Technical Documentation

Application Overview:

The **DashBus** project is a Spring Boot-based Bus Ticketing System designed to streamline the booking process for long-distance trips.

Below is an overview of the application's architecture, implementation details, and key components.

Project Structure:

The application follows a modular structure organized under one package with the following components:

- **controller:** Contains controllers responsible for handling HTTP requests and managing the flow of data between the client and the backend.
- **domain:** Includes domain objects representing entities within the application, such as Ticket, User, TicketOrder, etc.
- **repository:** Houses repositories responsible for data access and interaction with the underlying database.
- **service:** Implements business logic and services necessary for the application's functionality.

Libraries and Frameworks:

The project utilizes the following libraries and frameworks:

1. **Spring Boot:**
   - **Description:** The core framework providing a simplified, convention-over-configuration, opinionated approach to building Spring-based applications.
   - **Usage:** Used as the foundation for developing the backend components of the Bus Ticketing System.
2. **Spring Boot Starter Web:**
   - **Description:** A starter for building web applications with Spring MVC.
   - **Usage:** Enables the development of RESTful APIs and handles HTTP requests.
3. **Spring Boot DevTools:**
   - **Description:** Provides development-time tools to enhance the development experience.
   - **Usage:** Enables automatic application restarts and other development-oriented features.
4. **H2 Database:**
   - **Description:** An in-memory database for development and testing purposes.
   - **Usage:** Used as a runtime dependency to facilitate local data storage during development.
5. **MySQL Connector/J:**
   - **Description:** The official MySQL JDBC driver for connecting the application to a MySQL database.
   - **Usage:** Enables the application to interact with a MySQL database in production environments.
6. **Lombok:**
   - **Description:** A library to simplify Java code by reducing boilerplate code through annotations.

    - **Usage:** Streamlines the creation of entity classes and getter/setter methods.
7. **Spring Boot Starter Test:**
    - **Description:** A starter for testing Spring Boot applications.
    - **Usage:** Provides testing support for the application, ensuring code reliability.
8. **Spring Boot Starter Data JPA:**
    - **Description:** A starter for using Spring Data JPA with Hibernate.
    - **Usage:** Simplifies data access through the Java Persistence API (JPA) and Hibernate.
9. **Tomcat JDBC:**
    - **Description:** A connection pool provided by Tomcat.
    - **Usage:** Helps manage database connections efficiently.
10. **Spring Security Starter:**
    - **Description**: Integrates Spring Security into the Bus Ticketing System, providing default configurations for authentication, authorization, and security best practices.

Application Flow:

1. **Controller Layer:**
    - Handles incoming HTTP requests.
    - Orchestrates the flow of data between the client and the backend.
2. **Service Layer:**
    - Implements business logic and services.
    - Interacts with the repository layer for data access.
3. **Repository Layer:**
    - Manages data access and communication with the underlying database.
4. **Domain Layer:**
    - Defines entities representing the core data structure of the application.

Build and Execution:

The project uses Maven as the build tool. The Spring Boot Maven Plugin is configured to package the application as a standalone JAR file. The application can be executed using the `java -jar` command.

You can also use an appropriate IDE to run the project, I recommend using IntelliJ IDEA.

Conclusion:

This Bus Ticketing System leverages the Spring Boot framework and associated technologies to provide a scalable, efficient, and modular solution for online bus ticket bookings.

The use of industry-standard libraries and frameworks contributes to the application’s maintainability and robustness.