Healthcare Reception Verification System

Step-by-Step Workflow

Step 1: Receptionist Loads Appointments

- The **Delphi Desktop App** retrieves **appointment data** from the **Web API** (C#/ASP.NET Core).
- The **Web API** fetches relevant records from the **MS-SQL Database** and sends them back.
- The Delphi App **displays appointments** grouped by status (e.g., Pending, Confirming, Confirmed, Not Confirmed, Cancelled, Completed).

✅ **Best Practice:** The system ensures that only the **required data** is retrieved for efficiency.

Step 2: Receptionist Selects a Patient for Verification

- The receptionist selects an appointment in the **Delphi Desktop App**.
- The app **retrieves available kiosks** from the Web API.
- The receptionist **chooses a kiosk** and clicks **"Start Verification."**

✅ **Best Practice:** The system prevents duplicate verification requests by **checking kiosk availability** first.

Step 3: Delphi App Sends Verification Request via WebSocket

- The **Delphi App** sends a verification request **to the WebSocket Server**.
- The **WebSocket Server** forwards the request to the **Kiosk App** assigned for verification.
- The **Delphi App updates the appointment's status** to **"In Progress"**.

✅ **Best Practice:** Using **WebSockets ensures real-time, event-driven communication**, preventing delays.

Step 4: Patient Completes Verification on Kiosk

- The **Kiosk App (React.js)** receives the request and **displays verification questions**.
- The patient **responds using the touchscreen interface**.
- Once completed, the Kiosk App **sends the verification result** back to the **WebSocket Server**.

✅ **Best Practice:** The UI is designed for **touchscreen usability**, improving **patient experience**.

Step 5: WebSocket Server Notifies Delphi App

- The **WebSocket Server** receives the verification result from the **Kiosk App**.
- It **forwards the result to the Delphi App** handling the patient's appointment.
- The Delphi App **updates the UI** to reflect the verification outcome.

✅ **Best Practice: Instant notifications eliminate polling delays, improving system responsiveness.**

Step 6: Delphi App Updates the Database

- The **Delphi App** updates the patient’s appointment status (e.g., "Confirmed") by sending a request to the **Web API**.
- The **Web API** commits the update to the **MS-SQL Database**.
- The **appointment status is updated** across all active **Delphi Apps** through WebSocket notifications.

✅ **Best Practice:** A **centralized database with controlled access** ensures data consistency and security.

Step 7: Receptionist Proceeds with Check-In

- The receptionist **sees the updated verification status** in the **Delphi App**.
- If successful, the receptionist **proceeds with the patient check-in process**.
- If verification **fails**, the receptionist may manually verify the patient's identity.

✅ **Best Practice:** The **status-based UI design** helps the receptionist **quickly make informed decisions**.

Key Features and Benefits

- **Event-Driven Architecture:** Eliminates delays and **ensures real-time updates**.
- **Decoupled & Scalable:** Modules interact **via APIs and WebSockets**, making the system easy to extend.
- **User Experience Focused:** Both receptionists and patients have a **smooth, guided experience**.
- **Security & Data Integrity:** Only the **Web API accesses the database**, enforcing **strict validation rules**.