PAWA Software Requirement Specifications

HCMC Metro Booking Web Application

For Guests and Passengers

Table of Contents

Introduction

The purpose of this project is to develop two integrated web applications to support the successful implementation of the Ho Chi Minh City (HCMC) Metro system. As HCMC embarks on an ambitious plan to construct 12 metro lines by 2050 (See English version here), these applications will address both passenger and operator needs.

The first web application will enable passengers to search for metro routes, book tickets, and access metro services directly from their mobile phones or PCs. The second web application will support metro operators in managing stations, tracking delays, and analyzing traffic statistics to optimize schedules and service efficiency. Ultimately, this software will facilitate smoother urban transport, reduce traffic congestion, and contribute to the city's economic and environmental well-being.

System Overview

This project involves the development of two subsystems, each serving a distinct user group. **Team PAWA** is responsible for the development of the web application for passengers. This application will allow users to search for routes, book tickets, and use digital ticket scanning for entry into the metro system. **Team OPWA** will develop the web application for metro operators, enabling them to manage metro lines, monitor station status, update delays, and view real-time traffic data to adjust schedules as necessary.

Both subsystems must be designed to communicate seamlessly with each other. The specific requirements for data exchange, including the communication channels and data structures, will be outlined in the corresponding sections of the document.

The software requirements will be categorized into three levels, inspired by the Scoville scale of spiciness — **Black Pepper**, **Habanero**, and **Red Savina** —to reflect the complexity and progression of development. These levels will help guide both teams in managing and prioritizing tasks throughout the project.

I. Black Pepper Level

I.G. General Requirements

These requirements apply to both Team **PAWA** (PAssenger Web Application) and Team **OPWA** (OPerator Web Application).

I.G.1. Frontend Architecture

- The frontend application must implement an **N-Tier architecture**.
    - For **React applications**, the architecture must include **at least** the following three tiers:
        - **Page Tier**: Responsible for rendering pages and managing routing.
        - **UI Components Tier**: Contains reusable user interface components.
        - **Backend API Calls Tier**: Handles all interactions with backend APIs.
    - For **Vanilla JavaScript applications**, the architecture must include at least the following three tiers:
        - **Presentation Tier**: Manages the user interface and rendering of content.
        - **Event Handler Tier**: Handles user interactions and events.
        - **Backend API Call Tier**: Manages all communication with the backend services.

I.G.2. Backend Architecture

- The backend application must be developed using **Spring Boot** and follow an **N-Tier architecture** with at least four tiers:
    - **Controller Tier**: Handles HTTP requests and routes them to the appropriate services.
    - **Service Tier**: Contains business logic and processes requests from the controller.
    - **Repository Tier**: Manages data persistence and database operations.
    - **Model Tier**: Defines the data structures used throughout the application.

I.G.3. Frontend Utility Class for REST Requests

- To enhance **usability** and **maintainability**, the frontend application must include a **Utility class** that:
    - Centralizes the logic for sending REST requests to the backend.
    - Handles request headers, authentication tokens, and error handling consistently.

- Ensure that modifications to REST interactions can be made in a single location, reducing duplication.

I.G.4. Global URL Configuration

- The frontend application must maintain a **Global URL configuration** file that:
  - Contains all backend endpoint path definitions.
  - Allows **easy modification of endpoint patterns from a single source**, minimizing the effort required to update endpoints across the application.

I.PA. PAWA System Requirements

I.PA.1. Registration

The Passenger Web Application (PAWA) shall provide a registration feature that allows passengers to create an account by submitting the following information:

- Email address
- Password (hashed using Bcrypt for security)
- First name
- Middle name
- Last name
- National ID
- Date of birth
- Residence address
- Phone number
- Student ID (optional)
- Disability status (yes/no)
- Revolutionary contribution status (yes/no)

The system shall check whether **all required fields contain value**. Upon successful registration, the passenger account shall be created and stored in the database with secure password hashing.

I.PA.2. Authentication & Authorization

PAWA shall implement user authentication using Basic Authentication. The system shall:

- Authenticate users by verifying the provided email address and password against stored credentials.
- Utilize **Bcrypt** to compare hashed passwords during the authentication process.
- Deny access if authentication fails and provide appropriate error messages.

Guest users can register, view Metro Lines (See I.PA.3) and Ticket Booking (I.PA.6) pages. **Other pages must be accessible only to authenticated users with Passenger role**.

I.PA.3. View Available Metro Lines

PAWA shall provide a feature to view available metro lines with the following specifications:

- Display the following details for each metro line:
  - Line name
  - First station
  - Last station
  - Total travel duration in minutes
- The total number of stations in the metro line shall be shown by default, while the name and order of stations remain hidden.
- Upon user interaction (clicking on a metro line), the system shall expand to display the order and names of all stations within the line.
- The data for metro lines must be retrieved from the Operator Web Application (OPWA) system/module. PAWA shall not develop backend features for creating or retrieving metro line data.

I.PA.4. Passenger e-Wallet

PAWA shall provide an e-Wallet feature for passengers to top-up and manage funds for ticket payments. The requirements include:

- The top-up functionality shall require only the input of the desired top-up amount.
- The wallet balance shall be updated immediately upon successful top-up. For example, if the wallet balance is 100,000đ and the passenger tops up 50,000đ, the new balance shall be 150,000đ.
- No credit card, QR code payment, or NAPAS payment integrations are required at the Black Pepper Level.

I.PA.5. Passenger Profile

PAWA shall provide a user profile page with the following capabilities:

- Display all the information provided during the registration process.
- Allow passengers to update their:
  - Email address
  - Residence address
  - Phone number
  - Password
- Display the current e-Wallet balance prominently within the profile page.

I.PA.6. Purchasing Ticket

PAWA shall support purchasing **one** ticket, offering the following ticket types:

1. **One-way Ticket**
   - Active within **24 hours after purchase**. Expire after activating at the station.
   - Usable only at stations of the trip in the ticket. For example, one-way ticket from Ben Thanh to Ba Son Station cannot be used in a Bay Hien – Pham Van Hai trip.
   - Pricing:
     - 8,000đ for up to 4 stations
     - 12,000đ for up to 8 stations
     - 20,000đ for more than 8 stations

2. **Daily Ticket**
    - Valid for **24 hours after activation**.
    - Usable across all metro lines.
    - Price: 40,000đ
3. **Three-day Ticket**
    - Valid for **72 hours after activation.**
    - Usable across all metro lines.
    - Price: 90,000đ
4. **Monthly Ticket for Students**
    - Valid for **30 days after activation.**
    - Usable across all metro lines.
    - Price: 150,000đ
    - Available only if the passenger provided a student ID during registration.
5. **Monthly Ticket for Adults**
    - Valid for **30 days after activation.**
    - Usable across all metro lines.
    - Price: 300,000đ
6. **Free Tickets** (issued to eligible groups):
    - Elderly passengers aged 60 and above
    - Children aged 6 and below
    - People with disabilities
    - People with revolutionary contribution status

I.PA.7. View History of Ticket Purchase

PAWA shall provide a ticket purchase history feature that:

- Lists all previously purchased tickets.

- Indicates the status of each ticket (**active**, **inactive**, or **expired**).
- Displays the expiry date for all active tickets.
- Displays the ticket purchase date and time

I.PA.8. Ticket Activation Simulation

PAWA shall include a ticket activation simulation feature to mimic activation at the metro gate. The system shall:

- Provide a user interface where passengers can input the ticket ID and confirm activation.
- **Accept only inactive** or **active** tickets for simulation.
- Change the ticket status **from inactive to active upon activation**.
- Maintain the active status unchanged if the ticket is already active.

II. Habanero Level

II.G. General Requirements

The following requirements apply to both **PAWA** and **OPWA** teams to ensure architectural consistency, security, and interoperability.

II.G.1. Frontend Architecture

- **Separation of Concerns:** UI components must adhere to a modular design, separating **API communication** from **event handling** and **presentation logic** to enhance maintainability and scalability.
- **Component Abstraction in React:**
    - UI elements must be structured into **Pages** and **Components**:
        - **Pages**: High-level containers that compose multiple UI components to form a complete interface (e.g., a **Dashboard Page** includes a **Search Bar** for metro stations and a **Carousel** for announcements).
        - **Components**: Reusable UI elements that encapsulate a specific feature (e.g., **Carousel**, **Table**, **Search Bar**, **Button**).

- **State Management:**
  - Utilize **local component state** for isolated interactions and **global state management** (e.g., Redux, React Context) for shared data across components.

II.G.2. Backend Architecture

The **Spring Boot application must follow modular architecture** with the following constraints to enhance maintainability, scalability, and separation of concerns:

- **Layered Structure for Each Module:**
  - Each module must have **at least four layers**:
    1. **Controller Layer**: Handles incoming HTTP requests and responses.
    2. **Service Layer**: Implements business logic and interacts with repositories.
    3. **Repository Layer**: Manages database operations using JPA/Hibernate.
    4. **Model Layer**: Defines the domain objects/entities for the module.
  - **Modules must be dedicated to specific business capabilities.**
    - Example: The **PaymentModule** should handle only payment-related services and must not contain business logic related to ticketing or user management.
- **Encapsulation through Service Interfaces:**
  - Each module must provide a **Service Interface** that **exposes only necessary functions** for external modules.
  - Other modules must **only interact with the exposed Service Interface** and must not directly access internal Service layer functions.
    - Example:
      - The **TicketModule** may expose a getTicketDetails(ticketId) function via its service interface.

        - The **PaymentModule** should **only access this function** instead of querying the Ticket database directly.

- **Data Transfer Object (DTO) Usage:**
    - Modules must define **DTOs to control data exposure**.
    - A DTO should represent **only a subset** of the Model, ensuring that unnecessary attributes **are not exposed** to the frontend or external modules.
        - Example:
            - A **UserDTO** may expose only userId, fullName, and email, while keeping passwordHash and userRoles hidden.

By enforcing this structured **modular backend architecture**, PAWA and OPWA will achieve:

- **Better maintainability** through clear separation of responsibilities.
- **Stronger security** by restricting direct access between modules.
- **Improved scalability** as new modules can be integrated with minimal impact on existing ones.

II.G.2. HTTPS Enforcement

- All backend and frontend communications must be secured using **HTTPS (HyperText Transfer Protocol Secure)** to prevent data interception and unauthorized access.
- Each team must generate **self-signed asymmetric key pair(s)** to enable HTTPS protocol on **the backend and frontend** services.

II.G.3. Cross-Origin Resource Sharing (CORS) Policy

- Each backend service must enforce a **restrictive CORS policy** to allow requests exclusively from the PAWA and OPWA frontend applications rather than using wildcard (*) for trusted origins.

II.G.4. Responsiveness

The web application must be designed to provide optimal user experience across various device types, including desktops, laptops, and mobile devices. The system shall:

- Utilize **responsive design principles**, ensuring all UI components adjust dynamically based on screen size and resolution.
- Implement a **mobile-friendly layout**, where navigation, buttons, and forms remain accessible and user-friendly on smaller screens.
- Support **fluid resizing**, ensuring that content and interactive elements maintain usability when the browser window is resized.

This requirement ensures accessibility and usability for the users across different devices, improving the satisfaction of passengers.

II.PA. PAWA System Requirements

II.PA.1. Validation of Registration Data

The Passenger Web Application (PAWA) shall enforce strict validation rules for user registration data to ensure data integrity and security. Each input field must adhere to the following validation criteria:

- **Email Address:**
  - Must follow a valid email format (e.g., example@domain.com).
  - Must end with **.com** or **.vn** domain extensions.
  - Must not contain spaces or special characters other than @ and .
  - Must be unique within the system.
- **Password:**
  - Must be hashed using Bcrypt before storage.
  - Must be at least **8 characters** long.
  - Must contain at least **one uppercase letter, one lowercase letter, and one digit, and one special character** (e.g., @, #, $, %)

- **First Name, Middle Name, and Last Name:**
  - Must support both English and Vietnamese alphabet characters, including diacritics (e.g., à, Ô, ê, Ừ, ơ).
  - Must not contain numbers or special characters.
  - Maximum length: **50 characters** per field.
- **National ID:**
  - Must be exactly **12 digits** (Vietnamese Citizen ID format).
  - Must consist of only numeric characters.
  - Must be unique within the system.
- **Date of Birth:**
  - Must follow the format **dd/mm/yyyy** (e.g., 15/08/2000).
  - Must represent a valid calendar date (e.g., 30/02/2000 is invalid).
  - Must indicate an age of at least **6 years old**.
- **Residence Address:**
  - Must be split into Number, Street, Ward, District, and City
  - Must not contain special symbols except , . - /
- **Phone Number:**
  - Must follow the Vietnamese phone number format.
  - Must contain exactly **10 digits**, starting with 0.
- **Student ID (Optional):**
  - Must contain only alphanumeric characters.
  - Maximum length: **15 characters**.
- **Disability Status:**
  - Must be a binary option (Yes or No).
- **Revolutionary Contribution Status:**
  - Must be a binary option (Yes or No).

Each field has a placeholder value with an example of a valid input. For example, Phone number's placeholder is 0921 123 456; Date of birth's placeholder is 21/12/1995.

The system shall **reject any registration attempt** that fails to meet the above validation criteria and return appropriate error messages specifying the reason for rejection.

The error message must clearly explain the **cause** of the validation failure and provide an **example** of a valid input. For instance, if an invalid email is provided, the system shall return:

"Invalid email format. The email must end with '.com' or '.vn' (e.g., example@domain.com)."

Similarly, if an incorrect date format is entered, the system shall return:

"Invalid date of birth format. Please use 'dd/mm/yyyy' (e.g., 15/08/2000)."

This ensures users receive clear feedback, enabling them to correct their input efficiently.

II.PA.2. Authentication & Authorization

The PAWA backend shall **implement authentication using JSON Web Tokens (JWT).** Upon receiving valid credentials (a registered email and the correct password), the system must generate a signed JWT token containing user identification details and authorization claims.

- The JWT must be used in authorizing PAWA and OPWA users **in both the frontend and backend of PAWA**:
    - At minimum, the JWT token should be included in the **Authorization** header of every request that requires authentication.
    - The PAWA frontend shall store the token securely and include it in all protected API requests. **The token must NOT be stored in local storage or session storage to prevent vulnerabilities such as cross-site scripting (XSS).**
    - The PAWA backend shall validate the JWT in every request to ensure that the user is authorized to access the requested resource. **Tokens with invalid signatures or expired timestamps must be rejected**, and **an appropriate error response shall be returned**.
- When receiving requests from OPWA, the JWT generated by OPWA in the request must be verified:

- When OPWA interacts with PAWA's services, it must include a JWT token in the request for authentication.
- The PAWA backend shall verify the OPWA JWT's **signature**, and **expiration time** before granting access to requested resources.
- If the OPWA token is invalid or unauthorized, PAWA must return an HTTP 401 Unauthorized response with an error message specifying the reason for rejection.

II.PA.3. View Available Metro Lines with Scheduling Information

- In addition to the requirements of **I.PA.3**, PAWA must allow **searching the trains from a station to another in one metro line based on departure time**. The OPWA system shall calculate these trips using the metro line's predefined schedule, including the first departure time and train frequency, **then display the nearest three trips**.

- For example, if a passenger searches for a **Ben Thanh - Suoi Tien** trip at **05:46 AM** on Monday 03 March 2025, and trains arrive every **10 minutes** with the **first departure at 05:30 AM**. The duration from Ben Thanh to Suoi Tien is 30 minutes, so PAWA shall display the following starting station, start time, end station, end time, and current date of next three available trips:

  Ben Thanh to Suoi Tien on Mon. 03 March 2025

  - **From 05:50 AM to 06:20 AM**
  - **From 06:00 AM to 06:30 AM**
  - **From 06:10 AM to 06:40 AM**

- PAWA shall display the information of next train in a user-friendly and compact approach. E.g., do not show all the intermediate stations by default.

- **Lazy Loading for Additional Trip Options:**

  - PAWA shall implement a **Lazy Loading mechanism** to allow users to load **the next three available trips** upon request.
  - This feature ensures passengers can **view more alternative departure times** beyond the initial three suggested options.

  - The system shall fetch the next batch of trips **dynamically** without reloading the entire page, ensuring a seamless user experience.

- PAWA shall display announcements regarding any news on the events or operations of the HCMC Metro.

- **All metro line and schedule data must be retrieved from the OPWA system. PAWA shall not implement backend features for creating or managing this data.**

II.PA.4. Passenger e-Wallet and Direct Payment

In addition to the requirements of **I.PA.4**, PAWA must support direct ticket purchases and credit-card-based wallet top-ups using a third-party Payment Gateway API, such as **Stripe** or **PayPal**.

- **Direct Ticket Purchase**:
  - Both **guests** and **authenticated passengers** can purchase metro tickets without using the e-Wallet.
  - Payments must be processed via a **secure third-party Payment Gateway API**.
  - Upon successful payment, the system must issue a digital ticket and provide a confirmation receipt.
- **Wallet Top-Up via Credit Card**:
  - Instead of manually entering the top-up amount, passengers must use a **credit card** to add funds to their e-Wallet.
  - The payment must be handled securely through a **third-party Payment Gateway API**.
  - Upon successful transaction confirmation, the wallet balance must be updated accordingly.

II.PA.5. Purchasing Tickets with Persistent Ticket Cart

In addition to the requirements specified **in I.PA.6**, PAWA must implement a **Ticket Cart** system to enhance the ticket purchasing experience. The Ticket Cart must provide the following features:

- **Multi-Ticket Selection:** Passengers can add multiple ticket types and quantities to the cart before completing the purchase. For example, a passenger can add one **Monthly Ticket for Students** and two **Three-day Tickets** in a single transaction.
- **Cart Management:** The cart must support functionalities to **remove individual tickets** or **clear all items** before finalizing the purchase.
- **Real-Time Pricing Information:** The cart must display the **subtotal cost** for each ticket type and the **total cost** of all selected tickets. For instance, if a **Monthly Ticket for Students** costs **150,000 VND** and two **Three-day Tickets** cost **180,000 VND**, the cart must show both subtotals and a final total of **330,000 VND** before checkout.
- **Persistent Access Across PAWA Pages:** The Ticket Cart must be accessible from any page within PAWA, allowing passengers to view or modify their cart without interrupting their browsing experience.
- **Session Persistence:** The cart must retain its contents when passengers navigate between different PAWA web pages, ensuring that selected tickets are not lost until explicitly removed or purchased.

This feature aims to provide passengers with **seamless** and **flexible ticket purchasing experience** while ensuring ease of access and real-time pricing visibility.

II.PA.6. Real-Time Metro Line Suspension Alerts

In addition to the features specified in **I.PA.3 (Metro Line Search and Trip Listing)**, PAWA must process and display real-time metro line suspension alerts received from OPWA. This ensures passengers are immediately informed of service disruptions due to emergencies or maintenance.

Trigger Conditions:

- When OPWA sends a **real-time notification** about a metro line or segment suspension due to:
    - **Emergency situations** (e.g., accidents, security issues).
    - **Scheduled maintenance** (e.g., track repairs, system upgrades).

System Behavior:

- **Immediate Alert Display:** PAWA must instantly display a **clear, high-visibility notification** in the UI when a suspension notification is received.
- **Notification Content:** The alert must include:
    - **Affected Metro Line and Stations**
    - **Reason for Suspension** (Emergency or Maintenance)
    - **Expected Restoration Time** (if provided)

UI & Passenger Experience:

- **In-App Banner Alerts:** PAWA must display a **banner** or **popup notification** on all relevant pages where passengers view metro line data.
- **Trip Search Adjustments:** If a passenger searches for a route affected by a suspension, PAWA must prominently warn the user and display alternate routes if OPWA provides.

System Acknowledgment:

- **Acknowledgment to OPWA:** PAWA must send a confirmation receipt to OPWA when it successfully processes a suspension notification.

III. Red Savina Level

III.G. Modularized Architecture for Reusability

The following requirements apply to both PAWA and OPWA teams to ensure a modularized architecture in both the frontend and backend, fostering reusability, maintainability, and scalability.

III.G.1. Frontend Architecture

The frontend application must implement a **componentized frontend architecture** with the following principles:

- **Encapsulation of UI Modules:**
    - UI elements, local state, event handlers, and backend API calls of a UI component **must be encapsulated within a dedicated module.**
    - Example: The Metro Line News Carousel component must be structured in a dedicated **carousel** folder. This folder should contain:

        - **Backend API calls** for retrieving announcement images.
        - **Local state** for managing the image list and the current slide.
        - **Event handlers** for navigating between slides.
        - **The JSX/HTML structure** of the component.
    - This ensures that the component is **self-contained and reusable** in different parts of the project or future projects.
- **Separation of Concerns (SoC):**
    - Each UI module must **separate logic into distinct files**:
        1. **Component file** (.jsx / .tsx): Defines the UI structure.
        2. **State and Hooks file**: Manages local state and event handlers.
        3. **API Services file**: Handles backend API communication.
- **Reusability Across the System:**
    - Common UI components must be designed as **shared components** and placed in a shared or common directory.
    - Reusable API services must be stored in a central services module.
    - Utility functions must be abstracted into a utils directory.

III.G.2. Backend Architecture

In addition to requirement **II.G.2**, the **Spring Boot backend must follow a modular monolith approach**, ensuring **clear separation of concerns** and **controlled access between modules.** Each module's interface and DTO shall be structured as follows:

1. **External Interface**
    - Defines functions that **can be accessed by other modules**.
    - Typically implemented as **Spring Services** with **@Component** or **@Service** annotations.
    - Example: A **TicketModule** exposes a method purchaseTicket() that can be called from **PaymentModule**.
2. **Internal Interface**

- Contains functions **only accessible within the same module**.
- Internal logic should be encapsulated to prevent direct access from external modules.
- Example: The **TicketModule** contains an internalCalculateFare() method that is only available within the module.

3. **External DTOs (Data Transfer Objects)**
   - Defines **DTOs shared between different modules** for structured communication.
   - Example: A **TicketResponseDTO** is used to transfer ticket purchase data from TicketModule to PaymentModule.
4. **Internal DTOs**
   - Defines **DTOs restricted to internal module use**.
   - Example: A **TicketValidationDTO** is used **only within the TicketModule** for processing ticket validation logic.

By enforcing this **modularized frontend and backend architecture**, PAWA and OPWA will achieve:

- **High reusability** of components and services.
- **Better maintainability** by keeping modules independent.
- **Improved scalability** to support future features without disrupting existing functionality.

III.PA. PAWA Advance System Requirements

III.PA.1. Allowing Adding Images of National ID and Student ID

PAWA shall allow Passengers to upload the image of their **National ID** or **Student ID card** to enable identify verification.

- The **storage size** of the image must be less than 5 MB.
- For each ID card, Passenger must upload two images to present the front and back side of the card

---

III.PA.2. Allow Passengers to Login with Google Account

PAWA shall support authentication via Google Accounts, allowing passengers to sign in without manually registering in the system.

- **OAuth 2.0 Integration**: PAWA backend must implement the OAuth 2.0 protocol to securely authenticate Google users.
- **Personal Information Collection**: Upon first-time login, PAWA shall prompt the user to provide all required personal information as stated in **II.PA.2**, except for the password.
- **Account Linking**: If a passenger has already registered with an email and password, they must be able to link their Google Account to their existing PAWA account to prevent duplicate accounts.

III.PA.3. Metro Lines Route Visualization on Map API

In addition to the requirements specified in **II.PA.3**, PAWA must enhance the user experience by integrating a **Map API** for visualizing metro routes and supporting cross-line trip searches. The system shall implement the following features:

- **Default Search by Departure Time:**
    - When passengers search for train routes, the system shall **automatically set the search time to the current time and date** unless manually adjusted.
    - The system must display the **next three upcoming trips** for each metro line based on the requested departure time. The trip schedule shall be calculated using **predefined metro line schedules**, including **first departure time and train frequency**.
- **Cross-Line Trip Search:**
    - PAWA must allow passengers to search for trips between **any two stations within the Metro Network**, even if they are on **different metro lines**.
    - If a journey requires **switching between multiple metro lines**, the system shall display the necessary transfer stations and estimated transfer times.
    - For example, a passenger searching for a trip from **Suoi Tien Station (Line 1) to Nguyen Van Linh Station (Line 4)** should receive a route that includes **transfer points and estimated travel durations.**

- **Route Visualization via Map API:**
    - PAWA shall **display the stations on an interactive map** to help passengers understand the route.
    - The system is **not required to show direct connections between stations** due to the metro lines' incomplete status. Instead, **individual station locations** should be shown as markers.
    - The **start station and end station of the searched trip must be clearly highlighted** on the map.

By implementing **real-time schedule-based searching**, **map-based route visualization**, and **cross-line trip support**, PAWA significantly enhances trip planning for passengers, offering dynamic**, intuitive, and highly informative** commuting experience.

III.PA.4. Anonymize Sensitive Payment Data

PAWA must not store **sensitive credit card information** in plaintext. To enhance security and comply with industry standards:

- **Credit card details** must never be stored in their original form.
- **Transaction IDs** received from the third-party payment provider must be **securely stored** using encryption or hashing, depending on the data type and intended use.
- All payment processing must be handled **entirely by the chosen third-party Payment Gateway API**, ensuring that PAWA does not directly process, transmit, or expose sensitive payment credentials.

III.PA.5. Re-upload National ID and/or Student ID Proof

PAWA shall allow passengers to upload updated images of their **National ID** or **Student ID** in case they are issued a new card. The system must enforce the following requirements to ensure proper document submission:

- **File Size Limitation:** Each uploaded image must not exceed **5 MB** to optimize storage and loading performance.
- **Dual-Sided Submission:** For each card type, passengers must upload **two images**:
    - **Front side** of the card

  - **Back side** of the card

- **Format Validation:** The uploaded images must be in **JPEG or PNG** format to ensure compatibility.

By implementing these requirements, PAWA ensures that passengers can seamlessly update their identification documents while maintaining **data integrity, usability, and security**.

III.PA.6. Dynamic, Informative, and Persisted Ticket Cart

In addition to the requirements specified in **II.PA.6**, the **Ticket Cart** must provide the following advanced features to enhance the passenger experience, usability, and security:

- **Adjustable Ticket Quantities:** Passengers must be able to increase or decrease the quantity of each ticket type within the cart dynamically. Changes in quantity must instantly update the subtotal and total price displayed.
- **Real-Time Cart Overview:** The Ticket Cart label, which is visible on all PAWA pages, must dynamically display the **total number of tickets** and the **cumulative price** of the cart's contents. If the cart is empty, only the cart icon must be displayed.
- **Persistent Data Across Sessions:** The cart contents must remain intact even after the page is refreshed or the passenger navigates away from PAWA.
- **Secure Storage Implementation:** If browser storage (such as **localStorage** or **IndexedDB**) is used to store cart data, it must be **secured properly**. Specifically:
  - Passenger ID **must not** be stored in plaintext.
  - Sensitive identifiers must be **hashed or encrypted** to prevent unauthorized access or data tampering.

By implementing these features, the **Ticket Cart** will offer a **dynamic, informative, and resilient** ticket purchasing experience, ensuring data persistence while maintaining security best practices.

III.PA.7. Partition and Shard Ticket Purchases

To optimize database performance and ensure efficient retrieval of ticket data, PAWA must implement partitioning and sharding strategies for ticket storage.

- **Partitioning:**
    - PAWA shall partition ticket records based on their status: **active** (valid for use)**, inactive (**purchased but not yet activated)**, and expired**.
- **Sharding:**
    - In each partition, PAWA shall distribute ticket data across multiple database shards based on ticket type.
    - Each shard shall handle a subset of ticket read and write transactions to prevent performance bottlenecks and enhance scalability.
    - The sharding mechanism must support efficient retrieval of tickets when passengers request ticket history or make new purchases.

By implementing partitioning and sharding, PAWA enhances database performance, ensures quick access to relevant ticket data, and prevents database overload as ticket volume grows.

IV.PAAC. PAWA API Consumption Requirements

The PAWA system must consume the following OPWA API endpoints to facilitate seamless integration with the OPWA system in providing metro line services to passengers.

IV.PAAC.1. Metro Line Information

Metro Line Data PAWA must obtain the data of Metro Line from the OPWA system. The PAWA Team must not develop their own backend API for CRUD Metro Line data.

IV.PAAC.2. Metro Line Announcement **(Bonus Feature)**

PAWA must obtain the most recent announcements from the OPWA system. Upon clicking on the link, the PAWA system must open the page containing information about the recent announcement. The content of the news must be provided by the OPWA system.

IV.PAAC.3. Metro Line Suspension Alert

PAWA must actively listen, process, and display real-time metro line suspension alerts

received from OPWA. PAWA must send a confirmation receipt to OPWA when it successfully processes and displays a suspension notification.

V.PAAP. PAWA API Provision Requirements

PAWA must provide the following API endpoints to facilitate seamless integration with OPWA. These APIs shall enable OPWA to access essential passenger data and perform necessary transactions related to ticketing and wallet management.

V.PAAP.1. Booking Record API

PAWA must expose an API endpoint that allows OPWA operators to retrieve the list of purchased tickets. The API shall adhere to the following specifications:

- **Authentication**: Required (**Operator** role-based access control)
- **Response Data**:
    - Ticket ID
    - Passenger ID
    - Ticket type (One-way, Daily, Three-day, Monthly Student, Monthly Adult, Free Ticket)
    - Activation status (Inactive, Active, Expired)
    - Activation timestamp (if applicable)
    - Expiry timestamp (if applicable)
    - Travel route details: Line Name, Start Station, and End Station (if applicable)

V.PAAP.2. Wallet Deposit and Payment API

To facilitate wallet management operations, PAWA must provide an API that enables T**icket Agents** at metro stations to assist passengers in paying for the ticket by cash. The API must ensure secure transactions and prevent unauthorized modifications.

- **Payment API**:
    - Authentication Role: Ticket Agent
    - Request Parameters:
        - Passenger ID
        - Ticket Type ID

        - Payment amount
    - Response:
        - Payment confirmation
        - Updated wallet balance
    - Constraints:
        - The system must verify that the passenger has sufficient wallet balance before processing the payment.

These API endpoints will ensure efficient communication between PAWA and OPWA while maintaining data security and system integrity.

V.PAAP.3. Ticket Policy API

To help the OPWA system collecting the recent ticket policies for Ticket Agent to conduct payment on behalf of the passenger, PAWA shall provide a ticket policy API adhering to the following requirements:

- **Authorization**: Required (**Ticket Agent** role-based access control)
- **Response Data**:
    - Ticket type (One-way, Daily, Three-day, Monthly Student, Monthly Adult, Free Ticket)
    - Unit Price
    - Eligibility: who can purchase this ticket type
    - Other information the teams consider necessary

The End