

Week 1 — Foundations: Spring Boot, REST, and Project Setup

Goal: Get comfortable with Spring Boot basics, REST APIs, and the dev workflow.

Topics: - Spring Boot project structure - REST concepts: resources, verbs, JSON, status codes - Maven/Gradle basics - Static assets serving

Resources: - Spring Boot Getting Started Guide - Spring Boot Hello World tutorials - REST API Tutorial: <https://restfulapi.net/> - YouTube: Java Brains Spring Boot - GitHub Copilot / ChatGPT for scaffolding

Exercises: 1. Scaffold a new Spring Boot project 2. Implement `HelloController` with GET and POST 3. Serve a static `index.html` with image and buttons 4. Hook buttons to REST endpoints using JS `fetch()` 5. Test endpoints with curl/Postman

AI Integration: - Scaffold controllers and endpoints - Suggest imports and dependencies - Generate test cases

Week 2 — REST API Mastery & Data Modeling

Goal: Design and implement RESTful endpoints with persistent data.

Topics: - Spring Boot REST annotations - JSON request/response mapping - Spring Data JPA basics - PostgreSQL basics - Entity relationships: WorkOrder, Customer, Item

Resources: - Spring Boot Data JPA Docs - PostgreSQL tutorials: <https://www.postgresqltutorial.com/> - OpenAI Cookbook for REST examples - YouTube: Spring Boot REST API with PostgreSQL

Exercises: 1. Create PostgreSQL database for ski shop 2. Implement entities: WorkOrder, Customer, Item 3. Implement CRUD REST endpoints 4. Test endpoints 5. Scaffold repository and service layers

AI Integration: - Generate entity classes and repositories - Auto-generate JSON request/response examples - Write unit tests for CRUD

Week 3 — Frontend & Full-Stack Integration

Goal: Connect REST API to frontend and handle dynamic data.

Topics: - Static HTML/CSS/JS serving - React + TypeScript basics - Fetch API and async calls - Event handling and UI updates

Resources: - React Docs: <https://react.dev/> - TypeScript + React Guide - FreeCodeCamp React + TypeScript tutorials - TailwindCSS / Material UI docs

Exercises: 1. Convert static page to React components 2. Fetch WorkOrders from API and display 3. Add buttons/forms to create/update WorkOrders 4. Display real-time status updates

AI Integration: - Scaffold React components - Generate fetch calls for API - Generate CSS/JS snippets - Auto-generate unit tests with Jest

Week 4 — Notifications, Validation, and MVP Polish

Goal: Implement notifications, validation, error handling, and deployment.

Topics: - Email (SendGrid) or SMS (Twilio) notifications - Input validation and error handling - Application properties and environment config - Testing end-to-end - Deployment (Render, Railway, Fly.io)

Resources: - Twilio Docs: <https://www.twilio.com/docs/sms> - SendGrid Docs: <https://docs.sendgrid.com/> - Spring Boot Validation Docs - Deployment tutorials: Render / Railway / Fly.io

Exercises: 1. Add endpoint to trigger notifications on WorkOrder completion 2. Add validation for WorkOrder creation 3. Add global error handling 4. Deploy app and test 5. Polish frontend: notifications, tables, styling

AI Integration: - Generate Twilio/SendGrid integration code - Write validation and error handling - Generate deployment scripts (Dockerfile/YAML) - Auto-generate unit and integration tests

Outcome by end of 4 weeks

- Fully working ski shop tracker MVP
- REST API with CRUD operations
- PostgreSQL persistence
- Static/React frontend
- Buttons trigger updates
- Customer notifications
- Validation and error handling
- Deployed for testing
- Fluency with Spring Boot, REST, PostgreSQL, React/TypeScript
- Comfortable using AI-assisted development