# **ELECTRONIC STORE**

**AN INTERNSHIP REPORT**

***Submitted by***

# **Patel Anshul Jayeshbhai**

**GTU Enrollment No.: 210670107047**

***In fulfillment for the award of the degree of***

# **BACHELOR OF ENGINEERING**

***In***

**Computer Engineering**

## **SAL Institute of Technology and Engineering Research,**

**Ahmedabad**

 

## **Gujarat Technological University, Ahmedabad**

**May, 2025**

 

**SAL Institute of Technology and Engineering Research**

**opposite Science City, Sola, Ahmedabad, Gujarat 380060.**

**CERTIFICATE**

This is to certify that the project report submitted along with the project entitled **TechBazaar**

has been carried out by **Soni Dhruv Alpesh** **Bhai** under my guidance in partial

fulfillment for the degree of Bachelor of Engineering in Computer Engineering, 8th Semester

of Gujarat Technological University, Ahmadabad during the academic year 2024-2025.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Krishna Hingrajiya Dr. Nimisha Patel

Internal Guide Head of the Department

**Industry Letter Head**

Date:

**TO WHOM IT MAY CONCERN**

This is to certify that Soni Dhruv Alpesh Bhai, a student of SAL Institute of Technology and Engineering Research has successfully completed his internship in the field of MERN Stack from 20/01/2025 to 20/04/2025 (Total number of Weeks: 12) under the guidance of Yagnesh Modh.

His internship activities include MERN Stack.

During the period of his internship program with us, he had been exposed to different processes and was found diligent, hardworking and inquisitive.

We wish him every success in his life and career.

For The Special Character,

Authorised Signature with Industry Stamp



**SAL Institute of Technology and Engineering Research**

**opposite Science City, Sola, Ahmedabad, Gujarat 380060.**

# **DECLARATION**

We hereby declare that the Internship / Project report submitted along with the

Project entitled **Internship / TechBazaar** submitted in partial fulfillment for the degree of Bachelor of Engineering in Computer Engineering to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at The Special Character under the supervision of Yagnesh Modh and that no part of this report has been directly copied from any students’ reports or taken from any other source, without providing due reference.

Name of the Student Sign of Student

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **Acknowledgment**

I wish to express our sincere gratitude to our External guide **Mr.** **Yagnesh Modh** for continuously guiding me at the company and answering all my doubts with patience. I would also like to thank my Internal Guide **Dr. Krishna Hingrajiya** for helping us through our internship by giving us the necessary suggestions and advice along with their valuable co-ordination in completing this internship.

We also thank our parents, friends and all the members of the family for their precious support and encouragement which they had provided in completion of our work. In addition to that, we would also like to mention the company personals who gave us the permission to use and experience the valuable resources required for the internship.

Thus, In conclusion to the above said, we once again thank the staff members of **The Special Character** for their valuable support in completion of the project.

Thank You,

Soni Dhruv A.

**Abstract**

TechBazaar is an innovative and feature-rich e-commerce web application specifically tailored for the online sale of electronic products. Built using the powerful and scalable MERN stack (MongoDB, Express.js, React.js, and Node.js), TechBazaar delivers a modern, responsive, and interactive user experience that mirrors the capabilities of leading e-commerce giants like Amazon and Flipkart, while maintaining a focused niche in the electronics domain.

The platform hosts a wide variety of electronic goods including smartphones, laptops, tablets, smartwatches, televisions, audio devices, computer accessories, and more. Each product is stored in a well-structured JSON-based product catalog that supports easy searching, filtering, and categorization.

TechBazaar is not just a shopping platform; it's a scalable and maintainable solution for managing large volumes of electronic products, offering a robust structure for future integration of features such as order tracking, wishlists, customer support chat, and AI-based product recommendations.

**List of Figures**

**List of Tables**

**List of Abbreviations**

**Table of Contents**

Acknowledgement ......................................................................................................i

Abstract .....................................................................................................................ii

List of Figures...........................................................................................................iii

List of Tables............................................................................................................iv

List of Abbreviations..................................................................................................v

Table of Contents......................................................................................................vi

**Chapter 1: Title of chapter........................................................................................1**

1.0 Overview of the Company.....................................................................................2

1.1.1 How to do .................................................................................................3

1.2 Introduction to Topic ................... .....................................................................4

1.3 Introduction to Topic ........................................................................................5

1.4 Introduction to Topic ........................................................................................6

**Chapter 2 Title of chapter........................................................................................8**

2.1 Introduction to Topic ........................................................................................9

2.1.1 How to do ...............................................................................................10

**Chapter 3 Title of chapter......................................................................................11**

3.1 Introduction to Topic ......................................................................................14

3.1.1 How to do ...............................................................................................16

3.2 Introduction to Topic ......................................................................................17

References...............................................................................................................40

Appendix ................................................................................................................42

**Details of Chapters**

**Chapter 1: Overview of the Company**

**1.1 History**

**Learn about founder:**

Yagnesh Modh, A computer engineer from Nirma University has provided his expertise to HP, Nokia, Thought works and many startups like Turing.com, Jupiter Money & Visionary Schoolmen. He has been a corporate trainer since 2018 and hosted over 700+ corporate training sessions to date.

**The Idea of TSC:**

Our founder is a real dynamo of intellectual and entrepreneurial activity. After returning back from Bangalore to his hometown Ahmedabad in 2020, he noted the significant importance of tech-grooming students to boost their learning and placement in good companies, eventually expanding Ahmedabad's tech value. Thus, in 2021, he launched 'The Special Character' on his quest as a freelance corporate trainer to bridge the university-industry gap and establish a superior tech student resource.

**1.2 Different product / scope of work**

Innovation underpins our working methods. To improve the world for coming generations, we innovate. relationship building is openness. With all of our hearts, we aspire to serve knowledge. "Available from anywhere" educational framework

**1.3 Organization chart**

Project Manager

|

---------------------------------------------------------------------------------------

| | |

Frontend Team Backend Team Database Admin

| | |

UI/UX Designer API Developer (Next.js) MongoDB Architect

Next.js Developer Auth & Payment Integration Data Model Designer

State Management Security & Middleware Backup & Performance

(Redux/Context API) Developer Optimization Expert

|

Quality Assurance

|

Testing & Debugging Team

(Unit, Integration, UI tests)

|

Deployment & DevOps

|

Version Control & CI/CD Specialist

Hosting, Monitoring & Maintenance

**1.4 Capacity of plant**

**1.4.1 User Capacity:**

* Capable of handling up to 10,000+ concurrent users under standard load conditions.
* Built with scalability in mind, allowing horizontal scaling using containerization.

**1.4.2 Product Catalog Capacity**

* Product data is stored efficiently using MongoDB, which allows for high-volume document-based storage with rapid query performance.

**1.4.3 Future Scalability**

* The system is architected in a modular way, allowing for easy integration of new features like:
  + Multi-vendor support
  + AI-powered recommendation engines
  + Mobile app versions of the platform

**Chapter 2: Overview of different plant/unit/department/shop of the organization and Layout of the production/process being carried out in company**

**2.1 It includes the details about the work being carried out in each department.**

| **Unit / Department** | **Description** |
| --- | --- |
| **Software Development Unit** | This unit is responsible for the core development of web and mobile applications. It includes frontend and backend developers who collaborate to build responsive and high-performance software systems. |
| **UI/UX Design Unit** | Handles user interface and user experience design. The team creates design mockups, wireframes, and prototypes to ensure applications are intuitive and user-friendly. |
| **Quality Assurance (QA) Unit** | Ensures software quality through rigorous testing—manual and automated. They identify bugs, validate features, and perform performance testing. |
| **Project Management Unit** | Oversees the planning, execution, and delivery of projects. Project managers assign tasks, set deadlines, and ensure that the project aligns with client goals. |
| **Database Administration Unit** | Manages the design, security, and performance of databases. This unit ensures data consistency and handles large-scale data operations. |

**2.2 List the technical specifications of major equipment used in each department.**

During my internship at the software development company, each department is equipped with modern systems and tools to support smooth workflow, high performance, and collaborative development. Below is a summary of the technical equipment and software used across departments:

**1. Frontend Development Department:**

* **Systems**: Intel Core i5/i7 or AMD Ryzen 5+, 16 GB RAM, 512 GB SSD
* **Monitors**: Full HD or 4K 24” monitors for accurate UI design
* **Tools**: Visual Studio Code, Chrome Developer Tools, React.js, Next.js
* **Design Support**: Access to Figma and Adobe XD for interface collaboration

**2. Backend Development Department:**

* **Systems**: Intel i7/Ryzen 7, 32 GB RAM, 1 TB SSD
* **Tools**: Node.js, Express.js, Postman for API testing, GitHub for version control
* **Cloud Platforms**: AWS, Heroku, or Vercel for deployment and server management

**3. UI/UX Design Department:**

* **Laptops**: MacBook Pro / Windows laptops with Retina or high-res displays
* **Software**: Figma, Adobe XD, Illustrator, Photoshop
* **Accessories**: Graphic tablets for sketching and precision design

**4. Database Management Department:**

* **Systems**: 16–32 GB RAM, SSD-based machines for fast query handling
* **Tools**: MongoDB Compass, SQL Server Management Studio, MongoDB Atlas
* **Backup Systems**: Cloud storage for automated backups and scaling

**5. Quality Assurance (QA) Department:**

* **Machines**: Mid-to-high performance PCs with 8–16 GB RAM
* **Testing Tools**: Selenium, Cypress, Jest, Postman for API and UI testing
* **Devices**: Real Android/iOS phones for cross-device compatibility testing

**2.3 Prepare schematic layout which shows the sequence of operation for**

**manufacturing of end product.**

1. Requirement Gathering

↓

2. Planning & Analysis

↓

3. UI/UX Design (Wireframes & Prototypes)

↓

4. Frontend Development (React/Next.js)

↓

5. Backend Development (Node.js, Express.js)

↓

6. Database Design & Integration (MongoDB)

↓

7. API Development & Integration

↓

8. Testing & Debugging (QA Process)

↓

9. Deployment (Vercel, Heroku, AWS)

↓

10. Maintenance & Updates

**2.4 Explain in details about each stage of production.**

**1. Requirement Gathering:**

* **Objective**: Understand what the client or end-user needs.
* **Activities**:
  + Conduct meetings/interviews with stakeholders.
  + Define functional and non-functional requirements (performance, security).
  + Create user stories or use cases.
* **Output**: Software Requirement Specification (SRS) document.

**2. Planning & Analysis**

* **Objective**: Plan the project roadmap and analyze technical feasibility.
* **Activities**:
  + Decide on the tech stack (e.g., MERN: MongoDB, Express.js, React.js, Node.js).
  + Break down the project into milestones and sprints.
  + Assign roles and responsibilities.
* **Output**: Project plan, timeline (e.g., Gantt chart), and team setup.

**3. UI/UX Design**

* **Objective**: Design the user interface and enhance user experience.
* **Activities**:
  + Create wireframes, mockups, and interactive prototypes.
  + Choose design components (fonts, colors, buttons, icons).
  + Review and get approval from stakeholders.
* **Tools Used**: Figma, Adobe XD, Sketch.
* **Output**: Approved UI designs and style guide.

**4. Frontend Development**

* **Objective**: Convert UI designs into a functional interface.
* **Activities**:
  + Code components using React.js or Next.js.
  + Implement routing, navigation, and state management.
  + Ensure responsiveness across all devices (mobile-first design).
* **Output**: Functional user interface of the application.

**5. Backend Development**

* **Objective**: Handle business logic and server-side operations.
* **Activities**:
  + Create APIs using Node.js and Express.js.
  + Handle authentication, session management, and server configuration.
  + Manage secure data transactions.
* **Output**: Working APIs and a secure backend environment.

**6. Database Design & Integration**

* **Objective**: Store and retrieve application data efficiently.
* **Activities**:
  + Design data models using MongoDB (collections, schemas).
  + Set up cloud databases (MongoDB Atlas).
  + Integrate database with backend APIs.
* **Output**: A connected and optimized database system.

**7. API Development & Integration**

* **Objective**: Enable frontend to communicate with backend.
* **Activities**:
  + Connect APIs to frontend forms and features.
  + Fetch, update, delete, and display data in real time.
  + Handle errors and validation.
* **Output**: Seamless interaction between user interface and server.

**8. Testing & Debugging**

* **Objective**: Ensure the product works as expected.
* **Activities**:
  + Unit Testing: Test individual functions or components.
  + Integration Testing: Test how components work together.
  + UI/UX Testing: Ensure user flow and visuals are correct.
  + Bug Fixing and Performance Optimization.
* **Tools Used**: Jest, Cypress, Postman, Chrome DevTools.
* **Output**: A bug-free and stable application ready for deployment.

**9. Deployment**

* **Objective**: Make the application accessible to end users.
* **Activities**:
  + Set up hosting platforms (e.g., Vercel, Heroku, AWS).
  + Configure environment variables and deployment settings.
  + Monitor live performance.
* **Output**: The live version of the application is available online.

**10. Maintenance & Updates**

* **Objective**: Continuously improve and support the application.
* **Activities**:
  + Monitor bugs, crashes, or security issues.
  + Update features based on user feedback.
  + Perform backups and server maintenance.
* **Output**: An evolving, user-friendly, and reliable product.

**Chapter 3: Introduction to Project / Internship and Project**

**3.1 Project / Internship Summary – Key To a good summary is the FIRST sentence,**

I worked on the development of “**TechBazaar**,” a full-stack e-commerce web application for electronic products during my internship at a software development firm.

The platform is designed to provide users with a seamless and responsive shopping experience for purchasing electronics like smartphones, laptops, TVs, and accessories. It includes key e-commerce features such as product listing, likes and ratings, cart management, checkout process, and payment integration.

Throughout the internship, I was actively involved in all stages of development—from planning, UI/UX design, and frontend interface building using React.js, to backend API creation with Node.js and database handling using MongoDB. I also participated in code reviews, testing, debugging, and deployment using GitHub and Vercel.

This internship significantly enhanced my practical knowledge of full-stack development, team collaboration, and project lifecycle management, preparing me for real-world software engineering roles.

**3.2 Purpose**

The purpose of **TechBazaar** is to create a modern, user-friendly, and efficient **e-commerce platform** dedicated to the online sale of **electronic products** such as smartphones, laptops, smartwatches, TVs, and accessories.

**It aims to:**

1. Provide a seamless shopping experience for users by offering intuitive navigation, attractive UI, real-time cart updates, and secure checkout options.
2. Simplify product management for administrators with features like product CRUD operations, inventory updates, and real-time catalog control.
3. Serve as a scalable and customizable solution for tech-focused businesses that want to sell electronic goods online.

**3.3 Objectives**

* **Develop a full-stack e-commerce web application** using the MERN stack to demonstrate practical skills in modern web development.
* **Implement key e-commerce features** such as product listing, cart management, likes, and secure payment integration.
* **Enable product management for administrators**, including the ability to add, update, and delete products using a dashboard or API.
* **Ensure mobile responsiveness and user accessibility** to support shopping across all devices (desktop, tablet, mobile).
* **Maintain data security and integrity** through user authentication, input validation, and secure server-side processing.
* **Gain hands-on experience with industry-standard tools** such as Git, GitHub, Postman, and MongoDB Atlas for collaboration and testing.

**3.4 Scope (what it can do and can’t do)**

**What TechBazaar Can Do:**

1. **Product Browsing & Display**  
   Users can view a wide range of electronic products like smartphones, laptops, TVs, smartwatches, and accessories.
2. **Search & Filtering**  
   Users can search for products and apply filters to narrow down results based on categories or preferences.
3. **Cart**   
   Users can add or remove products from the cart and view a cart summary with total pricing.
4. **User Interactions**  
   Users can like products, rate them, and leave reviews.
5. **Authentication System**  
   Users can sign up, log in, and securely manage their accounts.
6. **Admin Panel**  
   Admins can add, edit, or delete products (CRUD operations) and manage the product catalog.
7. **Secure Checkout Integration**  
   The app supports basic payment gateway integration for processing orders.
8. **Responsive Design**  
   Works smoothly across mobile, tablet, and desktop devices.

**What TechBazaar Can’t Do:**

1. **Multi-Vendor Functionality**  
   It doesn’t allow multiple sellers to manage their own product listings (like in Amazon Marketplace).
2. **Real-Time Order Tracking**  
   There's no integration with courier/delivery systems for tracking orders after purchase.
3. **AI-Powered Recommendations**  
   Product suggestions are not based on user behavior or purchase history.
4. **Live Chat Support**  
   It does not offer live customer support or chatbot integration.
5. **Inventory Auto-Update on Purchase**  
   Inventory does not automatically update in real-time when purchases are made (unless explicitly implemented).
6. **Mobile App Version**  
   It is a web-only application; there is no native Android/iOS version.

**3.5 Technology and Literature Review**

**Technology Review**

The development of **TechBazaar** involved the use of modern and efficient web technologies, particularly the MERN stack, which enables rapid and scalable development of full-stack web applications.

**1. MongoDB (Database)**

* A NoSQL document-oriented database used for storing flexible and scalable product and user data.
* Supports dynamic schema, making it suitable for managing diverse electronic product listings.

**2. Express.js (Backend Framework)**

* A lightweight Node.js web framework used to build RESTful APIs.
* Handles routing, middleware, server logic, and integration with the MongoDB database.

**3. React.js (Frontend Library)**

* A component-based JavaScript library used to build dynamic and responsive user interfaces.
* React Hooks and Context API/Redux were used for state management and real-time updates.

**4. Node.js (Runtime Environment)**

* A JavaScript runtime environment that allows backend development using JavaScript.
* Used for API creation, server logic, and real-time data handling.

**5. Tailwind CSS (Styling Framework)**

* A utility-first CSS framework that allowed for custom design implementations with ease and responsiveness.
* Enabled rapid UI development without writing lengthy custom CSS.

**6. Deployment Tools**

* **MongoDB Atlas**: Cloud-based database used for storing real-time data securely.

**Literature Review**

The development and architecture of TechBazaar were guided by various research materials, technical documentation, and real-world case studies related to e-commerce systems and modern web development practices.

**1. E-Commerce Platform Analysis**

* Platforms like Amazon, Flipkart, and BestBuy were analyzed to understand essential e-commerce features such as product filtering, cart management, rating systems, and responsive design principles.
* User experience (UX) design patterns were studied for layout structure and checkout flow optimization.

**2. Research on Full-Stack Web Development**

* Articles and tutorials from MDN Web Docs, FreeCodeCamp, and GeeksforGeeks were referred to for best practices in building secure and scalable applications using the MERN stack.
* GitHub repositories and open-source projects helped in understanding integration patterns and component structure.

**3. Design and UI/UX Trends**

* Key concepts like mobile-first design, accessibility, and color contrast ratios were incorporated to enhance usability.

**3.6 Project / Internship Planning**

**3.6.1 Project / Internship Development Approach and Justification**

**Development Approach**

The **development approach** adopted for the **TechBazaar** project was based on the **Agile methodology**, specifically following an **incremental and iterative model**. The project was divided into small, manageable phases (sprints), where each phase focused on building and refining core functionalities. Regular testing, feedback, and updates ensured flexibility and continuous improvement.

The MERN stack was chosen to deliver a full-stack solution, enabling the creation of a single-page application (SPA) with a dynamic user interface, efficient data flow, and a scalable backend.

**Justification for Approach**

* **Agile Methodology**: Chosen to support iterative development, frequent feedback loops, and flexibility to adapt to changes during the project.
* **MERN Stack**: Offered a unified JavaScript-based environment across frontend and backend, reducing complexity and increasing efficiency.
* **Component-Based Frontend (React.js)**: Enabled reusability and faster development of UI components.
* **NoSQL Database (MongoDB)**: Ideal for e-commerce platforms where product data is dynamic and varies across categories.

**3.6.2 Project / Internship Effort and Time, Cost Estimation**

The project, developed during my internship, required consistent planning, focused development efforts, and time-bound execution. Although the project was part of a learning experience and did not involve direct financial investment, this section outlines the estimated time and effort involved, along with a theoretical cost analysis to reflect its real-world value.

**Effort and Time Estimation**

The project spanned approximately **11 weeks**, with work divided across various development phases including requirement gathering, UI/UX design, frontend and backend development, testing, and deployment.

**Effort Distribution Across Phases**

| **Development Phase** | **Duration** | **Estimated Hours** |
| --- | --- | --- |
| Requirement Analysis & Planning | 1 week | 10 hours |
| UI/UX Design | 1 week | 12 hours |
| Frontend Development (React.js) | 2 weeks | 30 hours |
| Backend Development (Node.js) | 2 weeks | 30 hours |
| Database Design & Integration | 1 week | 10 hours |
| API Integration & Testing | 1 week | 12 hours |
| Debugging & Optimization | 1 week | 10 hours |
| Deployment & Hosting | 1 week | 8 hours |
| Documentation & Finalization | 1 week | 8 hours |
| **Total Time & Effort** | **11 weeks** | **130 hours approx.** |

**3.6.3 Roles and Responsibilities**

I was assigned multiple responsibilities as a **full-stack web developer** working on the TechBazaar e-commerce platform. The roles were distributed based on the project phases and development requirements. This hands-on involvement allowed me to contribute to both the frontend and backend development process, while also gaining experience in design, testing, and deployment.

**3.6.4 Group Dependencies**

The successful development of the **TechBazaar** project relied heavily on effective collaboration, communication, and coordination among various team members and functional units. As a full-stack development project, different parts of the system were interdependent and required timely contributions from each group or individual to ensure smooth progress and integration.

**3.7 Project / Internship Scheduling (Gantt Chart/PERT/Network Chart)**

**Phases Used in Scheduling**

| **Phase Name** | **Description** |
| --- | --- |
| **Platform Structure Setup** | Setting up the folder structure, initializing the Next.js project, installing required packages, and organizing routes/components. |
| **Design Integration Phase** | Converting Figma/UI mockups into reusable and responsive React components using Tailwind CSS. |
| **Frontend Features Development** | Building major frontend features like Product Cards, Search Bar, Ratings, Cart System, and Checkout Page. |
| **Backend API Construction** | Creating Express.js APIs for user login, signup, product retrieval, cart management, and order placement. |
| **Database Collection Setup** | Designing MongoDB collections and schemas for users, products, and orders. |
| **Authentication Module Setup** | Implementing user registration, login, JWT token-based auth, and protected routes. |
| **Admin Dashboard Implementation** | Creating the interface and logic for product CRUD, stock management, and category control by admin users. |
| **Frontend-Backend Integration** | Connecting frontend features with backend APIs using Axios/Fetch and managing state flow. |
| **Deployment & Hosting Phase** | Deploying the frontend on Vite and backend APIs on Render/Heroku, with MongoDB hosted on Atlas. |
| **Final Testing & Documentation** | Conducting manual testing of features, final bug fixes, and preparing the final project report, demo presentation, and documentation. |

**Gantt Chart Representation (Week-wise Summary)**

| **Phase** | **W 1** | **W 2** | **W 3** | **W 4** | **W 5** | **W 6** | **W 7** | **W 8** | **W 9** | **W 10** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Platform Structure Setup | ✅ ✅ |  |  |  |  |  |  |  |  |  |
| Design Integration Phase |  | ✅ ✅ | ✅ ✅ |  |  |  |  |  |  |  |
| Frontend Features Development |  |  | ✅ ✅ | ✅ ✅ | ✅ ✅ |  |  |  |  |  |
| Backend API Construction |  |  |  | ✅ ✅ | ✅ ✅ |  |  |  |  |  |
| Database Collection Setup |  |  |  | ✅ ✅ | ✅ ✅ |  |  |  |  |  |
| Authentication Module Setup |  |  |  |  | ✅ ✅ | ✅ ✅ |  |  |  |  |
| Admin Dashboard Implementation |  |  |  |  |  | ✅ ✅ | ✅ |  |  |  |
| Frontend-Backend Integration |  |  |  |  |  | ✅ ✅ | ✅ ✅ | ✅ ✅ |  |  |
| Deployment & Hosting Phase |  |  |  |  |  |  |  | ✅ ✅ | ✅ ✅ |  |
| Final Testing & Documentation |  |  |  |  |  |  |  |  | ✅ ✅ | ✅ ✅ |

**Chapter 4: System Analysis**

**4.1 Study of Current System**

An analysis of existing e-commerce platforms and their limitations was conducted to better understand current systems in the online electronics marketplace. This study involved evaluating platforms like Amazon, Flipkart, and other niche tech stores to identify their workflows, strengths, and potential areas for improvement in user experience and management features.

**Overview of Current E-Commerce Systems**

Most current e-commerce systems offer a wide variety of features such as:

* Product search and filtering
* User accounts and order tracking
* Ratings and reviews
* Secure checkout and payment integration
* Admin control panels for managing products and users

**4.2 Problem and Weaknesses of Current System**

The evaluation of existing e-commerce platforms revealed several problems and limitations that can affect both end-users and administrators—especially for platforms focusing on electronics. These challenges justified the need for a simplified and customized system like TechBazaar.

**Problems Identified in Existing Systems**

1. **Complex User Interfaces**
   * Most platforms have cluttered layouts with too many features, which confuses new users.
2. **Slow Loading and Performance Issues**

* Some platforms have slow response times due to unnecessary features, affecting the user experience.

1. **Difficult Admin Controls**

* Non-technical users find it difficult to manage inventories, pricing, or categories due to complicated dashboards and lack of visual guidance

**4.3 Requirements of New System**

Based on the limitations of existing e-commerce platforms and the specific goals of the **TechBazaar** project, a set of functional and non-functional requirements was defined. These requirements ensure that the new system is tailored for selling electronic products online with better performance, usability, and management capabilities.

**Functional Requirements**

1. **User Registration and Login**
   * Users must be able to register, log in, and manage their profiles securely.
2. **Product Browsing**
   * Visitors can browse products by category (e.g., smartphones, laptops, TVs).
   * Search and filter functionality should be available.
3. **Product Details Page**
   * Each product will have a dedicated page with specifications, images, and reviews.
4. **Add to Cart and Cart Management**
   * Logged-in users can add products to the cart, update quantities, and remove items.
5. **Checkout and Payment**
   * Users can proceed to checkout with their cart and complete payment securely.
6. **Admin Panel**
   * Admins can add, update, delete, or view products.
   * Admins can manage inventory and categories.
7. **Order Management**
   * Orders placed by users are saved in the database and visible to the admin.

**Non-Functional Requirements**

1. **Responsiveness**
   * The website must be fully responsive and work smoothly on mobile, tablet, and desktop.
2. **Scalability**
   * The system should be able to handle a growing number of users and products.
3. **Performance**
   * Fast page loads and API response times are required to provide a good user experience.
4. **Security**
   * User data must be encrypted and protected.
   * Secure authentication and authorization using JWT tokens.
5. **Maintainability**
   * Code should be modular, clean, and easy to update or extend in the future.
6. **Deployment**
   * The system must be deployed on reliable cloud platforms with a live database.

**4.4 System Feasibility**

Before proceeding with the development of **TechBazaar**, a detailed feasibility study was conducted to determine whether the project was practical, achievable, and valuable. This analysis included technical, operational, economic, and schedule feasibility to ensure the system could be developed successfully with the available resources.

**Technical Feasibility:**

* The project was built using the MERN stack, which is a widely adopted and well-supported technology for web development.
* All required development tools (VS Code, Git, Postman, MongoDB Atlas) are open-source or free, making them accessible and sustainable for long-term use.

**Operational Feasibility:**

* The system addresses real-world e-commerce needs with features like product search, cart management, admin control, and user login.
* It provides a clean, user-friendly interface, ensuring ease of use for customers and admins.
* Admin dashboard allows easy product management without requiring technical expertise.

**Economic Feasibility:**

* Since this project was developed during an internship, no major financial investment was required.
* Development was done using free-tier services and open-source technologies, minimizing costs.
* If developed commercially, it would still be cost-effective due to low infrastructure needs and minimal licensing fees.

**Schedule Feasibility:**

* The entire project was completed within the allocated **12-week internship duration**.
* Tasks were divided into manageable phases, and progress was tracked regularly.
* A realistic project timeline was followed with sufficient time for development, testing, and documentation.

**4.4.1 Does the system contribute to the overall objectives of the organization?**

Yes, the **TechBazaar** system significantly contributes to the overall objectives of the organization by aligning with its core mission of delivering modern, user-friendly, and scalable e-commerce solutions—particularly focused on electronics.

**Alignment with Organizational Goals**

1. **Digital Transformation**

TechBazaar supports the organization’s goal of expanding into the digital marketplace by offering a fully functional and scalable online platform for electronic product sales.

1. **Improving User Experience**

The system enhances customer engagement through a responsive UI, easy navigation, real-time cart updates, and seamless checkout, fulfilling the organization's goal of delivering a smooth shopping experience.

1. **Simplifying Product Management**

With a robust admin panel, TechBazaar simplifies inventory management and product updates—reducing operational overhead and enabling better control for the organization.

**Practical Contributions**

* Facilitates **online sales** of electronics without the need for physical stores.
* Increases **brand visibility** through a modern web presence.
* Encourages **data-driven decisions** by storing structured product and customer data.
* Lays the foundation for future enhancements such as order tracking, multi-vendor support, and mobile app integration.

**4.4.2 Can the system be implemented using the current technology and within the given cost and schedule constraints.**

**Yes**, the **TechBazaar** system can be effectively implemented using the current available technologies and within the defined cost and schedule constraints.

**Technology Compatibility**

* The system was developed using the **MERN stack**, which is modern, open-source, and widely used for scalable web applications.
* All development and deployment tools such as **VS Code, GitHub, Postman, MongoDB Atlas, Vercel**, and **Render/Heroku** were readily available and supported by the current infrastructure.
* The technology stack used is fully compatible with both development and deployment environments, ensuring a smooth implementation process.

**Cost Constraints**

* The project utilized **free-tier plans** for hosting and databases, making it cost-effective.
* No additional hardware or paid software licenses were required.
* Estimated total cost remained minimal and within projected, which is affordable for small-to-medium-scale implementation.

**4.4.3 Can the system be integrated with other systems**

**Yes**, the **TechBazaar** system is built in a modular and flexible architecture, making it highly capable of being integrated with other systems and services as needed.

**Integration Possibilities**

1. **Payment Gateways**
   * The system can easily be integrated with services like **Razorpay**, **Stripe**, or **PayPal** to handle secure online transactions.
2. **Analytics & Reporting**
   * Integration with tools like **Google Analytics**, **Mixpanel**, or **Firebase Analytics** is possible to monitor user behaviour and site performance.
3. **Inventory or ERP Systems**
   * TechBazaar can connect with inventory or enterprise management software for real-time stock updates, reporting, and supplier management.

**4.5 Activity / Process in New System / Proposed System**

The **TechBazaar** system introduces a streamlined and modern e-commerce process that simplifies product browsing, cart management, and order placement for users, while also providing easy product and inventory control for administrators.

Customers can register, log in, browse electronic products, add items to their cart, and complete purchases through a secure checkout process. On the admin side, the system allows for efficient management of product listings, stock updates, and order tracking through a dedicated dashboard.

Each activity is handled through well-structured interfaces and backend processes, ensuring a smooth, responsive, and secure experience for both users and administrators.

**Key Activities for Users (Customers)**

1. **User Registration & Login**
   * Users can create an account or log in securely using their credentials.
   * Authentication is managed using tokens (JWT).
2. **Product Browsing**
   * Users can view a categorized list of electronics such as smartphones, laptops, and smartwatches.
   * Search and filtering options help narrow down results.
3. **Viewing Product Details**
   * Users can click on any product to view details including images, specifications, price, ratings, and availability.
4. **Add to Cart and Cart Management**
   * Items can be added to the shopping cart.
   * Users can update quantity, remove items, or view the total cost in real-time.
5. **Checkout and Payment**
   * Users proceed to checkout and complete the payment using an integrated payment gateway.
   * Confirmation messages are displayed after successful transactions.
6. **Ratings and Feedback**
   * After purchase, users can rate and review the products to help other customers.

**Key Activities for Admin**

1. **Login and Access Admin Dashboard**
   * Only authenticated admins can access backend controls.
2. **Product Management**
   * Admins can create, read, update, and delete products from the inventory.
   * They can update pricing, descriptions, and stock levels.
3. **Order Management**
   * Admins can view placed orders, update order status, and manage order history.
4. **User Management (Optional)**
   * Admins may review user accounts and their activity (depending on feature implementation).

**4.6 Features of New System / Proposed System**

The proposed system offers a range of modern features designed to enhance the user experience and simplify online electronics shopping. It includes a clean, responsive user interface, secure user authentication, real-time cart management, and smooth product browsing and checkout.

Administrators benefit from a dedicated dashboard to easily manage product listings, inventory, and orders. Additional features like product ratings, likes, and search filters improve functionality and user engagement. The system is scalable, mobile-friendly, and built with the latest web technologies for performance and reliability.

**4.7 List Main Modules / Components / Processes / Techniques of New System / Proposed System**

The system is developed using a modular and component-based structure to ensure scalability, maintainability, and ease of development. The main modules and techniques used in the system include:

* **User Module**: Handles user registration, login, authentication, and session management.
* **Product Module**: Manages product listings, details, categories, and search/filter functionalities.
* **Cart Module**: Allows users to add, update, and remove items in the shopping cart.
* **Order Module**: Processes user orders and manages order history and checkout flow.
* **Admin Module**: Enables administrators to perform product CRUD operations and manage inventory.
* **Authentication System**: Uses JWT (JSON Web Token) for secure login and role-based access control.
* **Frontend Component System**: Built using React.js and Tailwind CSS for a responsive UI/UX.
* **Backend API Services**: Developed using Node.js and Express.js for handling server-side logic.
* **Database Layer**: Uses MongoDB for storing user data, product info, and orders.

**4.8 Selection of Hardware / Software / Algorithms / Methodology / Techniques / Approaches and Justification**

Modern and efficient tools, techniques, and approaches were selected to ensure fast, scalable, and user-friendly performance. The **MERN stack** was chosen for its end-to-end JavaScript environment, enabling smooth integration between frontend and backend.

* **Software Tools**:
  + React.js for building dynamic frontend components.
  + Node.js and Express.js for server-side APIs.
  + MongoDB Atlas for cloud-based NoSQL database management.
  + Postman, GitHub, and VS Code for testing, collaboration, and development.
* **Algorithms & Techniques**:
  + JWT for secure authentication,
  + RESTful API design for modular communication between frontend and backend,
  + Responsive design techniques using Tailwind CSS for mobile-first UI.
* **Methodology**:
  + Agile-based development with task-based sprints and regular progress reviews.
* **Hardware Requirements**:
  + Standard development laptop/PC with internet access, minimum 8 GB RAM, and browser support.

**Justification**: These selections were made to ensure cost-effective development, platform independence, real-time performance, easy deployment, and simplified maintenance perfectly suited for a scalable e-commerce system like TechBazaar.

**Chapter 5: System Design**

**5.1 System Design & Methodology**

The system was designed using a **modular, component-based architecture** to ensure scalability, maintainability, and ease of development. The system is structured using the **MERN stack**, which allows seamless communication between the frontend, backend, and database.

The chosen **development methodology** is **Agile**, which supports iterative development, regular feedback, and continuous improvement. The design process included the creation of wireframes, database schemas, RESTful APIs, and reusable frontend components.

The system is divided into functional modules such as user management, product handling, cart operations, admin dashboard, and order processing each developed and tested independently before integration.

**5.2 Database Design / Data Structure Design / Circuit Design / Process Design / Structure Design**

The system uses a well-organized **database design and modular structure** to manage user data, products, and orders efficiently. The system is built using **MongoDB**, a NoSQL database, which allows flexible schema design suited for e-commerce platforms with varying product attributes.

* **Database Design**: Collections were created for users, products, orders, and cart data. Relationships are maintained through IDs and references for fast access and updates.
* **Data Structure Design**: JSON-based document structures are used to store dynamic data such as product specifications, user profiles, and cart items, which simplifies data handling.
* **Process & Structure Design**: The application is split into functional modules (user, admin, product, cart) using a **component-based architecture** in React for frontend and REST APIs in Node.js for backend.

**5.3 Input / Output and Interface Design**

The system features a user-friendly interface designed to ensure smooth interactions between the user and the application. Both the **input and output** designs focus on simplicity, responsiveness, and accessibility.

**Input Design:**

* **User Inputs**: Login credentials, product search queries, filter selections, cart updates, checkout forms, and product reviews.
* **Admin Inputs**: Product data entry, inventory updates, and content management via the admin dashboard.
* All input fields include validation to prevent errors and ensure data integrity.

**Output Design:**

* **User Outputs**: Product listings, detailed product views, cart summaries, order confirmations, and rating feedback.
* **Admin Outputs**: Dashboard data, product stats, user activity logs, and order details.
* Outputs are displayed dynamically using React components and updated in real time using API responses.

**Interface Design:**

* Built using **React.js and Tailwind CSS**, the UI is clean, responsive, and mobile-friendly.
* Navigation is intuitive, with clearly defined sections like Home, Products, Cart, and Admin Panel.
* The interface uses consistent colors, icons, and layout patterns to enhance the user experience.

**5.3.1 Samples of Forms, Reports and Interface**

In the context of system design, forms, reports, and interfaces are essential components that facilitate user interaction with the system, provide meaningful data presentation, and allow for input collection. These elements are crucial for ensuring that the system is both functional and user-friendly. Below is an overview of each component:

**1. Forms in System Design**

Forms are used to collect data from users or system inputs. A well-designed form ensures that the information collected is accurate, organized, and efficient for processing by the system. Forms are integral in many applications such as user registration, login, transaction processing, and feedback systems.

Key aspects to consider in form design:

* **Input validation**: Ensuring that the data entered by users is in the correct format.
* **User-friendly design**: Using clear labels, instructions, and intuitive layouts.
* **Data integrity**: Ensuring the correct capture of data without loss or inconsistency.

**Example**:  
In an e-commerce system, a **checkout form** is used to capture the user’s shipping address, payment details, and order confirmation.

**2. Reports in System Design**

Reports are structured presentations of data that allow stakeholders to review, analyze, and make informed decisions. In system design, reports are often generated from the system’s database and can vary in complexity—from simple summaries to detailed analytical reports.

Key considerations for report design:

* **Clarity and readability**: Reports must be easy to read and understand, using tables, graphs, and concise language.
* **Accuracy and relevance**: Ensuring that the report pulls the right data, and presents it in a manner that makes sense to the user.
* **Customization**: Allowing users to customize the report parameters, such as date ranges or categories, for more granular insights.

**Example**:  
In a financial system, a **monthly sales report** would summarize all transactions, showing revenue, number of items sold, and comparisons to previous months.

**3. Interfaces in System Design**

Interfaces define how users interact with the system. They are the visual and functional elements that facilitate communication between the user and the underlying system. A well-designed interface improves usability and ensures that the user can navigate and utilize the system efficiently.

Key elements to consider in interface design:

* **Usability**: The interface should be intuitive, responsive, and efficient, reducing the learning curve for users.
* **Consistency**: Interface elements such as buttons, colors, and icons should remain consistent throughout the system to provide a seamless experience.
* **Accessibility**: The interface should accommodate all users, including those with disabilities, by following accessibility guidelines.

**Example**:  
In a content management system, the **admin dashboard interface** allows the admin to manage users, content, and view system analytics.

**5.3.2 Access Control / Mechanism / Security**

**1. Access Control in E-commerce:**

Access control is used to manage who can access different parts of the system and what actions they can perform. It ensures that only authorized users have access to sensitive or restricted areas.

* **User Roles and Permissions**:
  + **Admin**: Has full access to manage products, orders, users, and other critical system functions.
  + **Customer**: Can browse products, make purchases, view their order history, etc.
  + **Guest**: Has limited access (e.g., browsing products, but cannot checkout without registration).

**2. Mechanisms in E-commerce Security:**

Security mechanisms refer to the methods used to protect the system and its data from unauthorized access or tampering. For e-commerce websites, this is crucial for protecting sensitive user information (e.g., personal data, payment information).

* **Authentication**: Ensures that users are who they say they are.
  + Examples: **Username/Password** combination, **Two-Factor Authentication (2FA)**, **Social Media Login**.
* **Authorization**: After authenticating, the system grants access to resources based on user roles.
  + Example: Only customers with successful payment should be able to view order details or download invoices.
* **Encryption**: Ensures that sensitive data (like payment details, passwords) is transmitted securely across the network.
  + Example: Use of **SSL/TLS** protocols to secure data during transmission (HTTPS).
* **Session Management**: Ensures users’ sessions are securely managed, including timeout and automatic logout for idle users.

**3. Security in E-commerce**

Security is a broad area and plays a vital role in an e-commerce project. It involves preventing various types of cyberattacks, protecting user data, and ensuring that transactions are safe.

* **Data Encryption**: E-commerce websites must encrypt sensitive customer data, such as credit card numbers, to prevent data breaches.
  + Example: Implement **AES encryption** to securely store payment information and personal details.
* **Secure Payment Gateways**: Use well-established and secure payment gateways (like **PayPal**, **Stripe**, **Razorpay**) for processing transactions, ensuring that the customer’s payment details are handled securely.
* **Account Protection**:
  + **Password Hashing**: Use algorithms like **bcrypt** to securely hash user passwords before storing them in the database.
  + **Multi-Factor Authentication (MFA)**: Implement MFA for admins and users to add an extra layer of security when accessing accounts or processing orders.
* **Backup and Recovery**: Implement regular backups of critical data and ensure you have a disaster recovery plan in place in case of a breach or failure.

**Chapter 6: Implementation**

**6.1 Implementation Platform / Environment**

**1. Hardware Environment**

The **hardware environment** defines the physical infrastructure needed to host and run the e-commerce application. For most modern e-commerce websites, this environment consists of cloud servers, network infrastructure, and databases.

**For example**:

* **Database**: For an e-commerce application, you'll need a relational or NoSQL database to store product details, user data, orders, etc.
  + **Relational Database**: **MySQL** or **PostgreSQL** for structured data storage.
  + **NoSQL Database**: **MongoDB** for flexible, scalable data storage.
* **Payment Gateway Integration**: If applicable, you will also integrate third-party payment services like **Stripe**, **PayPal**, or **Razorpay** for processing payments.

**Example Hardware Environment**:

* AWS RDS or MongoDB Atlas (Database)
* Stripe API (Payment Gateway)

**2. Software Environment**

The **software environment** defines the software tools, programming languages, and frameworks used to develop and run the application. This includes the frontend and backend frameworks, databases, APIs, and any tools you are using for deployment, testing, and monitoring.

**For example**:

* **Frontend Development**: Since you are building your e-commerce site with the **MERN stack (MongoDB, Express, React, Node.js)**, the frontend will likely be developed using:
  + **React.js**: JavaScript framework to build user interfaces (UI).
  + **CSS**: For styling the application.
  + **Redux** or **Context API**: For state management in React.
* **Backend Development**: The backend will handle user requests, database operations, and business logic.
  + **Node.js**: Server-side JavaScript runtime.
  + **Express.js**: Web framework to handle routing, middleware, and APIs.
* **Database**:
  + **MongoDB**: MongoDB atlas database to store product, order, and customer data. Alternatively, you can use a relational database like **MySQL** or **PostgreSQL** depending on the structure of your data.
* **Payment Gateway**:
  + **Stripe** **API**: For handling online payments securely.

**3. Development Tools and IDEs**

The development tools include any Integrated Development Environments (IDEs), text editors, and additional tools used during development.

**For example**:

* **IDE/Text Editor**:
  + **Visual Studio Code**: A popular code editor with support for JavaScript, React, Node.js, etc.
  + **WebStorm**: Another IDE for JavaScript development.
* **Version Control**:
  + **Git**: Version control system to manage code changes.
  + **GitHub**: Platform to host repositories, manage version control, and collaborate.
* **Package Manager**:
  + **npm** or **yarn** for managing dependencies in both frontend and backend.

**6.2 Process / Program / Technology / Modules Specification**

**1. Process Specification**

* **User Registration and Login Process:**
  + The user enters their email and password on the registration page.
  + The system checks if the email is unique. If so, it stores the user’s data, securely hashes the password, and creates an account.
  + Upon login, the system validates the email and password, then generates a JWT token that is used for subsequent requests to authenticate the user.
* **Product Ordering Process:**
  + The customer browses the product catalog and adds items to the shopping cart.
  + The system calculates the total cost, taxes, and shipping.
  + The user proceeds to checkout, where they provide payment and shipping information.
  + The system processes the payment via a payment gateway like **Stripe** or **PayPal**.
  + Once payment is successful, the system confirms the order, generates an invoice, and sends a confirmation email to the customer.
* **Admin Order Management Process:**
  + Admin users can view orders in the admin dashboard.
  + Admins can update the status of orders (e.g., processing, shipped, delivered).
  + The system sends automated email notifications to customers based on the status updates.

**2. Program Specification**

The **program specification** defines the key components of the e-commerce application. This includes the backend server, frontend user interface, and how the two interact. It also specifies the language and frameworks used for each component.

**Key Components:**

* **Backend**:
  + **Node.js**: Server-side JavaScript runtime environment that handles HTTP requests, serves data, and interacts with the database.
  + **Express.js**: Web framework built on Node.js for handling routes and HTTP requests.
  + **MongoDB**: NoSQL database to store products, orders, customer data, etc. You may also use **Mongoose** to interact with MongoDB in a structured way.
* **Frontend**:
  + **React.js**: A JavaScript library for building the user interface, allowing dynamic rendering of content like product listings, cart, and user profile.
  + **Redux** or **Context API**: Used for state management in React, keeping track of the shopping cart, user authentication status, and order details.
* **Payment Gateway Integration**:
  + **Stripe API**: To process payments securely. These APIs provide easy integration with the system for handling transactions and ensuring payment security.

**3. Technology Specification**

The **technology specification** describes the tools, libraries, and technologies used in your e-commerce project. These technologies are chosen based on the project requirements, scalability, security, and ease of development.

**Technologies Used:**

* **Frontend:**
  + **React.js**: JavaScript library for building user interfaces.
  + **Redux**: For state management, especially useful for managing the shopping cart, user data, and product list.
  + **CSS/SASS**: For styling the frontend components and ensuring a responsive design.
  + **Axios**: For making HTTP requests to the backend.
* **Backend:**
  + **Node.js**: JavaScript runtime environment for building the backend API.
  + **Express.js**: Framework for routing and handling HTTP requests.
  + **MongoDB**: NoSQL database for storing unstructured data (products, orders, users).
  + **JWT (JSON Web Tokens)**: For user authentication and secure session management.
  + **bcrypt**: To hash and store passwords securely.
* **Payment:**
  + **Stripe API**: To handle online payments securely.

**4. Module Specification**

Modules in an e-commerce system represent independent, reusable components that handle specific functionalities. Each module focuses on a particular feature and interacts with other modules to complete the system's operations.

**Key Modules:**

* **Authentication and User Management Module**:
  + **Login/Logout**: Allows users to log into their accounts and securely logout.
  + **Registration**: Facilitates user account creation.
  + **Profile Management**: Allows users to update their personal details, such as name, email, and password.
* **Product Catalog Module**:
  + **Product Listings**: Displays available products for customers to browse.
  + **Search and Filter**: Allows customers to search and filter products based on category, price, and other attributes.
  + **Product Detail**: Displays detailed information for each product, including images, descriptions, and prices.
* **Shopping Cart Module**:
  + **Add/Remove Items**: Allows users to add products to the cart and adjust quantities.
  + **View Cart**: Displays the current cart with items, total price, and any discounts.
  + **Cart Summary**: Displays a summary of the cart, including the subtotal, taxes, and shipping charges.
* **Order Management Module**:
  + **Checkout Process**: Handles the checkout, where users input shipping details and payment information.
  + **Order Confirmation**: Displays the order summary and confirmation details once payment is successful.
  + **Admin Order Management**: Admins can view and update order statuses (processing, shipped, delivered).
* **Payment Module**:
  + **Stripe Integration**: Processes payments securely and provides a seamless payment experience for customers.

**6.3 Finding / Results / Outcomes**

**1. Project Achievements and Successes**

**System Performance**:  
The e-commerce platform was successfully developed using the **MERN stack,** ensuring high scalability and performance. The use of **MongoDB** for data storage provided flexibility in managing product listings, customer information, and order data. The system’s response time has been optimized, ensuring a smooth user experience, even with a large volume of products and users.

**2. Challenges and Issues Encountered**

**Integrating Third-Party APIs:**  
Integrating Stripe and other third-party services like email notification services posed challenges due to the complex nature of their APIs and the need for precise error handling. Ensuring secure and correct payment processing was critical, and there were moments when API limitations or updates caused temporary delays.

**3. Outcomes and Benefits**

**Business Efficiency:**

The e-commerce platform automated many manual processes like order processing, payment handling, and inventory management, significantly reducing the workload for administrators and improving operational efficiency. This allowed for better scalability as business operations grew.

**6.4 Result Analysis / Comparison / Deliberations**

The **Result Analysis / Comparison / Deliberations** section of your internship report evaluates the outcomes of the e-commerce project in more depth. It involves comparing the actual performance of the system against the initial expectations, analyzing the results in the context of industry standards, and reflecting on any deliberations or decisions made during the development process. This section aims to provide insights into the success and areas for improvement based on the results of the project.

**Chapter 7: Testing**

**7.1 Testing Plan / Strategy**

The **Testing Plan / Strategy** for the e-commerce project outlines the structured approach taken to verify the system's functionality, performance, reliability, and security. The goal was to ensure that all modules of the system—from user registration to payment processing—worked correctly and met the intended requirements before deployment.

**1. Objectives of Testing**

* To ensure the system meets all functional requirements (e.g., login, product listing, cart operations, checkout).
* To identify and fix bugs or errors before deployment.
* To verify the security of user data and transaction processes.
* To validate system performance under varying loads.
* To confirm the application delivers a user-friendly experience.

**2. Testing Types Used**

**Functional Testing**:  
Focused on business-critical functions like user registration, adding items to the cart, placing orders, and processing payments.  
*Example*: Ensured correct redirection after successful payment.

**Integration Testing**:  
Verified data flow between front-end and back-end systems, such as product retrieval from the database.  
Tools Used: Supertest, Postman.

**7.2 Test Results and Analysis**

The **Test Results and Analysis** section presents the outcomes of the testing phase conducted on the e-commerce application. This includes a summary of the test cases executed, bugs discovered, performance observations, and overall system reliability based on the applied testing strategies.

**1. Key Observations**

* **High Pass Rate**: Most core features such as login, registration, product browsing, cart operations, and checkout worked flawlessly across browsers and devices.
* **Failed Cases**:
  + One failure in **Integration Testing** was due to an unexpected response format from a third-party payment API (Stripe).
  + One **End-to-End test** failed due to incorrect session expiration handling.
  + A **Security Test** failed due to lack of rate-limiting on the login API, which could allow brute-force attempts.

**2. Bug Reports**

* **Total Bugs Identified**: 12
* **Critical Bugs**: 2 (fixed immediately)
  + Payment gateway integration issue
  + Session expiration error on checkout
* **Medium Severity Bugs**: 5 (e.g., UI misalignment, alert not closing)
* **Low Severity Bugs**: 5 (e.g., minor spelling errors, tooltip issues)

**3. Performance Testing Results**

* **Load Testing**:
  + Simulated 100 concurrent users: system handled smoothly with average response time of **1.2 seconds**.
  + Simulated 500 users: response time increased to **2.7 seconds**, acceptable but slightly degraded UX.
* **Frontend Performance (Google Lighthouse)**:
  + Performance Score: **91/100**
  + Accessibility: **95/100**
  + Best Practices: **100/100**

**7.2.1 Test Cases (test ID, test condition, expected output, actual output,**

**remark)**

| **Test ID** | **Test Condition** | **Expected Output** | **Actual Output** | **Remark** |
| --- | --- | --- | --- | --- |
| TC01 | User registration with valid input | User account created and redirected to login page | As expected | Pass |
| TC02 | User login with correct credentials | User logged in and redirected to dashboard | As expected | Pass |
| TC03 | User login with incorrect password | Error message "Invalid Credentials" | As expected | Pass |
| TC04 | Add product to cart | Product added to cart and cart count updated | As expected | Pass |
| TC05 | Add same product twice to cart | Quantity of product increases in cart | As expected | Pass |
| TC06 | Checkout with valid address and payment | Order placed, payment successful, confirmation message | As expected | Pass |
| TC07 | Checkout with invalid payment details | Payment failed message shown | As expected | Pass |
| TC08 | Admin adds new product | Product appears in product list for users | Product not showing initially | Fail – Fixed |
| TC09 | Product search using keyword | Relevant products displayed | As expected | Pass |
| TC10 | View order history after successful order | Order listed under user order history | As expected | Pass |
| TC11 | Unauthorized access to admin panel | Redirect to login or error message | As expected | Pass |
| TC12 | Logout from user account | Redirect to home page and session cleared | As expected | Pass |
| TC13 | Load homepage under 100 users | Page loads within 2 seconds | Page loaded in 1.3 seconds | Pass |
| TC14 | Login API rate-limiting test | Block multiple failed login attempts after threshold | No rate limit implemented | Fail – Fixed |
| TC15 | Apply filters (price range/category) | Products displayed match filter criteria | As expected | Pass |

**Chapter 8: Conclusion and Discussion**

**8.1 Overall Analysis of Internship / Project Viabilities**

**8.2 Photographs and date of surprise visit by institute mentor**

**8.3 Dates of Continuous Evaluation (CE-I and CE-II)**

**8.4 Problem Encountered and Possible Solutions**

**8.5 Summary of Internship / Project work**

**8.6 Limitation and Future Enhancement**