**Project Overview: Port La Goulette Digital Hub API**

A comprehensive **RESTful API** for managing and interacting with real-world data from **Port La Goulette**, including cruise schedules, bookings, passenger feedback, parking availability, and user schedules. The API will support **CRUD operations**, **advanced filtering**, **user authentication**, and **real-time notifications**.

**Core Entities**

**1. Cruise Schedules**

Attributes:

* Schedule ID
* Cruise Name
* Arrival Time
* Departure Time
* Origin
* Destination
* Status (e.g., "On Time," "Delayed")

**2. Bookings**

Attributes:

* Booking ID
* User ID (foreign key)
* Service Type (e.g., "Taxi," "Shuttle," "Parking")
* Service Details (e.g., taxi vendor, parking lot location)
* Booking Time
* Confirmation Status

**3. Passenger Feedback**

Attributes:

* Feedback ID
* User ID (foreign key)
* Feedback Type (e.g., "Complaint," "Suggestion," "Compliment")
* Description
* Date Submitted
* Resolution Status

**4. Parking Availability**

Attributes:

* Parking ID
* Lot Name
* Capacity
* Available Spots
* Pricing (hourly/daily rates)
* Location

**5. User Schedules**

Attributes:

* Schedule ID
* User ID (foreign key)
* Cruise Schedule ID (foreign key)
* Notification Enabled
* Reminder Time (in minutes before event)

**6. Users**

Attributes:

* User ID
* Name
* Email
* Password (hashed)
* Role (e.g., "Admin," "Passenger")

**API Endpoints**

**Authentication Endpoints**

http

Copier le code

POST /auth/signUp - Register a new user

POST /auth/signIn - Login and receive a token

PUT /auth/password - Update user's password

**Cruise Schedules Endpoints**

http

Copier le code

GET /schedules - Retrieve all cruise schedules

GET /schedules/{id} - Retrieve a specific schedule by ID

POST /schedules - Add a new cruise schedule (admin only)

PUT /schedules/{id} - Update a cruise schedule (admin only)

DELETE /schedules/{id} - Delete a cruise schedule (admin only)

**Bookings Endpoints**

http

Copier le code

GET /bookings - Retrieve all bookings (admin only)

POST /bookings - Create a new booking

PUT /bookings/{id} - Update a booking

DELETE /bookings/{id} - Cancel a booking

**Feedback Endpoints**

http

Copier le code

GET /feedback - Retrieve all feedback (admin only)

POST /feedback - Submit new feedback

PUT /feedback/{id} - Update feedback resolution status (admin only)

**Parking Endpoints**

http

Copier le code

GET /parking - Retrieve parking availability

PUT /parking/{id}/reserve - Reserve a parking spot

**User Schedule Endpoints**

http

Copier le code

GET /schedule - Retrieve user’s scheduled cruises

POST /schedule/{scheduleId} - Add a cruise to user schedule

DELETE /schedule/{scheduleId} - Remove a cruise from user schedule

PUT /schedule/notifications - Update notification preferences

**Database Schema**

**Cruise Schedules**

sql

Copier le code

CREATE TABLE CruiseSchedules (

id INT PRIMARY KEY,

cruise\_name VARCHAR(100),

arrival\_time TIMESTAMP,

departure\_time TIMESTAMP,

origin VARCHAR(100),

destination VARCHAR(100),

status VARCHAR(50)

);

**Bookings**

sql

Copier le code

CREATE TABLE Bookings (

id INT PRIMARY KEY,

user\_id INT,

service\_type VARCHAR(50),

service\_details TEXT,

booking\_time TIMESTAMP,

confirmation\_status BOOLEAN DEFAULT FALSE,

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

**Feedback**

sql

Copier le code

CREATE TABLE Feedback (

id INT PRIMARY KEY,

user\_id INT,

feedback\_type VARCHAR(50),

description TEXT,

date\_submitted TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

resolution\_status BOOLEAN DEFAULT FALSE,

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

**Parking**

sql

Copier le code

CREATE TABLE Parking (

id INT PRIMARY KEY,

lot\_name VARCHAR(100),

capacity INT,

available\_spots INT,

pricing DECIMAL(5, 2),

location VARCHAR(100)

);

**User Schedule**

sql

Copier le code

CREATE TABLE UserSchedule (

id INT PRIMARY KEY,

user\_id INT,

schedule\_id INT,

notification\_enabled BOOLEAN DEFAULT TRUE,

reminder\_time INT DEFAULT 30,

FOREIGN KEY (schedule\_id) REFERENCES CruiseSchedules(id),

FOREIGN KEY (user\_id) REFERENCES Users(id)

);

**Users**

sql

Copier le code

CREATE TABLE Users (

id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100) UNIQUE,

password VARCHAR(255),

role VARCHAR(50) DEFAULT 'Passenger'

);

**Advanced Features**

**Notifications**

* Use a task scheduler like **Celery** with **Redis** for background tasks and reminders.
* Send booking confirmations or cruise delay alerts via email using **Flask-Mail**.

**Authentication**

* Implement secure JWT-based authentication using **PyJWT**.
* Role-based access control for admin-specific actions.

**API Documentation**

Use **Swagger/OpenAPI** with **Flask-RESTX** to document all endpoints. The documentation will provide clear details for:

1. **Endpoint Descriptions**:  
   Each endpoint will have a concise description explaining its purpose and functionality.
2. **Request/Response Formats**:  
   Document supported HTTP methods, required parameters, expected request bodies, and possible response statuses.
3. **Interactive Documentation**:  
   Developers will be able to test endpoints directly from the Swagger UI.

**Technology Stack**

* **Backend:** Python with Flask
* **Database:** PostgreSQL with SQLAlchemy
* **Authentication:** JWT (via PyJWT)
* **Notifications:** Celery with Redis
* **API Documentation:** Flask-RESTX with Swagger

**Implementation Steps**

1. **Database Setup**
   * Design schema using **SQLAlchemy** and populate initial data for schedules, users, and parking.
2. **API Development**
   * Build endpoints incrementally for schedules, bookings, and feedback.
3. **Authentication**
   * Secure endpoints with JWT and implement role-based access using Flask-Security.
4. **Testing**
   * Use **Postman** or **Swagger UI** to test all endpoints.
5. **Task Scheduling and Notifications**
   * Implement reminders using **Celery** with **Redis**.
   * Use Flask-Mail for email notifications.
6. **Deployment**
   * Host the API on **AWS**, **Heroku**, or another cloud provider.

This project framework enables the development of a robust digital API for Port La Goulette, ensuring enhanced passenger experiences and operational efficiency.