Great! Since you already have **mobile number + OTP login** ready, your next steps should be focused on **role selection, backend setup, and printing flow**. Here’s a detailed roadmap:

**1. Role Selection**

* On login, ask: **“Are you Owner or User?”**
* Store the role in your app state (React Native Context or Redux) or locally.
* Depending on the role, load different screens:
  + **Owner** → Owner Dashboard
  + **User** → User Dashboard

**2. Owner Mode Setup**

1. **Start backend server** (Express) on owner device or cloud:
   * Create endpoints:
     + /upload-print-job → accepts PDFs from users
     + /verify-payment → checks payment status
     + /print-job → sends file to printer
2. **Generate QR code** for users to connect:
   * QR contains backend URL and a temporary token for security
3. **Printer Integration**:
   * USB printer → use Node.js package printer or system commands
   * Wi-Fi printer → send PDF via IPP or printer SDK
4. **Job Dashboard** (optional):
   * Show pending print jobs, payment status, and history

**3. User Mode Setup**

1. **QR Scanner**:
   * Scan owner’s QR code → retrieve server URL + token
2. **File Picker**:
   * Let user select PDF using react-native-document-picker
3. **Print Options & Price Calculation**:
   * Number of pages, B/W or color → calculate amount
4. **Payment Integration**:
   * UPI / Razorpay SDK
   * Send payment confirmation to backend
5. **Upload PDF**:
   * After payment, upload PDF to owner’s server
6. **Print Confirmation**:
   * Backend triggers printing and responds with success

**4. Backend (Express + MongoDB)**

* **PrintJob Schema**: userId, filename, filePath, pages, color, amount, paymentStatus, printed
* **Endpoints**:
  + POST /upload-print-job → receives file, stores temporarily, triggers printing
  + POST /verify-payment → confirms payment success
  + GET /jobs → optional, for owner dashboard

**5. Recommended Development Order**

1. **Owner backend endpoints** → test manually with Postman
2. **Owner dashboard** → start server, generate QR, connect to printer
3. **User QR scan + file selection** → connect to owner backend
4. **Payment integration** → Razorpay / UPI
5. **File upload → printing** → verify end-to-end
6. Optional: real-time updates, multi-user handling

If you want, I can **make a step-by-step diagram showing the flow from login → role selection → QR code → payment → printing**, so you know exactly what happens in each step.

Do you want me to make that diagram?

**The Big Picture: How It Works**

The process involves the owner's phone creating a temporary, local Wi-Fi network (using Wi-Fi Direct) and then encoding the network's name and password into the QR code. The customer's phone scans the code, gets the network details, and joins it. Once they are both on the same local network, they can share files directly.

**On the Owner's Phone (The "Host" 📱)**

1. **Create a Local Network:** When the owner wants to share, they press a "Receive Files" button. Your app calls a function from the Wi-Fi P2P library to create a new, private group. The owner's phone now acts like a mini Wi-Fi hotspot.
2. **Get Connection Details:** Once the group is successfully created, the library will provide the network details:
   * **SSID** (the network's name, e.g., DIRECT-xy-Android\_1234)
   * **Passphrase** (the network's password, e.g., aB1cD2eF)
   * **Host IP Address** (the owner's IP address on this new network, usually 192.168.49.1)
3. **Encode Details into QR Code:** You'll combine these three pieces of information into a structured format, like a JSON string:

JSON

{

"ssid": "DIRECT-xy-Android\_1234",

"passphrase": "aB1cD2eF",

"ip": "192.168.49.1"

}

1. **Display the QR Code:** Your app takes this JSON string and uses a library like **react-native-qrcode-svg** to display it on the owner's screen.

**On the Customer's Phone (The "Client" 📲)**

1. **Scan the QR Code:** The customer taps a "Send File" or "Scan to Connect" button. Your app opens the camera using a library like **react-native-qrcode-scanner**.
2. **Decode the Information:** The scanner reads the QR code and returns the JSON string. Your app immediately parses this string to get the ssid, passphrase, and the owner's ip.
3. **Join the Local Network:** Your app now uses the Wi-Fi P2P library to connect to the network using the ssid and passphrase it just obtained from the QR code.
4. **Connection Established!** 🤝 After a few seconds, the customer's phone will be connected to the owner's private Wi-Fi Direct network.