Cab Aggregator App with IVR & Offline Sync - Development Design

This document outlines the design and implementation of the Cab Aggregator App with offline booking support via IVR (Interactive Voice Response). The system ensures users can book rides even without internet, while syncing seamlessly once online.

# 1. System Architecture

- Mobile App (Flutter, Android/iOS)  
- Backend (FastAPI on Replit)  
- Database (SQLite for MVP)  
- IVR Integration (Twilio Voice API)  
- AI/NLP Layer (for voice commands, pricing, and DigiPin location resolution)  
  
The system ensures both online app bookings and offline IVR bookings flow into the same backend, providing a unified ride history.

# 2. Workflows

## 2.1 Online Booking (App)

1. User searches for cabs via `/search` API.  
2. Backend finds nearest driver (mock in MVP).  
3. User confirms and `/book` is called.  
4. Booking stored in DB with channel='APP'.  
5. App fetches booking via `/history/{phone}`.

## 2.2 Offline Booking (IVR)

1. User calls Twilio-powered IVR number.  
2. IVR captures user input (DTMF or speech → text).  
3. Backend `/ivr` endpoint processes booking.  
4. Booking stored in DB with channel='IVR'.  
5. IVR confirms booking to user by voice + SMS.  
6. When user comes online, `/history/{phone}` shows same ride in app.

## 2.3 Syncing IVR & App

- All bookings (APP/IVR) share a unified `booking\_id` format.  
- History API fetches both seamlessly.  
- App UI tags IVR rides as 'Booked via Call'.  
- Driver updates/cancellations flow back when user is online.

# 3. Backend APIs

- `/search` (POST): Returns nearby driver & fare.  
- `/book` (POST): Confirms ride & stores booking.  
- `/history/{phone}` (GET): Returns booking history across APP + IVR.  
- `/ivr` (POST): Handles Twilio IVR booking requests.

# 4. Replit Setup

1. Create new Replit (Python + FastAPI).  
2. Install dependencies: `pip install fastapi uvicorn twilio sqlalchemy sqlite3`.  
3. Add `main.py` with backend code provided.  
4. Configure Twilio webhook to point to `/ivr` endpoint.  
5. Run backend with `uvicorn main:app --host 0.0.0.0 --port 8000`.

# 5. Flutter App Integration

- App calls `/search` and `/book` for online rides.  
- At startup, app fetches `/history/{phone}`.  
- Offline IVR rides show up automatically with 'Booked via Call' label.  
- Seamless ride history = one platform experience.

# 6. Future Enhancements

- AI pricing engine (city/area-aware dynamic fares).  
- DigiPin integration for accurate offline location tracking.  
- Multi-lingual IVR (English + local languages).  
- Integration with Uber/Ola/Rapido real APIs.  
- Call masking & privacy protection for local drivers.