BAHIR DAR UNIVERSITY

Bahir Dar Institute of Technology (BIT)

Faculty of computing

Department of Software Engineering

PROJECT TITLE: Tana Market E-Commerce Platform

Project Documentation For industrial project on Tana Market E-Commerce Platform

Submitted to the faculty of computing in partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering

Group Members

	Name	ID Number
1.	Alazar Andualem	1305794
2.	Anwar Ebrahim	1305786
3.	Dawit Genetu	1305901

ADVISOR : Mr. Kelemork

Bahir Dar University, Bahir Dar Institute of Technology

January 2026

Declaration

We declare that this project has been composed solely by ourselves and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where stated otherwise by reference or acknowledgement, the work presented is entirely our own.

Name	Signature
1. Alazar Andualem …………………………	________________
2. Dawit Genetu ……………………………..	_________________
3. Anwar Ebrahim……………………………	_________________

Faculty: Computing

Program: Software Engineering

Project Title Tana Market E-Commerce Platform

This is to certify that I have read this project and that in my supervision and the students' performance, it is fully adequate, in scope and quality, as a project for the degree of Bachelor of Science.

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

Nameof Advisor **Signature**

Approval Sheet

This project entitled **“Tana Market: A Full-Stack E-Commerce Platform with Role-Based Access Control”** has been read and approved as meeting the preliminary project requirements of the Department of Software Engineering in partial fulfillment for the award of a Bachelor of Science degree in Software Engineering, Bahir Dar University, Ethiopia.

Approved by:

Name of Advisor: Mr. Kelemework

Date: 24/01/2026 GC

Signature: ______________________

Examining committee members	signature	date
1. Examiner1………………………	_________	
2. Examiner2………………………	_________	

It is approved that this project has been written in compliance with the formatting rules laid down by the faculty.

Roles and Responsibilities of the Group Members

List of tasks	List of memebers		
	Alazar Andualem	Anwar Ebrahim	Dawit Genetu
Backend development	✓		✓
Frontend development		✓	
UI/UX design		✓	✓
Backend API design	✓		✓

Acknowledgment

First and foremost, we would like to express our sincere gratitude to Almighty God for His guidance, strength, and continuous support throughout our academic journey. Without His divine assistance, the successful completion of this project would not have been possible.

We would also like to extend our deepest appreciation to our advisor, Mr. Kelemework, for their invaluable guidance, unwavering support, and encouragement throughout the duration of this project. Their expertise, constructive feedback, and dedicated mentorship have been crucial to the successful completion of this work.

Special thanks to the Bahir Dar Institute of Technology, Faculty of Computing, for providing the necessary resources and environment for this research. We also extend our appreciation to the businesses and customers who participated in the requirement gathering process and provided valuable feedback during system testing.

The knowledge and insights shared with us have significantly enhanced both the quality and impact of our project. We are truly grateful for their mentorship, and we will carry the lessons learned under their guidance with us throughout our academic and professional careers.

List of Figures

List of tables

Abbreviations and Acronyms

Acronym	Full Form
API	Application Programming Interface
JWT	JSON Web Token
REST	Representational State Transfer
SPA	Single Page Application
UI	User Interface
UX	User Experience
PDF	Portable Document Format
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MVC	Model-View-Controller
ODM	Object Document Mapper
CRUD	Create, Read, Update, Delete
RBAC	Role-Based Access Control
SDLC	Software Development Life Cycle
UML	Unified Modeling Language
RAD	Requirement Analysis Document
RDD	Requirement Design Document

Table of Contents

CHAPTER ONE

INTRODUCTION

In today's digital era, e-commerce has revolutionized the way consumers purchase goods and services globally. In Ethiopia, the digital marketplace is experiencing significant growth, driven by increasing internet penetration, mobile device adoption, and the need for convenient shopping solutions. Traditional brick-and-mortar stores face challenges in reaching wider customer bases, managing inventory efficiently, and providing seamless shopping experiences.

The rapid advancement of web technologies and payment gateways has created opportunities to develop comprehensive e-commerce platforms that assist businesses in establishing online stores while providing customers with secure and convenient shopping experiences. Modern web applications provide accessibility from any device, while responsive frameworks enable optimal user experiences across desktop, tablet, and mobile devices.

This project focuses on the development of Tana Market, a comprehensive full-stack e-commerce platform designed specifically for the Ethiopian market. The platform enables businesses to create and manage online stores, allows customers to browse and purchase products with secure payments, and provides administrators with complete oversight capabilities. The system integrates Chapa payment gateway, automated order tracking with unique TANA tracking numbers, and comprehensive analytics dashboards.

By integrating modern web technologies with automated workflows, the proposed system aims to enhance e-commerce operations, improve user experience, and support data-driven decision making. The application is intended to serve businesses of all sizes, from small entrepreneurs to larger enterprises, who require a robust, secure, and user-friendly e-commerce solution.

1.1 Background of the Project

The development of this project is undertaken within an academic environment as part of the Bachelor of Science degree in Software Engineering at Bahir Dar University. The project reflects the growing demand for integrated software solutions that address real-world problems using modern web technologies and payment processing systems.

With the increasing digitalization of commerce, web-based e-commerce platforms have become essential tools for businesses to expand their reach and improve operational efficiency. E-commerce systems, in particular, play a critical role in helping businesses manage products, process orders, handle payments, and analyze sales performance. However, many existing solutions are either too complex for small businesses or lack essential features for comprehensive management.

This project is motivated by the need to design an integrated, user-friendly e-commerce platform that leverages modern web technologies to deliver seamless shopping experiences while maintaining security and reliability. The system is developed using modern software engineering principles and tools to ensure scalability, maintainability, and ease of use.

1.2 Statement of the Problem

Despite the availability of various e-commerce tools, many Ethiopian businesses face challenges in effectively establishing and managing online stores. Existing solutions often suffer from fragmentation, requiring multiple disconnected systems for different aspects of e-commerce operations.

The existing e-commerce landscape in Ethiopia faces several critical challenges:

1. **Limited Role-Based Management**: Many existing platforms lack sophisticated role-based access control, making it difficult for businesses to delegate responsibilities effectively between administrators and operational staff.
2. **Inadequate Order Tracking**: Customers often lack visibility into their order status, leading to confusion and reduced trust in online shopping platforms.
3. **Payment Integration Challenges**: Limited integration with local payment gateways (such as Chapa) restricts the ability of businesses to accept payments from Ethiopian customers.
4. **Review and Feedback Systems**: Inadequate mechanisms for customers to provide product reviews and for businesses to manage customer feedback.
5. **Inventory Management**: Lack of real-time stock management and product availability tracking.
6. **Reporting and Analytics**: Insufficient tools for businesses to analyze sales, track performance, and make data-driven decisions.
7. **Security Concerns**: Weak authentication and authorization mechanisms in existing systems pose security risks.

There is a clear need for an integrated, user-friendly, and feature-rich e-commerce platform that assists all stakeholders in managing online stores efficiently while providing automated workflows and actionable insights.

1.3 Objective of the Project

1.3.1 General Objective

The general objective of this project is to design and implement a comprehensive, secure, and scalable full-stack e-commerce platform that enables businesses to manage online stores efficiently while providing customers with a seamless shopping experience, integrated payment processing, and real-time order tracking.

1.3.2 Specific Objectives

- ➢ To design a responsive, modern user interface that provides optimal user experience across desktop, tablet, and mobile devices.
- ➢ To develop a comprehensive product management system that allows administrators and managers to create, update, delete, and manage product inventory with support for categories, images, pricing, and stock levels.

- To develop a shopping cart system with persistent storage, automatic price calculations, discount handling, and seamless checkout processes.
- To integrate Chapa payment gateway for secure payment processing, enabling customers to make payments using local Ethiopian payment methods.
- To create a review and comment system that allows verified customers to rate and review products, with moderation capabilities for administrators and managers.
- To implement a unique order tracking system using TANA tracking numbers (format: TANA-YYYYMMDD-XXXX) that provides real-time order status updates throughout the order lifecycle
- To implement a notification system that keeps users informed about order status changes, payment confirmations, and other important events.
- To develop comprehensive reporting and analytics dashboards for administrators to monitor sales, track performance, and generate reports.
- To implement activity logging that records system events for audit purposes and security monitoring.
- To implement a robust role-based access control system that supports three distinct user roles (Admin, Manager, Customer) with appropriate permissions and access levels.

1.4 Methodology

1.4.1 Requirement Gathering Methods

The following requirement gathering methods were employed: - Literature review of existing e-commerce platforms and shopping systems - Stakeholder interviews with potential users including business owners, managers, and customers - Questionnaire surveys to gather quantitative data on user preferences - Observation of existing manual e-commerce processes - Prototype review and feedback collection - Competitive analysis of existing e-commerce platforms

1.4.2 Analysis and Design Methodology

The system analysis and design followed an object-oriented approach with: - Unified Modeling Language (UML) diagrams for system modeling - Iterative/incremental development approach - Component-based architecture design - RESTful API design principles - Database normalization techniques - Security-first design considerations

1.4.3 Implementation Methodology

The implementation followed these approaches: - Technology stack: Node.js with Express.js, MongoDB, React, Tailwind CSS - Version control using Git - Modular code organization (MVC pattern) - Environment-based configuration - Service-oriented architecture - Integration with Chapa payment gateway and notification services

1.5 Feasibility Study

Economic Feasibility

The project is economically feasible due to: - Use of open-source technologies eliminating licensing costs - Affordable cloud hosting options with free tiers for development - Competitive transaction fees from Chapa payment gateway - Scalable architecture allowing cost-effective growth - Reduced long-term maintenance costs through modular design - Revenue generation potential for businesses using the platform

Technical Feasibility

The project is technically feasible based on: - Mature and well-documented technologies with large community support - Adequate developer expertise with the selected technology stack - Availability and good documentation of required external APIs - Sufficient standard web hosting infrastructure - Capability to integrate with existing systems through REST APIs - Ability to handle expected load and performance requirements

Time Feasibility

The project timeline is feasible considering: - Iterative development approach allowing incremental feature delivery - Parallel development of frontend and backend components - Reusable components and libraries accelerating development - Well-defined requirements preventing scope creep - Adequate project team size for the project scope

1.6 Beneficiaries and Significance

Primary Beneficiaries

1. **Business Owners/Administrators**: Complete control over the e-commerce platform, user management capabilities, comprehensive reporting and analytics, system configuration and maintenance tools.
2. **Store Managers**: Product management and inventory control, order processing and fulfillment, customer interaction through reviews and comments, operational dashboard for daily tasks.
3. **Customers**: Seamless shopping experience, secure payment processing, real-time order tracking, product reviews and ratings, account management.

Secondary Beneficiaries

1. **Payment Gateway Providers (Chapa)**: Increased transaction volume through platform integration, expanded user base.
2. **Software Developers**: Reference implementation for e-commerce platforms, learning resource for full-stack development.
3. **Academic Community**: Case study for software engineering projects, research material for e-commerce systems.

Project Significance

1. **Academic Significance**: Demonstrates software engineering principles, full-stack development skills, security implementation, API integration, and database design.

2. **Practical Significance**: Provides businesses with ready-to-use e-commerce platform, contributes to digital economy growth in Ethiopia, facilitates digital payments, creates employment opportunities.
3. **Technical Significance**: Demonstrates modern technology stack including React 18, Node.js, Express.js, MongoDB, state management with Zustand, RESTful API design, responsive design with Tailwind CSS.

1.7 Limitations

The current version of the system has the following limitations:

1. **Payment Gateway Dependency**: The system relies on Chapa payment gateway. Any downtime or issues with Chapa will affect payment processing capabilities.
2. **Internet Connectivity**: The platform requires stable internet connectivity for both customers and administrators. Poor connectivity may affect user experience.
3. **Single Currency**: The system currently supports only Ethiopian Birr (ETB). Multi-currency support is not implemented.
4. **Limited Shipping Integration**: The system does not integrate with external shipping providers. Delivery tracking is manual and based on estimated delivery times.
5. **No Offline Mode**: The platform requires an active internet connection. Offline functionality is not available.
6. **Storage Limitations**: Product image storage is limited by server storage capacity. Large-scale image management may require cloud storage solutions.
7. **Scalability Constraints**: The current implementation may require optimization for handling very large numbers of concurrent users (10,000+ simultaneous users).
8. **Browser Compatibility**: While the system is designed to work on modern browsers, older browser versions may experience compatibility issues.
9. **Mobile App Absence**: Native mobile applications are not available. Users must access the platform through web browsers.
10. **Limited Internationalization**: The system is primarily designed for the Ethiopian market. Multi-language support is limited.

1.8 Scope of the Project

In-Scope Features

1. **User Management**: Registration, authentication, profile management, and role-based access control with three roles (Admin, Manager, Customer)
2. **Product Management**: CRUD operations for products, category management, stock control, image uploads
3. **Shopping Cart**: Add/remove items, quantity management, price calculations, persistent storage
4. **Order Management**: Order creation, payment processing, status tracking, cancellation, return requests
5. **Payment Integration**: Chapa payment gateway integration for Ethiopian payment methods

6. **Order Tracking**: Unique TANA tracking number generation (TANA-YYYYMMDD-XXXX) and real-time status updates
7. **Review System**: Product ratings (1-5 stars), comments, moderation, and replies
8. **Notifications**: Real-time notifications for order updates, payment status, and system events
9. **Reporting**: Sales reports, analytics dashboards, activity logs
10. **Admin Dashboard**: Comprehensive management interface for administrators
11. **Manager Dashboard**: Operational interface for order and product management
12. **Customer Interface**: Shopping interface, order history, profile management

Out-of-Scope Features

1. **Mobile Applications**: Native iOS or Android mobile applications
2. **Multi-vendor Support**: Support for multiple sellers/vendors on a single platform
3. **Advanced Inventory Management**: Warehouse management, barcode scanning, automated reordering
4. **Customer Support Chat**: Real-time chat or ticketing system for customer support
5. **Email Marketing**: Automated email campaigns and marketing tools
6. **Social Media Integration**: Direct integration with social media platforms for sharing
7. **International Shipping**: Shipping calculations and tracking for international orders
8. **Multi-currency Support**: Support for multiple currencies beyond Ethiopian Birr (ETB)
9. **Advanced Analytics**: Machine learning-based recommendations, predictive analytics
10. **Pagenition** : pagenition of products is not functional products are not classified by pages just listed

CHAPTER TWO

Requirement Analysis

2.1 Current System Description

In the current situation, most e-commerce operations in Ethiopia rely on manual methods and fragmented systems. Businesses typically use social media platforms, messaging apps, basic websites, and various disconnected tools for different aspects of e-commerce operations.

2.1.1 Major Functions of the Current System

The major functions of existing e-commerce approaches include: - Manual product listing on social media platforms - Order management through messaging apps (Telegram, WhatsApp) - Payment processing through bank transfers or mobile money - Manual inventory tracking using spreadsheets - Customer communication through phone calls and messages - Basic website with limited functionality - Manual order fulfillment and delivery coordination

2.1.2 Problems of the Existing System

The existing e-commerce approaches suffer from several limitations: - **Inefficiency**: Time-consuming manual processes requiring significant labor - **Error-Prone**: Human errors in order processing, inventory tracking, and payment reconciliation - **Lack of Integration**: Disconnected systems requiring manual data transfer between platforms - **Poor Customer Experience**: Customers must navigate multiple platforms and payment methods - **Limited Scalability**: Manual processes do not scale with increasing order volume - **No Real-time Updates**: Customers lack visibility into order status and delivery updates - **Security Concerns**: Limited security controls for payment processing and customer data - **Lack of Analytics**: Limited insights into sales performance, customer behavior, and trends - **Communication Gaps**: Inconsistent and delayed communication between businesses and customers - **Resource Waste**: Duplicate efforts and redundant data entry across different platforms

2.2 Proposed System Description

Tana Market is a comprehensive web-based e-commerce platform designed to automate and streamline the entire online shopping lifecycle. The system addresses all limitations of existing approaches through integrated, automated workflows.

2.2.1 Overview

The proposed system provides a unified platform where businesses can create and manage online stores, customers can browse and purchase products with ease, and administrators can maintain oversight and control. The system integrates all aspects of e-commerce operations into a seamless workflow from product listing to order delivery.

2.2.2 Functional Requirements

Authentication and Authorization Module

- User registration with email, password, full name, phone number
- JWT-based authentication with three user roles: ADMIN, MANAGER, CUSTOMER
- Role-based access control and user profile management
- Account activation/deactivation by administrators

Product Management Module

- Product creation with title, description, price, category, stock, images
- Product categorization and search functionality
- Inventory tracking with automatic stock reduction on orders
- Product status management (active/inactive)
- Discount and promotional pricing support

Shopping Cart Module

- Add/remove products to/from shopping cart
- Quantity adjustment and price calculation
- Persistent cart storage across sessions
- Automatic price calculation including discounts
- Cart summary with subtotal and total amounts

Order Management Module

- Order creation from shopping cart
- Order status lifecycle: PENDING → PAID → APPROVED → SHIPPED → DELIVERED → CANCELLED
- Unique TANA tracking number generation (TANA-YYYYMMDD-XXXX)
- Order approval workflow for managers
- Order cancellation and return request support

Payment Processing Module

- Chapa payment gateway integration
- Payment initiation and verification
- Payment status tracking and history
- Support for multiple payment methods through Chapa
- Payment confirmation and receipt generation

Review and Comment Module

- Product rating system (1-5 stars)
- Review submission by verified customers
- Review moderation by administrators/managers
- Reply to reviews by business representatives
- Product rating calculation based on approved reviews

Notification Module

- Email notifications for order status changes
- In-app notification system for users
- Payment confirmation notifications
- Review approval/rejection notifications
- System-wide announcements

Analytics and Reporting Module

- Admin dashboard with system-wide statistics
- Manager dashboard with store performance metrics
- Sales reports with date range filtering
- Product performance analytics
- Customer activity reports
- Revenue tracking and financial reports

User Management Module

- Admin view of all users
- User role management and permissions
- User activity tracking and logs
- Account activation/deactivation

2.2.3 Non-Functional Requirements

Performance Requirements

- The system shall handle concurrent user requests efficiently
- Response time for all operations shall be under 2 seconds
- The system shall support up to 1000 concurrent users
- Page load time shall be under 3 seconds

Security Requirements

- The system shall implement secure authentication using JWT tokens
- User passwords shall be hashed using bcrypt with salt rounds
- Role-based access control shall be enforced for all operations
- Payment data shall be processed securely through Chapa gateway
- Input validation shall be implemented for all user inputs
- Protection against SQL injection (NoSQL injection) and XSS attacks

Usability Requirements

- The system shall provide an intuitive user interface
- The interface shall be responsive and work on various screen sizes
- Error messages shall be clear and helpful
- The system shall provide tooltips and guidance for complex operations
- Consistent design language across all pages

Reliability Requirements

- The system shall maintain data consistency through proper database design
- The system shall handle errors gracefully without crashing
- Data backup mechanisms shall be implemented
- The system shall recover gracefully from failures

Maintainability Requirements

- The system shall follow modular architecture
- Code shall be well-documented and follow coding standards
- The system shall support easy updates and enhancements
- Configuration shall be externalized for easy deployment

2.3 Business Rules

1. **Role Assignment Rule**: All new user registrations default to "customer" role. Only existing administrators can create manager or admin accounts.
2. **Product Status Rule**: Products can be soft-deleted but not hard-deleted if they have associated orders.
3. **Order Status Transition Rule**: Order status follows strict flow: pending → paid → approved → shipped → delivered. Status cannot be skipped.
4. **Tracking Number Generation Rule**: TANA tracking numbers are generated automatically when order status changes to "paid". Format: TANA-YYYYMMDD-XXXX.
5. **Order Cancellation Rule**: Orders can only be cancelled by customers if status is "pending" or "paid" (before shipping).
6. **Payment Confirmation Rule**: Order status automatically updates to "paid" only after successful payment verification with Chapa gateway.
7. **Review Submission Rule**: Customers can only review products they have purchased (verified through order history).
8. **Duplicate Review Prevention Rule**: Each customer can submit only one review per product.
9. **Review Moderation Rule**: Reviews require moderation before being visible to other customers.
10. **Stock Management Rule**: Orders cannot be placed if requested quantity exceeds available stock.

CHAPTER THREE

System Design

3.1 System Design Overview

System modeling is the process of creating abstract representations of a system to understand, analyze, and design its structure and behavior. Unified Modeling Language (UML) is employed as the standard modeling language to visually represent the Tana Market E-Commerce Platform.

3.2 Use Case Model

3.2.1 Identifying Actors

The following actors have been identified:

Administrator: Manages system, approves managers, views analytics, manages users

Manager: Manages products and orders, processes payments, moderates reviews

Customer: Browses products, makes purchases, processes payments, views orders.

3.2.2 Identifying Use Cases

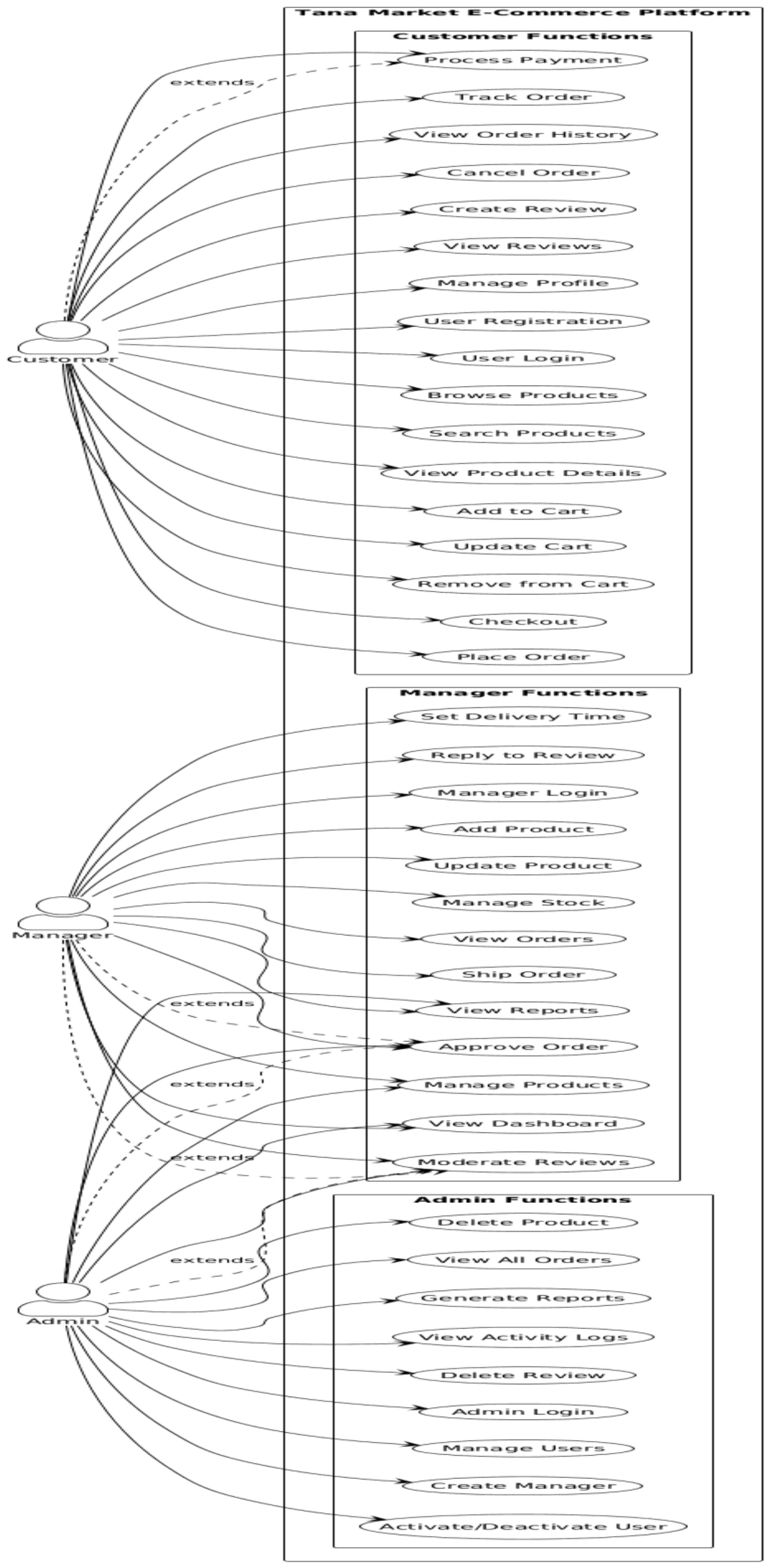
Administrator Use Cases: - Manage Users - Approve/Reject Manager Accounts - View System Analytics - Monitor Payments - Manage System Configuration - View Activity Logs

Manager Use Cases: - Register Account - Create Product - Update Product Details - Manage Inventory - Process Orders - Approve/Reject Orders - Moderate Reviews - View Store Analytics

Customer Use Cases: - Register Account - Browse Products - View Product Details - Add to Cart - View Cart - Checkout - Make Payment - View Orders - Track Orders - Submit Reviews - View Notifications

3.2.3 Use Case Diagram

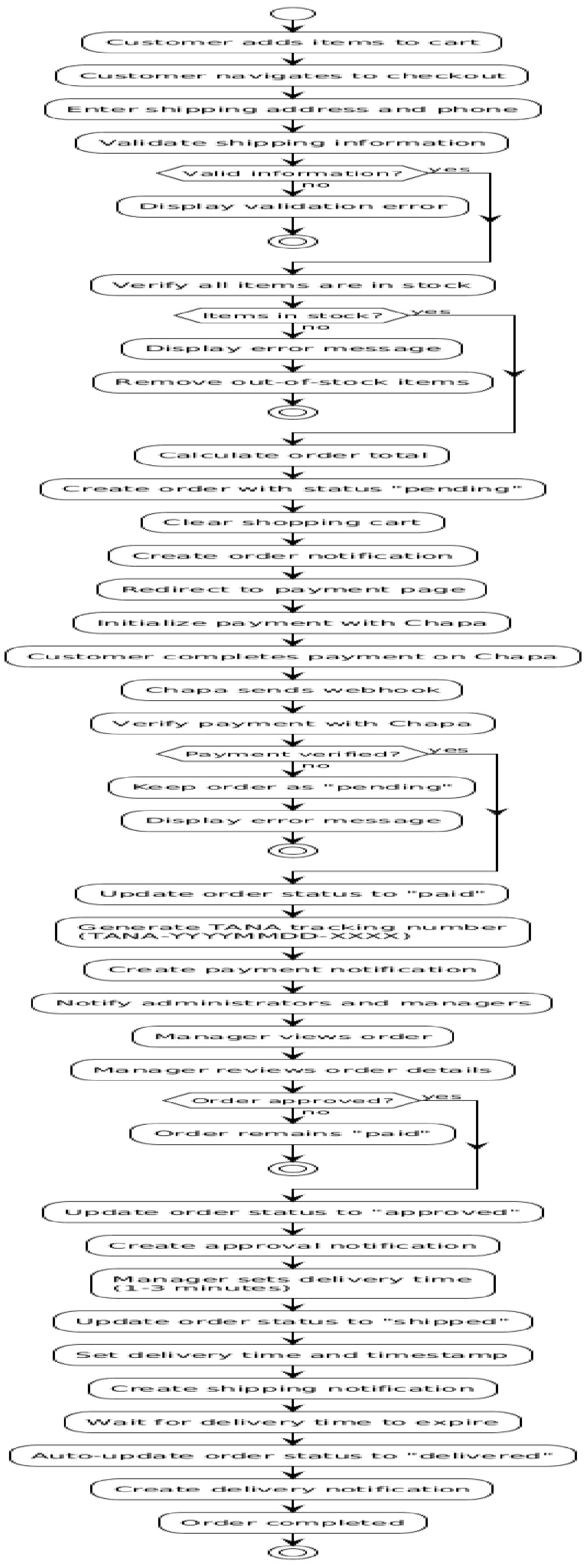
The Use Case Diagram illustrates the interactions between the three primary actors (Administrator, Manager, Customer) and the Tana Market system. It shows: - Customers interacting with shopping, order, and review features - Managers handling product management, order processing, and review moderation - Administrators having access to all features plus user management and reporting



3.3 Activity Diagrams

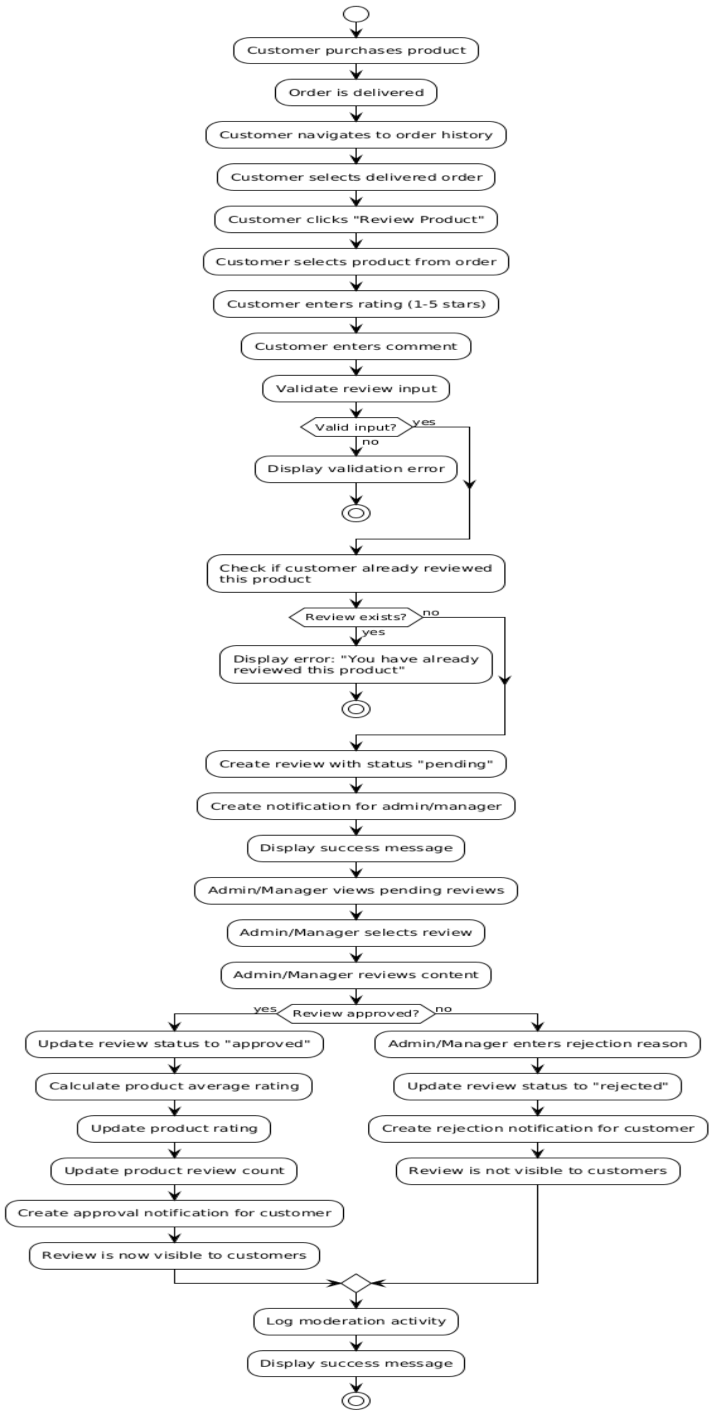
Order Processing Activity Flow:

1. Customer adds items to cart
2. Customer proceeds to checkout
3. System validates cart items and stock availability
4. Customer enters shipping information
5. System creates order with status “pending”
6. Customer initiates payment
7. System redirects to Chapa payment gateway
8. Customer completes payment on Chapa
9. Chapa sends callback to system
10. System verifies payment
11. System updates order status to “paid”
12. System generates TANA tracking number
13. System sends confirmation notifications
14. Manager approves order
15. Manager ships order
16. System marks order as delivered after estimated time
17. Customer receives order and can submit review


Customer adds items to cart
Customer navigates to checkout
Enter shipping address and phone
Validate shipping information
Valid information?
yes
no
Display validation error
Verify all items are in stock
Items in stock?
yes
no
Display error message
Remove out-of-stock items
Calculate order total
Create order with status "pending"
Clear shopping cart
Create order notification
Redirect to payment page
Initialize payment with Chapa
Customer completes payment on Chapa
Chapa sends webhook
Verify payment with Chapa
Payment verified?
yes
no
Keep order as "pending"
Display error message
Update order status to "paid"
Generate TANA tracking number (TANA-YYYYMMDD-XXXX)
Create payment notification
Notify administrators and managers
Manager views order
Manager reviews order details
Order approved?
yes
no
Order remains "paid"
Update order status to "approved"
Create approval notification
Manager sets delivery time (1-3 minutes)
Update order status to "shipped"
Set delivery time and timestamp
Create shipping notification
Wait for delivery time to expire
Auto-update order status to "delivered"
Create delivery notification
Order completed

Product Review Activity Flow:

1. Customer views delivered orders
2. Customer selects order to review
3. System checks if customer already reviewed products
4. Customer submits rating and comment
5. System creates review with status “pending”
6. System notifies managers/admins
7. Manager/Admin reviews submission
8. Manager/Admin approves or rejects review
9. If approved, system updates product rating
10. System notifies customer of review status



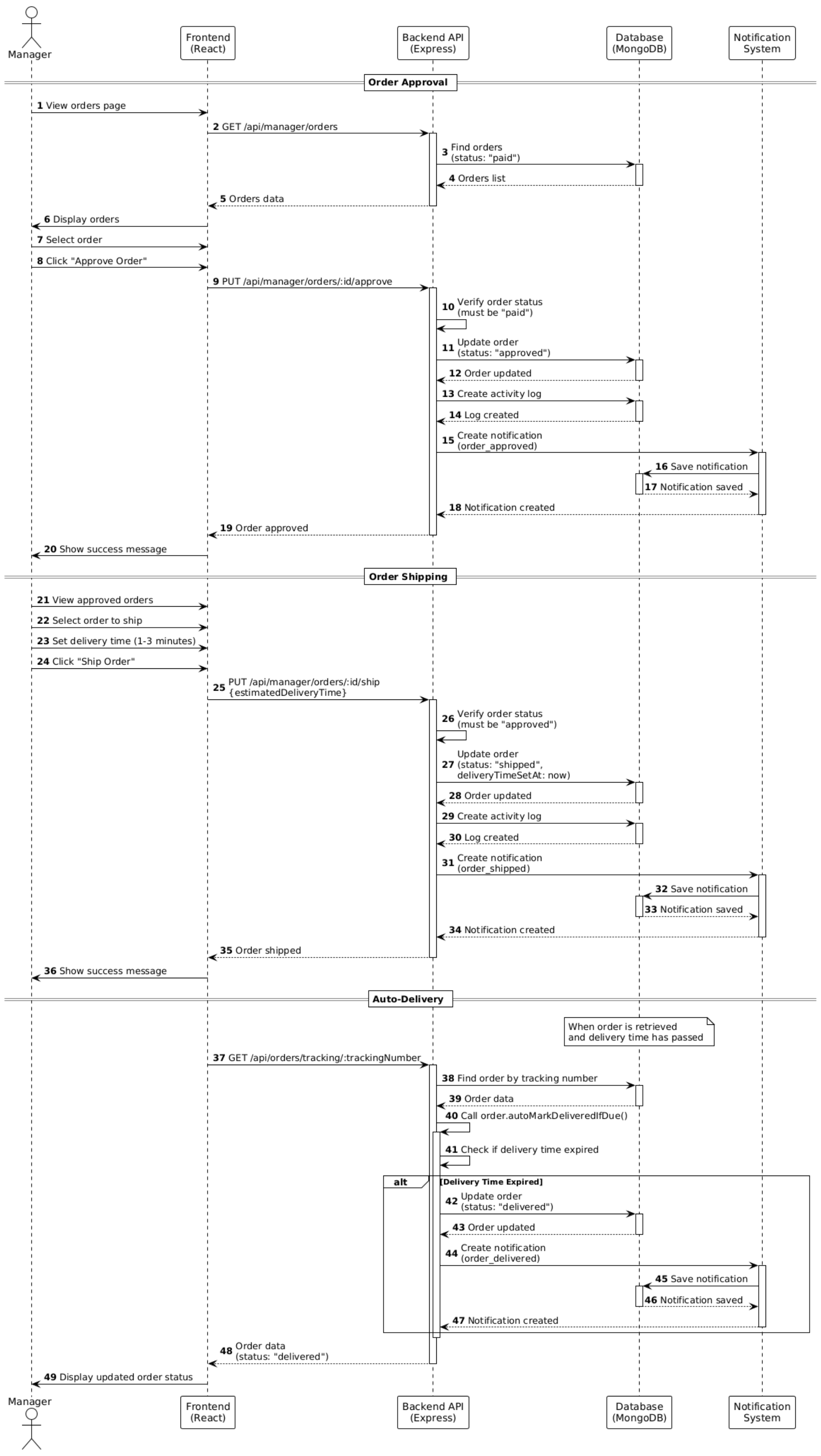
Customer purchases product
Order is delivered
Customer navigates to order history
Customer selects delivered order
Customer clicks "Review Product"
Customer selects product from order
Customer enters rating (1-5 stars)
Customer enters comment
Validate review input
Valid input?
yes
no
Display validation error
Check if customer already reviewed this product
Review exists?
no
yes
Display error: "You have already reviewed this product"
Create review with status "pending"
Create notification for admin/manager
Display success message
Admin/Manager views pending reviews
Admin/Manager selects review
Admin/Manager reviews content
Review approved?
yes
no
Update review status to "approved"
Calculate product average rating
Update product rating
Update product review count
Create approval notification for customer
Review is now visible to customers
Admin/Manager enters rejection reason
Update review status to "rejected"
Create rejection notification for customer
Review is not visible to customers
Log moderation activity
Display success message

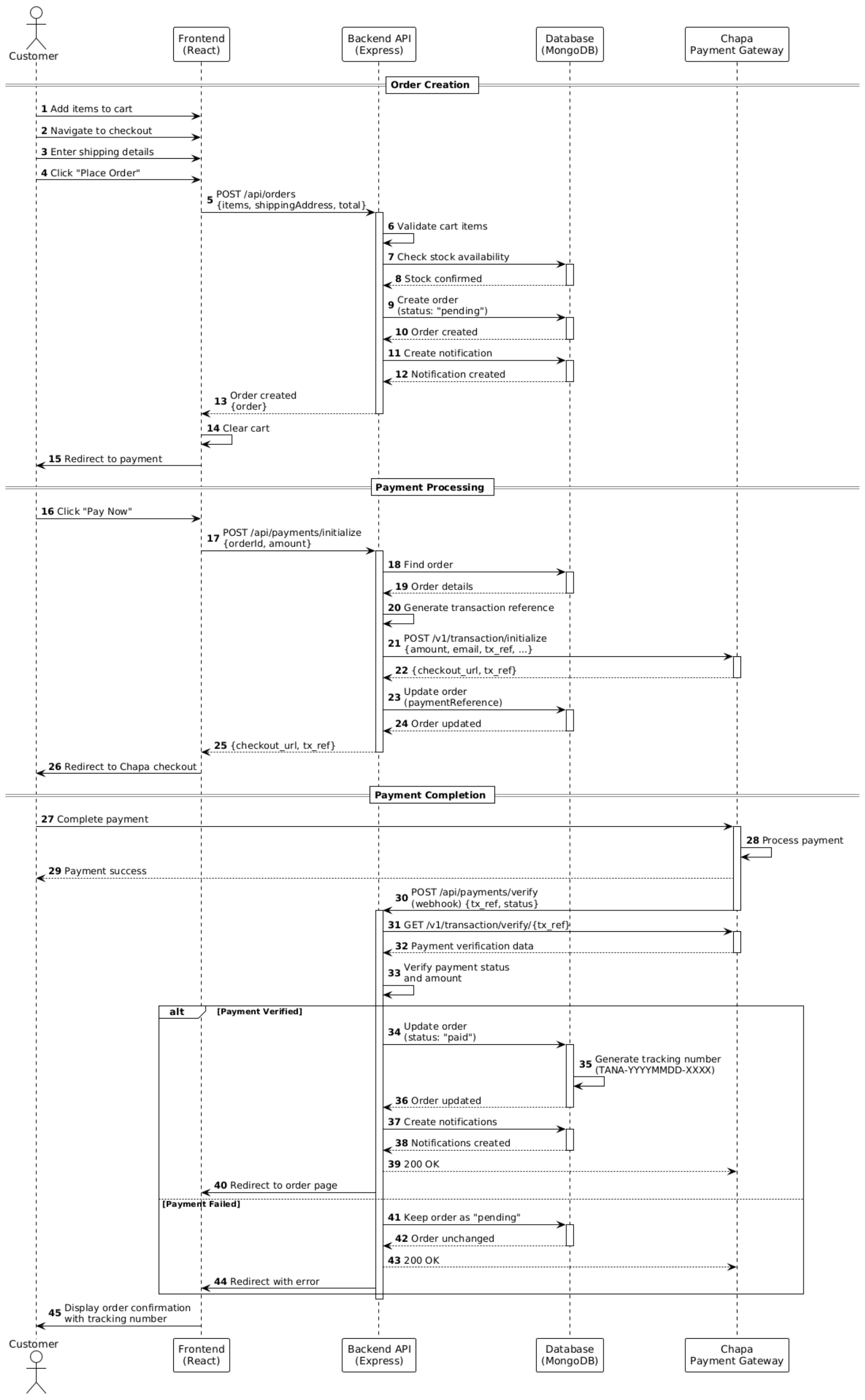
3.4 Sequence Diagrams

Order Placement and Payment Sequence:

1. Customer views product on frontend
2. Frontend requests product details from backend
3. Backend returns product data
4. Customer adds product to cart

5. Frontend updates cart state
6. Customer proceeds to checkout
7. Frontend sends order request to backend
8. Backend validates order and creates order record
9. Backend initiates payment with Chapa
10. Backend returns payment URL to frontend
11. Frontend redirects to Chapa checkout
12. Customer completes payment on Chapa
13. Chapa sends callback to backend
14. Backend verifies payment
15. Backend confirms order and generates tracking number
16. Backend sends email notifications
17. Backend returns success response to frontend


Manager
Frontend (React)
Backend API (Express)
Database (MongoDB)
Notification System
Order Approval
1 View orders page
2 GET /api/manager/orders
3 Find orders (status: "paid")
4 Orders list
5 Orders data
6 Display orders
7 Select order
8 Click "Approve Order"
9 PUT /api/manager/orders/:id/approve
10 Verify order status (must be "paid")
11 Update order (status: "approved")
12 Order updated
13 Create activity log
14 Log created
15 Create notification (order_approved)
16 Save notification
17 Notification saved
18 Notification created
19 Order approved
20 Show success message
Order Shipping
21 View approved orders
22 Select order to ship
23 Set delivery time (1-3 minutes)
24 Click "Ship Order"
25 PUT /api/manager/orders/:id/ship {estimatedDeliveryTime}
26 Verify order status (must be "approved")
27 Update order (status: "shipped", deliveryTimeSetAt: now)
28 Order updated
29 Create activity log
30 Log created
31 Create notification (order_shipped)
32 Save notification
33 Notification saved
34 Notification created
35 Order shipped
36 Show success message
Auto-Delivery
When order is retrieved and delivery time has passed
37 GET /api/orders/tracking/:trackingNumber
38 Find order by tracking number
39 Order data
40 Call order.autoMarkDeliveredIfDue()
41 Check if delivery time expired
alt [Delivery Time Expired]
42 Update order (status: "delivered")
43 Order updated
44 Create notification (order_delivered)
45 Save notification
46 Notification saved
47 Notification created
48 Order data (status: "delivered")
49 Display updated order status
Manager
Frontend (React)
Backend API (Express)
Database (MongoDB)
Notification System

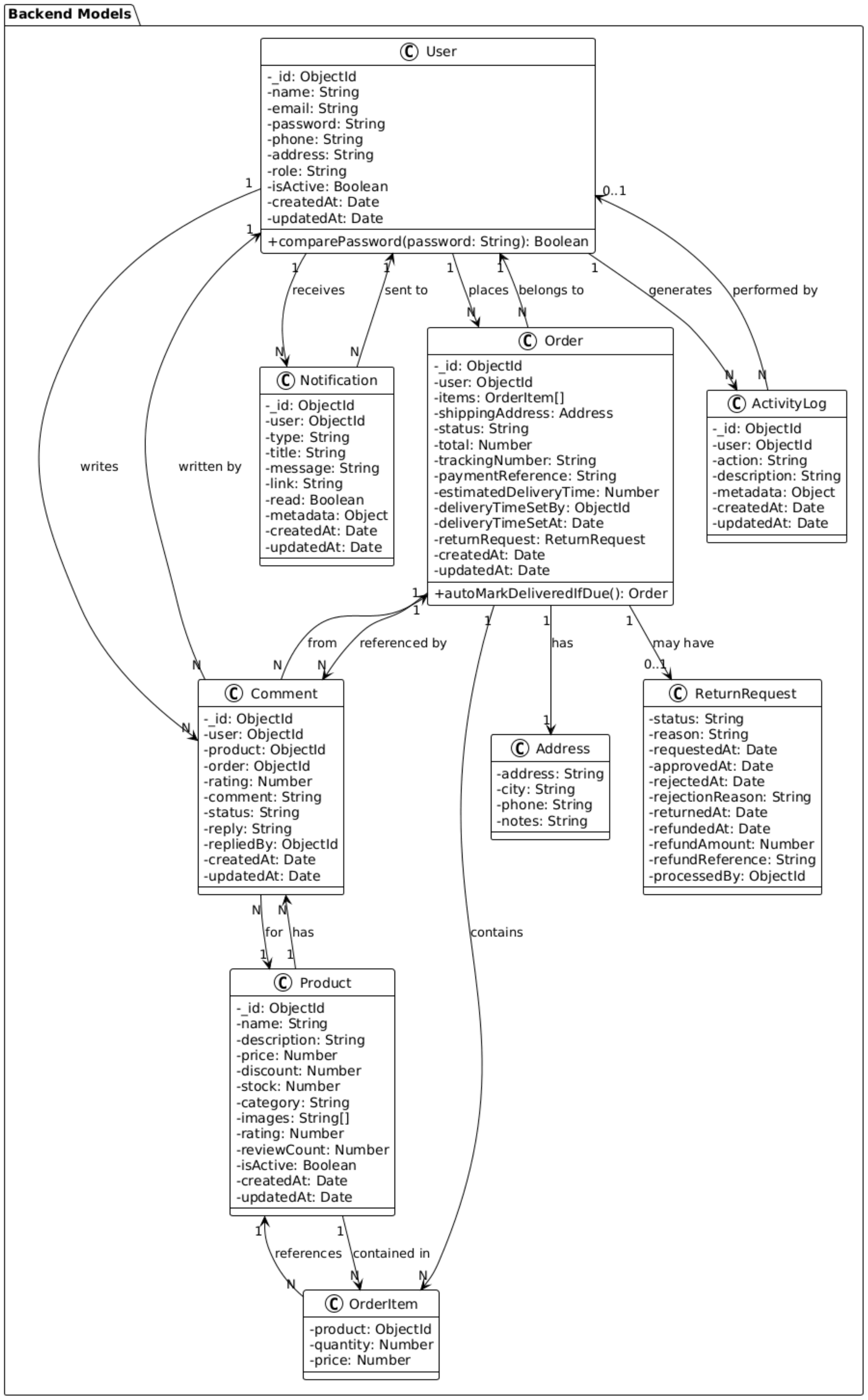

Customer
Frontend (React)
Backend API (Express)
Database (MongoDB)
Chapa Payment Gateway
Order Creation
1 Add items to cart
2 Navigate to checkout
3 Enter shipping details
4 Click "Place Order"
5 POST /api/orders {items, shippingAddress, total}
6 Validate cart items
7 Check stock availability
8 Stock confirmed
9 Create order (status: "pending")
10 Order created
11 Create notification
12 Notification created
13 Order created {order}
14 Clear cart
15 Redirect to payment
Payment Processing
16 Click "Pay Now"
17 POST /api/payments/initialize {orderId, amount}
18 Find order
19 Order details
20 Generate transaction reference
21 POST /v1/transaction/initialize {amount, email, tx_ref, ...}
22 {checkout_url, tx_ref}
23 Update order (paymentReference)
24 Order updated
25 {checkout_url, tx_ref}
26 Redirect to Chapa checkout
Payment Completion
27 Complete payment
28 Process payment
29 Payment success
30 POST /api/payments/verify (webhook) {tx_ref, status}
31 GET /v1/transaction/verify/{tx_ref}
32 Payment verification data
33 Verify payment status and amount
alt
[Payment Verified]
34 Update order (status: "paid")
35 Generate tracking number (TANA-YYYYMMDD-XXXX)
36 Order updated
37 Create notifications
38 Notifications created
39 200 OK
40 Redirect to order page
[Payment Failed]
41 Keep order as "pending"
42 Order unchanged
43 200 OK
44 Redirect with error
45 Display order confirmation with tracking number
Customer
Frontend (React)
Backend API (Express)
Database (MongoDB)
Chapa Payment Gateway

3.5 Class Diagram

The Class Diagram represents the system's data model and relationships:

Core Classes: - **User**: Represents all system users (customers, managers, admins) with attributes: name, email, password, role, isActive - **Product**: Represents products with attributes: name, description, price, discount, stock, category, images, rating - **Order**: Represents customer orders with attributes: user, items, shippingAddress, status, total, trackingNumber, paymentReference - **Comment**: Represents product reviews with attributes: user, product, order, rating, comment, status, reply - **Notification**: Represents user notifications with attributes: user, type, title, message, read - **ActivityLog**: Represents system activity logs with attributes: user, action, description, metadata

Relationships: - User has many Orders (1:N) - User has many Comments (1:N) - User has many Notifications (1:N) - Product has many Comments (1:N) - Order belongs to one User (N:1) - Order contains many OrderItems (1:N) - OrderItem references one Product (N:1) - Comment references one Product (N:1) and one Order (N:1)


Backend Models
User
-_id: ObjectId
-name: String
-email: String
-password: String
-phone: String
-address: String
-role: String
-isActive: Boolean
-createdAt: Date
-updatedAt: Date
+comparePassword(password: String): Boolean
Notification
-_id: ObjectId
-user: ObjectId
-type: String
-title: String
-message: String
-link: String
-read: Boolean
-metadata: Object
-createdAt: Date
-updatedAt: Date
Order
-_id: ObjectId
-user: ObjectId
-items: OrderItem[]
-shippingAddress: Address
-status: String
-total: Number
-trackingNumber: String
-paymentReference: String
-estimatedDeliveryTime: Number
-deliveryTimeSetBy: ObjectId
-deliveryTimeSetAt: Date
-returnRequest: ReturnRequest
-createdAt: Date
-updatedAt: Date
+autoMarkDeliveredIfDue(): Order
ActivityLog
-_id: ObjectId
-user: ObjectId
-action: String
-description: String
-metadata: Object
-createdAt: Date
-updatedAt: Date
Comment
-_id: ObjectId
-user: ObjectId
-product: ObjectId
-order: ObjectId
-rating: Number
-comment: String
-status: String
-reply: String
-repliedBy: ObjectId
-createdAt: Date
-updatedAt: Date
Address
-address: String
-city: String
-phone: String
-notes: String
ReturnRequest
-status: String
-reason: String
-requestedAt: Date
-approvedAt: Date
-rejectedAt: Date
-rejectionReason: String
-returnedAt: Date
-refundedAt: Date
-refundAmount: Number
-refundReference: String
-processedBy: ObjectId
Product
-_id: ObjectId
-name: String
-description: String
-price: Number
-discount: Number
-stock: Number
-category: String
-images: String[]
-rating: Number
-reviewCount: Number
-isActive: Boolean
-createdAt: Date
-updatedAt: Date
OrderItem
-product: ObjectId
-quantity: Number
-price: Number
receives
sent to
places
belongs to
generates
performed by
writes
written by
from
referenced by
has
may have
contains
for
has
references
contained in
1
N
0..1

CHAPTER FOUR

SYSTEM ARCHITECTURE AND IMPLEMENTATION

4.1 Introduction

System design and implementation are fundamental stages in the Software Development Life Cycle (SDLC). This chapter focuses on transforming requirements and models into a concrete and functional solution, describing the architectural design, system components, data storage mechanisms, security considerations, and deployment environment.

4.2 System Architecture Design

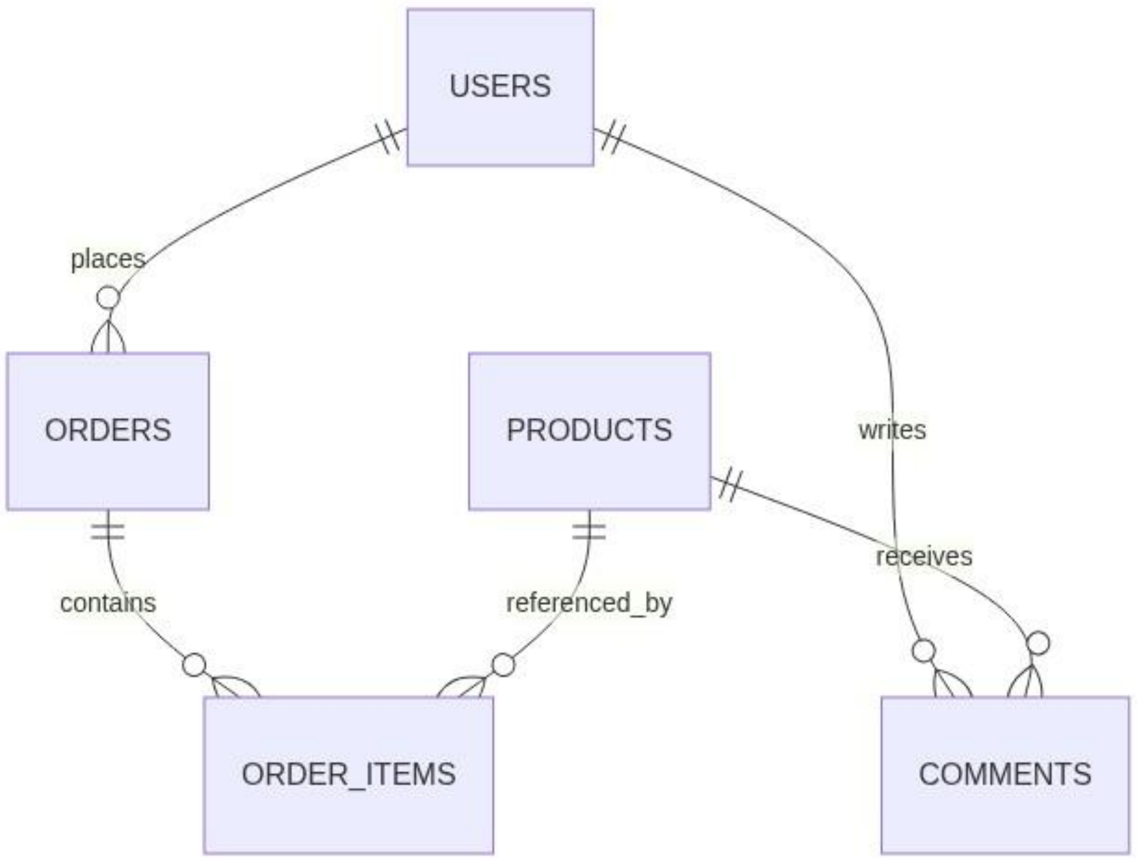
Key Points

Encapsulation:	All attributes are private (-) providing data hiding and security
Public Methods:	Handle business logic and provide controlled access to class data
Multiplicity:	Clearly shows one-to-many and many-to-many relationships
Role Associations:	Defines the role each class plays in relationships (e.g., User places Order)
Data Types:	Strong typing with MongoDB ObjectId for primary keys

3.2.2 Persistent Model (MongoDB – Object-Oriented Persistence)

Since Tana Market uses MongoDB (NoSQL, object-based storage), persistence models map directly to domain classes.

Figure 2: Entity-Relationship Diagram (MongoDB Collections)



Persistent Model – Collection Definitions

Users Collection

Attribute	**Type**	**Length**	**Key**	**Description**
_id	ObjectId	—	PK	Primary key, auto-generated
name	String	100		User's full name
email	String	100	Unique	User's email address (unique)
password	String	255		Hashed password using bcrypt
role	String	20		User role: guest/customer/manager/admin

Products Collection

Attribute	**Type**	**Length**	**Key**	**Description**
_id	ObjectId	—	PK	Primary key
name	String	150	Index	Product name (indexed for search)
description	String	500		Detailed product description
price	Number	—		Product price in ETB
stock	Number	—		Available quantity in stock
category	String	50	Index	Product category (indexed)
isActive	Boolean	—		Product availability status

Orders Collection

Attribute	**Type**	**Length**	**Key**	**Description**
_id	ObjectId	—	PK	Primary key
user	ObjectId	—	FK → users	Reference to user who placed order
total	Number	—		Order total amount
status	String	30		Order status: pending/processing/completed/cancelled
trackingNumber	String	50	Unique	Unique tracking number for order

Persistence Implementation Notes

MongoDB references (ObjectId) maintain relationships between collections
One-to-many relations handled via arrays or foreign key references
Indexing on frequently searched fields (name, category) improves query performance
Embedded documents (OrderItems) for denormalization and faster reads
Data validation enforced at application layer using Mongoose schemas

Connection pooling implemented for database performance optimization

3.3 User Interface Design

Implementation Tool: React.js with Vite build tool

Architecture: Component-Based Architecture with reusable UI components

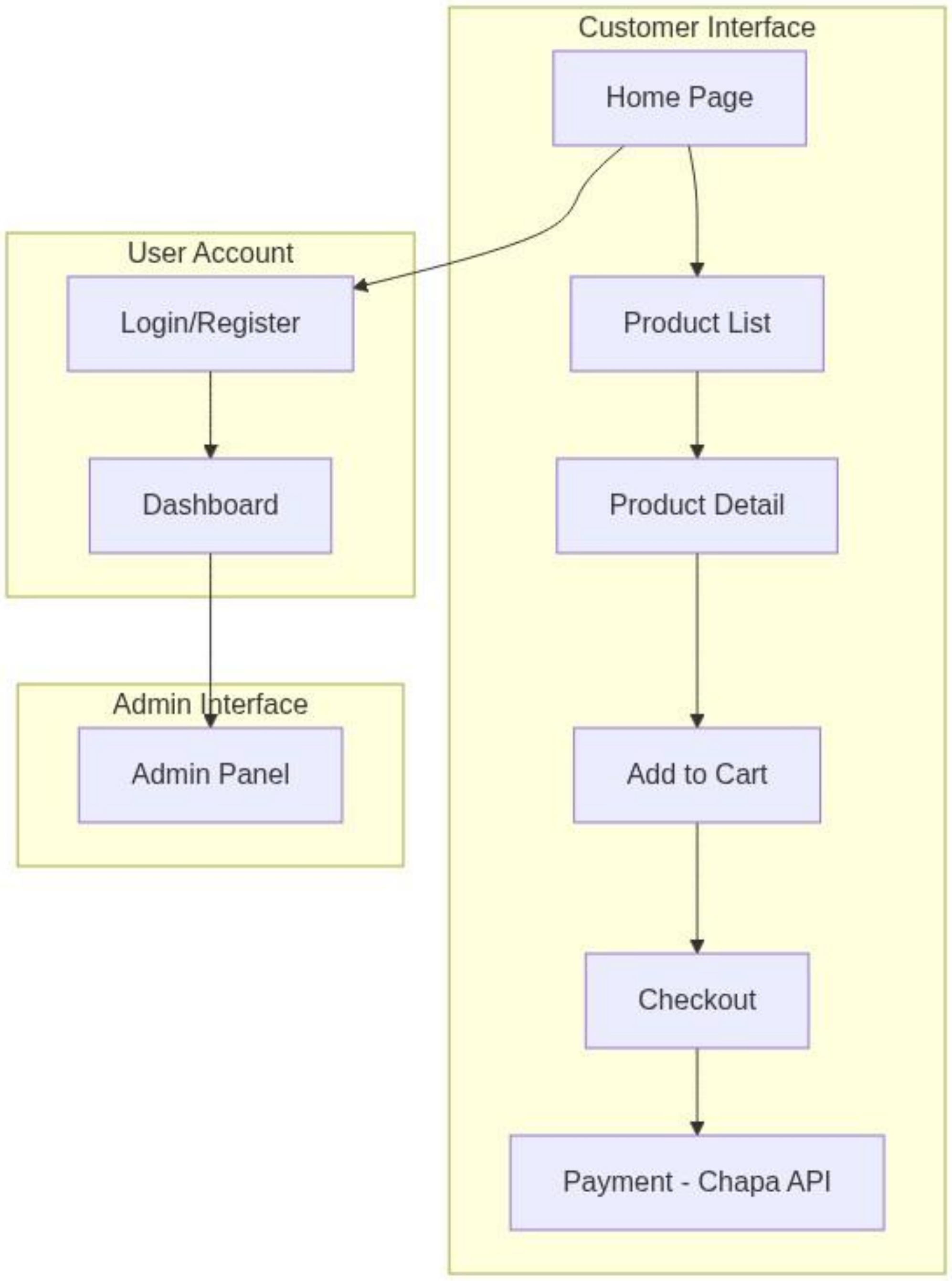
Responsive Design: Mobile-First approach with Bootstrap 5

State Management: React Context API with useReducer

Routing: React Router v6 for client-side navigation

Styling: CSS Modules with Sass preprocessor

Figure 3: User Interface Flow Diagram



Main UI Screens

Customer Interface

Home Page - Featured products, categories, promotions
Product Listing - Search, filter, sort, pagination
Product Detail - Images, description, reviews, add to cart
Shopping Cart - Modify quantities, calculate totals
Checkout Process - Address, payment method, order summary

Order History - Track orders, view past purchases
User Profile - Account management, saved addresses

Admin / Manager Dashboard

Product Management - CRUD operations, bulk import/export
Order Management - Process orders, update status, generate invoices
User Management - View users, modify roles (Admin only)
Inventory Management - Stock levels, low stock alerts
Sales Analytics - Dashboard with charts, reports, filters
Category Management - Product categories and attributes

3.4 Access Control and Security Design

Actors and Roles

Guest: Unauthenticated users who can browse products

Customer: Registered users who can purchase and review products

Manager: Staff with product and order management permissions

Admin: Full system access including user management

Role-Based Access Control (RBAC) Matrix

Feature	**Admin**	**Manager**	**Customer**	**Guest**
Browse Products	✓	✓	✓	✓
Search Products	✓	✓	✓	✓
Add to Cart	✓	✓	✓	X
Place Order	✓	✓	✓	X
View Own Orders	✓	✓	✓	X
Rate Products	✓	✓	✓	X
Manage Products	✓	X	X	X
Manage Orders	✓	✓	X	X
Manage Users	✓	X	X	X

View Analytics	✓	✓	X	X
System Configuration	✓	X	X	X

Security Mechanisms

Authentication: JWT (JSON Web Token) with 24-hour expiry

Authorization: Role-based middleware for route protection

Password Security: Bcrypt hashing with 12 rounds salt

Input Validation: Express Validator middleware, XSS protection

CORS Policy: Restricted to frontend origins only

Secure Headers: Helmet middleware for security headers

Rate Limiting: Express Rate Limit for API endpoints

Payment Security: Chapa API with webhook verification

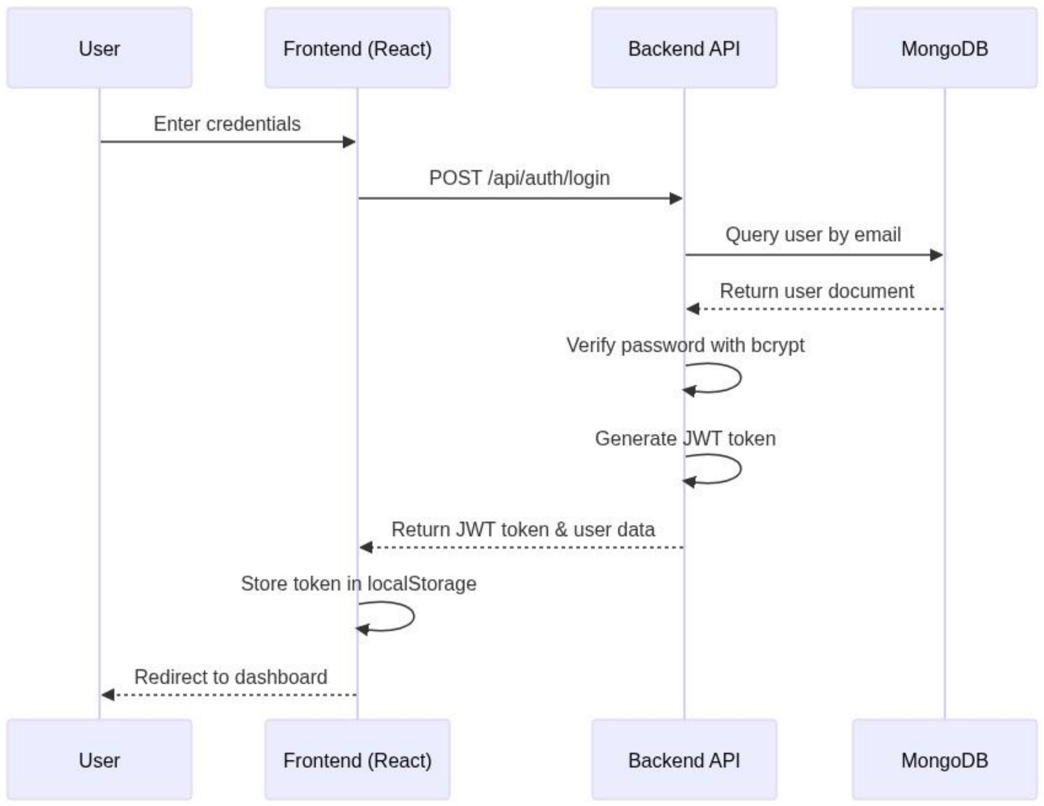
Data Encryption: TLS 1.3 for data in transit

Session Management: HTTP-only cookies, secure flag

3.5 Interaction Sequence Diagrams

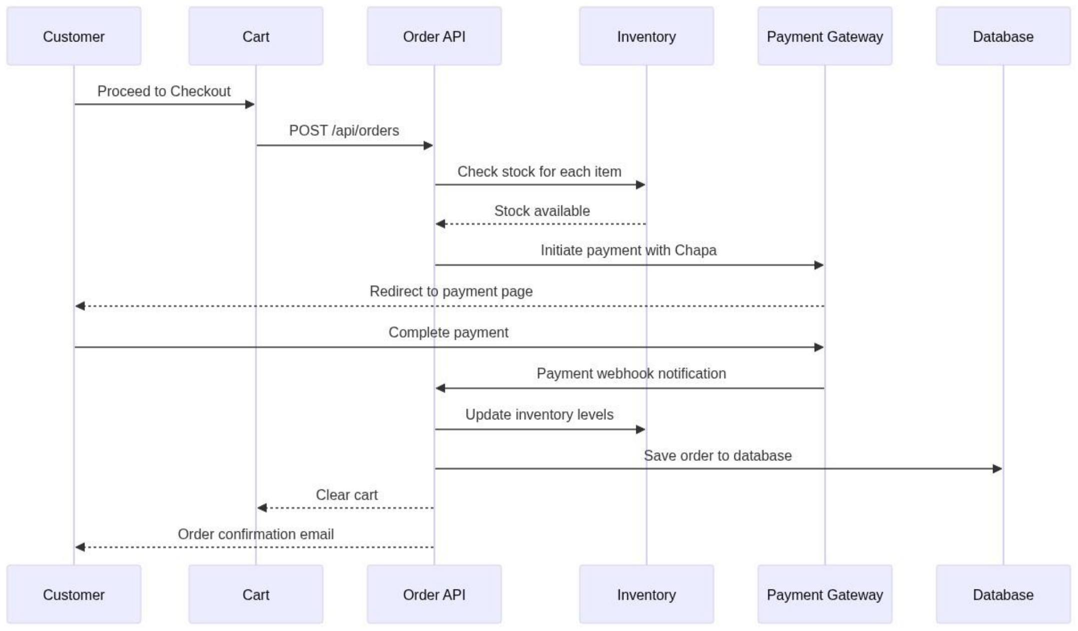
3.5.1 Login Sequence

Figure 4: Login Sequence Diagram



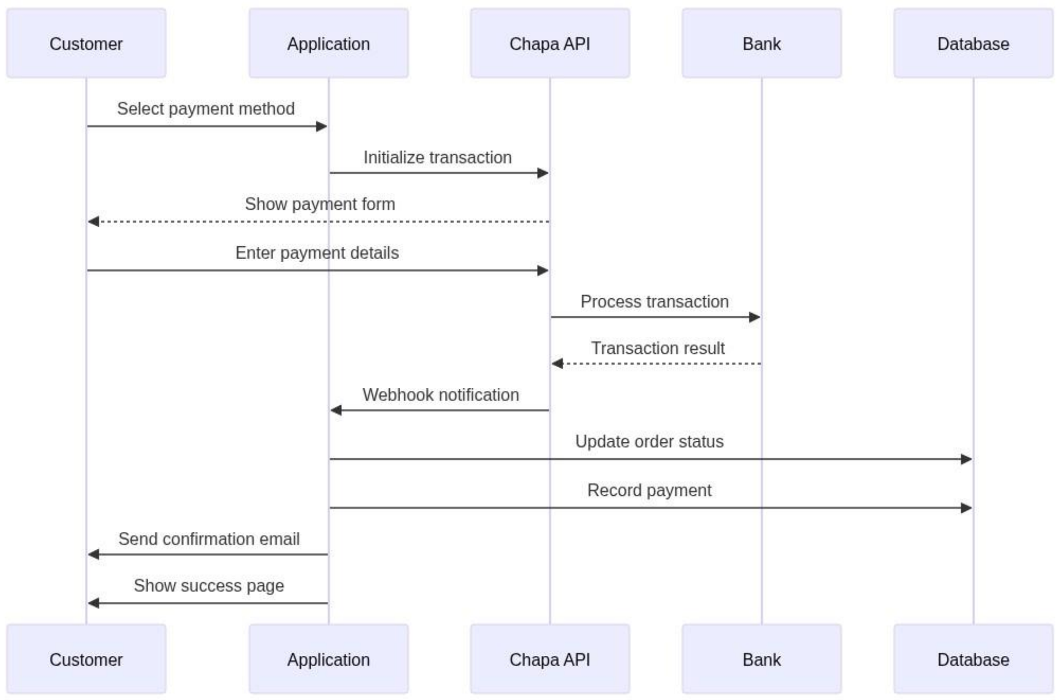
3.5.2 Order Placement Sequence

Figure 5: Order Placement Sequence Diagram



3.5.3 Payment Processing Sequence

Figure 6: Payment Processing Sequence Diagram



The Tana Market E-Commerce Platform is designed using a layered and modular architecture to improve maintainability, scalability, and clarity. The architecture separates responsibilities into different layers:

Presentation Layer

- Represents the web user interface built with React
- Responsible for user interactions, input reception, and output display
- Provides responsive design for various devices using Tailwind CSS

Application Logic Layer

- Handles core functionality using Node.js and Express.js
- Processes user requests and validates input data
- Coordinates communication between layers
- Implements business rules and workflows

Data Access Layer

- Manages interactions with MongoDB database
- Uses Mongoose ODM for data modeling and validation
- Implements data persistence and retrieval operations

Integration Layer

- Handles external service integrations
- Chapa payment gateway integration
- File upload handling using Multer

- Email notification integration

4.3 Proposed System Architecture

The system follows a client-server architecture with the following components:

Frontend Application (React)
- Built using React 18 with Vite build tool
- Uses React Router for client-side routing
- Implements responsive design with Tailwind CSS
- State management using Zustand
- Axios for HTTP requests to backend API

Backend API Server (Node.js/Express.js)
- RESTful API design with proper HTTP methods
- Modular route structure organized by resources
- Middleware for authentication, validation, and error handling
- Environment-based configuration
- Comprehensive logging and monitoring

Database (MongoDB)
- NoSQL database for flexible data modeling
- Collections for Users, Products, Orders, Payments, Comments, Notifications, ActivityLogs
- Indexes for frequently queried fields
- Data validation at application level using Mongoose

External Services Integration
- **Chapa Payment Gateway**: Secure payment processing for Ethiopian customers
- **File Upload Service**: Product image upload and management
- **Notification Service**: Email and in-app notifications

Security Architecture
- JWT token-based authentication
- Password hashing using bcrypt with 10 salt rounds
- Role-based access control (RBAC) with three roles
- Input validation and sanitization
- HTTPS for secure communication
- CORS configuration for frontend-backend communication

4.4 System Implementation

Technology Stack
- **Backend**: Node.js, Express.js, MongoDB, Mongoose
- **Frontend**: React, Vite, Tailwind CSS, Axios

- **Authentication**: JWT, bcrypt
- **Payment Integration**: Chapa API
- **File Upload**: Multer
- **State Management**: Zustand
- **Development Tools**: Git, Postman, VS Code

Project Structure

```
tana_market/
├── backend/
│   ├── models/         # MongoDB Mongoose models
│   ├── routes/         # API route handlers
│   ├── middleware/     # Authentication and authorization middleware
│   ├── utils/          # Utility functions
│   ├── scripts/        # Database setup and utility scripts
│   ├── uploads/        # Product image storage
│   └── server.js       # Main server entry point
├── frontend/
│   ├── src/
│   │   ├── components/  # Reusable UI components
│   │   ├── pages/       # Page components
│   │   ├── layouts/     # Layout components
│   │   ├── store/       # Zustand state management
│   │   ├── utils/       # Utility functions
│   │   └── App.jsx      # Main application component
│   └── public/         # Static assets
└── docs/               # Project documentation
```

Key Implementation Details

- **Authentication Implementation**: JWT token generation and verification, password hashing with bcrypt, role-based route protection
- **Product Management Implementation**: Product model with validation schemas, image upload handling, search and filter functionality
- **Order Management Implementation**: Order model with tracking number generation, status transition logic, payment integration
- **Payment Integration Implementation**: Chapa API integration for payment initiation, webhook handling for payment verification
- **Review System Implementation**: Review model with unique constraint (user, product), moderation workflow, rating calculation

4.5 System Testing

Testing Strategy

- Unit Testing: Individual components and functions
- Integration Testing: API endpoints and module interactions
- End-to-End Testing: Complete user workflows
- User Acceptance Testing: Stakeholder validation

Test Cases

- User registration and authentication
- Product creation and management
- Shopping cart operations
- Order placement and processing
- Payment processing integration
- Review submission and moderation
- Notification system
- Analytics and reporting

Testing Tools

- Jest for unit testing
- Supertest for API testing
- Postman for manual API testing
- React Testing Library for frontend testing
- MongoDB Memory Server for database testing

CHAPTER FIVE

IMPLEMENTATION AND TESTING

5.1 Implementation

Implementation is a crucial phase in the SDLC where the system design is transformed into a fully functional software solution. The project was divided into modules with team members contributing to specific areas:

Project Modules:

1. **User Module**: Authentication, registration, profile management, role-based access control
2. **Product Module**: Product creation, management, categorization, image handling
3. **Shopping Cart Module**: Cart operations, price calculation, persistence
4. **Order Module**: Order creation, status management, tracking number generation
5. **Payment Module**: Chapa integration, payment processing, verification
6. **Review Module**: Review submission, moderation, rating calculation
7. **Notification Module**: Email notifications, in-app notifications
8. **Analytics Module**: Dashboards, statistics, reporting
9. **Admin Module**: User management, system oversight, configuration

Implementation Approach

- **Modular Development**: Each module developed independently with clear interfaces
- **Continuous Integration**: Regular integration of modules to ensure compatibility
- **Code Reviews**: Peer code reviews to maintain quality and consistency
- **Documentation**: Comprehensive documentation including code comments and user guides

5.2 Testing

Testing ensures that the developed system functions correctly and meets defined requirements. A systematic testing process was applied:

Basic Functionality Testing

- Verified all UI components function as expected
- Tested forms, buttons, navigation, and user interactions
- Ensured no crashes or unexpected behavior

Code Review

- Peer-assisted code reviews to identify issues
- Security vulnerability assessment

- Code quality and consistency checks

Unit Testing

- Individual module testing including authentication, product management, order processing, payment
- Test cases designed to validate module behavior
- Compliance with design specifications verified

Sample Test Cases

Test Case 1: User Registration - Description: Verify user registration functionality - Pre-requisite: User email must not exist in database - Test Steps: 1. Navigate to Registration page 2. Enter valid user details (name, email, password, phone) 3. Click Register button - Test Data: Name: “Test User”, Email: “test@example.com”, Password: “TestPass123”, Phone: “0912345678” - Expected Result: User account created successfully, verification email sent, user logged in automatically

Test Case 2: Product Purchase Flow - Description: Verify complete product purchase workflow - Pre-requisite: User logged in, product available in stock - Test Steps: 1. Browse products and select a product 2. Add product to shopping cart 3. Proceed to checkout 4. Enter shipping information 5. Initiate payment 6. Complete payment on Chapa 7. Verify order confirmation - Expected Result: Order placed successfully, payment processed, order confirmation received, tracking number generated

Integration Testing

- Module combination testing
- API endpoint integration testing
- Payment gateway integration testing
- Database integration testing

System Testing

- Fully integrated application testing
- Data flow and operational timing validation
- End-to-end workflow testing
- Performance and load testing

User Acceptance Testing (UAT)

- End-user involvement in final testing phase
- Usability standards validation
- Stakeholder expectation verification
- Feedback collection for interface refinement
- Real-world scenario testing with actual users

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1 Conclusion

After dedicated effort and collaboration, it is evident that the proposed Tana Market E-Commerce Platform provides significant potential in improving online shopping experiences and business operations in Ethiopia. Throughout this project, the team gained important insights into the challenges faced by businesses, managers, and customers in managing e-commerce operations through fragmented, manual processes.

The motivation for this project stems from the understanding that efficient e-commerce platforms are crucial for business growth and customer satisfaction in the digital age. The application addresses these challenges by offering integrated workflows, secure payment processing, automated order tracking, and comprehensive analytics.

Specifically, the system is designed to: - Provide centralized e-commerce management platform - Streamline order processing and fulfillment workflows - Enable secure online payments through Chapa gateway - Automate order tracking with unique TANA tracking numbers - Deliver comprehensive analytics for data-driven business decisions - Maintain secure platform for sensitive customer and payment data - Ensure intuitive and responsive user interface across all devices

The team believes that the Tana Market platform is a reliable and effective tool for businesses seeking to establish and manage online stores. Although some limitations remain, the foundational framework for building and scaling the system has been clearly established. Moreover, this project lays the groundwork for future enhancements such as mobile applications, multi-vendor support, and advanced analytics capabilities.

The successful completion of this project demonstrates the application of modern software engineering principles and technologies, making it a valuable contribution to both the academic and practical domains. Through its role-based access control, payment integration, order tracking, and comprehensive management features, Tana Market serves as a complete e-commerce solution ready for real-world deployment.

6.2 Recommendations

Based on the analysis and findings of this project, the following recommendations are proposed:

1. **Strategic Implementation Planning**: Careful planning should be applied when deploying the system at scale, including risk identification, infrastructure readiness, and phased rollouts to different business segments.

2. **User Education and Support**: Users may require guidance to fully utilize system features. Training materials, in-app tutorials, user manuals, and support channels should be provided for both business users and customers.

3. **Expansion to Multiple Platforms**: Future versions should include native mobile applications for iOS and Android to reach broader audience and provide enhanced mobile shopping experiences.
4. **Advanced Analytics Integration**: Incorporating more sophisticated analytics and machine learning models can provide predictive insights, personalized recommendations, and inventory forecasting.
5. **Enhanced Security Measures**: Continuous updates to security protocols, implementation of two-factor authentication, and regular security audits are essential to protect sensitive customer and payment information.
6. **Multiple Payment Gateway Options**: Integration with additional payment gateways (in addition to Chapa) to provide users with more payment options and reduce dependency on single provider.
7. **Real-time Features Implementation**: WebSocket integration for real-time notifications, live inventory updates, and customer support chat to improve user experience.
8. **Cloud Storage Integration**: Implementation of cloud storage services for better scalability, reliability, and cost-effectiveness of product image storage.
9. **Multi-language and Localization Support**: Addition of multiple language options (including Amharic) and localization features to cater to diverse user base across Ethiopia.
10. **Mobile Application Development**: Development of native mobile applications for iOS and Android platforms with push notifications and mobile-optimized shopping experience.
11. **Advanced Inventory Management**: Implementation of barcode scanning, warehouse management, and automated reordering features for larger businesses.
12. **Customer Relationship Management**: Integration of CRM features for businesses to manage customer relationships, loyalty programs, and targeted marketing.
13. **API Development for Third-Party Integration**: Development of comprehensive API documentation and SDKs for third-party integrations with logistics providers, accounting software, and other business tools.
14. **Performance Optimization for Scale**: Continuous performance monitoring and optimization for handling large-scale deployments with thousands of concurrent users.
15. **Disaster Recovery and Business Continuity Planning**: Implementation of comprehensive backup, disaster recovery, and business continuity mechanisms to ensure platform reliability.

The Tana Market E-Commerce Platform represents a significant step forward in digital commerce solutions for Ethiopia. With continued development and implementation of these recommendations, the system has the potential to become a leading platform in the Ethiopian e-commerce industry, serving diverse needs across retail, services, and various business sectors.

References

1. Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education Limited.
2. Pressman, R. S., & Maxim, B. R. (2019). *Software Engineering: A Practitioner’s Approach* (9th ed.). McGraw-Hill Education.

3. MongoDB Inc. (2023). *MongoDB Documentation*. Retrieved from https://docs.mongodb.com/
4. React Team. (2023). *React Documentation*. Retrieved from https://react.dev/
5. Node.js Foundation. (2023). *Node.js Documentation*. Retrieved from https://nodejs.org/docs/
6. Express.js. (2023). *Express.js Documentation*. Retrieved from https://expressjs.com/
7. Chapa. (2023). *Chapa API Documentation*. Retrieved from https://developer.chapa.co/
8. IEEE Computer Society. (1998). *IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications*. IEEE.
9. World Wide Web Consortium (W3C). (2023). *Web Content Accessibility Guidelines (WCAG) 2.1*. Retrieved from https://www.w3.org/WAI/WCAG21/quickref/
10. OWASP Foundation. (2021). *OWASP Top 10 Web Application Security Risks*. Retrieved from https://owasp.org/www-project-top-ten/

Appendices

Appendix A: Technology Specifications

Backend Technologies

- **Node.js**: Version 18.0 or higher
- **Express.js**: Version 4.18.2
- **MongoDB**: Version 6.0 or higher
- **Mongoose**: Version 8.0.3
- **JWT**: Version 9.0.2
- **bcryptjs**: Version 2.4.3
- **Multer**: Version 1.4.5-lts.1

Frontend Technologies

- **React**: Version 18.2.0
- **Vite**: Version 7.3.1
- **React Router**: Version 6.20.1
- **Zustand**: Version 4.4.7
- **Tailwind CSS**: Version 3.3.6
- **Axios**: Version 1.6.2

Appendix B: Sample API Endpoints

Authentication Endpoints

- POST /api/auth/register - User registration
- POST /api/auth/login - User login
- GET /api/auth/profile - Get user profile

Product Endpoints

- GET /api/products - List products with pagination
- GET /api/products/:id - Get product details
- POST /api/products - Create product (admin/manager)
- PUT /api/products/:id - Update product (admin/manager)

Order Endpoints

- POST /api/orders - Create order
- GET /api/orders - Get user orders
- GET /api/orders/tracking/:trackingNumber - Track order
- PUT /api/orders/:id/cancel - Cancel order

Payment Endpoints

- POST /api/payments/initialize - Initialize payment
- POST /api/payments/verify - Payment webhook verification

Appendix C: System Requirements

Hardware Requirements

- **Server**: Minimum 2 CPU cores, 4GB RAM, 20GB storage
- **Client**: Modern web browser with JavaScript enabled

Software Requirements

- **Server**: Node.js 18+, MongoDB 6.0+
- **Client**: Chrome, Firefox, Safari, or Edge (latest 2 versions)

Network Requirements

- Stable internet connection
- HTTPS for production deployment
- Open ports for web traffic (80, 443)

Appendix D: Deployment Guide

Development Deployment

1. Clone repository
2. Install dependencies: `npm install` in both backend and frontend directories
3. Configure environment variables (.env files)
4. Start backend: `npm run dev` in backend directory
5. Start frontend: `npm run dev` in frontend directory

Production Deployment

1. Build frontend: `npm run build` in frontend directory
2. Configure production environment variables
3. Use process manager (PM2) for backend
4. Configure reverse proxy (Nginx/Apache)
5. Set up SSL certificates
6. Configure database backups and monitoring

Appendix E: User Manual Highlights

For Customers

1. **Registration**: Create account with email and password
2. **Shopping**: Browse products, add to cart, proceed to checkout
3. **Payment**: Complete payment through Chapa gateway
4. **Order Tracking**: Use TANA tracking number to track order status
5. **Reviews**: Submit reviews for purchased products

For Managers

1. **Product Management**: Add, edit, and manage products
2. **Order Processing**: Approve and ship orders
3. **Review Moderation**: Approve or reject customer reviews

4. **Analytics**: View sales reports and performance metrics

For Administrators

1. **User Management**: Manage all user accounts
2. **System Oversight**: Monitor all activities and payments
3. **Configuration**: Manage system settings and configurations