Project Documentation — ClientVault Spring Application

1. Project Overview

The ClientVault Spring Application is a Java Spring Boot–based backend system designed to manage client information securely through RESTful APIs. The project demonstrates core enterprise backend concepts including layered architecture, REST endpoint design, database integration, and configuration management.

The application is intended as a learning and development project to showcase best practices in building microservice-style backend systems using Spring Boot.

2. Objectives

- Provide a backend service for managing client records
- Demonstrate clean architecture using controller, service, and data layers
- Enable CRUD operations through REST APIs
- Integrate with a relational database
- Support secure and configurable application properties
- Prepare the application for containerization and deployment workflows

3. Technology Stack

- Java
- Spring Boot
- Spring Web
- Spring Data JPA
- Maven
- PostgreSQL
- Docker

4. System Architecture

The project follows a layered architecture:

Controller Layer — Handles HTTP requests and exposes REST endpoints.
Service Layer — Contains business logic.
Repository Layer — Manages database access.
Model/Entity Layer — Defines domain objects.

5. Key Features

- Create client records
- Retrieve client details
- Update client information
- Delete client entries

- RESTful API design
- Database persistence

6. Project Structure

```
src/
 main/
  java/
   controller/
   service/
   repository/
   model/
  resources/
   application.properties
```

7. Prerequisites

- Java JDK 17 or later
- Maven
- PostgreSQL
- Git

8. Installation

Clone the repository and build:

```
git clone https://github.com/CodeForgeCapaciti/ClientVault-Spring-Application.git
cd ClientVault-Spring-Application
mvn clean install
```

9. Configuration

Configure database settings in application.properties including database URL, username, password, and server port.

10. Running the Application

```
mvn spring-boot:run
```

The application starts on the configured port (typically 8080).

11. API Usage

Use tools like Postman or curl to test endpoints for creating, retrieving, updating, and deleting clients.

12. Testing Strategy

Includes unit testing, integration testing, and database testing to ensure reliability.

13. Deployment Considerations

Use environment variables, secure credentials, containerization, and CI/CD pipelines for deployment.

14. Security Considerations

Avoid storing secrets in source control, validate inputs, and implement authentication if extended.

15. Future Enhancements

- Add authentication
- Add API documentation
- Improve logging and monitoring
- Add caching

16. Contribution Guidelines

Fork the repository, create a branch, commit changes, and submit a pull request.

17. License

Refer to the repository license for usage terms.