**User Requirements Document**

**Bike Rental Web Application**

### 1. Overview

This document defines the user requirements for the bike rental web application. The platform aims to deliver a seamless and efficient bike rental experience in Dublin, catering to various user groups, including casual riders, subscribers, business owners, and maintenance staff.

### 2. Goals of the Project

**1. Provide Transport**

The main goal of our web application is to offer a reliable and accessible transport solution for people in Dublin. By providing bicycles, e-bikes, and e-scooters, we aim to reduce dependence on cars and public transport for short-distance travel.

* Traditional transport options like buses and taxis can be expensive, slow, and unreliable, especially during peak hours. Our system ensures that users always have an alternative mode of transport readily available.
* Shared micro-mobility solutions reduce congestion on city roads, leading to faster travel times for everyone.
* Encouraging cycling and e-scooter usage contributes to a cleaner urban environment by lowering CO₂ emissions.

**2. Make City Travels Easier**

Our platform should simplify urban mobility by allowing users to quickly locate and rent vehicles through an intuitive web interface. The system will ensure a smooth rental process, reducing waiting times and making travel hassle-free.

* The platform will feature an interactive map displaying available vehicles in real-time, ensuring users can easily find the nearest option.
* Users can unlock a bike or scooter with just a few clicks, eliminating the need for lengthy sign-ups or paperwork.
* Unlike traditional bike rental shops, which often require deposits and specific return locations, our system allows for flexible pick-up and drop-off points.
* By integrating with navigation apps, we can suggest the fastest and safest routes for users, improving overall experience.

**3. Make Travel Cheaper**

We aim to offer an affordable alternative to traditional transportation methods. Our pricing model will include pay-as-you-go options and subscriptions, making it accessible to students, tourists, and daily commuters.

* Car ownership and taxi rides are expensive due to fuel, insurance, and maintenance costs. Our service provides a budget-friendly alternative.
* Public transport fares can add up quickly, especially for those who need to make multiple daily trips. A flexible subscription plan can help users save money.
* The ability to rent a vehicle for short periods ensures users only pay for what they use, making it a cost-effective option for quick trips.
* Offering student discounts and corporate plans will further increase affordability and accessibility.

**4. Good Selection of Vehicles and Subscription Plans**

To meet different user needs, our service will provide a variety of transport options, including bicycles, electric bikes, and scooters. Additionally, we will introduce flexible subscription plans to accommodate occasional and frequent users.

* Traditional bicycles are ideal for those who prefer exercise and do not need additional power assistance.
* Electric bikes offer a balance between effort and convenience, making longer trips easier without requiring as much physical exertion.
* E-scooters are perfect for short-distance travel and provide a fun, efficient alternative to walking.
* By offering daily, weekly, and monthly subscription plans, users can choose an option that best suits their travel habits.
* Families, students, and office workers have different commuting needs, so a variety of options ensures that everyone finds a suitable solution.

**5. Provide a Flexible and Usable System**

The platform should be user-friendly, with a responsive design that works on different devices. It should also offer multiple payment options, real-time vehicle availability tracking, and easy customer support to enhance user experience.

* The web application will be mobile-friendly, allowing users to rent a vehicle from any device without needing to download an app.
* Multiple payment options, including credit/debit cards, and prepaid transport cards, will ensure accessibility for all users.
* Real-time tracking will allow users to see vehicle availability and battery levels before making a decision.
* An easy-to-use support system, including chatbots and customer service representatives, will provide quick assistance with any issues.

**3.1 User Roles & Needs**

#### 1. Tourists / Pay-As-You-Go Users

**Role Description:**  
Casual or one-off users who are visiting Dublin and need a convenient transportation option without a long-term commitment.

**Needs:**

* **Quick Access & Simplicity:**
  + Fast and easy registration/login (possibly even guest checkout).
  + Simple interface to search for and book vehicles.
* **Efficient Booking Process:**
  + Access to a real-time map view of available vehicles.
  + A streamlined booking process using Django views like BookingCreateView.
* **Flexible Payment Options:**
  + Support for pay-as-you-go payments using integrated payment gateways (Stripe, PayPal).
* **Minimal Friction:**
  + Minimal required fields during registration, allowing quick access to the service.

**Django Considerations:**

* **Models:** Use Django’s built-in User model, plus custom models for Vehicle, Booking, and Payment.
* **Views:** Create views like SearchView, BookingCreateView, and PaymentView to handle core interactions.
* **Templates:** Develop simple, intuitive pages that focus on fast booking and payment completion.

#### 2. Special Categories (e.g., Students and Other Eligible Groups)

**Role Description:**  
Users like students or other special groups who qualify for discounts or special offers.

**Needs:**

* **Discounted Pricing:**
  + Special rates and offers tailored to their category.
* **Verification Process:**
  + A system for verifying eligibility (e.g., uploading student IDs).
* **Tailored Subscription Options:**
  + Access to exclusive subscription plans that offer additional benefits.
* **Enhanced Customer Support:**
  + Dedicated support channels for resolving issues related to eligibility and benefits.

**Django Considerations:**

* **Models:**
  + Extend the User model to include a user\_category field (e.g., "student") and a VerificationStatus field.
* **Views:**
  + Create a SpecialCategoryRegistrationView to handle the verification process.
* **Templates/Forms:**
  + Develop registration forms that allow file uploads (for ID verification) and display category-specific information.

#### 3. Hobbist’s and athletes

**Role Description:**  
Users who lead an active lifestyle, preferring classic bicycles over motorized options. They are motivated by low prices and a focus on fitness and reducing pollution.

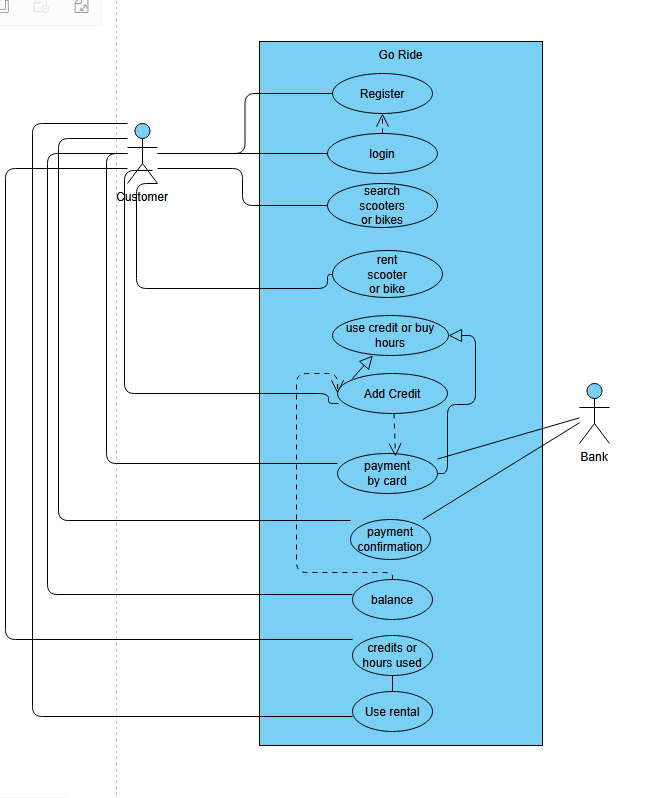
**Needs:**

* **Dedicated Vehicle Options:**
  + Exclusive access to classic bicycles, with filtering options in the search functionality.
* **Competitive Pricing:**
  + Lowest possible rates for classic bicycles to encourage frequent use.
* **Fitness and Activity Tracking:**
  + Optional features to track rides and fitness metrics (could be an integration with third-party APIs).
* **Community and Incentives:**
  + Potential community features like leaderboards or challenges to foster a healthy, active lifestyle.

**Django Considerations:**

* **Models:**
  + Modify the Vehicle model to include a category field (e.g., "classic", "e-bike", "scooter") for filtering.
* **Views:**
  + Adjust SearchView to allow users to filter exclusively for classic bicycles.
* **Templates:**
  + Create sections in the user dashboard that highlight cost savings and ride statistics specific to athletes.

**3.2** **Use Case Diagram for Customers Using The Website**



**3.3 Use Case Descriptions**

|  |  |
| --- | --- |
| **Use Case Id** | 1 |
| **Use Case Name** | Adding credit |
| **Created by** | Yones Ananzeh |
| **description** | The user will try to add credit to their account the user will have to access the system |
| **primary Actors** | Customer |
| **Secondary Actors** |  |
| **Pre-Conditions** | 1. The Customer must be on the website. 2. The Customer must have an account. 3. The Customer must have a debit or credit card. |
| **Post Conditions** | 1. The Customer receives the credits. |
| **Main Path** | 1. The Customer accesses the website 2. The Customer logs in to the website 3. The Customer goes to their balance and adds credits 4. The Customer their payment card to pay for the credits 5. The bank confirms the payment 6. The Customer has added credits to their account |
| **Alternate Path** | 1. The Customer accesses the website 2. The Customer has no account 3. The Customer registers 4. The Customer goes to their balance and adds credits 5. The Customer their payment card to pay for the credits 6. The bank denies the payment 7. The Customer has no credits. |

|  |  |
| --- | --- |
| **Use Case Id** | 2 |
| **Use Case Name** | Renting a scooter |
| **Created by** | Yones Ananzeh |
| **description** | The Customer will try to rent a scooter from the website. |
| **primary Actors** | Customer |
| **Secondary Actors** | Bank |
| **Pre-Conditions** | 1. The customer has access to the website 2. The customer must login 3. The customer must have goride credit or a payment card |
| **Post Conditions** | 1. The customer is able to use the rental |
| **Main Path** | 1. The Customer accesses the website 2. The Customer logs in to the website 3. The customer browses scooters 4. The customer rents a scooter. 5. The customer selects their scooter to rent. 6. The customer uses credit to rent the scooter. 7. The customer can now use the scooter. |
| **Alternate path** | 1. Steps 1-5 of the main path. 2. The Customer selects how much time they will use the scooter for 3. The customer uses payment card. 4. The bank confirms the payment. 5. The User can now use the scooter. |

**4.Functional Requirements**

**Core features**

***1. Search Transport***

**Description:** Users should be able to find available bicycles, e-bikes, and e-scooters near them.

**Implementation**

**Model:**

Vehicle model with fields: vehicle\_id, type, location, status, battery\_percentage (for e-bikes and scooters), price\_per\_hour.

**Views:**

A search view (search\_transport\_view) that queries the database for available vehicles based on location.

**Templates:**

A search results page that displays a list of nearby vehicles with filters (e.g., vehicle type, battery level).

**2. Login**

**Description:** Users must be able to log in to access booking and payment features.

**Implementation**

**Django Authentication System** will be used for handling user login.

**Model:**

User model (Django’s built-in User model with possible extensions for extra details like phone number).

**Views:**

LoginView to authenticate users.

**Forms:**

Login form with email/username and password fields.

**Templates:**

A login page with form validation.

**3. Registration**

**Description:** New users should be able to create an account to book and rent vehicles.

**Implementation**

**Model:**

Extension of Django’s User model to include phone\_number.

**Views:**

RegisterView to handle user registration.

**Forms:**

Registration form with fields for email, password, phone number, and optional profile picture.

**Templates:**

A registration page with error handling and form validation.

**4. Payment**

**Description:** Users should be able to pay for their bookings using different payment methods.

**Implementation**

**Model:**

Payment model with fields: user\_id, amount, payment\_method, status, payment\_id,payment\_date.

**Views:**

PaymentView to handle payment processing.

**External Integration:**

Integration with Stripe or PayPal for handling transactions.

**Templates:**

A payment confirmation page showing transaction details.

**5. Admin Panel**

**Description:** The admin panel will allow admins to manage users, vehicles, and bookings.

**Implementation**

**Django Admin:**

Admin interface to manage User, Vehicle, Booking, and Payment models.

**Model Admins:**

List filters for vehicles (availability, location, type).

Payment history and transaction tracking.

**Superuser:**

Admins will be able to log in as superusers for complete control.

**6. Booking Process**

**Description:** Users should be able to book a vehicle and track their ongoing rental.

**Implementation**

**Model:**

Booking model with fields: user\_id,rental\_id, vehicle\_id, start\_time, end\_time, status, price.

**Views:**

BookingCreateView to allow users to start a booking.

BookingDetailView to show active bookings.

**Templates:**

A booking confirmation page.

A user dashboard to view active and past bookings.

**7. Support**

**Description:** Users should be able to contact support for issues related to booking, payments, or technical problems.

**Implementation**

**Model:**

SupportTicket model with fields: user\_id,chat\_id, subject, message, status, created\_at.

**Views:**

SupportTicketCreateView to submit a new ticket.

SupportTicketListView (for admins) to manage tickets.

**Templates:**

A contact support form.

A support ticket tracking page.

**Extra features**

#### 1. ****Subscription Plans****

**Description:**  
Expand the payment system by offering various rental options such as daily, weekly, and monthly plans, including special rates for students and a pay-as-you-go model.

**Implementation:**

* **Model:**
  + Create a SubscriptionPlan model with fields:
    - plan\_name (e.g., "Daily", "Weekly", "Monthly", "Student")
    - duration (in days)
    - price
    - discount\_rate (if applicable)
* **Views:**
  + Implement views to display available plans (SubscriptionPlanListView) and handle user subscriptions (SubscriptionCreateView).
* **Templates:**
  + A subscription selection page integrated within the user dashboard.

**Example:**  
Users can choose a daily pass when traveling short distances or opt for a weekly/monthly plan for regular commuting, with a special discounted rate available for students.

#### 2. ****Map Integration****

**Description:**  
Enhance the user experience by displaying an interactive map to locate available vehicles, view their status, and see service coverage areas (e.g., within Dublin or specific zones).

**Implementation:**

* **Model:**
  + Create models such as VehicleLocation, ParkingSpot, and ServiceArea with geographic coordinates.
* **Views:**
  + Develop a map view (MapView) that queries these models and integrates with mapping libraries (e.g., Leaflet or Google Maps API).
* **Templates:**
  + An interactive map page that shows:
    - Nearby vehicles and their statuses
    - The nearest route to a selected vehicle or destination
    - Service coverage boundaries and approved parking spots

**Example:**  
Users can see available e-scooters on the map, choose the one nearest to their location, and even get directions to the vehicle or to the designated return spots.

#### 3. ****Cashback System****

**Description:**  
Offer a cashback incentive if users return vehicles to designated parking spots, promoting proper use of the service and enhancing sustainability.

**Implementation:**

* **Model:**
  + Either extend the Booking or Payment model to include a cashback field or create a separate Cashback model.
* **Views:**
  + Implement logic in the booking process to check if the vehicle is returned to an approved location and then update the user’s cashback balance.
* **Templates:**
  + Show cashback details in the user dashboard and booking history.

**Example:**  
If a user returns a bike to a verified parking zone (visible on the map), a portion of their fare is credited back to their account as cashback.

#### 4. ****Promocode Feature****

**Description:**  
Allow users to apply promocodes during the payment process to receive discounts.

**Implementation:**

* **Model:**
  + Create a PromoCode model with fields:
    - code
    - discount\_percentage
    - expiry\_date
    - usage\_limit
* **Views:**
  + Integrate promocode validation in the payment view to adjust the total cost if a valid code is entered.
* **Templates:**
  + Add an input field for promocodes on the payment page.

**Example:**  
A user enters a promocode to receive a 10% discount on their rental, and the total cost is recalculated accordingly.

#### 5. ****Referral System****

**Description:**  
Encourage word-of-mouth marketing by allowing users to refer friends and earn rewards.

**Implementation:**

* **Model:**
  + Extend the User model or create a separate Referral model to track referral codes and reward statuses.
* **Views:**
  + Build a referral dashboard where users can share a unique referral link and monitor referral progress.
* **Templates:**
  + Create a referral invitation page that explains the benefits and displays the unique referral code/link.

**Example:**  
When a referred friend signs up and completes their first booking, the referrer earns a bonus (e.g., credit towards future rentals).

#### 6. ****Language Versions****

**Description:**  
Support multiple languages to serve a diverse user base, starting with English and Gaeilge , with potential future support for languages like French or Spanish.

**Implementation:**

* **Django Internationalization (i18n):**
  + Utilize Django’s built-in i18n framework to translate text in models, views, and templates.
* **Templates:**
  + Implement translation tags ({% trans "Text" %}) and provide language selection options in the user interface.

**Example:**  
Users can switch between English and Gaeilge via a language dropdown menu, and all text on the site will dynamically change based on the selection.

#### 7. ****Extended Support Features****

**Description:**  
Broaden the support channels by including options like email, messenger integration, and in-app chat for real-time assistance.

**Implementation:**

* **Model:**
  + Enhance the existing SupportTicket model to include chat history or integrate a third-party chat service.
* **Views:**
  + Develop a dedicated support view (SupportChatView) that handles live chat functionality.
* **Templates:**
  + A support page offering multiple channels: email contact form, live chat interface, and links to messenger services.

**Example:**  
Users facing issues with booking or payment can choose to chat directly within the app instead of waiting for an email response.

#### 8. ****Credit System****

**Description:**  
Introduce an internal wallet allowing users to top up an internal balance, reducing the need for repeated external transactions.

**Implementation:**

* **Model:**
  + Create a Wallet or Credit model linked to the User model that tracks the current balance.
* **Views:**
  + Implement views for adding credits (WalletTopUpView) and using credits during the booking process.
* **Templates:**
  + A wallet management section in the user dashboard displaying balance, top-up options, and transaction history.

**Example:**  
Users can preload funds into their wallet and then pay for rentals using their account balance, streamlining the payment process.

#### 9. ****Vehicle Access****

**Description:**  
Offer multiple methods for users to unlock vehicles, including QR code scanning and a fallback manual key generation system.

**Implementation:**

* **Model:**
  + Extend the Vehicle model to include fields for storing qr\_code data or temporary access keys.
* **Views:**
  + Develop a view for QR code scanning (VehicleAccessView) and an alternative view for manual key generation.
* **Templates:**
  + An access page displaying the QR code and an option to request a key if the QR scanning fails.

**Example:**  
Users scan the QR code on a vehicle using their mobile device; if the camera is not accessible, they can request a unique access key as an alternative.

#### 10. ****Notification System****

**Description:**  
Provide timely notifications to users via SMS, email, and browser pop-ups, ensuring they stay informed about booking statuses and promotions.

**Implementation:**

* **Model:**
  + Create a Notification model with fields such as:
    - user
    - message
    - notification\_type (SMS, email, browser)
    - status (read/unread)
* **Views:**
  + Integrate notification triggers within booking, payment, and support workflows.
* **Templates:**
  + Customizable email templates and browser pop-up designs.
* **External Integration:**
  + Use services like Twilio for SMS and SendGrid for email notifications.

**Example:**  
Users receive an SMS when their rental starts, an email confirmation for payments, and browser pop-ups for timely updates or promotional offers.

### ****11. User Stats & Insights****

#### ****Description:****

Enhance user experience by providing detailed statistics about their trips, including total distance traveled, number of rides, average speed, and personal insights. Additionally, an annual summary (similar to Spotify Wrapped) will highlight key riding habits, favorite transport types, and the user’s ranking in the community.

### ****Implementation:****

#### ****Model:****

Create a UserStats model with the following fields:

* user (ForeignKey to User) – The user to whom the stats belong.
* total\_rides (IntegerField) – The total number of rides completed.
* total\_distance (FloatField) – The cumulative distance traveled in kilometers.
* average\_speed (FloatField) – The user’s average speed across all rides.
* most\_active\_day (DateField, nullable) – The user’s busiest riding day.
* favorite\_transport\_type (CharField) – The most frequently used vehicle type.
* longest\_ride\_distance (FloatField) – The longest single trip recorded.
* co2\_saved (FloatField) – Estimated CO₂ reduction compared to car usage.

#### ****Views:****

* UserStatsView (DetailView) – Displays the user’s statistics.
* AnnualStatsView – Generates a yearly summary of the user’s activity.
* LeaderboardView – Shows rankings based on distance traveled, rides taken, or CO₂ saved.

#### ****Templates:****

* **User Dashboard:** Displays personal ride statistics, such as total distance and speed.
* **Annual Summary Page:** Visual representation of yearly stats (e.g., most active day, CO₂ savings).
* **Leaderboard Page:** Compares user performance with others in their city or area.

### ****Example:****

John, a frequent e-bike user, checks his dashboard and sees that he has taken **150 rides**, traveled **500 km**, and saved **80 kg of CO₂** this year. His favorite transport type is an **electric scooter**, and his most active day was **May 10th**. At the end of the year, he receives a **personalized Wrapped-style summary** comparing his stats to other users.

#### 12. Regular Subscribers

**Role Description:**  
Users who frequently use the service and prefer subscription plans (daily, weekly, monthly) for convenience and cost savings.

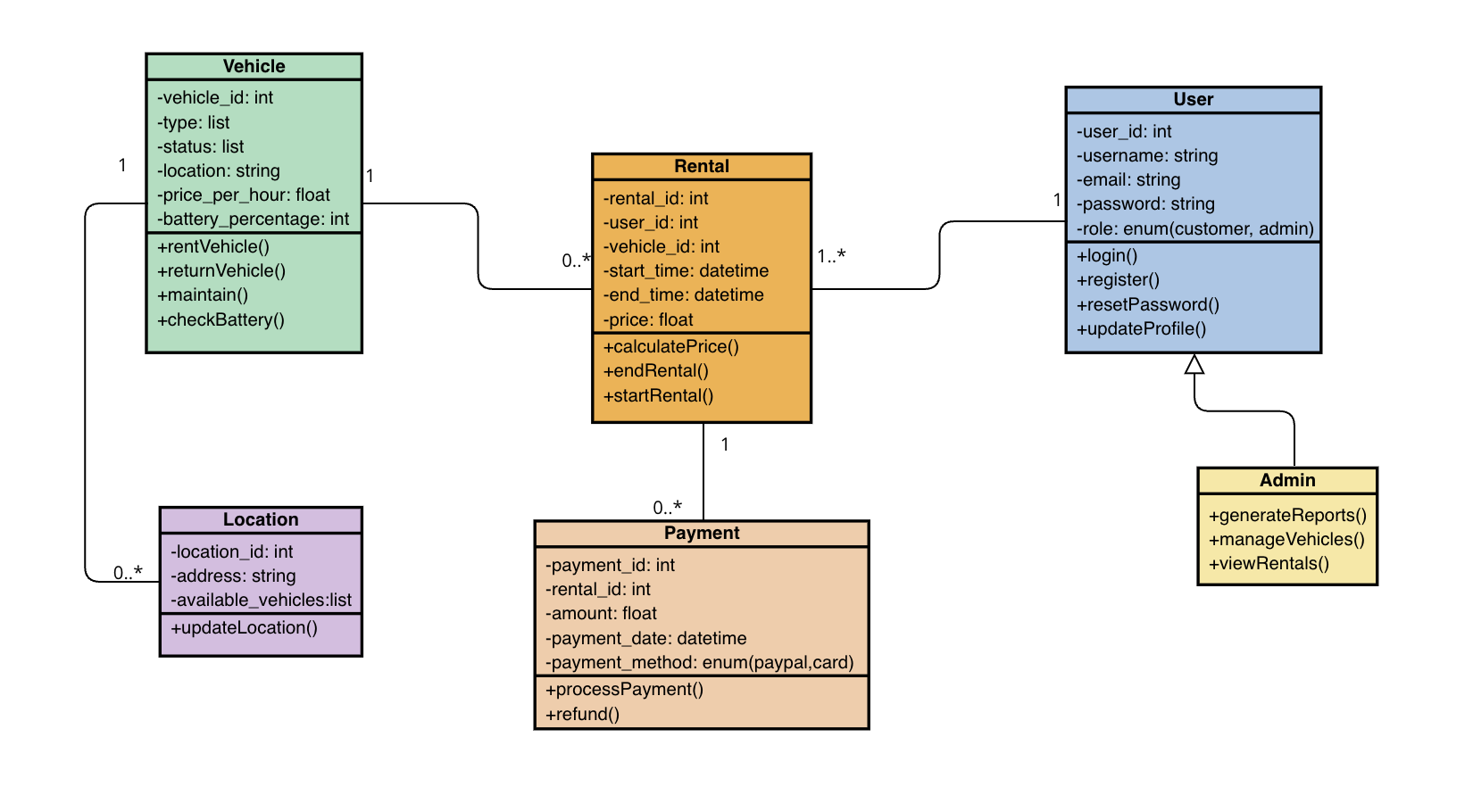
**Needs:**

* **Subscription Management:**
  + Ability to select and manage different subscription plans.
  + A clear overview of subscription status, ride history, and benefits in a personal dashboard.
* **Automated Payments:**
  + Option for auto-renewal payments to streamline recurring subscriptions.
* **Enhanced Booking Features:**
  + Priority booking options and enhanced filtering/search capabilities.
* **Detailed Usage Tracking:**
  + Access to ride statistics, billing history, and plan benefits.

**Django Considerations:**

* **Models:** Create a SubscriptionPlan model and a Subscription model linked to the User model (extended if necessary).
* **Views:** Develop views such as SubscriptionListView, SubscriptionDetailView, and SubscriptionUpdateView to manage subscriptions.
* **Templates:** Design a user dashboard that integrates subscription details, ride history, and payment options.

**4.6 Class Diagram**

****

**5. Non-Functional Requirements**

**5.1 Performance & Scalability**

* The system must handle **high user traffic** and support future expansion.
* Optimized for **fast loading times** and **real-time updates.**

**5.2 Security & Compliance**

* **End-to-end encryption** for user data & transactions.
* **GDPR compliance** for data protection.
* **Regular security audits** to prevent vulnerabilities.

**5.3 User Experience (UX) & Accessibility**

* **Mobile-first design** for seamless access across devices.
* **Intuitive user interface** for effortless navigation.
* **Accessibility features** to accommodate users with disabilities.

**6. Project Feasibility**

**6.1 Timeline & Development Plan**

* **Three-month development period (Jan 27 – Apr 28)** with a **prototype demo every 3 weeks**.
* **Prioritize core features** (authentication, rentals, payments, mapping) before enhancements.
* **Development milestones:**
  + **Week 3:** Basic booking system.
  + **Week 6:** Payment integration.
  + **Week 9:** Admin dashboard & analytics.

**6.2 Potential Risks & Contingency Plans**

1. **Limited Team Capacity**
   * **Risk:** A high workload could lead to burnout.
   * **Solution:** Focus on MVP features & ensure knowledge-sharing within the team
   * .
2. **Technical Challenges**
   * **Risk:** Potential issues with database, APIs, or security.
   * **Solution:** Allocate debugging time & maintain thorough documentation.
3. **Strict Deadlines & Unexpected Delays**
   * **Risk:** Bugs or unforeseen problems affecting timelines.
   * **Solution:** Include buffer time in the final weeks to resolve last-minute issues.
4. **User Testing & Feedback Implementation**
   * **Risk:** Major usability issues discovered late in development.
   * **Solution:** Conduct early-stage testing with simulated users.