IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Design and Implementation of TechZone

YAGMUR ÖZLER[1], (Fellow, MCBU), NURSENA ZENGIN[2], AND BILAL AYAKDAS, [3], (STUDENT, MCBU)

[1]Computer Engineering Department, Manisa Celal Bayar University, Yunusemre, Manisa 45000 TÜRKİYE(e-mail: 200316067@ogr.cbu.edu.tr)
[2]Computer Engineering Department, Manisa Celal Bayar University, Yunusemre, Manisa 45000 TÜRKİYE(e-mail: 230315007@ogr.cbu.edu.tr)
[3]Computer Engineering Department, Manisa Celal Bayar University, Yunusemre, Manisa 45000 TÜRKİYE (e-mail: 220316002@ogr.cbu.edu.tr)

ABSTRACT The development of an online technology store involves a comprehensive approach to meet user needs and business objectives effectively. This feasibility study evaluates the technical, operational, and economic aspects of creating a robust e-commerce platform. The system encompasses user-friendly features, including product categorization, advanced filtering, shopping cart management, secure checkout, and order tracking, designed to enhance the customer experience. The backend leverages Spring Boot, PostgreSQL for database management, and JUnit for testing, while the frontend incorporates HTML, CSS, Bootstrap, and jQuery/AJAX for responsive and dynamic interfaces. External integrations with payment gateways, logistics partners, and email services ensure streamlined operations and regulatory compliance. Employing the KANBAN methodology for agile project management, the platform ensures efficient workflows and timely delivery.

The requirements engineering process, covering elicitation, analysis, validation, and management, establishes a robust foundation, with attention to functional and non-functional requirements such as performance, scalability, and security. Challenges such as stakeholder alignment and technical feasibility are addressed through structured processes and tools like GitHub for version control and Postman for API testing. Future enhancements include advanced search capabilities, personalized recommendations, and additional security measures. The project delivers a scalable, accessible, and user-centric platform, leveraging modern technologies to meet market demands and regulatory standards.

This study provides a roadmap for successful implementation while identifying potential improvements to ensure long-term sustainability and user satisfaction.

INDEX TERMS Online Technology Store, E-commerce Platform, User Experience, Product Categorization, Shopping Cart Management, Secure Checkout, Order Tracking, Spring Boot, PostgreSQL Database, Responsive Design, Bootstrap, jQuery/AJAX, Agile Project Management, KANBAN Methodology, Requirements Engineering, Functional Requirements, Non-Functional Requirements, System Integration, Payment Gateway, Logistics Management, Email Notifications, Scalability, Performance Optimization, Security Compliance, GDPR, Accessibility, Usability, Personalization, System Monitoring, Future Enhancements.

I. FEASIBILITY STUDY AND PLAN

A. INTRODUCTION

The purpose of this feasibility study is to assess whether the development of an online technology store is technically, economically, and operationally feasible within the proposed timeline and budget. By identifying potential challenges and risks, this document aims to provide a roadmap for mitigating issues and ensuring project success.

The online technology store will include the following features:

Category-based Browsing: Users will be able to browse products by categories such as computers, mobile phones, and other technology items.

Search and Filter Functionality: Advanced search tools will help customers locate products quickly and apply filters based on brand, price, and specifications.

Shopping Cart Management: Customers can add, remove, or update items in their shopping cart before finalizing their purchase.

Wishlist (Favorites): Users will have the ability to save products for later consideration by marking them as favorites.

Order History and New Orders: The platform will allow customers to view their previous orders and place new orders seamlessly.

B. TECHNICAL FEASIBILITY

Frontend Technologies: The frontend will be developed using HTML, CSS and Javascript for the core structure and styling, ensuring the platform is visually appealing and intuitive to use. Bootstrap will be incorporated to provide responsive design elements, enabling the application to function efficiently across devices of various screen sizes. Additionally, jQuery/AJAX will be used to dynamically render content and handle user interactions without requiring full-page reloads, enhancing the user experience.

Backend Technologies: The backend of the application will be powered by a Java-based API, ensuring reliability and scalability for handling business logic and data persistence. RESTful APIs will be used to facilitate seamless communication between the frontend and backend systems. The backend will also incorporate secure and optimized database queries to manage product and customer data effectively. It will depend on the PostgreSQL database.

Testing Framework: JUnit will be employed as the testing framework for the backend API. This framework ensures that backend functions, such as user authentication, shopping cart updates, and order processing, work as intended by enabling developers to write unit tests and validate the API's reliability and robustness. A separate test will be applied for each function.

Challenges: Some technical challenges that might arise include ensuring seamless integration between the frontend and backend components, managing concurrency issues when multiple users interact with the shopping cart simultaneously, and maintaining system performance under high traffic.

Proposed Solutions: To overcome these challenges, the development team will adopt RESTful API principles to ensure modular and scalable communication between the frontend and backend. Concurrency issues will be addressed by implementing locks or queues to manage simultaneous updates to shared resources like the shopping cart. Performance optimization will be achieved by using caching mechanisms and efficient database queries.

C. OPERATIONAL FEASIBILITY

User Perspective: From the user's perspective, the application will cater to technology enthusiasts seeking a convenient online shopping experience. The platform will prioritize ease of use, fast loading times, and reliable order fulfillment to ensure customer satisfaction.

Development Team: The development team will consist of frontend developers responsible for building the user interface, backend developers managing server-side logic and APIs, and test developers conducting rigorous testing to ensure the application's reliability and performance.

Project Management: KANBAN methodology will be adopted for managing project tasks. KANBAN provides a visual framework that helps the team prioritize tasks, track progress, and manage workloads efficiently. KANBAN allows for continuous improvement by encouraging regular reflection on the process. After each milestone, the team will conduct retrospectives to evaluate what worked well and what could be improved, ensuring a dynamic and adaptive project management approach. By using the KANBAN methodology effectively, the project team aims to maintain a steady workflow, deliver features incrementally, and meet project deadlines without sacrificing quality. GitHub will serve as the central platform for version control and collaboration, enabling team members to work on different components of the project simultaneously while avoiding conflicts.

D. SCHEDULE FEASIBILITY

Weeks 1-2: Gather detailed requirements, design wireframes, and prepare technical documentation.

Weeks 3–5: Develop the frontend interface and backend API.

Weeks 6–8: Integrate frontend and backend components, followed by thorough testing to identify and resolve bugs.

Week 9: Deploy the application to the production environment and collect user feedback for further enhancements.

E. ECONOMIC FEASIBILITY

The development of the online technology store involves several cost components:

Hosting Costs: Hosting costs are estimated depending on the server's capacity and additional features.

Maintenance Costs: Regular maintenance activities such as bug fixes, feature updates, and performance optimizations are expected to incur.

Expected Revenue: The primary revenue streams for the online technology store will include direct product sales and potential partnerships with brands for featured listings and advertisements. By providing a user-friendly shopping experience and a wide range of products, the platform aims to attract and retain a large customer base, thereby maximizing revenue potential.

F. CONCLUSION

The feasibility study has demonstrated that the development of the proposed online technology store is both viable and practical from technical, economic, and operational perspectives. By leveraging modern technologies and tools, the project has the potential to deliver a robust and user-friendly platform that aligns with current market demands and customer expectations.

From a technical standpoint, the use of HTML, CSS, Javascript and Bootstrap for frontend development ensures that the platform will have an appealing and responsive design, suitable for various devices and screen sizes. jQuery and AJAX provide a seamless user experience by enabling dynamic content updates without requiring page reloads. On the backend, a Java-based API ensures scalability, reliability, and efficient handling of business logic. The integration of Postman for API testing and JUnit for backend unit testing

further ensures that the system will be rigorously evaluated for quality and reliability before deployment.

Economically, the project has been assessed to be feasible within the defined budget constraints. Initial development costs are expected to include resources for developers, testers, and project management. Ongoing expenses such as hosting and maintenance have also been considered and are within a manageable range. The platform is expected to generate revenue through direct product sales and potential partnerships with technology brands for advertisements and featured listings. With a growing demand for online shopping platforms, especially in the technology sector, this project presents a significant opportunity for sustained revenue generation.

Operationally, the project is well-supported by the adoption of KANBAN as the project management methodology and GitHub as the collaborative platform for task tracking and version control. KANBAN allows for clear visualization of tasks and their progression through various stages, while GitHub ensures streamlined communication and version management. The development team, composed of frontend developers, backend developers, and test developers, is well-equipped to address the challenges of integrating multiple features and ensuring a seamless user experience.

In conclusion, the feasibility study supports the decision to proceed with the development of the online technology store. The project has a clear roadmap for success, including a well-defined timeline, robust technical foundation, and a comprehensive management strategy. By adhering to the proposed plan and continuously improving based on stakeholder feedback, the project team is poised to deliver a high-quality product that meets both business objectives and user expectations.

II. REQUIREMENTS

A. FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirements: Functional requirements describe the key features and functionalities the system must provide to meet user and business needs.

1) User Management

User Registration: Users should be able to create an account with basic details (e.g., name, email, password).

Login/Logout: Users must be able to log in securely using their credentials and log out when done. Profile Management: Users should be able to update their personal information.

2) Product Catalog

Category Browsing: Users must be able to browse products by categories (e.g., Computers, Mobile Phones, Televisions).

Search Functionality: Users should be able to search for products using keywords. Filter and Sort: Products must be filterable by brand, price, and specifications.

3) Shopping Cart

Add to Cart: Users should be able to add products to their shopping cart.

Edit Cart: Users must be able to update quantities or remove items from the cart.

Cart Summary: A summary page showing total cost and selected items should be displayed.

4) Order Management

Place Orders: Users should be able to place orders for the items in their shopping cart. **Order History:** Users must be able to view a list of past orders. **Order Details:** Each order should display the purchase date, items, and total cost.

5) Favorites

Save Products: Users should be able to mark products as favorites.

View Favorites: Users must be able to view all saved products in a separate section.

6) Administrator Features

Product Management: Admins should be able to add, edit, or remove products. Order Tracking: Admins must be able to monitor and manage customer orders.

Non-Functional Requirements: Non-functional requirements define the system's operational standards and constraints.

7) Performance

Response Time: The system should load product search results within 5 seconds under normal load conditions. Checkout operations should complete within 10 seconds after the payment is initiated. Scalability: The system must support at least 10,000 concurrent users during peak shopping hours.

8) Security

Data Encryption: All sensitive user data (e.g., passwords, payment information) must be encrypted using AES-256 standards. PCI Compliance: Payment processes must adhere to the Payment Card Industry Data Security Standards (PCI DSS).

9) Usability

Responsive Design: The user interface should be fully functional across devices with varying screen sizes, including desktops, tablets, and smartphones.

10) Maintainability

Modularity: The system must follow a modular architecture to simplify updates and debugging. Documentation: Comprehensive technical documentation must be provided for future maintenance and enhancements.

11) Compliance

GDPR: The system must comply with General Data Protection Regulation (GDPR) for user data protection. Legal

Compliance: All e-commerce operations must comply with local and international trade laws. metrics to quantify non-functional goals:

Requirement	Metric	Target Value
Response Time	Time to load search results	≤ 5 seconds
Scalability	Concurrent users supported	$\geq 10{,}000$
Security	Encryption standard used	AES-256
Uptime	Percentage of operational time	$\geq 99.9\%$
Accessibility	WCAG compliance level	WCAG 2.1 AA
Backup Frequency	Number of backups per day	1 (daily)

TABLE 1. System Requirements and Metrics

B. USER REQUIREMENTS

To ensure the successful implementation of the online technology store, it is essential to define the requirements from the perspective of different user groups. This ensures that the platform addresses the needs and expectations of all stakeholders, including customers, administrators, and technical staff.

1) Customer Requirements

Browsing and Product Discovery

- Customers should be able to browse through products categorized under labels such as Computers, Mobile Phones, and Televisions.
- Users must have access to search functionality with support for keyword-based searches.
- Filters and sorting options (e.g., by brand, price range, and specifications) should help users refine their product search results.

Product Details

- Each product page should display key information, including name, description, price, specifications, and availability status.
- Users should be able to view product images with options for zoom.

Shopping Cart and Favorites

- Users must be able to add multiple items to their shopping cart, view the cart summary, and update item quantities.
- The platform should provide a favorites feature where customers can save products for later review.
- Customers should be notified when items in their favorites go on sale or are about to go out of stock.

Order Placement and Tracking

- Users should be able to place an order by completing a secure checkout process.
- Payment methods should include credit/debit cards, and cash on delivery.
- After placing an order, customers must receive confirmation with a unique order ID.
- Users should be able to track their order status (e.g., processing, shipped, delivered).

Order History

- Customers must have access to a detailed history of their previous orders, including itemized bills, order dates, and payment methods.

User Accounts

- Customers should be able to create accounts with personal details (name, email, address, and phone number).
- Users must be able to update their profile information and shipping address.
- Password recovery options must be available through email or phone number verification.

Customer Support

- Users should have access to customer support via email, or phone for resolving issues related to orders or payments.

2) Administrator Requirements

Product Management

- Administrators must be able to add, edit, or remove products from the inventory.
- Bulk upload functionality for adding multiple products at once should be available.
- Admins must be able to set product pricing, discounts, and availability status.

Order Management

- Administrators should have access to a dashboard that provides a real-time view of all orders.
- They must be able to update order statuses (e.g., processing, shipped, delivered) and handle cancellations or returns.

User Management

- Admins must be able to view and manage customer accounts, including the ability to deactivate accounts in cases of fraudulent activity.

Notifications

- Admins must receive alerts for low inventory levels to ensure products are restocked promptly.

3) Technical Staff Requirements

System Monitoring

- The technical team should have access to monitoring tools to ensure the system's performance and uptime.
- Alerts for system errors, downtime, or database issues should be configured.

Maintenance and Updates

- Developers should be able to deploy updates (e.g., new features, bug fixes) without causing downtime.
- The system should support modular development to make future enhancements easier.

Data Backup and Recovery

- Technical staff must have access to recovery tools to restore the system in case of failures. Security

- The technical team must ensure compliance with data protection laws like GDPR.
- Tools for monitoring potential threats (e.g., unauthorized access, data breaches) should be available.

C. GENERAL REQUIREMENTS FOR ALL USERS

Performance

- The platform must respond quickly under normal load conditions (e.g., search results within 5 seconds).

By addressing these user requirements, the project will meet the expectations of all stakeholders, ensuring usability, functionality, and scalability.

D. SYSTEM REQUIREMENTS

System requirements focus on system properties, while stakeholder requirements focus on stakeholder needs. System requirements are stated in engineering language. System requirements tend to be more quantitative than their corresponding stakeholder requirements.

1) Functional System Requirements

These requirements define the specific behaviors and functionalities of the system. **Browsing and Product Discovery**

- The system shall categorize products into predefined labels such as "Computers," "Mobile Phones," ,"Televisions", "Home", and "Personal,", dynamically fetched from the database.
- The system shall implement a search functionality capable of processing keyword-based searches with a response time of less than 5 seconds.
- The system shall support filtering options based on category, price and brand implemented through database queries and indexed searching.

Product Details The system shall fetch and display filtered products, including:

- Image
- Name
- Price
- Add to cart
- Add to favorites

When a user selects a product from the product listing, the system shall open a dedicated product detail page, which displays the following information:

- A large, high-resolution image of the product.
- Product name.
- Product price.
- Detailed product description (e.g., features, specifications).
- Current stock status (e.g., "In Stock," "Out of Stock").
- Product category (e.g., "Computers," "Mobile Phones").
- Product brand (e.g., "Apple," "Samsung").
- Two actionable buttons:
  - Add to Cart: Adds the product to the shopping cart.
  - Add to Favorites: Saves the product to the user's favorites for future reference.

The product detail page shall ensure all fields dynamically update based on the database entry for the selected product, ensuring accuracy and consistency.

The system shall support high-resolution product image uploads (up to 10 MB per image) with a zoom-in feature implemented using JavaScript libraries.

Shopping Cart and Favorites

The system shall allow users to add, remove, and update quantities of products in their shopping cart, ensuring that all updates reflect in real-time.

A favorites feature shall allow users to save products, which shall be stored in the database and retrievable in under 5 seconds.

The system shall generate notifications for users (via email or in-app alerts) when items in their favorites:

- Are nearing stock depletion.

Order Placement and Tracking

- The system shall support a secure checkout process that includes order review, payment options, and order confirmation steps.
- The system shall integrate with third-party payment gateways for:
  - Credit/Debit card payments.
  - Cash-on-delivery options.
- Upon successful order placement, the system shall generate a unique order ID, which shall be sent to the user via email and displayed in the user interface.
- The system shall provide an order-tracking feature that dynamically updates statuses such as "Processing," "Shipped," and "Delivered."

Order History

- The system shall maintain a detailed history of each user's orders, including:
  - Product details.
  - Itemized bills.
  - Payment methods.
  - Purchase dates.

User Accounts

- The system shall allow users to register accounts by providing personal details such as name, email, address, and phone number.
- The system shall enable users to update their profile information and shipping addresses.
- A password recovery mechanism using email or phone number verification shall be implemented to assist users in resetting their passwords securely.

Customer Support

- The system shall provide customer support access via email and phone numbers displayed on a dedicated support page.

E. ADMINISTRATOR SYSTEM REQUIREMENTS

1) Product Management

- The system shall include an administrative interface that allows administrators to:

  - Add, edit, or delete product records from the inventory.
  - Upload product details in bulk with data validation.
- The system shall allow administrators to configure:
  - Product pricing.
  - Availability status.

2) Order Management

- The system shall provide a real-time order management dashboard where administrators can:
  - View all active, pending, and completed orders.
  - Update order statuses to reflect real-time progress.
  - Process order cancellations and returns.

3) User Management

- Administrators shall have tools to:
  - View customer accounts and their order history.
  - Deactivate accounts flagged for suspicious or fraudulent activity.

4) Notifications

- The system shall generate low-inventory alerts for administrators when stock levels fall below a predefined threshold, which is 10.

F. NON-FUNCTIONAL SYSTEM REQUIREMENTS

1) Performance

- The system shall ensure a response time of less than 5 seconds for product searches, filtering, and cart operations under normal load conditions.
- The system shall support a peak user load of at least 10,000 concurrent users with no performance degradation.

2) Security

- All sensitive user data, including passwords and payment information, shall be encrypted using AES-256 encryption standards.
- User authentication shall follow OAuth 2.0 protocols to ensure secure login and session management.
- The system shall comply with GDPR and other applicable data protection laws, ensuring user data is securely stored and processed.

3) Usability

- The system shall adhere to WCAG 2.1 AA guidelines to ensure accessibility for users with disabilities.
- The system interface shall support a fully responsive design, optimized for desktops, tablets, and smartphones.

4) Availability

- The platform shall maintain an uptime of 99.9%, with downtime limited to scheduled maintenance windows.
- A disaster recovery plan shall include daily backups of critical data, stored securely in a cloud environment.

5) Maintainability

- The system shall adopt a modular architecture to enable future updates without affecting the core functionalities.
- All system modules shall include documentation for developers, detailing APIs, database structures, and code design.

6) Monitoring and Recovery

- The system shall integrate with performance monitoring tools to track server health, response times, and resource usage.
- Recovery tools shall enable administrators to restore the system to a previous state in case of hardware or software failure.

Measurable Metrics To validate compliance with the system requirements, the following metrics will be used:

Requirement	Metric	Target Value
Search Response Time	Seconds to display search results	5 seconds
Concurrent User Load	Maximum supported users	10,000 users
Uptime	Percentage of operational time	99.9%
Backup Frequency	Daily backups	1 backup per day
Data Encryption	Encryption standard	AES-256

TABLE 2. Measurable Metrics for System Requirements

This detailed system requirements document ensures that the stakeholder needs are translated into clear, measurable, and actionable technical specifications, which will guide the development process.

G. SYSTEM STAKEHOLDERS

Stakeholders are individuals or entities with an interest in the development, operation, or outcomes of the online technology store. Projects need to know the roles involved and the viewpoints of the stakeholders playing those roles [**?**].

1) Internal Stakeholders

Agile Coach

Role: The Agile Coach plays a vital role in ensuring that the team adopts and adheres to Agile principles, practices, and workflows. In this project, the Agile Coach is Didem Abidin.

Responsibilities:

- Guide the team in understanding and implementing the KANBAN framework effectively, focusing on workflow visualization and limiting work in progress (WIP).
- Continuously improve processes by facilitating retrospectives and suggesting actionable improvements.
- Help identify and eliminate bottlenecks in the development workflow.
- Foster a culture of collaboration and continuous delivery.
- Coach team members on Agile principles, such as customer-centric thinking, adaptability, and incremental delivery.

- Ensure that stakeholders outside the team (e.g., management, funders) align with Agile practices and expectations.

Product Owner

Role: The Product Owner is the voice of the customer and business. They are responsible for defining the product vision and ensuring that the team focuses on delivering value to the end-users and stakeholders.

Responsibilities:

- Maintain a well-defined Product Backlog, including tasks, features, and bug fixes prioritized based on business value and user needs.
- Clearly communicate requirements and expectations to the development team, ensuring there is no ambiguity.
- Work closely with the development team to break down high-level requirements into smaller, actionable tasks.
- Continuously refine and prioritize the Product Backlog to ensure that the team is working on the most valuable tasks at any given time.
- Act as a liaison between the customer and the development team, gathering feedback and translating it into actionable improvements.
- Make decisions on feature trade-offs to balance the needs of customers and business goals.

Development Team

Role: The Development Team is a cross-functional group responsible for the creation, testing, and delivery of the product. In this project, the team consists of:

- **Bilal Ayakdaş:** Responsible for backend design.
- **Yağmur Özler:** Responsible for frontend design.
- **Nursena Zengin:** Responsible for testing.

Responsibilities:

- Develop and implement features defined in the Product Backlog, ensuring they meet the acceptance criteria.
- Collaborate on the design, coding, testing, and deployment of system components.
- Regularly update the KANBAN board to reflect the current status of tasks (e.g., Backlog, In Progress, Testing, Done).
- Participate in daily stand-ups to discuss progress, challenges, and next steps.
- Test features at each stage of development to ensure quality and functionality.
- Deliver working software in small, incremental releases.
- Actively participate in retrospectives to identify ways to improve teamwork and processes.

Project Manager

Role: The Project Manager coordinates high-level activities and external dependencies, despite Agile minimizing traditional project management roles.

Responsibilities:

- Oversee the overall project roadmap and ensure alignment with company goals.
- Manage resource allocation, budget constraints, and timelines.
- Handle external dependencies, such as communication with suppliers, funders, or logistics partners.
- Ensure that project deliverables meet stakeholder expectations and quality standards.

Company/Organization

Role: The organization provides resources and infrastructure needed to develop, deploy, and maintain the system.

Responsibilities:

- Define the business objectives and desired outcomes for the online store.
- Provide tools, technologies, and support for the Agile team.
- Ensure alignment between the project and overall organizational strategy.
- Support ongoing maintenance and operations post-deployment.

2) External Stakeholders

Customers

Role: Customers are the end-users of the system, providing critical input for the requirements and validating the product through feedback.

Responsibilities:

- Provide initial requirements and feedback during the requirements elicitation phase.
- Validate features and functionality through user testing (e.g., beta testing).
- Share insights on user experience, helping the Product Owner refine the backlog.
- Use the system for browsing, purchasing, and managing orders.

Suppliers

Role: Suppliers provide the products and information needed for the platform.

Responsibilities:

- Ensure accurate and up-to-date information about inventory, pricing, and specifications.
- Collaborate with administrators to restock inventory as required.

Government and Regulators

Role: Regulatory bodies ensure compliance with laws and policies governing e-commerce operations.

Responsibilities:

- Enforce legal requirements such as GDPR for data protection and PCI DSS for payment security.
- Approve and monitor taxation, trade regulations, and other legal aspects of the platform's operation.

Payment Gateway Providers

Role: Third-party services facilitating secure payment processing for the platform.

Responsibilities:

- Provide APIs for credit/debit card payments and other methods like digital wallets.
- Ensure compliance with financial regulations and security standards.

Logistics Partners

Role: Handle the delivery of orders from the platform to the customers.

Responsibilities:

- Provide integration for real-time tracking of shipments.
- Ensure timely delivery of orders and update order status in the system.

H. COMPLETENESS AND CONSISTENCY OF REQUIREMENTS

The completeness and consistency of the requirements are foundational to the successful development of the online technology store. Analyzing these aspects reveals that the outlined requirements largely address the functional, non-functional, and stakeholder-specific needs of the system. However, careful examination highlights areas of strength as well as opportunities for refinement, ensuring that all system expectations align cohesively.

1) Completeness of Requirements

The requirements demonstrate a robust level of completeness by thoroughly covering both functional and non-functional elements necessary for the system's operation. For instance, the functional requirements effectively outline key user-facing features, such as browsing products through categories like Computers and Mobile Phones, enabling users to perform keyword-based searches, and refining their results using filters for price range, brand, and specifications. These functionalities directly address the primary needs of the customers, ensuring a seamless product discovery experience.

Moreover, the requirements extend beyond basic browsing and incorporate detailed specifications for the shopping cart and favorites features. Users can add multiple items to their cart and update quantities. The favorites functionality allows users to save preferred items. The requirements also account for post-purchase experiences by providing access to a detailed order history, ensuring that users can review their past transactions. These features together create a comprehensive journey for the customer, from product discovery to order completion.

From an administrative perspective, the requirements address the needs of platform managers by including functionalities like product inventory management, order monitoring, and customer account control. Administrators are equipped with tools to add, edit, or remove products, manage discounts and pricing, and monitor inventory levels, ensuring smooth operational workflows. Similarly, non-functional requirements, such as performance benchmarks, security protocols, and scalability to handle up to 10,000 concurrent users, add another layer of completeness by setting clear technical expectations for the system.

2) Consistency of Requirements

The requirements are not only complete but also demonstrate a high level of consistency, with well-aligned functionalities that avoid contradictions and ambiguities. For example, the customer-facing requirements, such as the need to display real-time product availability and stock status, are seamlessly integrated with the administrator-facing requirements for managing inventory and updating product data. This ensures that both user roles operate within a unified framework, enhancing the platform's reliability and reducing the likelihood of conflicts.

Furthermore, the adoption of Agile methodologies, particularly the KANBAN framework, reinforces the consistency of the requirements by emphasizing incremental delivery and iterative improvements. The clearly defined tasks, such as developing the shopping cart feature or implementing search filters, can be broken down and visualized on the KANBAN board, ensuring that all functionalities are developed systematically without overlapping responsibilities.

Conclusion

Overall, the requirements exhibit a strong degree of completeness and consistency, covering the major functional and non-functional aspects required for the successful implementation of the online technology store. They address the needs of all stakeholders—customers, administrators, and technical staff—while setting clear performance, security, and usability benchmarks. Nonetheless, addressing edge cases like downtime behavior, enhancing accessibility, and refining overlapping functionalities would further strengthen the requirements. By ensuring that these areas are incorporated, the system can achieve a seamless, reliable, and inclusive user experience while maintaining alignment with stakeholder goals and technical feasibility. This holistic approach lays a solid foundation for development and ensures the project's success.

I. USABILITY REQUIREMENTS

Usability requirements ensure that the online technology store provides a user-friendly experience, is easy to navigate, and meets the expectations of diverse users. These requirements focus on accessibility, intuitiveness, responsiveness, and user satisfaction.

1) General Usability Requirements

- **Ease of Navigation:** The platform must provide a simple, intuitive navigation menu that allows users to access key sections such as "Home," "Categories," "Shopping Cart," "Favorites," and "Order History" within three clicks from any page.

- **Consistent Design Language:** The user interface must maintain a consistent design style across all pages, including fonts, colors, buttons, and icons, to reduce cognitive load and enhance familiarity. Call-to-action buttons (e.g., "Add to Cart," "Place Order") must be clearly distinguishable and follow a uniform color scheme throughout the platform.

2) Accessibility Requirements

- **Compliance with WCAG 2.1 Guidelines:** The platform must adhere to WCAG 2.1 AA accessibility standards to ensure usability for individuals with disabilities.
  - Provide text alternatives for all non-text content (e.g., product images, buttons).
  - Enable keyboard navigation for users unable to use a mouse.

3) Search and Discovery Usability

- **Search Functionality:** The search bar must be prominently displayed on all pages and support keyword-based searches.
- **Filtering and Sorting:** Filters for brand, price, specifications, and availability must be easy to use, with real-time updates reflecting applied filters. Sorting options (e.g., "Price: Low to High," "Newest Arrivals") must be accessible through dropdown menus and should retain the user's preferences during browsing.

4) Shopping Cart and Purchase Flow Usability

- **Simplified Checkout Process:** The checkout process must follow a clear, linear flow with the following steps:
  - Review cart.
  - Enter shipping details.
  - Select payment method.
  - Confirm order.

  Users must be able to edit their cart or shipping details without restarting the checkout process.
- **Progress Indicators:** A progress bar or step indicator must be displayed during multi-step processes, such as checkout, to inform users of their current stage and remaining steps.
- **Payment Confirmation:** After placing an order, users must receive a confirmation message on the platform and via email with details such as order ID, items purchased, and estimated delivery date.

5) Personalization and Engagement

- **Favorites Management:** Users must be able to save products to a favorites list, which should persist across sessions. Notifications (e.g., price drops, restocks) for favorite items should be easy to enable and disable.

By incorporating these usability requirements, the online technology store ensures a seamless, efficient, and inclusive user experience that meets the expectations of all stakeholders while maintaining high engagement and customer satisfaction.

J. REQUIREMENTS ENGINEERING PROCESS

The requirements engineering process involves a structured approach to identify, analyze, validate, and manage the requirements for a system. This ensures the development of a high-quality product that meets the needs of all stakeholders. Below is a detailed explanation of the four stages: Elicitation, Analysis, Validation, and Management.

1) Requirements Elicitation

Requirements elicitation is the process of gathering information about the needs and expectations of stakeholders for a software system. This stage focuses on engaging stakeholders through various techniques to ensure all requirements are captured comprehensively.

2) Stakeholder Identification and Engagement

- Identify all internal and external stakeholders, such as customers, administrators, technical staff, Agile roles (product owner, Agile coach, development team, and project manager), suppliers, and regulatory authorities.
- Ensure active participation of stakeholders during elicitation to capture diverse perspectives and needs.

3) Techniques for Gathering Requirements

- **Interviews:** Conduct structured interviews with customers to understand their expectations for features like product browsing, payment options, and order tracking.
- **Workshops:** Organize collaborative workshops with administrators and technical staff to identify operational and technical requirements, such as inventory management and system scalability.
- **Surveys and Questionnaires:** Distribute surveys to potential customers to gather data on their shopping habits, preferred features, and usability expectations.
- **Observation:** Observe existing e-commerce platforms to identify best practices and common pain points, ensuring that the new system addresses these effectively.
- **Document Analysis:** Analyze existing documentation, such as business plans, compliance regulations, and stakeholder guidelines, to ensure the requirements align with organizational goals and legal standards.

4) Output of Elicitation

A preliminary list of functional, non-functional, and usability requirements is created, providing a foundation for further refinement during analysis.

K. REQUIREMENTS ANALYSIS

Requirements analysis involves refining, prioritizing, and ensuring the feasibility of the gathered requirements. This stage focuses on resolving conflicts, defining dependencies, and ensuring the requirements are clear and achievable.

1) Categorizing Requirements

- **Functional Requirements:** Core system functionalities, such as browsing products, adding items to the cart, managing favorites, placing orders, and tracking deliveries.
- **Non-Functional Requirements:** Define performance, security, scalability, and operational constraints.
- **Usability Requirements:** Ensure a seamless and intuitive user experience, including accessibility and responsiveness.

2) Constraints

- **Technical Constraints:** The technologies to be used have been determined in advance:
  - Frontend: HTML, CSS, Bootstrap, and jQuery/AJAX.
  - Backend: A Java-based API.
  - Database: PostgreSQL.
- **Time Constraints:** The project must be completed within the established timeline.

3) Feasibility Analysis

- Assess technical feasibility by evaluating whether the system can support complex features, such as real-time inventory updates or integration with third-party payment gateways.
- Evaluate economic feasibility by estimating development costs, hosting expenses, and ongoing maintenance to ensure alignment with the allocated budget.

4) Conflict Resolution

- Identify conflicts among requirements, such as customer demands for real-time updates versus technical staff concerns about performance issues.
- Engage stakeholders to resolve conflicts by balancing expectations with feasibility.

5) Output of Analysis

A refined and categorized list of functional, non-functional, and usability requirements ready for implementation.

L. REQUIREMENTS VALIDATION

This stage ensures that the refined requirements meet stakeholder expectations and align with the project's goals. Validation reduces the risk of errors, ambiguities, and misunderstandings during development.

1) Prototyping for Validation

- **Low-Fidelity Prototypes:** Develop wireframes or mockups for critical features such as the product listing page, product detail page, and checkout process.

2) Requirement Reviews

Conduct formal reviews with stakeholders to ensure requirements are clear, feasible, and aligned with business goals.

3) Validation Criteria

- **Validity:** Confirm that requirements represent stakeholder needs and align with project objectives.
- **Completeness:** Ensure all critical requirements are included.
- **Consistency:** Verify that requirements align with one another and do not conflict.
- **Verifiability:** Ensure each requirement can be verified through testing or demonstrations.
- **Realism:** Validate that all requirements are feasible within the constraints.

4) Output of Validation

A validated requirements document with stakeholder approval, ready for development.

M. REQUIREMENTS MANAGEMENT

Requirements management ensures the requirements remain valid and relevant throughout the project lifecycle.

1) Version Control and Traceability

- Use tools like GitHub to maintain version-controlled repositories of requirements.
- Link requirements to their sources and related artifacts for traceability.

2) Change Management

- Document new requirements or modifications.
- Evaluate the impact of changes on timelines and budgets.
- Seek stakeholder approval before incorporating changes.

3) Monitoring Requirements Implementation

- Use traceability matrices to track requirements from design to deployment.
- Assign requirement owners to ensure successful implementation.

N. POSSIBLE PROBLEMS IN REQUIREMENTS ENGINEERING PROCESS AND THEIR SOLUTIONS

1) Problems in Requirements Elicitation

Stakeholder Miscommunication

- **Problem:** Stakeholders may not clearly communicate their needs due to technical knowledge gaps or conflicting interests.
- **Solution:** Conduct structured interviews and workshops; employ prototyping for better stakeholder articulation.

Overlooked Stakeholders

- **Problem:** Certain stakeholders, such as regulatory authorities, may be missed.
- **Solution:** Perform comprehensive stakeholder analysis to include all relevant parties.

TABLE 3. Summary of Problems and Solutions in Requirements Engineering

Process Step	Potential Problem	Proposed Solution
2***Elicitation**	Stakeholder miscommunication	Conduct structured interviews, workshops, and prototyping.
	Overlooked stakeholders	Perform comprehensive stakeholder analysis early.
2***Analysis**	Conflicting requirements	Use MoSCoW prioritization and resolve conflicts through trade-offs.
	Infeasibility of requirements	Conduct feasibility studies and propose phased implementation.
2***Validation**	Missed errors or ambiguities	Perform peer reviews, prototype validation, and define acceptance criteria.
	Stakeholder disengagement	Use visual aids and schedule regular validation meetings.
2***Management**	Scope creep	Implement change control processes and visualize changes with KANBAN boards.
	Version control issues	Use version-controlled repositories (e.g., GitHub) and ensure traceability.

2) Problems in Requirements Analysis

Conflicting Requirements

- **Problem:** Conflicts between stakeholders, such as technical feasibility versus user demands.
- **Solution:** Use prioritization techniques like MoSCoW and engage stakeholders in trade-off discussions.

Infeasibility of Requirements

- **Problem:** Unrealistic requirements due to resource constraints.
- **Solution:** Conduct feasibility studies and propose phased delivery for complex features.

O. PROBLEMS IN REQUIREMENTS VALIDATION

Missed Errors During Validation

- **Problem:** Requirements may lack clarity, leading to costly changes later.
- **Solution:** Use peer reviews and prototyping to validate and define clear acceptance criteria.

Stakeholder Disengagement

- **Problem:** Stakeholders may not actively participate in validation.
- **Solution:** Schedule regular meetings and use engaging visual aids like prototypes.

P. PROBLEMS IN REQUIREMENTS MANAGEMENT

1) Scope Creep

- **Problem:** New requirements may arise during development that were not initially taken into account, leading to delays and budget overruns. For example, a late request to integrate a loyalty reward system may disrupt the project timeline.
- **Solution:**
  - Implement a change control process in which all new requirements are documented, analyzed for impact, and approved before implementation.
  - Utilize KANBAN boards to visualize scope changes and their impact on ongoing tasks.

2) Version Control Issues

- **Problem:** Changes to the requirements may not be documented correctly, leading to confusion about the latest version of the requirements.
- **Solution:**
  - Use tools such as GitHub to maintain version-controlled requirements repositories.
  - Ensure traceability by linking the requirements to the relevant design, development, and testing structures.

3) Communication Gaps Among Stakeholders

- **Problem:** Poor communication can lead to misunderstandings or delays in the implementation of requirements. For example, the development team may misinterpret a requirement due to ambiguous documents.
- **Solution:**
  - Plan regular progress reviews and status updates to keep all stakeholders informed.
  - Maintain a centralized pool of requirements accessible to all team members, providing transparency and clarity.

4) Inadequate Monitoring of Requirement Implementation

- **Problem:** Failure to monitor progress may lead to partial implementation of requirements or complete evasion.
- **Solution:**
  - Use traceability matrices to track the status of each requirement from design to deployment.
  - Appoint requirement owners responsible for ensuring successful implementation.

Q. STORIES, USER SCENARIOS, AND THEIR TASKS

1) Ahmet Wants to Change His Password

User Story: As a user (Ahmet), I would like to change my password because I have a hard time remembering it, so that I can log in to my account more easily.

User Scenario: Ahmet, a registered user of the TechZone website, logs into his account using his current password. After successfully logging in, he navigates to the Account Settings page by clicking on the person icon. On the Account Settings page, Ahmet notices the "Change Password" option and clicks on it. The system redirects him to the Change Password page, where he is prompted to enter his current password for verification. Ahmet enters his current password. The system now displays fields to create a new password. Ahmet is required to enter the new password and confirm it by re-entering it in a second field. The system provides guidelines for the new password, including:

- At least 8 characters long.
- Must include at least 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special symbol (e.g., @, #, or $).

Ahmet chooses a new password that meets these criteria, such as "Ahmet@1234", and types it into both fields. After entering the password, he clicks the "Save" button. The system validates Ahmet's input against the specified password requirements. If the password does not meet the criteria, the system displays an error message, such as: "Your password must include at least 8 characters, 1 uppercase letter, 1 lowercase letter, 1 number, and 1 special character." Since Ahmet's new password is valid, the system successfully updates his password and displays a confirmation message: "Your password has been successfully changed." Ahmet logs out of his account to test his new password. On the login page, he enters his email and the updated password ("Ahmet@1234") and logs in successfully. Ahmet is now satisfied that his new password is easier to remember while maintaining strong security.

Tasks:

- Provide a "Change Password" option in the Account Settings menu.
- Require users to enter their current password for security verification.
- Allow users to create a new password and confirm it by re-entering.
- Validate the new password against system requirements (e.g., minimum 8 characters, including letters and numbers).
- Display a confirmation message: "Your password has been successfully updated."

2) Filtering Computers by Brand

User Story: As a user (Ahmet), I would like to see only Lenovo brand computers so that I can choose the brand computer I want without looking at other computers.

User Scenario: Ahmet, a senior computer engineering student, logs into the TechZone website with his account credentials. After successfully logging in, he navigates to the Computers category by clicking on it from the main menu. On the Computers page, Ahmet sees a wide variety of computers from different brands listed in a grid layout. Ahmet, who is specifically interested in purchasing a Lenovo brand computer, notices the filtering options on the left-hand side of the page. He scrolls through the filter options and finds a section labeled Brand. Within this section, Ahmet selects the checkbox for Lenovo to apply the brand filter. After selecting the checkbox, he clicks on the Filter button located below the options. The system processes his request and updates the product list on the page. Now, only Lenovo computers are displayed on the screen, while all other brand computers are hidden. Ahmet browses through the filtered results and reviews the Lenovo computers, comparing their specifications, prices, and features. Pleased with the filtering functionality, Ahmet finds it much easier to focus on the brand he prefers without being distracted by other options. This streamlined experience allows him to quickly identify the Lenovo computer that best suits his needs and proceed to add it to his "Favourites" section for further consideration.

Tasks:

- Provide a filter option on the product listing page under the "Brand" filter section.
- Include a list of available brands, including Lenovo, as selectable options.
- Allow users to select Lenovo from the brand filter options.
- Once Lenovo is selected, dynamically update the product list to only display Lenovo computers, hiding other brands.
- Ensure that the filtering is real-time without needing to refresh the page.
- Implement a clear filter button so that users can easily remove the filter and see all available computers again.
- Ensure the system efficiently handles the filtering on both the frontend (user interface) and the backend (database queries).

3) Adding Computers to the Favourites Section

User Story: As a user (Ahmet), I would like to add the computers I like to the 'Favourites' section, so that I can decide which computer to buy more easily.

User Scenario: Ahmet, a senior computer engineering student, logs into the TechZone website using his account credentials. Once logged in, he navigates to the Computers category by clicking on it from the main menu. On the computers page, Ahmet sees a list of computers from different brands and begins browsing through the available options. As Ahmet explores the page, he notices a favourite icon (heart symbol) next to each computer. While examining the products, Ahmet identifies a few computers that meet his requirements and budget. He clicks on the favourite icon for each of these computers, and the system adds them to his Favourites section. After completing his browsing, Ahmet clicks on the Favourites icon of the application to review his saved items. The system redirects him to the Favourites page, where he can see all the computers he previously marked as favourites. On this page, Ahmet compares the specifications, prices, and features of the computers more conveniently.

By narrowing down his options in the Favourites section, Ahmet finds it easier to decide which computer to purchase. Satisfied with the comparison, he selects the computer he wants to buy and proceeds to the next steps of adding it to his cart and completing the order. The Favourites feature has helped Ahmet organize his preferences and make an informed decision efficiently.

Tasks:

- Provide a heart icon next to each computer listed on the product pages.
- Allow users to click the heart icon to add a computer to their Favourites section.
- Once clicked, change the icon to a filled heart to indicate that the product has been added to Favourites.
- Store the favourited items in the user's account and ensure they are accessible when logged in.
- Provide a "Favourites" link in the navigation menu or user profile to view all favourited computers.
- Ensure that the Favourites section shows all the computers added by the user in an organized list, displaying key details such as the computer name, image, price, and brand.
- Ensure that favourited computers are easily distinguishable within the Favourites list by including a "Remove" button for quick deletion.

4) Notification of Out-of-Stock Item

User Story: As a user (Ahmet), I would like to receive an information mail if the product I ordered is out of stock, so that I know when I can receive it.

User Scenario: Ahmet successfully completes his purchase of a Lenovo computer on the TechZone application. Shortly after, he receives an order confirmation e-mail indicating that his order is being processed. However, the next day, Ahmet receives another e-mail from TechZone notifying him that the product he ordered is currently out of stock. The e-mail begins with an apology for the inconvenience and provides Ahmet with the details of the situation. It explains that due to high demand, the Lenovo computer he ordered is temporarily unavailable. However, the e-mail assures Ahmet that new stock will arrive within a short period, specifically within 5 business days. To minimize the delay and inconvenience, the e-mail also promises fast shipping as soon as the product is restocked. This means the computer will be delivered to Ahmet immediately after it becomes available, without any additional costs or delays. Reassured by the estimated timeline and the fast delivery promise, Ahmet decides to wait for the restocking. He feels satisfied that TechZone is proactively addressing the issue and keeping him informed. A few days later, Ahmet receives another e-mail notifying him that the computer is back in stock and is being shipped with expedited delivery. Shortly after, his computer is delivered as promised, and Ahmet appreciates the professional handling of the situation.

Tasks:

- When a product is marked as out of stock after an order has been confirmed, trigger an automatic email notification to the user informing them of the stock status.
- The email should contain details such as:
  - The name and description of the out-of-stock product.
  - An estimated time for restocking.
- Ensure the email template is clear and professional, providing necessary details and a possible solution for the user (e.g., expected restock date).
- Ensure that the email notification is sent immediately after the system updates the stock status, with real-time inventory checks.
- Use email segmentation to target only users whose specific orders are impacted by the out-of-stock status.

Other Tasks From Kanban Board:

- Class Design.
- Frontend Design-Navigation Bar for All Pages and About Us And Contact Pages:
  - Designin main theme, adding logo of the e-commerce site, creation of links to all pages like main page to About Us page, and contact page, then designin these two page first.
- Brands API Implementation,Categories API Implementation
- Brands Unit Tests
- Frontend Design- Main Page Design and Getting Brands /Categories.
- Products API Implementation.
- Categories Unit Tests.
- Frontend Design- Main Page Filter Products:
  - After getting all products, categories and products from API and designing them, providing a filter option on the product listing page under the Brand and Categories section. Making the filtering real time without needing to refresh page. Implementation of clearing the filter.
- Product Unit Tests.
- Security Congfiguration and User Operations Implementation.
- Frontend Design- From Main Page to Login Page:
  - Get mail and password and check is there a registered user, if such a user do not exist or wrong password/mail, show a message. Adding a register link.
- Favori Products API Implementation.
- Frontend Design- From Login Page to Register Page:
  - Check for validation of every input, first name, last name, age, email, phone number, user name, and password. Show what rules for these inputs and if it is done properly, then click register button and transfer to the login page.
- Frontend Design-Main Page Product Adding to Favorites and Creation of Favorites Page:

  - Providing a heart icon for each product and allow users add that product their favorites list if they are logged in, unless show a message for login. Once added to the list of favorites, show a message that it is already has been added. In the Favorites page, a button for removing the product from favorites is added. .
- Security Configuration and User Operations Unit Tests,User Address API Implementation
- Frontend Design- Profile Page- User Information Design:
  - Getting user information from API and added functionality to edit these info.
- Frontend Design- Profile Page-Address Information Design.
- Order API Implementation, Autharization Implementation
- Favori Products Unit Tests, Cart API Implementation
- Frontend Design-Main Page Product Adding to Cart:
  - Providing a cart icon for each product and allow users add that product their cart list if they are logged in, unless show a message for login.
- Order Tests.
- Frontend Design- Admin Page Design:
  - After authentication as Admin in login Page, redirect to admin page. Design control of Users, Orders and modify ,add or delete from Brands, Categories, Products as admin.
- .Frontend Design- Cart Page Details and Order Creation:
  - Delete from cart, seeing total price on the cart items and make payment button added which opens a model for card information and enables creation of order. After that , Cart page is refreshed as empty and order is created.
- Test for Authorization,Frontend Design- Profile Page- User's Past Orders,Cart Unit Tests

R. STRUCTURED SPECIFICATION FOR REQUIREMENTS

1) Requirement: Product Categorization

- **Function:** The system shall categorize products into predefined labels dynamically fetched from the database.
- **Description:** Organizes products into categories such as "Computers," "Mobile Phones," "Televisions," "Home," and "Personal," ensuring accurate categorization and dynamic updates when categories or products are added or modified in the database.
- **Inputs:**
  - Product data from the database.
  - Category mapping logic defined in the database schema.
- **Source:** Product inventory database.
- **Outputs:** Categorized product listings displayed on the user interface.
- **Destination:** End-user device (web browsers or mobile apps).
- **Action:**
  - Query the database for products and their associated categories.
  - Generate and render categorized lists dynamically.
- **Requirements:**
  - Database schema must support category-product relationships.
  - Query performance should remain under 5 seconds.
- **Pre-condition:**
  - Database contains valid product and category data.
  - Connection to the database is established.
- **Post-condition:**
  - Categories are displayed accurately on the user interface.
- **Side Effects:** None expected under normal operation.

2) Requirement: Order Tracking Feature

- **Function:** The system shall provide an order-tracking feature that dynamically updates statuses such as "Processing," "Shipped," and "Delivered."
- **Description:** Users will be able to track the status of their orders in real-time, including updates to shipping and delivery statuses.
- **Inputs:**
  - Order ID and status updates from the order management system.
- **Source:**
  - Order management system.
  - Shipping and delivery service integrations.
- **Outputs:** Real-time order status updates displayed on the user interface.
- **Destination:** User interface (order tracking page).
- **Action:**
  - Query the order management system for the latest status.
  - Display dynamic updates to users based on order status.
- **Requirements:**
  - Integration with shipping services to retrieve status updates.
  - User interface must refresh order status in real-time.
- **Pre-condition:**
  - Order has been placed and is processed in the system.
  - Integration with the shipping service is established.
- **Post-condition:**
  - Users see real-time updates on their order status.
- **Side Effects:** None expected under normal operation.

3) Requirement: Shopping Cart Updates

- **Function:** The system shall allow users to add, remove, and update quantities of products in their shopping cart, ensuring that all updates reflect in real-time.
- **Description:** Users should be able to modify their shopping cart by changing the quantity of items, removing products, or adding new ones. All changes must be reflected immediately in the cart view, including updates to the total price.
- **Inputs:**
  - Product ID and updated quantity or removal request.
- **Source:**
  - User input from the shopping cart interface.
  - Shopping cart database (user session data).
- **Outputs:** Updated shopping cart displaying modified product quantities or removed items.
- **Destination:** User interface (shopping cart page).
- **Action:**
  - Modify the shopping cart based on the user's actions (add, remove, update).
  - Recalculate the total price and update the cart in real-time.
- **Requirements:**
  - Shopping cart database must support real-time updates.
  - Total price calculations must be accurate and up-to-date.
- **Pre-condition:**
  - Shopping cart contains products, or it is empty before actions.
  - User inputs valid changes (quantity updates, removals).
- **Post-condition:**
  - Cart is updated correctly, reflecting changes in quantity and price.
- **Side Effects:** None expected under normal operation.

4) Requirement: Low Inventory Alerts for Users

Function: The system shall generate notifications for users when items in their favorites are nearing stock depletion.

Description: Sends notifications to users when products in their favorites list are nearing out-of-stock levels, allowing them to make informed purchase decisions.

Inputs:

- Inventory stock levels from the product database.
- User favorites list (products marked for interest).

Source:

- Product inventory database.
- User favorites database.

Outputs:

- Email or in-app alert notifying users of low stock on favorited products.

Destination:

- User device (email or mobile app notification).

Action:

- Check stock levels for favorited products.
- Send alert notification when a product's stock is below a predefined threshold.

Requirements:

- Stock levels must be monitored regularly.
- Alerts must be sent in a timely manner when stock is low.

Pre-condition:

- User has at least one product in their favorites.
- Product inventory data is accurate and updated regularly.

Post-condition:

- Users are notified of low stock for favorited products.

Side Effects:

- Users may be prompted to make a purchase decision sooner.

Tabular Specification:

Condition	Actio
$S < T$ (Stock level S falls below threshold T, where $T = 10$)	Notify
$S = 0$ (Product added to cart but becomes out of stock)	Notify your c
$Qu > S$ (User quantity Qu exceeds stock S)	Displa
$P \in F$ and $D > 0$ (Product P in favorites and discount D applied)	Notify
$P \in F$ and $S < T$ (Product P in favorites and stock S below threshold T)	Notify

TABLE 4. Low Inventory Alerts for Users: Tabular Specification

5) Requirement: Display Product Details

Function: The system shall display detailed product information, including name, price, description, image, stock status, and brand.

Description: When a user selects a product, the system shall open a dedicated product detail page, presenting all relevant product information dynamically fetched from the database.

Inputs:

- Product ID selected by the user.

Source:

- Product database containing detailed product information.

Outputs:

- A detailed product page displaying the selected product's name, price, description, stock status, and brand.

Destination:

- User interface (product detail page).

Action:

- Query the product database for details about the selected product.

- Display product details (image, name, price, description, etc.) on the product detail page.

Requirements:

- Database schema must support detailed product information (image, name, price, description, stock status).
- Page rendering must be fast and responsive.

Pre-condition:

- User selects a product from the listing.
- Database contains accurate and up-to-date product details.

Post-condition:

- Product details are displayed correctly.

Side Effects:

- None expected under normal operation.

6) Requirement: Favorites Feature

Function: The system shall allow users to save products to a favorites list, which shall be stored in the database and retrievable in under 5 seconds.

Description: Enables users to mark products they are interested in for future reference. The favorites list is stored in the database and should be retrievable quickly when the user returns.

Inputs:
- Product ID to be added to favorites.

Source:
- User input (add to favorites action).
- Favorites database (user favorites list).

Outputs:
- List of favorited products displayed on the user's account page.

Destination:
- User interface (favorites page or profile).

Action:
- Save selected product to the user's favorites list in the database.
- Retrieve and display the favorites list when the user visits their profile or favorites page.

Requirements:
- Database schema must support storing user favorites.
- Retrieval time must be under 5 seconds.

Pre-condition:
- User has logged in and has a valid account.
- Database is accessible and contains user favorites data.

Post-condition:
- User's favorites list is updated and displayed correctly.

Side Effects:
- None expected under normal operation.

7) Requirement: Product Management

Function: Manage products in inventory.

Description: Administrators can add, edit, or remove products.

Inputs:
- Product details (name, price, description).

Source:
- Admin input via the backend interface.

Outputs:
- Updated product list in the inventory.

Destination:
- Product management system and database.

Action:
- Admin updates inventory.
- Product details are updated or removed.

Requirements:
- Admin authentication.
- Inventory database.
- Bulk upload.

Pre-condition:
- Admin must be logged in.

Post-condition:
- Inventory is updated with new or edited products.

Side Effects:
- Product details may affect customer browsing or availability.

8) Requirement: Order Management

Function: Manage orders.

Description: Allows admins to view, update, and track orders.

Inputs:
- Order ID.
- Order status (processing, shipped, delivered).

Source:
- Admin input via the dashboard.

Outputs:
- Order status updates.

Destination:
- Order management system.
- Customer notifications.

Action:
- Update order statuses.
- Manage returns/cancellations.

Requirements:
- Real-time order dashboard.
- Order tracking.

Pre-condition:
- Admin must be logged in.

Post-condition:
- Order status is updated in the system.

Side Effects:
- Customer notifications are triggered.

9) Requirement: Secure Checkout Process

Function: The system shall support a secure checkout process that includes order review, payment options, and order confirmation steps.

Description: Ensures that users can securely review their order, select payment methods, and confirm the order for processing. It must adhere to industry-standard security practices.

Inputs:
- User cart items.
- Personal details.
- Payment information.

Source:
- User input from checkout page (order details, payment method).

- Secure payment gateway (payment processing).

Outputs:

- Order confirmation page with order ID and summary.
- Payment confirmation notification (email or in-app).

Destination:

- User interface (checkout confirmation page).
- Payment gateway (for transaction processing).

Action:

- Validate user input (address, payment details).
- Process payment via third-party gateway.
- Generate an order ID and confirmation page upon successful payment.

Requirements:

- Secure payment processing integration (e.g., PCI DSS compliance).
- Order confirmation must be immediate upon successful payment.

Pre-condition:

- User has items in their cart and is logged in (if required).
- Payment gateway is accessible and functional.

Post-condition:

- User receives order confirmation, and payment is processed.

Side Effects:

- None expected under normal operation.

10) Requirement: User Management

Function: Manage customer accounts.

Description: Administrators can view, edit, or deactivate customer accounts in cases of fraudulent activity.

Inputs:

- Customer account details (name, email, account status).

Source:

- Admin actions via the management dashboard.

Outputs:

- Updated customer account details or deactivated accounts.

Destination:

- Customer database.
- User management logs.

Action:

- Admin selects a customer account.
- Updates details or disables the account.

Requirements:

- Admin authentication.
- Secure database access.
- Activity logs.

Pre-condition:

- Admin must be logged into the system with sufficient permissions.

Post-condition:

- Customer account reflects the updated or deactivated status.

Side Effects:

- Customers may lose access if their account is deactivated.
- Notifications to users may be triggered for account status changes.

11) Requirement: Notifications

Function: Receive inventory alerts.

Description: Administrators are notified when product inventory reaches a low threshold to enable restocking.

Inputs:

- Inventory level thresholds.
- Product details.

Source:

- Automated inventory monitoring system.

Outputs:

- Notification or alert sent to administrators.

Destination:

- Admin dashboard.
- Email system.

Action:

- The system checks inventory levels.
- Triggers alerts when predefined thresholds are met.

Requirements:

- Inventory monitoring tool.
- Admin contact information.
- Notification system.

Pre-condition:

- Inventory monitoring system is active.
- Thresholds are set.

Post-condition:

- Admin receives a notification for action.

Side Effects:

- Alerts may create a backlog if restocking is delayed.

12) Requirement: System Monitoring

Function: System performance monitoring.

Description: Technical staff must monitor system performance and uptime.

Inputs:

- System logs.
- Alerts for errors.

Source:

- System monitoring tools.
- Logs.

Outputs:

- Alerts for issues such as downtime or performance degradation.

Destination:

- Monitoring dashboard.

- Technical support team.

Action:

- Set up alerts.
- Monitor system performance.

Requirements:

- Uptime monitoring.
- Error tracking.
- Alert system.

Pre-condition:

- System must be deployed and operational.

Post-condition:

- Alerts sent if performance drops or errors occur.

Side Effects:

- Immediate intervention required to avoid service disruptions.

13) Requirement: Security Compliance

Function: Ensure system security and compliance with data protection laws.

Description: Ensures the platform is secure and complies with regulations like GDPR.

Inputs:

- Security audit logs.
- Compliance requirements.

Source:

- Security monitoring tools.
- Legal documentation.

Outputs:

- Security status reports.
- Compliance updates.

Destination:

- Security management tools.
- Compliance team.

Action:

- Conduct regular security audits.
- Apply security patches.

Requirements:

- Secure login.
- Encryption.
- GDPR compliance tools.

Pre-condition:

- System must be operational and connected.

Post-condition:

- Platform is secure and compliant.

Side Effects:

- User privacy concerns.
- Potential system slowdowns due to extra security measures.

S. REQUIREMENTS CHECKING

1) Validity

Question: Does the system provide the functions which best support the customer's needs?

Assessment: The functionalities (e.g., user management, product catalog, shopping cart, favorites, order tracking) directly address customer needs for an e-commerce platform. Real-time updates for shopping carts and order tracking meet the needs for transparency and convenience.

- **Product Categorization:**
  - Customers need to browse products by categories (e.g., "Computers," "Mobile Phones").
  - System includes a dynamic categorization mechanism that reflects updates in real-time, fully meeting this need.
- **Order Tracking:**
  - Real-time order tracking supports the customer's expectation of staying informed about their orders' statuses (e.g., "Processing," "Shipped").
  - This feature directly addresses customer needs for transparency.
- **Shopping Cart and Checkout:**
  - The ability to add, remove, and update items in the shopping cart in real-time reflects essential customer functionality.
  - Secure checkout with multiple payment methods (e.g., credit card, cash on delivery) fulfills customer expectations for flexibility and security.
- **Low Inventory Alerts & Favorites Feature:**
  - Notifications about low inventory and discounts for favorited items cater to customer preferences for timely decision-making.

Conclusion: System provides functions that closely align with the customer's needs. The design and functionalities are valid and appropriate for the problem domain.

2) Consistency

Question: Are there any requirements conflicts?

Assessment: There are no conflicts between functional and non-functional requirements. User needs (e.g., real-time updates for cart) align with system capabilities (e.g., database support for concurrent operations). Conflicts between admin features (e.g., product management, order tracking) are avoided. For instance, deleting a product does not create inconsistencies in the order history.

- **Shopping Cart & Security:**
  - Real-time shopping cart updates must not conflict with secure session handling (e.g., no conflicts between live updates and encrypted connections).
- **Performance vs Scalability:**
  - Ensure that performance goals (e.g., <5 seconds search response time) are feasible even under scalability constraints (10,000 concurrent users).

- **Low Inventory Alerts and Shopping Cart Integration:**
  - Alerts for low inventory in favorites do not interfere with cart updates.
- **Favorites Feature and Notifications:**
  - Notifications about out-of-stock or discounted products work seamlessly with the favorites feature.
- **Pre-Conditions and Post-Conditions:**
  - Pre-conditions and post-conditions across requirements (e.g., shopping cart updates, order tracking) are logically consistent and do not contradict each other.

Conclusion: The requirements are consistent and free from conflicting conditions or behaviors. There are no overlaps or contradictions that would hinder implementation.

3) Completeness

Question: Are all functions required by the customer included?

Assessment: Most user-facing functionalities (e.g., shopping cart updates, secure checkout, order history, and product browsing) are covered. Requirements include key operational constraints such as performance, scalability, security, and compliance.

- **Customer-Facing Features:**
  - Product browsing, filtering, and sorting.
  - Shopping cart updates and secure checkout.
  - Order tracking and order history.
  - Favorites and low inventory alerts.
- **Administrator Features:**
  - Product management (add, edit, delete products).
  - Order management (update statuses, manage cancellations/returns).
  - Notifications for low inventory alerts.
- **Technical Features:**
  - System monitoring for performance and uptime.
  - Security compliance (GDPR, PCI DSS).

Conclusion: The requirements are nearly complete, with minor opportunities to enhance by including advanced search options and accessibility standards.

4) Realism

Question: Can the requirements be implemented given available budget and technology?

Assessment: Features like user registration, shopping cart updates, and secure checkout are realistic with the chosen technologies.

- **Technologies Used:**
  - Frontend: HTML, CSS, Bootstrap, jQuery/AJAX.
  - Backend: Java-based API.
  - Database: Assumed to support dynamic queries for categorization, shopping cart updates, and order tracking.
- **Performance:**
  - Non-functional requirements specify performance metrics (e.g., search results under 5 seconds). These are realistic, given the technologies used.
- **Budget:**
  - Realistic costs include hosting, payment gateway integration, and developer effort.

Conclusion: The requirements are realistic, achievable, and align with the capabilities of the chosen technologies and tools.

5) Verifiability

Question: Can the requirements be checked?

Assessment: Functional requirements are highly verifiable, with clear actions, inputs, and outputs defined. Non-functional requirements include measurable metrics, making them testable.

- **Product Categorization:**
  - Test: Verify that products are dynamically categorized and displayed correctly based on database entries.
- **Order Tracking:**
  - Test: Place an order and track its status through each stage ("Processing," "Shipped," "Delivered"). Ensure the status updates are visible in real-time.
- **Shopping Cart Updates:**
  - Test: Add, remove, and update item quantities in the shopping cart. Validate that the total price reflects the changes immediately.
- **Low Inventory Alerts:**
  - Test: Simulate inventory levels falling below the threshold and verify that the system sends alerts to the user.
- **Secure Checkout:**
  - Test: Complete the checkout process using various payment methods and validate order confirmation and security compliance.

Conclusion: All requirements are testable through well-defined conditions and outputs. Metrics like response times (e.g., “under 5 seconds”) ensure additional verifiability during testing.

Criteria	Assessment
Validity	The system meets customer needs (e.g., shopping cart, secure checkout, product browsing). All features are aligned with user and business goals.
Consistency	Requirements are consistent, with no visible conflicts between functional/non-functional needs or user/system goals.
Completeness	Requirements cover most functionalities. Advanced search and accessibility standards could be added for improvement.
Realism	Implementation is feasible with the chosen technologies and a realistic budget, though scalability and compliance may require careful planning.
Verifiability	Requirements are testable with well-defined metrics, pre-conditions, and post-conditions (e.g., <5s search response time, secure checkout validation).

TABLE 5. Assessment of Requirements Based on Key Criteria

T. DO YOU FORESEE ANY CHANGES IN YOUR SYSTEM? IF SO, WHAT DO YOU PLAN TO MAKE CHANGES?

There are potential areas for upgrade or revision as the project progresses. These changes could arise from evolving user needs, technological developments, or feedback during testing phases.

1) Improving the User Experience

Foreseen Change: Adding advanced features for user convenience to stay competitive with other e-commerce platforms.

Planned Changes:

- **Wishlist Sharing Feature:** Allow users to share their favorite list with friends or family via email or social media. This helps collaborative shopping and increases user engagement.
- **Personalized Recommendations:** Introduce a recommendation engine that suggests products based on user browsing or purchase history.

Reason for Change: These enhancements align with modern e-commerce trends and improve customer satisfaction.

2) Improving Search Functionality

Foreseen Change: The current search functionality supports keyword-based search but could be enhanced for better usability.

Planned Changes:

- **Full-Text Search:** Enable users to search among product names, descriptions, and tags.
- **Autocomplete Suggestions:** Provide search suggestions while users type to improve search efficiency.
- **Search Filters for Discounts or Promotions:** Add filters like "On Sale" or "New Arrivals."

Reason for Change: Enhanced search functionality makes it easier for users to find desired products, improving their overall experience.

3) Notifications System

Foreseen Change: The current notification system focuses on low inventory alerts and order confirmations. Expanding its scope would improve user engagement.

Planned Changes:

- **Order Shipment Notifications:** Notify users via email or SMS when their order is shipped or delivered.
- **Personalized Marketing Notifications:** Inform users about discounts on their favorite products or abandoned cart reminders.

Reason for Change: Proactive communication increases user satisfaction and encourages repeat purchases.

4) Administrator Features

Foreseen Change: Administrative features may require enhancements for better efficiency and control.

Planned Changes:

- **Advanced Analytics Dashboard:** Provide admins with insights on sales trends, inventory turnover, and customer behavior.

Reason for Change: Enhancing admin tools streamlines operations and improves decision-making.

5) Security Enhancements

Foreseen Change: With evolving cybersecurity threats, additional layers of security may be required.

Planned Changes:

- **Two-Factor Authentication (2FA):** Require 2FA for user and admin logins to strengthen account security.
- **Fraud Detection System:** Monitor transactions for suspicious activities (e.g., unusually large orders or multiple failed payment attempts).

Reason for Change: Security enhancements ensure data protection and maintain user trust.

Summary: The planned changes aim to improve the platform's usability, scalability, security, and overall user satisfaction. While the system is well-designed for its initial goals, adjusting to evolving needs and feedback ensures long-term success. These changes are not immediately critical but are foreseen enhancements as the system evolves.

U. SYSTEM BOUNDARIES

The system boundary defines the scope of the online technology store, determining what functionalities and components are included within the system and which external systems or processes interact with it. This boundary ensures a clear understanding of the system's responsibilities, dependencies, and relationships between internal and external entities.

1) Internal Components (Inside the System Boundary)

These are the features, functionalities, and modules that are entirely part of the online technology store system. They represent the core system that the development team is responsible for building, maintaining, and managing.

a: Frontend

Technologies Used: HTML, CSS, Bootstrap, jQuery/AJAX.
Key Features:

- Home page displaying product categories (e.g., Computers, Mobile Phones).
- Product listing pages with brand-based filtering options.
- Product detail pages showing product specifications, availability, and price.
- Interactive user functionalities:
  - Add products to Favorites.
  - Add products to the Shopping Cart.
  - Manage the shopping cart (update quantities, remove items).
- User account management pages:
  - Login and registration forms.
  - Profile editing.
  - Viewing order history.

b: Backend

Technologies Used: Java (Spring Boot, Spring Security).
Key Features:

- User Management:
  - Registering new users.
  - Authenticating users during login.
  - Managing user sessions.
- Product Management:
  - Retrieving product lists by category and brand.
  - Managing product availability and stock.
- Order Management:
  - Storing new orders in the database.
  - Retrieving order history for users.
  - Handling payment processing through external payment systems.
- Favorites Management:
  - Adding/removing items to/from a user's favorites.

c: Database

Technologies Used: PostgreSQL. **Key Data Structures:**

- **User Table:** Stores detailed user information, including ID, name, email, and password.
- **UserAddress Table:** Stores users' delivery addresses linked via user IDs.
- **Brand Table:** Manages product brand information.
- **Category Table:** Organizes products into categories (e.g., Computers, Mobile Phones).
- **Product Table:** Stores detailed product information, including name, price, stock, and category.
- **FavouriteProduct Table:** Tracks user-favorited products for personalization.
- **Order Table:** Manages customer orders with details like status, total price, and timestamps.
- **OrderItem Table:** Links products to orders, tracking quantities and prices.
- **Cart Table:** Manages user shopping carts.
- **CartItem Table:** Links products to carts with quantity details.

d: Project Management and Development Tools

- **GitHub:** Used for version control and KANBAN workflows to manage tasks (e.g., To Do, In Progress, Testing, Done).
- **JUnit:** Used for backend testing to ensure API functionality correctness.
- **Postman:** Used for API testing to verify communication between the frontend and backend.

2) External Components (Outside the System Boundary)

These are systems, services, or entities that interact with the online technology store but are not part of the core development scope. They provide inputs or support external functionalities critical for the system's operation.

a: Payment Gateway System

Interaction with the System: Handles secure payment processing. The system sends payment details to the gateway, which confirms payment status.

b: Delivery and Logistics Partners

Interaction with the System: Shares order details (e.g., delivery address, product information) with logistics partners for order fulfillment.

c: Social Media Integration

Interaction with the System: Allows users to log in using their Google accounts via Google's API for authentication.

d: E-mail Notification Service

Interaction with the System: Sends automated e-mails for registration, order confirmations, and stock-related issues via a third-party e-mail service.

e: Regulatory Compliance Authorities

Interaction with the System: Ensures compliance with legal standards like GDPR for data protection.

3) Implications of System Boundaries

- **Scope Definition:** Externalizing functionalities like payment processing and logistics allows the project to focus on core components (e.g., user management, product browsing).
- **Technical Dependencies:** Integration with external systems requires secure APIs, robust protocols, and data-sharing agreements.
- **Regulatory Constraints:** Compliance with standards like GDPR influences data handling and storage practices.
- **Resource Allocation:** Outsourcing complex functionalities enables efficient resource allocation for frontend, backend, and database development.

By establishing these system boundaries, the team ensures a clear division of responsibilities between internal components and external systems, optimizing development efforts and aligning with project goals.

III. DESIGN DOCUMENTATION FOR TECHZONE

A. INTRODUCTION

1) Project Name

Techzone

2) Project Description

This project involves building a modern, secure, and user-friendly e-commerce platform using Spring Boot, where users can browse, search, and purchase technology products. The system will include authentication, authorization, and secure payment functionalities.

3) Goals and Objectives

- Provide an intuitive user interface for customers to browse and purchase products.
- Offer secure user authentication and role-based access control using Spring Security.
- Ensure scalability, maintainability, and high performance through the N-Tier Architecture.

4) Target Users

- **Customers:** Can browse products, add items to their cart, and place orders.
- **Administrators:** Can manage products, monitor sales, and manage users.

5) Technologies Used

- **Backend:** Java (Spring Boot, Spring Security)
- **Frontend:** HTML, CSS, JavaScript, JQuery
- **Database:** PostgreSQL
- **Deployment:** Cloud-based deployment with Docker and Maven

B. SYSTEM OVERVIEW

1) System Architecture

The system leverages the Spring Boot Framework to implement the N-Tier Architecture:

- **Presentation Layer:** Spring Controllers handle HTTP requests and interact with the frontend (HTML, CSS, JavaScript).
- **Business Logic Layer:** Spring Services implement the core application logic.
- **Data Access Layer:** Spring Data JPA Repositories manage database interactions with PostgreSQL.
- **Database:** PostgreSQL stores application data in normalized tables.

2) System Modules

- **Authentication & Authorization Module:** Handles login, signup, and role-based access control using Spring Security.

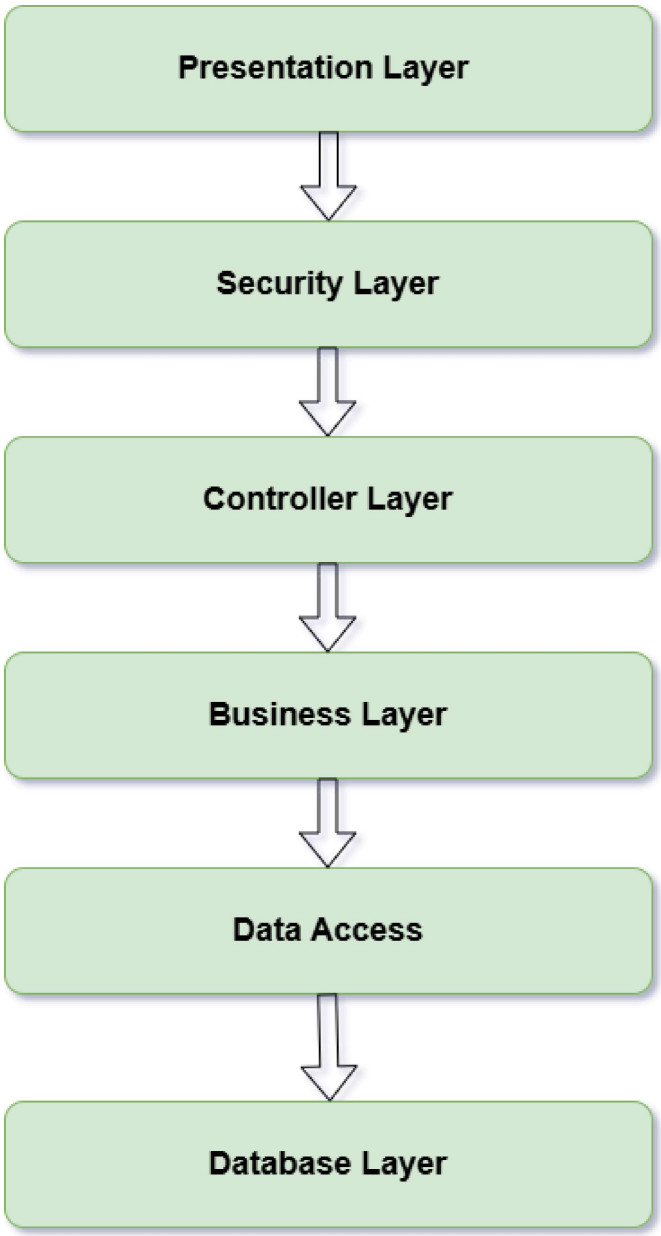


FIGURE 1. System Architecture,

- **Product Management Module:** Allows administrators to add, update, and delete products.
- **Cart Management Module:** Manages user carts and checkout processes.
- **Order Management Module:** Handles order creation, payment processing, and order tracking.

C. ARCHITECTURAL DESIGN

1) N-Tier Architecture Design

The system's architecture is divided into the following layers:

- **Controller Layer:**
  - Handles HTTP requests from the frontend and returns appropriate responses.
  - Responsible for interacting with the Security Layer and forwarding valid requests to the Service Layer.

  - *Example:* /api/products for fetching product details.
- **Security Layer (Spring Security Filter):**
  - Acts as a filter between the Controller Layer and incoming HTTP requests.
  - Responsible for:
    * **Authentication:** Verifying user credentials (e.g., JWT tokens).
    * **Authorization:** Ensuring users have the necessary permissions (e.g., roles like ROLE USER, ROLE ADMIN) to access specific endpoints.
  - Requests that fail security checks are rejected here and do not proceed to the Controller Layer.
- **Service Layer:**
  - Contains the application's core business logic.
  - Orchestrates operations by coordinating between the Controller Layer, Repository Layer, and other services.
  - *Example:* ProductService for managing product-related operations, such as calculating prices or checking stock.
- **Repository Layer:**
  - Interacts with the database via Spring Data JPA.
  - Responsible for querying, inserting, updating, and deleting data in the database.
  - *Example:* ProductRepository for executing database queries related to products.
- **Database Layer:**
  - Stores application data in normalized tables using PostgreSQL.
  - Ensures data integrity, consistency, and relationships between entities (e.g., User, Product, Order, Cart).

2) Why Spring Boot?

- **Embedded Server:** Built-in support for Tomcat or Jetty eliminates the need for external server configuration.
- **Dependency Management:** Maven and Spring Boot Starter dependencies simplify library integration.
- **Auto-Configuration:** Reduces boilerplate code with automatic configurations.
- **Spring Security Integration:** Provides out-of-the-box support for secure authentication and authorization.
- **Microservices Ready:** Modular and scalable architecture for large applications.

3) Security Design

- **Authentication:** Verifies user credentials using JWT tokens.
- **Authorization:** Implements role-based access control (e.g., ROLE ADMIN, ROLE USER).
- **Password Hashing:** Uses BCrypt to securely hash user passwords.

4) Deployment Design

- **Spring Boot Application:** The backend will be packaged as a JAR file and deployed using Docker for scalability.
- The application will run on cloud platforms like AWS or Azure.

D. DATABASE DESIGN

1) ER Diagram

Entities and relationships:

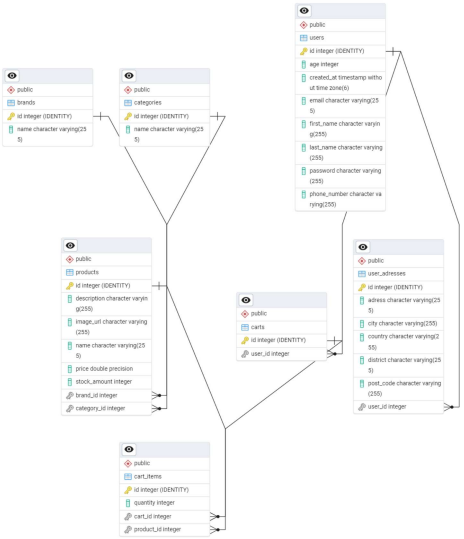


FIGURE 2. ER Diagram

E. TABLE SCHEMA

Column	Type
id	NUMBER
name	VARCHAR
surname	VARCHAR
age	NUMBER
phonenumber	VARCHAR
email	VARCHAR
password	VARCHAR
created_at	DATE

TABLE 6. User Table Schema

Column	Type
id	NUMBER
user_id	NUMBER
country	VARCHAR
district	VARCHAR
city	VARCHAR
post_code	VARCHAR
address	VARCHAR

TABLE 7. User Address Table Schema

Column	Type
id	NUMBER
name	VARCHAR
price	NUMBER
stock	NUMBER
image	VARCHAR
description	VARCHAR
category_id	NUMBER
brand_id	NUMBER

TABLE 8. Products Table Schema

Column	Type
id	NUMBER
user_id	NUMBER

TABLE 9. Cart Table Schema

Column	Type
id	NUMBER
cart_id	NUMBER
product_id	NUMBER
quantity	NUMBER

TABLE 10. Cart Item Table Schema

Column	Type
id	NUMBER
user_id	NUMBER
product_id	NUMBER
added_at	NUMBER

TABLE 11. Favorite Products Table Schema

Column	Type
id	NUMBER
name	VARCHAR

TABLE 12. Categories Table Schema

Column	Type
id	NUMBER
name	VARCHAR

TABLE 13. Brands Table Schema

Column	Type
id	NUMBER
user_id	NUMBER
total_price	NUMBER
status	VARCHAR
created_at	DATE
updated_at	DATE

TABLE 14. Orders Table Schema

F. SYSTEM MODULES

1) Authentication & Authorization

- **Spring Boot Integration:** Authentication and authorization processes are implemented via Spring Security.

Column	Type
id	NUMBER
order_id	NUMBER
product_id	NUMBER
quantity	VARCHAR
price	DATE

TABLE 15. Order Items Table Schema

- **Flow:**
  1) User sends credentials to `/api/auth/login`.
  2) Spring Security validates credentials and generates a JWT token.
  3) User includes the token in subsequent requests to access protected resources.

2) Product Management

- **Flow:**
  1) Admin sends product details to `/api/products`.
  2) Backend validates input and updates the database using `ProductRepository`.

3) Order Management

- **Flow:**
  1) User finalizes the cart and sends a request to `/api/orders`.
  2) Spring Boot processes the request and stores the order details in the database.

G. USER INTERFACE DESIGN

- **Homepage:** Displays products in a grid layout.
- **Product Page:** Shows detailed information about selected products.
- **Cart Page:** Allows users to view and manage items in their cart.
- **Admin Dashboard:** Provides tools for managing products, users, and orders.

IV. DIAGRAMS

A. USECASE DIAGRAM

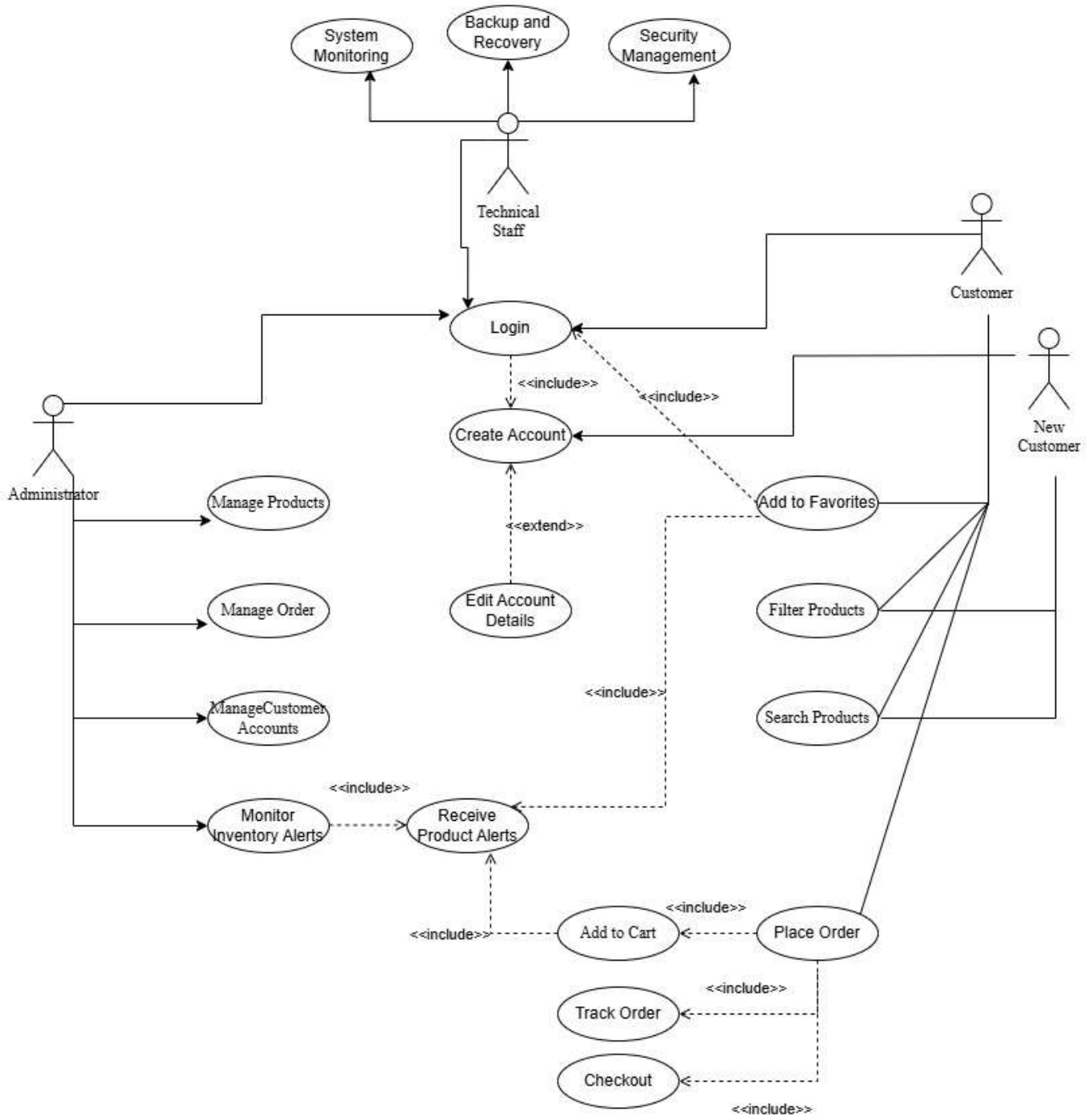


FIGURE 3. Usecase Diagram

B. STATE DIAGRAM

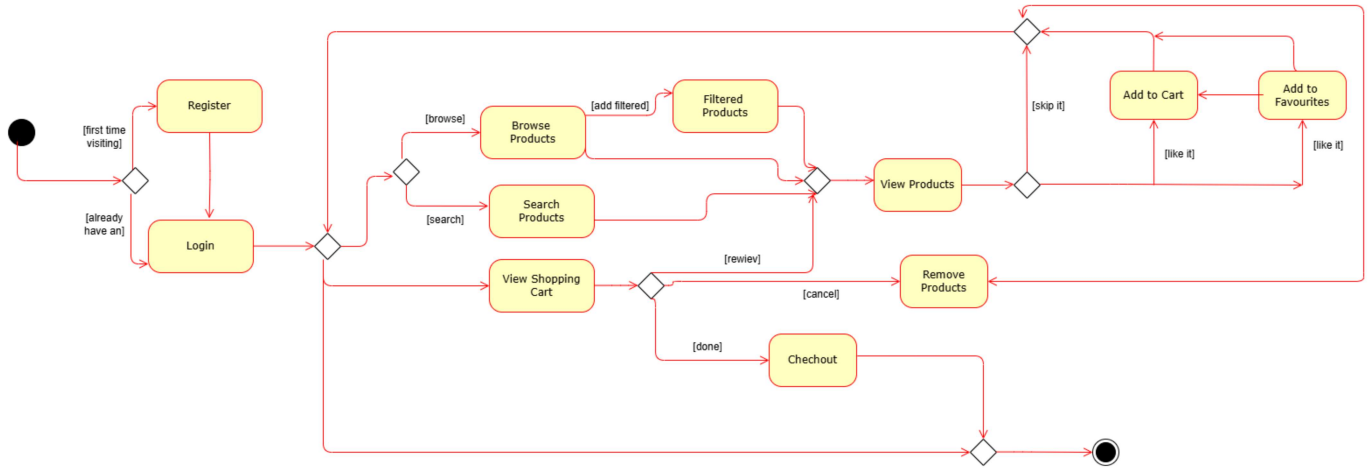


FIGURE 4. State Diagram

C. CLASS DIAGRAM

Defines relationships between User, Product, Order, Cart, and CartItem.

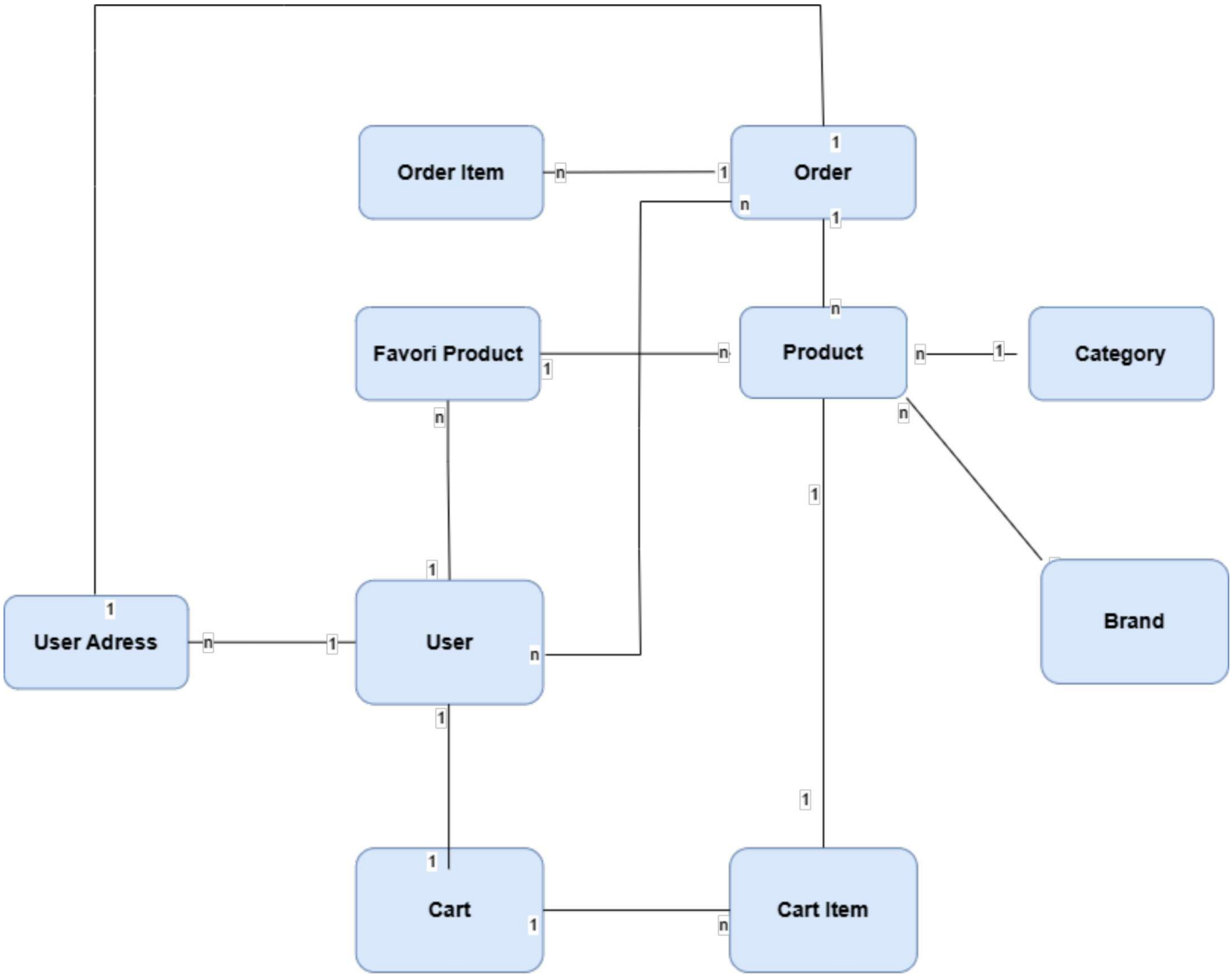


FIGURE 5. Class Diagram

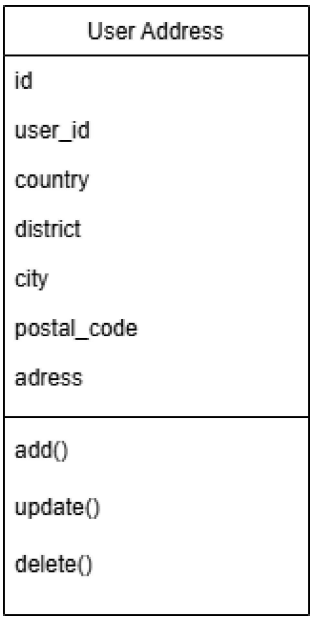


FIGURE 6. User Adress Class Diagram

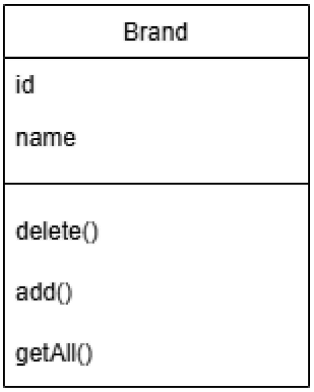


FIGURE 7. Brand Class Diagram

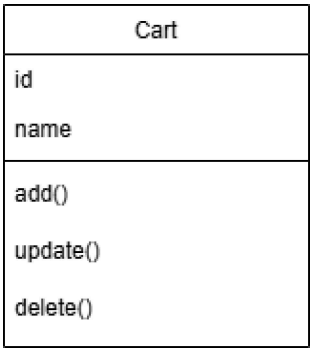


FIGURE 8. Cart Class Diagram

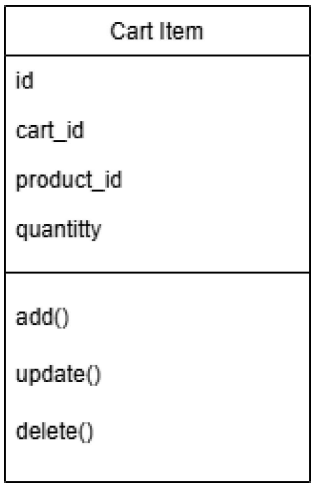


FIGURE 9. Cart Item Class Diagram

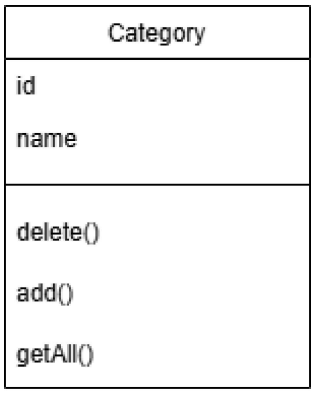


FIGURE 10. Category Class Diagram

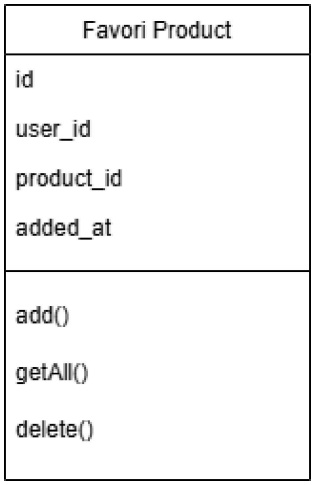


FIGURE 11. Favori Product Class Diagram

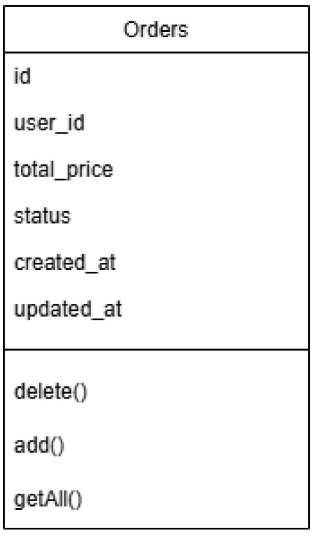


FIGURE 12. Order Class Diagram

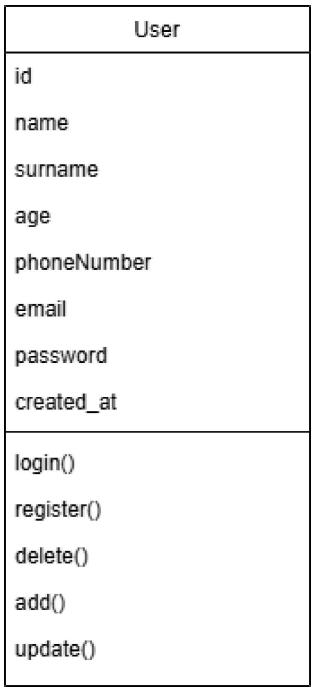


FIGURE 14. User Class Diagram

FIGURE 13. Product Class Diagram

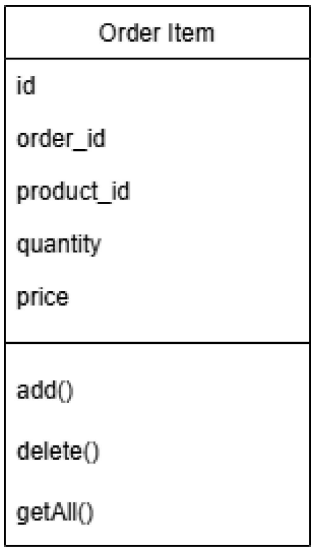


FIGURE 15. Order Item Class Diagram

D. SEQUENCE DIAGRAM

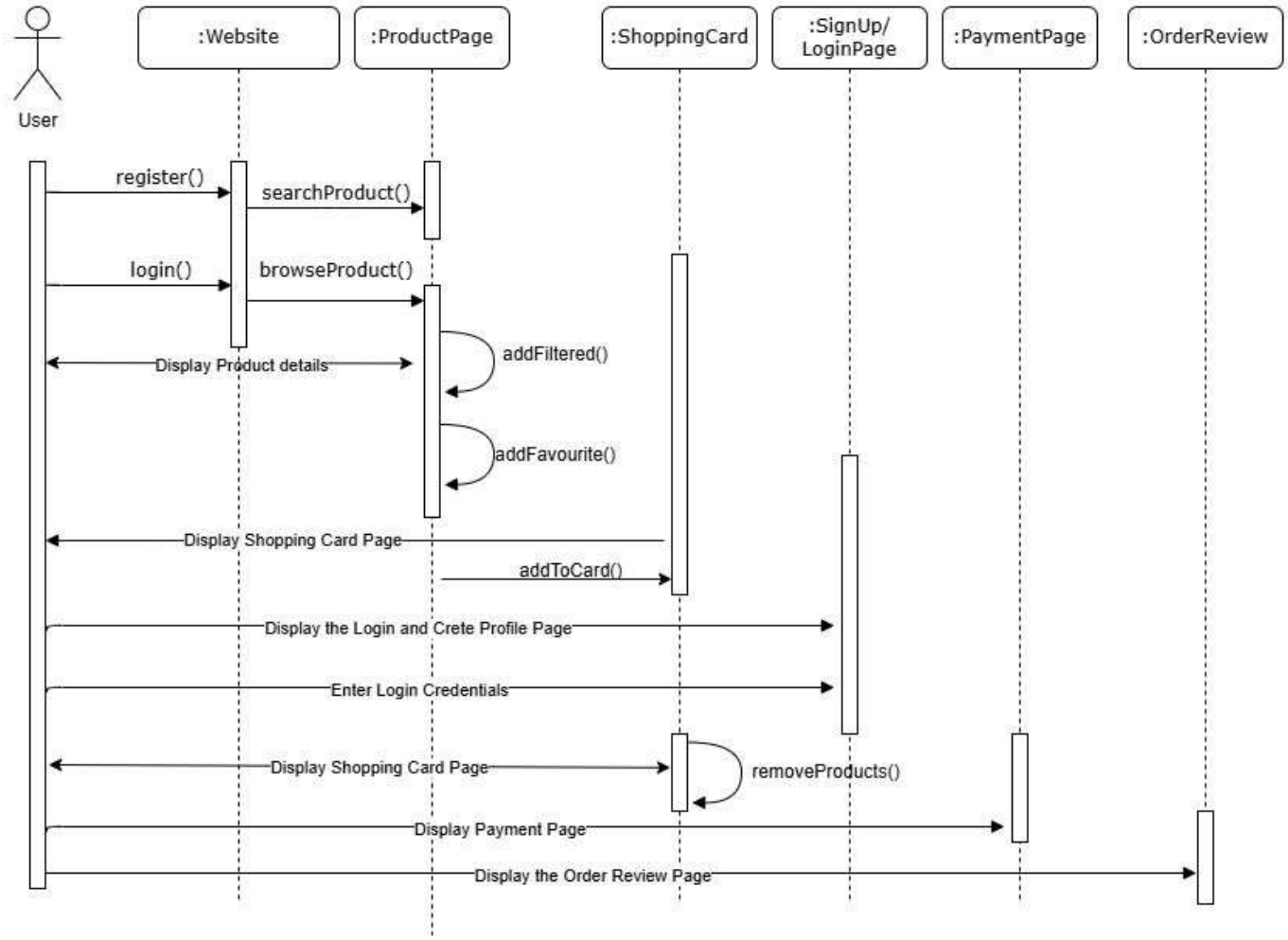


FIGURE 16. User Sequence Diagram

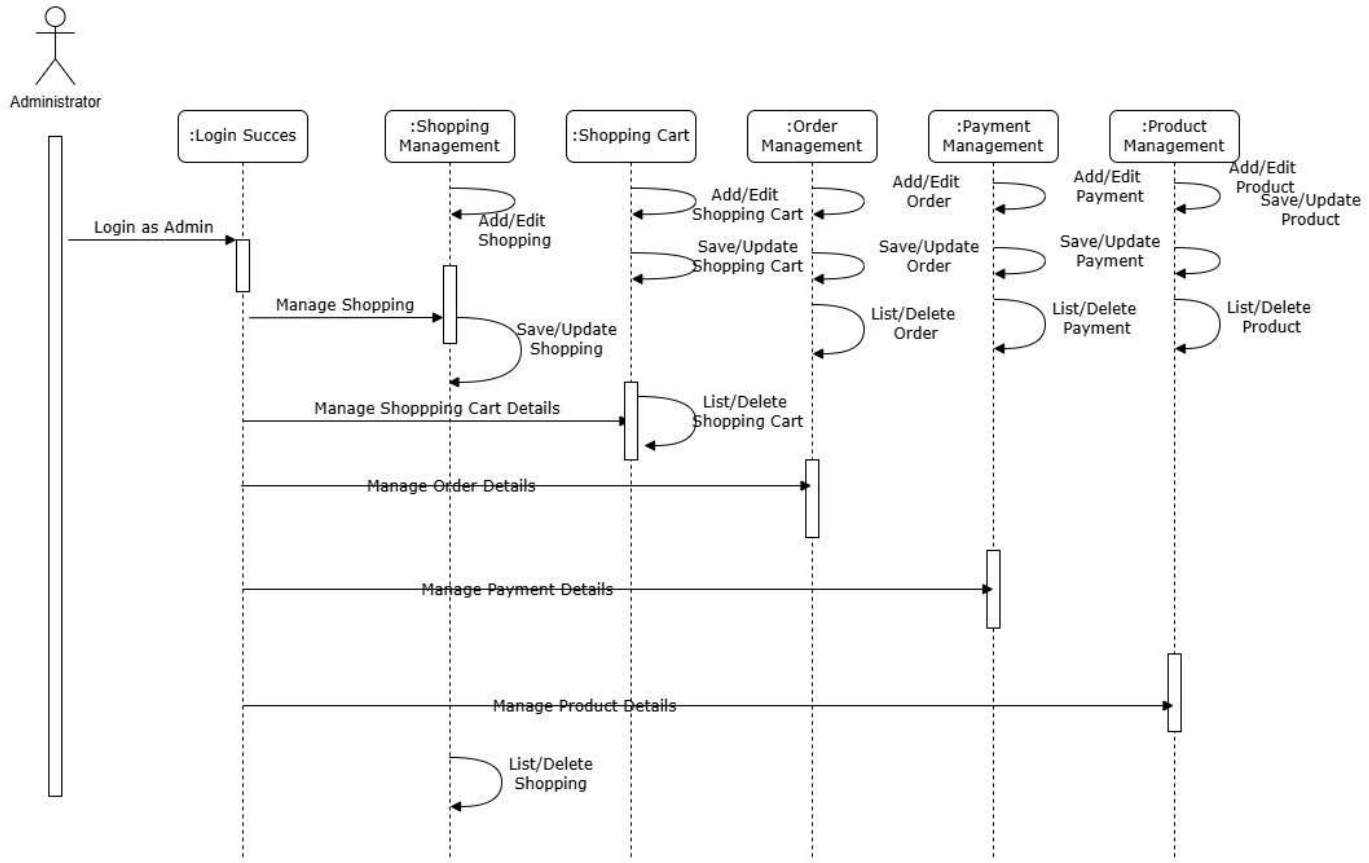


FIGURE 17. Administrator Sequence Diagram

E. SECURITY DESIGN

Spring Boot Integration with Spring Security:

- **Authentication:** JWT tokens are generated upon successful login.
- **Authorization:**Access control based on user roles (ROLE USER, ROLE ADMIN).
- **Password Hashing:** BCrypt ensures secure password storage.

V. DOCUMENTATION FOR TECHZONE

A. RISK MANAGEMENT

Risk Identification:

- **Estimation:** - Developing API integrations may take longer than estimated. Designing complex frontend features, such as real-time filtering, may take more time than expected. The duration of testing processes might be underestimated.
- **Organizational:**- Lack of task delegation among team members may delay the completion of tasks.
- **People** - Miscommunication among team members due to unnecessary workload may arise.
- **Requirements** - Last-minute requirement changes (e.g., adding a new payment method). - Misunderstanding user needs could lead to incomplete or incorrect requirement designs.
- **Technology** - Integration issues due to incompatible versions of APIs or database tools. - Inadequate security configurations could compromise user data.
- **Tools** - - Unexpected issues may arise in the frontend framework (e.g., JavaScript, Bootstrap).

Risk Analysis:

- **Estimation:** - Likelihood: High Impact: Moderate Severity: High Developing API integrations, complex frontend features like real-time filtering, and underestimating testing durations may lead to project delays.
- **Organizational:**- Likelihood: Medium Impact: High Severity: High Lack of proper task delegation among team members could cause delays in task completion, affecting the project timeline.
- **People** - Likelihood: Medium Impact: Moderate Severity: Medium Miscommunication due to excessive workloads might slow progress and reduce productivity.
- **Requirements** - Likelihood: High Impact: High Severity: Critical Last-minute requirement changes (e.g., adding a payment method) or misunderstandings of user needs could result in incomplete or incorrect designs, leading to rework.
- **Technology** - Likelihood: High Impact: High Severity: Critical Integration issues due to incompatible versions of APIs or database tools and inadequate security configurations could compromise functionality and user data.
- **Tools** - Likelihood: Medium Impact: Moderate Severity: Medium Unexpected issues with frontend frameworks, such as JavaScript or Bootstrap, may disrupt development and require additional troubleshooting.

B. ERROR MESSAGES

Admin Site

1) Page for adding brand to the site and its message to the screen when it is successful:

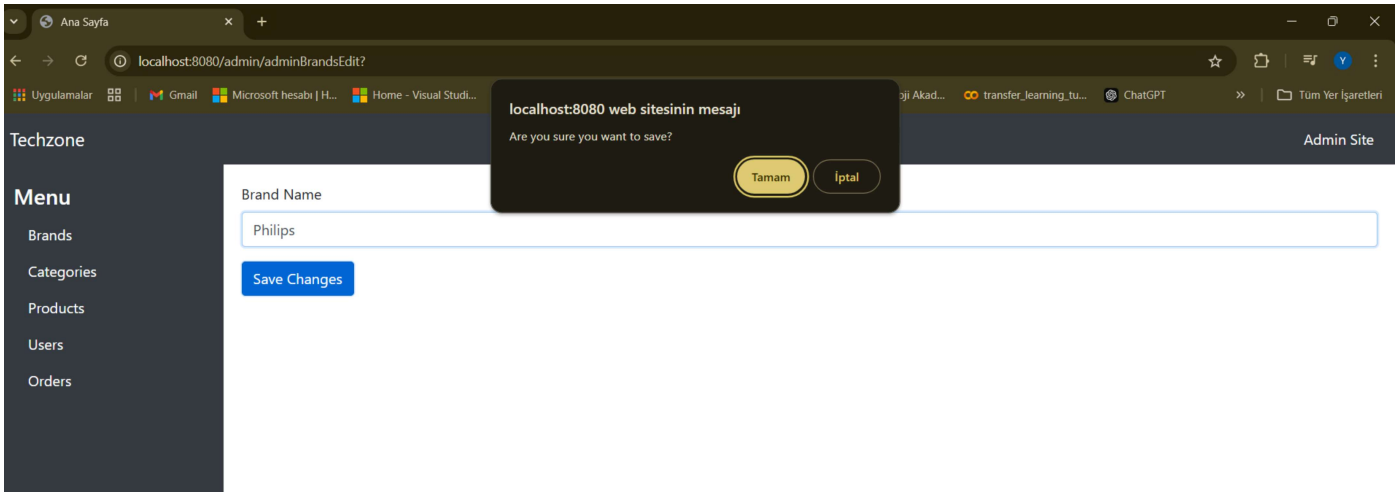


FIGURE 18. Brand Add Picture

2) Page for editing existing brand name with button, and successfully saved message for it:

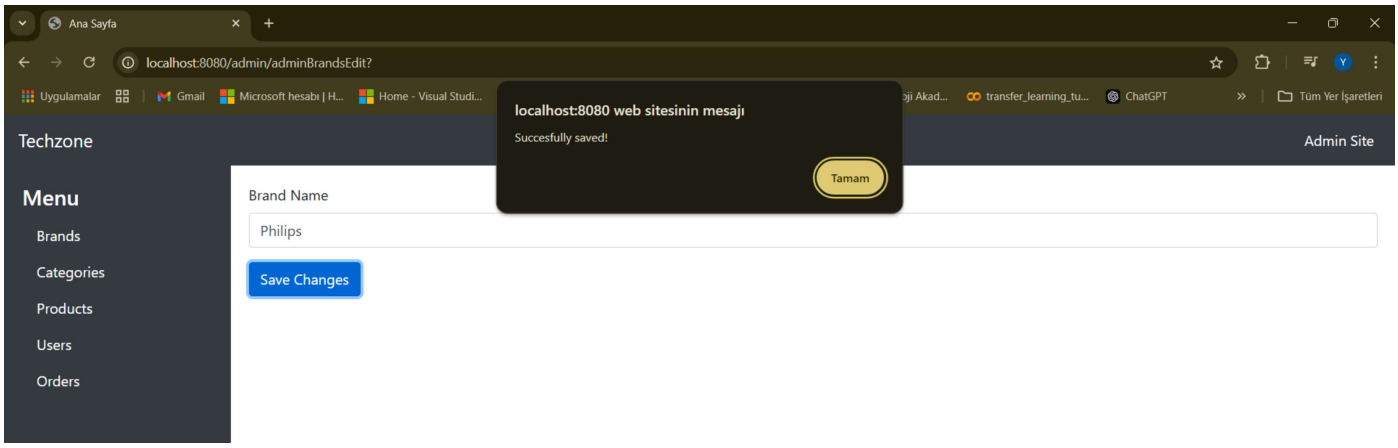


FIGURE 19. Brand Edit

3) After adding a new brand and editing it, image of the page:

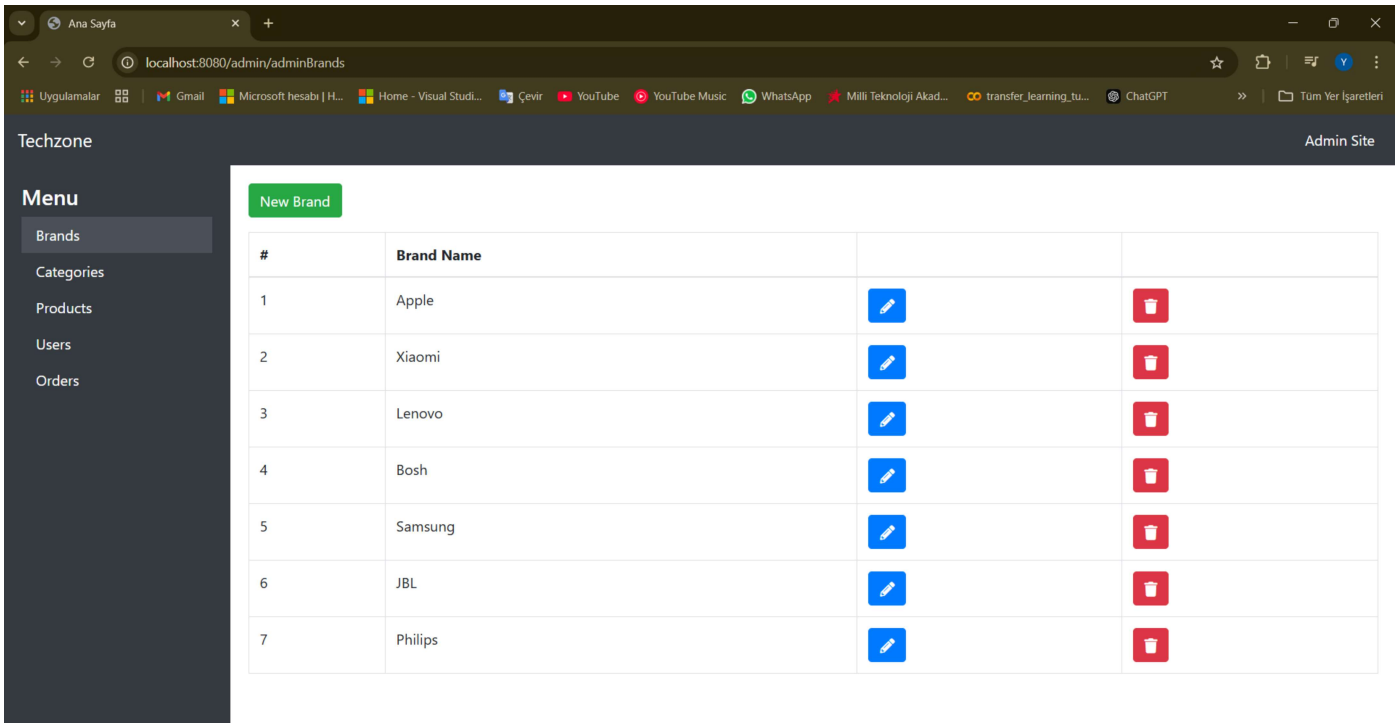


FIGURE 20. Enter Caption

4) In the categories tab, adding a new category to the site, and its success message:
5) Edit category name and successfully save it:

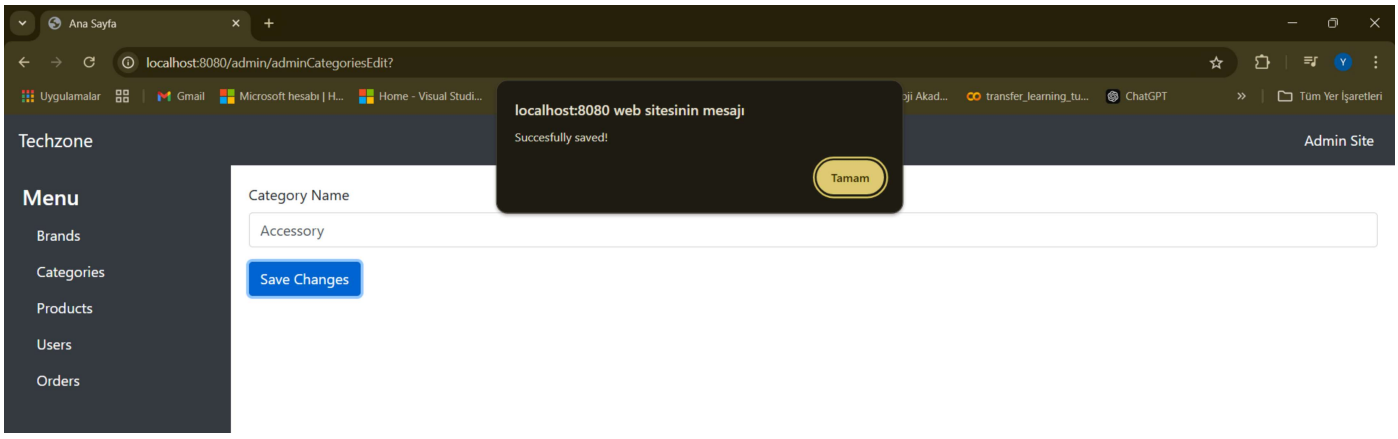


FIGURE 21. Enter Caption

6) When admin wants to delete a category, a confirmation message is displayed and with the approval with “Tamam”, page is reloaded and category name is deleted and there is full page:

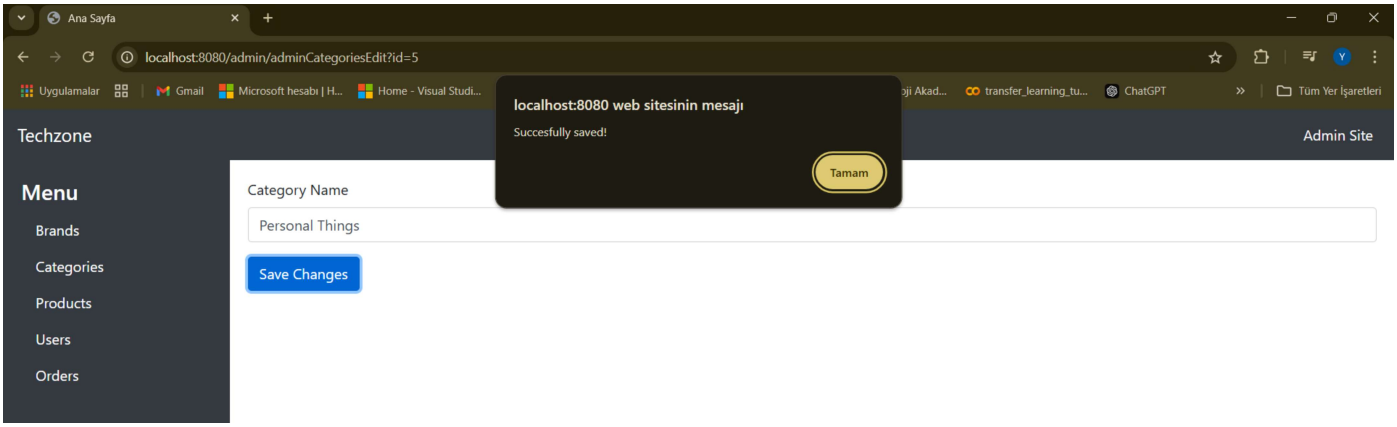


FIGURE 22. Enter Caption

7) In the product page, Admin has the authority to add products, set their name, price, stock, category, brand, image, and description, and this page is used for this and success message is showed in this example, but otherwise an error message is showed.

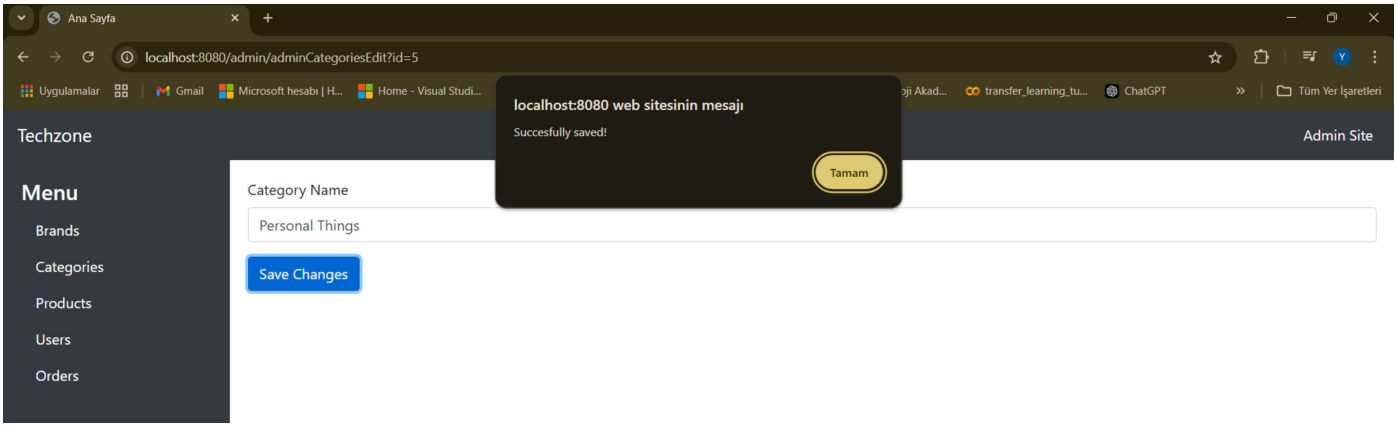


FIGURE 23. Enter Caption

8) All products including the last added product are shown on this page and all of them have an edit button:

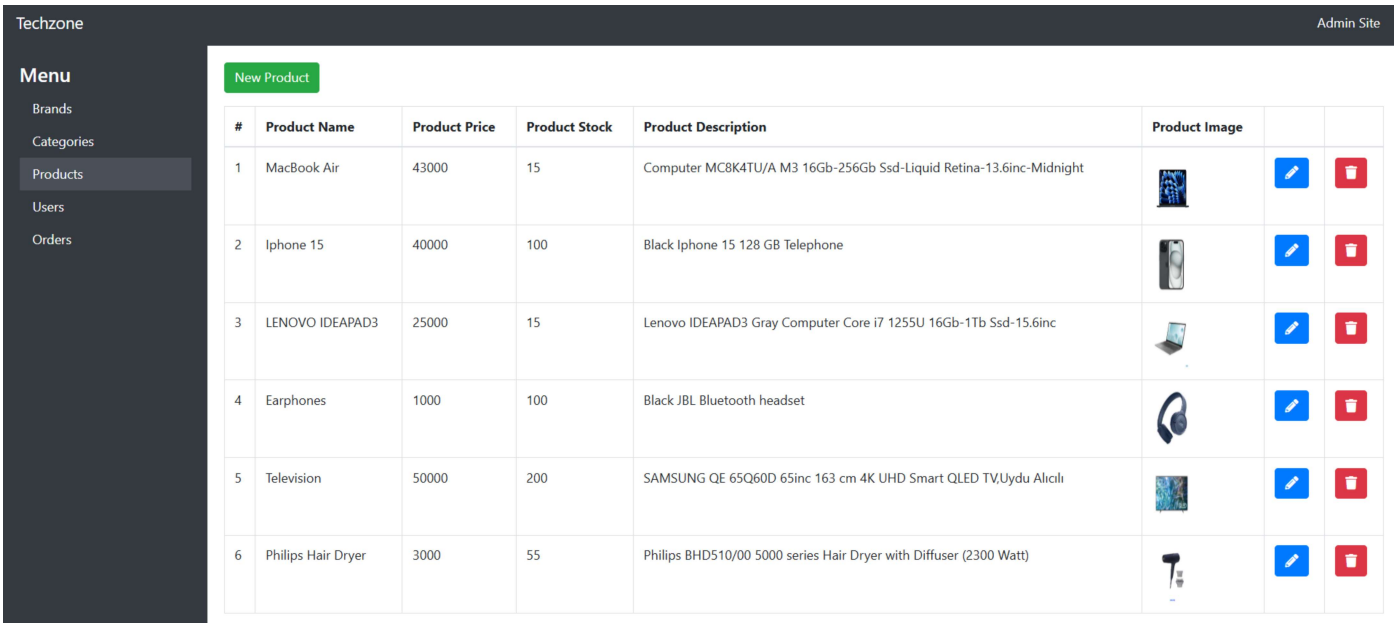


FIGURE 24. Enter Caption

9) Admin can see all users with this page and delete these users from the system when necessary, a confirmation message is displayed and with the approval with “Tamam”, page is reloaded and user is deleted and there is full page:

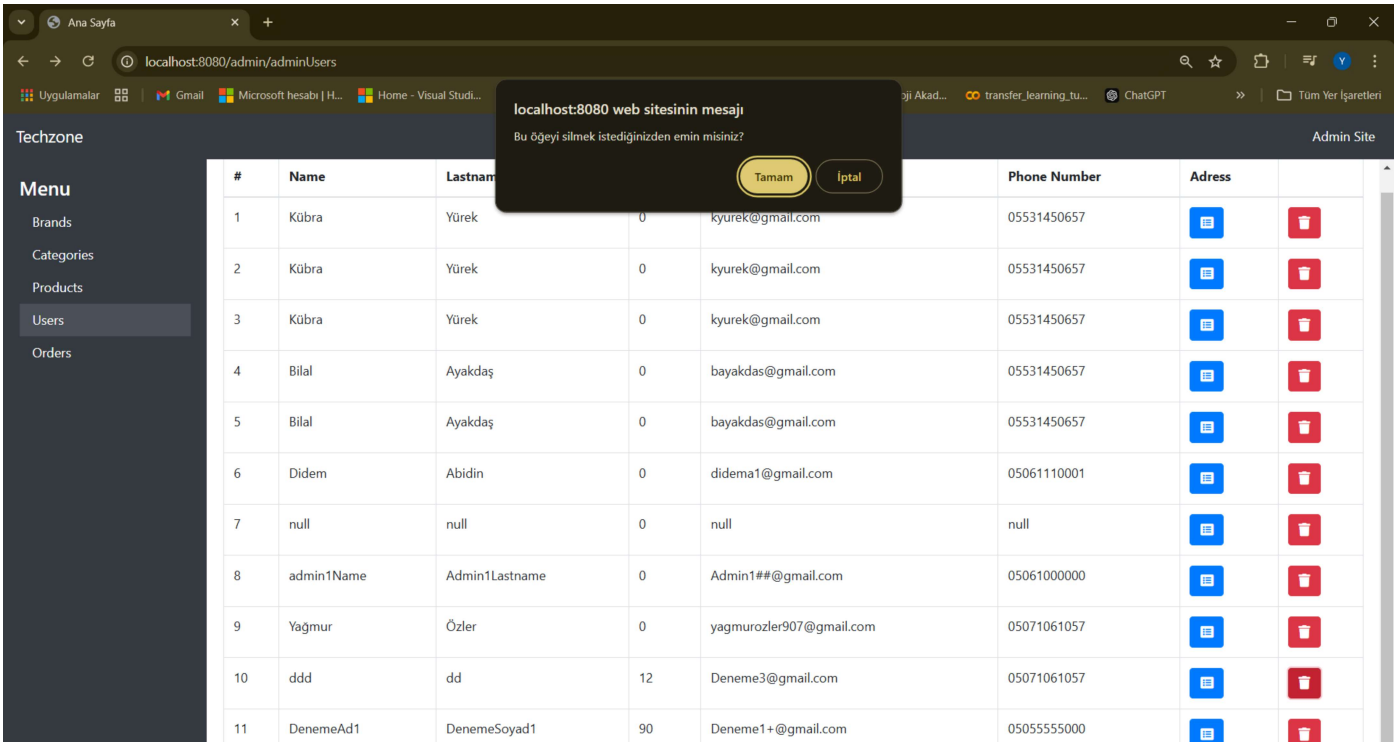


FIGURE 25. Enter Caption

10) After deleting the null user at seventh row, a success message is shown:

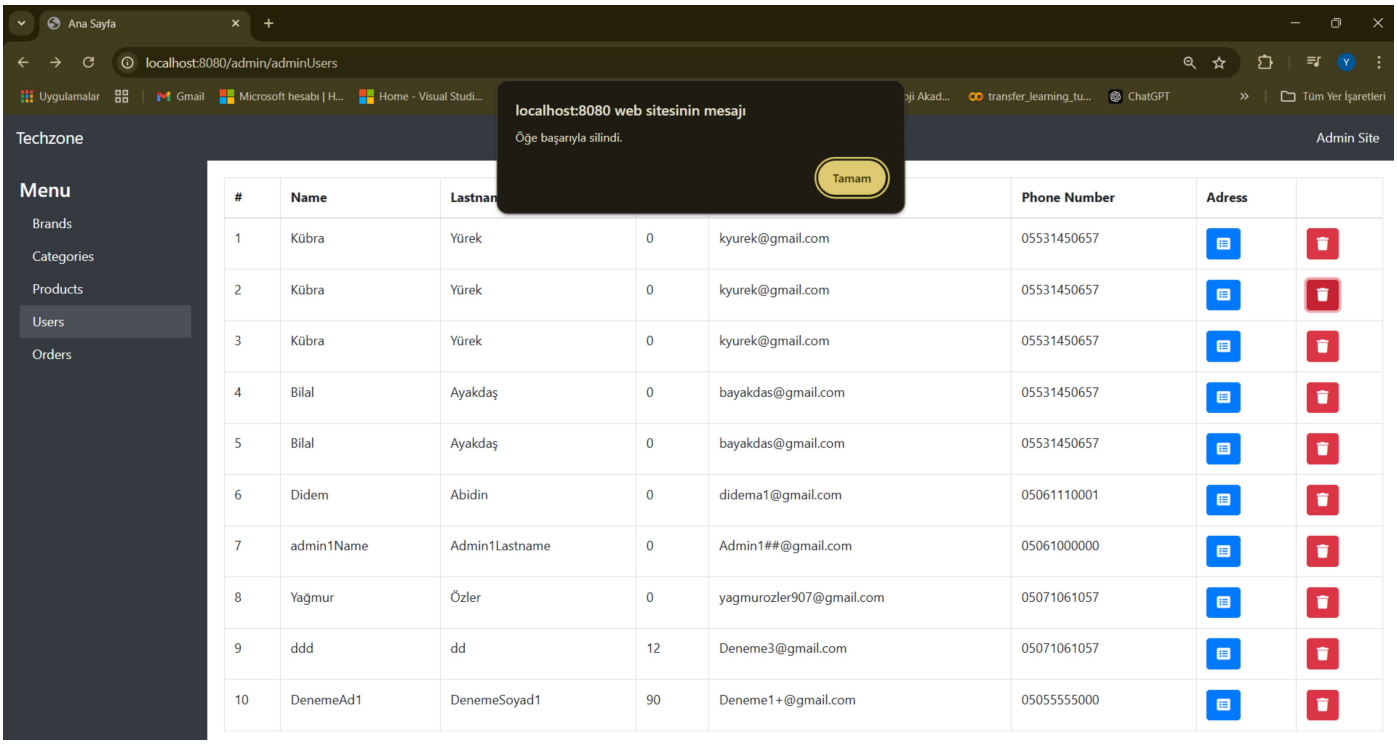


FIGURE 26. Enter Caption

11) Admin can see user details too:

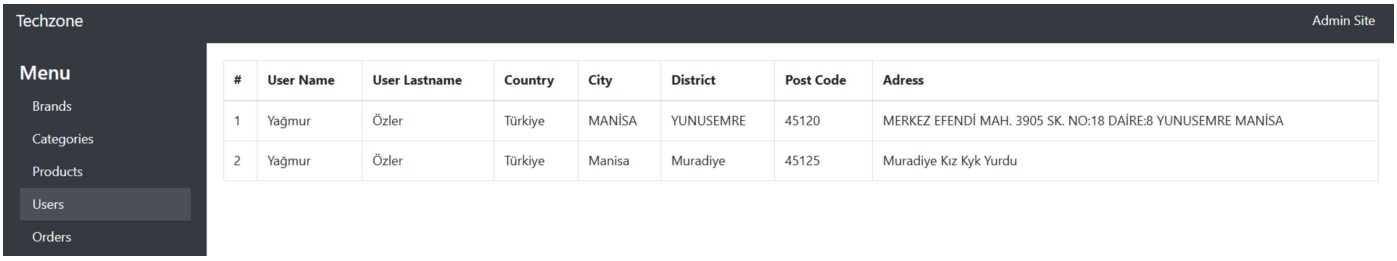
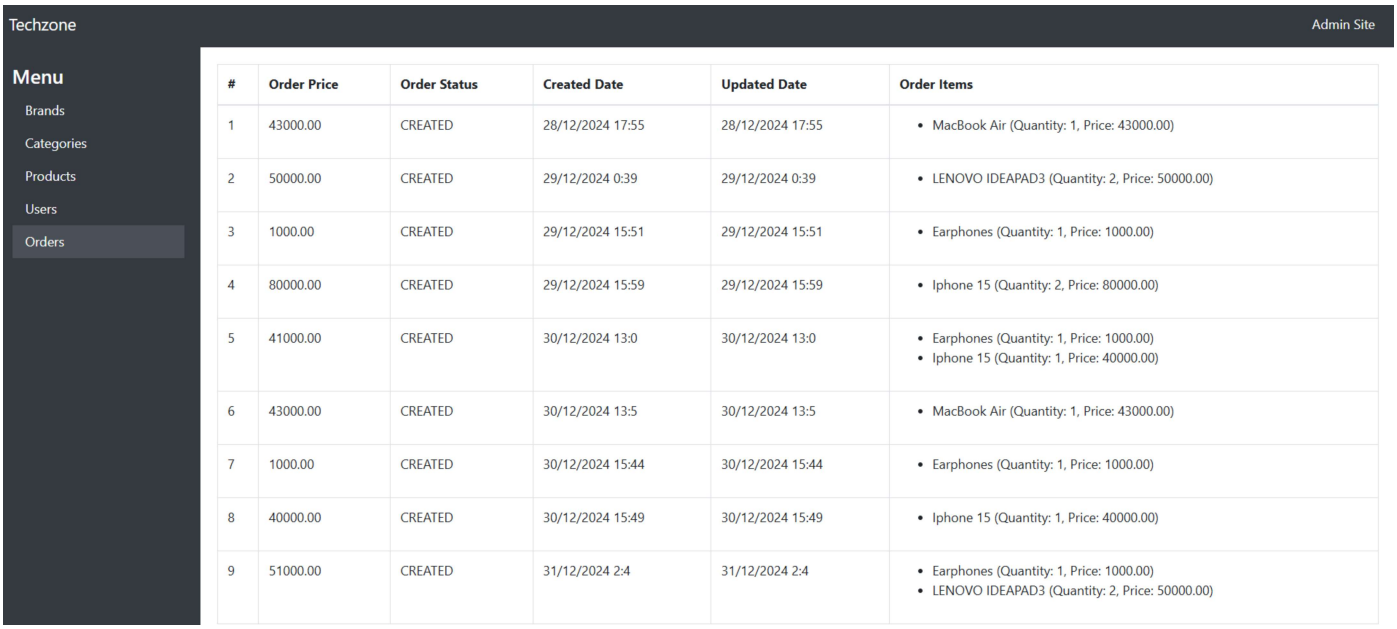


FIGURE 27. Enter Caption

12) Admin can see all orders made by all users and their price, status, creation date, update date, and all products in them, their quantity and price:



#	Order Price	Order Status	Created Date	Updated Date	Order Items
1	43000.00	CREATED	28/12/2024 17:55	28/12/2024 17:55	• MacBook Air (Quantity: 1, Price: 43000.00)
2	50000.00	CREATED	29/12/2024 0:39	29/12/2024 0:39	• LENOVO IDEAPAD3 (Quantity: 2, Price: 50000.00)
3	1000.00	CREATED	29/12/2024 15:51	29/12/2024 15:51	• Earphones (Quantity: 1, Price: 1000.00)
4	80000.00	CREATED	29/12/2024 15:59	29/12/2024 15:59	• Iphone 15 (Quantity: 2, Price: 80000.00)
5	41000.00	CREATED	30/12/2024 13:0	30/12/2024 13:0	• Earphones (Quantity: 1, Price: 1000.00) • Iphone 15 (Quantity: 1, Price: 40000.00)
6	43000.00	CREATED	30/12/2024 13:5	30/12/2024 13:5	• MacBook Air (Quantity: 1, Price: 43000.00)
7	1000.00	CREATED	30/12/2024 15:44	30/12/2024 15:44	• Earphones (Quantity: 1, Price: 1000.00)
8	40000.00	CREATED	30/12/2024 15:49	30/12/2024 15:49	• Iphone 15 (Quantity: 1, Price: 40000.00)
9	51000.00	CREATED	31/12/2024 2:4	31/12/2024 2:4	• Earphones (Quantity: 1, Price: 1000.00) • LENOVO IDEAPAD3 (Quantity: 2, Price: 50000.00)

FIGURE 28. Enter Caption

C. CUSTOMER SITE

1) Home page and trying to add products to the cart and favorites without logging in and their error messages are displayed to the user:

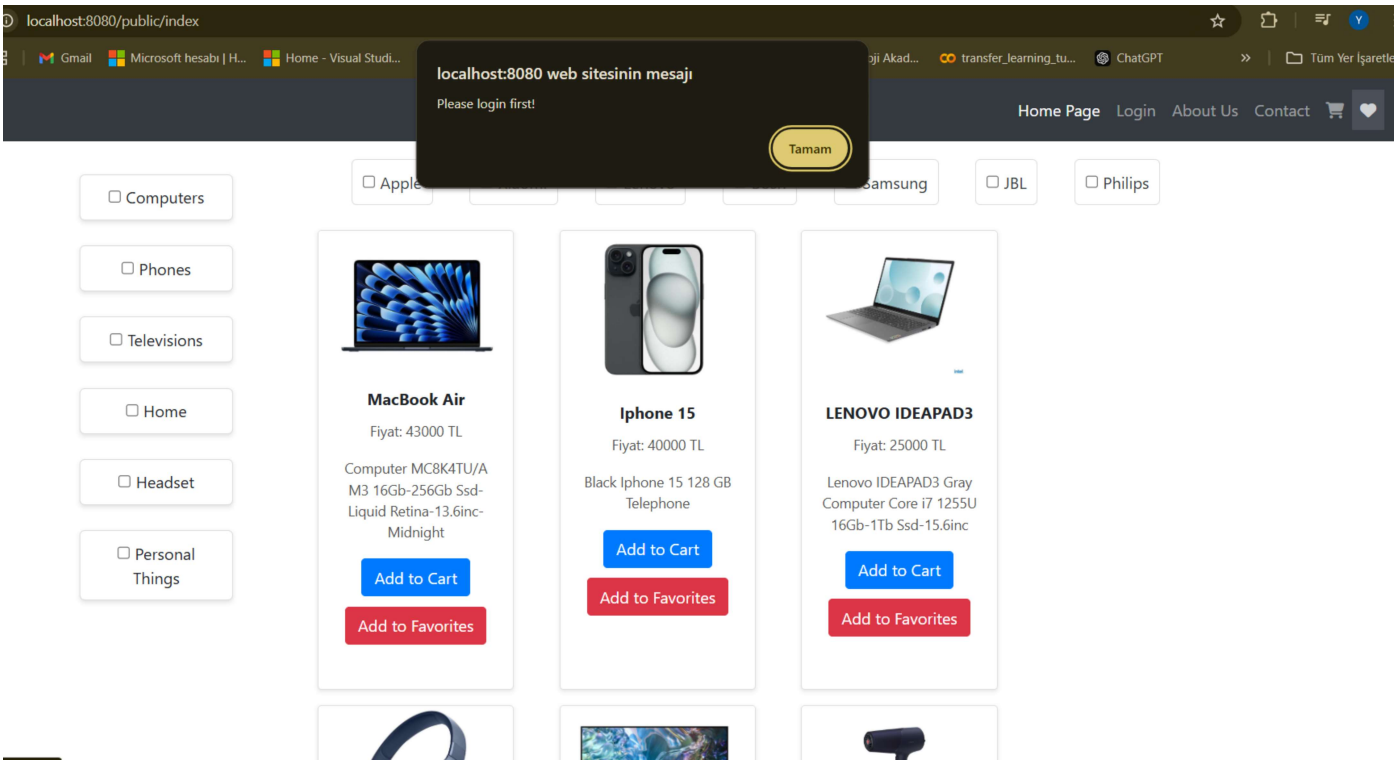


FIGURE 29. Enter Caption

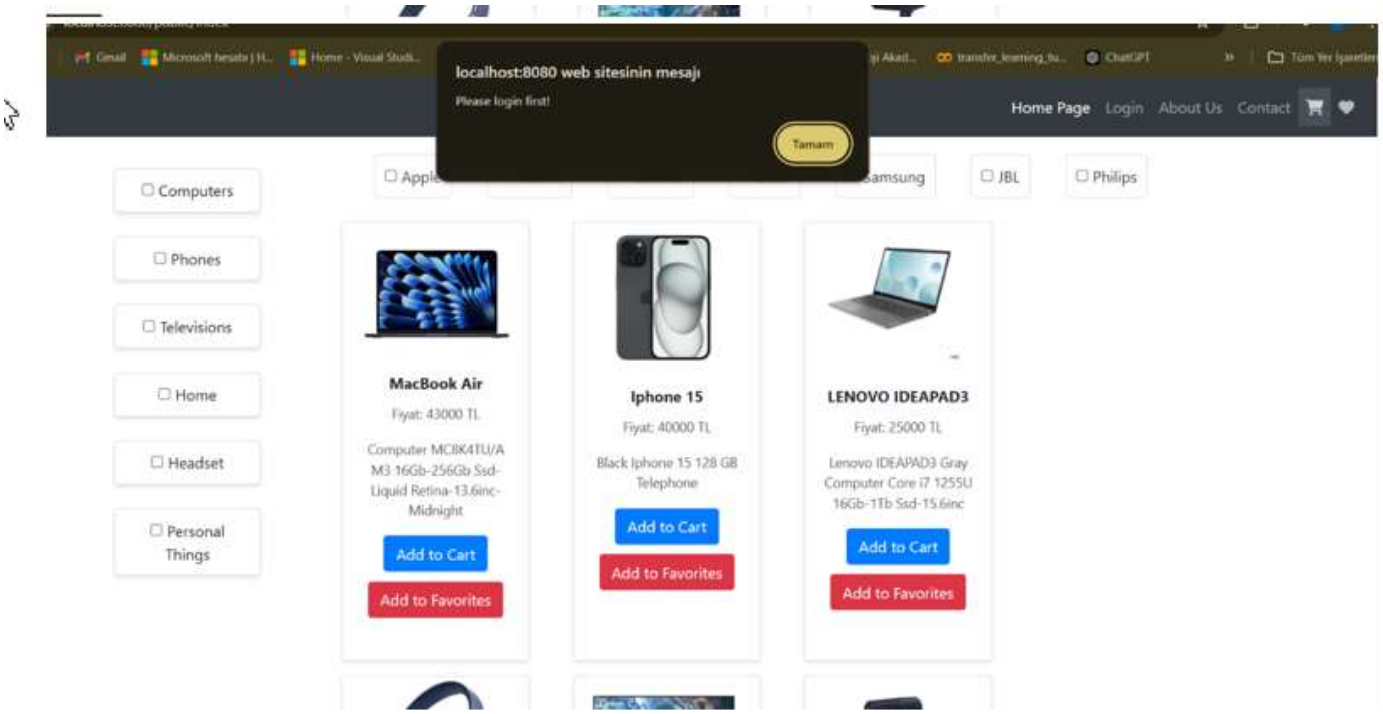


FIGURE 30. Enter Caption

2) Home page and trying to go first to my favorites and then my cart with the icons in the navigation bar, without logging in and their error messages are displayed to the user:

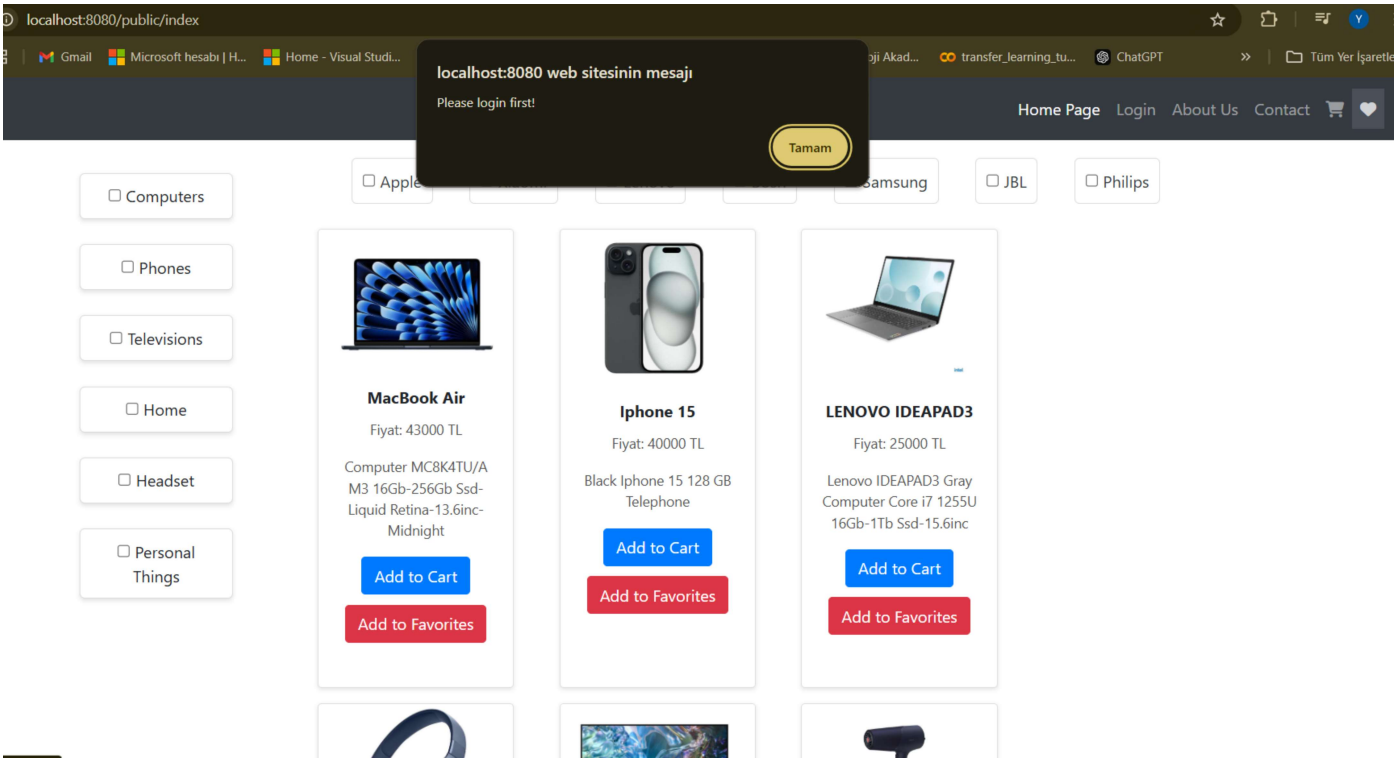


FIGURE 31. Enter Caption

3) After pushing login button, login page comes and if user mail and password is correct, successful login message is shown, otherwise try again error is shown :

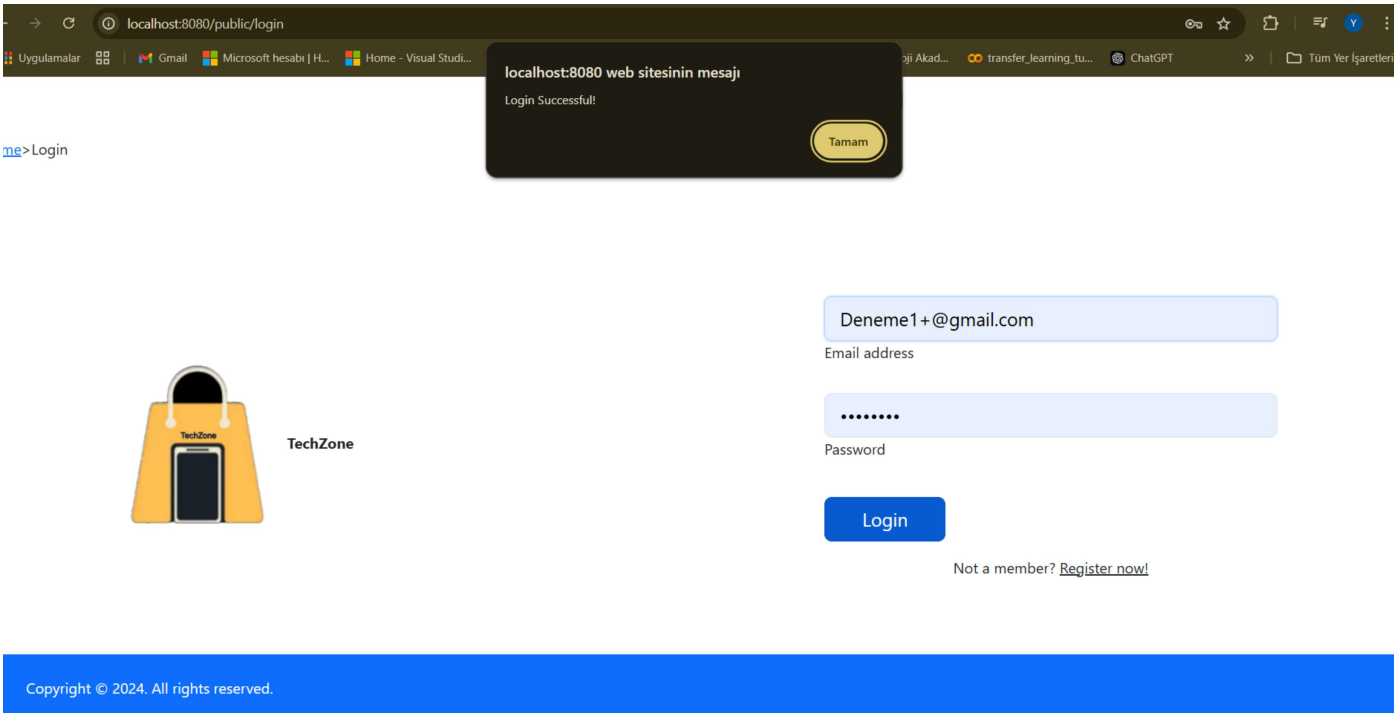


FIGURE 32. Enter Caption

4) If users forget their password, an exchange page is comes with Email, old password, new password and confirmation password and the error messages shown are as follows:

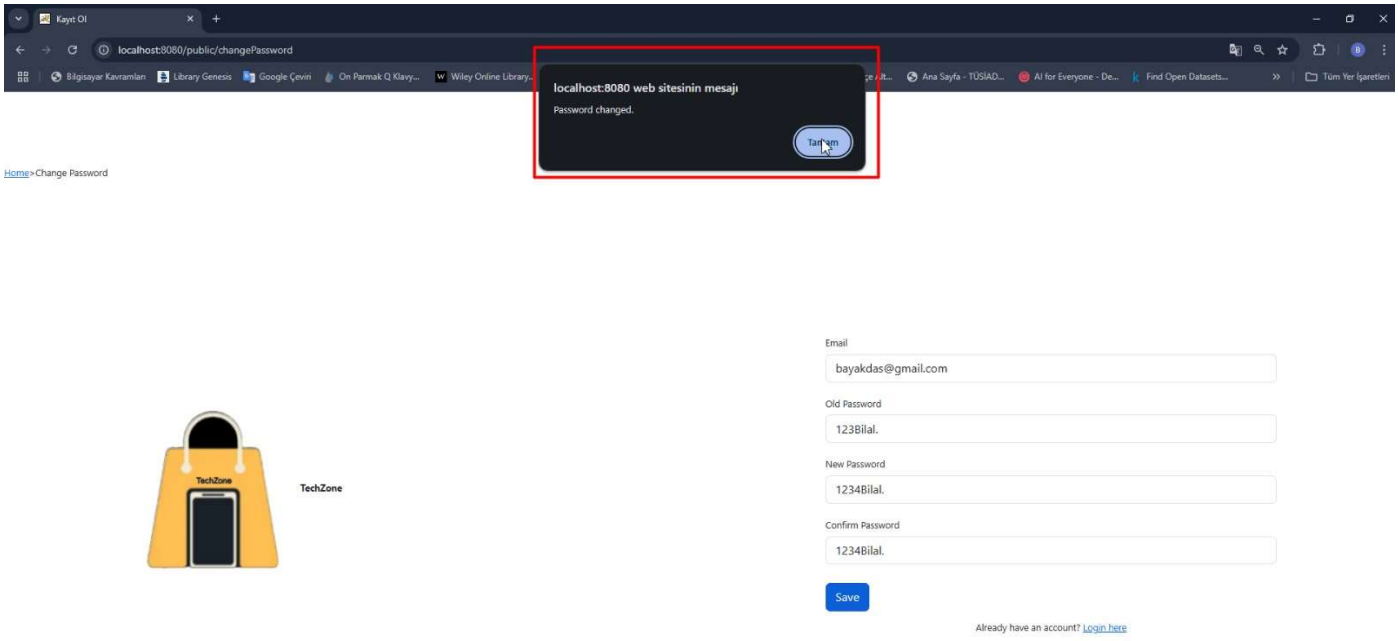


FIGURE 33. Enter Caption

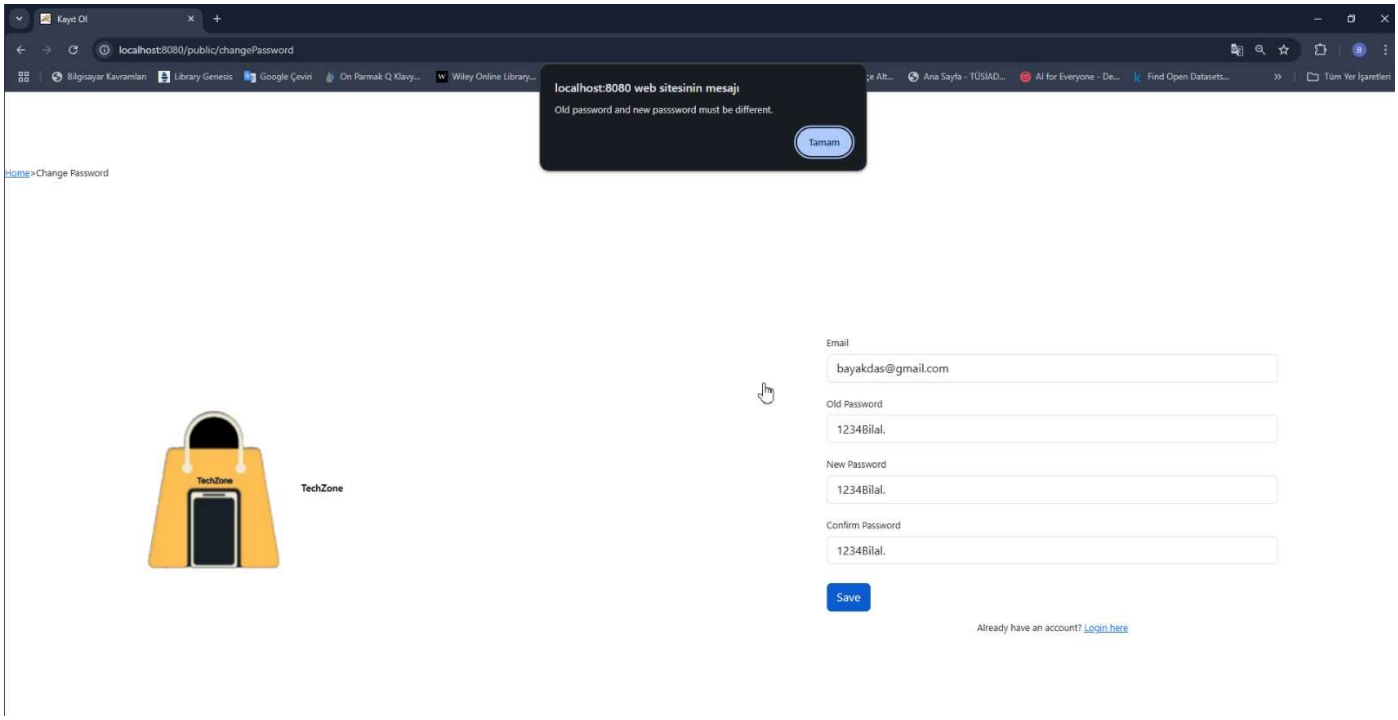


FIGURE 34. Enter Caption

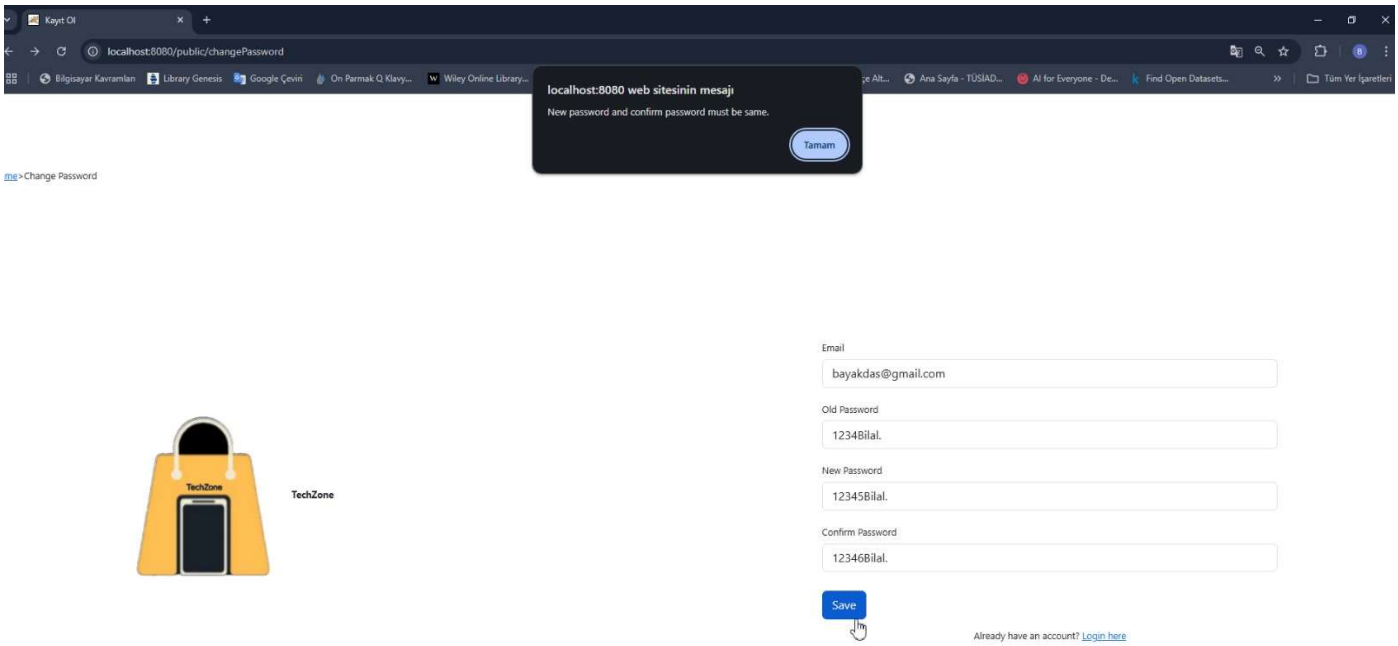


FIGURE 35. Enter Caption

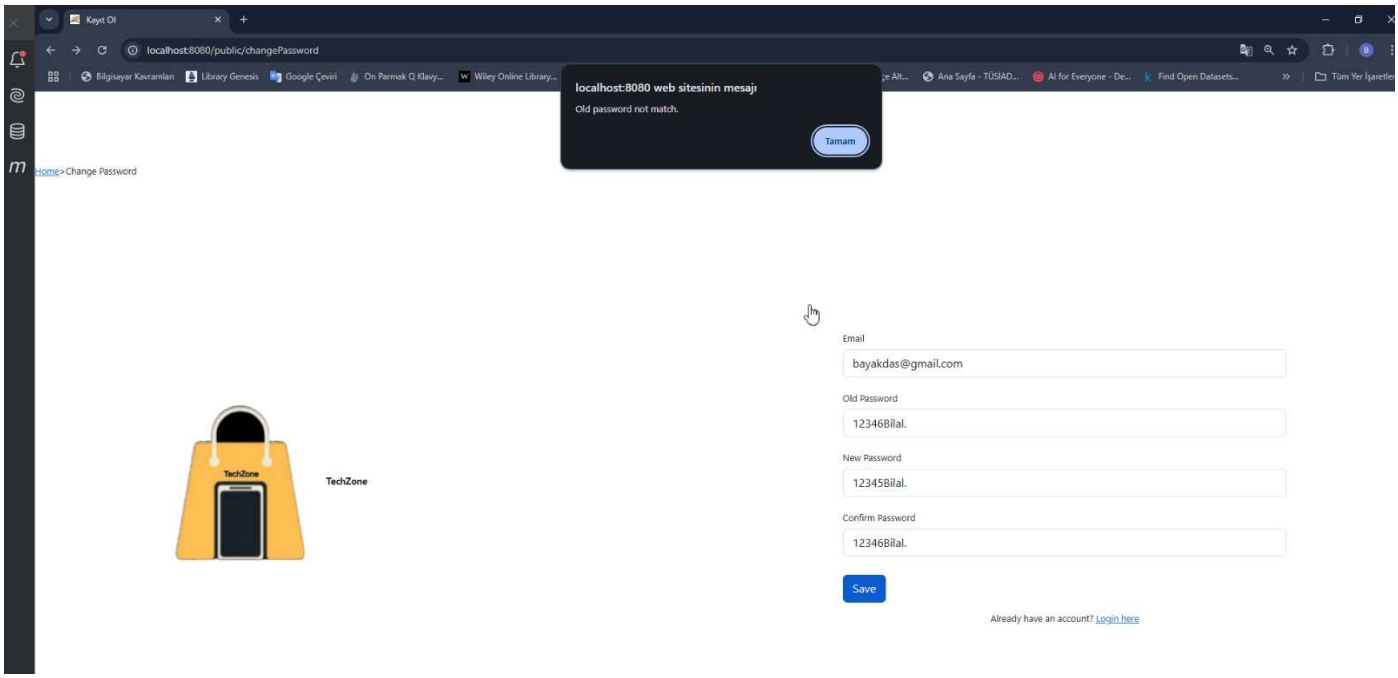


FIGURE 36. Enter Caption

5) If User does not have an account and want to register, the register link can be clicked and the registration page will open. On this page, there are requirements for each piece of information and if any information is entered incorrectly, messages regarding this and the correct entry will be displayed below that entry as seen here:

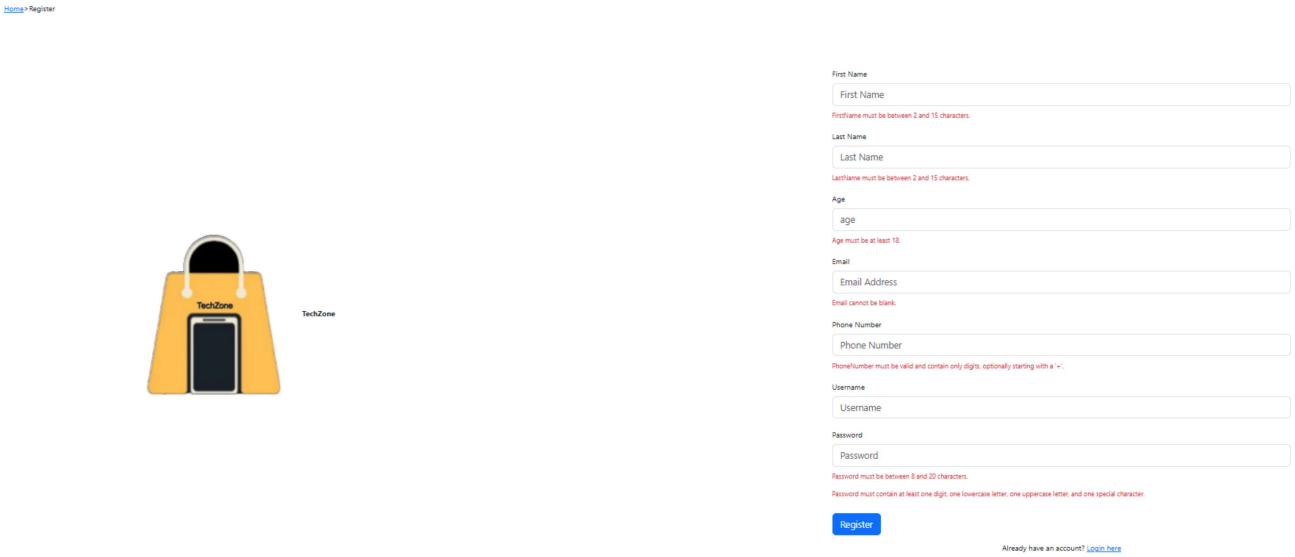


FIGURE 37. Enter Caption

After pressing the register buttons, the user is directed to the login page.

6) Filtering the products with many options is possible but here with only one category, and two category with one brand is shown in these images:

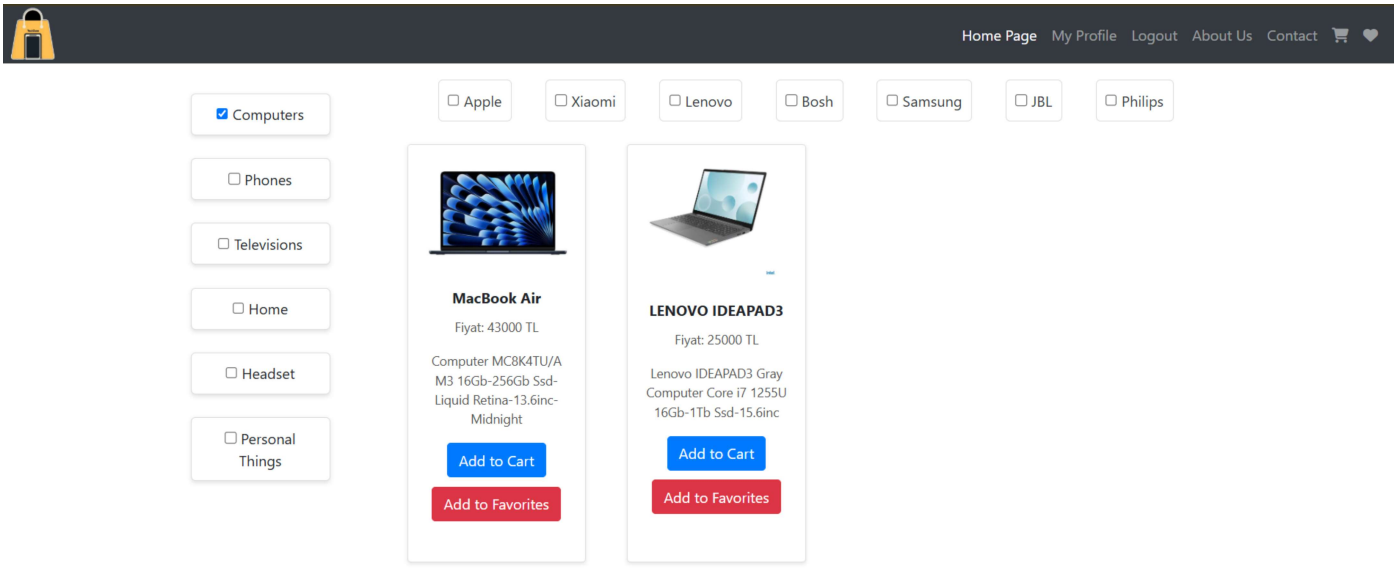


FIGURE 38. Enter Caption

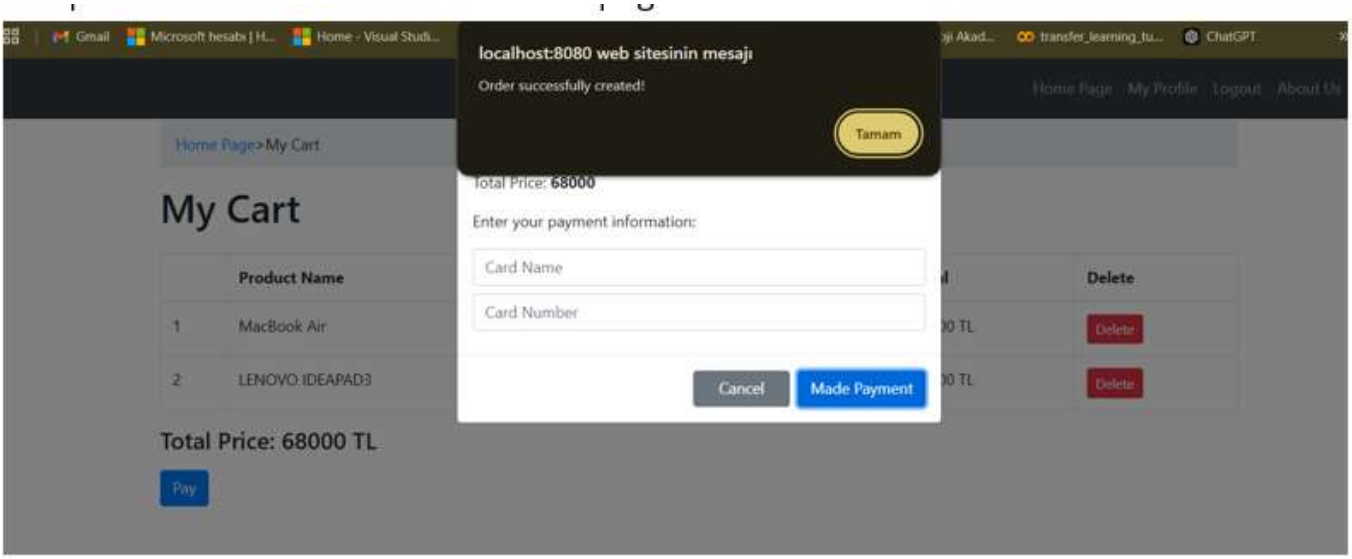


FIGURE 39. Enter Caption

Also, home page is shown here if user logged in, navigation bar's Login button changes with Logout and My Profile tab is added.

7) If user logged in, pressing the favorite button is enabled, and there is success message for it, here for television:

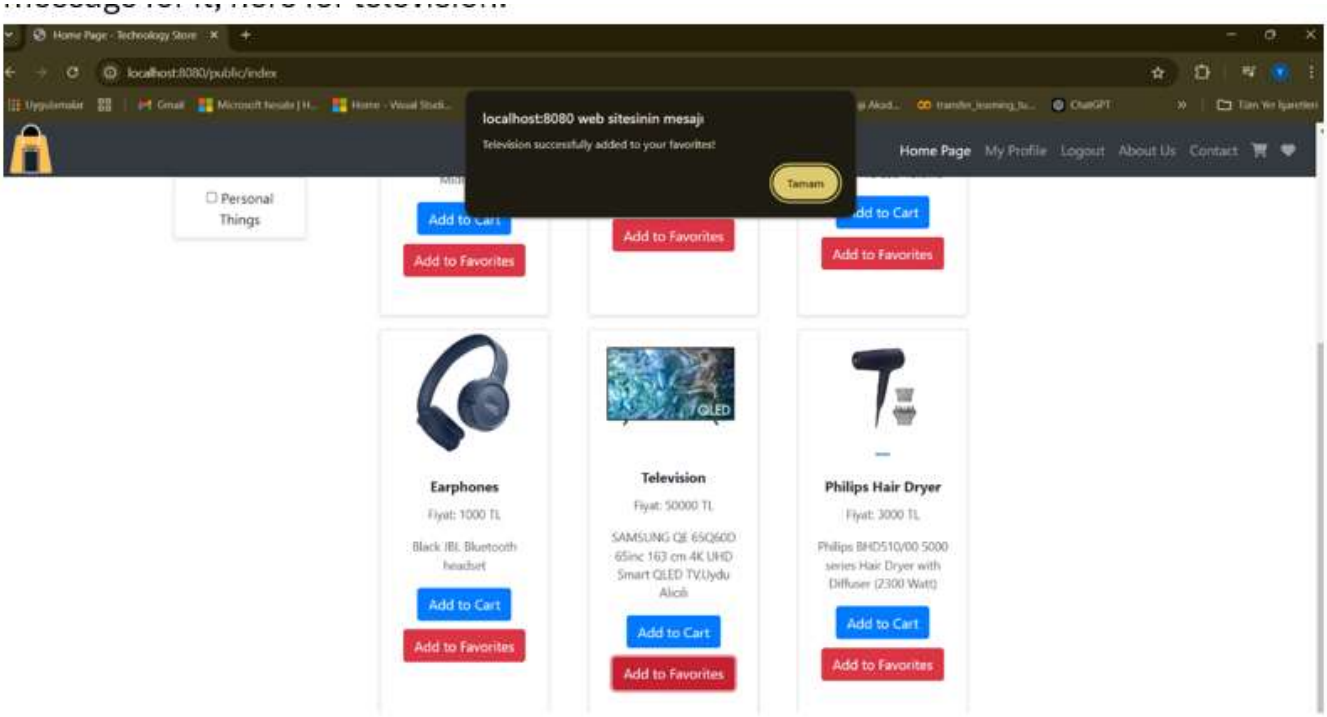


FIGURE 40. Enter Caption

8) If the same product which is television here is trying to added to favorites, an error message is shown that explains it already added to favorites:

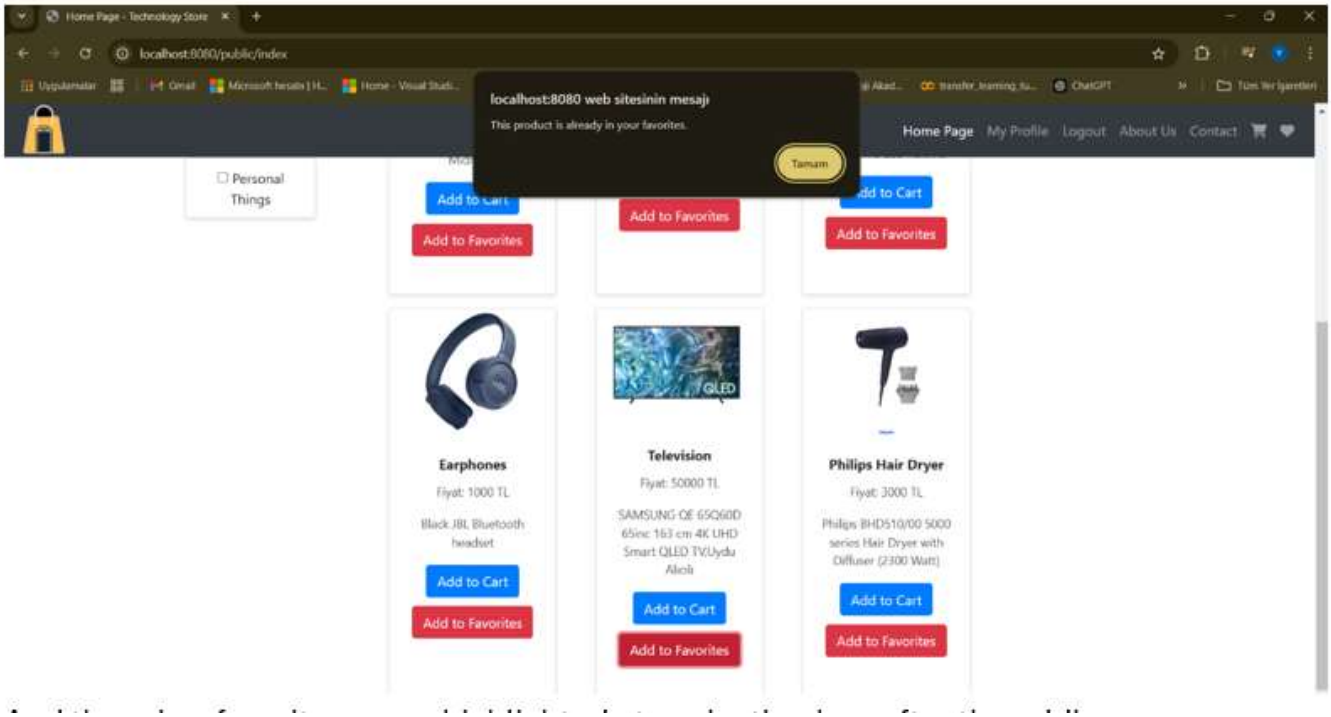


FIGURE 41. Enter Caption

9) And there is a favorites page, highlighted at navigation bar, after the adding the television and its delete button:

FIGURE 42. Enter Caption

10) If user logged in, pressing the cart button is enabled, and there is success message for it, here for computer product:

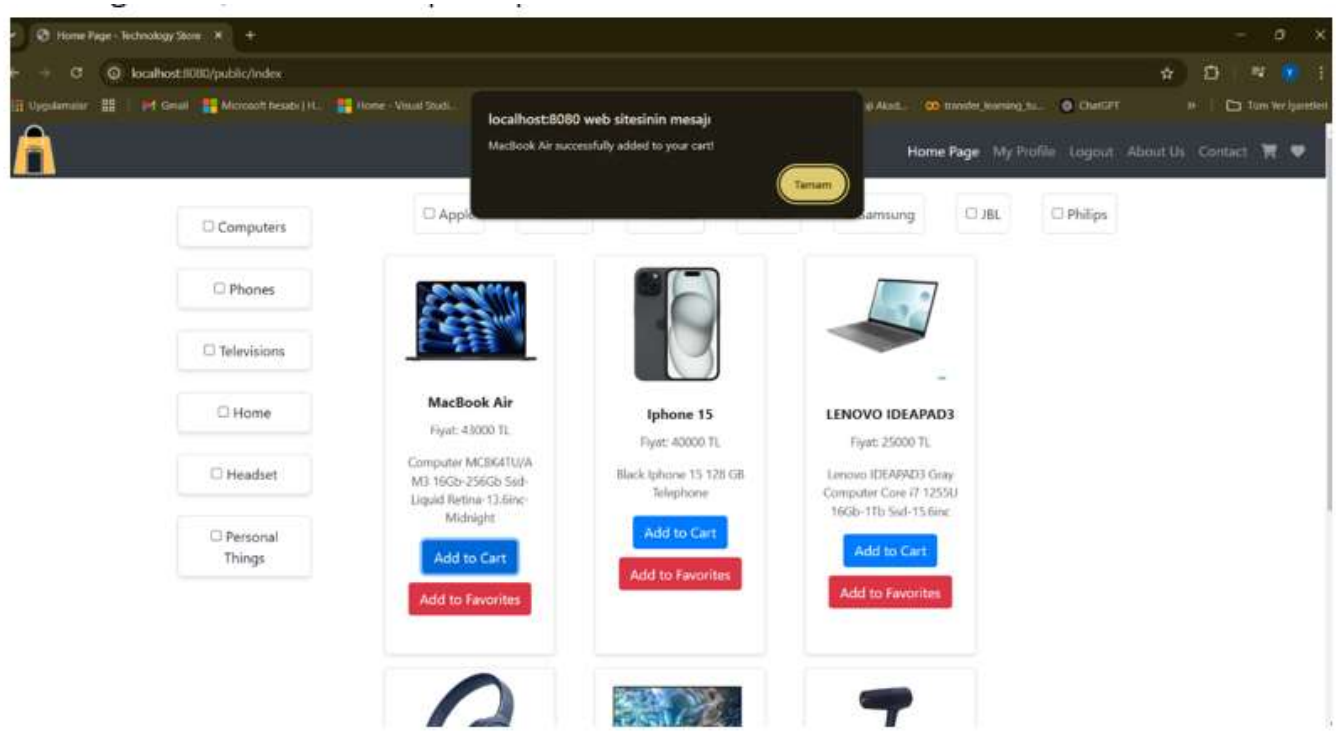


FIGURE 43. Enter Caption

11) There is my cart page for user and success message for deleting a product, after that page is reloaded:

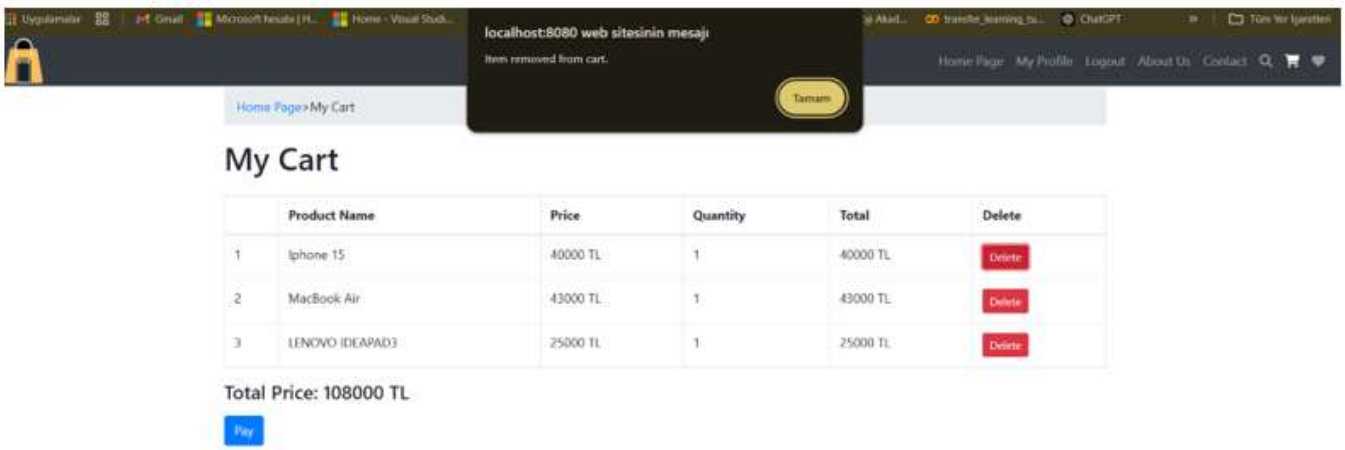


FIGURE 44. Enter Caption

12) If pay button is pressed, a modal comes to the page for card information with pay and cancel buttons:

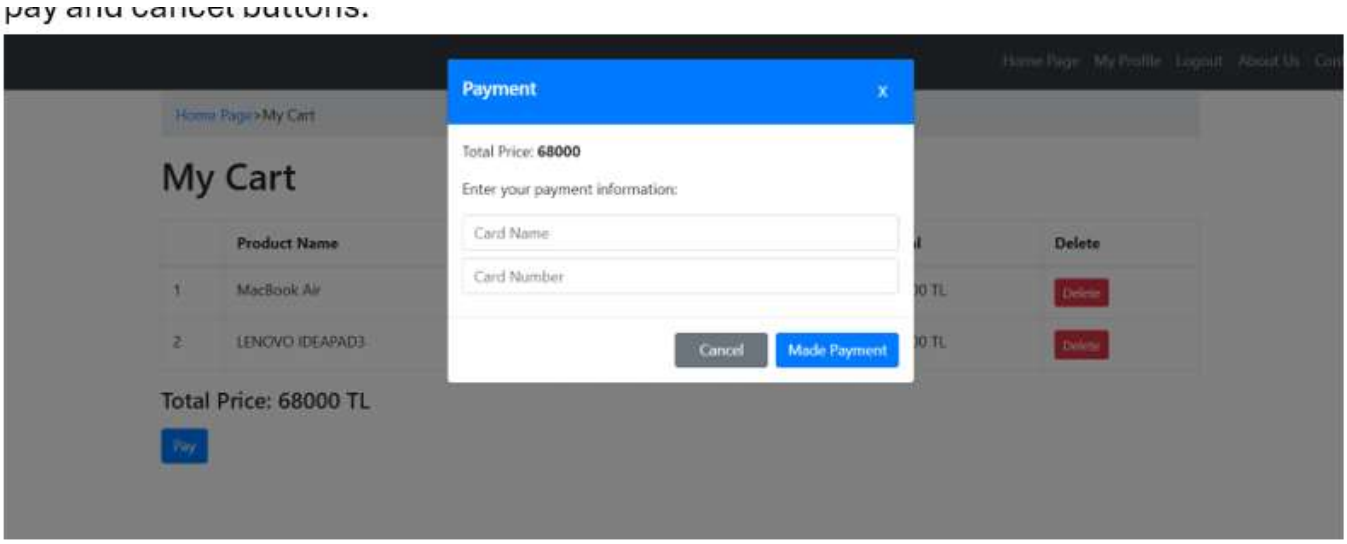


FIGURE 45. Enter Caption

13) A message is shown if payment is successful, and with that order is created and products in the cart is deleted and page is reloaded :

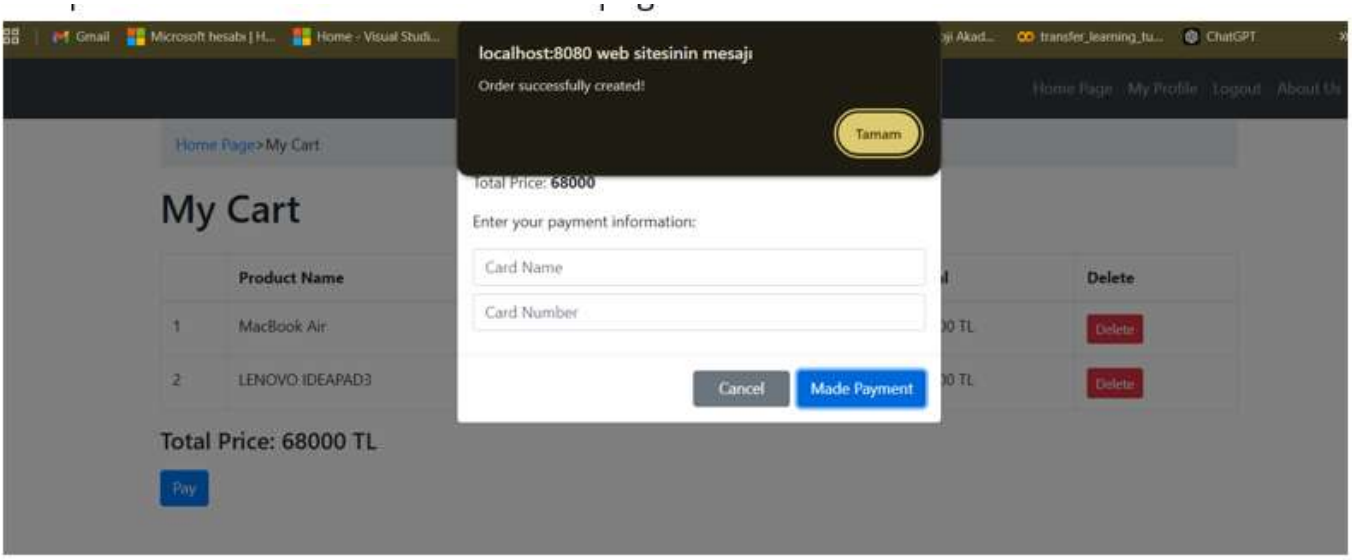


FIGURE 46. Enter Caption

14) The My Profile page displays user information and the user can change it if they wish with pressing the Edit My Profile button:

FIGURE 47. Enter Caption

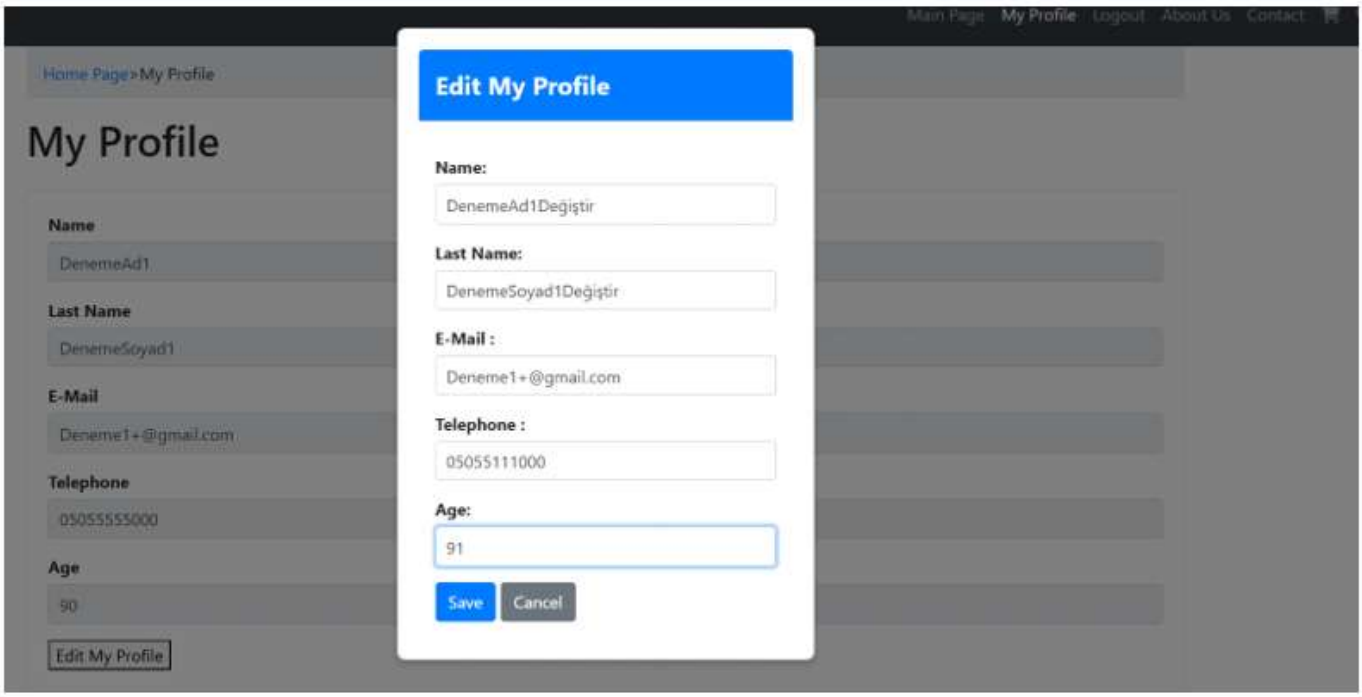


FIGURE 48. Enter Caption

And message is shown is saveing the new information is successful:

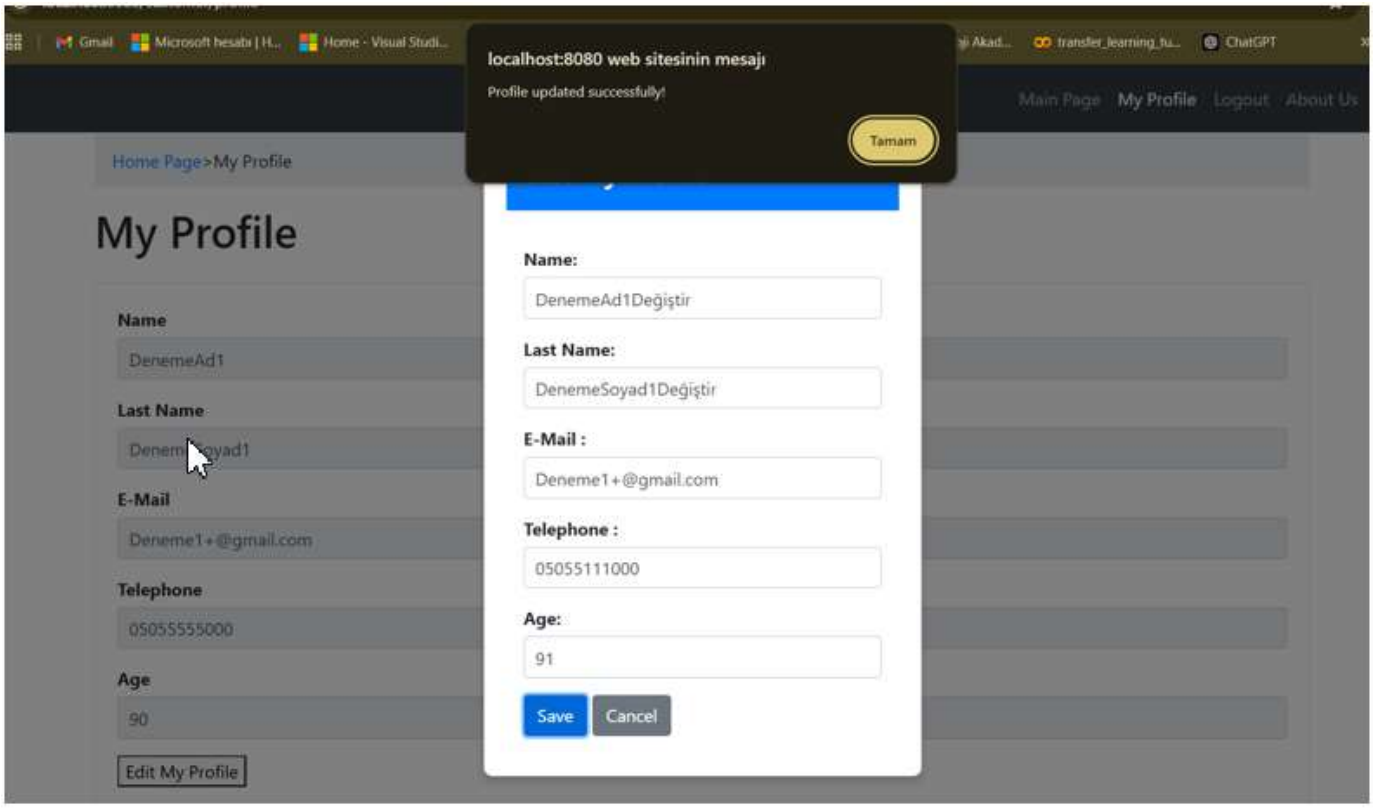


FIGURE 49. Enter Caption

15) User can see its addresses and can add-delete if necessary, there is shown a modal for adding an address and its success messaage:

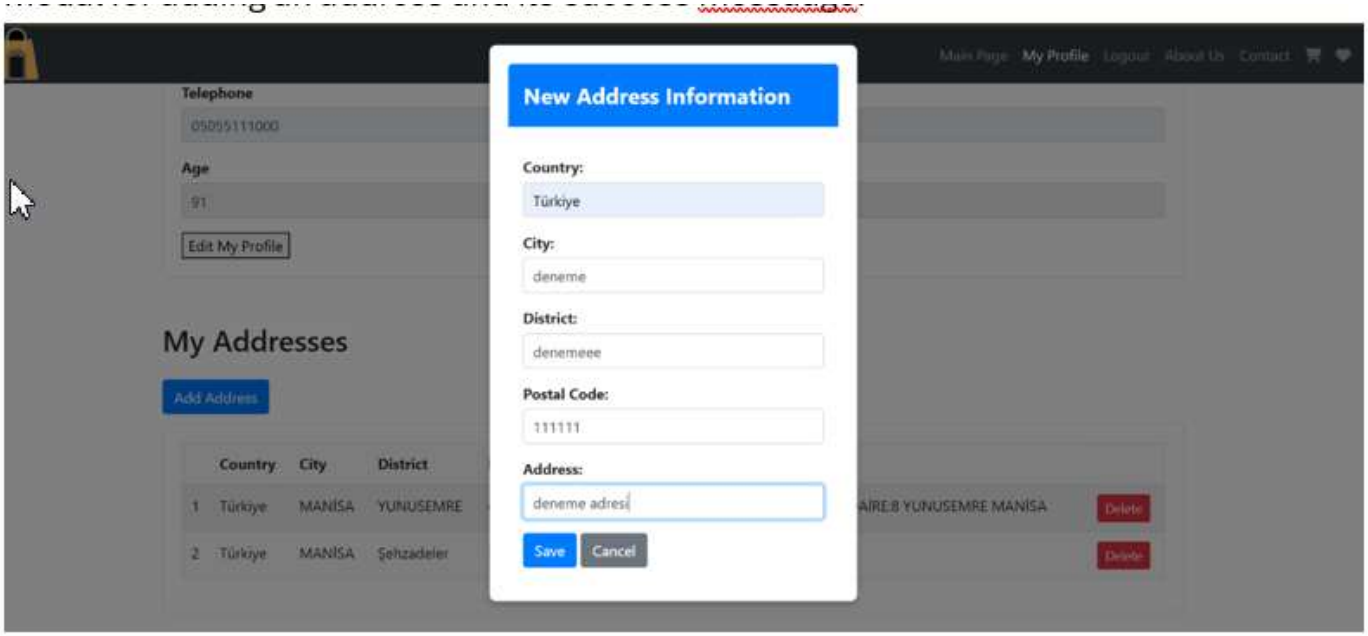


FIGURE 50. Enter Caption

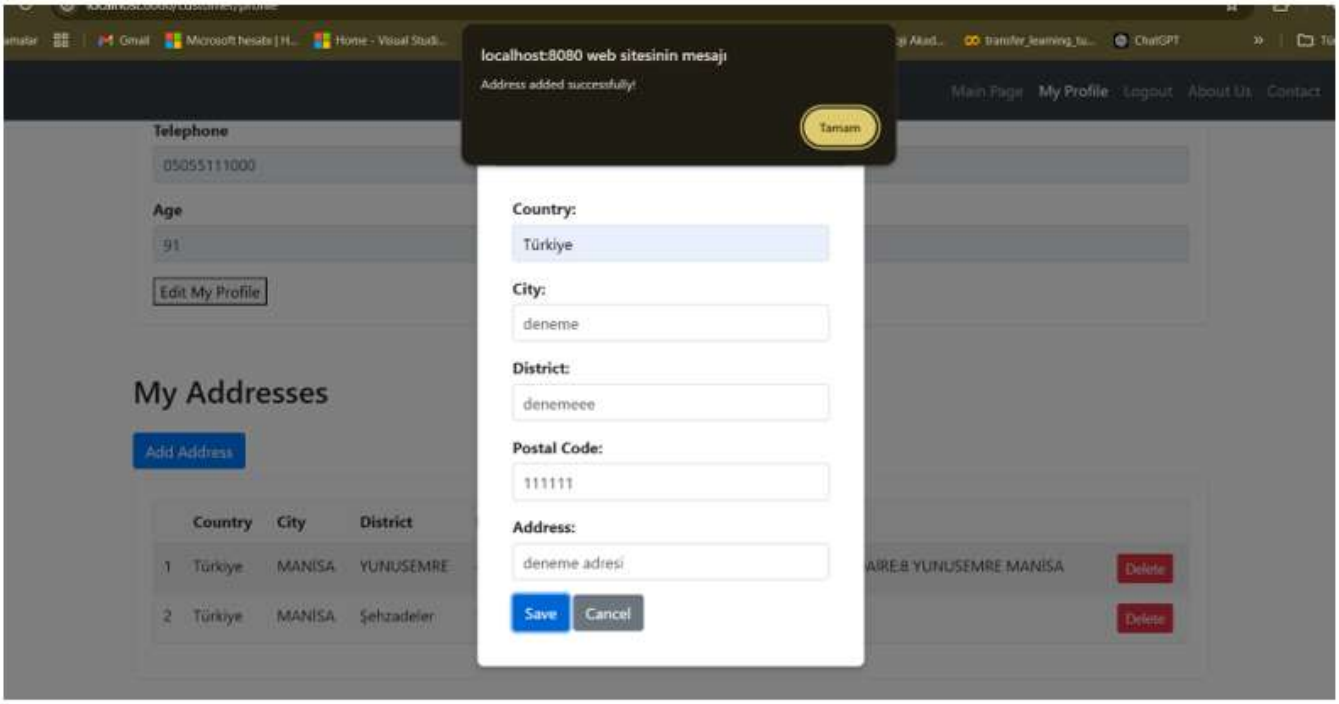


FIGURE 51. Enter Caption

16) After adding the new address, page is reloaded, and also in this My Profile page, there is that user's past orders:

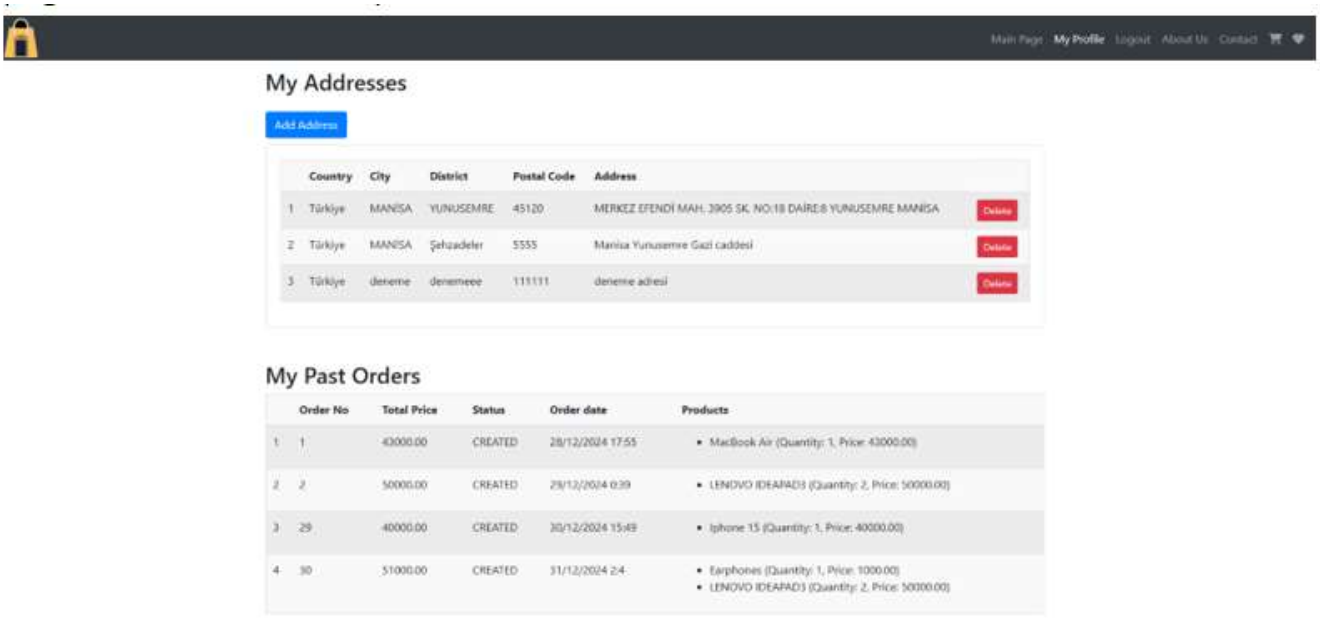


FIGURE 52. Enter Caption

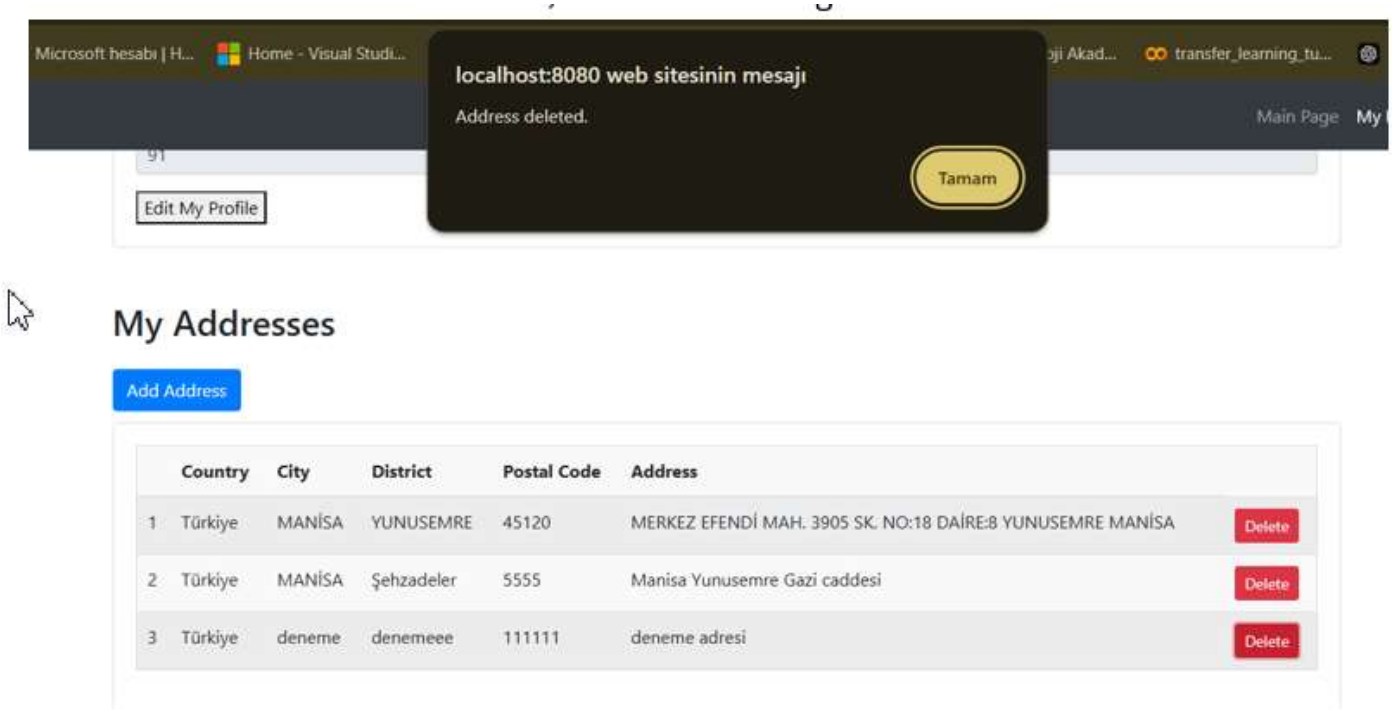


FIGURE 54. Enter Caption

17) If user wants to delete an address, there is a message if it is successful:

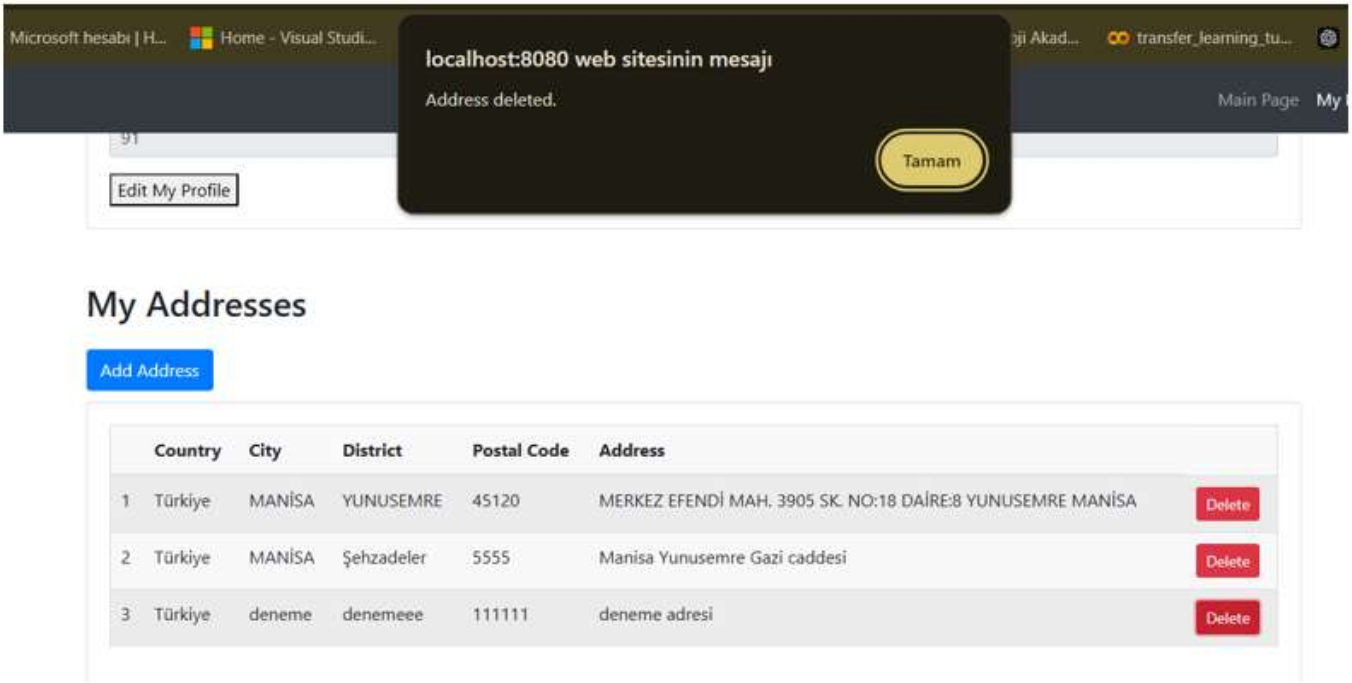


FIGURE 53. Enter Caption

18) When user wants to logged out, a message shown if it is successful and navigation bars elements change back to the login, also this functionality is shown at About Us page:

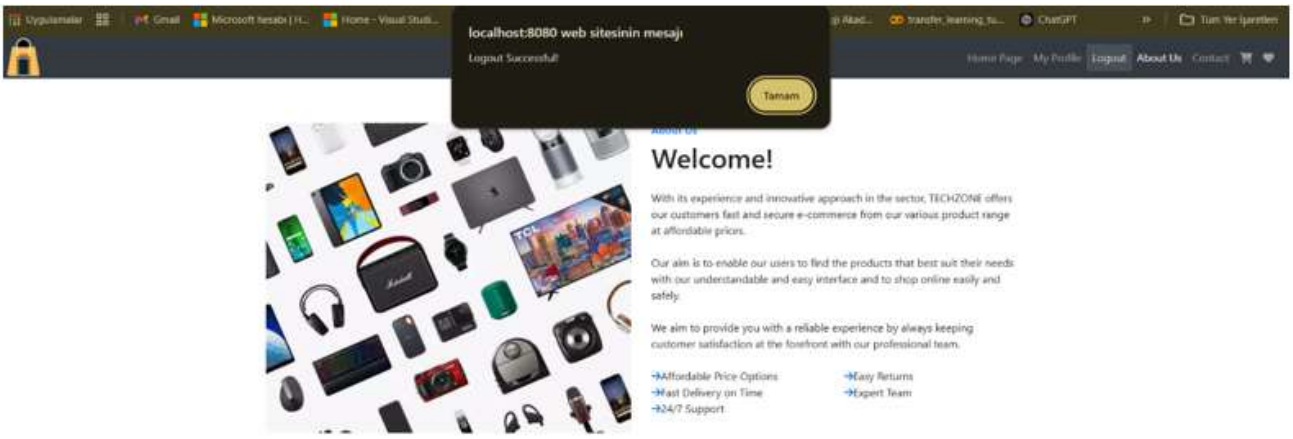


FIGURE 55. Enter Caption

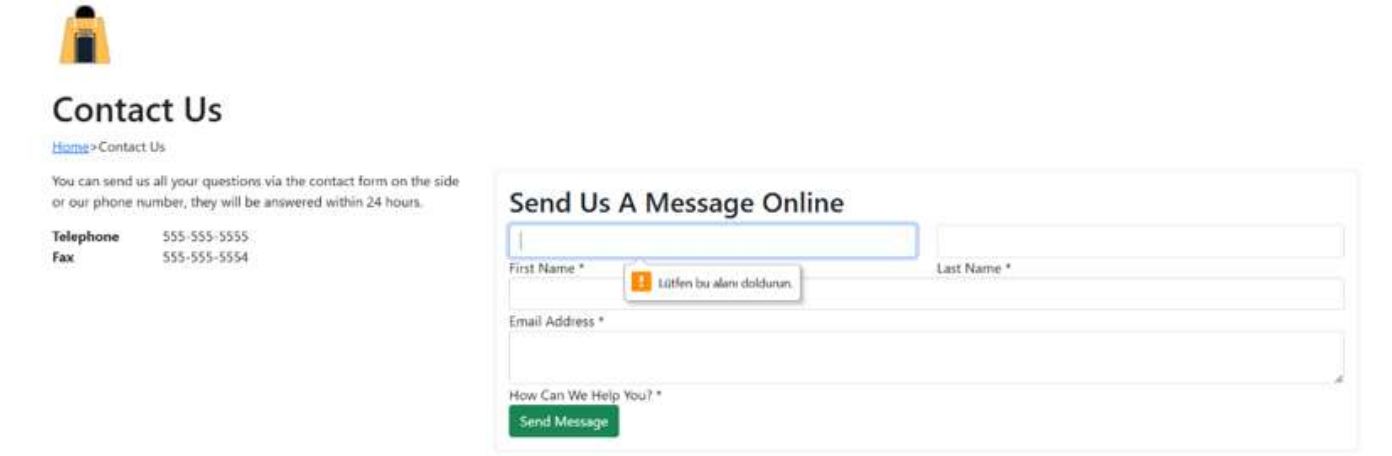


FIGURE 56. Enter Caption

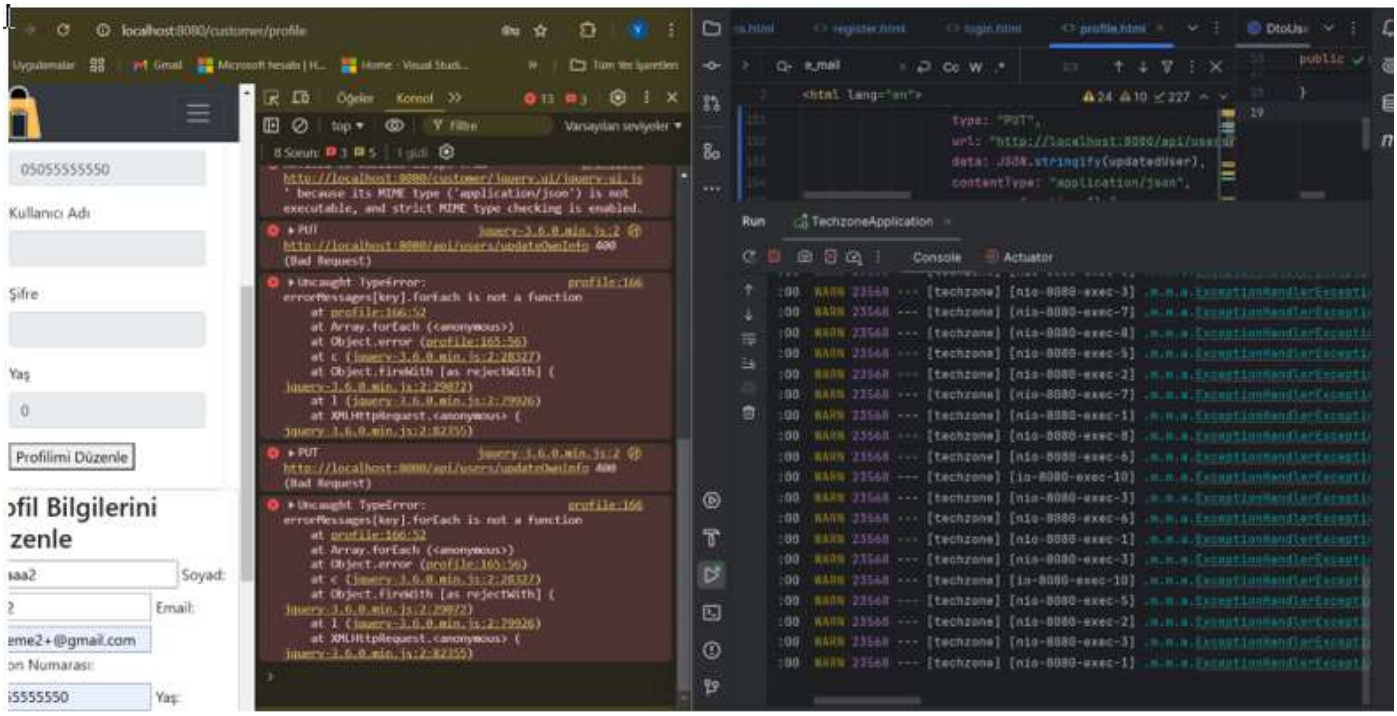
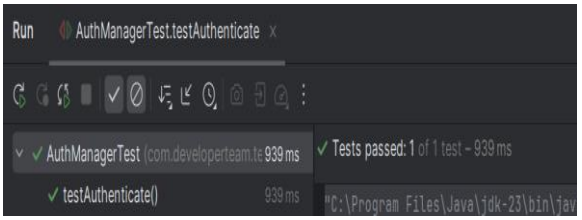


FIGURE 57. Enter Caption

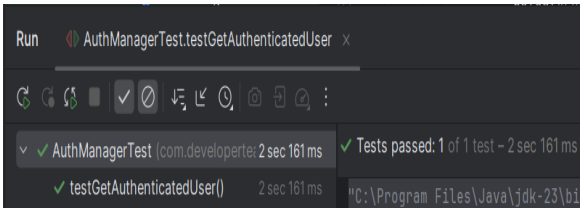
19) On the Contact page, if labels are empty, an error message is displayed:
Also, at the web side console, there were error messages like this:

AuthManager Tests

This code defines a unit test to verify the user authentication process by testing an AuthRequest object with the authentication provider.

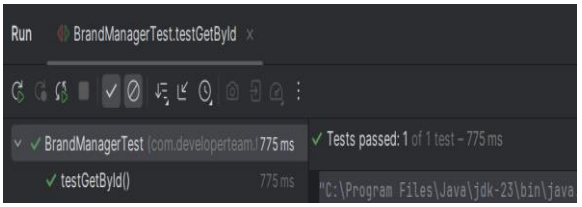


This code tests the retrieval of the authenticated user by verifying the email and ID of the user obtained from the authentication manager and JWT service.

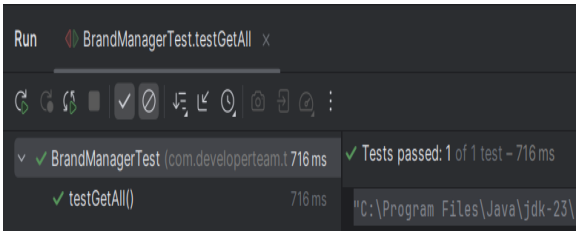


BrandManager Tests

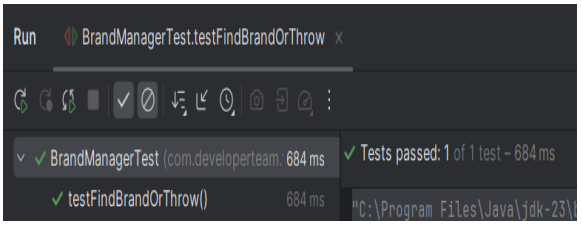
This method tests retrieving all brands and verifies their IDs and names.



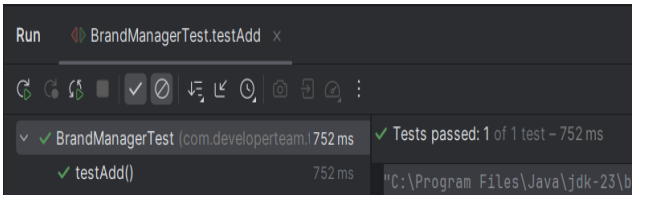
This method tests retrieving all brands and verifies their IDs and names.



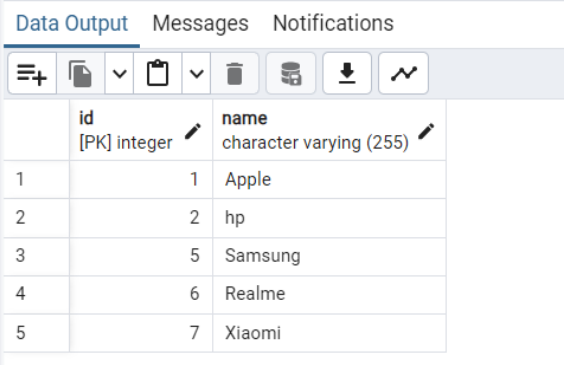
This method tests finding a brand by ID, ensuring it exists and matches the expected attributes.



This method tests adding a new brand, ensuring the brand is saved correctly and matches the expected name.

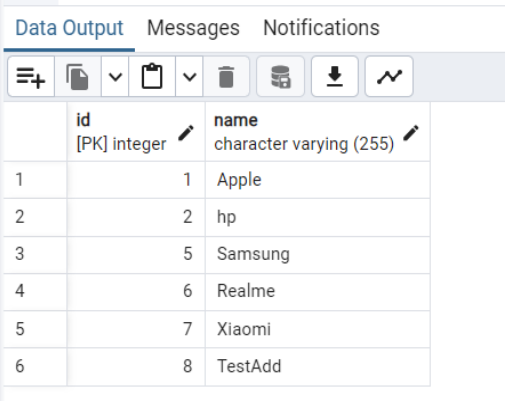


Database before testAdd() method runs:



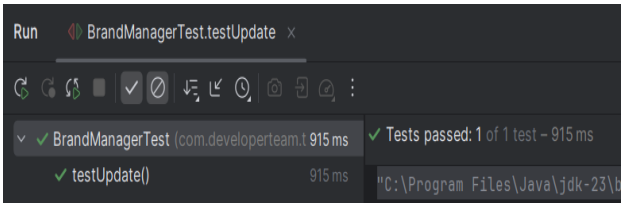
	id [PK] integer	name character varying (255)
1	1	Apple
2	2	hp
3	5	Samsung
4	6	Realme
5	7	Xiaomi

Database after testAdd() method runs:



	id [PK] integer	name character varying (255)
1	1	Apple
2	2	hp
3	5	Samsung
4	6	Realme
5	7	Xiaomi
6	8	TestAdd

This method tests updating an existing brand, verifying the updated name in both the result and the database.

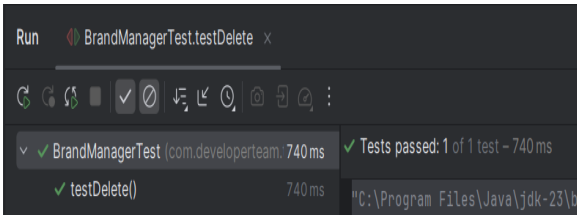


Database after testUpdate() runs:

Data Output | Messages | Notifications

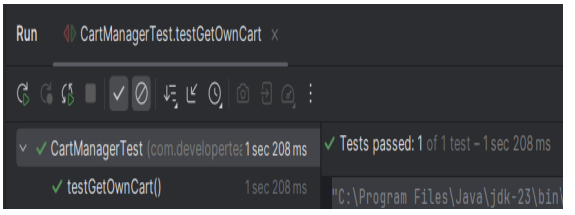
	id [PK] integer	name character varying (255)
1	1	Apple
2	2	hp
3	5	Samsung
4	6	Realme
5	7	Xiaomi
6	8	testUpdate

This method tests deleting a brand by confirming its existence before removal.

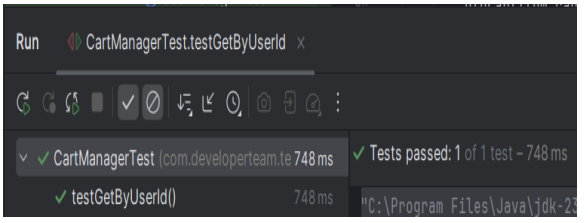


Cart Tests

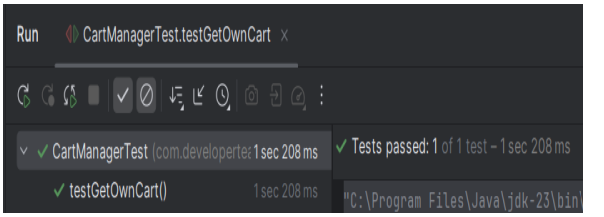
This method tests retrieving all carts, ensuring that the list is not empty and verifying the IDs of the first and second cart items.



This method tests fetching cart items by user ID, verifying the quantity and product ID of the first item.

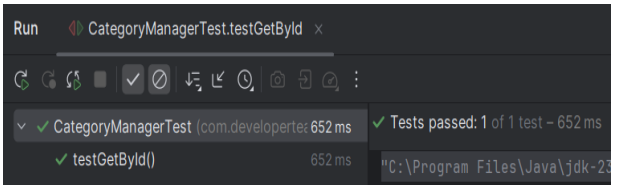


This method tests retrieving the authenticated user's own cart, ensuring the cart items match the expected details such as ID, quantity, and product ID.

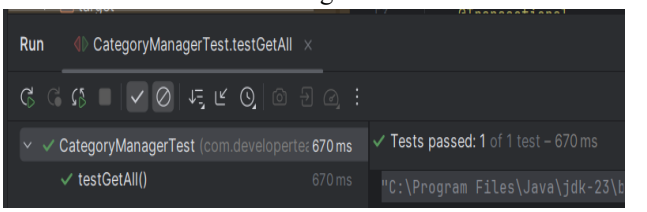


Category Tests

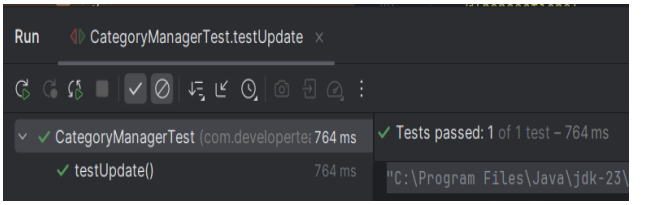
This method tests fetching a category by its ID, validating that the category's name and ID match the expected values.



This method tests retrieving all categories, ensuring the list is not null and verifying the ID and name of the first two categories.



This method tests updating an existing category by verifying the updated name in both the result and the database.

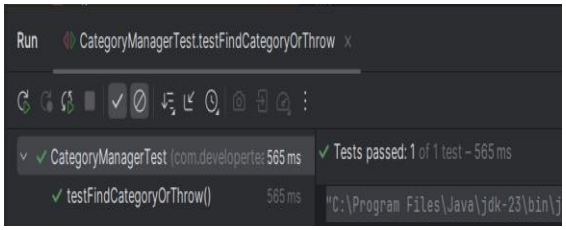


Database after testUpdate() runs:

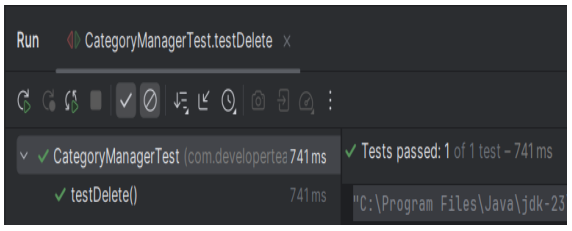
Data Output | Messages | Notifications

	id [PK] integer	name character varying (255)
1	1	Bilgisayar
2	2	Telefon
3	4	Tablet
4	6	Giyilebilir Teknoloji

This method tests finding a category by ID, ensuring the category exists and its attributes match the expected values.

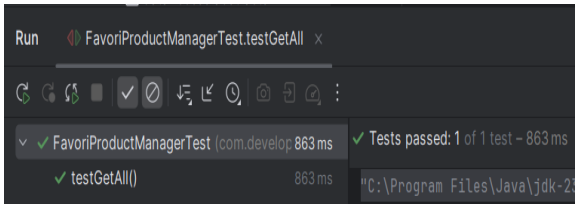


This method tests deleting a category by ensuring the category exists before it is removed from the database.

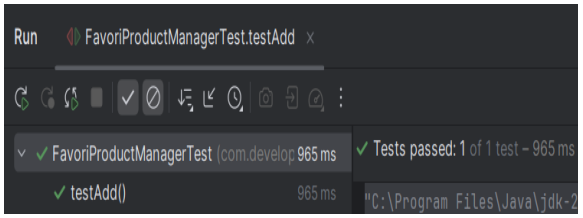


FavoriProductManager Tests

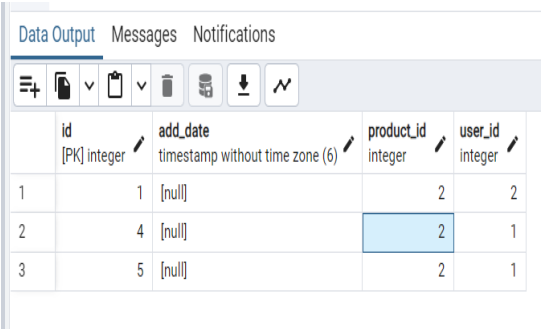
This method tests retrieving all favorite products, verifying that the list is not null and checking the ID of the first product and its associated product ID.



This method tests adding a new favorite product, ensuring the product is saved correctly and matches the expected values in both the result and the database.

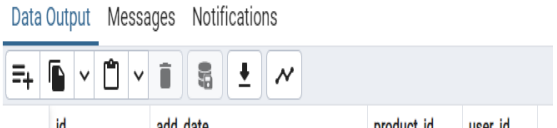


Database before testAdd() runs:



	id [PK] integer	add_date timestamp without time zone (6)	product_id integer	user_id integer
1	1	[null]	2	2
2	4	[null]	2	1
3	5	[null]	2	1

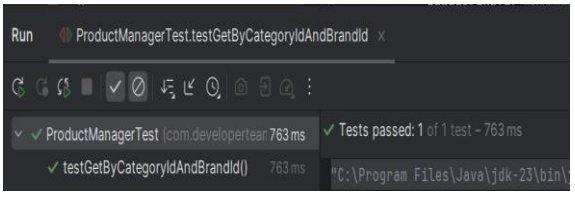
Database after testAdd() runs:



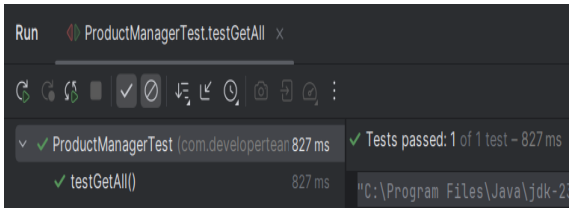
	id [PK] integer	add_date timestamp without time zone (6)	product_id integer	user_id integer
1	1	[null]	2	2
2	4	[null]	2	1
3	5	[null]	2	1
4	6	[null]	2	15

ProductManager Tests

This method tests retrieving products by category and brand ID, ensuring that the list is not null and verifying the product's attributes, such as ID, description, price, stock amount, and associated category and brand IDs.

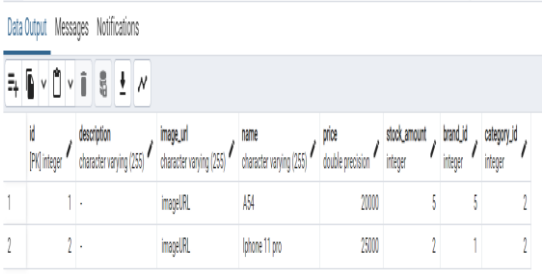


This method tests retrieving all products, ensuring the list is not null, verifying the correct number of products, and checking that the attributes of the first product match the expected values, including ID, description, price, stock amount, and associated brand and category IDs.



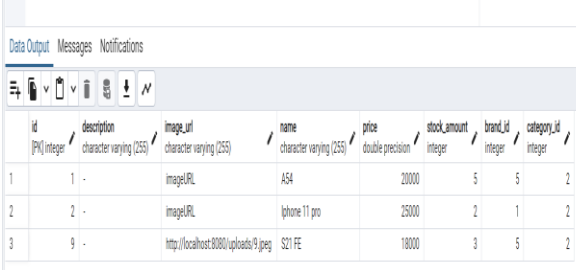
This method tests adding a new product, verifying that the product is created correctly with the expected attributes, the image is uploaded successfully, and the product is saved in the database with the correct details.

Database before testAdd() runs:



	id [PK] integer	description character varying (255)	image_url character varying (255)	name character varying (255)	price double precision	stock_amount integer	brand_id integer	category_id integer
1	1	-	imageURL	A54	20000	5	5	2
2	2	-	imageURL	Iphone 11 pro	25000	2	1	2

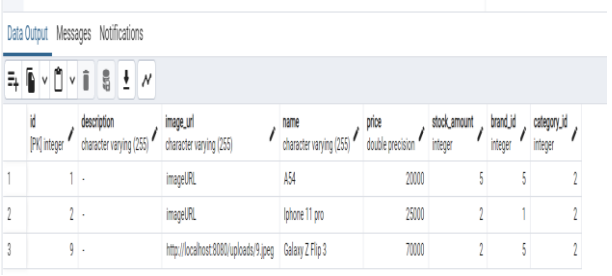
Database after testAdd() runs:



	id [PK] integer	description character varying (255)	image_url character varying (255)	name character varying (255)	price double precision	stock_amount integer	brand_id integer	category_id integer
1	1	-	imageURL	A54	20000	5	5	2
2	2	-	imageURL	Iphone 11 pro	25000	2	1	2
3	9	-	http://localhost:8080/uploads/9.jpeg	S21 FE	18000	3	5	2

This test verifies the functionality of updating a product's details, including its name, description, price, stock amount, category, and brand. It also checks that the image file is uploaded correctly and that the updated product is saved in the database with the correct attributes. The test ensures that the product's information is updated and stored as expected, including checking the presence of the uploaded image file in the specified directory.

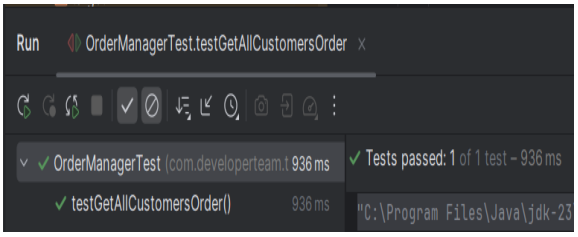
Database after testUpdate() runs:



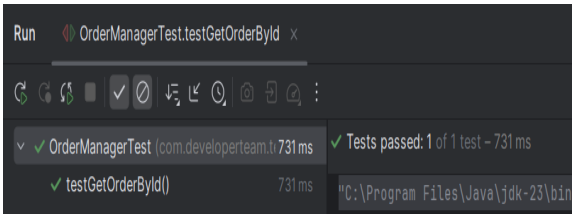
	id [PK] integer	description character varying (255)	image_url character varying (255)	name character varying (255)	price double precision	stock_amount integer	brand_id integer	category_id integer
1	1	-	imageURL	A54	20000	5	5	2
2	2	-	imageURL	Iphone 11 pro	25000	2	1	2
3	9	-	http://localhost:8080/uploads/9.jpeg	Galaxy Z Flip 3	70000	2	5	2

Order Tests

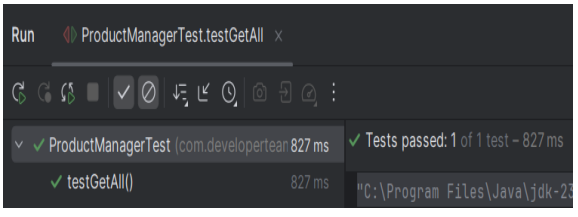
This test verifies the functionality of retrieving all orders. It checks that the list of orders returned by orderManager.getAllOrders() is not null and not empty. Additionally, it validates that the first order in the list matches the expected attributes (ID, total price, and status) from the database. The test also ensures that the order contains the correct items, specifically verifying that the quantity of the first item in the order is 3. This ensures that the order retrieval and its associated items are correctly handled.



This test verifies that the order retrieved by orderManager.getOrderById() matches the expected order details from the database.

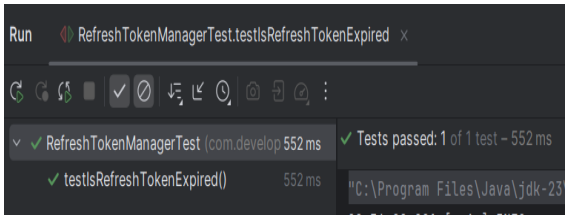


This test ensures that all orders are fetched correctly, validating the order's ID, total price, status, and the quantity of items in the order.



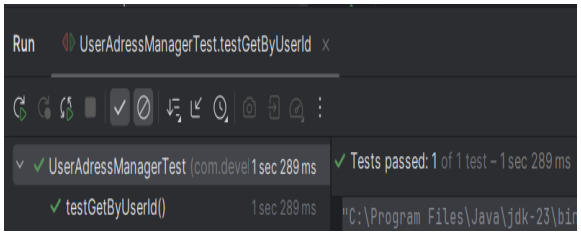
RefreshTokenManager Tests

This test checks if a refresh token is still valid by comparing the current date with the token's expiration date.

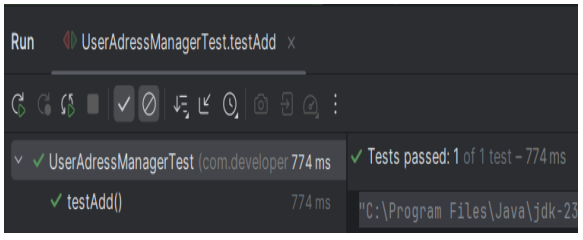


UserAdressManager Tests

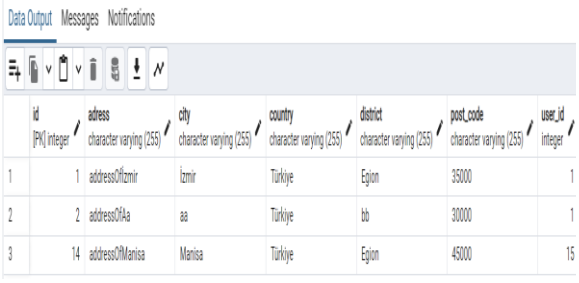
This test verifies that the getByUserId method correctly retrieves user addresses by checking their properties against expected values.



This test ensures that the add method correctly adds a new user address and validates the saved data against the expected values.

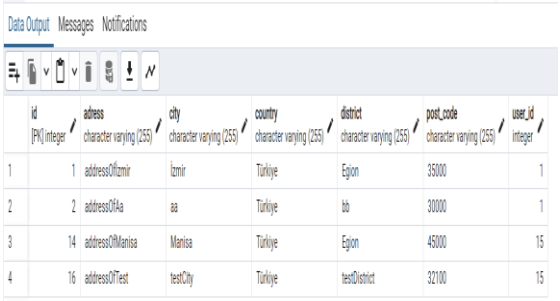


Database before testAdd() runs:



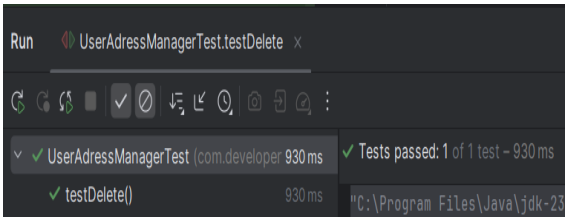
	id [PK] integer	adress character varying (255)	city character varying (255)	country character varying (255)	district character varying (255)	post_code character varying (255)	user_id integer
1	1	addressOfIzmir	İzmir	Türkiye	Egion	35000	1
2	2	addressOfAa	aa	Türkiye	bb	30000	1
3	14	addressOfManisa	Manisa	Türkiye	Egion	45000	15

Database after testAdd() runs:

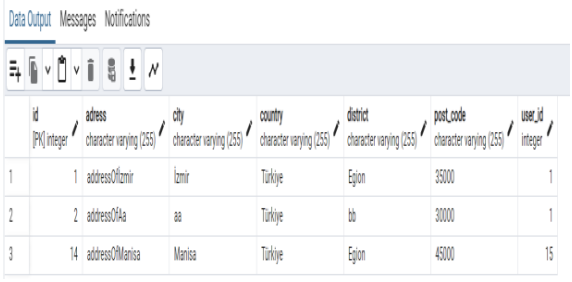


	id [PK] integer	adress character varying (255)	city character varying (255)	country character varying (255)	district character varying (255)	post_code character varying (255)	user_id integer
1	1	addressOfIzmir	İzmir	Türkiye	Egion	35000	1
2	2	addressOfAa	aa	Türkiye	bb	30000	1
3	14	addressOfManisa	Manisa	Türkiye	Egion	45000	15
4	16	addressOfTest	testCity	Türkiye	testDistrict	32100	15

This test verifies that the delete method removes a user address by its ID after confirming its existence.



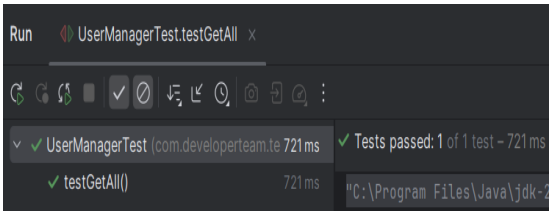
Database after testDelete() runs:



	id [PK] integer	adress character varying (255)	city character varying (255)	country character varying (255)	district character varying (255)	post_code character varying (255)	user_id integer
1	1	addressOfIzmir	İzmir	Türkiye	Egion	35000	1
2	2	addressOfAa	aa	Türkiye	bb	30000	1
3	14	addressOfManisa	Manisa	Türkiye	Egion	45000	15

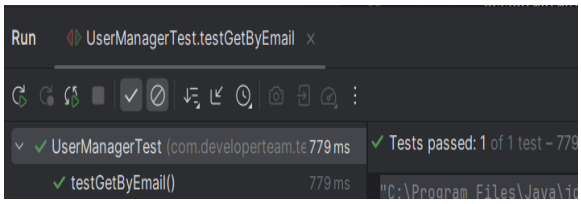
User Tests

Ensures all users are retrieved, with correct details like first name, last name, age, and email.



Verifies a user can be fetched by their ID with accurate details.

Confirms a user can be fetched using their email, validating fields such as name, age, password, phone number, and ID.



Checks the retrieval of a user by their email and password, ensuring correct user details.