1. Feature List (Mapped to Problem Statement + DB)

A. Authentication & Roles (Platform-Level – Mandatory)

Tech: Supabase Auth + FastAPI role checks

Features

- User signup / login (email + password)
- Role-based access:
  - `customer`
  - `provider`
  - `admin`
- Auto-create `profiles` row on signup
- Route protection:
  - Customers cannot access provider/admin routes
  - Providers cannot access admin routes

DB Used

- `profiles`
- `user_role` enum

B. Customer Features

1. Browse & Search Providers (Core Feature)

Hyperlocal requirement – CRITICAL

Features

- Browse providers by:
  - Category
  - Location (nearby providers)
- Distance-based sorting (nearest first)
- Filter by:
  - Category
  - Minimum rating
  - Price range (optional)

Implementation

- PostGIS `ST_DWithin` / `ST_Distance`
- Uses `provider_profiles.location`
- Indexed with `GIST`

DB Used

- `categories`
- `provider_profiles`
- `provider_location_idx`

---

2. Provider Profile View

Features

- View provider details:
  - Name
  - Category
  - Bio
  - Base price
  - Average rating
  - Verification status
- Show availability (weekly schedule)
- Show reviews

DB Used

- `profiles`
- `provider_profiles`
- `provider_availability`
- `reviews`

---

3. Service Booking

Features

- Submit service request:
  - Select date & time
  - Enter service address
  - Notes (optional)
- Booking lifecycle:
  - `pending → confirmed → completed / cancelled`
  - `rescheduled` supported

Rules

- Prevent booking outside provider availability
- Prevent double booking for same time

DB Used

- `bookings`
- `booking_status` enum

---

4. Booking Tracking

Features

- View all bookings
- See status updates
- Cancel booking (before confirmation or policy-defined)

---

5. Reviews & Ratings

Features

- Leave review **only after completed booking**
- One review per booking
- Update provider average rating

DB Used

- `reviews`
- `bookings`
- `provider_profiles.avg_rating`

C. Service Provider Features

1. Provider Profile Management

Features

- Create provider profile
- Select category
- Set base price
- Add bio

- Set geolocation (lat/long)

Admin approval required before visible

DB Used

- `provider_profiles`
- `categories`

---

2. Availability Management

Features

- Set weekly availability:
  - Day of week
  - Start time
  - End time
- Edit availability

DB Used

- `provider_availability`

---

3. Booking Management

Features

- View incoming requests
- Accept / reject bookings
- Reschedule booking
- View booking history

Rules

- Cannot accept booking outside availability
- Status transitions validated

D. Admin Features (Bonus but Strongly Recommended)

1. User & Provider Moderation

Features

- View all users
- Approve / suspend providers
- Remove providers

DB Used

- `profiles`
- `provider_profiles.is_verified`

---

2. Review Moderation

Features

- View reviews
- Remove inappropriate reviews

---

3. Platform Monitoring

Features

- View:
  - Total users
  - Total providers
  - Total bookings
- Simple metrics (counts only)

E. Non-Functional Features (Must Mention in Evaluation)

- Input validation (FastAPI + Pydantic)
- Error handling (401, 403, 404, 409)
- Scalable API structure
- Clean UI/UX
- Index-based fast location search
- Role-based authorization

---

2. API Feature Mapping (FastAPI)

Core API Modules

/auth
/users
/providers
/categories
/bookings
/reviews
/admin

Example Endpoints

POST /auth/signup
POST /auth/login

GET /categories

GET /providers/search
GET /providers/{id}

POST /providers/profile
PUT /providers/profile
POST /providers/availability

POST /bookings
GET /bookings/my
PUT /bookings/{id}/status

POST /reviews

GET /admin/users
PUT /admin/providers/{id}/verify

---

3. Project Structure (Recommended & Clean)

Backend (FastAPI)

```
backend/
├── app/
│   ├── main.py
│   ├── core/
```

```
│   │   ├── config.py
│   │   ├── security.py
│   │   └── database.py
│   │
│   ├── models/
│   │   ├── profiles.py
│   │   ├── providers.py
│   │   ├── bookings.py
│   │   └── reviews.py
│   │
│   ├── schemas/
│   │   ├── auth.py
│   │   ├── providers.py
│   │   ├── bookings.py
│   │   └── reviews.py
│   │
│   ├── api/
│   │   ├── auth.py
│   │   ├── providers.py
│   │   ├── bookings.py
│   │   ├── reviews.py
│   │   └── admin.py
│   │
│   ├── services/
│   │   ├── location.py
│   │   ├── booking_logic.py
│   │   └── ratings.py
│   │
│   └── utils/
│       ├── permissions.py
│       └── exceptions.py
│
└── requirements.txt
```

Frontend (React + Tailwind)

```
frontend/
├── src/
│   ├── pages/
│   │   ├── Home.jsx
│   │   ├── Search.jsx
│   │   ├── ProviderProfile.jsx
│   │   ├── Bookings.jsx
│   │   ├── Login.jsx
│   │   └── AdminDashboard.jsx
│   │
```

```
│   ├── components/
│   │   ├── ProviderCard.jsx
│   │   ├── BookingForm.jsx
│   │   ├── ReviewForm.jsx
│   │   └── Navbar.jsx
│   │
│   ├── api/
│   │   ├── auth.js
│   │   ├── providers.js
│   │   ├── bookings.js
│   │   └── reviews.js
│   │
│   ├── hooks/
│   │   ├── useAuth.js
│   │   └── useLocation.js
│   │
│   └── utils/
│       └── protectedRoute.jsx
│
└── tailwind.config.js
```

---

4. What Evaluators Will Look For (Important)

✔ Correct role separation
✔ Real hyperlocal search (PostGIS used)
✔ Booking workflow correctness
✔ Clean API structure
✔ Logical DB usage
✔ Trade-offs explained (e.g., simple metrics vs analytics)
