Invoice Management System Documentation

1. Introduction

The Invoice Management System is a Spring Boot–based application developed to manage invoices efficiently using CRUD (Create, Read, Update, Delete) operations. It helps organizations handle customers, products, invoices, and invoice items in a structured and automated way.

This system is designed using a layered architecture with RESTful APIs and follows best practices such as separation of concerns, exception handling, and database persistence using Spring Data JPA.

2. Objectives

- To create and manage invoices digitally
- To perform CRUD operations on customers, products, and invoices
- To reduce manual errors in billing
- To provide a scalable and maintainable backend system

3. Technologies Used

- Java 17+
- Spring Boot
- Spring Data JPA
- Hibernate
- RESTful Web Services
- MySQL / H2 Database
- Maven
- Postman (for API testing)

4. System Architecture

The application follows a layered architecture:

- Controller Layer – Handles HTTP requests and responses
- Service Layer – Contains business logic
- Repository Layer – Handles database operations
- Entity Layer – Represents database tables
- Exception Handling – Manages errors globally

5. Modules

5.1 Customer Module

Manages customer-related information.

Attributes: - customerId - customerName - email - phoneNumber - address

Operations: - Add new customer - View all customers - View customer by ID - Update customer details - Delete customer

5.2 Product Module

Manages product details used in invoices.

Attributes: - productId - productName - price - description

Operations: - Add new product - View all products - View product by ID - Update product - Delete product

5.3 Invoice Module

Handles invoice generation and management.

Attributes: - invoiceId - invoiceDate - customer - totalAmount

Operations: - Create invoice - View all invoices - View invoice by ID - Update invoice - Delete invoice

5.4 Invoice Item Module

Links products to invoices.

Attributes: - invoiceItemId - invoice - product - quantity - price

Operations: - Add items to invoice - Update invoice items - Remove invoice items

6. CRUD Operations Explanation

Create

Used to insert new records into the database using HTTP POST requests.

Read

Used to fetch data using HTTP GET requests.

Update

Used to modify existing data using HTTP PUT requests.

Delete

Used to remove data using HTTP DELETE requests.

7. REST API Endpoints (Sample)

Customer APIs

- POST /api/customers
- GET /api/customers
- GET /api/customers/{id}
- PUT /api/customers/{id}
- DELETE /api/customers/{id}

Product APIs

- POST /api/products
- GET /api/products
- GET /api/products/{id}
- PUT /api/products/{id}
- DELETE /api/products/{id}

Invoice APIs

- POST /api/invoices
- GET /api/invoices
- GET /api/invoices/{id}
- DELETE /api/invoices/{id}

8. Exception Handling

Global exception handling is implemented using @ControllerAdvice to handle: - ResourceNotFoundException - InvalidInputException - General exceptions

This improves error handling and user-friendly responses.

9. Database Design

Tables: - Customer - Product - Invoice - Invoice_Item

Relationships: - One Customer Many Invoices - One Invoice Many Invoice Items - One Product Many Invoice Items

10. Advantages

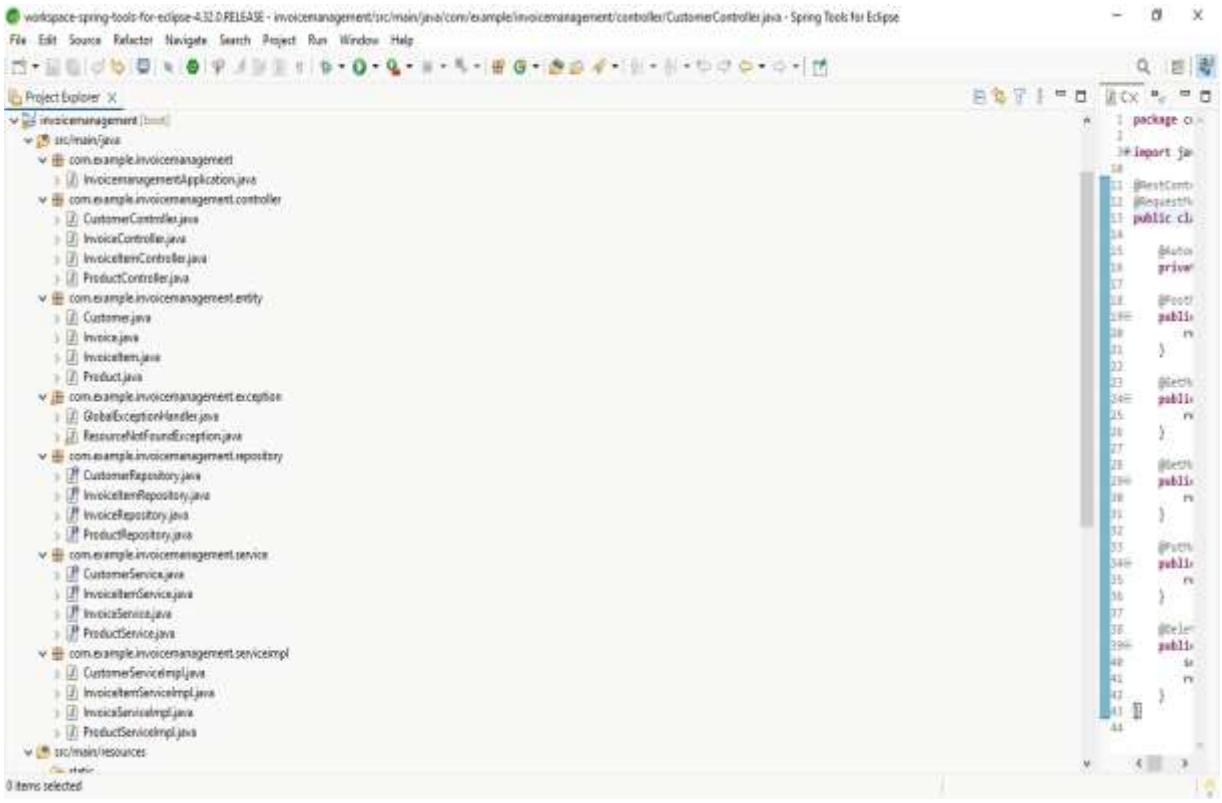
- Easy invoice tracking
- Reduced manual work
- Scalable architecture
- REST API–based integration

11. Future Enhancements

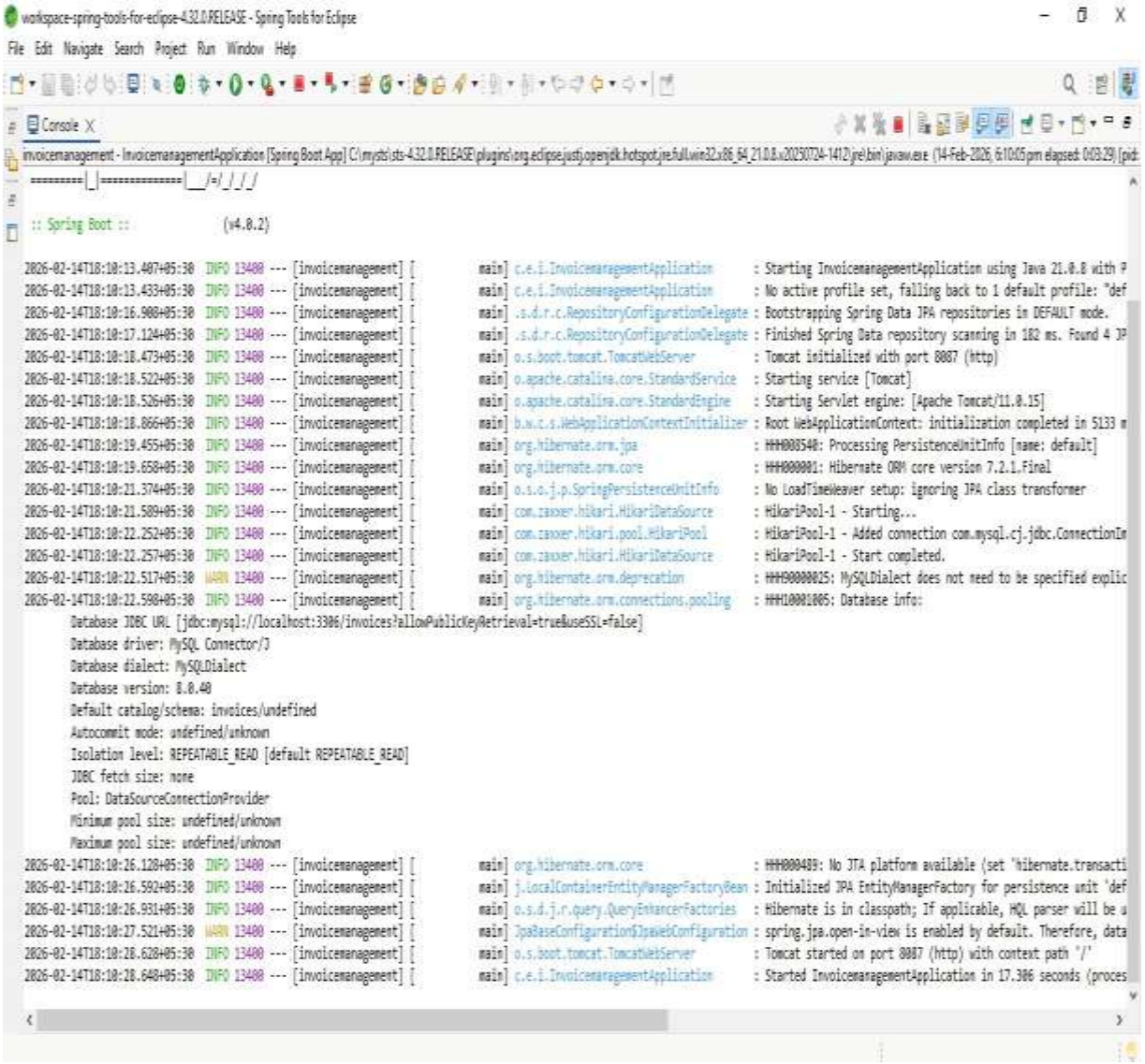
- Authentication and authorization (Spring Security)
- PDF invoice generation
- Payment gateway integration
- Reporting and analytics

12.Screenshots

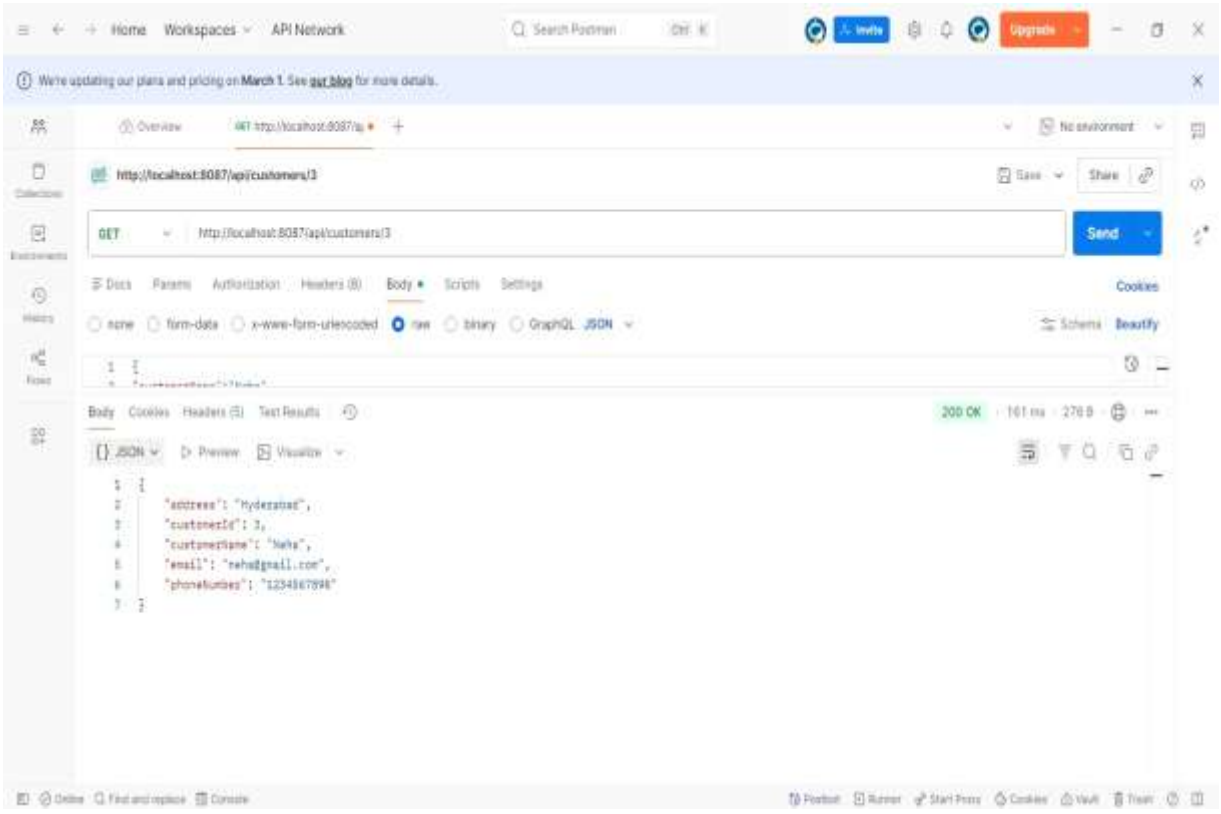
The screenshot shows the structured package architecture of the Invoice Management System developed using Spring Tool Suite



The screenshot shows the Spring Boot application console confirming successful server startup and database connection.



The screenshot shows the successful running of RESTful web services in the Spring Boot application.



12. Conclusion

The Invoice Management System using Spring Boot CRUD operations provides an efficient and reliable solution for managing invoices. It demonstrates the practical implementation of Spring Boot, REST APIs, and database integration, making it suitable for real-world enterprise applications and interviews.