# Secure Delivery Lockers Spring Boot Application Documentation

Muhammad Omer Khan

December 15, 2025

## Contents

## 1 Project Overview

**Project Name:** Secure Delivery Lockers Backend
**Tech Stack:**

- Spring Boot 3
- Spring Security with JWT
- Spring Data JPA / Hibernate
- PostgreSQL
- Java 17+
- Maven

**Description:**
Backend service for managing secure delivery lockers. Supports user authentication (signup, login, OTP verification), locker management, slot creation, reservations, and opening lockers using OTP.

## 2 Base URLs

- Base URL (local): `http://localhost:8080/api`
- Modules:
  - Auth: `/api/auth/**`
  - Lockers: `/api/lockers/**`
  - S3/File uploads: `/api/s3/**`

## 3 Authentication

### 3.1 Signup / Login

**Endpoint:** `POST /api/auth/register`
**Request Body:**

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

**Responses:**

- OTP Sent:

```
{
  "success": true,
  "message": "OTP sent to your email",
  "data": {
    "token": null,
    "status": "OTP_SENT"
  }
}
```

- Login Successful:

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "token": "JWT_TOKEN_HERE",
    "status": "LOGIN_SUCCESS"
  }
}
```

### 3.2 OTP Verification

**Endpoint:** `POST /api/auth/verify-otp`
**Request Body:**

```
{
  "email": "user@example.com",
  "otp": "123456"
}
```

**Response Example:**

```
{
  "success": true,
  "message": "OTP verified successfully",
  "data": {
    "token": "JWT_TOKEN_HERE",
    "status": "LOGIN_SUCCESS"
  }
}
```

## 4 Lockers API

**Note:** All locker endpoints require JWT Authorization header:

```
Authorization: Bearer <JWT_TOKEN>
```

## 4.1 Create Locker (For Admin)

**Endpoint:** `POST /api/lockers/create`
**Request Body:**

```
{
  "name": "Locker A",
  "location": "Street 123, City",
  "lockerImage": "MultipartFile",
  "totalSlots": "12"



}
```

**Response:**

```
{
  "success": true,
  "message": "Locker created successfully",
  "data": {
    "id": "UUID",
    "name": "Locker A",
    "location": "Street 123, City",
    "slots": []
  }
}
```

## 4.2 Get All Lockers

**Endpoint:** `GET /api/lockers/get-all`
**Response:**

```
{
  "success": true,
  "message": "All Lockers Fetched Successfully",
  "data": [ ... ]
}
```

## 4.3 Create Locker Slot

**Endpoint:** `POST /api/lockers/lockerId/create-slot`
**Request Body:**

```
{
  "slotNumber": 1,
  "size": "MEDIUM"
}
```

**Response:**

```
{
  "success": true,
  "message": "Slot Created Successfully",
  "data": { ... }
}
```

### 4.4 Reserve Locker

**Endpoint:** `POST /api/lockers/reserve-locker`
**Request Body:**

```
{
  "lockerId": "UUID",
  "slotId": "UUID",
  "userEmail": "user@example.com",
  "reservationTime": "2025-12-15T20:00:00"
}
```

**Response:**

```
{
  "success": true,
  "message": "Reserved Successfully",
  "data": { ... }
}
```

### 4.5 Open Locker

**Endpoint:** `POST /api/lockers/open-locker`
**Request Body:**

```
{
  "otp": 123456
}
```

**Response:**

```
{
  "success": true,
  "message": "Opened Successfully",
  "data": { ... }
}
```

## 5 Error Handling

- 401 → Unauthorized
- 403 → Forbidden
- 400 → Bad Request
- 500 → Internal Server Error

## 6 Security

- JWT-based authentication
- Passwords stored with BCrypt
- `/api/auth/**` → public
- `/api/lockers/**` → JWT protected

## 7 Notes

- OTP is sent via email for signup/reservation
- JWT token expires after configurable time
- Slots have `slotNumber` and `size`
- All IDs are UUIDs