Application Design Document

Project Title:
Supply Chain Management

Developer Name : Meghavath Mahender
Contact Information: meghavath.mahender21b@iiitg.ac.in
mahendermeghavath@gmail.com

March 22, 2025

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to serve as a detailed roadmap for the development of the Supply Chain Management System, leveraging the power of modern web technologies and microservices architecture. We will use the Spiral Model for this project. Beyond a mere listing of requirements, it comprehensively outlines both functional and non-functional aspects, design constraints, and other vital considerations, ensuring a methodical and successful development journey. This application aims to streamline supply chain operations by integrating order management, transportation management, and distribution management systems, providing real-time inventory tracking, route optimization, and automated freight billing capabilities.

1.2 Document Conventions

In maintaining top-notch development standards, we adhere to industry best practices, adopting API-based integration, modular design, and cloud-based deployment for scalability. For backend services, we follow RESTful API design principles and utilize standardized documentation with OpenAPI/Swagger. Frontend components adhere to the Airbnb JavaScript Style Guide and implement consistent naming conventions. In-code documentation leverages JSDoc for enhanced clarity, while Entity-Relationship diagrams illustrate database structures, providing visual representations to aid developers in understanding data relationships.

1.3 Intended Audience and Reading Suggestions

Designed for a diverse audience, this document caters to developers, testers, project managers, and stakeholders actively engaged in the Supply Chain Management System project. Developers will find technical specifications to guide implementation, project managers can delve into project scope, milestones, and challenges, while stakeholders gain valuable insights into the application's technical nuances. By catering to varied roles, this document facilitates a collaborative and informed approach to the project.

1.4 Project Scope

The Supply Chain Management System project ambitiously encompasses the development of a comprehensive solution designed to integrate different supply chain functions into a unified system. The scope includes a cloud-based architecture implementing Order Management (OMS), Transportation Management (TMS), and Distribution Management (DMS) modules. Key functionalities include real-time inventory tracking, AI-driven route optimization, automated warehouse operations, and demand forecasting. The system will streamline the entire supply chain process from order processing to final delivery, ensuring maximum operational efficiency and visibility across all stages.

1.5 References

For guidance and best practices, refer to MongoDB Documentation (link), Express.js Documentation (link), React.js Documentation (link), and Node.js Documentation (link). These resources provide invaluable support in harnessing the capabilities of the MERN stack and ensuring the successful implementation of the ’Dil Mange More’ web apxplication.

2 Overall Description

2.1 Product Perspective

The Supply Chain Management System is a cloud-based solution designed to integrate different supply chain functions into a unified system. It operates as a comprehensive platform that interfaces with various external systems such as third-party logistics providers, payment gateways, and shipping carriers. While functioning as a cohesive unit, it maintains distinct modules (OMS, TMS, DMS) that communicate through well-defined APIs, allowing for independent scaling and maintenance. This microservices architecture ensures flexibility while maintaining system integrity.

2.2 Product Functions

The major functions of the Supply Chain Management System include:

- Order Management: Captures and validates customer orders, manages real-time inventory.
- Transportation Management: Optimizes delivery routes, manages carriers, and tracks shipments.
- Distribution Management: Manages warehouse operations, demand forecasting, and returns processing.
- Multi-channel order intake and processing
- Real-time inventory management and allocation
- AI-driven route optimization
- Carrier and freight management
- Automated warehouse operations
- Demand forecasting using historical data
- Comprehensive reporting and analytics

2.3 User Classes and Characteristics

The application caters to several main user classes:

- Customers: Place and track orders.
- Logistics Managers: Manage transportation and distribution operations.
- Warehouse Staff: Handle picking, packing, and inventory management.
- Administrators: Oversee the system and handle configurations.
- Suppliers: Monitor inventory levels and receive purchase orders.
- Carriers: Update shipment status and coordinate deliveries.

Each user class has distinct roles, permissions, and interfaces tailored to their specific needs within the supply chain ecosystem.

2.4 Operating Environment

The Supply Chain Management System is a web-based application accessible from desktops, tablets, and mobile devices. It is hosted on a cloud infrastructure for scalability and high availability. The system is designed to operate across major browsers (Chrome, Firefox, Safari, Edge) and is built with responsive design principles to ensure usability across different screen sizes and devices.

2.5 Design and Implementation Constraints

The Supply Chain Management System employs a microservices architecture with API-driven communication between modules. Key constraints include:

- Adherence to RESTful API design principles
- Implementation of cloud-native technologies
- Containerization using Docker for consistent deployment
- Service orchestration through Kubernetes
- Automated testing and CI/CD pipelines
- Compliance with data protection regulations
- Performance requirements for real-time operations

These constraints guide development decisions and foster a system characterized by modularity, scalability, and maintainability.

2.6 User Documentation

A comprehensive user documentation package will be developed to enhance user experience. This will include:
- Detailed user manuals for each module
- Interactive tutorials and walkthroughs
- API documentation for integration partners
- Administrator guides for system configuration
- Training materials for all user classes

The documentation will offer clear instructions for users engaging with the Supply Chain Management System, empowering them to effectively utilize all features while minimizing the learning curve.

2.7 Assumptions and Dependencies

Integral to the Supply Chain Management System project are several key assumptions:
- Stable internet connectivity for real-time data exchange
- Integration with third-party logistics providers and cloud APIs
- Availability of accurate geolocation services for route optimization
- Compatibility with existing enterprise systems
- Access to historical data for demand forecasting algorithms
- Reliable cloud infrastructure for system hosting

These assumptions play a pivotal role in shaping the project's implementation and overall functionality.

3 External Interface Requirements

3.1 User Interfaces

- **Login Page:** Implemented using React with secure JWT-based authentication. Integration of OAuth for social logins (Google, Facebook) using Passport.js.
- **Registration Form:** React-based form with extensive input validation using Formik and Yup. User data is stored in MongoDB with Mongoose for schema validation.
- **Product Page (Menu Display):** Utilizes React for dynamic rendering. Axios is used for API calls to Node.js backend for real-time menu updates.
- **Shopping Cart:** Interactive React components with Redux for global state management of cart items. Optimized for responsiveness and user-friendly interactions.
- **Order Confirmation:** Node.js with the pdfkit library for server-side PDF generation of order summaries. Nodemailer is used for automated email dispatches.

3.2 Hardware Interfaces

- **Internet Connectivity:** Designed for compatibility with diverse internet-connected devices including laptops, smartphones, and tablets. Optimized for varying network speeds.
- **Peripheral Support:** Facilitates connection with peripherals like printers and POS systems for physical receipt generation, using browser-based APIs.

3.3 Software Interfaces

- **Operating System:** Platform-independent web application, compatible with Windows, macOS, Linux, and mobile operating systems.
- **Browser Support:** Ensures compatibility with major browsers (Chrome, Firefox, Safari, Edge) using responsive design principles and polyfills for cross-browser support.
- **Third-party APIs:** Integration with Stripe for payment processing, Google Maps API for location services, and Twilio for SMS notifications.

4 System Features

4.1 Order Management System (OMS)

- **Description:** Comprehensive order intake, processing, and fulfillment system.
- **Functional Requirements:** Multi-channel order intake (web, mobile, API)
- - Real-time inventory availability checking
- - Order validation and prioritization
- - Customer-specific pricing and discounts
- - Backorder management and automated notifications
- - Order status tracking and updates
- - Integration with payment processing systems.

4.2 Transportation Management System (TMS)

- **Description:** Advanced transportation planning, execution, and monitoring system.
- **Functional Requirements:** AI-driven route optimization considering multiple constraints
- - Carrier selection and rate management
- - Load planning and consolidation
- - Real-time shipment tracking and ETAs
- - Freight audit and payment automation
- - Performance analytics and carrier scorecards
- - Exception management and automated alerts.

4.3 Distribution Management System (DMS)

- **Description:** Comprehensive warehouse and inventory management solution.
- **Functional Requirements:** - Warehouse layout optimization
- - Picking and packing optimization
- - Inventory allocation and replenishment
- - Returns processing and management
- - Cycle counting and inventory reconciliation
- - Cross-docking capabilities
- - Labor management and productivity tracking.

4.4 Inventory Management

- **Description:** Real-time inventory tracking and optimization system.
- **Functional Requirements:** Multi-location inventory visibility
- - Safety stock calculation and management
- - Automated reordering based on demand forecasts
- - Lot tracking and expiration date management
- - Serial number tracking for high-value items
- - Inventory valuation and reporting
- - ABC analysis and inventory classification

4.5 Reporting and Analytics

- **Description:** Comprehensive business intelligence and analytics platform.
- **Functional Requirements:** Real-time dashboards with customizable KPIs

- - Historical performance analysis
- - Predictive analytics for demand forecasting
- - Scenario planning and simulation
- - Cost analysis and profitability reporting
- - Export capabilities in multiple formats

5 Nonfunctional Requirements

5.1 Performance Requirements

The system's performance is crucial to ensure seamless operations across the supply chain. The following criteria should be met:

- **Response Time:** The system must respond to user inputs within few seconds. Achieving low latency in API responses is essential for a responsive user interface. This involves optimizing backend processes, such as database queries and server-side computations, to meet the specified time constraint.
- **Throughput:** The system should scale to real time traffic. This includes both read and write operations on the database, ensuring efficient data processing. To achieve this, implement efficient data retrieval and storage mechanisms, consider indexing database fields, and optimize algorithms for data manipulation.

5.2 Safety Requirements

Ensuring the safety of user data and system operations is paramount. The following safety requirements should be met:

- **Data Redundancy:** Implementation of failover mechanisms and regular backups to prevent data loss.
- **Error Handling:** The system should provide clear error messages to users in case of failures to avoid confusion. Implementing comprehensive error logging and monitoring will aid in identifying and resolving issues promptly. Use tools for real-time error tracking and implement user-friendly error messages to assist users in troubleshooting.
- **Validation Checks:** Input validation at all levels to prevent processing of invalid or potentially harmful data.
- **Audit Trails:** Complete logging of all system activities, especially those involving critical data modifications.

5.3 Security Requirements

Security is of utmost importance to safeguard the system against potential threats. Key security requirements include:

- **Authentication:** Multi-factor authentication for all users with role-based access control.
- **Data Encryption:** AES-256 encryption for data at rest and TLS for data in transit.
- **API Security**: OAuth 2.0 and JWT for secure API authentication and authorization**.**
- **Secure Development**: Implementation of secure coding practices and regular code reviews.
- **Monitoring:** Real-time monitoring for suspicious activities and automated alerts for potential security
-

5.4 Software Quality Attributes

Software quality attributes define the characteristics of the system that contribute to its understanding, maintainability, and overall quality. Key quality attributes of the system should include:

- **Maintainability:** The system should be designed to facilitate updates and maintenance. This involves using clean, modular code with comprehensive documentation to ensure that future developers can easily understand and modify the system.

- **Scalability:** The application should be able to handle an increasing number of users, transactions, and data without a degradation in performance. This involves using scalable architecture patterns and technologies.
- **Reliability:** The system should operate consistently under defined conditions, and in case of a failure, it should fail gracefully with minimal impact on the user experience.
- **Usability:** The user interface should be intuitive and user-friendly, allowing users to perform their tasks efficiently and without unnecessary complexity.
- **Testability:** The system should be designed to facilitate easy testing of all components to ensure functionality, performance, and security.
- **Portability:** The system should be adaptable to different environments without significant changes in the software.
- **Interoperability:** The system should be capable of interacting with other systems and exchange data in a seamless manner.

6 Decision Applications

6.1 Decision Tree

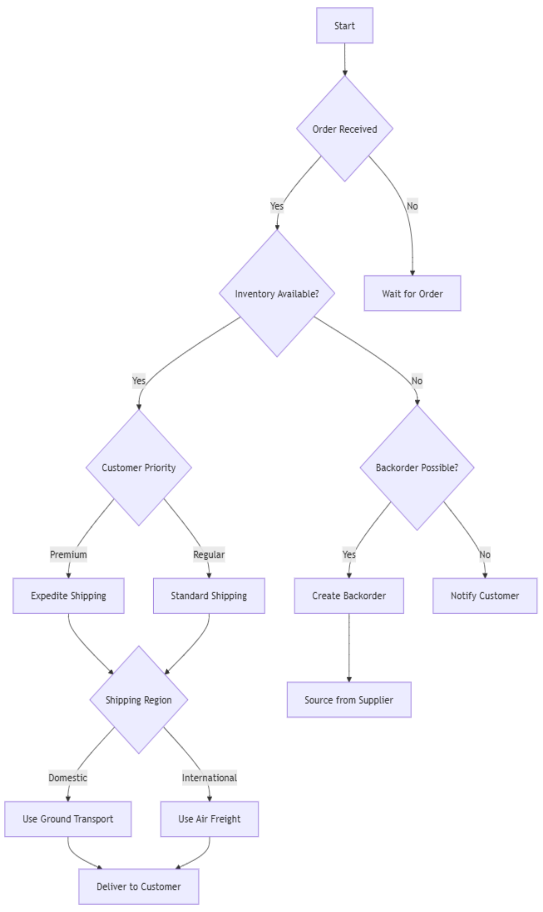

Figure 1: Decision Tree

7 DFD

7.1 Data Flow diagram

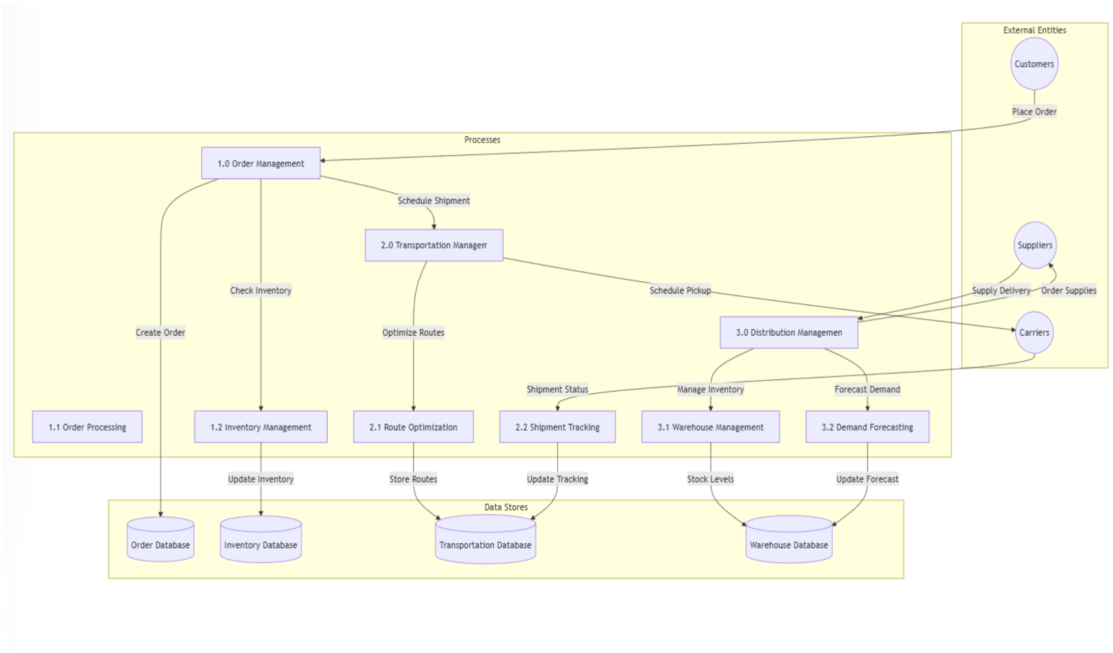

Figure 2: Data Flow Diagram

8 Structure Chart

8.1 Overview of System Architecture

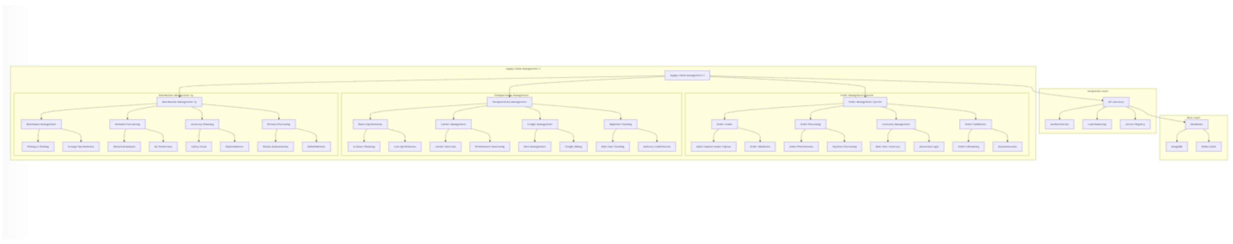

Figure 5: Structure Chart illustrating the system architecture.

9 Unified Modeling Language

9.1 UML Diagram

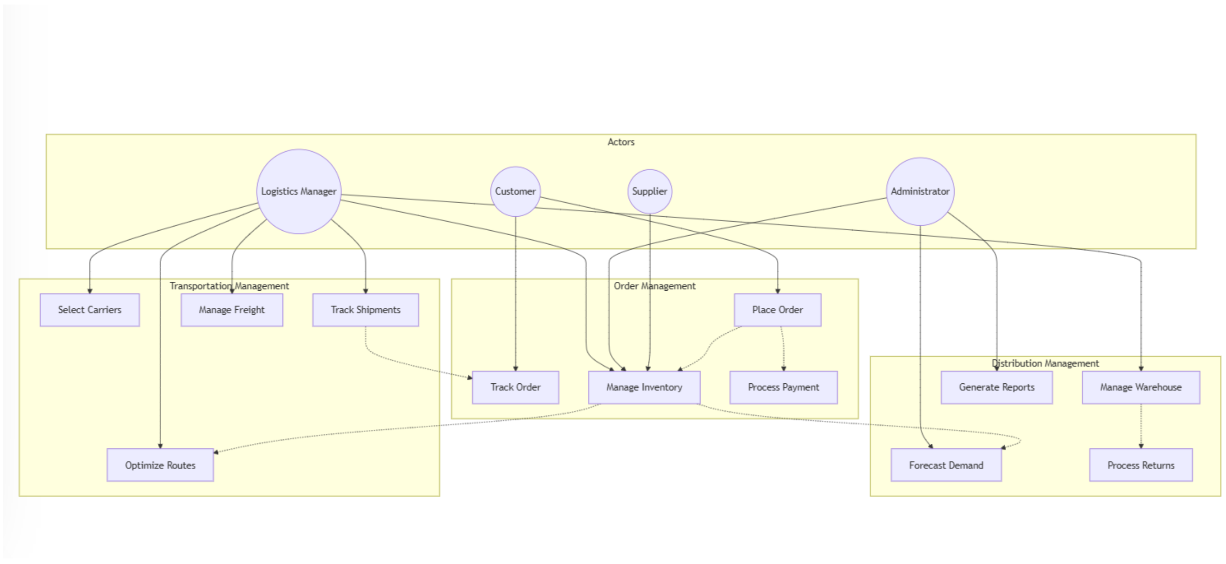

Figure 6: UML Diagram illustrating the Use Cases

10 Unified Modeling Language Sequence Diagram

10.1 UML Diagram

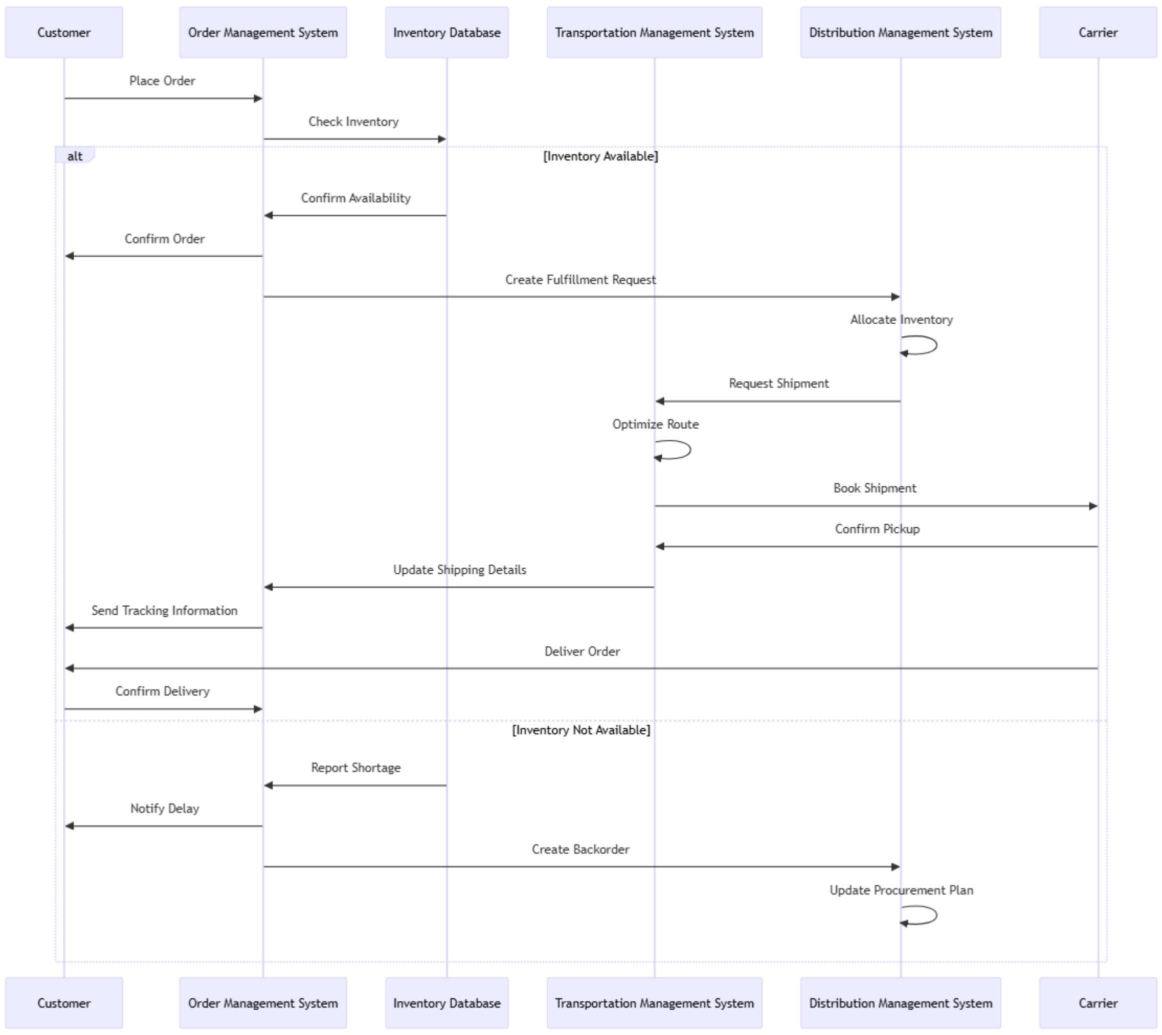

Figure 7: UML Diagram illustrating the Use Cases

11 UML Collboration diagram

11.1 UML Collboration Diagram

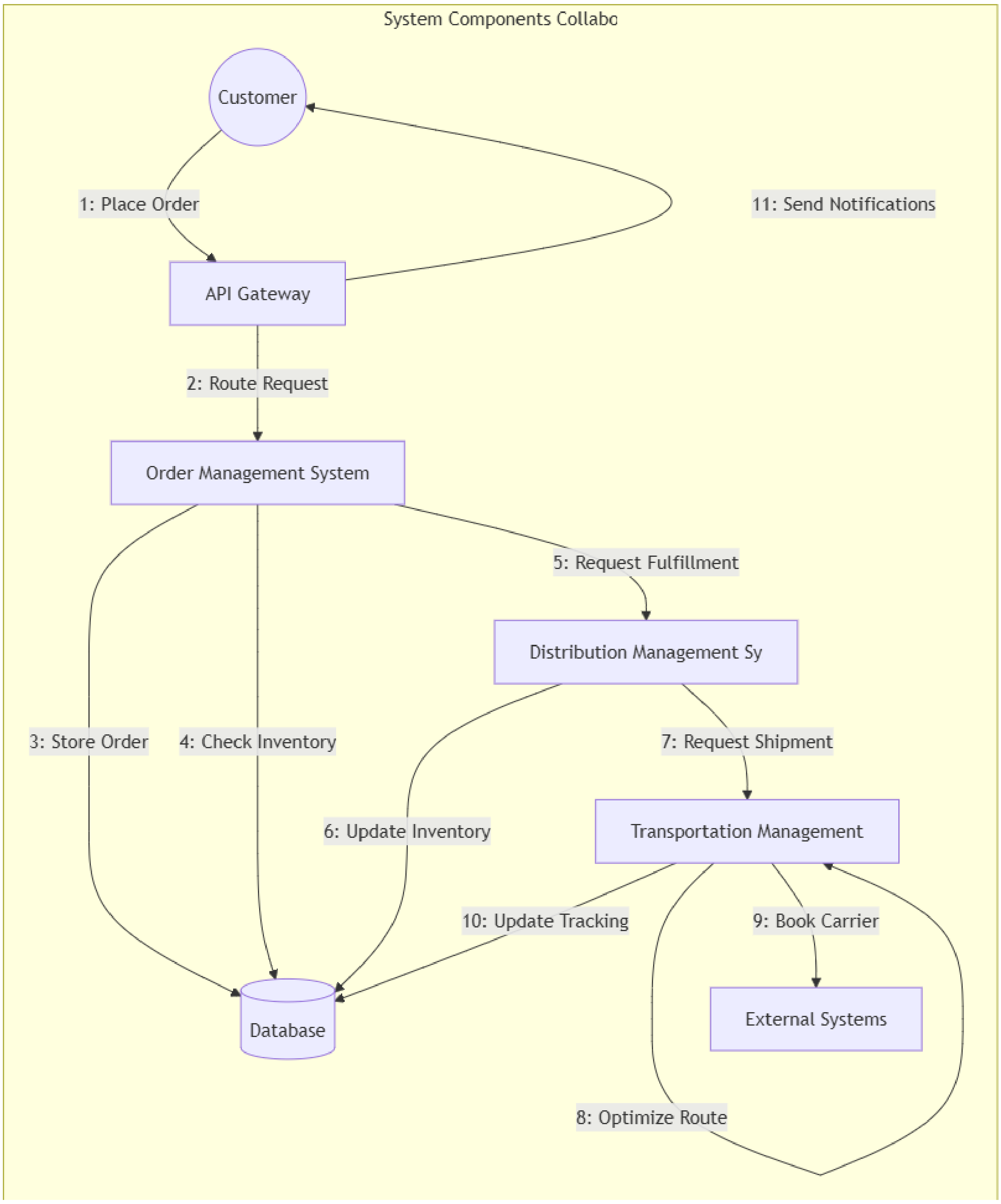
System Components Collabo
Customer
1: Place Order
11: Send Notifications
API Gateway
2: Route Request
Order Management System
5: Request Fulfillment
Distribution Management Sy
3: Store Order
4: Check Inventory
7: Request Shipment
6: Update Inventory
Transportation Management
10: Update Tracking
9: Book Carrier
Database
External Systems
8: Optimize Route

Figure 8: UML Diagram illustrating the Use Cases

12 ER Diagram

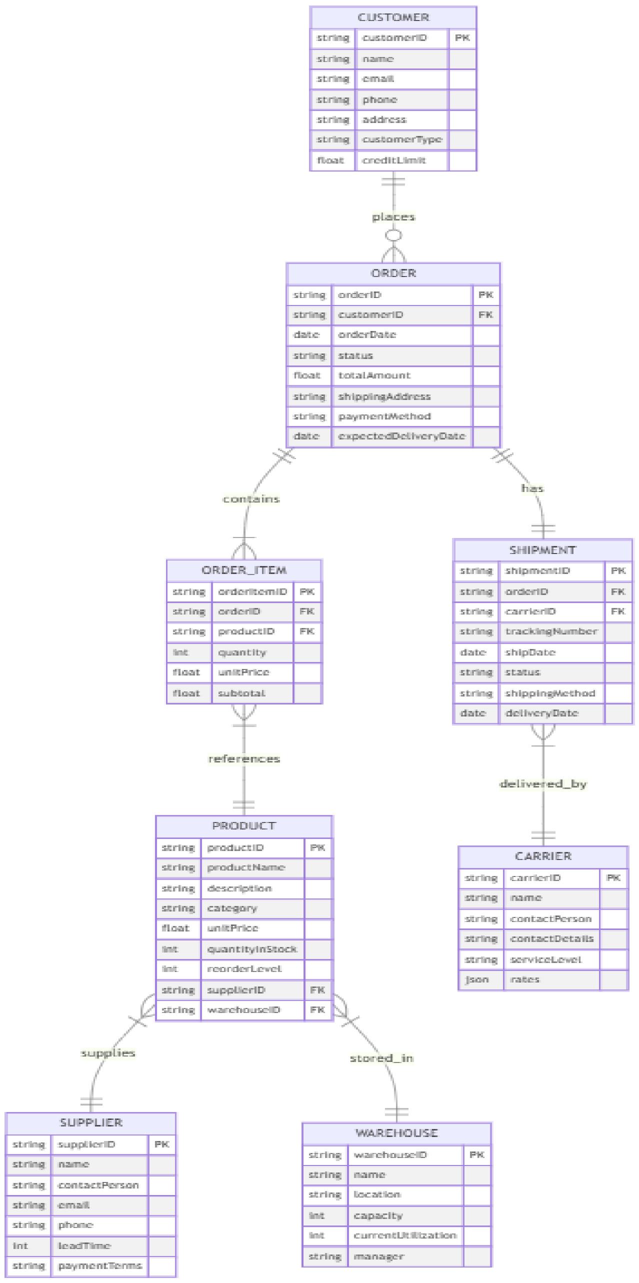

Figure 9: ER Diagram

13 Technology Stack : MERN Stack (MongoDB, Express.js, React.js, Node.js).

14 Timeline and Milestones

14.1 The project is planned to be completed within 25 days(upto April 21), covering:

. Requirements Gathering and Analysis

. Design and Prototyping

. Development and Unit Testing

. Integration and System Testing

. Documentation

15 Scalability and Performance Considerations

.Scalability: Horizontal scaling, optimized database queries, load balancing and caching.

.Performance: Asynchronous processing, static file compression, and real-time monitoring with auto-scaling.

16 Assumptions and Constraints

Stable internet and reliable APIs assumed; constrained to MERN stack for development.

17 Approval Request to Proceed with Coding

Respected sir,

I seek your approval to proceed with the coding phase of the " Supply Chain Management " project. All necessary preparations are complete, and I am ready to begin coding. Kindly confirm if I can move forward or if additional steps are required.

Thank you for your consideration.

Best regards,
Meghavath Mahender

meghavath.mahender21b@iiitg.ac.in,

mahendermeghavath@gmail.com,