1. **Authentication Flow:**

/api/auth/

├── route.js (POST)

- Purpose: Handles user authentication and profile management

- Input: Firebase user data (uid, name, email, photoURL)

- Actions:

1. Creates/updates user in users table

2. Creates session in user\_sessions

- Used by: Navbar.js during login

/api/users/

├── [firebaseUid]/

└── route.js (GET)

- Purpose: Fetches user's database UUID using Firebase UID

- Input: Firebase UID (from URL parameter)

- Used by: Services.js before creating bookings

1. **Services Flow:**

/api/services/

├── route.js (GET)

- Purpose: Fetches available services

- Output: List of services with details

- Used by: Services.js component for display

/api/bookings/

├── route.js (POST)

- Purpose: Creates new booking and associated payment

- Input: {user\_id, service\_type, scheduled\_time, total\_cost, status}

- Actions:

1. Creates booking record

2. Creates associated payment record

- Used by: Services.js when "Book Now" clicked

1. **Logical Flow Diagram:**

User Login Flow:

Firebase Auth → /api/auth → users table → user\_sessions table

↓

Profile displayed in Navbar

Booking Flow:

Services Display → Book Now Click → Check Login Status

↓

If Logged In:

↓

Get User UUID (/api/users/[firebaseUid]) → Create Booking (/api/bookings)

↓

Create Payment Record

↓

Success Notification

1. **Database Relationships:**

users

↓ (1:many)

├── user\_sessions

├── bookings ─── (1:1) ─── payments

├── user\_addresses

├── user\_phones

└── user\_auth\_providers

service\_providers

↓ (1:many)

└── bookings

1. **Suggested Additional Routes (Not Yet Implemented):**

/api/bookings/

├── [userId]/

└── route.js (GET) - Get user's booking history

├── [bookingId]/

└── route.js (GET, PUT) - Get/Update booking details

/api/payments/

├── [paymentId]/

└── route.js (GET, PUT) - Get/Update payment status

/api/providers/

├── route.js (GET) - List available service providers

├── [serviceType]/

└── route.js (GET) - Get providers by service type

1. **Error Handling Flow:**

Each API Route

├── Try-Catch Block

├── Input Validation

├── Database Transaction

│ ├── Begin

│ ├── Operations

│ ├── Commit/Rollback

│ └── Release Connection

└── Error Response

1. **Authentication State Management:**

Firebase Auth State

↓

useAuth Hook

↓

Components (Navbar, Services)

↓

API Requests with User Context