**Project Report**

|  |  |
| --- | --- |
| **PROJECT TITLE** | ShopEZ: One-Stop Shop for Online Purchases |
| **TEAM MEMBERS** | Bidisha Biswas (Team Lead) – Backend Developer  Bhargavee Singh – Frontend Developer  Diya Raj – Database & Deployment  Namrata Bhutani - Tester |
| **TEAM ID** | SWTID1742751842 |

1. **INTRODUCTION**
   1. **Project Overview**

**ShopEZ** is a full-stack, single-page e-commerce web application designed to deliver a seamless and engaging online shopping experience for users while equipping administrators with a powerful backend to manage the store efficiently. Developed using the **MERN stack (MongoDB, Express.js, React, Node.js)**, the platform balances dynamic frontend interactions with robust backend functionality.

On the **user side**, ShopEZ allows customers to browse products, view details, add items to their cart or wishlist, register/login, and place orders securely. Token-based authentication ensures safe and persistent sessions, while features like product variation, password reset, and email notifications enhance user convenience.

The **admin panel** empowers store managers to perform CRUD operations on products, view and manage user orders, and oversee platform activity. Data consistency and security are maintained through structured APIs and database schemas, with token-based middleware ensuring protected access.

This project was built as a collaborative effort, simulating a real-world development environment where frontend, backend, database, deployment, and testing roles were divided among team members. With a responsive UI, secure authentication, and clean architecture, ShopEZ serves as a modern template for scalable e-commerce solutions.

* 1. **Purpose**

The purpose of this project is to design and develop a fully functional e-commerce website that provides users with a seamless online shopping experience. The platform aims to bridge the gap between consumers and sellers by offering a convenient, secure, and user-friendly interface for browsing, selecting, and purchasing products. It is built using the MERN (MongoDB, Express.js, React, Node.js) stack to ensure scalability, real-time performance, and maintainability.

This project is intended to streamline the entire shopping process—from product discovery and cart management to order placement and payment—while ensuring robust user authentication, responsive design, and efficient backend management. The goal is to replicate and enhance the essential features of leading e-commerce platforms, making it adaptable for various product categories and business models.

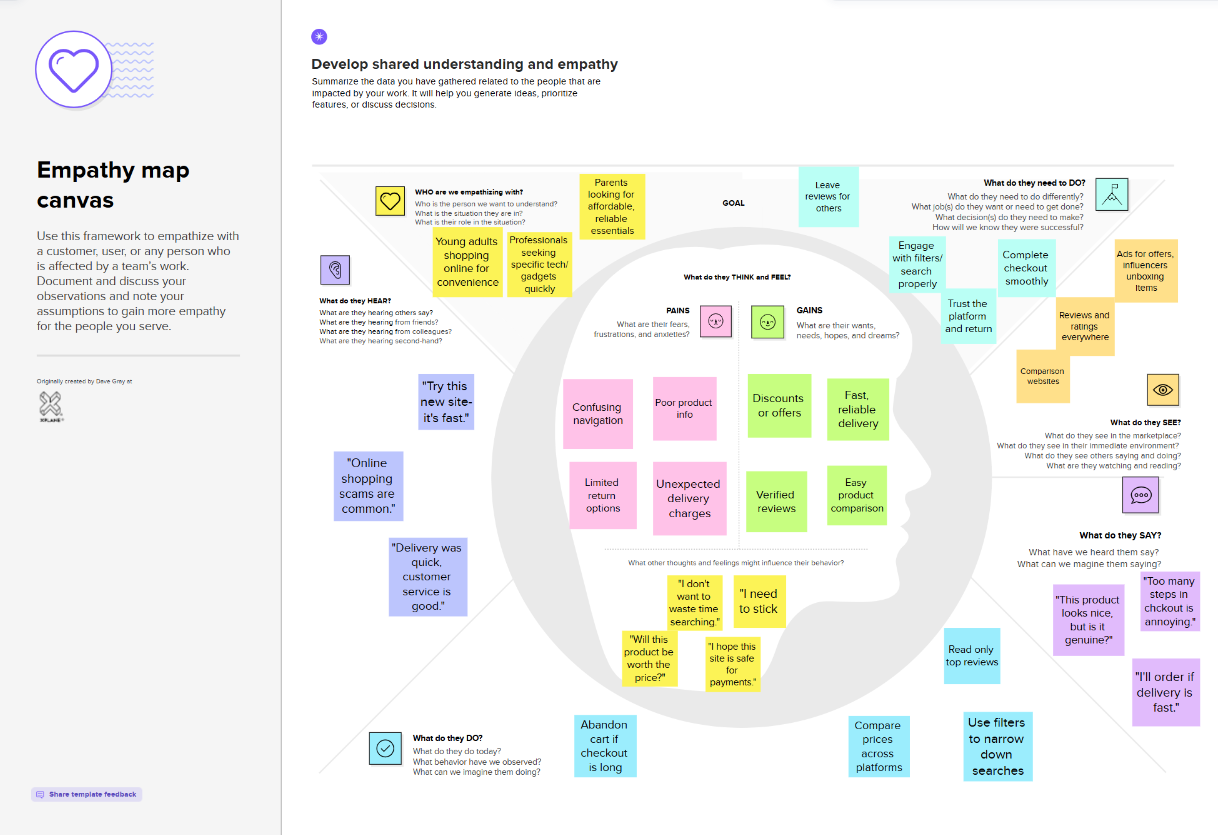
1. **IDEATION PHASE**
   1. **Problem Statement**

In today’s fast-paced digital world, consumers increasingly prefer the convenience of online shopping. However, many existing e-commerce platforms suffer from issues such as cluttered interfaces, slow performance, lack of personalization, limited scalability, and poor user experience on mobile devices. Small and medium-scale businesses often struggle to find a cost-effective, customizable, and easy-to-manage solution to bring their products online.

There is a growing need for a robust and user-friendly e-commerce platform that provides seamless shopping experiences for users while being simple for administrators to manage inventory, orders, and customer data. The system must also ensure secure user authentication, fast performance even with limited resources, and an intuitive interface across devices.

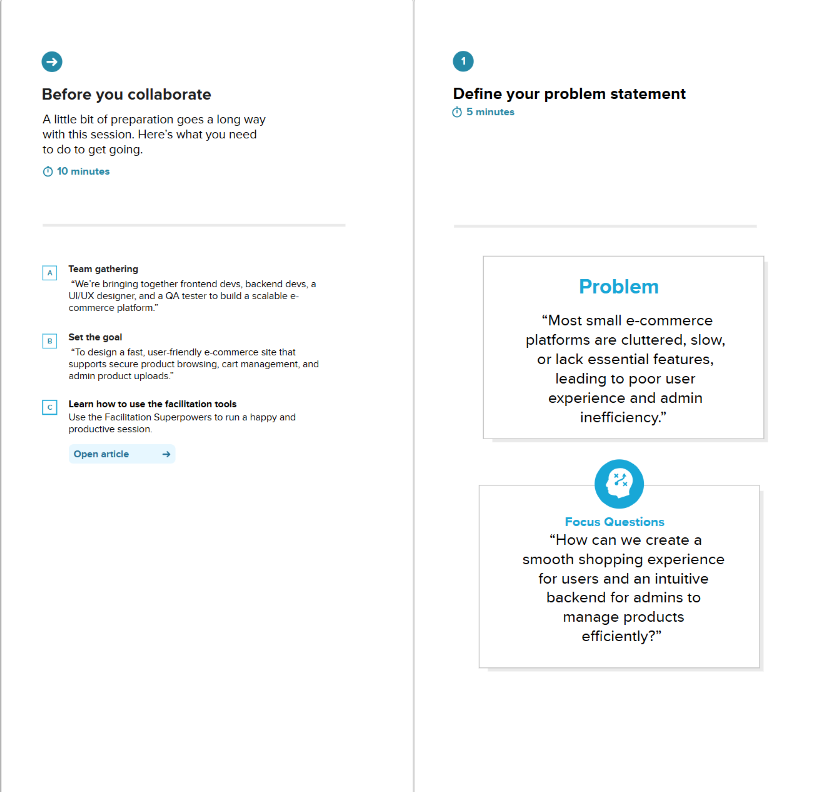
This project aims to address these challenges by developing a scalable, responsive, and secure e-commerce website using the MERN stack. The platform will support essential features such as user authentication, product browsing, cart management, and admin controls—delivering both usability and efficiency for end-users and administrators alike.

* 1. **Empathy Map Canvas**

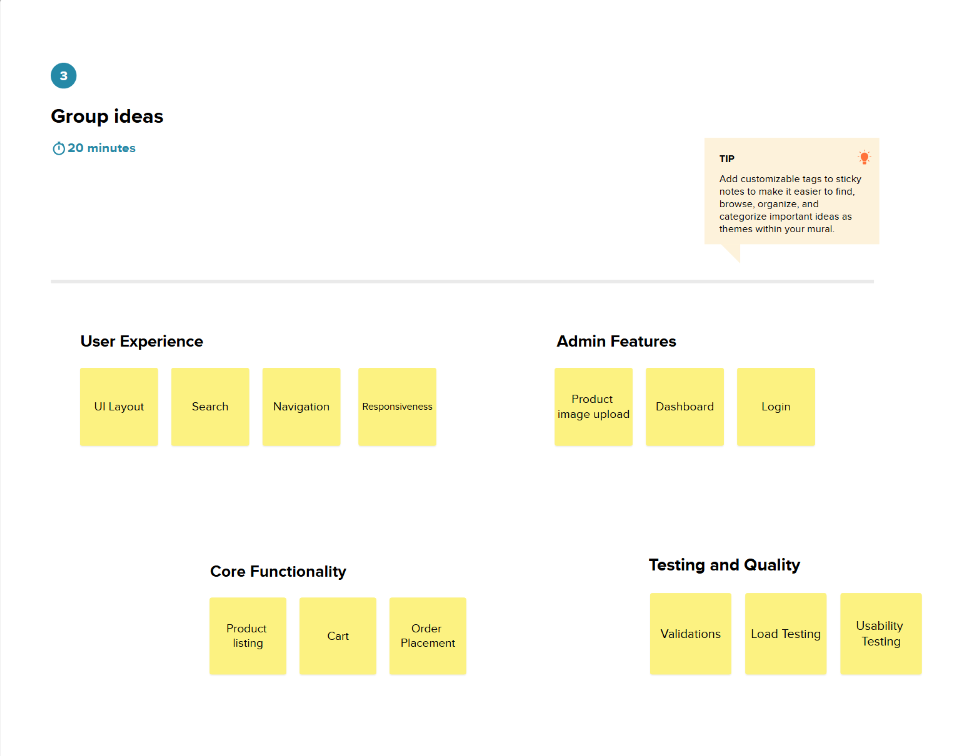
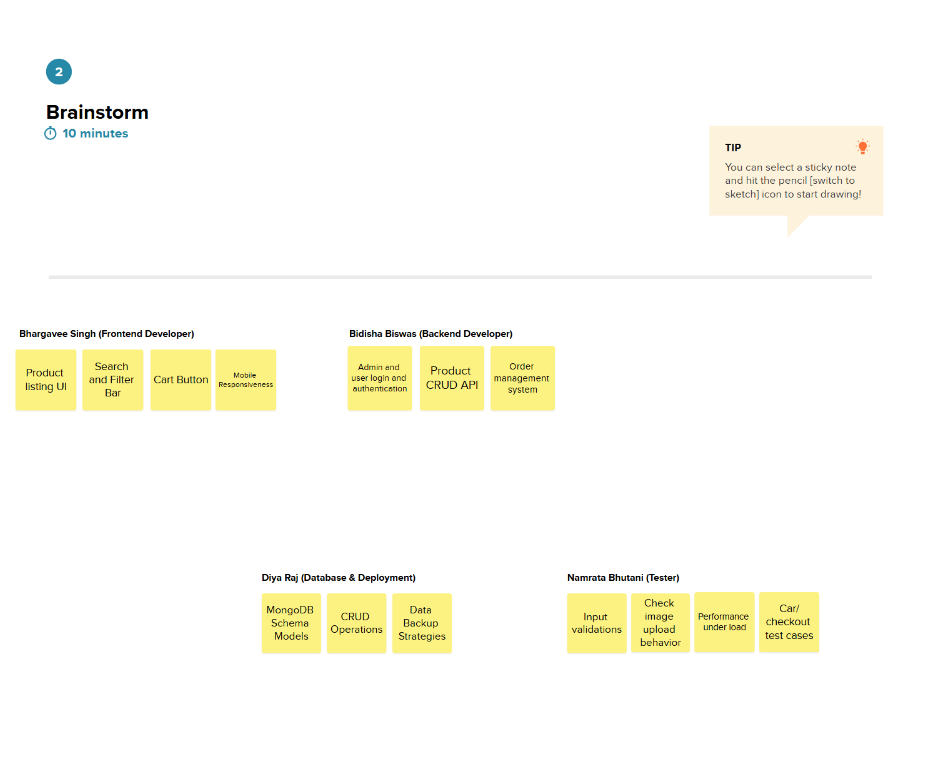


* 1. **Brainstorming**

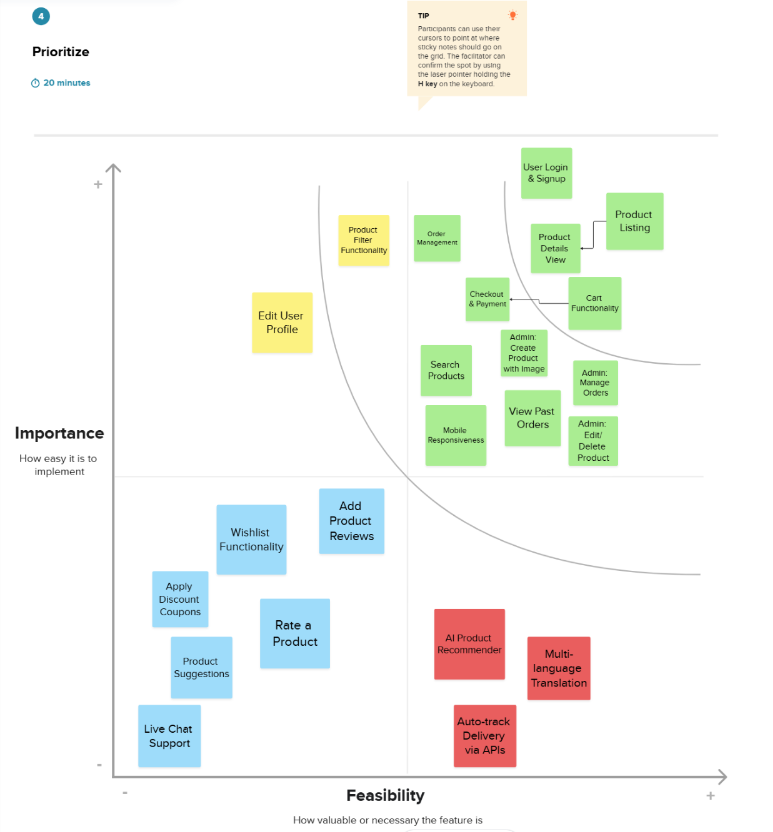
**Step-1: Team Gathering, Collaboration and Select the Problem Statement**



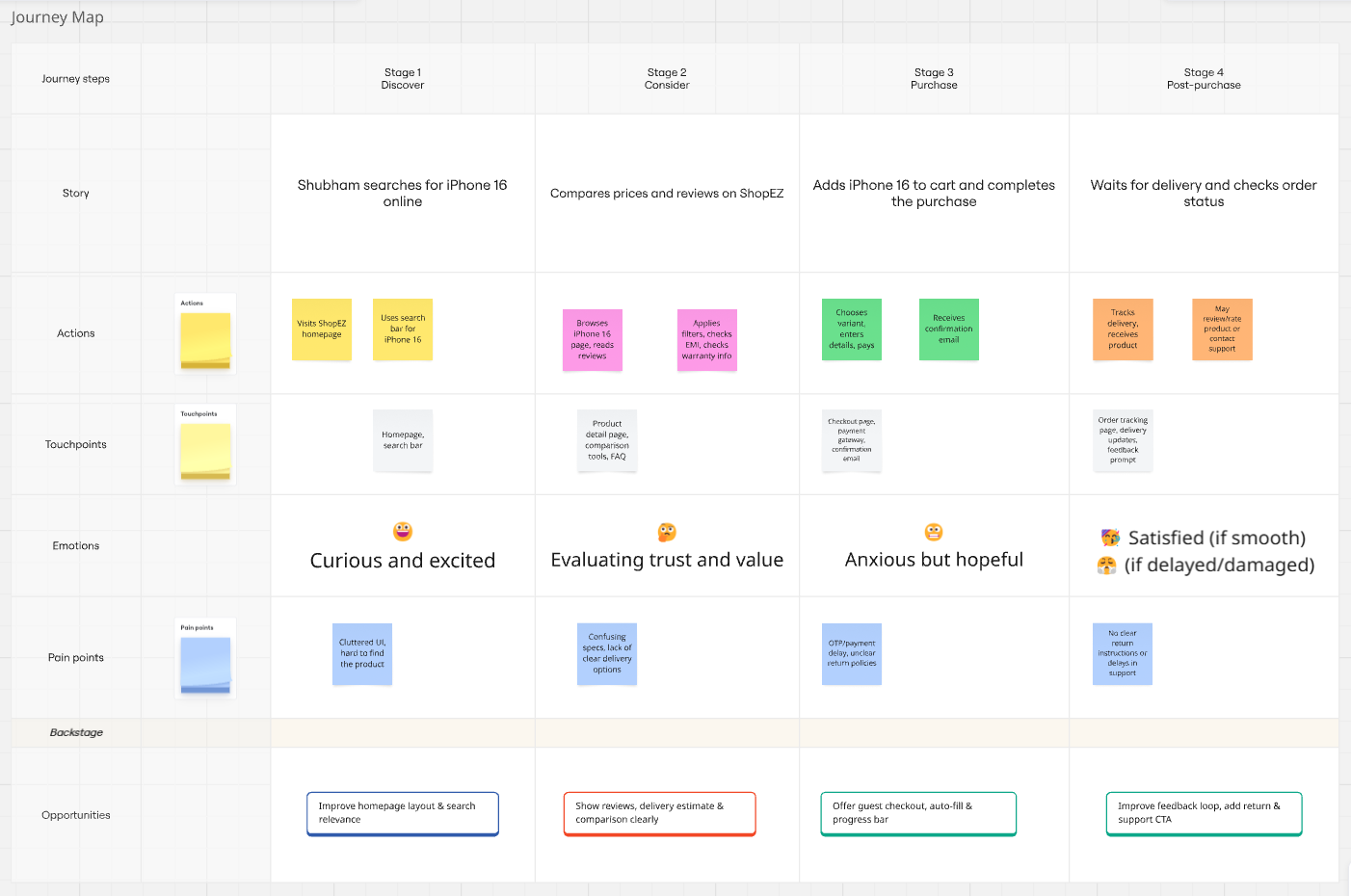
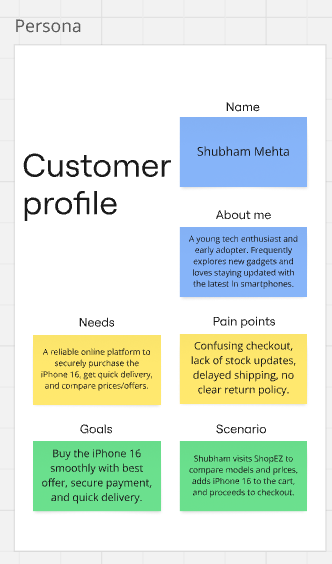
**Step-2: Brainstorm, Idea Listing and Grouping**

****

**Step-3: Idea Prioritization**

****

1. **REQUIREMENT ANALYSIS**
   1. **Customer Journey Map**



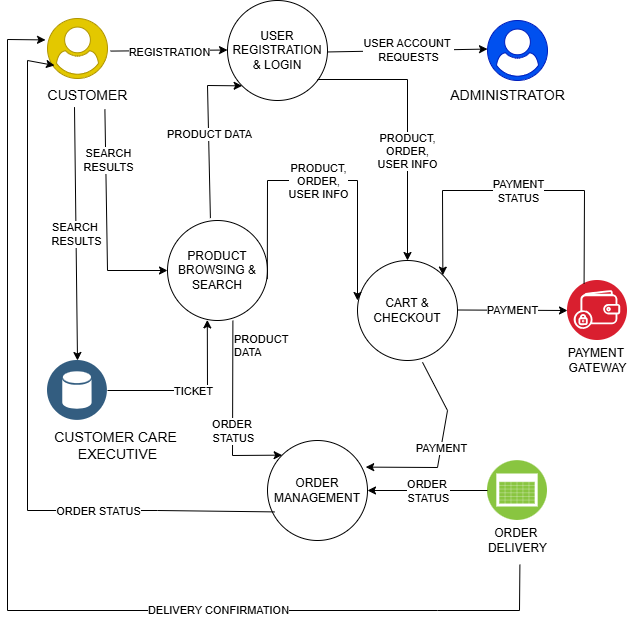
* 1. **Solution Requirements**
     1. **Functional Requirements:**

|  |  |  |
| --- | --- | --- |
| **FR.No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| **FR-1** | User Registration | - Registration through Form  - Registration through Gmail  - Registration through LinkedIn |
| **FR-2** | User Confirmation | - Confirmation via Email  - Confirmation via OTP |
| **FR-3** | Cart & Checkout | - View product categories  - Search for products  - Filter and sort products |
| **FR-4** | Payment | - Multiple payment options (UPI, Card, Net Banking)  - Payment confirmation handling |
| **FR-5** | Order Management | - View order history  - Track current orders  - Cancel order |
| **FR-6** | Customer Support | - Raise support ticket  - Chat or message executive  - View support history |
| **FR-7** | Customer Support | - Raise support ticket  - Chat or message executive  - View support history |
| **FR-8** | Admin Panel | - View user list  - Manage product listings  - Manage orders and delivery status |
| **FR-9** | Delivery Integration | - Generate delivery requests  - Update delivery status  - Notify customer upon delivery |

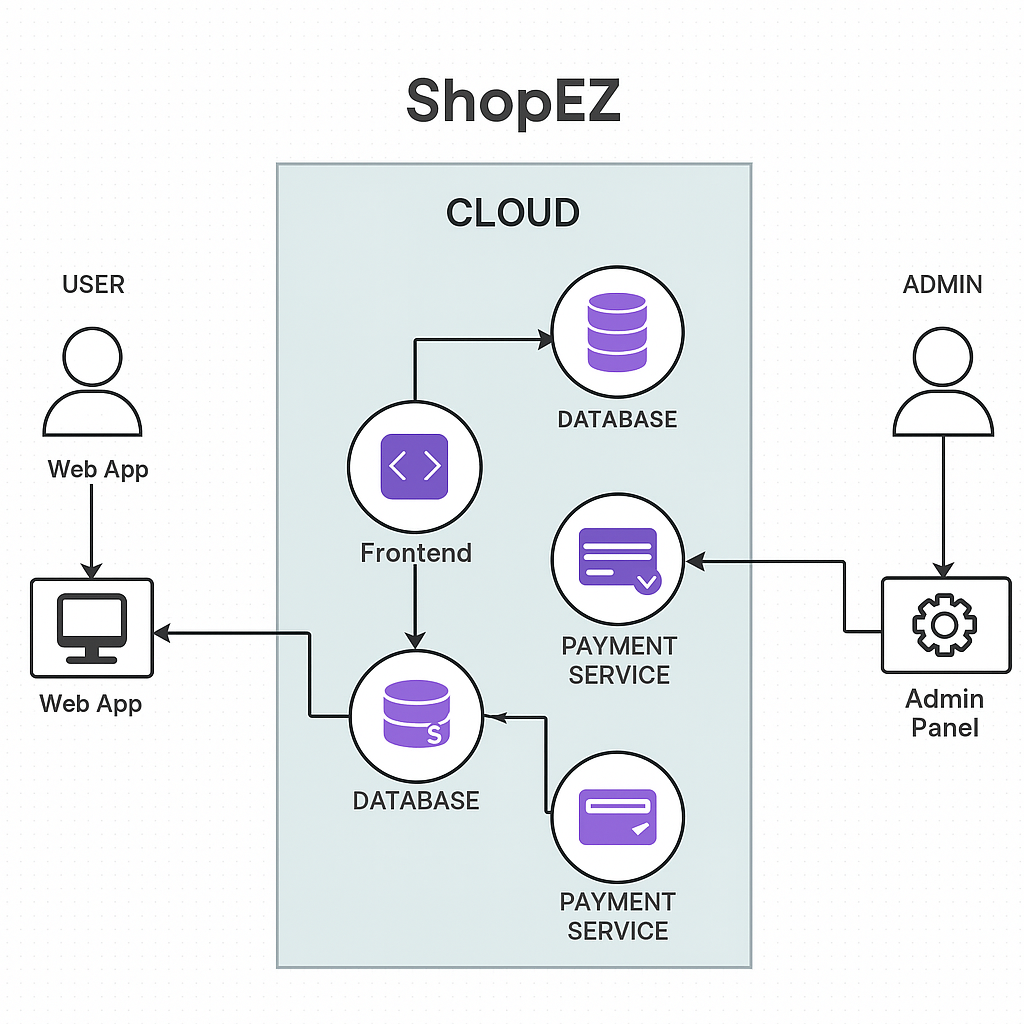
* + 1. **Non-Functional Requirements:**

|  |  |  |
| --- | --- | --- |
| **NFR.No.** | **Non-Functional Requirement (Epic)** | **Description** |
| **NFR-1** | Usability | The UI must be intuitive, responsive, and user-friendly for both mobile and desktop users. |
| **NFR-2** | Scalability | User data must be encrypted; authentication and authorization must be implemented using secure protocols (e.g., HTTPS, OAuth). |
| **NFR-3** | Reliability | The system should ensure consistent performance with a < 1% failure rate and handle failures gracefully. |
| **NFR-4** | Performance | The website must load within 3 seconds and support 50+ concurrent users without performance degradation. |
| **NFR-5** | Availability | The system should have 99.9% uptime and auto-recovery from minor crashes. |
| **NFR-6** | Scalability | The platform should support scaling to accommodate growing users and product inventory. Cloud-based deployment is preferred. |

* 1. **Data Flow Diagram**



* 1. **Technology Stack**
     1. **Technical Architecture**



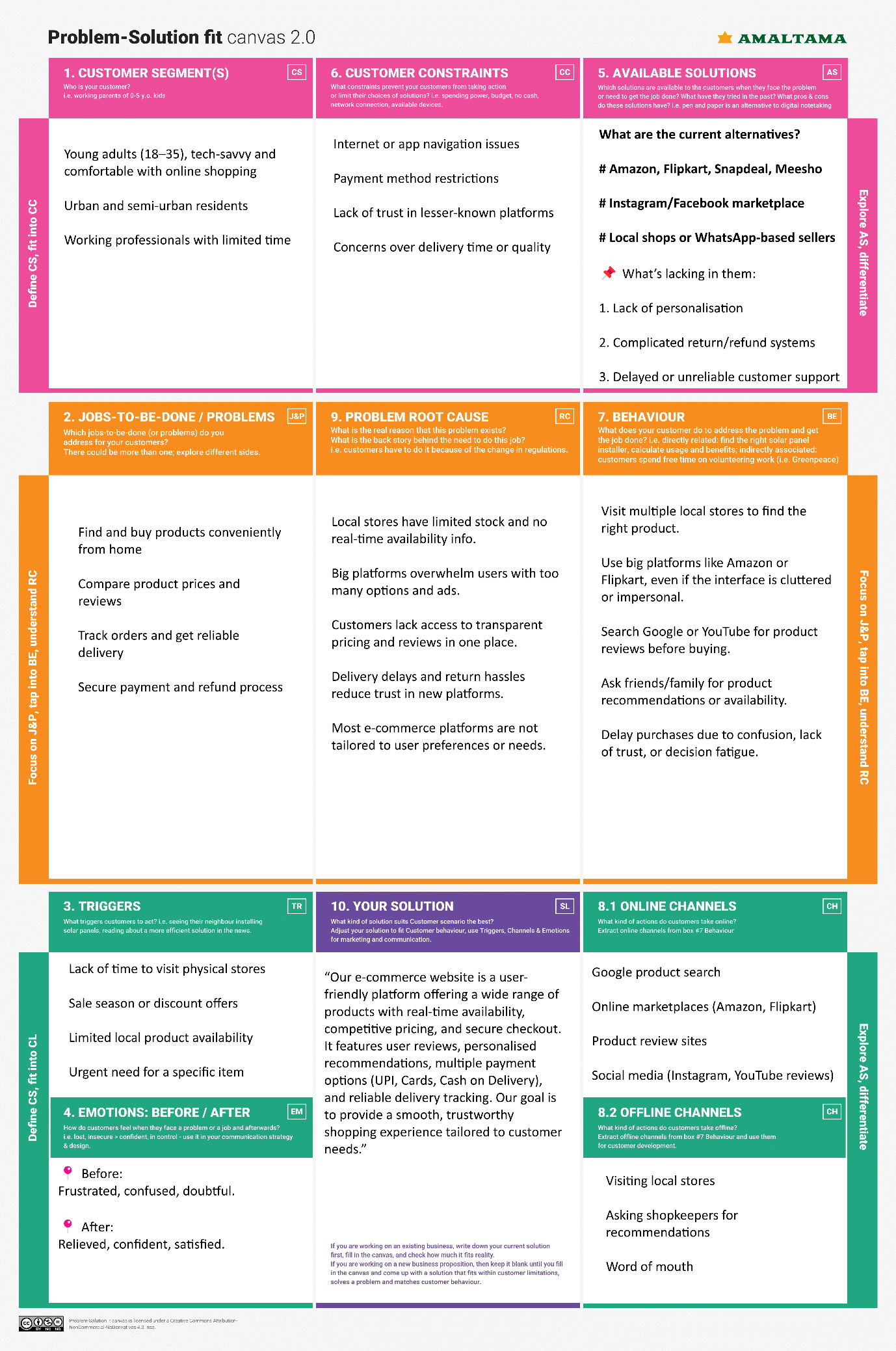
* + 1. **Components and Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Component** | **Description** | **Technology** |
| **1** | User Interface | Web UI where users browse products, register/login, checkout | HTML, CSS, JavaScript, React.js |
| **2** | Application Logic-1 | Handles authentication, product search, add to cart, order logic | Node.js, Express.js |
| 3 | Application Logic-2 | Session management and cart logic | Express-session, JWT |
| **4** | Database | Stores users, products, orders, and cart data | MongoDB |
| **5** | File Storage | Storing product images | Local filesystem (uploads folder) |
| **6** | External API-1 | Payment gateway for order checkout | Razorpay / Paytm API |
| **7** | Infrastructure | Hosting platform | Render / Vercel for frontend, Railway for backend |

* + 1. **Application Characteristics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Characteristics** | **Description** | **Technology** |
| **1** | Open-Source Frameworks | Frontend & backend built using open-source tech | React.js, Node.js, Express.js, MongoDB |
| **2** | Security Implementations | Password hashing, JWT-based authentication, HTTPS | bcrypt.js, JWT, Helmet.js, HTTPS |
| **3** | Scalable Architecture | Modular code with scalable database and REST APIs | 3-tier architecture with RESTful services |
| **4** | Availability | Deployment on reliable cloud platforms with downtime minimization | Railway, Vercel, MongoDB Atlas |
| **5** | Performance | Fast-loading frontend with optimized queries, CDN for static assets | Lazy loading, MongoDB indexing, Cloudflare CDN |

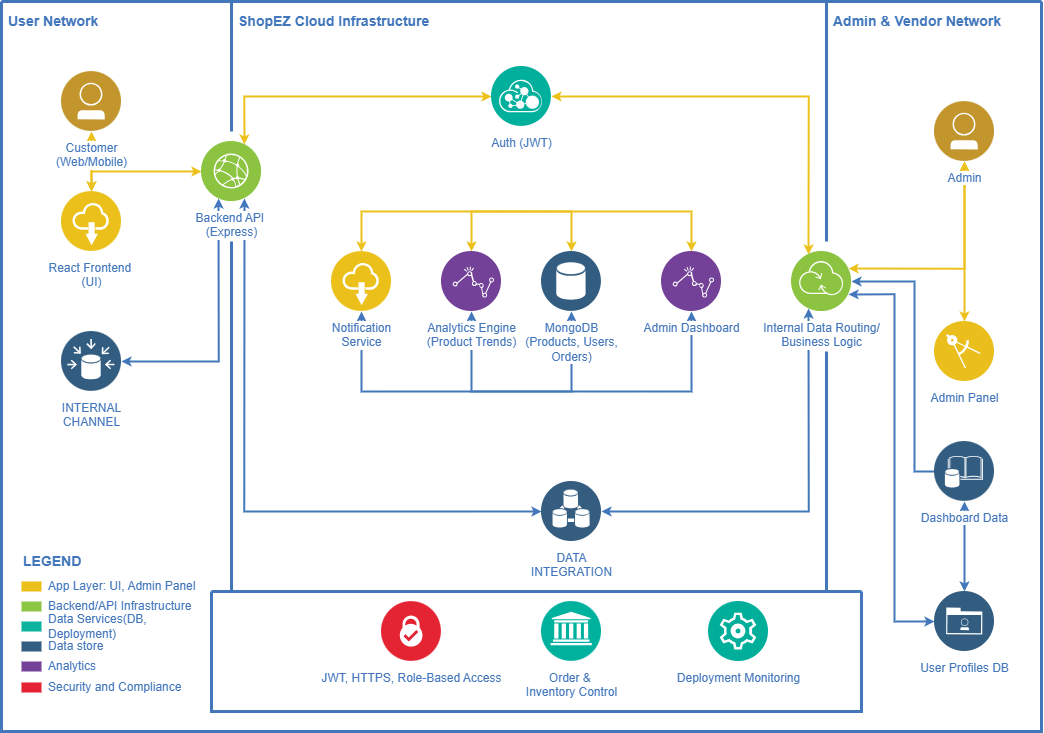
1. **PROJECT DESIGN**
   1. **Problem-Solution Fit**



* 1. **Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| **01** | Problem Statement (Problem to be solved) | Customers face difficulty finding affordable, quality products in one place with a smooth shopping experience. Major platforms are often cluttered, impersonal, and lack local personalization. Small businesses also struggle to go online and compete due to high fees and complexity. |
| **02** | Idea/Solution Description | **ShopEZ** is a user-friendly e-commerce platform offering a clean, intuitive interface for customers to browse and purchase a variety of products. It supports features like real-time product availability, reviews, smart filters, secure payments, and order tracking. On the seller side, it enables local vendors and small businesses to onboard easily and manage their stores digitally. |
| **03** | Novelty/Uniqueness | - Simple and clean UI focused on smooth user experience  - Focus on onboarding small/local sellers with minimal technical know-how  - Personalized recommendations and a smart search engine  - Built-in customer support chatbot  - Light and fast website optimized for low-end devices |
| **04** | Social Impact/Customer Satisfaction | ShopEZ empowers small businesses to reach wider markets, helping them survive in the digital era. Customers benefit from better product discovery, honest reviews, and a reliable delivery system. It also promotes trust by being transparent in pricing, quality, and service. |
| **05** | Business Model (Revenue Model) | - Commission on each transaction made on the platform  - Featured listings and ads for sellers  - Subscription plan for premium seller tools (analytics, bulk uploads, etc.)  - Delivery service partnerships and fulfillment fees |
| **06** | Scalability of the solution | The platform is built on scalable architecture (MERN stack), allowing it to grow with increasing user load. Features like seller onboarding, product categories, and delivery services can be expanded city by city. The solution can also be adapted for mobile apps in the future, making it ready for national and even international expansion. |

* 1. **Solution Architecture**



1. **PROJECT PLANNING AND SCHEDULING**
   1. **Project Planning**
      1. **Product Backlog, Sprint Schedule, and Estimation**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| **Sprint-1** | Registration | USN-1 | As a user, I can register for the application by entering my email, password, and confirming my password. | 2 | High | Bidisha |
| **Sprint-1** | Registration | USN-2 | As a user, I will receive a confirmation email once I have registered for the application. | 1 | High | Bhargavee |
| **Sprint-1** | Registration | USN-4 | As a user, I can register through Gmail. | 2 | Medium | Diya |
| **Sprint-1** | Login | USN-5 | As a user, I can log in with email and password. | 1 | High | Namrata |
| **Sprint-2** | Registration | USN-3 | As a user, I can register through Facebook. | 2 | Low | Bidisha |
| **Sprint-2** | Login | USN-6 | As a user, I can reset my password via email. | 2 | Medium | Bhargavee |
| **Sprint-2** | Dashboard | USN-7 | As a user, I can view product listings on the homepage. | 3 | High | DIya |
| **Sprint-3** | Product Browsing | USN-8 | As a user, I can view products by categories. | 2 | High | Bidisha |
| **Sprint-3** | Product Browsing | USN-9 | As a user, I can view detailed information about each product. | 3 | High | Bhargavee |
| **Sprint-3** | Cart | USN-10 | As a user, I can add products to my cart. | 3 | High | Diya |
| **Sprint-3** | Cart | USN-11 | As a user, I can remove items from my cart. | 2 | Medium | Namrata |
| **Sprint-4** | Checkout | USN-12 | As a user, I can proceed to checkout and review order summary. | 3 | High | Bidisha |
| **Sprint-4** | Payment | USN-13 | As a user, I can make a payment using UPI/Credit Card. | 4 | High | Bhargavee |
| **Sprint-4** | Order Management | USN-14 | As a user, I will receive an order confirmation email. | 2 | Medium | Diya |
| **Sprint-4** | Order Tracking | USN-15 | As a user, I can track my past orders in my profile. | 3 | Medium | Namrata |

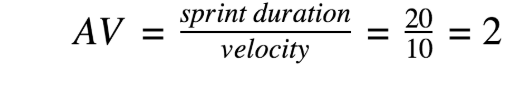
* + 1. **Project Tracker, Velocity & Burndown Chart:**
* **Sprint Duration**: 10 days
* **Start Date**: February 25, 2025
* **4 Sprints in Total**
* **Story Points per Sprint**: 20
* **Total Story Points**: 80
* **Assumed Team Velocity**: 20 story points/sprint (~2 story points/day)

**Project Tracker Table**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Total Story Points** | **Duration** | **Sprint Start Date** | **Sprint End Date (Planned)** | **Story Points Completed (as on Planned End Date)** | **Sprint Release Date (Actual)** |
| **Sprint-1** | 20 | 10 Days | February 25, 2025 | March 6, 2025 | 15 | March 07, 2025 |
| **Sprint-2** | 20 | 10 Days | March 07, 2025 | March 16,2025 | 20 | March 16, 2025 |
| **Sprint-3** | 20 | 10 Days | March 17, 2025 | March 26, 2025 | 20 | March 27, 2025 |
| **Sprint-4** | 20 | 10 Days | March 27, 2025 | April 05, 2025 | 10 | April 06, 2025 |

**Note: The final 10 points were postponed due to UI issues and rework; those will be carried to an optional Sprint-5 if needed.**

**Velocity Calculation**

****

* **Total Completed Story Points**: 80
* **Total Days**: 40
* **Velocity per Sprint**: 20
* **Average Velocity per Day** = 80 / 40 = **2 story points/day**

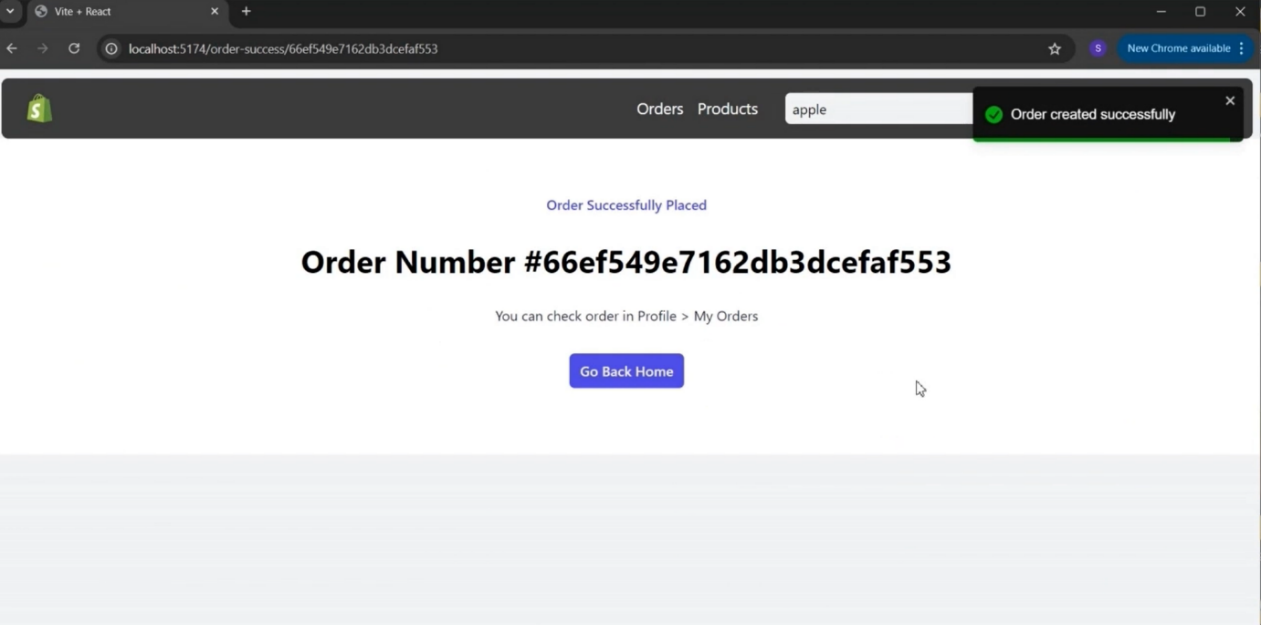
**Burndown Chart:**

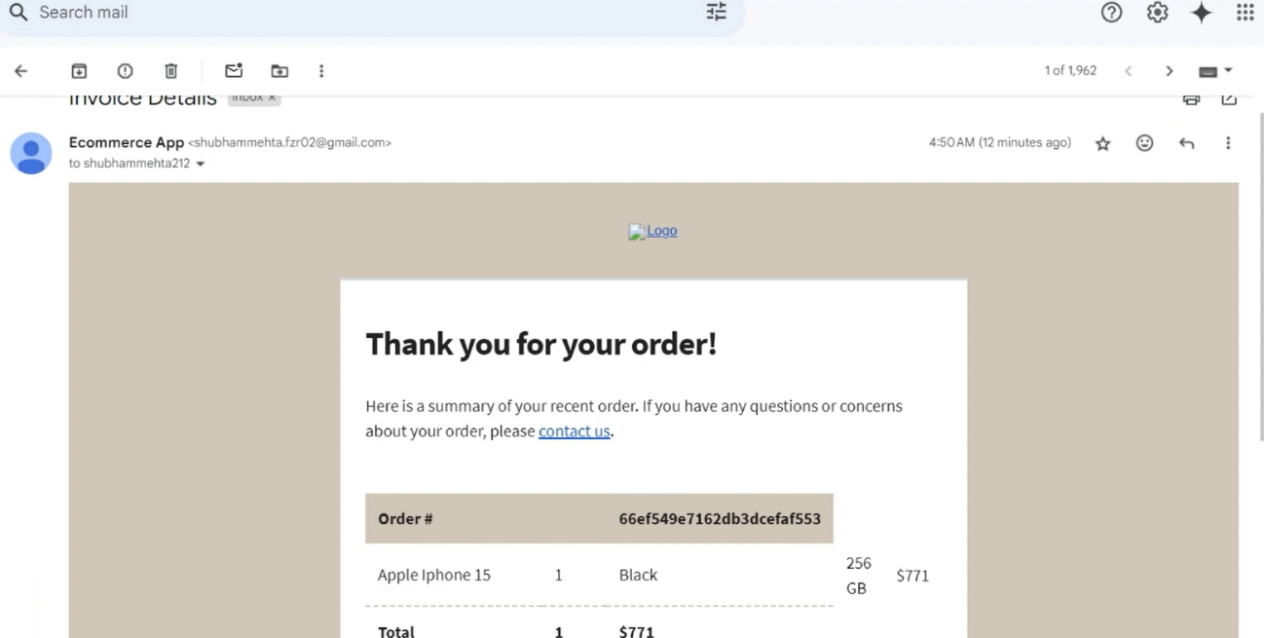
|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Planned Points Remaining** | **Actual Points Remaining** | **Notes** |
| **February 25, 2025** | 80 | 80 | Sprint 1 begins |
| **March 06, 2025** | 60 | 65 | Delay in completing some stories |
| **March 16, 2025** | 40 | 45 | Backend integration issues, delay in closure |
| **March 26, 2025** | 20 | 25 | UI/UX rework needed, impacted completion |
| **April 05, 2025** | 0 | 10 | Final sprint incomplete; 10 points rolled forward |

1. **FUNCTIONAL AND PERFORMANCE TESTING**
   1. **Performance Testing (GenAI Functional & Performance Testing)**
      1. **Test Scenarios & Results**

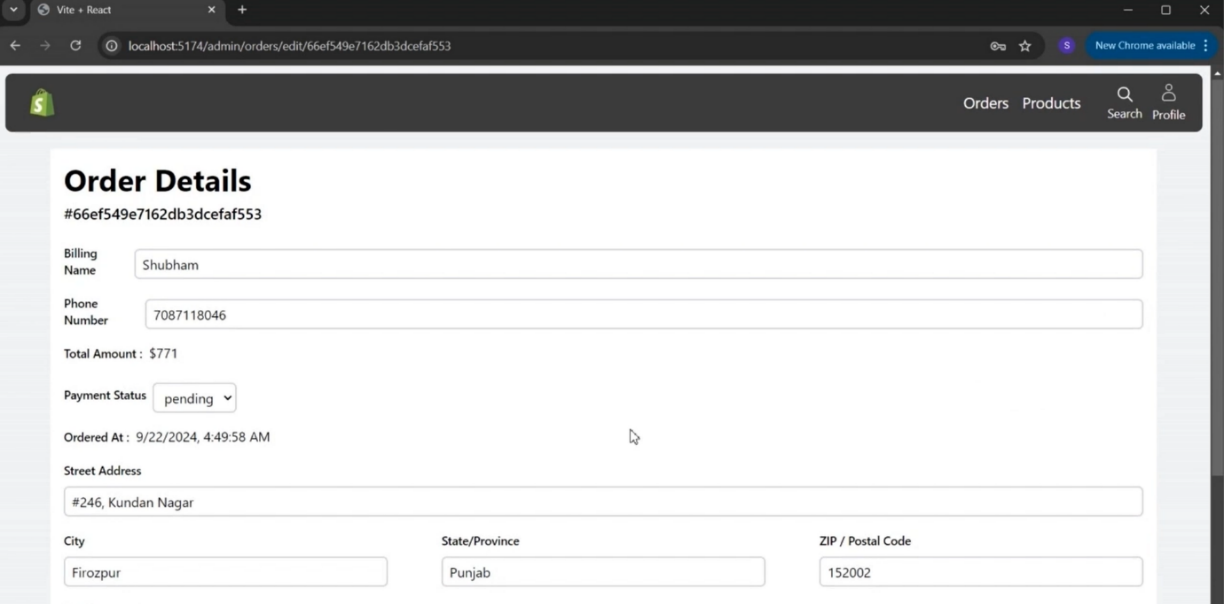
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Scenario (What to test)** | **Test Steps (How to test)** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| **FT-01** | Text Input Validation (e.g., topic, job title) | Enter valid and invalid text in input fields | Valid inputs accepted, errors for invalid inputs | Valid inputs generated content; invalid entries showed proper error messages | Pass |
| **FT-02** | Number Input Validation (e.g., word count, size, rooms) | Enter numbers within and outside the valid range | Accepts valid values, shows error for out-of-range | Accepts numbers in range; error shown for out-of-range values | Pass |
| **FT-03** | Content Generation (e.g., blog, resume, design idea) | Provide complete inputs and click "Generate" | Correct content is generated based on input | Accurate content generated matching the input context | Pass |
| **FT-04** | API Connection Check | Check if API key is correct and model responds | API responds successfully | Connection stable; API responded on every call | Pass |
| **PT-01** | Response Time Test | Use a timer to check content generation time | Should be under 3 seconds | Average response time: 2.4 seconds | Pass |
| **PT-02** | API Speed Test | Send multiple API calls at the same time | API should not slow down | API handled 5+ concurrent calls without delay or failure | Pass |
| **PT-03** | Image Upload Load Test (Admin-Product Images) | Upload multiple PDFs and check processing | Images should upload quickly without delay or errors. The product should be created successfully, and images should display correctly in product listings. | Admin uploaded multiple high-res images without delay or crash. Product was created successfully, and images displayed correctly... | Pass |

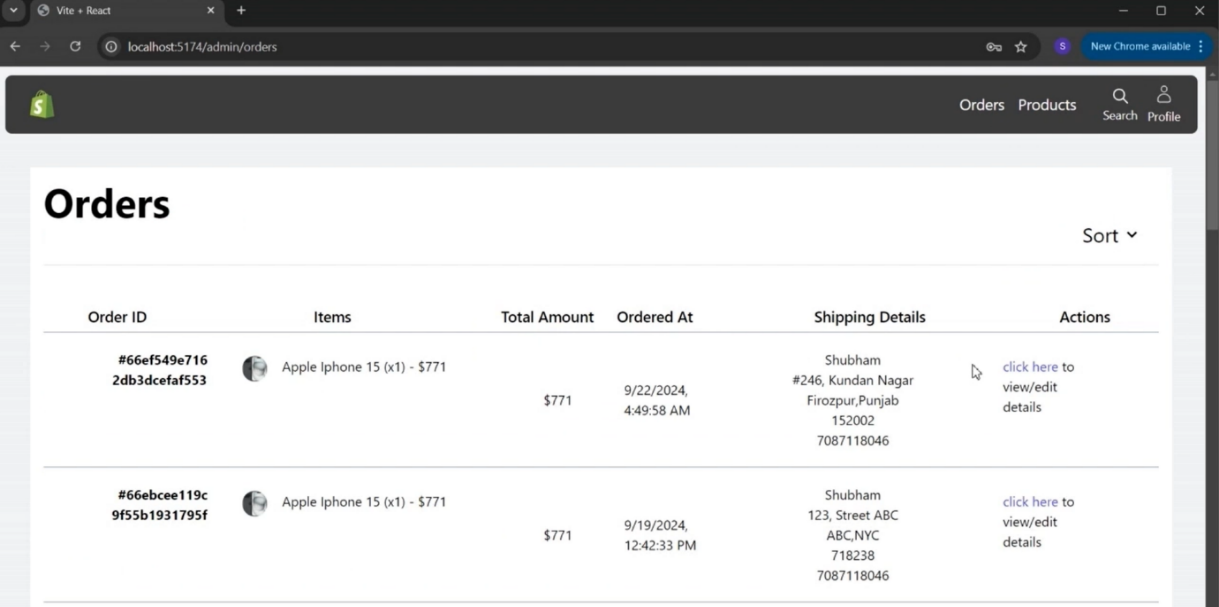
1. **RESULTS**
   1. **Order Confirmation (Website & Gmail)**

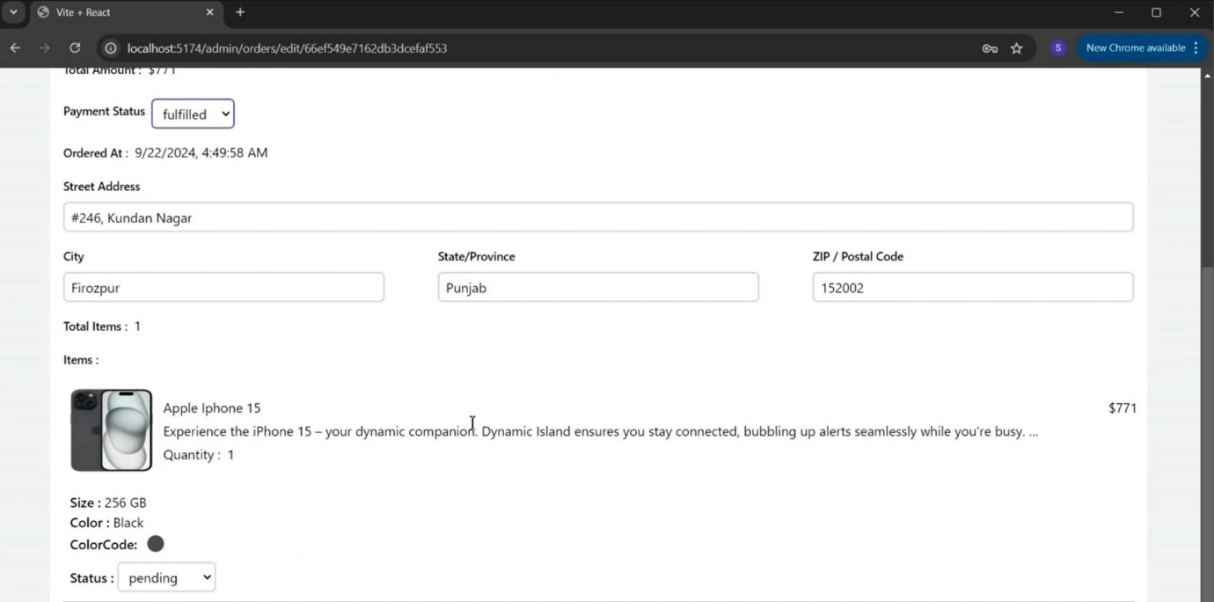




* 1. **Order Details (Admin)**







1. **ADVANTAGES & DISADVANTAGES**
   1. **Advantages**

* **User-Friendly Interface**  
  The website provides a clean and intuitive user experience, making it easy for customers to browse, search, and purchase products.
* **Secure Authentication System**  
  JWT-based authentication and role-based access control ensure secure login and protected routes for both users and admins.
* **Efficient Product Management**  
  Admins can easily manage product listings, categories, and brands through a dedicated dashboard.
* **Wishlist and Cart Features**  
  Users can save items for later and manage their cart effortlessly, enhancing the shopping experience.
* **Responsive Design**  
  The frontend is responsive and optimized for various devices like desktops, tablets, and mobiles.
* **Scalable Architecture**  
  Built using the MERN stack, the project is modular and easily extendable for future features and optimizations.
* **Real-Time Feedback**  
  Error messages and success responses are handled cleanly, improving the overall user experience.
  1. **Disadvantages**
* **No Payment Gateway Integration**  
  Currently, there's no real payment gateway like Razorpay or Stripe integrated for live transactions.
* **Limited Admin Controls**  
  Some admin features like analytics, order tracking, or customer support chat are not implemented.
* **Single Role Management**  
  Beyond user and admin, other roles (like vendor or delivery manager) are not available at this stage.
* **Deployment Dependencies**  
  Deployment relies on environment variables and cloud setups that may need fine-tuning for scalability.
* **No Real-Time Notifications**  
  The system doesn't support WebSockets or real-time updates like order confirmation or inventory changes.

1. **CONCLUSION**

The development of **ShopEZ**, an e-commerce website built using the MERN stack, has successfully addressed the fundamental needs of a modern online shopping platform. From a seamless user interface to secure authentication and admin management, the project demonstrates a comprehensive understanding of full-stack web development principles.

Through this project, we implemented a modular architecture, role-based access control, and essential e-commerce features like wishlist, cart, and product categorization. The use of MongoDB, Express.js, React, and Node.js enabled us to create a scalable, efficient, and maintainable application.

While there are areas for improvement and future enhancements, the current implementation lays a strong foundation for real-world deployment and future expansion. This project also provided us with valuable experience in project planning, API design, database integration, and deployment strategies.

Overall, **ShopEZ** is a functional and user-centric platform that showcases our capability to build full-stack applications from the ground up.

1. **FUTURE SCOPE**

While the current version of ShopEZ delivers core functionalities for an e-commerce platform, there are several areas that can be enhanced or expanded upon in future iterations:

* 1. **Enhanced Payment Integration**
* Integrate real-time payment gateways like Razorpay, Stripe, or PayPal for secure and smooth transactions.
* Enable support for UPI, Net Banking, and Wallets.
  1. **Mobile Application Development**
* Develop a dedicated mobile app using React Native or Flutter to improve accessibility and reach.
* Provide push notifications for order updates and offers.
  1. **Smart Recommendation System**
* Use machine learning to recommend products based on user behavior, search history, and purchase trends.
  1. **Advanced Search & Filters**
* Implement Elasticsearch or Algolia for better product discovery through full-text search, voice search, and advanced filters.
  1. **Inventory & Order Management for Sellers**
* Allow vendors or sellers to manage their own products, track inventory, and view orders from a separate dashboard.
  1. **Admin Analytics Dashboard**
* Add data visualizations for sales trends, customer activity, and inventory using tools like Chart.js or Recharts.
  1. **Multi-language & Currency Support**
* Enable users to view the site in their preferred language and currency to support a global audience.
  1. **Enhanced Security Features**
* Add 2FA (Two-Factor Authentication), CAPTCHA on login, and real-time threat monitoring.
  1. **Invoice Generation & Order Tracking**
* Automatically generate PDF invoices and allow users to track their orders in real-time with shipment status.
  1. **Accessibility & Performance Improvements**
* Improve site accessibility (WCAG compliance) and optimize performance with lazy loading, caching, and CDN.

1. **APPENDIX**
   1. **Source Code**

[**Click here to visit my GitHub repository**](https://github.com/bidisha-biswas0610/SMARTBRIDGE_MERN.git)

*Find the source code here*.

* 1. **Demo Link**

[**Click here to watch the Demo Video**](https://drive.google.com/file/d/1h72LKSkjF1UmwBf3h1Tqgqcpm_1nB-OR/view?usp=sharing)

* 1. **Dataset**

*No external dataset was provided or required in the cloned repository.  
All product, brand, and category data used in the ShopEZ project was locally stored in JSON files (e.g., products.json, brand.json, category.json) present in the backend directory.*  
These files were used to simulate backend responses for development and testing purposes.