Functional Requirements

ID	Requirement	Category
REQ_001	The system **must** allow users to register and log in using email, phone, or social media.	Must Have
REQ_002	The system **must** allow users to search and filter services by type (e.g., cleaning, fridge repair, carpentry, etc.).	Must Have
REQ_003	The system **must** allow users to view detailed service provider profiles, including ratings, reviews, and certifications.	Must Have
REQ_004	The system **must** allow users to book a service for a selected date and time.	Must Have
REQ_005	The system **must** allow service providers to accept or reject bookings.	Must Have
REQ_006	The system **should** provide real-time status updates (e.g., "Service Confirmed," "In Progress," "Completed").	Should Have
REQ_007	The system **should** allow users to cancel bookings with appropriate cancellation policies.	Should Have
REQ_008	The system **should** support payment via credit/debit cards, wallets, and cash on delivery.	Should Have
REQ_009	The system **could** allow users to chat with service providers via an in-app chat function.	Could Have
REQ_010	The system **could** send automated reminders and notifications for scheduled services.	Could Have
REQ_011	The system **won't** allow guest users to book services without registration.	Won’t Have
REQ_012	The system **must** allow admins to add, edit, or remove services and service providers.	Must Have
REQ_013	The system **must** include a review and rating feature for completed services.	Must Have
REQ_014	The system **should** generate monthly reports on bookings, payments, and reviews for admins.	Should Have
REQ_015	The system **could** include location-based suggestions for nearby service providers.	Could Have

Non-Functional Requirements

ID	Requirement	Category
REQ_016	The system **must** support at least 10,000 concurrent users.	Performance
REQ_017	The system **must** ensure data encryption for sensitive information like passwords and payment details.	Security
REQ_018	The system **must** comply with GDPR or relevant local data protection laws.	Security
REQ_019	The system **should** respond to user actions (e.g., search, booking) within 3 seconds.	Performance
REQ_020	The system **must** be accessible via desktop, tablet, and mobile devices.	Usability
REQ_021	The system **should** recover from failures and errors within 10 seconds.	Reliability
REQ_022	The system **should** have 99.9% uptime annually.	Reliability
REQ_023	The system **could** be scalable to accommodate new services and providers in different cities.	Scalability
REQ_024	The system **must** log all critical events, such as logins, payments, and cancellations.	Security
REQ_025	The system **must** provide a user-friendly interface with easy navigation and readability.	Usability

Use Case Scenarios for Functional Requirements

Use case scenarios describe how users interact with the system to achieve specific goals.

Use Case 1: User Registration and Login

- **Actor**: User
- **Steps**:
    1. User opens the app and clicks on "Register."
    2. Users enter email, phone, or social media credentials.
    3. The system validates details and creates a user profile.
    4. User logs in successfully.
- **Related Requirements**: REQ_001

Use Case 2: Search and Filter Services

- **Actor**: User
- **Steps**:
    1. User navigates to the service search page.
    2. User selects a category (e.g., cleaning, fridge repair).
    3. User applies filters like ratings, distance, or availability.
    4. The system displays matching service providers.
- **Related Requirements**: REQ_002

Use Case 3: Booking a Service

- **Actor**: User
- **Steps**:
    1. User selects a service provider from the search results.
    2. User views provider details (REQ_003).
    3. The user selects a date and time for service.
    4. The system confirms availability and creates a booking.
    5. User receives booking confirmation.
- **Related Requirements**: REQ_004

Use Case 4: Accept or Reject Bookings

- **Actor**: Service Provider
- **Steps**:
    1. Service provider logs into the system.
    2. Service provider views pending bookings.
    3. The service provider accepts or rejects a booking.
    4. System updates booking status accordingly.
- **Related Requirements**: REQ_005

Use Case 5: Payment Processing

- **Actor**: User
- **Steps**:
    1. After completing the service, the user chooses a payment method (credit, wallet, or cash).
    2. The system processes the payment securely.
    3. User receives a payment confirmation receipt.
- **Related Requirements**: REQ_008

Use Case 6: Admin Management

- **Actor**: Admin
- **Steps**:
    1. Admin logs into the system.
    2. Admin adds or removes a service or service provider.
    3. Admin views reports and reviews for monitoring.
- **Related Requirements**: REQ_012, REQ_014