**SAVIFY**

|  |
| --- |
|  |

**By:**

**Muzammil Arif**

**35747**

**Farhan Ahmed**

**32621**

**Supervised by:**

**Mr. Hassaan Ali Shah**

**Faculty of Computing**

**Riphah International University, Islamabad**

**Spring 2025**

**A Dissertation Submitted To**

**Faculty of Computing,**

**Riphah International University, Islamabad**

**As a Partial Fulfillment of the Requirement for the Award of the Degree of**

**Bachelors of Science in Computer Science**

**Faculty of Computing**

**Riphah International University, Islamabad**

Date: December, 2024

**Final Approval**

This is to certify that we have read the report submitted by ***Muzammil Arif (35747)***, ***Farhan Ahmed (32621)*** for the partial fulfillment of the requirements for the degree of the Bachelors of Science in Computer Science (BSCS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelors of Science in Computer Science (BSCS).

**Committee:**

|  |  |
| --- | --- |
| **1** | Hassaan Ali Shah  (Supervisor) |
|  |  |
| **2** | Dr. Musharraf  (Head of Department/chairman) |

**Declaration**

We hereby declare that this document “**Savify**” neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers, especially our supervisor **Hassaan Ali Shah**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muzammil Arif**

**35747**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Farhan Ahmed**

**32621**

**Dedication**

Our project is dedicated to our parents, teachers, friends, and our supervisor " Hassaan Ali Shah" who has been our mentor and inspiration throughout out educational journey. We are pleased to dedicate our project to such motivational and inspiring people.**Acknowledgement**

First of all, we are obliged to Allah Almighty the Merciful, the Beneficent and the source of

all Knowledge, for granting us the courage and knowledge to complete this Project.

We are greatly indebted to our project supervisor “Hassaan Ali Shah”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Muzammil Arif**

**35747**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Farhan Ahmed**

**32621**

**Abstract**

In today’s digital age, online shopping has become an integral part of consumer lifestyle, yet many existing platforms fall short of delivering an engaging and intuitive experience. Shoppers often face challenges like limited **interaction**, and the **inability to bargain**, which diminishes the appeal of online shopping compared to traditional markets. Moreover, searching for products can be **time-consuming**, especially for users who want to browse visually or use natural language rather than text search. In response to these limitations, there is a need for an e-commerce platform that combines the **convenience** of online shopping with features that closely mimic the traditional shopping experience. This approach would provide consumers with greater **flexibility**, **personalized interaction**, and a more efficient way to find products tailored to their **preferences**.

Table of Contents

[Table of Contents i](#_Toc113957278)

[List of Tables iii](#_Toc113957279)

[List of Figures iv](#_Toc113957280)

[Abstract 1](#_Toc113957281)

[Chapter 1: Introduction 2](#_Toc113957282)

[1.1 Goals and Objectives 2](#_Toc113957283)

[1.2 Scope of the Project 2](#_Toc113957284)

[Chapter 2: Literature Review 3](#_Toc113957285)

[2.1 Introduction 3](#_Toc113957286)

[2.2 Background and Problem Elaboration 3](#_Toc113957287)

[2.3 Detailed Literature Review 3](#_Toc113957288)

[2.3.1 Definitions 3](#_Toc113957289)

[2.3.2 Related Research Work 1 3](#_Toc113957290)

[2.3.3 Related Research Work 2 3](#_Toc113957291)

[2.4 Literature Review Summary Table 3](#_Toc113957292)

[2.5 Research Gap 3](#_Toc113957293)

[2.6 Problem Statement 3](#_Toc113957294)

[Chapter 3: Requirements and Design 4](#_Toc113957295)

[3.1 Requirements 4](#_Toc113957296)

[3.1.1 Functional Requirements 4](#_Toc113957297)

[3.1.2 Non-Functional Requirements 4](#_Toc113957298)

[3.1.3 Hardware and Software Requirements 4](#_Toc113957299)

[3.2 Proposed Methodology 4](#_Toc113957300)

[3.3 System Architecture 4](#_Toc113957301)

[3.4 Use Cases 4](#_Toc113957302)

[3.4.1 Sample Use Case Name Here 4](#_Toc113957303)

[3.5 Database Design *(Optional)* 6](#_Toc113957304)

[3.6 Class diagram *(Optional)* 6](#_Toc113957305)

[3.7 Sequence diagram *(Optional)* 6](#_Toc113957306)

[3.8 Any Other Artifact… 6](#_Toc113957307)

[3.9 Graphical User Interfaces(GUI) (*Optional)* 6](#_Toc113957308)

[Chapter 4: Implementation and Test Cases 7](#_Toc113957309)

[4.1 Implementation 7](#_Toc113957310)

[4.1.1 Implementation of First Component/Algorithm 7](#_Toc113957311)

[4.2 Test case Design and description 7](#_Toc113957312)

[4.2.1 Sample Test case No.1 7](#_Toc113957313)

[4.2.2 Sample Test case No.2 7](#_Toc113957314)

[4.3 Test Metrics 8](#_Toc113957315)

[4.3.1 Sample Test case Matric.No.1 8](#_Toc113957316)

[4.3.2 Sample Test case Metric.No.2 8](#_Toc113957317)

[4.3.3 Sample Test case Metric.No.3 8](#_Toc113957318)

[Chapter 5: Experimental Results and Analysis 9](#_Toc113957319)

[Chapter 6: Conclusion and Future Directions 10](#_Toc113957320)

[References 11](#_Toc113957321)

[Appendix 12](#_Toc113957322)

[Appendix A: Guidelines 12](#_Toc113957323)

[Appendix B: Heading of Sample Appendix B 12](#_Toc113957324)

[Formatting Guidelines 13](#_Toc113957325)

[Chapter 1: Heading 1 13](#_Toc113957326)

[1.1 Heading 2 13](#_Toc113957327)

[1.1.1 Heading 3 13](#_Toc113957328)

[Tables and Figures 14](#_Toc113957329)

[Equations 15](#_Toc113957330)

[Header/Footer 15](#_Toc113957331)

[Other Formatting Guidelines 15](#_Toc113957332)

[References 15](#_Toc113957333)

List of Tables

[Table 1: This is Sample table caption 1](#_Toc55559466)

[Table 2: This is Sample table caption 1](#_Toc55559467)

List of Figures

Figure 1: List of Styles 1

Figure 2: IEEE Reference style 1

# Abstract

*“Savify”* is a web-based multi-vendor e-commerce platform designed to provide an enhanced online shopping experience, seamlessly connecting buyers and sellers. Built using the *MERN Stack* technology, *Savify* combines ease of use with efficient functionality, catering specifically to online shoppers who seek a more interactive and traditional approach to e-commerce.

The platform enables sellers to upload product details while offering buyers intuitive ways to search for items, explore product information, and negotiate prices in real time. With AI-driven sentiment analysis, additionally user can simply search for products using image upload, image capture and image auto detection using camera.

*Savify* redefines online shopping, making it user-focused and accessible, while preserving the engagement and integrity of a traditional shopping experience.

# Introduction

**Savify** is a web-based multi-vendor platform designed to create a more interactive and personalized online shopping experience. The platform allows sellers to upload product information, while buyers can explore, negotiate, and search for items using advanced search functionalities. As current e-commerce platforms often lack personalization and effective engagement features, customers frequently experience impersonal transactions, limited communication options, and challenges in finding relevant products.

A significant issue addressed by Savify is the limited interaction between buyers and sellers, as well as the lack of real-time communication tools, which often results in a detached shopping experience.. To address these gaps, Savify incorporates mechanisms to improve buyer-seller interaction, enhance search capabilities, and maintain a respectful online environment through AI-driven moderation.

Savify aims to provide a transparent, secure, and efficient shopping experience, enhancing customer satisfaction and promoting innovation in online retail. Ultimately, Savify supports a more dynamic e-commerce environment, contributing to the growth and engagement of digital marketplaces.

**.**

## Goals and Objectives

The primary objective of *Savify* is to develop an interactive e-commerce platform where customers can browse, search, and negotiate confidently, creating an experience that feels as engaging as traditional shopping.

**Goals:**

* To provide a web-based platform enabling shoppers to negotiate on product and get in their desired price.
* To enhance the online shopping experience by enabling visual base search which makes search process easier.

**Objectives:**

* Create a platform that allows multiple vendors to list and sell products seamlessly, providing a user-friendly experience for both sellers and buyers.
* Develop an AI-driven feature that enables real-time price negotiations between customers and vendors, offering a personalized and interactive shopping experience.
* Incorporate image and voice search capabilities to enhance product discovery and improve user convenience.

*Savify* aims to set a new standard in e-commerce by addressing gaps in personalization, interaction, and accessibility that are prevalent in current platforms.

## Scope of the Project

* Our Website will be developed on MERN.
* Create a web-based system that allows multiple vendors to register, list, and manage their products, including inventory, pricing, and order fulfillment.
* Implement secure user registration and login functionalities for both customers and vendors, along with profile management features.
* Integrate image search functionality enabling users to search for products using images, and incorporate voice search to enhance user experience and accessibility.
* Develop an AI-driven bargaining feature that allows customers to negotiate prices in real-time, providing a dynamic and personalized shopping experience.
* Include features for customer reviews, ratings, and a support system to address inquiries and enhance overall service quality.
* Problem will be solved with Machine Learning.

The platform would be accessible and user-friendly, and simplified for both buyer and seller.

# Literature Review

## Introduction

**Savify** is a web-based multi-vendor e-commerce platform developed to bring a traditional shopping experience into the digital space. While e-commerce has seen rapid growth globally, many platforms **lack interactive** and **personalized features** that could improve user experience and buyer-seller engagement. Savify is tailored specifically to provide customers with a more dynamic and connected shopping journey. The platform enables sellers to easily share product details and interact with buyers, addressing common issues such as limited negotiation options, impersonal interactions, and difficulty in finding products intuitively.

The primary challenges addressed by Savify include the lack of personalized interaction and buyer engagement, which often reduces customer satisfaction on existing platforms. By integrating features like real-time bargaining, Savify has taken steps to bridge this gap, making online shopping both interactive and efficient.

## Background and Problem Elaboration

While multi-vendor e-commerce platforms have broadened online selling opportunities, they often lack features that enhance personalization and user interaction. Traditional platforms miss the personalized negotiations and interactive experiences of physical stores, leading to less customer engagement.

Text-based search functionalities can be limiting due to language barriers or vague descriptions, making product discovery frustrating. By integrating an **AI Bargaining System**, **image search**, and **voice search**, the platform can simulate in-store experiences and improve accessibility.

This project aims to create a web-based multi-vendor e-commerce platform that addresses these shortcomings by incorporating advanced AI features to enhance user satisfaction and streamline the shopping experience.

## Detailed Literature Review

### Definitions

### Related Research Work 1

### Related Research Work 2

## Literature Review Summary Table

The columns in the table depend upon your problem and should be specific to your project.

Table 1: History of Computing Devices

The summary of various computing devices invented in the past from 1833-1901 is presented here.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | Name, reference | Inventor | Year | Input | Output | Description |
| 1. | Artificial Intelligence Based Price Negotiating E-commerce Chat Bot System [1] | A. Porselvi1, Pradeep Kumar Ramesh2, A.V Hemakumar3, S.Manoj Kumar4 | 2021 | Punch cards | Printer, curve plotter, bell | First general purpose computer that had an arithmetical logic unit and could compute using conditional branching and loops. Also incorporated integrated memory. |

## Research Gap

Despite advancements in e-commerce technologies, there is a notable gap in integrating advanced AI features into multi-vendor platforms to enhance user interaction and personalization. Current multi-vendor e-commerce platforms often lack the incorporation of AI-driven bargaining systems that allow for dynamic price negotiations, a feature that could simulate the personalized experience of physical store shopping. While some platforms have implemented image or voice search independently to improve product discovery, there is limited research and practical application combining these functionalities within a single platform.

Existing studies have primarily focused on the individual implementation of AI bargaining agents, image search algorithms, or voice recognition systems in e-commerce settings. However, they do not address the challenges and benefits of integrating these technologies holistically in a multi-vendor environment. This integration could bridge the gap between the impersonal nature of online shopping and the interactive experiences of brick-and-mortar stores.

The research gap lies in developing a comprehensive, web-based multi-vendor e-commerce platform that seamlessly incorporates an AI bargaining system alongside advanced image and voice search capabilities. Such an integration remains underexplored in academic research and commercial applications. Addressing this gap can lead to a more engaging and accessible shopping experience, meeting modern consumer expectations and providing vendors with innovative tools to enhance customer satisfaction and loyalty.

## Problem Statement

Despite the proliferation of multi-vendor e-commerce platforms, many lack advanced features that provide personalized and interactive shopping experiences akin to physical stores. Traditional platforms often miss opportunities for real-time price negotiations, leading to reduced customer engagement and satisfaction. Additionally, reliance on text-based search functionalities presents challenges for users facing language barriers or when product descriptions are insufficient, making product discovery cumbersome. There is a pressing need for an innovative e-commerce solution that integrates an AI-driven bargaining system, image search, and voice search capabilities to enhance user interaction, accessibility, and overall satisfaction in the online shopping experience

# Requirements and Design

In this chapter, we have developed the functional requirements for our actors, i.e., **Customer, Seller, and Admin**. The requirements are specifically designed for the **Savify** platform.

Savify is a web-based e-commerce platform designed to provide an interactive and efficient way for customers and sellers to connect and engage with each other. The platform is user-friendly, easy to navigate and search, and offers features such as AI bargaining system, image search, AI assistant, and speech-to-text support. These functionalities ensure a convenient and seamless experience for all users.

We created system use cases based on each functional requirement and developed corresponding use case diagrams. Additionally, we prepared fully dressed use cases for the main actors, i.e., **Customer, Seller, and Admin**, ensuring that each role's interactions and responsibilities are clearly outlined within the Savify system.

## Requirements

### Functional Requirements

**Buyer:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-1.1 | Buyer shall be able to sign up on website |
| FR-1.2 | Buyer shall be able to login to the website |
| FR-1.3 | Buyer shall be able to edit their profile |
| FR-1.4 | Buyer shall be able to recover passwords |
| FR-1.5 | Buyer shall be able to add product to cart |
| FR-1.6 | Buyer shall be able to buy product |
| FR-1.7 | Buyer shall be able to bargain from seller |
| FR-1.8 | Buyer shall be able to view products |
| FR-1.9 | Buyer shall be able to view orders. |
| FR-1.10 | Buyer shall be able to view cart. |
| FR-1.11 | Buyer shall be able to chat with seller. |
| FR-1.12 | Buyer shall be able to view chats. |

**Seller:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-2.1 | Seller shall be able to register their account. |
| FR-2.2 | Seller shall be able to login to their account. |
| FR-2.3 | Seller shall be able to edit their profile. |
| FR-2.4 | Seller shall be able to recover passwords. |
| FR-2.5 | Seller shall be able to add products. |
| FR-2.6 | Seller shall be able to view products. |
| FR-2.7 | Seller shall be able to delete products. |
| FR-2.8 | Seller shall be able to edit products. |
| FR-2.9 | Seller shall be able to view orders. |
| FR-2.10 | Seller shall be able to manage orders. |
| FR-2.11 | Seller shall be able to reply to customers. |

**Admin:**

|  |  |
| --- | --- |
| **ID** | **Requirements** |
| FR-3.1 | Admin shall be able to login to account. |
| FR-3.2 | Admin shall be able to edit profile. |
| FR-3.3 | Admin shall be able to view sellers. |
| FR-3.4 | Admin shall be able to add sellers. |
| FR-3.5 | Admin shall be able to delete sellers. |
| FR-3.6 | Admin shall be able to edit sellers. |
| FR-3.7 | Admin shall be able to view buyers. |
| FR-3.8 | Admin shall be able to add buyers.. |
| FR-3.9 | Admin shall be able to delete buyers. |
| FR-3.10 | Admin shall be able to edit buyers. |

### Non-Functional Requirements

 User**-Friendly Interface:** Savify offers a simple and intuitive user interface, ensuring customers and sellers can easily navigate the platform and access desired features without any confusion.

 HighPerformance: The system is designed to handle large numbers of concurrent users, ensuring fast response times and smooth functionality during peak usage.

 Scalability**:** Savify is built to scale as the number of users and products grows, allowing for seamless expansion without degrading performance.

 Data **Security and Privacy:** The platform ensures the confidentiality and security of user data through robust encryption protocols and compliance with data protection standards.

### Hardware and Software Requirements

**Hardware Requirements:**

* **Server:** Server should run windows 10-11 for the latest requirements.
* **Storage:** Moderate Storage to save all the data during and after project completion.
* **Processors:** High performance Processors such as GPUs to efficiently compute the projects.
* **Camera:** Webcam for product Search through image detection.

**Software Requirements:**

* **Operating System:** Operating system such as Windows, Linux or MacOS.
* **Database:** We used MongoDB as our Database for storage purpose.
* **Programming Languages:**
* The website can be built using the MERN stack, which includes:

➢ JavaScript: for server-side and client-side scripting.

➢ Node.js: A JavaScript runtime environment for server-side development.

➢ Express.js: a web application framework for building the server-side application.

➢ React.js: A JavaScript library for building the client-side user interface.

* Python: Trained YOLOv11 latest model on product images.
* **Development Tools:** Development tools such as Google Colab, Visual Studio code to run and debug codes. Furthermore, we used Roboflow to annotate images of dataset.
* **Version Control:** A version control system like Git to manage source code and collaborate with multiple developers.

## Proposed Methodology

Savify is a web-based platform designed to connect customers and sellers easily. The platform is tailored specifically for e-commerce, providing features that facilitate seamless interaction between both parties. Sellers can upload their products, and customers can browse, search, and purchase items conveniently.

As current e-commerce platforms in Pakistan often lack features like real-time bargaining and AI-assisted shopping, Savify addresses these gaps by creating a user-friendly and innovative experience. The platform allows sellers to showcase their products with images and descriptions, while customers can negotiate prices using the built-in bargaining system.

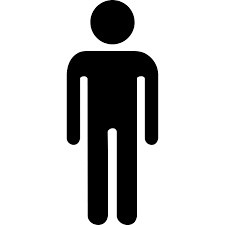
We are building Savify using MERN Stack technology. The platform also features an \*\*AI-powered assistant\*\* to enhance the user experience. This assistant can help customers with queries related to products or features, ensuring they get the information they need. Additionally, we have implemented \*\*AI sentiment analysis\*\* to monitor reviews and remove abusive comments, maintaining a safe and respectful environment.

If a seller doesn't respond to a customer’s bargaining offer or query within a specified timeframe, customers can also interact with Savify’s chatbot, which provides automated responses tailored to their needs and queries.

## System Architecture

## Use Cases

**Admin:**



Login

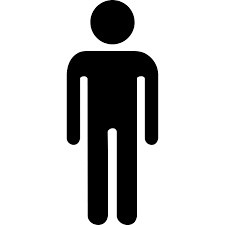
Edit profile

Admin

Manage Buyers

Manage sellers

**Seller:**



Sign up

Login

Add products

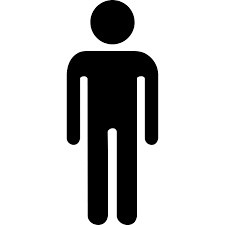
Edit profile

Edit products

Seller

Manage Orders

**Buyer:**



Sign up

Login

Add to cart

Bargain

Buy products

Edit profile

Check out

Buyer

**Fully-Dressed Use Cases:**

### Login:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Login to System | | |
| Actors | | Admin, Seller, Buyer | | |
| Summary | | The user provides their login credentials. If valid, they are granted access to the system. | | |
| Pre-Conditions | |  The user must be registered in the system database.   The user must not already be logged in. | | |
| Post-Conditions | |  The user’s session is initiated.   The user is redirected to their respective dashboard. | | |
| Special Requirements | |  Ensure encryption of passwords during verification.   Provide feedback for invalid credentials. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user opens the login page. | | 2 | The login page is displayed asking for email and password. |
| 3 | The user enters valid email and password. | | 4 | The system verifies the credentials, starts the session, and redirects to the appropriate dashboard. |
| **Alternative Flow** | | | | |
| 3 | The user enters invalid email or password. | | 4-A | The system displays an error message: "Invalid email or password." |

### Sign Up:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign up | | |
| Actors | | Seller, Buyer | | |
| Summary | | The user registers for an account by providing necessary details. | | |
| Pre-Conditions | |  The user must not already have an account.   All required fields in the sign-up form must be valid. | | |
| Post-Conditions | |  A new account is created.   The user is redirected to the login page for authentication. | | |
| Special Requirements | |  Validate email format.   Check for duplicate email addresses during registration. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user navigates to the sign-up page. | | 1 | The user navigates to the sign-up page. |
| 3 | The user fills out the form and submits it. | | 3 | The user fills out the form and submits it. |
| **Alternative Flow** | | | | |
| 4.1 | The user provides invalid or duplicate information. | | 4.2 | The user provides invalid or duplicate information. |

### Edit Profile Information:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Edit Profile Information | | |
| Actors | | Admin, Seller, Buyer | | |
| Summary | | A logged-in user updates their profile details. | | |
| Pre-Conditions | |  The user must be logged in.   The user must have access to the "Edit Profile" section. | | |
| Post-Conditions | |  The user’s updated details are stored in the database.   Changes are reflected in the user’s account. | | |
| Special Requirements | |  Validation for email format, phone numbers, etc.   Real-time feedback for successful updates. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user navigates to "Edit Profile." | | 2 | The system displays the user’s current profile information. |
| 3 | The user modifies the desired fields and submits the form. | | 4 | The system validates inputs and updates the user’s profile in the database, displaying a success message. |
| **Alternative Flow** | | | | |
| 4.1 | The user enters invalid data (e.g., invalid email format). | | 4.2 | The system highlights errors in the form and prompts the user to fix them. |

### :Add Sellers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Add Sellers | | |
| Actors | | Admin | | |
| Summary | | The Admin can add a new seller to the platform by providing seller details such as name, email, contact number etc. and store information. Upon successful addition, the seller is registered in the system. | | |
| Pre-Conditions | |  The Admin must be logged into the system with valid credentials.   The Admin must have the necessary privileges to manage sellers.   The required seller details (e.g., name, email, contact number, and store name) must be available. | | |
| Post-Conditions | | The new seller is successfully added to the database. | | |
| Special Requirements | | The system must validate that the seller’s email is unique. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The Admin selects the "Add Seller" option in the system. | | 2 | The system displays a form for entering seller details (name, email, contact number, store name, etc.). |
| 3 | The Admin enters the seller's details and submits the form. | | 4 | The system validates the input (e.g., unique email, valid contact number). |
| 5 | If the input is valid, the Admin confirms the action. | | 6 | The system saves the seller details in the database. |
|  |  | | 7 | The system displays a success message confirming the seller has been added. |
| **Alternative Flow** | | | | |
| 4.1 | The Admin submits incomplete or invalid seller details (e.g., duplicate email or invalid contact number). | | 4.2 | The system displays an error message indicating the validation failure and highlights the invalid fields. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Manage Buyers | | |
| Actors | | Admin | | |
| Summary | | Admin views, updates, or removes buyer accounts. | | |
| Pre-Conditions | |  Admin must be logged in.   Buyers must be registered in the system. | | |
| Post-Conditions | |  Changes to buyer accounts are reflected in the system.   Updated buyer records are stored in the database. | | |
| Special Requirements | |  Similar structure to "Manage Sellers."   Audit logs for admin actions. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The admin navigates to "Manage Buyers." | | 2 | The system displays a list of registered buyers. |
| 3 | The admin selects a buyer to view, update, or remove. | | 4 | The system performs the action (view details, update record, or delete buyer) and confirms the changes. |
| **Alternative Flow** | | | | |
| 3 | The admin attempts to delete a non-existent buyer. | | 4-A | The system displays an error message: "Buyer not found." |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign up | | |
| Actors | | Seller, Buyer | | |
| Summary | | The user registers for an account by providing necessary details. | | |
| Pre-Conditions | |  The user must not already have an account.   All required fields in the sign-up form must be valid. | | |
| Post-Conditions | |  A new account is created.   The user is redirected to the login page for authentication. | | |
| Special Requirements | |  Validate email format.   Check for duplicate email addresses during registration. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user navigates to the sign-up page. | | 2 | The sign-up page is displayed with required fields (name, email, password, etc.). |
| 3 | The user fills out the form and submits it. | | 4 | The system validates the input, creates a new account, and displays a success message. |
| **Alternative Flow** | | | | |
| 3 | The user provides invalid or duplicate information. | | 4-A | The system highlights errors in the form and prompts the user to fix them. |
| Name | | Purchase Products | | |
| Actors | | Buyer | | |
| Summary | | The buyer selects products, proceeds to checkout, and completes the purchase. | | |
| Pre-Conditions | |  The buyer must be logged in.   The buyer must have items in their cart. | | |
| Post-Conditions | |  The purchase is recorded in the system.   The buyer receives a confirmation of the order. | | |
| Special Requirements | | Provide an option to review the order before confirmation. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The buyer navigates to their cart.. | | 2 | The system displays the items in the cart. |
| 3 | The buyer clicks "Proceed to Check out." | | 4 | The system displays the payment and delivery form. |
| 5 | The buyer confirms the order. | | 6 | The system displays a confirmation message. |
| **Alternative Flow** | | | | |
| 3 |  | | 4-A |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Sign up | | |
| Actors | | Seller, Buyer | | |
| Summary | | The user registers for an account by providing necessary details. | | |
| Pre-Conditions | |  The user must not already have an account.   All required fields in the sign-up form must be valid. | | |
| Post-Conditions | |  A new account is created.   The user is redirected to the login page for authentication. | | |
| Special Requirements | |  Validate email format.   Check for duplicate email addresses during registration. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The user navigates to the sign-up page. | | 2 | The sign-up page is displayed with required fields (name, email, password, etc.). |
| 3 | The user fills out the form and submits it. | | 4 | The system validates the input, creates a new account, and displays a success message. |
| **Alternative Flow** | | | | |
| 3 | The user provides invalid or duplicate information. | | 4-A | The system highlights errors in the form and prompts the user to fix them. |
| Name | | Add Products to Cart | | |
| Actors | | Buyer | | |
| Summary | | The buyer adds selected products to their shopping cart. | | |
| Pre-Conditions | | The buyer must be logged in. | | |
| Post-Conditions | |  The selected product is added to the cart.   The buyer’s cart is updated in real time. | | |
| Special Requirements | | Allow the buyer to specify quantity before adding to the cart. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The buyer clicks "Add to Cart" for a product. | | 2 | The system adds the product to the cart and updates the cart’s total. |
| **Alternative Flow** | | | | |
| 3 | The buyer tries to add an out-of-stock product | | 4-A | The system displays a message: "This product is currently out of stock." |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Complete Checkout | | |
| Actors | | Buyer | | |
| Summary | | The buyer finalizes their order by providing payment and shipping details. | | |
| Pre-Conditions | |  The buyer must be logged in.   The buyer must have items in their cart. | | |
| Post-Conditions | |  The order is placed successfully.   Payment is processed and confirmation is sent to the buyer. | | |
| Special Requirements | | Allow buyers to review and modify their order before confirming. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The buyer clicks "Checkout" in the cart. | | 2 | The system displays a summary of the cart items and the total price. |
| 3 | The buyer provides payment and shipping details. | | 4 | The system validates the input, processes the payment, and displays an order confirmation. |
| **Alternative Flow** | | | | |
| 3 | The buyer provides invalid shipping details | | 4-A | The system highlights the errors and prompts the buyer to correct them. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Add New Products | | |
| Actors | | Seller | | |
| Summary | | The seller adds new products to the platform by providing details such as name, price, description, and images. | | |
| Pre-Conditions | |  The seller must be logged in.   Required product details must be available. | | |
| Post-Conditions | |  The product is successfully added to the system.   The product becomes available for buyers to view and purchase. | | |
| Special Requirements | |  Validate input fields (e.g., price, name, and description length).   Allow multiple images to be uploaded. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The seller navigates to the "Add Products" page. | | 2 | The system displays a form for entering product details (name, price, description, and images) |
| 3 | The seller fills out the form and uploads images. | | 4 | The system validates the input, saves the product, and displays a success message. |
| **Alternative Flow** | | | | |
| 3 | The seller provides incomplete or invalid details. | | 4-A | The system highlights the errors and prompts the seller to correct them. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | View and Manage Buyer Orders | | |
| Actors | | Seller | | |
| Summary | | The seller reviews and manages orders placed by buyers for their products. | | |
| Pre-Conditions | |  The seller must be logged in.   Orders for the seller's products must exist in the system. | | |
| Post-Conditions | |  The seller updates the status of orders (e.g., confirmed, shipped).   Buyers are notified of order status changes. | | |
| Special Requirements | |  Display order details clearly, including buyer information.   Allow filtering orders by name | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The seller navigates to the "Manage Orders" page. | | 2 | The system displays a list of orders for the seller's products. |
| 3 | The seller selects an order and updates its status. | | 4 | The system updates the order status and notifies the buyer of the changes. |
| **Alternative Flow** | | | | |
| 3 | The seller attempts to update an invalid order. | | 4-A | The system displays an error message: "Unable to update order. Please try again." |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | | Edit Existing Products | | |
| Actors | | Seller | | |
| Summary | | The seller modifies details of their existing products. | | |
| Pre-Conditions | |  The seller must be logged in.   The product to be edited must exist in the system and belong to the seller. | | |
| Post-Conditions | |  The updated product details are saved in the system.   Buyers see the updated product details immediately. | | |
| Special Requirements | |  Validate all updated fields (e.g., price must be numeric, description must meet length criteria).   Ensure no duplicate product names within the seller's product list. | | |
| Basic Flow | | | | |
| Actor Action | | | **System Response** | |
| 1 | The seller navigates to the "Edit Products" page. | | 2 | The system displays a list of products added by the seller. |
| 3 | The seller selects a product to edit. | | 4 | The system displays the product details in an editable form. |
| 5 | The seller updates the details and submits the form. | | 6 | The system validates the input, saves the changes, and displays a success message. |
| **Alternative Flow** | | | | |
| 5 | The seller enters invalid or duplicate product details. | | 6-A | 6-A. The system highlights errors and prompts the seller to fix them. |

## Database Design *(Optional)*

## Class Diagram (*Optional)*

## Sequence diagram *(Optional)*

## Any Other Artifact…

## GUI Graphical User Interfaces (*Optional)*

This section should give the GUI dumps of each screen, with reference to the user. The navigation flow of each user is also required, and each GUI should mark the functionality/use case that it covers.

# Implementation and Test Cases

**For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.**

## Implementation

Whatever implementation that you have done so far, please elaborate here.

Give clear details of the algorithms that were implemented along with the platform and the APIs which were used. **For FYP-1, this chapter can be changed to description of prototype developed.**

### Implementation of First Component/Algorithm

Write implementation of first component of your system here.

## **Test case Design and description**

**This section will be added in FYP-II.** Summarize the common attributes of test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies. The following scheme is recommended for describing test cases in detail.

### Sample Test case No.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **<Software component Name>** | | | | | |
| **<Reference>** | | | | | |
| Test Case ID: | | *Reference Number* | Test Date: | | *Date* |
| Test case Version: | | *Version number* | Use Case Reference(s): | | *Relation to use cases* |
| Revision History: | | *Refer to previous test case identity (if any)* | | | |
| Objective | | *Need and scope of the testing* | | | |
| Product/Ver/Module: | | *Refer to overall system being built and the place of this test case in it.* | | | |
| Environment: | | *Necessary and desired properties of the test environment. (hardware/software)* | | | |
| Assumptions: | | *Assumptions that might affect the testing process.* | | | |
| Pre-Requisite: | | *Necessary condition that needs to be fulfilled prior to the test case.* | | | |
| Step No. | Execution description | | | Procedure result | |
|  | *Events being tested.* | | | *Mention software response.* | |
| Comments: | | | | | |
| *Passed* *Failed* *Not Executed* | | | | | |

### Sample Test case No.2

.

.

.

## Test Metrics

Summarize here the common ground of attributes of test case metrics.

### Sample Test case Matric.No.1

|  |  |
| --- | --- |
| Metric: | Purpose |
| Number of Test Cases: | Total number of test cases that you have developed for your system. |
| Number of Test Cases Passed: | The number of test cases that successfully passed |
| Number of Test Cases Failed: | The number of test cases that failed |
| Test Case Defect Density: | (No of test cases failed \* 100)  No of test cases executed |
| Test Case Effectiveness: | No of defects detected using test cases \*100  Total number of defects detected |
| Traceability Matrix: | Traceability is the ability to determine that each feature has a source in requirements and each requirement has a corresponding implemented feature. |

### Sample Test case Metric.No.2

### Sample Test case Metric.No.3

.

.

.

# Experimental Results and Analysis

**This chapter will be added in FYP-II.** Give proper analysis and discussion of experimental results (in plain English text) along with tables of results. **For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.**

# Conclusion and Future Directions

**This chapter is mandatory.** Give conclusions and summary of the work done. What were your findings and what were the results? Discuss in detail whether the scope of your project was entirely covered or not and whether the objectives of the project were met or not. What challenges did you face and what has been left out and why?

Sum up all the conclusions of all the chapters here to make a conclusion chapter. Do not repeat any text, just summarize it in different words.

Give recommendations for future work also. How your project can be further enhanced or improved? Future recommendations if someone wants to work on it. **For FYP-1 it is mandatory to list down a plan of the work to be done for FYP-2.**

# References

List all important sources of information which have been consulted for this project

# Appendix

## Appendix A: Guidelines

This section should include all supporting information from the project that was not included in the body of the report.  You should include surveys, complex statistical calculations, certain detailed tables and other such information in an appendix.  The information presented in this section is important to support the work presented in the body of the report but would make it more difficult to read and understand if presented within the body of the report.

Cite the appendix items in the report narrative (write "see Appendix A") and organize appendices (e.g., Appendix A, Appendix B,

Any tables, figures, forms, or other materials that are not totally central to the analysis but that need to be included are placed in the Appendix.

## Appendix B: Heading of Sample Appendix B

Following is a sample code with “code” style format.

Void SampleFunction(){

Print “Hello World.”;

}

# Formatting Guidelines

This document also serves as style guide for final year project reports. In order to give a similar high-quality appearance to all final year software project reports this template uses a collection of predefined Microsoft Word formatting styles. **These styles should be used without modification or replacement.** Font in the document is ***“Time New Roman”.*** This template provides following styles:

* **Title** – the main title style
* **Title2** – the subtitle style
* **Body Text** – style for paragraphs
* **Caption** – the style for a figure or table caption
* **Table Description** – the style for description of table, it must be added after caption.
* **Figure Description** - the style for description of figure, it must be added after caption.
* **Code** – the style for program source code

**int x** = 10; // Writing important code

* **Table Header Row** – Style for the header row of table
* **Table Grid** – the style for the data rows in the tables
* **Reference** – The style for references
* **Bullets** – The style for the bullet lists
* **Numbered** **List**– Style for numbered lists

All Heading styles with different level numbers are listed below.

# Heading 1

## Heading 2

### Heading 3

#### Heading 4

##### Heading 5

###### Heading 6

Heading 7

Heading 8

Heading 9

## Tables and Figures

Tables and figures should be centered horizontally. The caption button should be used to insert caption for both the figures and tables. All figures and tables must be numbered properly. Always refer to tables and figures according to their numbers. A table or figure can be cited as follows: ‘see Table1’ or ‘as shown in Table1’. The caption of table should be centered above the table and figure caption should be centered below the figure. Place the tables/figures close to their reference. Use “Table Header Row” and ‘Table Grid’ style for table’s header and data rows respectively. It is compulsory to provide brief description of table/figure after its caption. Styles for table and figure descriptions are “Table Description” and “Figure Description” respectively.

Press Ctrl+Shift+S to see list of styles mentioned above. Figure 1 shows the Apply Style window displaying the list of styles. Select any text then press Ctrl+Shift+S, the Apply Style window will show you the current style applied on that text and if required, you can change the style by selecting any other style from the “Style Name” dropdown.

This is brief description of above figure.

Figure 1: List of Styles

Table 1: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

Table 2: This is Sample table caption

This is brief description of following Table.

|  |  |  |  |
| --- | --- | --- | --- |
| Header row | Header row | Header row | Header row |
| Row1 col1 | Row1 col2 | Row1 col3 | Row1 col4 |
| Row2 col1 | Row2 col2 | Row2 col3 | Row2 col4 |

## Equations

Use equation editor to write equations in this report. Use last button of the custom tool bar to invoke equation editor. Similar to tables and figures, equations should also be aligned centered horizontally. Number all equations and insert them in parenthesis. Below is a sample equation and its reference number. An equation can be referenced like this: ‘it is clear from (1)’.

 (1)

## Header/Footer

Notice the headers in this document, before Introduction (i.e. the main content of this document) page numbers are in roman numerals. The page numbers of the actual content start with Arabic numerals i.e. 1, 2, 3 and so on. All of the **odd numbered pages** contain title of your project while the **even numbered pages** contain the section heading (i.e. chapter’s name) in the headers.

## Other Formatting Guidelines

* Keep 2-4 GUIs in one page. Consume as much space as possible. Do not leave most of page blank unnecessarily.
* Do not break tables (or use cases) in multiple pages unless the table is too large to fit in one page.
* Re-arrange the content i.e., text, images, and tables properly to meet above two guidelines.

## References

Always refer to the source of information by inserting the reference number in square brackets like this [5]. The reference numbers can either be added at the end of the sentence or within the sentence without changing the punctuation of sentence. A reference can also be cited as follows: ‘as Ruskey [2] mentioned’. List each source only once on your reference page.



Figure 2: IEEE Reference style

This figure represents the styling information for adding references in IEEE format

**Following is a list of sample reference for various typed of sources in IEEE format.**

1. P.M. Morse and H. Feshback, *Methods* of *Theoretical Physics*. New York: McGraw Hill, 1953. **//Format for Book**
2. S.K. Kenue and J.F. Greenleaf, “Limited angle multifrequency diffiaction tomography,” *IEEE Trans. Sonics Ultrason*., vol. SU-29, no. 6, pp. 213-2 17, July 1982. **//Format for Journal Article**
3. B. Tsikos, “Segmentation of 3-D scenes using multi-modal interaction between machine vision and programmable mechanical scene manipulation,” Ph.D. dissertation, Univ. of Pennsylvania, BCE Dept., Philadelphia, 1987. [Add if applicable: University Microfilms, Inc., University of Michigan, Ann Arbor, Michigan.] **//Format for Dissertation or thesis**
4. R. Finkel, R. Taylor, R. Bolles, R. Paul, and J. Feldman, “An overview of AL, programming system for automation,” in *Proc. Fourth Int. Joint Conf Artif. Intell*., pp. 758-765, Sept. 3-7, 1975. **//Format for Proceedings paper**
5. “Technology threatens to shatter the world of college textbooks, *The Wall Street Journal*, vol 91, pp. Al, A8, June 1, 1993. **//Format for Newspaper article**
6. R. Cox and J. S. Turner, “Project Zeus: design of a broadband network and its application on a university campus,” Washington Univ., Dept. of Comp. Sci., Technical Report WUCS-91-45, July 30, 1991. **//Format for Technical Report**
7. M. Janzen, *Instant Access Accounting*. Computer software. Nexus Software, Inc IBM-PC, 1993. **//Format for** **Software**
8. Fuminao Okumura and Hajime Takagi, “Maglev Guideway On the Yamanashi Test Line,” *http://www.rtri.or.jp/rd/maglev2/okumura.html*, October 24, 1998. **//Format for** **World Wide Web** (give author and title if named)
9. “AT&T Supplies First CDMA Cellular System in Indonesia,” http://www.att.com/press/1095/951011.nsa.html, Feb 5, 1996. **//Format for World Wide Web**