### **Programmer Guide - Recipe Website by Team 21**

### Welcome to the comprehensive programmer guide for our Recipe Website. This application serves as a one-stop solution for users looking to explore, manage, and share recipes. It features a client-side interface built with React, offering dynamic interactions with recipes, meal plans, and shopping lists. The server side is powered by Node.js and Express, with MongoDB as the database, ensuring efficient data handling and persistence.

Instructions for running the project can be found in our readme file at the repo:

<https://github.com/amir3x0/A6-Final.git>

Link to DB:  
<https://cloud.mongodb.com/v2/65e0d5280bd1ec352566abc0#/metrics/replicaSet/65e0d56cd361746d3d381a65/explorer/test/recipes/find>  
  
\*\* **note:** you will have to make a request to gain access to the B

### **Client-Side Component Overviews**

#### RecipeSection Component

Purpose: Facilitates the display, filtering, and selection of recipes, allowing users to interact dynamically with the application's offerings based on categories or search terms. It acts as the primary interface for users to explore available recipes.

Key Features:

* Dynamically fetches recipes from a backend service, categorizing them for easy navigation.
* Implements a search functionality that lets users filter recipes by keywords.
* Integrates with custom React contexts to manage global states like selected recipes and user-preferred UI themes.
* Offers navigational controls to other parts of the application, enhancing user flow.

#### PlanSection Component

Purpose: Empower users to create customized meal plans by selecting recipes from predefined categories. It supports the addition of new recipes to categories, viewing selected recipes, and saving the curated meal plan with a unique name.

Key Features:

* Displays selected recipes categorized by meal types, dynamically updating as users add or remove recipes.
* Facilitates adding new recipes to each category through navigation with state passing, promoting content discovery and user engagement.
* Allows for the naming and saving meal plans, integrating closely with the user's profile for a personalized experience.

#### ShoppingSection Component

Purpose: Provides a comprehensive interface for managing shopping lists, including the manual addition of ingredients or the automatic inclusion of selected recipes. It aids in the preparation process by organizing ingredients and quantities needed.

Key Features:

* Manages state for individual ingredient fields (name, quantity, unit), allowing users to add ingredients to their shopping list with precision.
* Calculates and displays ingredients required for recipes selected for the shopping list, streamlining the meal preparation process.
* Integrates with ShoppingListContext and RecipesForShoppingListContext for effective global state management of the shopping list and recipe selections.

#### SettingsModal Component

Purpose: Serves as a modal interface for users to update their personal information, including bio, theme preference, and profile image URL. It emphasizes user customization and personalization within the application.

Key Features:

* Dynamically populated form fields with the current user data ensure a user-centric approach to management settings.
* Implements form submission handling to update user settings, reinforcing the application's responsiveness to user inputs.

#### MyYummy Component

Purpose: Renders the user's profile page, showcasing their favorite recipes, uploaded recipes, and meal plans, and providing access to settings for managing their profile. It centralizes user-related content and preferences for easy access.

#### Share Component

* Purpose: Allows users to share their recipes with the community. This component offers fields for inputting recipe details, including ingredients, instructions, and nutritional information, with an integration to an external API for ingredient details.
* Key Features:
  + Enables users to search and select ingredients from an external API, providing a dynamic ingredient selection experience.
  + Calculates and displays the total nutritional information of the shared recipe based on selected ingredients and quantities.
  + Facilitates the sharing of detailed recipes, including images, difficulty levels, and categories, directly to the application's database.

#### MealCard & RecipeCard Components

* Purpose: Enhances the display of meals and recipes with functionalities such as expand/collapse details for MealCard and liking, selecting, and adding ingredients to the shopping list for RecipeCard.
* Key Features:
  + MealCard visually presents a meal through a collage of recipe images, with an interactive option to view more details.
  + RecipeCard integrates with the user's favorite recipes, allowing users to like or unlike recipes and perform recipe-specific actions.

#### SignInPage, SignUpPage, and JoinUs Components

* Purpose: Manages user registration, sign-in, and community engagement by inviting users to join. These components collectively support user authentication, new user registration, and community building.
* Key Features:
  + SignInPage provides a form for user login, with backend authentication and navigation to user-specific pages upon successful login.
  + SignUpPage offers a user-friendly registration form, guiding new users through the sign-up process with options for navigation to the sign-in page.
  + JoinUs invites users to become part of "The Fatties" community, providing reasons to join and a toggleable sign-up form for new users.

#### Welcome Component

* Purpose: Serves as the landing page, offering a welcoming environment with a call-to-action for recipe exploration. It visually engages users with a selection of popular recipes.
* Key Features:
  + Presents a visually appealing introduction to the application, with images of popular recipes and an enticing call-to-action button.
  + Promotes immediate user engagement by navigating users to explore recipes upon interaction.

### State Management and Hooks

* useState is leveraged across all components for managing local states, including user inputs, selection states, UI states, and more. This hook facilitates reactive UI updates and state encapsulation within components.
* useEffect plays a crucial role in performing side effects such as fetching data upon component mount and responding to state or prop changes, ensuring that the data displayed is always up-to-date and relevant to user interactions.
* Context API Integration: The application employs React contexts (useUser, useSelectedRecipes, etc.) extensively for global state management. These contexts provide a seamless way to share state across components, enabling functionalities like user preferences, selected recipes for meal planning or shopping lists, and theme settings to be consistently managed and accessed application-wide.

### Backend and API Integration

* External API and Backend Communication: The Share component, for instance, interacts with external APIs for ingredient searches, fetching detailed nutritional information, and also communicates with the application's backend for actions like recipe sharing. This demonstrates a sophisticated integration pattern, allowing the application to offer rich features and functionalities.
* Asynchronous Operations: Components perform asynchronous operations to communicate with backend services (e.g., fetchRecipes, saveMealPlan), effectively linking user actions on the frontend with database updates and data fetching processes. This ensures a dynamic and interactive user experience, supported by current and relevant data.

### Implementation Highlights and Code Insights

#### RecipeSection: Fetching and Categorizing Recipes

* Demonstrates fetching recipes from the backend and categorizing them for display, highlighting effective data manipulation and UI rendering based on fetched data.

#### PlanSection: Adding Recipes to Meal Plan

* Illustrates the process of adding selected recipes to a user-defined meal plan, showcasing how user actions can influence the application state and interact with backend services for data persistence.

#### ShoppingSection: Managing Shopping List

* Exemplifies managing a dynamic shopping list based on user-selected recipes and manual ingredient additions, showing complex state management and interaction with external APIs for ingredient details.

#### Share Component: Ingredient Selection and Nutrition Calculation

* Features dynamic ingredient search and selection from an external API Spoonacular, and calculates the total nutritional information of a shared recipe. This component stands out for its interactive UI, real-time feedback, and integration with external data sources.

#### SignInPage: User Authentication Flow

* Implements a comprehensive authentication flow, handling form submission, user feedback, and navigation based on authentication status, which is critical for securing user sessions and personalizing user experiences.

#### JoinUs Component: Community Engagement

* Aims to build community engagement by inviting users to join through a compelling sign-up form, bolstered by reasons to join the community. This component emphasizes user acquisition and community building.

### **Server-Side Component Overviews**

* Environment Configuration: Utilizes dotenv for managing environment variables, ensuring sensitive information like the database connection string is securely loaded and accessed.
* Express Application Setup: Leverages Express for server creation and configuration, establishing a robust foundation for handling HTTP requests.
* Middleware Integration:
  + CORS: Enables Cross-Origin Resource Sharing, allowing the frontend application to interact with the backend without facing cross-origin issues.
  + JSON Parser: Uses express.json() to parse incoming JSON payloads, facilitating easy access to request bodies.
* Request Logging: Implements a simple middleware for logging request paths and methods, aiding in debugging and monitoring server activity.

### Routing and Controllers

* Modular Routing: Defines routes for users, recipes, meal plans, and ingredients, each managed through separate files for clarity and maintainability.
* Controller Functions: Incorporates a variety of controller functions (e.g., createUser, getRecipe, createMeal) that encapsulate business logic for handling specific tasks like user registration, recipe fetching, and meal plan creation.
* Authentication Middleware: Integrates verifyToken middleware to protect routes that require user authentication, ensuring that only authenticated users can access certain functionalities.

### Database Interaction and Model Integration

* Mongoose for MongoDB: Utilizes Mongoose to connect to MongoDB, offering a schema-based solution to model application data. This integration facilitates operations like creating, fetching, updating, and deleting records with ease.
* Model Definitions: Employs Mongoose models (User, Recipe, Meal) to define and enforce the structure of the data within the database, supporting consistency and data integrity.

### Feature Implementation Details

* User Authentication and Profile Management: Handles user authentication, profile updates, and preference settings, showcasing secure password handling and personalized user experiences.
* Recipe Management: Offers comprehensive functionalities for recipe management including creation, retrieval, updating, and deletion, catering to the core features of the application.
* Meal Planning: Supports meal plan creation and association with user profiles, illustrating the application's capability to provide value-added services like meal planning.
* Ingredient Information Fetching: Integrates external APIs for fetching detailed ingredient information, enhancing the recipe creation process with valuable data like nutritional content.

### Security and Error Handling

* Implements security best practices such as password hashing and route protection with authentication middleware, ensuring user data protection and secure access control.
* Incorporates error handling mechanisms throughout the application, providing meaningful feedback to the client and logging errors for server-side monitoring.

### References

* - [Full-Stack JavaScript by The NetNinja]

https://www.youtube.com/watch?v=98BzS5Oz5E4&list=PL4cUxeGkcC9iJ\_KkrkBZWZRHVwnzLIoUE

* - [React & Node Tutorial by Academind]

https://www.youtube.com/watch?v=BbRmXEpvWwY&list=PL9UYLHIX7qsR41B7N80fEXbALVMK2Q9lV

### APIs:

Imagekit:

ImageKit is a real-time image optimization and transformation service that helps improve the performance and visual experience of websites and mobile applications. It provides a suite of tools to manage, optimize, and deliver images and videos efficiently, supporting responsive images, image resizing, format conversion, and quality adjustment on the fly.

Spoonacular:

Spoonacular is a comprehensive API that provides access to an extensive database of recipes, food products, and nutritional information. It allows developers to integrate a variety of food-related features into their applications, such as recipe search and discovery, meal planning, and nutritional analysis.