Splitter: Effortless Bill Management

Project Team: Daniel Distor

User Audience and Needs: Splitter is designed for anyone who enjoys dining out with friends, family, or colleagues and seeks a hassle-free way to split bills. Whether it's a casual outing, a birthday dinner, or a business lunch, our app ensures that each participant pays only for what they ordered, including proportional shares of tax and tips.

Features and Functionality

Splitter streamlines the process of bill management through its intuitive interface:

Welcome Screen: Easy access with options to register or sign in.
Event Management: Users can create events, adding participants and detailing the shared expenses.
Meal Exploration: For those cooking at home, the meal tab allows users to search for recipes or receive random dish suggestions.

App In Action
1. **Starting the App**: Upon launching Splitter, users are greeted with the welcome screen. New users can register by clicking the "Register" button.
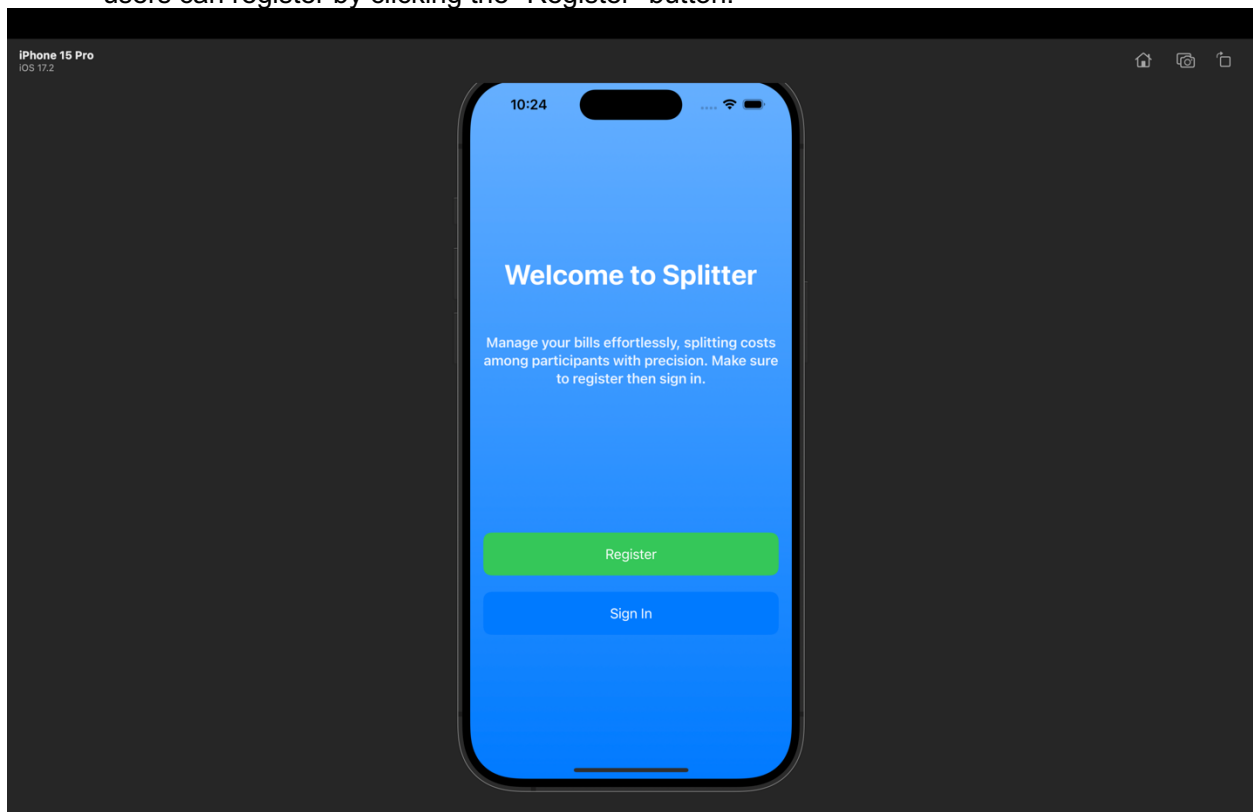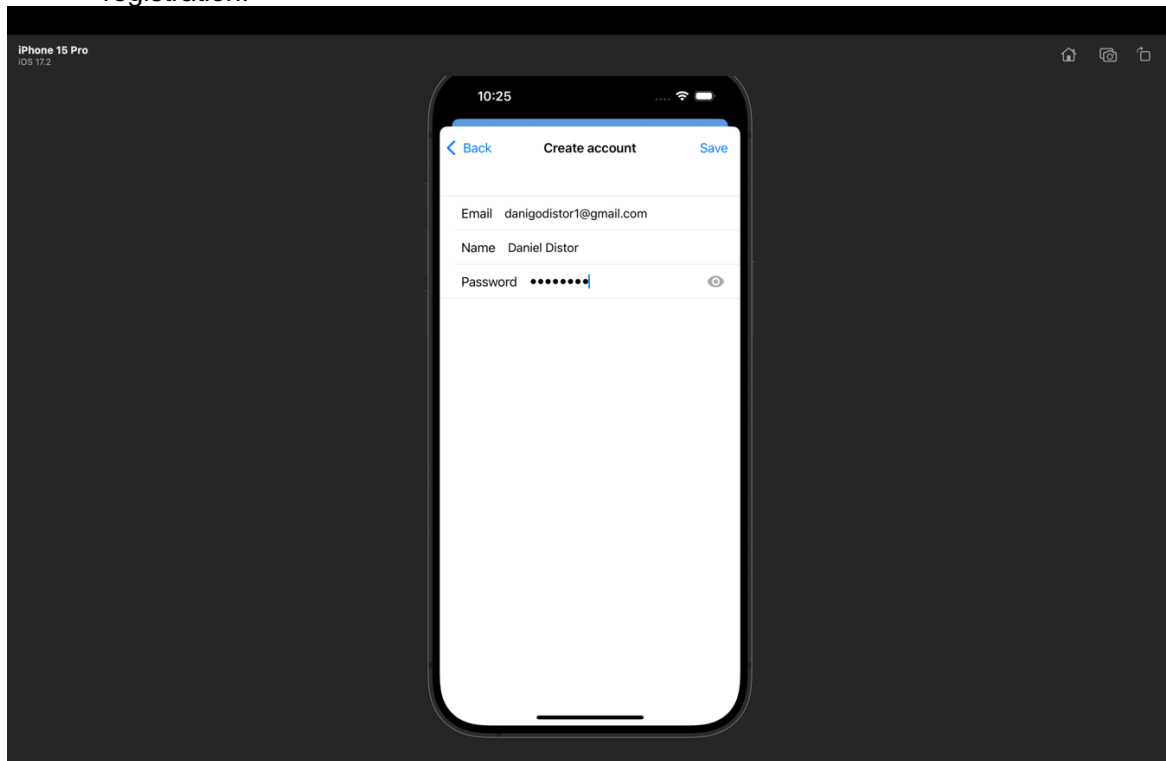
2. **Registration Process**: After clicking "Register", users are prompted to "Sign in with email". Selecting this option advances the process.
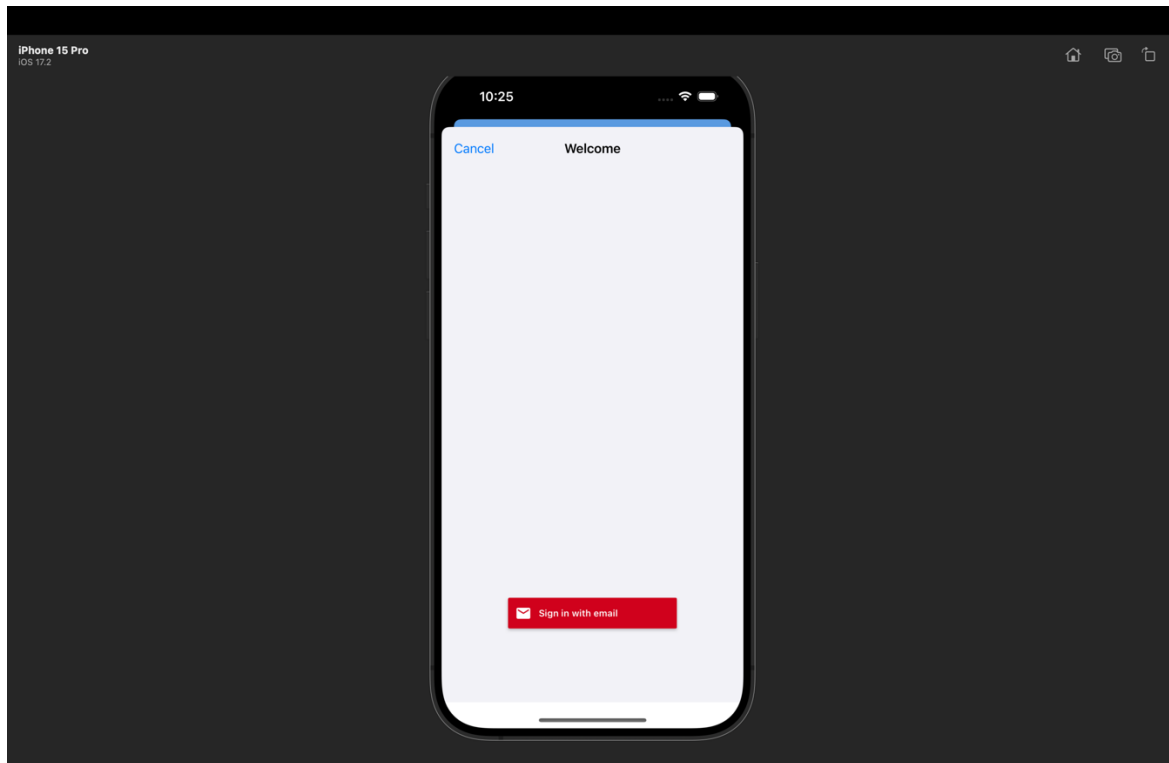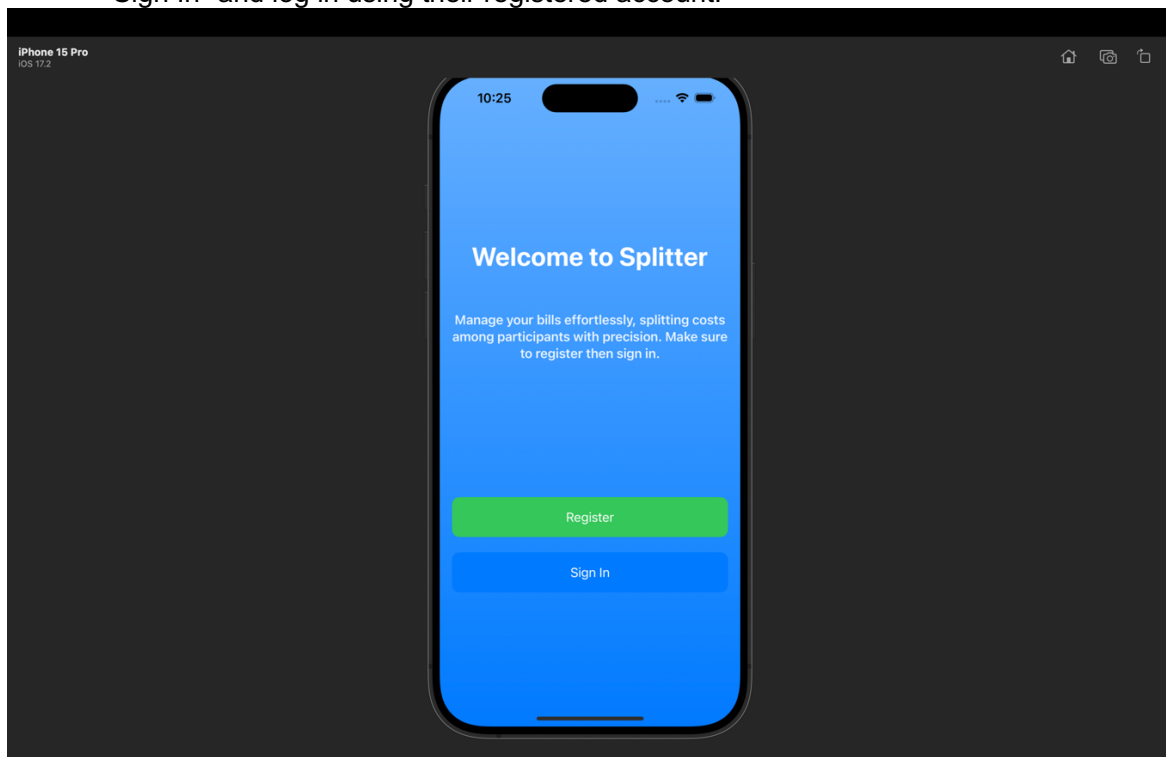

3. **Email Entry**: Users should enter their email address and click "Next" to continue.
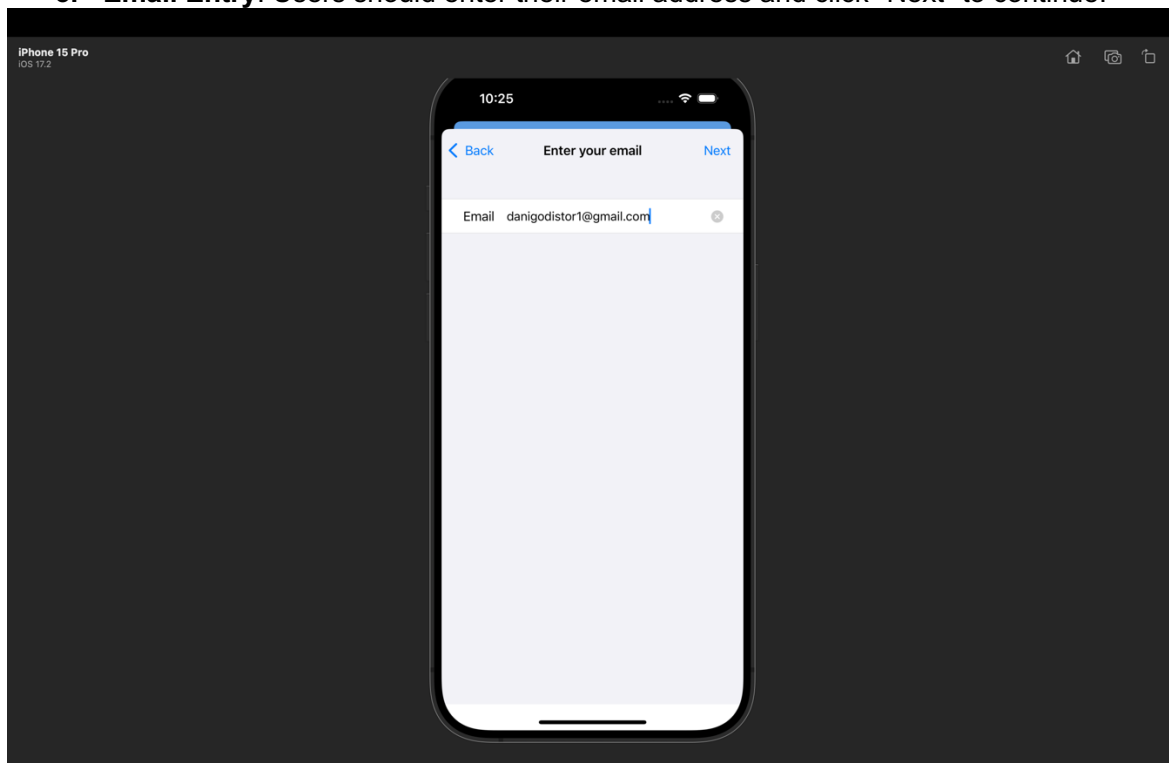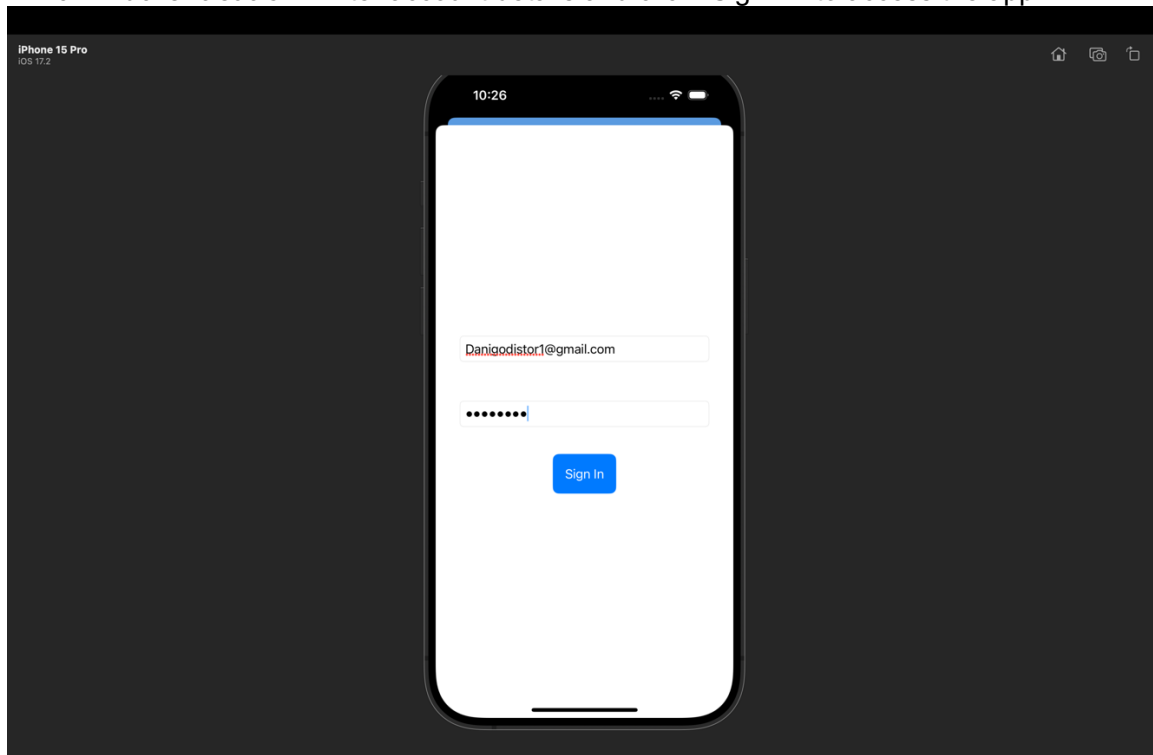
4. **Account Creation**: Input name and desired password, then click "Save" to finalize the registration.

5. **Sign In**: After saving, the app redirects back to the welcome screen. Here, users can click "Sign In" and log in using their registered account.

6. **Authentication**: Enter account details and click "Sign In" to access the app.

7. **Navigating to Split Tab**: Authenticated users are taken to the Split tab where existing events are listed. To add a new event, click "New" located at the top right of the screen.
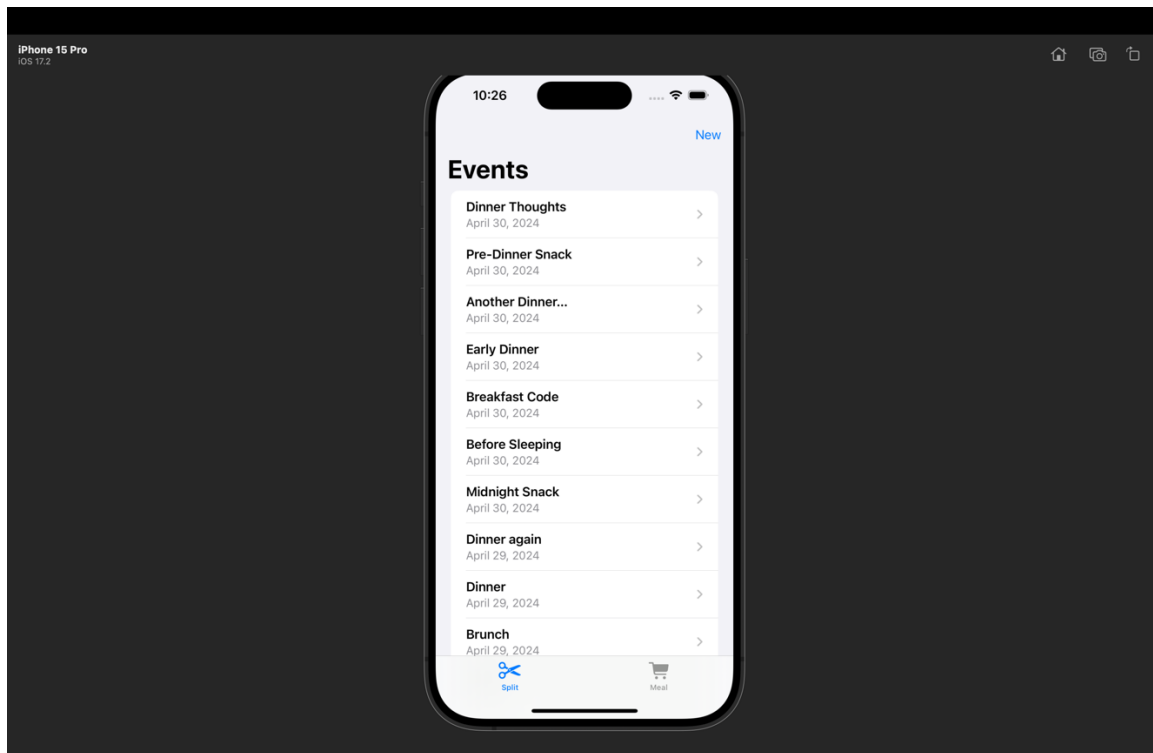
8. **Creating an Event**: Fill in the fields for participants and dishes. Click "Add Participant" and/or "Add Dish" if needed.

9. **Managing Expenses**: Scroll to the expenses section, enter all relevant financial details such as subtotal, tax, tip, and total. Click "Save Event".
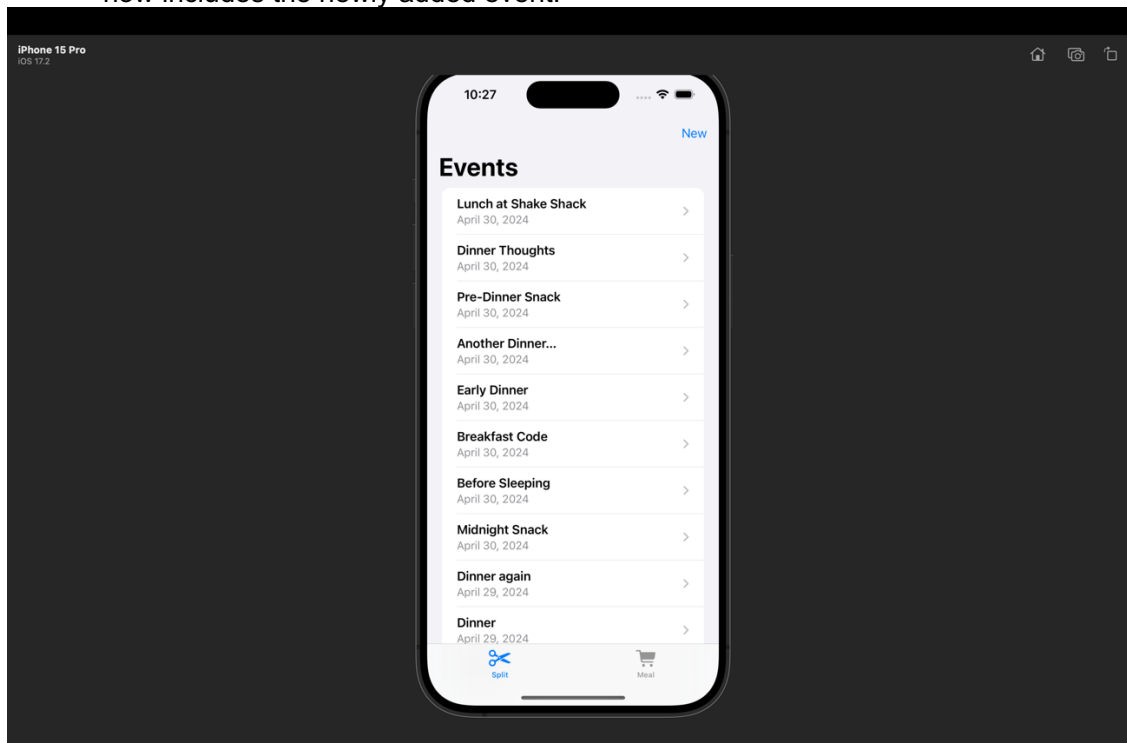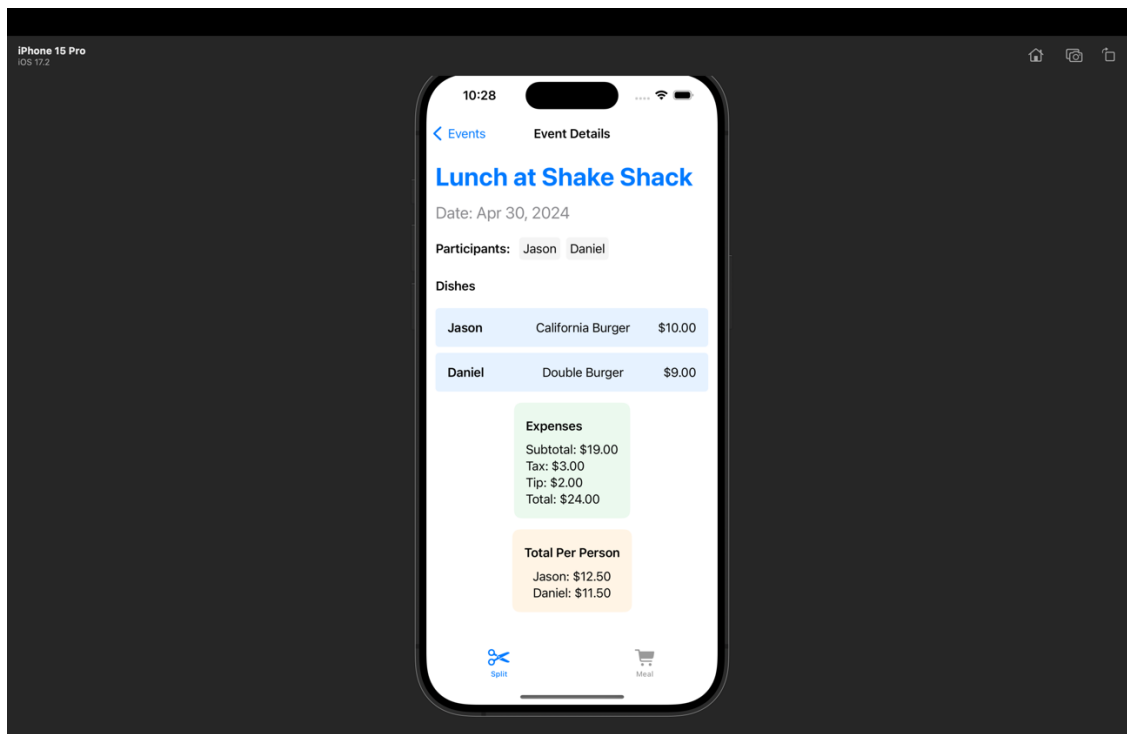
10 View Updated Events: Post-event creation, users are redirected back to the Split tab which now includes the newly added event.


11. Event Details: Clicking on an event opens a detailed view where all particulars are displayed, including the calculated total for each participant respective of the tax and tips.

12. **Meal Search**: If opting to stay in and cook, users can search for recipes by entering a dish name in the search bar and pressing enter.
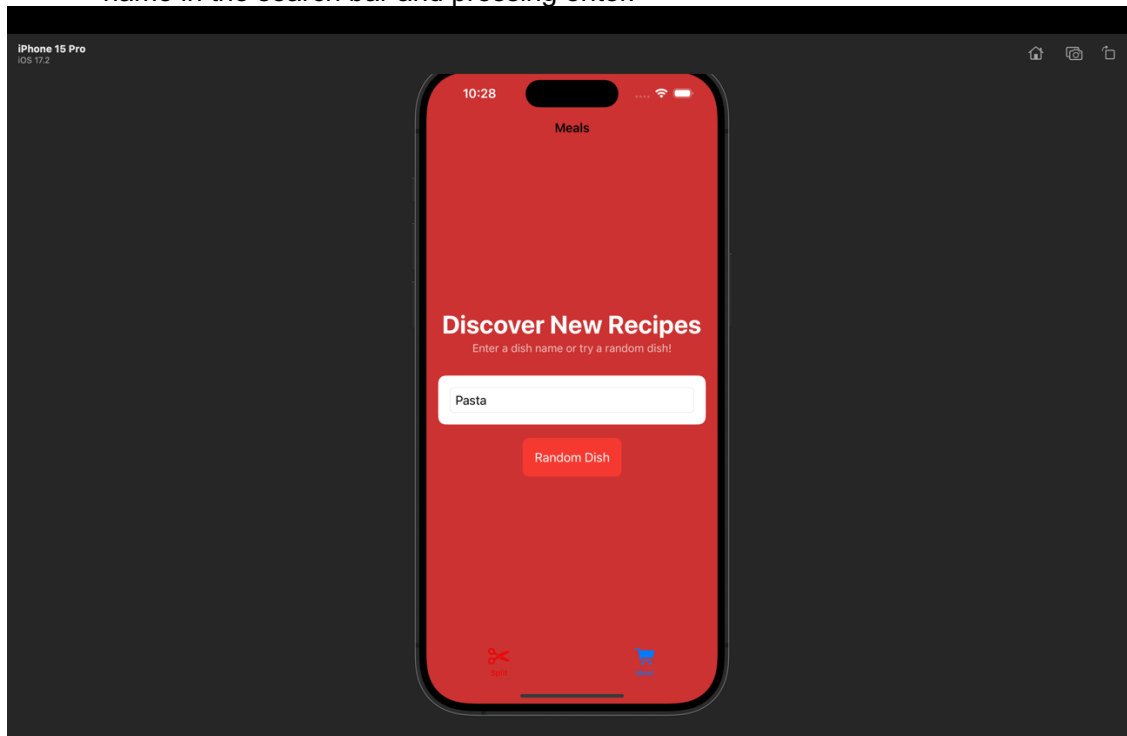
13. **Exploring Dish Details**: The search reveals detailed information about the dish, including images, full name, category, are/origin, cooking instructions, tags, and a YouTube tutorial link.
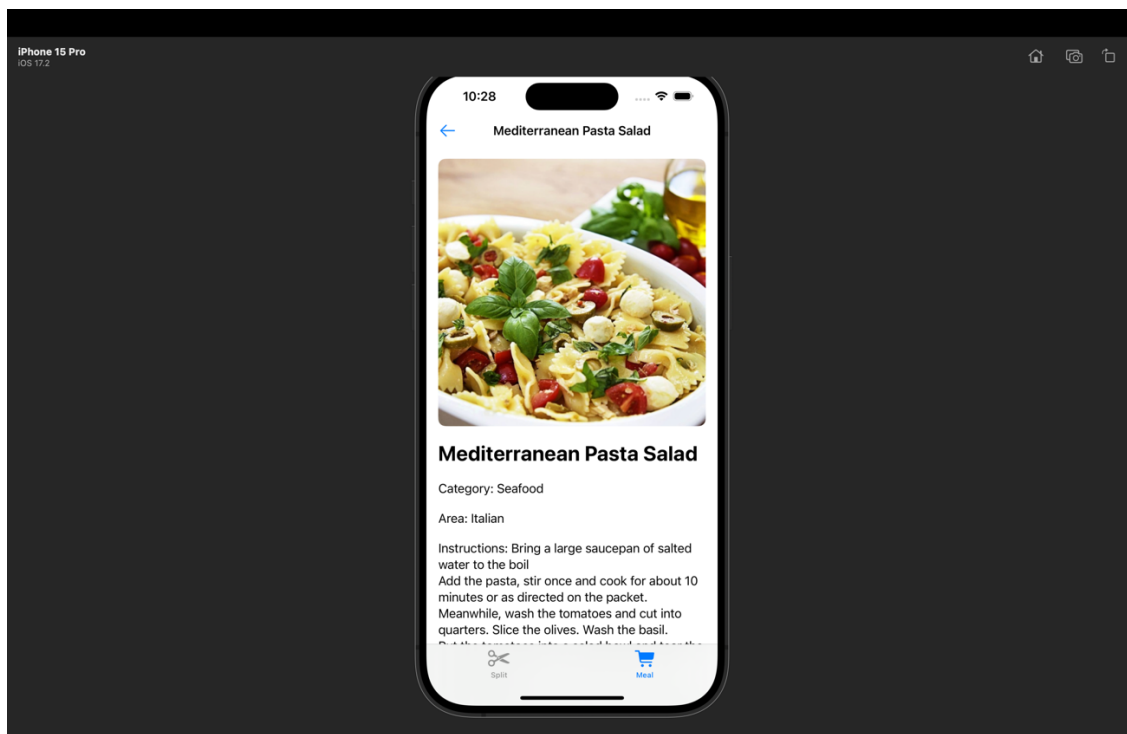
14. Discovering New Dishes: To explore new recipes, click the "Random Dish" button.
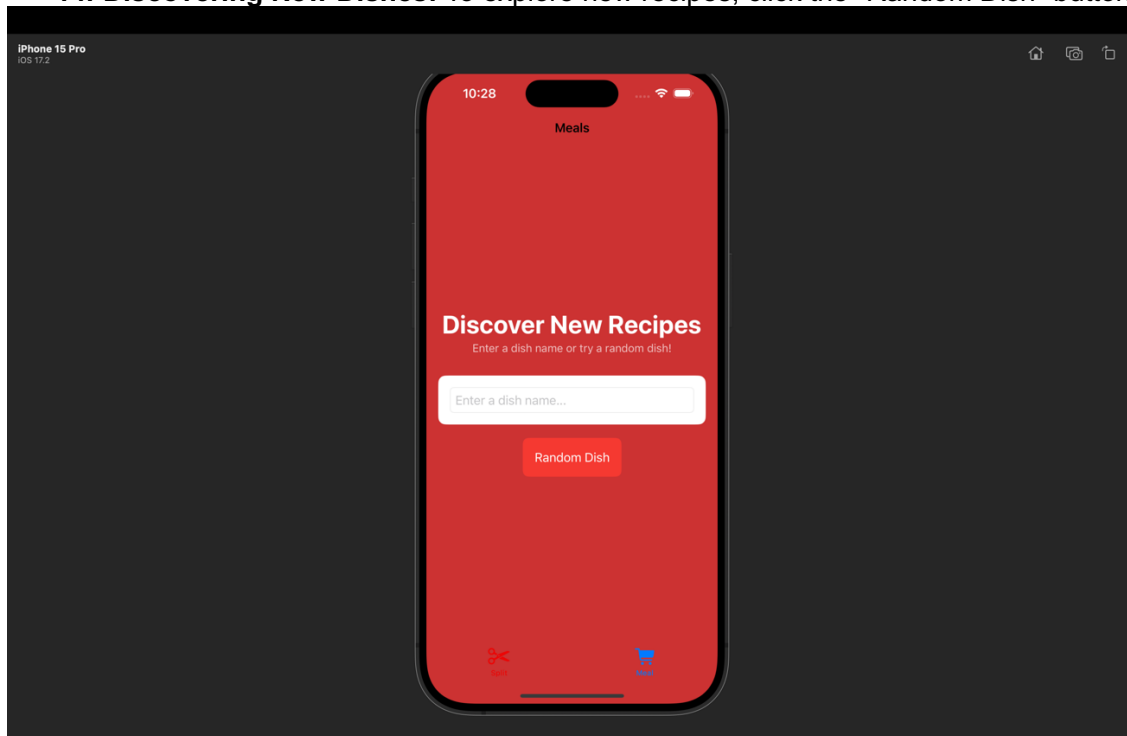
15. Random Dish Display: A random dish will appear, complete with comprehensive details.

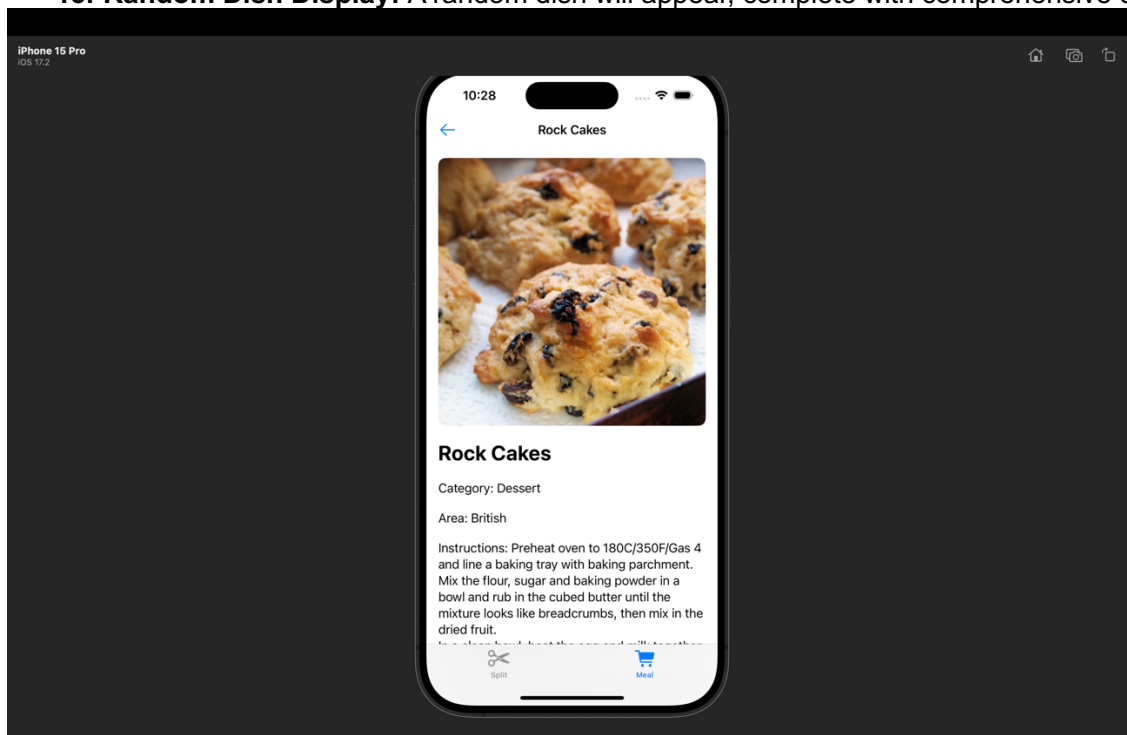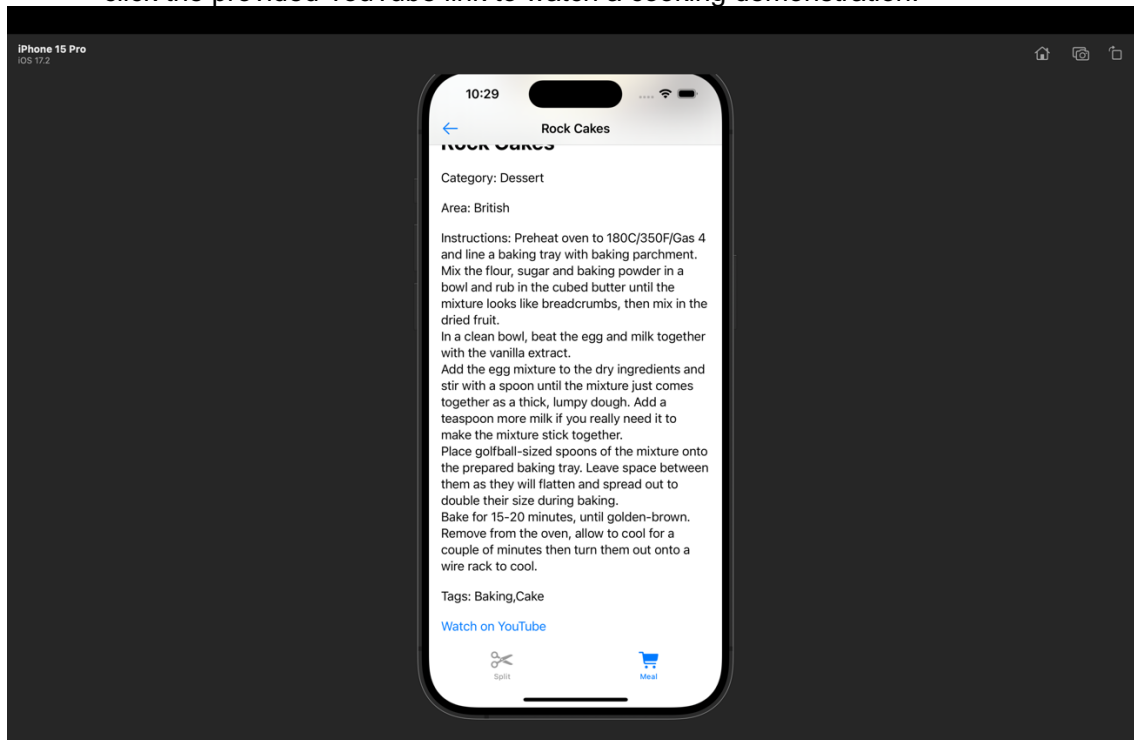16. **Additional Information:** Further scrolling provides more insights into the dish. Users can click the provided YouTube link to watch a cooking demonstration.

Technology Highlights

Firebase Authentication: Manages user authentication processes.
Cloud Firestore: Stores and retrieves user events and associated participants, dishes, and expenses.
API: Utilizes external APIs to fetch dish recommendations.

• Model objects
• Abstraction of back-end functionality
• Feedback for operations-in-progress
• Error handling and messaging
• Professional finance inclined design and color palette
• Proper choice of input views and controls
• Animations and Transitions
• Programmed graphics
• Custom app icon

Acknowledgements and Credits:

ThemealDB for my API
Link to their website: https://www.themealdb.com/api.php

Rules for Cloud Firestore:

```
service cloud.firestore {
  match /databases/{database}/documents {

    // Matches any document in the 'events' collection
    match /events/{eventId} {
      // Allow read/write if the user is authenticated
      allow read, write: if request.auth != null;

      // Matches any document in the 'participants' sub-collection of an event
      match /participants/{participantId} {
        allow read, write: if request.auth != null;
      }

      // Matches any document in the 'dishes' sub-collection of an event
      match /dishes/{dishId} {
        allow read, write: if request.auth != null;
      }

      // Matches any document in the 'expenses' sub-collection of an event
      match /expenses/{expenseId} {
        allow read, write: if request.auth != null;
      }
    }
  }
}
```