MERN COMMERCE

By

KRUNAL MISTRY (23MCA085)

NEEL PATEL (23MCA131)

UTSAV SHAH (23MCA181)

Under Guidance

of

Internal Guides

Ravi Nimavat

Submitted to

Smt. Chandaben Mohanbhai Patel Institute of Computer Applications
CHARUSAT
Changa

March-April 2025

Accredited with Grade A+ by NAAC,
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Changa

Acknowledgement

Knowledge in itself is a continuous process. At this moment of our substantial enhancement, We rarely find words to express our gratitude towards those who were constantly involved with us.

The completion of any inter disciplinary project depends upon coordination, cooperation and combined efforts of several resources of knowledge, creativity, skill, energy and time. The work being accomplished now, we feel our sincerest urge to recall and knowledge through these lines, trying our best to give full credit wherever it deserves.

We would like to thank our project guide **Ravi Nimavat,** I/C Principal **Dr. Dharmendra Patel** and I/C Dean **Dr. Sanskruti Patel** who advised and gave us moral support through the duration of our project. Without their constant encouragement we could not have been able to achieve what we have.

It's our good fortune that we had support and well wishes of many. We are thankful to all and those names which have been forgotten to acknowledge here but contributions have not gone unnoticed.

With Sincere Regards,
Krunal Mistry (23MCA085)
Neel Patel (23MCA131)
Utsav Shah (23MCA181)

Table of Contents

Sr.No		Subject	Page Number
1		**Project Profile & Company Profile**	**1**
2		**Introduction to tools**	**3**
3		**System Study**	
	1.	Existing System	**10**
	2.	Proposed System	**10**
	3.	Scope of the Proposed System	**11**
	4.	Aim and Objective of the Proposed System	**11**
	5.	Feasibility Study	**13**
		Operational Feasibility	**13**
		Technical Feasibility	**14**
		Economical Feasibility	**14**
4		**System Analysis**	
		Requirements Specification (along with System Modules)	**16**
		Use Case Diagram	**18**
		Activity Diagram	**19**
		Class Diagram	**24**
5		**System Design**	
		Data Dictionary	**19**
		Screen Layouts	
		Reports	
6		**System Testing**	
		Testing Strategies	
		Test Cases	
7		**Future Enhancement**	
8		**Bibliography/References**	
9		**Reporting Report (to be collected from respective internal guide)**	

PROJECT PROFILE

❖ Project Profile

Project Name: MERNCommerce

Type of Application: Web Application

Project Description:

- MERNCommerce is a dynamic and fully functional e-commerce platform designed to provide a seamless shopping experience for customers while ensuring smooth business operations for administrators and management teams. Built using the MERN stack (MongoDB, Express.js, React.js, and Node.js), the system leverages modern web technologies to deliver a highly scalable and efficient solution. The platform is structured around five key user roles: Admin, Delivery Partner, Inventory Manager, Finance Manager, and End User, each playing a crucial role in maintaining the overall functionality. The admin has complete control over the platform, including product and category management, order processing, and promo code management. Sub-admin roles such as Delivery Partners handle order deliveries and cash-on-delivery payments, while Inventory Managers monitor stock levels and receive notifications for low or out-of-stock products. Finance Managers track total sales and manage COD payments. End Users, or customers, can browse products, place orders, apply promo codes, and make online payments through secure payment gateways like Stripe or Razorpay. The platform ensures a smooth user experience with real-time notifications for order confirmations, delivery updates, and promotional offers.

- One of the key highlights of MERNCommerce is its efficient order management system, designed to streamline the entire process from product selection to order delivery. Admins receive instant notifications when a customer places an order and can assign orders to Delivery Partners, who are notified via email. Delivery Partners can track their assigned deliveries, confirm cash-on-delivery payments, and update the system once an order is successfully delivered. The admin also has visibility into all pending COD transactions, ensuring financial transparency. Additionally, the system features a dedicated Finance Manager role to oversee sales reports, categorize revenue, and process COD payments. This structured workflow ensures that every role within the platform functions efficiently, minimizing errors and improving overall operational effectiveness. To further enhance usability, the platform provides role-based access control (RBAC), ensuring that each user only has access to the necessary features and data relevant to their role. Security is prioritized through JWT-based authentication, secure password handling with bcrypt, and email verification for sub-admin accounts.

- For customers, MERNCommerce offers a convenient and feature-rich shopping experience with advanced product search, filtering, and personalized recommendations. Users can manage their profiles, save multiple addresses, and track their order history. The platform also supports gift orders, allowing customers to include gift wraps and personalized notes during checkout. A well-integrated email notification system keeps users updated on every stage of their order, from placement to delivery. Additionally, customers can download invoices for their purchases, ensuring complete transparency in transactions. The platform also offers promotional campaigns, where admins can create and manage discount codes, providing customers with exciting offers to enhance their shopping experience. Payment security is a top priority, with support for both online payments and cash-on-delivery, ensuring flexibility for users.

- MERN E-Commerce goes beyond a simple online shopping experience by integrating robust inventory and finance management features. Inventory Managers can monitor stock levels in real-time and receive notifications when products run low, ensuring efficient restocking. The platform also includes comprehensive sales tracking, with Finance Managers able to generate detailed reports on revenue, sales by category, and overall business performance. This data-driven approach allows business owners to make informed decisions and optimize their operations. Furthermore, the system includes automation features, such as notifications for out-of-stock products and order assignment, reducing manual workload and increasing efficiency. The platform's architecture is designed to handle high traffic, making it suitable for businesses of all sizes.

- MERN E-Commerce will continue to evolve with advanced features like AI-powered product recommendations, automated discount calculations, and advanced analytics for better business insights. Future updates will optimize pricing strategies and enhance customer experiences. Additionally, a mobile app version is in development to improve accessibility. With its scalable and feature-rich architecture, MERN E-Commerce is a reliable solution for modern businesses.

Team Size: 3

Front End: React.js, HTML, CSS, Material UI

Back End: Node.js, Express.js, MongoDB

Tools used: Visual Studio Code (VSCode), MongoDB Atlas, Postman API

❖ Company Profile

Company Name: Triosphere Tech

Website: www.triospheretech.com

Company Overview:

Triosphere Tech is an innovative technology company specializing in software development, digital transformation, and IT solutions. The company focuses on providing high-quality, scalable, and cost-effective solutions to businesses across various industries.

Core Services:

- Web & Mobile App Development
- Cloud Computing Solutions
- E-commerce Development
- Enterprise Software Solutions

Mission Statement:

Triosphere Tech aims to empower businesses with cutting-edge technology, helping them scale and achieve digital excellence.

Vision:

To be a global leader in technology solutions by delivering innovation-driven products and services.

Clients & Industries Served:

- Retail & E-commerce
- Healthcare
- Finance
- Logistics
- Education

Triosphere Tech is committed to delivering innovative, scalable, and cost-effective technology solutions, empowering businesses across industries to achieve digital excellence.

INTRODUCTION TO TOOLS

❖ Introduction to Tools

➢ Front End Tool:

- **React.js**
  - React.js is a powerful JavaScript library used for building dynamic and responsive user interfaces. It allows for component-based architecture, enabling reusable UI elements and efficient rendering using a virtual DOM. In MERNCommerce, React.js is responsible for handling the front-end, ensuring smooth navigation, state management, and interactive features such as filtering products, managing user profiles, and updating order statuses. React hooks like useState and useEffect are used for managing component behavior efficiently.

- **HTML**
  - HTML (Hypertext Markup Language) is the backbone of web pages, defining the structure and content layout of the website. In MERN-ECommerce, HTML is used to create semantic page structures such as product listings, category sections, user dashboards, and forms for user interactions like checkout and login. It provides accessibility and SEO-friendly features, ensuring a better user experience.

- **CSS**
  - CSS (Cascading Style Sheets) is used for styling and designing the user interface to make the platform visually appealing. In MERNCommerce, CSS is utilized for layout design, color themes, and responsiveness, ensuring that the platform works well across different screen sizes. Media queries and Flexbox/Grid are used for structuring product categories, navigation menus, and checkout pages.

- **Material UI**
  - Material UI is a React component library that enhances the user interface with pre-built, customizable components following Google's Material Design principles. In the billing system, it is used to create consistent UI elements such as buttons, tables, modals, and forms. Material UI ensures an intuitive and professional design, providing a seamless experience for users interacting with invoices, company profiles, and dashboards.

➢ Back-End Tools

• Node.js

- Node.js serves as the backbone of the application's back end, handling all server-side operations efficiently. Built on Chrome's V8 engine, Node.js allows developers to execute JavaScript on the server, enabling full-stack development using a single programming language. In the billing system, Node.js is responsible for user authentication, company data management, invoice processing, and financial tracking. Its event-driven, non-blocking I/O model ensures that multiple client requests can be handled simultaneously without affecting performance. Additionally, Node.js facilitates easy integration with third-party services like email notifications, enabling automatic invoice delivery to clients.

• Express.js

- Express.js is a minimal and powerful web application framework for Node.js that simplifies server-side development. It provides an efficient way to manage API routes and middleware, making it easier to handle authentication, database interactions, and business logic. In the billing system, Express.js is used to implement RESTful APIs for managing user authentication, invoice creation, payment status updates, and dashboard data retrieval. Middleware functions in Express help in security measures such as authentication, request validation, and error handling. By using Express.js, the back-end structure remains clean, scalable, and easy to maintain.

• MongoDB

- MongoDB is a NoSQL database that provides a flexible and scalable solution for storing and managing financial and company-related data. Unlike traditional relational databases, MongoDB uses a document-oriented approach, allowing data to be stored in JSON-like structures. This flexibility is beneficial for managing dynamic records such as pending bills, received payments, and company profiles. The billing system leverages MongoDB to efficiently store invoice details, track payment statuses, and retrieve financial reports. With MongoDB's ability to handle large volumes of data and perform fast queries, the system remains responsive even as the number of transactions grows. Additionally, features like indexing and aggregation help generate detailed financial insights efficiently.

➢ Tools Used:

The back end of the Company Secretary Billing System is designed to handle server-side logic and database management using Node.js and MongoDB. Together, they ensure fast and scalable data processing, supporting a seamless user experience.

1. Visual Studio Code (VSCode)

- VSCode is a widely used code editor that provides a rich development environment with features like IntelliSense, debugging, and Git integration. In MERNCommerce, VSCode is used for writing, testing, and debugging both front-end and back-end code. It supports extensions like Prettier, which help maintain clean and error-free code. The built-in terminal allows developers to run Node.js servers, execute database queries, and manage version control with Git seamlessly. Additionally, its vast marketplace offers extensions for React, MongoDB, and API testing, enhancing productivity and streamlining the development process.

• MongoDB Atlas

- MongoDB Atlas is a cloud-based database service used in MERNCommerce for managing and storing data securely. It provides automated scaling, backups, and real-time performance monitoring, ensuring high availability and reliability. With built-in security features like encryption and access control, Atlas allows developers to manage products, orders, and user data efficiently. It also supports advanced analytics, indexing optimization, and seamless integration with the MERN stack, enabling faster query execution and better database performance. The cloud-based nature of Atlas eliminates the need for manual database maintenance, making development and deployment more efficient.

• Postman API

- Postman is a powerful API testing tool used for sending, receiving, and debugging HTTP requests. In MERNCommerce, Postman is used to test backend APIs, ensuring proper communication between the front-end and database. It helps verify endpoints for product management, user authentication, and order processing before integrating them into the application.

SYSTEM STUDY

❖ System Study

➢ Existing System

- Traditional e-commerce platforms provide basic functionalities for browsing products, placing orders, and making payments. However, many existing systems have inefficiencies due to a lack of structured role management and automation. **Admins** often have to manage product listings manually, assign orders one by one, and track payments without real-time updates. **Delivery partners** do not have a dedicated dashboard to track assigned orders and cash-on-delivery (COD) payments, leading to potential mismanagement. **Inventory managers** rely on manual stock checks without automated low-stock alerts, which can result in inventory shortages or overstocking. **Finance teams** struggle to monitor sales performance effectively as existing systems lack category-wise or monthly sales tracking features. **End users** face difficulties in tracking their orders, receiving timely notifications, and utilizing promo codes efficiently. Overall, traditional systems are **time-consuming, lack automation, and do not provide a seamless multi-role experience.**

➢ Proposed System

- The **MERNCommerce** platform eliminates inefficiencies by offering a **robust, multi-role e-commerce solution** that automates tasks, improves efficiency, and enhances user experience. The system introduces **role-based access control (RBAC)**, where each user type (Admin, Delivery Partner, Inventory Manager, Finance Manager, and End User) has specific permissions and functionalities.

1. **Admins** can efficiently manage products, categories, sub-admins, and orders while assigning delivery tasks and tracking COD payments.
2. **Delivery Partners** receive real-time notifications when assigned an order and can track payments collected from customers, ensuring smooth financial accountability.
3. **Inventory Managers** are notified about low-stock or out-of-stock items, enabling timely restocking and reducing supply chain disruptions.
4. **Finance Managers** can track total sales, view sales breakdowns by category, and generate financial reports for data-driven decision-making.
5. **End Users** enjoy a seamless shopping experience, with advanced filtering, order tracking, promo codes, and email notifications.

The system also integrates **JWT authentication for security**, **Stripe/Razorpay for secure online payments**, and **NodeMailer/SendGrid for email notifications.** By automating crucial operations and ensuring clear communication between different roles, **MERNCommerce enhances efficiency, security, and customer satisfaction.**

➢ Scope of the Proposed System

- The **MERNCommerce** system is designed to **cater to small and mid-sized e-commerce businesses** that require a scalable and efficient management solution. This system provides **full-stack e-commerce functionality** by integrating:

  1. **Comprehensive Role Management:** Ensuring that Admins, Delivery Partners, Inventory Managers, Finance Managers, and End Users have dedicated functionalities, reducing workload and errors.
  2. **Automated Order Processing:** Orders are instantly assigned to Delivery Partners with notifications, and COD payments are tracked in real-time to ensure transparency.
  3. **Inventory Optimization:** The system monitors stock levels and notifies managers about low or out-of-stock items to prevent sales loss.
  4. **Advanced Payment Management:** Supports both **COD and online payment processing**, ensuring a smooth transaction experience for customers and businesses.
  5. **Secure Authentication:** Implements **JWT authentication** for API security, **bcrypt for password encryption,** and **email verification for Sub-Admins.**
  6. **Sales & Financial Analytics:** Finance Managers can track **monthly and yearly sales reports**, providing insights for business growth strategies.
  7. **Customer Engagement & Experience:** The system supports **promo codes, order tracking, email notifications, invoice downloads, and gift options**, ensuring an **enhanced user experience.**

This system can be **further enhanced** with AI-driven recommendations, automated discount calculations, and a mobile app version in the future.

➢ Aim and Objective of the Proposed System:

- **Aim** The aim of MERNCommerce is to develop a feature-rich, scalable, and secure e-commerce platform that streamlines business operations, enhances user experience, and ensures efficient order and inventory management. By leveraging the MERN stack, the system aims to provide seamless interactions between Admins, Delivery Partners, Inventory Managers, Finance Managers, and End Users while maintaining high security standards and real-time notifications.

- ### Objective

  - To automate order assignments and allow bulk assignment of orders to Delivery Partners.
  - To ensure seamless tracking of COD payments with real-time updates.
  - To enable product variant-wise pricing and discounts (e.g., 250g, 500g, 1kg).
  - To implement Stripe/Razorpay for secure online transactions and track COD payments.
  - To provide real-time email notifications for order status updates, promotions, and payment

confirmations.

- To develop a user-friendly frontend with React.js for fast, responsive shopping.
- To implement role-based access control (RBAC) for security and data integrity.
- To expand the system with AI-driven recommendations, advanced analytics, and a mobile app version in the future.

1. Optimize Order and Delivery Management

- Automate order assignments and allow bulk assignment of orders to Delivery Partners.
- Ensure seamless tracking of COD payments with real-time updates.
- Provide notifications for all role-based operations (order placement, payment received, low stock, etc.).

2. Enhance Product & Inventory Management

- Enable product variant-wise pricing and discounts (e.g., 250g, 500g, 1kg).
- Allow Admins to manage product tags for improved search and categorization.
- Alert Inventory Managers when products reach low stock levels or go out of stock.

3. Secure Transactions & Financial Transparency

- Implement Stripe/Razorpay for secure online transactions.
- Track and manage COD payments, ensuring Delivery Partners and Admins confirm fund transfers.
- Generate detailed sales reports based on date, category, and payment type.

4. Streamline Communication and Notifications

- Use NodeMailer/SendGrid to send real-time email notifications for order status updates, promotions, and payment confirmations.
- Keep End Users engaged by sending special offer notifications via email.

5. Improve Customer Experience and Engagement

- Provide an easy-to-use frontend with React.js for fast, responsive shopping.
- Enable customers to track orders, download invoices, and apply promo codes easily.
- Allow gift-ordering options with gift wrap and custom notes.

6. Ensure Data Security and Access Control

- Use JWT authentication to prevent unauthorized access.
- Implement role-based access control (RBAC) so that users only access the data relevant to their roles.
- Encrypt passwords using bcrypt, ensuring secure credential storage.

➢ Feasibility Study

1. **Operational Feasibility**
2. **Technical Feasibility**
3. **Economical Feasibility**

- **Operational Feasibility**
    - **User Accessibility**: MERNCommerce is designed with distinct roles (Admin, Delivery Partner, Inventory Manager, Finance Manager, and End User) ensuring smooth workflow and easy accessibility for all users.
    - **Order Management Efficiency**: Automated email notifications and role-based dashboards streamline order processing, reducing manual effort.
    - **Security & Access Control**: Role-based access control (RBAC) ensures that users only access the functionalities relevant to their role, enhancing security.
    - **Scalability**: Built using the MERN stack, the system can handle increasing product listings, user registrations, and transactions without performance issues.
    - **Inventory & Payment Tracking**: Real-time inventory updates and COD payment tracking enhance operational control and minimize discrepancies.
    - **Customer Engagement**: Features like promo codes, personalized emails, and order tracking ensure a smooth shopping experience for end users.
    - **Future Expansion**: The system is flexible for future enhancements like AI-based product recommendations and advanced analytics.

- **Technical Feasibility**
    - **Technology Stack**: MERN (MongoDB, Express.js, React.js, Node.js) provides a robust, scalable, and modern architecture for building a high-performance e-commerce platform.
    - **Database Management**: MongoDB, a NoSQL database, allows efficient handling of product listings, orders, and user data with high scalability.
    - **Authentication & Security**: JWT-based authentication ensures secure access, while bcrypt encrypts passwords for enhanced security.
    - **API Communication**: RESTful API architecture allows smooth communication between frontend and backend services, ensuring reliability and efficiency.
    - **Email & Notification Services**: NodeMailer or SendGrid ensures reliable email notifications for orders, payments, and promotions.
    - **Payment Gateway Integration**: Stripe or Razorpay provides secure and seamless online payment processing for users.
    - **Hosting & Deployment**: Can be deployed on cloud services like AWS, DigitalOcean, or Vercel for high availability and performance.

- **Economic Feasibility**
  - **Development Cost**: The MERN stack is open-source, reducing software licensing costs, and development can be managed efficiently with a small team.
  - **Operational Costs**: Cloud hosting costs can be optimized with services like AWS free tier, DigitalOcean, or Firebase for certain functionalities.
  - **Revenue Generation**: The platform can generate revenue through direct product sales, subscription models, and advertising of featured products.
  - **Return on Investment (ROI)**: Given the demand for e-commerce platforms, a well-marketed solution can quickly break even and generate profit.
  - **Maintenance Costs**: Regular updates and security patches require minimal expenses due to the availability of open-source libraries and frameworks.
  - **Scalability Considerations**: The system is designed to handle increasing traffic and product data without significant additional costs.
  - **Competitive Advantage**: Features like AI-based recommendations, automated discounts, and advanced analytics can drive higher user engagement and sales.

SYSTEM ANALYSIS

❖ System Analysis:

- The E-Commerce Order Management System is designed to handle product orders, manage user payments, and track deliveries efficiently. It supports various user roles, including Admin, Customers, and Delivery Partners, ensuring seamless order processing

➢ Requirements:

1. Functional Requirements

- **User Management**
  - Users can register and log in to the system.
  - Role-based access for Admin, Customer, and Delivery Partner**.**
  - Customers can update profile details and manage addresses**.**
- **Product Management**
  - Admin can add, update, and delete products.
  - Products can have multiple variants (weight, price, discount, etc.).
  - Product details, including images, descriptions, and GST rates, must be maintained.
- **Cart Management**
  - Customers can add, update, and remove products from the cart.
  - The cart should display total cost, including GST and discounts.
- **Order Management**
  - Customers can place orders with selected variants.
  - The system generates unique order IDs for tracking.
  - Orders contain delivery details, payment status, and applied promo codes.
  - Admin can assign orders to delivery partners.
  - Delivery partners can update order status (e.g., Out for Delivery, Delivered).
- **Payment & Invoices**
  - Supports CASH ON DELIVERY (COD) and Online Payment methods.
  - Stores payment transaction IDs for tracking.
  - Generates invoices for each completed order.
- **Delivery Management**
  - Users can track order status.
  - The system maintains delivery addresses and assigns orders to delivery partners.
  - Delivery partners can update the delivery status and time.

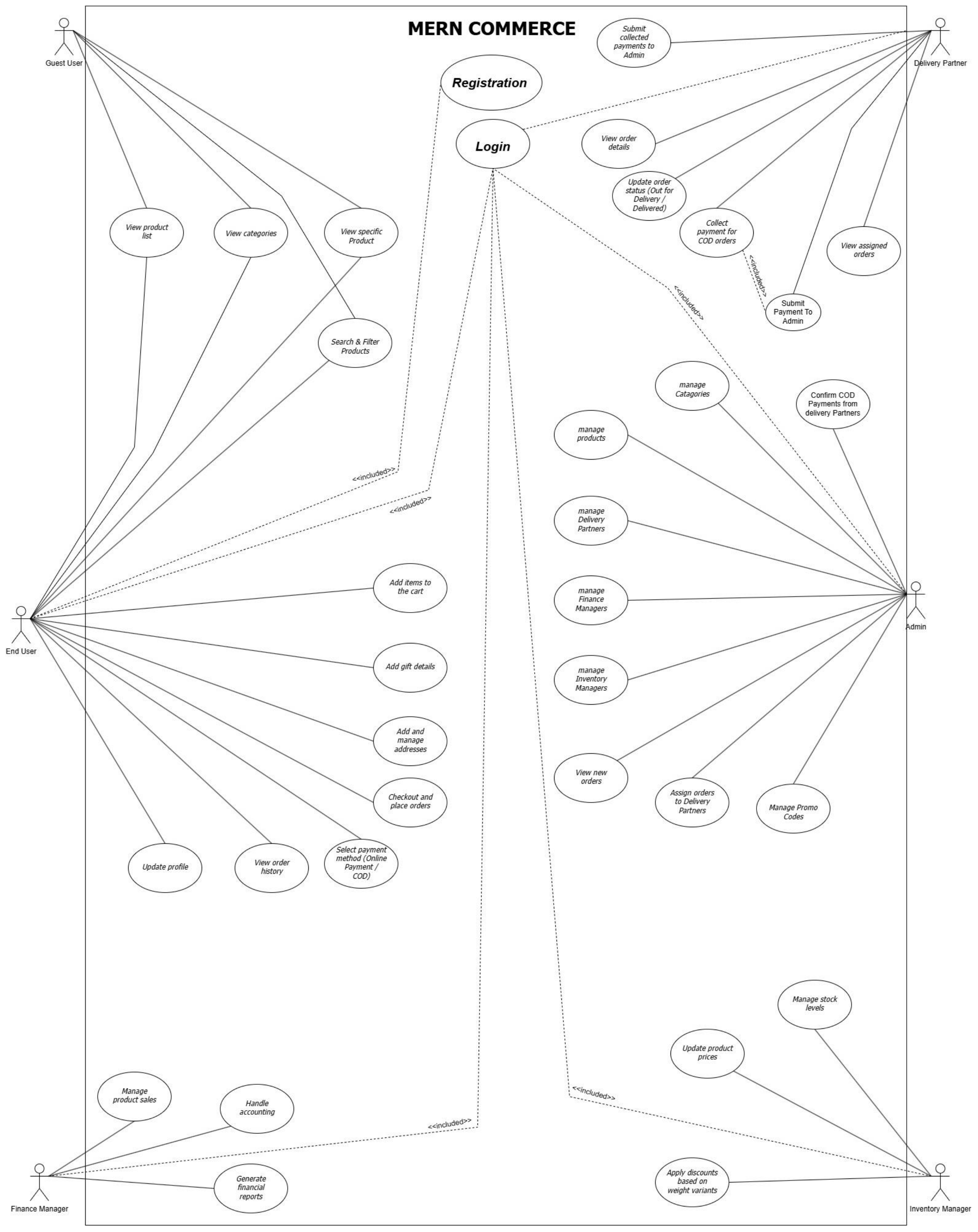
2. Non-Functional Requirements

- **Performance**
  - The system should handle multiple concurrent users efficiently.
  - Order placement and payment processing should be fast and seamless.
- **Scalability**
  - The system should support an increasing number of products, users, and orders.
  - Designed for future expansion, including additional payment gateways and delivery tracking.
- **Security**
  - Implements role-based authentication for users.
  - Encrypts sensitive data such as passwords and payment information.
  - Prevents unauthorized access to orders, payments, and user data.
- **Usability**
  - Provides a user-friendly interface for customers, admins, and delivery partners.
  - The admin panel should allow easy product and order management.
- **Reliability**
  - Ensures consistent data storage and retrieval.
  - The system should be available 24/7 with minimal downtime.

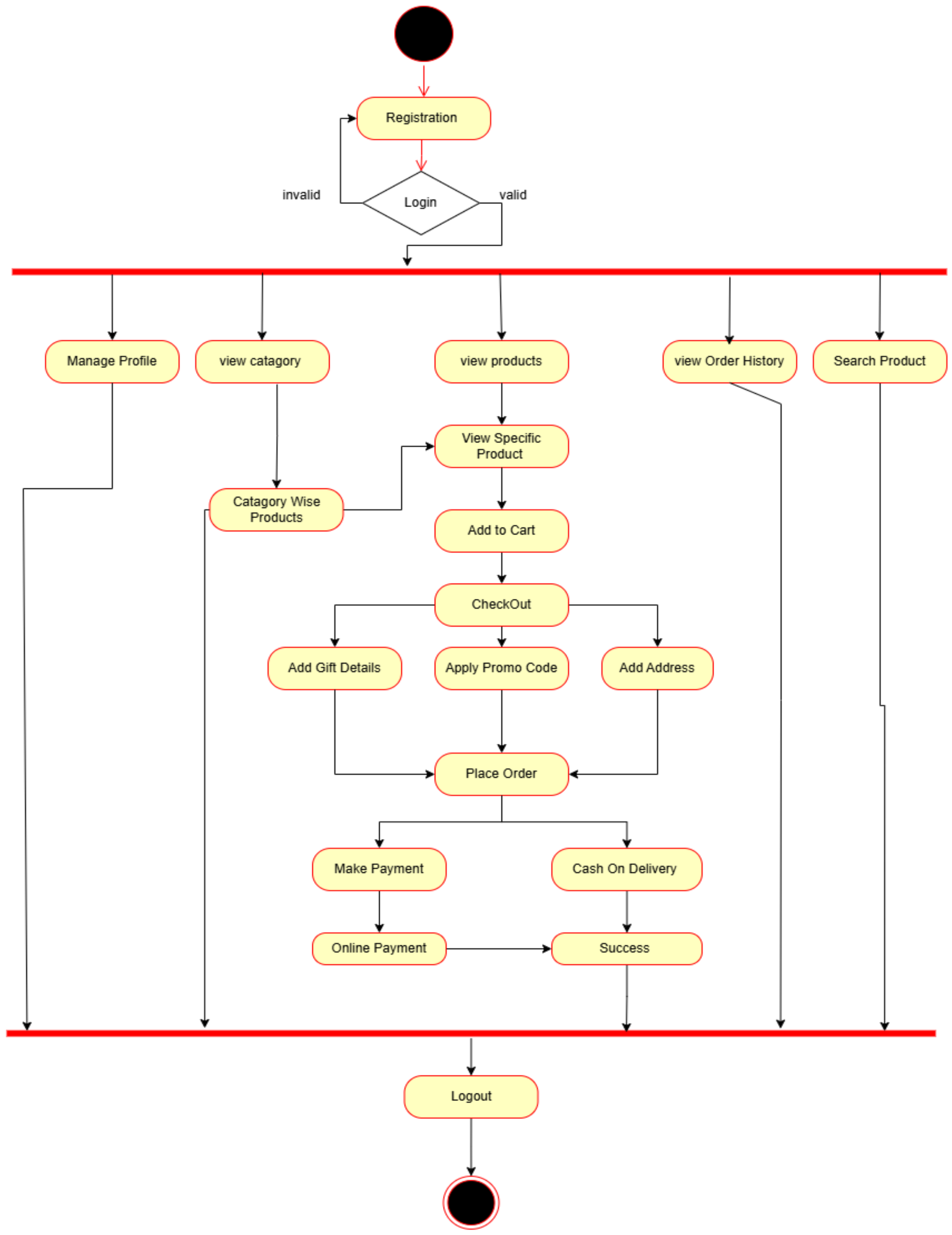
➢ System Modules

1. **User Module:** Manages authentication, user profiles, addresses, and order history.
2. **Product Management Module:** Allows admins to manage products, variants, and GST details.
3. **Cart Module:** Handles cart operations, pricing, and tax calculations.
4. **Order Processing Module:** Manages order creation, tracking, and delivery assignments.
5. **Payment Module:** Supports COD and online payments, stores transactions, and applies discounts.
6. **Delivery Management Module:** Tracks delivery addresses, assigns orders, and updates statuses.
7. **Admin Dashboard Module**: Provides order monitoring, user management, and sales analytics.

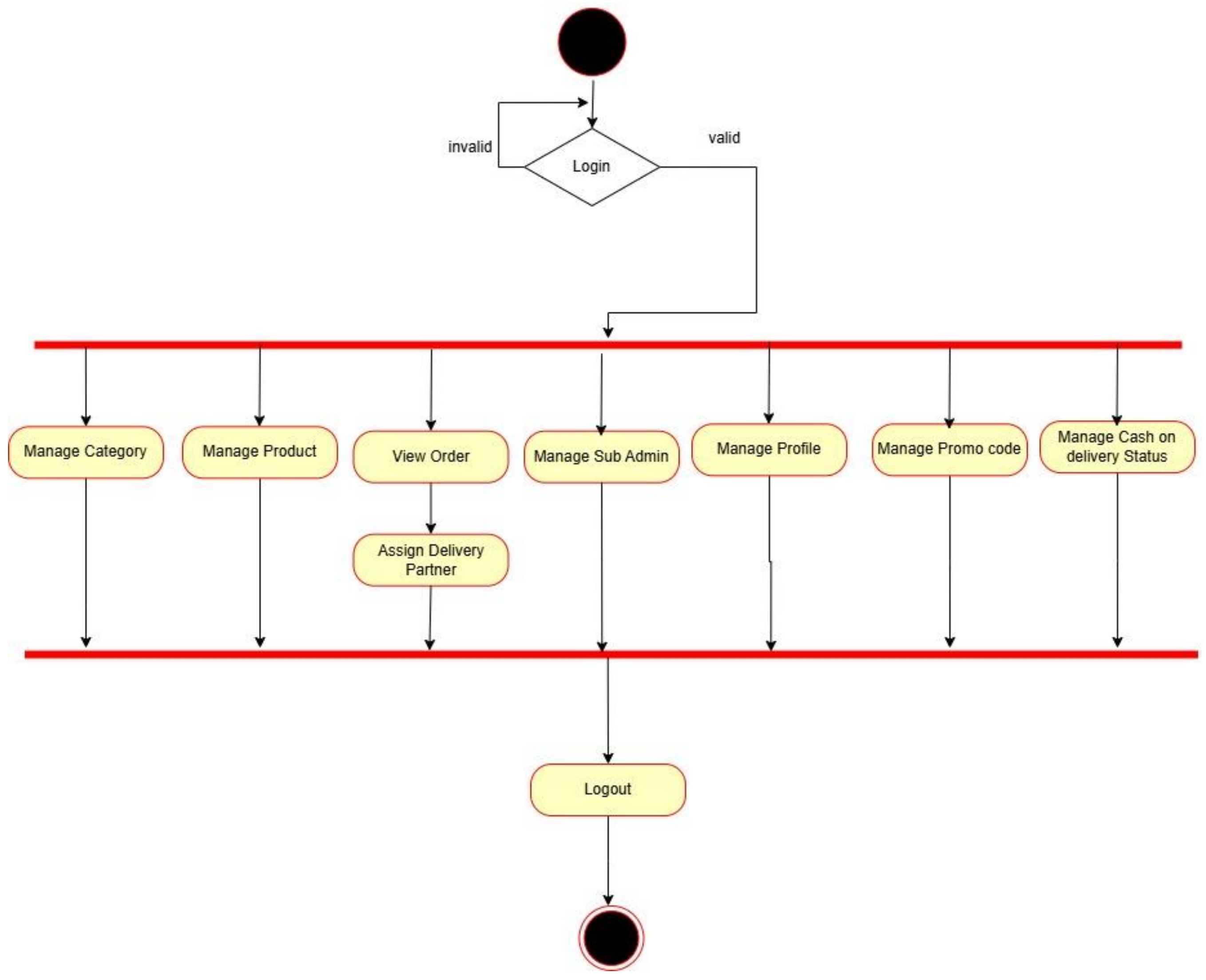
❖ Use Case Diagram:



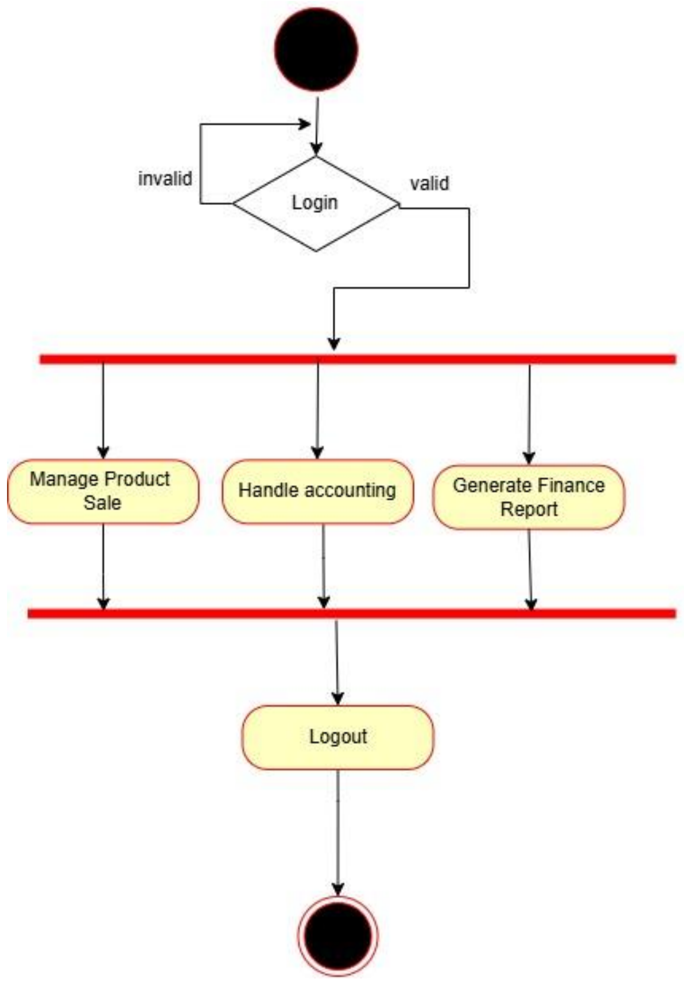
❖ User Activity Diagram:



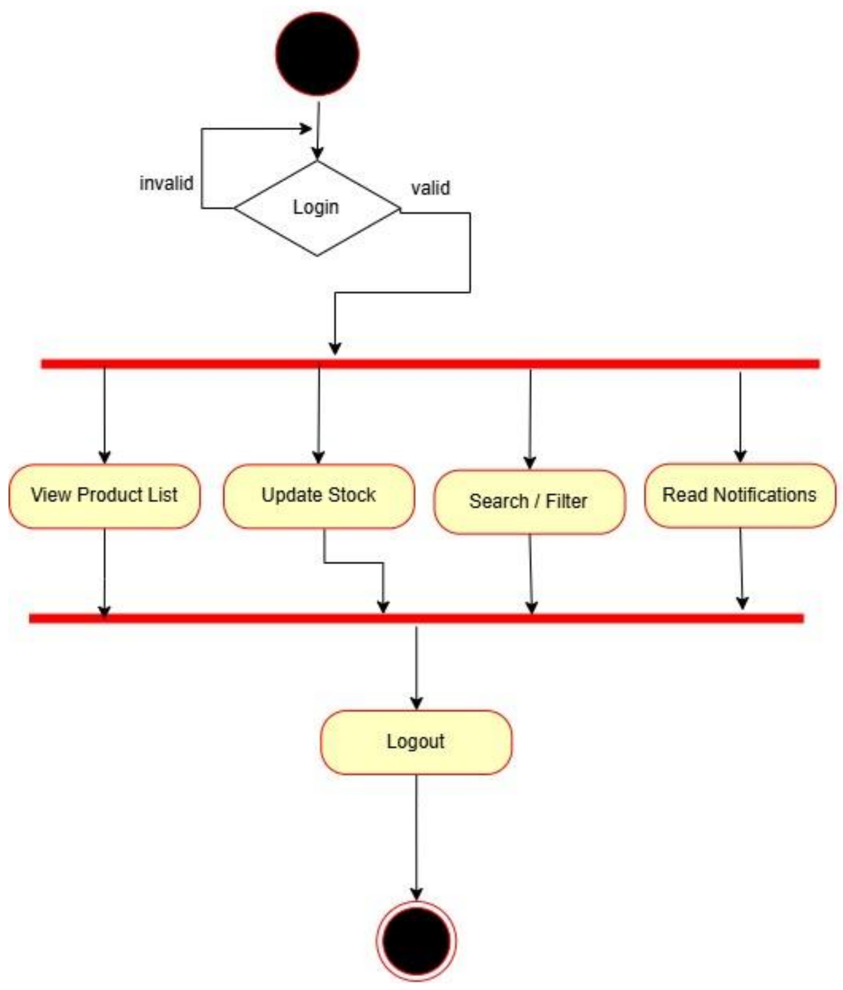
❖ **Admin Activity Diagram:**



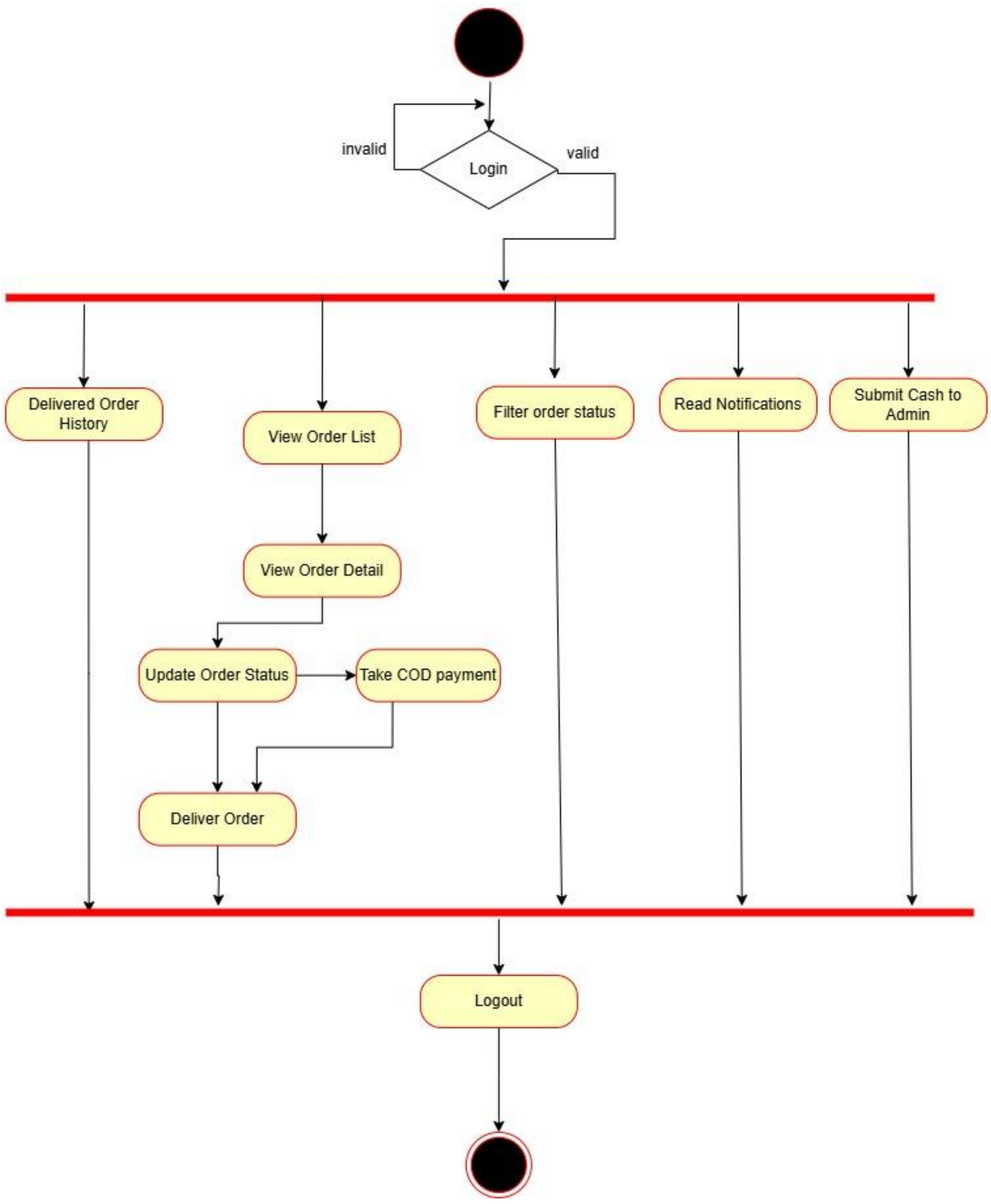
❖ Finance Manager Activity Diagram:



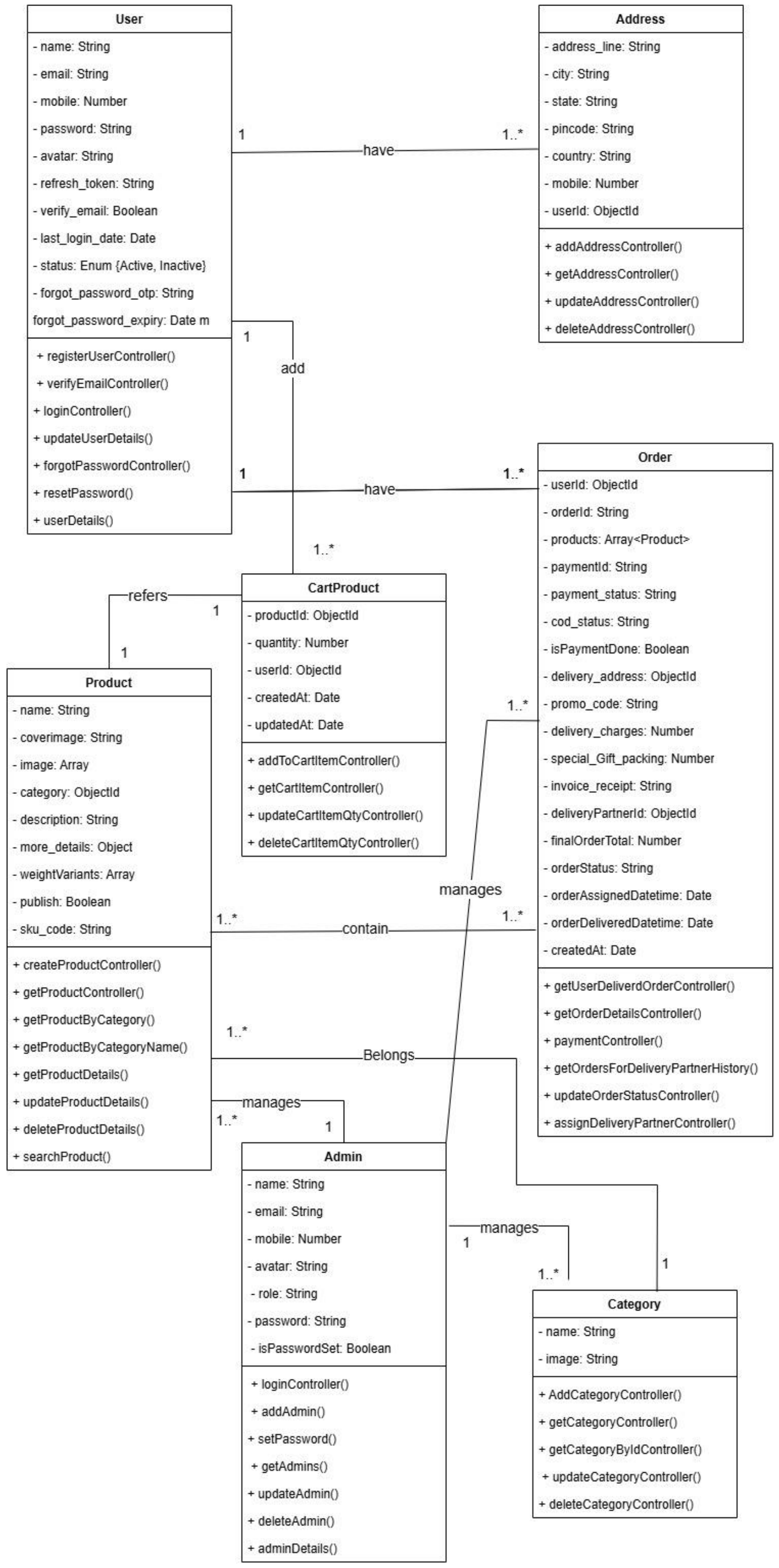
❖ **Inventory Manager Activity Diagram:**



❖ Delivery Partner Activity Diagram:



❖ Class Diagram:



SYSTEM DESIGN

❖ Data Dictionary

1. User Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the user	Auto-generated
`name`	String	Full name of the user	Required
`email`	String	User's email address	Required, Unique
`password`	String	Encrypted user password	Required
`avatar`	String	URL of the user's profile picture	Default: ""
`mobile`	Number	User's mobile number	Default: null
`refresh_token`	String	Token used for authentication refresh	Default: ""
`verify_email`	Boolean	Email verification status	Default: false
`last_login_date`	Date	Last login timestamp	Default: ""
`status`	String	Account status	Enum: [Active, Inactive, Suspended], Default: Active
`address_details`	Array(ObjectId)	List of saved addresses	References `address`
`shopping_cart`	Array(Object)	User's shopping cart items	Embedded Document
`orderHistory`	Array(ObjectId)	User's past orders	References `order`
`forgot_password_otp`	String	OTP for password reset	Default: null
`forgot_password_expiry`	Date	Expiry date for password reset OTP	Default: ""
`role`	String	User's role	Enum: [ADMIN, USER], Default: USER
`createdAt`	Date	Timestamp when the record was created	Auto-generated
`updatedAt`	Date	Timestamp when the record was last updated	Auto-generated

2. Admin Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the admin	Auto-generated
`name`	String	Full name of the admin	Required
`email`	String	Admin's email address	Required, Unique
`mobile`	Number	Admin's mobile number	Default: null
`avatar`	String	URL of the admin's profile picture	Default: ""
`role`	String	Admin's role	Required
`password`	String	Encrypted admin password	Optional
`isPasswordSet`	Boolean	Indicates if password has been set	Default: false
`createdAt`	Date	Timestamp when the record was created	Auto-generated
`updatedAt`	Date	Timestamp when the record was last updated	Auto-generated

3. Category Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the category	Auto-generated
`name`	String	Name of the category	Default: ""
`image`	String	URL of the category image	Default: ""
`createdAt`	Date	Timestamp when created	Auto-generated
`updatedAt`	Date	Timestamp when updated	Auto-generated

4. Address Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the address	Auto-generated
`address_line`	String	Street address	Default: ""
`city`	String	City name	Default: ""
`state`	String	State name	Default: ""
`pincode`	String	Postal code	Required
`country`	String	Country name	Required
`mobile`	Number	Contact number	Default: null
`status`	Boolean	Address status (Active/Inactive)	Default: true
`userId`	ObjectId	Reference to the `User` collection	Required
`createdAt`	Date	Timestamp when created	Auto-generated
`updatedAt`	Date	Timestamp when updated	Auto-generated

5. Product Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the product	Auto-generated
`name`	String	Product name	Required
`coverimage`	String	Cover image URL	Default: ""
`image`	Array	Array of image URLs	Default: []
`category`	ObjectId	Reference to the `Category` collection	Required
`description`	String	Product description	Default: ""
`more_details`	Object	Additional product details	Default: {}
`weightVariants`	Array	List of weight variants for the product	Embedded document
`publish`	Boolean	Product visibility status (Published/Unpublished)	Default: true
`sku_code`	String	Unique stock-keeping unit (SKU) code	Required
`createdAt`	Date	Timestamp when created	Auto-generated
`updatedAt`	Date	Timestamp when updated	Auto-generated

6. Cart Product Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for the cart product item	Auto-generated
`productId`	ObjectId	Reference to the `Product` collection	Required
`quantity`	Number	Quantity of the product in the cart	Default: 1
`userId`	ObjectId	Reference to the `User` collection	Required
`createdAt`	Date	Timestamp when the cart item was added	Auto-generated
`updatedAt`	Date	Timestamp when the cart item was updated	Auto-generated

7. Order Collection

Field Name	Data Type	Description	Constraints & Default
`_id`	ObjectId	Unique identifier for each order	Auto-generated
`userId`	ObjectId (Ref: User)	Reference to the user placing the order	Required
`orderId`	String	Unique order identification number	Required, Unique
`products`	Array (Embedded)	List of products in the order	Required
`paymentId`	String	Payment transaction ID	Required
`payment_status`	String (Enum)	Payment method	Required, Enum: CASH ON DELIVERY, ONLINE PAYMENT
`cod_status`	String (Enum)	Status of cash-on-delivery	Enum: NOT COMPLETED, PENDING, COMPLETED
			Default: NOT COMPLETED
`isPaymentDone`	Boolean	Indicates if payment has been successfully processed	Default: false
`delivery_address`	ObjectId (Ref: Address)	Reference to the delivery address	Required
`promo_code`	String	Applied promo code (if any)	Default: null
`delivery_charges`	Number	Shipping or delivery charges	Default: 0

`special_Gift_packing`	Number	Additional charges for gift packaging	Default: 0
`invoice_receipt`	String	Path to the invoice receipt file	Optional
`deliveryPartnerId`	ObjectId (Ref: Admin)	Reference to the assigned delivery partner	Default: null
`finalOrderTotal`	Number	Total order amount after discounts and charges	Required
`orderStatus`	String	Order processing status	Default: Not Assigned
`orderAssignedDatetime`	Date	Date & time when the order was assigned	Default: null
`orderDeliveredDatetime`	Date	Date & time when the order was delivered	Default: null
`createdAt`	Date	Timestamp when the order was created	Auto-generated
`updatedAt`	Date	Timestamp when the order was last updated	Auto-updated