

# Title: INVENTORY MANAGEMENT SYSTEM

**A CAPSTONE PROJECT REPORT**

# Submitted by

**K. MOHAN SRI SAI (192210467)**

**CSE Department.**

In Partial Fulfillment for the completion of the course

CSA0912-PROGRAMMING IN JAVA FOR ACCESSING DATABASE

# SEPTEMBER -2024



SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

CHENNAI - 602105 TAMIL NADU, INDIA



### BONAFIDE CERTIFICATE

This is to certify that the project report entitled **Inventory Management System** submitted by **K. MOHAN SRI SAI (192210467)** to Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institution and in my appraisal meets the required standards for submission.

Dr.K.Jayasakthi Velmurugan

COURSE FACULTY

*Department of Deep Learning. Saveetha School of Engineering, SIMATS, Chennai - 602105*

**ACKNOWLEDGEMENT**

This project work would not have been possible without the contribution of many people. It gives me immense pleasure to express my profound gratitude to our Honorable Chancellor **Dr. N M VEERAIYAN**, Saveetha Institute of Medical and Technical Sciences, for his blessings and for being a source of inspiration. I sincerely thank our Director of Academics **Dr. DEEPAK NALLASWAMY,** SIMATS, for his visionary thoughts and support. I am indebted to extend my gratitude to our Director **Dr. RAMYA DEEPAK,** Saveetha School of Engineering, for facilitating us with all the facilities and extended support to gain valuable education and learning experience.

I register my special thanks to **Dr. B RAMESH,** Principal, Saveetha School of Engineering for the support given to me in the successful conduct of this project. I wish to express my sincere gratitude to my Course faculty **Dr.K.Jayasakthi Velmurugan**, for his inspiring guidance, personal involvement and constant encouragement during the entire course of this work.

I am grateful to Project Coordinators, Review Panel External and Internal Members and the entire faculty of the Department of Design, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

# INDEX

|  |  |
| --- | --- |
| ABSTRACT | 4 |
| INTRODUCTION | 5 |
| ARCHITECTURE DIAGRAM | 6 |
| FLOWCHART | 7 |
| UML DIAGRAM | 8 |
| CLASS DIAGRAM | 8 |
| CODE IMPLEMENTATION | 9-14 |
| OUTPUT SCREENSHOT | 14-15 |
| CONCLUSION | 16 |
| REFERENCES | 17 |

1. **ABSTRACT**

The Inventory Management System (IMS) is a software application designed to facilitate the effective management of inventory within businesses. It provides real-time tracking of stock levels, supports order management, and enhances decision-making through accurate data on stock movements. The system allows users to perform functions like adding new stock, updating quantities, generating reports, and managing supplier relationships.

By automating inventory processes, the IMS reduces manual errors, improves accuracy, and minimizes the risk of overstock or understock situations. The system supports integration with point-of-sale systems (POS), procurement, and accounting modules, ensuring seamless business operations.

Additionally, the IMS features user authentication, role-based access control, and auditing tools to secure and manage data. With a user-friendly interface and customizable reporting tools, the IMS is suitable for small to large enterprises, optimizing inventory flow, enhancing productivity, and reducing operational costs.

In addition to operational efficiency, the IMS enhances decision-making by providing actionable insights through data analytics. Historical sales data, stock movement trends, and supplier performance reports are made available to help businesses forecast demand, optimize purchasing strategies, and reduce holding costs. The system can be integrated with other enterprise resource planning (ERP) tools, ensuring end-to-end supply chain visibility. This comprehensive approach to inventory management helps businesses reduce wastage, improve customer satisfaction, and ultimatel drivehig0herprofitability.

In today's fast-paced digital world, convenience and efficiency are paramount, especially in the entertainment industry. Online movie ticket booking systems are innovative solutions designed to meet the changing needs of movie lovers and cinema operators. As online transactions continue to grow in popularity, this system allows users to easily purchase movie tickets from the comfort of their own home or on the go, eliminating the traditional hassle of in-person bookings.

# INTRODUCTION

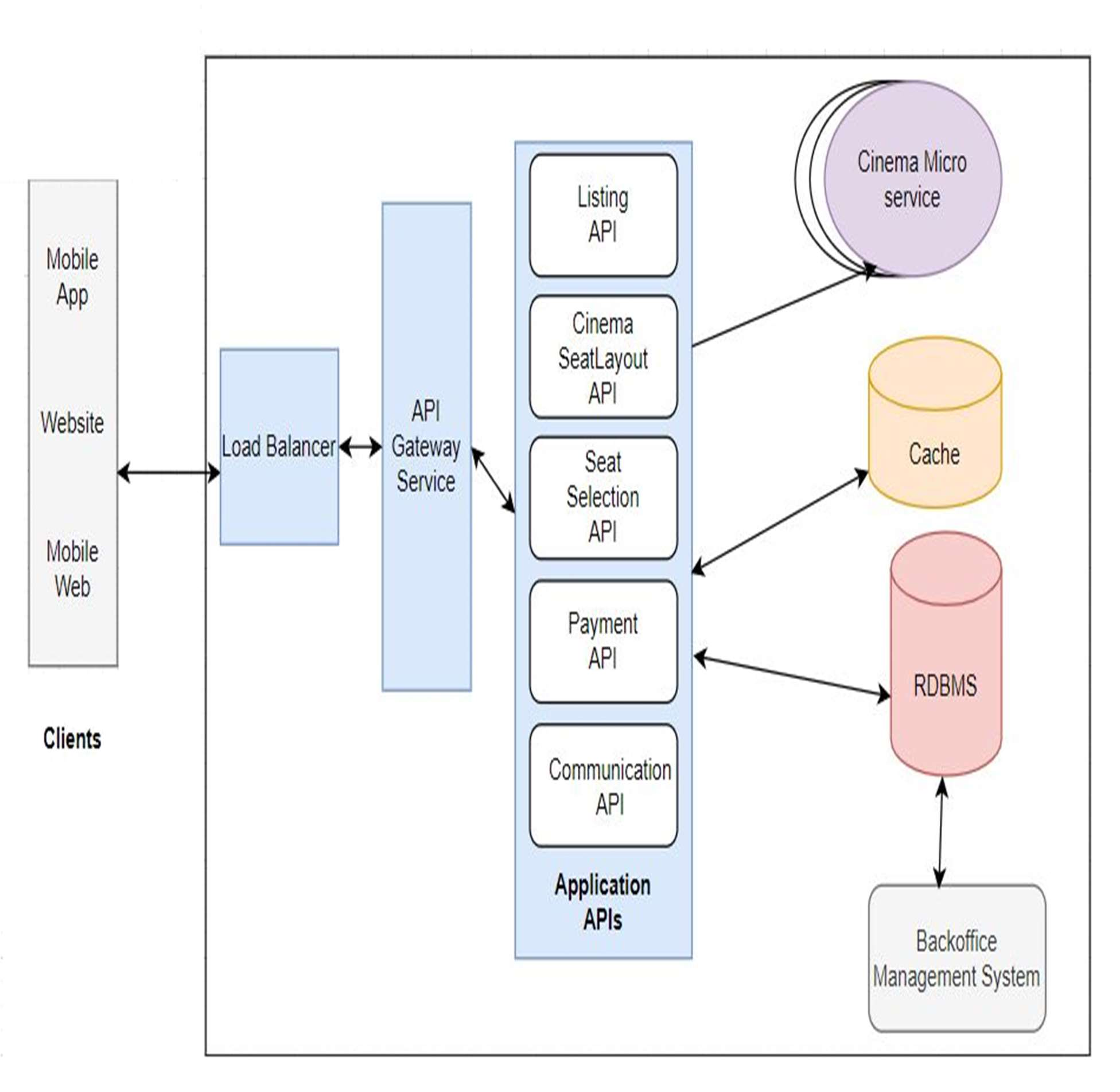
Effective inventory management is a critical component of modern business operations, as it directly impacts profitability, customer satisfaction, and operational efficiency. In many industries—such as retail, manufacturing, and logistics—managing stock levels, tracking product movement, and predicting demand can become highly complex. Traditionally, these tasks were done manually, leading to inaccuracies, inefficiencies, and increased operational costs. To overcome these challenges, businesses are increasingly adopting automated **Inventory Management Systems (IMS)**, which provide real-time data, improve accuracy, and streamline processes.

An Inventory Management System is a digital tool designed to simplify the oversight of inventory levels, orders, sales, and deliveries. It ensures that businesses can maintain optimal stock levels, reducing the risk of both overstocking and stockouts. The system automates key functions like order generation, stock tracking, and reporting, providing businesses with up-to-date insights on stock levels and enabling timely decision-making. By integrating advanced technologies like barcode scanning, RFID, and cloud-based platforms, an IMS can track inventory across multiple locations, warehouses, or retail outlets, ensuring efficient supply chain management.

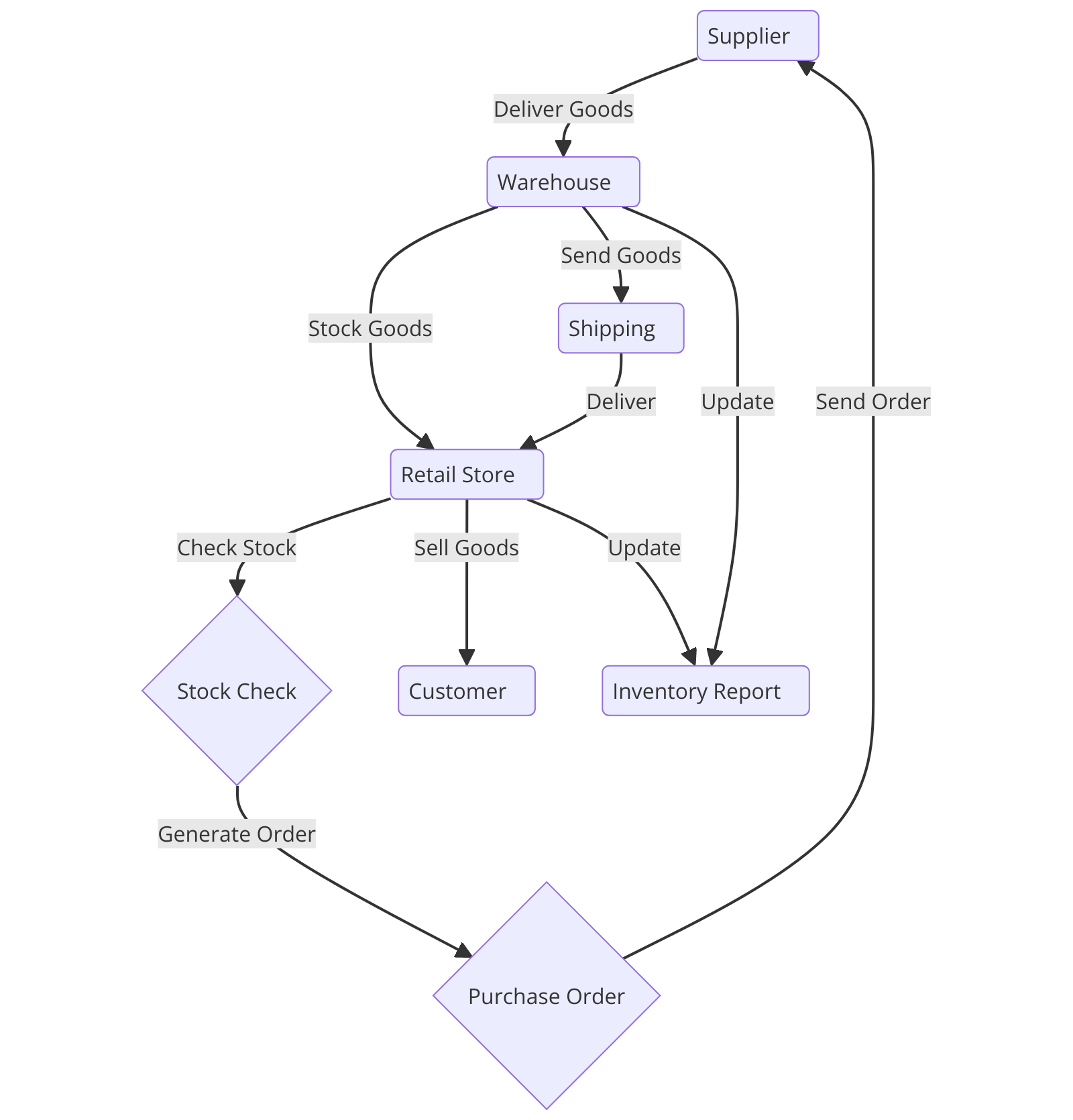
The growing need for greater efficiency, scalability, and cost control has driven many organizations to adopt Inventory Management Systems. These systems not only provide operational benefits but also strategic advantages by offering detailed analytics on stock performance, supplier relationships, and demand forecasting. With the ability to integrate seamlessly with other enterprise solutions such as point-of-sale (POS) systems, procurement software, and accounting tools, IMS plays a crucial role in creating a connected, data-driven business environment.

The system automates key functions like order generation, stock tracking, and reporting, providing businesses with up-to-date insights on stock levels and enabling timely decision-making. By integrating advanced technologies like barcode scanning, RFID, and cloud-based platforms, an IMS can track inventory across multiple locations, warehouses, or retail outlets, ensuring efficient supply chain management.

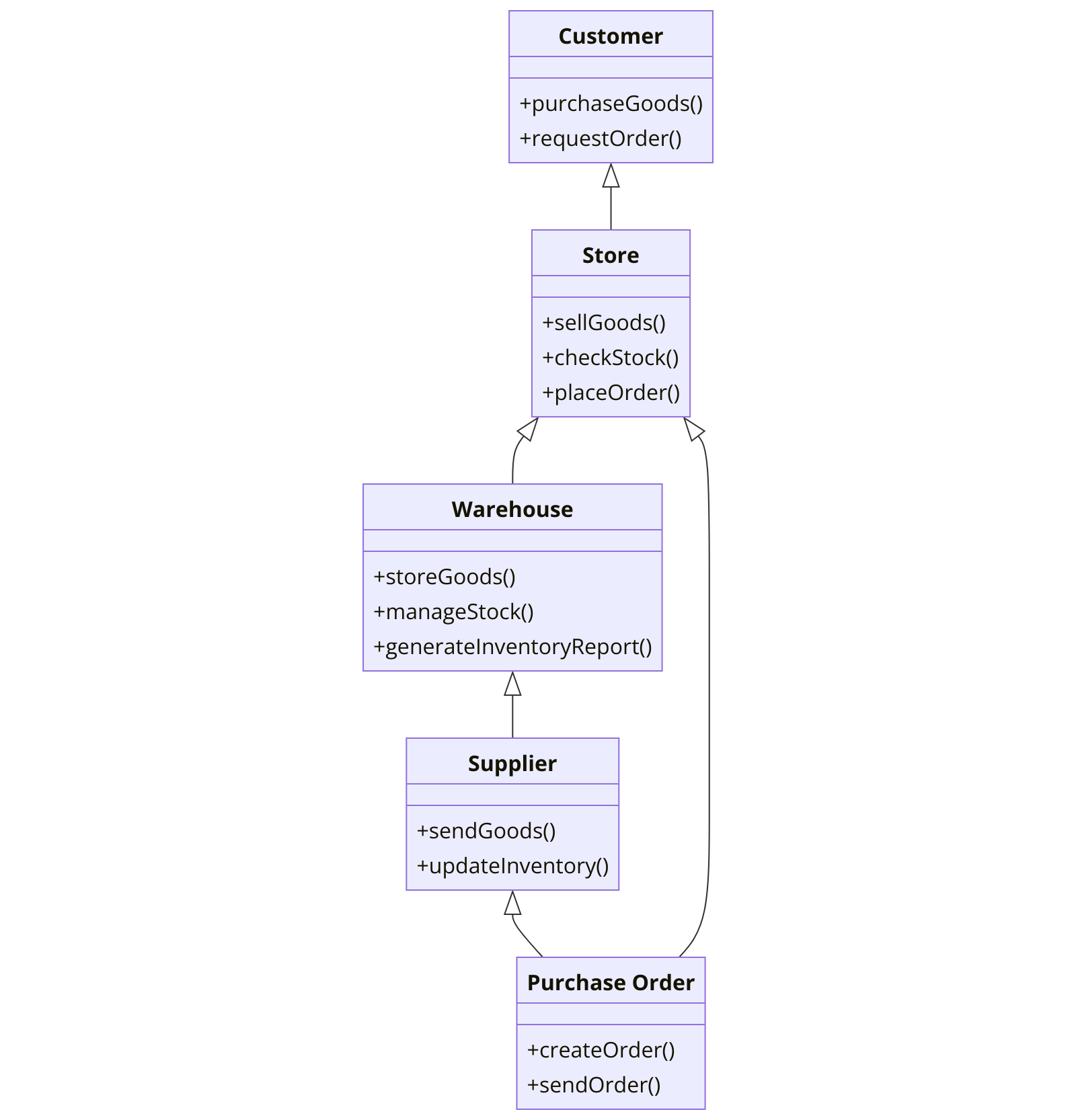
# ARCHITECTURE DIAGRAM



1. **FLOW CHART**

****

1. **CLASS DIAGRAM**



## Code Implementation

// Inventory.java

import java.util.ArrayList;

import java.util.List;

public class Inventory {

private List<Item> items;

private int nextId;

public Inventory() {

items = new ArrayList<>();

nextId = 1; // Starting ID

}

// Add a new item

public void addItem(String name, int quantity, double price) {

Item item = new Item(nextId++, name, quantity, price);

items.add(item);

System.out.println("Item added successfully.");

}

// View all items

public void viewItems() {

if (items.isEmpty()) {

System.out.println("Inventory is empty.");

return;

}

for (Item item : items) {

System.out.println(item);

}

}

// Update an existing item

public boolean updateItem(int id, String name, int quantity, double price) {

for (Item item : items) {

if (item.getId() == id) {

item.setName(name);

item.setQuantity(quantity);

item.setPrice(price);

System.out.println("Item updated successfully.");

return true;

}

}

System.out.println("Item with ID " + id + " not found.");

return false;

}

// Delete an item

public boolean deleteItem(int id) {

for (Item item : items) {

if (item.getId() == id) {

items.remove(item);

System.out.println("Item deleted successfully.");

return true;

}

}

System.out.println("Item with ID " + id + " not found.");

return false;

}

// Search for an item by name

public void searchItem(String name) {

boolean found = false;

for (Item item : items) {

if (item.getName().toLowerCase().contains(name.toLowerCase())) {

System.out.println(item);

found = true;

}

}

if (!found) {

System.out.println("No items found with name containing \"" + name + "\".");

}

}

}

// Main.java

import java.util.Scanner;

public class Main {

private static Inventory inventory = new Inventory();

private static Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

int choice;

do {

showMenu();

choice = getIntInput("Enter your choice: ");

switch (choice) {

case 1:

addItemUI();

break;

case 2:

inventory.viewItems();

break;

case 3:

updateItemUI();

break;

case 4:

deleteItemUI();

break;

case 5:

searchItemUI();

break;

case 6:

System.out.println("Exiting Inventory Management System. Goodbye!");

break;

default:

System.out.println("Invalid choice. Please select between 1-6.");

}

} while (choice != 6);

}

private static void showMenu() {

System.out.println("\n=== Inventory Management System ===");

System.out.println("1. Add Item");

System.out.println("2. View All Items");

System.out.println("3. Update Item");

System.out.println("4. Delete Item");

System.out.println("5. Search Item");

System.out.println("6. Exit");

}

private static void addItemUI() {

System.out.println("\n--- Add New Item ---");

String name = getStringInput("Enter item name: ");

int quantity = getIntInput("Enter quantity: ");

double price = getDoubleInput("Enter price: ");

inventory.addItem(name, quantity, price);

}

private static void updateItemUI() {

System.out.println("\n--- Update Item ---");

int id = getIntInput("Enter item ID to update: ");

String name = getStringInput("Enter new name: ");

int quantity = getIntInput("Enter new quantity: ");

double price = getDoubleInput("Enter new price: ");

inventory.updateItem(id, name, quantity, price);

}

private static void deleteItemUI() {

System.out.println("\n--- Delete Item ---");

int id = getIntInput("Enter item ID to delete: ");

inventory.deleteItem(id);

}

private static void searchItemUI() {

System.out.println("\n--- Search Item ---");

String name = getStringInput("Enter item name to search: ");

inventory.searchItem(name);

}

// Helper methods for input

private static String getStringInput(String prompt) {

System.out.print(prompt);

return scanner.nextLine().trim();

}

private static int getIntInput(String prompt) {

int num;

while (true) {

System.out.print(prompt);

try {

num = Integer.parseInt(scanner.nextLine().trim());

return num;

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid integer.");

}

}

}

private static double getDoubleInput(String prompt) {

double num;

while (true) {

System.out.print(prompt);

try {

num = Double.parseDouble(scanner.nextLine().trim());

return num;

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid number.");

}

}

}

}

1. **OUTPUT SCREENSHOT**

**// Item.java**

**public class Item {**

**private int id;**

**private String name;**

**private int quantity;**

**private double price;**

**// Constructor**

**public Item(int id, String name, int quantity, double price) {**

**this.id = id;**

**this.name = name;**

**this.quantity = quantity;**

**this.price = price;**

**}**

**// Getters and Setters**

**public int getId() {**

**return id;**

**}**

**public String getName() {**

**return name;**

**}**

**public void setName(String name) {**

**this.name = name;**

**}**

**public int getQuantity() {**

**return quantity;**

**}**

**public void setQuantity(int quantity) {**

**this.quantity = quantity;**

**}**

**public double getPrice() {**

**return price;**

**}**

**public void setPrice(double price) {**

**this.price = price;**

**}**

**// To display item details**

**@Override**

**public String toString() {**

**return String.format("ID: %d | Name: %s | Quantity: %d | Price: %.2f",**

**id, name, quantity, price);**

**}**

**}**

**// Inventory.java**

**import java.util.ArrayList;**

**import java.util.List;**

**public class Inventory {**

**private List<Item> items;**

**private int nextId;**

**public Inventory() {**

**items = new ArrayList<>();**

**nextId = 1; // Starting ID**

**}**

**// Add a new item**

**public void addItem(String name, int quantity, double price) {**

**Item item = new Item(nextId++, name, quantity, price);**

**items.add(item);**

**System.out.println("Item added successfully.");**

**}**

**// View all items**

**public void viewItems() {**

**if (items.isEmpty()) {**

**System.out.println("Inventory is empty.");**

**return;**

**}**

**for (Item item : items) {**

**System.out.println(item);**

**}**

**}**

**// Update an existing item**

**public boolean updateItem(int id, String name, int quantity, double price) {**

**for (Item item : items) {**

**if (item.getId() == id) {**

**item.setName(name);**

**item.setQuantity(quantity);**

**item.setPrice(price);**

**System.out.println("Item updated successfully.");**

**return true;**

**}**

**}**

**System.out.println("Item with ID " + id + " not found.");**

**return false;**

**}**

**// Delete an item**

**public boolean deleteItem(int id) {**

**for (Item item : items) {**

**if (item.getId() == id) {**

**items.remove(item);**

**System.out.println("Item deleted successfully.");**

**return true;**

**}**

**}**

**System.out.println("Item with ID " + id + " not found.");**

**return false;**

**}**

**// Search for an item by name**

**public void searchItem(String name) {**

**boolean found = false;**

**for (Item item : items) {**

**if (item.getName().toLowerCase().contains(name.toLowerCase())) {**

**System.out.println(item);**

**found = true;**

**}**

**}**

**if (!found) {**

**System.out.println("No items found with name containing \"" + name + "\".");**

**}**

**}**

# CONCLUSION

In conclusion, this simple Inventory Management System provides a solid foundation for understanding key Java programming concepts like object-oriented design, data manipulation, and user interaction. It allows users to perform basic operations such as adding, viewing, updating, deleting, and searching for items in an inventory. While the current implementation is console-based and stores data in memory, it can be extended to include more advanced features such as data persistence, graphical user interfaces, reporting, and user authentication. By further developing this system, you can gain deeper insights into building scalable and practical applications in Java.

For administrators, the system simplifies the process of managing movie listings, tracking reservations, and generating real-time reports, providing valuable insights into customer behavior and performance. Mobile responsiveness, cross-device compatibility, and robust backend infrastructure ensure that users can access your services anytime, anywhere.

Moreover, this project is highly modular and easily scalable. Each component, such as the Item class and the Inventory management system, is designed to be reusable and adaptable. This allows future enhancements like integrating a database for persistent storage, adding new fields to the items, or even implementing inventory tracking across multiple locations. Additionally, incorporating exception handling and improving the user experience with validation checks ensures the system's robustness. As you continue to build on this foundation, you’ll also improve your understanding of software architecture and development best practices in Java.

### REFERENCES

* 1. GitHub Repositories: Explore various open-source projects related to movie ticket booking systems on GitHub. These repositories often include code, documentation, and implementation details1.
  2. GeeksforGeeks: This article provides a comprehensive guide on designing a movie ticket booking system, including requirements, architecture, and implementation details2.
  3. Filmgrail Blog: Learn about the essentials of an online cinema booking system, including features like browsing showtimes, seat selection, and payment processing3.
  4. BookMyShow: One of the most popular platforms for booking movie tickets in India. It offers a user-friendly interface and integrates well with mobile apps4.
  5. Research Papers: Look for academic papers on digital ticketing systems and their impact on user experience and business operations.
  6. API Documentation: Check out APIs like TMDB (The Movie Database) for integrating movie data into your booking system.
  7. Payment Gateway Integration: Learn about integrating payment gateways like PayPal, Stripe, or Razorpay for secure transactions.
  8. User Authentication: Implement secure user authentication using OAuth or JWT (JSON Web Tokens).
  9. Real-time Updates: Use WebSockets or similar technologies to provide real-time updates on seat availability.
  10. UI/UX Design: Focus on creating an intuitive and responsive user interface, possibly using frameworks like React or Angular.
  11. Database Management: Understand how to manage and optimize databases for storing movie schedules, user information, and booking details.
  12. Security Best Practices: Learn about securing your application against common threats like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
  13. Scalability: Explore techniques for scaling your application to handle high traffic, such as load balancing and microservices architecture.
  14. Performance Optimization: Study methods to optimize the performance of your application, including caching strategies and efficient query handling.