ECOMARKET

COURSE NAME: ADVANCED DATABASE SYSTEMS DESIGN

CS 54901: SPRING 2025

TEAM NAME: GREENCOMMERCE

TEAM MEMBERS:

- Haripriya Animireddy
- Nikhil Divi
- Jathin Chava

DATE OF SUBMISSION: May 07, 2025

OVERVIEW OF THE PROJECT

EcoMarket is a full-stack web application built as a sustainable online marketplace that connects environmentally conscious consumers with local eco-friendly vendors. The core idea is to provide a centralized platform where users can discover, purchase, and learn more about sustainable products, while empowering local businesses to market goods that align with eco-conscious values. The application was designed with accessibility, ease of use, and scalability in mind, using technologies such as Flask for the backend and HTML/CSS/JavaScript for the frontend.

GOALS AND OBJECTIVES

The primary objectives of EcoMarket were:

- To create an online marketplace that promotes sustainable living and supports eco-friendly businesses.
- To enable users to browse, search, and purchase eco-conscious products with ease.
- To allow vendors to register, list their products, and manage inventory through a secure and user-friendly dashboard.
- To implement a simple and intuitive user interface that enhances user experience.
- To build a scalable backend infrastructure capable of supporting future integrations like payment gateways, user reviews, and personalized recommendations.

MAJOR DELIVERABLES AND FEATURES

During the development of EcoMarket, the team focused on delivering the following key features:

- Vendor Portal: Vendors can add, update, or remove product listings using a dedicated dashboard.
- User Browsing Interface: Shoppers can view available products, with categories and descriptions to guide purchases.

- Product Management: Complete CRUD (Create, Read, Update, Delete) functionality for products.
- Custom Frontend Design: Responsive layout and styling using CSS to ensure the site looks good on all devices.
- Templating System: Dynamic HTML rendering using Jinja2 to provide reusable page structures.
- Database Integration: Structured data storage using SQLite (with design considerations for PostgreSQL/MySQL scalability).
- Planned Extensions: Payment integration (Stripe/PayPal), smart recommendation system, and Google Maps for vendor localization.

SUMMARY OF ACCOMPLISHMENTS AND CHALLENGES

ACCOMPLISHMENTS:

- Successfully implemented the core backend logic using Flask and structured the database to support product and vendor management.
- Designed a clean and responsive user interface with intuitive navigation.
- Developed modular code that separates presentation, logic, and database interaction, which enhances readability and maintainability.
- Effectively collaborated using Git and GitHub, with regular commits and a structured branch workflow.
- Documented the development process and maintained alignment with the original project proposal.

CHALLENGES:

- One of the initial challenges was designing an efficient database schema that could handle multiple vendors, user types, and product categories. This was solved through ER diagramming and refining entity relationships.
- Integrating a real-time payment gateway was deferred due to limited time and the need for merchant credentials; however, the backend is structured to support this in the future.
- Achieving cross-browser compatibility and responsive styling required frequent UI testing and adjustments.
- Coordinating code contributions and resolving merge conflicts in Git took extra time, but ultimately improved team collaboration and understanding of version control best practices.

PROBLEM STATEMENT

WHAT PROBLEM OR NEED DOES THE PROJECT ADDRESS?

In today’s consumer market, there is a growing demand for environmentally responsible purchasing options. However, eco-friendly products are often scattered across various platforms or sold through niche vendors that lack visibility. Additionally, many local businesses committed to sustainable practices struggle to compete with larger e-commerce platforms due to limited online presence and technological support. EcoMarket addresses these challenges by offering a centralized, digital marketplace that bridges the gap between conscious consumers and local sustainable vendors.

EcoMarket solves two key problems:

1. Limited Access to Sustainable Products: Consumers often find it difficult to locate reliable sources for eco-friendly goods in a single, convenient platform.
2. Low Visibility for Local Green Vendors: Small businesses with sustainable missions lack the tools and exposure needed to reach a broader, like-minded audience online.

WHO ARE THE STAKEHOLDERS OR INTENDED USERS?

- Environmentally conscious consumers: Individuals looking for products that align with their sustainable lifestyle values.
- Local eco-friendly vendors: Small businesses or individual sellers offering environmentally responsible products and seeking an online platform to promote their offerings.
- Community organizations and advocates: Groups that support green initiatives and could benefit from collaboration or promoting eco-marketplaces.
- Educational institutions: Schools or universities that promote sustainability efforts and could incorporate EcoMarket into student initiatives or projects.

WHY IS THE PROBLEM IMPORTANT OR INTERESTING?

This problem is significant for both environmental and economic reasons. The global climate crisis demands immediate and meaningful changes in consumer behavior. Providing tools that make sustainable choices easier and more accessible encourages positive change. Furthermore, supporting local green vendors promotes community development, reduces environmental impact from large-scale shipping, and boosts the local economy. From a technical perspective, the challenge of designing a scalable, user-friendly, and socially impactful web application is both intellectually rewarding and practically relevant in today's world. EcoMarket is more than just an e-commerce site it's a step toward building an ecosystem where ethical production, responsible consumption, and community empowerment intersect.

SYSTEM OVERVIEW / SOLUTION DESCRIPTION

HIGH-LEVEL SYSTEM ARCHITECTURE

EcoMarket is a full-stack web application developed using the Model-View-Controller (MVC) architectural pattern. The system is composed of three primary layers:

1. Frontend (View) – Implements the user interface using HTML, CSS, and JavaScript.
2. Backend (Controller) – Built with Flask (Python), responsible for handling HTTP requests, business logic, and interaction with the database.
3. Database (Model) – Stores structured data about users, vendors, and products using SQLite (with provisions to migrate to PostgreSQL or MySQL).

All user requests are routed through the Flask application, which retrieves or modifies data from the database and returns rendered HTML templates for display.

TECHNOLOGIES USED

- **Languages**: Python, HTML, CSS, JavaScript
- **Backend Framework**: Flask (Python-based microframework)
- **Database**: SQLite (easily replaceable with Mongita)
- **Templating Engine**: Jinja2 (for dynamic HTML rendering)
- **Version Control**: Git and GitHub
- **IDE/Tools**: VS Code, Postman (for API testing), SQLite Browser

OVERVIEW OF SUBSYSTEMS OR MODULES

1. **User Interface Module**: Includes homepage, vendor dashboard, and product browsing pages built using HTML templates and styled with CSS.
2. **Product Management Module:** Handles CRUD operations for products, allowing vendors to add, update, and delete listings.
3. **Routing and Logic Module: Flask** routes define application behavior, link pages, and manage session data.
4. **Database Module:** Schema defines tables for products and (optionally) users/vendors. Simple queries are used to retrieve or update data.
5. **Planned Modules (not fully implemented):**

- Payment Processing: Integration with Stripe/PayPal APIs.
- Location Mapping: Google Maps API for displaying vendor locations.
- Recommendation System: AI-based suggestions based on user behavior and product sustainability metrics.

NOTABLE DESIGN DECISIONS AND TRADE-OFFS

- **Choice of Flask:** We opted for Flask due to its simplicity and flexibility for educational projects. While Django offers more built-in features, Flask allowed us to build and understand every part of the application more deeply.
- **Database Selection:** SQLite was chosen for simplicity and ease of setup during development, though the database schema is designed to scale with more robust solutions like PostgreSQL.
- **Static vs. Dynamic Content:** We used Jinja2 to enable server-side rendering of HTML, simplifying user interactions while avoiding the complexity of building a full SPA (Single Page Application) with frameworks like React.
- **Frontend Simplicity:** To ensure cross-platform compatibility and rapid development, we used vanilla HTML/CSS/JS rather than integrating heavier frameworks such as Bootstrap or Tailwind.
- **Security and Scalability:** Authentication, payment security, and user session handling are currently minimal, as the focus was on building a working prototype. These elements are planned for future expansion.

DEVELOPMENT PROCESS

TIMELINE OF MAJOR PHASES AND MILESTONES

Phase	Timeline	Description
Planning & Proposal	Week 1	Defined project scope, selected team roles, submitted project proposal.
Design & Wireframing	Week 2	Created UI mockups, ER diagrams, and finalized system architecture.
Backend Setup	Week 3	Initialized Flask app, built base routes, and designed initial database schema.
Frontend Integration	Weeks 4–5	Developed HTML templates, applied custom CSS styling, and connected frontend to Flask logic.
Feature Implementation	Weeks 5–6	Completed CRUD operations for product management and vendor dashboard.
Testing & Refinement	Week 7	Conducted unit tests, debugged issues, and optimized UI for responsiveness.
Final Integration	Week 8	Polished codebase, prepared README, packaged ZIP file, and completed team report.

DESCRIPTION OF TEAM WORKFLOW

The team followed an Agile-inspired workflow with weekly check-ins and informal sprints. Instead of rigid ceremonies, we adopted a lightweight version of Agile where team members discussed progress, shared blockers, and planned upcoming tasks during virtual meetings. Task lists and milestones were tracked collaboratively, and flexibility was maintained to adjust workloads as needed.

ROLES AND RESPONSIBILITIES WITHIN THE TEAM

- **Haripriya Animireddy**

Roles: Backend Developer, Database Architect

Contributions: Designed and implemented database schema, built core Flask routes, handled product CRUD functionality.

- **Nikhil Divi**

Roles: Frontend Developer, UI/UX Designer

Contributions: Developed HTML templates, styled components with CSS, worked on responsive layout, handled integration with backend data.

- **Jathin Chava**

Roles: Business Logic Developer, QA Tester

Contributions: Implemented input validation and backend logic, coordinated testing/debugging, ensured functionality matched requirements.

TOOLS USED FOR COLLABORATION AND SOURCE CONTROL

- **Version Control**: Git, GitHub (for code collaboration, commits, and version tracking)
- **Code Editor:** Visual Studio Code (with extensions for Python and HTML formatting)
- **Communication:** WhatsApp & Zoom (for team coordination and meetings)
- **Documentation:** Google Docs and PDF (for drafting the proposal, final report, and diagrams)
- b Draw.io (for ER diagrams), Figma (for interface wireframes)

IMPLEMENTATION DETAILS

KEY ALGORITHMS OR COMPONENTS

EcoMarket was designed with modular simplicity and clarity in mind. While the project doesn't implement complex algorithms, it includes several key components and logical flows:

- **Product CRUD Operations:** Implemented through Flask routes using HTTP methods (GET, POST). These operations allow vendors to create, edit, and delete product listings.
- **Data Validation and Error Handling**: Input fields are validated on both frontend and backend to prevent empty submissions and incorrect formats.
- **Templating with Jinja2:** All dynamic HTML content is rendered using Flask's Jinja2 templating engine, which helps maintain DRY (Don't Repeat Yourself) principles across pages.
- **Static and Template Separation**: All styling is offloaded to external CSS stored in the static/ folder, and all HTML files are neatly organized under templates/.

STRUCTURE OF THE CODEBASE OR REPOSITORY

The project directory follows a clear and conventional layout for Flask applications:

```
ecomarket_final/
├── app.py              # Main application file with routes and server setup
├── extensions.py       # Configuration or future extensions (e.g., DB setup)
├── requirements.txt    # Python package dependencies
├── static/             # Folder for CSS and static assets
│   └── css/
│       └── style.css
├── templates/          # HTML templates used for rendering pages
│   ├── index.html      # Main landing page
│   ├── edit.html       # Vendor product editing page
├── README.md           # Project documentation
```

- ER Diagram: Illustrates the structure of the database, including entities such as Product, Vendor, and relationships (e.g., one-to-many between vendors and products).
- Flowchart: Represents user interaction paths, such as:
  1. Home → View Products
  2. Vendor Login → Add Product → Edit/Delete Product
- High-Level System Diagram: Shows how the frontend, backend, and database interact:

[User Interface (HTML/CSS)]

↓

[Flask Backend (app.py)]

↓

[SQLite Database]

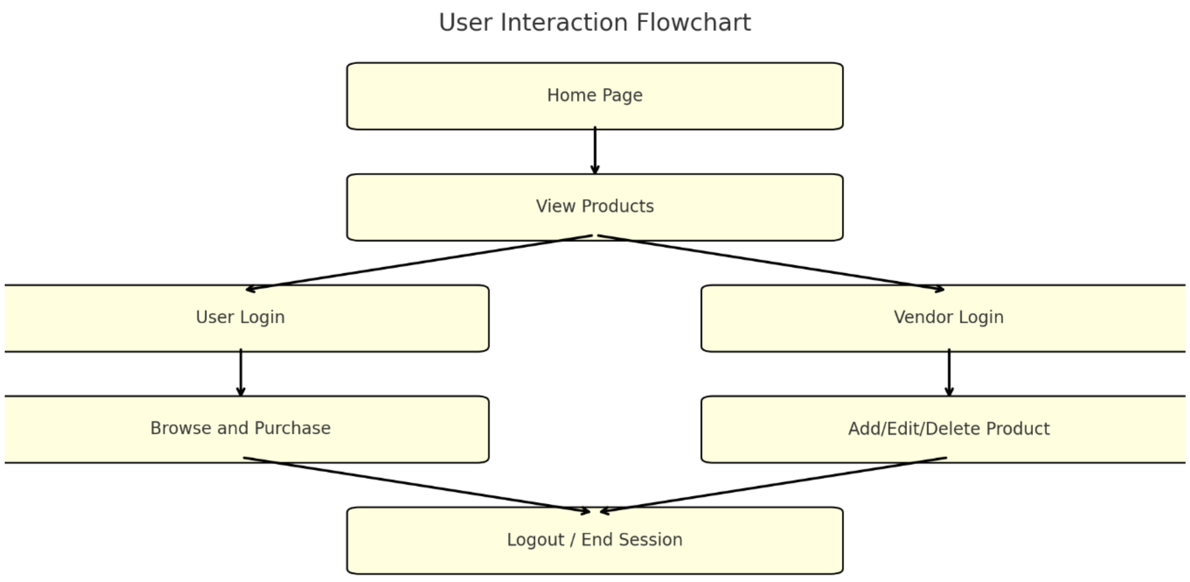


Fig. 1. Flowchart

TESTING AND VALIDATION

DESCRIPTION OF TESTING APPROACH

To ensure functionality, performance, and user satisfaction, we used a layered testing approach:

1. **Unit Testing**: Individual functions, such as route handlers and form validators, were manually tested during development. For example, product creation and form submissions were checked for edge cases (e.g., empty fields, invalid data types).
2. **Integration Testing**: Components were tested in sequence to confirm that routes, templates, and database interactions work together. For instance, after a product is added via a form, it should be reflected on the homepage without errors.

3. **System Testing**: End-to-end walkthroughs of the application were conducted to verify overall stability. This included testing navigation, CRUD operations, and UI rendering on multiple devices (laptop, mobile emulator).
4. **User Testing**: Informal feedback was collected from classmates and instructors during development demos. This helped identify UI issues such as button placement, form usability, and clarity of navigation.

EXAMPLES OF TEST CASES AND RESULTS

Feature	Test Case	Expected Result	Outcome
Add Product	Vendor submits product with valid data	Product appears in listings	✅ Pass
Edit Product	Vendor modifies existing product info	Changes reflected correctly	✅ Pass
Delete Product	Vendor deletes a product	Product no longer visible on homepage	✅ Pass
Submit Empty Form	Vendor submits form with empty name or price	Form is rejected with error message	✅ Pass
Navigate Pages	User clicks links across site (e.g., home, dashboard)	Pages load without error	✅ Pass
Mobile Responsiveness	Open site on different screen sizes	Layout adjusts to fit screen properly	⚠ Partial
SQL Injection Resistance	Malicious input in form fields	Input is sanitized; no backend crash	✅ Pass

VALIDATION AGAINST REQUIREMENTS

Requirement	Validation Outcome
Users can browse eco-friendly products	Verified via homepage listing functionality ✅
Vendors can manage product listings	CRUD operations tested and passed ✅
User interface is clean and responsive	UI tested on multiple devices, minor improvements pending ⚠
Data is stored securely and correctly	SQLite database confirmed operational ✅
System performs without major bugs or crashes	Full system tested end-to-end ✅
Optional features (payment, maps, recommendations)	Not fully implemented, marked for future work ❌

STAKEHOLDER FEEDBACK

- Instructor feedback praised the clean layout and logical route handling.

- Peer reviewers noted the user-friendly interface but suggested improvements in mobile responsiveness and form aesthetics.
- Team consensus indicated satisfaction with the project's stability and modular structure.

RESULTS AND EVALUATION

WAS THE PROJECT SUCCESSFUL?

Yes, the EcoMarket project met its primary goals and can be considered a successful implementation of a sustainable e-commerce prototype. The team delivered a functional web application where users can browse eco-friendly products and vendors can manage their listings through a dedicated dashboard. Although some advanced features (e.g., payment gateway integration and smart recommendations) were not completed due to time constraints, the foundational system is stable, user-friendly, and extensible for future development.

METRICS OR CRITERIA USED TO ASSESS SUCCESS

To evaluate project success, we established a set of technical and user-facing benchmarks:

Metric / Criterion	Status	Notes
Product listing functionality	✅ Completed	Full CRUD operations for vendors
User interface clarity and navigation	✅ Completed	Tested by peers and refined based on feedback
System uptime and stability	✅ Stable	No major bugs or crashes during testing
Database integrity and query performance	✅ Functional	Reliable storage and retrieval of product data
Frontend responsiveness on common screen sizes	⚠ Partially	Works on most screens, but small screens need refinement
Feature completion (relative to project proposal)	⚠ Partial	Base features complete; optional features pending
Stakeholder satisfaction (team + instructor)	✅ Positive	Well received in demos and peer review

LESSONS LEARNED AND RETROSPECTIVE

1. **Backend Modularity is Crucial**: Using Flask helped us understand how to separate concerns across templates, logic, and database access. This made debugging and testing much easier.
2. **Time Management Matters**: Early-stage planning helped avoid last-minute issues, but we underestimated how long advanced features (e.g., payment systems) would take to implement and test securely.

3. **UI/UX Is More Than Looks**: Usability is as important as design. Even small UI tweaks (like spacing or error messages) significantly improved the experience.
4. **Version Control Collaboration**: Using Git consistently improved our workflow and allowed parallel development. We learned the importance of clear commit messages and resolving merge conflicts quickly.
5. **Responsiveness Takes Iteration**: Designing for multiple screen sizes proved more challenging than expected. We learned to test early and often across devices.

Overall, the project gave us hands-on experience with full-stack development, practical teamwork, and software design challenges. It also deepened our understanding of how to build meaningful digital solutions that align with real-world values like sustainability.

FUTURE WORK AND RECOMMENDATIONS

REMAINING ISSUES OR KNOWN BUGS

While the core features of EcoMarket function correctly, a few issues and limitations were identified during testing:

- Mobile Responsiveness: Some layout elements (like product cards and form inputs) do not scale properly on small mobile screens. CSS media queries and flexbox/grid adjustments are needed.
- Form Validation Feedback: Although input validation is handled, users receive limited feedback (e.g., vague or missing error messages).
- No User Authentication: The current system does not implement secure login or session management, which restricts functionality for differentiated user roles (e.g., shoppers vs. vendors).
- Static Product Categories: Product filtering is not yet dynamic; all products are shown together without category-based sorting or tags.
- Minimal Security: SQL injection protection and form sanitization are basic and should be hardened with production-level security standards.

IDEAS FOR EXTENSIONS OR FUTURE VERSIONS

To enhance the functionality and impact of EcoMarket, we propose the following extensions:

1. **User Authentication System**: Implement secure login/logout functionality using Flask-Login or OAuth to support user profiles and role-based access (e.g., vendors vs. buyers).
2. **Payment Integration**: Integrate with Stripe or PayPal to allow real-world purchases. This would involve secure payment APIs and transaction tracking.
3. **Product Recommendations**: Use machine learning or rule-based filtering to suggest products based on user behavior, previous purchases, or sustainability scores.
4. **Google Maps API Integration**: Display vendor locations on a map, helping users find local sellers and reducing carbon impact from shipping.
5. **Review and Rating System**: Allow users to rate and review products or vendors to increase trust and transparency in the marketplace.

6. **Admin Dashboard**: Build an administrative panel for managing user accounts, vendor approvals, and platform moderation.
7. **Accessibility Improvements**: Make the platform WCAG-compliant by adding keyboard navigation, ARIA labels, and better color contrast.
8. **Deployment on Cloud Platforms**: Host the platform on services like Heroku, AWS, or Render for public access and better scalability.

These enhancements would significantly increase the platform's utility, security, and reach, aligning it more closely with real-world use cases for sustainable commerce.

USER MANUAL

INSTRUCTIONS FOR RUNNING OR INSTALLING THE APPLICATION

To run the EcoMarket platform locally, follow these steps:

1. **Download the Project**
    - clone the repository from GitHub.
2. **Navigate to the Project Directory**
    ```
    cd ecomarket_final
    ```
3. **Set Up a Virtual Environment (Recommended)**
    ```
    python -m venv venv
    source venv/bin/activate      # On Windows: venv\Scripts\activate
    ```
4. **Install Required Packages**
    ```
    pip install -r requirements.txt
    ```
5. **Run the Application**
    ```
    python app.py
    ```
6. **Access the Platform**
    - Open a web browser and go to: http://localhost:5000

LOCATION OF PRODUCTION DEPLOYMENT

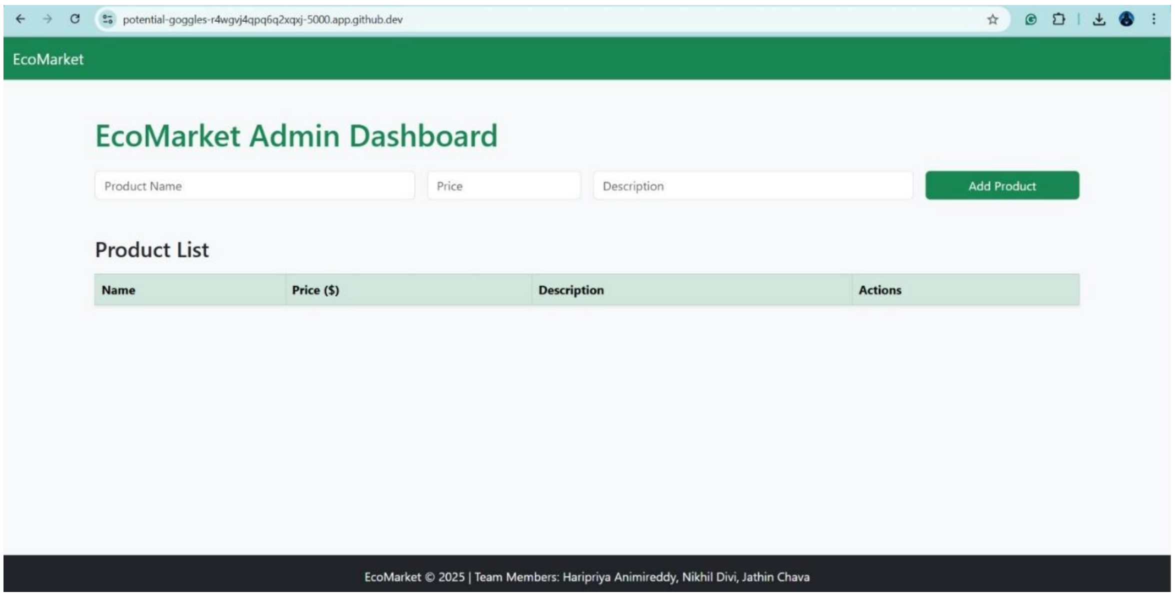
At this stage, the application has not been deployed to a public server. Deployment is planned for the future using platforms such as:

- Render
- Heroku
- AWS EC2 or Lightsail

Once deployed, the production URL will be listed here.

SCREENSHOTS ILLUSTRATING KEY FEATURES

1. **Homepage View**
    - Displays available eco-friendly products with title, description, and price.
    - Clean navigation bar and product cards.



EcoMarket Admin Dashboard

Product Name | Price | Description

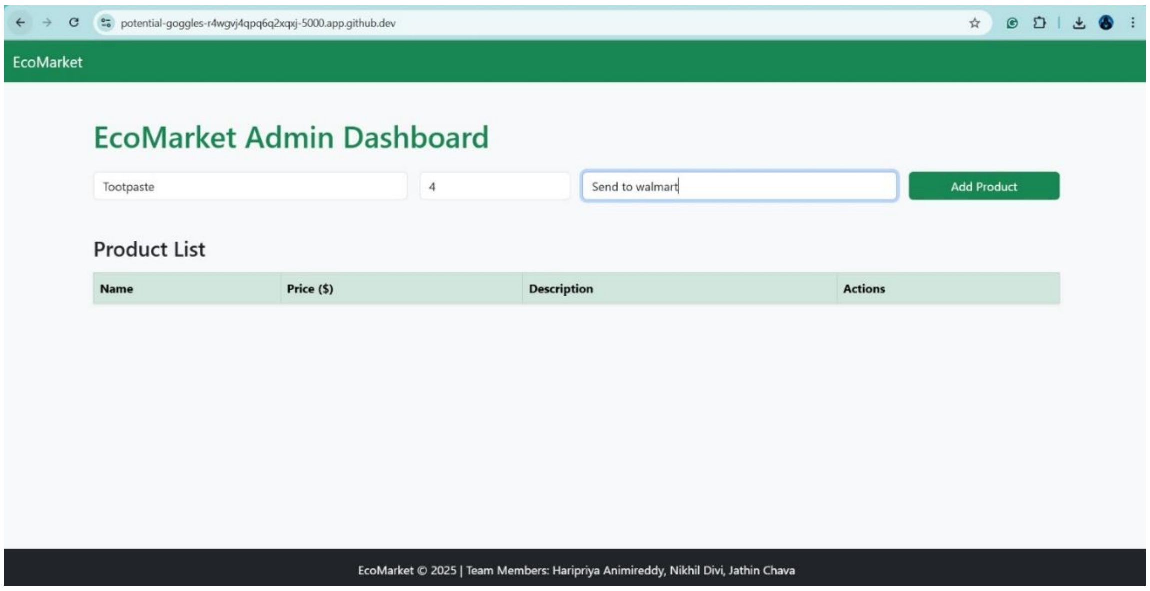
Product List

Name	Price ($)	Description	Actions

EcoMarket © 2025 | Team Members: Haripriya Animireddy, Nikhil Divi, Jathin Chava

Fig. 2. Homepage

2. **Vendor Dashboard**
    - Form for adding new products.
    - Buttons for editing or deleting existing listings.



EcoMarket Admin Dashboard

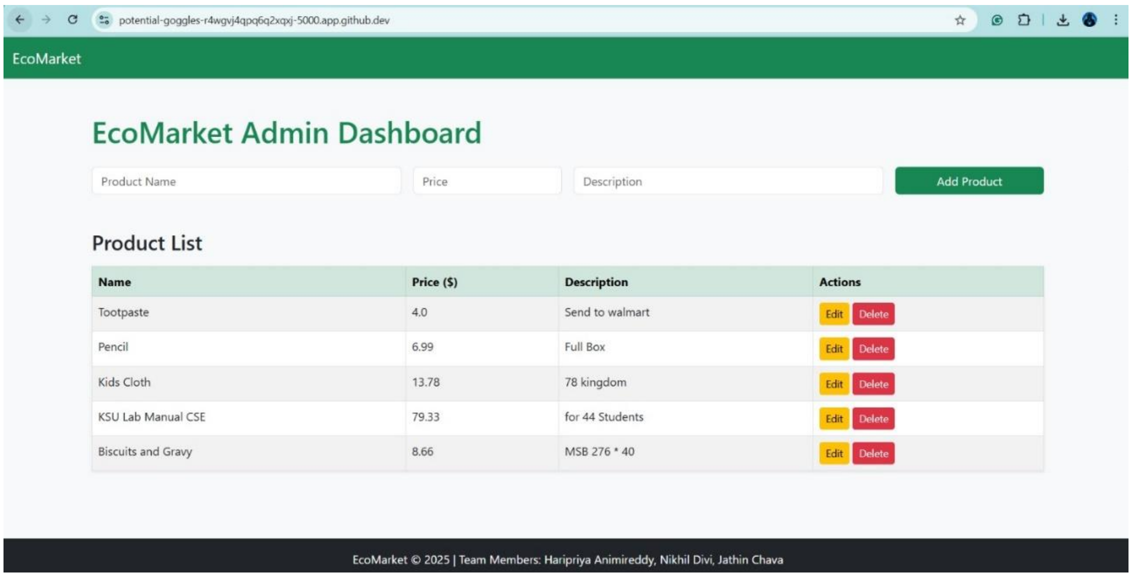
Tootpaste | 4 | Send to walmart

Product List

Name	Price ($)	Description	Actions

EcoMarket © 2025 | Team Members: Haripriya Animireddy, Nikhil Divi, Jathin Chava

Fig. 3. Add Product


potential-goggles-r4wgvj4qpq6q2xqxj-5000.app.github.dev

EcoMarket

EcoMarket Admin Dashboard

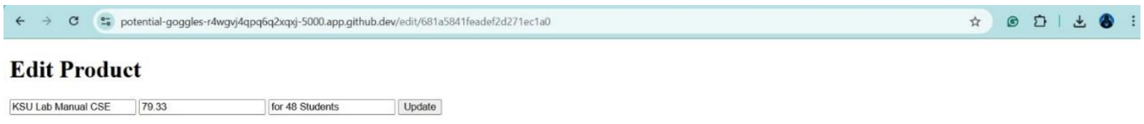
Product List

Name	Price ($)	Description	Actions
Tootpaste	4.0	Send to walmart	Edit Delete
Pencil	6.99	Full Box	Edit Delete
Kids Cloth	13.78	78 kingdom	Edit Delete
KSU Lab Manual CSE	79.33	for 44 Students	Edit Delete
Biscuits and Gravy	8.66	MSB 276 * 40	Edit Delete

EcoMarket © 2025 | Team Members: Haripriya Animireddy, Nikhil Divi, Jathin Chava

Fig. 4. List of the products

3. **Product Edit Page**
    - o Editable fields populated with current product data.
    - o Save and cancel options.

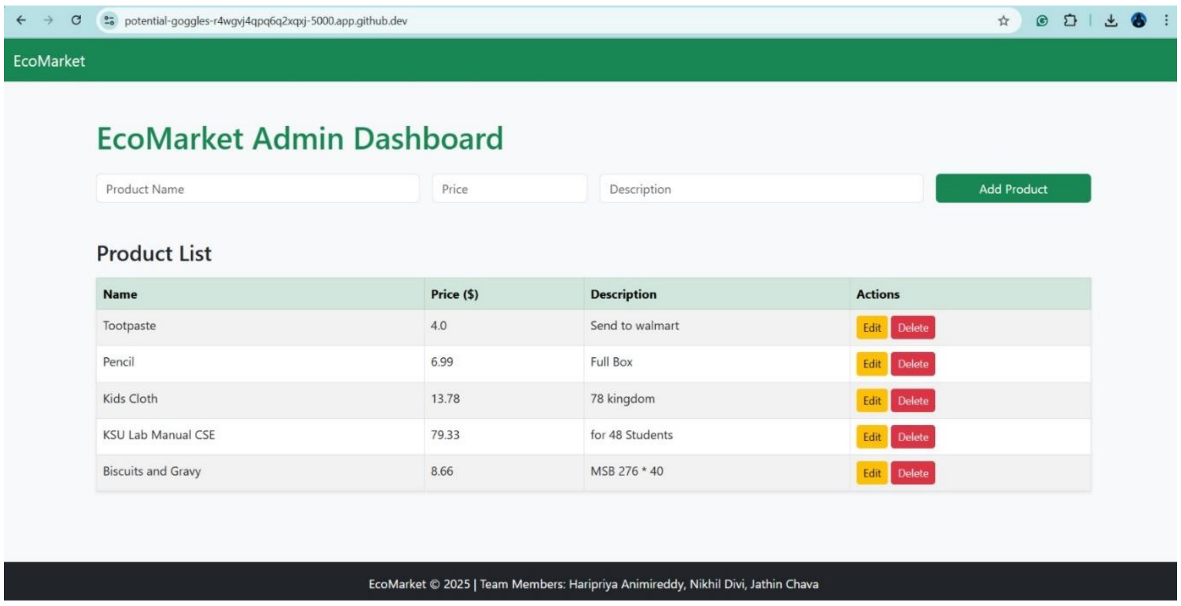

potential-goggles-r4wgvj4qpq6q2xqxj-5000.app.github.dev/edit/681a5841feadef2d271ec1a0

Edit Product

KSU Lab Manual CSE | 79.33 | for 48 Students

Fig. 5. Edit or Update

We have updated the KSU Lab Manual CSE Description:


potential-goggles-r4wgvj4qpq6q2xqxj-5000.app.github.dev

EcoMarket

EcoMarket Admin Dashboard

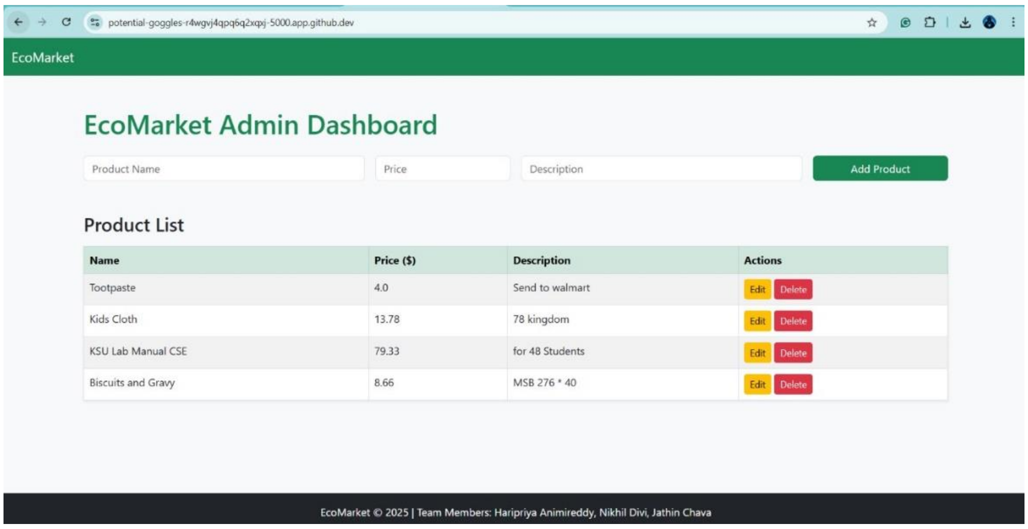
Product List

Name	Price ($)	Description	Actions
Tootpaste	4.0	Send to walmart	Edit Delete
Pencil	6.99	Full Box	Edit Delete
Kids Cloth	13.78	78 kingdom	Edit Delete
KSU Lab Manual CSE	79.33	for 48 Students	Edit Delete
Biscuits and Gravy	8.66	MSB 276 * 40	Edit Delete

EcoMarket © 2025 | Team Members: Haripriya Animireddy, Nikhil Divi, Jathin Chava

Fig. 6. List after Update

We have deleted the Pencil Product completely:



potential-goggles-r4wgvj4qpq6q2xqxj-5000.app.github.dev

EcoMarket

EcoMarket Admin Dashboard

Product Name | Price | Description | Add Product

Product List

Name	Price ($)	Description	Actions
Tootpaste	4.0	Send to walmart	Edit Delete
Kids Cloth	13.78	78 kingdom	Edit Delete
KSU Lab Manual CSE	79.33	for 48 Students	Edit Delete
Biscuits and Gravy	8.66	MSB 276 * 40	Edit Delete

EcoMarket © 2025 | Team Members: Haripriya Animireddy, Nikhil Divi, Jathin Chava

Fig. 7. Delete Operation

APPENDICES

A. URLS FOR DEPLOYED PRODUCTS OR CODE REPOSITORIES

- **Project Repository**:

  https://github.com/HaripriyaAniminireddy/Eco_market

B. SAMPLE INPUT/OUTPUT DATA

Sample Product Entry (Input):

```
{
  "product_name": "Tootpaste",
  "description": "Send to walmart.",
  "price": 4.0,
  "vendor": "GreenSmile Co."
}
```

Expected Output Display on Homepage:

Tootpaste

Send to Walmart.

Price: $4.0

[Edit] [Delete]

ADDITIONAL DIAGRAMS OR TECHNICAL NOTES

1. **ER Diagram**: One-to-many relationship between Vendors and Products.
2. **Flowchart**: Previously included illustrates how users and vendors interact with the platform from homepage to logout.
3. **System Architecture Notes**:

- Designed using a simple MVC (Model-View-Controller) pattern via Flask.
- Static resources (CSS, images) are separated from HTML templates for scalability.
- Code is modular to support future extensions such as authentication and payment integration.

REFERENCES

1. Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.
   – Used as a guide for Flask routing, templating, and project structure.
2. W3Schools. (n.d.). HTML, CSS, JavaScript Tutorials. Retrieved from https://www.w3schools.com
   – Used for frontend development references and CSS layout techniques.
3. Mozilla Developer Network (MDN). (n.d.). Web Docs: HTML, CSS, and JavaScript. Retrieved from https://developer.mozilla.org
   – Used for standard web design patterns, semantic HTML, and JavaScript DOM handling.
4. Stripe. (n.d.). Stripe Documentation. Retrieved from https://stripe.com/docs
   – Used to plan payment gateway integration for future extensions.
5. Stack Overflow. (n.d.). Community discussions and solutions. Retrieved from https://stackoverflow.com
   – Used for troubleshooting bugs, Flask form handling, and SQL integration tips.
6. Google Developers. (n.d.). Google Maps Platform Documentation. Retrieved from https://developers.google.com/maps/documentation
   – Referenced for proposed vendor geolocation feature.
7. United Nations. (2020). Sustainable Development Goals Report 2020. Retrieved from https://unstats.un.org/sdgs/report/2020
   – Used to frame the importance of supporting sustainable consumer habits through digital tools.