Based on the wireframes and features described for **Crowd Nest**, here’s a breakdown of the **APIs** you’ll need to develop using **Node.js**. These APIs will handle communication between the frontend (React Native app) and the backend (Node.js server). I’ve categorized them by functionality:

**1. Authentication APIs**

* **POST /api/auth/register**: Register a new user (customer or restaurant owner).
* **POST /api/auth/login**: Log in a user and return a JWT token.
* **POST /api/auth/logout**: Log out a user (invalidate the token).
* **POST /api/auth/forgot-password**: Handle password reset requests.
* **POST /api/auth/reset-password**: Reset the user’s password.

**2. Restaurant APIs**

* **GET /api/restaurants**: Fetch a list of restaurants with filters (e.g., location, cuisine, crowd status).
* **GET /api/restaurants/:id**: Fetch details of a specific restaurant.
* **POST /api/restaurants**: Create a new restaurant profile (for owners).
* **PUT /api/restaurants/:id**: Update restaurant details (e.g., seating capacity, open/closed status).
* **DELETE /api/restaurants/:id**: Delete a restaurant profile.

**3. Crowd Management APIs**

* **GET /api/restaurants/:id /crowd-status**: Fetch real-time crowd status for a restaurant.
* **PUT /api/restaurants/:id /crowd-status**: Update the crowd status (e.g., occupied seats).

**4. Reservation APIs**

* **GET /api/reservations**: Fetch all reservations for a customer or restaurant.
* **GET /api/reservations/:id**: Fetch details of a specific reservation.
* **POST /api/reservations**: Create a new reservation.
* **PUT /api/reservations/:id**: Update a reservation (e.g., change time or number of guests).
* **DELETE /api/reservations/:id**: Cancel a reservation.

**5. Waitlist Management APIs**

* **GET /api/waitlist**: Fetch the current waitlist for a restaurant.
* **POST /api/waitlist**: Add a customer to the waitlist.
* **DELETE /api/waitlist/:id**: Remove a customer from the waitlist.

**6. Notification APIs**

* **POST /api/notifications**: Send notifications to users (e.g., reservation confirmation, table ready).
* **GET /api/notifications**: Fetch notifications for a user.

**7. Analytics APIs**

* **GET /api/analytics/reservations**: Fetch reservation trends and statistics for a restaurant.
* **GET /api/analytics/crowd**: Fetch crowd trends (e.g., peak hours, average occupancy).
* **GET /api/analytics/ratings**: Fetch customer ratings and feedback.

**8. User Profile APIs**

* **GET /api/users/:id**: Fetch user profile details.
* **PUT /api/users/:id**: Update user profile details.
* **DELETE /api/users/:id**: Delete a user account.

**9. Search and Filter APIs**

* **GET /api/search**: Search for restaurants by name, location, or cuisine.
* **GET /api/filter**: Filter restaurants by criteria (e.g., open now, low crowd).

**10. Miscellaneous APIs**

* **GET /api/locations**: Fetch available locations for filtering.
* **GET /api/cuisines**: Fetch available cuisine types for filtering.

**Example API Workflow**

Here’s an example of how the APIs would work together for a reservation flow:

1. **Customer searches for restaurants**: GET /api/restaurants (with filters).
2. **Customer views restaurant details**: GET /api/restaurants/:id.
3. **Customer checks crowd status**: GET /api/restaurants/:id/crowd-status.
4. **Customer creates a reservation**: POST /api/reservations.
5. **Restaurant owner updates crowd status**: PUT /api/restaurants/:id/crowd-status.
6. **Customer receives a notification**: POST /api/notifications.

**Tools for API Development**

1. **Express.js**: For building the RESTful APIs.
2. **Mongoose**: For interacting with MongoDB (if using a NoSQL database).
3. **JWT**: For authentication and authorization.
4. **WebSocket**: For real-time updates (e.g., crowd status).
5. **Swagger**: For API documentation.