MERN Stack Development:

Meal Planner App

Asia Shvets, Joshua Emery, Anahita Raeiszadeh,

Matthew Thibault, Vaishnavi Mandage

CS624 Full Stack Development I

City University of Seattle

School of Technology and Computing

[shvetsasia@cityuniversity.edu,](mailto:Shvetsasia@cityuniversity.edu) [emeryjoshua@cityuniversity.edu](mailto:emeryjoshua@cityuniversity.edu),

[raeiszadehanahita@cityuniversity.edu,](mailto:raeiszadehanahita@cityuniversity.edu) [thibaultmatthew@cityuniversity.edu](mailto:thibaultmatthew@cityuniversity.edu)

mandagevaishnavi@cityuniversity.edu

**Abstract**

This paper introduces a mobile application developed using React Native, designed to streamline meal planning and enhance the overall user experience. Mobile food applications establish partnerships with numerous restaurants, serving as an intermediary connecting eateries and consumers. Multiple factors contribute to the growth in their sales, including ease of use, convenient payment options, a diverse range of food and dining establishments, delivery efficiency, and quality customer service, among others. The React Native Meal Planner app offers an intuitive and user-friendly interface, making it easy for users to create, customize, and manage their meal plans for a week. Key features include a vast recipe library, and a user-friendly interface for tracking nutritional information. User can set their own dietary goals and the app can make suggestions for recipes based on the user’s criteria. The app's cross-platform compatibility ensures accessibility on both Android and iOS devices. Through an examination of its development and user interface design, this paper explores the potential benefits of leveraging React Native for the creation of efficient and versatile meal planning applications. This research proved instrumental in gaining insights into the role that applications play in the contemporary landscape and the transformative impact they have had on the food service industry.

**Keywords:** React Native, Recipe finder API, Food app, Mobile app development, User interface design, iOS, Android, Cross-Platform, Full Stack Development, MERN

**1. INTRODUCTION**

In today's fast-paced lifestyle, the role of meal planning has transcended convenience to become a fundamental component of how individuals approach nutrition and dining. Due to the demands of contemporary lifestyles, Mobile Food Apps have gained popularity. Many individuals now favor online food ordering over home cooking. The introduction and progress of these mobile food applications have been significantly influenced by technology. This paper delves into the development and implications of an innovative mobile application—the React Native Meal Planner app—crafted to streamline meal planning processes and elevate the user experience to new heights.

The youth of the nation are showcasing exceptional prowess in embracing the latest technological advancements and innovations. Professionals in the workforce aspire to excel within their organizations, and technology stands as a pivotal asset aiding them in this pursuit. What once appeared unattainable is now within reach, courtesy of technological progress. Presently, diverse businesses are establishing online storefronts, a trend facilitated by technological advancements. Factors like affordable smartphones and accessible internet data packs have propelled a surge in mobile app usage among people.

In the contemporary landscape, where mobile applications serve as conduits connecting consumers with a plethora of culinary offerings from various restaurants, the React Native Meal Planner app emerges as a comprehensive solution. Its utilization of React Native technology ensures a seamless experience across diverse platforms, catering to both Android and iOS users. Through this exploration, this paper seeks to unravel the multifaceted advantages of employing React Native in creating efficient and adaptable meal planning applications.

**Usefulness of Recipe Planning**

Meal planning has evolved from a mere routine to a crucial aspect of modern living, significantly aided by mobile applications. These applications offer a myriad of advantages:

**Intuitive Functionality:** Mobile food applications simplify meal planning, offering interfaces that enable users to effortlessly create, customize, and manage their meal plans.

**Access to Culinary Diversity:** Integration with Recipe Finder APIs provides access to extensive recipe libraries, empowering users to explore diverse cuisines, dietary preferences, and personalized suggestions.

**User-Centric Design:** Features like nutritional information tracking prioritize user preferences and dietary goals.

**Cross-Platform Accessibility:** Utilizing React Native ensures compatibility across different devices, enhancing accessibility for a broader user base across Android and iOS platforms.

**2. BACKGROUND**

In recent years, the confluence of technology and dietary habits has catalyzed a profound shift in meal planning approaches, prominently influenced by the integration of Recipe Finder APIs. These APIs have emerged as pivotal elements in the development of meal planning applications, as highlighted in various scholarly works and resources.

Studies by Iqbal and Permadi (2021) underscore the instrumental role of APIs, particularly in fusion with automated meal planners like dietducate, enhancing personalized meal planning experiences. Similarly, Mauch's (2019) extensive investigation into cross-platform mobile application development using React Native emphasizes the efficiency and advantages of API integration, as exemplified in "Plan-cook-eat," a meal planner app focusing on personalized dietary planning based on optimal macronutrient distribution by Garcia (2019).

Simultaneously, consumer behavior has undergone a notable shift, especially regarding online food ordering, as elucidated by Kimes' comprehensive study (2011). This shift, driven by factors like convenience and control, has reshaped interactions with restaurant staff and propelled the surge in online food orders, signifying a transformative phase in consumer expectations.

Moreover, academic research has augmented this evolution. Tamim et al.'s study (2023) on the Mess Meal Planner Application, Vinaik et al.'s exploration (2019) into consumer interest in mobile food ordering apps, and Kumar et al.'s analysis (2021) of the 'Foodie' Online Food Delivery App provide comprehensive insights into functionalities, user experiences, and consumer behaviors within meal planning applications.

This evolution in meal planning, facilitated by technology, aligns with the proliferation of mobile apps supporting healthy eating practices within families, as highlighted by Golley et al.'s (2018) systematic assessment of popular commercially available apps.

Additionally, the academic discourse surrounding React Native technology, detailed by Eisenman (2015) and Danielsson (2016), has elucidated its robust capabilities in mobile app development using JavaScript. These insights into React Native's adaptability and cross-platform compatibility have been pivotal in creating versatile applications catering to diverse user bases across Android and iOS platforms.

The amalgamation of these technological advancements, consumer behavioral shifts, and scholarly research underscores the dynamic evolution of meal planning applications, emphasizing the transformative impact of Recipe Finder APIs in reshaping dietary management within the contemporary technological landscape.

**3. PROBLEM STATEMENT**

**Optimal Recipe Data Management:** There exists a challenge in determining the most effective approach to managing recipes sourced from external Recipe Finder APIs within the Users database. The debate continues whether to duplicate and store recipes or dynamically fetch them from the Recipe Finder API during each application launch. This ambiguity presents a dilemma in defining the most efficient recipe data handling process.

**Database Architecture Evolution:** The ongoing development and evolution of the database architecture aims to facilitate seamless association and management of recipes within user profiles. However, this critical phase necessitates refinement to enhance usability, personalization, and the efficient handling of recipe data. Determining the best strategy for handling recipe data integration within the Users database remains pivotal.

**Efficiency Centered on User Needs:** There's a need for an approach that ensures seamless accessibility, minimizes data redundancy, and enhances overall performance in managing recipe data within user profiles. Striving for improved user personalization and usability while efficiently managing and retrieving recipe data to optimize the user experience is a fundamental concern.

These problem statements revolve around the challenge of effectively handling recipe data, evolving database architecture for improved usability and personalization, and ensuring user-centric design and efficiency within the Meal Planner App. They form the core aspects that require resolution and optimization in the application's development and database integration.

**4. LITERATURE REVIEW**

The integration of Recipe Finder APIs is foundational in the development of meal planning applications, as evidenced by key scholarly works and resources:

Iqbal and Permadi (2021) explored the fusion of dietducate and an automated meal planner, highlighting APIs' critical role in augmenting personalized nutritional experiences.

Mauch (2019) extensively investigated cross-platform mobile application development using React Native, accentuating its practical implications, especially the efficient integration of APIs.

Garcia (2019) introduced "Plan-cook-eat," emphasizing API integration for personalized dietary planning, focusing on optimal macronutrient distribution based on personal energy expenditure.

Kimes (2011) conducted a comprehensive study on the surge in online food ordering, indicating a significant shift in consumer behavior and perceptions towards mobile food ordering applications, emphasizing convenience and control factors.

Tamim et al. (2023) meticulously analyzed the Mess Meal Planner Application from Sonargaon University, offering insights into functionalities, user experiences, and technological underpinnings for further development and enhancement.

Furthermore, Vinaik et al.'s (2019) research delved into consumer interest in mobile food ordering apps, providing crucial insights into user preferences and behaviors, pivotal for user-centric design in meal planning applications.

Similarly, Kumar et al.'s (2021) study on the 'Foodie' Online Food Delivery App detailed specific user interface elements, functionalities, and user engagement aspects, contributing valuable insights into mobile food application dynamics.

The cumulative findings from these scholarly works collectively underscore the vital role of Recipe Finder APIs in shaping the landscape of meal planning applications, emphasizing their significance in personalized user experiences, and shifting consumer behaviors.

**5. METHODOLOGY**

The development of the React Native Meal Planner app followed a systematic and comprehensive approach, integrating various components to ensure a seamless user experience. The design methodology encompasses the following key stages.

**User Authentication and Integration**

The app incorporates a user-friendly authentication system, allowing users to register and log in with their credentials. With seamless authentication, users receive JWT tokens for personalized and secure access to meal plans and recipes. This addition not only improves functionality but also contributes to the overall visual appeal, reflecting the app's commitment to a user-centric and visually engaging meal planning experience. Integration with OAUTH through Google was also tested and implemented. This topic is discussed in detail in the Authentication section.

**Recipe Finder API Integration**

A crucial aspect of the app is its integration with a Recipe Finder API. This API serves as a rich source of diverse recipes, enabling users to explore a vast library of culinary options. By leveraging this API, the app dynamically fetches recipes based on user preferences, dietary goals, and criteria, providing a personalized and tailored experience.

**Database design**

The integration of the Users database within MongoDB plays a pivotal role in personalizing the user experience within the React Native Meal Planner App. Currently, the database architecture is evolving to allow users to associate and manage recipes seamlessly within their profiles. This functionality is in the developmental phase, presenting an opportunity to enhance the app's usability and personalization.

One key consideration in the ongoing development is the handling of recipes sourced from the Recipe Finder API within the Users database. There's deliberation on whether retrieved recipes should be duplicated and stored within our “Users” database or if they should be dynamically fetched from the Recipe Finder API during each application launch.

**Backend Development**

The backend of the React Native Meal Planner is developed using a Full Stack MERN (MongoDB, Express.js, React, Node.js) architecture. This ensures robust data storage, efficient data retrieval, and seamless communication between the server and the client. The backend handles user data and recipe information based on the selected meal plans.

**Frontend Development**

The frontend of the app is meticulously designed to offer an intuitive and visually appealing interface. Using React Native, the development team crafted a cross-platform solution compatible with both iOS and Android devices. The interface allows users to effortlessly navigate through various features, promoting a smooth and enjoyable meal planning process.

**UI Design**

**1. Navigation Tabs**

The app features organized navigation tabs to streamline user interaction. Tabs include "My Recipes" for personalized recipe collections, "Recipe Finder" for exploring recipes based on cuisine, diet, intolerance, and other criteria, and "Search results" for creating and managing meal plans for the week. This tab structure ensures that users can easily access and utilize the app's core functionalities.

**2. My Dietary Preferences**

The "Dietary Preferences" tab is a central component of the app, empowering users to create, customize, and manage their meal plans efficiently. Users can set dietary goals.

In summary, our methodology blends secure authentication, API integration, robust backend development, and an intuitive frontend design to deliver a comprehensive meal planning experience. By adopting this approach, we aim to showcase the advantages of React Native in creating versatile and efficient meal planning applications. This study explores the development intricacies and user interface design considerations, shedding light on the transformative impact such applications have on the contemporary food service industry.

**6. AUTHENTICATION**

For user authentication we explored both third party authentication through Google as well as creating our own login system using MongoDB and JWT. Both techniques were implemented and below we have contrasted these techniques.

**Google login**

Implementing Google login in our React app was a pivotal step to reduce friction for users, especially considering Google's widespread use.

In the development of our application, we explored and successfully implemented Gmail sign-in. Initially deployed on the web, we seamlessly extended this functionality to Android and iOS platforms using React Native.

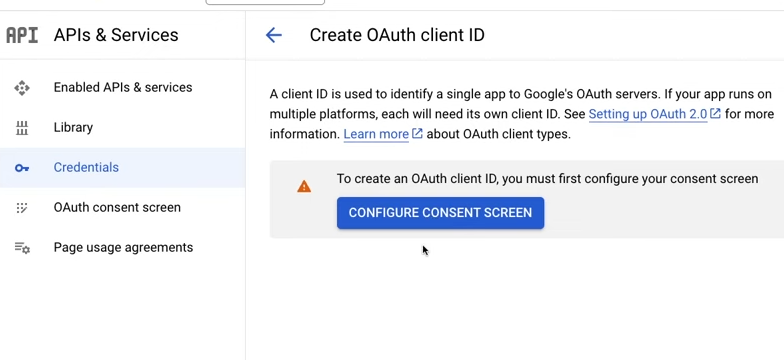


Figure 1 - “Create OAuthClentID”

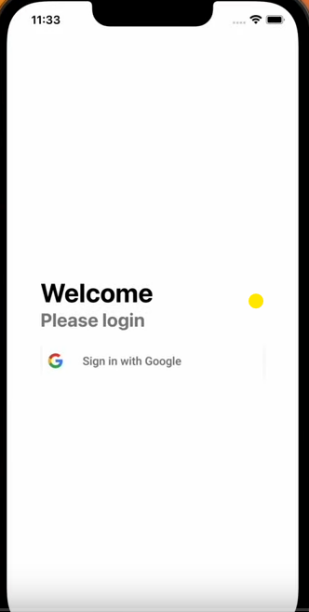


Figure 2 - “Welcome page”

During this process, we enriched our understanding by integrating Google login through the @react-oauth/google package. Key steps included obtaining a Google client ID, configuring a consent screen, and creating a web client ID. The final touch involved installing the @react-oauth/google package, wrapping our app with GoogleOAuthProvider, and providing the web client ID for a smooth Google login experience.

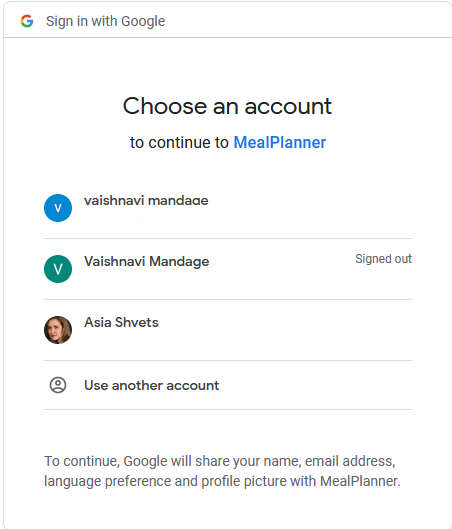


Figure 3 “Second page”

Beyond being a feature addition, this journey became a deep learning experience. It provided valuable insights into working across platforms, integrating with React Native, and the importance of user-centric design. Gmail sign-in wasn't just a feature; it became a cornerstone of our learning, helping us navigate platform diversity, user behavior, and the art of creating a consistent experience across various settings.

**Creating our own Authentication Process**

Based on the requirements of our application we needed a permanent store of data that would enable users to save recipes and dietary information. This mandated the creation of a database. This database would need to store user information. As the creation of this database and corresponding API was already required, we also explored simply using our database for the user to login and logout. We researched different techniques to accomplish this and settled on a user's table and Json Web Tokens (JWT). JWT is an open-source standard that provides secure transmission of data between networked resources. When a user successfully logs in a token is generated. This token must be included with all future requests to protected resources of the API. The token is passed in the Header of the request in a Authentication key. Tokens not only allow for users to remain logged in for a given person of time they also reduce the amount of data that must be sent for each request. Consider the following example: A request is sent to an API to retrieve all of the saved recipes for a user. In a more traditional endpoint, we could require either a userId or perhaps an API key. We would need to validate the user exists and then also validate the key. In the case of JWT we can simply include the token. Verification of the user is automatic assuming that the token is still valid. All data associated with that user is easily accessible based on the token. Consider the following request:



Figure 4 Request getSavedRecipes

When this request was sent no user object was included nor was there a user id. The only data sent to this request was the JWT Token. The user object is automatically populated because getSavedRecipes is a protected route. The JWT middleware automatically locates the user:

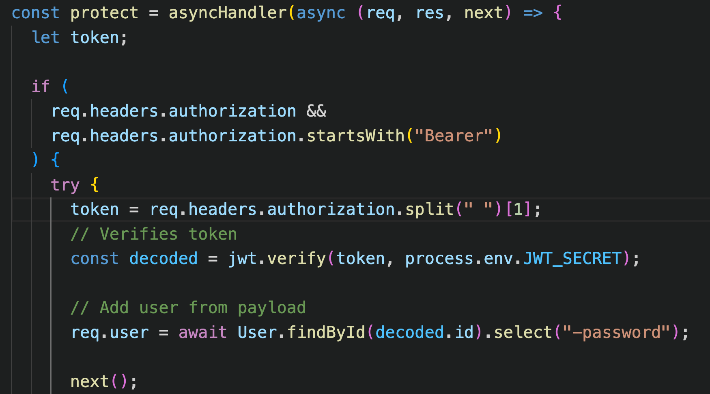


Figure 5

This incredible ease of integration into our existing API was the deciding factor on whether to use Google OAuth or JWT via mongoDB.

The registration process was handled via a post request to users/ and required an email and a password. The bcrypt library was used to encrypt all user passwords. This is a one-way encryption algorithm; all comparisons of password are performed on the encrypted value. At no point is a user password stored in plain text.

**7. THE USERS DATABASE**

In addition to allowing for users to log in and out, users are able to save dietary restrictions and other criteria that will be used to generate custom and recipe data tailored to each user’s preferences. We have developed a MongoDB that serves as persistent storage for this application. Access to the database is provided with a web API. See below for an example user object:



Figure 6 – User object

Saved recipes are recorded as ids which correspond to an entry in the [Spoonacular](https://spoonacular.com/) Web API. It is important to distinguish that there are two APIS that are involved in this application. The Users API handles Authentication, user preferences and saved recipes. The Recipes API is Spoonacular.

**7. THE USERS API**

Access to the Users database is facilitated by an Express based node web API. The API has the following endpoints.

## User Endpoints

### **1. Get User**

* **Description**: Retrieves a user object based on their given JWT token.
* **Endpoint**: **GET /users**
* **Access**: Private
* **Usage**: **await getUsers(req, res);**

### **2. Update User Preferences**

* **Description**: Modifies the preferences of a user based on the JWT token.
* **Endpoint**: **PUT /users/preferences**
* **Access**: Private
* **Usage**: **await updateUserPreferences(req, res);**

### **3. Register User**

* **Description**: Registers a new user.
* **Endpoint**: **POST /users/register**
* **Access**: Public
* **Usage**: **await register(req, res);**

### **4. Login User**

* **Description**: Authenticates a user and returns a JWT token.
* **Endpoint**: **POST /users/login**
* **Access**: Public
* **Usage**: **await login(req, res);**

Other endpoints were developed for testing; however these were the only ones required by production.

The results of a requests to GET/users is pictured below:

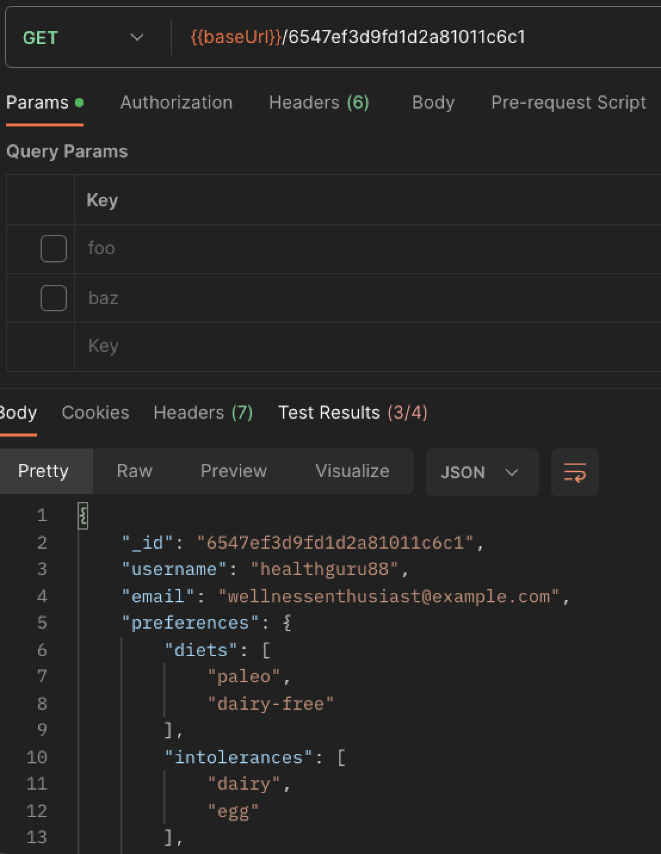


Figure 7 – GET request and response

To achieve an MVP (minimum viable product), string arrays were used to represent most data types to make the models easier to alter or expand upon. These models will need to be refactored once we have determined the exact criteria which users will be providing and storing in the Users database. Basic functionality is complete for the users section of the API. A collection of recipes endpoints was also developed. A basic workup of a potential recipe model has been developed so that the development of the recipe endpoints can be easily completed if they are needed. A portion of the recipe model is included below.

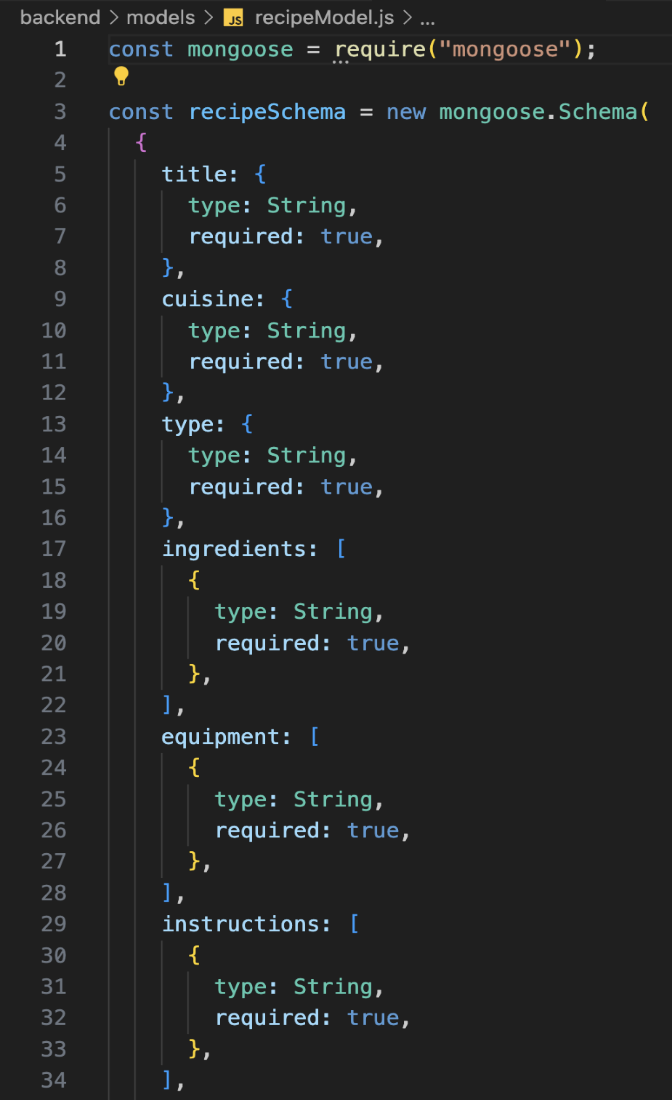


Figure 8 – "Recipe" model

This API is currently deployed to Azure and can be accessed at the following URL:

<https://624api.azurewebsites.net/users>

**8. LOGGING INTO THE APPLICATION**

Logging into the application was accomplished using fetch with a post request to /users. This request would return a token. The token would need to be accessed by nearly every component in the client side application. For this reason, we chose a global state hook called useContext to store the user token. If the token is found to be expired it will be removed from context and the user would be prompted to login again. Registration works similarly with the added step of encrypting the user password and storing it in the monogdb users table. A token is then returned and is added to context.

**9. ACCESSING THE Recipe API**

Access to the recipes api was also accomplished via a fetch request. We primarily used the complexSearch endpoint available at <https://api.spoonacular.com/recipes/complexSearch>. This endpoint is capable of taking multiple parameters that can be used for filtering the results. For example a request for a main course that includes chicken, potatoes, and peas but does not include carrots and has high protein content is handled well by this endpoint.

One of the main challenges we encountered when connecting the front and back end was how to handle multiple asynchronous requests that needed to fire and impact the virtual DOM. Initially I thought that while coding this project I may need to go outside of the normal react workflow of useState and useEffect and create a simple reference to accomplish this task. Consider the following scenario. First a list of saved recipe ids must be retrieved from the user's database via an asynchronous request. This was accomplished via a useEffect and the ids tracked via state. However, these ids would then need to be passed to another asynchronous request to retrieve the recipe information which was also tracked by state. This initially was done with a useEffect as well based on the ids array. This functionality did not work as expected as initially we created an infinite loop. We also ran into the problem where the setState method of the ids would not have updated in time for the second api call as setState is asynchronous itself. Here is our final solution for this situation.



Figure 9 – "Recipe" model

Notice that we removed the ids array from the state, as this was not displayed to the user. The other main change which enabled the functionality was the use of Promise.all() function. This allows us to await a series of asynchronous calls. This ensured that the recipe details were complete before they were passed to set recipe details.

**10. USER INTERFACE**

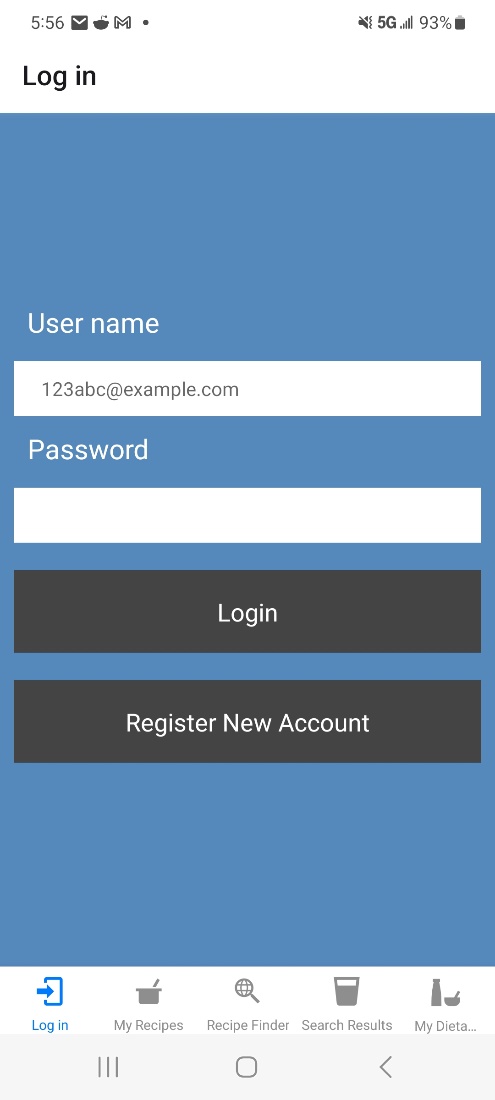


Figure 10 - “Login Page”

The Meal Planner app includes the integration of a user-friendly login screen featuring email and password fields. With seamless authentication, users receive JWT tokens for personalized and secure access to meal plans and recipes. This addition not only improves functionality but also contributes to the overall visual appeal, reflecting the app's commitment to a user-centric and visually engaging meal planning experience.

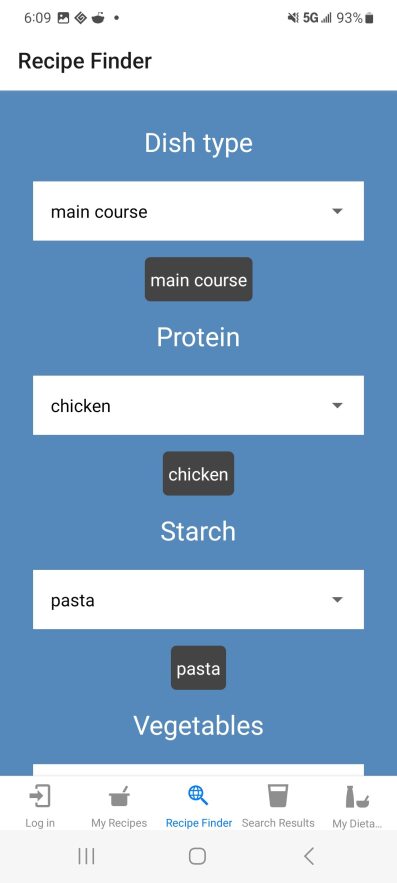


Figure 11 - “Recipe Finder screen”

The Recipe Finder screen introduces a range of user-friendly dropdown menus, including options for dietary types, protein sources, vegetables, and starches. This intuitive feature allows users to effortlessly customize their recipe searches, catering to specific preferences and dietary needs. The inclusion of these dropdown menus streamlines the exploration process, providing a more personalized and efficient experience for users seeking diverse and tailored culinary options.

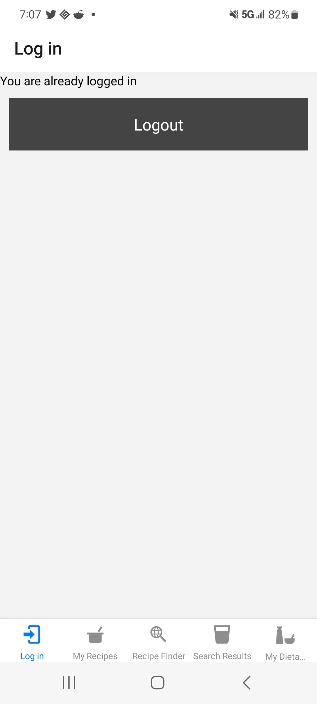


Figure 12 - “Logout screen”

The Meal Planner app offers users a secure and intuitive way to log out of their accounts. This feature enhances user control and privacy by terminating the current session, ensuring that sensitive information remains protected. This addition emphasizes the app's commitment to providing a secure and streamlined experience, empowering users with the ability to manage their sessions effortlessly.

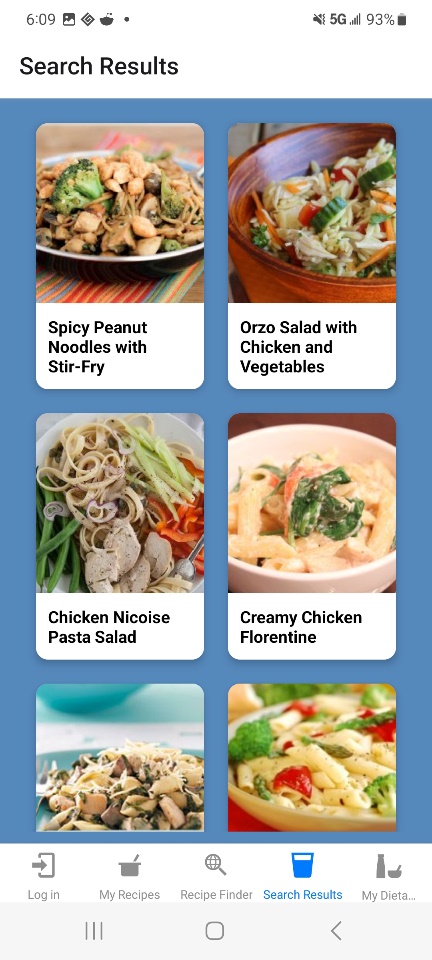


Figure 13 - “Search result screen”

Search Result screen, enhancing user engagement by dynamically presenting personalized recipe options based on specific search criteria. This feature enhances the user experience by offering a dynamic and visually appealing overview of diverse meal options, allowing users to effortlessly explore and select recipes based on their preferences. The inclusion of names and pictures provides users with a quick and informative glance at each recipe, streamlining the decision-making process and contributing to a more immersive and enjoyable meal planning journey.



Figure 14 - “My Dietary preferences screen”

Preferences screen, leveraging React Native's UI capabilities for seamless dropdown menu interactions. Users can efficiently customize diets, intolerances, and cuisines with state-of-the-art "include" and "exclude" options. The screen, integrated into the app's navigation using React Navigation.

**11. CONCLUSION**

In conclusion, the Meal Planner app not only transforms the meal planning experience but also establishes a benchmark for a user-friendly and visually captivating design. The seamless integration of features, starting from a secure login procedure issuing JWT tokens for personalized access, to the inventive Recipe Finder screen with adaptable dropdown menus, underscores the app's commitment to user convenience. Additionally, the exploration of the Google login option reflects the app's dedication to enhancing accessibility, although this feature has not been implemented in the current version. The Logout screen underscores the app's commitment to user privacy and control. The Preferences screen, utilizing React Native's UI capabilities, showcases the app's technical proficiency. Overall, the app is a standout solution, focusing on user needs, security, and innovative design in the meal planning space. Whether users are creating recipes or customizing preferences, the app offers a hassle-free journey.

**12.** **REFERENCES**

React Native Paper: official documentation, https://callstack.github.io/react-native-paper/

Iqbal, M., & Permadi, M. R. (2021, March). Analysis of integration dietducate and automated meal planner for nutritional purposes. In *IOP Conference Series: Earth and Environmental Science* (Vol. 672, No. 1, p. 012079). IOP Publishing.

Eisenman, B. (2015). *Learning react native: Building native mobile apps with JavaScript*. " O'Reilly Media, Inc.".

Swarna, I., Purnama, J., & Anthony, R. (2020). Cross-Platform Analysis and Development of Online Catering Platform (Kunyahku). *Journal of Applied Information, Communication and Technology*, *7*(2), 79-89.

Kuitunen, M. (2019). *Cross-Platform Mobile Application Development with React Native* (bachelor's thesis).

Garcia, M. B. (2019, November). Plan-cook-eat: a meal planner app with optimal macronutrient distribution of calories based on personal total daily energy expenditure. In *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)* (pp. 1-5). IEEE.

Mauch, C. E., Wycherley, T. P., Laws, R. A., Johnson, B. J., Bell, L. K., & Golley, R. K. (2018). Mobile apps to support healthy family food provision: systematic assessment of popular, commercially available apps. *JMIR mHealth and uHealth*, *6*(12), e11867.

Danielsson, W. (2016). React Native application development. *Linköpings universitet, Swedia*, *10*(4), 10.

Li, Y. (2022). *Building a Cross-Platform Bridging Library for Native Mobile SDKs* (Doctoral dissertation, Massachusetts Institute of Technology).

Harris, A., & Harris J. (2009). Publishing in JISE. *Journal of Information Systems Educators*, *7*(1), 12-15.

Wilson, E. (2018). MERN Quick Start Guide: Build Web Applications with MongoDB, Express. js, React, and Node. Packt Publishing Ltd.

Chatzidimitriou, K. C., Papamichail, M. D., Diamantopoulos, T., Oikonomou, N. C. I., & Symeonidis, A. L. (2019, July). npm Packages as Ingredients: A Recipe-based Approach. In ICSOFT (pp. 544-551).

Tamim, M., Hasan, M. M., Khan, M. S., & Rana, M. A. S. (2023). *A Comprehensive Study on Mess Meal Planner Application* (Doctoral dissertation, Sonargaon University (SU)).

Leong, W. H. (2016). *Food ordering system using mobile phone* (Doctoral dissertation, UTAR).

Vinaik, A., Goel, R., Sahai, S., & Garg, V. (2019). The study of interest of consumers in mobile food ordering apps. *International Journal of Recent Technology and Engineering*, *8*(1), 3424-3429.

Kumar, H., Jain, M., & Bajwa, M. S. (2021). Online Food Delivery App ‘Foodie’. *Journal of University of Shanghai for Science and Technology*, *23*(7), 761-771.