a) System Architecture

Applied Architectural Pattern(s):
We applied the **Model-View-Controller (MVC)** architectural pattern and **Client-Server Architecture** for this system.

- **Why MVC?**
    - **Model:** Manages data and business logic (e.g., Ticket details, Train schedules, Delay calculation).
    - **View:** The user interface (HTML/CSS interfaces for users/admins).
    - **Controller:** Handles user input, updates the model, and returns views.
    - **Reason:** It separates concerns, enhances maintainability, and supports simultaneous frontend/backend development.
- **Why Client-Server Architecture?**
    - Users interact with the frontend via browsers (clients).
    - Backend (PHP) serves requests and connects to the database (MySQL).
    - This pattern is suitable for distributed systems like online ticketing.

System Components:

- Client Side: Web Browser (HTML, CSS, JavaScript)
- Server Side: PHP (Controllers, Business Logic)
- Database Server: MySQL
- Optional: JSON API layer for third-party integration (e.g., mobile app)