College Canteen Ordering & Token Management System

Resume-Worthy Spring Boot Backend Project

Balanced Complexity | 3-4 Weeks | Team of 5

Professional Project Documentation

November 14, 2025

Project Philosophy: Learn by Building Real Systems

This project strikes the perfect balance: **professional enough for your resume**, yet **achievable for beginners** learning Spring Boot. You'll build a production-grade REST API with authentication, database operations, and real-world business logic.

Resume Highlights

Why This Project Stands Out on Your Resume:

- **Full-Stack Backend:** REST APIs, Database Design, Business Logic
- **Real Authentication:** JWT-based security (industry standard)
- **Complex Domain Logic:** Token queue management, order workflows
- **Modern Tech Stack:** Spring Boot 3.x, MySQL, Postman, Git
- **Production Practices:** Exception handling, validation, documentation
- **Only 12-15 Core Files:** Manageable for beginners, impressive for recruiters

START HERE - Team Action Items

Before Week 1:

- Read this guide (15 min)
- Join GitHub repo and clone project
- Install: Java 17, MySQL, Maven, Postman, Git
- Watch 1 Spring Boot tutorial (3 hours)

Week 1-3 Responsibilities: See Section 5 for your specific role and tasks

Contents

1 Executive Summary

1.1 What You'll Build

A professional backend system where students can order food digitally, receive queue tokens, and track their orders in real-time. Canteen staff can manage orders through a FIFO (First-In-First-Out) token system. Built with Spring Boot 3.x, MySQL, JWT authentication—production-grade and resume-worthy.

1.2 Perfect Balance: Not Too Simple, Not Too Complex

Aspect	What We Skip	What We Include
Files	30+ files with microservices	12-15 focused files
Security	Complex OAuth2, roles	Simple JWT (industry standard)
Features	Payment gateways, AI	Core ordering + token system
Database	10+ tables	4 essential tables
Frontend	React/Angular	Postman (API testing)
Deployment	Kubernetes, CI/CD	Optional Docker

Table 1: Balanced scope for 3-4 weeks

2 Technology Stack

Component	Technology
Backend Framework	Spring Boot 3.x
Language	Java 17 (LTS)
Security	Spring Security + JWT
Database	MySQL 8.0
ORM	Spring Data JPA
API Testing	Postman
Build Tool	Maven
Version Control	Git + GitHub
IDE	IntelliJ IDEA / Eclipse / VS Code

Table 2: Resume-worthy tech stack

Why This Stack?

- **Spring Boot:** #1 Java framework, used by 70% of enterprises
- **JWT:** Industry-standard authentication
- **MySQL:** Most popular database, easy to learn
- **Postman:** Professional API testing tool

3 Project Structure (12-15 Files)

Listing 1: Manageable file structure

```
canteen-system/
  src/main/java/com/canteen/
    CanteenApplication.java

    model/
      User.java
      MenuItem.java
      Order.java
      OrderItem.java

    repository/
      UserRepository.java
      MenuRepository.java
      OrderRepository.java
      OrderItemRepository.java

    service/
      AuthService.java
      MenuService.java
      OrderService.java

    controller/
      AuthController.java
      MenuController.java
      OrderController.java

    dto/
      LoginRequest.java
      RegisterRequest.java
      OrderRequest.java
      ApiResponse.java

    security/
      JwtUtil.java
      SecurityConfig.java

    exception/
      GlobalExceptionHandler.java

  src/main/resources/
    application.properties
    data.sql

  pom.xml
  README.md
```

4 Database Design (4 Tables)

4.1 ER Diagram

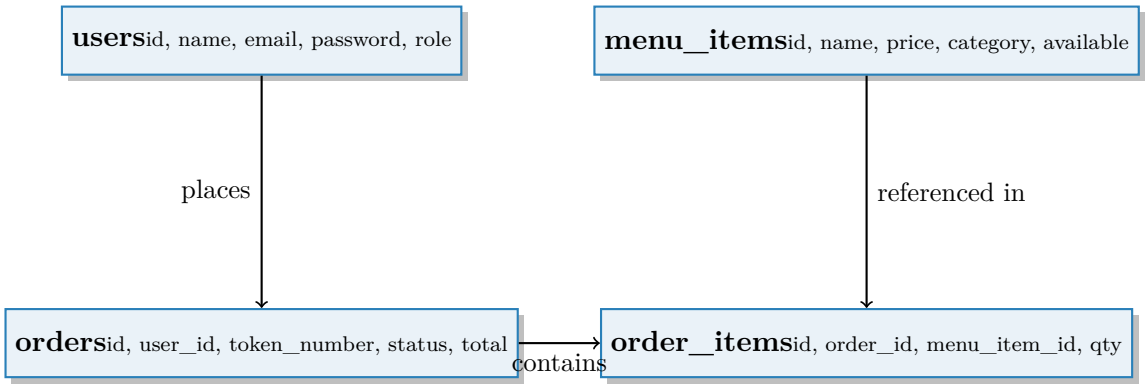

Figure 1: Simple but complete ER diagram

4.2 SQL Schema

Listing 2: Complete database schema (4 tables)

```sql
-- Table 1: Users
CREATE TABLE users (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('STUDENT', 'STAFF') DEFAULT 'STUDENT',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Table 2: Menu Items
CREATE TABLE menu_items (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description VARCHAR(255),
    price DECIMAL(10,2) NOT NULL,
    category VARCHAR(50),
    available BOOLEAN DEFAULT TRUE,
    image_url VARCHAR(255)
);

-- Table 3: Orders
CREATE TABLE orders (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    token_number INT NOT NULL,
    total_price DECIMAL(10,2) NOT NULL,
    status ENUM('PLACED', 'COOKING', 'READY', 'COMPLETED')
        DEFAULT 'PLACED',
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Table 4: Order Items
CREATE TABLE order_items (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    order_id BIGINT NOT NULL,
    menu_item_id BIGINT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (menu_item_id) REFERENCES menu_items(id)
);

-- Sample data for testing
INSERT INTO menu_items (name, description, price, category)
VALUES
('Masala Dosa', 'Crispy dosa with filling', 40.00, 'South Indian'),
('Veg Sandwich', 'Grilled with vegetables', 30.00, 'Snacks'),
('Coffee', 'Hot filter coffee', 15.00, 'Beverages'),
('Samosa', 'Crispy fried samosa', 20.00, 'Snacks');
```

5 REST API Endpoints (12 Total)

5.1 Authentication APIs

Endpoint	Method	Description
/api/auth/register	POST	Register new user. Body: `{name, email, password, role}`
/api/auth/login	POST	Login. Returns JWT token. Body: `{email, password}`

Table 3: Auth endpoints

5.2 Menu APIs

Endpoint	Method	Description
/api/menu	GET	Get all menu items (public)
/api/menu/{id}	GET	Get single menu item
/api/menu	POST	Add menu item (STAFF only, requires JWT)
/api/menu/{id}	PUT	Update menu item (STAFF only)
/api/menu/{id}	DELETE	Delete menu item (STAFF only)

Table 4: Menu CRUD endpoints

5.3 Order & Token APIs

Endpoint	Method	Description
/api/orders	POST	Place order. Returns token number
/api/orders/my-orders	GET	Get logged-in user's orders (JWT required)
/api/orders/token/{tokenNo}	GET	Check order status by token
/api/orders/queue	GET	Get pending orders (STAFF only)
/api/orders/{id}/status	PUT	Update order status (STAFF only)

Table 5: Order and queue endpoints

6 Team Division (5 Members)

Member 1: Database & Setup Lead

Responsibilities:

- Setup Spring Boot project skeleton
- Configure MySQL database connection
- Create all 4 entity classes
- Write database schema SQL
- Create sample data
- Help others with setup issues

Timeline: Week 1 (Days 1-7)

Learning: JPA entities, relationships, database basics

Member 2: Authentication & Security

Responsibilities:

- Implement user registration and login
- JWT token generation and validation
- Password encryption (BCrypt)
- Spring Security configuration
- AuthController with 2 endpoints

Timeline: Week 2 (Days 8-14)

Learning: JWT, Spring Security, authentication flow

Member 3: Menu Management

Responsibilities:

- Complete CRUD operations for menu items
- Input validation
- MenuController with 5 REST endpoints
- Role-based access (STAFF can add/edit)
- Test all endpoints in Postman

Timeline: Week 2 (Days 8-14)

Learning: REST APIs, service layer, validation

Member 4: Order Management & Token System

Responsibilities:

- Order placement logic (most complex!)
- Token number generation (auto-increment with daily reset)
- Calculate total price from items
- Save order + order items in transaction
- Order history for students

Timeline: Week 2-3 (Days 12-18)

Learning: Transactions, complex business logic

Member 5: Queue Management & Testing

Responsibilities:

- Order status updates (workflow)
- Queue view for staff (pending orders)
- Token lookup functionality
- Complete Postman collection
- Integration testing
- Documentation and README

Timeline: Week 3 (Days 15-21)

Learning: Status workflows, testing, documentation

7 Week-by-Week Plan (3-4 Weeks)

Week 1: Foundation & Learning (Days 1-7)

Goals: Spring Boot basics learned, project structure set up, database ready.

Tasks:

1. Days 1-2: Watch Spring Boot tutorial (2-3 hours each)
2. Days 3-4: Member 1 sets up project, everyone clones from Git
3. Days 5-6: Create all 4 entities, test database connection
4. Day 7: Team meeting - review progress, resolve blockers

Deliverable: Running Spring Boot app with database connected

Week 2: Core Features (Days 8-14)

Goals: Authentication working, menu CRUD complete, basic order placement.

Tasks:

1. Member 2: Implement registration + login (JWT)
2. Member 3: Build menu CRUD APIs
3. Member 4: Start order placement logic
4. Member 1: Support others, fix bugs
5. Member 5: Start Postman collection

Deliverable: Login working, menu APIs working, tested in Postman

Week 3: Advanced Features (Days 15-21)

Goals: Complete order system with token, staff queue management, full integration.

Tasks:

1. Member 4: Complete order placement + token generation
2. Member 5: Order status updates, queue view
3. Member 3: Role-based access control for menu
4. Member 2: Exception handling, security refinements
5. All: Full system testing, documentation

Deliverable: Complete working system, all 12 APIs tested, README done

8 Key Concepts You'll Master

8.1 Spring Boot Concepts

- **Dependency Injection:** `@Autowired`, `@Service`, `@Repository`
- **REST Controllers:** `@RestController`, `@GetMapping`, `@PostMapping`
- **JPA Entities:** `@Entity`, `@Id`, `@ManyToOne`, `@OneToMany`
- **Repositories:** `JpaRepository<Entity, ID>`
- **DTOs:** Separating API layer from database layer
- **Exception Handling:** `@ControllerAdvice`, custom exceptions

8.2 Database Concepts

- Entity relationships (One-to-Many)
- Foreign keys and referential integrity
- Transactions (`@Transactional`)
- SQL queries with JPA

8.3 Security Concepts

- JWT token structure (Header.Payload.Signature)
- Password hashing with BCrypt
- Role-based access control
- Securing REST endpoints

9 Critical Code Examples

9.1 Token Generation Logic (Core Feature)

Listing 3: Token generation in OrderService.java

```
@Service
public class OrderService {

    @Autowired
    private OrderRepository orderRepository;

    public Order placeOrder(OrderRequest request, User user) {
        // 1. Calculate total price
        double total = calculateTotal(request.getItems());

        // 2. Generate token number (FIFO)
        int tokenNumber = generateTokenNumber();

        // 3. Create order
        Order order = new Order();
        order.setUser(user);
        order.setTotalPrice(total);
        order.setTokenNumber(tokenNumber);
        order.setStatus(OrderStatus.PLACED);

        // 4. Save order
        Order savedOrder = orderRepository.save(order);

        // 5. Save order items
        saveOrderItems(savedOrder, request.getItems());

        return savedOrder;
    }

    private int generateTokenNumber() {
```

```
        LocalDate today = LocalDate.now();
        Integer maxToken = orderRepository
            .findMaxTokenForDate(today);
        return (maxToken == null) ? 1 : maxToken + 1;
    }
}
```

9.2 JWT Authentication

Listing 4: Login with JWT in AuthService.java

```
@Service
public class AuthService {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private JwtUtil jwtUtil;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public AuthResponse login(LoginRequest request) {
        User user = userRepository.findByEmail(request.getEmail())
            .orElseThrow(() -> new BadCredentialsException(
                "Invalid␣credentials"));

        if (!passwordEncoder.matches(request.getPassword(),
            user.getPassword())) {
            throw new BadCredentialsException("Invalid␣credentials");
        }

        String token = jwtUtil.generateToken(user.getEmail(),
            user.getRole());

        return new AuthResponse(token, user.getName(),
            user.getRole());
    }
}
```

10 Demo Flow for Presentation

10.1 Demo Script (5-7 minutes)

1. **Show GitHub Repository** (30 sec)
    - Clean commit history
    - Good README with setup instructions

- Each member's contributions visible

2. **Start Backend** (30 sec)
   - Run Spring Boot app: `mvn spring-boot:run`
   - Show console: "Application started on port 8080"
3. **Postman Demo - Student Flow** (2 min)
   - Register: POST /api/auth/register
   - Login: POST /api/auth/login → Get JWT token
   - View menu: GET /api/menu
   - Place order: POST /api/orders → Token #5
   - Check status: GET /api/orders/token/5 → PLACED
4. **Postman Demo - Staff Flow** (2 min)
   - Login as staff
   - View pending queue: GET /api/orders/queue
   - Update status: PUT /api/orders/5/status → COOKING
   - Mark as READY → COMPLETED
5. **Show Code** (1-2 min)
   - Explain `generateTokenNumber()` logic in OrderService
   - Show SecurityConfig.java and JWT filter
   - Show database schema in MySQL Workbench
6. **Conclusion** (30 sec)
   - Summarize: JWT Security, Token Queue, 12 APIs
   - Mention team collaboration

11 Team Communication & Daily Sync

11.1 Daily Standup Template (15 minutes)

Each member answers 3 questions:

- **What did I complete yesterday?** (specific code files, tested features)
- **What am I doing today?** (tasks from the timeline)
- **What blockers do I have?** (database issues, unclear logic, test failures)

11.2 Git Workflow

1. Create your branch: `git checkout -b feature/your-name`
2. Make commits with clear messages:
   - GOOD: `git commit -m "feat: Add JWT token generation"`
   - BAD: `git commit -m "fix stuff"`
3. Push and create Pull Request, get 1 review, merge

11.3 How to Help Each Other

- **Debugging:** If stuck > 30 min, ask teammate
- **Code Review:** Everyone reviews someone else's PR before merging
- **Testing:** Test endpoints before pushing, don't break main branch

12 Quick Troubleshooting

- **MySQL Connection Failed:** Check MySQL running, verify credentials in `application.properties`, ensure database exists
- **Invalid JWT Token:** Pass token in Authorization header as `Bearer <token>`, check expiry settings
- **Order Items Not Saving:** Use `@Transactional`, enable cascade delete, save parent first then children
- **CORS Error:** Add `@CrossOrigin(origins = "*")` to controllers (Postman doesn't have CORS issues)

13 Testing & Deployment

13.1 Testing Strategy

- **Manual Testing:** Postman collection with all 12 endpoints
- **Unit Tests:** JUnit 5 + Mockito for services
- **Integration Tests:** TestRestTemplate for full flow
- **Database:** Verify all data saved correctly

13.2 Local Development

Listing 5: Running the application

```
# Clean and build
mvn clean package

# Run the application
java -jar target/canteen-system.jar

# OR run directly
mvn spring-boot:run
```

13.3 Optional Deployment

- **Docker:** Containerize Spring Boot app and MySQL
- **Heroku:** Deploy free tier with ClearDB MySQL
- **Railway:** Simple cloud deployment with GitHub integration

14 Learning Resources

14.1 Required Tutorials

- **Spring Boot Basics:** "Spring Boot in 100 Steps" by Udemy (2-3 hours)
- **JWT Authentication:** "Spring Security + JWT Tutorial" (1-2 hours)
- **Database Design:** "MySQL Fundamentals" (2 hours)
- **REST API Design:** "RESTful API Principles" (1 hour)

14.2 Documentation Links

- **Spring Boot Official:** https://spring.io/projects/spring-boot
- **Spring Data JPA:** https://spring.io/projects/spring-data-jpa
- **Spring Security:** https://spring.io/projects/spring-security
- **MySQL Documentation:** https://dev.mysql.com/doc/

14.3 Useful Tools

- **Postman:** https://www.postman.com/downloads/
- **MySQL Workbench:** Visual database design and querying
- **IntelliJ IDEA Community:** Free Spring Boot IDE
- **Git/GitHub:** Version control and portfolio showcase

15 FAQ

- **Q: Can we add payment integration?** A: Possible in Phase 2. This project focuses on core ordering.
- **Q: What if a team member drops out?** A: Redistribute tasks among 4 members, adjust timeline slightly.
- **Q: Should we build a frontend?** A: Optional as Week 4 extension. Focus on backend first.
- **Q: How do we handle concurrent orders?** A: Use database locking for token number atomicity.
- **Q: Can we use MongoDB instead of MySQL?** A: Yes, with Spring Data MongoDB.
- **Q: How much learning time needed?** A: Budget 8-10 hours for Spring Boot fundamentals in Week 1.

16 Future Scope (Bonus Features)

This project is a complete backend and a great foundation for more:

- **Live Queue (WebSockets):** Push live updates to staff queue screen
- **Payment Integration:** Razorpay or Stripe integration
- **Basic Frontend:** React or Angular frontend to consume API
- **Docker Deployment:** Containerize with Docker Compose
- **Unit Testing:** JUnit 5 tests for services and controllers
- **Email Notifications:** Spring Mail when order is READY

17 Conclusion

By completing this project, your team will have a professional, end-to-end backend system. You'll master in-demand Java ecosystem skills (Spring Boot, JPA, Security) and build a complex application solving a real-world problem. This project will impress recruiters.

Resume Highlights

Your Resume Blurb:
"Developed a RESTful API for a College Canteen Token System using Spring Boot, Spring Security (JWT), and MySQL. Implemented core business logic for a FIFO token queue, complex order management, and role-based (STUDENT/STAFF) access control. Documented and tested all 12 endpoints in a team environment using Git and Postman."

Good luck! This project will definitely impress recruiters.