Absolutely! Let’s create a **Node.js API** for the **walk-in customer experience** in **Crowd Nest**. I’ll provide a complete implementation for the key endpoints we discussed earlier. This will include:

1. **Adding a customer to the waitlist**.
2. **Fetching real-time crowd status**.
3. **Sending notifications**.
4. **Updating table status**.

**Setup**

Before we start, ensure you have the following installed:

* **Node.js** and **npm**.
* **Express.js** for building the API.
* **Mongoose** for MongoDB interaction.
* **Twilio** for SMS notifications (optional).
* **WebSocket** or **Firebase** for real-time updates (optional).

**API Code Implementation**

**1. Project Setup**

Create a new Node.js project and install dependencies:

bash

Copy

mkdir crowd-nest-api

cd crowd-nest-api

npm init -y

npm install express mongoose body-parser twilio

**2. Folder Structure**

Copy

crowd-nest-api/

├── models/

│ ├── Waitlist.js

│ ├── Restaurant.js

│ └── Table.js

├── routes/

│ ├── waitlist.js

│ ├── restaurant.js

│ └── table.js

├── utils/

│ └── notifications.js

├── app.js

└── server.js

**3. MongoDB Models**

**Waitlist Model (models/Waitlist.js)**

javascript

Copy

const mongoose = require('mongoose');

const waitlistSchema = new mongoose.Schema({

restaurantId: { type: mongoose.Schema.Types.ObjectId, ref: 'Restaurant', required: true },

name: { type: String, required: true },

phone: { type: String, required: true },

partySize: { type: Number, required: true },

createdAt: { type: Date, default: Date.now }

});

module.exports = mongoose.model('Waitlist', waitlistSchema);

**Restaurant Model (models/Restaurant.js)**

javascript

Copy

const mongoose = require('mongoose');

const restaurantSchema = new mongoose.Schema({

name: { type: String, required: true },

totalSeats: { type: Number, required: true },

occupiedSeats: { type: Number, default: 0 }

});

module.exports = mongoose.model('Restaurant', restaurantSchema);

**Table Model (models/Table.js)**

javascript

Copy

const mongoose = require('mongoose');

const tableSchema = new mongoose.Schema({

restaurantId: { type: mongoose.Schema.Types.ObjectId, ref: 'Restaurant', required: true },

status: { type: String, enum: ['available', 'occupied'], default: 'available' },

customerId: { type: String }

});

module.exports = mongoose.model('Table', tableSchema);

**4. API Routes**

**Waitlist Routes (routes/waitlist.js)**

javascript

Copy

const express = require('express');

const router = express.Router();

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

const { sendSMS } = require('../utils/notifications');

// Add customer to waitlist

router.post('/', 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);

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

} catch (error) {

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

}

});

module.exports = router;

**Restaurant Routes (routes/restaurant.js)**

javascript

Copy

const express = require('express');

const router = express.Router();

const Restaurant = require('../models/Restaurant');

// Get real-time crowd status

router.get('/:id/crowd-status', async (req, res) => {

const { id } = req.params;

try {

const restaurant = await Restaurant.findById(id);

if (!restaurant) {

return res.status(404).json({ message: 'Restaurant not found' });

}

res.status(200).json({

totalSeats: restaurant.totalSeats,

occupiedSeats: restaurant.occupiedSeats,

crowdStatus: restaurant.occupiedSeats >= restaurant.totalSeats ? 'Full' : 'Available'

});

} catch (error) {

res.status(500).json({ message: 'Error fetching crowd status', error });

}

});

module.exports = router;

**Table Routes (routes/table.js)**

javascript

Copy

const express = require('express');

const router = express.Router();

const Table = require('../models/Table');

// Update table status

router.put('/:id', async (req, res) => {

const { id } = req.params;

const { status, customerId } = req.body;

try {

const table = await Table.findByIdAndUpdate(id, { status, customerId }, { new: true });

if (!table) {

return res.status(404).json({ message: 'Table not found' });

}

res.status(200).json({ message: 'Table status updated successfully', data: table });

} catch (error) {

res.status(500).json({ message: 'Error updating table status', error });

}

});

module.exports = router;

**5. Notification Utility (utils/notifications.js)**

javascript

Copy

const twilio = require('twilio');

const accountSid = 'your\_twilio\_account\_sid';

const authToken = 'your\_twilio\_auth\_token';

const client = new twilio(accountSid, authToken);

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

try {

await client.messages.create({

body: message,

to: phone,

from: 'your\_twilio\_phone\_number'

});

console.log(`SMS sent to ${phone}`);

} catch (error) {

console.error('Error sending SMS:', error);

}

};

module.exports = { sendSMS };

**6. App Configuration (app.js)**

javascript

Copy

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const waitlistRoutes = require('./routes/waitlist');

const restaurantRoutes = require('./routes/restaurant');

const tableRoutes = require('./routes/table');

const app = express();

// Middleware

app.use(bodyParser.json());

// Routes

app.use('/api/waitlist', waitlistRoutes);

app.use('/api/restaurants', restaurantRoutes);

app.use('/api/tables', tableRoutes);

// Database connection

mongoose.connect('mongodb://localhost:27017/crowdnest', {

useNewUrlParser: true,

useUnifiedTopology: true

})

.then(() => console.log('Connected to MongoDB'))

.catch(err => console.error('MongoDB connection error:', err));

module.exports = app;

**7. Server Setup (server.js)**

javascript

Copy

const app = require('./app');

const PORT = process.env.PORT || 5000;

app.listen(PORT, () => {

console.log(`Server running on port ${PORT}`);

});

**Running the API**

1. Start MongoDB locally or use a cloud service like **MongoDB Atlas**.
2. Run the server:

bash

Copy

node server.js

1. Test the API using **Postman** or **cURL**.

**Example API Requests**

**Add to Waitlist**

bash

Copy

POST /api/waitlist

Body:

{

"restaurantId": "64f1a2b3c9e77b001f8e4f1a",

"name": "John Doe",

"phone": "+1234567890",

"partySize": 4

}

**Get Crowd Status**

bash

Copy

GET /api/restaurants/64f1a2b3c9e77b001f8e4f1a/crowd-status

**Update Table Status**

bash

Copy

PUT /api/tables/64f1a2b3c9e77b001f8e4f1b

Body:

{

"status": "occupied",

"customerId": "12345"

}

This API implementation covers the core functionality for walk-in customers. Let me know if you’d like to expand on any part or add more features! 🚀