PocketChef

Anh Le, Ritika Roy Choudhury, Vani Anilkumar
Group #42

Brief Introduction

- **Problem**: The difficulty of meal preparation in lives of adults who don’t have time & energy to spend on meal-prepping.
- **Solution**: A smart recipe recommendation system to recommend diverse dishes to users based on their preferences and simplify the meal-planning and grocery shopping process.

1.
What is PocketChef?

Goals, User Flow

Goals and Functionality

- Get a **personalized menu plan** daily based on user's personal model: allergens, diet preferences, etc.
- Get a recommended **average daily calories** score from personal model: weight, height, average activity, gender.
- **Search for recipes** from your personalized index or from the internet to put in menu.
- Dynamic list of **grocery items** needed every day easily accessible.

How The Personalization Works

- User’s personal model **(input)**:
  - Height, weight, age, gender, diet type, allergies
- User’s personal model **(dynamic)**:
  - User’s steps and calories burned data from the iPhone Health app.
- Recipes from the Spoonacular API initialize the inverted index for each user to fill the menu and keep track of user’s most frequently eaten and Favorite Recipes

Flutter

The Setup

- Used **Flutter** and Xcode to build and run this **iPhone** application.
  - Flutter uses the Dart.js framework
- Used a **Firebase** server to store the databases for the user's information.
- Used the **Spoonacular API** to store and fetch all dishes and full recipes.

2. Components

The What's and The How's

List of JS Components Used

Front-End:

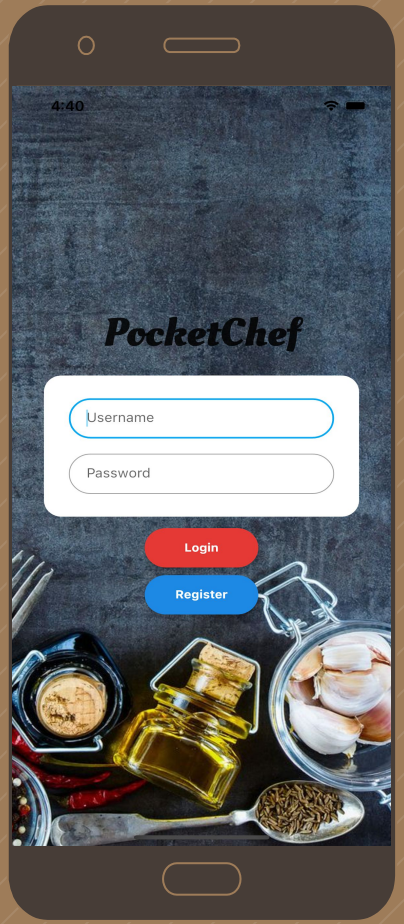
- **LoginScreen**: Login or register
- **HomeScreen**: View personal stats and Daily Menu
- **SearchScreen**: Search for recipes online or in user's personal database.
- **GroceryListScreen**: View ingredients required for the three dishes on the menu

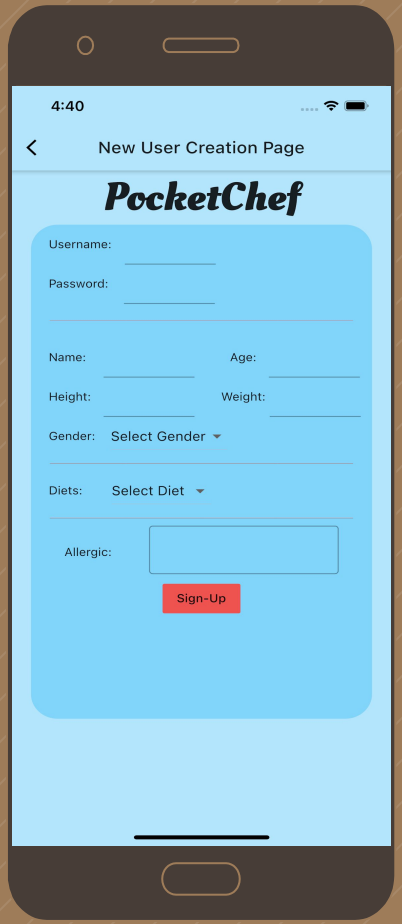
Back-end:

- **MainUser**: Store/manipulate the personal model
- **Dish** and **DishInfos**: Store/ manipulate dish information from the JSON API response
- **Recipe**: Full Recipe from JSON API response
- **UserInputScreen**: Populate user's personal database and the day's menu

Login Screen

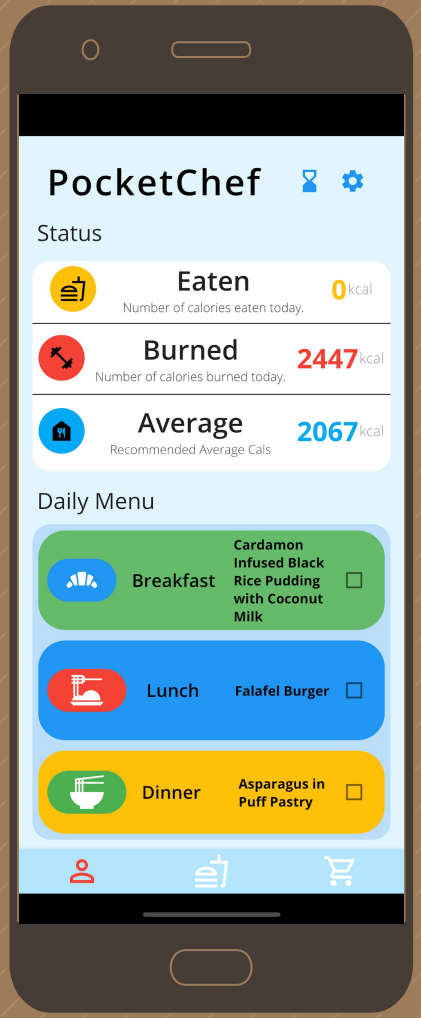
- Either login or register for a new account.
- Fields in registration page used to build personal model.
- Push user data to the Firestore database.
-

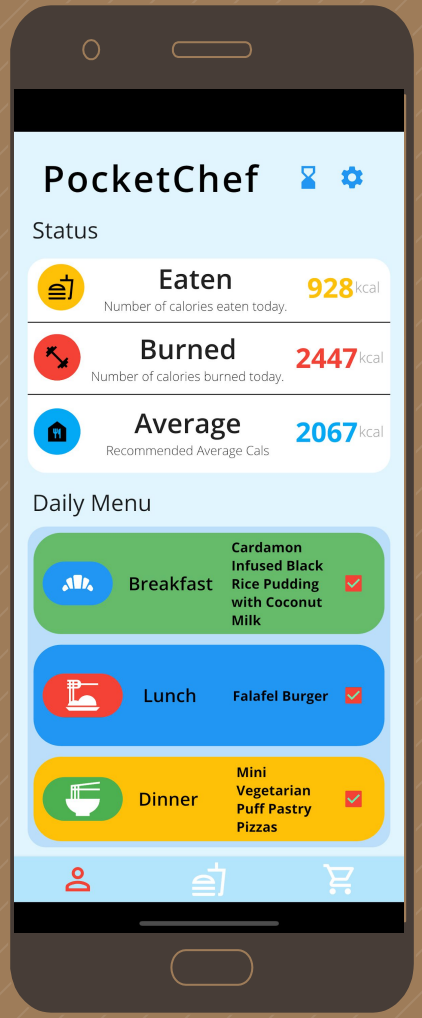




Home Screen

- View stats for calories eaten (dynamic based on checked items), calories burned (Health App data), and recommended daily calories
- Once all recipes are checked off/eaten, we generate a new menu for the user (recommendation algorithm)
- Click on any meal icon to view and favorite a recipe.



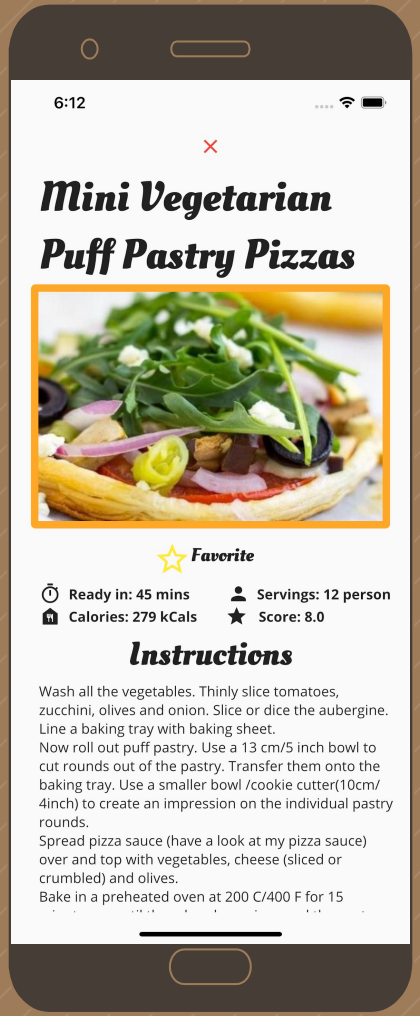


How Do We Calculate Recommended Calories

- We use the **Harris-Benedict formula** to calculate **BMR** (total number of calories a person needs):
  - For women: BMR = 655.1 + (9.563 * weight in kg) + (1.850 * height in cm) - (4.676 * age in years) * activity factor
  - For men: BMR = 66.47 + (13.75 * weight in kg) + (5.003 * height in cm) - (6.755 * age in years) * activity factor
- For the activity factor, we use the data from the Health App to categorize how active the user is based on the calories they burn and calculate accordingly.
- Sources: https://www.verywellfit.com/how-many-calories-do-i-need-each-day-2506873, https://fitfolk.com/average-calories-burned-per-day-men-women/

Show Recipe in HomeScreen

- See an image of the dish and other information.
- View the instructions.
- Favorite button to mark that you like the recipe and want to see more like it.

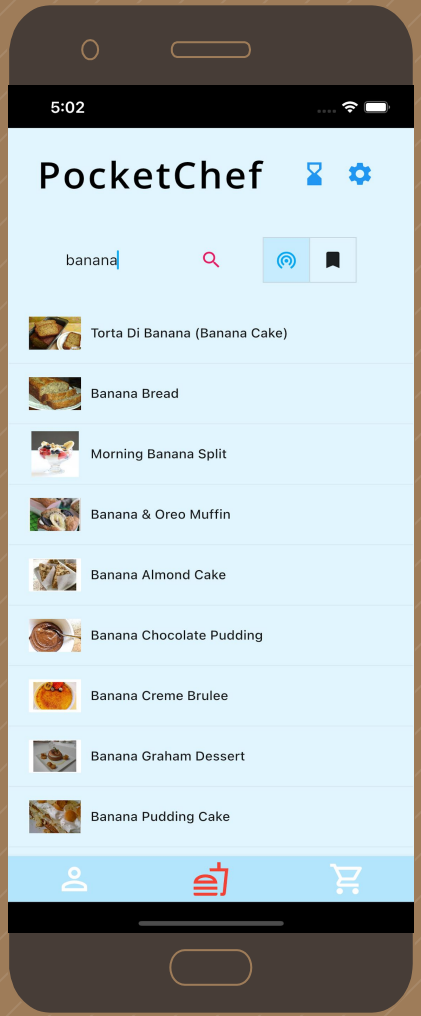


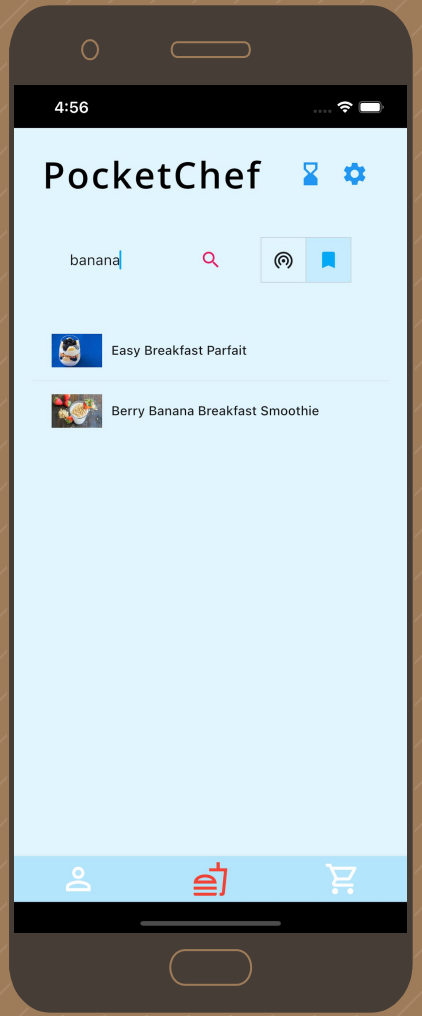
How Do We Recommend a New Menu?

- We initialize each user with a personal inverted index of dishes upon creation that suit their diet/etc.
- 2 of the generated meals:
  - If the user has favorited meals, we recommend **similar dishes** from a **Spoonacular** API.
  - If not, we recommend meals similar to those the user has in their personalized index.
- 1 of the generated meals:
  - We recommend a meal from the inverted index that the user has eaten frequently and enjoys, or a meal that they haven’t eaten in a while.
- As the user keeps using the app, the more robust and diverse the recommendations get.

Search Screen

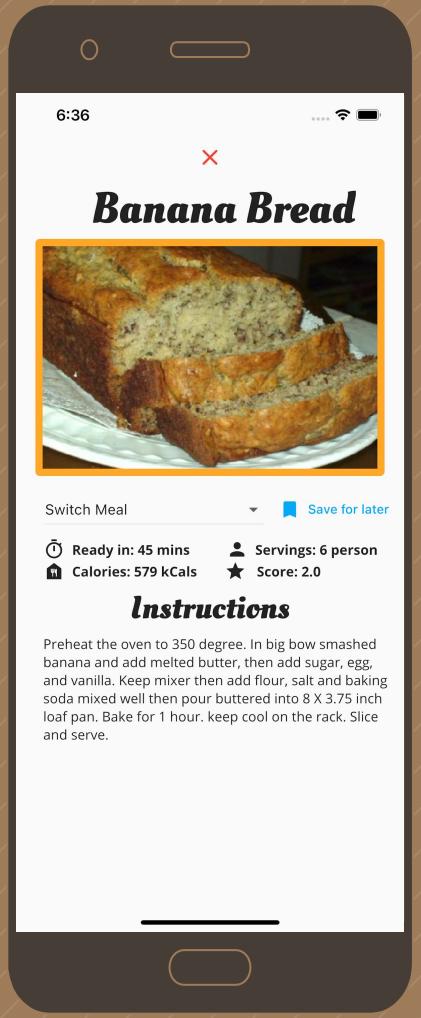
- Toggle between showing recipes from the web or the personal index.
- Search and scroll through top results.
- We implemented our own search system for the personal index search (using inverted index).

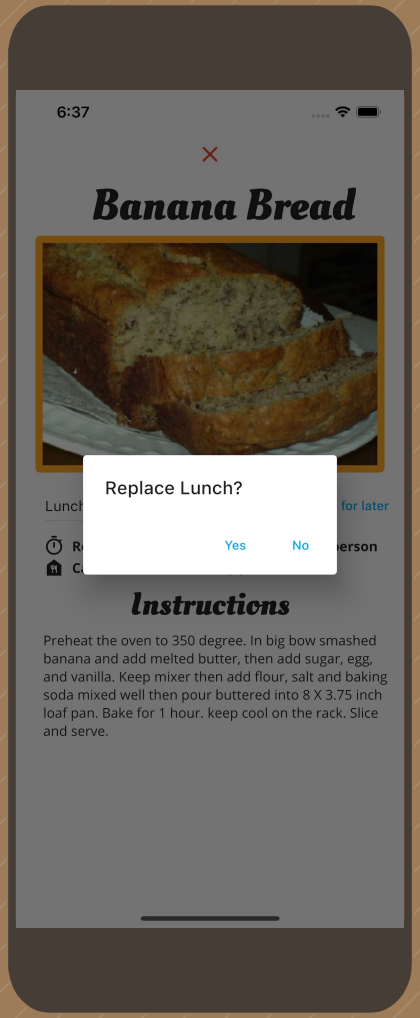




Clicking on a Recipe

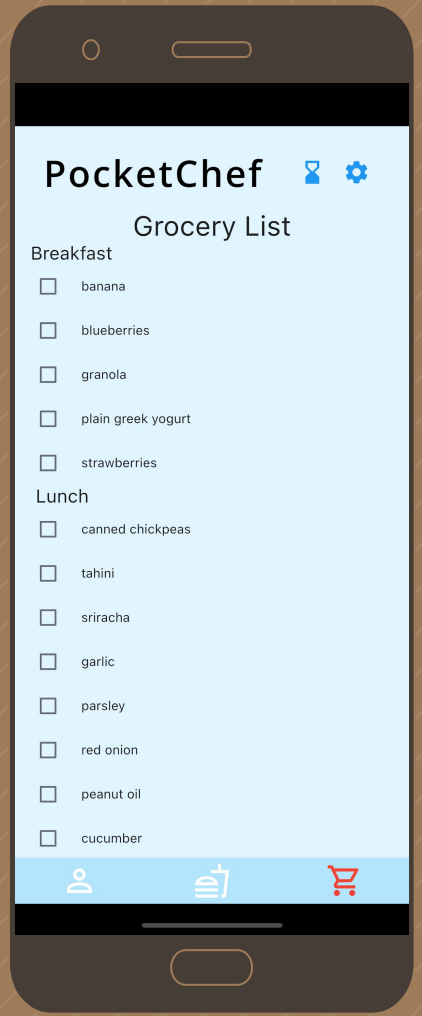
- Recipe page shows image, instructions, etc.
- You can choose to replace a meal in the daily menu with the selected dish, or save it for later and add to the user's index.

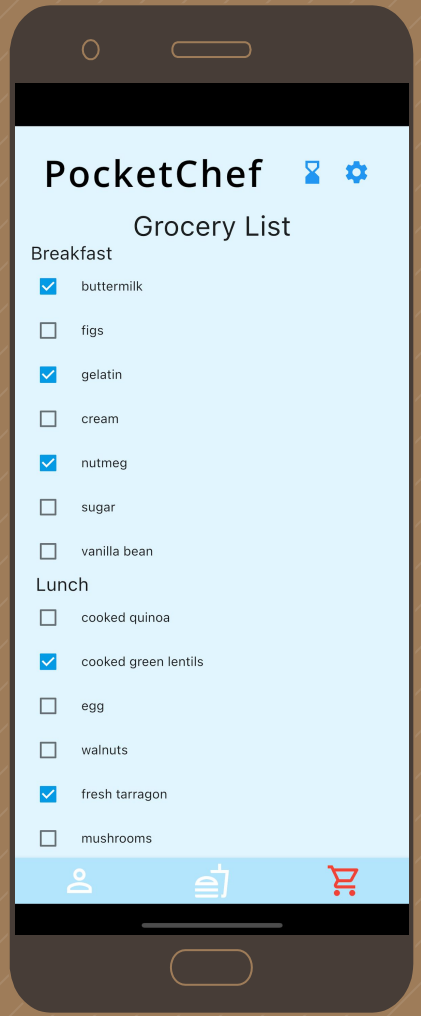




Grocery List Screen

- Dynamic list of ingredients for each meal from the Daily Menu.
- Separated into breakfast, lunch, and dinner categories.





Spoonacular APIs used

- **Generate meal plan**: Initial index and Daily Menu created.
- **Fetch dishes** based on a user query: Search Screen; returns recipe IDs, titles.
- **Fetch full recipe:** Uses recipe ID to get full instructions, extended ingredients list, etc. Our most used API!
- **Fetch similar recipes**: Used to create a new Daily Menu once all the dishes have been eaten. Takes in recipe ID and returns list of recipe IDs, titles.

3.
Further Considerations

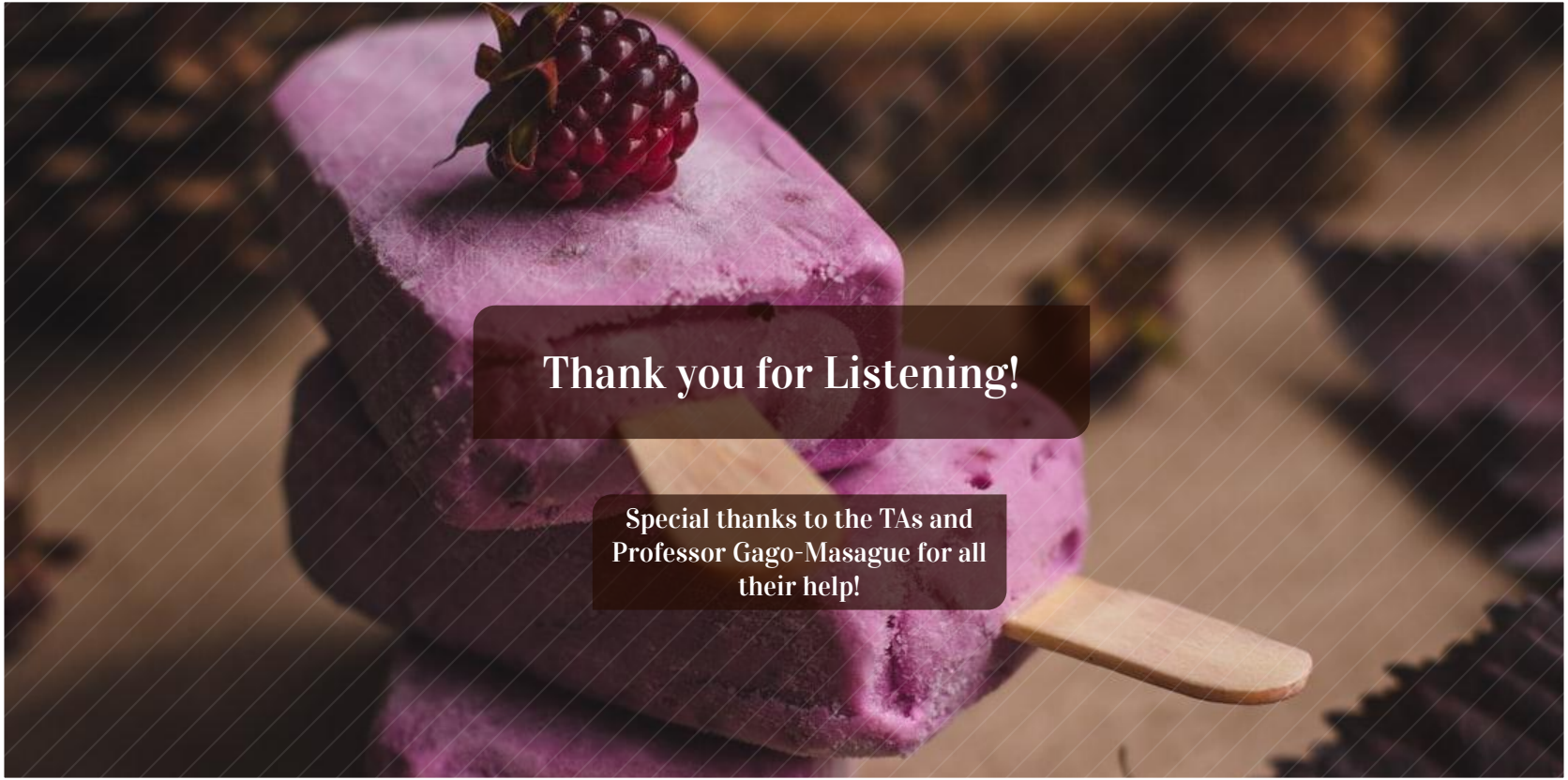
Phew! That was a lot.

Challenges and Roadblocks

- Asynchronous request handling! Future methods in Dart.js
- Appropriately using the Firebase database to store information, call information, and model information in a way that makes sense to store online.
- Getting the right data and parsing the JSON response for the recipes from Spoonacular.

What We'd Improve

- Automatically switch the daily menu at midnight every day.
- Add in functionality to change their diet/allergens, and other information after creating the user.
- Expand on the grocery list functionality to route users to local grocery stores nearby with the items to buy.



Thank you for Listening!

Special thanks to the TAs and Professor Gago-Masague for all their help!