Development of an AI-Driven Personal Meal-Planning Application

by
MADISON COBB

Advisor
DR. JOSHUA ECKROTH

A Senior Research Paper submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mathematics and Computer Science
in the College of Arts and Science
at Stetson University
Deland, Florida

Spring Term
2024

ABSTRACT

In this paper, we detail the process of building a mobile phone application that uses ChatGPT as the primary means by which to generate recipes, and plan schedules of meals to be prepared and eaten by the user over the course of their day. The goal of this project is to create a product that helps the user in the course of planning and making meals for themselves, and hopefully along the course of use obtain a better understanding of the process of preparing food in the kitchen, allowing for more independence and agency over their diets and the food that goes into their body. Utilizing this AI-driven application, the cognitive overhead that normally comes with planning and preparing a course of meals to eat for themselves or for a larger group, like a family or house unit, will ultimately be reduced by expediting multiple key steps in the process. These steps include finding recipes that cater to the user's specific dietary and nutritional needs, planning out those recipes to eat over a given period of time, going through those recipes and deriving what ingredients and cooking supplies you need and comparing to what you have readily available, and trying to find the most effective way to balance having a nutritionally whole diet while not breaking the bank in buying unnecessarily expensive ingredients. Ultimately, using this app should make it easier and more accessible than ever to learn how to cook food for oneself, while simultaneously allowing for the curated creation of a personalized, sustainable diet consisting of foods the user enjoys making and eating, especially for those who might not otherwise have the knowledge, skills, money, or time to prepare these things for themselves. By the end of this paper, you will see how we've researched , planned, and implemented each of these components, and ultimately how we've leveraged the generative power of Large Language Models like ChatGPT to make a better and more robust product for the end user than could be accomplished through conventional logical programming.

I. Introduction

Cooking meals for oneself is oftentimes a daunting and laborious undertaking that many people simply don't have the time or energy to accomplish. The labor and planning involved just in maintaining a functional kitchen with a sufficient set of appliances and a consistently stocked pantry full of staple ingredients you'll actually use often becomes overwhelming, even to the most seasoned of chefs. And, in a time where on-demand meals from your favorite restaurants are just a button's click away through any number of meal-delivery apps, never has it been more tempting to put thoughts of home-made meals aside in favor of the vastly more convenient, albeit less economically-viable option of having food dropped off right outside your door. The convenience of such options naturally leads to more people shirking the effort it takes to prepare a meal for oneself and, understandably, putting their energy into matters that take higher precedence, like work for a job or for school.

With this project, we propose a solution to the conundrum of eating better vs. eating easier by reducing the amount of work and decision making done on the individual's end. This will be accomplished by automating the processes of what can be some of the most obstructive hurdles when it comes to home-cooking: deciding what and when to cook for oneself, and keeping necessary ingredients stocked in the pantry to pull out for nutritionally whole, staple meals. This project also aims to optimize the individual's usage of their ingredients, to the end of reducing the financial burden placed on the user by buying less, and using more. Our solution will take the form of an android mobile phone application, of which the primary functionality will be centered around generating a day-by-day schedule of meals to eat, drawing from a database of ChatGPT-generated recipes that the user can continually populate with meals they like or want to try. Having a single central list of desired meals and an automatically generated schedule of when to eat each one, along with a ready-made grocery list of all the ingredients needed to make them, the user can have a much easier experience in getting to consistently prepare meals for themselves.

The primary components that will make this app work consist of a ChatGPT-powered recipe generation system that uses advanced prompt engineering techniques to craft personalized recipes, a meal scheduler that automatically generates ready-made meal plans for a given period of time and continually adapts itself to the user's food consumption habits, and a robust grocery list generator that takes into account ingredients on hand and tracks ingredient usage to automatically suggest when certain stock may need to be replenished, which allows for grocery store trip planning that fits in with the user's busy schedule and caters to any financial limitations the user may have. In order to understand how exactly we can build our app's functionality around ChatGPT-based generation methods, let's take a moment to explore how exactly GPT and, by extension, Large Language Models as a whole work on a mechanical level, what specific goals they're engineered to accomplish and how we can leverage their utility for our own purposes.

II. Background and Related Works

The term "Large Language Model" refers to currently the most robust and powerful form of a language model, utilizing neural networks and deep reinforcement machine learning to create a probabilistic model of a given human language, which would be English for our purposes. The concept of a language model has roots stretching back to the 1980s, when linguists proposed a way to model the most probable continuation of a given sequence of words in a language. The first language models were pure statistical models, using complex manual diagramming in order to model the probabilities of given word sequence continuations. More recently, researchers have been able to leverage the power of various implementations of neural networks to make these predictive models more dynamic, robust, and most importantly accurate than ever. These developments are what have ultimately brought us to the stage of ChatGPT, what a state-of-the-art predictive language model that can produce deeply complex, conversational, human-understandable outputs that are predictive responses to a given human-written query.

So, the fundamental question we're left to ask is, what kind of practical implications does this technology preclude us in the state we have it today? Well, language models are the technical basis of a plethora of modern technologies that many of us utilize in our day-to-day lives. Probably the most direct example is the Auto-Complete feature present in practically all modern smartphones. There's not much more of a directly analogous use case for predictive text generation than literally guessing and suggesting the most likely word that a user is going to type next. Other examples of practical implementations of this technology include audial speech recognition, machine interlingual translation, and automated information retrieval. That last use case is what we're focused on in our implementation of this application: we seek to utilize this human-readable generative text in order to synthesize the most likely linguistic representation of a recipe that is followed by a user to make a food dish. Our goal here is ultimately to test the mettle of this technology by applying it to one of the most important aspects of day-to-day human activity, that is deciding and preparing the food that we put into our bodies, and seeing how well this predictive generation holds up to snuff when it comes to practically informing decisions made by human users in ways that will directly impact their health, wellbeing, and financial stability. Next, we'll explore other contemporaneous implementations of this technology towards similar ends, or put more simply, other currently existing apps that utilize ChatGPT to make recipes and meal plans.

Upon initial investigation, it’s easy to see that there are already many applications that implement ChatGPT for meal planning and recipe generation in a similar manner to how we plan to. Some examples include ChefGPT, DishGen, and FoodAI, all of which slightly differ in their approach to making recipes using ChatGPT and other AI implementations. A lot can be learned from observing the different approaches that each app takes in implementing AI to generate recipes, with different degrees of user input. Some applications allow users to select some from a list of ingredients, and generate recipes that use those ingredients exclusively or non-exclusively, based on user choice. However, in the implementation of our application, we plan to take an approach that takes as much of the dictation of minute details out of the hands of the user as possible. Many of these apps also have the features we plan to develop, like meal scheduling and dietary considerations, locked behind paywalls, which is less than optimal from an accessibility standpoint. And while the financial aspect of things would definitely need to be considered and reckoned with in a full-scale production and distribution of our application, in an ideal scenario we would like to make these tools as widely and easily accessible as possible.

Another important topic of research going into the development of this app is finding ways to make the best use of ChatGPT to produce more optimal results for our application, and ultimately make a better experience for the user. By restructuring and fine tuning the prompts that you give to an LLM like ChatGPT, you can minutely alter the responses it outputs to gradually work towards generating optimal outcomes, which for our purposes will be the production of consistently formatted and indexed JSON files that will act as the recipes stored in our app for consistent code results. This concept of prompt engineering has become an important subject of discussion and research ever since the latest generative AI tools like ChatGPT and DALL-E have been released to be used in technological fields, and even a much wider public audience than has ever really been reckoned with when it comes to AI, with swathes of time and resources put into continually developing better and better prompt structures to increase the effectiveness of these already very powerful tools. There's a financial aspect to generating well written and efficient prompts as well; the ChatGPT API charges per token when using newer models or implementing it in such a wide scale such that a large degree of queries are sent from the user end through the application that implements it. Naturally, in either of these cases, we'd want to make our prompts as succinct as possible while still producing favorable results in order to minimize the financial burden placed on us as the developers implementing this API. Similar to before, this would be a higher priority consideration if we were to develop a full distribution of this application, but for research and development purposes at the point where we are now, we can concede consolidating the size of our prompts if it means producing better results.

III. Design and Methodology

Our application has been under development using Android Studio, primarily utilizing a Java interface to Android's native Kotlin language for programming logic, and using XML markdown for the design and structuring of our UI. The application's functionality is structured around 3 primary components, implemented in a pseudo-hierarchical structure with each piece generating data and then feeding those output results to the next component in the chain. The first component is the user-specific database of recipes generated by ChatGPT, that will grow as the user continues to use the app.

In our current scope, there are two ways that ChatGPT can generate recipes to populate the user's Recipe Book. First, given the name of a meal that the user selects or inputs, generate a recipe for that meal. The below visualizations show the system level interactions with ChatGPT that take place when a user goes to generate a recipe with this method, and the user-end implementation of this function will be shown in a later section.

Figure 1.

User Input	**System Input**
(Generate Recipe By Name) -> Input Recipe Name: **Chicken Caesar Salad**	chatGPT("Write a list of ingredients needed to prepare and make [**Chicken Caesar Salad]**. Return the recipe in JSON format. Minimize unnecessary ingredients, making it as simple and cost-effective to construct as possible. ")
System Output	

{"recipe": "Simple Chicken Caesar Salad",
"ingredients": {
"chicken": {
"type": "chicken breasts",
"amount": "2",
"preparation": "cooked and shredded"
},
"romaine lettuce": {
"type": "romaine lettuce",
"amount": "1 head",
"preparation": "washed and chopped"
},
"croutons": {
"type": "croutons",
"amount": "1 cup",
"preparation": "store-bought"
},
"Caesar dressing": {
"type": "Caesar dressing",
"amount": "1/2 cup",
"preparation": "store-bought"
},
"parmesan cheese": {
"type": "Parmesan cheese",	

<table>
<tr><th>User Input</th><th>System Input</th></tr>
<tr><td colspan="2">"amount": "1/4 cup",
"preparation": "shredded"
}
},
"instructions": [
"Cook the chicken breasts until fully cooked and shred into bite-sized pieces.",
"In a large bowl, combine the chopped romaine lettuce, shredded chicken, and croutons.",
"Add Caesar dressing to the salad and toss until ingredients are evenly coated.",
"Sprinkle shredded Parmesan cheese over the top and toss again.",
"Serve immediately, optionally garnishing with extra Parmesan and croutons."
]
}</td></tr>
<tr><td colspan="2">Output to User</td></tr>
<tr><td colspan="2">“Recipe Added: Chicken Caesar Salad. We’ll start including this meal in your Meal Plans!”</td></tr>
</table>

The second method we have implemented uses ChatGPT to generate a recipe based on the list of ingredients provided by the user that they already have on hand. This list will be what consists of the user’s Pantry, a part that will be further detailed later in this paper. At any point, the user is able to go to their pantry page (in the same place where the Grocery List is), and press a button that tells the app to generate a recipe from their pantry contents. The user can either approve or disapprove of the generated meal; if approved, it will be slotted into the soonest available/thematically appropriate meal slot. If disapproved, the user can ask for a second generation, or back out of the tool and exit to the main app. The following figure again visualizes the system-level interaction that takes place with ChatGPT when the user decides to generate using this method.

Figure 2.

<table>
<tr><th>User Input</th><th>Sample Pantry Contents</th><th>System Input</th></tr>
<tr><td>Viewing pantry -> (Generate Recipe from Contents)</td><td>Avocados
Bananas
Apples
Bread
Peanut Butter
Jelly
Salsa
Tortilla Chips</td><td>chatGPT("Create a simple recipe from a list of possible ingredients. Return the recipe in JSON format. You do not need to use all of the ingredients listed.
Ingredients: [Sample Pantry Contents]")</td></tr>
<tr><td colspan="3">System Output
{
"recipe": {
"name": "Peanut Butter and Jelly Sandwich",
"servings": 1,
"ingredients": {
"bread": "2 slices",
"peanutButter": "2 tablespoons",
"jelly": "2 tablespoons"
},
"instructions": [
"Take 2 slices of bread.",
"Spread 1 tablespoon of peanut butter on one side of each slice.",
"Spread 1 tablespoon of jelly on top of the peanut butter on one of the slices.",
"Place the other slice of bread on top, peanut butter side down, creating a sandwich.",
"Press the slices together gently.",
"If desired, cut the sandwich in half for easier handling.",
"Enjoy your Peanut Butter and Jelly Sandwich!"
]
}
}</td></tr>
<tr><td colspan="3">Output to User
"We've found that with your current pantry contents, you can make a Peanut Butter and Jelly Sandwich. Would you like to queue this recipe up in your Meal Plan? " Yes/No
"Okay, it has been added to your Meal Plan. Would you like to save it to your Recipe Book?" Yes/No</td></tr>
</table>

When a recipe is generated via either of these methods, it is stored and added to the user's Recipe Book, so that they can easily look through and see what meals are going to be put into their Meal Plans. Visualizations of the user’s recipe book will be seen in the Implementation section of this paper.

Next, with the user’s initial Recipe Book entries generated, the next component can be used: the Meal Plan generator. ChatGPT is also utilized for this portion of the app’s functionality, making meal plans for the day based off of the recipes stored in the user’s Recipe Book. For a single given day, ChatGPT will pick at least 3 meals in the user’s Recipe Book to be scheduled for that day’s breakfast, lunch, and dinner. Exactly how many meals or what each time slot can be occupied by should be able to be manually adjusted by the user, depending on their schedule or general preferences and habits when it comes to eating, however the app’s current functions just account for the 3 primary meals of the day. The following figure shows the system level interaction with ChatGPT that takes place when performing this function.

Figure 3.

<table>
<tr><th>User Input</th><th>Sample Recipe Book Contents</th><th>System Input</th></tr>
<tr><td>(Generate Meal Plan)*

*Will generate if Recipe Book population > 3</td><td>Bagel with Cream Cheese
Deli Meat Sandwich
Orange Chicken Rice
Peanut Butter Jelly Sandwich
Chicken Caesar Salad</td><td>chatGPT(“Generate a single day's meal plan, consisting of 3 meals based on the following list of recipes. You do not need to use all of the recipes listed. Output the plan in JSON format. Recipes: [Sample Recipe Book Contents]”)</td></tr>
<tr><td colspan="3">System Output
{
  "day": "Monday",
  "meal_plan": [
        {
        "meal": "Breakfast",
        "recipe": "Bagel with Cream Cheese"
        },
        {</td></tr>
</table>

```
        "meal": "Lunch",
        "recipe": "Deli Meat Sandwich"
        },
        {
        "meal": "Dinner",
        "recipe": "Orange Chicken Rice"
        }
 ]
}
```


Output to User
Today's Meal Plan:
Breakfast
- Bagel with Cream Cheese

Lunch
- Deli Meat Sandwich

Dinner
- Orange Chicken Rice

Finally, with a Meal Plan set for the next stretch of planning days, the app will take all of the upcoming selected recipes and put all of the ingredients needed to make them into the third piece: the Grocery List. Since the recipes are stored in the JSON format, the grocery list can be easily procedurally generated just by parsing through the files set to be scheduled for the user's meal plan, and each unique ingredient field can be added to the grocery list. First, the ingredient item is checked against the contents of the user's pantry, to make sure we aren't adding something they already have access to. If it's not present, then the item is added to the grocery list. This process is repeated for every ingredient needed for the upcoming meal plan. Then, when an ingredient is checked off of the grocery list, indicating that it has been purchased, it's summarily added to the user's Pantry. When the ingredient has been used in enough recipes generated by the meal planner, the ingredient can then be removed from the pantry, and added back onto the grocery list if it's something the user will need again for upcoming meal plans.

Ultimately, with these interconnected pieces put in place, we now have a pretty powerful tool that is able to generate customized recipes for the user while utilizing a fraction of the processing power and logical procedural programming that would normally need to be implemented to make an application as robust, dynamic, and powerful as we have created now utilizing the power of generative AI. A tool like this is also extremely helpful to the user, allowing for much of the effort and decision-making that normally goes into manually planning out meals to be taken out of their hands and relegated to this handy AI assistant, allowing them to focus more of their energy and time on the things that matter, like actually making the food.

In creating an app that utilizes ChatGPT to the capacity that we do, we find that one of the problems that lie at the core of making this app useful is one of finding optimal ways to query the AI so that it may output responses that are both consistent and useful; Consistent in that the formatting and style of the bot's answers will always be the same so that all procedural logic left can be uniformly applied to anything the AI generates, and useful in that these outputs generated by ChatGPT will be suitable to the wide variety of tastes and preferences that an end user may have. So, a large part of the research and developmental experimentation that has been done over the past year is on how best to structure the phrasing of our queries to ChatGPT in order to produce these optimal results.

As you can see upon examination of the demonstrations of the two recipe-generating tools depicted in Figures 1 & 2, despite having similarly structured prompts, the actual content of the JSON outputs are structured slightly differently. This is found in testing when different recipes have different degrees of specificity when it comes to what and how ingredients are used. Our solution to this is to provide a sample JSON output with just the bare structure in place so that every recipe that is held in the app's database can be processed and read in the same way.

The JSON structure has been omitted from the previous figures for sake of brevity. The main downside of using this method is that a great deal more of our valuable space in the prompts to GPT is taken up in specifying the formatting of its answers, which ultimately decreases the financial efficiency of each prompt sent through. Ultimately, the benefits in this case outweigh the downsides, as this is just an experimental implementation of this technology. However, if this app were to be fully developed and distributed to a wider user base, we would need to do a lot more work to maximize the efficiency of the prompts we send through to ChatGPT.

There are lots of places in this app where use of optimization techniques could prove beneficial for both the user and us as developers. On a user end, different forms of optimization could be implemented to decrease the costs associated with going to the grocery store, by intelligently considering how much of a certain ingredient by mass is needed, the app/ChatGPT could find the best brands, the best package sizes, and ideally the lowest cost/weight ratio available to the user when purchasing their ingredients. In the meal planning step, the app could consider how many meals contain similar ingredients, and plan accordingly to minimize the number of unique ingredients needed to make multiple different meals when scheduling for the next planning period. With these factors in mind, let’s look at the steps of optimization that have been put in place thus far.

The majority of optimization that has been put in place has taken the form of modifying the prompts we pass through to ChatGPT in order to optimize the generation of recipes to use as few extraneous and unnecessary additive ingredients as possible. Additionally, steps of optimization have been put in place in the Meal Planning step in order to try and re-use as many ingredients present in the pantry and on the grocery list as possible, by scheduling meals for different days that use similar sets of ingredients to both break up monotony of eating the same dish for multiple days or meals in a row while still making optimal use of the ingredients the user has available to them. Addendums to ChatGPT prompts we've implemented to achieve these optimization goals are shown in Figures 1-3, including the phrases:

"Minimize unnecessary ingredients, making it as simple and cost-effective to construct as possible."

"You do not need to use all of the recipes (Meal Plan Gen)/ingredients (Pantry Recipe Gen) listed."

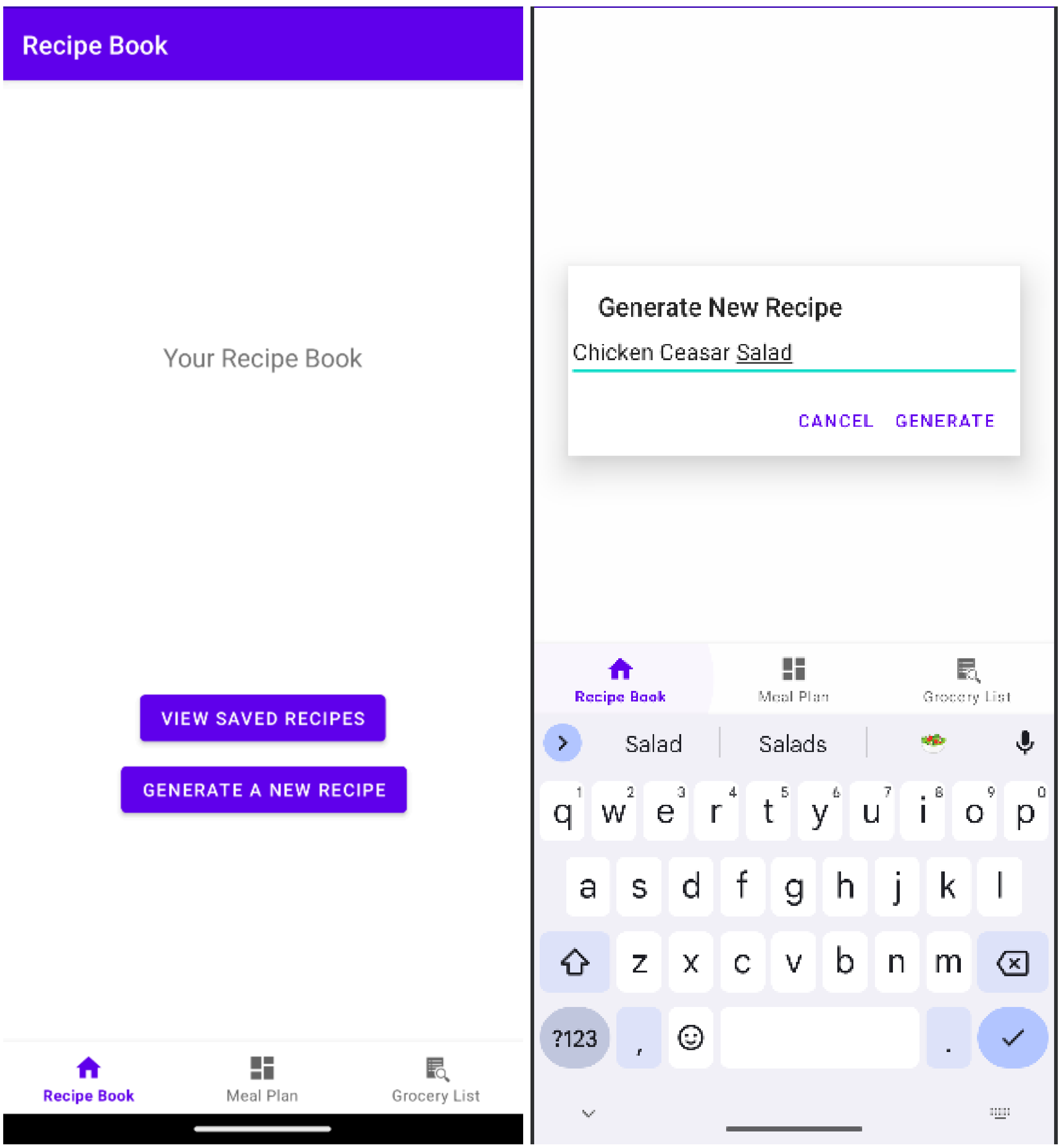
Additionally, adding the keyword "simple" to most prompts makes it so ChatGPT consistently outputs better and easier to read results. While these statements may seem few and inconsequential, they do a lot of work in making sure that the results that ChatGPT outputs are both indeed consistent and useful in making easy to execute meal plans for our user. There's a lot more potential for development and experimentation on making more optimal prompts that could take place if this app were to be developed further for full production, and we're confident that this application has the base it needs at this stage for that to be a possibility, further outside of the scope of this senior research project. Finally, let's take a look inside the app and see how it functions for the user.

IV. Implementation

The application's functionality has come a long way this past semester. All core components are implemented and are able to interact with each other, and while some more of the technical features are yet to be added, generally regarding user feedback and agency over what gets produced and put into their meal plans, the app's functionality based off of ChatGPT's outputs already proves to be a valuable tool. Evaluation of the efficacy of these outputs has been underway over the most recent few weeks, as will be described in the next section. The following figures show how the app's functions look and respond to the user.

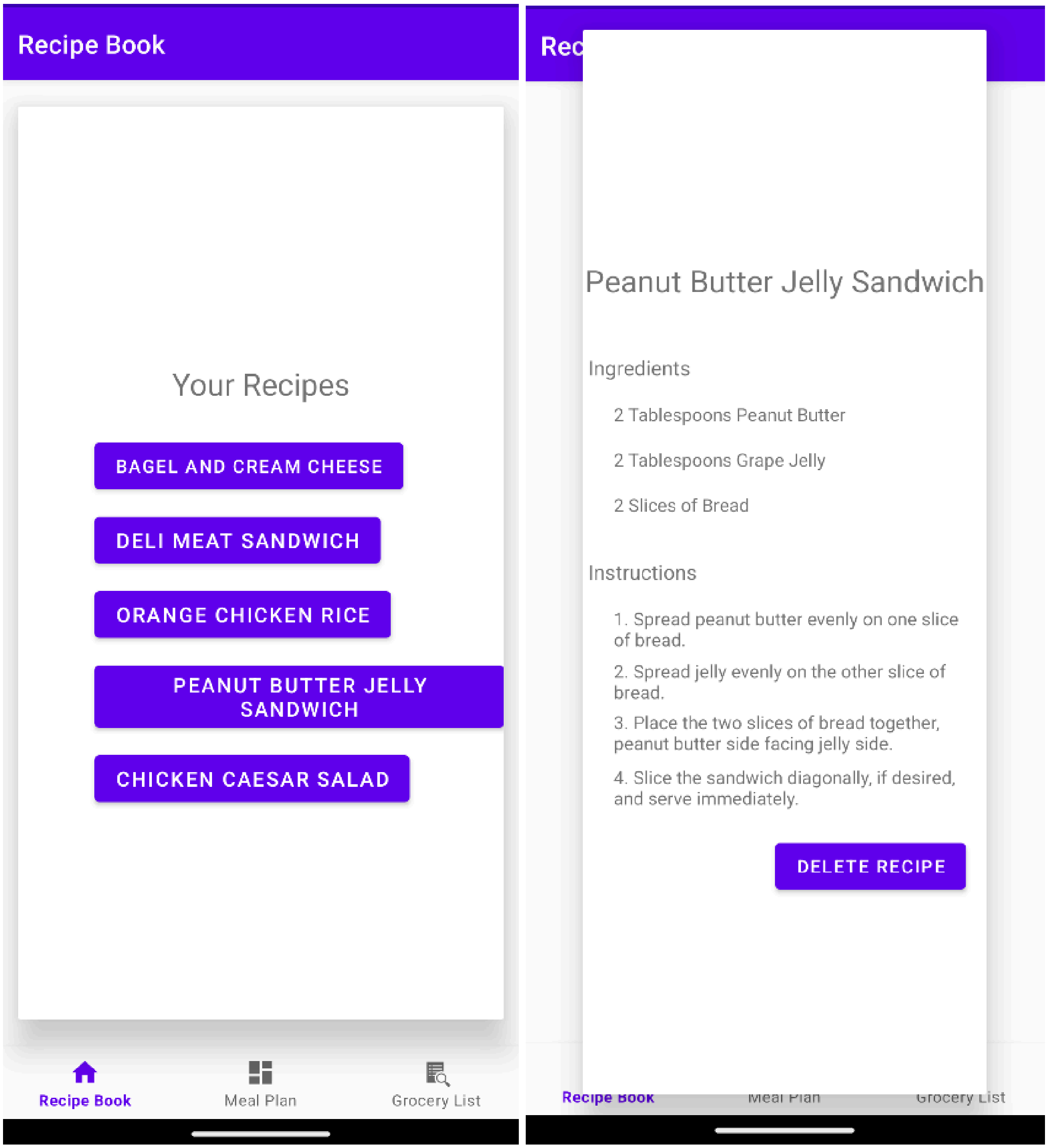
Figures 4-5.

A view of the landing screen a user views when opening the app, starting on the Recipe Book tab, and the User Input prompt for generating a new recipe by name to the Recipe Book using **(Generate a New Recipe)**.



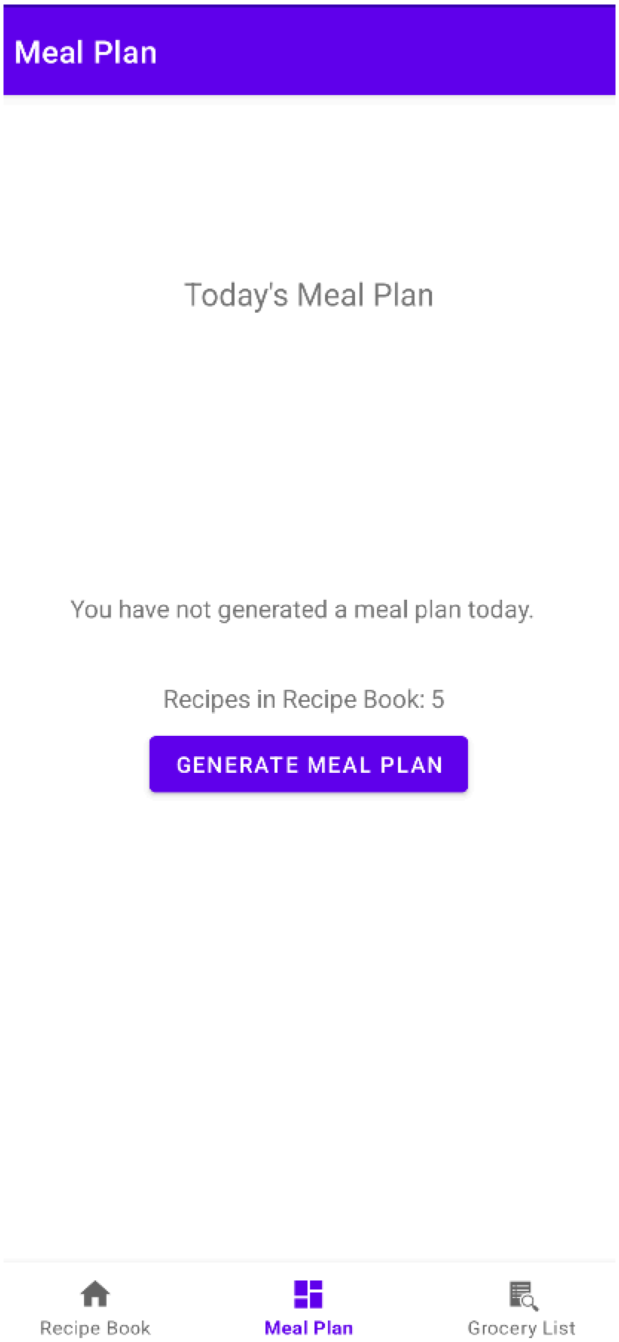
Figures 6-7.

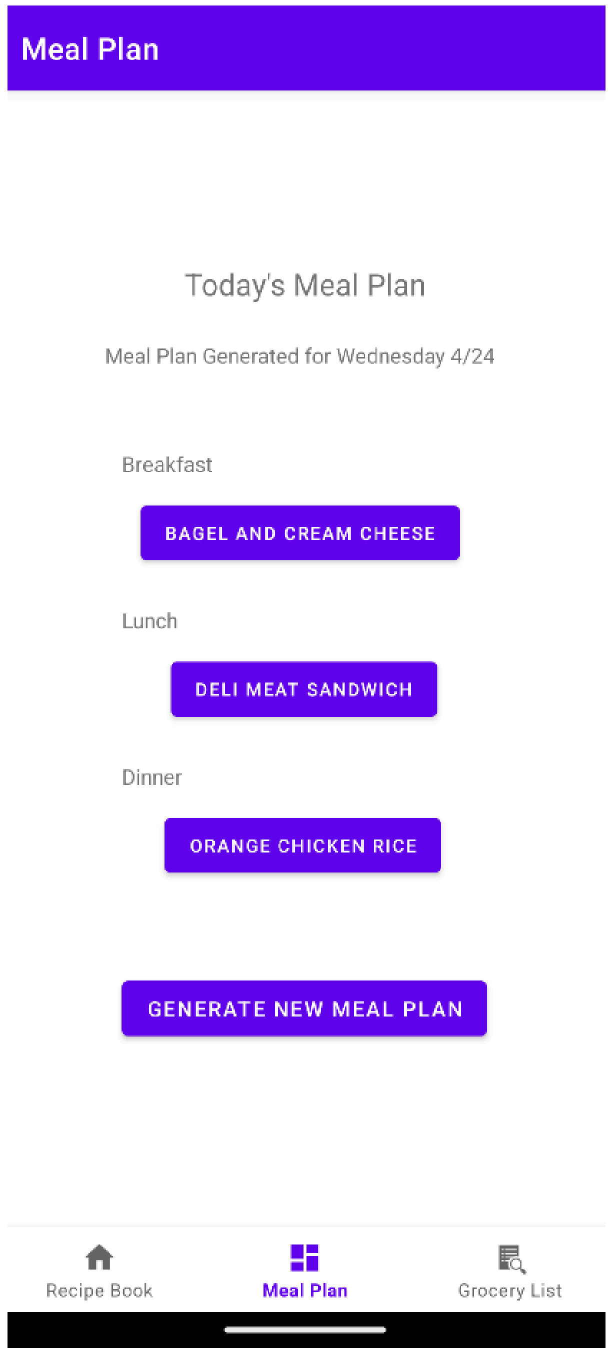
A view of the in-app Recipe Book browser using **(View Saved Recipes)**, and the contents of a single recipe using **(Recipe Name Button)**.



Figures 8-9.

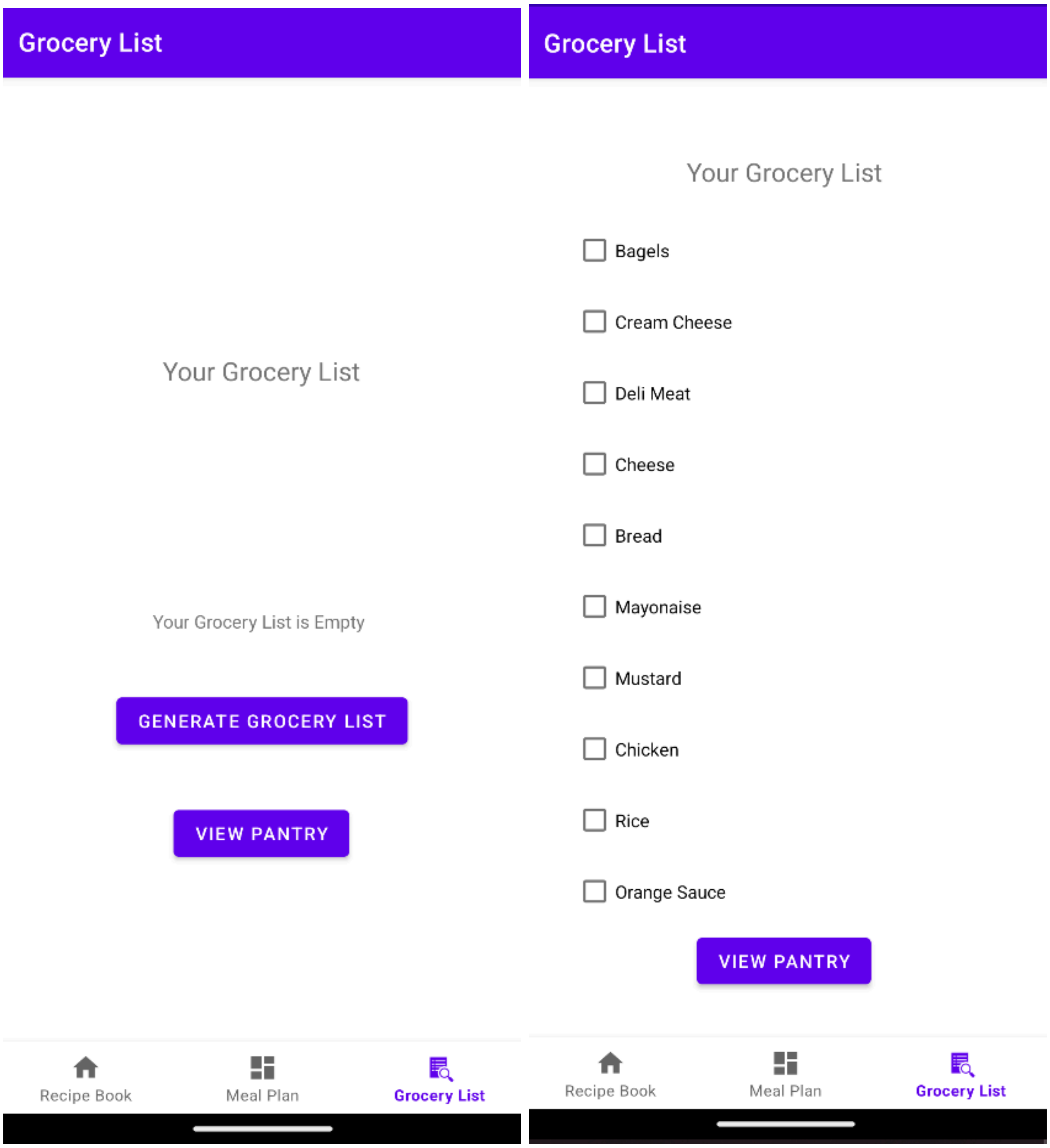
A view of the base screen under the Meal Plan tab with no plan generated, and with a plan generated for the day using **(Generate Meal Plan)**.





Figures 10-11.

A view of the Grocery List tab with no list generated, and with a list generated based on the Meal Plan in Fig. 9 using **(Generate Grocery List)**.



V. Evaluation

The evaluation process we've undertaken for this project has been less conventional than one might approach standard logic and loop-based algorithms. While there are definitely places in our app where time and space complexity could be measured and summarily optimized, what we're focused on in this implementation is the overall efficacy of the systems that are being driven by ChatGPT, that is the recipe generation and the meal planning steps. Of course, that begs the question: how exactly do we quantitatively measure how good the outputs of an AI are for our purposes? In the following section, we detail the processes and tests we've put our product through in order to evaluate how well it performs through multiple different lenses of analysis.

Evaluation #1: Quantitative Analysis of Cost Efficiency

First and foremost, we have an element of this app's utilization that involves number values we can directly compare and contrast with one another, and that is the total financial cost of trips to the grocery store between making meal plans manually and extemporaneously, and going off of the grocery list that is generated for us based on the meal plans made by the AI. So, over the course of two weeks, we tracked the costs associated with all of the meals we ate on our own schedule for one week, and compared it to a week's worth of meals generated by the application, using the same pool of meals prepared in the control period. The following figures show the results of this testing.

Figures 12-13.

MANUAL	(NC) = No Cost			
Day (4/10-4/16)	Breakfast	Lunch	Dinner	Other
Wednesday 10	Bagel/Cream Cheese	Meal Swipe (NC)	Orange Chicken Rice	Coffee
Thursday 11	Peanut Butter/Jelly Sandwich	Chicken Nuggets	Rice and Stew	Coffee
Friday 12	Bagel/Cream Cheese	Deli Sandwich	Cheeseburger (Out)	Coffee
Saturday 13	Bagel/Cream Cheese	Pulled Pork (Out)	Chicken Nuggets	Blueberry Shortcake
Sunday 14	Cereal	Frozen Pizza	Frozen Shrimp	Coffee
Monday 15	Bagel/Cream Cheese	Meal Swipe (NC)	Chicken Nuggets	Coffee
Tuesday 16	Peanut Butter/Jelly Sandwich	Salad	Meal Swipe (NC)	Coffee

Manual Groceries Bought	Cost
Bagels (6 Pack)	$1.95
Cream Cheese	$1.99
Bag of Rice	$2.65
Canned Beef Stew	$2.26
Peanut Butter	$1.99
Jelly	$2.55
Bread	$2.09
Deli Meat	$3.19
Deli Cheese	$1.89
Mayonaise	$3.29
Mustard	$0.95
Espresso Powder	$3.85
Brown Sugar	$2.15
Milk	$3.01
Creamer	$2.55
Frozen Chicken	$6.99
Buffalo Sauce	$2.98
Leaf Greens	$2.89
Caesar Dressing	$2.79
Croutons	$1.25
Cheerios	$4.72
Frozen Shrimp	$5.45
Frozen Pizza	$4.39
Blueberries	$5
Shortcakes	$4.41
Whipped Cream	$2.99
Orange Sauce	$2.19
TOTAL	$82.41

Figures 14-15.

W/ PLANNER APP				
	Breakfast	Lunch	Dinner	Other
Day 1	Cereal	Deli Sandwich	Orange Chicken Rice	
Day 2	Bagel with Cream Cheese	Peanut Butter Jelly Sandwich	Frozen Pizza	
Day 3	Cereal	Chicken Caesar Salad	Frozen Shrimp	
Day 4	Bagel with Cream Cheese	Deli Sandwich	Chicken Nuggets	
Day 5	Cereal	Peanut Butter Jelly Sandwich	Frozen Pizza	
Day 6	Bagel with Cream Cheese	Chicken Caesar Salad	Orange Chicken Rice	
Day 7	Cereal	Deli Sandwich	Frozen Shrimp	

Planner App Groceries Needed	Cost
Bagels (6 Pack)	$1.95
Cream Cheese	$1.99
Peanut Butter	$1.99
Jelly	$2.55
Bread	$2.09
Deli Meat	$3.19
Deli Cheese	$1.89
Mayonaise	$3.29
Mustard	$0.95
Frozen Chicken	$6.99
Leaf Greens	$2.89
Caesar Dressing	$2.79
Croutons	$1.25
Frozen Shrimp	$5.45
Frozen Pizza	$4.39
Orange Sauce	$2.19
TOTAL	$45.84

As we can clearly see, the total cost of groceries associated with the AI-generated meal plan ended up being much lower than the groceries purchased when shopping and planning manually. There are a few factors that go into this result. First, the AI only being able to select from a limited pool of recipes naturally precludes there from being any spur-of-the-moment meal decisions, like going out to eat or trying to make a new recipe for the sake of it without planning ahead using the app first. Second, the app also eliminates the "Other" category, whereas when eating manually there were additional grocery costs present considering things like drinking coffee every day, or making that blueberry shortcake as a dessert. Since the app only accounts for a rigid 3 meals-per-day schedule, extra consumption like this is not planned for and thus excludes the costs associated with that. However, looking at the generation, we can see that there are some definite efficiencies made clear in the schedule. The AI was able to effectively portion out which meals were eaten across different days, reducing repeated meals and making even more effective use of the groceries bought than we saw on the manual side. Ultimately, some tweaking would definitely need to be made to the prompts by which we generated our meals using ChatGPT to account for some of the natural spontaneity when it comes to human dietary habits, but the results speak for themselves regarding the cost effectiveness of using our tool to plan out the primary meals a user would eat in a day. This lens of evaluation was successful in demonstrating the great potential efficacy of our app in day-to-day utilization.

Evaluation #2: Comparative Analysis of Different AI Models

There are many different contemporary models of generative AI available to the public right now, so while our app's architecture is currently based solely on ChatGPT 4, we thought it would be insightful to compare the meal plans produced by our application with what might be produced by another generative AI using similar prompting methodology. The following figures demonstrate the results of this evaluation.

Test 1: ChatGPT4 AI App vs. ChatGPT 3.5 Turbo

First and foremost, let's compare the performance of the current state-of-the-art model we implement in our app with an advanced version of one of its previous iterations, to compare which branch of iteration provides more value to us. Our baseline will be the single-week meal plan produced shown in Fig. 14.

Figure 16.

3.5 TURBO	Breakfast	Lunch	Dinner
Day 1	Bagel with Cream Cheese	Deli Meat Sandwich	Orange Chicken Rice
Day 2	Cereal	Chicken Caesar Salad	Chicken Nuggets
Day 3	Peanut Butter Jelly Sandwich	Frozen Shrimp	Frozen Pizza
Day 4	Bagel with Cream Cheese	Deli Meat Sandwich	Chicken Caesar Salad
Day 5	Cereal	Chicken Nuggets	Orange Chicken Rice
Day 6	Peanut Butter Jelly Sandwich	Frozen Shrimp	Frozen Pizza
Day 7	Bagel with Cream Cheese	Chicken Caesar Salad	Deli Meat Sandwich

We can see that the two different models seem to have slightly differing "philosophies", per se, on how these meals are to be categorized between breakfast, lunch, and dinner. The output produced by 3.5 Turbo actually seems to be better at breaking up monotony in the meals that it generates, allowing for more unique combinations of dishes eaten throughout the day than the output produced by 4 in our app, which is very interesting. Perhaps more experimental users could be given the option to use this model if they have less stringent personal preference as to what kinds of dishes they eat for a given meal, and those who like to stick to more rigid mealtime categorizations can use the former.

Test 2: App vs. Google Gemini

Venturing outside the domain of OpenAI and its models, let's take a look and see what can be produced by various competing AI models on the market. To start, we'll prompt Gemini AI's chat in the same way as we did the others and see what it comes up with for us. For reference, since we're out of the realm of application logic, here is the prompt being input to Gemini (as well as was for ChatGPT 3.5 Turbo):

"Generate a week's meal plan, consisting of 3 meals a day based on the following list of recipes. You do not need to use all of the recipes listed. Bagel with Cream Cheese, Deli Meat

Sandwich, Orange Chicken Rice, Chicken Caesar Salad, Peanut Butter Jelly Sandwich, Chicken Nuggets, Cereal, Frozen Shrimp, Frozen Pizza"

Figure 17.

GEMINI	Breakfast	Lunch	Dinner
Day 1	Cereal	Orange Chicken Rice	Chicken Caesar Salad
Day 2	Bagel with Cream Cheese	Leftover Orange Chicken Rice	Frozen Pizza
Day 3	Peanut Butter Jelly Sandwich	Chicken Caesar Salad	Frozen Shrimp with Rice
Day 4	Cereal	Deli Meat Sandwich	Chicken Nuggets
Day 5	Bagel with Cream Cheese	Leftover Chicken Nuggets	Chicken Noodle Soup
Day 6	Peanut Butter Jelly Sandwich	Deli Meat Sandwich	Free Night (Pick Your Own Meal!)
Day 7	Leftovers or Cereal	Leftovers or Deli Meat Sandwich	Breakfast for Dinner!

We've got a lot of interesting results to unpack with this one. Overall, Gemini's output seems to be a lot more creative with how it uses the list of meals that you input to it. For example, dinner on Days 3 and 5 specify meals that weren't listed in the prompt, instead combining known components of some of the recipes listed to create a new dish entirely (its specific suggestion for Chicken Noodle Soup was to "use leftover chicken from Caesar Salad, add vegetables and noodles"). Additionally, we have the consideration of leftover servings baked into the meal plan, which if implemented could definitely be leveraged to further increase the efficiency of our usage of ingredients, thereby decreasing the associated costs of groceries. We also have the suggestion on Day 6 to have a night where the user decides their own meal for a change, designating it as a "night off from cooking". While this definitely seems to run contrary to the philosophy behind the application we're developing, this kind of result may not be an entirely unwelcome change of pace depending on the user. Finally, we get a plan that completely ignores the recipe list and just suggests having breakfast for dinner (late night breakfast, anyone?), specifically recommending pancakes, waffles, or eggs as the dish for that night. All in all, Gemini's response to our test leaves some very interesting implications, but the unpredictability of it makes it seem like it would be less suited than ChatGPT to be implemented into our app. Still, with some clever prompt engineering, we can definitely see potential in utilizing this model and leveraging the element of creativity that it could bring to our meal plans.

Test 3: App vs. HuggingChat

As most other contemporary mainstream AI Chat models are already based off of OpenAI's GPT models (Like Microsoft Copilot or Jasper), to round off this round of evaluation, we used a unique chatbot called HuggingChat, based on the proprietary open-source model HuggingFace. Since there are definite benefits that could be leveraged from hypothetically having full source access to the model being utilized for our app, let's see what this model has to offer as far as its generative capabilities go.

Figure 18.

HUGCHAT	Breakfast	Lunch	Dinner
Day 1	Bagel with Cream Cheese	Chicken Caesar Salad	Orange Chicken with Rice
Day 2	Cereal	Peanut Butter and Jelly Sandwich	Chicken Nuggets with Frozen Shrimp
Day 3	Bagel with Cream Cheese	Deli Meat Sandwich	Frozen Pizza
Day 4	Cereal	Chicken Caesar Salad	Orange Chicken with Rice
Day 5	Peanut Butter and Jelly Sandwich	Deli Meat Sandwich	Chicken Nuggets with Frozen Shrimp
Day 6	Bagel with Cream Cheese	Chicken Caesar Salad	Frozen Pizza
Day 7	Brunch Option: Deli Meat Sandwich	Leftovers or Snacks	Dinner Out: Cereal for Late Night Option

This response seems to strike a happy medium between the rigidness of ChatGPT 4 and the creativity of Google Gemini. Interestingly, this model liked to combine 2 of the recipes listed, namely Chicken Nuggets and Frozen Shrimp, into one meal, which would make for a more hearty dish than just having one or the other, while conversely using more ingredients to make that so. This model also accounts for leftovers, but only for 1 meal slot. The suggestions of Brunch and Dinner Out on day 7 are also unique to this model's response, and seem a lot more effective breaks from the monotony of the limited recipe list than were generated by Gemini, but eating dinner out will also drive up the overall cost of food for the week, although it's never a bad thing to treat yourself to once in a while. Overall, this model would probably not be as effective in saving on food costs for the week as ChatGPT, but seems to provide a much more human-oriented experience in eating meals for the week. Plus, with the added utility of this model being open-source, the possibilities are really endless with this one. We could create and fine tune a model made specifically for the purpose of planning meals like we're doing here, so for a long-term course of development, this kind of model would definitely be an appealing path to go down.

With all those evaluations done, we come to multiple substantive conclusions about our app in its current implementation. While some fine-tuning could definitely be warranted to make our app's meal planning responses more robust and considerate of the fickleness of human dietary habits, we believe that the utilization of this app has great potential to provide benefits to its users, in terms of lower costs and less cognitive burden when it comes to planning out what meals they can eat over a given period of time. Additionally, our implementation of the ChatGPT 4 model was ultimately the right decision to make, as while the other models demonstrated more capacity to have some leeway for human input and decision-making when it comes to what they eat, for the purposes of making an app that rigidly and consistently generates meal plans based on a finite pool of recipes, ChatGPT provides us the most utility.

VI. Potential Implementations

There are many possible features that could be implemented to help greatly with the productivity and usability of our app. For example, small utilities like a measurement conversion chart detailing comparisons of different measurements would be helpful in scaling recipes up or down for more or fewer servings. Having ChatGPT be able to generate information regarding how to best store ingredients or leftovers would help increase the efficiency with which the user is able to use their ingredients, saving costs and reducing the need to go to the grocery store to replace them if they were to go bad. Leftovers of a given meal slot could also be indicated after the user is finished eating, and can be factored in planning near future meals to decrease how much the user has to prepare a new recipe for each meal.

To ensure that new recipes are consistently being cycled into the Recipe Book for diversification purposes, we could have ChatGPT intelligently recommend a few new recipes during each planning cycle, based on what the user has indicated they like the best out of the recipes they've had so far. In a similar vein, broad categories of food that can take on different forms, like salads or pastas, can be recommended to a user based on the core components of those categories that they already have. For instance, if the user consistently eats Caesar Salad as a meal of choice, then it would be easier to incorporate a different kind of salad in their Meal Plans, like Balsamic Vinaigrette or Ranch flavor, since they already have the base components of lettuce and croutons. Intelligently reusing ingredients in this manner will help the user save on resources, and get through more meals while using fewer unique ingredients.

Another very important factor that has yet to be considered in our current scope is the availability of kitchen supplies and utensils to the user. There are a wide variety of tools and appliances needed in the kitchen in order to make certain dishes, like mixers, toasters, and air fryers, that can be a prohibitive step to users with less expansive kitchen experience. Even basic appliances like ovens, stoves, pots, and pans can be inaccessible to some users, so if this were to be distributed to a wider audience, accessibility of these resources would have to be something considered to make our app usable to many people who would need it.

The last potential category of development that could potentially be implemented to improve the user experience is in the form of letting users have more degrees of control over what meals are put into their Recipe Books and Meal Plans. For example, the app could generate a recipe for a meal that the user likes, but use different ingredients or methods that the user isn't privy to. For this purpose, an interface that allows the user to modify the data of a recipe JSON file could be implemented to give recipes a higher degree of customization and usability for the user. Another mechanic of the Recipe Book that could help further tailor the Meal Plans to each specific user could be a ranking or favorability system, that allows the user to weigh each recipe in their Recipe Book depending on how much the user likes that meal, which would then affect how frequently that meal is put into their Meal Plans. If the user really enjoys having Bagels and Cream Cheese as a staple breakfast, they can give that recipe a high favorability so that the app knows to schedule it more frequently for them. While one of our primary design philosophies is centered around reducing the amount that the user has to eat the same meals in succession, it's also important for the user to be able to exercise and implement their own personal preferences when it comes to how their meals are planned out.

VII. Conclusion

There are a great many ways in which we could improve and build on what we've come up with over the course of this app's development. Many ideas we had planned to give more agency and personalization capabilities to the user had to go by the wayside in order to focus in on making the core mechanics of this application the best that they can be, and we're happy with what we've ultimately come up with. Exploring the potential considerations to the quandary of generating meal plans that other AI models were able to demonstrate to us has additionally left us with a lot of ideas on how our results could be better improved to cater to the individual user in further potential development. But, when all is said and done, we believe that we've created a powerful tool that cleverly leverages the full power of Generative AI in order to make a product that can help the end user in many ways, especially financially.

We've learned a great deal about the potential implementations of generative AI over the course of this senior research, and we're more excited than ever to continue exploring into the new frontiers of what this and other emerging technologies have to offer to us as developers and to humanity as a whole. Menial, repetitive tasks that continually produce the same outcome and take up our time and mental processing power, like meal planning for our purposes, can be easily relegated to AI to produce those same outcomes with a similar degree of precision and infinitely higher degree of time and resource efficiency. Of course, meal planning is a relatively low-stakes application of this technology, and in many fields where it could be applied like healthcare and governmental affairs, the models will need to be held to a much higher level of scrutiny in order to assure their accuracy and efficacy. But, as these models continue to be rapidly developed and enhanced, it's easy to conclude that AI will only continue to be able to be applied to more and more complicated and important problems, that will ultimately increase the productivity and efficiency of our society as a whole.

Through the power of generative Large Language Models like ChatGPT, we can bypass a lot of the logical and memory limitations that would normally be imposed on the production of an application of this scale. And, in the grand scheme of things, this is just one of the many ways that AI can be used to make humans' lives easier. As our technology stands now and probably will for a long while, no computer or AI could hold a candle to the sheer complex and creative power of the human brain. However, that's all the more reason we should continue finding ways to implement it to make our lives and jobs easier, so that we can apply our full potential brain power to tackling even greater problems facing our species, and work together towards making this world a better place to live in for all of humanity.

Articles, Tools, and AI Models Referenced

"22 Common Dietary Restrictions to Look Out For." *American Dining Creations*, 21 Jan. 2022, adc-us.com/blog/common-dietary-restriciions-to-look-out-for/.

Bengio, Yoshua, et al. "A Neural Probabilistic Language Model." *The Journal of Machine Learning Research*, 1 Mar. 2003, dl.acm.org/doi/10.5555/944919.944966.

Ekin, Sabit. "Prompt Engineering for CHATGPT: A Quick Guide to Techniques, Tips, and Best Practices." *ResearchGate*, Apr. 2023, www.researchgate.net/publication/370554061_Prompt_Engineering_For_ChatGPT_A_Quick_Guide_To_Techniques_Tips_And_Best_Practices.

Roumeliotis, Konstantinos I., and Nikolaos D. Tselikas. "Chatgpt and Open-AI Models: A Preliminary Review." *MDPI*, Multidisciplinary Digital Publishing Institute, 26 May 2023, www.mdpi.com/1999-5903/15/6/192.

White, Jules, et al. "A Prompt Pattern Catalog to Enhance Prompt Engineering with Chatgpt." *arXiv.Org*, 21 Feb. 2023, arxiv.org/abs/2302.11382.

Zhong, Qihuang, et al. "Can CHATGPT Understand Too? A Comparative Study on CHATGPT and Fine-Tuned Bert." *arXiv.Org*, 2 Mar. 2023, arxiv.org/abs/2302.10198.

https://www.foodai.app/

https://www.dishgen.com/

https://www.chefgpt.xyz/

https://chat.openai.com/

https://gemini.google.com/app

https://huggingface.co/chat/