**🧱 1. System Overview**

This system is a **role-based web application** designed for mobile-first usage. It allows three types of users to interact with a central parking service platform:

* **Users** – book parking spots
* **Marshals** – manage on-site operations, payments, and validations
* **Admins** – oversee marshals, validate their shifts, and handle reporting

**🏛️ 2. Architectural Pattern: MVC (Model-View-Controller)**

The system is built using the **MVC architectural pattern**, which is ideal for web applications and promotes separation of concerns:

* **Model (M):**  
  SQLAlchemy ORM models like User, Booking, MarshalShift, Payment, and ParkingZone.
* **View (V):**  
  HTML templates rendered using **Jinja2** with Bootstrap for layout and jQuery for dynamic behavior.
* **Controller (C):**  
  Flask route handlers in app.py process requests and interface between models and views.

**🗂️ 3. System Components**

**3.1 Frontend (Mobile-First Web Interface)**

* Built using **Bootstrap 5** for responsive design and mobile compatibility.
* Uses **Jinja2 templating** for rendering dynamic views based on profile type.
* **jQuery** handles:
  + Dynamic form field labeling
  + Auto-geolocation and AJAX location posting
  + Payment submission logic

**3.2 Backend (Flask Server)**

* Built using **Flask** (microframework for Python).
* Handles all HTTP requests, session management, and routing.
* Modular functions handle:
  + Authentication and registration
  + Role-based access logic
  + Booking lifecycle
  + Marshal shift tracking
  + Payment processing and end-of-day reporting

**3.3 Database**

* **Relational Database (e.g., PostgreSQL or SQLite in development)** using **SQLAlchemy ORM**.
* Tables:
  + User: Stores all profiles (users, marshals, admins)
  + Booking: Tracks parking reservations
  + MarshalShift: Logs shift info, location, approval status
  + Payment: Tracks transaction info per booking
  + ParkingZone: Holds zone configuration and spot data

**🔗 4. Interfaces & Integration Points**

* **Browser–Server Interface**: HTTP requests from web client to Flask routes
* **AJAX API**: Location captured via browser and posted using jQuery to /update\_location
* **(Optional)** Reverse Geocoding API:
  + Integrate with **OpenStreetMap (Nominatim)** or **Google Maps API** to convert lat/lng → street + city

**⚙️ 5. Deployment Model**

* **Development:** Localhost with Flask dev server + SQLite
* **Production (recommended):**
  + Flask app behind **Gunicorn** or **uWSGI**
  + **Nginx** as a reverse proxy for HTTPS, caching, compression
  + **PostgreSQL** or **MySQL** for production-grade RDBMS
  + Containerization via **Docker** for environment consistency
  + Deployment options:
    - VPS (e.g., DigitalOcean, Linode)
    - PaaS (e.g., Heroku, Railway)
    - Kubernetes (for scalability at scale)

**🧩 6. Key Design Principles Applied**

| **Principle** | **Application in System** |
| --- | --- |
| **Modularity** | Separate templates, route logic, and models |
| **Reusability** | Shared form logic (e.g., dynamic label fields via jQuery) |
| **Scalability** | Database-first design, supports future features like APIs or native mobile apps |
| **Maintainability** | Clean MVC structure, readable Flask routes, ORM-based data models |
| **Security** | Role-based access control, login with hashed passwords, session protection |
| **Extensibility** | Easy to add new roles (e.g., zone managers), APIs, or mobile apps |

**🔒 7. Security Considerations**

* Passwords stored securely using hashing (e.g., Werkzeug or Bcrypt)
* Login and signup are role-aware
* Each route checks the profile\_type and uses @login\_required
* Payments are only processed by marshals, and shift approval is admin-only

**📈 8. Future Scalability Enhancements**

* **Switch to RESTful API** to enable native apps or React frontend
* **Asynchronous job processing** (e.g., with Celery) for email receipts, shift reports
* **Map-based UI** using Leaflet or Google Maps to visualize bookings and marshal locations
* **Rate limiting & audit logging** for security