

Private University SESAME – Tunis

#### Department of Computer Engineering

**Ecommerce Full stuck web app**

##### Souli mohamed

##### Supervised by: Ferchichi Olfa

#### Academic Year: 2024 – 2025

# Abstract

The Amazona project is a full-featured e-commerce web application developed using the MERN stack (MongoDB, Express.js, React.js, and Node.js). It offers an intuitive shopping experience for users and a comprehensive admin panel for managing products, users, and orders.

This application addresses the limitations of traditional commerce systems by incorporating modern features such as real-time inventory tracking, order placement, user authentication, and payment integration. Designed with scalability and maintainability in mind, Amazona serves as a robust solution for digital retail.

**Contents**

**Abstract** 1  
**General Introduction** 5

### ****1 Project Framework**** 6

**Introduction** 6  
  **1.1 Project Context** 6  
    1.1.1 Project Background 6  
    1.1.2 Problem Statement 6  
    1.1.3 Proposed Solution 7  
  **Conclusion** 7

### ****2 Requirements Analysis and System Design**** 8

**Introduction** 8  
  **2.1 Requirements Analysis** 8  
    2.1.1 Functional Requirements 8  
    2.1.2 Non-Functional Requirements 9  
  **2.2 System Design** 9  
    2.2.1 Module Descriptions 9  
    2.2.2 Use Case Diagrams 10  
    2.2.3 Class Diagram 11  
    2.2.4 Activity Diagram 12  
  **Conclusion** 13

### ****3 Software Environment and Programming Languages**** 14

**Introduction** 14  
  **3.1 Software Environment** 14  
  **3.2 Programming Languages and Frameworks Used** 16  
  **Conclusion** 18

### ****4 Implementation**** 19

**4.1 System Features and UI Demonstration** 19  
    4.1.1 User Authentication 19  
    4.1.2 Customer Dashboard 20  
    4.1.3 Product Browsing and Filtering 20  
    4.1.4 Shopping Cart and Checkout Flow 21  
    4.1.5 Admin Dashboard Interface 22  
    4.1.6 Order History and Tracking System 22  
    4.1.7 Responsive Design and Mobile Support 22

**General Conclusion** 23

**List of Figures:**

2.1 System Use Case Diagram 10

2.2 System Class Diagram 11

2.3 Customer Side – Activity Diagram 12

2.4 Admin Side – Activity Diagram 12

3.1 Visual Studio Code Logo 14

3.2 MongoDB Compass Logo 15

3.3 GitHub Logo 15

3.4 Postman Logo 15

3.5 MongoDB Atlas Logo 16

3.6 React Logo 16

3.7 Node.js & Express Logo 17

3.8 MongoDB Logo 17

4.1 Login Interface 19

4.2 Customer Dashboard 20

4.3 Product Grid and Filter Sidebar 20

4.4 Shopping Cart Page 21

4.5 Checkout and Payment Flow 21

4.6 Admin Dashboard Overview 22

# General Introduction

In modern organizations, efficient leave management is crucial for maintaining operational continuity while ensuring employee well-being. Traditional paper-based or spreadsheet systems are prone to errors, lack transparency, and create administrative burdens. This project addresses these challenges by developing a comprehensive Leave Management System that digitalizes and streamlines the entire process.

The system features:

* + - Role-based access with separate interfaces for managers and employees
    - Leave request submission with two distinct types (sick and vacation)
    - Manager approval workflow through a dedicated interface
    - Visual distinction of rejected, sick leaves and vacation leaves requests (highlighted in red, purple and blue),
    - The colors can identify if a request has been approved or stil in pending review.
    - Leave balance tracking and enforcement
    - Request modification capability for pending leaves This report is structured as follows:
    - Chapter 1 outlines the project framework and context.
    - Chapter 2 details requirements analysis and system design.
    - Chapter 3 describes the software environment and technologies.
    - Chapter 4 presents the implementation and user interface.
    - Finally, we conclude with findings and future improvements.

**Chapter 1**

**Project Framework**

## Introduction

This chapter presents the context of our end-of-year project, assigned by the university as part of our academic program. It outlines the motivation behind the development of the Amazona E-Commerce Platform, identifies the challenges that traditional retail systems typically face in the digital era, and describes the proposed solution that our system offers.

## Project Context

### Project Background

With the rapid rise of digital commerce, traditional physical retail and basic static websites are no longer sufficient to meet the evolving expectations of consumers. These outdated systems often suffer from:

• Limited scalability and availability  
• Lack of real-time inventory updates  
• Absence of secure and user-friendly checkout processes  
• Manual, error-prone order management  
• Poor user experience on mobile devices

In the competitive landscape of online shopping, businesses require dynamic, reliable, and secure platforms to provide efficient services and retain customers.

### Problem Statement

To ensure smooth retail operations and meet the demands of both users and administrators, an e-commerce platform must:

• Offer real-time product and inventory management  
• Support secure user authentication and order history tracking  
• Provide a simple and intuitive interface for shoppers  
• Include a powerful admin dashboard for product, user, and order control  
• Be scalable and accessible across all modern devices

Without these features, businesses struggle with inefficiency, customer dissatisfaction, and loss of revenue.

### Proposed Solution

Our proposed solution is **Amazona**, a modern web-based e-commerce platform developed using the **MERN stack** (MongoDB, Express.js, React.js, Node.js). It aims to meet modern commerce needs through:

• A responsive and intuitive interface for seamless user interaction  
• A secure login and registration system using JWT authentication  
• A fully functional shopping cart and order placement workflow  
• Real-time product listing, filtering, and inventory updates  
• An admin dashboard for managing products, users, and orders  
• Compatibility with desktop and mobile devices for broad accessibility

## Conclusion

This chapter clarified the motivation and rationale behind our project. By identifying the limitations of outdated commerce systems, we proposed a scalable and modern e-commerce platform that uses web technologies to enhance both the customer and admin experience. The next chapter will explore the specific requirements and design of the Amazona system.

**Chapter 2**

**Requirements Analysis and System Design**

## Introduction

This chapter outlines the functional and non-functional requirements of the **Amazona E-Commerce Platform**. It also presents the system’s structural design, including key modules and their responsibilities. These elements form the foundation of the system’s implementation, ensuring it meets performance expectations, user needs, and industry standards.

## Requirements Analysis

### Functional Requirements

**User Authentication**  
– Login and registration system for customers and administrators  
– JWT-secured sessions and password encryption

• **Customer Features**  
– Browse and filter products by category, rating, and price  
– Add products to shopping cart  
– Place orders with delivery details and payment method  
– View order history and track order status  
– Manage user profile and shipping addresses

• **Admin Features**  
– Add, update, or delete products  
– Manage user accounts (view, promote, or remove users)  
– View and update order statuses (e.g., paid, delivered)  
– Access dashboard with metrics on sales, users, and inventory

• **System Features**  
– Real-time product availability tracking  
– Role-based access control for admin and users  
– Automated order and payment confirmation  
– Secure storage of user credentials and transaction data  
– Shopping cart persists across sessions  
– Full responsiveness on desktop and mobile

### Non-Functional Requirements

**Usability**: Simple and modern interface for all users with minimal learning curve  
• **Performance**: Fast loading time and smooth navigation even under load  
• **Security**: JWT authentication, role-based access, and hashed password storage  
• **Availability**: 24/7 uptime with minimal interruptions

## System Design

The **Amazona** platform is composed of several well-defined modules that interact through RESTful APIs. The system is structured around two key roles: **customers** and **administrators**. Customers use the site to explore products, make purchases, and manage personal details. Administrators have full control over the platform’s content and operations.

The modular architecture promotes scalability, maintainability, and code separation of concerns.

### Module Descriptions

**Authentication Module**:  
Handles user registration, login, and JWT session management. Ensures secure access and differentiates roles (admin/user) for access control.

• **Product Management Module**:  
Allows admins to create, edit, or delete products. Each product contains attributes like title, price, stock count, rating, and category. Users can browse and view product details.

• **Cart and Order Module**:  
Enables users to add/remove products to their cart, choose delivery preferences, and place orders. Orders are stored with unique identifiers and associated with the user.

• **Admin Dashboard Module**:  
Provides analytics and insights for administrators including charts, product summaries, order lists, and user counts. Includes management interfaces for all entities.

• **User Profile and Order Tracking Module**:  
Users can update profile info, view previous orders, and track the status of ongoing deliveries.

• **Payment Integration Module**:  
Handles the checkout process and integrates payment method selection (e.g., PayPal, Stripe, or Cash on Delivery). Ensures secure transactions and confirms payment status.

### Use Case Diagrams

**A diagram of a product

AI-generated content may be incorrect.**

Figure 2.1: System Use Case Diagram

### Class Diagram

**A computer flowchart with many white boxes

AI-generated content may be incorrect.**

Figure 2.2: System Class Diagram

### Activity Diagram

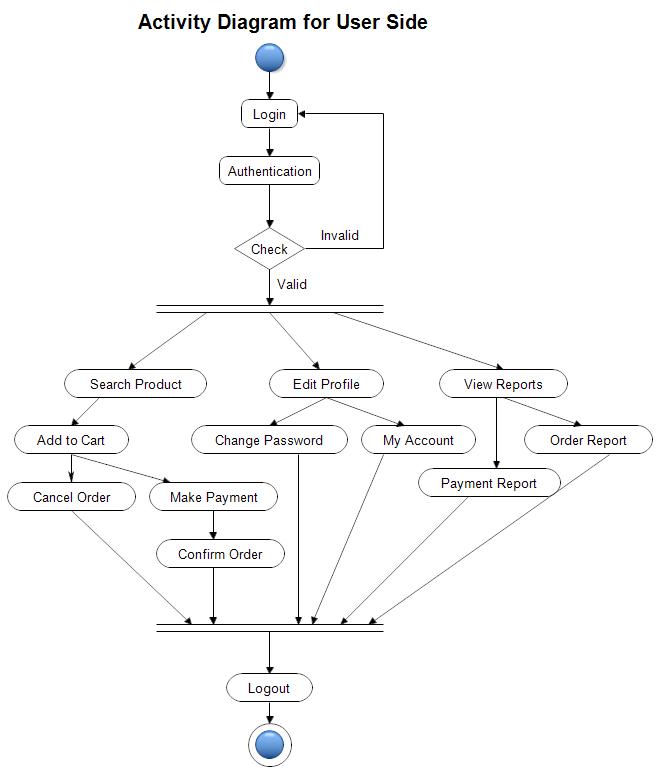
****

Figure 2.3: user side - Activity Diagram

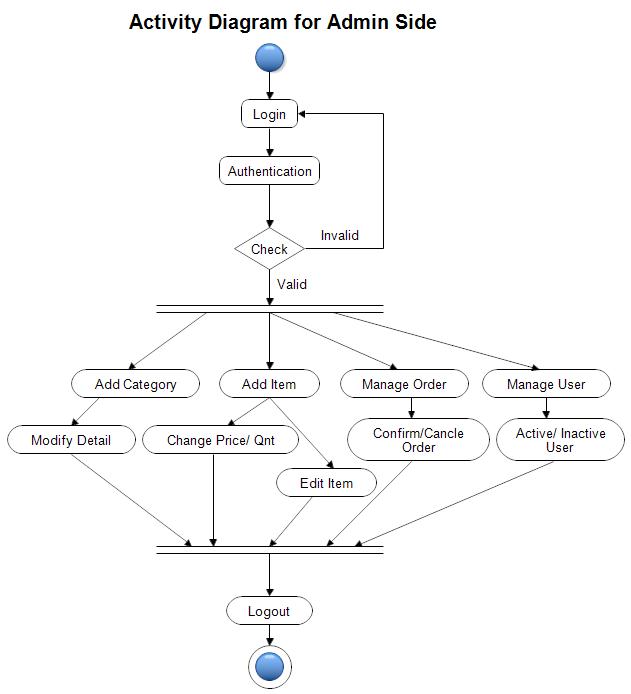


Figure 2.4: admin side - Activity Diagram

## Conclusion

This chapter defined the system’s key requirements and outlined the architecture through its main modules. These specifications serve as the blueprint for implementation and ensure the solution meets the functional, security, and usability needs of its users. The following chapter will focus on the technologies and tools used to build the web app.

**Chapter 3**

**Software Environment and Programming Languages**

## Introduction

This chapter introduces the development tools, software environments, and programming technologies that supported the implementation of the Amazona e-commerce platform.

## Software Environment

**Visual Studio Code**

Visual Studio Code (VS Code) is a lightweight yet powerful source code editor developed by Microsoft. It supports numerous programming languages and includes features like IntelliSense, debugging, version control integration, and extensions. In this project, VS Code was primarily used for writing frontend code in Angular and working on HTML, TypeScript, and SCSS files.

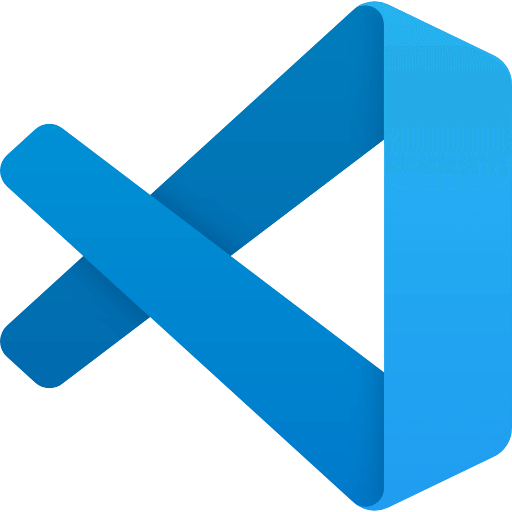


Figure 3.1: Visual Studio Code Logo

**MongoDB Compass:**

MongoDB Compass is the graphical user interface for MongoDB. It was used in this project to visualize, monitor, and manage the non-relational database. Developers could easily explore collections, run queries, and ensure data consistency across users, products, and orders.

A green leaf with brown text

AI-generated content may be incorrect.

Figure 3.2: MongoDB Compass Logo

**Git & GitHub:**

Git was used as the version control system, while GitHub was used for code hosting and collaboration. Git branches were employed to manage feature development and bug fixing, supporting continuous integration and teamwork.

A logo of a cat

AI-generated content may be incorrect.

Figure 3.3: github Logo

**MongoDB Atlas (Optional Deployment):**

MongoDB Atlas is the cloud-hosted version of MongoDB. It was optionally used for deploying the production database, providing scalability and security with minimal configuration.

A green leaf with brown text

AI-generated content may be incorrect.

Figure 3.4: MongoDB Atlas Logo

**SQL Server Management Studio (SSMS)**

SQL Server Management Studio (SSMS) is a Microsoft tool for configuring, managing, and administering SQL Server databases. It was used to create and manage the appli- cation’s database, write and test SQL queries, and interact with the Entity Framework Core backend. SSMS provides a graphical interface for managing tables, views, stored procedures, and database security.

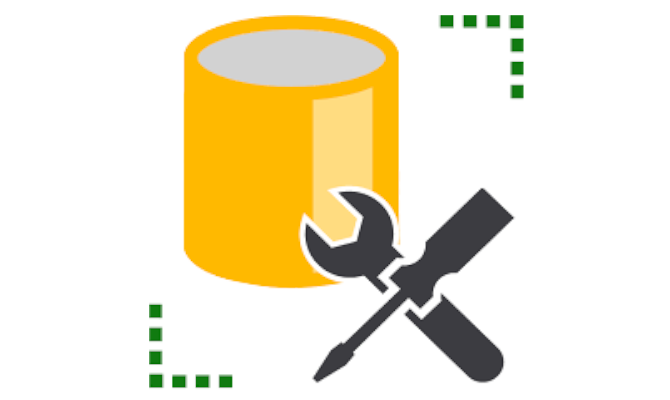


Figure 3.5: SQL Server Management Studio (SSMS) Logo

## Programming Languages and Frameworks Used

**React.js (Frontend)**

React.js is a JavaScript library developed by Facebook for building user interfaces. It was used to build the front end of Amazona, including routing, dynamic product pages, shopping cart, and responsive layout. React Router was used for client-side routing, and Context API/Redux was used for state management.

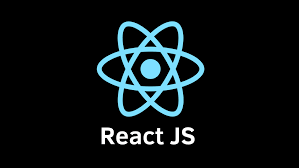


Figure 3.6: react js Logo

**Node.js & Express.js (Backend):**

Node.js is a runtime environment that allows JavaScript to run on the server side. Combined with Express.js, it was used to build the RESTful API for user authentication, product and order management, and administrative features. The API supports JWT-based access control and secure HTTP endpoints.

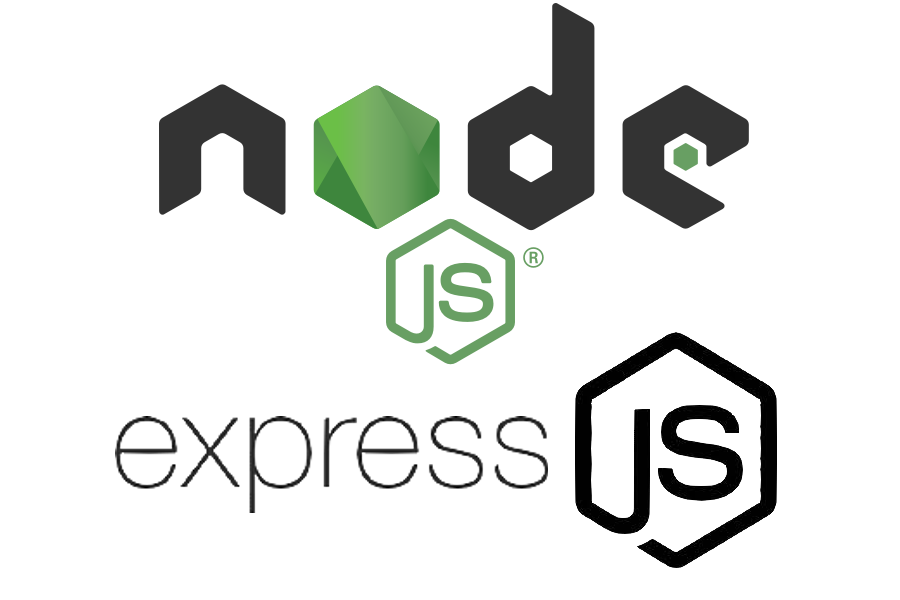


Figure 3.7: node js Logo

**JWT Authentication:**

JSON Web Tokens (JWT) were used for implementing secure authentication and session management. JWT ensures stateless login sessions for both customers and administrators.

***Conclusion:***

This chapter outlined the technical tools, software environments, and programming languages that formed the core of the Amazona project. These technologies enabled the creation of a secure, scalable, and responsive e-commerce application. The next chapter will present the system’s main features and demonstrate its implementation through the user interface.

**Chapter 4 Implementation**

## System Features and UI Demonstration

### User Authentication

the login interface allows users (customers and administrators) to securely access the system using their email and password. Authentication is handled via JWT tokens, ensuring that only authorized users can perform actions such as ordering, reviewing orders, or managing product listings.

A screenshot of a sign in form

AI-generated content may be incorrect.

Figure 4.1: Login Interface

### Home page :

Personalized welcome section with quick access to order history  
• Product search and category filtering  
• Shopping cart icon showing real-time quantity  
• Navigation links to profile, address book, and logout

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.2: home page

### Product Browsing and Filtering

Dynamic product cards with images, names, ratings, and prices  
• Real-time filtering by category, price, and rating  
• Pagination for large product inventories  
• Out-of-stock indicator and add-to-cart buttons

A screenshot of a website

AI-generated content may be incorrect.

Figure 4.3: product filter

A screenshot of a phone

AI-generated content may be incorrect.

Figure 4.4: side bar filter

### Shopping Cart and Checkout Flow:

Add/remove items with live subtotal updates  
• Quantity adjustment directly in the cart  
• Checkout redirection requiring login if not authenticated  
• Selection of shipping address and payment method (e.g., PayPal, Cash)

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.4: shopping cart

A screenshot of a checkout form

AI-generated content may be incorrect.

Figure 4.5: Checkout and Payment Flow

A screenshot of a credit card

AI-generated content may be incorrect.

### Admin Dashboard Interface:

Admin access to analytics: total users, orders, and product count  
• Dedicated sections to manage:  
– Products: add, update, delete  
– Users: list, block/unblock, promote  
– Orders: view order list and update delivery status

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.6: Manager Interface

A screenshot of a graph

AI-generated content may be incorrect.

Figure 4.7: Admin product tracking Overview

* + - * **4.1.6 Order History and Tracking System**

Customers can view all past orders with date, status, and total  
• Clickable orders display shipping info and items purchased  
• Real-time status: pending, paid, shipped, delivered  
• Secure visibility (only visible to logged-in users)

### ****4.1.7 Responsive Design and Mobile Support****

• Fully responsive layout using CSS Flexbox/Grid and media queries  
• Hamburger menu for small screens  
• Adaptive product grid, mobile cart access, and checkout flow  
• Designed for usability across desktops, tablets, and smartphone

**General Conclusion**

The developed **Amazona E-Commerce Platform** effectively addresses the challenges of traditional retail systems by providing a feature-rich, scalable, and responsive digital solution. Key accomplishments include:

• Secure user and admin authentication  
• Full shopping experience: from browsing to checkout  
• Real-time order and inventory management  
• Mobile-friendly and fast-loading UI

**Future improvements may include:**

• Mobile application version (React Native or Flutter)  
• Integration with third-party payment providers like Stripe or Apple Pay  
• Support for product reviews and wishlists  
• Multi-vendor marketplace capability with seller dashboards

This project demonstrates how modern web technologies can transform the e-commerce experience, benefiting both customers and administrators through enhanced usability, automation, and scalability.