# **User Registration and Login – Module Specification**

## **Overview**

The User Registration and Login Modules provide secure and efficient access management for the Smart Parking Spot Finder App. These modules allow users to create accounts, log in using email or mobile number, and manage credentials securely. The goal is to ensure user identity verification, data security, and a seamless onboarding experience.

## **1. User Registration – Module Specification**

### **1.1 Functional Requirements**

#### **1.1.1 Registration Methods**

* Register using:  
  + Mobile Number + OTP + Password

#### **1.1.2 Required Fields**

* Full Name
* Mobile Number (must be unique)
* Password
* OTP
* Optional: Vehicle Information (license plate, vehicle type)

#### **1.1.3 Workflow**

* User selects registration method ( Mobile).
* Fills out registration form with required details.
* System verifies:  
  + Mobile number validity.
  + Password strength (for Phone number registration).
* System sends OTP for mobile number verification.
* On successful OTP verification:  
  + User is registered and redirected to the homepage.

### **1.2 Validation Rules**

* Unique mobile number
* Valid OTP (expires in 1 minute)
* Password must meet complexity requirements:
  + 8–16 characters
  + At least one uppercase, one lowercase, one digit, and one special character

### **1.3 Error Handling**

* Duplicate mobile: show “Already registered” message
* Invalid OTP: show error and allow retry
* Weak password: show password strength guide

## **2. User Login – Module Specification**

### **2.1 Functional Requirements**

#### **2.1.1 Login Methods**

* Mobile Number + Password

#### **2.1.2 Required Fields**

* Mobile Number
* Password

#### **2.1.3 Workflow**

* User enters credentials.
* System verifies credentials against the database.
* On success: redirect to dashboard/homepage.
* On failure: show appropriate error message.

### **2.2 Additional Features**

* “Remember Me” checkbox for persistent session
* “Forgot Password” link

### **2.3 Password Security**

* Store passwords using bcrypt or Argon2 hashing
* Prevent reuse of last 5 passwords
* Password strength validation during creation/reset

## **User Stories**

* As a new user, I want to register using my mobile number so I can access the app.
* As a user, I want to log in securely using my credentials.
* As a user, I want to reset my password if I forget it.
* As a user, I want to be notified of logins from unknown device

## **3. Forgot Password – Module Specification**

### **Overview**

The Forgot Password module enables users to securely reset their password if they cannot recall it. This feature verifies user identity using OTP verification sent to the registered mobile number, ensuring secure password recovery and account protection.

### **Functional Requirements**

#### **3.1 Triggering Password Reset**

* Accessible from the Login screen via a “Forgot Password?” link.
* User can initiate the process by entering:  
  + Registered Mobile Number

#### **3.2 OTP Verification**

* System sends a One-Time Password (OTP) to the registered mobile number.
* **OTP is:**
  + Numeric (e.g., 6 digits)
  + Valid for 1 minute
* User enters OTP to verify identity.
* System validates OTP against backend service.

#### **3.3 New Password Entry**

* Once OTP is verified, user is redirected to the Password Reset screen.
* User is prompted to enter a new password and confirm it.

#### **Password Rules:**

* Length: 8–16 characters
* Must contain:  
  + At least one uppercase letter
  + At least one lowercase letter
  + At least one number
  + At least one special character (e.g., @, #, $, %, etc.)

#### **3.4 Confirmation**

* **On successful password reset:**
  + User receives a confirmation notification
  + Redirected to Login screen
* If the password reset request was unauthorized, users can report it as a security issue.

### **Technical Requirements**

#### **3.5 Backend Services**

* OTP generation and delivery via SMS gateway
* OTP validation API
* Password hashing with bcrypt or Argon2
* Password reuse check against last 5 stored hashes

#### **3.6 Security Controls**

* OTP expiry and retry limits
* Brute-force protection (e.g., lockout after multiple failed attempts)
* Password validation on the client and server sides

### **Error Handling**

* **I**nvalid/expired OTP: show “OTP is incorrect or expired”
* Unregistered email/mobile: show “No account associated with this contact”
* Weak password: prompt user to follow strength rules

### 

### 

### **User Stories**

* As a user, I want to securely reset my password if I forget it.
* As a user, I want to receive an OTP to verify my identity during password reset.
* As a user, I want to be assured that my new password is secure and unique.

## **4. View Available Parking Slots – Module Specification**

### **Overview**

The View Available Parking Slots module enables users to see real-time availability of parking spaces on a map interface. It integrates with IoT sensors and location services to deliver accurate, up-to-date data on parking slot status, enhancing convenience and reducing time spent searching for parking.

### **Functional Requirements**

#### **4.1 User Interface**

* Display a map-based view of parking locations and slot availability.
* Real-time visualization of:  
  + Available (highlighted in green)
  + Occupied (highlighted in red)
  + Reserved (highlighted in yellow)
* Parking locations are marked with icons or pins.
* Tooltip or detail pane shows:  
  + Total slots
  + Available slots
  + Price/hour
  + Distance from current location
  + Slot features (EV, disabled access, etc.)

#### **4.2 Real-Time Data Integration**

* Fetch real-time data from:  
  + IoT parking sensors
  + Camera analytics
  + Backend APIs of parking providers
* Update slot availability every X seconds (configurable, e.g., every 30 seconds)
* Synchronization with backend database to reflect latest availability

#### **4.3 Location Awareness**

* Use GPS to detect user's current location
* Center map on user's current or searched location
* Allow map drag and zoom to explore other areas
* Geofencing to display parking spots within a specific radius (e.g., 5 km)

#### **4.4 Filtering & Sorting Options**

* Filters:  
  + Parking type (public, private, valet)
  + Slot type (compact, EV, disabled access)
  + Price range
  + Distance
  + Availability (now, within next 30 minutes)
* Sort by:  
  + Nearest first
  + Cheapest
  + Highest availability

#### **4.5 Slot Detail View**

* Tap/click on a parking pin opens a detailed view:  
  + Address
  + Availability count
  + Pricing
  + Booking option
  + Time restrictions (if any)
  + User ratings (optional)

### **Technical Requirements**

#### **4.6 APIs & Data Sources**

* REST APIs to fetch:  
  + Slot status
  + Location metadata
  + Pricing and features
* WebSocket (optional) for real-time updates
* Integration with mapping services (Google Maps, Mapbox)

#### **4.7 Error Handling**

* Display fallback message if data cannot be loaded
* Retry mechanism for API failures
* Offline mode (optional): show last known availability with a timestamp

### **User Stories**

* As a user, I want to view nearby available parking slots on a map so I can make quick parking decisions.
* As a user, I want to filter parking spots based on my needs (e.g., EV charging), so I can find suitable options.
* As a user, I want to see detailed information about each parking spot so I can evaluate and compare options.

## **5.Parking Slots Overview**

### **Overview**

This screen provides users with a clear, interactive view of available and occupied parking slots in a specified area. It enables real-time monitoring, filtering, and booking functionality for different slot types.

## Layout Structure

## Top Section – Header / Search / Filters

#### **Header**

* **App Title**: ParkMate
* Positioned at the top center for clear branding

#### **Search & Filters (Optional but Recommended)**

* **Search Bar**: Placeholder - *“Search slot Type…”*
* **Filter Options**:  
  + **Slot Type**:  
    - Two**-wheeler 🛵**
    - **Four-wheeler 🚗**
    - **Three-wheeler 🛺**
  + **Availability**:  
    - All
    - Available only
  + **Sort By**:  
    - Price (Low to High / High to Low)
    - Proximity (if GPS is active)

### **Main Section – Slot Grid/List View**

#### **Card Layout for Each Slot**

Use a card or tile view (grid preferred for mobile) with the following details:

| **Slot #** | **Type** | **Status** | **Price/hr** |
| --- | --- | --- | --- |
| A1 | 🚗 Four-wheeler | ✅ Available | ₹50 |
| A2 | 🛵 Two-wheeler | ❌ Occupied | ₹20 |
| B3 | 🚗 Four-wheeler | ✅ Available | ₹45 |

#### **Slot Card UI Elements**

* **🚦 Status Badge** (color-coded):  
  + Green = Available
  + Red or Grey = Occupied
* **📌 Slot Number & Type Icon**
  + E.g., “A1 🚗”
* **💸 Price/hr**: “₹XX/hr”
* **🔘 Book Now Button**:  
  + **Enabled only if** slot is **Available**
  + Action: Initiates booking workflow

### **Bottom Section – Pagination / Summary Info**

#### **Scrolling Behavior**

* **Infinite scroll** OR **“Load More”** button
* Loading indicator during data fetch

#### **Summary Text (Optional)**

* Display real-time summary:  
   **“12 Available | 8 Occupied”**

## **6. UI Screen: Book a Parking Slot**

### **Overview**

This screen enables users to complete the booking of a selected parking slot by providing essential details. It includes dynamic pricing calculation, real-time status updates, and form validation to ensure accuracy and a smooth user experience.

## **Layout Structure**

## **Top Section – Slot Information**

#### **Title**

* Heading: Book Slot A1 *(Dynamic: based on selected slot)*

#### **Slot Details Display**

* Slot Type: 🚗 Four-Wheeler *(or 🛵 for Two-Wheeler, etc.)*
* Price/Hour: ₹50
* Status: ✅ Available

*All information displayed in a visually distinct info card or header box.*

### **Booking Form Fields**

1. **👤 Full Name**
   * Input Type: Text
   * Placeholder: Enter your full name
2. **🚘 Vehicle Number**
   * **I**nput Type: Text
   * Placeholder: e.g., KA01AB1234
   * Format: Valid Indian vehicle number pattern
3. **🕒 Duration**
   * Input Type: dropdown
   * Options: 1 to 4 hours (configurable)
   * Label: Select Duration (in hours)
4. **💰 Estimated Price (Auto-calculated)**
   * Formula: ₹50 x [Selected Duration] = ₹\_\_\_
   * Updates in real-time when duration is changed

### **Confirm Booking Button**

* Button Text: ✅ Confirm Booking
* State:  
  + Enabled only if all fields are correctly filled
  + Disabled otherwise

#### **Post-Click Actions**

* Validate all form inputs
* Update slot status to “Booked” in database
* Display confirmation message:  
    
    
   “Your booking for Slot A1 has been confirmed.”

### Validation & Error Messages

* "Please enter all fields" – if any field is empty
* "Invalid vehicle number format" – if vehicle number doesn't match format
* "Duration must be at least 1 hour" – if duration is less than 1

## **7.UI Screen: Check-In / Check-Out**

### **Overview**

This screen allows users to manage the parking session for a booked slot. It supports real-time check-in, live session tracking, and check-out with payment calculation based on time used. It ensures accurate billing and enhances the user experience with timely status updates.

## **Layout Structure**

### **Top Section – Current Booking Summary**

Display a summary of the user’s active booking:

| **Field** | **Example Value** |
| --- | --- |
| Slot Number | A1 |
| Slot Type | 🚗 Four-Wheeler |
| Vehicle Number | KA01AB1234 |
| Booking Duration | 2 Hours |
| Price per Hour | ₹50 |
| Status | 🟡 Not Checked In / ✅ Checked In |

* Highlight status clearly using badges (Yellow for *Not Checked In*, Green for *Checked In*).

### **Middle Section – Timer Area**

#### **Before Check-In**

* **Button Displayed**: 🟢 Check In Now
* On tap:  
  + Record current time as **Check-In Time** (e.g., 10:05 AM)
  + Update booking status: ✅ Checked In
  + Start **Live Timer** (Optional, e.g., Ongoing: 00:01:12)

#### **After Check-In**

* Show the following:  
  + **Check-In Time**: e.g., 10:05 AM
  + **Ongoing Timer**: ⏳ 1h 13m
  + **Button**: 🔴 Check Out

### **Check-Out Flow**

On tap of **Check Out**:

1. Calculate **Total Time Parked** (e.g., 1h 45m)
2. Compute **Final Payment**:  
   * E.g., ₹50/hour × 1.83 hours = ₹92
   * (Rounding rules to be defined)
3. Show:  
   * **Total Time**: 1h 45m
   * **Final Payment**: ₹92
   * **Button**: ✅ Confirm Payment

### **Payment Confirmation Section**

After user confirms payment:

* Display **Success Message**:  
    
    
   “Thank you! Your parking session is complete.”
* Show **Booking Summary**:  
  + Slot, Duration, Total Time, Amount Paid
* Options:  
  + 🔁 Book Another Slot
  + 📜 View History

### **Optional Features**

* **Auto-alerts**: Notify user if time exceeds booked duration
* **Overtime Charges**: Automatically apply extra charges if parked time exceeds booking (e.g., ₹10 per extra 15 mins)

## **8.Booking History**

### **Overview**

This screen presents a record of all past parking sessions booked by the user. It allows users to review slot usage, payment details, and session durations. Filters and sorting options improve usability and data accessibility.

## **Layout Structure**

### **Top Section – Screen Title & Filter Controls**

* **Title**: 📜 Booking History
* Placed prominently at the top of the screen

#### **Optional Filters:**

* **Date Range Picker**:  
  + Custom date range
  + Presets: Last 7 Days, Last 30 Days
* **Sort Options**:  
  + Latest First
  + Oldest First

### **Main Section – Booking Cards/List View**

Each booking is displayed in an individual card or list item with detailed information.

#### **Booking Card Format (Example)**

* **Slot**: A1 (Four-wheeler 🚗)
* **Vehicle**: KA01AB1234
* **Duration**: 2h 15m
* **Check-in Time**: 10:05 AM
* **Check-out Time**: 12:20 PM
* **Payment**: ₹112
* **Status**: ✅ Completed

Cards should be scrollable and optionally support tap actions for more details (e.g., invoice, location)

### **🧾 Optional Features**

* **Search by vehicle number**
* **Export as PDF or CSV**
* **Click to view full booking receipt**