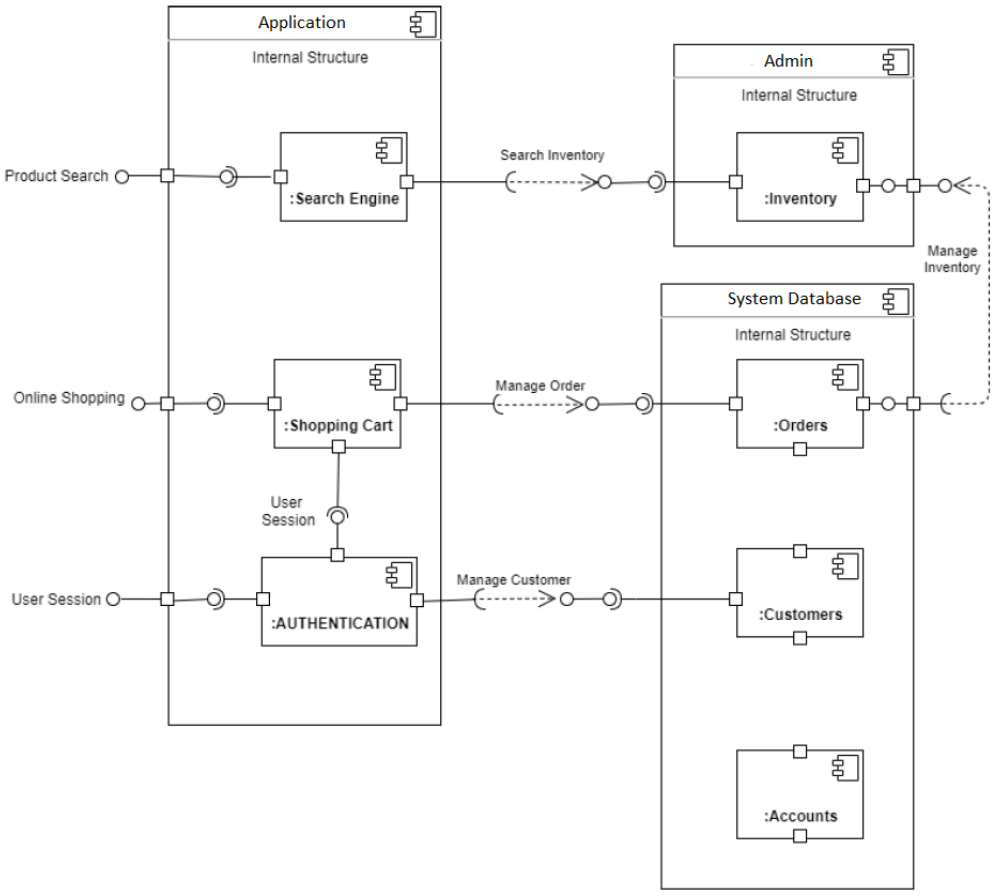
State Diagram User



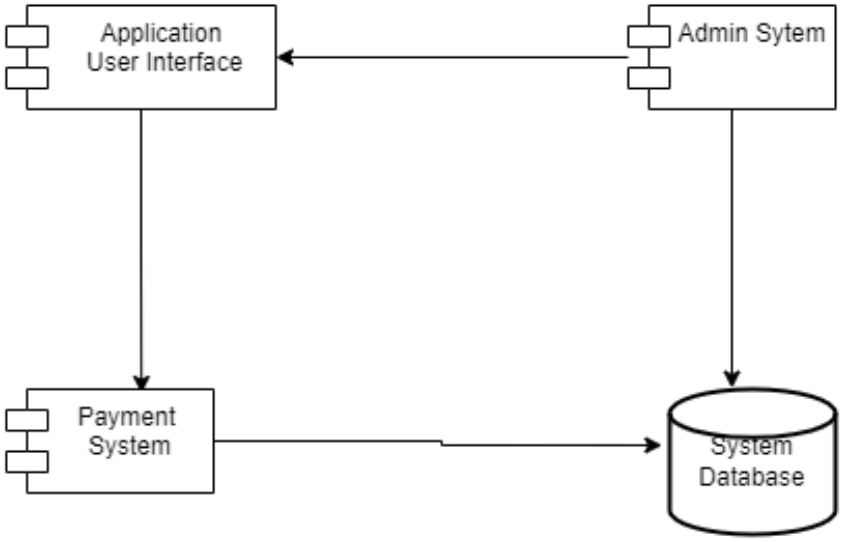
State Diagram:



4.9. Component Diagram



4.10. Deployment Diagram



Chapter 5

Implementation

Chapter 5: Implementation

This chapter covers the key technical aspects of implementing the Swift Buy Hub e-commerce platform. It delves into the important flow control logic, the components and libraries utilized, the deployment environment, the tools and techniques employed, the coding standards followed, and the version control practices adopted.

5.1. Important Flow Control/Pseudo codes

1. **Start**

2. Sign Up an account and log in:

- Function registerUser(userType):

- Input: userType (buyer/seller), registrationDetails
- If userType == "buyer":
  - Save buyer details in buyer database
- Else if userType == "seller":
  - Save seller details in seller database
- Output: confirmation message

- Function loginUser(credentials):

- Input: username, password
- Authenticate user
- Output: login status (success/failure)

3. Smart NLP-Based Buyer Searches:

- Function searchProducts(searchTerms):
  - Input: searchTerms
  - Use NLP to process search terms

- Output: list of matching products

4. Product Display List:

- Function displayProducts(productList):
  - Input: productList
  - Display products to user
  - Output: user selection

5. The User Chooses the Products:

- Function chooseProducts(selectedItems):
  - Input: selectedItems
  - Add selected items to user's cart
  - Output: updated cart

6. Start Chatting with the Seller:

- Function startChat(sellerId):
  - Input: sellerId, message
  - Initialize chat session with seller
  - Output: chat interface

7. Examining AI Models:

- Function compareWithAIModel(selectedItems):
  - Input: selectedItems
  - Use AI to analyze and compare items
  - Output: estimated cost and comparison details

8. Bid-Seller Discussion:

- Function bidDiscussion(bidDetails):
  - Input: bidDetails (price, terms)
  - Facilitate negotiation between buyer and seller
  - Output: agreed terms or updated bidDetails

9. Order Confirmation for Product:

- Function confirmOrder(cartDetails):

- Input: cartDetails
- Verify purchase details
- Process payment
- Output: order confirmation

10. **End**

5.2. Components, Libraries, Web Services and Stubs

The Swift Buy Hub platform leverages a variety of components, libraries, and web services to achieve its functionality:

- Frontend: Angular framework, RxJS, NgRx for state management
- Backend: Node.js, Express.js, MongoDB for database
- AI/ML: Python, Scikit-learn for price prediction models
- Chatbot: Dialogflow for natural language processing
- Payment Integration: PayPal and Stripe APIs
- Messaging: Twilio for SMS notifications

Additionally, stubs were created for mocking external dependencies during development and testing.

5.3. Deployment Environment

The Swift Buy Hub application is deployed on the AWS cloud platform. The specific infrastructure includes:

- EC2 Instances: For hosting the Node.js application servers
- MongoDB Atlas: Managed MongoDB database service
- AWS Lambda: Serverless functions for price prediction models
- API Gateway: Managed service for REST API endpoints
- CloudFront: Content delivery network for static assets
- Route 53: Domain name management and DNS

This cloud-native architecture ensures scalability, reliability, and ease of maintenance for the ecommerce platform.

5.4. Tools and Techniques

The development team utilized a range of tools and techniques to streamline the implementation process:

- Agile Methodology: Iterative sprints with daily standups and retrospectives
- Integrated Development Environment (IDE): Visual Studio Code with relevant plugins
- Version Control: Git for source control, GitHub for remote repositories
- Continuous Integration: GitHub Actions for automated build and testing
- Containerization: Docker for packaging and deploying application components
- Infrastructure as Code: Terraform for provisioning cloud resources

These tools and practices enabled efficient collaboration, consistent code quality, and reliable deployments.

5.5. Best Practices / Coding Standards

The development team adhered to the following best practices and coding standards:

- Clean Code: Followed principles of readability, maintainability, and modularity
- SOLID Principles: Designed the system with a focus on single responsibility, open-closed, Liskov substitution, interface segregation, and dependency inversion
- Design Patterns: Leveraged patterns like Factory, Singleton, Observer, and Decorator to address specific design challenges
- Security Guidelines: Implemented OWASP Top 10 security measures, such as input validation, encryption, and role-based access control
- Testing Practices: Wrote comprehensive unit, integration, and end-to-end tests to ensure code reliability

These best practices ensured the codebase was well-structured, secure, and easy to extend and maintain.

5.6. Version Control

The Swift Buy Hub project utilized Git for version control, with the following key practices:

- Repository Structure: Separate repositories for frontend, backend, and infrastructure code
- Branching Strategy: Followed the Git Flow model with distinct branches for features, releases, and hotfixes
- Commit Guidelines: Enforced descriptive and atomic commit messages following conventional commit standards
- Code Reviews: Mandatory peer reviews for all pull requests before merging to main branches
- CI/CD Integration: Automated build, test, and deployment pipelines triggered on push and pull request event

Chapter 6

Testing and Evaluation

Chapter 6: Testing and Evaluation

6.1. Use Case Testing

This phase helps us test the entire system before deploying the final product. The whole system includes 3 actors the admin, customer and client. All these users will perform the actions like sign up, sign in, product search, product display, chatting, payment, invoices slip.

User Sign Up Testing Case

Main Success Scenario	Step	Description
Login A: Actor S: System	1.	**A: User Enter New User & Password**
	2.	**S: Registers User Name & Password**
	3.	**S: Validate Account Access**
	4.	**S: Allow Account Access**
Extensions	2a	**Password Not Valid** S: Display Message and Ask For Re-Entry 5 Times
	2b	**Password Not Valid 5 Times** S: Login In Timeout

User Sign In Testing Case

Main Success Scenario	Step	Description
Login A: Actor S: System	1.	**A: User Enter User & Password**
	2.	**S: Validate User Name & Password**
	3.	**S: Allow Account Access**
Extensions	2a	**Password Not Valid** S: Display Message and Ask For Re-Entry 5 Times
	2b	**Password Not Valid 5 Times** S: Login In Timeout

Employer Store Creation Testing Case

Main Success Scenario	Step	Description
Employer Store Creation A: Actor S: System	1.	**A: Enter New Store Details**
	2.	**S: Registers New Store Details**
	3.	**S: Allow To Post Products**
Extensions	2a	**Store Does Not Create** S: Display Message and Ask For To Re-Enter Details
	3a	**Products Not Publish** S: Check Product`s Details And Publish

Product Search Testing Case

Main Success Scenario	Step	Description
Product Search A: Actor S: System	1.	**A: User Enter Product Details**
	2.	**S: Validate Product Details**
	3.	**S: Product Details Searches**
Extensions	2a	**Product Detail Not Valid** S: Product Not Found. Search Other Products
	2b	**Password Data Matched** S: Shows Product Details

Products Buy Testing Case

Main Success Scenario	Step	Description
Products Buy C: Customer E: Employer S: System	1.	**C: Selects Product**
	2.	**C: Buy The Product**
	3.	**S: Adds Buy-In Product To Processing**
	4.	**E: Confirms Products Order**
Extensions	2a	**Product Buy-In Error**

		S: Display Error Message and Ask For Re-Order
	3a	**Product Not Issued** S: Asks The Employer To Re-Confirm Order

6.1. Equivalence Partitioning

6.1.1. Testing the functionalities of users roles and their authority level like Administrator, Employer & Users to check if all their assigned roles and functioning are in order.
6.1.2. Testing for core features of system such as Advanced AI based product searching and product ordering to ensure the main functionality of the system whether it is working properly or not.
6.1.3. Testing for secure transactions to ensure the whole transaction process is secured and well-functioning as intended.
6.1.4. Testing for Stack functionality of NodeJS, MongoDB, and Angular regarding backend and interface aspects implemented
6.1.5. Testing for application's performance under different conditions, including peak loads and varying network speeds.

6.2. Boundary Value Analysis

Testing with minimum and maximum number of users with assigned roles and privileges. Testing core features such as transactions, price predictions and searching. Testing system scalability and performance and behavior based on logged in user and task performed. Ensure application compatibility on various web browsers and devices. Test system response to invalid and extremely long usernames/passwords. Verify behavior when attempting to log in with a locked account. Gradually increase load on the application to determine maximum capacity. Test application response time with minimum and maximum network speeds.

6.3. Unit Testing

6.3.1. Evaluate individual products (functions, methods).
6.3.2. Monitor the behavior of each unit.
6.3.3. Display input/output for desired results.
6.3.4. Detect and treat diseases in the early stages of development.
6.3.5. Use a testing framework (e.g. JUnit for Java) for automation.
6.3.6. Make sure the unit is working properly before integration.
6.3.7. Focus on small sections of code rather than the entire application.
6.3.8. Checks testing tools such as PyTest, NUnit or Mocha, depending on the programming language.

6.4. Integration Testing

Verifying interactions between components ensuring working. Errors, inconsistencies or problems that may occur during the integration of the integration. During this testing, developers evaluate the communication, data flow, and performance of the interface. Integration testing helps identify issues such as incorrect data transfer, compatibility issues, or product compatibility issues. These tests are important to identify errors that may occur if one integrates to ensure that the entire system works as expected. To fully test the capabilities of the integration, automation tools such as JUnit, TestNG or Jest can be used to simplify and streamline the integration process.

6.5. Performance Testing

6.5.1. Load testing: Testing the physical ability to control the product to ensure that it performs well under normal conditions.
6.5.2.Stress testing: testing its behavior under extreme conditions, pushing the system beyond its capabilities to identify malfunctions and inefficiencies. work increases or decreases.
6.5.3. Batch Testing: Involves testing a process using large data sets to ensure it is efficient and functional.
6.5.4. Evaluation Tests: Test the machine's ability to solve stable tasks over a long period of time, detect performance degradation or memory loss.

6.6. Stress Testing

6.6.1. Define voltage conditions that simulate high loads.

6.6.2. Set performance indicators and thresholds.

6.6.3. Use tools like JMeter or Gadling for testing. 4. Gradually increase users or their willingness to trust the system.

6.6.4. Pay attention to important signs of malfunction. 6. Find breaking points and weaknesses.

6.6.5. Analyze the results, optimize the code and configuration.

6.6.6. Repeat the test after optimization to check the improvement.

6.6.7. including return events; Measure system endurance. 10. Data discovery is constantly being developed and prepared for unexpected usage **patterns.**

Chapter 7

Summary, Conclusion and Future Enhancements

7.1. Project Summary

The project aims to develop an online shopping platform with integrated artificial intelligence capabilities for small and medium retailers. The solution called Swift Buy Hub provides an easy-to-use webstore to enable transactions along with smart price predictions using machine learning to suggest optimal pricing. Additionally, it incorporates a conversational chatbot to offer voice-based shopping assistance. Built on cloud infrastructure leveraging NodeJS, MongoDB and Angular, Swift Buy Hub hopes to make advanced technologies accessible for niche ecommerce segments. The automated and intelligent features distinguish it from typical catalog-driven platforms to solve pain points around inventory planning, demand forecasting and customer engagement for virtual storefronts. After launch, the next phase involves extending its recommendations engine for personalized promotions and loyalty programs to further boost customer experience.

This summarizes the objective, key capabilities, technology stack, target user base, overall vision and future roadmap for the Swift Buy Hub ecommerce web application. Please let me know if you need any modifications or have additional guidelines on structuring the executive summary. I'm happy to refine it further

7.2. Achievements and Improvements

Swift Buy Hub has successfully revolutionized online shopping for small and medium retailers with its integrated AI capabilities. The platform offers a user-friendly webstore for seamless transactions and employs machine learning for optimal pricing. The addition of a conversational chatbot has boosted customer engagement, while cloud infrastructure and technologies like NodeJS, MongoDB, and Angular ensure accessibility for niche ecommerce segments.

By addressing pain points such as inventory planning and demand forecasting, Swift Buy Hub stands out from traditional platforms. Looking ahead, it aims to enhance customer experience with personalized promotions and loyalty programs. To further improve, ongoing enhancements in AI algorithms and chatbot capabilities could refine the platform's intelligence. Additionally, scalability and performance optimization within the technology stack will ensure smooth operations as the user base grows.

7.3. Critical Review

Swift Buy Hub shows promise in transforming online shopping for small and medium retailers through integrated AI capabilities. While its user-friendly webstore and machine learning for pricing optimization are commendable, ongoing refinement of AI algorithms and chatbot capabilities is necessary.

- Swift Buy Hub transforms online shopping for small and medium retailers with integrated AI capabilities.
- Features a user-friendly webstore and machine learning for pricing optimization.
- Ongoing refinement of AI algorithms and chatbot capabilities is necessary.
- Addressing scalability concerns as the user base expands is crucial for long-term success.
- Stands out for addressing critical pain points like inventory planning and customer engagement.
- Plans to enhance the customer experience further through personalized promotions and loyalty programs.

7.4. Lessons Learnt

The Swift Buy Hub project has provided valuable lessons for future endeavors in the ecommerce domain. Firstly, the importance of user-centric design and intuitive interfaces cannot be overstated. Prioritizing user experience ensures higher engagement and adoption rates among retailers and customers alike.

Secondly, the integration of advanced technologies like AI requires continuous iteration and improvement. Investing in ongoing refinement of AI algorithms and chatbot capabilities is essential to deliver on the promise of enhanced intelligence and responsiveness

- User-centric design and intuitive interfaces are paramount for higher engagement.
- Continuous refinement of AI algorithms and chatbot capabilities is essential.
- Addressing scalability concerns early ensures sustainable growth.
- Fostering a culture of innovation and adaptability is vital for competitiveness.

- Incorporating lessons learned will drive success in future endeavors.

7.5. Future Enhancements/Recommendations

- Further refine AI algorithms and chatbot capabilities to enhance intelligence and responsiveness.
- Explore opportunities to integrate additional advanced technologies, such as natural language processing or computer vision, to enrich the platform's capabilities.
- Implement robust analytics tools to gather and analyze data for deeper insights into customer behavior and market trends.
- Continuously optimize the platform's performance and scalability to accommodate growing user demand.
- Enhance personalization features to offer tailored recommendations and experiences for individual users.
- Expand integration capabilities to allow seamless connectivity with other business tools and platforms, enabling retailers to streamline their operations effectively

Reference and Bibliography

References

[1] Atlassian, "Agile Project Management," [Online]. Available: https://www.atlassian.com/agile. .

[2] "Work Breakdown Structure WBS)," " Project Management Institute, [Online]. Available: https://www.pmi.org/learning/library/work-breakdown-structure-fundamentals-6961.

[3] ""Empathy Mapping: A Guide to Getting Inside a User's Head"," Nielsen Norman Group, [Online]. Available: https://www.nngroup.com/articles/empathy-mapping/.

[4] "Robert, “MEAN vs MERN | The Basics and Differences Between the Two,”," eLuminous , [Online]. Available: https://eluminoustechnologies.com/blog/mean-vs-mern/.

[5] S. Thakkar, "Software product & app development company in USA, India and Canada," Iconflux, [Online]. Available: https://iconflux.com/blog/ai-machine-learning-with-node-js.

[6] N. N. a. P. Singh, "The MEAN Stack," *International Research Journal of Engineering* , vol. 4, no. 5, pp. 3237-3240, 2017.

[7] S. Thakkar, "Integrating AI and Machine Learning with Node.js for Mobile App," Iconflux, 10 2023. [Online]. Available: https://iconflux.com/blog/ai-machine-learning-with-node-js.

[8] "IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications," IEEE, [Online]. Available: https://ieeexplore.ieee.org/document/720574..

[9] "“Usability Requirements of Web Software’s”," [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016814000568.

[10] "Use Case Modeling," Rational Unified Process," [Online]. Available: https://www.ibm.com/docs/en/rational-system-architect/9.6.1?topic=concepts-use-case-modeling.

[11] "UML Use Case Diagrams: Examples and Tutorial," Lucidchart," [Online]. Available: https://www.lucidchart.com/pages/uml-use-case-diagram.

[12] "Use Case Description Template, Confluence," [Online]. Available: https://www.atlassian.com/confluence/templates/use-case-description.

[13] "Software Architecture Diagrams," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/software-architecture-diagrams.

[14] "UML Class Diagram Tutorial," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/uml-class-diagram.

[15] "UML Sequence Diagram Tutorial," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/uml-sequence-diagram.

[16] "UML Activity Diagram Tutorial," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/uml-activity-diagram.

[17] "UML State Machine Diagram Tutorial,," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/uml-state-machine-diagram.

[18] E. H. R. J. R. &. V. J. Gamma, "Design Patterns: Elements of," [Online].

[19] M. Fowler, "Refactoring: Improving the Design of Existing Code. Addison-Wesley," [Online].

[20] "Angular Documentation," [Online]. Available: https://angular.io/docs.

[21] "Express Js Documentation," [Online]. Available: https://expressjs.com/en/api.html.

[22] "MongoDB Documention," [Online]. Available: https://www.mongodb.com/docs.

[23] "Skikit-Learn Documentation," [Online]. Available: https://scikit-learn.org/stable/user_guide.html.

[24] "Dialogflow Documentation," [Online]. Available: https://cloud.google.com/dialogflow/docs.

[25] "AWS Documentation," [Online]. Available: https://docs.aws.amazon.com/.

[26] "TerraForm Documentation," [Online]. Available: https://www.terraform.io/docs.

[27] "Atlassian," Agile Methodolgy, [Online]. Available: https://code.visualstudio.com/docs.

[28] "Visual Studio Code Documentation," [Online]. Available: https://code.visualstudio.com/docs.

[29] " Git Documentation," [Online]. Available: https://git-scm.com/doc.

[30] " GitHub Actions Documentation.," [Online]. Available: https://docs.github.com/en/actions.

[31] " Docker Documentation," [Online]. Available: https://docs.docker.com/.

[32] R. C. Martin, "Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall," [Online].

[33] R. C. Martin, ". Agile Software Development, Principles, Patterns, and Practices.," [Online].

[34] "OWASP," OWASP Top 10, [Online]. Available: https://owasp.org/www-project-top-ten/.

[35] " Chacon, S., & Straub, B," Pro Git. Apress. [Online].

[36] "Conventional Commits Specification," [Online]. Available: https://www.conventionalcommits.org/en/v1.0.0/.

[37] "OWASP," OWASP Top 10, [Online]. Available: https://owasp.org/www-project-top-ten/.