**Chronological Frontend Design Roadmap for Integrating Information into Application**

**Phase 1: Initial Setup and Planning**

1. **Review the Existing Backend Setup:**
   * Understand the current backend configuration, controllers, models, and routes.
   * Ensure all backend functionalities required for the frontend are available.
2. **Project Initialization:**
   * Set up the frontend project structure in the Nour2.0/frontend directory.
   * Install necessary dependencies and tools (React, Redux, React Router, Styled Components, etc.).

**Phase 2: Component Design and Development**

1. **Design System and Component Library:**
   * Create a design system with the defined color schemes, typography, icons, and spacing.
   * Develop a component library with reusable UI components as described:
     + Buttons, Cards, Form Elements, Navigation, Modals, Tables, Notifications, Charts, Lists, and Miscellaneous components.
2. **Implement Key Components:**
   * **Header:** Logo, user profile icon, notification bell, and search bar.
   * **Footer:** Links to support, user guides, terms of service, social media links, and latest blog posts or updates.

**Phase 3: User Authentication & Data Gathering**

1. **User Registration and Login:**
   * **Registration:**
     + Welcome screen with an animated message and "Get Started" button.
     + Step-by-step registration form with animated transitions and real-time validation.
     + Confirmation screen with a success message and smooth transition to the login screen.
     + Initial data points gathering, first / last name, email, DoB, password, height, weight, gender,
   * **Login:**
     + Simple login form with animated input fields, social login options, and a "Forgot Password" link.
     + Interactive login button with a loading animation upon submission and feedback for incorrect credentials.
     + Smooth transition to the dashboard upon successful login.
   * **Data Gathering:**
     + Connect users wearable for all the wearable datapoints
     + DoctorAI Chat function, STT & TTS model that interacts with the user like a initial doctors intro consultant session, gathering as much data about the user as possible. 10 questions to gather a lot of information about the user

**Phase 4: Core Application Pages**

1. **Home/Dashboard Page:**
   * **Today's Summary:** Interactive cards displaying key metrics with progress bars.
   * **Quick Links:** Vertical list of clickable cards with hover effects and tooltips.
   * **Notifications:** Scrollable section of recent notifications with filter options.
2. **Nutrition (Meal Plans) Page:**
   * **Current Meal Plan:** Expandable cards for each meal with detailed nutritional information and action buttons.
   * **Sidebar:** Quick links to nutritional recommendations, food diary, and collapsible sections for additional resources.
3. **Inventory Page:**
   * **Inventory List:** Searchable and sortable table with filter options, alerts, and inline action buttons.
   * **Sidebar:** Quick links to recipe suggestions, expiry notifications, inventory statistics, and collapsible sections for additional resources.

**Phase 5: Interactive Prototypes**

1. **Meal Plan Customization:**
   * **Accessing Meal Plans:**
     + Clickable card on the dashboard with a hover effect and preview of today’s meal plan.
     + Transition to the meal plan page with a fade-in effect and calendar view for weekly meal plans.
   * **Customizing Meals:**
     + Expandable meal cards with smooth animations and swipe gestures for mobile view.
     + Interactive buttons for customizing ingredients and portions with real-time preview of nutritional changes.
     + Save button with a confirmation animation and undo option for recent changes.
   * **Feedback:**
     + Success message with a smooth transition back to the meal overview and personalized recommendations for future meals.
2. **Inventory Management:**
   * **Viewing Inventory:**
     + Clickable card on the dashboard with a hover effect and quick stats for current inventory.
     + Transition to the inventory page with a slide-in effect and filter options for categories and expiry status.
   * **Managing Items:**
     + "Add Item" button triggers a modal with a slide-up animation and barcode scanning for quick entry.
     + Form fields with real-time validation and feedback, and suggestions for common items.
     + Inline edit buttons with hover effects, batch actions for managing multiple items, and real-time updates.
     + Alert messages with fade-in/out animations and snooze option for non-critical alerts.
     + Action buttons for quick fixes with smooth transitions and color-coded alerts for different priority levels.

**Phase 6: Integration and Testing**

1. **Backend Integration:**
   * Connect frontend components with backend APIs for dynamic data fetching and interactions.
   * Ensure seamless communication between frontend and backend for user authentication, data management, and other functionalities.
2. **State Management:**
   * Implement Redux for global state management.
   * Use React hooks for managing component-specific states.
3. **Accessibility and Responsiveness:**
   * Ensure all components and pages meet WCAG standards.
   * Implement responsive design techniques for a seamless user experience across devices.
4. **Testing and Quality Assurance:**
   * Write unit tests for each component using Jest and React Testing Library.
   * Perform end-to-end testing using tools like Cypress.
   * Ensure high test coverage and include tests for edge cases.

**Phase 7: Deployment and Documentation**

1. **Documentation:**
   * Document each component’s API, props, and usage examples using tools like Storybook.
   * Create a comprehensive user guide for the application.
2. **Deployment:**
   * Deploy the application using services like Netlify or Vercel.
   * Set up CI/CD pipelines for automated testing and deployment.

**Phase 8: Maintenance and Updates**

1. **Regular Updates:**
   * Continuously update the application based on user feedback and new requirements.
   * Maintain the codebase for performance, security, and usability improvements.

**Phase 1: Initial Setup and Planning**

**Review the Existing Backend Setup**

1. **Understand the Current Backend Configuration:**
   * **Configuration Files:**
     + config.json: Contains environment-specific configurations like database connection strings, API keys, and other settings.
     + database.js: Configures and initializes the database connection using the settings from config.json.
     + logger.js: Sets up logging mechanisms to track application events and errors.
     + passport.js: Configures authentication strategies using Passport.js for user authentication.
   * **Controllers:**
     + Controllers are responsible for handling requests and returning responses. Key controllers include:
       - authController.js: Manages user authentication processes such as login, registration, and password management.
       - inventoryController.js: Handles CRUD operations for inventory items.
       - mealController.js: Manages meal-related data and operations.
       - profileController.js: Deals with user profile information and updates.
       - recommendationController.js: Provides personalized recommendations to users.
   * **Models:**
     + Models define the structure of the data and interact with the database. Key models include:
       - User.js: Represents user data, including personal information, authentication details, and preferences.
       - InventoryItem.js: Represents items in the inventory, including details like name, quantity, and expiry date.
       - Meal.js: Represents meals, including nutritional information and ingredients.
       - UserPreferences.js: Captures user preferences for personalized recommendations.
   * **Routes:**
     + Routes define the endpoints for API requests and map them to the appropriate controllers. Key routes include:
       - authRoutes.js: Endpoints for authentication-related operations.
       - inventoryRoutes.js: Endpoints for inventory management.
       - mealRoutes.js: Endpoints for meal data operations.
       - profileRoutes.js: Endpoints for user profile operations.
       - recommendationRoutes.js: Endpoints for fetching personalized recommendations.
   * **Middlewares:**
     + Middlewares handle common tasks such as authentication and validation before passing the request to the controllers.
       - authMiddleware.js: Ensures that the user is authenticated before accessing certain routes.
2. **Ensure All Backend Functionalities Required for the Frontend Are Available:**
   * Verify that all necessary APIs are implemented and functioning correctly.
   * Ensure that the backend provides endpoints for all the functionalities required by the frontend, such as user authentication, data fetching, and updates.

**Project Initialization**

1. **Set Up the Frontend Project Structure in the Nour2.0/frontend Directory:**
   * **Directory Structure:**
     + Create a well-organized directory structure to manage components, pages, assets, and other resources effectively.
     + Example structure:

java

Copy code

Nour2.0/frontend/

├── public/

│ ├── index.html

│ └── favicon.ico

├── src/

│ ├── assets/

│ ├── components/

│ ├── pages/

│ ├── redux/

│ ├── services/

│ ├── styles/

│ ├── App.js

│ ├── index.js

│ └── setupTests.js

├── .gitignore

├── package.json

├── package-lock.json

└── README.md

1. **Install Necessary Dependencies and Tools:**
   * **Core Libraries:**
     + **React:** A JavaScript library for building user interfaces.
     + **React DOM:** Integrates React with the DOM.
   * **State Management:**
     + **Redux:** A state management library for JavaScript applications.
     + **React-Redux:** Official React bindings for Redux.
     + **Redux Thunk:** Middleware for Redux to handle asynchronous actions.
   * **Routing:**
     + **React Router:** A collection of navigational components for React applications.
   * **Styling:**
     + **Styled Components:** A library for writing CSS-in-JS.
     + **Tailwind CSS (optional):** A utility-first CSS framework.
   * **HTTP Client:**
     + **Axios:** A promise-based HTTP client for making API requests.
   * **Form Management:**
     + **Formik:** A library for building forms in React.
     + **Yup:** A schema builder for value parsing and validation.
   * **Testing:**
     + **Jest:** A JavaScript testing framework.
     + **React Testing Library:** A set of helpers for testing React components.
   * **Example of Dependencies to Install:**

bash

Copy code

npm install react react-dom redux react-redux redux-thunk react-router-dom styled-components axios formik yup jest @testing-library/react @testing-library/jest-dom

1. **Configuration and Setup:**
   * **Create React App:** Initialize the React application using Create React App for a quick setup.

bash

Copy code

npx create-react-app frontend

cd frontend

* + **Configure Redux:**
    - Set up the Redux store, reducers, and middleware.
  + **Configure Routing:**
    - Set up React Router for navigation between different pages.
  + **Global Styles and Theme:**
    - Create a global styles file and theme configuration using Styled Components or Tailwind CSS.
  + **Environment Variables:**
    - Set up environment variables for different configurations (e.g., API endpoints).

1. **Development Tools:**
   * **Code Quality and Formatting:**
     + Set up ESLint and Prettier for code linting and formatting.
   * **Version Control:**
     + Initialize Git for version control and create a .gitignore file to exclude unnecessary files.

By following these steps, you will have a solid foundation for your frontend project, ensuring a well-organized structure and all necessary tools and dependencies in place for efficient development.

### Monochromatic Design System

#### Color Palette

1. **Primary Colors**
   * **Black**: #000000
   * **Dark Gray**: #333333
   * **Gray**: #666666
   * **Light Gray**: #CCCCCC
   * **White**: #FFFFFF
2. **Accent Colors**
   * **Accent Dark Gray**: #444444
   * **Accent Light Gray**: #EEEEEE

#### Typography

1. **Primary Font**
   * **Font Family**: 'Helvetica Neue', Arial, sans-serif
   * **Font Weights**: 300 (Light), 400 (Regular), 500 (Medium), 700 (Bold)
2. **Font Sizes**
   * **Headings**
     + H1: 48px, Bold
     + H2: 36px, Medium
     + H3: 28px, Regular
     + H4: 24px, Regular
     + H5: 20px, Medium
     + H6: 16px, Medium
   * **Body Text**
     + Body1: 16px, Regular
     + Body2: 14px, Regular
   * **Caption**
     + Caption: 12px, Regular
   * **Button Text**
     + Button: 14px, Medium, Uppercase

#### Iconography

1. **Icon Style**
   * **Type**: Outline and Filled
   * **Line Weight**: 2px for Outline
   * **Size Variations**: 16px, 24px, 32px, 48px
2. **Icon Library**
   * Use a consistent icon library such as Material Icons or FontAwesome in a monochrome style.
3. **Common Icons**
   * **Navigation Icons**: Home, Menu, Settings, Profile
   * **Action Icons**: Add, Edit, Delete, Save
   * **Status Icons**: Success, Error, Warning, Info

#### Reusable UI Components

1. **Buttons**
   * **Primary Button**
     + Background: Black
     + Text Color: White
     + Hover: Dark Gray
     + Disabled: Light Gray
   * **Secondary Button**
     + Background: White
     + Border: Black
     + Text Color: Black
     + Hover: Light Gray
     + Disabled: Very Light Gray
   * **Icon Button**
     + Background: Transparent
     + Icon Color: Black
     + Hover: Light Gray
2. **Cards**
   * **Basic Card**
     + Background: White
     + Border: Light Gray
     + Shadow: Subtle shadow for elevation
     + Padding: 16px
   * **Image Card**
     + Image at the top, followed by Title and Description
     + Padding: 16px
     + Border: Light Gray
     + Shadow: Subtle shadow
3. **Form Elements**
   * **Input Field**
     + Border: Light Gray
     + Focus: Dark Gray border
     + Placeholder Text: Gray
   * **Select Dropdown**
     + Border: Light Gray
     + Focus: Dark Gray border
     + Dropdown Icon: Dark Gray
   * **Checkboxes and Radio Buttons**
     + Unchecked: Gray
     + Checked: Black
   * **Toggle Switch**
     + Off: Light Gray background, Gray toggle
     + On: Black background, White toggle
4. **Navigation**
   * **Header**
     + Background: Black
     + Text Color: White
     + Icons: White
   * **Sidebar**
     + Background: White
     + Text Color: Black
     + Active Item: Light Gray background
   * **Breadcrumbs**
     + Text Color: Dark Gray
     + Separator: Light Gray
   * **Pagination**
     + Active Page: Black background, White text
     + Inactive Page: White background, Black text
5. **Modals**
   * **Basic Modal**
     + Background: White
     + Border: Light Gray
     + Shadow: Subtle shadow
     + Header: Black text
   * **Confirmation Modal**
     + Background: White
     + Border: Light Gray
     + Shadow: Subtle shadow
     + Header: Dark Gray text for warnings
6. **Tables**
   * **Basic Table**
     + Header Background: Dark Gray
     + Header Text: White
     + Row Background: White
     + Row Border: Light Gray
     + Hover Row Background: Light Gray
7. **Notifications**
   * **Toast Notification**
     + Background: Dark Gray
     + Text Color: White
     + Icon: White
   * **Alert Banner**
     + Background: Dark Gray
     + Text Color: White
     + Icon: White
   * **Snackbar**
     + Background: Dark Gray
     + Text Color: White
     + Action Text: Light Gray
8. **Charts and Graphs**
   * **Line Chart**
     + Line Color: Black
     + Point Color: Dark Gray
     + Grid Lines: Light Gray
   * **Bar Chart**
     + Bar Color: Black
     + Grid Lines: Light Gray
9. **Lists**
   * **Basic List**
     + Background: White
     + Text Color: Black
     + Divider: Light Gray
   * **Expandable List**
     + Background: White
     + Text Color: Black
     + Divider: Light Gray
10. **Miscellaneous**
    * **Avatar**
      + Background: Light Gray for initials, Image otherwise
      + Border: Gray
    * **Tooltip**
      + Background: Dark Gray
      + Text Color: White
    * **Loader/Spinner**
      + Color: Dark Gray
    * **Progress Bar**
      + Background: Light Gray
      + Fill: Dark Gray
    * **Accordion**
      + Background: White
      + Border: Light Gray
      + Expanded Background: Light Gray
    * **Carousel**
      + Arrow Color: Dark Gray
      + Dot Color: Gray (active: Black)

#### Implementation Guidelines

1. **Component-Based Architecture**
   * Develop components as independent, reusable modules.
   * Use a consistent naming convention for components and their props.
2. **Styling**
   * Use a centralized theming system to manage colors, fonts, and spacing.
   * Implement styles using CSS-in-JS libraries like Styled Components or Emotion.
3. **State Management**
   * Use React hooks and Redux for managing component state.
   * Ensure components are stateless where possible, passing state and actions as props.
4. **Accessibility**
   * Follow WCAG (Web Content Accessibility Guidelines) standards.
   * Implement ARIA (Accessible Rich Internet Applications) roles and attributes.
5. **Testing**
   * Write unit tests for each component using Jest and React Testing Library.
   * Ensure high test coverage and include tests for edge cases.
6. **Documentation**
   * Document each component’s API, props, and usage examples.
   * Use tools like Storybook to create interactive component documentation