Food Management System – Full Project Explanation

# Overview

This project is a full-stack food donation and management platform. It allows users to donate food, find hunger spots, reserve meals, and manage their profiles. The stack includes:

Frontend: Next.js (React), Tailwind CSS, Material UI, Google Maps API

Backend: Node.js (the backend is currently a placeholder and not implemented)

# Frontend Logic

## 1. App Layout and Structure

The main layout is defined in src/app/layout.tsx:

export default function RootLayout({ children }) {  
 return (  
 <html lang="en">  
 <body className={`${geistSans.variable} ${geistMono.variable} antialiased`}>  
 {children}  
 </body>  
 </html>  
 );  
}

Purpose: Sets up global fonts and wraps all pages with a consistent layout.

## 2. Home Page (src/app/home/page.tsx)

Displays: Welcome message, top donators, nearby donations, and a footer navigation.

Logic: Uses static arrays for donator info and top donators, and renders them with reusable components.

Key Snippet:

const DonaterInfo = [ ... ];  
const topDonaters = [ ... ];  
return (  
 <div className='main-container'>  
 {/\* ...header, top donators, nearby donations... \*/}  
 <div className='donater-container'>  
 {DonaterInfo.map((donater) => <Donater key={uuid()} {...donater} />)}  
 </div>  
 <Footer />  
 </div>  
)

## 3. Donating Food (src/app/donate-food/page.tsx)

Form: Users can enter meal details, upload an image, and confirm donation.

State Management: Uses React useState for form fields and image preview.

Navigation: Uses Next.js router to navigate after submission.

Key Snippet:

const [mealName, setMealName] = useState('');  
const [mealQuantity, setMealQuantity] = useState(0);  
// ...other states...  
  
const handleConfirmDonation = () => {  
 // Here you would send data to the backend  
 router.push('/donation-confirmation');  
};

## 4. Hunger Spot Map (src/app/hungerspot/MapView.tsx)

Google Maps Integration: Uses @react-google-maps/api to display restaurants as markers.

Search: Users can search for restaurants using Google Places API.

Reservation: Clicking a marker opens a modal to reserve a meal.

Key Snippet:

<GoogleMap ...>  
 {defaultMarkers.map((marker) => (  
 <Marker position={{ lat: marker.lat, lng: marker.lng }} onClick={() => handleMarkerClick(marker)} />  
 ))}  
</GoogleMap>  
{showModal && activeMarker && (  
 <div>  
 {/\* Modal with reservation button \*/}  
 <button onClick={handleReservation}>Reserve your meal</button>  
 </div>  
)}

## 5. Components

Header/Footer: Consistent navigation and branding.

Donater: Displays info about a donator and their meals.

Dock: Custom bottom navigation bar.

SearchBar: Integrated with Google Places for restaurant search.

Example: Donater Component

const Donater: React.FC<DonaterProps> = ({ name, date, location, noofmeal, images }) => (  
 <div>  
 <h3>{name}</h3>  
 <div>{date} - {location}</div>  
 <div>{noofmeal} meal donated</div>  
 {/\* Images and action buttons \*/}  
 </div>  
)

## 6. UI Utilities

Button/Input: Custom styled components using Tailwind and utility functions.

Utils: cn function for merging class names.

# Backend Logic

Note: The backend (server/index.js) is currently empty. The project is set up for a Node.js backend, but all business logic and API endpoints are yet to be implemented.

Planned Structure:

index.js: Will initialize the server, set up middleware (body parsing, CORS), and define API routes.

routes/: (To be created) Will contain route handlers for food donations, reservations, user management, etc.

# Data Flow

User Interaction: Users interact with forms and maps on the frontend.

API Calls: (Planned) Data from forms (e.g., food donation) will be sent to backend endpoints.

State Management: React state is used for UI interactivity; Next.js router handles navigation.

Google Maps/Places: Used for searching and displaying restaurant locations.

## Example: Restaurant Search Flow

1. User types a restaurant name in the search bar.

2. searchBar.tsx uses Google Places API to fetch matching restaurants.

3. Results are displayed; user selects one.

4. Map centers on the selected restaurant and opens a modal for reservation.

# How to Run

Install dependencies:

cd .\Hackathon\_code\client\hackproject  
npm install

Start the frontend:

npm run dev

(Planned) Start the backend:

cd ..\..\server  
npm install  
node index.js

# Conclusion

Frontend: Fully functional with modern UI, Google Maps integration, and interactive forms.

Backend: Placeholder, ready for API and database logic.

Extensible: Easy to add backend endpoints and connect to a database for full-stack functionality.