University of Messina

Bachelor of Data Analysis

ACADEMIC YEAR - 2023/2024

Software Engineering

Marketplace Platform

(Project report)

Supervisor:
Professor Salvatore Distefano

Students:
Sahil Nakrani (533173)
Henok Shimelis Taddese (537462)

Acknowledgments

We would like to extend our deepest gratitude to all the individuals and team who contributed to the successful completion of this project.

Firstly, we are immensely grateful to our development team for their unwavering dedication, hard work, and innovative solutions that have been instrumental in bringing this project to life. Our technical expertise, collaboration, and commitment to excellence have been the foundation of our success.

A special thank you goes to our supervisor Professor Salvatore Distefano. Your input has been crucial in shaping the functionalities and ensuring that the platform meets the needs and expectations of our diverse user base.

Our sincere appreciation extends to our partners and third-party service providers, including Stripe, FastBots, and GTranslate. Your services and support have significantly enhanced the capabilities and reach of our platform.

We are grateful to our quality assurance team for their thorough testing and dedication to maintaining the highest standards of quality and reliability in our product. Your attention to detail has ensured a seamless user experience.

Lastly, we would like to thank our families and friends for their understanding and support during the long hours of development and testing. Your encouragement has been a source of strength and motivation for our team.

Thank you all for your contributions, collaboration, and commitment to making this project a succes

Executive Summary

Project Overview

The project began with the goal of developing a comprehensive e-commerce platform, which was later expanded to evolve into a fully functional multi-vendor marketplace. This expansion was driven by the need to enhance the platform's capabilities, allowing multiple sellers to list and manage their products, thereby increasing its reach and scalability.

Initial Development Phase: The initial phase of the project focused on implementing essential e-commerce functionalities, which included:

- **User Authentication:** Secure login and registration for both users and vendors, ensuring a safe and personalized shopping experience.
- **Product Management:** Tools for vendors to add, update, and manage their product listings, including inventory tracking and categorization.
- **Payment Processing:** Integration with multiple payment gateways to facilitate secure and efficient transactions.
- **Notification Service:** Real-time notifications for order updates, payments, and system alerts, enhancing communication between users and vendors.
- **Admin Panel:** A centralized interface for administrators to manage users, vendors, and overall platform operations.
- **Multi-Language Support:** Integration of a translation feature to cater to a global audience, making the platform accessible to users worldwide.
- **AI-Powered Chatbot:** A 24/7 customer support tool using AI-driven chatbots to assist users with queries and improve their overall experience.
- **Dark/Light Mode Toggle:** A feature allowing users to switch between dark and light themes, enhancing the visual experience based on personal preferences.

Expansion to a Multi-Vendor Marketplace: Following the completion of the core e-commerce functionalities over six sprints, our professor suggested expanding the platform into a multi-vendor marketplace. This led to the addition of two more sprints, during which we focused on the following key enhancements:

- **Multi-Vendor Integration:** Enabled multiple sellers to register, list products, and manage their storefronts independently. This feature was critical in transitioning the platform from a single-vendor model to a scalable marketplace.
- **Root Admin Panel:** Developed a comprehensive control panel for the root admin, providing centralized management of all vendors, products,

transactions, and platform settings. This ensured efficient and streamlined marketplace operations.
- **Seller/Vendor Panel:** Created a tailored interface for vendors, empowering them to manage their products, orders, pricing, and subscriptions with ease.
- **Pricing Model and Subscription Options:** Implemented a flexible pricing structure that included subscription tiers for vendors. This allowed vendors to choose different levels of access and features based on their needs, creating a sustainable revenue model for the platform.

Methodology

The development process was guided by the Agile Scrum methodology, which provided a structured yet flexible approach to managing the project's complexity. Scrum's iterative nature allowed the team to continuously refine the product, incorporating feedback from stakeholders at regular intervals, ensuring that the project stayed aligned with the evolving requirements and expectations.

We started by creating a detailed product backlog that outlined all required features and functionalities. This backlog was continuously refined as the project progressed, with user stories prioritized based on their importance and impact on the overall platform. Each sprint began with a planning meeting where specific tasks were assigned to team members, ensuring that the goals of each sprint were clear and achievable.

Daily stand-ups were held to track progress, address challenges, and make necessary adjustments. At the end of each sprint, we conducted sprint reviews to demonstrate the completed features and gather feedback, which was then used to inform the next sprint's planning. This iterative process allowed the team to remain agile, adapting quickly to changes and new requirements.

Key Achievements

The project achieved several critical milestones that significantly enhanced the platform's functionality and user experience:

- **User Authentication:** Implemented secure and seamless login and registration processes for both users and vendors, establishing a strong foundation for user management.
- **Product Management:** Developed a robust system for vendors to manage their product listings, including inventory tracking, pricing, and categorization.
- **Payment Processing:** Integrated multiple payment gateways to ensure secure and efficient transactions, supporting various payment methods for user convenience.
- **Notification Service:** Implemented real-time notifications for orders, payments, and system updates, keeping users and vendors informed and engaged.

- **Admin and Root Admin Panels:** Developed comprehensive control panels for both regular and root administrators, providing centralized management of the platform, including user and vendor oversight, transaction monitoring, and policy enforcement.
- **Multi-Vendor Integration:** Successfully integrated multiple sellers/vendors into the platform, allowing them to independently manage their products, orders, and profiles, a cornerstone of the platform's transition to a marketplace model.
- **Pricing Model and Subscription Options:** Implemented a flexible pricing model that includes various subscription tiers for vendors, providing different levels of access and features based on their subscription, enabling the platform to generate recurring revenue while offering value to vendors.
- **Enhanced User Experience:** Integrated features such as multi-language translation to cater to a global audience, chatbot support to provide real-time customer assistance, and dark/light mode options to enhance the visual experience for users. These enhancements were designed to improve user engagement and satisfaction, making the platform more accessible and user-friendly.

In summary, the project successfully expanded the e-commerce platform into a robust multi-vendor marketplace, delivering on both functional requirements and user experience goals. The use of Agile Scrum methodology ensured that the project was completed on time, within scope, and with high-quality results. The platform now stands as a scalable, user-friendly marketplace that is ready to serve a diverse and growing community of users and vendors.

Introduction

Background

The project initially started as a straightforward e-commerce platform designed to operate in a single-vendor environment. This initial platform provided the essential functionalities typical of an e-commerce site, including secure user authentication, product management capabilities, payment processing, and an administrative panel to oversee site operations. While this platform was functional and met the basic needs of a single seller, it lacked the scalability and flexibility required to support a more complex, multi-vendor marketplace.

As the project progressed, we get a suggestion from our supervisor, the potential for significant growth by expanding the platform to accommodate multiple vendors. This expansion would allow the platform to support various sellers, each managing their storefronts, thereby increasing the variety of products available to users and enhancing the platform’s competitive edge in the market. The decision to pivot towards a marketplace model was strongly influenced by feedback from our professor, who suggested that this direction would maximize the platform's potential and broaden its scope.

Transitioning from a single-vendor system to a multi-vendor marketplace required not only technical enhancements but also a comprehensive rethinking of the platform's administrative functions. The shift involved implementing new mechanisms for managing multiple vendors, ensuring seamless user experience, and maintaining the platform's operational efficiency despite the increased complexity.

Objectives

The primary objective of this project was to create a user-friendly e-commerce platform with essential features, then expand this platform into a scalable and robust multi-vendor marketplace. The key goals of this expansion included:

Scalability: Develop the platform to support multiple sellers, allowing them to register, manage their products, and handle transactions independently within the marketplace.

Enhanced Administrative Controls: Introduce a root admin panel with comprehensive controls to oversee all vendors, ensuring adherence to platform policies and providing a centralized system for managing the marketplace.

Vendor Management System: Implement a vendor management system that includes features such as vendor registration, product listing, order management, and reporting. This system needed to be intuitive and efficient, allowing vendors to focus on their business operations with minimal friction.

Adaptable Pricing Model: Create a flexible pricing model that allows for various subscription tiers, offering different levels of service and features to vendors based on their needs. This model would also provide a steady revenue stream for the platform through subscription fees.

User Experience Enhancements: Improve the overall user experience by adding features such as multi-language translation, chatbot support, and the option for dark/light modes. These enhancements were aimed at making the platform more accessible, engaging, and user-friendly.

Scope

The scope of the project was carefully defined to ensure that development efforts were focused, and the final product met the desired objectives. The scope was divided into in-scope and out-of-scope items:

In-Scope:

- **User Authentication:** Secure login and registration processes for users and vendors.

- **Product Management:** Tools for vendors to add, update, and manage their product listings.
- **Payment Processing:** Integration with multiple payment gateways to facilitate smooth and secure transactions.
- **Notification Service:** Real-time notifications for orders, payments, and updates for users and vendors.
- **Admin Panel:** A comprehensive panel for the root admin to manage the platform, including user and vendor management.
- **Multi-language Translation:** Support for multiple languages to cater to a global user base.
- **Chatbot:** AI-driven chatbot to provide 24/7 customer support and assist users in navigating the platform.
- **Dark/Light Modes:** A toggle feature allowing users to switch between dark and light themes for a customized experience.
- **Vendor Integration:** Mechanisms to onboard and manage multiple vendors, each with their own product listings and storefronts.
- **Root Admin Panel:** A centralized control panel for managing all aspects of the marketplace, including vendor oversight.
- **Seller/Vendor Panel:** A dedicated interface for sellers to manage their products, orders, and account settings.
- **Pricing Model:** A tiered subscription model offering different levels of service to vendors, with corresponding fees.
- **Subscription Service:** A system to manage vendor subscriptions, including billing and service level management.

Out of Scope:

- **Advanced Analytics:** Although useful for business insights, implementing advanced analytics was not prioritized in the current project scope but is planned for future enhancements.
- **Recommendation Systems:** The development of personalized recommendation systems was also deferred to a later phase, as the focus was on building the core marketplace functionalities first.

Stakeholders

The project involved multiple stakeholders, each playing a critical role in the development and success of the platform:

Project Team: The core team responsible for the development, testing, and deployment of the platform. The team included developers, designers, testers, and a project manager who coordinated the overall project.

Professor (Mentor and Advisor): Provided guidance, feedback, and suggestions throughout the project. The professor’s input was instrumental in shaping the direction of the project, particularly the decision to expand into a multi-vendor marketplace.

Agile Methodology Overview

Agile methodology is a project management approach widely used in software development and other industries that emphasizes flexibility, team collaboration, customer satisfaction, and high adaptability to change. It involves iterative development, where projects are broken into smaller, manageable segments that are completed in work sessions known as sprints, which typically last from one week to one month.

Key Principles of Agile

The Agile Manifesto, the foundational document for agile development, outlines four core values:

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

It also highlights 12 principles which emphasize customer satisfaction, welcoming changing requirements, frequent delivery, collaboration, motivation, face-to-face conversation, working software as the primary measure of progress, sustainable development pace, continuous attention to technical excellence, simplicity, self-organizing teams, and regular reflections to improve effectiveness.

Types of Agile Methodologies

• **Scrum:** Focuses on managing tasks within a team-based development environment. It is the most widely implemented agile method.
• **Kanban:** Emphasizes continuous delivery without overburdening the development team. It’s visualized via a Kanban board, showing all the tasks in various stages of the process.
• **Extreme Programming (XP):** Encourages frequent "releases" in short development cycles, which improves productivity and introduces checkpoints where new customer requirements can be adopted.
• **Lean:** Focuses on minimizing waste and improving efficiency. This methodology is particularly useful in helping businesses with limited resources maximize their development efficiency.

• **Feature Driven Development (FDD):** Focuses on features, which are actually small pieces of functionality that are valued by the customer. It is model-driven, short-iteration process.
• **Dynamic Systems Development Method (DSDM):** An agile method that is part of the wider AgilePM framework and emphasizes the full project lifecycle and a detailed, foundation phase.

Extreme Programming (XP)
Adaptive System Development (ASD)
Feature-Driven Development
Scrum
Kanban
Agile Methodology

The Scrum and the motivation behind it

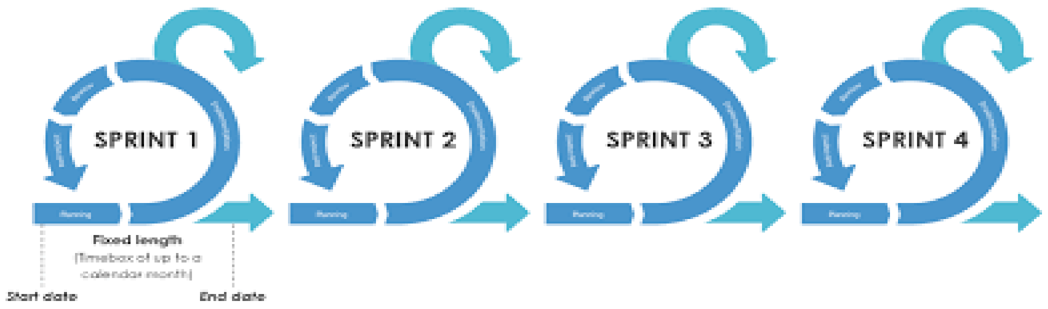
Scrum is a well-established framework within Agile methodologies designed specifically for managing complex projects in a flexible and iterative manner. It allows teams to work incrementally, delivering parts of the product in short, manageable cycles known as sprints. The primary focus of Scrum is on collaboration, continuous improvement, and frequent delivery of functional software, which is especially important in dynamic environments where requirements can evolve rapidly.

Scrum's iterative approach aligns with the principles of Agile, which prioritize customer satisfaction through early and continuous delivery of valuable software. By breaking down the project into smaller, manageable increments, Scrum enables the team to quickly adapt to changing requirements, incorporate stakeholder feedback, and continuously improve the product with each iteration.

Key Features of Scrum:

1. **Iterative Process:**

Scrum breaks down the development process into short, manageable periods known as **SPRINTS**, which typically last between one to four weeks. Each sprint is aimed at producing a shippable product increment, thereby ensuring regular feedback and the ability to adapt to changing requirements.



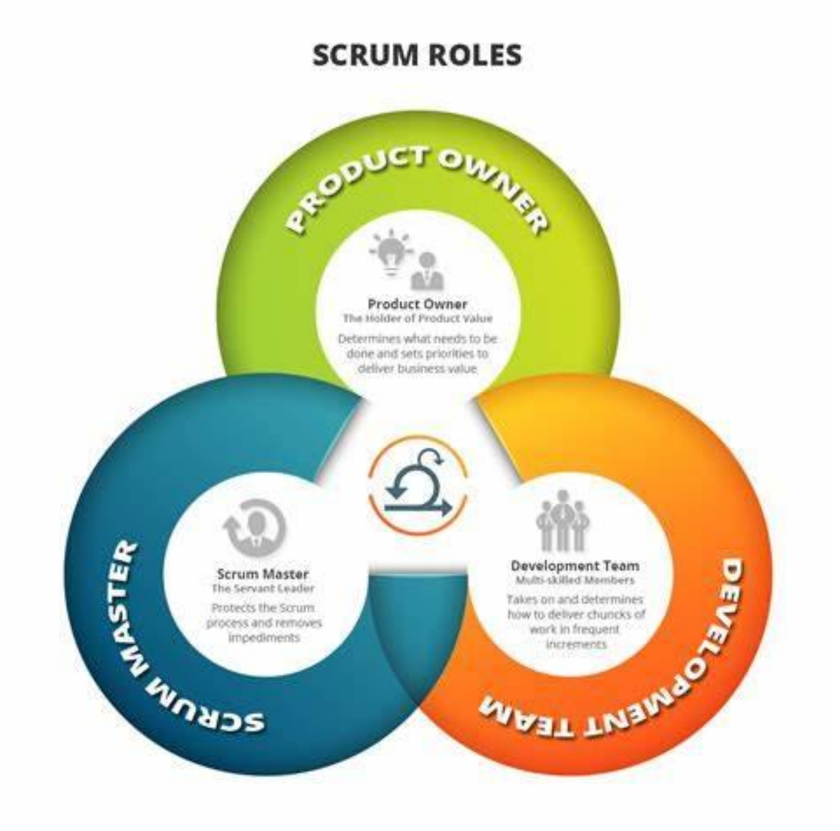
2. Roles:

Scrum defines three key roles, each with specific responsibilities, to ensure the smooth execution of the project:

Product Owner: The Product Owner was responsible for managing the product backlog, which included all the features, enhancements, and bug fixes to be addressed during the project. This role was crucial in prioritizing features based on their value to the project, feedback from our professor, and team discussions. The Product Owner ensured that the team focused on delivering the most important and high-impact features first, aligning the project with its overall goals.

Scrum Master: The Scrum Master facilitated all Scrum ceremonies, such as sprint planning, daily stand-ups, sprint reviews, and retrospectives. Their primary responsibility was to ensure that the team adhered to Scrum principles and practices. The Scrum Master also acted as a buffer between the team and any obstacles that could impede progress, helping to remove these impediments to keep the project on track.

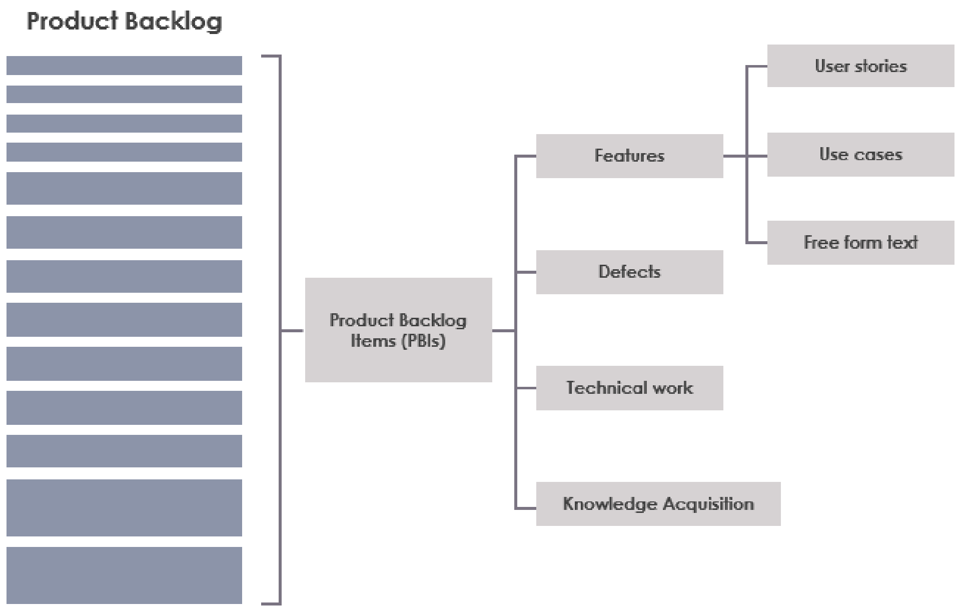
Development Team: The Development Team was responsible for implementing the features, conducting tests, and delivering the product increments at the end of each sprint. The team worked collaboratively, with each member contributing to different aspects of the project, from coding and testing to deploying and reviewing the software. The development team was cross-functional, meaning they possessed all the skills necessary to deliver the product increment within the sprint.

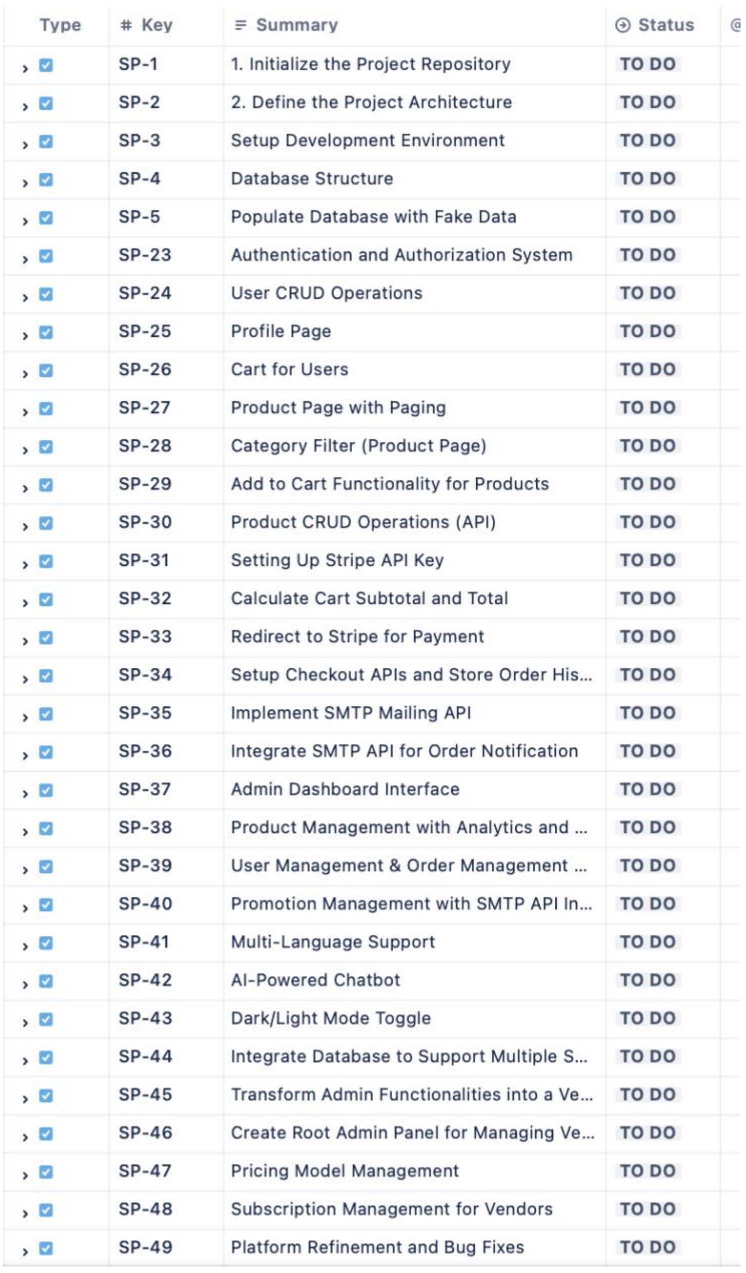

SCRUM ROLES
PRODUCT OWNER
Product Owner
The Holder of Product Value
Determines what needs to be done and sets priorities to deliver business value
SCRUM MASTER
Scrum Master
The Servant Leader
Protects the Scrum process and removes impediments
DEVELOPMENT TEAM
Development Team
Multi-skilled Members
Takes on and determines how to deliver chuncks of work in frequent increments

3. Artifacts:

Scrum utilizes several artifacts to track progress and ensure that the team is aligned with the project goals:

> **Product Backlog:** The Product Backlog was a comprehensive list of all the user stories and tasks that needed to be completed during the project. It included both the initial core e-commerce features and the additional marketplace functionalities. The Product Owner continuously refined and prioritized the backlog, ensuring that the most valuable features were tackled first.
>
> **Sprint Backlog:** The Sprint Backlog was a subset of the Product Backlog, consisting of the tasks selected for implementation during a particular sprint. This backlog was created during the sprint planning meeting and served as the team's to-do list for that sprint. It provided a clear focus for the team and helped track the progress of the sprint.
>
> **Increment:** At the end of each sprint, the team delivered a potentially shippable product increment. This increment included the features developed during that sprint and was integrated with the existing functionality of the platform. Each increment was tested and functional, ensuring that the product was always in a usable state.


Product Backlog
Product Backlog Items (PBIs)
Features
Defects
Technical work
Knowledge Acquisition
User stories
Use cases
Free form text



Type	# Key	Summary	Status
› ✓	SP-1	1. Initialize the Project Repository	TO DO
› ✓	SP-2	2. Define the Project Architecture	TO DO
› ✓	SP-3	Setup Development Environment	TO DO
› ✓	SP-4	Database Structure	TO DO
› ✓	SP-5	Populate Database with Fake Data	TO DO
› ✓	SP-23	Authentication and Authorization System	TO DO
› ✓	SP-24	User CRUD Operations	TO DO
› ✓	SP-25	Profile Page	TO DO
› ✓	SP-26	Cart for Users	TO DO
› ✓	SP-27	Product Page with Paging	TO DO
› ✓	SP-28	Category Filter (Product Page)	TO DO
› ✓	SP-29	Add to Cart Functionality for Products	TO DO
› ✓	SP-30	Product CRUD Operations (API)	TO DO
› ✓	SP-31	Setting Up Stripe API Key	TO DO
› ✓	SP-32	Calculate Cart Subtotal and Total	TO DO
› ✓	SP-33	Redirect to Stripe for Payment	TO DO
› ✓	SP-34	Setup Checkout APIs and Store Order His...	TO DO
› ✓	SP-35	Implement SMTP Mailing API	TO DO
› ✓	SP-36	Integrate SMTP API for Order Notification	TO DO
› ✓	SP-37	Admin Dashboard Interface	TO DO
› ✓	SP-38	Product Management with Analytics and ...	TO DO
› ✓	SP-39	User Management & Order Management ...	TO DO
› ✓	SP-40	Promotion Management with SMTP API In...	TO DO
› ✓	SP-41	Multi-Language Support	TO DO
› ✓	SP-42	AI-Powered Chatbot	TO DO
› ✓	SP-43	Dark/Light Mode Toggle	TO DO
› ✓	SP-44	Integrate Database to Support Multiple S...	TO DO
› ✓	SP-45	Transform Admin Functionalities into a Ve...	TO DO
› ✓	SP-46	Create Root Admin Panel for Managing Ve...	TO DO
› ✓	SP-47	Pricing Model Management	TO DO
› ✓	SP-48	Subscription Management for Vendors	TO DO
› ✓	SP-49	Platform Refinement and Bug Fixes	TO DO

4. Ceremonies:

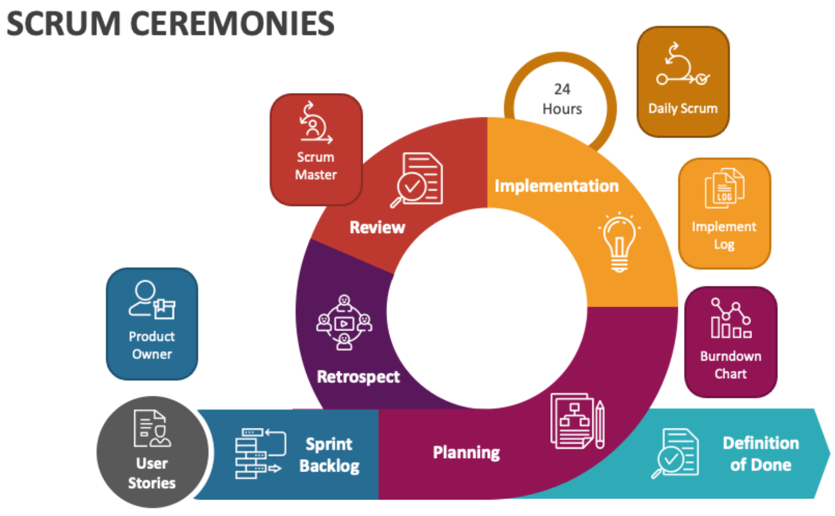
Scrum ceremonies are the structured events that keep the team synchronized and the project moving forward:

Sprint Planning: Sprint Planning meetings were conducted at the start of each sprint. During these meetings, the team and the Product Owner discussed the user stories in the Product Backlog, selected the stories to be implemented in the sprint, and set a clear sprint goal. This ceremony ensured that the team had a shared understanding of what needed to be achieved during the sprint.

Daily Stand-ups: The Daily Stand-up was a short meeting held every day, where each team member briefly discussed what they worked on the previous day, what they planned to work on that day, and any obstacles they were facing. This ceremony helped in identifying issues early and keeping the entire team aligned.

Sprint Review: At the end of each sprint, the team held a Sprint Review meeting to demonstrate the completed work to the Product Owner, the professor, and other stakeholders. This review provided an opportunity for stakeholders to give feedback on the increment, which could then be incorporated into the Product Backlog for future sprints.

Sprint Retrospective: The Sprint Retrospective was a reflection meeting held after the Sprint Review, where the team discussed what went well, what could have been better, and what actions could be taken to improve in the next sprint. This ceremony was crucial for continuous improvement and learning.



Why We Chose Scrum:

The decision to use Scrum for this project was driven by several key factors:

Complexity of the Project: The project involved not only the development of a traditional e-commerce platform but also the expansion into a multi-vendor marketplace, which introduced additional layers of complexity. Scrum's iterative nature was ideal for managing this complexity, allowing us to focus on one set of features at a time, review the progress, and adjust the plan as needed.

Need for Flexibility: Given the evolving nature of the project, particularly after receiving feedback from our professor, Scrum's flexibility allowed us to adapt to changes in requirements without derailing the entire project. This adaptability was crucial as we expanded the platform's scope mid-project.

Focus on Collaboration: Scrum emphasizes collaboration, both within the team and with stakeholders. Regular Scrum ceremonies ensured that the entire team was aligned, and the iterative reviews with our professor provided opportunities to incorporate feedback early and often. This collaborative environment was essential for ensuring that the final product met the expectations of all stakeholders.

Continuous Delivery and Improvement: The goal of delivering functional increments at the end of each sprint aligned with our need to have a working product at various stages of development. This approach not only helped in tracking progress but also in identifying potential issues early, allowing for continuous improvement throughout the project.

Managing Risk: By breaking the project into smaller sprints, we were able to identify and mitigate risks incrementally. This was particularly important in a project of this scale, where unforeseen challenges could arise at any stage.

Workflow

The project was broken down into several sprints, each focused on delivering specific features or enhancements. The workflow involved the following steps:

Sprint Initiation: Each sprint began with a Sprint Planning meeting where the sprint goal was defined, and user stories were selected from the Product Backlog to be included in the Sprint Backlog.

Daily Development: Throughout the sprint, the team worked on the tasks defined in the Sprint Backlog, with progress tracked during Daily Stand-ups. The team focused on delivering a fully functional increment by the end of the sprint.

Sprint Conclusion: At the end of the sprint, a Sprint Review was held to demonstrate the completed increment, followed by a Sprint Retrospective to reflect on the process and plan for improvements in the next sprint.

Marketplace Expansion: After the initial six sprints, which focused on building the core e-commerce platform, two additional sprints were added to integrate marketplace functionalities. These final sprints introduced the vendor integration, root admin panel, and pricing model, transforming the platform into a multi-vendor marketplace.

Software Requirements Analysis

The Software Requirements Analysis phase is crucial for ensuring that the project meets its intended objectives. This analysis involves a detailed examination of the system's functional and non-functional requirements, providing a clear roadmap for development. For our marketplace platform, these requirements are outlined as follows:

Functional Requirements

Functional requirements specify the core functionalities and capabilities that the system must deliver. These are essential to meeting the needs of the users and stakeholders.

1. User Authentication and Authorization

- **Description:** The system must provide secure mechanisms for user authentication and authorization, catering to different roles such as buyers, sellers, and administrators.
- **Details:**
    - **User Registration and Login:** Users can create accounts and log in using secure authentication protocols.
    - **Role-Based Access:** The system distinguishes between roles (e.g., buyer, seller, admin) and grants access based on these roles.
    - **Profile Management:** Users can manage personal information, update profiles, and view account history.

2. Product Management (Admin and Vendor Panels)

- **Description:** The system must enable both administrators and sellers to manage product listings, including creating, updating, and deleting products.
- **Details:**
    - **Admin Product Management:** Administrators can manage all product listings across the platform, including adding new products, updating details, and removing products.
    - **Vendor Product Management:** Sellers have access to their own vendor panel, where they can manage their product listings independently.
    - **Product Details:** Each product listing must include essential information such as name, description, price, images, and stock levels.

3. Payment Processing

- **Description:** The system must facilitate secure payment transactions, supporting multiple payment methods and ensuring proper fund distribution to sellers.
- **Details:**
    - **Payment Gateway Integration:** The platform integrates with third-party payment gateways (e.g., Stripe, l) for secure transaction processing.
    - **Multiple Payment Options:** Users can choose from various payment methods, including stripe and cash on delivery.

4. Marketplace Functionality

- **Description:** The system must support multi-vendor operations, allowing various sellers to list products and enabling buyers to make purchases from multiple vendors.
- **Details:**
    - **Multi-Vendor Cart:** Users can add items from different sellers to a single shopping cart and checkout seamlessly.
    - **Order Management:** The system tracks orders from placement to delivery, providing updates to both buyers and sellers.

5. Notification and Mailing Service

- **Description:** The system must notify users of important events, such as order confirmations and shipping updates, and send promotional emails.
- **Details:**
    - **Order Notifications:** Automatic emails for order confirmations, shipping updates, and delivery confirmations.
    - **Promotional Emails:** Targeted emails for sales, discounts, and new product arrivals.

6. Multi-Language Translation

- **Description:** The platform must cater to a global audience by supporting multiple languages, allowing users to interact in their preferred language.
- **Details:**
    - **Content Translation:** The system automatically translates product descriptions, user reviews, and other text content.

7. Chatbot Integration

- **Description:** The system must provide a chatbot for customer support, handling common queries and providing assistance.
- **Details:**

    - **Automated Support:** The chatbot answers frequently asked questions and guides users through the platform.

8. Dark/Light Mode Feature

- **Description:** The system must offer an option to switch between dark and light modes, enhancing the user experience.
- **Details:**
    - **Toggle Option:** Users can switch between dark and light modes easily.
    - **Consistent Design:** The platform must maintain consistent styling across both modes.

9. Vendor and Root Admin Panels

- **Description:** The system must include dedicated panels for vendors and a root admin to manage their respective areas of the marketplace.
- **Details:**
    - **Vendor Panel:** Sellers can manage their storefronts, including product listings, orders, and payments.
    - **Root Admin Panel:** The root admin has overarching control of the platform, including managing sellers, overseeing transactions, and configuring system-wide settings.

10. Pricing Model and Subscription Service

- **Description:** The platform must implement a flexible pricing model and subscription service to monetize the marketplace.
- **Details:**
    - **Subscription Tiers:** Sellers can choose from different subscription plans, each offering varying levels of access and benefits.
    - **Pricing Flexibility:** The system allows for different pricing strategies, including discounts, promotional pricing, and dynamic pricing based on demand.

Non-Functional Requirements

Non-functional requirements describe the system’s performance characteristics and operational qualities, ensuring it meets expectations beyond just functionality.

1. Performance

- **Requirement:** The platform must respond to user actions, such as page loading, product searches, and transaction processing, within 2 seconds under typical conditions.
- **Rationale:** Quick performance is essential for user satisfaction and reducing bounce rates.

2. Scalability

- **Requirement:** The platform must scale to handle increasing numbers of users, products, and transactions, especially during peak periods.
- **Rationale:** Scalability is crucial for supporting the platform's growth without sacrificing performance.

3. Security

- **Requirement:** The platform must adhere to security best practices, protecting user data and ensuring secure transactions.
- **Rationale:** Security is vital for building trust and ensuring compliance with regulations.

4. Reliability and Availability

- **Requirement:** The platform must maintain a high level of reliability, with uptime exceeding 99.9% and robust disaster recovery mechanisms in place.
- **Rationale:** Reliable operation is critical for user trust and continuous business operations.

5. Usability

- **Requirement:** The platform’s user interface must be intuitive and accessible to users of varying technical expertise.
- **Rationale:** A user-friendly interface ensures that users can easily navigate the platform and complete transactions.

6. Maintainability

- **Requirement:** The platform’s codebase should be modular and well-documented, allowing for easy updates and maintenance.
- **Rationale:** Maintainability is crucial for the platform’s long-term success, enabling quick adaptation to new requirements.

7. Compliance

- **Requirement:** The platform must comply with relevant legal and regulatory requirements, including data protection laws and e-commerce standards.
- **Rationale:** Compliance ensures that the platform operates within legal boundaries, avoiding fines and protecting its reputation.

Tools and Technologies Used

In the development of our e-commerce clothing platform, a diverse range of tools and technologies were utilized to ensure a robust, scalable, and user-friendly application. Each technology was chosen based on its suitability for the specific tasks and requirements of the project. Below is a detailed report of the key tools and technologies used in this project:

1. For Backend Development

Flask:

- **Description**: Flask is a lightweight and flexible Python web framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.
- **Usage**: Flask was used for developing the RESTful APIs that handle CRUD operations for product management. Its simplicity and flexibility allowed for rapid development and integration with other backend components.

MySQL:

- **Description**: MySQL is an open-source relational database management system.
- **Usage**: MySQL was chosen for its reliability and performance in handling structured data. It was used to store and manage product information, user data, order details, and other essential records.

2. For Frontend Development

Nginx:

- **Description**: Nginx is a high-performance web server that can also be used as a reverse proxy, load balancer, and HTTP cache.
- **Usage**: Nginx was used to serve the frontend of the platform, providing a fast and efficient way to handle HTTP requests and deliver content to users. Its ability to manage high concurrency made it ideal for ensuring smooth user interactions.

Darkmode.js:

- **Description**: Darkmode.js is a lightweight JavaScript library that allows for easy implementation of a dark mode toggle.
- **Usage**: The library was used to implement a dark mode feature, enabling users to switch between light and dark themes seamlessly. It

provided a straightforward interface to detect system preferences and apply the appropriate styles.

3. API Tools

- **Chat-Bot Integration**

  FastBots:

  - **Description**: FastBots is a platform that offers easy-to-setup chat-bot solutions for websites and applications.

  - **Usage**: FastBots was used to integrate a chat-bot into the platform, providing users with real-time assistance and support. The chat-bot was configured to handle common queries, guide users through the site, and improve overall user engagement.

- **Multi-Language Translator**

  GTranslate:

  - **Description**: GTranslate is a powerful website translation widget that allows websites to support multiple languages.

  - **Usage**: The GTranslate widget was integrated to provide multi-language support, making the platform accessible to a global audience. It offered seamless language switching and accurate translations, enhancing user experience for non-English speakers.

- **Email Service**

  Python SMTP Library:

  - **Description**: smtplib is a Python library used for sending emails using the Simple Mail Transfer Protocol (SMTP).

  - **Usage**: The smtplib library was used to implement the email notification service within our platform. By creating a custom API, we ensured that users receive timely updates on account activities, promotions, and order statuses. This approach allowed for greater control over email customization and delivery processes.

- **Payment Processing**

  Stripe API:

  Description: Stripe is a technology company that builds economic infrastructure for the internet, providing payment processing software and application programming interfaces for e-commerce websites and mobile applications.

  Usage: Stripe's robust API was integrated into the platform to handle payment processing. It allowed us to securely manage transactions, support multiple payment methods, and ensure compliance with various financial regulations. The API provided a seamless checkout experience for users, managing payment authorizations efficiently.

4. Containerization and Deployment

Docker:

- **Description**: Docker is a platform for developing, shipping, and running applications in containers. Containers allow developers to package applications with all their dependencies, ensuring consistency across different environments.

- **Usage**: The entire project was containerized using Docker, which facilitated consistent development and deployment environments. Docker allowed us to package the backend (Flask and MySQL), frontend (Nginx), and other services into isolated containers, simplifying the deployment process and enhancing scalability.

In general, the strategic selection and integration of these tools and technologies were important in the successful development of our e-commerce clothing platform. By leveraging Docker for containerization, Flask and MySQL for backend development, Nginx for frontend serving, Darkmode.js for dark mode functionality, FastBots for chat-bot integration, GTranslate for multi-language support, the smtplib library for email notifications, and Stripe for payment processing, we created a robust, scalable, and user-friendly application. These technologies not only met the immediate needs of the project but also provided a strong foundation for future enhancements and scalability.

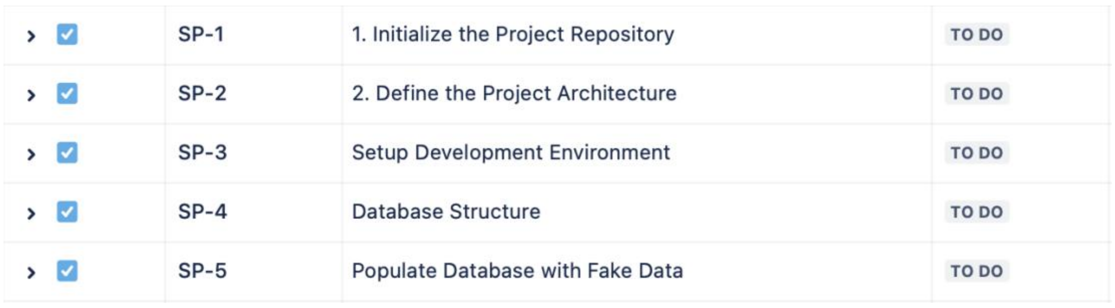
Sprints Overview

1st Sprint : Project Initialization and Core Setup

1. Sprint Planning

Objective:
The primary goal of Sprint 1 was to lay the foundational groundwork for the e-commerce by setting up the project environment, defining its architecture, and establishing the initial database structure.

Sprint 1 backlog (originated from the main product backlog):



› ✓	SP-1	1. Initialize the Project Repository	TO DO
› ✓	SP-2	2. Define the Project Architecture	TO DO
› ✓	SP-3	Setup Development Environment	TO DO
› ✓	SP-4	Database Structure	TO DO
› ✓	SP-5	Populate Database with Fake Data	TO DO

User Stories (from Product Backlog):

1. **As a developers**, we want to initialize a version-controlled repository to manage the project codebase efficiently.
2. **As a developers**, we want to define the project architecture using the MVC pattern to maintain separation of concerns.
3. **As a developers**, we want to set up the development environment to ensure all team members can work consistently.
4. **As a developers**, we want to design the database schema to store and manage all necessary data for the e-commerce platform.
5. **As a developers**, we want to populate the database with fake data for testing to validate the system's core functionalities.

2. Sprint Backlog

Tasks Derived from User Stories:

1. **Initialize the Project Repository:**

    1. Create a GitHub repository.

2. Establish the repository structure.
3. Define a branching strategy.

2. **Define the Project Architecture:**

1. Select and document the MVC architecture.
2. Choose the technology stack (Flask for backend, Nginx for frontend, MySQL for database).

3. Setup Development Environment:

1. Install necessary tools and dependencies.
2. Configure Docker for environment consistency.
3. Set up Python virtual environments.
4. Document the setup process.

4. Database Structure:

1. Create an Entity-Relationship (ER) diagram.
2. Implement database tables in MySQL.
3. Ensure data normalization and indexing.

5. Populate Database with Fake Data:

1. Write a script to generate fake data using the Faker library.
2. Populate the database with users, products, orders and etc.
3. Validate the integrity and performance of the data.

3. Development

Execution:

Project Repository Initialization:
The team created the GitHub repository and set up the directory structure, which facilitated organized code management. The branching strategy was implemented, allowing parallel development efforts.

Architecture Definition:
The MVC pattern was documented, and Flask was selected for the backend due to its flexibility and community support. Nginx was chosen for frontend management, while MySQL was selected for its reliability and relational characterstics.

Environment Setup:
Docker containers were configured, ensuring that all team members could work in identical environments.

Database Structure:
The team designed an ER diagram to visualize the relationships between entities such as Users, Products, Orders and etc. The schema was implemented in SQL, ensuring efficient data storage. Indexes were created on frequently queried columns to optimize performance.

Populating Fake Data:
The Faker library was used to generate realistic test data. This data was then populated into the database, providing a reliable dataset for subsequent testing and development.

4. Testing

Approach:

Initial Validation:
After setting up the database, the team performed initial validations to ensure that the data was correctly structured and that relationships between entities were maintained.

Data Integrity Checks:
Ensured that all foreign key relationships were intact and that there were no orphaned records. The team also tested CRUD operations (Create, Read, Update, Delete) on the database to confirm its readiness.

Performance Testing:
Basic performance tests were conducted using the populated fake data to ensure the database could handle typical queries without significant lag or bottlenecks.

Outcome:

- The testing phase confirmed that the project’s foundational elements were correctly implemented and that the system could support the upcoming development phases.

5. Sprint Review

Summary:

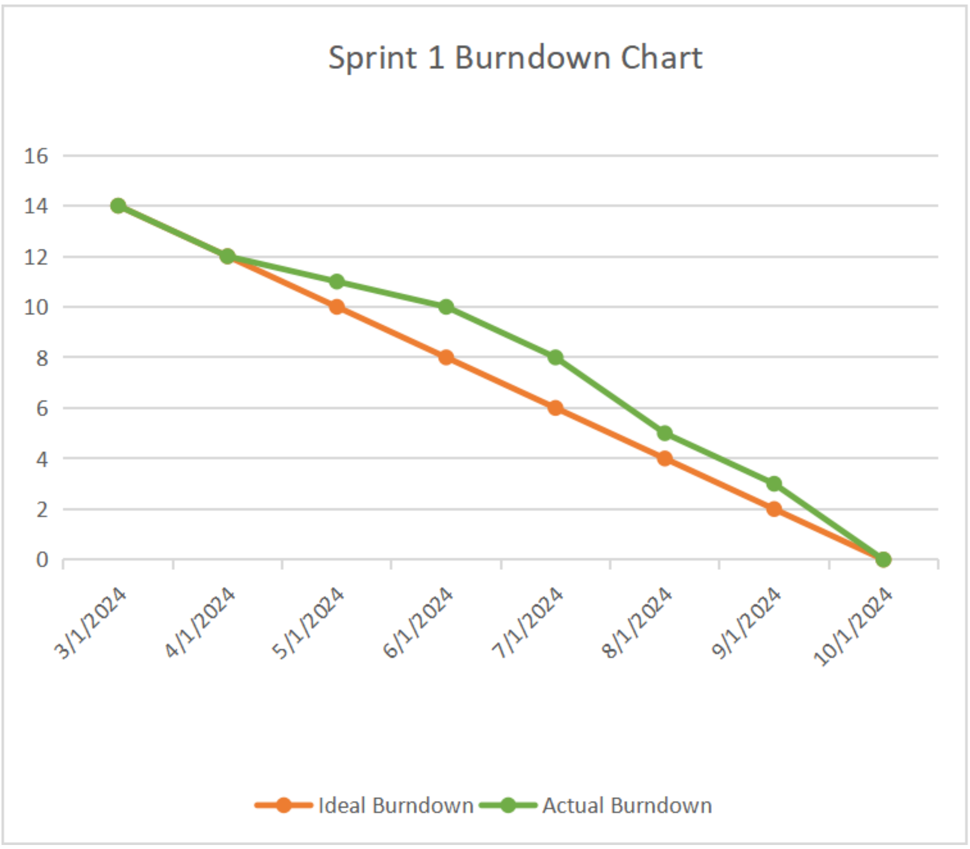
Achievements:
All tasks planned for Sprint 1 were completed successfully. The project repository was initialized, the architecture was defined, the environment was

set up, the database structure was implemented, and test data was populated. This established a strong foundation for future development.

Demonstration:
The team demonstrated the repository structure, the ER diagram, and the populated database during the Sprint Review meeting. The demonstration highlighted the successful setup and readiness of the project to proceed with further development.

Feedback:
Feedback from the team members was positive, with suggestions to enhance the documentation of the database schema for easier understanding and future modifications.

6. Sprint Retrospective



Burndown Chart Analysis:

Burndown Chart Overview:

- **Ideal Line:** Represented the expected steady progress throughout the sprint.
- **Actual Line:** Showed some initial delays due to environment setup issues but caught up towards the end.

Analysis:

- **Initial Phase:** Progress was steady during the first few days, with tasks like repository setup and architecture definition proceeding as planned.
- **Mid-Sprint:** The team encountered challenges with Docker configurations, leading to a slight dip in progress.
- **End Phase:** After resolving the configuration issues, the team accelerated and completed the remaining tasks on time.

Retrospective Discussion:

Successes:
The team successfully set up the foundational elements of the project, with Docker and virtual environments proving effective in standardizing the development environment.

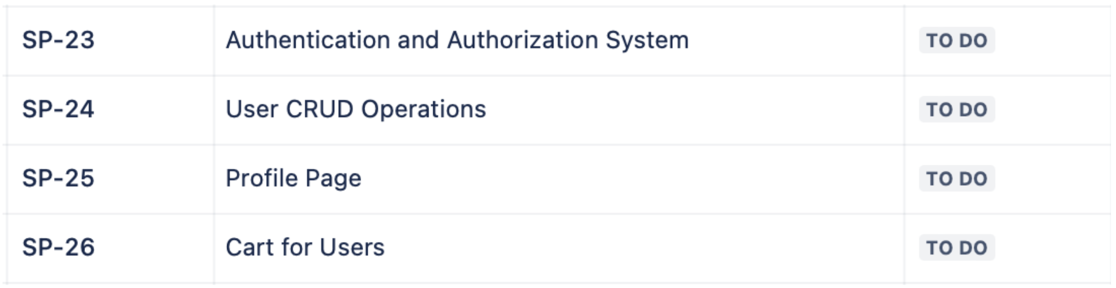
Areas for Improvement:

- **Documentation:** Enhance the detail of the database schema documentation.
- **Environment Setup:** Improve the initial setup guide to prevent delays in future sprints.

Action Items:

- Update and expand documentation for the database schema.
- Refine the setup guide to include troubleshooting steps for Docker and virtual environments.

7. Product Backlog Refinement



SP-23	Authentication and Authorization System	TO DO
SP-24	User CRUD Operations	TO DO
SP-25	Profile Page	TO DO
SP-26	Cart for Users	TO DO

New User Stories:

- Based on the successful completion of Sprint 1, new user stories were added to the product backlog for Sprint 2. These included tasks related to implementing user authentication, product management features, and other core e-commerce functionalities.

Refinement Activities:

- The product backlog was reviewed and updated, with user stories being prioritized based on the project's roadmap and the feedback received during the Sprint Review.

Outcome:

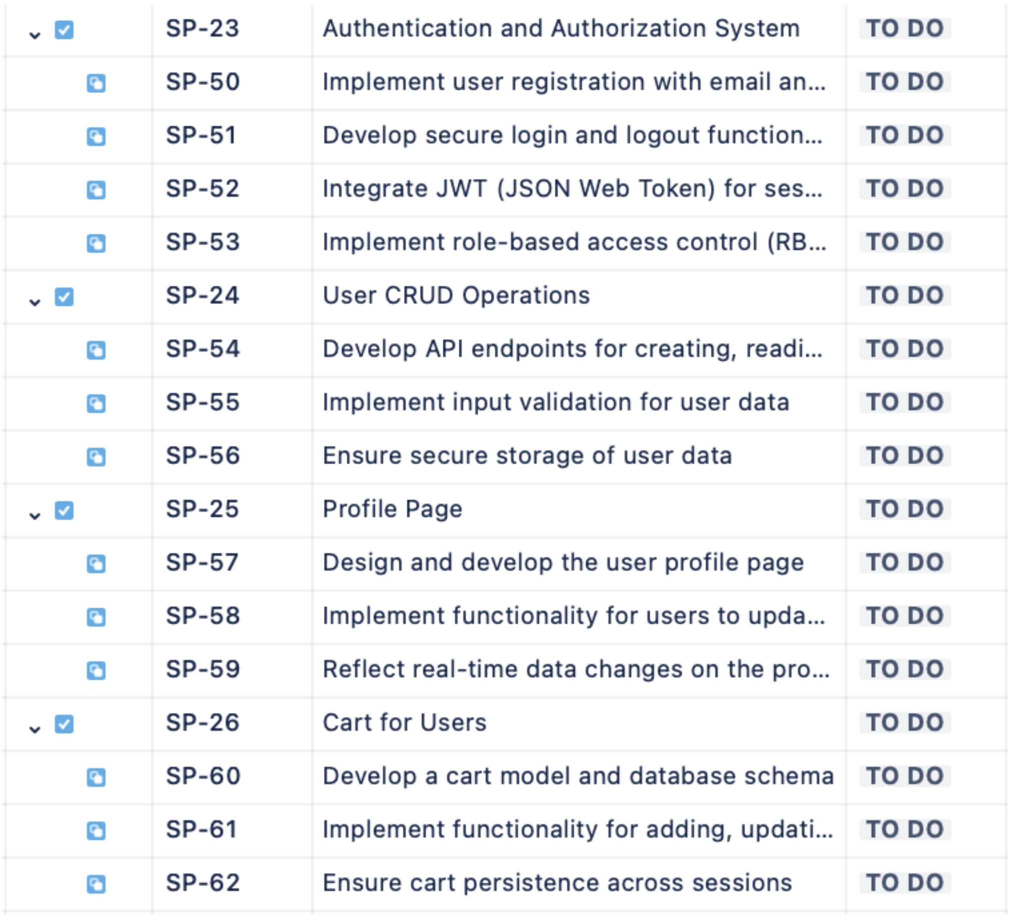
- The refined product backlog served as the input for Sprint 2, ensuring that the project continued to progress in alignment with its goals.

2nd Sprint : Implementation of Authentication and Authorization System, User CRUD Operations, Profile Page, and Cart for Users

1. Sprint Plannings

Objective:
The goal of Sprint 2 was to build the core user-related functionalities, focusing on secure authentication, user management, profile handling, and shopping cart capabilities.

Sprint 2 backlog (originated from the main product backlog):



⌄ ✓	SP-23	Authentication and Authorization System	TO DO
	SP-50	Implement user registration with email an...	TO DO
	SP-51	Develop secure login and logout function...	TO DO
	SP-52	Integrate JWT (JSON Web Token) for ses...	TO DO
	SP-53	Implement role-based access control (RB...	TO DO
⌄ ✓	SP-24	User CRUD Operations	TO DO
	SP-54	Develop API endpoints for creating, readi...	TO DO
	SP-55	Implement input validation for user data	TO DO
	SP-56	Ensure secure storage of user data	TO DO
⌄ ✓	SP-25	Profile Page	TO DO
	SP-57	Design and develop the user profile page	TO DO
	SP-58	Implement functionality for users to upda...	TO DO
	SP-59	Reflect real-time data changes on the pro...	TO DO
⌄ ✓	SP-26	Cart for Users	TO DO
	SP-60	Develop a cart model and database schema	TO DO
	SP-61	Implement functionality for adding, updati...	TO DO
	SP-62	Ensure cart persistence across sessions	TO DO

User Stories (from Product Backlog):

User Story 1: Authentication and Authorization System

As a user, I want to create an account and log in securely so that I can access the platform’s features.

User Story 2: User CRUD Operations

As an admin, I want to perform CRUD operations on user accounts so that I can manage the platform effectively.

User Story 3: Profile Page

As a user, I want to manage my personal profile so that I can update my information as needed.

User Story 4: Cart for Users

As a user, I want to add products to my cart so that I can purchase them later.

Sprint Goal:
Implement and integrate authentication and authorization systems, user CRUD operations, user profile page, and shopping cart functionality to enhance the user experience and ensure secure access.

2. Sprint Backlog

Tasks Derived from User Stories:

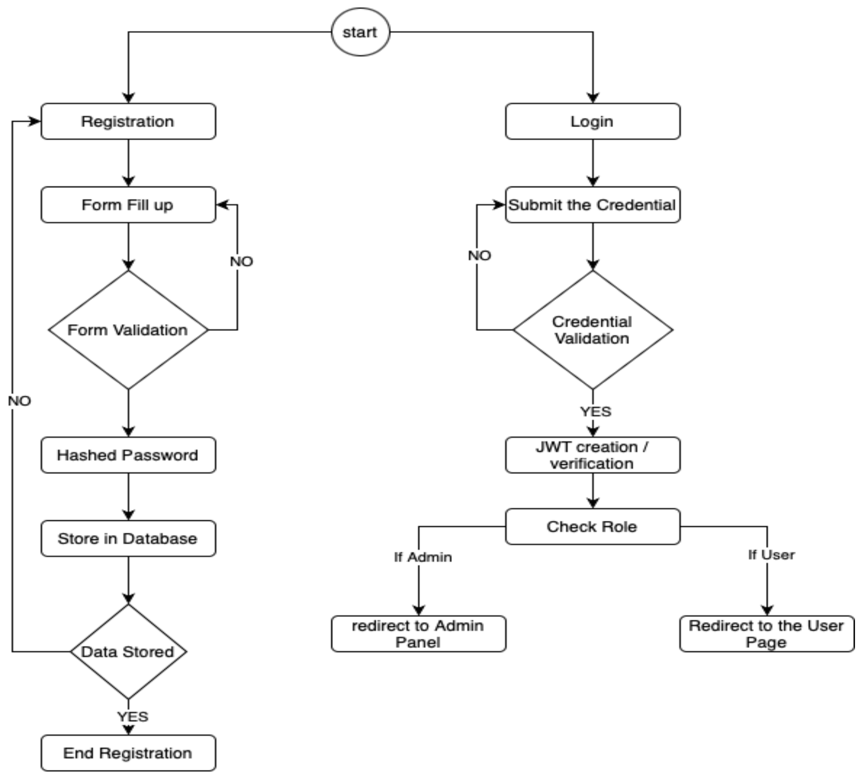
User Story 1: Authentication and Authorization System

1. Implement user registration with email and password.
2. Develop secure login and logout functionality.
3. Integrate JWT (JSON Web Token) for session management.
4. Implement role-based access control (RBAC).

Acceptance Criteria:

1. Users can register with a valid email and password.
2. Users can log in and receive a JWT token.
3. Access to certain endpoints is restricted based on user roles.
4. Users can log out, invalidating their session token.

Activity Diagram :



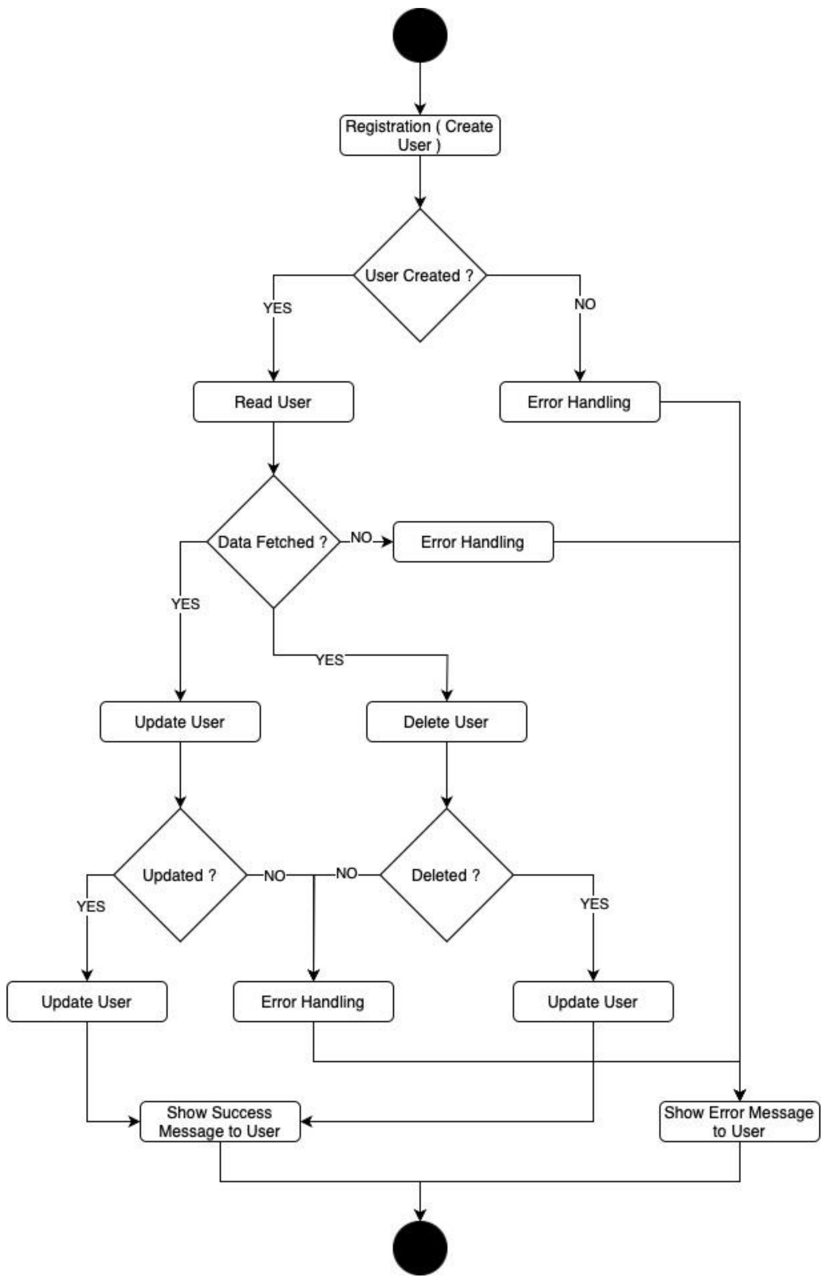
User Story 2: User CRUD Operations

1. Develop API endpoints for creating, reading, updating, and deleting user accounts.
2. Implement input validation for user data.
3. Ensure secure storage of user data.

Acceptance Criteria:

1. Admins can create, read, update, and delete user accounts.
2. The system rejects invalid input (e.g., incorrect email format).
3. Passwords are stored securely using encryption.

Activity Diagram :



User Story 3: Profile Page

1. Design and develop the user profile page.
2. Implement functionality for users to update personal information.
3. Reflect real-time data changes on the profile page.

Acceptance Criteria:

1. Users can view and update their profile information.
2. Updates are saved and reflected immediately on the profile page.

3. Data validation ensures accurate input (e.g., phone number format).

User Story 4: Cart for Users

1. Develop a cart model and database schema.
2. Implement functionality for adding, updating, and removing products in the cart.
3. Ensure cart persistence across sessions.

Acceptance Criteria:

1. Users can add items to their cart.
2. Users can update quantities or remove items from their cart.
3. The cart contents persist even if the user logs out and back in.

3. Development

Execution:

User Story 1: Authentication and Authorization System:

- Implemented secure user registration and login with Flask, incorporating JWT for session management.
- Role-Based Access Control (RBAC) was introduced, ensuring only authorized users could access certain endpoints.

User Story 2: User CRUD Operations:

- Developed the CRUD API endpoints with secure data handling and validation.
- Integrated robust input validation to ensure data integrity.

User Story 3: Profile Page:

- Designed a user-friendly profile page where users could update personal details, which were immediately reflected across the system.

User Story 4: Cart for Users:

- Created a cart model that allowed users to add, update, and remove items, with data persistence ensuring cart contents were saved across sessions.

4. Testing

Approach:

User Story 1: Authentication and Authorization System:

- Unit tests ensured the authentication system handled various scenarios, such as invalid credentials.
- Security testing was conducted to prevent vulnerabilities like unauthorized access.

User Story 2: User CRUD Operations:

- API testing validated the CRUD operations, ensuring data accuracy and security.
- Tests confirmed that user input was properly validated and stored.

User Story 3: Profile Page:

- Functional testing ensured users could seamlessly update their profiles with real-time feedback.
- UI/UX testing ensured the profile page was intuitive and responsive.

User Story 4: Cart for Users:

- Integration testing ensured all cart operations worked together smoothly.
- Performance testing validated that the cart could handle various operations efficiently.

Outcome:

- All features met their respective acceptance criteria, with successful test results across all user stories. No critical issues were identified during the testing phase.

5. Sprint Review

Summary:

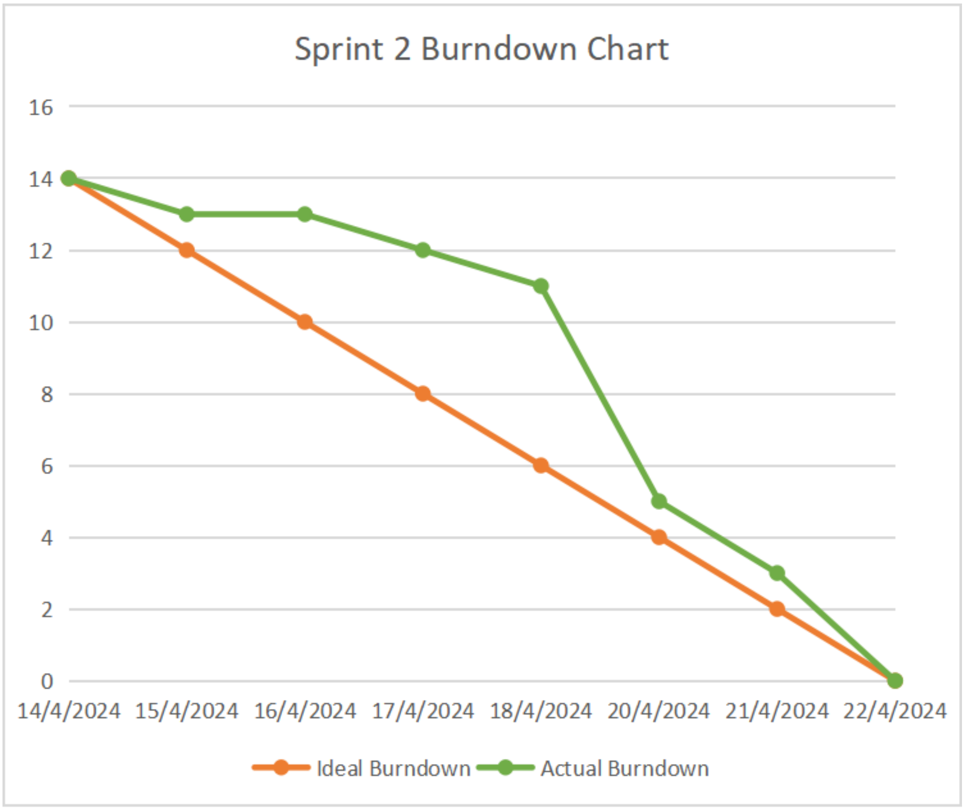
Achievements:
The team delivered the authentication system, user management features, profile page, and cart functionality on time and within scope. These were demonstrated in the Sprint Review.

Demonstration:
Each user story was demonstrated, showcasing the working authentication system, CRUD operations, profile management, and cart functionality. Stakeholders, including the professor, provided positive feedback.

Feedback:
The feedback emphasized minor UI improvements and suggested adding a profile picture upload feature in a future sprint.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:** Progress was steady in the early part of the sprint.
- **Mid-Sprint Challenges:** The team encountered challenges during the integration of JWT, but these were resolved quickly.
- **Final Phase:** The sprint concluded with all tasks completed, meeting the sprint goal.

Retrospective Discussion:

Successes:
The sprint was successful, with all features implemented and tested as planned.

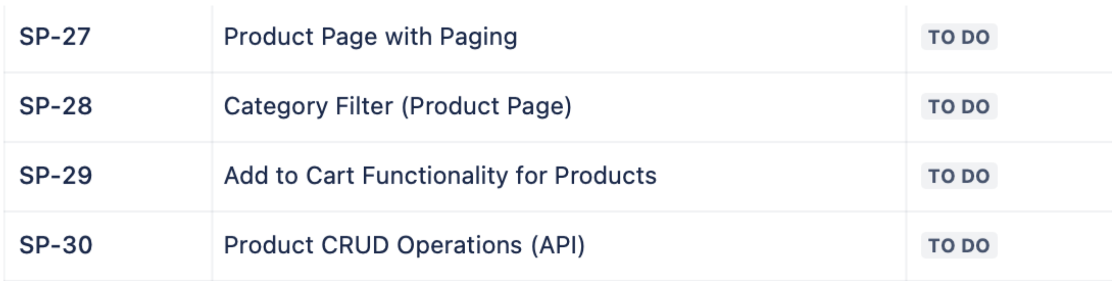
Areas for Improvement:

- Enhance documentation for JWT integration.
- Focus on iterative UI improvements based on feedback.

Action Items:

- Plan for UI enhancements in future sprints.
- Improve JWT-related documentation for the development team.

7. Product Backlog Refinement



SP-27	Product Page with Paging	TO DO
SP-28	Category Filter (Product Page)	TO DO
SP-29	Add to Cart Functionality for Products	TO DO
SP-30	Product CRUD Operations (API)	TO DO

New User Stories:

- Based on feedback, new user stories were added for the next sprint, including the enhancements to the cart and payment processing.

Refinement Activities:

- The product backlog was reviewed and prioritized for Sprint 3, focusing on payment processing and additional user experience improvements.

Outcome:

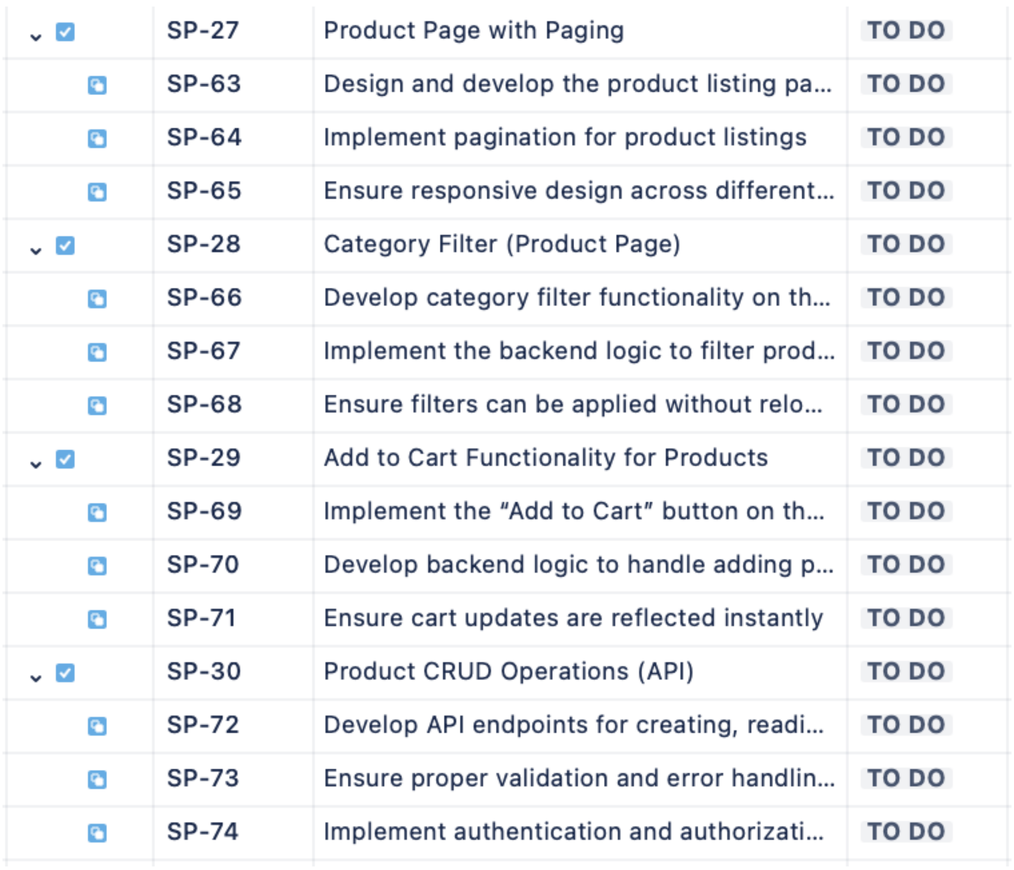
- The product backlog was refined, setting a clear path for Sprint 3, ensuring continued progress toward project goals.

3rd Sprint : Implementation of Product Page with Paging, Category Filter, Add to Cart Functionality, and Product CRUD Operations

1. Sprint Planning

Objective:
The goal of Sprint 3 was to enhance the user experience by developing a dynamic product page with paging and filtering capabilities, integrate the "Add to Cart" functionality, and establish robust Product CRUD operations via API.

Sprint 3 backlog (originated from the main product backlog):



⌄ ☑	SP-27	Product Page with Paging	TO DO
	SP-63	Design and develop the product listing pa...	TO DO
	SP-64	Implement pagination for product listings	TO DO
	SP-65	Ensure responsive design across different...	TO DO
⌄ ☑	SP-28	Category Filter (Product Page)	TO DO
	SP-66	Develop category filter functionality on th...	TO DO
	SP-67	Implement the backend logic to filter prod...	TO DO
	SP-68	Ensure filters can be applied without relo...	TO DO
⌄ ☑	SP-29	Add to Cart Functionality for Products	TO DO
	SP-69	Implement the "Add to Cart" button on th...	TO DO
	SP-70	Develop backend logic to handle adding p...	TO DO
	SP-71	Ensure cart updates are reflected instantly	TO DO
⌄ ☑	SP-30	Product CRUD Operations (API)	TO DO
	SP-72	Develop API endpoints for creating, readi...	TO DO
	SP-73	Ensure proper validation and error handlin...	TO DO
	SP-74	Implement authentication and authorizati...	TO DO

User Stories (from Product Backlog):

User Story 1: Product Page with Paging

As a user, I want to view products on a paginated interface so that I can easily navigate through large product lists.

User Story 2: Category Filter (Product Page)

As a user, I want to filter products by categories so that I can quickly find the items I'm interested in.

User Story 3: Add to Cart Functionality for Products

As a user, I want to add products to my shopping cart from the product page so that I can purchase them later.

User Story 4: Product CRUD Operations (API)

As an admin, I want to manage product listings via an API so that I can maintain an up-to-date product catalog.

Sprint Goal:
To deliver a fully functional product page with paging and filtering, enable users to add products to their cart, and provide admins with the ability to manage product listings through a RESTful API.

2. Sprint Backlog

Tasks Derived from User Stories:

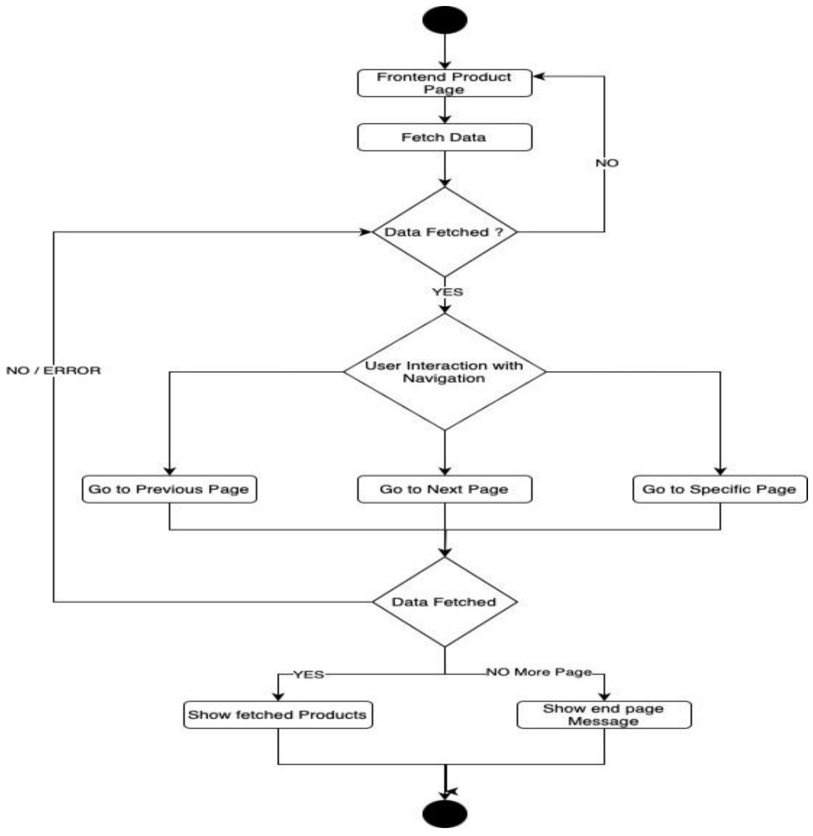
User Story 1: Product Page with Paging

1. Design and develop the product listing page.
2. Implement pagination for product listings.
3. Ensure responsive design across different devices.

Acceptance Criteria:

1. Products are displayed in a paginated format.
2. Users can navigate through pages easily.
3. The product page is responsive and accessible on all devices.

Activity Diagram :



User Story 2: Category Filter (Product Page)

1. Develop category filter functionality on the product page.
2. Implement the backend logic to filter products by category.
3. Ensure filters can be applied without reloading the page.

Acceptance Criteria:

1. Users can filter products by category.
2. The filtered results are displayed instantly without page reload.
3. The category filter is easy to use and intuitive.

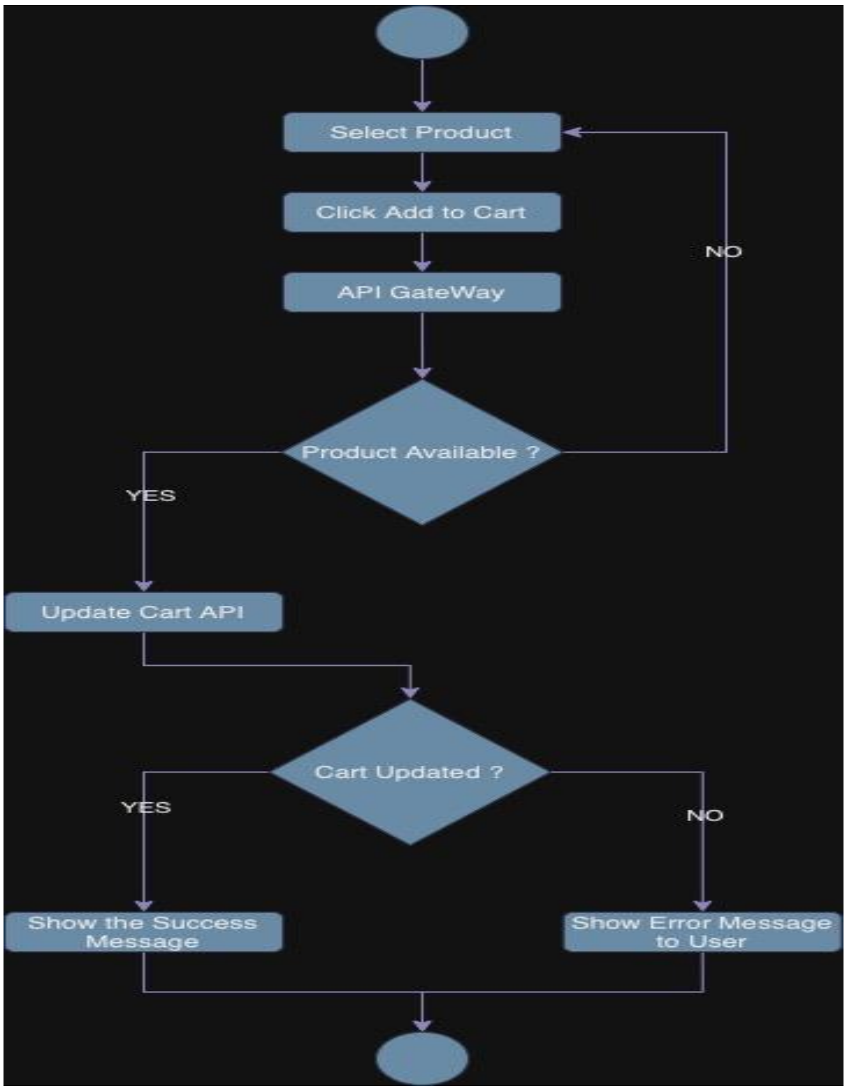
User Story 3: Add to Cart Functionality for Products

1. Implement the “Add to Cart” button on the product page.
2. Develop backend logic to handle adding products to the user’s cart.
3. Ensure cart updates are reflected instantly.

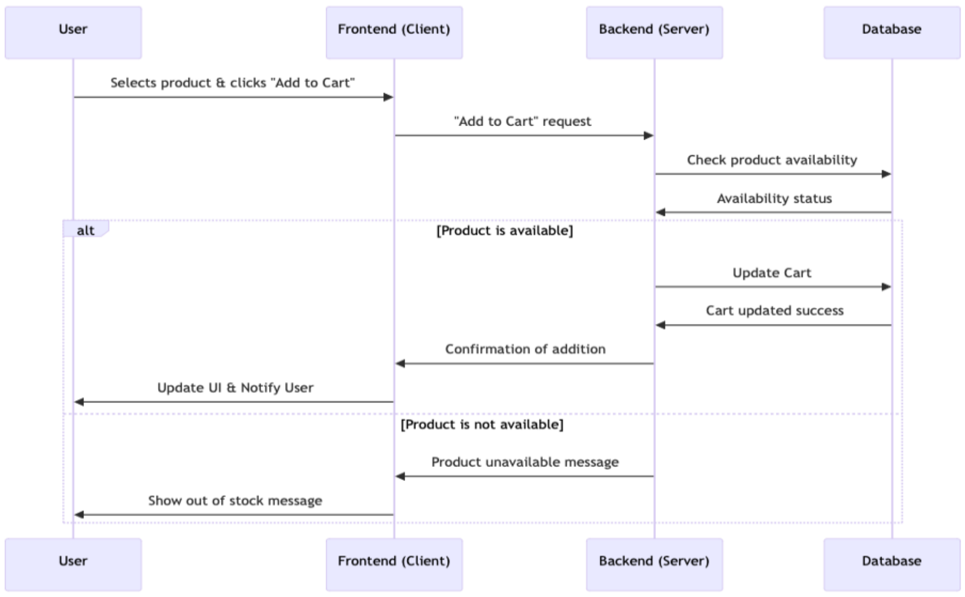
Acceptance Criteria:

1. Users can add products to their cart directly from the product page.
2. The cart updates automatically, showing the newly added items.
3. Users receive a confirmation that an item has been added to their cart.

Activity Diagram :



Sequence Diagram :



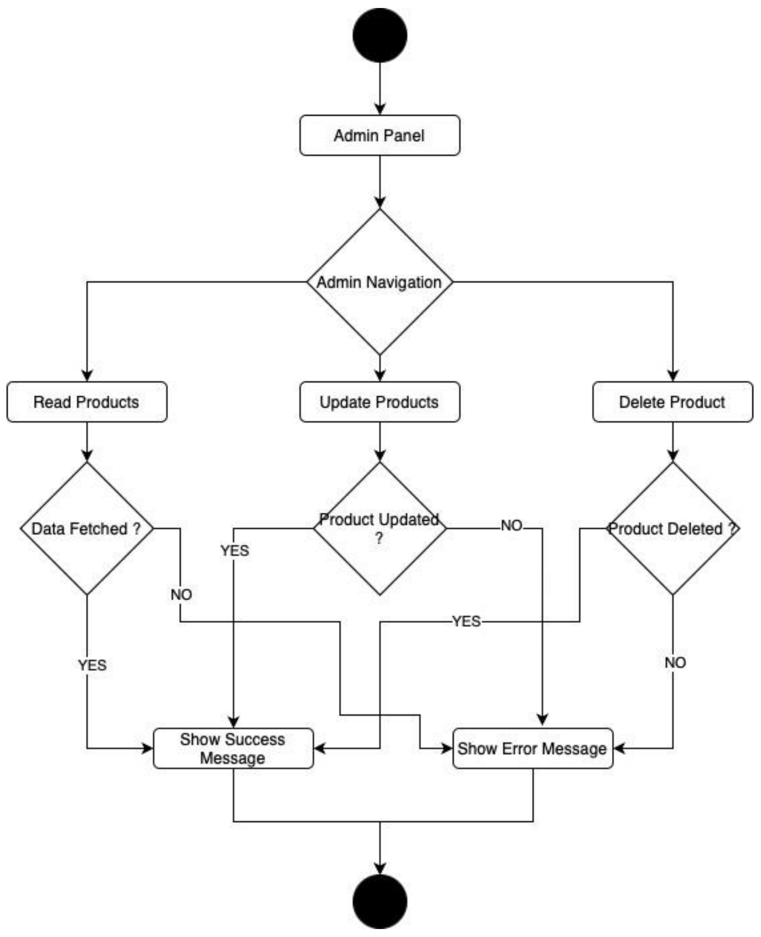
User Story 4: Product CRUD Operations (API)

1. Develop API endpoints for creating, reading, updating, and deleting products.
2. Ensure proper validation and error handling for product data.
3. Implement authentication and authorization checks for API access.

Acceptance Criteria:

1. Admins can perform CRUD operations on products via the API.
2. API responses are accurate and provide meaningful feedback.
3. Unauthorized users are prevented from accessing the API.

Activity Diagram :



3. Development

Execution:

User Story 1: Product Page with Paging:

- The product page was designed and developed with a clean, paginated layout to handle large inventories.
- Pagination was implemented using Flask’s backend logic and dynamic front-end updates, ensuring smooth user experience across all devices.

User Story 2: Category Filter (Product Page):

- A category filter was integrated into the product page, allowing users to refine their searches. The filter logic was efficiently handled on the server-side. The front-end was designed to be intuitive, with a collapsible menu for category selection.

User Story 3: Add to Cart Functionality for Products:

- The “Add to Cart” button was added to each product on the product page. When clicked, products were immediately added to the user’s cart, and the cart was updated without refreshing the page.
- The functionality was integrated with the existing cart system, ensuring that the user’s cart persisted across sessions.

User Story 4: Product CRUD Operations (API):

- RESTful API endpoints were created for product management, enabling admins to add, update, delete, and retrieve product information.
- Authentication checks ensured that only authorized users could perform CRUD operations, with detailed validation ensuring data integrity.

4. Testing

Approach:

User Story 1: Product Page with Paging:

- Functional testing ensured that pagination worked correctly across different product counts and that the design was responsive.
- Load testing was conducted to ensure the system handled large numbers of products without performance degradation.

User Story 2: Category Filter (Product Page):

- Unit testing focused on the correctness of filter logic.
- Usability testing was conducted to ensure the filtering experience was smooth and intuitive.

User Story 3: Add to Cart Functionality for Products:

- Integration testing ensured that adding products to the cart functioned correctly in conjunction with existing cart and payment systems.
- User acceptance testing (UAT) confirmed that users found the cart interaction seamless.

User Story 4: Product CRUD Operations (API):

- API testing verified that all CRUD operations functioned as expected, with proper data validation and error handling.

    - Security testing ensured that only authorized users could access and manipulate product data.

Outcome:

- All features passed their respective tests, with no critical bugs or issues identified. The sprint successfully delivered a fully functional product page with paging and filtering, integrated cart functionality, and robust product management via API.

5. Sprint Review

Summary:

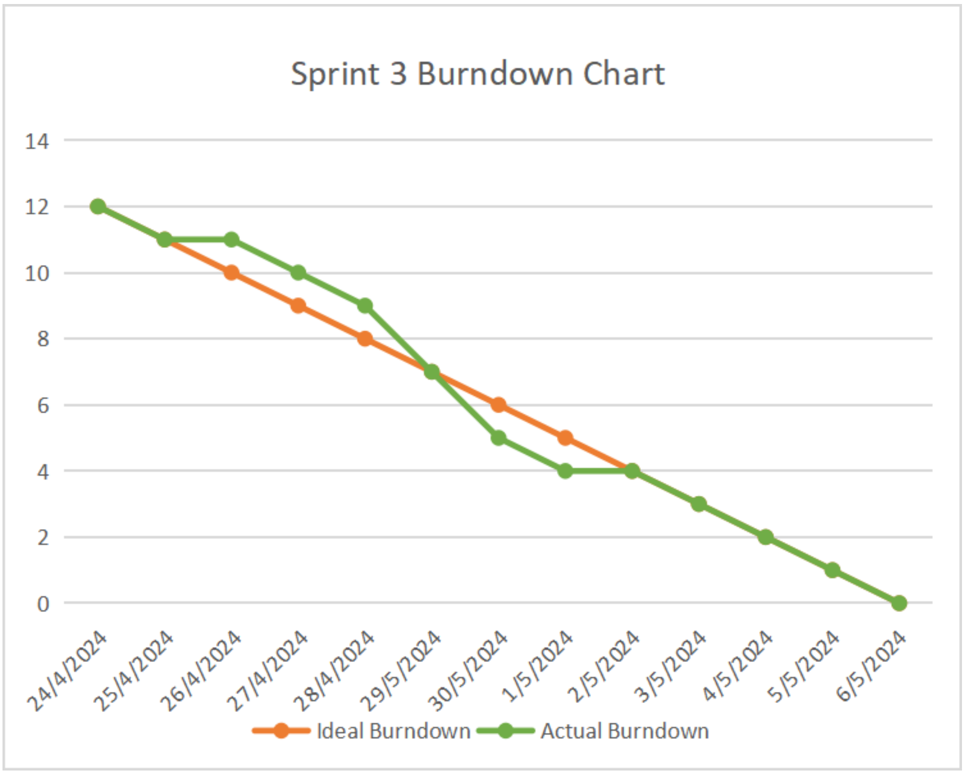
Achievements:
The team successfully developed a dynamic product page with paging and filtering capabilities, implemented the "Add to Cart" functionality, and established comprehensive CRUD operations for product management.

Demonstration:
Each user story was demonstrated to stakeholders, highlighting the new features and their seamless integration with existing functionalities.

Feedback:
Positive feedback was received, especially regarding the usability of the product page and the speed of the "Add to Cart" functionality. Minor suggestions were made for enhancing the category filter UI.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:** The team maintained a steady pace, completing the design and early development tasks on time.
- **Mid-Sprint Challenges:** Some challenges arose in ensuring the category filter worked seamlessly with paging, but these were resolved quickly.
- **Final Phase:** The sprint concluded successfully with all tasks completed and the sprint goal achieved.

Retrospective Discussion:

Successes:
The sprint was highly productive, with all planned features delivered on time. The integration of paging and filtering was particularly well-received.

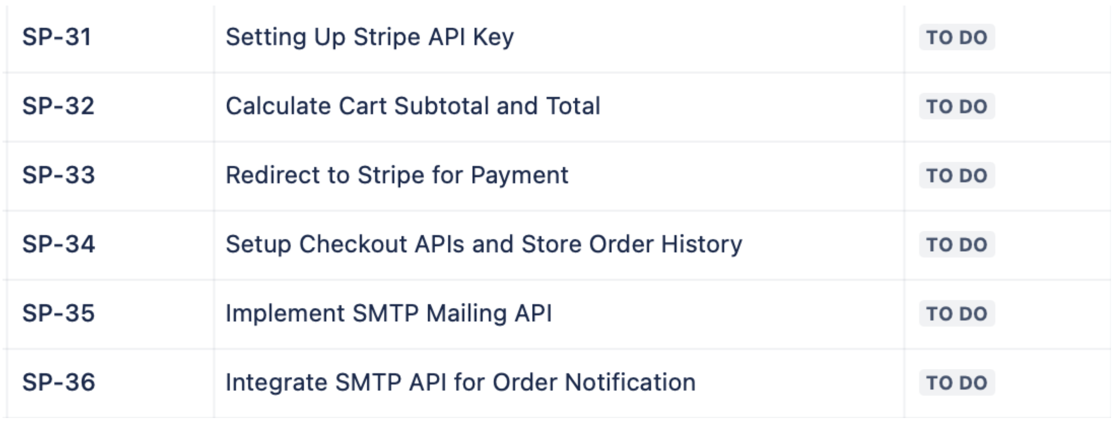
Areas for Improvement:

- Additional focus is needed on UI refinements to improve user experience further.
- Better initial estimation of tasks related to UI/UX could help in more accurate sprint planning.

Action Items:

- Plan for UI improvements in future sprints based on user feedback.
- Enhance collaboration between front-end and back-end teams to streamline the development process for features like filtering and paging.

7. Product Backlog Refinement



SP-31	Setting Up Stripe API Key	TO DO
SP-32	Calculate Cart Subtotal and Total	TO DO
SP-33	Redirect to Stripe for Payment	TO DO
SP-34	Setup Checkout APIs and Store Order History	TO DO
SP-35	Implement SMTP Mailing API	TO DO
SP-36	Integrate SMTP API for Order Notification	TO DO

New User Stories:

- Based on feedback, new user stories were added for the next sprint, including UI improvements for the category filter and enhancements to the product detail page.

Refinement Activities:

- The product backlog was reviewed, and priorities were adjusted to focus on refining the user experience in the next sprint, alongside the introduction of advanced features like payment integration.

Outcome:

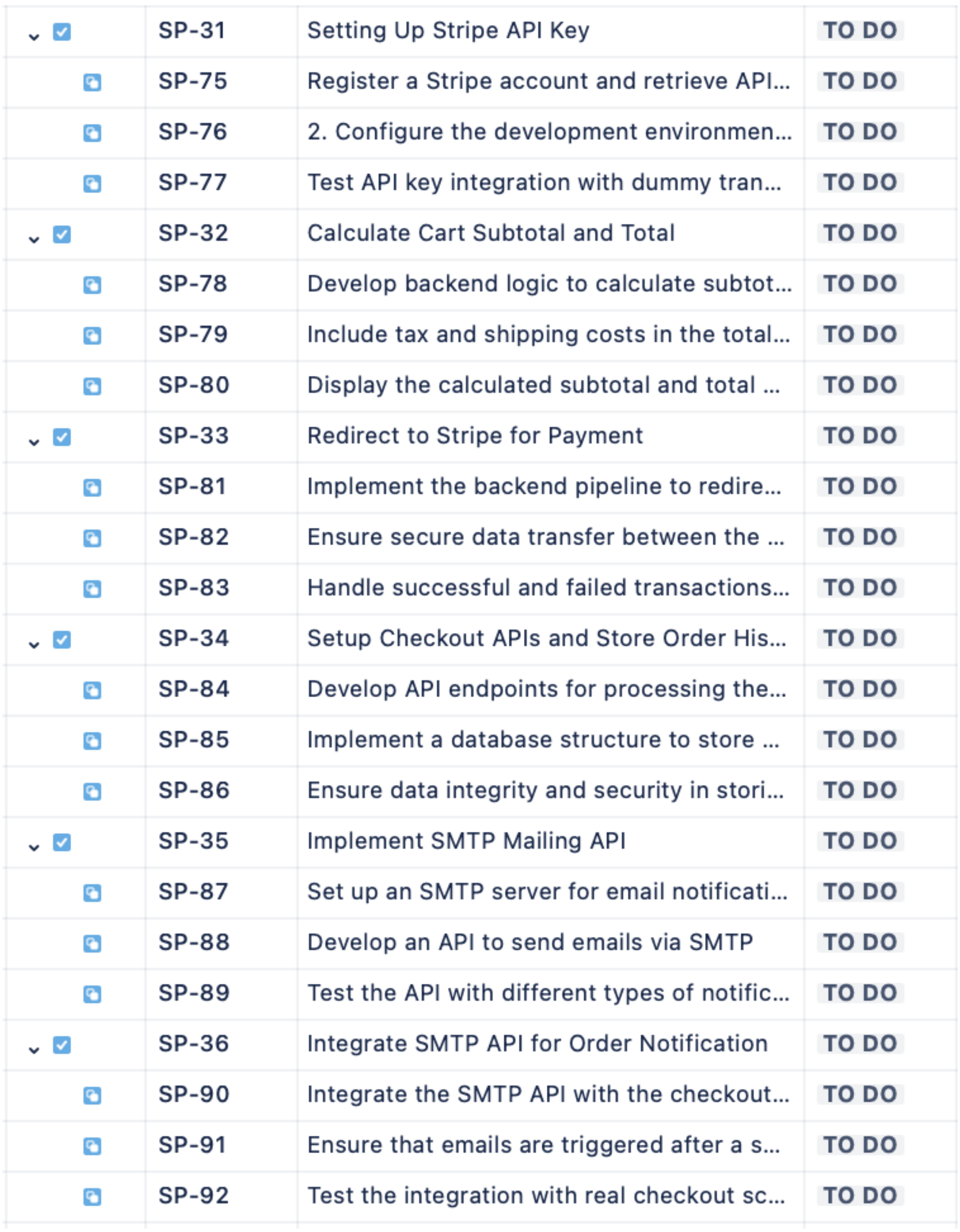
- The product backlog was updated and refined, providing a clear direction for Sprint 4, with a focus on continuing to improve the platform’s core functionalities and user interface.

4th Sprint : Integration of Stripe Payment System, Order Calculation, Checkout Process, and Notification System

1. Sprint Planning

Objective:
The focus of Sprint 4 was to integrate a secure and efficient payment system using Stripe, calculate cart totals for users, set up the entire checkout process, and implement an SMTP-based mailing system to notify users of their order status.

Sprint 3 backlog (originated from the main product backlog):



⌄ ☑	SP-31	Setting Up Stripe API Key	TO DO
	SP-75	Register a Stripe account and retrieve API...	TO DO
	SP-76	2. Configure the development environmen...	TO DO
	SP-77	Test API key integration with dummy tran...	TO DO
⌄ ☑	SP-32	Calculate Cart Subtotal and Total	TO DO
	SP-78	Develop backend logic to calculate subtot...	TO DO
	SP-79	Include tax and shipping costs in the total...	TO DO
	SP-80	Display the calculated subtotal and total ...	TO DO
⌄ ☑	SP-33	Redirect to Stripe for Payment	TO DO
	SP-81	Implement the backend pipeline to redire...	TO DO
	SP-82	Ensure secure data transfer between the ...	TO DO
	SP-83	Handle successful and failed transactions...	TO DO
⌄ ☑	SP-34	Setup Checkout APIs and Store Order His...	TO DO
	SP-84	Develop API endpoints for processing the...	TO DO
	SP-85	Implement a database structure to store ...	TO DO
	SP-86	Ensure data integrity and security in stori...	TO DO
⌄ ☑	SP-35	Implement SMTP Mailing API	TO DO
	SP-87	Set up an SMTP server for email notificati...	TO DO
	SP-88	Develop an API to send emails via SMTP	TO DO
	SP-89	Test the API with different types of notific...	TO DO
⌄ ☑	SP-36	Integrate SMTP API for Order Notification	TO DO
	SP-90	Integrate the SMTP API with the checkout...	TO DO
	SP-91	Ensure that emails are triggered after a s...	TO DO
	SP-92	Test the integration with real checkout sc...	TO DO

User Stories (from Product Backlog):

User Story 1: Setting Up Stripe API Key

As a developer, I want to set up the Stripe API so that the platform can securely process payments.

User Story 2: Calculate Cart Subtotal and Total

As a user, I want to see the subtotal and total of my cart before checking out so that I can review the final cost.

User Story 3: Redirect to Stripe for Payment

As a user, I want to be redirected to Stripe for payment processing so that I can securely complete my purchase.

User Story 4: Setup Checkout APIs and Store Order History

As a user, I want my order details to be securely stored so that I can view my order history later.

User Story 5: Implement SMTP Mailing API

As a developer, I want to create an API to send notifications via SMTP so that users receive order confirmations.

User Story 6: Integrate SMTP API for Order Notification

As a user, I want to receive a summary of my checkout via email so that I have a record of my purchase.

Sprint Goal:
To integrate the Stripe payment system, calculate and display cart totals, handle checkout processing, and implement email notifications for order summaries.

2. Sprint Backlog

Tasks Derived from User Stories:

User Story 1: Setting Up Stripe API Key

1. Register a Stripe account and retrieve API keys.
2. Configure the development environment to securely store and use Stripe API keys.
3. Test API key integration with dummy transactions.

Acceptance Criteria:

1. Stripe API keys are securely stored and functional.
2. The platform can connect to Stripe without errors.
3. Dummy transactions are processed successfully.

User Story 2: Calculate Cart Subtotal and Total

1. Develop backend logic to calculate subtotal based on cart items.
2. Include tax and shipping costs in the total calculation.
3. Display the calculated subtotal and total on the checkout page.

Acceptance Criteria:

1. Subtotal and total amounts are accurately calculated.
2. Calculations include applicable taxes and shipping fees.
3. Users can view these amounts before proceeding to payment.

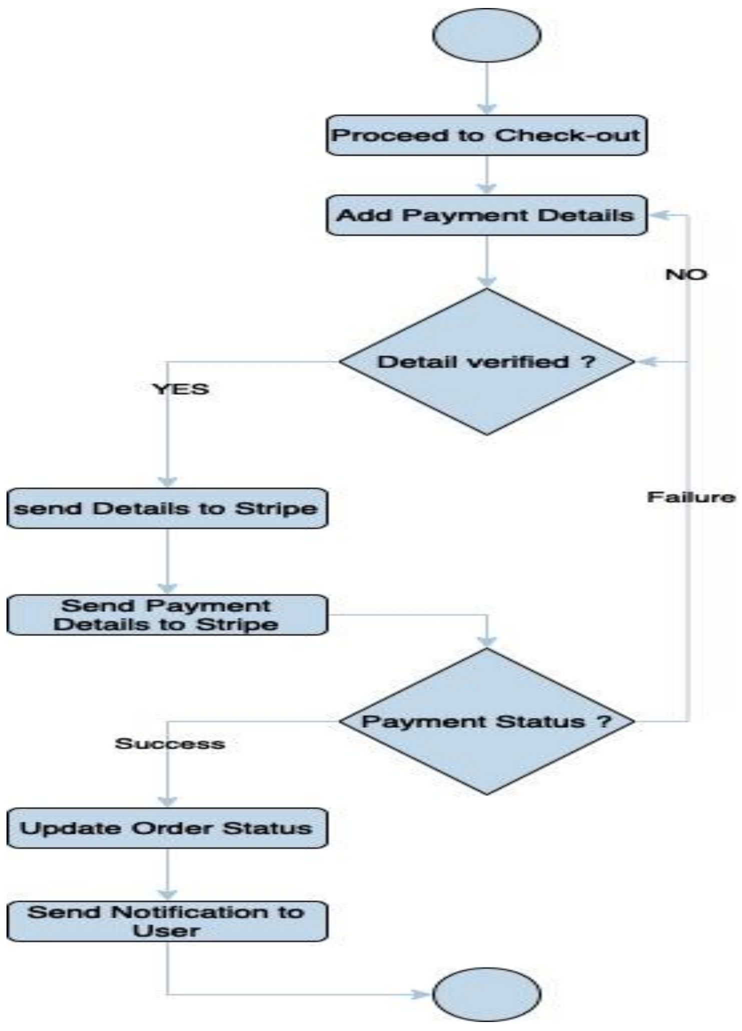
User Story 3: Redirect to Stripe for Payment

1. Implement the backend pipeline to redirect users to Stripe's payment gateway.
2. Ensure secure data transfer between the platform and Stripe.
3. Handle successful and failed transactions, redirecting users accordingly.

Acceptance Criteria:

1. Users are securely redirected to Stripe for payment.
2. Successful payments are confirmed, and users are redirected to an order confirmation page.
3. Failed payments prompt an error message and allow users to retry.

Activity Diagram :



User Story 4: Setup Checkout APIs and Store Order History

1. Develop API endpoints for processing the checkout and storing order details.
2. Implement a database structure to store order history.
3. Ensure data integrity and security in storing user order information.

Acceptance Criteria:

1. Checkout API processes payments and stores order data securely.
2. Users can view their order history after checkout.
3. Order data is stored and retrieved efficiently.

User Story 5: Implement SMTP Mailing API

1. Set up an SMTP server for email notifications.
2. Develop an API to send emails via SMTP.

3. Test the API with different types of notifications (e.g., order confirmation).

Acceptance Criteria:

1. Emails are sent successfully using the SMTP API.
2. Notifications include all necessary details and are formatted correctly.
3. The API is secure and scalable.

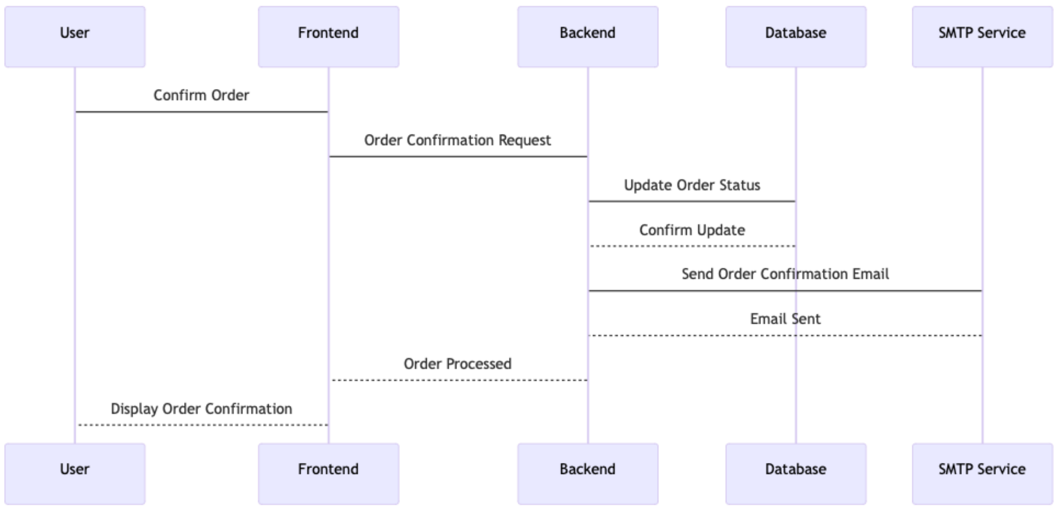
User Story 6: Integrate SMTP API for Order Notification

1. Integrate the SMTP API with the checkout process to send order summaries.
2. Ensure that emails are triggered after a successful payment.
3. Test the integration with real checkout scenarios.

Acceptance Criteria:

1. Users receive order confirmation emails after checkout.
2. Emails contain accurate order details, including subtotal, total, and items purchased.
3. The integration is seamless and reliable.

Sequence Diagram :



3. Development

Execution:

User Story 1: Setting Up Stripe API Key:

- The team registered a Stripe account, retrieved API keys, and securely configured them in the development environment using environment variables. This ensured that sensitive information was not exposed in the codebase.

User Story 2: Calculate Cart Subtotal and Total:

- The backend logic for calculating the cart's subtotal, including tax and shipping costs, was developed. The frontend was updated to display these amounts on the checkout page, providing users with a clear view of their purchase costs before payment.

User Story 3: Redirect to Stripe for Payment:

- A secure backend pipeline was implemented to redirect users to Stripe's payment gateway. This included handling session tokens and ensuring data passed between the platform and Stripe was encrypted. Upon successful payment, users were redirected to a confirmation page, while failed transactions prompted error messages.

User Story 4: Setup Checkout APIs and Store Order History:

- API endpoints were developed to handle checkout processing, including payment confirmation and order storage. A new database schema was created to store order history, allowing users to view past orders. The system was tested to ensure data integrity and security.

User Story 5: Implement SMTP Mailing API:

- An SMTP server was configured and an API was developed to send out email notifications. The API was tested with various scenarios, including order confirmations and account-related notifications, to ensure reliability.

User Story 6: Integrate SMTP API for Order Notification:

- The SMTP API was integrated into the checkout process, ensuring that users received a confirmation email immediately after a successful checkout. The email included a detailed summary of the order, including the subtotal, total, and list of purchased items.

4. Testing

Approach:

User Story 1: Setting Up Stripe API Key:

  - Connection tests were conducted to ensure the platform could securely interact with Stripe. Dummy transactions were processed to verify that the API keys were functioning correctly.

User Story 2: Calculate Cart Subtotal and Total:

  - Unit testing was performed on the calculation logic to ensure accuracy across various scenarios, including different tax rates and shipping options.

User Story 3: Redirect to Stripe for Payment:

  - Integration testing confirmed that users were correctly redirected to Stripe for payment. Security testing was also conducted to ensure data encryption between the platform and Stripe.

User Story 4: Setup Checkout APIs and Store Order History:

  - API testing ensured that checkout processes completed correctly and that order data was stored securely in the database. Data retrieval from the order history was also tested.

User Story 5: Implement SMTP Mailing API:

  - The SMTP API was tested across multiple scenarios, ensuring that emails were sent promptly and contained the correct information.

User Story 6: Integrate SMTP API for Order Notification:

  - End-to-end testing was conducted to confirm that order confirmation emails were sent immediately after a successful checkout. The content of the emails was verified to ensure accuracy and completeness.

Outcome:

- All features passed their respective tests, with successful integration between the payment system and the notification system. The checkout process was smooth, and users were able to receive detailed email confirmations without any issues.

5. Sprint Review

Summary:

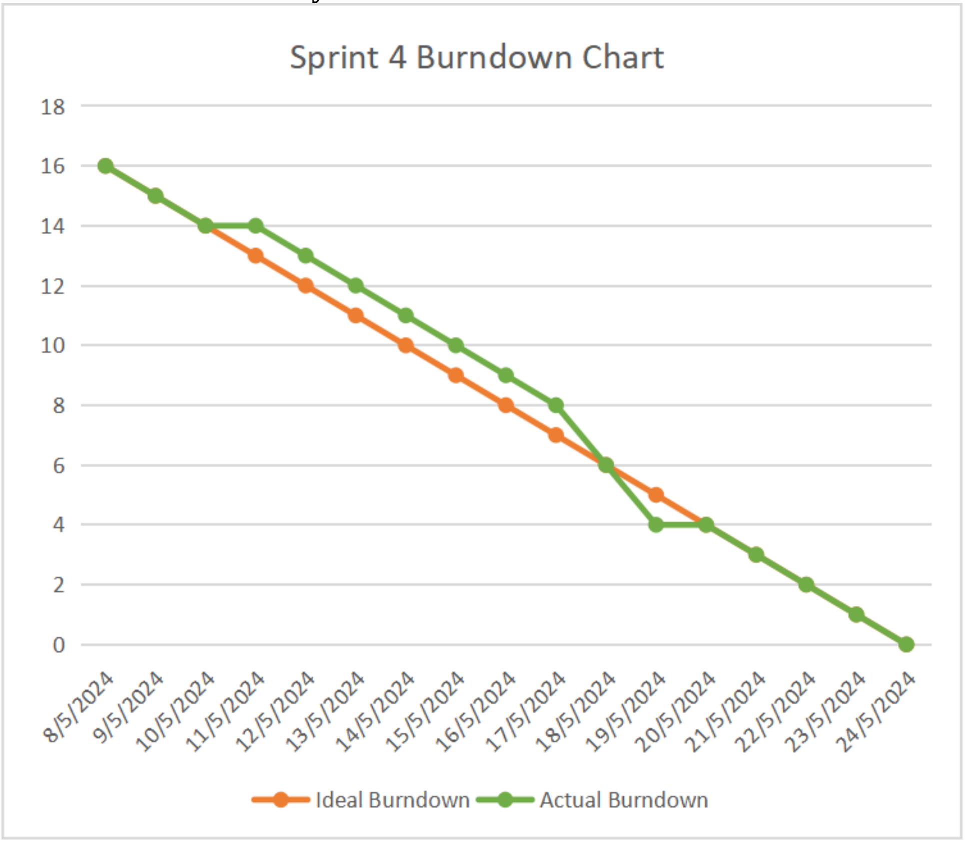
Achievements:
The team successfully integrated the Stripe payment gateway, developed robust checkout and notification systems, and ensured that users received accurate order summaries via email.

Demonstration:
The team demonstrated the entire checkout process from adding items to the cart, calculating totals, processing payments via Stripe, and receiving an order confirmation email.

Feedback:
Stakeholders were pleased with the seamless payment integration and the clarity provided by the order confirmation emails. Minor suggestions were made regarding the layout of the checkout page.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:** The sprint started with a focus on setting up Stripe and calculating totals, with the team maintaining a steady pace.
- **Mid-Sprint Challenges:** The integration of Stripe and ensuring security in data handling took slightly longer than estimated, but these issues were resolved within the sprint.

- ○ **Final Phase:** The sprint concluded successfully with all tasks completed, although some last-minute refinements were made to the email notification system.

Retrospective Discussion:

Successes:
The team successfully integrated third-party services (Stripe and SMTP) into the platform, significantly enhancing the checkout experience.

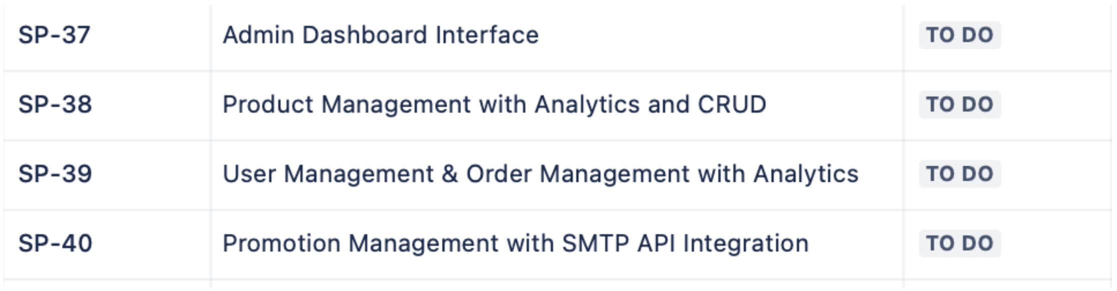
Areas for Improvement:

- ○ Better estimation for tasks involving third-party integrations could help in future sprints.
- ○ More focus on UI/UX testing could have been beneficial, especially for the checkout process.

Action Items:

- Allocate more time for testing and refining UI/UX elements in future sprints.
- Improve documentation for third-party integrations to streamline the process for future developers.

7. Product Backlog Refinement



SP-37	Admin Dashboard Interface	TO DO
SP-38	Product Management with Analytics and CRUD	TO DO
SP-39	User Management & Order Management with Analytics	TO DO
SP-40	Promotion Management with SMTP API Integration	TO DO

New User Stories:

- Based on feedback, new user stories were added for the next sprint, including further UI/UX improvements for the checkout page and exploring alternative payment methods.

Refinement Activities:

- The product backlog was reviewed, and priorities were adjusted to focus on enhancing the user experience during checkout and integrating additional features like order tracking.

Outcome:

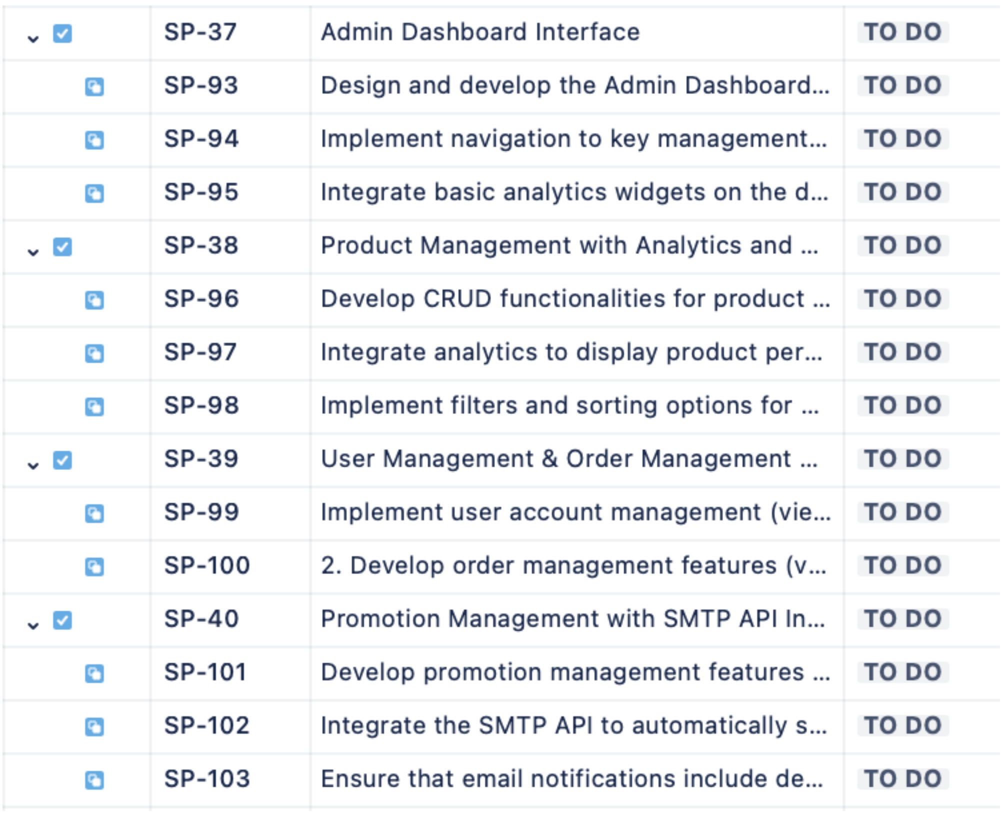
- The product backlog was updated, and the team is well-prepared for Sprint 5, which will focus on refining the checkout experience and introducing new features.

5th Sprint : Development of Admin Dashboard with Product, User, Order, and Promotion Management

1. Sprint Planning

Objective:
The focus of Sprint 5 was to develop a comprehensive Admin Dashboard that would allow administrators to manage products, users, and orders with integrated analytics. Additionally, this sprint aimed to enhance the promotion management system by integrating it with the SMTP API to notify users about new promotions.

Sprint 5 backlog (originated from the main product backlog):



⌄ ✓	SP-37	Admin Dashboard Interface	TO DO
	SP-93	Design and develop the Admin Dashboard...	TO DO
	SP-94	Implement navigation to key management...	TO DO
	SP-95	Integrate basic analytics widgets on the d...	TO DO
⌄ ✓	SP-38	Product Management with Analytics and ...	TO DO
	SP-96	Develop CRUD functionalities for product ...	TO DO
	SP-97	Integrate analytics to display product per...	TO DO
	SP-98	Implement filters and sorting options for ...	TO DO
⌄ ✓	SP-39	User Management & Order Management ...	TO DO
	SP-99	Implement user account management (vie...	TO DO
	SP-100	2. Develop order management features (v...	TO DO
⌄ ✓	SP-40	Promotion Management with SMTP API In...	TO DO
	SP-101	Develop promotion management features ...	TO DO
	SP-102	Integrate the SMTP API to automatically s...	TO DO
	SP-103	Ensure that email notifications include de...	TO DO

User Stories (from Product Backlog):

User Story 1: Admin Dashboard Interface

As an admin, I want a centralized dashboard where I can access key management features (products, users, orders) and view analytics so that I can efficiently oversee platform operations.

User Story 2: Product Management with Analytics and CRUD

As an admin, I need the ability to manage product listings, including creating, reading, updating, and deleting (CRUD) products, while also viewing product performance analytics to make informed decisions.

User Story 3: User Management & Order Management with Analytics

As an admin, I want to manage user accounts and view order history with integrated analytics to monitor user activity and sales performance.

User Story 4: Promotion Management with SMTP API Integration

As an admin, I want to manage promotions and automatically notify users of new promotions via email using the existing SMTP API to enhance user engagement and boost sales.

Sprint Goal:
To build a fully functional Admin Dashboard that allows for comprehensive management of products, users, orders, and promotions, with integrated analytics and automated email notifications for new promotions.

2. Sprint Backlog

Tasks Derived from User Stories:

User Story 1: Admin Dashboard Interface

1. Design and develop the Admin Dashboard layout.
2. Implement navigation to key management sections (Products, Users, Orders, Promotions).
3. Integrate basic analytics widgets on the dashboard (e.g., total sales, total products).

Acceptance Criteria:

1. The dashboard is accessible to admins and provides a clear overview of platform statistics.
2. Navigation to management sections is intuitive and functional.
3. Basic analytics are displayed and update in real-time.

User Story 2: Product Management with Analytics and CRUD

1. Develop CRUD functionalities for product management.
2. Integrate analytics to display product performance (e.g.stock levels).
3. Implement filters and sorting options for product management.

Acceptance Criteria:

1. Admins can create, read, update, and delete products from the dashboard.
2. Product analytics are accurate and displayed in an easily interpretable format.
3. Filters and sorting options are functional and aid in product management.

User Story 3: User Management & Order Management with Analytics

1. Implement user account management (view, edit).
2. Develop order management features (view orders, update order status).
3. Integrate analytics for user activity and order performance

Acceptance Criteria:

1. Admins can manage user accounts, including editing details and deactivating accounts.
2. Order management features allow viewing and updating order statuses.
3. Analytics provide insights into user activity and order performance.

User Story 4: Promotion Management with SMTP API Integration

1. Develop promotion management features (create, update, delete promotions).
2. Integrate the SMTP API to automatically send promotion notifications to users.
3. Ensure that email notifications include details of the promotion and are sent to the correct user segments.

Acceptance Criteria:

1. Admins can manage promotions efficiently through the dashboard.
2. Promotion details are sent to users via email using the SMTP API.
3. Emails are correctly formatted and reach the intended recipients.

3. Development

Execution:

User Story 1: Admin Dashboard Interface

- The team designed the Admin Dashboard with a user-friendly interface, focusing on accessibility and ease of navigation. Key sections such as Product Management, User Management, Order

Management, and Promotion Management were integrated into the dashboard. Basic analytics, including total sales, active users, and recent orders, were displayed in real-time on the dashboard to give admins an immediate overview of platform performance.

User Story 2: Product Management with Analytics and CRUD

- CRUD functionalities for product management were developed, allowing admins to add new products, update existing ones, delete outdated items, and view product listings. The team integrated analytics tools to provide insights into product performance, such as sales trends, stock levels, and product views. Filters and sorting options were added to help admins quickly find and manage products based on various criteria like category, price range, and popularity.

User Story 3: User Management & Order Management with Analytics

- User management features were implemented, allowing admins to view and edit user details, manage user roles, and deactivate accounts if necessary. The order management system was also developed, enabling admins to view all orders, update their statuses, and analyze order trends. Analytics were integrated to highlight key metrics such as the most active users, top-selling products, and the total number of orders processed.

User Story 4: Promotion Management with SMTP API Integration

- The team developed promotion management features that allowed admins to create, update, and delete promotional campaigns. The SMTP API was integrated to automate the process of sending out promotion notifications to users via email. The team ensured that emails were formatted correctly, included all relevant promotion details, and were sent to the correct segments of users.

4. Testing

Approach:

User Story 1: Admin Dashboard Interface

- The Admin Dashboard was tested for usability and functionality, ensuring that all navigation links worked correctly and that real-time analytics were displayed accurately.

User Story 2: Product Management with Analytics and CRUD

- Extensive testing was conducted on the CRUD functionalities, ensuring that products could be managed without any issues. The accuracy of the product analytics was verified against known data sets. Filters and sorting options were also tested to ensure they provided the correct results.

User Story 3: User Management & Order Management with Analytics

- User management features were tested for security and functionality, ensuring that only admins could edit user details or deactivate accounts. Order management features were tested to ensure that orders could be updated and tracked accurately. The analytics integration was tested to ensure that user activity and order performance metrics were accurate.

User Story 4: Promotion Management with SMTP API Integration

- The promotion management features were tested to ensure that promotions could be created, updated, and deleted without issues. The SMTP integration was tested by sending test emails to ensure they were delivered correctly, contained accurate promotion details, and were sent to the appropriate users.

Outcome:

- All functionalities passed their respective tests, with the Admin Dashboard providing a seamless management experience. The integration of analytics was successful, and the SMTP-based promotion notifications were sent without any issues.

5. Sprint Review

Summary:

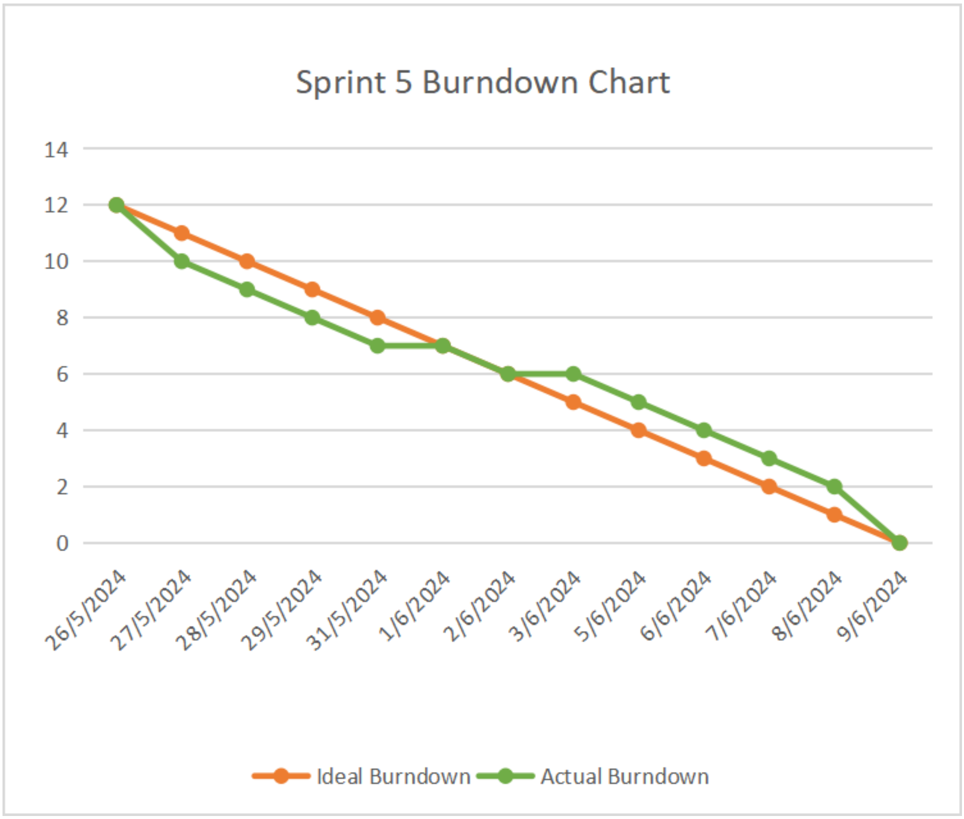
Achievements:
The team successfully developed and deployed a comprehensive Admin Dashboard, complete with product, user, order, and promotion management features. The integration of analytics provided valuable insights for admins, and the SMTP API ensured that users were promptly notified of new promotions.

Demonstration:
The team demonstrated the Admin Dashboard's key features, including the real-time analytics, CRUD operations for product management, user and order management functionalities, and the process of creating and sending out promotion notifications.

Feedback:
Stakeholders were impressed with the functionality and usability of the Admin Dashboard. They particularly appreciated the integrated analytics and the efficiency of the promotion management system. Suggestions were made to enhance the analytics dashboard with more customizable options in future sprints.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:** The sprint started with a focus on the Admin Dashboard design and product management features. The team maintained a consistent pace, tackling the most complex features first.
- **Mid-Sprint Adjustments:** Integrating the SMTP API and ensuring email deliverability took slightly longer than anticipated, but the team adjusted their workload to stay on track.
- **Final Phase:** The sprint concluded successfully with all tasks completed on time, and the dashboard ready for deployment.

Retrospective Discussion:

Successes:
The team effectively developed a robust Admin Dashboard with integrated analytics and efficient management features. The use of real-time analytics and automated email notifications significantly enhanced the platform's functionality.

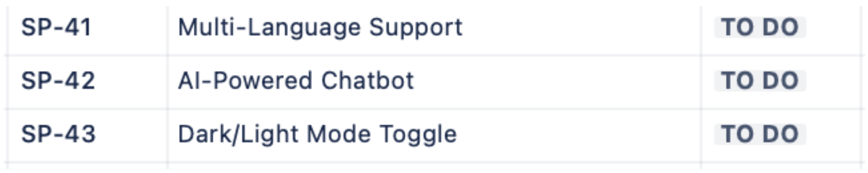
Areas for Improvement:

- There was a slight delay in integrating the SMTP API due to unexpected configuration challenges. Improving the team's familiarity with third-party API configurations could help avoid similar delays in the future.
- More user testing on the Admin Dashboard could provide additional insights into potential usability improvements.

Action Items:

- Provide additional training on third-party API integrations to reduce future delays.
- Increase the focus on user testing, particularly for admin-facing features, to ensure optimal usability.

7. Product Backlog Refinement



SP-41	Multi-Language Support	TO DO
SP-42	AI-Powered Chatbot	TO DO
SP-43	Dark/Light Mode Toggle	TO DO

New User Stories:

- Based on feedback and the successful completion of Sprint 5, new user stories were added to enhance the Admin Dashboard's analytics features, including more customizable data visualizations and the ability to export data.

Refinement Activities:

- The product backlog was reviewed and updated, with new priorities focusing on further refining the admin tools and adding additional analytics capabilities.

Outcome:

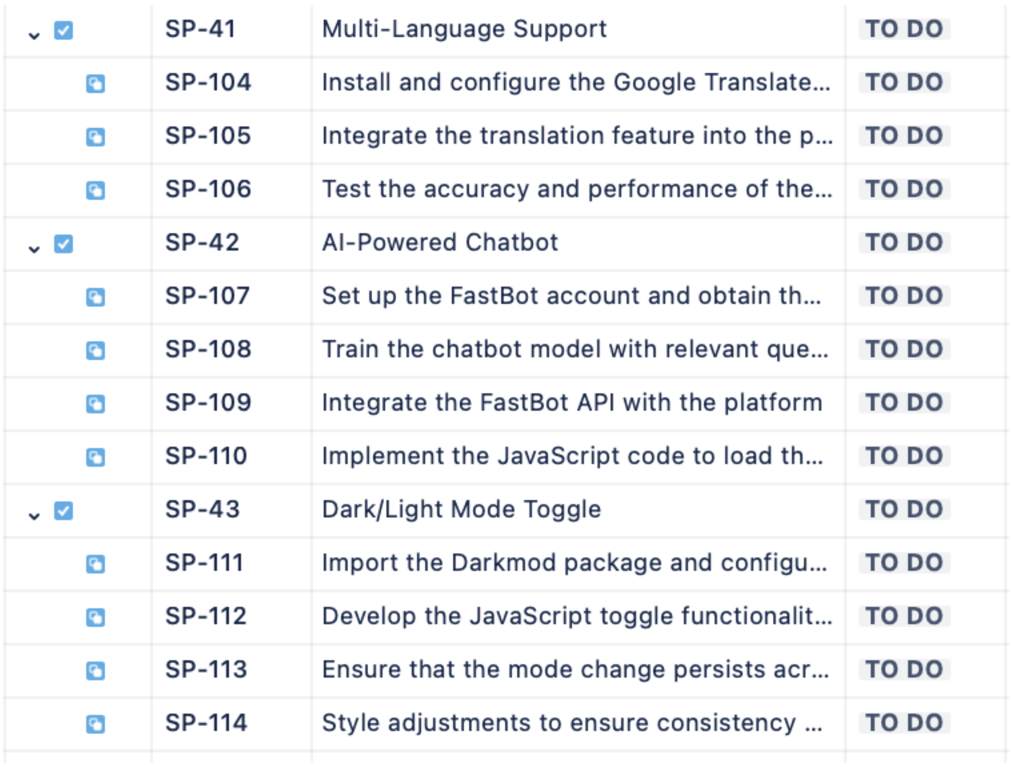
- The product backlog was updated, and the team is well-prepared for Sprint 6, which will focus on enhancing the analytics dashboard and exploring new admin features.

6th Sprint : Enhancing User Experience with Language Translation, AI Chatbot, and Dark/Light Mode

1. Sprint Planning

Objective:
The final sprint of the e-commerce platform focused on enhancing the user experience by integrating multi-language support using Google Translate, implementing an AI-powered chatbot with FastBot, and adding a Dark/Light mode feature for better accessibility. These features aimed to make the platform more user-friendly and accessible to a broader audience.

Sprint 6 backlog (originated from the main product backlog):



⌄ ☑	SP-41	Multi-Language Support	TO DO
	SP-104	Install and configure the Google Translate...	TO DO
	SP-105	Integrate the translation feature into the p...	TO DO
	SP-106	Test the accuracy and performance of the...	TO DO
⌄ ☑	SP-42	AI-Powered Chatbot	TO DO
	SP-107	Set up the FastBot account and obtain th...	TO DO
	SP-108	Train the chatbot model with relevant que...	TO DO
	SP-109	Integrate the FastBot API with the platform	TO DO
	SP-110	Implement the JavaScript code to load th...	TO DO
⌄ ☑	SP-43	Dark/Light Mode Toggle	TO DO
	SP-111	Import the Darkmod package and configu...	TO DO
	SP-112	Develop the JavaScript toggle functionalit...	TO DO
	SP-113	Ensure that the mode change persists acr...	TO DO
	SP-114	Style adjustments to ensure consistency ...	TO DO

User Stories (from Product Backlog):

User Story 1: Multi-Language Support

As a user, I want to view the platform in my preferred language so that I can navigate and understand the content more easily.

User Story 2: AI-Powered Chatbot

As a user, I want to interact with a chatbot that can answer my questions and provide assistance, making my shopping experience more seamless.

User Story 3: Dark/Light Mode Toggle

As a user, I want to switch between Dark and Light modes for a more comfortable viewing experience depending on the time of day and my environment.

Sprint Goal:
To integrate multi-language support, an AI-powered chatbot, and a Dark/Light mode toggle, significantly improving the overall user experience on the platform.

2. Sprint Backlog

Tasks Derived from User Stories:

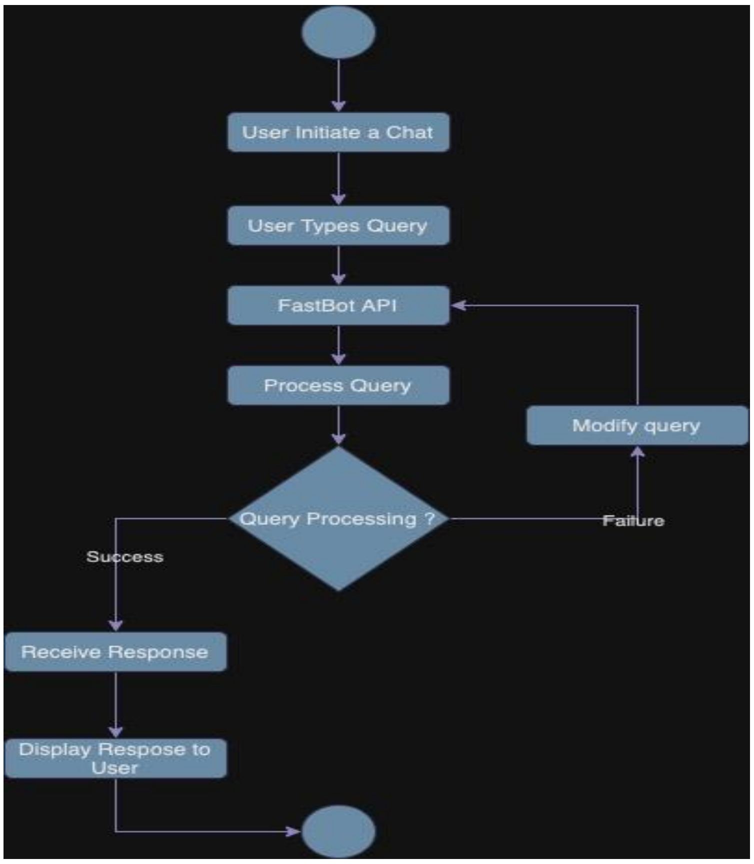
User Story 1: Multi-Language Support

1. Install and configure the Google Translate package.
2. Integrate the translation feature into the platform, allowing users to switch between languages.
3. Test the accuracy and performance of the translation service.

Acceptance Criteria:

1. Users can switch languages easily from a dropdown menu.
2. All major sections of the platform (product descriptions, menus, etc.) are translated accurately.
3. The translation process is fast and does not affect page load times.

Activity Diagram :



User Story 2: AI-Powered Chatbot

1. Set up the FastBot account and obtain the API key.
2. Train the chatbot model with relevant question-answer pairs tailored to the platform's offerings.
3. Integrate the FastBot API with the platform.
4. Implement the JavaScript code to load the chatbot on all pages of the platform.

Acceptance Criteria:

1. The chatbot is accessible on every page and can handle common queries effectively.
2. Responses are relevant and improve with continued use.
3. The chatbot does not interfere with the overall user experience (e.g., it is non-intrusive and easy to access).

User Story 3: Dark/Light Mode Toggle

1. Import the Darkmod package and configure it.
2. Develop the JavaScript toggle functionality for Dark/Light mode.

3. Ensure that the mode change persists across sessions (using cookies or local storage).
4. Style adjustments to ensure consistency across both modes.

Acceptance Criteria:

1. Users can switch between Dark and Light modes easily from a toggle button.
2. The platform's UI remains consistent and readable in both modes.
3. User preference for mode (Dark or Light) is saved and applied on subsequent visits.

3. Development

Execution:

User Story 1: Multi-Language Support

- The Google Translate package was installed and configured. The team integrated the translation feature across the platform, allowing users to select their preferred language from a dropdown menu. Major sections of the platform, such as product descriptions, navigation menus, and user instructions, were made translatable. The team focused on ensuring that the translation process was fast and seamless, so it did not negatively impact the user experience.

User Story 2: AI-Powered Chatbot

- The FastBot account was set up, and the API key was obtained. The team trained the chatbot model using a set of question-answer pairs specific to the platform, such as queries about product availability, shipping policies, return processes, and general information. The chatbot was then integrated into the platform using JavaScript, ensuring that it was accessible from all pages without being obtrusive. The chatbot was designed to improve its responses over time as it interacted with more users.

User Story 3: Dark/Light Mode Toggle

- The Darkmod package was imported, and the team developed the toggle functionality using JavaScript. The toggle button was placed in a prominent location on the platform, allowing users to easily switch between Dark and Light modes. Careful attention was given to styling, ensuring that both modes provided a consistent and comfortable user experience.

4. Testing

Approach:

User Story 1: Multi-Language Support

- The team tested the translation feature by switching between multiple languages and verifying the accuracy of translations across different sections of the platform. Performance testing was also conducted to ensure that translation did not slow down page loads or negatively impact the user experience.

User Story 2: AI-Powered Chatbot

- The chatbot was tested with a variety of user queries to ensure that it responded accurately and appropriately. The team also tested the chatbot's integration across different devices and screen sizes to ensure that it was accessible and functional for all users.

User Story 3: Dark/Light Mode Toggle

- The toggle functionality was tested across various browsers and devices to ensure compatibility and smooth transitions between modes.

Outcome:

- All features passed the testing phase successfully. The translation feature was accurate and quick, the chatbot was responsive and helpful, and the Dark/Light mode toggle provided a smooth and consistent user experience.

5. Sprint Review

Summary:

Achievements:
The sprint successfully enhanced the platform’s user experience by integrating multi-language support, an AI-powered chatbot, and a Dark/Light mode toggle. These features were well-received during the demonstration.

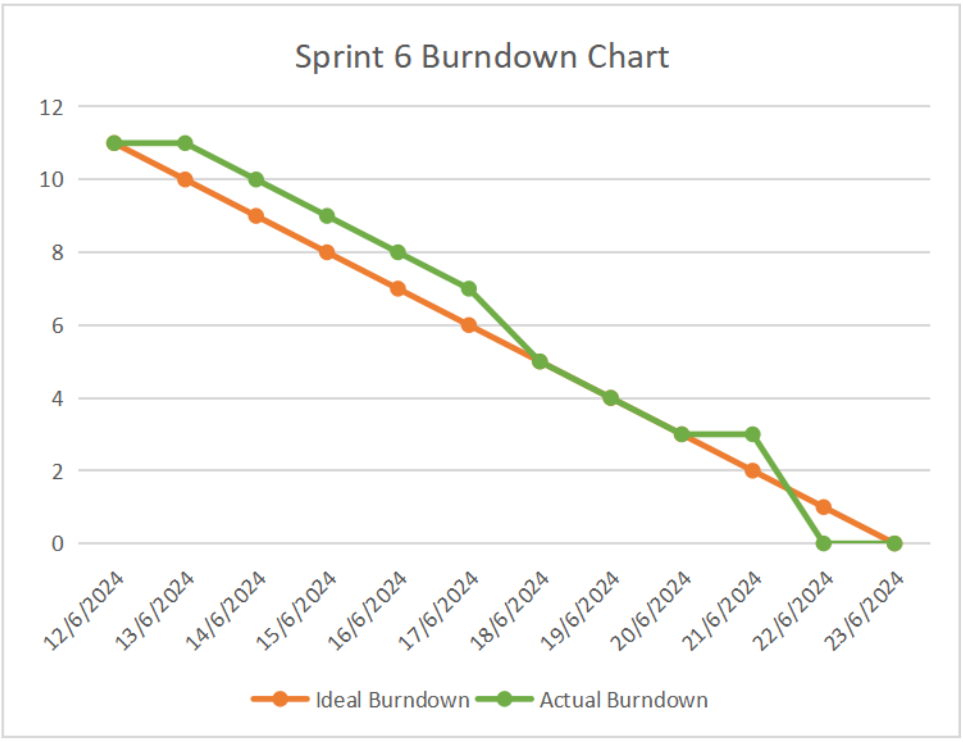
Demonstration:
The team demonstrated the language translation feature by switching the platform’s language to several options and showed how the chatbot handled common user queries effectively. The Dark/Light mode toggle was also demonstrated, highlighting the platform’s consistent styling across both modes.

Feedback:
We appreciated the improvements, particularly the chatbot’s functionality and

the Dark/Light mode feature. They suggested further expanding the language options and continuing to train the chatbot with more complex queries in the future.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:** The sprint began with setting up the language translation and chatbot services, which was successful.
- **Mid-Sprint Adjustments:** The team made minor adjustments to the chatbot integration to ensure it was non-intrusive. The Dark/Light mode toggle was developed smoothly.
- **Final Phase:** The sprint concluded with all tasks completed on time, with the team focusing on testing and refining the user experience.

Retrospective Discussion:

Successes:
The integration of advanced features such as multi-language support, an AI chatbot, and a Dark/Light mode significantly enhanced the platform's user experience. The team managed to deliver all these features without compromising the platform's performance.

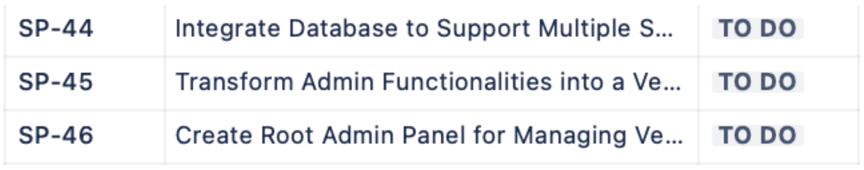
Areas for Improvement:

- While the translation feature worked well, some languages showed minor inconsistencies. Future sprints could focus on refining these translations.
- The chatbot could benefit from continuous training and expanding its knowledge base to handle more complex queries.

Action Items:

- Continuously monitor and refine the language translation feature to ensure accuracy.
- Continue training the chatbot to improve its responses and expand its capability to handle more diverse user queries.

7. Product Backlog Refinement



SP-44	Integrate Database to Support Multiple S...	TO DO
SP-45	Transform Admin Functionalities into a Ve...	TO DO
SP-46	Create Root Admin Panel for Managing Ve...	TO DO

Refinement Activities:

- The product backlog was reviewed and updated, with new priorities focusing on future enhancements based on user feedback and testing results.

Outcome:

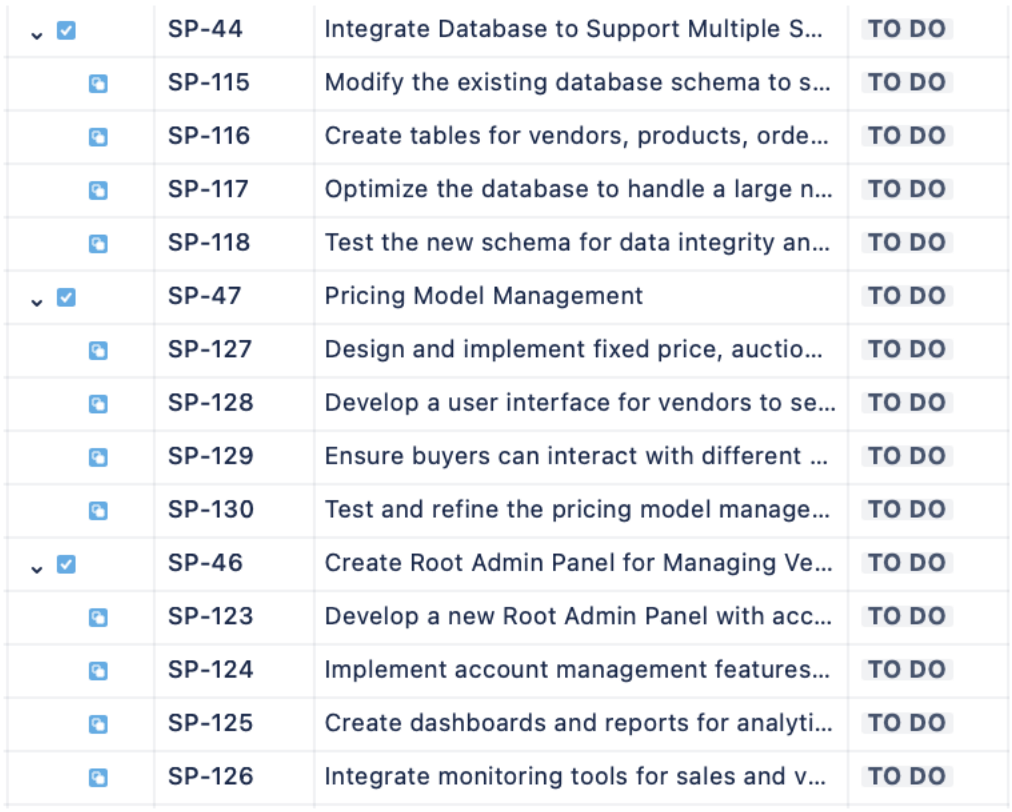
- The product backlog was updated, and the platform is now ready for potential expansion into a multi-vendor marketplace, leveraging the enhanced user experience features developed in this sprint.

7th Sprint : Database Expansion and Vendor/Seller Integration

1. Sprint Planning

Objective: The primary goal of Sprint 7 was to expand the existing e-commerce platform's database and system architecture to support a multi-vendor marketplace. This included creating a Vendor/Seller Panel, adapting existing admin functionalities, and developing a Root Admin Panel for overall marketplace management.

Sprint 7 backlog (originated from the main product backlog):



⌄ ✓	SP-44	Integrate Database to Support Multiple S...	TO DO
▣	SP-115	Modify the existing database schema to s...	TO DO
▣	SP-116	Create tables for vendors, products, orde...	TO DO
▣	SP-117	Optimize the database to handle a large n...	TO DO
▣	SP-118	Test the new schema for data integrity an...	TO DO
⌄ ✓	SP-47	Pricing Model Management	TO DO
▣	SP-127	Design and implement fixed price, auctio...	TO DO
▣	SP-128	Develop a user interface for vendors to se...	TO DO
▣	SP-129	Ensure buyers can interact with different ...	TO DO
▣	SP-130	Test and refine the pricing model manage...	TO DO
⌄ ✓	SP-46	Create Root Admin Panel for Managing Ve...	TO DO
▣	SP-123	Develop a new Root Admin Panel with acc...	TO DO
▣	SP-124	Implement account management features...	TO DO
▣	SP-125	Create dashboards and reports for analyti...	TO DO
▣	SP-126	Integrate monitoring tools for sales and v...	TO DO

User Stories (from Product Backlog):

User Story 1: Integrate Database to Support Multiple Sellers

As a marketplace administrator, **I want** the database to support multiple vendors with their own product listings and orders, **So that** each vendor can manage their products independently within the marketplace.

Acceptance Criteria:

1. The database must include tables specifically for vendors, products, orders, and inventory.
2. Each vendor’s data should be uniquely identifiable and separate from other vendors.
3. The database should handle a large number of vendors without performance degradation.

User Story 2: Transform Admin Functionalities into a Vendor/Seller Panel

As a vendor, **I want** to manage my products, orders, and inventory through a dedicated Vendor/Seller Panel, **So that** I can operate within the marketplace independently.

Acceptance Criteria:

1. The Vendor/Seller Panel should allow for adding, updating, and deleting products.
2. Vendors should be able to view and manage their own orders and inventory.
3. The user interface must be intuitive and easy to navigate.

User Story 3: Create Root Admin Panel for Managing Vendors and Marketplace Operations

As a root administrator, **I want** a comprehensive Root Admin Panel to oversee all vendors and marketplace operations, **So that** I can effectively manage the entire platform.

Acceptance Criteria:

1. The Root Admin Panel should provide access to all vendor data and activities.
2. Admins should be able to manage vendor accounts, monitor sales, and resolve disputes.
3. The panel should have dashboards and reports for analytics.

Sprint Goal: To successfully integrate support for multiple vendors into the database, transform the existing e-commerce admin functionalities into a Vendor/Seller Panel, and develop a Root Admin Panel for marketplace management.

2. Sprint Backlog

Tasks Derived from User Stories:

User Story 1: Integrate Database to Support Multiple Sellers

1. Modify the existing database schema to support multiple vendors.
2. Create tables for vendors, products, orders, and inventory with proper relationships.
3. Optimize the database to handle a large number of entries without performance degradation.
4. Test the new schema for data integrity and scalability.

User Story 2: Transform Admin Functionalities into a Vendor/Seller Panel

5. Adapt the existing admin panel for vendor-specific functionalities.
6. Implement product management features for vendors.
7. Enable vendors to manage their orders and inventory.
8. Ensure the panel's interface is user-friendly and intuitive.

User Story 3: Create Root Admin Panel for Managing Vendors and Marketplace Operations

1. Develop a new Root Admin Panel with access to all vendor data.
2. Implement account management features for the root admin.
3. Create dashboards and reports for analytics.
4. Integrate monitoring tools for sales and vendor activities.

3. Development

Execution:

User Story 1: Integrate Database to Support Multiple Sellers

- The database schema was expanded to include new tables for vendors, products, orders, and inventory. Each table was designed to be uniquely identifiable to support multiple vendors. The relationships between tables were carefully mapped to ensure data integrity. The team also optimized the database to handle a significant number of vendors without performance issues.

User Story 2: Transform Admin Functionalities into a Vendor/Seller Panel

- The existing e-commerce admin functionalities were adapted to create a Vendor/Seller Panel. This involved redesigning the interface and adjusting the back-end to support vendor-specific operations. Vendors were now able to add, update, and delete their products, as well as manage orders and inventory. The UI was also enhanced to be more intuitive, ensuring a smooth user experience.

User Story 3: Create Root Admin Panel for Managing Vendors and Marketplace Operations

- A new Root Admin Panel was developed to allow the root administrator to manage the entire marketplace. This panel included features for vendor account management, sales monitoring, and analytics. Dashboards were created to provide an overview of marketplace activities, and reports were integrated for detailed analysis. The panel was designed to be comprehensive yet user-friendly, enabling efficient marketplace management.

4. Testing

Approach:

User Story 1: Integrate Database to Support Multiple Sellers

- The database was rigorously tested for scalability, performance, and data integrity. Various scenarios were simulated to ensure that the database could handle a large number of vendors and their data without performance degradation.

User Story 2: Transform Admin Functionalities into a Vendor/Seller Panel

- The Vendor/Seller Panel was tested by simulating vendor operations such as product management and order processing. The team ensured that the panel was user-friendly and that all functionalities were working as expected.

User Story 3: Create Root Admin Panel for Managing Vendors and Marketplace Operations

- The Root Admin Panel underwent extensive testing to ensure that it provided comprehensive access to vendor data and allowed for effective management of the marketplace. The dashboards and reports were also tested to ensure accuracy and usability.

Outcome:

- All functionalities passed the testing phase successfully. The database integration supported multiple vendors seamlessly, the Vendor/Seller Panel was functional and user-friendly, and the Root Admin Panel provided comprehensive management tools.

5. Sprint Review

Summary:

Achievements:

- The sprint successfully expanded the platform to support multiple vendors and introduced new panels for both vendors and root administrators. These changes marked a significant step towards transforming the e-commerce platform into a full-fledged marketplace.

Demonstration:

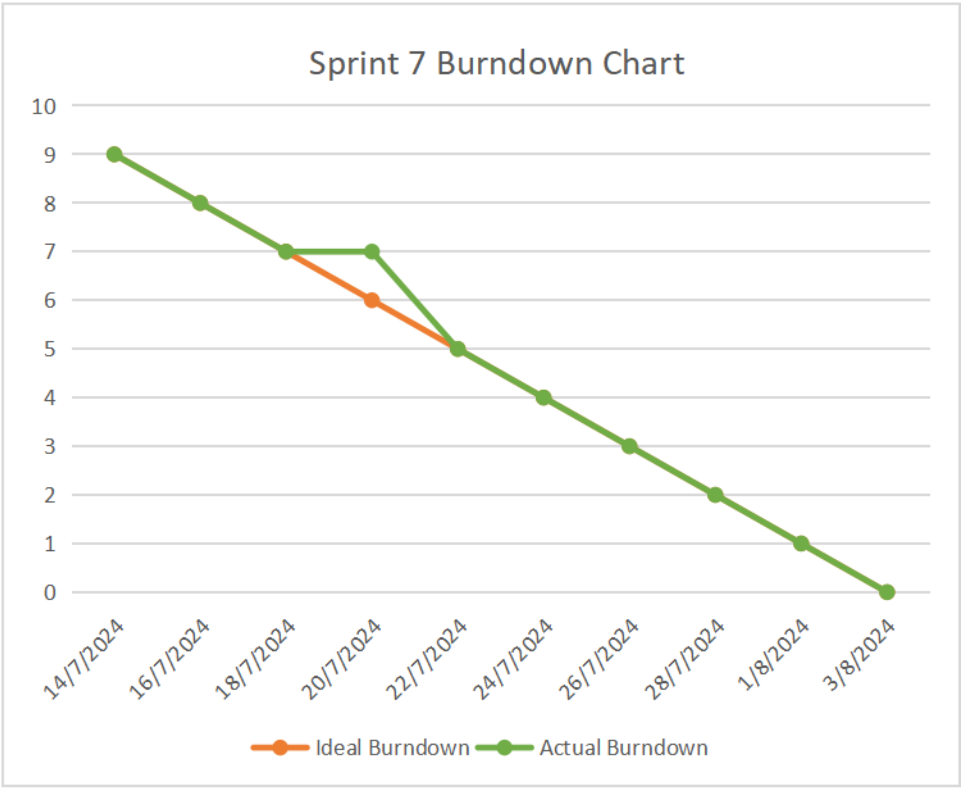
- The team demonstrated the new database structure, the Vendor/Seller Panel, and the Root Admin Panel. The demonstration showcased how vendors could manage their products and how the root admin could oversee the entire marketplace.

Feedback:

- Stakeholders appreciated the enhanced functionalities, especially the Root Admin Panel's ability to manage and monitor the marketplace. Some feedback included suggestions for further refining the vendor management features.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

- **Initial Progress:**
  - The sprint began with the database integration, which required significant planning and execution. Progress was steady, with initial tasks completed on time.
- **Mid-Sprint Adjustments:**
  - Some adjustments were made to the Vendor/Seller Panel's interface based on early testing feedback. These changes were implemented without affecting the overall timeline.
- **Final Phase:**
  - The sprint concluded successfully with all tasks completed as planned. Testing and refinement were the main focus during the final days, ensuring that the new features were polished and ready for deployment.

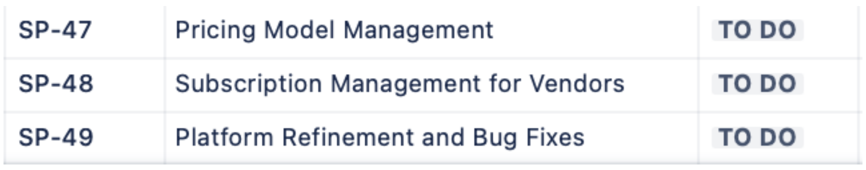
Retrospective Discussion:

- **Successes:**
  - The expansion of the platform to support a multi-vendor marketplace was a significant achievement. The team managed to deliver complex features on time and with high quality.
- **Areas for Improvement:**
  - The initial database design took longer than expected. Future sprints could benefit from more detailed planning in this area.

Action Items:

- Continue refining the Vendor/Seller Panel based on user feedback.
- Consider further optimization of the database as the marketplace grows.

7. Product Backlog Refinement



SP-47	Pricing Model Management	TO DO
SP-48	Subscription Management for Vendors	TO DO
SP-49	Platform Refinement and Bug Fixes	TO DO

New User Stories:

- **Vendor Analytics:** Develop detailed analytics for vendors to track their sales and performance.
- **Advanced Subscription Models:** Introduce more complex subscription models for vendors.

Refinement Activities:

- The product backlog was reviewed and updated, with new priorities focusing on enhancing vendor management and analytics based on the successful completion of Sprint 7.

Outcome:

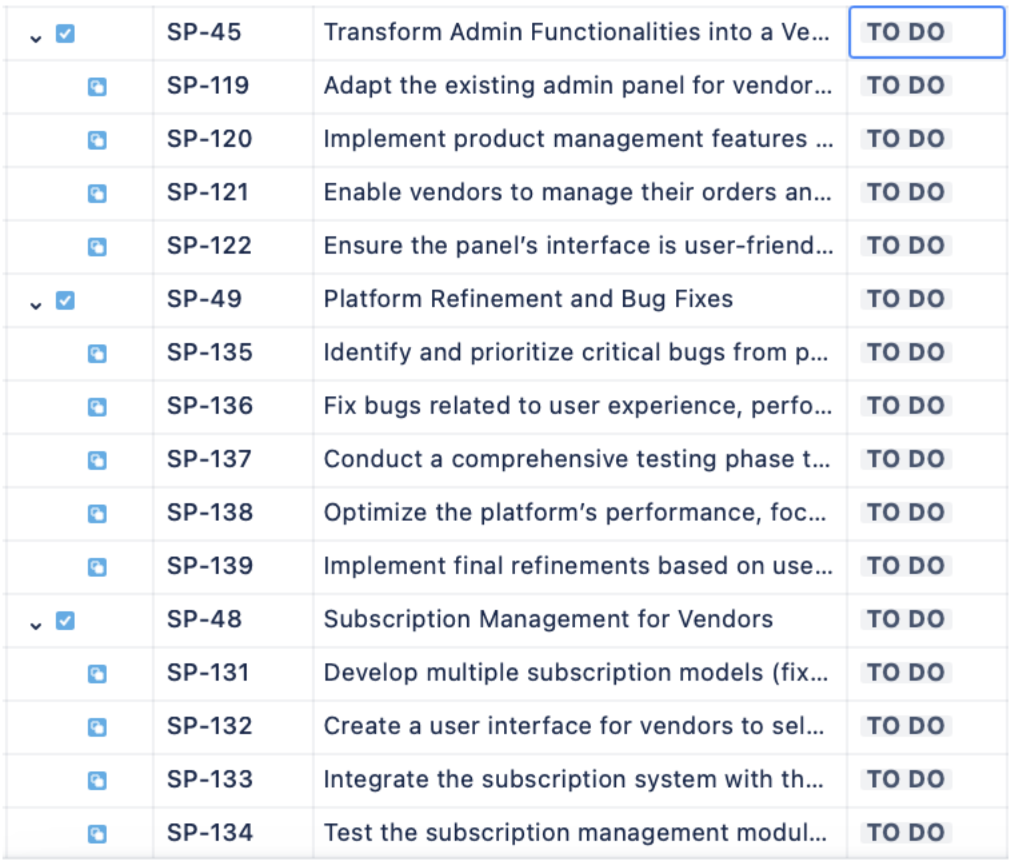
- The product backlog was updated, setting the stage for the next sprint, where the focus will be on further enhancing the marketplace features and optimizing the platform for scalability and performance.

8th Sprint : Marketplace Pricing Model, Subscription Management, and Final Touches

1. Sprint Planning

Objective: The final sprint for the marketplace expansion aimed to develop and integrate a comprehensive pricing model management module and a customizable subscription management system for vendors. Additionally, this sprint focused on refining the overall platform by addressing bugs and ensuring that all features work seamlessly together.

Sprint 8 backlog (originated from the main product backlog):



⌄ ☑	SP-45	Transform Admin Functionalities into a Ve...	TO DO
	SP-119	Adapt the existing admin panel for vendor...	TO DO
	SP-120	Implement product management features ...	TO DO
	SP-121	Enable vendors to manage their orders an...	TO DO
	SP-122	Ensure the panel's interface is user-friend...	TO DO
⌄ ☑	SP-49	Platform Refinement and Bug Fixes	TO DO
	SP-135	Identify and prioritize critical bugs from p...	TO DO
	SP-136	Fix bugs related to user experience, perfo...	TO DO
	SP-137	Conduct a comprehensive testing phase t...	TO DO
	SP-138	Optimize the platform's performance, foc...	TO DO
	SP-139	Implement final refinements based on use...	TO DO
⌄ ☑	SP-48	Subscription Management for Vendors	TO DO
	SP-131	Develop multiple subscription models (fix...	TO DO
	SP-132	Create a user interface for vendors to sel...	TO DO
	SP-133	Integrate the subscription system with th...	TO DO
	SP-134	Test the subscription management modul...	TO DO

User Stories (from Product Backlog):

User Story 1: Pricing Model Management

As a marketplace vendor, **I want** to choose between different pricing models (fixed price, auction-based, and custom pricing), **So that** I can sell my products using the model that best suits my business strategy.

Acceptance Criteria:

1. The platform must support fixed price, auction-based, and custom pricing models.
2. Vendors should be able to set up and manage their preferred pricing model easily.
3. Buyers should be able to interact with different pricing models without any issues.

User Story 2: Subscription Management for Vendors

As a marketplace administrator, **I want** to offer customizable subscription plans to vendors, **So that** they can choose a plan that fits their needs (e.g., fixed fee, percentage of sales, special offers).

Acceptance Criteria:

1. The platform should support multiple subscription models (e.g., fixed, percentage-based, special offers).
2. Vendors should be able to select and customize their subscription plan.
3. The subscription model should integrate seamlessly with the payment system.

User Story 3: Platform Refinement and Bug Fixes

As a user, **I want** a seamless and bug-free experience on the marketplace, **So that** I can navigate, purchase, and interact with the platform without any issues.

Acceptance Criteria:

1. All critical bugs identified in previous sprints should be resolved.
2. The platform's performance should be optimized.
3. User feedback should be addressed in the final refinements.

Sprint Goal: To finalize the marketplace by integrating pricing models, subscription management, and resolving any outstanding issues, ensuring a robust, user-friendly platform ready for deployment.

2. Sprint Backlog

Tasks Derived from User Stories:

User Story 1: Pricing Model Management

- Design and implement fixed price, auction-based, and custom pricing models.
- Develop a user interface for vendors to select and manage their pricing models.
- Ensure buyers can interact with different pricing models seamlessly.

- Test and refine the pricing model management system.

User Story 2: Subscription Management for Vendors

- Develop multiple subscription models (fixed fee, percentage-based, special offers).
- Create a user interface for vendors to select and customize their subscription plans.
- Integrate the subscription system with the payment gateway.
- Test the subscription management module to ensure accuracy and usability.

User Story 3: Platform Refinement and Bug Fixes

- Identify and prioritize critical bugs from previous sprints.
- Fix bugs related to user experience, performance, and functionality.
- Conduct a comprehensive testing phase to ensure all bugs are resolved.
- Optimize the platform's performance, focusing on load times and responsiveness.
- Implement final refinements based on user feedback.

3. Development

Execution:

User Story 1: Pricing Model Management

- The development team implemented a flexible pricing model system that allowed vendors to choose between fixed price, auction-based, and custom pricing strategies. The user interface was designed to be intuitive, providing vendors with easy access to manage their pricing models. Auction functionality included bidding mechanisms and time constraints, while custom pricing allowed vendors to set unique pricing rules. Integration testing ensured that buyers could interact with these models without issues, whether they were purchasing directly or participating in an auction.

User Story 2: Subscription Management for Vendors

- The subscription management module was developed to support a variety of plans, including fixed fees, percentage-based fees, and special offers. Vendors could easily select and customize their subscription plans through a user-friendly interface. The subscription module was tightly integrated with the payment gateway, ensuring that fees were automatically calculated and processed according to the selected plan. Extensive testing was conducted to ensure accuracy in billing and ease of use for vendors.

User Story 3: Platform Refinement and Bug Fixes

- The team conducted a thorough review of all features developed in previous sprints, identifying and prioritizing bugs based on their severity and impact on the user experience. Critical bugs, such as those affecting payment processing, product display, and user navigation, were fixed. The platform underwent extensive performance optimization, focusing on reducing load times and improving responsiveness. The team also implemented final refinements based on user feedback, ensuring that the platform was polished and ready for launch.

4. Testing

Approach:

User Story 1: Pricing Model Management

- The pricing models were tested across multiple scenarios, including fixed price purchases, auction participation, and custom pricing rules. The team verified that each model worked as expected and that buyers could easily interact with each pricing strategy. Edge cases, such as simultaneous bids in auctions, were also tested to ensure system reliability.

User Story 2: Subscription Management for Vendors

- The subscription management system was tested by simulating vendor sign-ups and subscription plan changes. The integration with the payment gateway was thoroughly tested to ensure accurate billing. Various scenarios, such as switching subscription plans and applying special offers, were also tested to ensure the system's robustness.

User Story 3: Platform Refinement and Bug Fixes

- A comprehensive testing phase was conducted to ensure that all identified bugs were resolved. The platform's performance was tested under various conditions, including high traffic scenarios, to ensure stability and responsiveness. User feedback was also revisited, with additional testing to verify that all refinements met user expectations.

Outcome:

- All new features passed testing successfully, and the platform was found to be stable, user-friendly, and ready for deployment. The pricing models worked seamlessly, the subscription management system was accurate and easy to use, and all critical bugs were resolved.

5. Sprint Review

Summary:

Achievements:

- The sprint successfully implemented key features that are crucial for a multi-vendor marketplace, including a flexible pricing model management system and a robust subscription management module. Additionally, all identified bugs were fixed, and the platform's performance was optimized.

Demonstration:

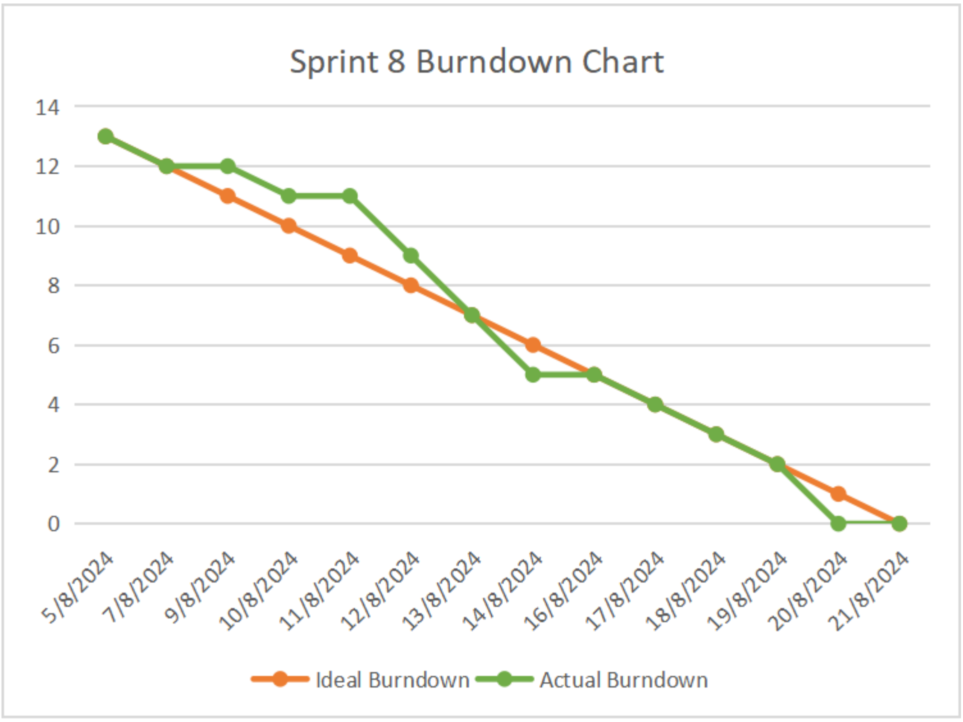
- The team demonstrated the new pricing models by showing how vendors could set up fixed prices, auctions, and custom pricing. The subscription management module was showcased, highlighting the ease of plan selection and integration with the payment gateway. A final review of the platform showed that all previously reported bugs had been resolved, and the platform was running smoothly.

Feedback:

- Stakeholders were impressed with the platform's comprehensive pricing and subscription features. They suggested monitoring the auction system post-launch to ensure fairness and functionality. The final refinements were well-received, with particular praise for the improved user experience and performance.

6. Sprint Retrospective

Burndown Chart Analysis:



Analysis:

Initial Progress:

- The sprint started with the implementation of the pricing model system, which required detailed planning and execution. Progress was steady, with the development team adhering to the schedule.

Mid-Sprint Adjustments:

- Minor adjustments were made to the subscription management module based on initial testing feedback. These adjustments were implemented quickly, allowing the sprint to stay on track.

Final Phase:

- The final days of the sprint focused on comprehensive testing and bug fixing. The team worked efficiently to ensure all critical bugs were resolved and that the platform was fully optimized for performance.

Retrospective Discussion:

Successes:

- The successful implementation of complex pricing models and subscription management within the sprint timeline was a major achievement. The bug-fixing process was also efficient, leading to a highly polished platform.

Areas for Improvement:

- The team noted that while the bug-fixing phase was successful, a more proactive approach to identifying and resolving potential issues earlier in the development process could have further streamlined the sprint.

Action Items:

- Post-launch, monitor the auction system closely to ensure it operates smoothly under real-world conditions.
- Consider implementing automated testing to catch bugs earlier in future development.

7. Bug Fixes Summary

Critical Bugs Resolved:

Payment Processing Error:

- **Issue:** In some instances, the payment gateway failed to process transactions correctly.

- **Fix:** The integration was reviewed, and the bug was traced to an API mismatch, which was corrected. Additional logging was added to monitor payment gateway activities.

Product Display Inconsistencies:

- **Issue:** Certain products were not displaying correctly on vendor pages due to a database indexing error.
- **Fix:** The indexing system was restructured, ensuring that all products are displayed accurately and in real-time.

Slow Load Times on Vendor Panel:

- **Issue:** Vendors experienced slow load times when accessing the Vendor/Seller Panel, especially when managing large inventories.
- **Fix:** The panel was optimized, including database query improvements and front-end adjustments, resulting in significantly faster load times.

Order Management Bug:

- **Issue:** Vendors encountered issues when attempting to update or cancel orders.
- **Fix:** The order management module was refactored to handle these operations more efficiently. Testing confirmed that updates and cancellations are now processed correctly.

Subscription Billing Error:

- **Issue:** Some vendors reported discrepancies in their subscription billing.
- **Fix:** The subscription module was reviewed, and the billing algorithm was corrected to ensure accurate fee calculation. Vendors affected by the error were refunded as necessary.

Minor Bugs Resolved:

- **UI Glitches:** Minor interface issues were fixed across various panels, including misaligned buttons and text.
- **Search Functionality:** The search functionality was optimized to return more accurate and faster results.
- **Notifications:** Fixed a bug where notifications were not being triggered for certain user actions, such as order confirmations and new subscription sign-ups.

Testing and Verification:

- All bugs were tested in a controlled environment after being fixed to ensure they did not recur. The platform was also stress-tested to confirm its stability under peak load conditions.

Final Notes:

- The sprint concluded with a fully operational platform, free of critical bugs, and optimized for performance. The team is confident in the platform's readiness for launch, with mechanisms in place to monitor and address any post-launch issues that may arise.

UML Introduction and Its Importance in Software Projects

Introduction to UML

Unified Modeling Language (UML) is a standardized modeling language used in software engineering to visualize, specify, construct, and document the structure and behavior of software systems. UML provides a comprehensive set of diagramming techniques that facilitate clear communication among stakeholders, aid in system design, and enhance understanding of complex software systems.

Why Use UML in Software Projects?

Clarity and Communication:

UML diagrams help convey the design and architecture of the software in a clear and standardized format. This clarity supports effective communication among stakeholders, including developers, designers, project managers, and clients.

Documentation:

UML provides detailed documentation of system requirements, design, and functionality. This documentation is crucial for future maintenance, debugging, and upgrades.

Design and Analysis:

UML diagrams assist in visualizing the system’s structure and behavior, which helps in identifying potential issues early in the design phase. This supports better planning and analysis, leading to more efficient development.

Standardization:

UML offers a common language and set of symbols that are widely understood in the software engineering community. This standardization helps in aligning the development process with industry practices.

Design Patterns:

UML can represent common design patterns and solutions, facilitating their reuse and implementation in new projects.

UML Diagrams Used in This Project

1. Use Case Diagram

The Use Case Diagram outlines the interactions between users (actors) and the system (use cases). It helps in identifying functional requirements and understanding how different actors interact with the system.

Use Case Diagram Overview:

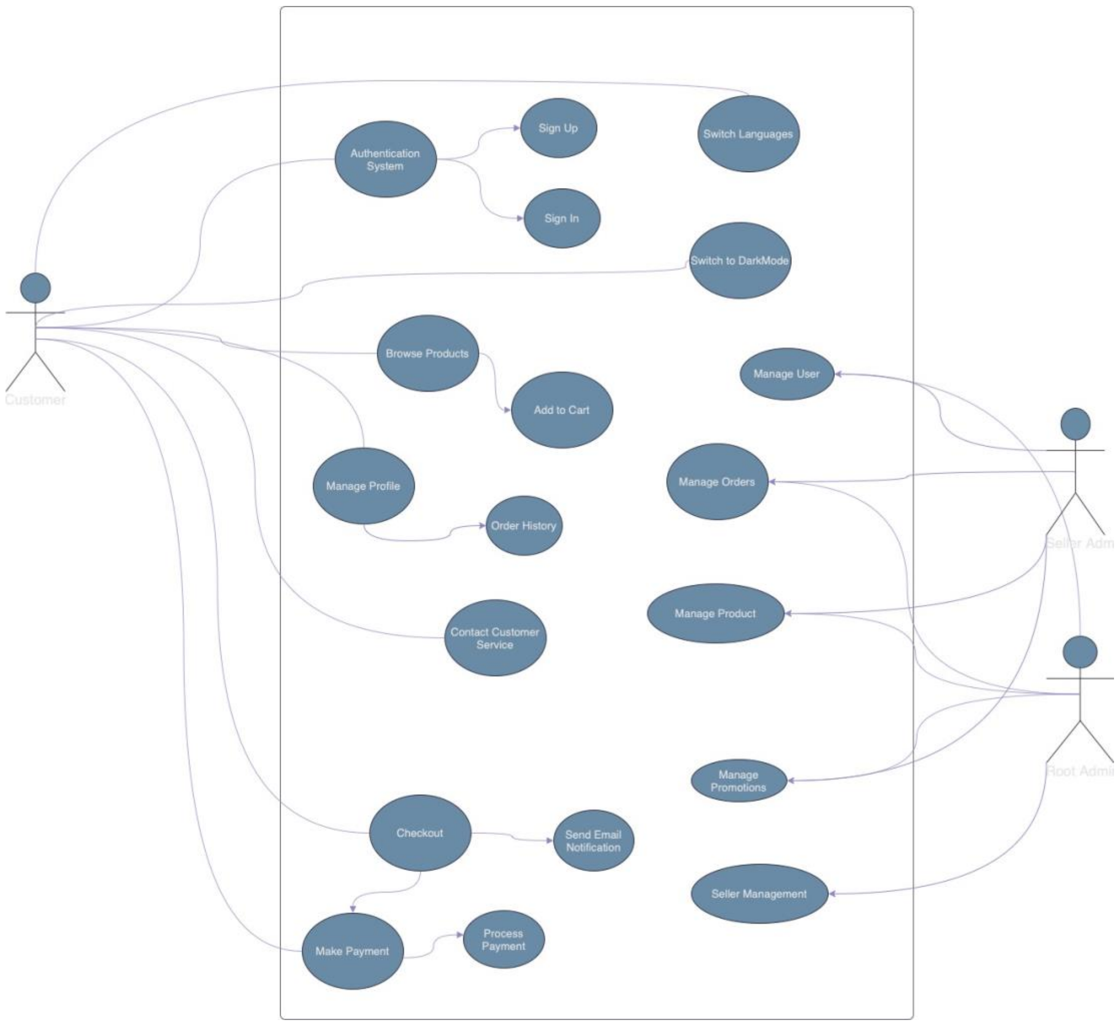
Actors:

- **Vendor:** Interacts with the platform to manage products, view sales, and handle orders.
- **Buyer:** Purchases products, participates in auctions, and interacts with the platform's features.
- **Admin:** Manages the overall platform, including users, products, and marketplace settings.
- **System:** Provides system-level interactions and services, such as payment processing and notifications.

Use Cases:

- **Manage Products:** Vendors add, update, and remove products.
- **View Sales Reports:** Vendors view sales analytics and reports.
- **Place Order:** Buyers purchase products and participate in auctions.
- **Manage Users:** Admin manages vendors and buyers.
- **Handle Payments:** System processes payments and manages financial transactions.
- **Send Notifications:** System sends notifications for orders, promotions, and system updates.

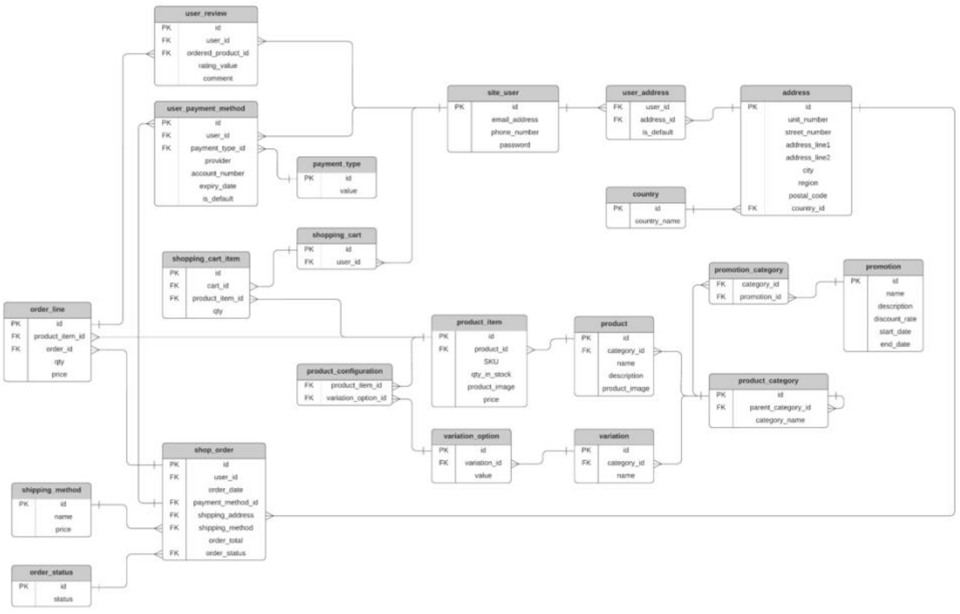
Use Case Diagram :

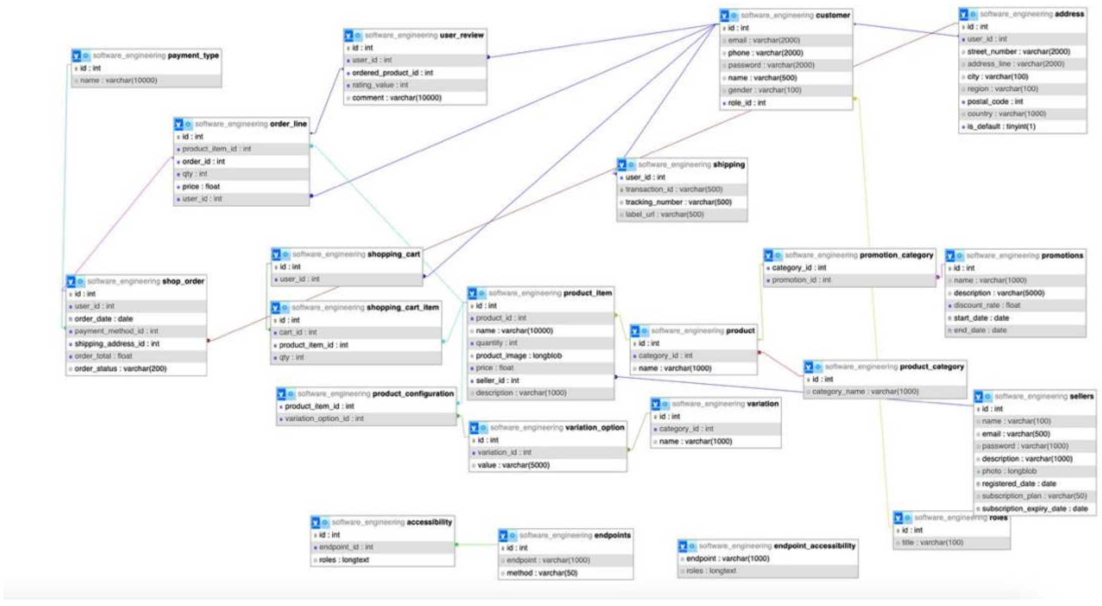


2. Class Diagram

The Class Diagram provides a detailed view of the system's static structure, showing the system's classes, their attributes, methods, and relationships. It is crucial for understanding the system's data model and interactions between different components.

Class Diagram :

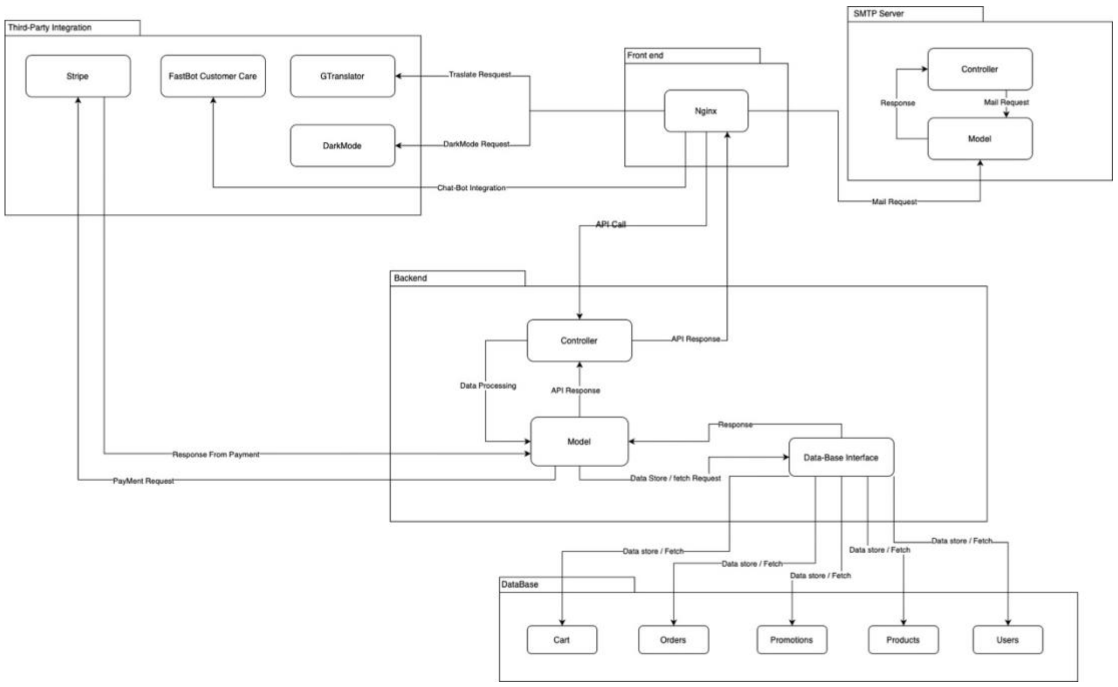




3. Component Diagram

The Component Diagram illustrates the system's high-level structure, showing how different software components interact with each other. It helps in understanding the system's architecture and the dependencies between components.

Component Diagram :


Third-Party Integration
Stripe
FastBot Customer Care
GTranslator
DarkMode
Traslate Resquest
DarkMode Request
Chat-Bot Integration
Front end
Nginx
SMTP Server
Controller
Model
Response
Mail Request
Mail Request
API Call
Backend
Controller
API Response
Data Processing
API Response
Model
Response
Data-Base Interface
Response From Payment
PayMent Request
Data Store / fetch Request
Data store / Fetch
Data store / Fetch
Data store / Fetch
Data store / Fetch
Data store / Fetch
DataBase
Cart
Orders
Promotions
Products
Users

Software Architecture Description

1. Introduction

1.1 Purpose

This document outlines the software architecture for the Multi-Vendor E-commerce Marketplace Platform, detailing the structure, components, and interactions within the system. It serves as a guide for developers, architects, and other stakeholders to understand the system's design and facilitate its development, deployment, and maintenance.

1.2 Scope

The architecture description covers:

- **Architectural Goals and Constraints**
- **High-Level Architectural Overview**
- **Logical View**
- **Process View**
- **Physical/Deployment View**
- **Data View**
- **Integration and APIs**

This document is intended for all stakeholders involved in the system's lifecycle, including software architects, developers, testers, and project managers.

1.3 Definitions, Acronyms, and Abbreviations

- **MVC:** Model-View-Controller
- **REST:** Representational State Transfer
- **API:** Application Programming Interface
- **DBMS:** Database Management System
- **UML:** Unified Modeling Language

2. Architectural Goals and Constraints

2.1 Architectural Goals

The architecture of the Multi-Vendor E-commerce Marketplace Platform is designed to meet the following goals:

- **Scalability:** Support a large number of users, vendors, and products, with the ability to scale horizontally and vertically as needed.

- **Modularity:** Enable easy maintenance and feature extensions by adhering to the MVC (Model-View-Controller) pattern, separating concerns across the system.
- **Security:** Ensure data protection and secure transactions through encryption, authentication, and authorization mechanisms.
- **Performance:** Optimize system performance to handle high traffic loads, ensuring fast response times and low latency.
- **Reliability:** Ensure the platform is reliable and available, minimizing downtime and ensuring business continuity.

2.2 Architectural Constraints

- **Technology Stack:** The architecture must leverage Flask for backend development, Nginx for frontend, MySQL for database management, and integrate with third-party services (e.g., Stripe for payments, FastBot for AI-powered chat).
- **Deployment Environment:** The system must be deployable in a cloud environment, supporting containerization (e.g., Docker) and orchestration (e.g., Kubernetes).
- **Compliance:** The system must comply with industry standards for data protection and privacy, such as GDPR.

3. High-Level Architectural Overview

3.1 System Context

The system is a multi-vendor e-commerce marketplace where multiple sellers can list their products, and users can browse, purchase, and interact with the platform's features. The system interacts with various external services, including payment gateways, AI-powered chatbots, and translation services.

Diagram :

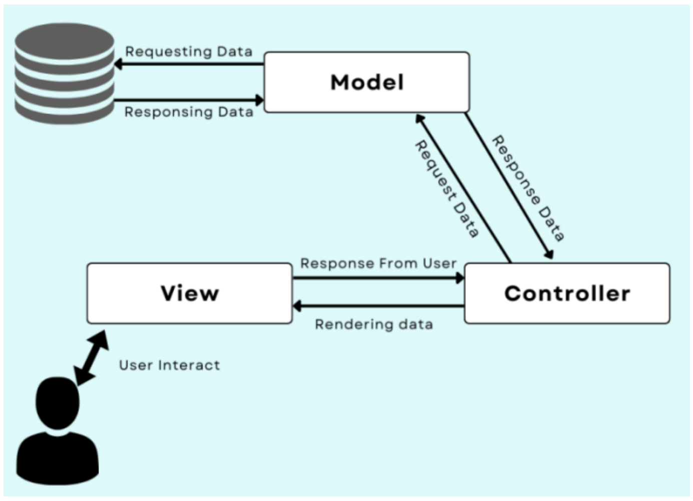
- (we can Insert System Context Diagram Here)

3.2 Architectural Style

The architecture follows the MVC (Model-View-Controller) pattern, which divides the application into three interconnected components:

- **Model:** Manages the data and business logic. It interacts with the database (MySQL) and handles data-related operations.
- **View:** Represents the UI, managed by Nginx, responsible for rendering data to the user and capturing user input.

- **Controller:** Acts as an intermediary between Model and View, processing user requests, updating the model, and selecting the appropriate view for response.



4. Logical View

4.1 Overview

The Logical View represents the static structure of the system, depicting how components are organized and how they interact with each other. The primary components include:

- **User Interface (UI):** Managed by Nginx, responsible for rendering the user interface and handling frontend operations.
- **Backend Services:** Developed using Flask, handling API requests, processing business logic, and interacting with the database.
- **Database Layer:** Managed by MySQL, responsible for storing and retrieving data.
- **Third-Party Integrations:** Interfaces for payment processing (Stripe), AI chatbot (FastBot), and language translation (Google Translate).

4.2 Key Components

- **Authentication & Authorization:** Manages user login, registration, and role-based access control.
- **Product Management:** Handles product creation, updating, deletion, and listing.
- **Order Management:** Manages the lifecycle of orders, from cart to payment and order history.
- **Vendor Management:** Provides functionalities for vendors to manage their products and view analytics.

- **Admin Panel:** Centralized control for Root Admin to manage users, vendors, products, and platform settings.

5. Process View

5.1 Overview

The Process View details the dynamic aspects of the system, showing how different components interact during runtime. It includes the workflows for key operations such as user authentication, product browsing, order processing, and admin functions.

5.2 Major Workflows

- **User Authentication Workflow:**
    - Users authenticate via OAuth or traditional login.
    - The Controller validates credentials and establishes a session.
- **Product Purchase Workflow:**
    - Users add products to their cart, calculate the total, and proceed to checkout.
    - The system interacts with Stripe to process the payment and store the order details in the database.
- **Vendor Operations Workflow:**
    - Vendors log in to their panel, manage products, and view sales analytics.
    - The system processes requests through Flask, updating the database and reflecting changes on the UI.

6. Physical/Deployment View

6.1 Overview

The Physical View outlines the deployment of the software across hardware resources, emphasizing containerization using Docker. This section includes server configurations, network topology, and details about the deployment environment, ensuring a consistent and scalable deployment process.

6.2 Docker Overview

Docker is a containerization platform that allows developers to package applications and their dependencies into standardized units called containers. These containers ensure that the application runs consistently across different environments, from development and testing to production. Docker containers are lightweight, portable, and isolated, which makes them ideal for modern software development and deployment strategies.

Key Benefits of Docker:

- **Consistency Across Environments:** Docker eliminates the "it works on my machine" problem by ensuring that the application environment is consistent across all stages of development and deployment.
- **Scalability:** Docker containers can be easily scaled horizontally, allowing for efficient management of resources and traffic handling.
- **Portability:** Docker containers can run on any system that supports Docker, making it easy to move applications between different servers or cloud environments.
- **Isolation:** Each Docker container runs in isolation, ensuring that the application's dependencies do not conflict with those of other applications.

6.3 Deployment Strategy

Frontend: The frontend application is deployed on Nginx servers, serving static files and handling client-side operations. The frontend is containerized using Docker, ensuring that it runs consistently across various environments, from development to production.

Backend: The Flask application, which serves as the backend, is also containerized using Docker. These Docker containers are deployed on application servers, where they expose APIs for frontend consumption. Docker ensures that the backend environment remains consistent and reduces the **"it works on my machine"** issue.

Database: MySQL is hosted on a dedicated database server or cluster. While the database itself might not be containerized in a production environment for performance reasons, Docker is used during development and testing to simulate the database environment, ensuring consistency across all stages of development.

Third-Party Services: Services such as Stripe for payment processing and FastBot for chatbot functionality are integrated via APIs. Docker containers may also be used for these services if they require specific configurations or dependencies.

Docker Containers: Both the frontend and backend services are packaged into Docker containers. This allows for easy scaling and deployment, as the containers can be quickly spun up, replicated, or moved across different servers or cloud environments.

Container Orchestration: To manage the deployment of multiple Docker containers, a container orchestration tool like Kubernetes or Docker Swarm may be used. This ensures that the application is resilient, can handle high traffic loads, and can be easily updated or scaled.

7. Data View

7.1 Overview

The Data View focuses on the data model of the system, including database schema, data flow, and data storage strategies.

7.2 Database Schema

- **Users Table:** Stores user information, including roles (buyer, seller, admin), credentials, and preferences.
- **Products Table:** Contains details about products, including seller information, pricing, and inventory.
- **Orders Table:** Tracks order details, including buyer information, product details, order status, and payment history.
- **Vendors Table:** Stores information related to vendors, including their products, sales data, and analytics.

8. Integration and APIs

8.1 External API Integrations

- **Stripe API:** Integrated for secure payment processing, supporting multiple payment methods.
- **Google Translate API:** Used for providing multi-language support across the platform.
- **FastBot API:** Integrated to offer AI-powered chatbot functionalities.

8.2 Internal APIs

- **RESTful APIs:** The platform exposes RESTful APIs for frontend-backend communication, following standard HTTP methods (GET, POST, PUT, DELETE).
- **Vendor APIs:** Vendors can interact with the system using dedicated APIs for managing products, viewing sales data, and handling orders.

Use Case Specification

1. Introduction to Use Cases

1.1 Purpose of Use Cases

Use cases are a crucial part of software design and development, providing a detailed description of how users will interact with the system to achieve specific goals. Each use case represents a scenario in which a user (referred to as an "actor") interacts with the system to accomplish a task. The purpose of these use cases is to ensure that all functional requirements are covered and that the system's design supports the needs of its users.

1.2 Scope

The use cases described here cover all primary interactions between users (buyers, vendors, admins) and the Multi-Vendor E-commerce Marketplace Platform. They outline the steps required to perform key activities such as user registration, product management, order processing, and administration. Each use case also includes details on actors, preconditions, triggers, and expected outcomes.

2. Use Case Diagram

2.1 Overview

A Use Case Diagram is a visual representation of the interactions between actors and the system. It provides an overview of the system's functionalities and how users engage with these functionalities.

2.2 Key Actors

- **Buyer:** A user who browses products, places orders, and manages their account.
- **Vendor:** A seller who lists products, manages inventory, processes orders, and views analytics.
- **Root Admin:** A system administrator with overarching control over all users, vendors, and system settings.
- **Guest:** A non-registered user who can browse products but must register or log in to make purchases.

3. Detailed Use Cases

Use Case 1: User Registration and Authentication

3.1.1 Use Case ID: UC-01

3.1.2 Use Case Name: User Registration and Authentication

3.1.3 Actors: Buyer, Vendor, Root Admin

3.1.4 Description: This use case describes the process through which users register for an account, log in, and authenticate their credentials to access the platform's features.

3.1.5 Preconditions:

- The user must have access to the platform's homepage.
- The user must have a valid email address and password for registration.

3.1.6 Trigger:

- The user navigates to the registration or login page.

3.1.7 Main Success Scenario:

Registration:

- The user clicks on the "Sign Up" button.
- The system presents a registration form.
- The user fills in the required details (name, email, password, role).
- The system validates the information.
- If valid, the system creates a new user account and sends a confirmation email.

Login:

- The user clicks on the "Log In" button.
- The system presents a login form.
- The user enters their email and password.
- The system verifies the credentials.
- If valid, the user is authenticated and redirected to their dashboard (Buyer, Vendor, or Admin).

3.1.8 Alternate Flows:

AF1: Invalid Credentials:

- If the user enters invalid credentials, the system displays an error message.
- The user is prompted to re-enter their credentials.

AF2: Forgotten Password:

   - The user clicks on "Forgot Password."
   - The system prompts the user to enter their email address.
   - The system sends a password reset link to the user’s email.

3.1.9 Postconditions:

- The user is logged in and redirected to the appropriate dashboard.
- The user account is active and ready for interaction with the platform.

3.1.10 Exceptions:

- **EX1: Registration Failure:**

   - If the email is already in use, the system prompts the user to log in or use a different email.
   - If any required fields are missing, the system highlights the errors and prompts the user to complete the form.

Use Case 2: Product Management by Vendors

3.2.1 Use Case ID: UC-02

3.2.2 Use Case Name: Product Management by Vendors

3.2.3 Actors: Vendor

3.2.4 Description: This use case covers the actions a vendor can perform to manage their products, including adding new products, updating existing ones, and removing products from their catalog.

3.2.5 Preconditions:

- The vendor must be logged in to their account.
- The vendor must have an active seller account.

3.2.6 Trigger:

- The vendor navigates to the product management section of their dashboard.

3.2.7 Main Success Scenario:

Add New Product:

   - The vendor clicks on "Add Product."
   - The system presents a form to input product details (name, description, price, inventory, category).
   - The vendor uploads product images.

    - The vendor submits the form.
    - The system validates the details and adds the product to the vendor’s catalog.

Update Existing Product:

    - The vendor selects a product from their catalog.
    - The system displays the product details.
    - The vendor edits the product information.
    - The vendor submits the changes.
    - The system updates the product in the database.

Remove Product:

    - The vendor selects a product from their catalog.
    - The vendor clicks on "Delete Product."
    - The system confirms the action.
    - The product is removed from the vendor’s catalog.

3.2.8 Alternate Flows:

AF1: Duplicate Product:

    - If the vendor tries to add a product that already exists, the system prompts them to edit the existing product instead.

AF2: Insufficient Inventory:

    - If the vendor tries to reduce inventory below the minimum threshold, the system displays a warning and prevents the update.

3.2.9 Postconditions:

- The product catalog is updated with the latest changes.
- The new or updated product is visible in the marketplace.

3.2.10 Exceptions:

- **EX1: Image Upload Failure:**
    - If the image upload fails, the system prompts the vendor to try again or upload a different image.

Use Case 3: Product Filtering by Buyers

3.3.1 Use Case ID: UC-03

3.3.2 Use Case Name: Product Filtering by Buyers

3.3.3 Actors: Buyer, Guest

3.3.4 Description: This use case describes how buyers and guests can search for products, apply filters, and view product details on the marketplace.

3.3.5 Preconditions:

- The buyer or guest must have access to the platform's homepage.

3.3.6 Trigger:

- The buyer or guest navigates to the product categories.

3.3.7 Main Success Scenario:

Filtering:

- o The buyer or guest applies filters (e.g., price range, category, brand).
- o The system updates the product list based on the selected filters.
- o The buyer or guest selects a product to view details.

View Product Details:

- o The buyer or guest clicks on a product.
- o The system displays the product details, including images, descriptions, pricing, and reviews.
- o The buyer or guest can add the product to the cart or wishlist.

3.3.9 Postconditions:

- The buyer or guest can proceed to purchase the selected product or continue browsing.
- The selected product is added to the cart or wishlist if the buyer or guest chooses to do so.

Use Case 4: Order Placement and Payment

3.4.1 Use Case ID: UC-04

3.4.2 Use Case Name: Order Placement and Payment

3.4.3 Actors: Buyer

3.4.4 Description: This use case describes the process through which buyers place orders and complete payments.

3.4.5 Preconditions:

- The buyer must be logged in to their account.
- The buyer must have items in their cart.

3.4.6 Trigger:

- The buyer clicks on the "Checkout" button.

3.4.7 Main Success Scenario:

Review Cart:

- The buyer reviews the items in their cart.
- The buyer can update quantities or remove items.
- The buyer clicks "Proceed to Checkout."

Enter Shipping Details:

- The buyer enters or selects a shipping address.
- The buyer reviews shipping options and costs.
- The buyer confirms the shipping details.

Make Payment:

- The buyer selects a payment method (Stripe or Cash on Delivery).
- The buyer enters payment details.
- The system processes the payment through the integrated payment gateway (Stripe).
- The system confirms payment and generates an order summary.

1.

Order Confirmation:

- The system sends an order confirmation email to the buyer.
- The order details are saved in the buyer's order history.

3.4.8 Alternate Flows:

AF1: Payment Failure:

- If the payment fails, the system displays an error message.
- The buyer can retry the payment or select a different payment method.

AF2: Shipping Address Issue:

    - If the entered address is invalid, the system prompts the buyer to correct the address.

3.4.9 Postconditions:

- The order is successfully placed and recorded in the system.
- The payment is processed, and the buyer receives an order confirmation.

3.4.10 Exceptions:

- **EX1: Insufficient Stock:**
    - If an item becomes unavailable during checkout, the system notifies the buyer, who must remove the item or adjust quantities before proceeding.

Use Case 5: Admin Management of Vendors and Platform

3.5.1 Use Case ID: UC-05

3.5.2 Use Case Name: Admin Management of Vendors and Platform

3.5.3 Actors: Root Admin

3.5.4 Description: This use case covers the administrative tasks performed by the Root Admin, including vendor management, platform settings, and overall marketplace monitoring.

3.5.5 Preconditions:

- The Root Admin must be logged in with appropriate privileges.

3.5.6 Trigger:

- The Root Admin navigates to the admin dashboard.

3.5.7 Main Success Scenario:

Manage Vendors:

- The Root Admin views a list of registered vendors.
- The Root Admin can approve, suspend, or remove vendor accounts.
- The Root Admin can review vendor performance and feedback.

Manage Platform Settings:

- The Root Admin adjusts platform-wide settings (e.g., pricing models, subscription tiers, payment settings).
- The Root Admin updates policies, terms, and conditions.

Monitor Marketplace:

- The Root Admin views real-time analytics on sales, traffic, and user engagement.
- The Root Admin identifies and resolves platform issues (e.g., payment processing errors, vendor disputes).

Manage User Accounts:

- The Root Admin reviews user reports and takes action (e.g., banning users, resolving disputes).
- The Root Admin manages roles and permissions for other admins.

3.5.8 Alternate Flows:

AF1: Vendor Appeal:

- If a vendor disputes an account suspension, the Root Admin can review the case and make a decision.

AF2: System Alerts:

- If the system detects a critical issue (e.g., security breach), the Root Admin is notified immediately and can take corrective action.

3.5.9 Postconditions:

- The platform continues to operate smoothly, with vendors and users managed effectively.
- All changes to platform settings and vendor statuses are logged.

3.5.10 Exceptions:

- **EX1: Admin Action Reversal:**
  - If an admin action is found to be in error, the Root Admin can reverse it, restoring the previous state.

Features Implemented in this project

1. Introduction

The Platform is an advanced online marketplace designed to facilitate interactions between multiple sellers and buyers. The platform allows vendors to manage their storefronts, list products, and handle transactions, while providing buyers with a seamless shopping experience. The platform is built with robust security measures, flexible pricing models, and advanced user experience features, making it suitable for a global audience. This report provides a detailed overview of the final product, highlighting the features implemented during the development process.

2. Overview of Features

The platform incorporates a comprehensive set of features aimed at enhancing both user and vendor experiences. These features can be broadly categorized into User Experience Enhancements, Core E-Commerce Functionality, and Administrative Tools.

3. User Experience Enhancements

3.1 User Authentication

Overview:
The platform features secure login and registration processes, ensuring that users and vendors can access the platform safely. The system uses strong encryption methods to protect user data during authentication.

Key Features:

- **Secure Registration:** Users and vendors can create accounts by providing necessary details such as email, password, and role (buyer or vendor).
- **Login Process:** The login system uses encrypted passwords and session management to maintain security.
- **Password Recovery:** Users can recover lost passwords securely using email verification.

Benefits:

- **Data Security:** Protects sensitive user information, preventing unauthorized access.
- **User Convenience:** Simplifies the process of managing accounts with secure and efficient authentication methods.

3.2 Multi-Language Support

Overview:
The platform supports multiple languages, making it accessible to users worldwide. This feature is critical for reaching a diverse audience and ensuring that users can navigate the platform in their preferred language.

Key Features:

- **Language Selection:** Users can choose their preferred language from a dropdown menu, which is available on the homepage.
- **Google Translate Integration:** The platform integrates with Google Translate to provide accurate translations of all major sections, including product descriptions, menus, and instructions.
- **Optimized Performance:** The translation feature is optimized to ensure it does not affect page load times.

Benefits:

- **Global Reach:** Enhances accessibility for non-English speakers, expanding the platform's user base.
- **User Satisfaction:** Improves the overall experience by allowing users to interact with the platform in their native language.

3.3 AI-Powered Chatbot

Overview:
The AI-powered chatbot provides 24/7 customer support, assisting users with common queries and guiding them through their shopping experience. The chatbot uses artificial intelligence to improve over time, offering increasingly relevant responses.

Key Features:

- **FastBot Integration:** The chatbot is integrated with FastBot, an AI service trained on a vast dataset relevant to the platform’s offerings.
- **24/7 Availability:** Accessible on all pages, the chatbot provides instant support to users at any time.
- **Learning Capability:** The AI chatbot improves its responses over time through machine learning.

Benefits:

- **Enhanced Support:** Offers immediate assistance, reducing the need for human customer support.
- **Increased Engagement:** Encourages users to interact more with the platform by providing quick and accurate responses.

3.4 Dark/Light Mode Toggle

Overview:
The platform includes a Dark/Light mode toggle, allowing users to switch between themes based on their preferences or environmental conditions. This feature enhances accessibility and improves the overall user experience.

Key Features:

- **Toggle Functionality:** A toggle button is prominently placed on the platform, allowing users to switch between Dark and Light modes easily.
- **Session Persistence:** The system remembers the user’s theme preference across sessions, providing a consistent experience.
- **UI Consistency:** The platform maintains a consistent and visually appealing interface in both Dark and Light modes.

Benefits:

- **User Comfort:** Reduces eye strain by allowing users to choose a theme that suits their environment.
- **Personalization:** Enhances user satisfaction by offering customization of the platform's appearance.

4. Core E-Commerce Functionality

4.1 Multi-Vendor Integration

Overview:
The platform allows multiple sellers to register and manage their products, creating a dynamic and diverse marketplace. Each vendor can manage their storefront independently, while the platform provides centralized control and monitoring.

Key Features:

- **Vendor Registration:** Sellers can create vendor accounts, listing their products on the platform.
- **Product Management:** Vendors have access to tools for adding, updating, and managing their product listings.
- **Inventory Management:** Vendors can track and manage inventory, ensuring accurate stock levels.

Benefits:

- **Diverse Marketplace:** Supports a wide variety of products, enhancing the platform's appeal to buyers.
- **Vendor Autonomy:** Allows vendors to manage their operations independently while benefiting from the platform's infrastructure.

4.2 Product Management

Overview:
Vendors can efficiently manage their products, including adding new listings, updating existing ones, and managing inventory. The system provides an intuitive interface for vendors to handle their product catalog.

Key Features:

- **Product Listing:** Vendors can add new products by entering relevant details such as name, description, price, and images.
- **Bulk Upload:** Allows vendors to upload multiple products at once, saving time and effort.
- **Inventory Tracking:** Provides real-time updates on stock levels, ensuring accurate product availability.

Benefits:

- **Efficiency:** Simplifies the process of managing a large product catalog, enabling vendors to focus on sales.
- **Accuracy:** Ensures that buyers see up-to-date information on product availability.

4.3 Payment Processing

Overview:
The platform supports multiple payment gateways, ensuring secure and flexible payment options for buyers. Vendors receive payments directly through the integrated payment systems.

Key Features:

- **Multiple Gateways:** Supports various payment gateways such as Stripe, PayPal, and credit/debit card processing.
- **Secure Transactions:** All transactions are encrypted, ensuring the security of sensitive financial information.
- **Transaction History:** Both buyers and vendors can view their transaction history, providing transparency and accountability.

Benefits:

- **Payment Flexibility:** Offers buyers a range of payment options, increasing the likelihood of completed transactions.
- **Security:** Protects sensitive payment information, building trust with users.

4.4 Notification Service

Overview:
The platform includes a real-time notification system that alerts users and vendors about important events such as order confirmations, payment receipts, and system updates.

Key Features:

- **Order Notifications:** Buyers receive notifications for order confirmations, shipping updates, and delivery statuses.
- **Payment Alerts:** Vendors are notified of successful payments and can track the payment status for their orders.
- **System Updates:** The platform sends out notifications for important updates, such as new features or scheduled maintenance.

Benefits:

- **User Engagement:** Keeps users informed, enhancing their experience and satisfaction.
- **Operational Efficiency:** Helps vendors stay updated on orders and payments, enabling prompt action.

5. Administrative Tools

5.1 Admin Panel

Overview:
The Admin Panel provides platform administrators with comprehensive control over the platform, including user management, vendor management, and system settings.

Key Features:

- **User Management:** Admins can manage user accounts, handle disputes, and monitor platform activity.
- **Vendor Oversight:** Admins can approve, suspend, or remove vendor accounts and monitor their performance.
- **Platform Settings:** Admins can adjust settings related to pricing models, subscription plans, and other operational parameters.

Benefits:

- **Control and Compliance:** Provides the tools necessary for maintaining order and ensuring compliance with platform policies.
- **Scalability:** Supports the platform’s growth by enabling efficient management of increasing numbers of users and vendors.

5.2 Root Admin Panel

Overview:
The Root Admin Panel offers centralized control over all vendors and platform settings, allowing for high-level management and strategic decision-making.

Key Features:

- **Vendor Management:** Root Admins can oversee all vendors, manage their accounts, and ensure compliance with marketplace standards.
- **System Configuration:** Allows Root Admins to configure global settings that affect the entire platform, such as payment processing and security protocols.
- **Reporting Tools:** Provides detailed analytics and reports.

Benefits:

- **Strategic Oversight:** Enables high-level management of the marketplace, ensuring that it operates smoothly and efficiently.
- **Data-Driven Decisions:** Supports informed decision-making through comprehensive reporting tools.

5.3 Seller/Vendor Panel

Overview:
The Seller/Vendor Panel provides vendors with a dedicated interface to manage their storefronts, including product listings, orders, and customer interactions.

Key Features:

- **Dashboard:** A central dashboard gives vendors an overview of their sales, orders, and inventory.
- **Order Management:** Vendors can track and manage orders, process returns, and communicate with customers.
- **Analytics:** Provides vendors with insights into their sales performance, helping them optimize their operations.

Benefits:

- **Operational Autonomy:** Empowers vendors to manage their businesses independently while benefiting from the platform’s infrastructure.
- **Performance Monitoring:** Helps vendors improve their sales and customer service through data-driven insights.

5.4 Pricing Model and Subscription

Overview:
The platform offers flexible pricing models and subscription plans for vendors, allowing them to choose the plan that best suits their business needs.

Key Features:

- **Subscription Plans:** Vendors can choose from various subscription plans based on their sales volume, product listings, and other factors.
- **Pricing Flexibility:** The platform supports different pricing models, including commission-based and subscription-based models.
- **Automated Billing:** The system automatically handles billing and renewals, ensuring a seamless experience for vendors.

Benefits:

- **Cost Efficiency:** Provides vendors with affordable and flexible pricing options, making it accessible to businesses of all sizes.
- **Revenue Generation:** Supports the platform’s revenue generation through diverse monetization strategies.

UI Mockup

1. Introduction to UI Mockups

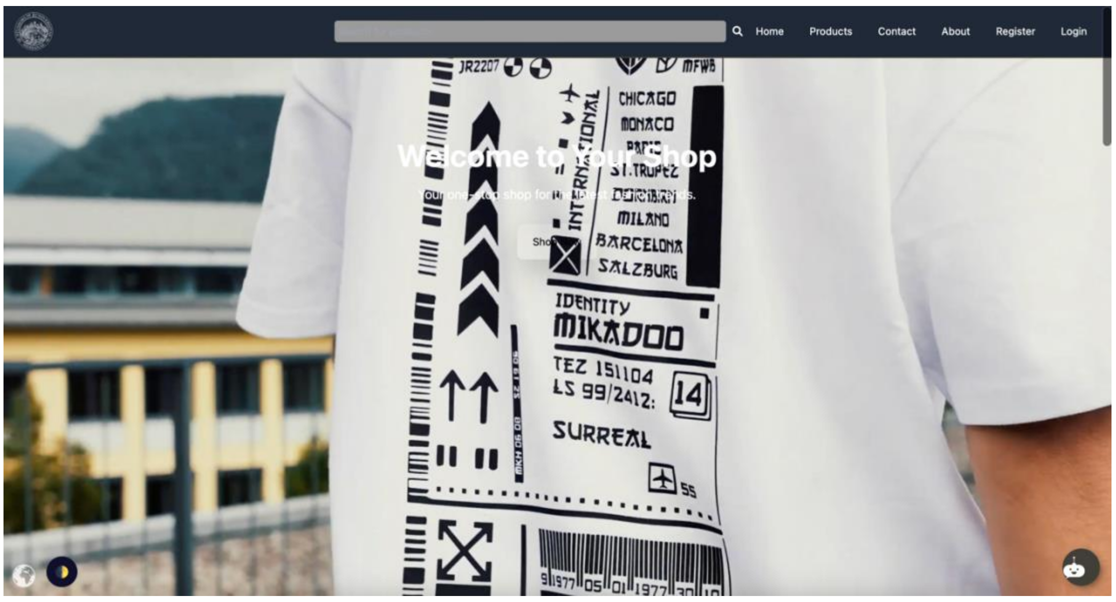
UI mockups provides a visual representation of the user interface before the actual development begins. They help in visualizing the layout, design elements, and overall look and feel of the application. By using mockups, we can ensure that the final product aligns with the desired user experience and functionality. In our project, UI mockups played a significant role in defining the interface for both users and vendors, ensuring that the platform was intuitive, visually appealing, and aligned with modern design standards.

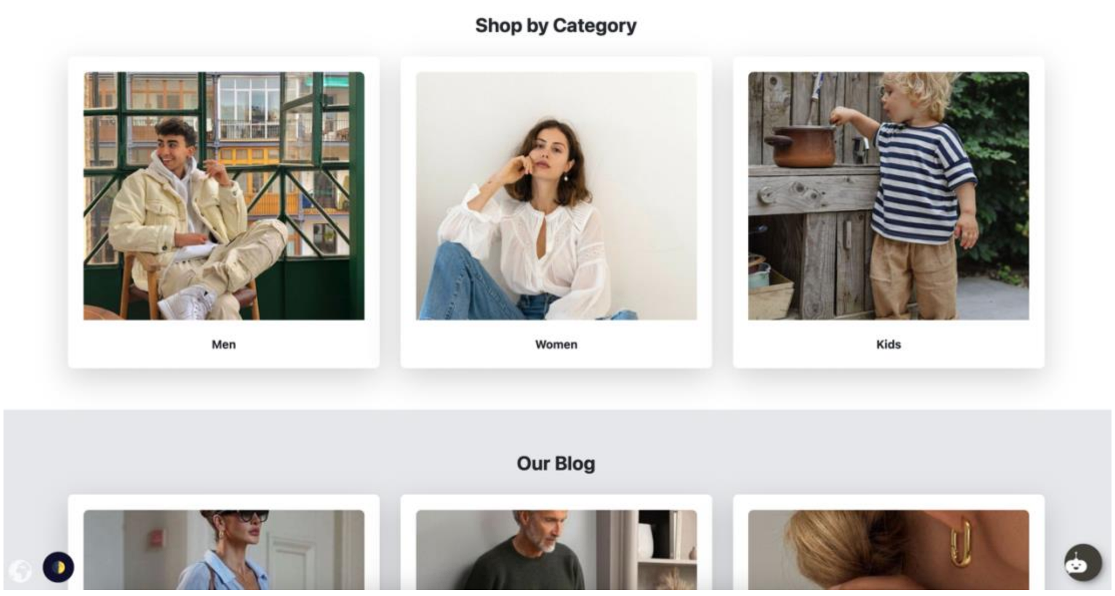
2. Key Screens in the Platform

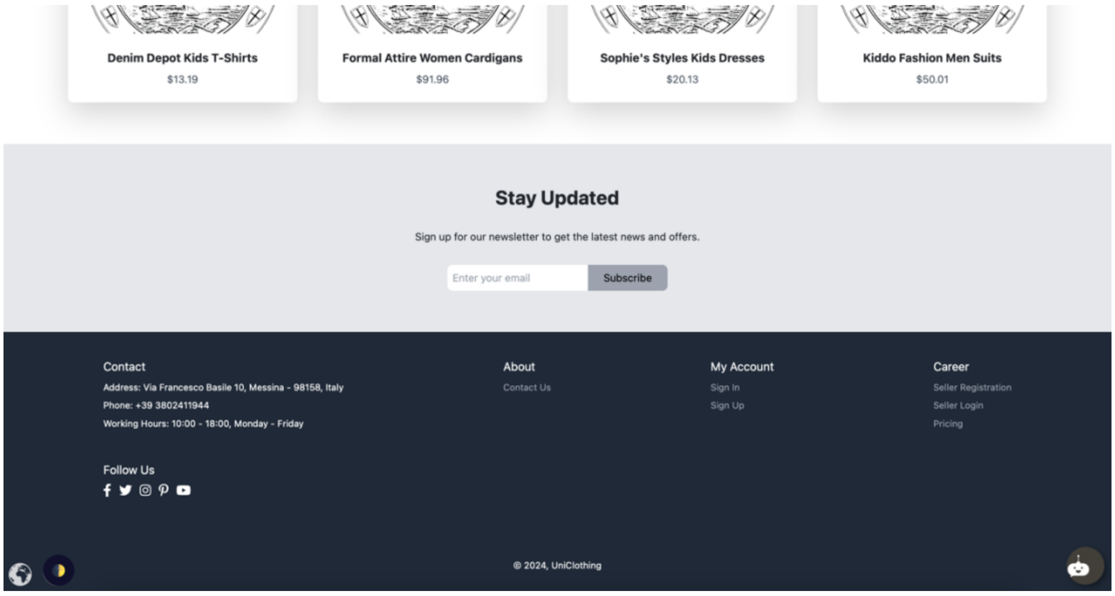
The following key screens were mockuped to guide the development process:

2.1 Home Page

- **Overview:** The home page serves as the gateway to the platform, showcasing featured products, popular categories, and promotional banners. It includes a prominent search bar, language selection dropdown, and links to various sections of the platform such as the user account, cart, and vendor registration.
- **Design Elements:**
  - **Header:** Contains the platform logo, search bar, language toggle, and user account links.
  - **Main Banner:** Displays current promotions or featured products.
  - **Product Categories:** Showcases top categories with images and links.
  - **Footer:** Provides links to important pages like About Us, Contact, and Terms of Service.







Denim Depot Kids T-Shirts
$13.19

Formal Attire Women Cardigans
$91.96

Sophie's Styles Kids Dresses
$20.13

Kiddo Fashion Men Suits
$50.01

Stay Updated

Sign up for our newsletter to get the latest news and offers.

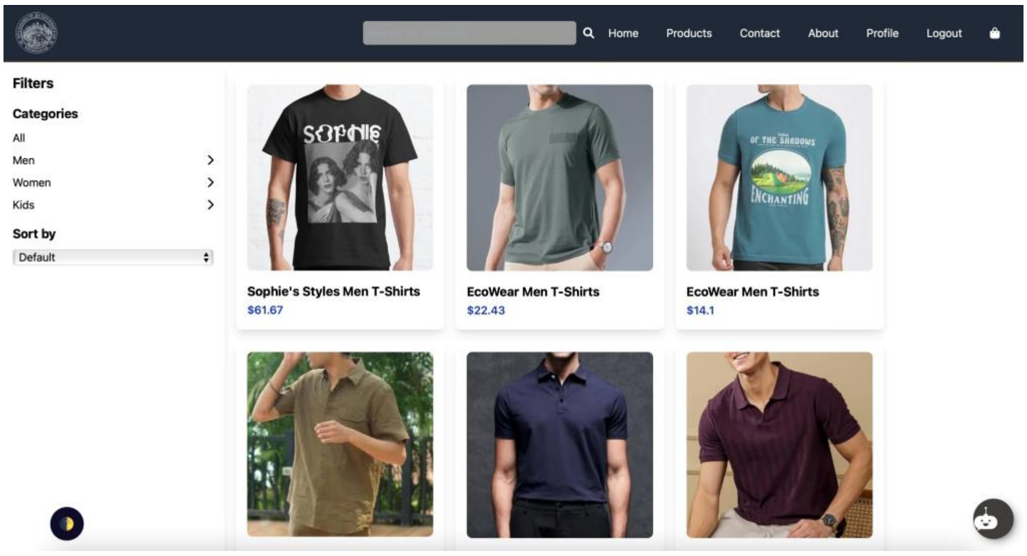
Contact
Address: Via Francesco Basile 10, Messina - 98158, Italy
Phone: +39 3802411944
Working Hours: 10:00 - 18:00, Monday - Friday

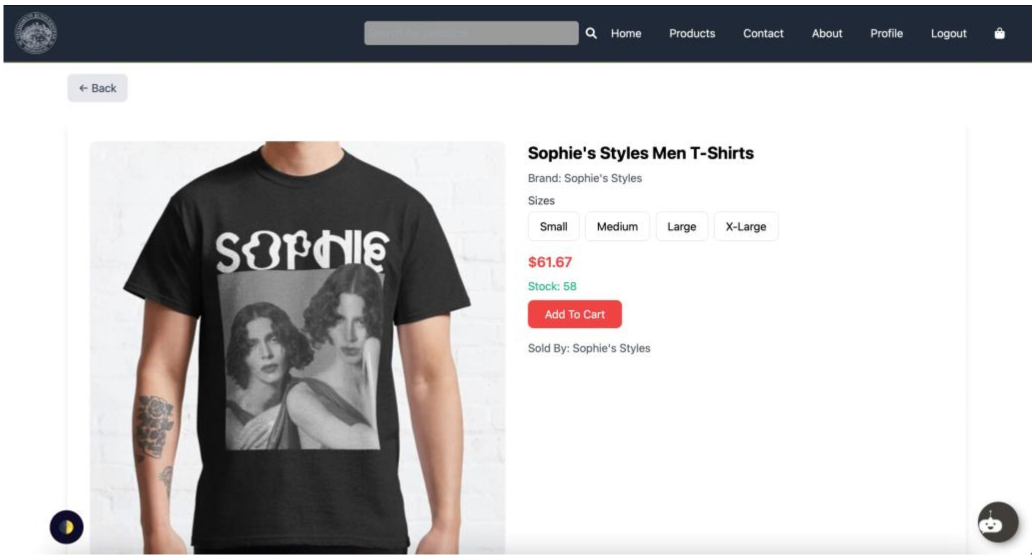
Follow Us

© 2024, UniClothing

2.2 Product Page

- **Overview:** The product page presents detailed information about a specific item, including images, descriptions, pricing, and vendor details. It also includes options for adding the item to the cart or wishlist.
- **Design Elements:**
  - **Image Gallery:** Allows users to view multiple images of the product.
  - **Product Details:** Displays the name, description, price, and availability.
  - **Vendor Information:** Shows details about the seller, including ratings and contact information.
  - **Action Buttons:** Includes options to add the product to the cart

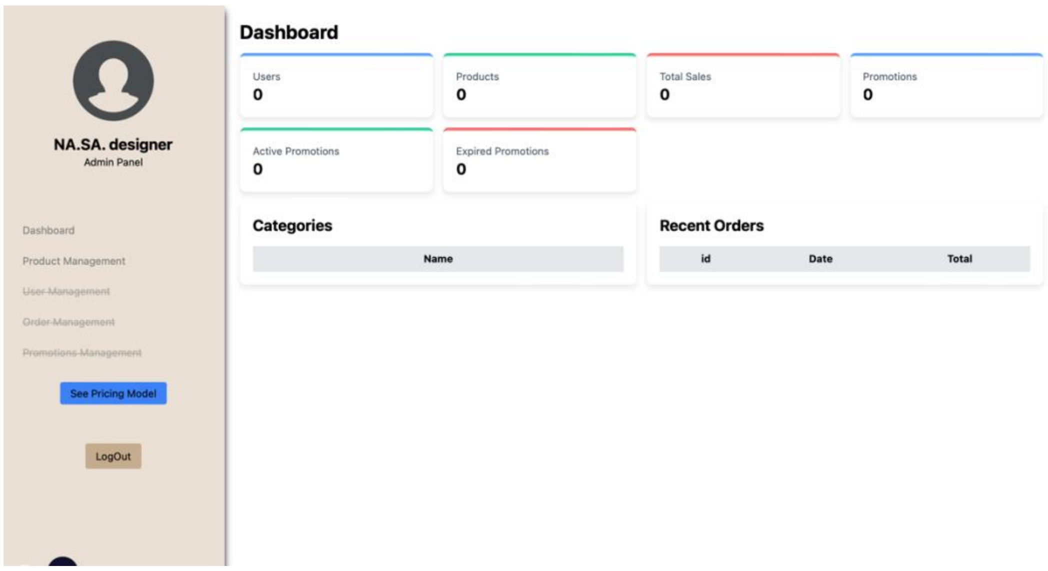




2.3 Vendor Dashboard

- **Overview:** The vendor dashboard provides sellers with an overview of their sales, orders, and product listings. It includes tools for managing inventory, tracking orders, and accessing sales analytics.
- **Design Elements:**
  - **Sales Overview:** Displays sales metrics such as total revenue, number of orders, and best-selling products.
  - **Product Management:** Provides a list of products with options to add, edit, or remove items.

- **Order Management:** Shows recent orders, their statuses, and options for processing returns.



NA.SA. designer
Admin Panel

Dashboard
Product Management
~~User Management~~
~~Order Management~~
~~Promotions Management~~

See Pricing Model

LogOut

Dashboard

Users	Products	Total Sales	Promotions
0	0	0	0

Active Promotions	Expired Promotions
0	0

Categories

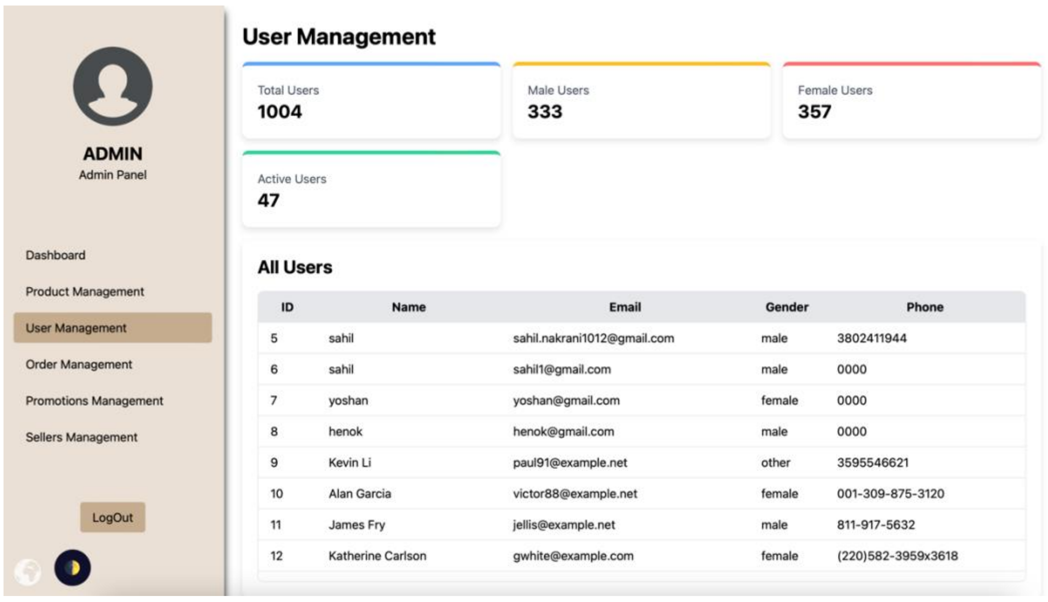
Name

Recent Orders

id	Date	Total

2.4 Admin Panel

- **Overview:** The admin panel offers administrators control over the entire platform, including user and vendor management, platform settings, and monitoring activities.
- **Design Elements:**
  - **User Management:** Allows admins to view, suspend, or delete user accounts.
  - **Vendor Management:** Includes tools for monitoring vendor activities, approving new vendors, and handling disputes.



ADMIN
Admin Panel

Dashboard
Product Management
User Management
Order Management
Promotions Management
Sellers Management

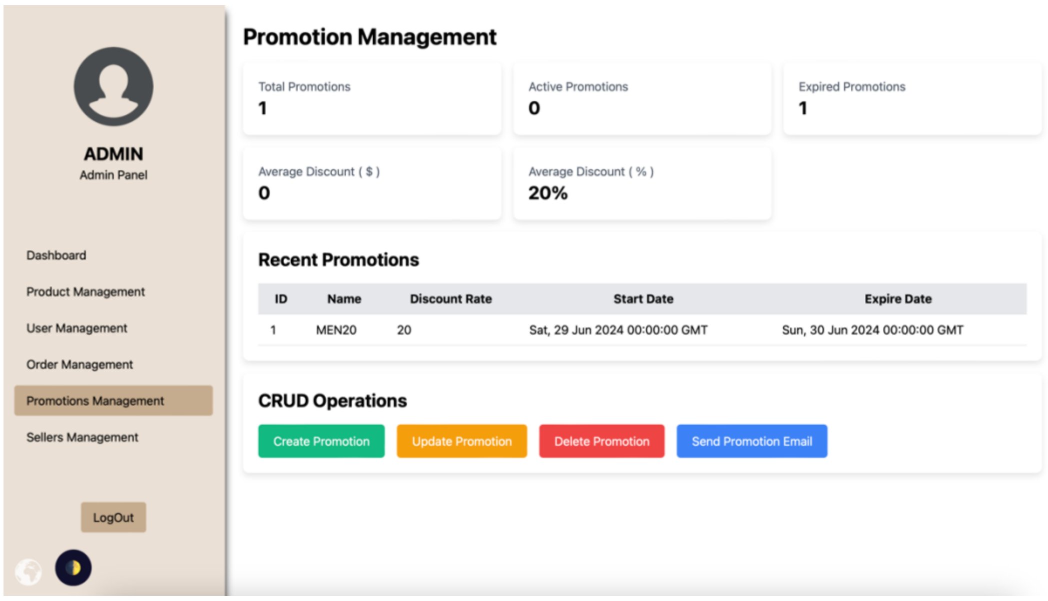
LogOut

User Management

Total Users	Male Users	Female Users	Active Users
1004	333	357	47

All Users

ID	Name	Email	Gender	Phone
5	sahil	sahil.nakrani1012@gmail.com	male	3802411944
6	sahil	sahil1@gmail.com	male	0000
7	yoshan	yoshan@gmail.com	female	0000
8	henok	henok@gmail.com	male	0000
9	Kevin Li	paul91@example.net	other	3595546621
10	Alan Garcia	victor88@example.net	female	001-309-875-3120
11	James Fry	jellis@example.net	male	811-917-5632
12	Katherine Carlson	gwhite@example.com	female	(220)582-3959x3618



ADMIN
Admin Panel

- Dashboard
- Product Management
- User Management
- Order Management
- Promotions Management
- Sellers Management

LogOut

Promotion Management

Total Promotions	Active Promotions	Expired Promotions
1	0	1

Average Discount ($)	Average Discount (%)
0	20%

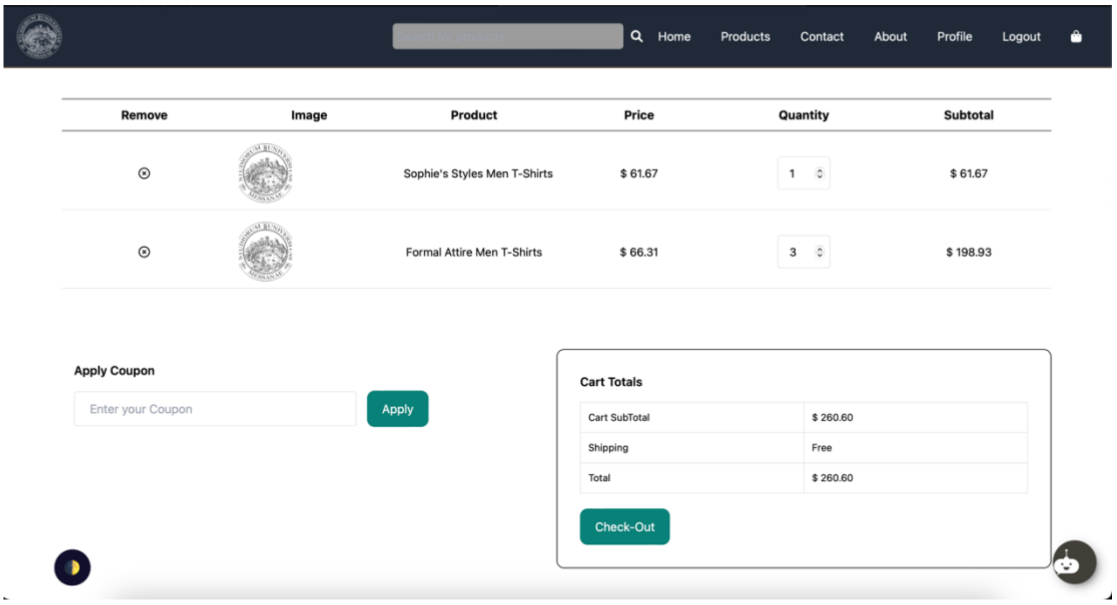
Recent Promotions

ID	Name	Discount Rate	Start Date	Expire Date
1	MEN20	20	Sat, 29 Jun 2024 00:00:00 GMT	Sun, 30 Jun 2024 00:00:00 GMT

CRUD Operations

2.5 Checkout Page

- **Overview:** The checkout page is where users review their order, enter shipping information, and select a payment method. It is designed to be simple and intuitive, guiding users through the purchase process smoothly.
- **Design Elements:**
  - **Order Summary:** Displays the products in the cart, their prices, and the total amount due.
  - **Shipping Information:** Provides fields for users to enter their shipping address.
  - **Payment Options:** Allows users to select their preferred payment method and enter payment details.
  - **Confirmation:** Includes a button to finalize the purchase and displays a confirmation message after successful payment.

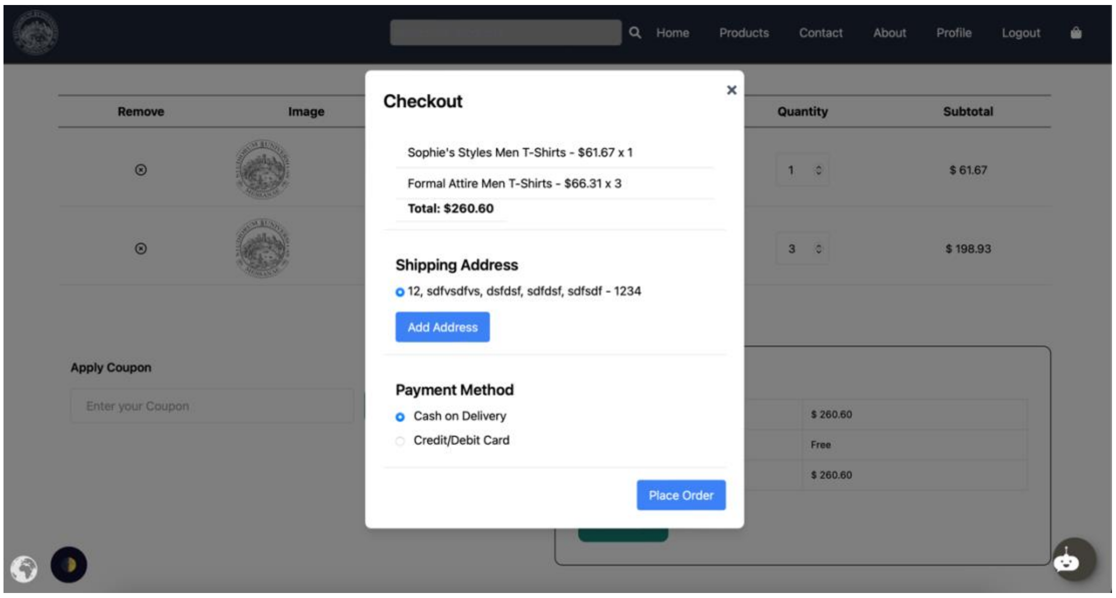


Remove	Image	Product	Price	Quantity	Subtotal
⊙		Sophie's Styles Men T-Shirts	$ 61.67	1	$ 61.67
⊙		Formal Attire Men T-Shirts	$ 66.31	3	$ 198.93

Apply Coupon

Cart Totals

Cart SubTotal	$ 260.60
Shipping	Free
Total	$ 260.60



Checkout

Sophie's Styles Men T-Shirts - $61.67 x 1

Formal Attire Men T-Shirts - $66.31 x 3

Total: $260.60

Shipping Address

12, sdfvsdfvs, dsfdsf, sdfdsf, sdfsdf - 1234

Add Address

Payment Method

(●) Cash on Delivery

( ) Credit/Debit Card

Place Order

Quantity	Subtotal
1	$ 61.67
3	$ 198.93

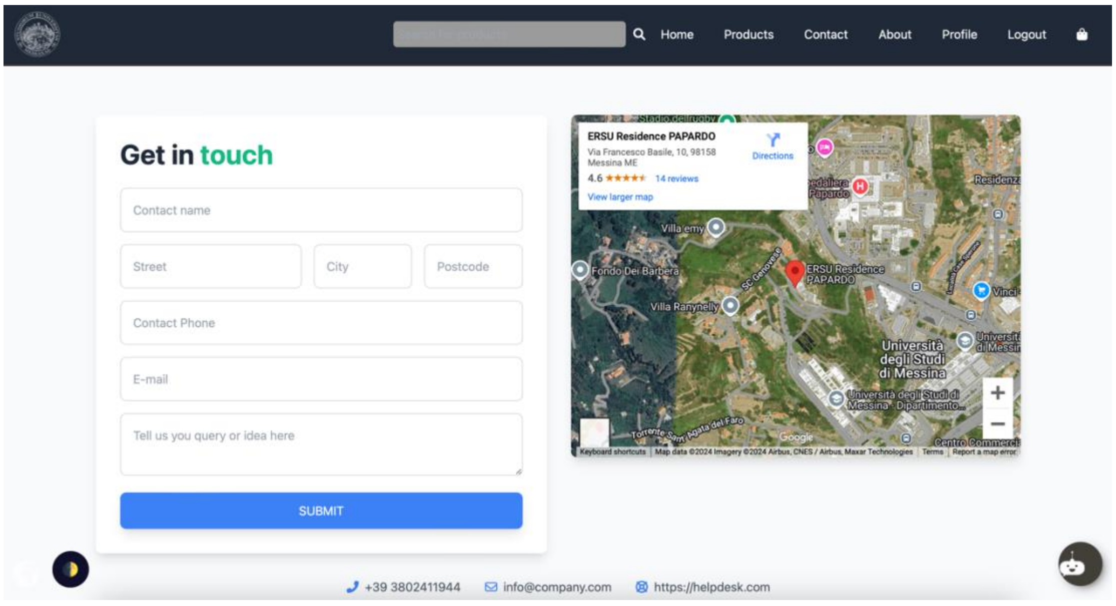
$ 260.60

Free

$ 260.60

2.6 Contact Us Page

- **Overview:** The Contact Us page allows users and vendors to get in touch with the platform's support team. It includes a contact form for queries, a FAQ section, and information about customer support availability. The page is designed to ensure users can easily find assistance when needed.
- **Design Elements:**
  - **Contact Form:** Users can enter their name, email, and message to send directly to the support team.
  - **Support Information:** Displays contact details for customer support, including email, phone number, and social media links.



Get in touch

Contact name

Street | City | Postcode

Contact Phone

E-mail

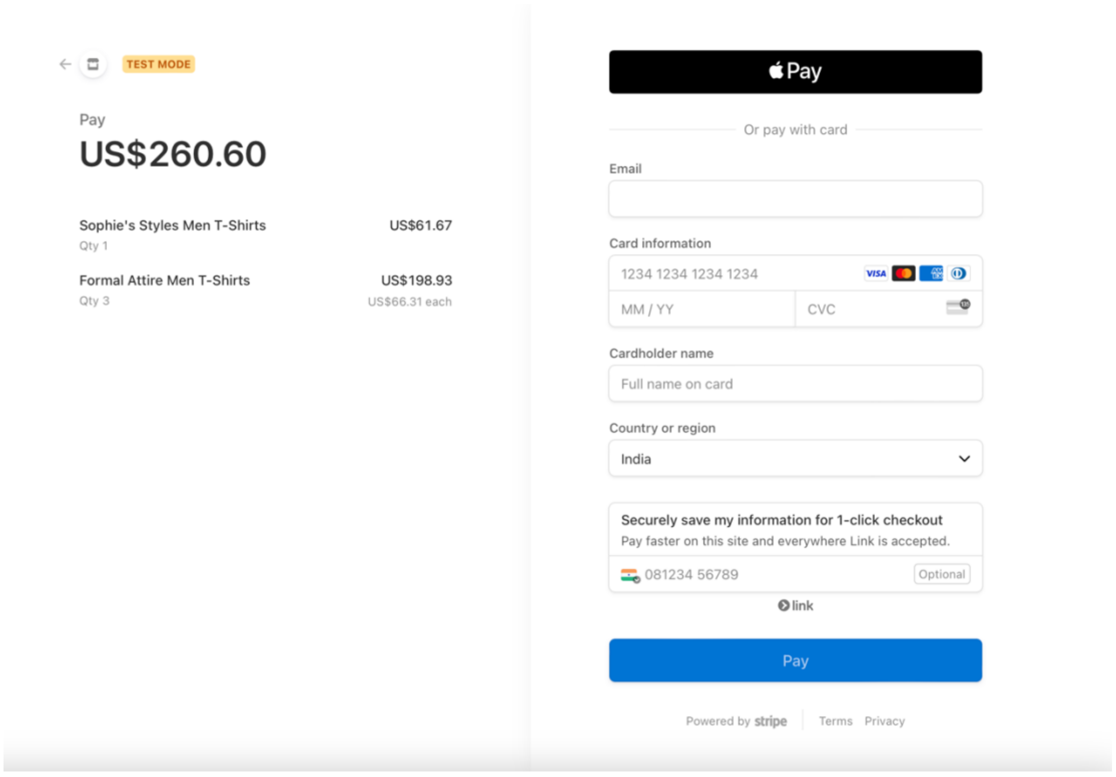
Tell us you query or idea here

SUBMIT

+39 3802411944 info@company.com https://helpdesk.com

2.7 Stripe Payment Method Integration

- **Overview:** The Stripe Payment Method page is part of the checkout process, providing users with a secure and seamless way to complete transactions. This page integrates with Stripe's API for secure payment processing and includes options for users to save payment information for future purchases.
- **Design Elements:**
  - **Payment Form:** Users enter their credit card details, including card number, expiration date, and CVV code.
  - **Security Badge:** Displays icons and messages to reassure users that their payment is processed securely via Stripe.
  - **Save Card Option:** Allows users to securely save their card details for quicker future checkouts.
  - **Payment Confirmation:** Once payment is processed, a confirmation message is displayed, summarizing the order.


TEST MODE

Pay
US$260.60

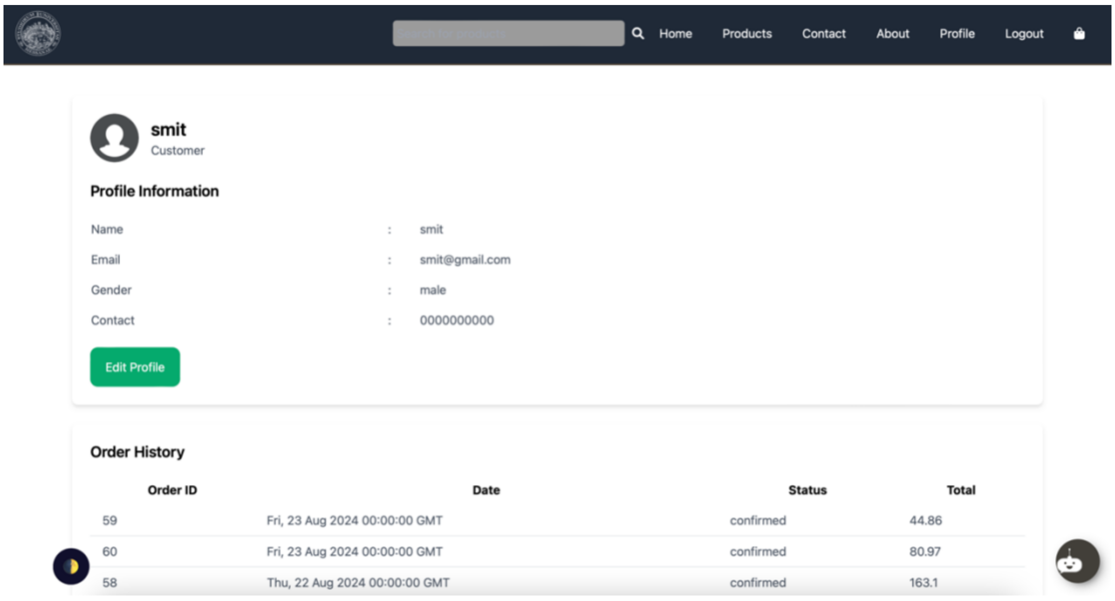
Sophie's Styles Men T-Shirts US$61.67
Qty 1

Formal Attire Men T-Shirts US$198.93
Qty 3 US$66.31 each

Securely save my information for 1-click checkout
Pay faster on this site and everywhere Link is accepted.

2.8 Profile Page

- **Overview:** The Profile page provides users and vendors with a personal dashboard to manage their account information, order history, and preferences. It offers options to update personal details, view past orders, and manage saved addresses and payment methods.
- **Design Elements:**
  - **Personal Information:** Displays fields for users to update their name, email, phone number, and password.
  - **Order History:** Lists previous orders with details such as order date, items purchased, and status.



smit
Customer

Profile Information

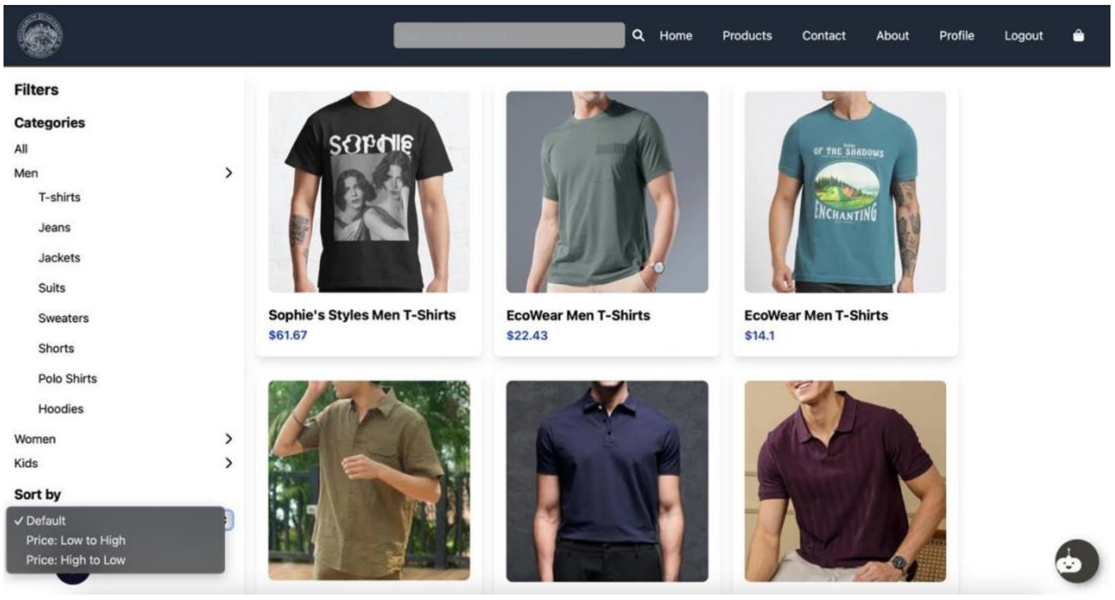
Name	:	smit
Email	:	smit@gmail.com
Gender	:	male
Contact	:	0000000000

Order History

Order ID	Date	Status	Total
59	Fri, 23 Aug 2024 00:00:00 GMT	confirmed	44.86
60	Fri, 23 Aug 2024 00:00:00 GMT	confirmed	80.97
58	Thu, 22 Aug 2024 00:00:00 GMT	confirmed	163.1

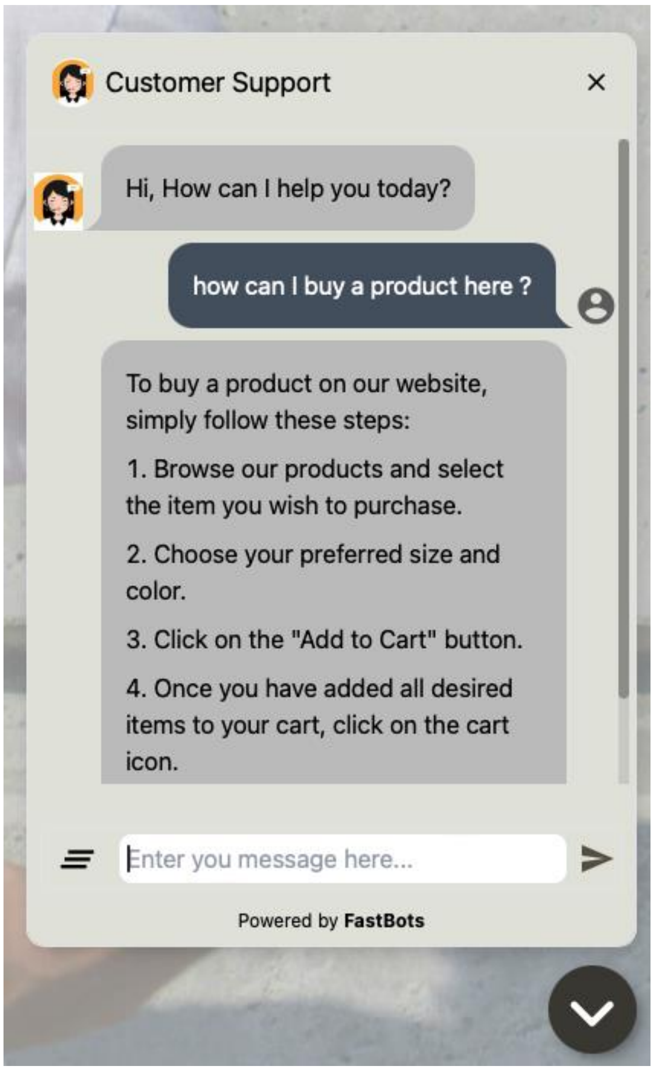
2.9 Product Filter

- **Overview:** The Product Filter page helps users quickly find the products they're looking for by allowing them to filter search results based on categories, price range, vendor, ratings, and availability. The filter sidebar is integrated into the product listing page.
- **Design Elements:**
    - **Category Filter:** Users can select product categories.



2.10 Chatbot Integration

- **Overview:** The Chatbot page is accessible from anywhere on the platform, providing users with 24/7 customer support. The AI-driven chatbot can answer common questions, assist with order tracking, and help users navigate the platform.
- **Design Elements:**
    - **Chat Window:** A small pop-up window where users can type questions.



Hi, How can I help you today?

how can I buy a product here ?

To buy a product on our website, simply follow these steps:

1. Browse our products and select the item you wish to purchase.

2. Choose your preferred size and color.

3. Click on the "Add to Cart" button.

4. Once you have added all desired items to your cart, click on the cart icon.

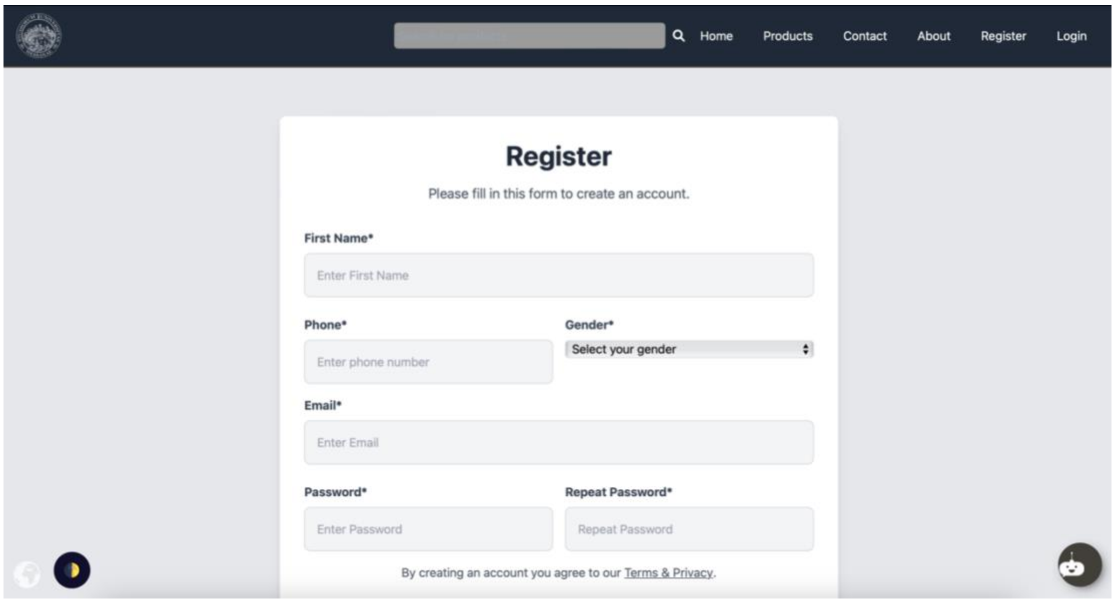
2.11 Multi-language Translator

- **Overview:** The Multi-language Translator feature allows users to select their preferred language from a dropdown menu, ensuring that the platform is accessible to a global audience. This feature is available across all pages and translates the platform's content, including product descriptions, buttons, and forms.
- **Design Elements:**
    - **Language Icon:** Located in the Bottom left, allowing users to choose their preferred language from a list.

    - **Instant Translation:** Once a language is selected, the page content is immediately translated into the chosen language.
    - **Supported Languages:** Includes multiple language options such as English, Italian, Amharic, Gujrati, and others.

2.12 Register and Login Pages

- **Overview:** The Register and Login pages provide users and vendors with secure access to the platform. The registration page allows new users to create an account, while the login page lets existing users and vendors sign in to their accounts securely.
- **Design Elements:**
    - **Register Page:**
        - **Form Fields:** Includes fields for name, email, password, and role selection (user or vendor).



Register

Please fill in this form to create an account.

First Name*

Enter First Name

Phone*

Enter phone number

Gender*

Select your gender

Email*

Enter Email

Password*

Enter Password

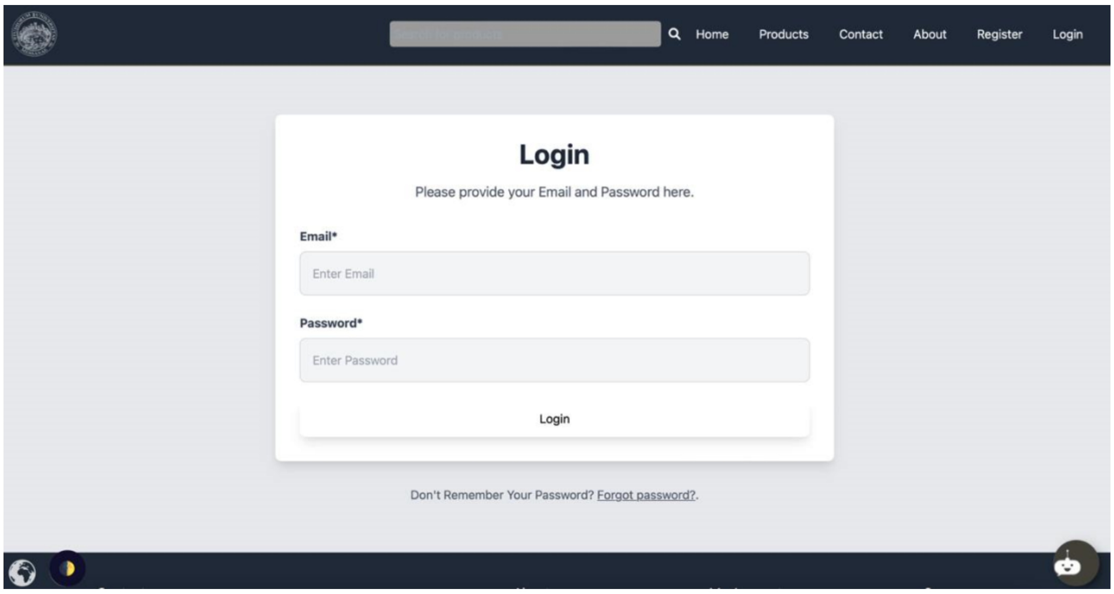
Repeat Password*

Repeat Password

By creating an account you agree to our Terms & Privacy.

    - **Login Page:**

- **Email/Password Fields:** Standard fields for email and password input.
- **Forgot Password Link:** Directs users to a password recovery page.



Login

Please provide your Email and Password here.

Email*

Enter Email

Password*

Enter Password

Login

Don't Remember Your Password? Forgot password?.

3. Importance of UI Mockups

UI mockups were crucial in the development process for the following reasons:

- **Alignment:** Ensured that all team members and stakeholders had a clear understanding of the design and functionality of the platform.
- **Feedback:** Allowed for early feedback from stakeholders, enabling adjustments before development began, saving time and resources.
- **User-Centered Design:** Helped in maintaining a focus on the user experience, ensuring that the interface was intuitive, accessible, and engaging.

Conclusion

1. Project Recap

The development of the Multi-Vendor E-Commerce Marketplace Platform has been a comprehensive and meticulous process, resulting in a robust, scalable, and user-friendly platform. Through several sprints, we successfully implemented a wide range of features aimed at enhancing both user and vendor experiences. From secure authentication and product management to advanced user interface options like multi-language support, AI-driven chatbots, and Dark/Light modes, the platform is well-equipped to cater to a global audience.

2. Achievements

- **User-Centric Features:** The platform was designed with the end-user in mind, ensuring a seamless and enjoyable shopping experience. Features like multi-language support and customizable themes have made the platform accessible and user-friendly.
- **Vendor Empowerment:** By providing vendors with powerful tools for managing their products, tracking sales, and engaging with customers, the platform fosters a dynamic and competitive marketplace.
- **Scalability and Flexibility:** The platform’s architecture, built on Flask with a MySQL database and fronted by Nginx, ensures that it can scale with increasing demand. The use of MVC architecture provides a solid foundation for future enhancements and expansions.

3. Lessons Learned

- **Importance of Continuous Testing:** Regular testing was crucial in identifying and fixing bugs, ensuring that the platform performed reliably across different scenarios and user interactions.
- **Feedback Integration:** Professor’s feedback was invaluable in refining features and aligning the platform with user expectations. Regular sprint reviews and retrospectives allowed the team to adapt quickly to changing requirements.
- **Scalable Design:** Focusing on a scalable architecture from the outset ensured that the platform could grow without significant rework, accommodating more vendors, products, and users.

4. Future Enhancements

While the platform is fully functional and ready for deployment, there are several areas where future enhancements could be made:

- **Expanded Language Support:** Adding more languages to cater to an even broader global audience.
- **Enhanced AI Capabilities:** Continuously training the AI chatbot to handle more complex queries and provide even more personalized support.
- **Advanced Analytics:** Implementing more sophisticated analytics tools for both vendors and administrators to gain deeper insights into sales, user behavior, and platform performance.

5. Final Thoughts

The Multi-Vendor E-Commerce Marketplace Platform is a testament to the team’s dedication, technical expertise, and commitment to delivering a high-quality product. With its rich feature set, user-friendly design, and scalable architecture, the platform is well-positioned to succeed in the competitive world of e-commerce. The journey from concept to final product has been challenging yet rewarding, and the platform stands ready to serve a diverse and growing community of users and vendors.