Senior Capstone Project Proposal

Project Name

Household Menu Planner and Ingredient Tracker (HoMePIT)

Team Members

- Erik Anderson
- Jerome Chestnut
- Maxwell Fugette

Abstract

Menu planning is an inescapable and oft time-consuming part of human existence. Some individuals attempt to avoid it by subsisting on fast and highly processed food, while others try their best to manage it using whatever tools are available to organize and categorize this part of their lives. With healthy eating being the preferred option, it is imperative for it to be made as simple and convenient as possible. We believe that many of the tasks associated with meal planning, such as the tracking of household ingredients and foodstuffs, planning what meals to eat on what days, and the composition of shopping lists are often too tedious and time-consuming for the average person, which will inevitably lead to unhealthy dietary choices. All of this can be simplified with the appropriate application of technology. HoMePIT is our attempt at making this simplification.

HoMePIT, as its expansion suggests, will assist primarily in the tasks associated with planning a household's menu and tracking its supply, usage, and replacement of ingredients and other foodstuffs. It will be designed to streamline the culinary experience of a household by making ingredient inventorying, grocery shopping, meal planning, and nutrition tracking easier to manage on an everyday basis. Users will be able to add ingredients, recipes, information relating to ingredients and recipes, and more to HoMePIT. Simultaneously, HoMePIT will assist users in tracking food expiration dates, daily menus, and nutrition, allow users to more easily compose daily menus based on previously input recipes and ingredients on hand, and produce shopping lists covering user-defined time periods. With this, HoMePIT is intended to not just be an application, but rather as essential to a kitchen as the food itself.

Description

Introduction

For many, kitchen related tasks are not only tedious, but outright inconvenient. In today's world many find themselves not having time or energy to devote to a consistent meal schedule, often opting for quick and convenient options such as fast food in order to not go hungry. As such, it is no wonder that health is on the decline. The fast-paced nature of the modern world has low compatibility with the maintenance of healthy dietary habits, which often demand sustained and thought-out effort on the part of the consumer. Effort in this sense may include preparing a shopping list, planning meals and their associated recipes for a given week, or even having to run calculations to ensure there are enough ingredients to go prepare all the meals that one desires in a week. The simple fact is that the time needed to engage in these tasks is scarce, with this being further compounded when there are multiple mouths to feed and certain dietary restrictions that one must consider. There have been attempts in the past to make these processes easier, such as in the case of MyFitnessPal, which has become a household name. Even a popular application such as this has its limitations, however; MyFitnessPal does not generate shopping lists, does not keep track of food items already in possession, and does not create a calendar for meal planning. When such limitations in existing services are so easily identified, one ends up desiring a service that can satisfy these unmet needs. This desire is amplified when considering the aforementioned demands of the modern world. From these considerations came the idea for the Household Menu Planner and Ingredient Tracker, also known as HoMePIT.

HoMePIT will be a web-based application designed to assist in a variety of kitchen related tasks. Unlike other applications of this nature, HoMePIT will allow for recipe saving, calorie and macronutrient tracking, and traditional meal planning. Simultaneously, HoMePIT will provide auxiliary services, such as the ability to create shopping lists that will automatically account for one's planned meals over a specified time period, track the quantity and use of leftovers, be alerted when leftovers or ingredients are preparing to spoil, view recipes that will use food that is preparing to spoil, and more. As previously stated, the demands that the modern world places on the average consumer means that they often have little time left in the day to properly pay attention to the kitchen, whether that be with respect to cooking or preparations made for cooking. If home cooking were to be made as convenient and engaging as picking up food from a restaurant or fast food chain, society would certainly have the opportunity to become healthier and happier. Creating nutritious, healthy meals in a way that saves time, maximizes satisfaction, and minimizes stress is an often underlooked aspect in other meal-prep applications that HoMePIT intends to solve.

Key Features

HoMePIT plans to be a comprehensive solution designed to revolutionize the way households approach meal planning, ingredient management, and culinary enjoyment. Within its digital ecosystem, users will find an array of essential features that will streamline nearly every facet of their kitchen logistics experience, which include but are not limited to ingredient management, recipe creation, effortless meal planning, and smart shopping lists. Unfortunately, nothing can be done to help with the clean up, but the time savings generated elsewhere will ensure that this is no problem.

Ingredient management will allow users to define and track ingredients, specify volumes or masses, nutritional information, purchase locations, costs, and user-defined tags that allow users to extend the power of HoMePIT. Furthermore, users may also define ingredients as substitutes for one another, potentially saving them a grocery trip in a pinch. HoMePIT keeps users informed about ingredient freshness and predicts when stocks might run low, ensuring kitchen efficiency. These features exist to assist the user in the logistics of keeping track of what is in their kitchen while minimizing waste, particularly when combined with the recipe and meal planning features.

Creating recipes will be quite simple with HoMePIT. Users will be able to select an icon to “add a recipe” to their virtual cookbook, wherein they may define ingredients, cook times, and user-defined tags. Sub-recipes allow for complex culinary creations, and the system automatically derives nutritional information and cost based on ingredient specifications. Furthermore, when sub-recipes, cook times, and meal planning functionality come together, the user is given the ability to specify when they want to cook sub-recipes that they are missing, should they want to spread their cooking endeavors over multiple days.

Meal planning will be made effortless with HoMePIT due to its introduction of a dynamic meal planning calendar, enabling users to schedule future meals effortlessly. Users can predict leftovers, divide meals for meal prepping, and receive alerts when leftovers are about to expire. Custom sorting options make meal selection a breeze, allowing users to select meals based on their own user-created tags, caloric or nutritional content, expiration date of leftovers or ingredients, and more.

HoMePIT intends to make unplanned and disorganized grocery trips a thing of the past by generating smart shopping lists that align with meal plans and ingredient thresholds. Users can customize shopping frequencies, confirm purchased items, and note substitutions or unavailable items. Users may also request that shopping lists be generated at user-defined intervals, meaning that those with a regular grocer-visiting schedule won’t have to worry about specifically requesting their shopping list before they leave.

These features are at the heart of HoMePIT, designed to simplify the daily culinary tasks that households face. HoMePIT is intended to be more than just a tool; it aspires to become an essential companion in the modern kitchen, enhancing the culinary experience and making it not only easy but truly enjoyable. With the potential for future enhancements, such as automated menu generation, push notifications, route optimization for grocery shopping, and barcode scanning, HoMePIT sets the stage for a culinary journey like no other.

We will leverage a range of technologies to ensure HoMePIT's seamless operation. The platform will be accessible through popular web browsers like Firefox and Chrome, allowing users the flexibility to access from various devices. It will be compatible with the Windows 10, Windows 11, and Android operating systems, ensuring a wide user base can benefit from its features. Being a browser-based application, the intent is that HoMePIT will be accessible anywhere, anytime and not faced with the limitations on portability that phone or desktop applications have.

Our development environment will primarily rely on tools such as Visual Studio and Visual Studio Code (VSCode), through which the application itself will be built using a combination of programming languages such as NoSQL on the database end, and ASP.NET, an open-source web framework developed by Microsoft that extends the .NET framework to assist in building web applications on the frontend. Given that it is based on the .NET framework, ASP.NET uses the C#, F#, and Visual Basic programming languages, which will enable the team to draw off of their shared experience with such technologies. The actual look and feel of the frontend will be created using Hypertext Markup Language (HTML) as well as Cascading Style Sheets (CSS), integrated with ASP.NET. The end result will be a sleek, functional website that meets the demands of what is asked of it.

For server management, data storage, and website hosting the team will utilize Google Firebase, a reliable and scalable solution that aligns with the goals of the project and offers much in the way of possible extensions, including but not limited to authentication, messaging, image resizing, and much more. Finally, communication within the development team will be facilitated through Discord and/or Short Message Service (SMS), ensuring efficient collaboration throughout the project's lifecycle.

Feature List

Guaranteed Features

- Ingredient-related features
    - The user may define ingredients and other foodstuffs, either specifically or when defining a recipe. When doing so, they may specify a given volume or mass of the added ingredient or foodstuff.
    - When defining ingredients, the user may add additional details, such as other ingredients that it may substitute or be substituted by, nutritional information, purchase locations, cost, and user-defined tags.
    - Ingredients may be assigned use-by dates or user-approximated expiration times, to allow for the user to keep better track of when ingredients are about to be wasted.
    - An ingredient may have a threshold volume or mass defined. If an ingredient is predicted to fall below its threshold but not run out due to recipe creation, then the user will be alerted to this fact.
- Recipe-related features
    - The user may define recipes, which contain a variety of ingredients of varying volumes and masses. Recipes may contain sub-recipes, which are portions of a recipe that are cooked or baked separately from the rest of the recipe and incorporated later. Recipes may also have cook times specified, with sub-recipe cook times added to the using recipe's cook time.
    - Recipes may have user-defined tags added to them, to allow for easier sorting and organization via other features.
    - Recipes will determine nutritional information, cost, and user-defined tags based off of the ingredients specified and their quantities.
    - Users may specify the number of portions a certain recipe cooks, as well as a default number of leftovers it provides.
- Meal planning features
    - Users will have access to a calendar that they may use to specify what meals they plan to cook and/or eat in the future. When specifying meals to be eaten, the user may select whether they intend to cook the meal from scratch using a recipe or consume leftovers (if they have any).
    - When a user has specified a meal to be cooked and eaten, they may choose to approximate the number of leftovers they predict they'll get from that meal. If they choose not to, they may specify the number of leftovers yielded, as well as leftover portion size if desired.
    - A user may also choose to specify that a meal will be cooked and then divided into "leftovers" rather than eaten, to permit users to account for meal prepping.
    - When leftovers are created, the user may predict how long the leftovers will remain fresh, such that HoMePIT may alert the user when leftovers are preparing to expire.
    - When cooking a meal from a recipe containing one or more subrecipes, the number of sub-recipe servings remaining will be displayed alongside any, if any, leftovers.

    - A user may choose to cook a meal and eat its leftovers in the same week. If they indicate that they want to eat more leftovers than a given meal preparation will produce, the user will be warned of this and allowed to either increase the size of meal being prepared or remove excess leftover-consuming meals.
    - When selecting a meal to eat, a user may sort recipes and leftovers by when they were most or least recently eaten, as well as by time until expiration. They may also choose to include or exclude recipes based on nutritional information or user-defined tags.
    - A user may access summaries of caloric information, nutrient information, and more on a weekly, monthly, or other user-specified timeframe basis.
- Shopping list features
    - The user will be able to request that a shopping list be created that enables the creation of meals for a specified time period. This shopping list will also contain ingredients that are set to fall below their threshold.
    - The user may also specify their frequency of grocery trips and be provided shopping lists for the appropriate time periods.
    - After a user has made their grocery trip, they may confirm that they obtained the ingredients listed, and specify any that were either substituted or unable to be obtained.

Nice-to-Have Features

- Ingredient-related features
    - One or more photos may be assigned to a given ingredient, to permit easier recognition of an ingredient in future shopping endeavors should a user have forgotten the ingredient’s appearance.
    - HoMePIT will alert the user in the webpage as to when ingredients are about to expire.
- Recipe-related features
    - One or more photos may be assigned to a given recipe, to permit a user to more easily remember what a given recipe looks like when prepared, should they have forgotten this.
- Meal planning features
    - When selecting a meal for a given day, the user may organize potential meals and recipes by how soon the included ingredients will expire.
    - Users will be informed of the latest time they can start preparing a given recipe, with this time calculation being based on the defined preparation time for a recipe and its sub-recipes, whether enough portions of sub-recipes exist to satisfy the recipe preparation in question, and the number of portions of recipe being prepared. If a certain sub-recipe requires significant time to prepare, then it may be separated from the main recipe and placed on the meal planning calendar as its own recipe to prepare.
    - HoMePIT will be able to analyze the rate at which a user consumes ingredients , leftovers, and sub-recipes, enabling it to inform a user of when they will likely run out of a given ingredient, leftover, or sub-recipe.

    - HoMePIT will alert the user in the webpage as to when leftovers and subrecipes are about to expire.
    - A user will be able to access a set of charts, graphs, and other visualizations of their food-consumption related statistics across predefined or user-defined time periods.
- Shopping list features
    - The user may opt to have their future shopping lists be updated to account for predicted usage rates of their ingredients and whether any are predicted to either run out or fall below a user-specified threshold.
    - The user may opt to ignore an ingredient that needs to be restocked, or an ingredient that has passed its threshold. A user may also specify that an ingredient ought to be allowed to run out, either temporarily or permanently.
    - When a shopping list is to be created, a user may choose to have images of ingredients linked or embedded within the shopping list.
    - When a shopping list is to be created, HoMePIT will attempt to minimize the number of user-specified grocers that need to be visited.

Future Features

- Ingredient-related features
    - Ingredients can be input and automatically populated with information by either submitting a Universal Product Code (UPC) or an image of a barcode from the ingredient in question.
- Recipe-related features
    - Recipes can be input by submitting a website Uniform Resource Locator (URL) containing a recipe or a file containing a recipe.
- Meal planning features
    - A user may request that HoMePIT generate a meal plan schedule based on constraints provided by the user regarding the ratio of different user-provided tags (such as those associated with cuisines), caloric content, nutritional content.
- Shopping list features
    - Shopping lists will be paired with a route provided by a third-party routing API that attempts to optimize a user's grocery shopping against time, distance, or other factors.
- Miscellaneous features
    - HoMePIT will use push notifications via email or SMS to remind a user outside of the webpage that certain ingredients or leftovers are close to expiring.
    - A user may choose to export or sync their planned menus to a Google, Outlook, or other third party calendar.
    - A user may authorize other users to consult with their meal calendar and permit them to make changes or suggestions, such as in the case of a household wanting to allow every member of the household to contribute to meal planning. Users may make comments regarding the menu for any given day. The original user may set constraints or dictate permissions for other users, such as limiting them to choosing a meal one day per month.

Technology

- Platform: Firefox Browser, Chrome Browser
- Operating System (OS): Windows 10/11, Android
- Integrated Development Environment (IDE): Visual Studio, VSCode
- Programming Languages: C#, Typescript, Javascript, HTML, CSS, others if necessary
- 3rd Party Tools/Libraries: ASP.NET, GitHub
- Server Software: Google Firebase
- Communication Software: Discord, SMS

Server Information

Google Firebase will be used as the server component for this project, due to the utility it provides for hosting web applications. Additionally, it is free-to-use for small-scale applications. The main selling point of Firebase is that it bypasses the need to manage servers when building the backend, allowing the team to focus more on the application itself and less on peripheral components. Within the scope of a senior capstone project, this is desirable as it will provide the group more time to implement wanted features and polish existing ones. In addition to the database aspect, the base plan for Firebase, Spark, allows for the hosting of a secure, quick-loading website that is backed by a global Content Delivery Network (CDN). Lastly, with Firebase user authentication, sign in, and onboarding is made easy, thus fulfilling multiple nice-to-have features of this project.

Data Sources

In the event that we have the opportunity to implement the feature of using an ingredient's UPC or barcode to automatically import its nutritional information, we will require a database of food-related UPCs and their associated product information. The United States Department of Agriculture (USDA) maintains such a database called FoodData Central (FDC), and this may be accessed using the FDC Application Programming Interface (API), which provides Representational State Transfer (REST) access to the database. Access to the API requires the usage of an API key, which can be freely signed up for. By default, this API key allows for 1000 requests per hour per Internet Protocol (IP) address, with the key being suspended for an hour if that rate is exceeded. A modified rate limit can be requested from FoodData Central if necessary. The DFC states that USDA food composition data is public domain and thus free from copyright, and simply requests that app creators credit FDC if the DFC API is used in an application.

Team Member Backgrounds and Responsibilities

Erik Anderson’s background involves the usage of C#, Java, MATLAB, SQL, R, and Python in the Windows 10 and 11 OSs. Erik is generally comfortable with C#, Java, and MATLAB, as he has used each for multiple years at this point in his academic career. He has passing familiarity with SQL, R, and Python, having used each during the course of a single college course each. He does not have a background in Typescript, Javascript, HTML, CSS, ASP.NET, or Google Firebase. Erik is familiar with the usage of the web platforms of Mozilla Firefox and Google Chrome, although he has never developed web applications for those platforms. Likewise, Erik is familiar with the usage of the Windows 10, Windows 11, and Android OSs, and he has developed code that has run on Windows 10 and 11. Erik has used the Visual Studio IDE for over a year, although he has not used the VSCode IDE. Erik has never developed an application within the ASP.NET framework, but he has become familiar with the usage of GitHub for version control. Erik has never used Google Firebase. Finally, Erik has used Discord as a communication platform for 7 years, and is highly confident in his usage of it. Erik’s responsibilities will mainly be associated with the creation and organization of databases, as well as the creation of code in C# within the ASP.NET framework relating to the functionality of HoMePIT.

Jerome Chestnut has a strong background in C# and Java, which he has used extensively throughout his academic and professional career. He is also proficient in Extensible Application Markup Language (XAML) and HTML, making him well-versed in both desktop and web development. Jerome has some experience with SQL, having utilized it during a college course, showcasing his ability to work with databases. Additionally, he has experience with Visual Studio and VSCode, further enhancing his development capabilities. In addition to his core programming skills, Jerome has a solid foundation in both frontend and backend development. He is comfortable working with XAML for Windows applications and HTML for web development. Jerome's experience in these areas makes him a versatile developer capable of contributing to various aspects of projects. He is also familiar with version control using tools like GitHub, ensuring effective collaboration within the team. Jerome's experience and skills make him a valuable asset to the team, particularly in areas related to C# and Java development, as well as frontend and backend tasks.

Maxwell Fugette primarily has a background in the programming languages of C#, Java, and Typescript, along with some knowledge of Rust, Haskell, and C. In addition, Maxwell has some experience using third-party applications such as Dreamweaver for User Experience (UX) design and Google Firebase to develop and host frontend applications, as well as having had substantial time put into WPF (Windows Presentation Foundation) and other User Interface (UI) type applications. Maxwell has used both the Visual Studio and VSCode IDEs for roughly half a decade, and is proficient in both. Likewise, Maxwell has used Google Chrome for several years, and through the use of Angular has developed simple web applications on it. Despite not being familiar with the ASP.NET framework, Maxwell is familiar with the .NET framework given his past experience, and as such is confident in his ability to learn new technology when necessary. Discord and GitHub for communication and version control are also familiar to him. Maxwell will focus primarily on the frontend of HoMePIT while also contributing to the project as a whole wherever necessary, one such area being the integration with Firebase and all features associated therein.

Dependencies, Limitations, and Risks

Dependencies

- Before much of the actual project starts production, the Firebase project associated with it must first be set up, as HoMePIT is a web application. Its functionalities require the usage of databases to manage user data, and this functionality will be provided via Firebase. This endangers the project due to the fact that if the web backend of HoMePIT cannot be set up in a timely manner, then it will be difficult to develop all other parts of the application, as the implementation specifics of everything web-based will be unknown. This will, in turn, delay the beginning of HoMePIT's development, relegating group members to searching for a new web backend and developing conceptual elements of HoMePIT. If difficulties are had in establishing the Firebase component of HoMePIT, then other services providing databases and user authentication may be considered, with a priority placed on ease of adoption.
- All group members must ensure that the necessary technologies are installed and set up on their machines. If a group member fails to have the necessary technologies available, then they will have difficulty assisting in any development using that technology should another group member require said assistance. If a team member is found to not have the necessary technologies installed, then the other team members will arrange a meeting with the goal of assisting the team member in question with the installation and setup of the required technologies, thus ensuring technological parity within the team.
- A Git repository must be set up prior to any other work being done to ensure proper version control, and team members must use it competently and regularly to ensure that they remain current in their development of HoMePIT. If this does not occur, then team members may run the risk of losing progress that they have made in developing HoMePIT, which could endanger the development timeline and lower group morale. Should difficulties arise in the use of Git or the Git-based service Github, then it may be necessary to use a different Git-based service to conduct version control. Likewise, if any team member or the team as a whole faces difficulty in the proper usage of Git and Github, it may be necessary for the team to dedicate time towards meeting and improving their usage of these services.
- In the event of licenses running out, being revoked, or changing, then the team will have to determine alternative services that provide the same or similar functionalities with more permissive licensing. This is because the loss of proper licenses would endanger the development of HoMePIT, as further development would break licensing and could incur legal troubles. Should such services not exist, certain features may need to have their method of implementation changed, or features may need to be replaced with combinations of alternative features that provide the same overall functionality.
- HoMePIT will be a web-based application, meaning that its operation necessitates a server that runs it. Given that it will not be allowed to be hosted on homebrew servers, a 3rd party host must be used that allows for the implementation of the functionality required by HoMePIT. These 3rd party services have a variety of limitations placed on them, depending on the tier of service utilized. It is possible that the tier of Google Firebase service that will be used to host HoMePIT may prove insufficient, which would in turn cause HoMePIT's functionality to be either limited or completely neutered.

Should this occur, the team will need to either search for a less limited server host, find a way to limit the usage of server resources, or pool funds to improve the level of service received from Google Firebase.

- For the shopping list feature to work as intended, the meal planning feature of HoMePIT must be at least partially implemented. This is because a shopping list will be created based on the ingredients that need to be purchased to allow a user to prepare the meals that they have planned. If the development of meal planning faces difficulties, then the shopping list feature can be partially implemented to at least create a shopping list containing ingredients that have run out or are currently below their threshold. When meal planning is later implemented, then the code handling shopping list creation can be updated to account for future meal plans.

Limitations

- Using Google Firebase is free via the Spark plan, although some features are limited unless paid for, such as Authentication (up to 10 SMS sent/day), Cloud Firestore, Cloud Functions, Cloud Storage, Hosting (10 gigabytes (GB), 360 megabytes (MB) per day), and more. We will work with these limitations to ensure that our application is feasible within the Spark plan. As noted earlier, if HoMePIT's development causes it to run into the limitations of the Spark plan, then a different and less limited server host may be identified, strategies for constraining resource usage may be developed, or an improved service level may be purchased.
- Due to the limitations of Google Firebase's Spark plan, some features, such as adding photos for recipes, may not be feasible due to storage concerns. In this event, we will opt to not include this or any such features. If such features do end up being desired towards the end of development, funds may be used to improve the level of service received, or strategies may be developed to allow for the implementation of data-heavy features, such as client-side storage of such data.
- As noted in the Feature List and Data Sources sections, HoMePIT may provide functionality for inputting UPCs or reading barcodes to input ingredients into the application. If this functionality is added, it is expected that the USDA's FDC API will be used to do so. Any given API key is limited to 1000 API requests per hour per IP address. We do not believe that we will exceed this limitation, as all team members are located in different households, and thus have different IP addresses. Thus, any given team member would have to exceed 1000 API requests in an hour (1 request per 3.6 seconds) before that limit becomes a problem. If this occurs, each team member will request a unique API key, such that one teammate cannot accidentally limit the other teammates. If the API request limit continues to be a problem, each team member will request a greater rate limit from the USDA FDC.
- Erik has no experience with web development, and thus will be unable to quickly assist either Jerome or Maxwell should they face difficulties in web development-related parts of HoMePIT. However, given that both Jerome and Maxwell each have web development experience, it is expected that they will be able to assist one another in web development tasks without the need for Erik. In the event that Jerome and Maxwell simultaneously experience difficulties that the other cannot resolve, Erik will attempt to help Jerome and Maxwell identify approaches that they have discounted or ignored.

- Jerome has some experience with databases, but none with Google Firebase. Jerome has limitations in his knowledge of Google Firebase and ASP.NET, as he may not have significant prior experience in these areas. While he brings expertise in database management to the team, he acknowledges that there's room for improvement in these specific technologies. Since Erik is more familiar with the backend and Maxwell with frontend, Jerome will play the more agile role in the project.
- Maxwell has little experience with working with databases, and has never worked with ASP.NET along with limited experience with Google Firebase. As such, he will be unable to assist Erik or Jerome with much of the backend of the project, focusing primarily on the frontend. In the event that there is some issue on the database side that Erik or Jerome cannot solve, Maxwell will step in to provide a third perspective on the matter.

Risks

- Erik is a source of risk because he is contributing to HoMePIT's development while simultaneously working 10 hours each week as a grader and tutor and contributing to 2 research projects in other departments. Furthermore, he intends to obtain his driver's license this semester. While Erik predicts that grading and tutoring should not exceed 10 hours each week, he also knows that some weeks at the end of the semester may be more demanding. Erik believes that he is nearing the end of work for one research project, with only some assembly, documentation, and potential presentation remaining. He is mainly operating in an auxiliary capacity for the other. He thus believes that his obligations towards these projects should decrease with time, leaving more time for developing HoMePIT. With respect to obtaining his driver's license, Erik believes that this will be done in periods of spare time that would not have been contributed towards the development of HoMePIT, and thus should not affect his ability to develop HoMePIT.
- Jerome is a source of risk on account of his freelance work, which could potentially lead to inconsistent availability and free time. Jerome's freelance commitments may vary in terms of workload and deadlines, which could impact his ability to dedicate consistent time to the project. While Jerome is dedicated to contributing effectively, it's essential to recognize that his freelance work may occasionally demand more of his time, leading to fluctuations in his availability.
- Maxwell is a source of risk due to his time being split between school duties and a 32 hour work week, which consists of two 16 hour shifts on Tuesday and Thursday. Despite technically being full-time at his job, only having two days out of the week dedicated to work leaves the remaining five days free to work on the project.
- The group as a whole is composed of seniors in their last semester of study at Austin Peay State University (APSU). Thus, they will be applying to job openings in the hopes that they will have a job secured prior to graduation. While the actual act of submitting a job application may be able to be done in a timely manner, any further interviews, tests, or other engagements required by a potential employer may take significant time from any team member's schedule. To avoid this problem, team members will endeavor to prioritize their project responsibilities above the process of submitting job applications, though they recognize that job interviews may take precedence over HoMePIT development due to the time-constrained nature of the former.

Timeline

- September 3 - September 9
    - Group: Submit Project Proposal
    - Erik: Research NoSQL, ASP.NET.
    - Jerome: Research ASP.NET, Firebase.
    - Maxwell: Research ASP.NET, Firebase.
- September 10 - September 16
    - Group: Set up an environment for project development, including Git repository. Begin coding. Determine rough idea of UI and UX desired.
    - Erik: Continue NoSQL and ASP.NET research, begin planning database.
    - Jerome: Continue NoSQL and ASP.NET research, begin planning database.
    - Maxwell: Set up Firebase.
- September 17 - September 23
    - Group:
    - Erik: Establish NoSQL database. Implement ingredients and recipe database entries as described in the Feature List.
    - Jerome: Finalize UI and UX plan.
    - Maxwell: Finalize UI and UX plan. Establish user sign in and authentication in Firebase.
- September 24 - September 30
    - Group:
    - Erik: Implement code to create, modify, and remove ingredient database entries. Implement code to create, modify, and remove recipe database entries without subrecipes.
    - Jerome: Implement code and/or implement a rough outline of the UI on the frontend.
    - Maxwell: Implement a rough outline of the UI on the frontend.
- October 1 - October 7
    - Group: Test each other's deliverables from the previous week. Integrate deliverables where possible and test the result.
    - Erik: Implement code to create recipe database entries with subrecipes. Implement code to automatically populate recipe nutritional information based on ingredients and recipes.
    - Jerome: Implement code and/or implement frontend application of creation, modification, and removal of ingredients.
    - Maxwell: Implement frontend application of creation, modification, and removal of ingredients. Implement additional features when applicable.
- October 8 - October 14
    - Group: Test each other's deliverables from the previous week. Integrate deliverables where possible and test the result.
    - Erik: Implement planned meals database. Implement code to add, modify, and remove planned meal database entries.
    - Jerome: Implement backend code and/or implement frontend application of recipe entry, display of nutritional information, etc.

    - Maxwell: Implement frontend application of recipe entry, display of nutritional information, and additional features when applicable.
- October 15 - October 21
    - Group: Test each other's deliverables from the previous week. Integrate deliverables where possible and test the result.
    - Erik: Implement database-adjacent code enabling described meal planning functionality.
    - Jerome: Implement meal planning database-adjacent code and/or implement frontend code. Assist in Implementing calendar functionality related to meal planning.
    - Maxwell: Implement frontend application of the addition of, modification of, and removal of meals from the planned meals database. Implement calendar functionality related to meal planning. Implement additional features when applicable.
- October 22 - October 28
    - Group: Test each other's deliverables from the previous week. Integrate deliverables where possible and test the result.
    - Erik: Implement code to create shopping lists from planned meal database entries and ingredient database entries below their threshold.
    - Jerome: Implement code to create shopping lists and/or implement further functionality from the week prior.
    - Maxwell: Implement further functionality from the week prior, and additional features when applicable.
- October 29 - November 4
    - Group: Test each other's deliverables from the previous week. Integrate deliverables where possible and test the result.
    - Erik: Implement supplementary database-adjacent shopping list code.
    - Jerome: Implement supplementary database-adjacent shopping list code and/or implement frontend shopping list feature.
    - Maxwell: Implement frontend application of the shopping list feature, and additional features when applicable.
- November 5 - November 11
    - Group: Begin creating the poster, if possible.
    - Erik: Delegate poster creation duties. Work on assigned poster duties. Test frontend functionality.
    - Jerome: Work on assigned poster duties. Assist in testing frontend and backend functionality.
    - Maxwell: Work on assigned poster duties. Test backend functionality.
- November 12 - November 18
    - Group: Begin poster if necessary, preferably begin finalizing poster. Begin finalizing the codebase.
    - Erik: Delegate poster creation duties. Work on assigned poster duties. Check that all guaranteed features have been completely implemented or are approaching that stage.
    - Jerome: Work on assigned poster duties. Assist Erik and Maxwell as necessary.

    - Maxwell: Work on assigned poster duties. Determine what changes need to be made to the frontend, if any.
- November 19 - November 25
    - Deadline: November 22: Poster due
    - Group: Finish and turn in the poster. Finalize codebase, aside from necessary changes mandated by testing.
    - Erik: Polish assigned sections of poster. Unify poster voice and style. Make any final backend changes necessary to ensure guaranteed features are available.
    - Jerome: Polish assigned sections of poster. Make any necessary frontend changes to ensure guaranteed features are available.
    - Maxwell: Polish assigned sections of poster. Make any necessary frontend changes to ensure guaranteed features are available.
- November 26 - December 2
    - Group: Prepare for technical interviews by individually studying the codebase, and asking each other questions concerning what one has worked on. Begin preparing the presentation, demo, final report, and supporting documentation.
    - Erik: Set up the basic outline of the final report. Delegate report writing duties. Contribute to necessary sections of the final report, presentation, demo, and documentation.
    - Jerome: Set up the basic outline of the presentation. Delegate presentation writing duties. Set up the basic outline of the demo. Contribute to necessary sections of the final report, presentation, demo, and documentation.
    - Maxwell: Document what documentation is required. Delegate supporting documentation duties. Contribute to necessary sections of the final report, presentation, demo, and documentation.
- December 3 - December 9
    - Deadline: December 6: Technical interviews begin
    - Group: Continue working on final report, presentation, demo, and documentation.
    - Erik: Contribute to necessary sections of the final report, presentation, demo, and documentation.
    - Jerome: Contribute to necessary sections of the final report, presentation, demo, and documentation.
    - Maxwell: Contribute to necessary sections of the final report, presentation, demo, and documentation.
- December 10 - December 16
    - Deadline: December 14: All code complete, final presentation and demo, final report and support documentation deadline
    - Group: Present presentation and demo, turn in final report and supporting documentation.
    - Erik: Polish final report. Double check presentation and supporting documentation.
    - Jerome: Polish presentation. Double check report and supporting documentation.
    - Maxwell: Polish supporting documentation. Double check report and presentation.