Based on your request, and from the perspective of a senior system designer, here is a detailed breakdown of all the features and requirements for the **first SDLC phase**, Requirements Gathering and Analysis.

This phase is about defining the **"what"** of the project before we get into the **"how."** We categorize these into functional and non-functional requirements to ensure we capture every aspect of the system.

**Functional Requirements (What the system must do)**

These are the core actions and services the SmartSheba app will perform for its users.

**1. User & Profile Management**

* **User Registration:** Users (both customers and providers) must be able to sign up with a phone number and/or email.
* **Secure Authentication:** The system must handle secure login and password management.
* **Profile Creation & Management:** Users can create and edit their profiles. Providers must be able to upload identity documents, certifications, and a work portfolio for verification.
* **Role-Based Access:** The system must differentiate between a Customer and a Service Provider role, with different features available for each.

**2. Service & Booking System**

* **Multi-Category Service Marketplace:** The platform must list services across multiple categories (e.g., Plumber, Tutor, Cleaner, Beauty) with sub-categories.
* **Search & Filtering:** Users must be able to search for specific services and filter results by category, location, rating, and price.
* **On-Demand & Scheduled Booking:** The system must allow for both immediate bookings and future-dated, scheduled appointments.
* **Recurring Booking:** Customers must have an option for regular, recurring services (e.g., weekly cleaning).
* **Pricing Transparency:** The system must display a fixed rate or a clear price estimation for each service.
* **Booking History:** Customers must be able to view their past and upcoming service bookings.

**3. Provider Management (for the Provider App)**

* **Request Dashboard:** Providers must have a dashboard to view incoming service requests, accept or decline them, and see a summary of their earnings.
* **Schedule Management:** A calendar or schedule view for providers to manage their availability and booked appointments.
* **Earning & Withdrawal System:** Providers must be able to view their earnings and request payouts.

**4. Communication & Safety Features**

* **In-App Messaging & Calling:** A secure messaging and calling feature must be integrated for customers and providers to communicate post-booking.
* **Live Tracking & ETA:** The system must use GPS to allow customers to track their provider's real-time location and estimated time of arrival (ETA).
* **Emergency SOS Button:** A one-tap emergency button must be available to notify pre-selected contacts, police, or ambulance services in an emergency.

**5. AI-Powered Features**

* **AI Chatbot:** The chatbot must be able to understand natural language requests (in Bangla) and suggest the appropriate service category or provider.
* **Image/Video Recognition:** Users must be able to upload a photo or video of a problem, and the AI must correctly identify the required service (e.g., a photo of a leaky pipe must be classified as a "Plumber Service").
* **AI-Based Estimation:** The AI must use historical data to provide an estimated cost and time for a service.
* **Fraud Detection:** An AI model must analyze user and provider behavior to detect and flag suspicious or fraudulent activities.

**Non-Functional Requirements (How the system must perform)**

These define the quality attributes of the system, ensuring it is reliable, secure, and user-friendly.

* **Usability:** The user interface (UI) must be simple, clear, and easy to navigate, with a **Bangla-first design** to cater to the local market.
* **Performance:** The app must be fast, with booking confirmations taking less than 5 seconds and screen loads taking no more than 2 seconds on a typical network.
* **Security:** All user data, especially personal and payment information, must be encrypted. Provider verification must be robust to build user trust.
* **Scalability:** The system architecture must be designed to handle a large and growing number of users, providers, and bookings without a drop in performance.
* **Reliability:** The platform must have a high uptime, targeting 99.9% availability to ensure services are accessible when needed.
* **Maintainability:** The code must be well-structured and documented to allow for easy updates, bug fixes, and feature additions in the future.

These requirements will serve as the foundation for the entire project.

Excellent. Now that we have a solid understanding of the project's requirements, we can begin the next phase: **Design**.

This phase is about creating the complete blueprint for the SmartSheba system. As a senior system designer, my goal is to break down the project into manageable, interconnected components and define how they will all work together to meet the requirements we've already established.

**1. High-Level Design: System Architecture**

This is the macro-level view of our entire system. We will define the major components and how they interact.

* **Frontend (Mobile Apps):** We will design two separate mobile applications, one for the **Customer** and one for the **Service Provider**, both built using Flutter. This choice allows us to target both Android and iOS with a single codebase, improving efficiency and reducing development time.
* **Backend (API Server):** A robust backend system will serve as the brain of the operation. It will handle all business logic, user authentication, service and booking management, and communication with other services. This will likely be built using a scalable framework like Node.js, Python/Django, or Go.
* **AI/ML Microservices:** The AI features will be developed as separate, specialized services. For example, the **AI Chatbot** and **Image Recognition** will each have their own dedicated microservice. This architecture keeps our main backend lean and allows us to scale or update AI models independently.
* **Third-Party Integrations:** We will define the APIs for integrating with essential external services:
  + **Payment Gateway:** For secure digital payments (e.g., bKash, Nagad, or SSLCommerz).
  + **Location & Maps Service:** For real-time GPS tracking and ETA calculations (e.g., Google Maps API).
  + **SMS & Notification Service:** For sending booking confirmations, reminders, and emergency alerts.

**2. Low-Level Design: Component & API Design**

This is the detailed plan for each module, defining its internal structure and communication methods.

* **API Endpoints:** We will design a complete set of RESTful APIs to facilitate all interactions between the mobile apps and the backend. This includes endpoints for user login, service search, booking creation, provider profile updates, and submitting ratings.
* **Class/Object Design:** We will formalize the core data models you listed. Each class (User, Service, Provider, Booking, Review) will be defined with all its attributes and a list of methods (functions) it can perform. This ensures consistency and clarity during the development phase.
* **Service & Provider Workflows:** We will create detailed flowcharts and sequence diagrams to map out complex processes, such as the complete booking lifecycle from a customer's request to a provider's completion and payment.

**3. Database Design**

The database is the foundation of our entire system. We must design it meticulously to ensure data integrity and efficient retrieval.

* **Schema Definition:** We will create a detailed schema for our database. This involves defining all the tables, their columns, data types, and the relationships between them (e.g., one-to-many, one-to-one). Key tables will include users, providers, services, bookings, reviews, and transactions.
* **Data Integrity:** We will enforce rules to ensure data is consistent and accurate. For instance, a booking entry cannot exist without a valid user\_id and provider\_id.

**4. UI/UX Design**

While we have the conceptual UI requirements, this is where we bring them to life visually.

* **Wireframing & Prototyping:** We will create initial low-fidelity wireframes to lay out the key screens for both the user and provider apps. These will be followed by high-fidelity prototypes that are interactive, allowing us to test the user flow before writing any code.
* **Design System:** A comprehensive design system will be created. This includes defining a color palette (with a focus on accessibility and branding), typography, iconography, and a library of reusable UI components that ensure a consistent look and feel across the entire app, especially for the **Bangla-first UI**.

This design blueprint provides everything the development team needs to begin building the SmartSheba application.

Based on our finalized design, we are ready for the next phase: **Implementation and Development**.

This is the phase where all the planning and blueprints come to life. The development team will now write the actual code for the SmartSheba application, meticulously following the high-level and low-level designs we created.

**1. Backend Development**

The backend is the core of our system, handling all the logic and data.

* **Server Environment Setup:** The team will set up the server infrastructure on a cloud provider, including the database and necessary services.
* **API Development:** The RESTful APIs designed in the previous phase will be built. This includes:
  + **Authentication & User Management:** Developing secure APIs for user registration, login, profile management, and role-based access control.
  + **Service & Booking Logic:** Writing the code for handling service discovery, booking creation, status updates (e.g., accepted, completed), and the recurring booking feature.
  + **Payment Integration:** Integrating with the chosen local payment gateways to process digital payments securely and handle cash-on-delivery transactions.
  + **Location & Tracking:** Implementing the APIs to receive and broadcast real-time GPS data for live tracking.
* **API Documentation:** We will use tools like Swagger or Postman to create clear and up-to-date documentation for all APIs, making it easier for the mobile app developers to consume them.

**2. Frontend (Mobile App) Development**

The user-facing part of the application will be built, ensuring a smooth and intuitive experience.

* **UI Component Creation:** The team will begin by creating all the reusable widgets from the design system, such as ServiceCard, RatingWidget, and CustomButton, ensuring consistency and speeding up the process.
* **Screen Development:** All the screens for both the **Customer App** and **Provider App** will be built, including the:
  + **Home Screen:** With service categories and search functionality.
  + **Service Details Page:** To display provider profiles, ratings, and pricing.
  + **Booking Flow:** A multi-step process from service selection to payment confirmation.
  + **Tracking Page:** To show the real-time location of the provider.
  + **SOS Screen:** A simple interface for activating the emergency feature.
* **API Integration:** The mobile apps will be connected to the backend APIs to send and receive data, ensuring all features are fully functional. We will use the providers module for state management to handle data flow efficiently.

**3. AI/ML Service Development**

The specialized AI features will be developed concurrently with the main app.

* **Model Development & Training:** The AI models for the chatbot, image recognition, and fraud detection will be trained using relevant datasets.
* **API Exposure:** Each AI service will be wrapped in its own API to allow the main backend to send requests and receive predictions. For example, the backend will send an image to the image recognition API and receive the predicted service category.

**4. Best Practices & Collaboration**

As a senior developer, I will ensure these practices are followed throughout the implementation:

* **Version Control:** All code will be managed using Git, with a clear branching strategy to support collaborative development.
* **Code Reviews:** Every piece of code will be reviewed by at least one other developer to maintain high quality and catch potential bugs early.
* **Modular & Clean Code:** We will adhere to the folder structure and class designs we agreed upon, ensuring the code is organized, readable, and easy to maintain.

With the implementation underway, the codebase will grow rapidly. The next crucial step is to ensure that everything works as intended.

Excellent. After the intensive **Implementation** phase, we now have a working application. The next crucial step is to ensure that it functions flawlessly and securely. This brings us to the **Testing** phase.

From a senior system designer's perspective, testing isn't just about finding bugs; it's about validating that the entire system works as designed and meets all the quality and security requirements we set in the first phase.

**1. Unit Testing**

This is the most granular level of testing. Developers will test individual, isolated components or functions of the code.

* **Purpose:** To ensure that each part of the code works correctly on its own.
* **SmartSheba Example:** Testing a function that calculates the estimated service time (AI-based Price & Time Estimation), or a module that validates a user's phone number during registration.

**2. Integration Testing**

This type of testing verifies that different modules or services communicate and work together as a single unit.

* **Purpose:** To catch bugs in the communication between components.
* **SmartSheba Example:** Testing the complete booking flow, from a customer selecting a service on the app, to the request being sent to the backend API, and a notification being delivered to the correct provider. We will also test the seamless integration with the chosen payment gateway.

**3. System Testing**

Here, we test the entire, integrated system from end to end, simulating real-world scenarios.

* **Purpose:** To verify that the complete system meets all the functional and non-functional requirements.
* **SmartSheba Example:** A full end-to-end test of the **SOS Button** feature. We would trigger it on a test device and verify that notifications are sent correctly to pre-selected contacts and emergency services, and that location data is accurately transmitted.

**4. User Acceptance Testing (UAT)**

This is where the product is tested by the intended end-users.

* **Purpose:** To ensure the app is intuitive, user-friendly, and meets the real-world needs of the customers and providers.
* **SmartSheba Example:** We will recruit a group of beta testers, including potential customers and service providers in Bangladesh. They will use the app to perform everyday tasks (e.g., booking a tutor, creating a provider profile) and provide feedback on the **Bangla-first UI** and overall user experience.

**5. Performance & Load Testing**

This testing is crucial for ensuring the scalability and stability of the platform.

* **Purpose:** To determine how the app performs under extreme conditions, such as high traffic.
* **SmartSheba Example:** Simulating thousands of concurrent users trying to book services or track providers. We will monitor the server response times and resource usage to ensure the system remains fast and reliable.

**6. Security Testing**

We will systematically test the application for any vulnerabilities.

* **Purpose:** To protect user data and prevent malicious attacks.
* **SmartSheba Example:** We will perform tests to check for common vulnerabilities like SQL injection, cross-site scripting (XSS), and data breaches. We will also verify that the provider verification process is secure and cannot be easily bypassed.

**7. AI/ML Model Testing**

This is a unique and vital part of our process.

* **Purpose:** To validate the accuracy and reliability of our AI features.
* **SmartSheba Example:** We will feed various test queries to the **AI Chatbot** to ensure it provides accurate service suggestions. We will also test the **Image Recognition** feature with a diverse set of images to confirm it correctly classifies problems like leaky pipes or broken electronics.

Once all these tests are successfully completed, we can be confident that the SmartSheba application is ready for the public.