Backend Development Crash Course 2025

📘 Backend Development Crash Course 2025: Spring Boot, Kotlin, JWT Auth & MongoDB

🎯 Overview

This document outlines a backend development architecture using modern technologies. It covers the essential components and practices required to build a secure and scalable web application with user authentication and data persistence.

🛠️ Technologies Used

- 🧬 Kotlin – Primary programming language.

- 🚀 Spring Boot – Backend framework.

- 🔐 JWT (JSON Web Tokens) – API authentication mechanism.

- 🍃 MongoDB – NoSQL database for data storage.

🧱 Project Structure

- config/ – Security and application configurations.

- controller/ – HTTP request handling and endpoint definitions.

- model/ – Data classes and DTOs.

- repository/ – MongoDB interfaces for CRUD operations.

- service/ – Business logic layer.

🔧 Spring Boot Setup

- Kotlin-based Spring Boot project with dependencies: Spring Web, Security, MongoDB, JWT.

- Configuration via application.yml for MongoDB URI and server settings.

🔐 JWT Authentication Flow

1. Registration – User submits username, email, and password.

2. Login – Valid credentials return a JWT token.

3. Authorized Requests – JWT included in headers for accessing secure endpoints.

Token contains claims (e.g., username, expiration) and is validated on every request through a custom filter.

🗃️ MongoDB Integration

- Managed via Spring Data MongoDB.

- Interfaces extend MongoRepository for data access.

- Entities such as User and Note stored as MongoDB documents.

📒 Application Features

👤 User Authentication

- Secure password hashing with BCryptPasswordEncoder.

- Custom UserDetailsService for loading users.

📄 Notes API

- POST /api/notes – Create a note.

- GET /api/notes – Retrieve user's notes.

- DELETE /api/notes/{id} – Remove a note by ID.

🔐 Security Configuration

- Public access for /register and /login.

- JWT-protected access for all other endpoints.

- Adjusted CORS and CSRF settings for frontend integration.

📦 Example Models

Registration Request:

{

"username": "exampleUser",

"email": "user@example.com",

"password": "password123"

}

JWT Login Response:

{

"token": "eyJhbGciOiJIUzI1NiIs..."

}

✅ Summary

- Developed REST APIs using Kotlin and Spring Boot.

- Implemented JWT for securing endpoints.

- Connected MongoDB for persistent, document-based storage.

- Applied modular architecture for clean and maintainable code.

📈 Further Improvements

- Add update and sharing functionality for notes.

- Introduce refresh tokens to extend JWT sessions.

- Build and integrate a frontend application (e.g., React or Angular).