# 🚀 Backend Intern Task – Login & Signup System (Spring Boot)

Hi Shivam 👋

Welcome aboard! This is your **first task** as a Backend Intern. The goal of this assignment is to help you understand our backend structure, coding standards, and real-world authentication flows.

Take your time, focus on **clean code**, **best practices**, and **proper structure**. Feel free to document assumptions where needed.

---

## 📌 Objective

You need to build a **complete authentication module** using **Spring Boot**, which includes:

1. ✅ User Registration (Signup)
2. ✅ User Login
3. 🔁 Account Recovery (Optional but appreciated)

This module should be **production-ready**, scalable, and secure.

---

## 🛠️ Tech Stack (Expected)

- Java 17+
- Spring Boot
- Spring Web
- Spring Data JPA
- Spring Security
- MySQL / PostgreSQL (any relational DB)
- Maven or Gradle
- Lombok (optional)

---

## 📂 Module Scope

You are expected to create:

- Controller layer
- Service layer
- Repository layer
- Entity models
- DTOs (Request/Response)
- Basic validation & exception handling

---

# 👤 User Entity (Minimum Fields)

### user table (users)

id (BIGINT, PK)
name (VARCHAR)
email (VARCHAR, UNIQUE)
password (VARCHAR)
phone (VARCHAR, nullable)
is_active (BOOLEAN)
created_at (TIMESTAMP)
updated_at (TIMESTAMP)

You may add extra fields if required.

---

# 🔐 OTP Entity (For Account Recovery)

### otp table (user_otp)

id (BIGINT, PK)
user_id (BIGINT, FK → users.id)
otp (VARCHAR)
purpose (ENUM: SIGNUP_VERIFICATION, PASSWORD_RESET)
expires_at (TIMESTAMP)
is_used (BOOLEAN)
created_at (TIMESTAMP)

**Notes:**

- One user can have multiple OTP records

- OTP should expire (5–10 mins)
- OTP must be marked used after successful verification

---

# 🔐 1. Signup API (Mandatory)

### Endpoint

POST /api/auth/signup

### Request Body (Example)

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "Password@123",
  "phone": "9876543210"
}
```

### Requirements

- Email must be unique
- Password must be **encrypted** (BCrypt)
- Validate inputs (email format, password length)
- Return meaningful success or error responses

---

# 🔑 2. Login API (Mandatory)

### Endpoint

POST /api/auth/login

### Request Body

```
{
  "email": "john@example.com",
  "password": "Password@123"
}
```

### Requirements

- Validate email & password
- Use Spring Security authentication
- Return:
    - Success message
    - Logged-in user details (no password)
    - (Optional) JWT token

---

# 🔁 3. Account Recovery (Optional – Bonus Points)

### Option A: Forgot Password (Preferred)

Flow:

1. User enters email
2. System generates a reset token
3. Token stored with expiry time
4. Password reset via token

Suggested Endpoints:

POST /api/auth/forgot-password
POST /api/auth/reset-password

### Option B: Simple Password Reset

- User resets password using email + OTP or token

---

# 🔒 Security Expectations

- Passwords must never be stored in plain text
- Use BCryptPasswordEncoder
- Proper HTTP status codes
- No sensitive data in logs or responses

---

## ⚠️ Error Handling

Implement global exception handling using:

@ControllerAdvice

Handle cases like:

- User already exists
- Invalid credentials
- User not found
- Validation errors

---

## 🧪 Testing

- APIs should be testable via Postman
- Provide sample requests/responses
- Edge cases should be handled

---

## 📦 Deliverables

1. Fully working Spring Boot project
2. Clean code with proper package structure
3. README explaining:
    - How to run the project
    - API list
    - DB configuration
4. SQL schema or JPA auto-generated tables

---

## ⭐ Evaluation Criteria

You'll be evaluated on:

- Code quality & structure
- Security practices
- API design

- Error handling
- Documentation

---

## 💡 Notes

- Focus on **clarity over complexity**
- If something is missing, **make reasonable assumptions and document them**
- Google & docs are allowed (copy-paste without understanding is not 😉)

---

All the best 🚀

— Eshan