Sustain Sports

(Eco Friendly Sports Equipment Store)

Submitted By

Ahnaf Sadik Saad

221-35-892

Supervised By

Mr. A.H.M Shahariar Parvez

Associate Professor

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

I declare that this thesis is classified as:

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:

PROJECT DECLARATION LETTER

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia.Dhaka,Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Authors Name: Ahnaf Sadik Saad

Project Title: Sustain Sports(Eco Friendly Sports Equipment Store)

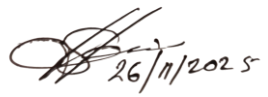
Reasons (i) Before commercialization and startup deployment.

(ii) Proprietary source code and algorithms shall be protected.

(iii) Secrecy of the future development path

Thank you.

Yours faithfully,



(Supervisor's Signature)

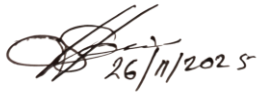
Date: 26/11/2025

Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Daffodil International University* with its copy attached to the thesis.

SUPERVISOR's DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.



(Supervisor's Signature)

Full Name : Mr. A.H.M Shahariar Parvez

Position : Associate Professor

Date : 26/11/2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

(Student's Signature)

Full Name : Ahnaf Sadik Saad

ID Number : 221-35-892

Date :

Sustain Sports
(Eco Friendly Sports Equipment Store)

Ahnaf Sadik Saad

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

NOVEMBER 2025

LETTER OF APPROVAL

APPROVAL

This thesis titled on "**Sustain Sports (Eco-Friendly Sports Equipment Store)**", submitted by **Ahnaf Sadik Saad (ID:221-35-892)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

Dr. S. M. Hasan Mahmud
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Chairman

A.H.M Shahariar Parvez
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 1

Tapushe Rabaya Toma
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Khalid Been md. Badruzzaman Biplob
Lecturer (Senior Scale)
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Dr. Md Sazzadur Rahman
Professor
Institute of Information technology
Jahangirnagar University, Bangladesh

External Examiner

ACKNOWLEDGEMENTS

First of all, I am grateful to The Almighty Allah for allowing us to complete this thesis.

Next, I would like to thank my supervisor, Mr. A.H.M Shahariar Parvez, Associate Professor in the Department of Software Engineering. I am very grateful and indebted to him for his expert, sincere, and valuable guidance and encouragement.

I also want to thank everyone who participated in the survey for this project. Without their passionate involvement and input, the validation survey could not have been successfully conducted.

I would like to express my sincere thanks to all the faculty members of the Department of Software Engineering for their help and support.

Finally, I want to thank my parents for their unconditional support and love. Without them, we would not have come this far.

DEDICATION

I declare that I completed this project with the guidance of Mr. A.H.M Shahariar Parvez, Associate Professor in the Department of Software Engineering at Daffodil International University. I also confirm that neither the whole record nor any part of it has been submitted elsewhere for my degree.

ABSTRACT

"Sustain Sports" is a premier online brand for high-performance sports when combined with a deep environmental concern. This online marketplace offers an attractively arranged platform for a conscientious consumer to facilitate the choice of one who seeks a comprehensive range to the best quality, eco-friendly sports equipment. The app makes the user's experience a very easy, safe, and secure one, so customers can go from product discovery to checkout in a very straightforward and reliable way. In a different manner than other businesses, "Sustain Sports" is very concerned about the environment and hence it is actively creating a circular economy through its integrated Donation & Recycling portal that allows users to give old equipment a new life.The whole network is supported by the powerful administrative core, which allows for the full control of all the products, user accounts, and requests related to sustainability. "Sustain Sports" is a brand new store concept, just like a movement that wants to supply athletes with the necessary sports equipment while at the same time saving the planet on which we make use of.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Programming Interface
BDT	Bangladeshi Taka
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
CPU	Central Processing Unit
NPM	Node Package Manager
MERN	MongoDB, Express.js, React.js, Node.js
JSON	JavaScript Object Notation
HTML	Hypertext Markup Language
SEO	Search Engine Optimization
OS	Operating System
FR	Functional Requirement
UX	User Experience
URL	Uniform Resource Locator
USD	United States Dollar
REST	Representational State Transfer
UI	User Interface
SRS	Software Requirements Specification
SPA	Single Page Application

CHAPTER 1 INTRODUCTION

1.1 Background

This project sits at the nexus of environmental sustainability and e-commerce, two fields that are expanding quickly in the contemporary economy. The lack of a specialized, reliable platform for customers looking for high-performance, environmentally friendly sports equipment is one of the specific gaps it fills in the athletic market. A full-stack MERN application called "Sustain Sports" was designed and developed as part of this project in response. Powered by a clean, modern, and visually appealing design backed up with an advanced Node JS backend for complete site management, this platform has it all – an eclectic collection of high-quality offerings right at your fingertips. The concrete context of the problem to solve, the general vision and very specifically what should be achieved and how far it will not go are described into details in the following parts.

1.1.1 Context and Relevance

This initiative, that monitors the correlation between retail and environmental sustainability, is one models of the rapidly progressing e-commerce industry. One of these is the so-called "conscious consumer" wave, which refers to a demographic segment increasingly caring more about sustainable materials and production approaches based on ethical and environmentally friendly concepts. This trend poses a particular problem for the athletic apparel and sporting goods industry: customers are frequently unable to confirm the "green" claims made by different brands or are forced to choose between high-performance gear and environmentally friendly, sustainable products.

1.1.2 Problem Identification

The problem at the core of this project is that there is no single, trustworthy, and dedicated marketplace for the kind of sports equipment that is both high-performing and environmentally friendly. Certain big retailers might have a few items in their inventory that can be labeled as "green," but a platform doesn't exist that is specially designed around the theme of sustainable sportswear and the brand and product catalog. The above-mentioned gap makes it possible for informed consumers to waste a substantial

amount of time on the internet in search of individual products and manufacturers, but they are never sure how transparent the latter actually are.

1.1.3 Purpose and Justification

The purpose of this project is the creation of a full-stack MERN e-commerce application named Sustain Sports that would be the perfect answer to a problem like this. The project is rationalized by its clear and concise value proposition. It offers a single, dependable, and elegantly designed platform for athletes to perform the necessary transactions not only to meet their requirements for good sports equipment but also to satisfy their ethical principles. The platform, with the help of initiatives like the "Sustainability Hub" centralizing eco-friendly products and giving sustainability information, gains users' confidence and makes their buying process easier.

1.1.4 Scope

This project's scope includes creating the "Sustain Sports" web application from start to finish. This comprises:

- React-built public-facing frontend with product browsing, search, and filtering capabilities.

- a comprehensive e-commerce workflow that includes a multi-step checkout process, a persistent wishlist, and a shopping cart.

- a secure, verified user area where users can manage their profiles, see their order histories, submitting product donation requests and receiving automated promo code rewards for approved donations.

- An extensive admin dashboard with restricted access for handling users, goods, orders, and donations.

- A backend RESTful API that manages all business logic, data persistence, and JWT-based authentication using Node.js, Express, and MongoDB.

1.2 Project Planning and Initiation

Feasibility Study (Step-by-Step)

The "Sustain Sports" project underwent a multi-phase feasibility study to confirm its viability. To identify the main issue and project scope, a preliminary analysis was conducted first. Market research was then conducted to verify the need for a specialized sustainable sports e-commerce platform. Following that, a technical analysis confirmed that the MERN stack was a suitable and up-to-date technological option. Ultimately, a financial feasibility study determined that the project, which mostly relied on open-source technologies and controllable development resources, was feasible.

Phase 1 Preliminary Analysis & Project Scope Definition:

The fundamental issue was discovered during this first stage: a need for a centralized, reliable online store selling high-performance, environmentally friendly athletic apparel. After that, the project scope was established in order to develop the "Sustain Sports" application. Three main elements make up the defined scope:

(1) A public-facing storefront: which features a sustainability content hub and full e-commerce workflow (cart, wish list, check out), browse products by category, search for them, or filter based on the specific merits of planet-friendly interest.

(2) A secure, authenticated user section: For account management, order history browsing, donations.

(3) A comprehensive admin dashboard, with controlled access: To manage orders, donations, users and products.

Phase 2 Market Feasibility Analysis (or Market Research):

The market potential for the project was demonstrated in this phase. Research shows there is a significant and growing trend of "conscious consumerism" where customers actively seek businesses that stand for values such as ethical and environmental responsibility. The market opportunity is there – a dedicated “go-to” platform does not currently exist that speaks to these two parts…and it needs to, as athletes and the health conscious gym-goers are now demanding green. Such gap is expected to be filled by concrete circular economy features like donating services and products, as well as brand trust on the basis of transparency as can be seen in the "Sustainability Hub". The study reveals that there is a strong demand on the market for this type of e-commerce solution.

Phase 3 Technical Feasibility Analysis:

The project is very feasible with current, open-source technologies, according to the technical feasibility study. For this kind of application, the chosen MERN stack—MongoDB, Express.js, React, and Node.js—was thought to be perfect. With the help of the Vite build tool, the frontend is constructed using the React library and styled using Tailwind CSS. A Node.js/Express server that offers a RESTful API and a MongoDB database for data persistence make up the backend. JWT is used to handle security for authorization and authentication. There aren't any major technical obstacles because the required technical knowledge and documentation for these technologies are widely accessible.

Phase 4 Financial Feasibility Analysis:

Due to its complete reliance on open-source technologies, this project has a high financial viability. There are no software licensing fees associated with the MERN stack (MongoDB, Express.js, React, and Node.js). The investment of human hours in creating the application is the main development cost. The only ongoing costs after development would be for a hosting service (to set up the static React frontend, MongoDB database, and Node.js backend) and yearly domain name registration. The project is financially feasible to develop and maintain due to these predictable and low operating costs.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

Three main categories can be used to categorize the "Sustain Sports" platform's target users and stakeholders:

Public Users (Shoppers/Guests): Any visitor to the location makes up this largest group. Their main objective is to read content on the Sustainability Hub or browse, search for, and learn about sustainable sports products.

Registered Users (Customers/Members): Users who have registered are included in this group. In addition to having access to private routes for managing their profile, viewing order history, managing a wishlist, and submitting product donation requests, they have all the features of Public Users.

Administrators (Site Managers): A tiny, internal user base whose database has isAdmin set to true. Using the admin dashboard, they are responsible for overseeing the operations and content of the application.

1.3.2 User profile

Table 1.1: User Profile for Public User (Guest)

User Class	**Note on Characteristics**
Type of user	Primary (The main target audience of the e-commerce site)
Age range	18-55 (Typical e-commerce demographic with an interest in sports and sustainability)
Frequency of use	Varies (from one-time purchase to regular seasonal browsing)
Mandatory	N/A (Use of the public site is voluntary)
Computer experience	Basic to Intermediate. Must be comfortable with online shopping and browsing
Education	Varies. Likely has an interest in or awareness of environmental/sustainability issues.
goal	To find, learn about, and purchase high-quality, sustainable sports equipment.
Language skills	English (All site text is in English)
Number of users	High (The largest and most open-ended user group)
Training	None required; the interface must be intuitive.
Others system use	Other e-commerce sites (e.g., Amazon, Nike, Patagonia), blogs, and review sites.
Way of working	Browses products via categories, uses the search bar, adds items to cart, and reads static content like the "About" or "Sustainability" pages.

Table 1.2: User Profile for Registered User

User Class	**Note on Characteristics**
Type of user	Secondary (A subset of Public Users who have created an account)
Age range	18-55
Frequency of use	Moderate to High (Likely repeat customers or those who want to track orders)
Mandatory	N/A (Registration is optional to browse, but required for checkout/features)
Computer experience	Basic to Intermediate. Must be comfortable with logging in and managing an account.
Education	Varies.
goal	To have a persistent experience: manage their profile, track past orders, save a wishlist, and submit donation requests.
Language skills	English.
Number of users	Moderate (Fewer than total public users).
Training	None required.
Others system use	E-commerce account portals, email clients.
Way of working	Logs in using credentials, accesses private routes, and fills out forms (checkout, profile, donation).

Table 1.3: User Profile for Administrator

User Class	**Note on Characteristics**
Type of user	Secondary (Internal user, Site Manager)
Age range	20-60
Frequency of use	Daily to Weekly (To manage the store operations)
Mandatory	Yes (Use of the admin panel is required for their job)
Computer experience	Intermediate to Advanced. Must be comfortable with web-based dashboards/CMS
Education	Varies (e.g., Business, IT, or Management background).
goal	To manage all aspects of the e-commerce store, including users, products, orders, and donations, to ensure smooth operation.
Language skills	English.
Number of users	Low (A small, trusted group, e.g., 1-5 users).
Training	Minimal training on the specific admin panel is required.
Others system use	Other CMS (Shopify, WordPress), analytics tools, inventory systems.
Way of working	Logs in with a special admin account, accesses the admin-only routes, and uses the dashboard UI to perform CRUD (Create, Read, Update, Delete) operations on data.

1.3.3 Elicitation Process

A variety of elicitation techniques were used to collect and specify the requirements for the "Sustain Sports" project in order to precisely record the needs of all stakeholders:

Stakeholder Interviews: This main technique entailed talking with the project's stakeholders to identify the main objectives and distinctive selling features. The necessity of a "closed-loop" sustainability feature was one of the main conclusions drawn from these interviews. Stakeholders stated that the system needed to provide incentives for users; merely accepting donations was not enough. This resulted in the need for an automated reward system, in which a special promo code is automatically generated and stored in the database upon administrative approval of a donation.

Competitive Analysis: Standard features which users likely expects some are actually added by comparison of another e-commerce platforms. This analysis also justified a set of "baseline" e-commerce capabilities, including more robust shopping cart functionality with total calculations, a permanent wishlist for item storage, and powerful product filtering and search options.

User Personas & Stories: User personas depicted the targeted audience (Public User, Registered User and Admin). Converting vague descriptions into action-orientated User Stories To channel this growth we turned them into actionable User Stories:

- **User Story:** I want a incentive for my donation, as a Registered User this would make me recycle again. (In the table for Donation History)
- **Admin Story:** "As an admin, I want to monitor products to keep the catalog up-to-date." (Added to the Admin Product List).

Prototyping: Visual prototypes and low-fidelity wireframes were created for the primary application user interfaces. This process also helped when designing the reusable UI components such as the Card and Button, the seductive HeroSection and the responsive Navbar before coding began.

1.4 Project Block Diagram

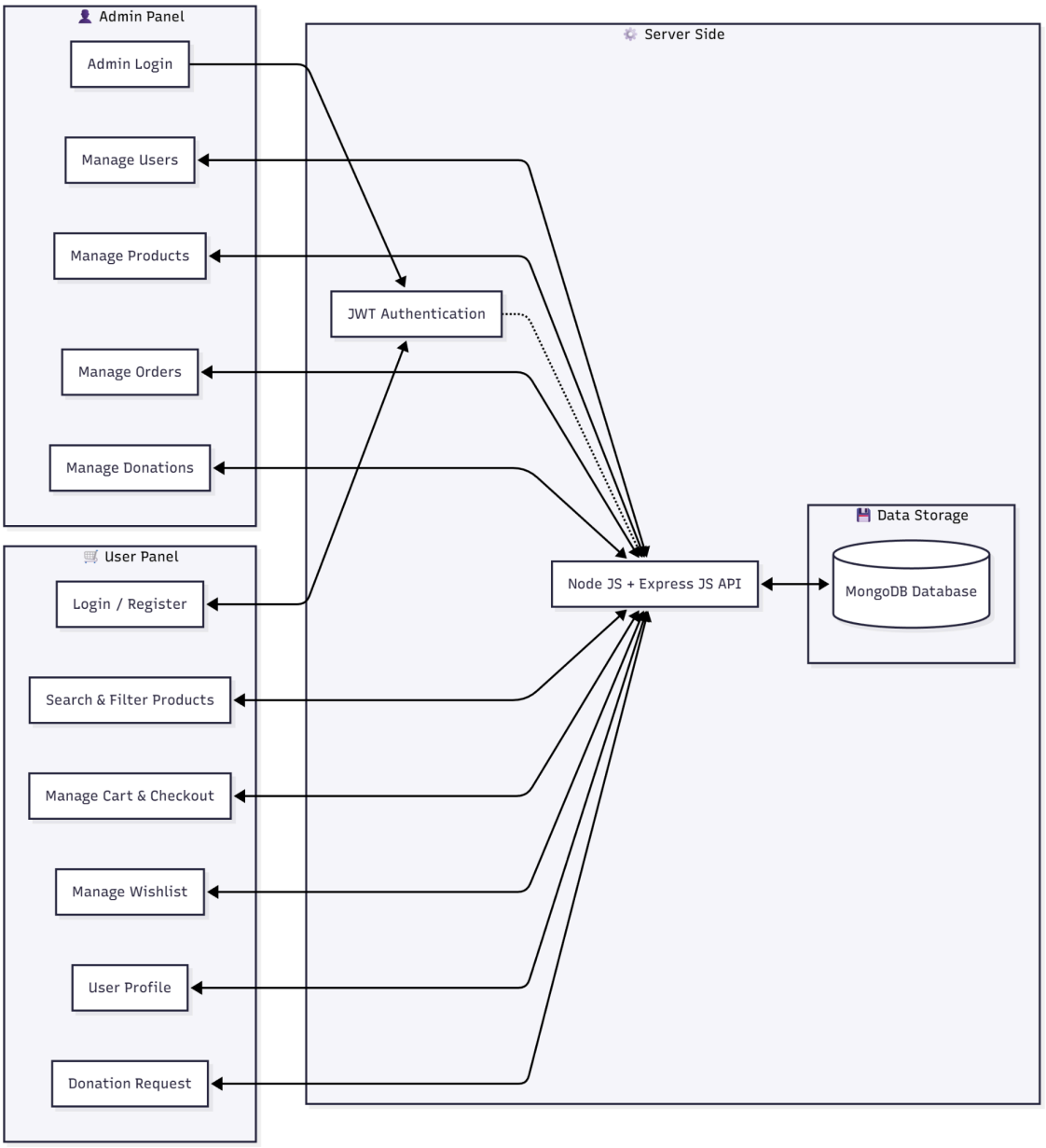


Figure 1.1: System Block Diagram

1.5 System Requirements

1.5.1 Hardware Requirements

Processor (CPU):

Least processor: Intel Core i3 or AMD Ryzen 3 (dual-core processor).

Recommended: Intel Core i5 or AMD Ryzen 5 (quad-core or higher) to realise your development potential.

Memory (RAM):

Minimum: 4 GB RAM.

Recommended: 8Gb RAM or more when developing, running react server and backend concurrently that may lag sometimes.

Storage (Disk Space):

Minimum: 10 GB free disk space for the OS, tooling (VS Code, Node. js), and project files.

Display:

Resolution: 1366 x 768 or above resolution monitor for comfortable code writing and UI design.

1.5.2 Software Requirements

Operating System:

Windows 10/11, macOS (10.15+) or a recent Linux distribution (Ubuntu 20.04+).

Runtime Environment:

Node. js: v20.19.1 (as defined in. nvmrc). This is important for executing backend server as well as frontend build tools.

Package Manager:

npm (Node Package Manager) -> dependencies according to what's in the file package. json.

Code Editor:

(IDE) such as Visual Studio Code (VS Code), or equivalent with JavaScript/React extensions.

Database:

MongoDB: Either local implementation or a cloud-based MongoDB Atlas cluster (you'll need the connection string in. env).

Web Browser:

A recent web browser (Google Chrome, Mozilla Firefox, Microsoft Edge) compatible with the frontend application.

Version Control:

Git: Source code management and collaboration Programmes in LaTeX 2e.

1.5.3 Constraints and Dependencies

1.5.3.1 Constraints

Technological Constraints:

MERN Stack: The application is built on MERN stack(MongoDB, Express. js, React, and Node. js. No SQL databases; no Python or PHP, either. Everything lives and breathes JavaScript.

Runtime Environment: You need Node. js version 20.19.1. Choose something else and you'll likely face compatibility problems with dependencies or build tools.

SPA (Single-Page Application): The frontend utilizes React Router and operates as a single-page application. That means doing SEO is more complicated than with server-side rendered (like Next. js or Nuxt). Special treatment is required for meta tags and indexing.

Hardware Constraints:

Development Resource Usage: Both the frontend Vite and backend Express being run on your machine! A minimum of 4GB RAM would do, but it's safer to have 8GB to ensure applications run smoothly.

Operational Constraints:

Admin-Only Actions: Difficult admin-only actions (promocode generation, modding data etc) can only be performed by admins (those with the powah!.isAdmin permission flag). You, the average user can not even touch these features — ever.

Supported Currencies: All e-commerce transactions are pegged to the USD ($). No other currencies for now. 1. 5. 3. 2 Dependencies.

1.5.3.2 Dependencies

Software Dependencies:

NPM Packages: The npm ecosystem is used by it to build. Some essentials: react and framer-motion on the UI side of things, tailwindcss styling, mongoose for defining data models and jsonwebtoken for handling security. If any of those packages become deprecated or have security vulnerabilities, the team needs to update them immediately.

Database Availability: Everything relies on the MongoDB related to the MONGO_URI. Store and admin panel would not work if db is down or connection incur.

Functional Dependencies:

LocalStorage: The shopping cart and user sessions depend on localStorage at the user's browser. The cart will not persist items across page reloads if you have someone who disables local storage.

How The Reward System Works: Promo codes are only made when someone requests a donation and an admin accepts. Those **t**wo steps are closely intertwined — you can't have a code that doesn't carry both.

1.6 Project Scheduling

Table 1.4: Project Scheduling

Phase	Task / Activity	Duration	Description
Phase 1	**Planning & Analysis**	**Week 1-2**	
	Requirement Elicitation	Week 1	Defining key features (Cart, Donation System) through competitive analysis and stakeholder interviews.
	Feasibility Study	Week 2	evaluating the MERN stack's technical feasibility and the market's need for environmentally friendly equipment.
Phase 2	**System Design**	**Week 3-4**	
	Database Schema Design	Week 3	creating MongoDB models for donations, products, and users.
	UI/UX Prototyping	Week 4	making wireframes for the admin dashboard, hero section, and navigation bar.
Phase 3	**Development (Core)**	**Week 5-8**	
	Backend API Setup	Week 5	Constructing authentication middleware, connecting MongoDB, and configuring the Express server.
	Frontend ImplementationWeek	6-7	Constructing Home, Shop, and Cart React pages and incorporating State Management (Context API).
	Admin Panel & Rewards	Week 8	Creating the automated Donation-to-Promo Code logic and the Admin Dashbord.

Phase 4	**Testing & Refinement**	**Week 9-10**	
	Functional Testing	Week 9	Testing the responsive design, BDT currency toggle, and entire checkout process.
	Bug Fixing	Week 10	Fixing routing problems and making database queries more efficient.
Phase 5	**Documentation & Finalization**	**Week 11-12**	
	Final Report Writing	Week 11-12	Creating the presentation materials and putting together the thorough project report.

1.7 Summary

Chapter 1 laid down the foundation for the “Sustain Sports” project. It began with a big hole in the e-commerce universe: there was no site devoted to selling sustainable athletic gear. In an effort to bridge this gap, the project's scope is aimed at developing a full-stack MERN application which can serve as both a retailer and recycling hub. A technical and commercial feasibility study identified that it was technically and economically viable to implement the project, based on open source technologies like React or MongoDB. We defined specific user personas (Public, Member, Admin) and gathered functional requirements through stakeholder interviews, including the distinct "Circular Economy Reward System". The system's architecture was presented in a block diagram that illustrates the interactions between frontend client, backend API and database. Last but not least we set for ourselves some hard frames (like using the MERN stack, and some hardware requirements) in order to had a well defined plan of action. If there were any doubt that the project is well conceived, needed and do-able, this chapter settles it.

CHAPTER 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

After the first step of Project planning and System analysis, the next chapter discusses "System Analysis" for the "Sustain Sports" as its topic is "sustainable system". The ultimate purpose of this review is to refine the high-level system scope and functionality requirements into a development-specific technical plan. This journey is about detecting the shortcomings of what already exists. Then, we will describe the architecture, flows and behavioral logic of system that we propose. The first task of this chapter will be to identify the limitations in the current system to reinforce the argument for developing the project. And then, we are going to architect the new application in terms of structure and processes so that all our requirements can be captured securely and effectively.

2.2 Functional Requirements

Functional Requirements These are the detailed descriptions of what the "Sustain Sports" system will do for its owner. They define what the system should do. These needs are extracted from user personas and their process of gathering. They are classified by core user roles.

FR01	**View Products**
Description	Shall be able to browse and view all products with filtering and sorting.
Stakeholder	Viewer , Registered User

FR02	**Search Products**
Description	Shall be able to search for products by name.
Stakeholder	Viewer,Registered User

FR03	**View Product Details**
Description	Shall be able to view individual product details, including images and descriptions.
Stakeholder	Viewer,Registered User

FR04	Manage Cart
Description	Shall be able to add/remove/update items in a persistent shopping cart.
Stakeholder	Viewer,Registered User

FR05	Sign Up
Description	Shall be able to **register for a new account** by providing a name, email, and password.
Stakeholder	Viewer

FR06	Sign In
Description	Shall be able to **log in to their account** using an email and password.
Stakeholder	Registered User

FR07	Manage Wishlist
Description	Shall be able to add/remove items from a persistent wishlist.
Stakeholder	Registered User

FR08	View Static Pages
Description	Shall be able to view static pages like "About Us" and "Sustainability".
Stakeholder	Viewer,Registered User

FR09	Complete Checkout
Description	Shall be able to complete the secure checkout process to place an order.
Stakeholder	Registered User

FR10	View Order History
Description	Shall be able to view a history of their past orders and their status.
Stakeholder	Registered User

FR11	Manage Profile
Description	Shall be able to update their personal profile information.
Stakeholder	Registered User

FR12	Submit Donation
Description	Shall be able to submit a product donation request.
Stakeholder	Registered User

FR13	View Reward
Description	Shall be able to see a generated Promo Code for their approved donations.
Stakeholder	Registered User

FR14	Access Dashboard
Description	Shall have access to a secure admin dashboard protected by credentials.
Stakeholder	Admin

FR15	Manage Products
Description	Shall be able to Create, Read, Update, and Delete all products.
Stakeholder	Admin

FR16	Manage Orders
Description	Shall be able to view and manage all user orders, including updating their status.
Stakeholder	Admin

FR17	Manage Users
Description	Shall be able to view and delete non-admin user accounts.
Stakeholder	Admin

FR18	Manage Donations
Description	Shall be able to approve or disapprove donation requests.
Stakeholder	Admin

FR19	Generate Reward
Description	Shall automatically generate a promo code in the database when a donation is "Approved".
Stakeholder	Admin

2.3 Non-Functional Requirements

The "Sustain Sports" system's operational standards, quality attributes, and limitations are defined by non-functional requirements. Instead of defining what the system should do, they outline how it should function.

2.3.1 Performance

Fast Load Times: To guarantee that the first page loads in less than two seconds, the system will make use of code-splitting (React.lazy) and contemporary build tools (Vite).

Effective Backend: To avoid blocking and guarantee that API responses are processed quickly, the backend must make use of asynchronous handlers (express-async-handler).

2.3.2 Reliability

Error Handling: To avoid system crashes, the backend must use specialized error middleware to gracefully handle 404 routes and runtime errors.

Data Persistence: Using localStorage, the contents of the wishlist and shopping cart must remain intact between sessions.

2.3.3 Portability

Cross-Platform (Backend): The Node.js backend works on all platforms, but it needs the particular Node.js v20.19.1 runtime.

Environment Independence (DB): Because the system only needs a working MONGO_URI connection string, it can be used in a variety of database hosting environments, including local and cloud-hosted ones.

2.3.4 Security

Password Hashing: Before being entered into the database, every user password needs to be one-way hashed with bcryptjs.

Authentication: JSON Web Token (JWT) validation middleware is required to safeguard private API routes.

2.3.5 Maintainability

Frontend Code Modularity: To make updates and debugging easier, the frontend codebase should be divided into modular components, pages, and contexts.

Backend Code Modularity: The logic in the backend will be separated into routes, controllers, and models in accordance with the separation of concerns principle.

2.3.6 Usability

Responsive Design: Using Tailwind CSS, the user interface must be completely responsive, adjusting fluidly to desktop, tablet, and mobile screens.

User Feedback: Using a toast notification system, the system must give users immediate, clear feedback when they take certain actions (such as "Added to cart").

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

Use Case Diagram:Sustain Sports System

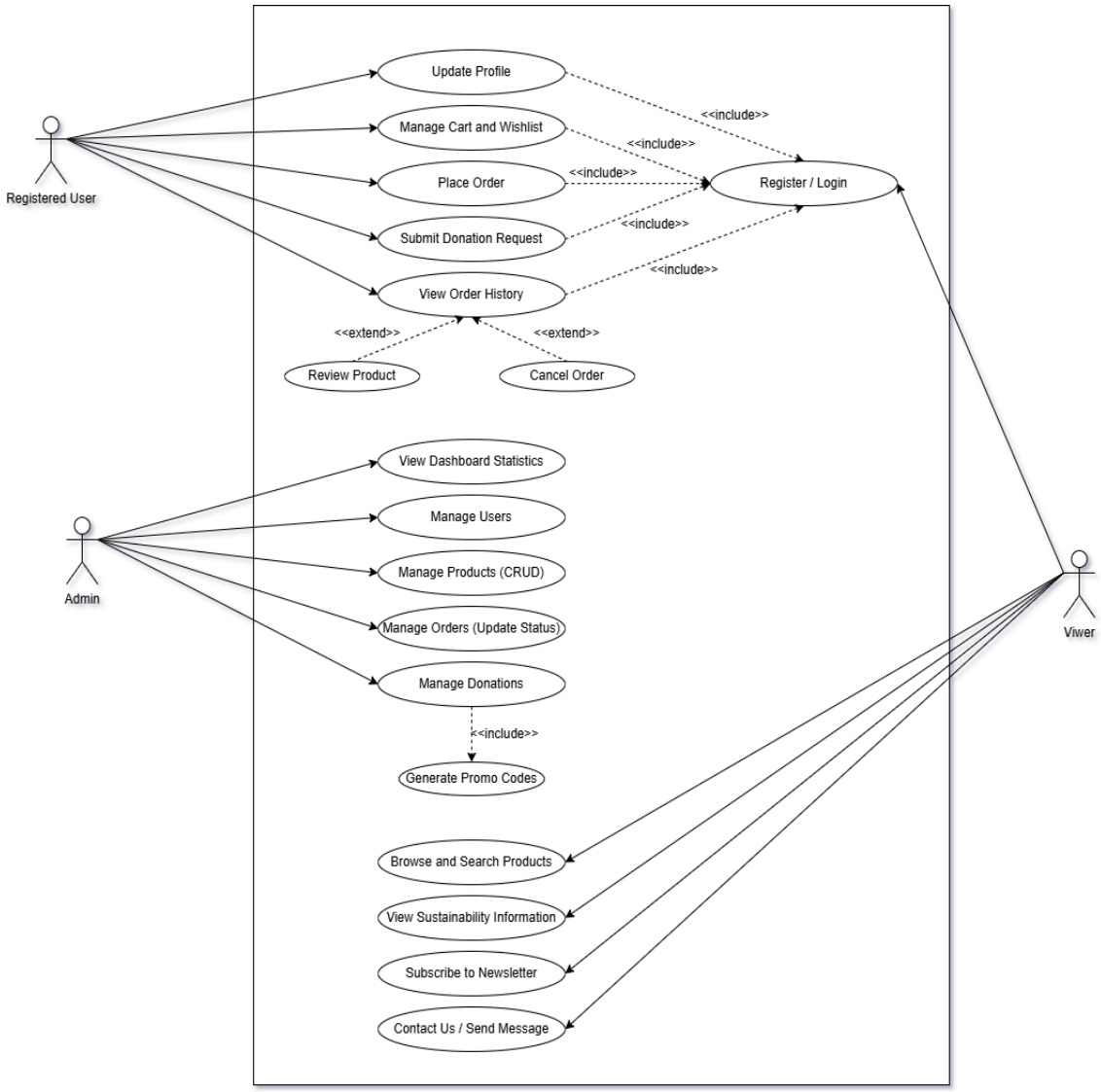


Figure 2.1: Use case Diagram

2.4.2 Case Description

Table 2.1: Case Description-01 Sign Up

<table>
<tr><td>Use Case</td><td>Sign Up</td></tr>
<tr><td>Goal</td><td>A new user (Viewer) can create a secure account to become a Registered User.</td></tr>
<tr><td>Precondition</td><td>The user must have a web browser and internet access to load the "Sustain Sports" website.</td></tr>
<tr><td>Success End Condition</td><td>Notification: "Account created successfully!" (User is logged in and redirected).</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Email already exists" or "Submission Failed".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Viewer
System</td></tr>
<tr><td>Trigger</td><td>The Viewer clicks the "Register" link in the Navbar.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table><tr><td>1.</td><td>Viewer clicks the "Sign Up" link.</td></tr><tr><td>2.</td><td>Provide registration form</td></tr><tr><td>3.</td><td>Enter Information</td></tr><tr><td>4.</td><td>Press “Submit” Button.</td></tr><tr><td>5.</td><td>Information saved</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>1.1</td><td>System Error</td></tr><tr><td></td><td>1.1.a. Please Try Again with Different Credentials!!</td></tr><tr><td>4.1</td><td>The user Did not fill up the details!</td></tr><tr><td></td><td>4.1.a. Checked By the system & Notify by “Please! Fill out the fields”.</td></tr><tr><td>5.1</td><td>The system did not respond</td></tr><tr><td></td><td>5.1.a. Show Error Message.</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>The user Will fill up all the details in 30 minutes.</td></tr>
</table>

Table 2.2: Case Description-02 Sign In

<table>
<tr><td>Use Case</td><td>Sign In</td></tr>
<tr><td>Goal</td><td>An existing user can securely authenticate to access their private account features.</td></tr>
<tr><td>Precondition</td><td>The user must have a pre-existing account.</td></tr>
<tr><td>Success End Condition</td><td>User is authenticated, session token is saved to localStorage, and user is redirected.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Invalid email or password".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Viewer
Registered User</td></tr>
<tr><td>Trigger</td><td>The Viewer clicks the "Login" link in the Navbar.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User navigates to the Login page</td></tr>
<tr><td>2.</td><td>User enters their Email and Password.</td></tr>
<tr><td>3.</td><td>User presses the "Login" button</td></tr>
<tr><td>4.</td><td>System (Frontend) calls <code>AuthContext</code>'s <code>login</code> function.</td></tr>
<tr><td>5.</td><td>System (Backend) finds the user by email.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>User enters an email that does not exist.</td></tr>
<tr><td></td><td>System (Backend) returns an error.</td></tr>
<tr><td></td><td>5.1.b System (Frontend) shows a toast notification: "Invalid email or password".</td></tr>
<tr><td>2.1</td><td>User enters a correct email but an incorrect password.</td></tr>
<tr><td></td><td>6.1.a System (Backend) matchPassword function fails.</td></tr>
<tr><td></td><td>System shows a toast notification: "Invalid email or password"</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The user Will fill up all the details in 30 minutes.</td></tr>
</table>

Table 2.3: Case Description-03 View Product List

<table>
<tr><td>Use Case</td><td>View Products & Filter</td></tr>
<tr><td>Goal</td><td>A user can browse the full catalog of available products, with options to sort and filter them.</td></tr>
<tr><td>Precondition</td><td>User has access to the website.</td></tr>
<tr><td>Success End Condition</td><td>A list of products is successfully displayed to the user.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Error: Could not load products."</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Viewer
System</td></tr>
<tr><td>Trigger</td><td>User clicks the "Products" link in the Navbar.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User clicks the "Products" navigation link.</td></tr>
<tr><td>2.</td><td>System navigates to the <code>/products</code> page.</td></tr>
<tr><td>3.</td><td>System (Frontend) fetches the list of all products from the backend API.</td></tr>
<tr><td>4.</td><td>System (Backend) queries the <code>Product</code> collection in the database</td></tr>
<tr><td>5.</td><td>System (Frontend) displays the products in a grid or list format.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>The backend API fails to respond or returns an error.</td></tr>
<tr><td></td><td>3.1.a System (Frontend) displays an error message to the user.</td></tr>
<tr><td>5.1</td><td>No products are found in the database.</td></tr>
<tr><td></td><td>5.1.a System displays a "No products found" message.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The page must be responsive and display correctly on mobile and desktop devices.</td></tr>
</table>

Table 2.4: Case Description-04 Search Product

<table>
<tr><td>Use Case</td><td>Search Products</td></tr>
<tr><td>Goal</td><td>A user can find specific products by searching for a keyword.</td></tr>
<tr><td>Precondition</td><td>User is on any page with the main Navbar.</td></tr>
<tr><td>Success End Condition</td><td>User is shown a list of products that match their search query.</td></tr>
<tr><td>Failed End Condition</td><td>User is shown a "No products found" message.</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Viewer
System</td></tr>
<tr><td>Trigger</td><td>User types a query into the Navbar search bar and presses "Enter".</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User types "Eco Ball" into the search bar and presses "Enter".</td></tr>
<tr><td>2.</td><td>System navigates the user to the /products page, adding the search query to the URL.</td></tr>
<tr><td>3.</td><td>The Products component loads and fetches all products from the API.</td></tr>
<tr><td>4.</td><td>The component reads the "Eco Ball" query from the URL.</td></tr>
<tr><td>5.</td><td>The component filters the full product list to find only products whose names include "Eco Ball" (case-insensitive).</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>No products in the list match the search query.</td></tr>
<tr><td></td><td>5.1.a System displays a "No products found for your search" message.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The search filtering must be fast and happen client-side after the initial product load.</td></tr>
</table>

Table 2.5: Case Description-05 Manage Cart

<table>
<tr><td>Use Case</td><td>Manage Cart</td></tr>
<tr><td>Goal</td><td>A user can add, remove, and update the quantity of items in their shopping cart.</td></tr>
<tr><td>Precondition</td><td>User is on the website.</td></tr>
<tr><td>Success End Condition</td><td>The CartContext state and localStorage are updated, and the UI reflects the change.</td></tr>
<tr><td>Failed End Condition</td><td>(N/A for these client-side operations).</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Registered User
System</td></tr>
<tr><td>Trigger</td><td>User adds an item from a product page, or changes quantity/removes an item on the Cart page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User navigates to the Cart page (<code>/cart</code>).</td></tr>
<tr><td>2.</td><td>User clicks the "+" or "-" button next to an item.</td></tr>
<tr><td>3.</td><td>System (Frontend) calls the updateQuantity function from CartContext.</td></tr>
<tr><td>4.</td><td>CartContext updates the quantity for that item in its internal state.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>User clicks "Add to Cart" on a product page.</td></tr>
<tr><td></td><td>1.1.a System calls the addToCart function, which adds the product to the items array</td></tr>
<tr><td>1.2</td><td>User clicks the "Delete" icon for an item on the cart page</td></tr>
<tr><td></td><td>1.2.a System calls the removeFromCart function, which filters the item out of the items array</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The cart state must persist after a page refresh, which is ensured by synchronization with localStorage.</td></tr>
</table>

Table 2.6: Case Description-06 Manage Wishlist

<table>
<tr><td>Use Case</td><td>Manage Wishlist</td></tr>
<tr><td>Goal</td><td>A Registered User can save products to a personal wishlist for later viewing.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in (authenticated).</td></tr>
<tr><td>Success End Condition</td><td>The item is added to (or removed from) the user's wishlist state, and the state is saved to localStorage.</td></tr>
<tr><td>Failed End Condition</td><td>(N/A for this client-side operation).</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Registered User
System</td></tr>
<tr><td>Trigger</td><td>User clicks the "Heart" (Wishlist) icon on a product card or views the /wishlist page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User clicks the "Heart" icon on a product.</td></tr>
<tr><td>2.</td><td>System (Frontend) calls the toggleWishlist function from WishlistContext.</td></tr>
<tr><td>3.</td><td>The context checks if the item is already in the list. It is not</td></tr>
<tr><td>4.</td><td>The context adds the product to its internal items array.</td></tr>
<tr><td>5.</td><td>The context saves the updated array to localStorage.</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>The item is already in the wishlist.</td></tr>
<tr><td></td><td>3.1.a The context removes the item from the items array.</td></tr>
<tr><td></td><td>3.1.b System shows a toast: "Item removed from wishlist."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The wishlist state must persist after a page refresh (ensured by localStorage).</td></tr>
</table>

Table 2.7 : Case Description-07 Complete Checkout

<table>
<tr><td>Use Case</td><td>Complete Checkout</td></tr>
<tr><td>Goal</td><td>A Registered User can purchase the items in their cart by providing shipping and billing information.</td></tr>
<tr><td>Precondition</td><td>User must be logged in. User must have at least one item in the cart.</td></tr>
<tr><td>Success End Condition</td><td>An "Order" is created in the database, the cart is cleared, and the user is redirected.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Order placement failed" or "Please fill out all required fields".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Registered User
System</td></tr>
<tr><td>Trigger</td><td>The user clicks the "Proceed to Checkout" button from the Cart page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User clicks "Proceed to Checkout".</td></tr>
<tr><td>2.</td><td>System (PrivateRoute) checks if the user is logged in.</td></tr>
<tr><td>3.</td><td>System displays the Checkout form (/checkout).</td></tr>
<tr><td>4.</td><td>User fills in Shipping, Billing, and Payment info.</td></tr>
<tr><td>5.</td><td>User clicks "Place Order".</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>2.1</td><td>User is not logged in.</td></tr>
<tr><td>2.1 .a</td><td>System intercepts the request and redirects the user to the /login page.</td></tr>
<tr><td>4.1</td><td>User did not fill in all required fields.</td></tr>
<tr><td>4.1 .a</td><td>System shows a validation error: "Please fill out all fields".</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The checkout process must be accessible only to authenticated users.</td></tr>
</table>

Table 2.8: Case Description-08 Manage Profile

<table>
<tr><td>Use Case</td><td>Manage Profile</td></tr>
<tr><td>Goal</td><td>A Registered User can update their personal information</td></tr>
<tr><td>Precondition</td><td>The user must be logged in.</td></tr>
<tr><td>Success End Condition</td><td>Notification: "Profile Updated Successfully". User info in localStorage is updated.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Error updating profile".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Registered User
System</td></tr>
<tr><td>Trigger</td><td>User navigates to the "/profile" page from the Navbar dropdown.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>System displays the Profile form, pre-filled with the user's current name and email.</td></tr>
<tr><td>2.</td><td>User types a new name and/or a new password.</td></tr>
<tr><td>3.</td><td>User presses the "Update Profile" button.</td></tr>
<tr><td>4.</td><td>System (Frontend) calls AuthContext's updateUserProfile function.</td></tr>
<tr><td>5.</td><td>System (Frontend) sends a PUT request to the backend API (/api/users/profile) with the new data.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>5.1</td><td>The API request fails (e.g., server error).</td></tr>
<tr><td>5.1 .a</td><td>a System (Frontend) shows an error toast: "Update failed".</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The user's new password must be securely hashed before being saved.
The user's session token must be updated if their details change.</td></tr>
</table>

Table 2.9 : Case Description-09 Submit Donation Request

<table>
<tr><td>Use Case</td><td>Submit Donation Request</td></tr>
<tr><td>Goal</td><td>A Registered User can submit their used gear for recycling to earn a reward.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in.</td></tr>
<tr><td>Success End Condition</td><td>Notification: "Donation Request Submitted!".</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Please fill in all fields" or "Submission Failed".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Registered User
System</td></tr>
<tr><td>Trigger</td><td>The user navigates to the "/recycling" page and fills out the donation form.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>User opens the "Recycling" page.</td></tr>
<tr><td>2.</td><td>User enters the "Item Name," "Item Condition," and "Description" in the form.</td></tr>
<tr><td>3.</td><td>User presses the "Submit Donation Request" button.</td></tr>
<tr><td>4.</td><td>ystem (Frontend) sends the form data (with auth token) to the backend API</td></tr>
<tr><td>5.</td><td>System (Backend) validates the user is authenticated.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>The user did not fill in all required fields.</td></tr>
<tr><td>3.1 .a</td><td>The system shows a toast notification: "Please fill in all fields".</td></tr>
<tr><td>4.1</td><td>The user is not logged in.</td></tr>
<tr><td>4.1 .a</td><td>The system redirects the user to the Login page.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>The form submission must be protected by authentication.</td></tr>
</table>

Table 2.10: Case Description-10 Manage Products

<table>
<tr><td>Use Case</td><td>Manage Products</td></tr>
<tr><td>Goal</td><td>An Admin can Create, Read, Update, and Delete products in the store.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in as an Admin.</td></tr>
<tr><td>Success End Condition</td><td>The product list in the database is modified (created, updated, or deleted).</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Product update failed" or "Product not found."</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System</td></tr>
<tr><td>Trigger</td><td>Admin navigates to the Admin Dashboard -> Products page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Admin clicks "Edit" on a product in the list.</td></tr>
<tr><td>2.</td><td>System provides the AdminProductEdit page, pre-filled with that product's data.</td></tr>
<tr><td>3.</td><td>Admin changes the price, name, description, or other fields.</td></tr>
<tr><td>4.</td><td>Admin presses the "Update" button.</td></tr>
<tr><td>5.</td><td>System (Frontend) sends a PUT request with the new data to the backend API</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Admin clicks "Create Product" instead of "Edit".</td></tr>
<tr><td>1.1 .a</td><td>System (Backend) creates a new "Sample Product" document.</td></tr>
<tr><td>1.1 .b</td><td>System redirects the Admin to the "Edit" page for that new product to fill in the details.</td></tr>
<tr><td>1.2</td><td>Admin clicks the "Delete" button on the product list.</td></tr>
<tr><td>1.2 .a</td><td>System sends a DELETE request to the API.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Only authorized Admins can perform product management.</td></tr>
</table>

Table 2.11: Case Description-11 Manage Users

<table>
<tr><td>Use Case</td><td>Manage Users</td></tr>
<tr><td>Goal</td><td>An Admin can remove a non-admin user from the system.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in as an Admin.</td></tr>
<tr><td>Success End Condition</td><td>Notification: "User removed". The user is deleted from the database.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Cannot delete an admin user" or "User not found".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System</td></tr>
<tr><td>Trigger</td><td>Admin navigates to the Admin Dashboard -> Users page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Admin opens the "Users" page.</td></tr>
<tr><td>2.</td><td>System displays a table of all registered users.</td></tr>
<tr><td>3.</td><td>Admin finds a non-admin user and clicks the "Delete" (trash) icon.</td></tr>
<tr><td>4.</td><td>System shows a confirmation dialog. Admin clicks "Confirm".</td></tr>
<tr><td>5</td><td>System (Frontend) sends a DELETE request to the backend API</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>Admin attempts to delete another Admin user.</td></tr>
<tr><td>3.1 a</td><td>System (Backend) deleteUser function returns an error: "Cannot delete an admin user".</td></tr>
<tr><td>3.1 .b</td><td>System (Frontend) shows an error toast with the message.</td></tr>
<tr><td>5.1</td><td>A non-admin user attempts to call the API directly.</td></tr>
<tr><td>5.1 .a</td><td>The adminMiddleware blocks the request and returns an "Unauthorized" error.</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>This function must be accessible only to authenticated Admins.</td></tr>
</table>

Table 2.12: Case Description-12 Manage Donations

<table>
<tr><td>Use Case</td><td>Manage Donations</td></tr>
<tr><td>Goal</td><td>An Admin can approve a donation, which automatically generates a reward code for the user.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in as an Admin.</td></tr>
<tr><td>Success End Condition</td><td>The donation status is "Approved" and a promoCode is saved to the database.</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Error updating status".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System, Registered User (receives reward)</td></tr>
<tr><td>Trigger</td><td>Admin navigates to the Admin Dashboard -> Donations page.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Admin opens the "Donations" page.</td></tr>
<tr><td>2.</td><td>System displays a table of all donation requests.</td></tr>
<tr><td>3.</td><td>Admin finds a "Pending" request and clicks "Approve".</td></tr>
<tr><td>4.</td><td>System (Frontend) sends a PUT request to the backend API</td></tr>
<tr><td>5.</td><td>System (Backend) validates the user is an Admin.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>Admin clicks "Disapprove".</td></tr>
<tr><td>3.1 .a</td><td>a System updates the status to "Disapproved" and does not generate a promo code.</td></tr>
<tr><td>5.1</td><td>A non-admin user attempts to call the API.</td></tr>
<tr><td>5.1 .a</td><td>The adminMiddleware blocks the request and returns an "Unauthorized" error</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>This action must be protected by both authentication and admin authorization.</td></tr>
</table>

2.4.3 Activity Diagram

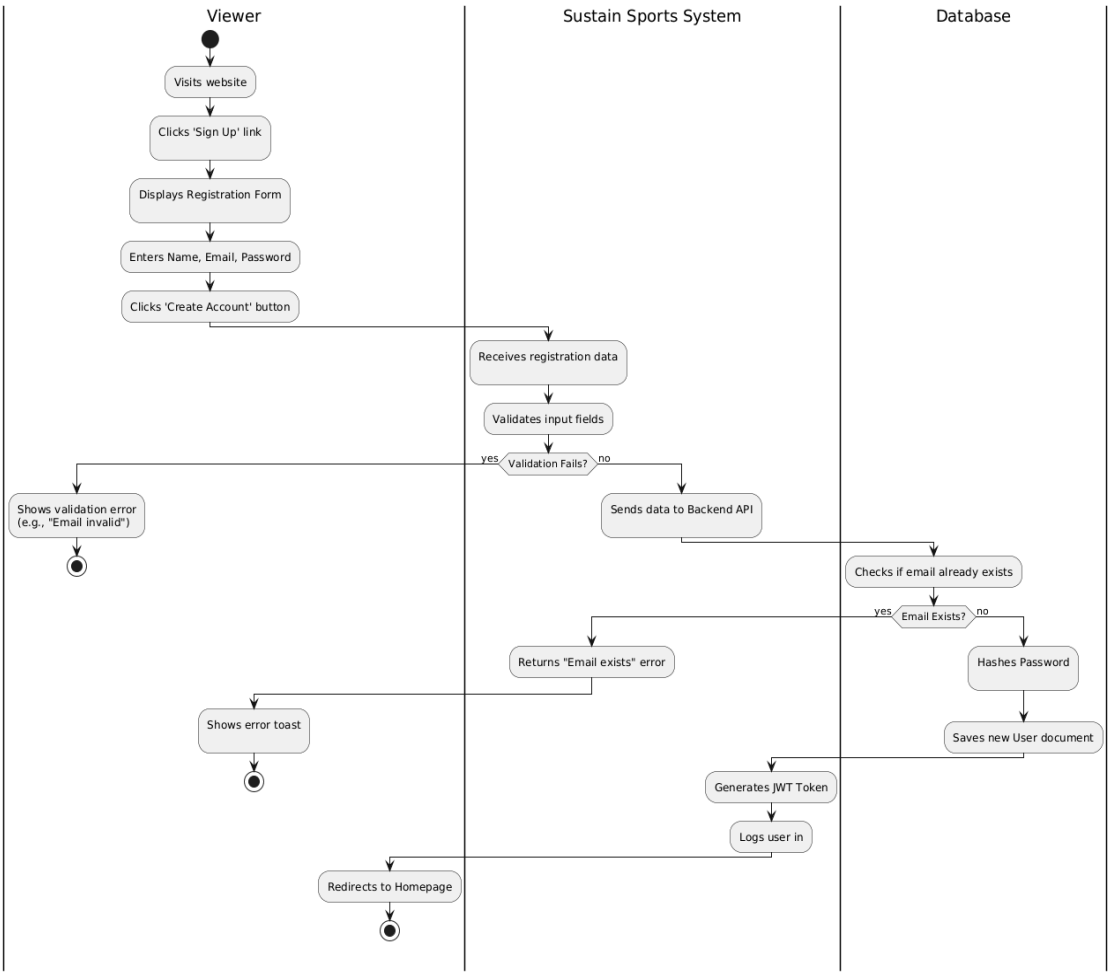


Figure 2.2.1 Activity Diagram : Sign Up

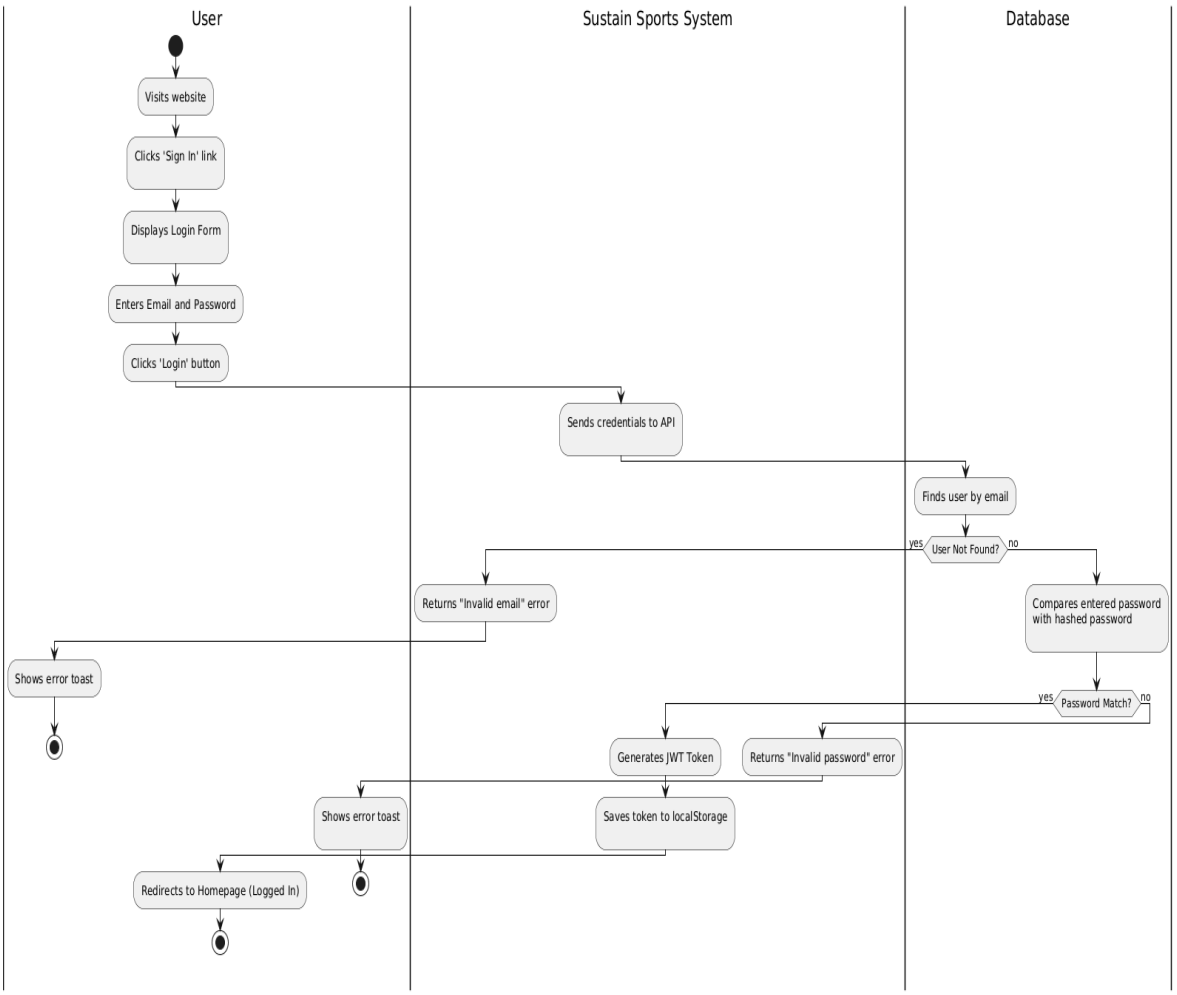


Figure 2.2.2 : Activity Diagram Sign In

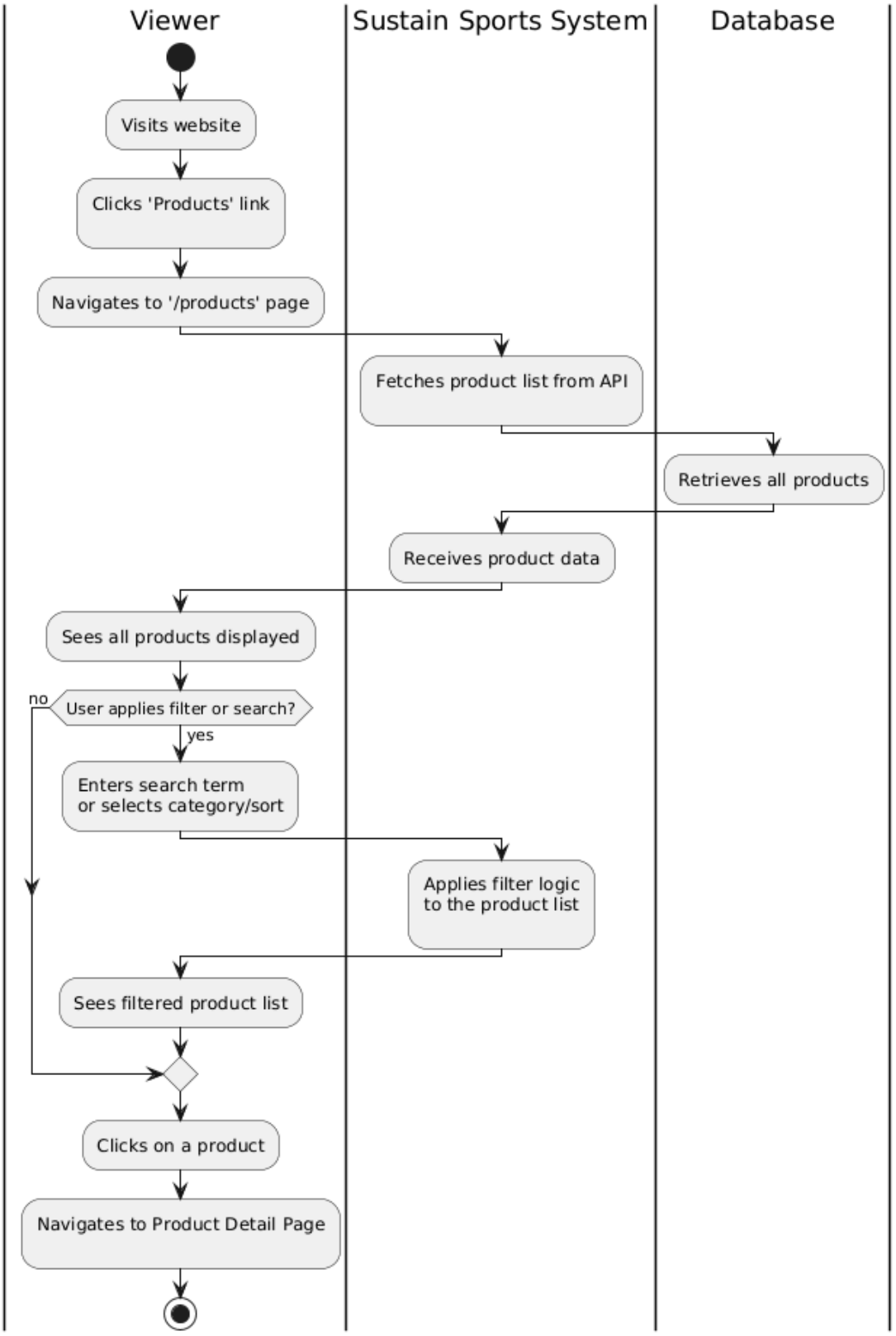


Figure 2.2.3: Activity Diagram Browse Product & Filtering

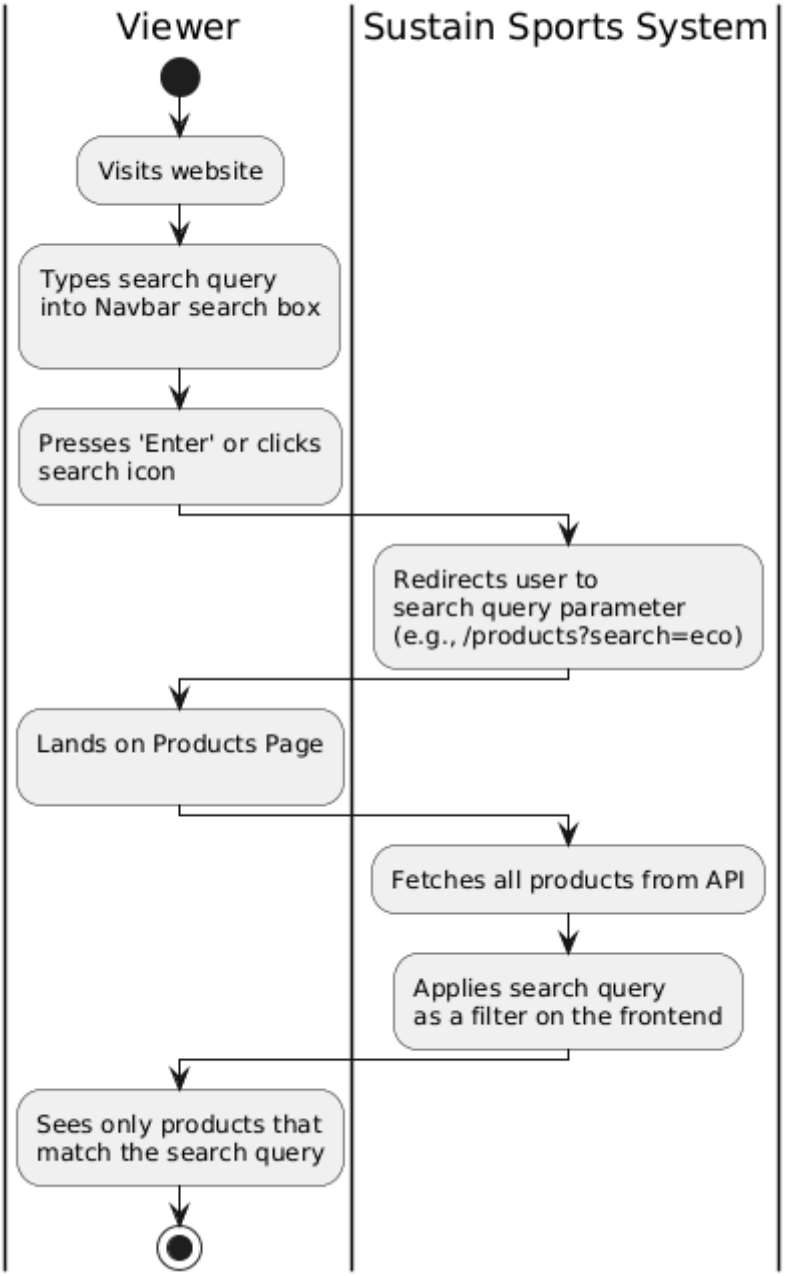


Figure 2.2.4: Activity Diagram Search Product

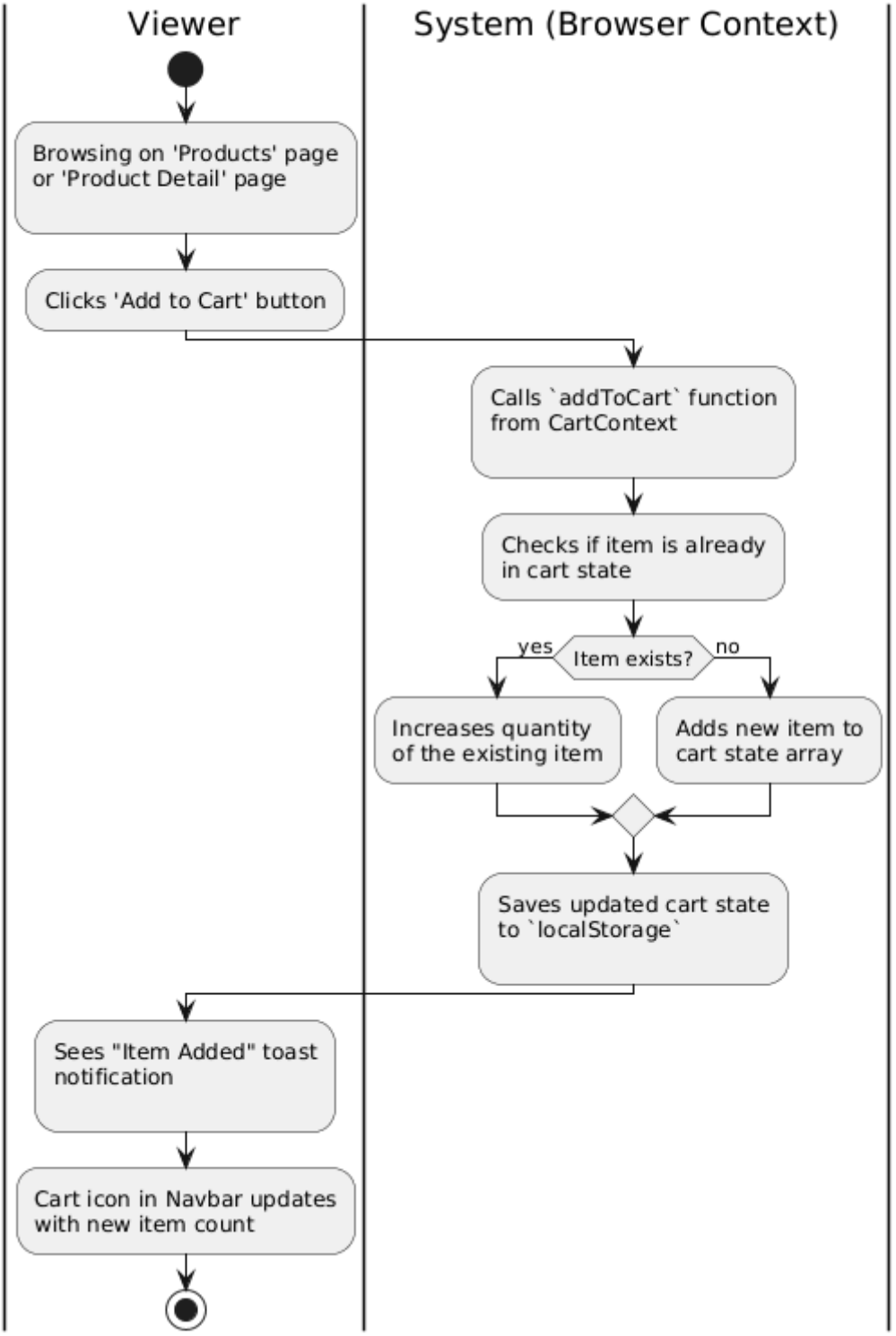


Figure 2.2.5 Activity Diagram Add to Cart

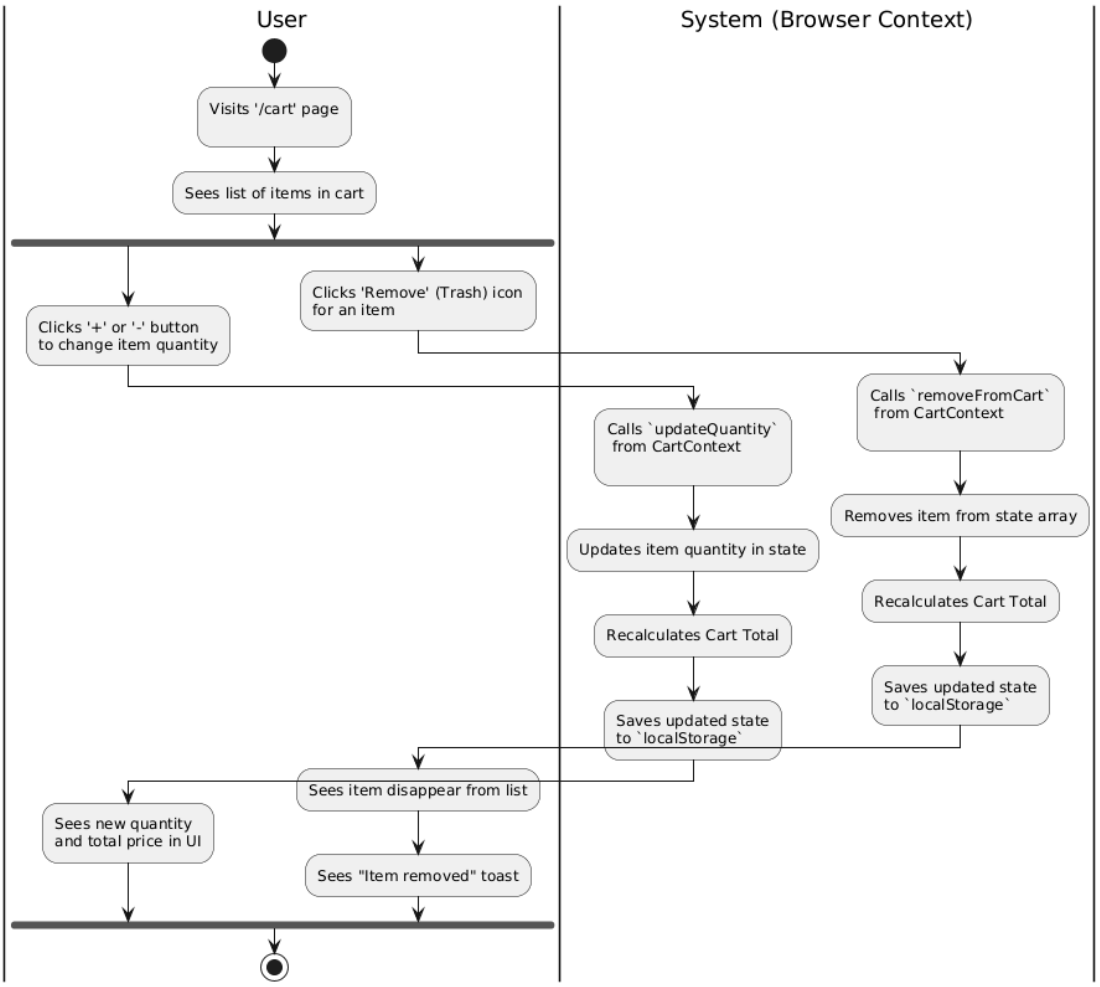


Figure 2.2.6:Activity Diagram Manage Cart

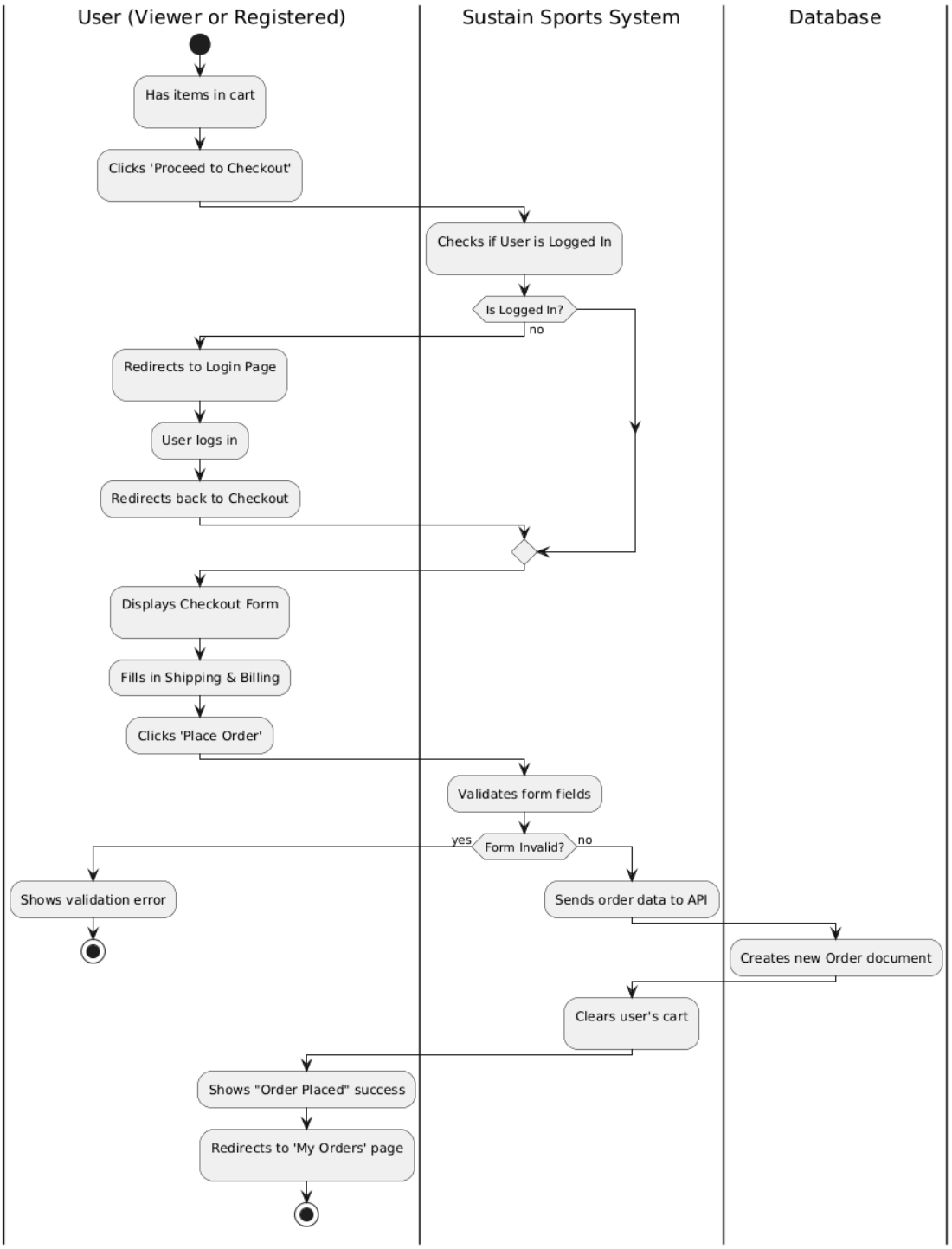


Figure 2.2.7: Activity Diagram Product Checkout

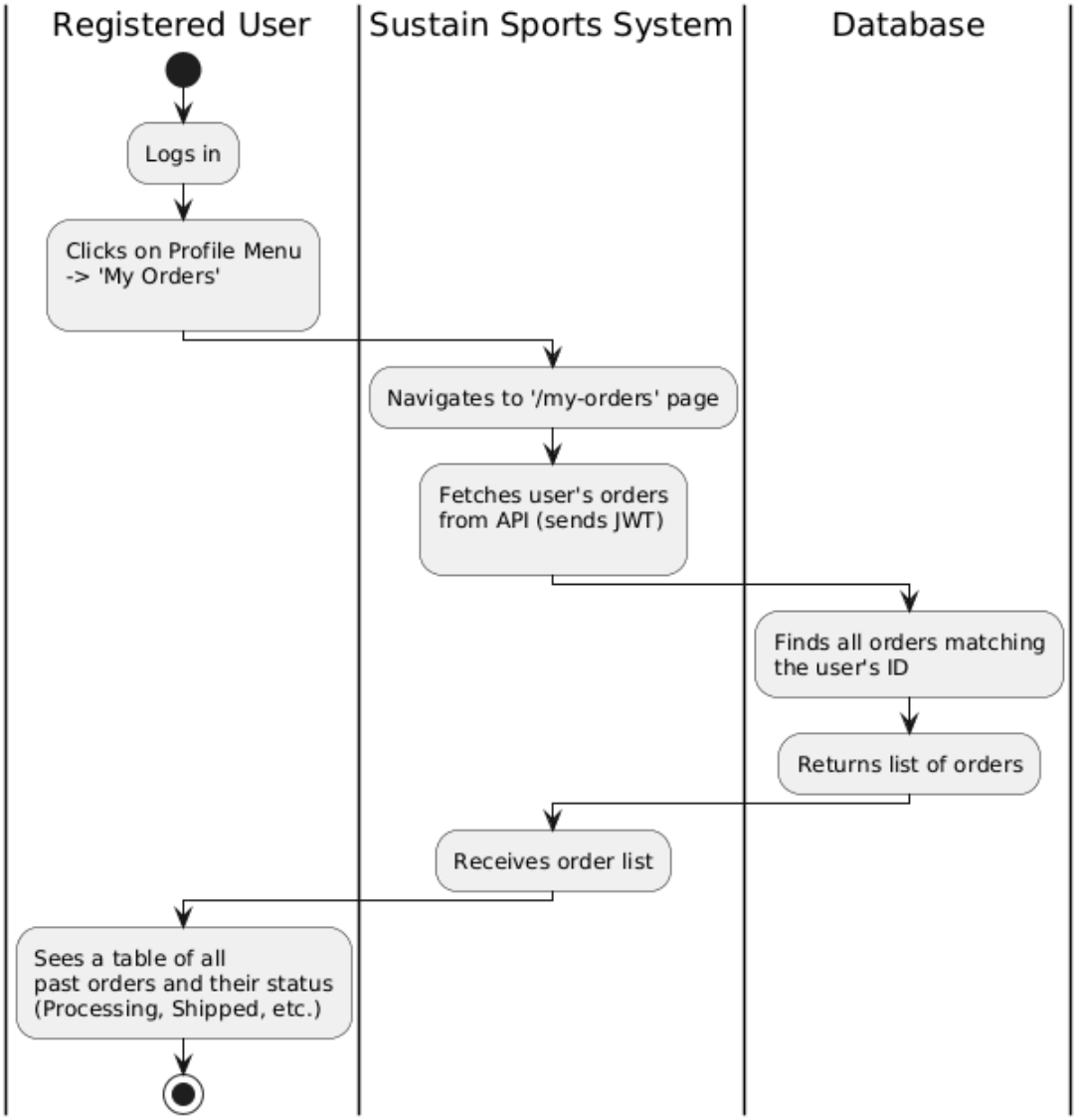


Figure 2.2.8: Activity Diagram View Order History

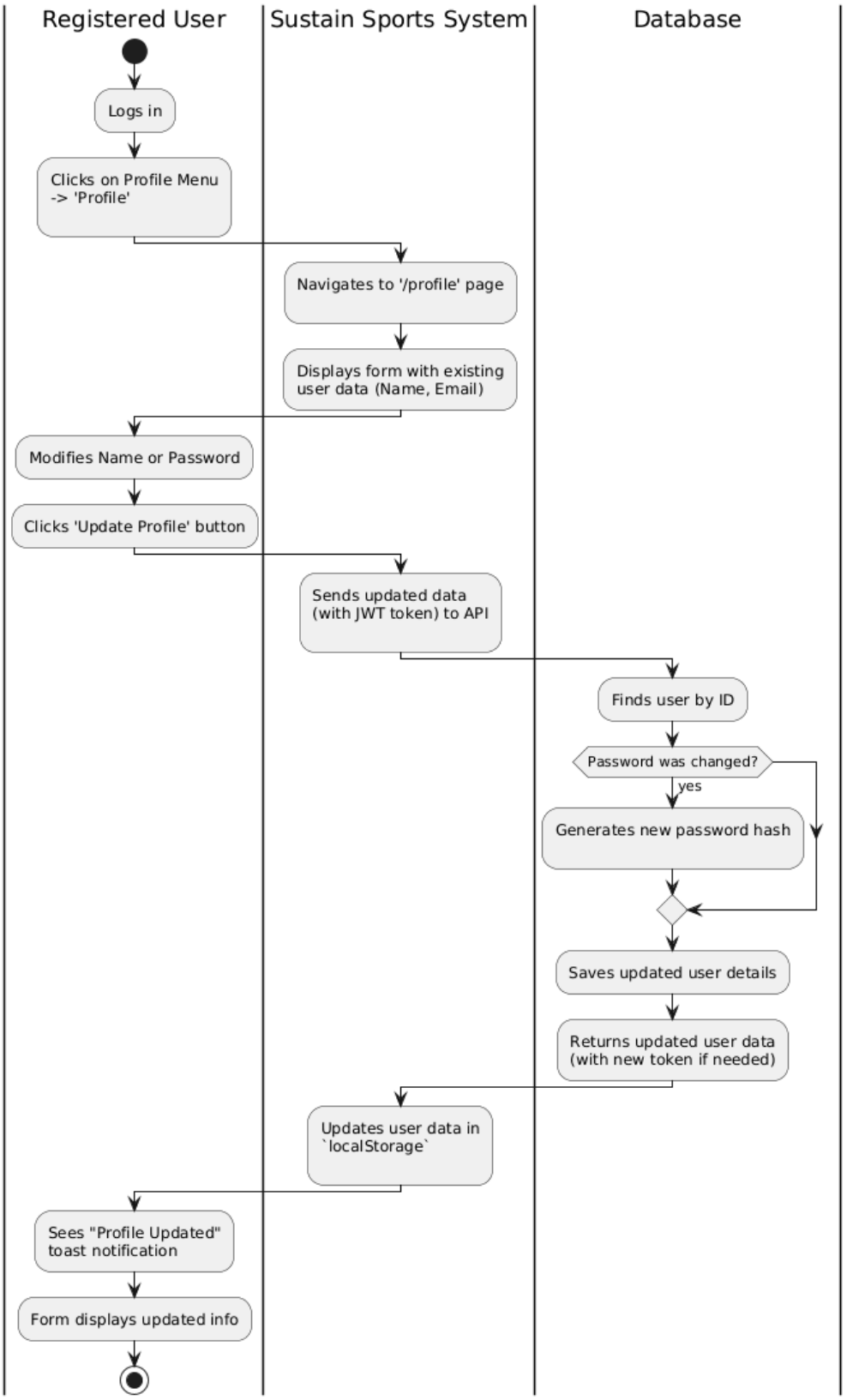


Figure 2.2.9: Activity Diagram Update User Profile

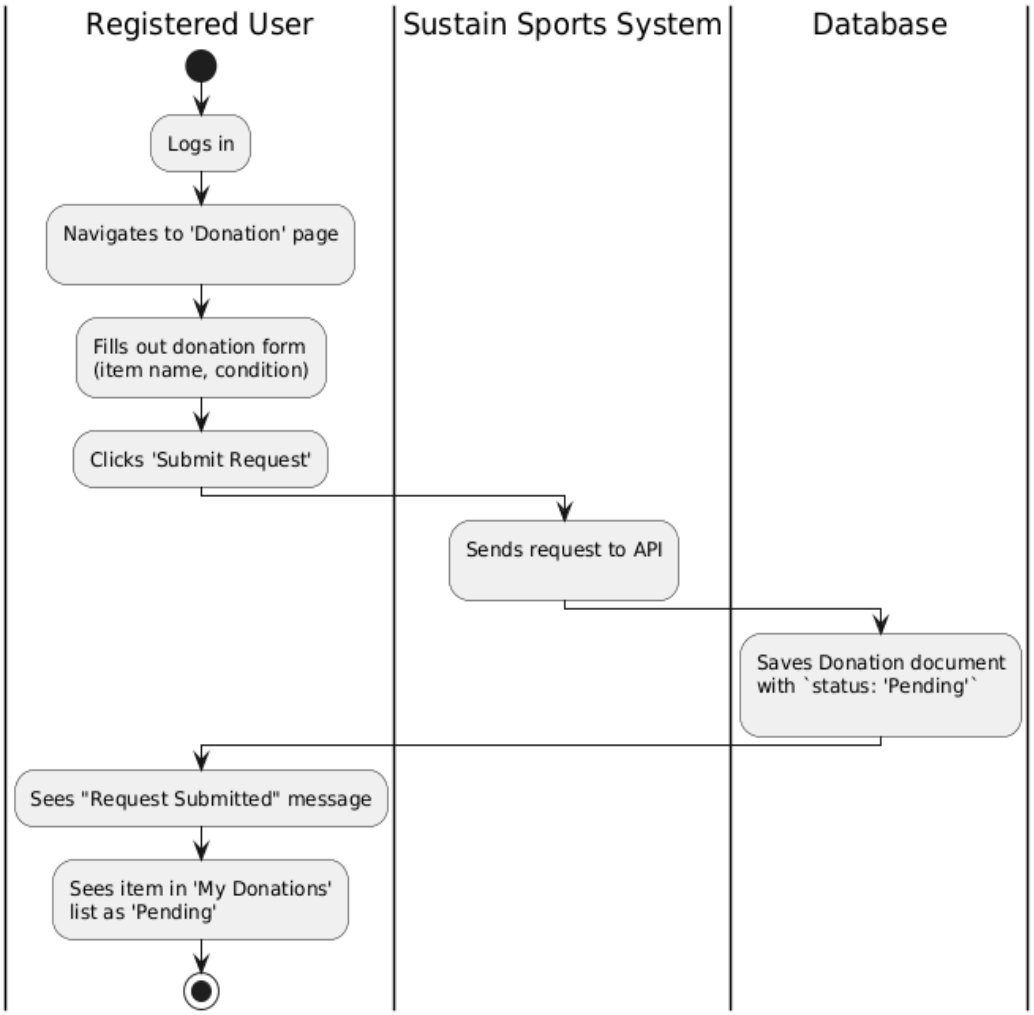


Figure 2.2.10:Activity Diagram Submit Donation Request

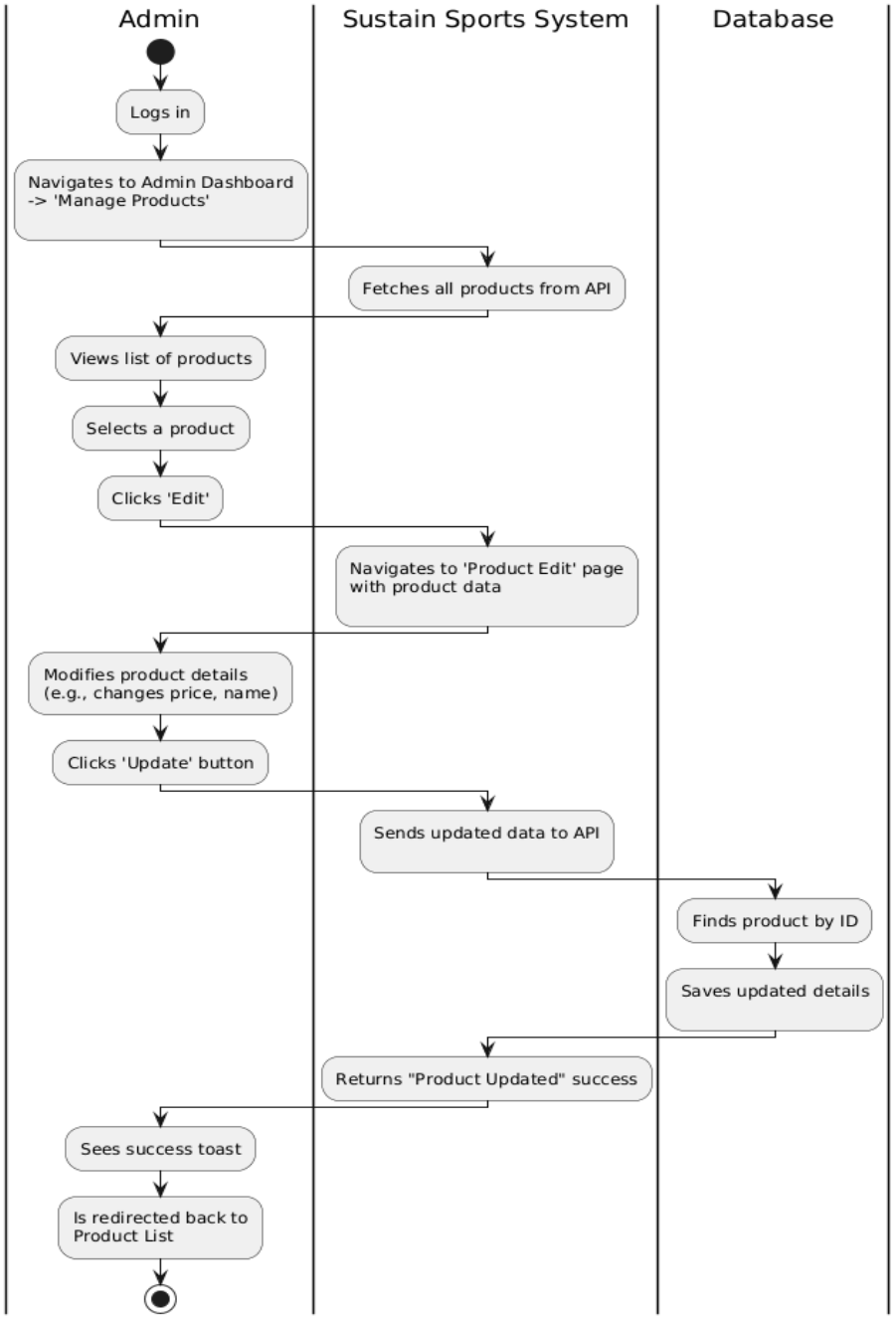


Figure 2.2.11:Activity Diagram Admin Update Product

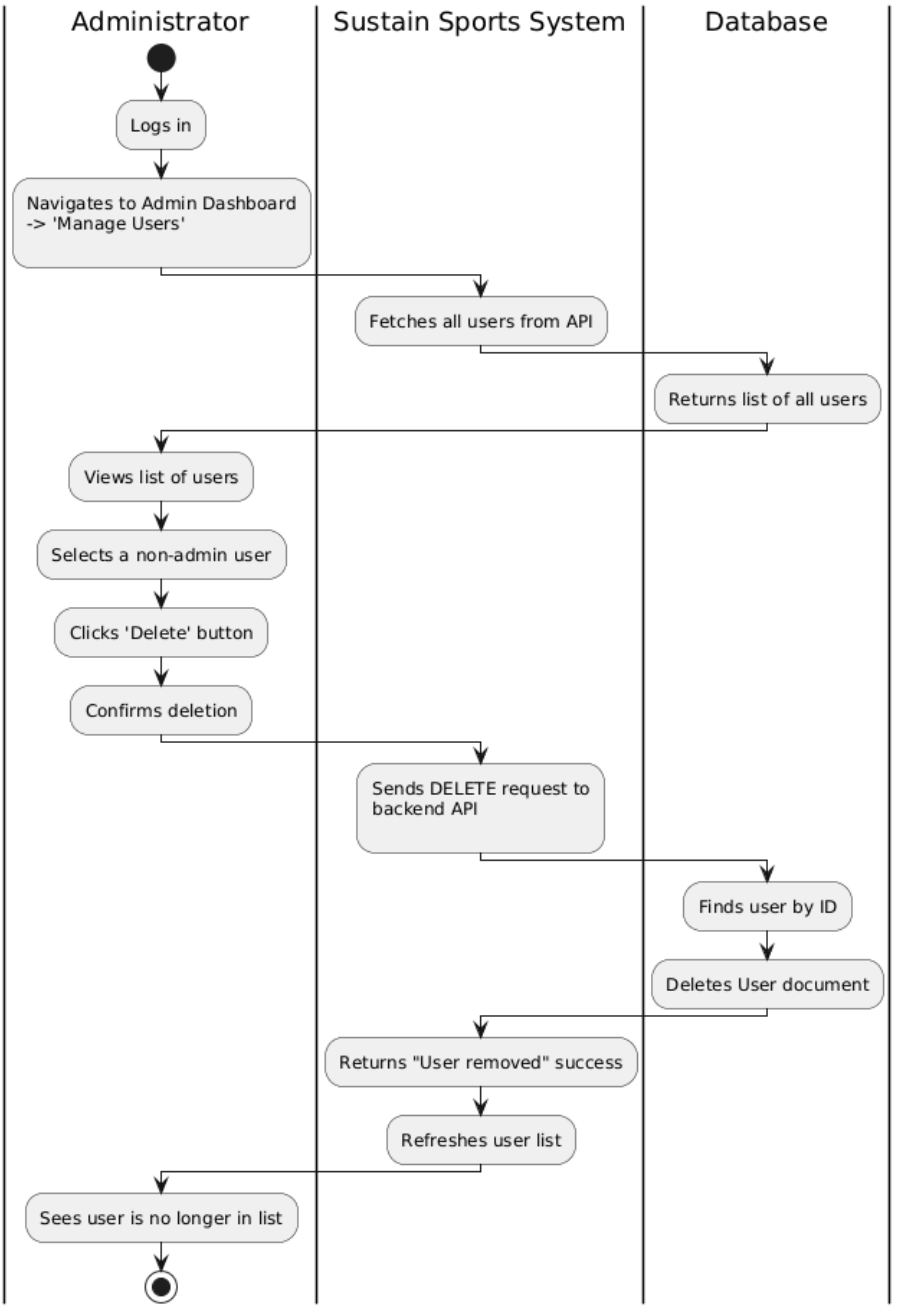


Figure 2.2.12: Activity Diagram Admin Delete Users

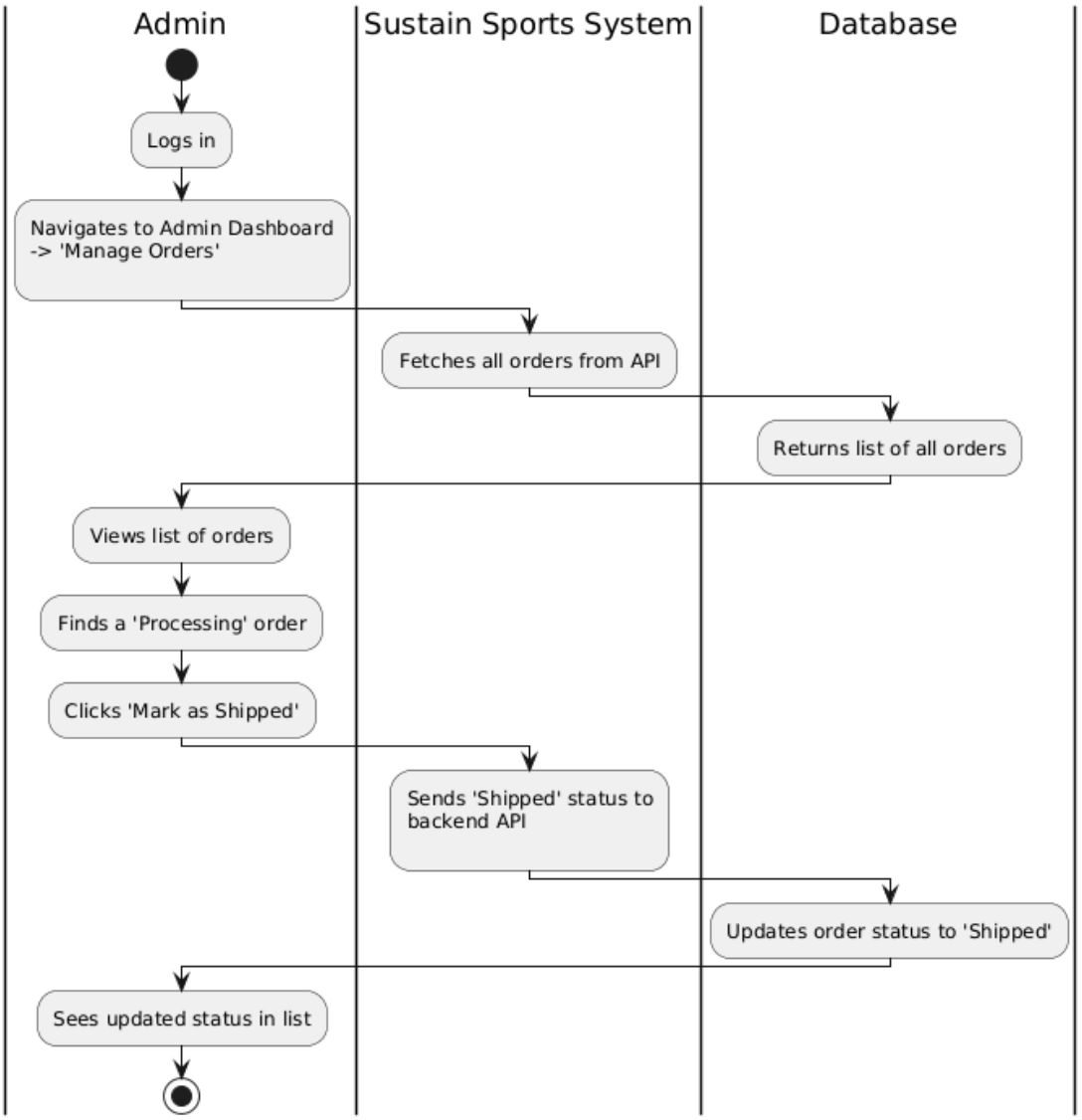


Figure 2.2.13: Activity Diagram Admin Update Orders

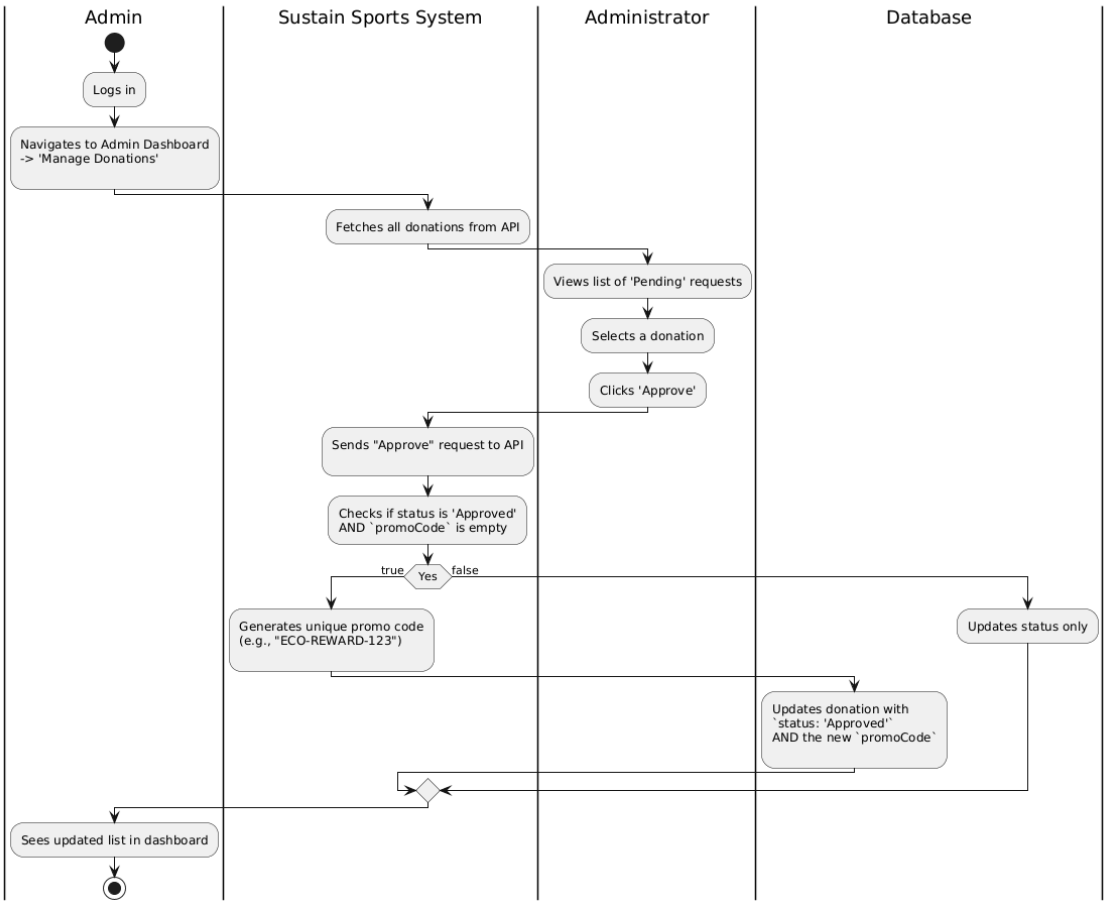


Figure 2.2.14: Activity Diagram Admin approve donation & Generate Rewards

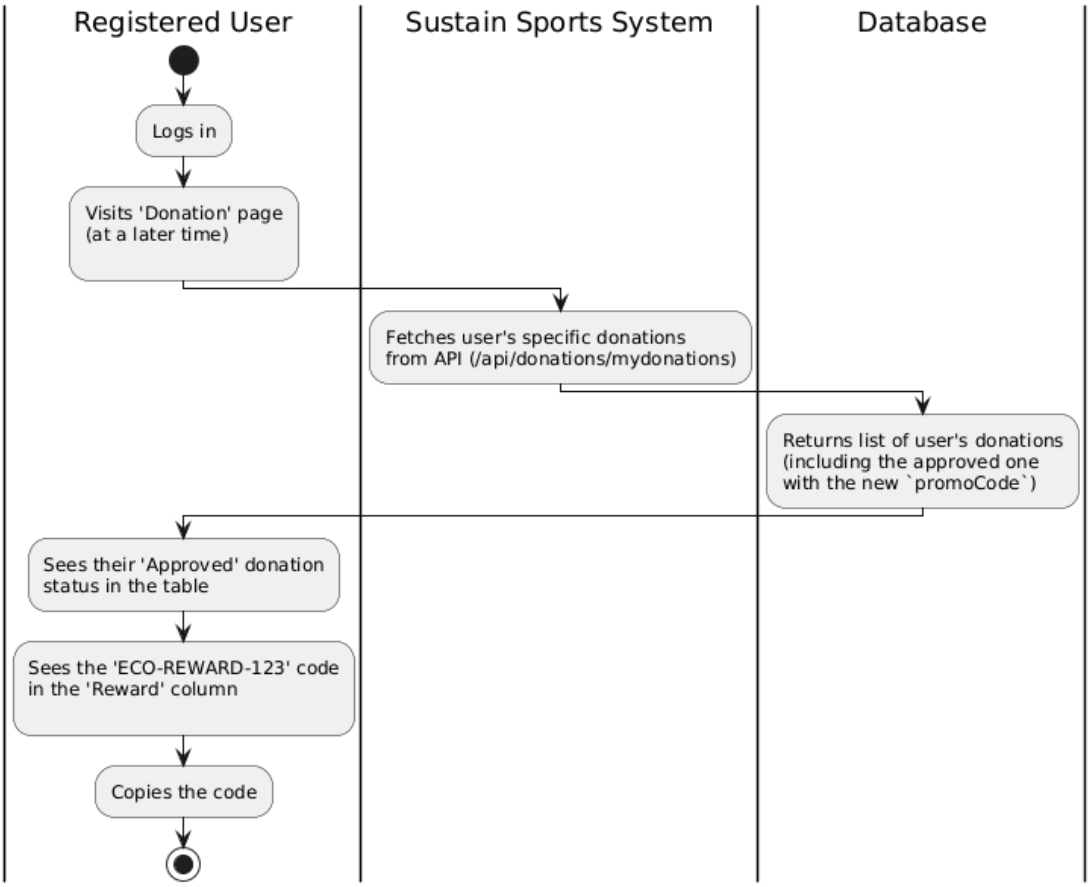


Figure 2.2.15: Activity Diagram View Donation Reward

2.4.4 Sequence Diagram

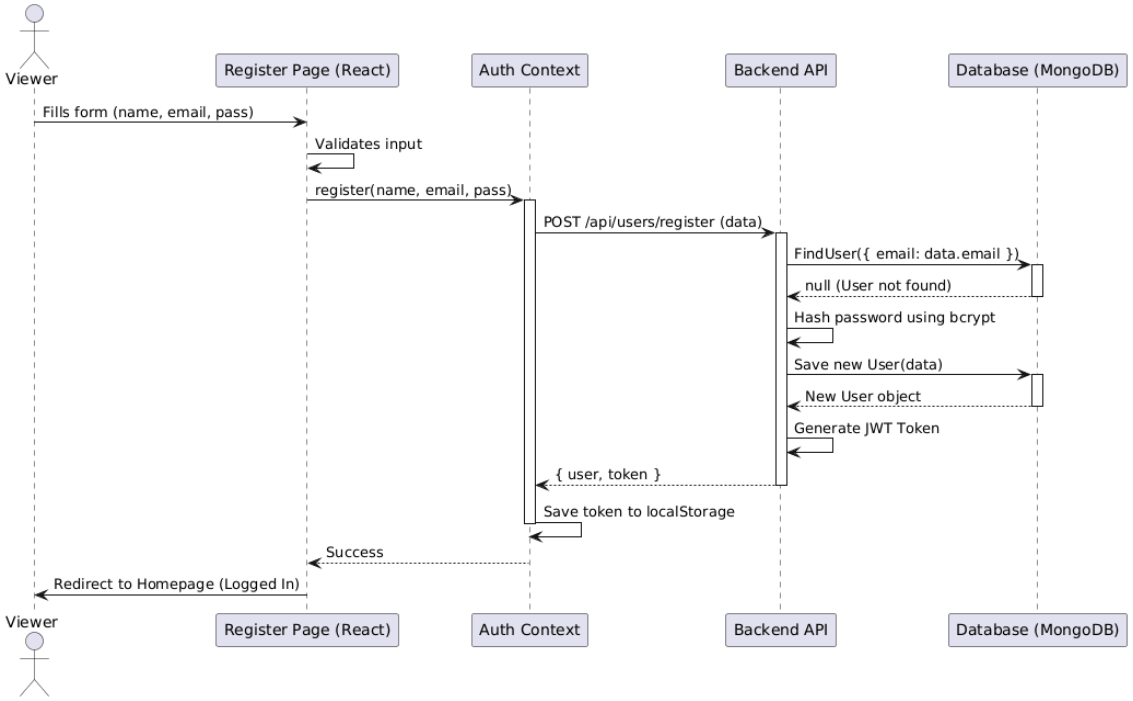


Figure 2.3.1: Sequence Diagram Sign UP

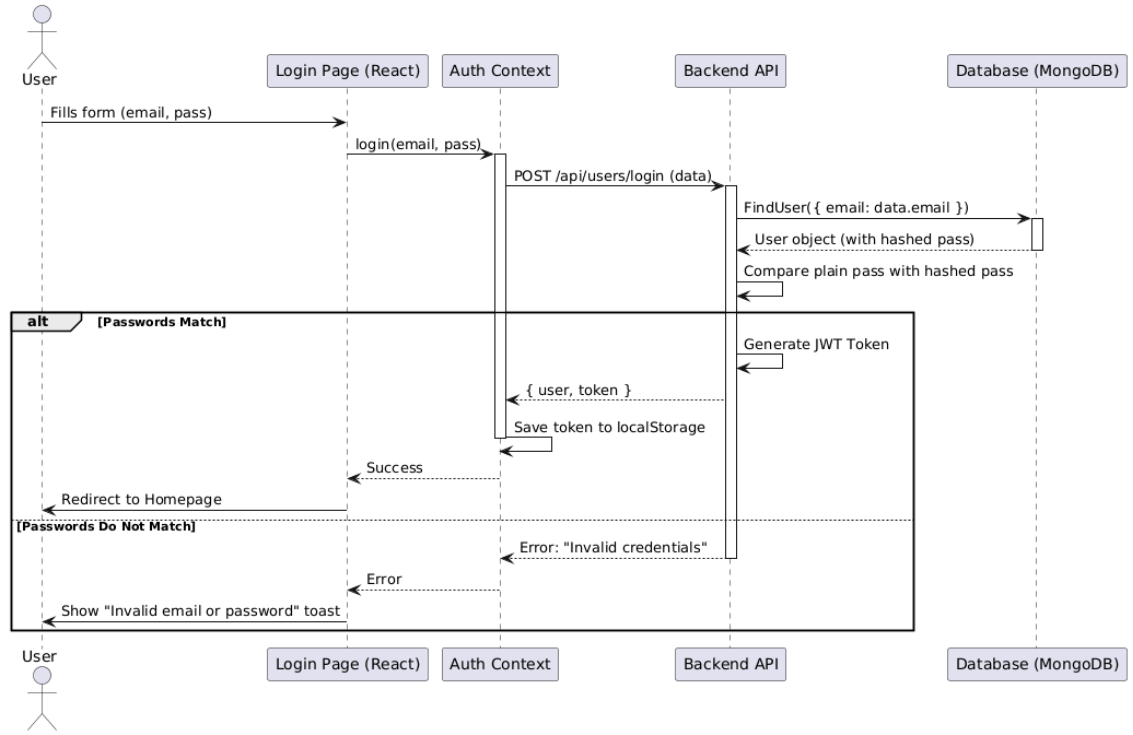


Figure 2.3.2: Sequence Diagram Sign In

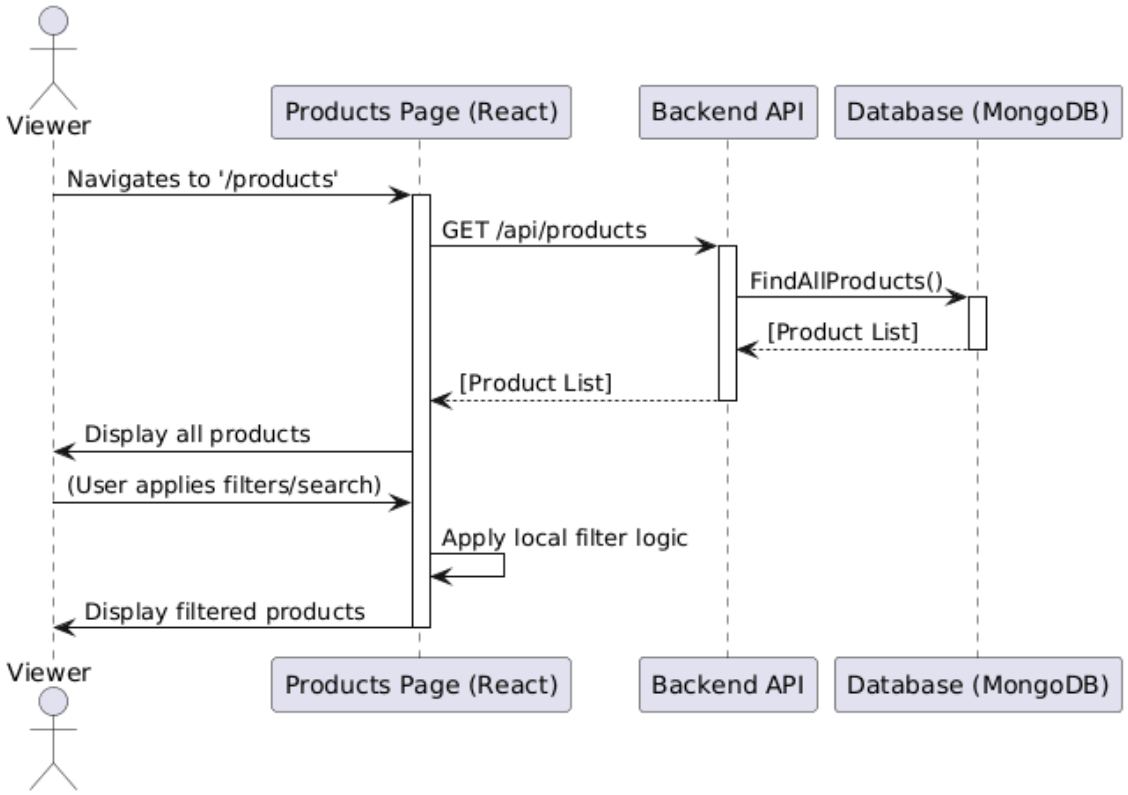


Figure 2.3.3:Sequence Diagram Browse Product & Filtering

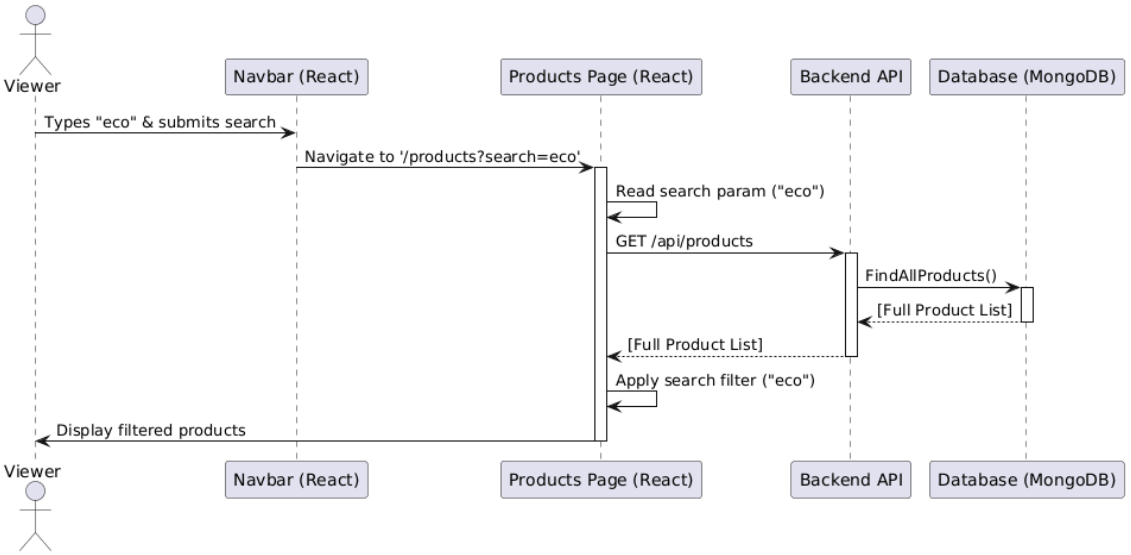


Figure 2.3.4: Sequence Diagram Search Product

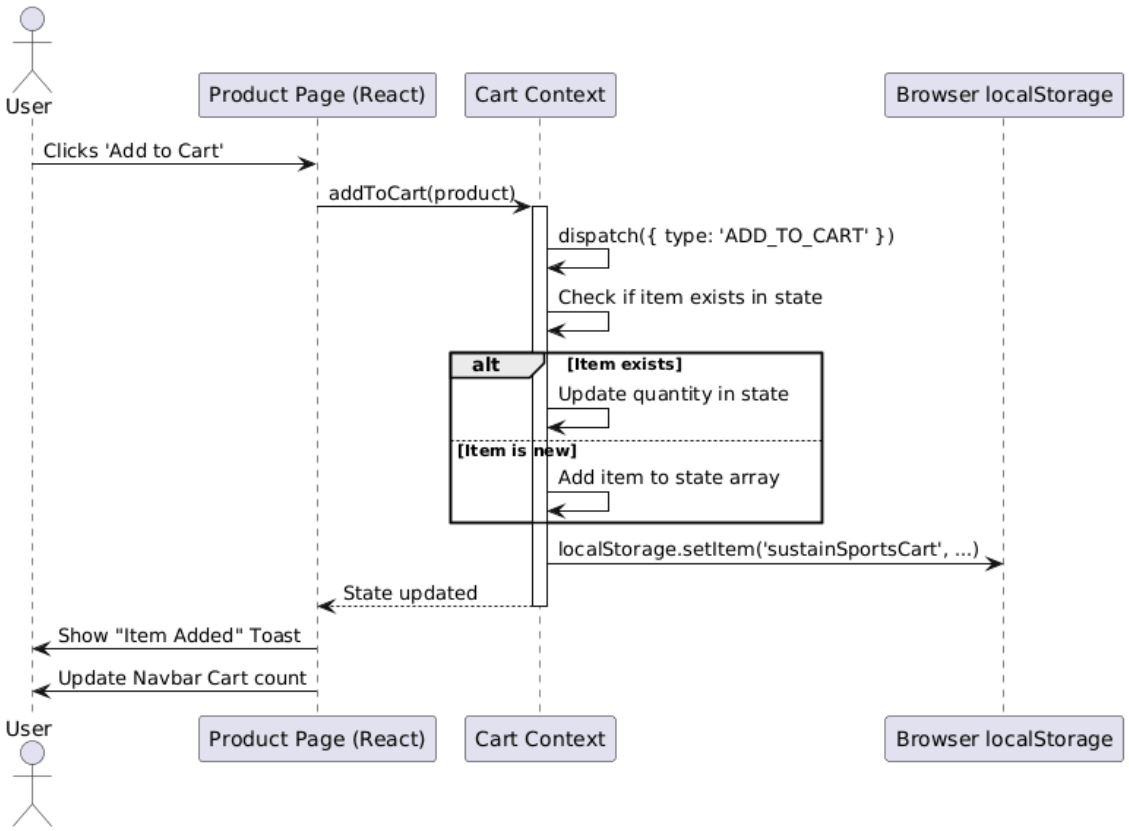


Figure 2.3.5: Sequence Diagram Add to Cart

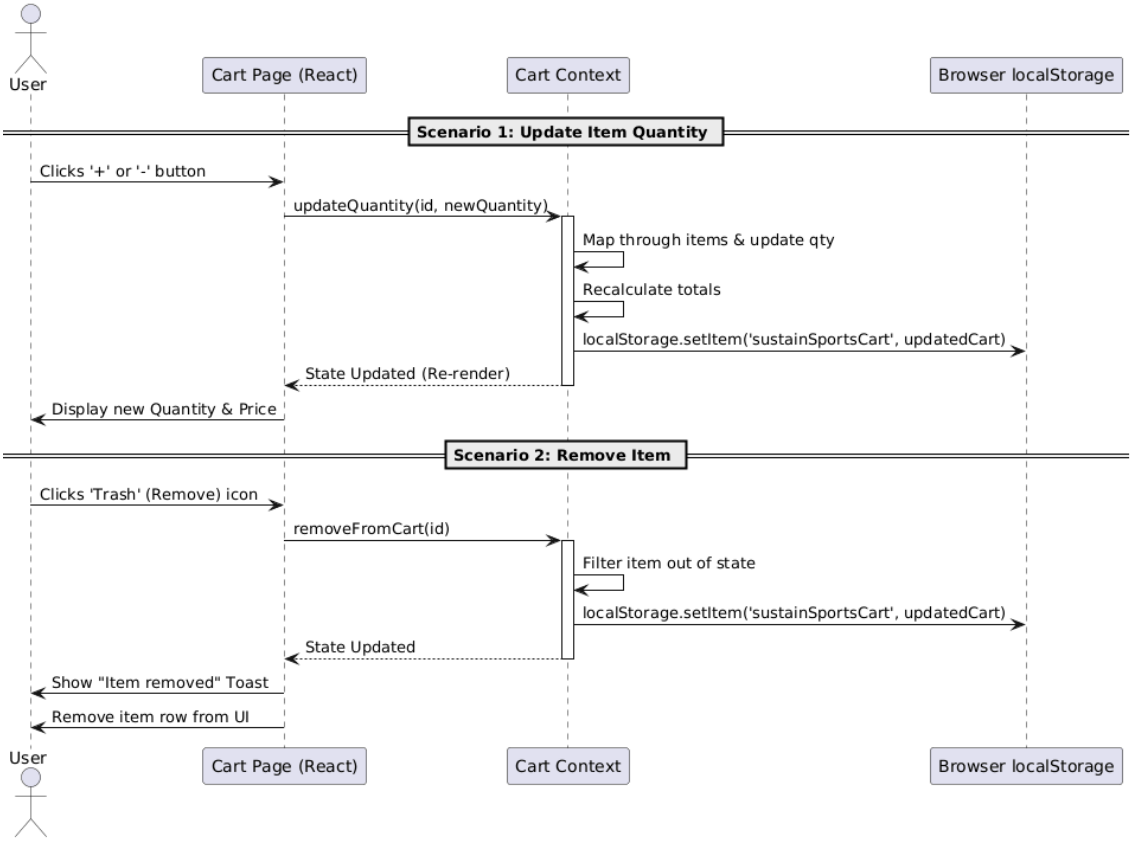


Figure 2.3.6:Sequence Diagram Manage Cart

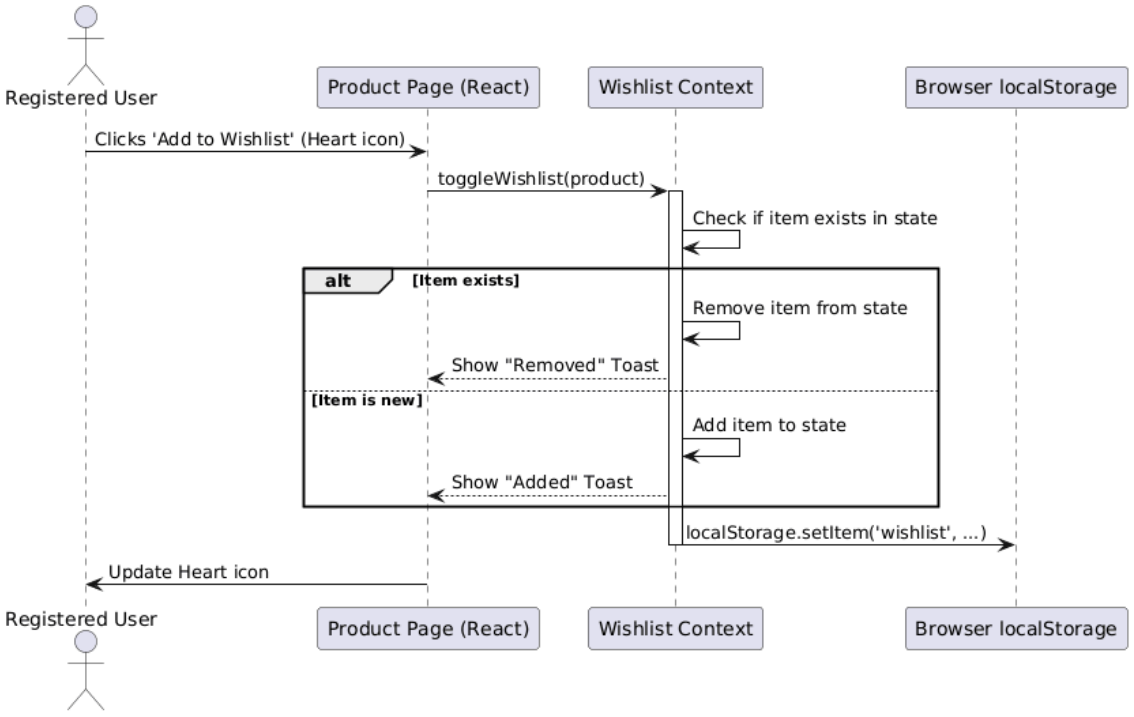


Figure 2.3.7:Sequence Diagram Manage Wishlist

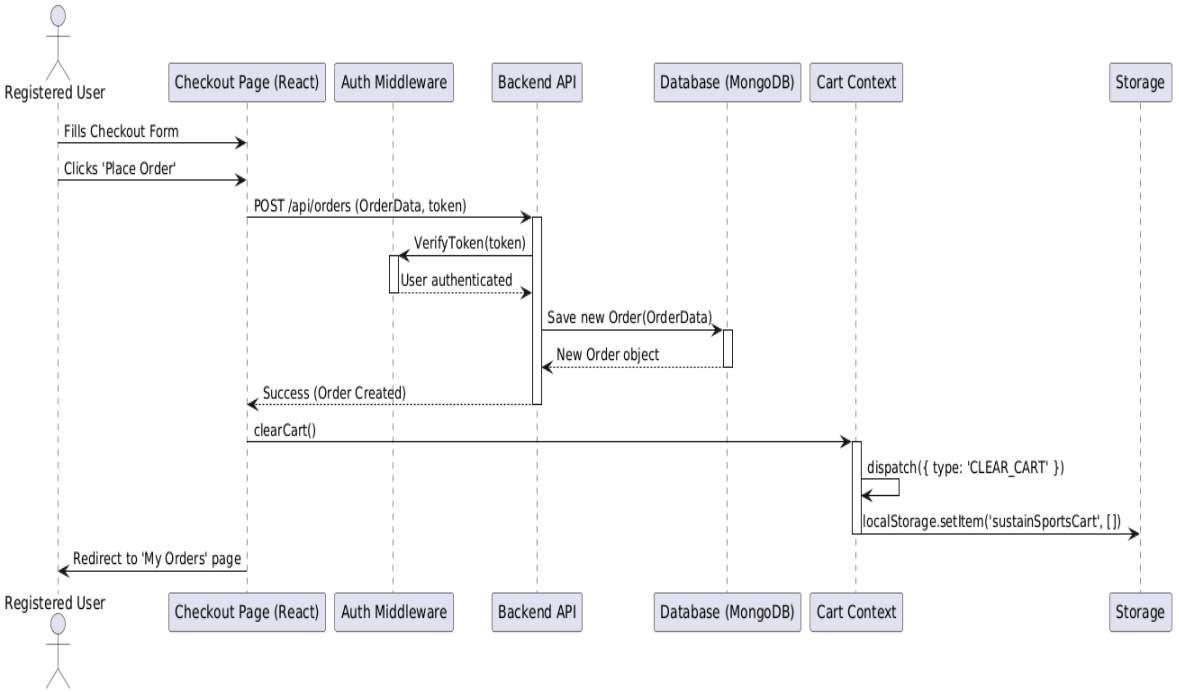


Figure 2.3.8:Sequence Diagram Complete Checkout

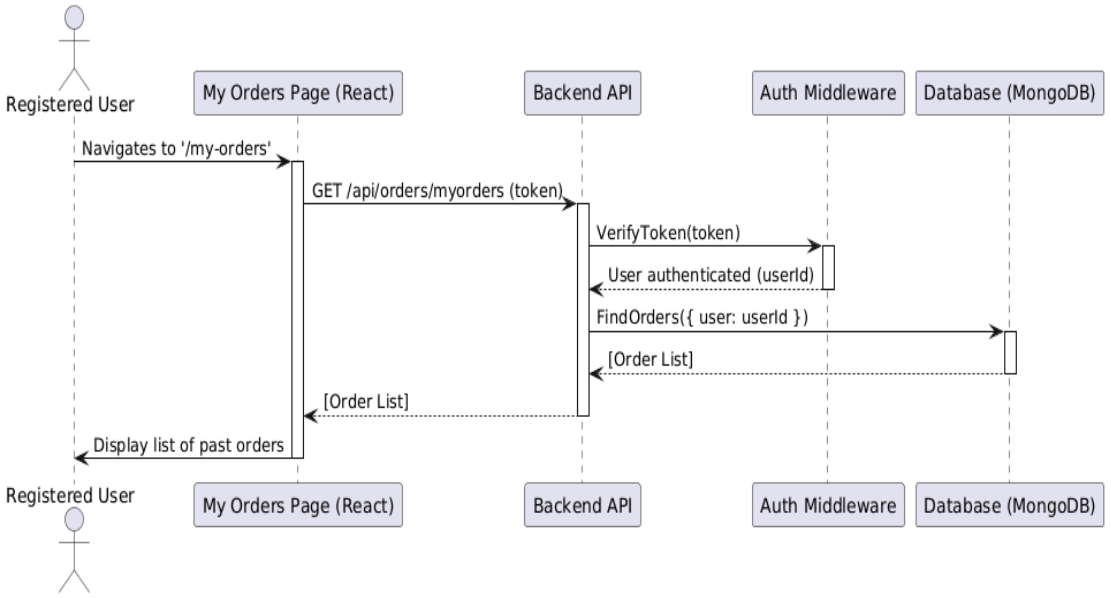


Figure 2.3.9:Sequence Diagram View Order History

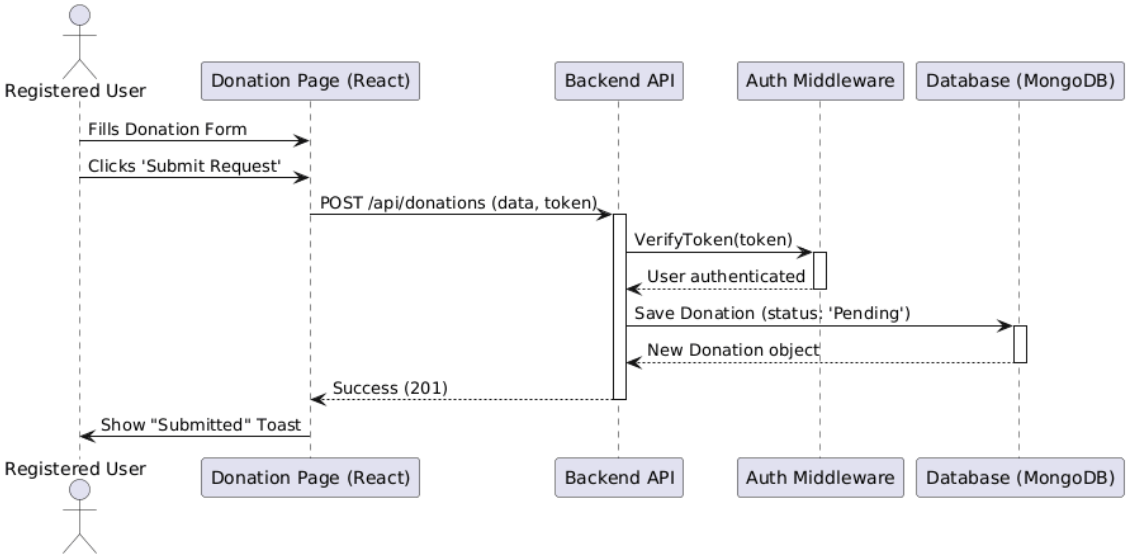


Figure 2.3.10: Sequence Diagram Submit Donation Request

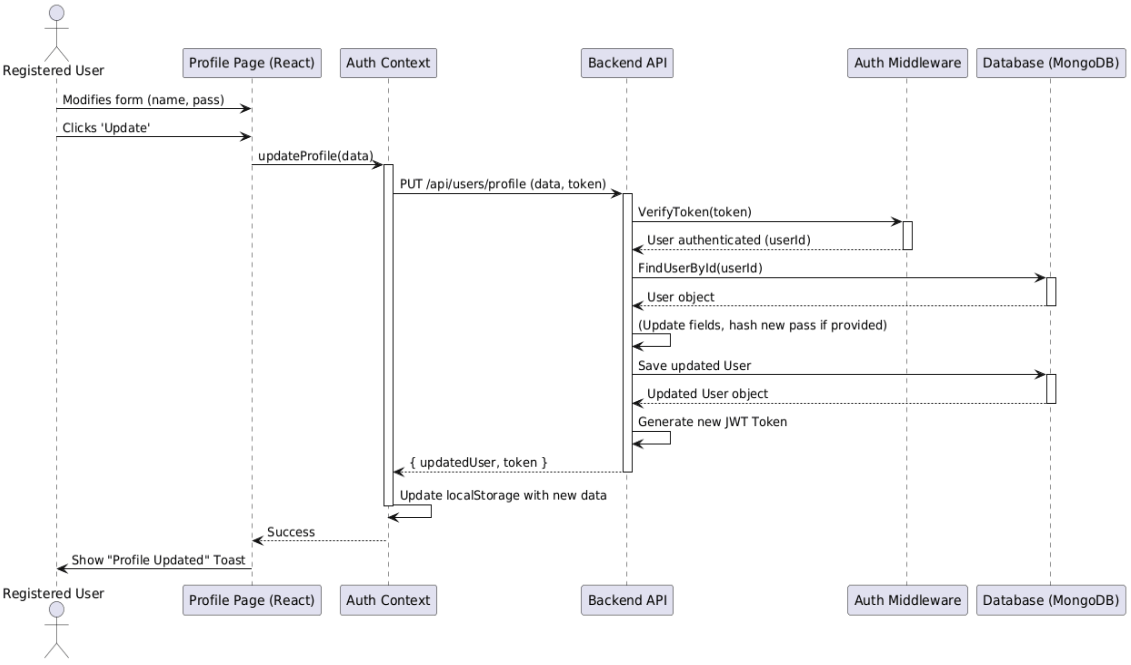


Figure 2.3.11: Sequence Diagram Update User Profile

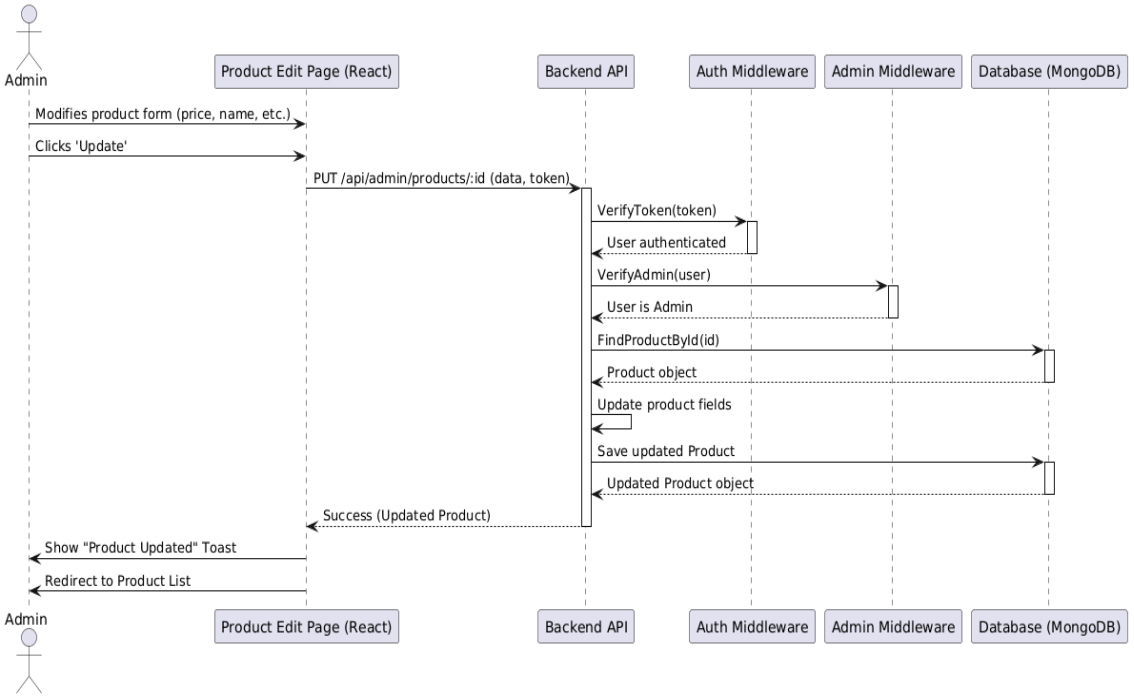


Figure 2.3.12: Sequence Diagram Admin Update Product

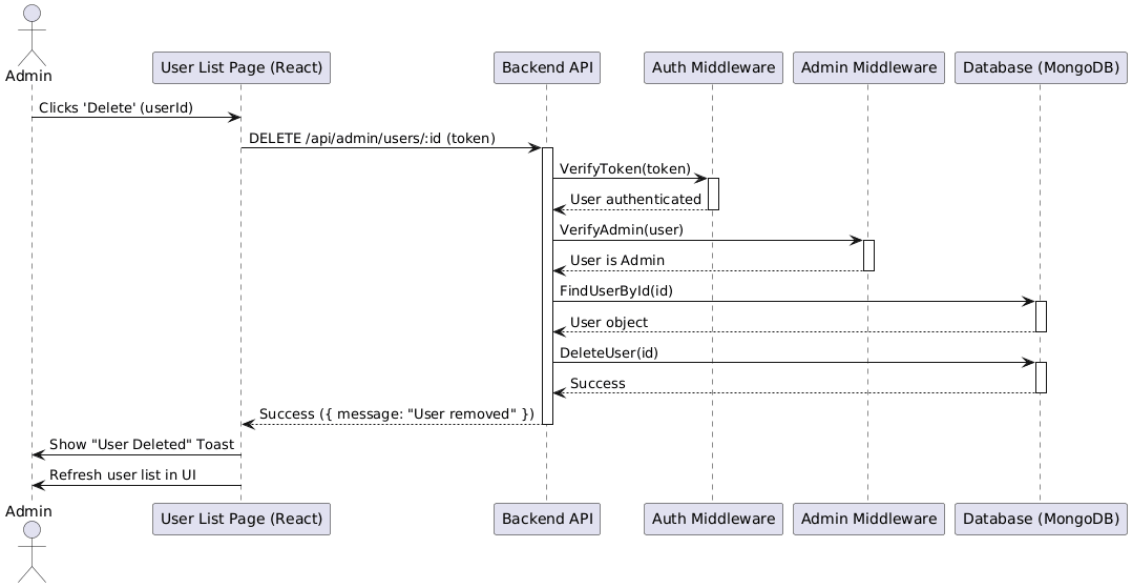


Figure 2.3.13: Sequence Diagram Admin Delete Users

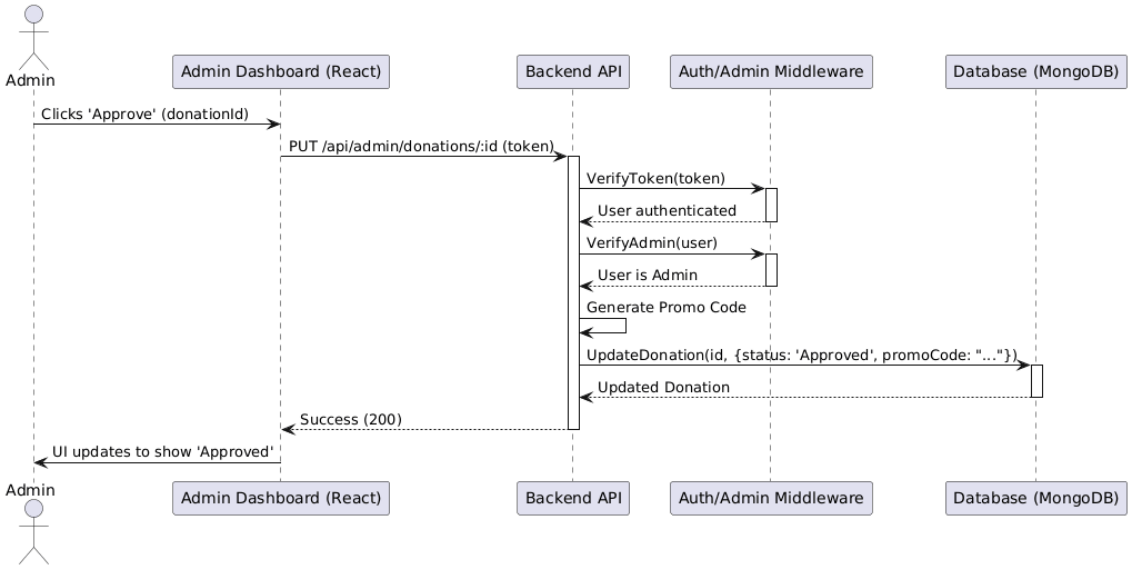


Figure 2.3.14: Sequence Diagram Admin Approve Donation & Generate Rewards

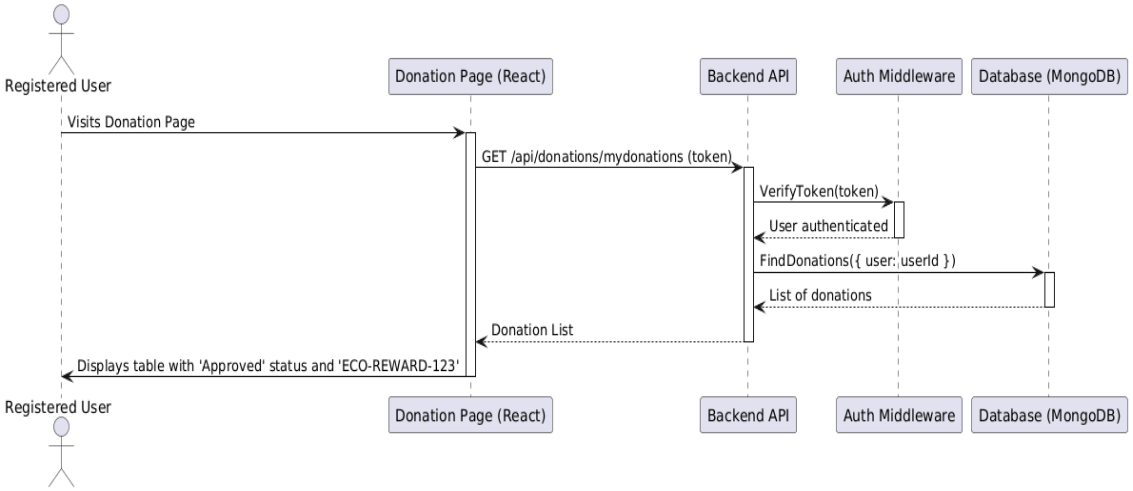


Figure 2.3.15: Sequence Diagram View Donation Reward

2.4.5 Class Diagram

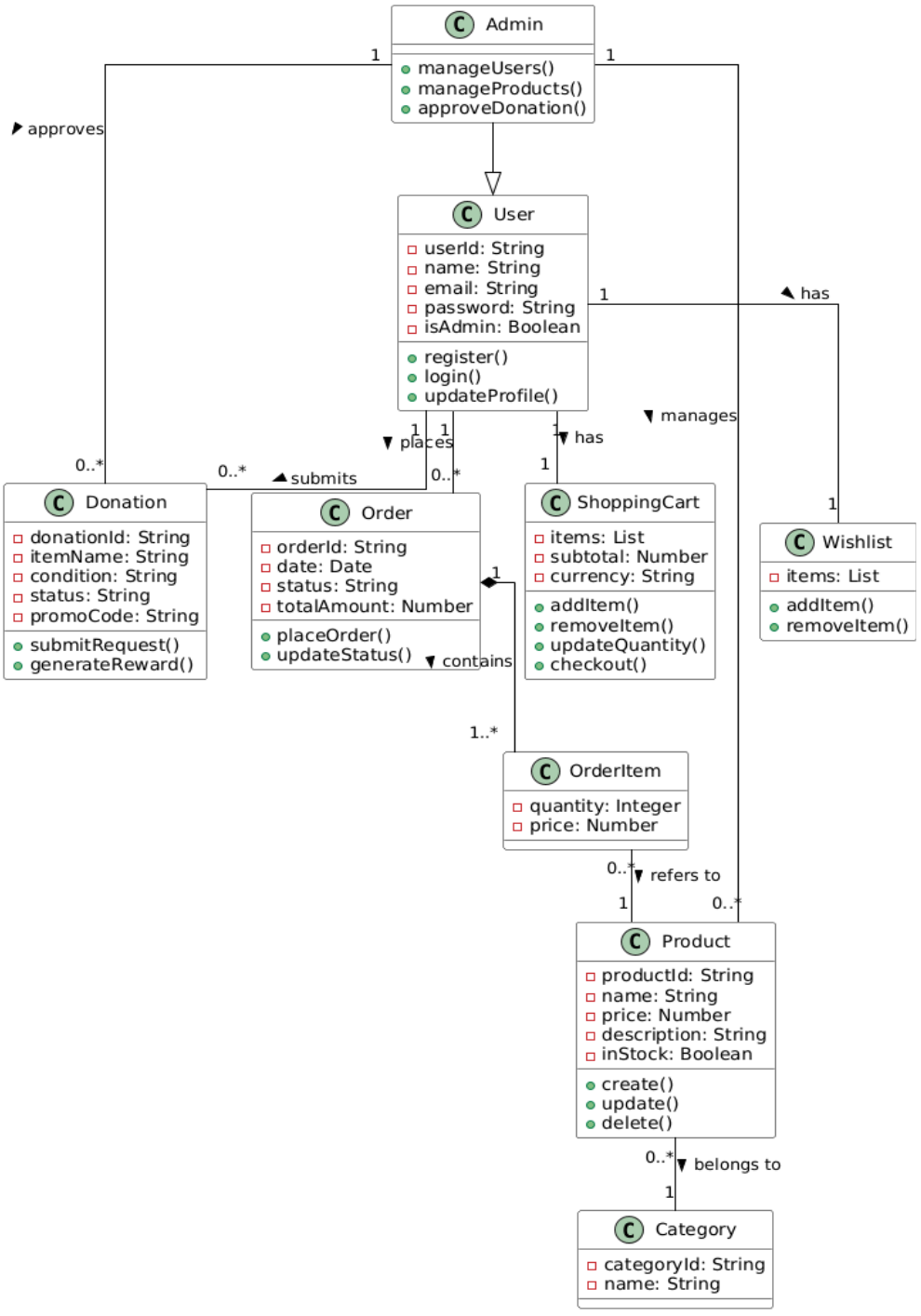


Figure 2.4: Class Diagram

2.4.6 ER Diagram

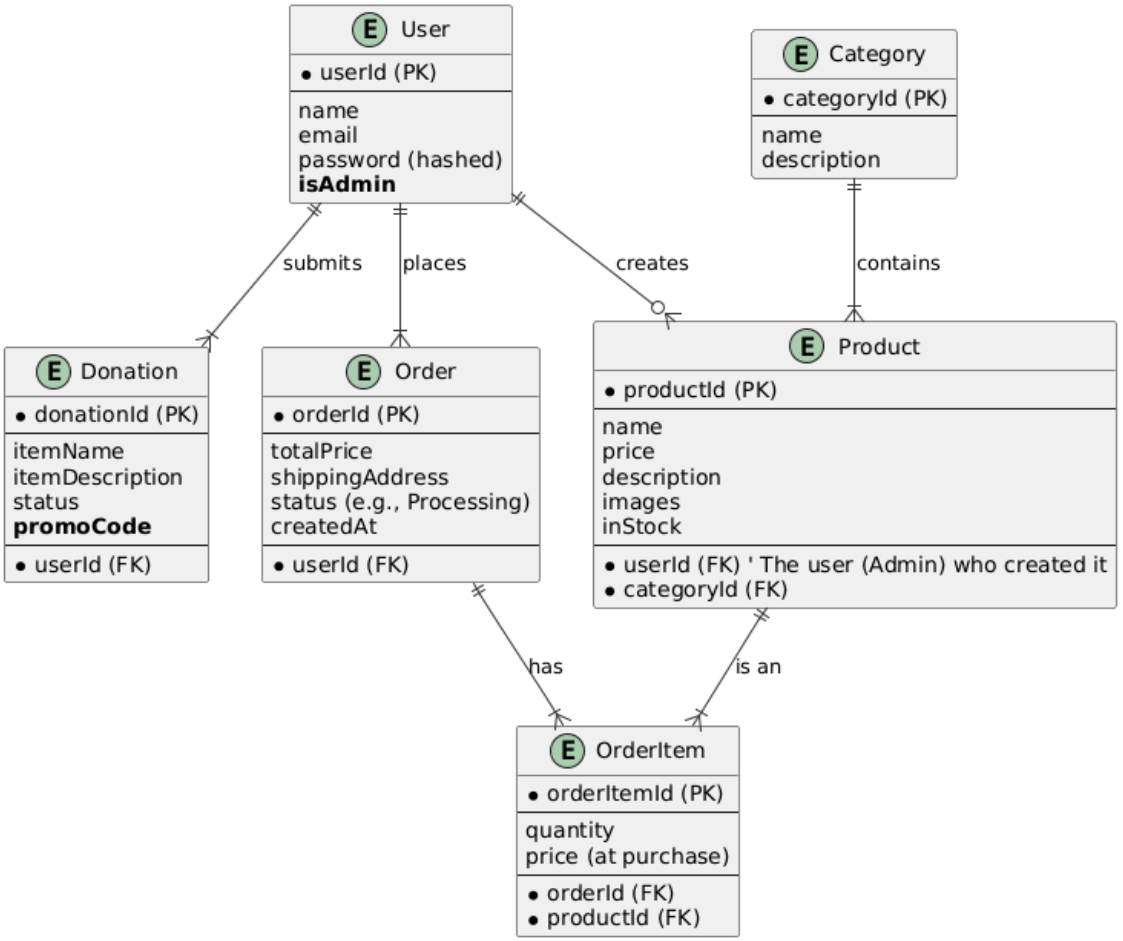


Figure 2.5: ER Diagram

2.5 Coding: Appendix A

All the parts (React) of the "Sustain Sports" project along with the backend (Node.js/Express), and the database setup code are available on GitHub.

Where the code lives:

URL: https://github.com/AhnafSaad/Sustain-Sports-Permanent.git

2.6 Summary

The structure "Sustain Sports" was detailed in Chapter 2. It took the rubber meets the road scope of work and turned it into nitty-gritty technical details. It outlined the main Functional Requirements and mentioned important features such as the secure check-out and their gamified concept; 'Donation & Reward'. Non-functional Requirements were described as well for the performance and security of usability.

To see both how the system should operate and be structured, I use classic UML Diagrams:

- The Use Case Diagram helped us understand the actors (Three core actors: Viewer, Registered User, Admin) and their roles/permissions.
- Activity Diagrams illustrated the various workflows involved in pivotal processes like User Registration and the Donation-Reward cycle.
- These processes were further detailed in Sequence Diagrams as to the specific interactions between the Frontend, Backend API, and Database.
- Lastly, the data architecture was designed with ER and Class Diagrams. They could specify how Users, Products, Orders and Donations related to each other in the database.

This analysis is the guide for both system design and realization (Refer to next chapters).

Chapter 3 Software Testing

3.1 Introduction

This section describes the software testing and test mechanisms of the 'Sustain Sports'. The primary objective was also to verify that the system fulfills the functionality requirements described in preceding chapters. The goal of the testing phase was to verify that the modules User Authentication, Admin Management and the 'special' Donation System function as intended and are secure. This section details the particularities of features tested, approaches followed and it shows the results from testing system.

3.2 Testing Features

3.2.1 Feature to Be Tested

A. User Registration
B. User Login
C. Product Browsing & Search
D. Product Checkout
E. Admin Login
F. Manage Products (Admin)
G. Submit Donation (User)
H. Approve Donation (Admin)
I. User/Admin Logout
J. Manage Orders

3.3 Testing Strategies

3.3.1 Test Approach

Unit Testing: This was more of a bottom up approach to testing the individual little bits and pieces. On the backend, this was testing Mongoose code such as matchPassword in userModel. js. On the front end, this meant verifying React Context logic like whether the CartContext calculated totals and currency formatted properly.

Integration Testing: This specifically focused on testing the way different modules communicated with each other, primarily between my React frontend and Express backend. Each API endpoint was manually tested using the Postman tool. This validated the correct behaviour of data responses, error codes and authentication middleware.

System Testing (Black Box) : This was the predominant method of testing, treating the application as a “black box”. End-to-end user scenarios provided coverage of the entire system working properly from a user’s perspective without caring about implementation details.

.

3.3.2 Pass/Fail Criteria

PASS: The actual result of the test matches the expected result perfectly. The feature works as defined in the functional requirements, and all UI elements show correctly. Expected error messages appear as intended.

FAIL: The actual result does not match the expected result. This includes:

- The application gives an incorrect calculation, shows the wrong data, or fails to save data.
- The application crashes or shows an unhandled exception.
- A security requirement is violated.
- The UI is broken, unusable, or does not display correctly on the target browser.

Any test case that results in a "Fail" was logged, fixed by the developer, and then re-tested (Regression Testing) until it produced a "Pass" result.

3.4 System Testing (Test Cases with Report)

Table 3.1 : Test Case of User Sign Up

Test Case: 3.1.1			**Test Case Name:** User Registration			
System: Sustain Sports			**Subsystem:** User Authentication			
Designed by: Admin			**Design Date:** 18/11/2025			
Executed by: User			**Execution Date:** 19/11/2025			
			Description: The user registers for the Sustain Sports system by providing valid registration information.			
			Pre-condition: The user accesses the registration page			
Step	**Name**	**Email**	**Password**	**Response**	**Pass / Fail**	**Comment**
1	Test User	valid@test.com	Saad1234	Registration Successful .	Pass	Successful registration with valid input.
2	User 2	valid@test.com	12345	Registration Failed	Fail	Invalid Password format detected.
3	(Empty)	(Empty)	saad12	Registration Failed.	Fail	System correctly prevents empty name.
Post-condition: The user is successfully registered with valid information, and the process is complete.						

Table 3.2 : Test Case of User Sign In

Test Case: 3.1.1			**Test Case Name:** User Login			
System: Sustain Sports			**Subsystem:** User Authentication			
Designed by: Admin			**Design Date:** 18/11/2025			
Executed by: User			**Execution Date:** 19/11/2025			
			Description: The user logs in to the Sustain Sports system by providing valid credentials.			
			Pre-condition: The user accesses the login page			
Step	**Name**	**Email**	**Password**	**Response**	**Pass / Fail**	**Comment**
1	Test User	valid@test.com	Saad1234	Login Successful. .	Pass	Valid credentials accepted.
2	User 2	wrong@test.com	12345	Login Failed.	Fail	System rejects incorrect password.
3	(Empty)	(Empty)	saad12	Login Failed.	Fail	System rejects unregistered email.
Post-condition: The user is verified and the token is saved for the cases that pass.						

Table 3.3 : Test Case of Product Browsing & Search

Test Case: 3.4.3	**Test Case Name**: Product Browsing
System: Sustain Sports	**Subsystem: Product Catalog**
Designed by: QA Engineer	**Design Date:** 25/12/2024
Executed by: Viewer	**Execution Date** 24/12/2024
	Description: Verify that a user can search for items, browse the catalog, filter results, and view product details..
	Pre-condition: The user is on the website Home page.

Step	**Action**	**Response**	**Pass / Fail**	**Comment**
1	Search Product	Search results displayed with matching items.	**Pass**	Search query logic functions correctly.
2	View Product	Product details displayed.	Pass	Navigation to product detail page successful.
3	Filter Products	Filter results displayed based on category.	Pass	Products are filtered correctly by category.
4	Add Product to Cart	Product successfully added to cart.	Pass	Cart count updates immediately.
Post-condition: The user can browse products, view details, and add items to the cart				

Table 3.4 : Test Case of Product Checkout

Test Case: 3.4.3			**Test Case Name**: Product Checkout	
System: Sustain Sports			**Subsystem:** E-Commerce	
Designed by: QA Engineer			**Design Date:** 18/11/2025	
Executed by: Registered User			**Execution Date**: 19/11/2025	
			Description: Verify that a logged-in user can successfully complete the checkout process and place an order.	
			Pre-condition: The user is logged in and has at least one item in the shopping cart.	
Step	**Action**	**Response**	**Pass / Fail**	**Comment**
1	Initiate Checkout	Click "Proceed to Checkout" from Cart page.	Pass	User is redirected to the `checkout` page with the order summary.
2	Enter Shipping Details	Fill in Address, City, and Postal Code.	Pass	Input fields accept valid data.
3	Submit Empty Form	Click "Place Order" without filling fields.	Fail	Browser/System displays "Please fill in this field" error.
4	Place Order	Click "Place Order" with valid data.	Pass	"Order Placed" toast appears, cart is cleared, and user is redirected to "My Orders".
Post-condition: A new order is created in the database with "Processing" status, and the shopping cart is empty.				

Table 3.5 : Test Case of Admin Login

Test Case: 3.1.1			**Test Case Name:** Admin Login			
System: Sustain Sports			**Subsystem:** User Authentication			
Designed by: Admin			**Design Date:** 18/11/2025			
Executed by: Admin			**Execution Date:** 19/11/2025			
			Description: Verify that an administrator can log in and gain access to the dashboard.			
			Pre-condition: The user has an admin account and is on the Login page			
Step	**Name**	**Email**	**Password**	**Response**	**Pass / Fail**	**Comment**
1	Test User	valid@test.com	Saad1234	Login Successful. .	Pass	Valid credentials accepted.
2	User 2	wrong@test.com	12345	Login Failed.	Fail	System rejects incorrect password.
3	(Empty)	(Empty)	saad12	Login Failed.	Fail	System rejects unregistered email.
Post-condition: The user is verified and the token is saved for the cases that pass.						

Table 3.6 : Test Case of Submit Donation

Test Case: 3.1.1			**Test Case Name:** Submit Donation			
System: Sustain Sports			**Subsystem:** Donation & Reward			
Designed by: User			**Design Date:** 18/11/2025			
Executed by: User			**Execution Date:** 19/11/2025			
			Description: Verify that a registered user can submit a donation request for recycling.			
			Pre-condition: The user is logged in and on the "Recycling" page			
Step	**Item Name**	**Condition**	**Description**	**Response**	**Pass / Fail**	**Comment**
1	Unused Gym Bag	New	Brand new, tags on.	Submission Successful. .	Pass	System accepts "New" condition correctly.
2	Tennis Racket	(Not Selected)	Broken strings.	Submissi on Failed.	Fail	System validation prevents submission without a condition.
3	Old Cricket Bat	Fair	Just old.	Submissi on Successfu l	Pass	Valid data submitted successfully.
Post-condition: For the passed case, a new "Pending" donation appears in the history.						

Table 3.7 : Test Case of Manage Products (Admin)

Test Case: 3.1.1			**Test Case Name:** Manage Products		
System: Sustain Sports			**Subsystem:** Admin Panel		
Designed by: QA Engineer			**Design Date:** 18/11/2025		
Executed by: Admin			**Execution Date:** 19/11/2025		
			Description: Verify that an administrator can create, read, update, and delete products in the store..		
			Pre-condition: User is logged in as Admin and is on the "Admin Dashboard".		
Step	**Action**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Create Product	Click "Create Product	Success.	Pass.	System creates placeholder correctly.
2	Update Details	Name: "Pro Mat", Price: "50"	Success.	Pass	Update functionality works.
3	Update Invalid	Name: (Empty)	Fail.	Fail	Backend validation prevents empty fields.
4	Delete Product	Click "Trash" icon	Success.	Pass	Delete function works.
Post-condition: The product catalog is updated reflecting the changes.					

Table 3.8 : Test Case of Manage Orders (Admin)

Test Case: 3.4.3	**Test Case Name**: Manage Orders (Admin)
System: Sustain Sports	**Subsystem:** Admin Panel
Designed by: QA Engineer	**Design Date:** 18/11/2025
Executed by: Registered User	**Execution Date**: 19/11/2025
	Description: Verify Admin can update order status.
	Pre-condition: A "Processing" order exists.

Step	**Action**	**Response**	**Pass / Fail**	**Comment**
1	Mark as Shipped	Status updates to "Shipped".	Pass	API call successful..
2	Mark as Delivered	Status updates to "Delivered".	Pass	Final status reached.
3	Verify User View	User sees "Delivered" in "My Orders"	Pass	Database update reflected for user..
4	Disconnect & Update	"Update failed". Status remains same.	Fail	System handles network failure.
Post-condition: The order status is permanently updated in the database.				

Table 3.9: Test Case of Approve Donation & Reward (Admin)

Test Case: 3.4.3	**Test Case Name**: Approve Donation & Reward
System: Sustain Sports	**Subsystem:** Admin Panel
Designed by: QA Engineer	**Design Date:** 18/11/2025
Executed by: Admin	**Execution Date**: 19/11/2025
	Description: Verify Admin approval triggers reward generation.
	Pre-condition: A "Pending" donation exists.

Step	**Action**	**Response**	**Pass / Fail**	**Comment**
1	Click "Approve"	.Status updates to "Approved".	Pass	API call successful; UI updates immediately.
2	Verify Reward Generation	Database shows a new promoCode for that item.	Pass	Backend logic correctly generates and saves the code.
3	Verify User View	(As User) "Donation History" shows the code.	Pass	User can see and copy the new reward code.
4	Fail Scenario	Click "Approve" (Network Error).	Fail	System handles network failure.
Post-condition: The donation is marked "Approved" and the user receives a promo code.				

Table 3.10:Test Case Of User/Admin Logout

Test Case: 3.4.3	**Test Case Name**: User/Admin Logout
System: Sustain Sports	**Subsystem:** Authentication
Designed by: QA Engineer	**Design Date:** 18/11/2025
Executed by: User/Admin	**Execution Date**: 19/11/2025
	Description: Verify that a logged-in user or admin can successfully log out of the system, clearing their session.
	Pre-condition: The user (User or Admin) is logged in and on any page.

Step	**Action**	**Response**	**Pass / Fail**	**Comment**
1	Click "Logout"	User is redirected to Home page.	Pass	Session termination triggered.
2	Check Navbar	"Sign In" and "Sign Up" links appear.	Pass	UI updates to public view.
3	Verify Token	Check browser localStorage.	Pass	Client-side token cleared successfully.
4	Force Fail (Simulated)	User remains on page after click.	Pass	Logout event handler failed to fire.

Post-condition: The user is logged out and cannot access protected routes.

3.5 Summary

This chapter presents a comprehensive testing approach to ensure the "Sustain Sports" system satisfies all of its objectives. The testing period employed a variety of methods/testing modes to make sure the application's structure and performance were sound. That began with Unit Testing, in which the critical pieces were tested independently. This involved auditing the CartContext state logic to ensure price computations were functioning correctly and ensuring our User model's security policies for password hashing held.

Then, Integration Testing verified seamless communication between the React frontend and Node. js/Express backend. This made sure the API endpoints managed data transmission correctly and that security middleware successfully secured sensitive routes from unauthorized users.

Furthermore, rigorous System (Black Box) Testing was conducted by mimicking real-user work patterns to validate end-to-end workflows. That helped stop the e-commerce cycle, from product discovery to secure checkout. It's also successfully tested Donation & Reward feature of the project. This means that administrative approvals have caused automatic rewards treating generation for a user. The success of these different test cases suggests that the system is secure, reliable and fully operational thus validating it for production use

Chapter 4 Deployment and Maintenance

4.1 Introduction

The last part of the "Sustain Sports" project lifecycle is deployment and maintenance, these phases are discussed in this chapter. Once development is complete, testing has been thorough and the app is ready for prime time, the app gets pushed from a local development server to another live production server. This section covers the Software Release Life Cycle (SRLC) for easy deployment, MERN stack strategies (React frontend and Node. js backend) and maintenance plan (to keep it safe, updated, working).

4.2 Try to follow the SRLC (software release life cycle)

"Sustain Sports" followed a well-defined Software Release Life Cycle (SRLC) to achieve stability and reliability. This process has several characteristic stages:

Phase 1: Development (Pre-Alpha)

Activity: Initial coding where we constructed the mern stack architecture.

Status: We used React, Vite, Node for the front end (frontend). js and Express. Specifically we added a the Donation System and Admin Dashboard tested on localhost:5173.

Phase 2: Alpha Testing (Internal)

Activity: In-House Internal “White Box” Testing conducted by the developer.

Actions:

- Tested each unit of CartContext.
- We also checked the schemes of our DB (User, Product and Donation) to complete our data integrity process.
- We verified the security middleware (authMiddleware. js) validations ensure that protected routes are not accessible without authorization.

Phase 3: Beta Testing (User Acceptance)

Activity: “Black Box” testing in which we tested (and reported on) the system as a user might use it (see Chapter 3).

Actions:

- We verified all the checkout process.
- "Donation & Reward" cycle was tested with Donate, approve Admin and check promo codes were generated.

Phase 4: Release Candidate (RC)

Activity: The code is ready for production.

Actions: * We refactored the code by removing console. log debugging statements.

We set up and sealed the environment variables (such as MONGO_URI or JWT_SECRET).

We built the frontend production by npm run build (Vite), which means we turned React components into static html, css and js files.

Phase 5: Production Release (Deployment)

Task: Release the app on public-facing live servers.

Database: We used the MongoDB database, hosted in a cloud-based cluster via MongoDB Atlas, to ensure fault tolerance.

Backend: We deployed the Node. js/Express server on a platform such as Render or Heroku that connects to the Atlas database.

Frontend: We hosted the React static build on a Content Delivery Network (CDN) service which is globally distributed, like Vercel or Netlify so that it will load fast.

Phase 6: Maintenance & Support

Action: Tracking & updates post GTM.

Bug Fixes: We fixed any bugs users reported after we launched.

Chapter 5 User Manual

5.1 Introduction

This section provides a detailed overview on the “SustainSports” application end-users. The goal is to offer simple, step-by-step guidance for the use of the system and include descriptions on how different features are used. The guide includes key functions needed for all user types, such as browsing and searching of green products, account management functionality, detail on the process to purchase product, in addition to participation within the recycling program. It also describes the management and operation functions that can be used in relation to content and platform. This guide is intended to make the experience of customers and administrators as smooth and intuitive as possible.

5.2 Project Functionalities

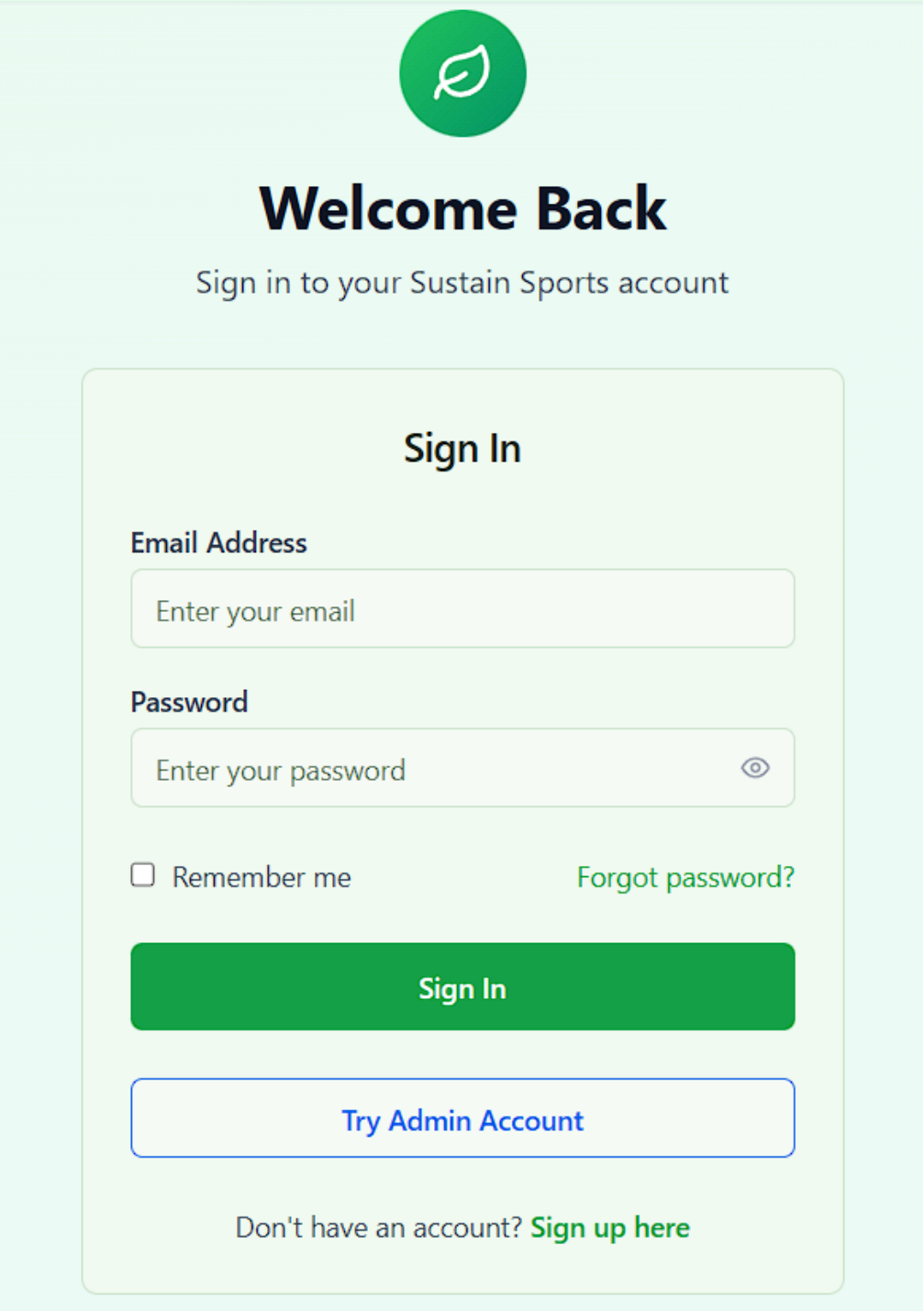


Figure 5.1: Admin & User Sign In

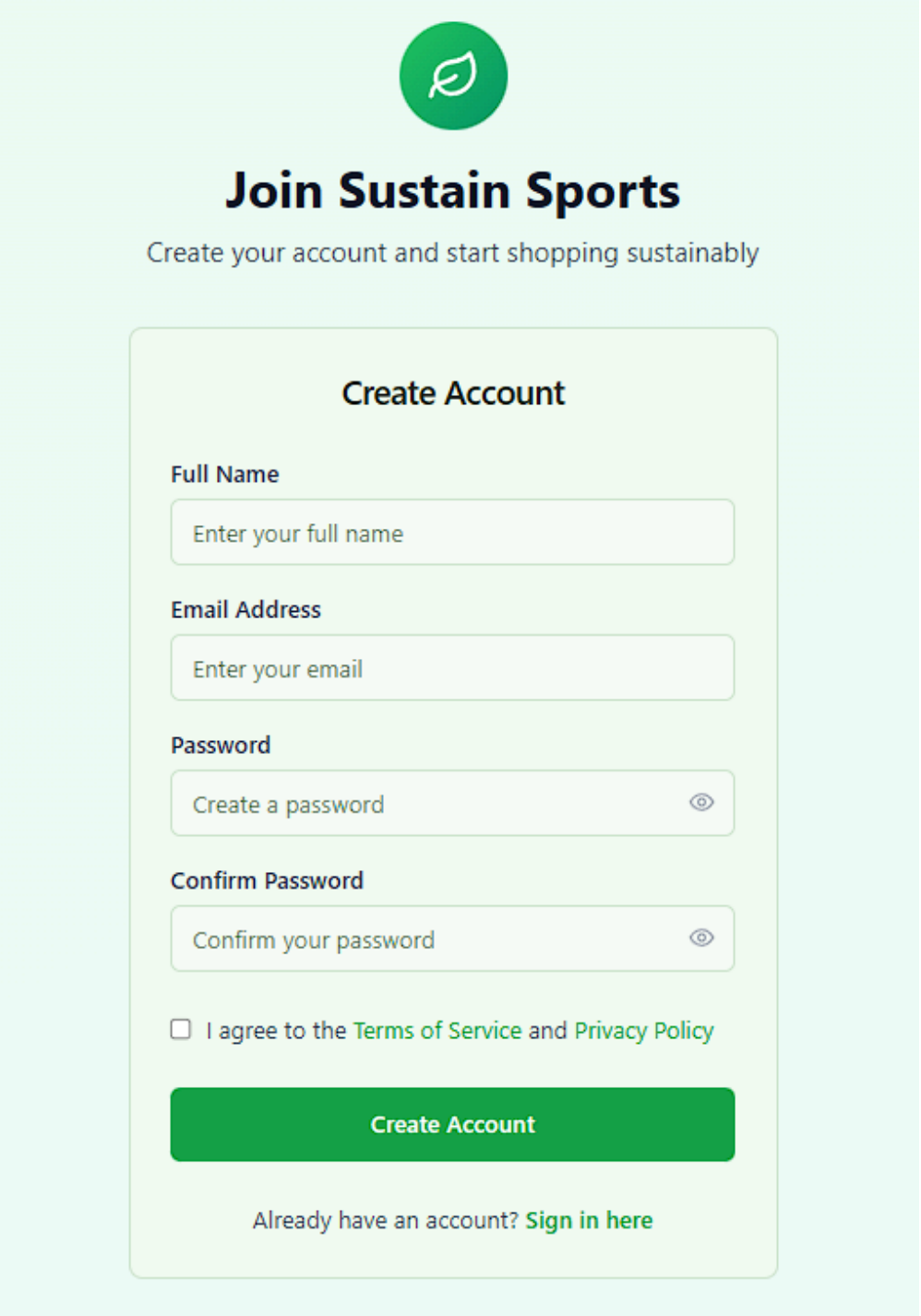


Figure 5.2: User Sign Up

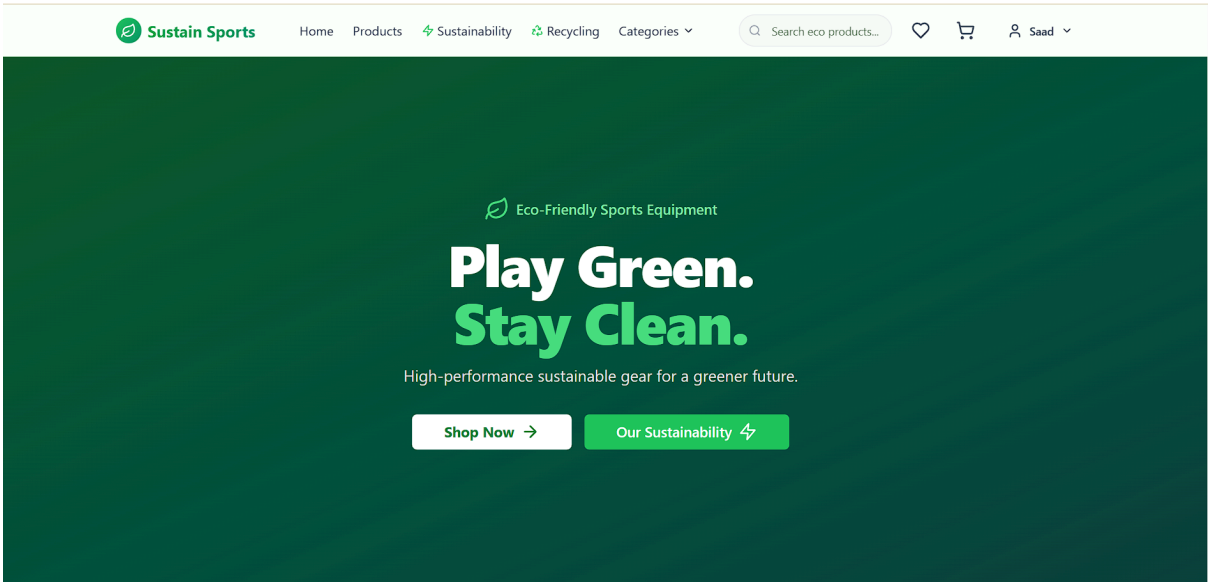


Figure 5.3: Home Page

Figure 5.4:Why Choose Sustain Sports

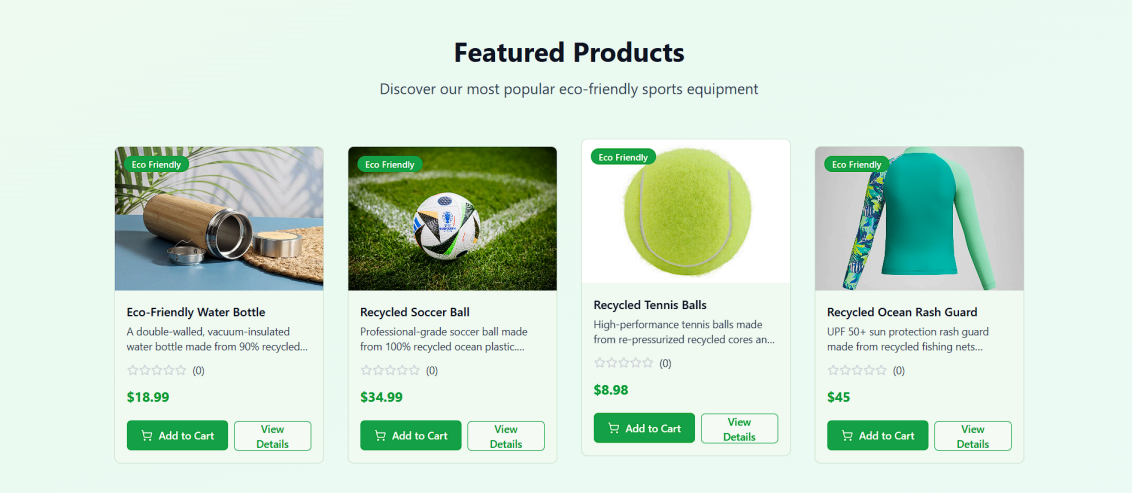


Figure 5.5:Featured Products

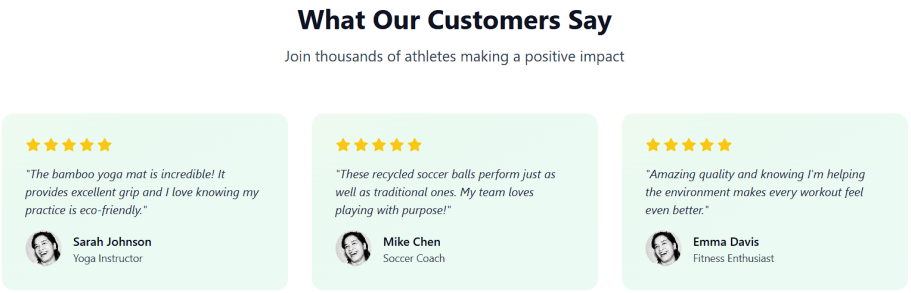


Figure 5.6:What Customers Say

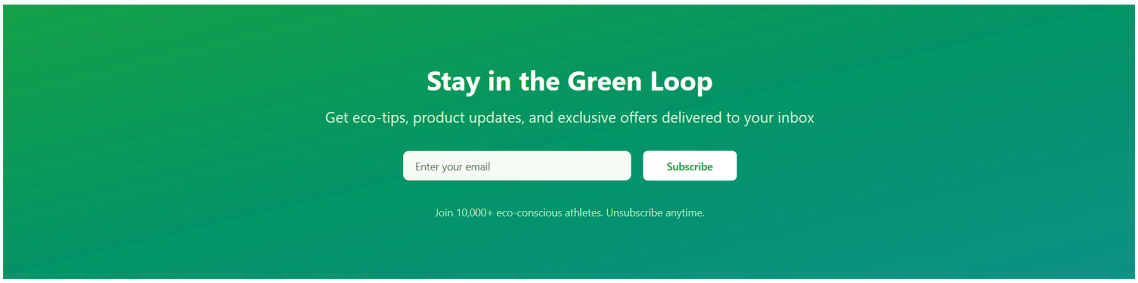


Figure 5.7:Subscribe

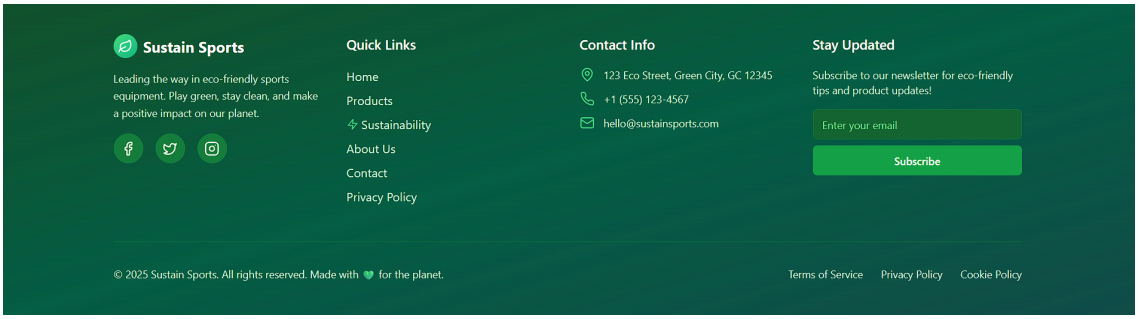


Figure 5.8:Footer

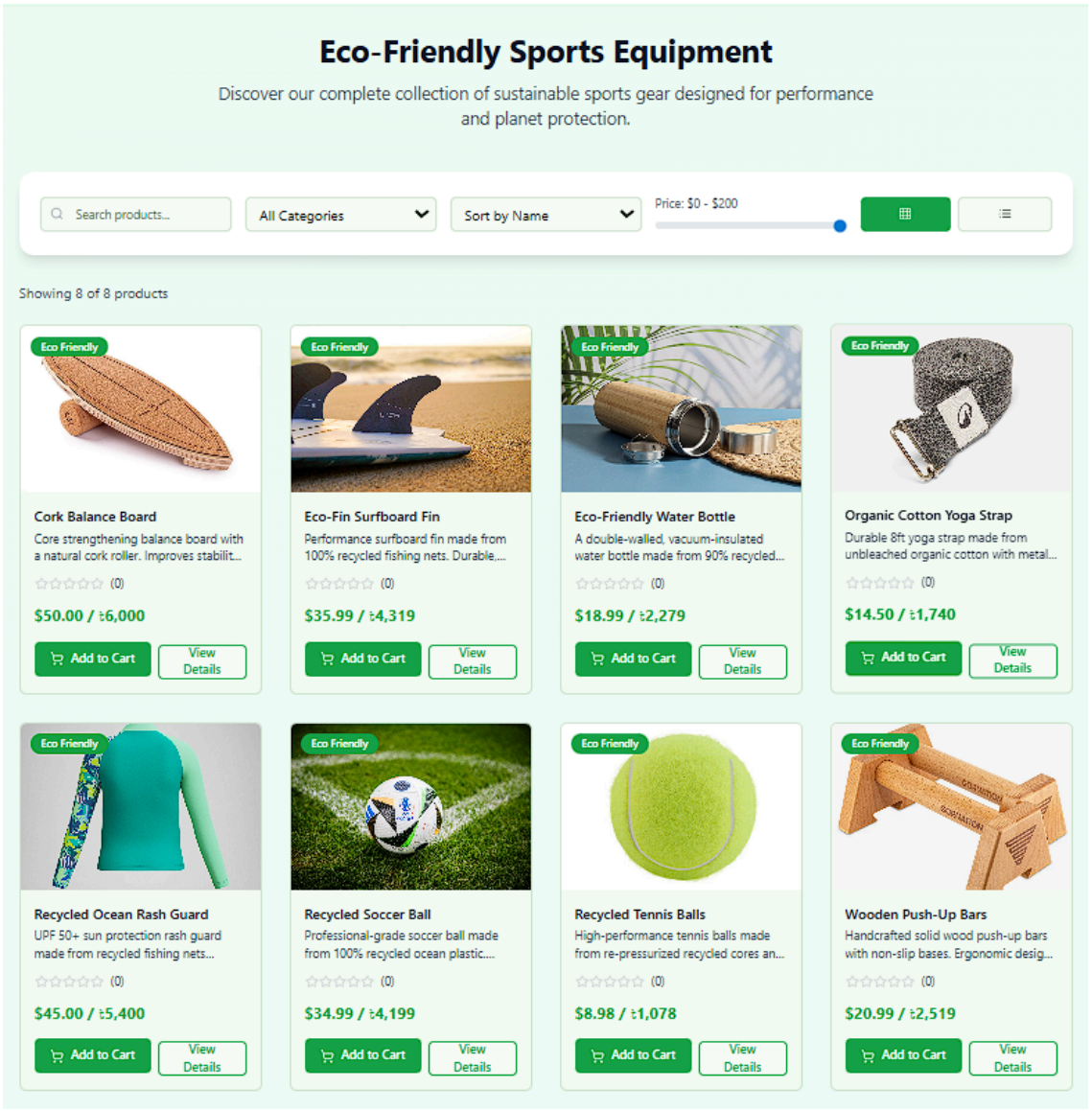


Figure 5.9:Browse Product & Filtering

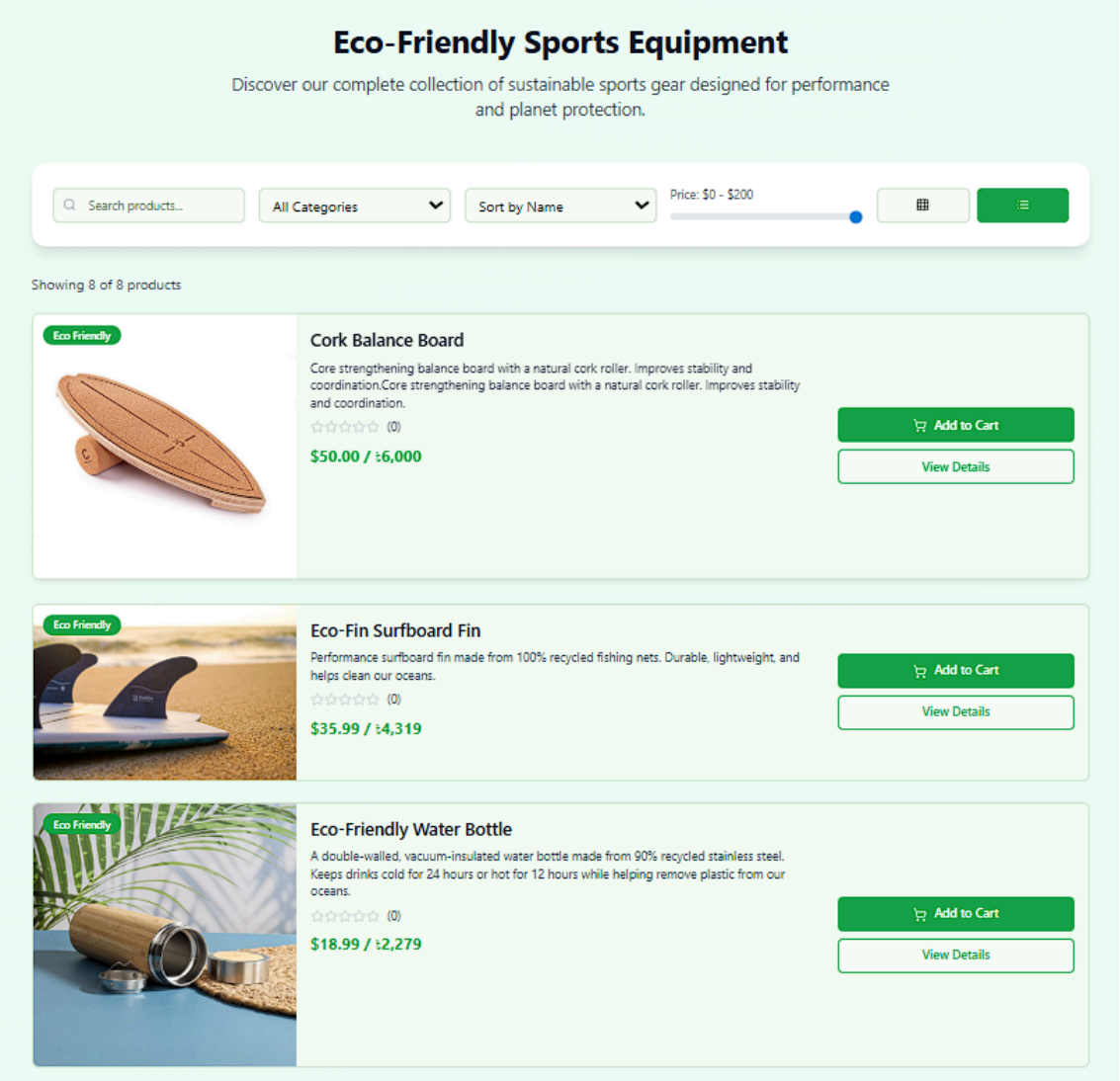


Figure 5.10:Browse Product & Filtering (List View)

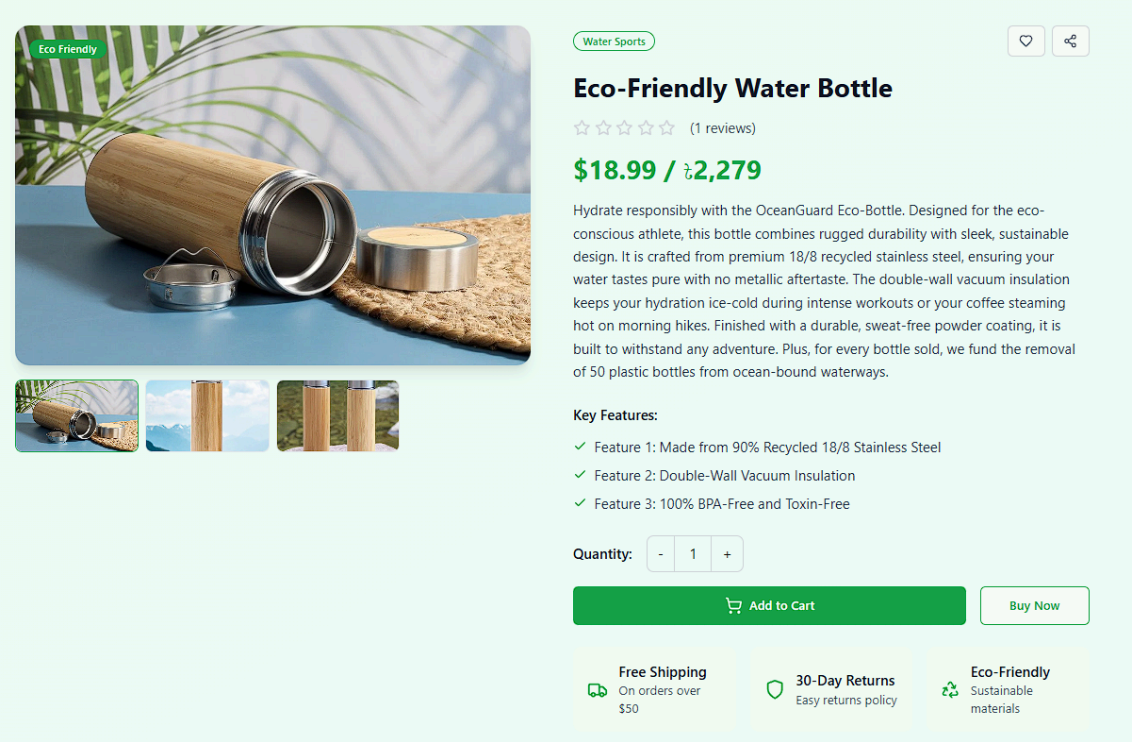


Figure 5.11:View Product Details

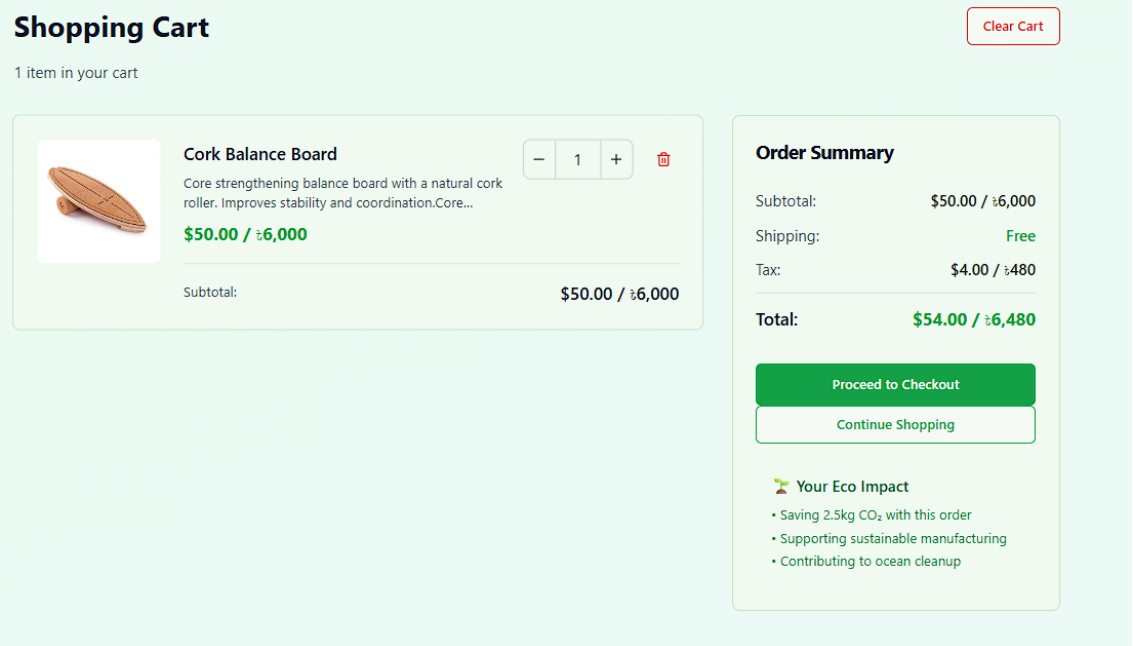


Figure 5.12:Manage Cart

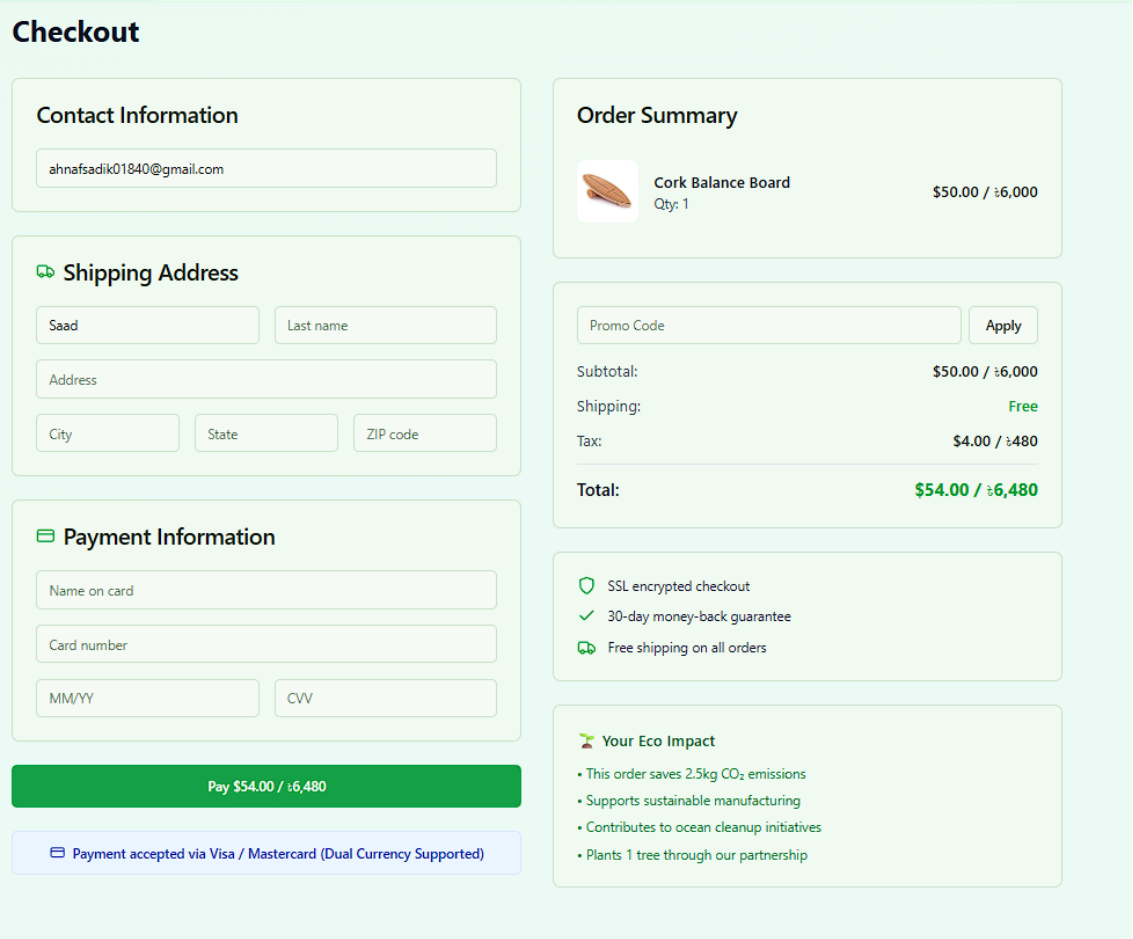


Figure 5.13: Product Checkout

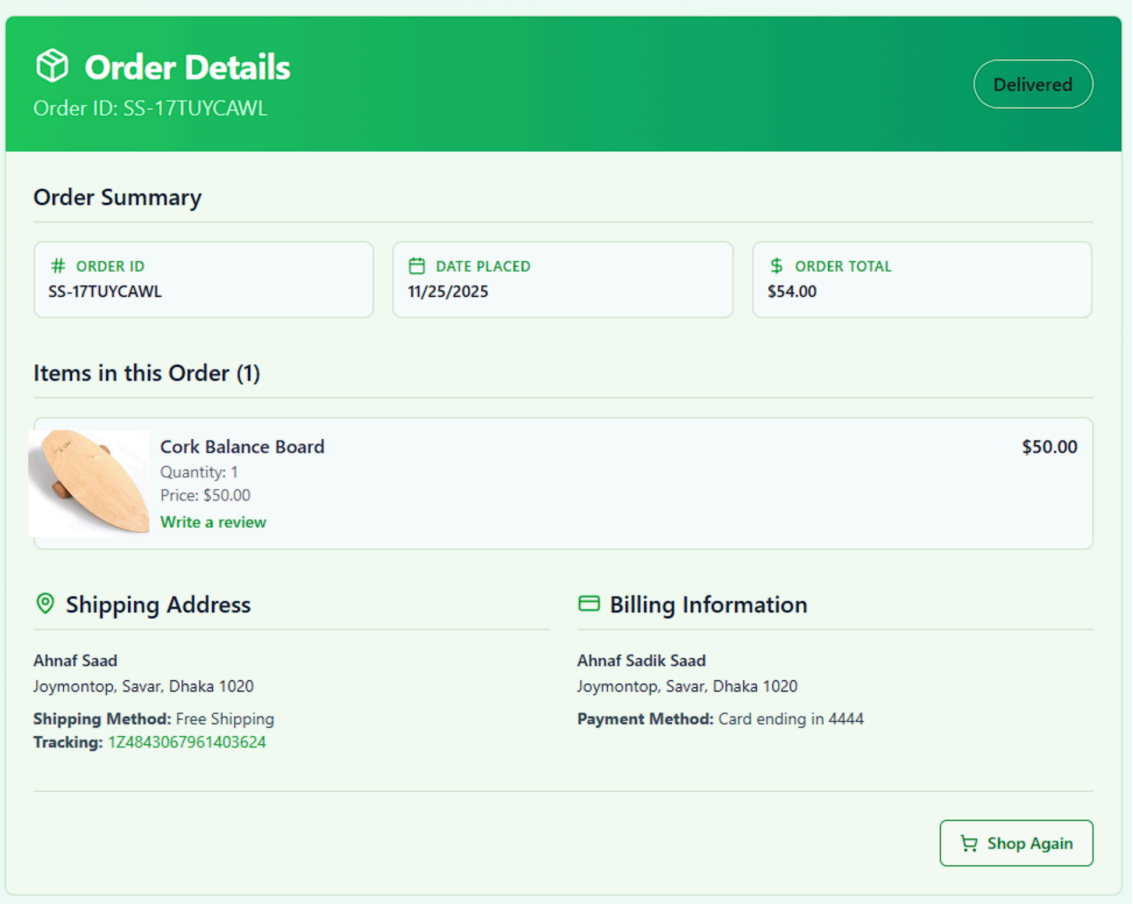


Figure 5.14: Order Details

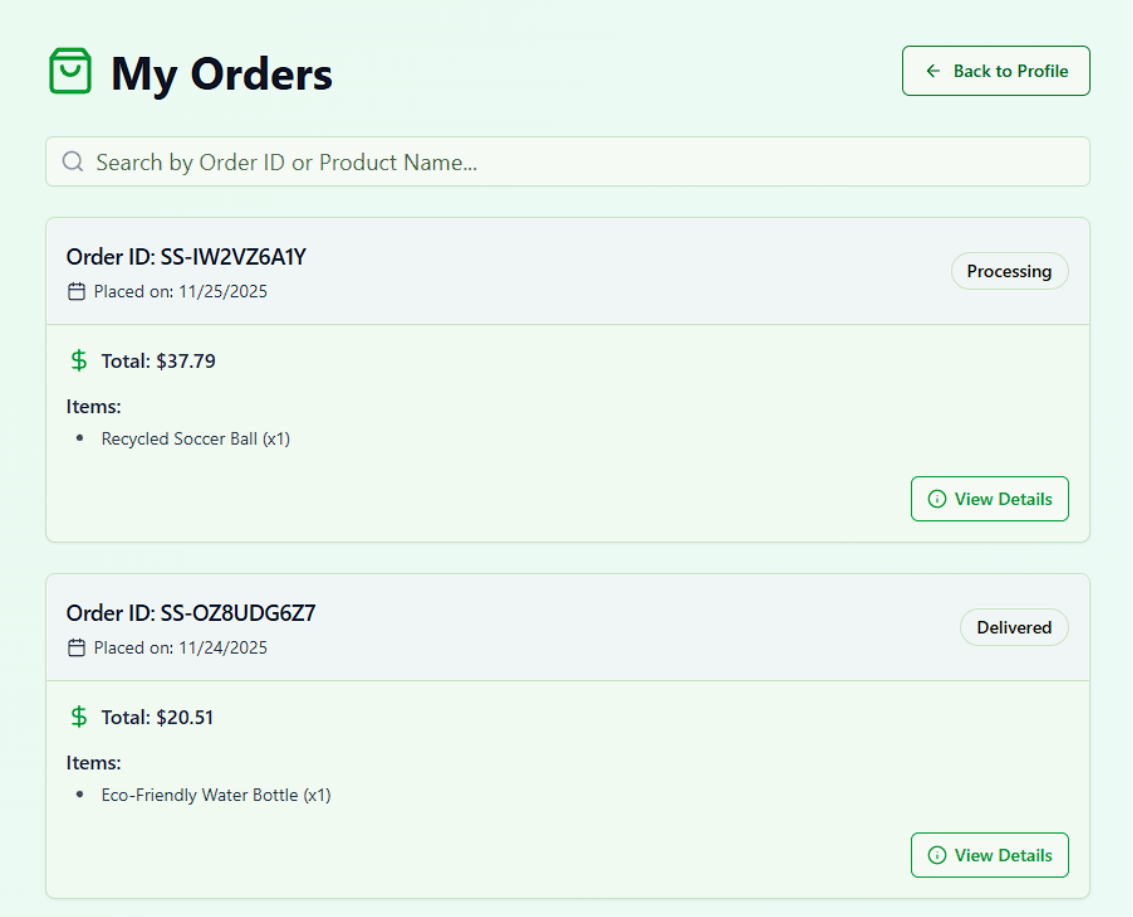


Figure 5.15: Orders History

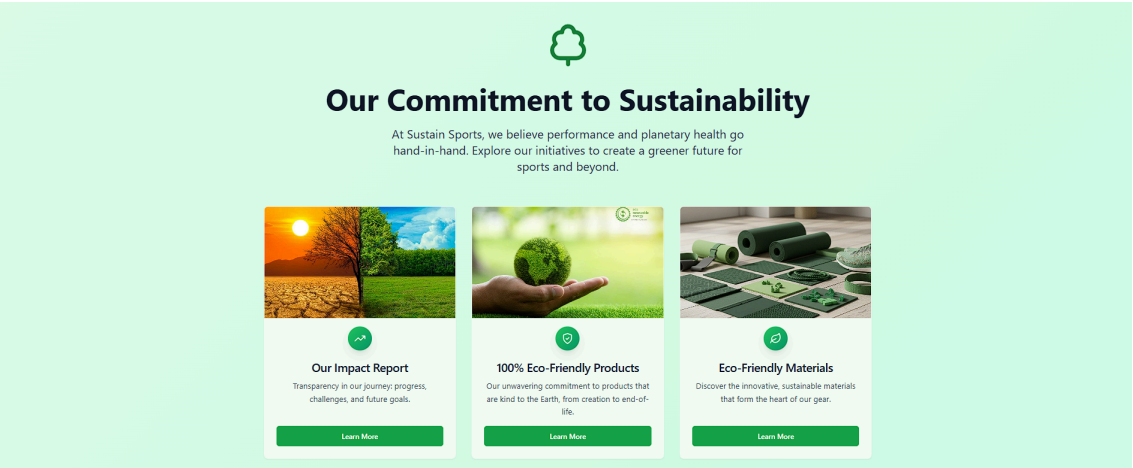


Figure 5.16: Sustainability

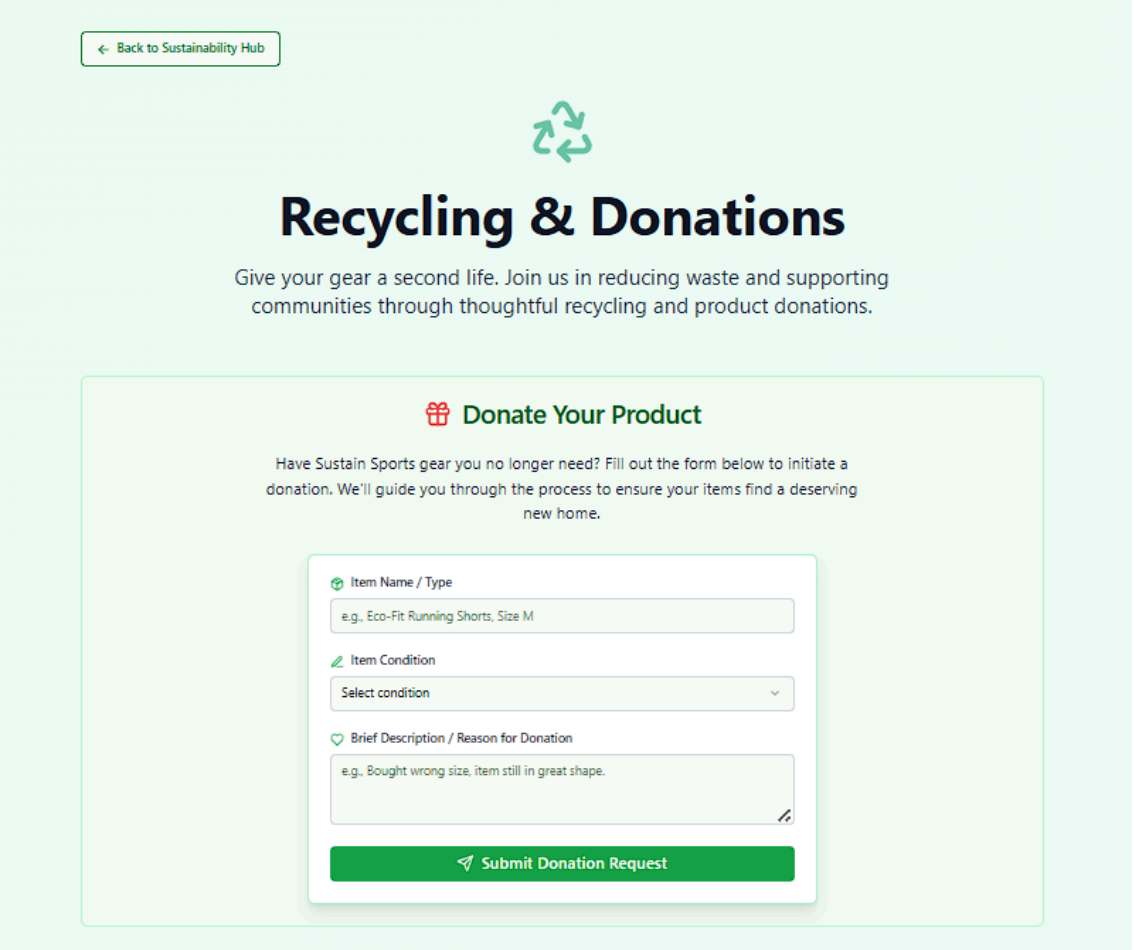


Figure 5.17: Submit Donation

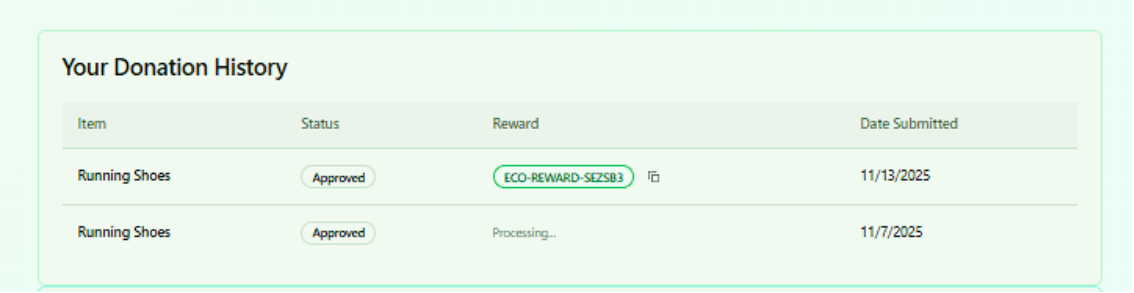


Figure 5.18: Donation Reward

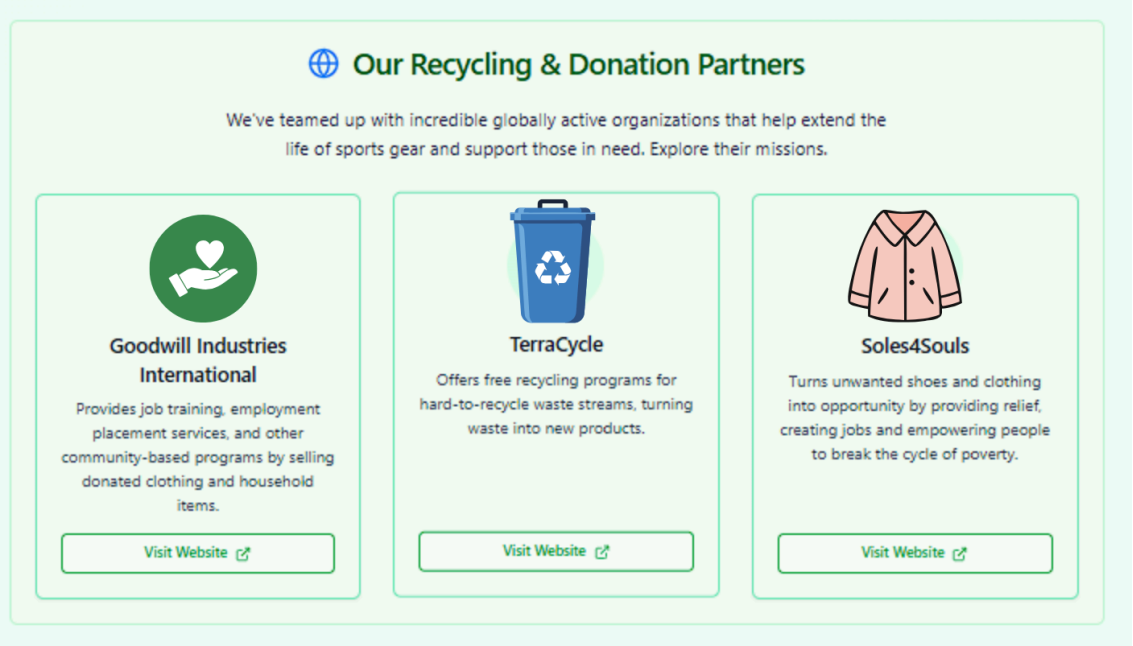


Figure 5.19: Recycling Partners

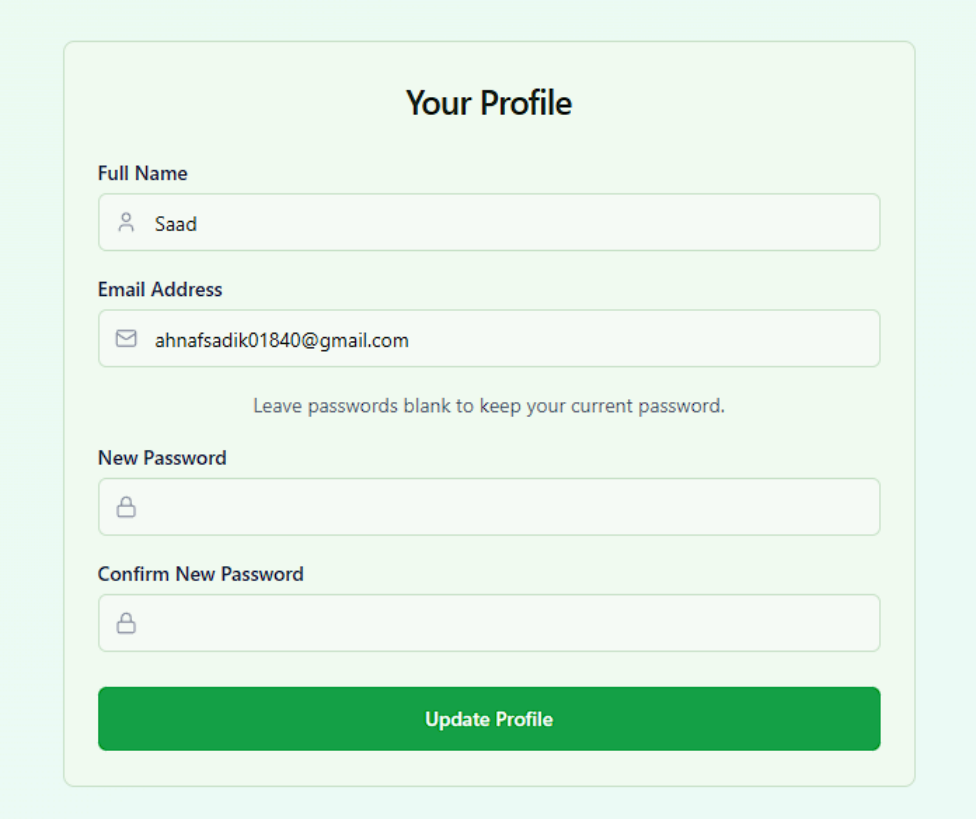


Figure 5.20: Update Profile

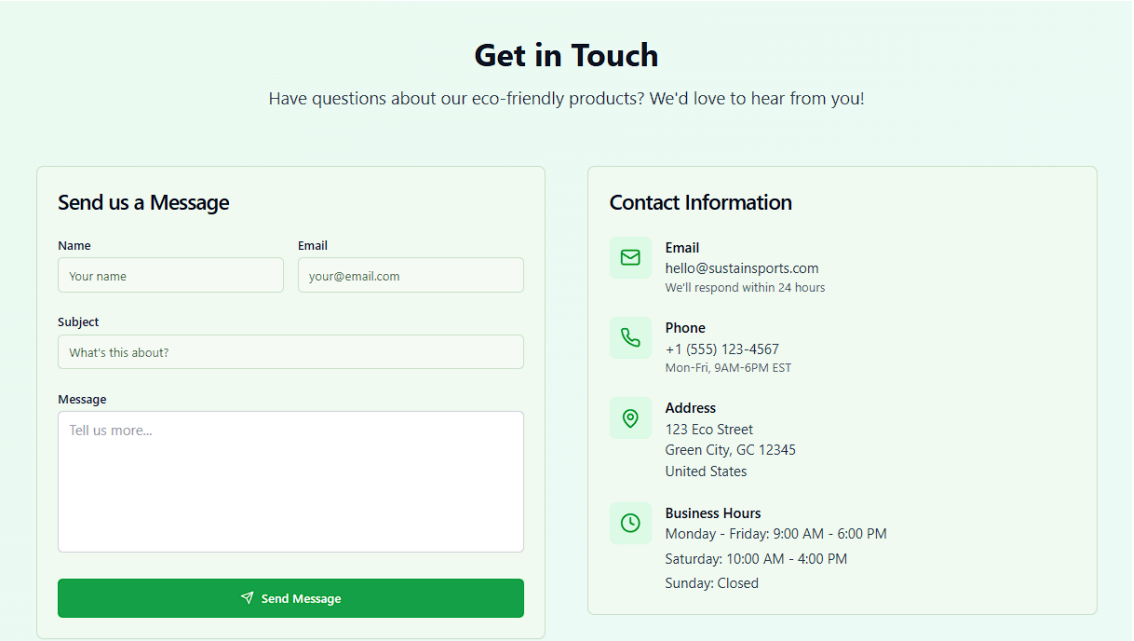


Figure 5.21: Contact Us

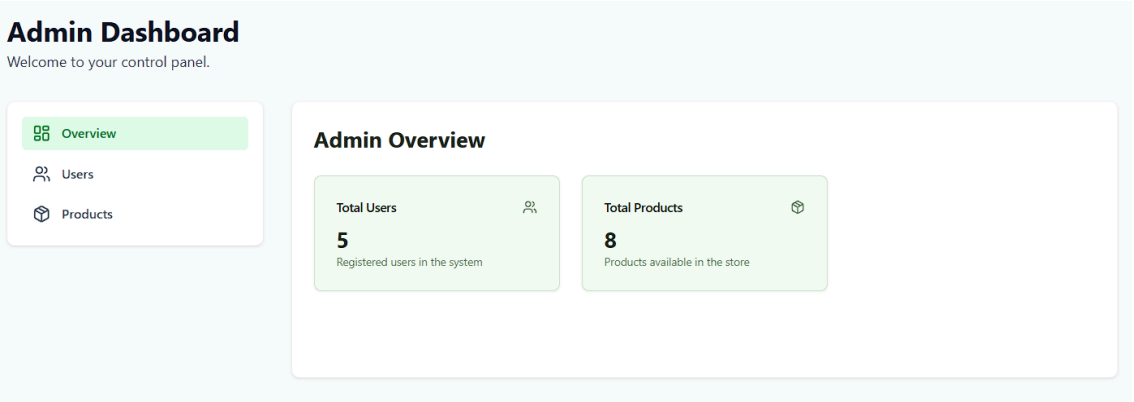


Figure 5.22: Admin Dashboard

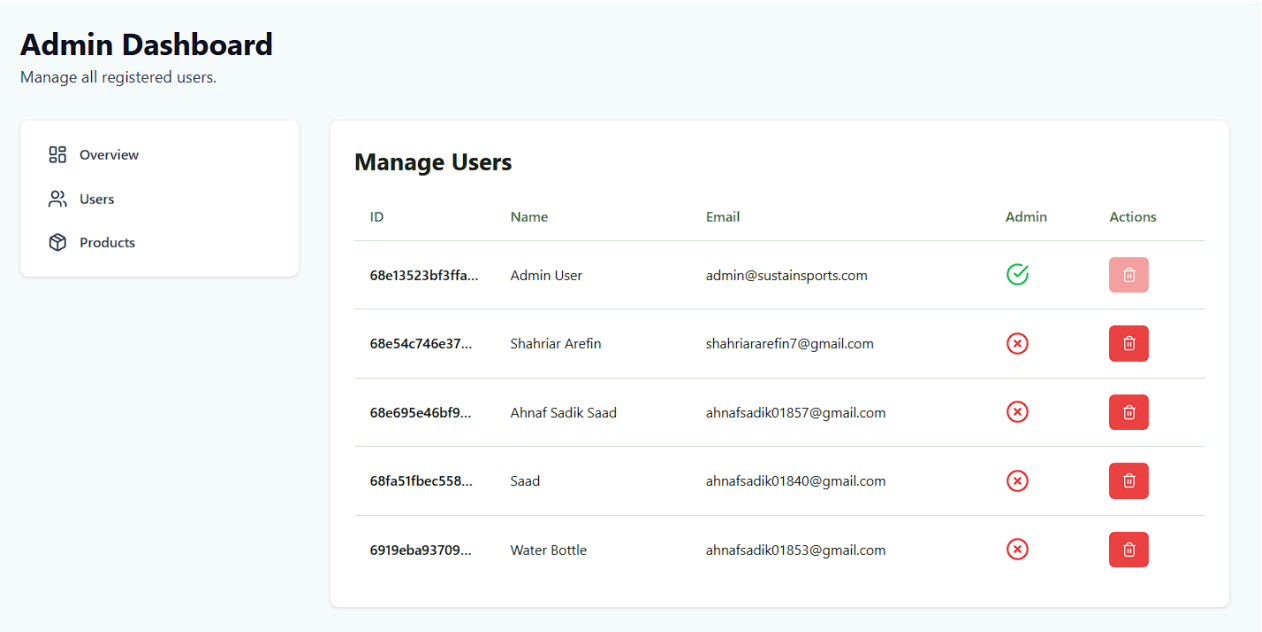


Figure 5.23: Manage Users

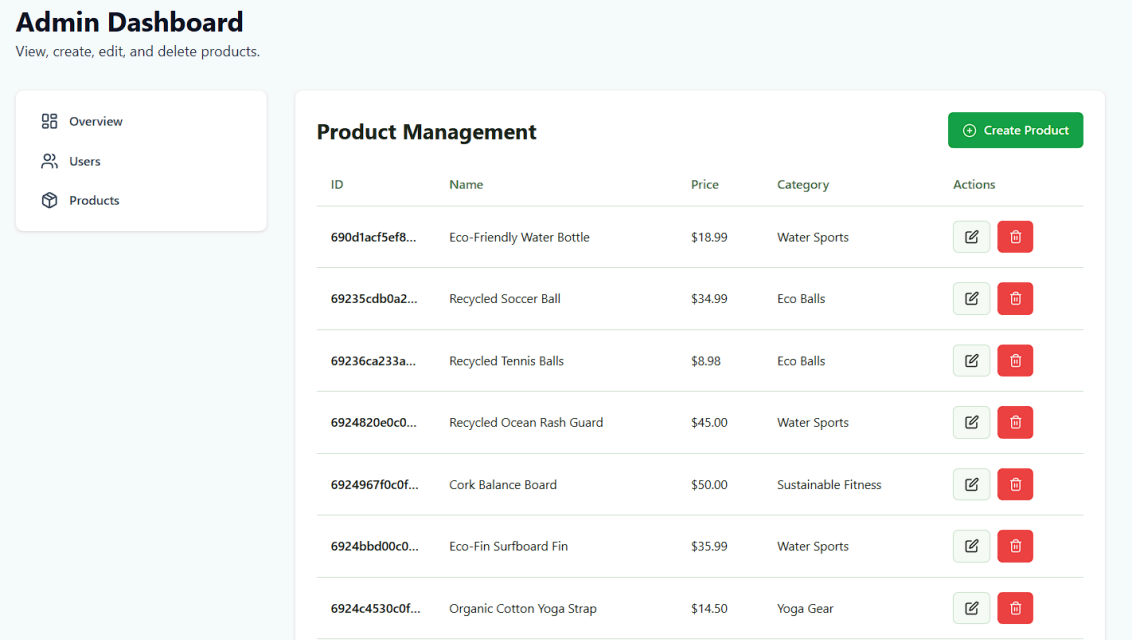


Figure 5.24: Manage Products

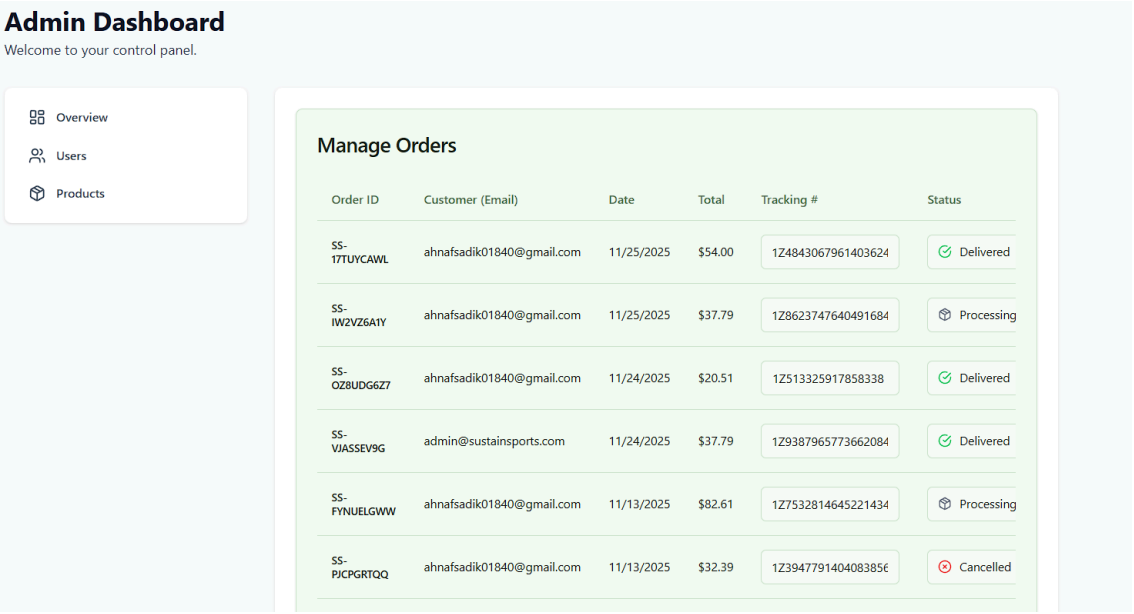


Figure 5.25: Manage Orders

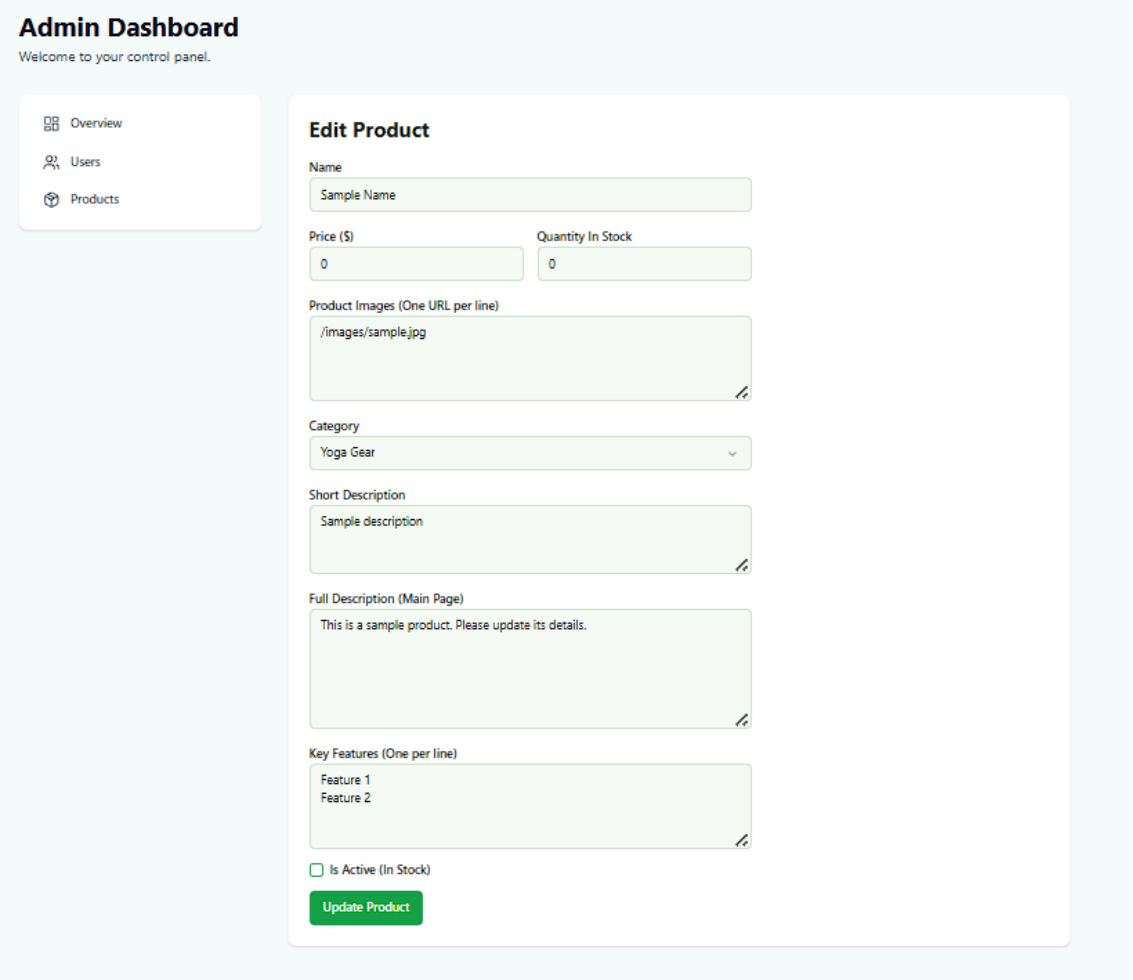


Figure 5.26: Create Product

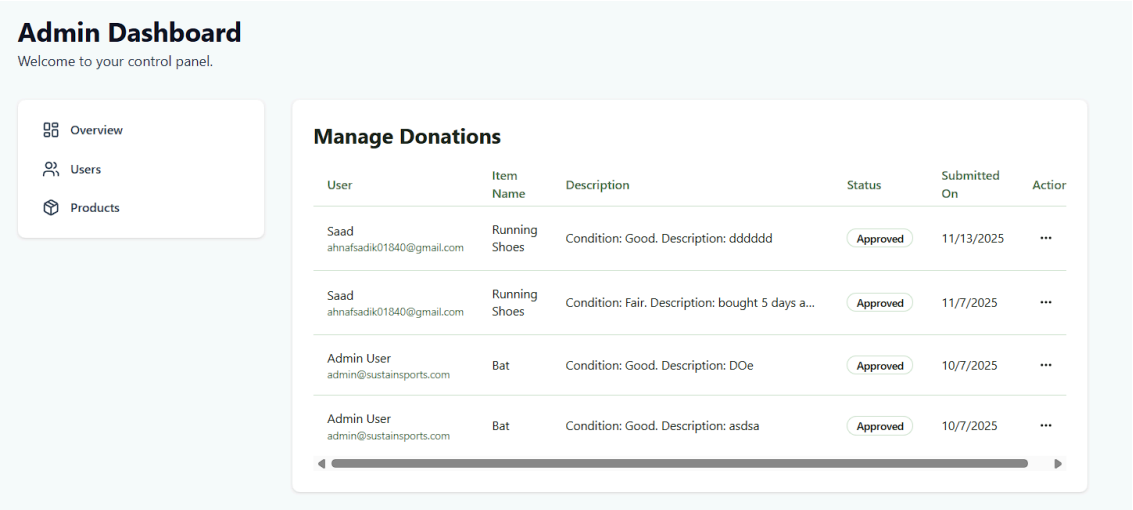


Figure 5.27: Manage Donations

5.3 Summary

Chapter 5 has offered very useful instruction formative contents on all users of the 'Sustain Sports' platform. It explained the shopping processes for Viewers, walking them through the ability to view and search for products or utilize filters as well as for Registered Users to access, secure transactions and participate in a one-of-a-kind donation-for-reward program. The chapter also contained an in depth user manual for Administrators, that details important operational responsibilities such as Managing inventory, Processing Orders and Validating Recycling requests to include a few examples. All parties will be able to effectively use the system and maximise throughput by understanding these principles.

Chapter 6 Project Summary

6.1 Introduction

The "Sustain Sports" project is designed to address an important bridge between contemporary e-commerce and environmental consciousness. Since the global sports equipment market is contributing to waste and carbon emissions, Preventas has identified a need for a service targeting "conscious consumers" where you don´t have to choose between performance and buying with a clear conscience. This project report documents all the phases of software development life cycle (SDLC) that addresses the void, starting from preliminary feasibility and requirements collections to complete system design, implementation and testing.

The outcome of this is a powerful full-stack web application that runs on the MERN stack (MongoDB, Express. js, React, Node. js) architecture. There's more to "Sustain Sports" than typical retail content. It features a dedicated "Circular Economy" rewards program that incentivizes the re-entry of users' old gear, to close the loop on product life cycles. Project Outputs are summarised in this final chapter. It provides a comprehensive summary of the work performed, discusses the problems encountered during development and outlines plans for future enhancements. These are some of the parameters we can use to assess global success of the project, in terms of achieving its objectives and its potential impact on sustainable sport market.

6.2 Project Limitation

However, some limitations and constraints were noted despite the successful inclusion of these elements:

SEO: As a React Single Page Application (SPA), the platform has native SEO challenges vs server side rendered application. This can impact its ability to be discovered organically.

Social Media Login: The app does not offer an option to sign up with social media. It now does not have link with OAuth for one click login using Google, Facebook etc.

Chat Support: There's no chat interface with a live human, such as chatbot or live widget utilizing Socket. io, to assist users instantly.

Multi language Support (Localisation): App is writen entirely in English. There's no system equivalent to i18n for dynamic language switching or localization for non-English speakers.

AI Product Recommendations: The ML and AI are not used by the system to study users behavior or what they had bought in order to suggest products.

6.3 Scope

The "Sustain Sports" project had a clear goal to create a strong web-based e-commerce solution.

The “Sustain Sports” project was tasking to develop a solid Web based e-commerce site.

What the Project Covers (Inclusions):

Public Storefront: A responsive front end for browse, search, filter of green products.

User account management: Secure register, login, profile settings and order history.

E-Commerce Capabilities: An active shopping cart, wish list ability, and full checkout features.

Admin Administration: A safe hub for managing customers, products, orders and donation requests.

Reward System: system which generate promo code automatically after user donation get accepted.

Boundaries (Exclusions):

App Development: A native iOS or Android app was omitted; the scope of the project is just a responsive website.

Live Inventory: There was no connection to on-site warehouse stock or third-party inventory software.

Automated Email Notifications: The software does not have the ability to email real time transactional emails (i.e. sending "Welcome" or "Order Confirmation") emails out to users and everything is run via in-app notifications currently.

6.4 Future Work

To enhance the capacity and commercial prospect of the platform, several future works are outlined below:

SaaS Transformation: Moving the platform to a multi-vendor Software as a Service (SaaS) offering. This would allow other sustainable brands to sign up as merchants, control their inventories and sell product through the "Sustain Sports" marketplace. And that becomes a hub for green business.

Artificial Intelligence Recommendations: Based on machine learning model which sees purchasing history and browsing pattern of the user. This will enable you to suggest products that are personalised and enhance user engagement.

Social Media Login: Implement OAuth services such as Google and Facebook to allow users to login easier, increasing signup conversion rates.

Live Chat Support: Integrate a customer support chatbot or live agent system so users can get answers to product questions in real time.

6.5 Conclusion

The main objective - creating a reliable marketplace for sustainable sport products SustainSports has achieved its most important goal. This is the full software development lifecycle (SDLC) of the application, where we went from a concept about combating “greenwashing” in sports to a working and verifiable web application.

Utilizing the latest MERN stack (MongoDB, Express. js, React, Node. js) the team designed a modular and scalable architecture. The application carries standard e-commerce functionality (secure log in, dynamic product catalog, and persistent shopping cart) yet it is infused with mission-centric features that make the tool one-of-a-kind. API backend was both secure and responsive, employing jwt authentication with role-based access to protect sensitive data and administrative tasks. React (with Vite and Tailwind CSS) was used on the frontend, resulting in a fast and responsive user interface that is great to use on all devices.

The successful production of the "Circular Economy Reward System" is an important result from this project. This component generates promotion codes when donation requests are approved by admin. It also neatly “closes the loop” on product lifecycles, and by doing so provides a clear end-user benefit to using the application in an ecologically friendly manner – aligning users’ day-to-day interaction with its predominantly eco-based objectives. It was through heavy testing (Unit, Integration and System Testing) that the solidity of this logic got established and system would scarcely have major defect had it been tested with proper data.

And, also, this project was a comprehensive full-stack software engineering project. We were taught the importance of global state control in SPAs, and strict input validation and testing to keep our code quality high. There are still some limits - you cannot live process a payment and there is no automated alert / emails setup, however the core system is 95% done and works GREAT! The "Sustain Sports" platform now also represents a robust, deployable base, with a blueprint for scaling into a multi-vendor SaaS ecosystem - well poised to make a serious dent in the market place for sustainable athletic equipment.

REFERENCES

React Documentation. (n.d.). *React: The library for web and native user interfaces*. Retrieved from https://react.dev/

Node.js Documentation. (n.d.). *Node.js v20.19.1 Documentation*. Retrieved from https://nodejs.org/en/docs/

Express.js. (n.d.). *Fast, unopinionated, minimalist web framework for Node.js*. Retrieved from https://expressjs.com/

MongoDB Documentation. (n.d.). *MongoDB Manual*. Retrieved from https://www.mongodb.com/docs/manual/

Mongoose ODM. (n.d.). *Mongoose v8.0.3 Documentation*. Retrieved from https://mongoosejs.com/docs/

Tailwind CSS. (n.d.). *Rapidly build modern websites without ever leaving your HTML*. Retrieved from https://tailwindcss.com/docs/

Vite. (n.d.). *Next Generation Frontend Tooling*. Retrieved from https://vitejs.dev/guide/

JSON Web Token (JWT). (n.d.). *Introduction to JSON Web Tokens*. Retrieved from https://jwt.io/introduction

Axios. (n.d.). *Promise based HTTP client for the browser and node.js*. Retrieved from https://axios-http.com/docs/intro

United Nations. (2015). *The 17 Sustainable Development Goals*. Retrieved from https://sdgs.un.org/goals

Ellen MacArthur Foundation. (n.d.). *What is a circular economy?*. Retrieved from https://ellenmacarthurfoundation.org/topics/circular-economy-introduction/overview

Shopify. (2023). *The Future of Commerce: Trends to Watch in 2023*. Retrieved from https://www.shopify.com/research/future-of-commerce

McKinsey & Company. (2020). *Consumer sentiment on sustainability in fashion*. Retrieved from https://www.mckinsey.com/industries/retail/our-insights/survey-consumer-sentiment-on-sustainability-in-fashion

Statista. (2023). *Global retail e-commerce sales 2014-2027*. Retrieved from https://www.statista.com/statistics/379046/worldwide-retail-ecommerce-sales/

221-35-892

ORIGINALITY REPORT

21%	14%	6%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Daffodil International University Student Paper	8%
2	dspace.daffodilvarsity.edu.bd:8080 Internet Source	2%
3	Submitted to NCC Education Student Paper	2%
4	Submitted to Universiti Teknologi Malaysia Student Paper	1%
5	Submitted to FICT Student Paper	1%
6	Submitted to Macquarie University Student Paper	1%
7	Submitted to Midlands State University Student Paper	1%
8	Submitted to De Montfort University Student Paper	<1%
9	slidetodoc.com Internet Source	<1%
10	docshare.tips Internet Source	<1%
11	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1%

12	umpir.ump.edu.my Internet Source	<1%
13	Submitted to TAR University College Student Paper	<1%
14	123dok.com Internet Source	<1%
15	Submitted to Dr. B R Ambedkar National Institute of Technology, Jalandhar Student Paper	<1%
16	Submitted to University of Greenwich Student Paper	<1%
17	Submitted to University of Portsmouth Student Paper	<1%
18	indah.ump.edu.my Internet Source	<1%
19	Submitted to Sabanci Universitesi Student Paper	<1%
20	Submitted to University of Huddersfield Student Paper	<1%
21	digilib.k.utb.cz Internet Source	<1%
22	Submitted to City University of Hong Kong Student Paper	<1%
23	Phenikaa University Publication	<1%
24	"Software Architecture. ECSA 2025 Tracks and Workshops", Springer Science and Business Media LLC, 2026	<1%

	Publication	
25	Ferreira, Maria Isabel Peixoto. "The Role of the Resource Owner and Product Presentation Used in Luxury Fashion Rental Services", Universidade do Porto (Portugal), 2025 Publication	<1%
26	Submitted to RMIT University Student Paper	<1%
27	Submitted to University of Sunderland Student Paper	<1%
28	Ge Chang, Yuhan Zhang, Xinsu Liu. "Chapter 24 The Construction of an Intelligent Autonomous Learning System Based on Machine Learning Algorithms", Springer Science and Business Media LLC, 2025 Publication	<1%
29	Babeș-Bolyai University Publication	<1%
30	Submitted to Gujarat Technological University Student Paper	<1%
31	Submitted to Higher Education Commission Pakistan Student Paper	<1%
32	Submitted to Coventry University Student Paper	<1%
33	David A. Fennell. "Routledge Handbook of Ecotourism", Routledge, 2021 Publication	<1%
	Submitted to University of East London	

34	Student Paper	<1 %
35	Fonseca, Francisca Maria Salgado. "Monitoring Pregnant Women Lifestyle", Universidade do Minho (Portugal), 2023 Publication	<1 %
36	ijirt.org Internet Source	<1 %
37	www.theseus.fi Internet Source	<1 %
38	Submitted to University of Liverpool Student Paper	<1 %
39	bird.com Internet Source	<1 %
40	www.numberanalytics.com Internet Source	<1 %
41	www.slideshare.net Internet Source	<1 %
42	su-plus.strathmore.edu Internet Source	<1 %
43	(2-9-13) http://129.16.157.207/dat255-vt12/?page_id=48 Internet Source	<1 %
44	Submitted to Harrisburg University of Science and Technology Student Paper	<1 %
45	Submitted to University of Waikato Student Paper	<1 %
	doczz.cz	

46	Internet Source	<1%
47	dspace.cvut.cz Internet Source	<1%
48	essay.utwente.nl Internet Source	<1%
49	moldstud.com Internet Source	<1%
50	opus4.kobv.de Internet Source	<1%
51	sekin.in Internet Source	<1%
52	trap.ncirl.ie Internet Source	<1%
53	www.faa.gov Internet Source	<1%
54	www.isteonline.in Internet Source	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		