Project Report: Vehicle Parking Management System

Developer

Name: Aastha
Roll Number: 22f2000645
Email: 22f2000645@ds.study.iitm.ac.in

I am currently pursuing my degree at IIT Madras. I've always enjoyed building **real-world applications** using Python and Flask. With this project, I was able to apply my knowledge of **full-stack web development**, **background job handling (Celery)**, and **performance optimization through caching (Flask-Caching with Redis)**."

Project Description

This project is a Vehicle Parking Management System that allows multiple users to register, log in, and reserve or release parking slots across various facilities. Admins manage the facilities and track user activities. Key functionality includes: managing asynchronous tasks via Celery for scheduled reminders, monthly usage reports, and CSV data export.

Technologies Used

- **Backend Framework: Flask**
- **Authentication: Flask-Security** (User management and role-based access control)
- **API Implementation: Flask-RESTful**
- **Database: SQLAlchemy** (ORM) and **SQLite** (Persistence)
- **Caching: Flask-Caching** with **Redis** (for frequently accessed data)
- **Background Tasks: Celery** + **Redis** (Job queuing)
- **Email Reports:** SMTP server (Google Mail configuration)

Database Schema Design

The database schema includes the following core entities:

- **Account**:
  account_id (p_key), mail (unique), display_name, password_hash**Role**:
  id, name, description

- **PermissionGroup:**
  group_id (p_key), name (unique), description**Parking_Lot**:
  id, location_name, price, pin_code, number_of_spots
  One-to-many with Parking_Spot

- **Facility**:
  facility_id (p_key), place_label, hourly_rate, zipcode, total_slots

- **Slot**:
  slot_id (p_key), facility_id (f_key), slot_state (A/O), assigned_user, reg_number
- **Booking:**

  booking_id (p_key), account_id (f_key), slot_id (f_key), start_time, end_time, cost_charged
- **AccountGroupLink:**

  Intermediate table linking Account and PermissionGroup

API Design

The application uses Flask-RESTful to expose APIs. Auth tokens are managed using Flask-Security.

API Endpoints Include:

- **Login**/auth/login Post
- **Register**/auth/register Post
- **Admin Dashboard Info**/admin/home GET
- **User Profile Info**/user/profile GET
- **Book Slot**/booking/reserve POST
- **Release Slot**/booking/release POST
- **View Booking History**/booking/history GET
- **Get All Facilities**/catalog/facility/api/lot GET, POST
- **Get Spot Details** /catalog/slot/api/spot/<facility_id>/<position> GET, DELETE
- **Get Admin Summary Metrics**/api/admin/summary GET
- **Get Occupancy/Stats Data**/api/admin/lot-stats GET
- **Get Revenue Data**/api/admin/revenue-per-lot GET
- **Queue CSV Export**/admin/export-csv/api/export GET
- **Get CSV Result**/admin/export-result/api/csv_result/<job_id> GET

Architecture and Features

The project is structured modularly for clarity and scalability:

- **app.py:** Main entry point, setting up core extensions (DB, Celery, Caching).
- **/backend/routes.py:** Handles fundamental application logic, including booking flow and task queuing.
- **/backend/resources.py:** Handles complex RESTful resource management for admin and facility management.

Features Implemented

Role-Based Access Control (RBAC): Access is controlled using @roles_required.

Real-Time Management: Users can book and release slots in real-time.

Asynchronous Tasks (Celery):

- **Data Export: Exports user history data to CSV.**
- **Reporting: Sends scheduled monthly parking summary reports via email.**
- **Reminders: Sends daily scheduled reminders to inactive users.**

Video

https://drive.google.com/file/d/1NcgNv8_jr3jNhmrUhucb3nUFDg0GJwPy/view?usp=drive_link