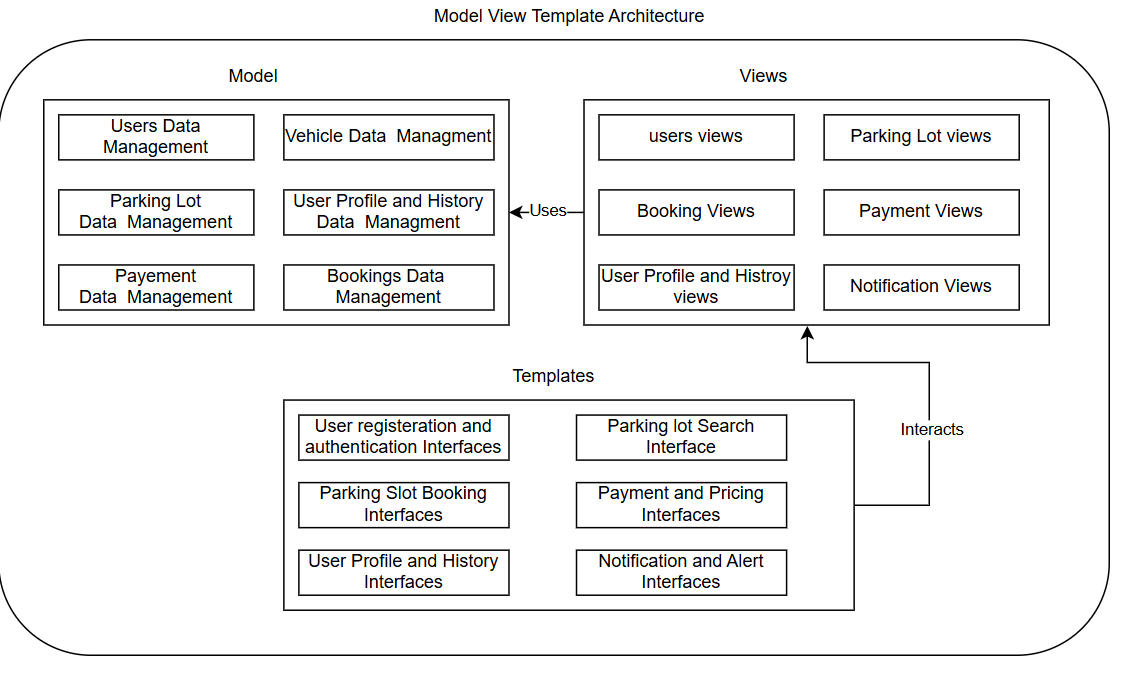
# **Specifications for VehiScan – Smart Car Parking & Booking System**

## **Quality Requirements**

1. **Scalability**: Scalability in the Smart Parking System is crucial to ensure it can efficiently handle a growing number of users and parking facilities. The system must support extensive user registration, authentication, vehicle management, real-time parking slot searches, reservations, and bookings without performance degradation. It should process numerous concurrent transactions, including payment processing, dynamic pricing adjustments, and real-time data updates.
2. **Usability**: Usability in our Smart Parking System means providing an intuitive, user-friendly interface that ensures a seamless experience for users. It is important for the system to facilitate easy navigation through features like user registration, vehicle management, parking slot search, booking, and payment processing. It should also offer clear visual cues for slot availability and status, provide real-time updates, and enable efficient booking with minimal steps. Furthermore, an intuitive interface encourages broader adoption, including among less tech-savvy users, and reduces the learning curve for the average urban resident, making the system accessible with minimal support.
3. **Performance**: Performance is a critical architectural quality requirement for our Smart Parking System as we need to ensure the system operates smoothly and responsively under various conditions. The system must deliver real-time updates on parking slot availability, process search queries, and manage bookings with minimal latency. This focus on performance will ensure a seamless user experience, even during high demand, and supports the system's reliability and scalability.
4. **Security**: Security is a fundamental architectural quality requirement for our Smart Parking System, to ensure the protection of user data and system integrity. The system must implement robust authentication mechanisms, including multi-factor authentication (MFA), to verify user identities securely. Furthermore, data encryption, both at rest and in transit, protects sensitive information such as personal details, payment data, and vehicle information from unauthorized access and breaches. Role-based access control (RBAC) ensures that only authorized users can access specific functionalities and data within the system. Regular security audits and vulnerability assessments are essential to identify and mitigate potential threats proactively

## **Software Architectural Pattern**

1. **Model View Template**

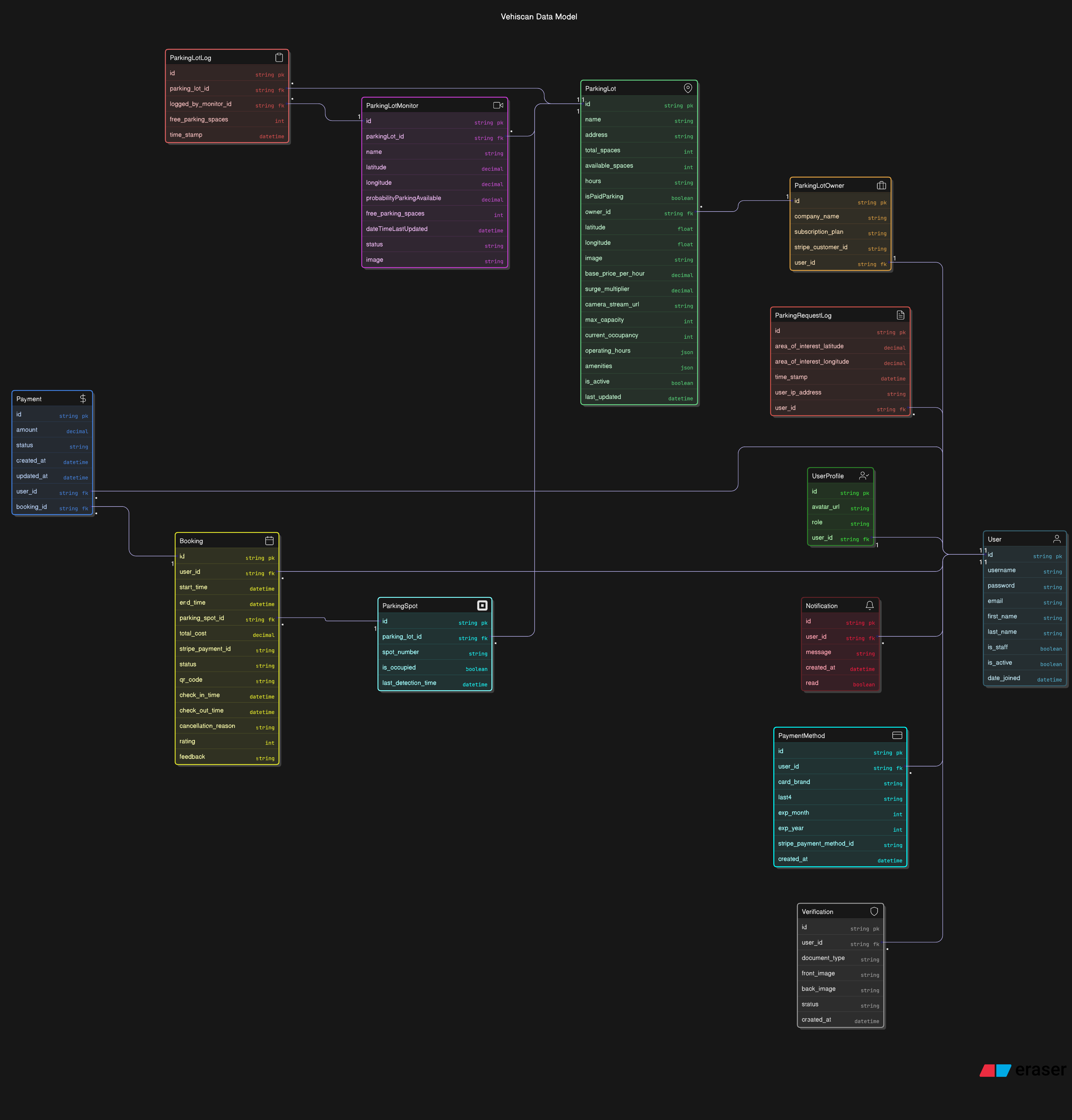


We also decided to make use of the **Model-View-Template (MVT)** architecture for the Smart Parking System, as MVT promotes a clear separation of concerns, dividing the system into three components: **Model** (handles data and business logic), **View** (processes requests and returns responses), and **Template** (controls the presentation layer or UI). This separation enhances maintainability by allowing us to work on the data layer, business logic, and frontend independently. It facilitates parallel development and simplifies debugging. The architecture also supports scalability, as each component can be updated or extended without heavily impacting the others. However, MVT can introduce complexity, especially when managing the interaction between views and templates. Ensuring a smooth flow of data from models to templates via views requires careful planning and a well-structured codebase.

## Constraints

1. **Hardware Limitations**:
   1. Restricted Environments: In narrow spaces or specific topographies, camera placement may be limited.
   2. Obstacles and Distances: The accuracy and distance of camera measurements depend on installation height and the presence of obstacles around the detection area.
2. **Privacy and Security:**
   1. Protect user data (personal information) captured by cameras. Implement encryption and comply with data protection regulations.
   2. Inform users about camera surveillance and obtain necessary consent.
   3. Balance security with privacy rights

## Class Diagram



## Technology requirements

#### Figma

#### We used **Figma** for the design and prototyping of the WeHiScan application. This step was essential to ensure the final product aligned with the project's vision and met all functional and aesthetic requirements. Figma served as a comprehensive blueprint for both the development and design teams, streamlining collaboration and maintaining consistency in UI/UX throughout the app.

#### **Advantages**

#### Easy to prototype the UI

#### Real-time collaboration

#### Version control

1. **Django**, a high-level Python web framework, was used for **backend development**. Django's robust features, built-in admin panel, and secure authentication system made it the perfect choice for VehiScan’s server-side logic. It enables rapid development and clean, pragmatic design while promoting scalability and maintainability. Django handled all core business logic, API routing, and backend integrations.

**Advantages**

* Built-in authentication and admin interface
* Secure and scalable
* Rapid development with reusable components

1. **PostgreSQL** served as the **relational database** solution for the application. It offers strong reliability, data integrity, and robust SQL support, which made it ideal for managing structured data such as user profiles, scan results, historical logs, and analytics. Its ACID compliance ensures that all operations are performed reliably and securely.

**Advantages**

* Powerful and reliable relational data storage
* Support for complex queries and data relationships
* Open-source with strong community support