

# 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 displays the Spring Tool Suite interface. On the left, the Project Explorer shows a hierarchical view of the project structure under the 'invoiceManagement' package. The structure includes: src/main/java (containing com.example.invoiceManagement, com.example.invoiceManagement.controller, com.example.invoiceManagement.entity, com.example.invoiceManagement.exception, com.example.invoiceManagement.repository, com.example.invoiceManagement.service, and com.example.invoiceManagement.service.impl); and src/main/resources. On the right, the Java Editor displays the code for 'CustomerController.java'. The code defines a controller class named 'CustomerController' with methods for handling customer-related requests. The editor shows syntax highlighting and code completion features.

```
1 package o;
2
3 import ja;
4
5 @RestController
6 @RequestMapping("/customers")
7 public class CustomerController {
8     @Autowired
9     private CustomerService customerService;
10
11     @GetMapping
12     public List<Customer> getAllCustomers() {
13         return customerService.getAllCustomers();
14     }
15
16     @PostMapping
17     public Customer createCustomer(@RequestBody Customer customer) {
18         return customerService.createCustomer(customer);
19     }
20
21     @PutMapping("/{id}")
22     public Customer updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {
23         return customerService.updateCustomer(id, customer);
24     }
25
26     @DeleteMapping("/{id}")
27     public void deleteCustomer(@PathVariable Long id) {
28         customerService.deleteCustomer(id);
29     }
30
31     @ExceptionHandler(GlobalExceptionHandler.class)
32     public ResponseEntity<String> handleGlobalException(GlobalException globalException) {
33         return new ResponseEntity<String>(globalException.getMessage(), HttpStatus.INTERNAL_SERVER_ERROR);
34     }
35
36     @ExceptionHandler(ResourceNotFoundException.class)
37     public ResponseEntity<String> handleResourceNotFoundException(ResourceNotFoundException exception) {
38         return new ResponseEntity<String>(exception.getMessage(), HttpStatus.NOT_FOUND);
39     }
40
41 }
```

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

The screenshot displays the STS IDE's application console window. The title bar reads "workspace-spring-tools-for-eclipse-4.32.0.RELEASE - Spring Tool for Eclipse". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Cut, along with other developer tools. The main console area shows the application's log output:

```
workspace-spring-tools-for-eclipse-4.32.0.RELEASE - Spring Tool for Eclipse
File Edit Navigate Search Project Run Window Help
Console X
Invoicemanagement - InvoicemanagementApplication [Spring Boot App] C:\mydists\sts-4.32.0.RELEASE\plugins\org.eclipse.jdt\openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (14-Feb-2026, 6:10:05 pm elapsed: 0:03:29) [pid: 1]
:: Spring Boot :: (v4.8.2)

2026-02-14T18:18:13,407+05:30 INFO 13408 --- [invoicemanagement] [main] c.e.i.InvoicemanagementApplication : Starting InvoicemanagementApplication using Java 21.0.8 with profile default
2026-02-14T18:18:13,433+05:30 INFO 13408 --- [invoicemanagement] [main] c.e.i.InvoicemanagementApplication : No active profile set, falling back to 1 default profile: "default"
2026-02-14T18:18:16,908+05:30 INFO 13408 --- [invoicemanagement] [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2026-02-14T18:18:17,124+05:30 INFO 13408 --- [invoicemanagement] [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 182 ms. Found 4 JPA repository interfaces.
2026-02-14T18:18:18,473+05:30 INFO 13408 --- [invoicemanagement] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat initialized with port 8087 (http)
2026-02-14T18:18:18,522+05:30 INFO 13408 --- [invoicemanagement] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2026-02-14T18:18:18,526+05:30 INFO 13408 --- [invoicemanagement] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/11.0.15]
2026-02-14T18:18:18,866+05:30 INFO 13408 --- [invoicemanagement] [main] b.w.c.s.WebApplicationContextInitializer : Root WebApplicationContext: initialization completed in 5133 ms
2026-02-14T18:18:19,459+05:30 INFO 13408 --- [invoicemanagement] [main] org.hibernate.orm.jpa : HHH000540: Processing PersistenceUnitInfo [name: default]
2026-02-14T18:18:19,658+05:30 INFO 13408 --- [invoicemanagement] [main] org.hibernate.orm.core : HHH000001: Hibernate ORM core version 7.2.1.Final
2026-02-14T18:18:21,374+05:30 INFO 13408 --- [invoicemanagement] [main] o.i.o.j.p.SpringPersistenceUnitInfo : HHH000311: LoadTimeWeaver setup: ignoring JPA class transformer
2026-02-14T18:18:21,589+05:30 INFO 13408 --- [invoicemanagement] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2026-02-14T18:18:22,252+05:30 INFO 13408 --- [invoicemanagement] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@63e0f3c
2026-02-14T18:18:22,257+05:30 INFO 13408 --- [invoicemanagement] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2026-02-14T18:18:22,517+05:30 [INFO] 13408 --- [invoicemanagement] [main] org.hibernate.orm.deprecation : HHH00000025: MySQLDialect does not need to be specified explicitly
2026-02-14T18:18:22,598+05:30 INFO 13408 --- [invoicemanagement] [main] org.hibernate.orm.connections.PoolingConnectionProvider : HHH10001005: Database info:
Database JDBC URL [jdbc:mysql://localhost:3306/invoices?allowPublicKeyRetrieval=true&useSSL=false]
Database driver: MySQL Connector/J
Database dialect: MySQLDialect
Database version: 8.0.40
Default catalog/schema: invoices/undefined
Autocommit mode: undefined/unknown
Isolation level: REPEATABLE_READ [default REPEATABLE_READ]
JDBC fetch size: none
Pool: DataSourceConnectionProvider
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2026-02-14T18:18:26,120+05:30 INFO 13408 --- [invoicemanagement] [main] org.hibernate.orm.core : HHH000489: No JTA platform available (set "hibernate.transaction.jta.platform")
2026-02-14T18:26,592+05:30 INFO 13408 --- [invoicemanagement] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2026-02-14T18:26,931+05:30 INFO 13408 --- [invoicemanagement] [main] o.s.d.j.r.query.QueryEntityCachingFactories : Hibernate is in classpath; If applicable, HQL parser will be a performance bottleneck
2026-02-14T18:27,521+05:30 [INFO] 13408 --- [invoicemanagement] [main] JpaBaseConfiguration$OpenInViewConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, data access methods will be synchronized at most once per request
2026-02-14T18:28,628+05:30 INFO 13408 --- [invoicemanagement] [main] o.s.boot.tomcat.TomcatWebServer : Tomcat started on port 8087 (http) with context path '/'
2026-02-14T18:28,648+05:30 INFO 13408 --- [invoicemanagement] [main] c.e.i.InvoicemanagementApplication : Started InvoicemanagementApplication in 17.306 seconds (process PID 1)
```

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

The screenshot displays the Postman application interface. At the top, there's a header with various icons and a search bar labeled "Search Postman". Below the header, a message says "We're updating our plans and pricing on March 1. See our blog for more details." The main workspace shows an "Overview" tab selected, with a URL "http://localhost:8087/api/customers/3" highlighted. A "Send" button is visible in the top right of the request panel. The "Body" tab is active, showing a JSON response:

```
{  
    "id": 3,  
    "name": "Neha",  
    "email": "neha@gmail.com",  
    "address": "Hyderabad",  
    "customerId": 3  
}
```

The status bar at the bottom indicates a "200 OK" response with a "161 ms" execution time and a "276 B" size.

## 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.