TaskFlow Project Study Guide
===========================

## 1. Tech Stack Overview

Frontend (React + Vite + TypeScript)
- Vite for bundling and dev server
- React Router for navigation
- React Query for data fetching and cache
- TypeScript for static typing

Backend (Spring Boot + Java)
- Spring MVC for REST controllers
- Spring Data JPA (Hibernate) for persistence
- Spring Security with JWT authentication
- MapStruct + Lombok for DTO mapping and boilerplate reduction
- Maven for build lifecycle

Database
- PostgreSQL in production configuration
- In-memory H2 in local profile (PostgreSQL compatibility mode)

## 2. Project Layout

```
backend/
  config/        Application wiring (properties, seed data)
  security/      JWT filters, services, and security rules
  project/       Project entity, DTOs, mapper, service, controller
  task/          Task entity, DTOs, mapper, service, controller, comments
  user/          User accounts, auth controller, service
  common/        Shared helpers (exception handler, security utils)
  application.yml         Default Postgres datasource
  application-local.yml   H2 datasource for local profile
frontend/
  api/           Axios client and typed API helpers
  features/auth/ Auth context, login/register forms, guard
  features/projects/ Project list/detail pages, task components
  App.tsx        Route setup + app shell
```

## 3. Backend Request Flow

- Controllers receive HTTP requests and validate payloads
- Services enforce business rules and ownership via SecurityUtils
- JPA repositories persist entities (Project, TaskItem, UserAccount, TaskComment)
- JWT filter authenticates requests and populates SecurityContext
- GlobalExceptionHandler converts IllegalArgumentException to 404 JSON responses

## 4. Authentication Journey

- POST /api/auth/login uses AuthenticationManager to validate credentials
- JwtService issues access + refresh tokens signed with app.security.jwt.secret
- Frontend stores tokens in AuthContext + localStorage

- Axios interceptor attaches Authorization: Bearer <token>
- JwtAuthenticationFilter verifies tokens on each request

5. Project + Task CRUD
Projects
- listMyProjects(): only owner projects via repository findByOwnerEmailIgnoreCase
- createProject(), updateProject(), deleteProject() ensure ownership
- DTO mapping via ProjectMapper exposes ownerEmail, createdAt, tasks

Tasks
- createTask() links to parent project, optional assignee lookup
- updateTaskStatus() updates enum state
- updateTask() edits core fields, allows clearing assignee
- deleteTask() removes task; orphanRemoval cascades comments deletion
- Comment endpoints allow add/list with author resolved from JWT subject

6. Frontend Data Flow
- React Query fetches projects, project detail, tasks, comments
- Mutations invalidate relevant caches to keep UI in sync
- AuthProvider syncs tokens with Axios and localStorage
- ProjectsPage supports create/edit/delete via window prompts
- ProjectDetailPage handles task CRUD, status cycling, comment posting

7. Profiles & Database Access
- application.yml -> PostgreSQL datasource
- application-local.yml -> H2 in-memory DB (jdbc:h2:mem:taskflow)
- H2 console enabled at /h2-console (profile local) with JDBC URL above
- Restarting with local profile reseeds sample data through DataInitializer

8. How to Explore the Codebase
- Start backend: mvn spring-boot:run -Dspring-boot.run.profiles=local
- Start frontend: npm run dev
- Browse http://localhost:5173 using seeded login admin@taskflow.dev / changeme
- Inspect DB via http://localhost:8080/h2-console (user sa, blank password)
- Run SHOW TABLES; SELECT * FROM PROJECTS; SELECT * FROM TASKS; etc.

9. Ideas for Extension
- Replace window.prompt UI with dedicated forms and validation
- Persist using PostgreSQL locally (docker-compose or local install)
- Add collaborators, task attachments, email notifications
- Write integration tests (Spring Boot Test) and frontend tests (RTL/Cypress)
- Containerize services, configure environment-based secrets

10. Learning Tips
- Trace a single API call front to back (e.g. create task)
- Modify entities/DTOs to see impact across layers
- Practice writing new endpoints following existing patterns
- Document findings as you explore to build personal reference