Absolutely! Let’s design a **user flow** and **technical implementation plan** for the **walk-in customer experience** in **Crowd Nest**. This will ensure a seamless and efficient process for both customers and restaurant staff.

**User Flow for Walk-In Customers**

**1. Customer Arrives at the Restaurant**

* **Action**: Customer approaches the restaurant entrance.
* **System**: A digital screen or tablet displays real-time crowd status (e.g., "20/50 seats occupied").
* **Customer Decision**: Decides whether to enter or wait based on the crowd status.

**2. Customer Joins the Waitlist**

* **Action**: Customer scans a QR code or uses a kiosk to join the waitlist.
* **Input**:
  + Name.
  + Phone number.
  + Party size.
* **System**:
  + Adds the customer to the digital waitlist.
  + Sends a confirmation SMS with an estimated wait time (e.g., "You are 5th in line. Estimated wait time: 15 minutes").

**3. Customer Waits Nearby**

* **Action**: Customer waits nearby (e.g., in a waiting area or a nearby café).
* **System**:
  + Sends periodic updates via SMS (e.g., "Your table will be ready in 5 minutes").
  + Allows the customer to check their wait status by scanning the QR code again.

**4. Table is Ready**

* **Action**: System notifies the customer when a table is available.
* **Notification**:
  + SMS: "Your table is ready! Please return within 5 minutes."
  + App notification (if the customer has the app installed).
* **System**:
  + Updates the crowd status (e.g., "25/50 seats occupied").
  + Assigns the table to the customer.

**5. Customer is Seated**

* **Action**: Customer returns to the restaurant and is seated.
* **System**:
  + Marks the table as occupied in the POS system.
  + Updates the crowd status in real-time.

**6. Post-Visit Feedback**

* **Action**: After the meal, the customer receives a feedback request.
* **System**:
  + Sends an SMS or email with a link to a quick survey.
  + Collects ratings and comments.
* **Customer Action**: Provides feedback (optional).

**Technical Implementation Plan**

**1. Real-Time Crowd Status Display**

* **Technology**: Use a **digital screen** or **tablet** connected to the backend.
* **Implementation**:
  + Develop an API endpoint to fetch real-time crowd status: GET /api/restaurants/:id/crowd-status.
  + Use **WebSocket** or **Firebase Realtime Database** for live updates.
  + Display the data on the screen using a simple frontend (e.g., React or plain HTML).

**2. Digital Waitlist System**

* **Technology**: Use a **QR code** or **kiosk** for customer input.
* **Implementation**:
  + Develop an API endpoint to add customers to the waitlist: POST /api/waitlist.

json

Copy

{

"restaurantId": "1",

"name": "John Doe",

"phone": "1234567890",

"partySize": 4

}

* + Store waitlist data in the database (e.g., MongoDB or PostgreSQL).
  + Use a **message queue** (e.g., RabbitMQ) to manage notifications.

**3. Notification System**

* **Technology**: Use **SMS APIs** (e.g., Twilio) or **push notifications** (e.g., Firebase Cloud Messaging).
* **Implementation**:
  + Develop an API endpoint to send notifications: POST /api/notifications.

json

Copy

{

"userId": "12345",

"message": "Your table is ready!",

"type": "waitlist"

}

* + Integrate with Twilio for SMS or Firebase for push notifications.

**4. Table Management**

* **Technology**: Integrate with the restaurant’s **POS system**.
* **Implementation**:
  + Develop an API endpoint to update table status: PUT /api/tables/:id.

json

Copy

{

"status": "occupied",

"customerId": "12345"

}

* + Sync with the POS system using its API or a middleware layer.

**5. Feedback Collection**

* **Technology**: Use an **email/SMS service** and a **survey tool** (e.g., Google Forms, Typeform).
* **Implementation**:
  + Develop an API endpoint to send feedback requests: POST /api/feedback.

json

Copy

{

"customerId": "12345",

"restaurantId": "1",

"message": "How was your experience?"

}

* + Integrate with a survey tool to collect and analyze feedback.

**Tools and Technologies**

1. **Frontend**:
   * **React Native** for the mobile app.
   * **HTML/CSS/JavaScript** for digital screens and kiosks.
2. **Backend**:
   * **Node.js** with **Express** for APIs.
   * **WebSocket** or **Firebase** for real-time updates.
3. **Database**:
   * **MongoDB** or **PostgreSQL** for storing data.
4. **Notifications**:
   * **Twilio** for SMS.
   * **Firebase Cloud Messaging** for push notifications.
5. **POS Integration**:
   * Use the POS system’s API or a middleware layer.
6. **Analytics**:
   * **Google Analytics** or **Mixpanel** for tracking user behavior.

**Example Code Snippet (Node.js)**

Here’s an example of how you could implement the POST /api/waitlist endpoint:

javascript

Copy

const express = require('express');

const router = express.Router();

const Waitlist = require('../models/Waitlist'); // MongoDB model

// Add customer to waitlist

router.post('/waitlist', async (req, res) => {

const { restaurantId, name, phone, partySize } = req.body;

try {

const newEntry = new Waitlist({ restaurantId, name, phone, partySize });

await newEntry.save();

// Send confirmation SMS

const message = `You are added to the waitlist. Estimated wait time: 15 minutes.`;

await sendSMS(phone, message); // Integrate with Twilio

res.status(201).json({ message: 'Added to waitlist successfully' });

} catch (error) {

res.status(500).json({ message: 'Error adding to waitlist', error });

}

});

// Function to send SMS

const sendSMS = async (phone, message) => {

// Integrate with Twilio API here

};

module.exports = router;

**Next Steps**

1. **Wireframe the User Interface**: Design the screens for the kiosk, digital display, and mobile app.
2. **Develop the Backend**: Build the APIs and integrate with the POS system.
3. **Set Up Notifications**: Integrate Twilio and Firebase for SMS and push notifications.
4. **Test the System**: Conduct user testing with real customers and restaurant staff.
5. **Deploy and Monitor**: Launch the system and monitor performance using analytics.