Project Documentation

1. Project Overview

This project is a web application with a modern, decoupled architecture. It consists of a React-based frontend that communicates with a backend built on a microservices architecture. The backend is composed of four distinct services, each responsible for a specific domain: authentication, file management, note-taking, and URL shortening. All backend services are developed using Kotlin and the Spring Boot framework.

2. Frontend Documentation

The frontend is a single-page application (SPA) built with React, a popular JavaScript library for building user interfaces. It is developed using modern web technologies and tools to ensure a fast and responsive user experience.

Key Technologies and Tools

- **React:** The core library for building the user interface components.
- **Vite:** A modern frontend build tool that provides a faster and leaner development experience compared to older tools like Webpack. It is used for both the development server and the production build.
- **ESLint:** A static code analysis tool for identifying and reporting on patterns found in ECMscript/JavaScript code.

Dependencies

The main dependencies are:

- `react` : The React library.
- `react-dom` : Provides DOM-specific methods that can be used at the top level of your app.

The development dependencies include tools for building, linting, and providing a better development experience, such as `@vitejs/plugin-react` and `eslint` .

3. Backend Documentation

The backend is built on a microservices architecture, which separates the application into a set of smaller, interconnected services. This approach allows for better scalability, maintainability, and flexibility.

Common Technologies

All microservices are built with a consistent technology stack:

- **Kotlin:** A modern, concise, and safe programming language.
- **Spring Boot:** A framework for building stand-alone, production-grade Spring-based applications.
- **Spring Security:** Provides authentication and authorization for the services.
- **JWT (JSON Web Tokens):** Used for securing the APIs.
- **JPA (Jakarta Persistence API):** Used for object-relational mapping and data persistence.
- **SQLite:** A lightweight, file-based SQL database.
- **Gradle:** The build automation tool.

Microservices

- **Auth Service:** Responsible for user authentication, registration, and issuing JWTs.
- **File Service:** Manages file uploads, storage, and retrieval.
- **Note Service:** Handles the creation, retrieval, updating, and deletion of notes.
- **URL Service:** Provides URL shortening and redirection functionality.

4. API Interaction

Communication between the frontend and backend microservices is handled via RESTful APIs.

Authentication Flow

1. The user signs up or logs in through the frontend.
2. The frontend sends the user's credentials to the **Auth Service**.
3. The **Auth Service** validates the credentials and, if successful, generates a JWT.
4. This JWT is sent back to the frontend and stored locally.
5. For subsequent requests to protected endpoints on other microservices (e.g., creating a note), the frontend includes the JWT in the `Authorization` header.
6. Each microservice is configured to validate the JWT, ensuring that the request is authentic and authorized.

Data Flow

The frontend makes direct calls to the respective microservices. For example, to create a note, the frontend sends a request to the **Note Service**. This decentralized communication pattern is typical for microservices architectures, although in a more complex production environment, an API Gateway might be used as a single entry point for all frontend requests.

5. Monitoring

The project includes a monitoring and observability stack using Prometheus and Grafana.

- **Prometheus:** A time-series database and monitoring system. It is configured to scrape metrics from each of the backend microservices. The Spring Boot Actuator module in each service exposes these metrics in a Prometheus-compatible format at the `/actuator/prometheus` endpoint.
- **Grafana:** A visualization and analytics tool. It is used to create dashboards that display the metrics collected by Prometheus, allowing for real-time monitoring of the application's health and performance.