



MyKitchen

Team 90

Elliott Gorman
Garrett Griffiths
Krish Suchdev
Rowenna Switzer
Thinh Le

Overview

We find that in 2022, we are more connected through technology than ever before. Past ideas and new emerging technological products/services are often made to make life easier. We see this in our cars with Android Auto or Apple CarPlay, on services such as Zoom or Microsoft Teams where hundreds of people may be connected at the click of a button, and more. We find that as engineers and computer scientists, we have a unique opportunity to help lead that change.

The MyKitchen app was designed to address the issue of food inventory management within household environments. We aimed to address the issue of forgetting precisely what food items one has and where it is stored, and how it could be used. As such, we developed an app that allows users to keep a running inventory of their food and a digital recipe book that accounts for current inventory and cooking options. To better support both of these functionalities, we included a shopping list section to easily keep track of what a user intends to buy on their next outing. We successfully managed to meet both of our deliverable goals: we have developed an app that allows a user to track and manage a pantry to use with various features, and the app is in a nearly-shippable state following some minor adjustments to the functionality of the app.

Two of the main motivations of MyKitchen is to provide ease of kitchen inventory management and provide users an easy way to find recipes to make. There are similar apps to MyKitchen, but these often require paid subscriptions to use where the free versions are limited in use. Where MyKitchen stands out is to provide these services free of charge to the user. Plans to monetize MyKitchen would not involve limiting users to their use of the app. In addition, we recognize features that other apps have, such as barcode scanning, checking ingredients for a recipe, and more, to be necessary parts of such an application such as MyKitchen. These features are considered to be required to make this lifestyle app useful. Where my MyKitchen plans to stand is with the addition of many additional features, some of which are listed below:

1. Integration with smart assistants such as Apple Siri and Google Assistant.
2. Multiple users managing the same pantry, with a transaction log for the pantry.
3. Scanning of nutrition labels and importing grocery lists from grocers to quickly add items to the pantry.
4. The ability to store more than food into the app, such as pots, pans, and other kitchen equipment. This is useful when sharing pantries so a friend coming over knows what they need / don't need to bring.
5. Share pantries with not only users of the same household, but with friends as well, for viewing.

Table Of Contents

Overview	1
Table Of Contents	2
Project Deliverables	4
Schedule / Timeline / Scope	4
Communication Plan	5
Tracking Plan	6
Risk Analysis	8
Risk Mitigation Plans	9
Project Assumptions	10
Performance Metrics	12
Technical Process Plan and Implementation Details	14
Process Model	14
Mock Designs	15
Technical Details - Tools and Techniques	15
iOS - Mobile	16
Database	16
Final Implementation Designs	17
Impact and Security	17
Impact	17
Security	18
Individual Assessment	19
Elliott Gorman	19
Garrett Griffiths	20
Krish Suchdev	20
Rowenna Switzer	21
Thinh Le	21
Issues and Lessons Learned	21
Elliott Gorman	21
Garrett Griffiths	22
Krish Suchdev	23
Rowenna Switzer	25
Thinh Le	26
Future Work	26

Ethics Discussion	27
Appendix	29
Useful Sources	29
Building, Running, and Testing MyKitchen	29
Signatures Page	31

Project Deliverables

The following is a list of all project deliverables.

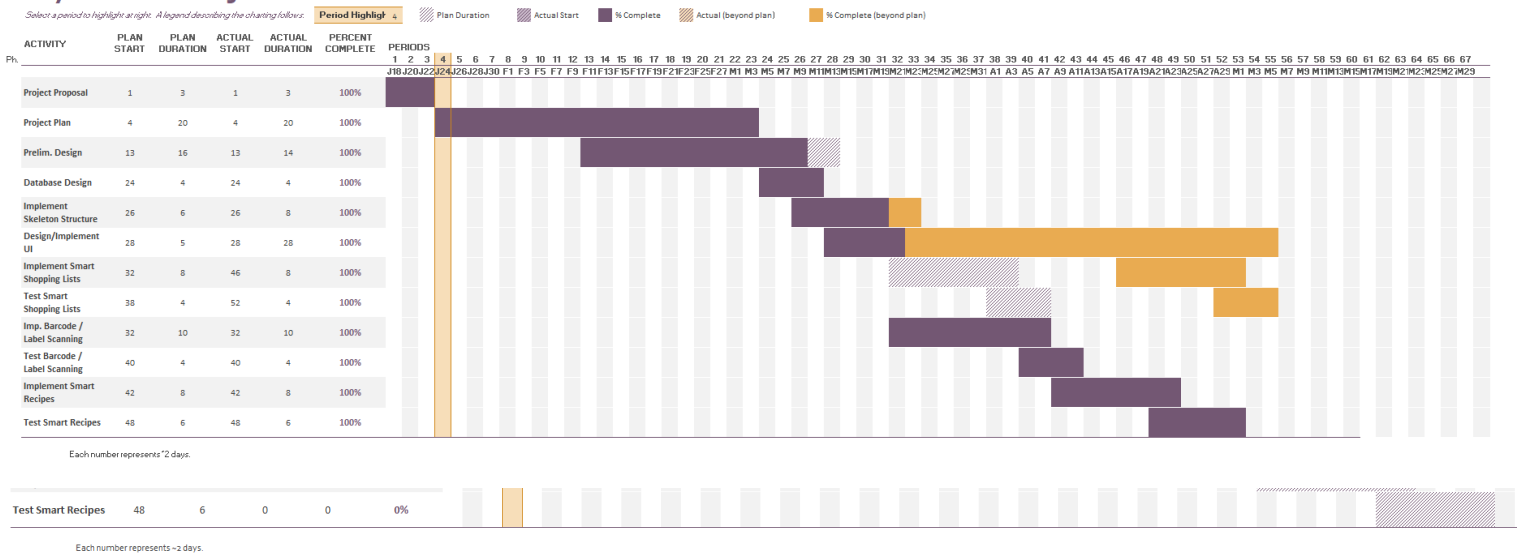
1. Deliver an app that allows users to track the items in their pantry, and use this information to provide various features described throughout this document.
2. The team ends the project with an app ready to be shipped to the app store come summertime (despite planned features past this point).

Schedule / Timeline / Scope

The Schedule / Timeline / Scope is provided in the form of a Gantt chart. The Gantt chart document is titled 'MyKitchen Project Plan - Gantt Chart.xlsx' and describes the milestones of the project, planned start/end and actual start/end dates. The Gantt chart has been used to track overall progress to help identify project progress continuously.

MyKitchen Project Timeline

Select a period to highlight at right. A legend describing the charting follows.





Task	Owner
Project Proposal (Internal)	Everyone
Project Plan	Everyone
Database Design	Everyone
Implement Skeleton Structure	Everyone
Design / Implement UI/UX	Everyone
UI Review/Evaluation (Testing)	Everyone
Implement Smart Shopping Lists	Garrett, Rowenna, Elliott
Test Smart Shopping List Feature	Garrett, Rowenna, Elliott
Implement Barcode / Label Scanning for entry into Database	Thinh
Test Barcode Scanning Feature	Thinh
Implement Smart Recipes	Thinh, Krish, Elliott
Test Smart Recipes Feature	Krish, Elliott

Communication Plan

The following was the communication plan for our team. It covers the type of communication, communication schedule, communication mechanism, who initiates, and the recipient. These meeting times were loose approximations of when communication would take place. During the course of the project, team meetings were generally held more often, and peer-to-peer meetings (not listed below) took place many times a week.

Type of Communication	Communication Schedule	Communication Mechanism	Who Initiates	Recipient
Team Meetings	Semiweekly	Discord	Project Team	Project Team
Project Status Reports	Weekly	MS Teams	Project Team	Faculty Mentor Project Team
Project Review	Weekly	MS Teams	Project Team	Faculty Mentor Project Team
Requirement Changes	As needed	MS Teams	Project Team	Faculty Mentor
Issue Reports	As needed	MS Teams	Project Team	Faculty Mentor Project Team
Kickoff meeting	Once	MS Teams	Project Team	Faculty Mentor Project Team
Project Proposal	Once	MS Teams	Project Team	Faculty Mentor Project Team
Project Plan	Once	MS Teams	Project Team	Faculty Mentor Project Team
Poster Approval	Once	MS Teams	Project Team	Faculty Mentor
Final Report	Once	MS Teams	Project Team	Faculty Mentor

Tracking Plan

We made use of Azure Repos (a host for Git) to monitor code changes and code related issues. Stakeholders will be added to the Azure DevOps project to provide them a means of tracking the project's progress in addition to what is listed below.

The following table is a list of items that we have collectively decided are worthwhile to develop. This list is not necessarily complete, nor should our predictions as to if/when they will be completed be taken as absolute. Work performed by members will be distributed equally and most milestones will be assigned to every member so that each deliverable is able to be reasonably checked and certified by each member of the team.

Phase	Milestone	Percent Completed	Predicted Date of Completion	Date Completed
1	Project Proposal (Internal)	100%	Feb 4th	Feb 4th
	Project Plan	100%	Feb 28th	Feb 28th
	Backend Database Framework for Data Storage	100%	Mar 18th	Mar 26th
	Implement Skeleton Structure	100%	Mar 25th	Apr 12th
2	Design / Implement UI/UX	100%	Apr 1st* (And Ongoing)	Mar 31st
	UI Review/Evaluation (Testing)	100%	Apr 1st - * (Ongoing)	Mar 31st
	Implement Smart Shopping Lists	100%	Apr 8th	Apr 14th
	Test Smart Shopping List Feature	100%	Apr 15th	Apr 14th
	Implement Barcode / Label Scanning for entry into Database	50%	Apr 22nd	April 18th
	Test Barcode Scanning Feature	50%	Apr 29th	May 6th
3	Implement Smart Recipes ¹	100%	Apr 22nd	April 7th
	Test Smart Recipes Feature	100%	Apr 29th	May 6th
	Allow Multiple Users to Manage The Same Pantry	0%	TBD	N/A
	Recognize and Lookup Expiration Dates for Scanned Items	0%	TBD	N/A
	Final Report	100%	May 2nd	May 6th

¹ Smart recipes are recipes that may be suggested based on items currently owned in your kitchen pantry.

Risk Analysis

The table below lists the details associated risks identified for our project. A breakdown of these risks is given; afterwards, risk mitigation and management is discussed.

The following table details risks identified, their likelihoods, impacts on the project from low (1) to high (5), and the risk exposure for each. These metrics were used when designing and creating the risk mitigation plan. The likelihood is an estimation based on how likely the risk is to occur during the lifetime of the project; the impact is the expected level of negative pushback on the project if a given risk occurs; the risk exposure, in this case, is the level at which we need to understand and mitigate high risks.

ID #	Risk	Likelihood (0-100%)	Impact (low 1 - high 5)	Risk Exposure (Likelihood * Impact)
1	Ramp-up time takes longer than anticipated	30%	3	0.9
2	Task(s) take longer than anticipated	50%	2.5	1.25
3	Team Members are unable to put in the needed work for a given week.	25%	3	0.75
4	Lack of clarity/cohesiveness in tasks and work expected	10%	2	0.2
5	1 (or more) project specifications are not met.	20%	4	0.8
6	Software Issues and Bugs	95%	2.5	2.375
7	Team Member is Quarantined / Out sick due to COVID-19	5%	4	0.2
8	Team Members are unable to obtain the necessary hardware or software to complete their work for a given week	10%	3	0.3

Risk Mitigation Plans

Risks **1, 2, 3**: Have the affected team member(s) commit to make up last week's work along with the current week's workload. Additionally, other team members will assist when able to ensure task completion.

Risk **4**: To mitigate this, we will ensure all team members understand the tasks for each week and what is expected. Additionally, we will perform reviews at weekly meetings to ensure everything is on track. If this issue occurs, we will plan the changes into the next week's sprint, using the allocated leeway time as necessary to replan sprints.

Risk **5**: This risk will have varying impacts. This issue will occur if there is a blocker that prevents the task(s) from being completed, or causes the task(s) to take much longer than originally anticipated. To mitigate this, we will use our initial development plan to identify major design decisions which will help us avoid unexpected issues. If this issue occurs, depending on the specification affected, we will either: 1) Change the design based in accordance with other project specifications OR 2) Adjust the design in order to meet deadlines while complying with the original goal.

Risk **6**: Software bugs are a given and hard to mitigate. Instead, we have a plan for dealing with bugs. When software bugs or issues are identified, the team member who found the bug will create a "Bug" work item in Azure DevOps with reproduction steps, priority, and other comments. This will then be integrated into our backlog and placed into coming sprints or fixed as they arise, depending on the issue.

Risk **7**: To rectify this issue, one of two solutions will be used: 1) The team member is able to work and no issue occurs; or the team member will be able to complete some work and other team members will finish the work OR 2) Other team members will take on the work and the leeway time will be used.

Have other team members divide up the affected person's workload for the week.

Risk **8**: Work will be reassigned to other team members that have the necessary hardware/software to complete the task. Work not requiring specific/hardware will then be assigned to the affected team member(s).

For each of the aforementioned risks, plans have been put in place to mitigate the impact of the risk. Now, we will discuss how these issues will be tracked.

Issues (and thereby bugs) will be tracked in weekly status reports and in communications with the team. When a risk or software issue arises, a team member will notify the team

and discuss the issue in their weekly report. We will then take into account this issue in the current and coming sprint, changing plans as needed. This will allow us to adjust our outlook to help ensure project completion within the allowed time frame. We will also allow for leeway time in the project plan to help add development time should risks occur.

Risks may often be overlooked or hard to identify. For all risks, including those not provided in the previous list, we will use the following procedure to identify risks:

1. During development meetings, identify any blockers both individually, and as a group.
2. For the next sprint, re-assign team members to prerequisite tasks of the identified blocked tasks to stay on track.
3. Additionally, team members will identify issues they face individually in accordance with the issue listed, and any risks that may arise.

To avoid risks, we will follow an Agile approach. Doing so will allow us to respond to risks and changes to the project specifications quickly. Individual risk (risks that apply to all team members individually) will be tracked and managed by each team member. Risks that are of larger scope and impact will be managed by the team during weekly team meetings. During these meetings, risks will be discussed and updated accordingly based on the current outlook of the project. This will allow us to track risks, and update them as necessary; additionally, this allows us to respond to changes in individual risks.

To manage risks, Elliott will oversee monitoring of major risks and ensuring the team tracks and monitors risks as necessary. It is estimated that 1-2 hours per week will be allocated to risk management activities and discussions amongst the team.

Project Assumptions

Assumptions made regarding this project involved expectations about either a tool or person. Therefore, there are no project assumptions being made concerning requirements. The following is a list of these assumptions; afterwards, these are discussed in greater detail.

ID #	Assumption
1	The barcode / label scanning will be easily implemented through the use of a technology that provides this ability.
2	Allocated ramp-up time allocated for team members is sufficient.
3	All equipment, e.g. MacBook, is acquired before the project goes into the

	development phase.
4	Everyone who was involved at the beginning of the project will remain in the project. That is, no one will drop this course.
5	There will not be many requirement changes.
6	Going into phases that go beyond the scope of this course, some members of the team will not be continuing with this project

Each of these assumptions posed a potential risk to project delivery. To understand these risks, each assumption is discussed in greater detail below.

1. The use of Barcode Scanning and relevant APIs should be reasonably streamlined to implement into the project, but it is entirely possible that it becomes more complicated than expected, which could delay deliverables. This assumption is tied directly to risk 2 and 5.
2. Two team members have experience with iOS development while the others have little or no experience. Therefore, ramp-up time is allocated for these team members without experience based on an assumption regarding the time needed to learn the relevant iOS skills. If more time is needed for these team members, the allocated leeway time may be used.
3. MyKitchen will be developed for iOS enabled devices, and therefore team members need access to an Apple Macintosh. A request was made to the University of Texas at Dallas to provide an Apple Macintosh to team members who do not have access to one. Failing to obtain a Mac device may result in delay in the development phase.
4. This project was intended to be a 5-person project and it is assumed that this number will remain throughout the project phases. Project delivery may be delayed in the event that one member drops this class for any reason. In such circumstances, workload must be reevaluated to ensure the delivery of the project.
5. Requirement changes are inevitable and it is anticipated. However, careful thought and much time has been spent on engineering the functional and nonfunctional requirements, so we expect that there will not be much change to the project requirements during the later phases of the project.
6. Each individual in this group has different plans for post-graduation, of which phases 3+ will fall under. As such, it is likely that the project will undergo development at a slower rate due to some members moving on.

Performance Metrics

In order to track performance metrics for the project, we identified the following measures at the beginning of this project.

1. Is a particular feature implemented in such a way that it fully meets our original goal/expectation?
2. Are we meeting weekly goals?
3. Implementation of the project
 - a. *Note: This metric includes the following items:*
 - b. Is each team member meeting their goals and contributing to the parts of the project that they committed to completing?
 - c. Is each team member able to implement their part of the project relatively smoothly without creating numerous bugs?
 - d. Can each team member explain the process to replicate the results of major features with relative ease?
4. User Feedback


These metrics aided us in ensuring project success and completion. In order to measure each metric, we identified the following methods:

Metric 1: During weekly meetings, team members' implementation of a given task will be compared to original goals and expectations for the feature. This will allow us to identify whether or not our goals need to be adjusted during the project.

Accomplished: The features implemented were successfully developed to meet our vision for the project. While we are not finished with this project and plan on continuing development on many features, with the features currently completed we have met our initial goals. With the MyKitchen app, people can rest easy knowing they do not have to monitor their pantry manually, and we've created many useful features to support this.

Metric 2: Through weekly reports and the use of Azure DevOps, we will be able to measure status completion, and completion towards sprint goals. This will allow us to determine how our weekly goals are being met.

Accomplished: All members were able to complete their weekly goals by ensuring work distributed was given with realistic and reasonable expectations. Through risk management, we accounted for emergency situations (accounting for scenarios such as COVID or family emergencies), ensuring everything was performed on time



even in such occasions. In occasions where a member was unable to make a weekly goal, the deficit was made up in the following week or assisted by other members to ensure that goals continued to be met

Metric 3: For each of the items listed in this metric, we will do the following: Measure team members' work ability to produce clean, reliable code through the use of code reviews; Team members should exemplify understanding of significant components they were involved in building. This will be measured by whether or not team members are able to explain how said component was made in varying levels of abstraction.

Accomplished: Development of features was handled with care and respect to other portions of the app and proved to help reduce the complexity of resolving merge conflicts and avoid problematic errors in code. As a result of these good practices, most features have been implemented without problems concerning erroneous code. Instances in which merge conflicts became or could become an issue were immediately handled by all relevant members and resolved to prevent the issue from escalating.

Metric 4: At each major feature testing release, we will create a testing environment with which to give users a platform to review the new featureset. Giving them categories such as "Ease of use", "Performance", "User Satisfaction", or other relevant categories to rate on a 1-5 scale, we can take their review as a metric with which to adjust our implementation for future versions.

Accomplished: We were unable to perform user testing in the manner described above due to the scope of this semester and the time sensitivity associated with its short duration. We were able to demonstrate most features to select individuals and were given useful feedback that improved the way we approached many features, but this does not reach the level of user testing we originally intended.

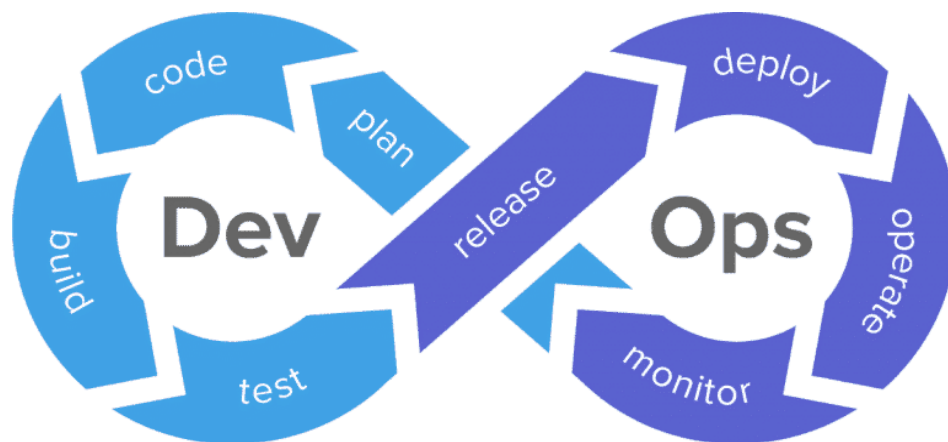
In addition to the metrics listed above, we have successfully implemented the following features on the MyKitchen app:

1. Allow users to add, track, search, and manage items in their pantry through the MyKitchen app.
2. Allow users to create recipes, and within those recipes allow users to easily see what items they have and what items they do not have, while also allowing users to export those items to a shopping list.
3. Allow users to create shopping lists that connect to the users recipes and pantry.

Technical Process Plan and Implementation Details

At the beginning of the project, we identified that we would be making use of the following skills: iOS Development, Artificial Intelligence / Machine Learning through the use of Image Recognition, Relational- or Object-based Database design, Front-End and Back-End Development, User design and User Experience, and Smart Assistant APIs such as Apple's Siri. Although not all of these skills have been utilized at the completion of this semester project, they will be utilized in work concerning future goals of MyKitchen.

Process Model



source: <https://nub8.net/>

For the development process model, we used DevOps (through the use of Azure DevOps) where everyone was responsible for implementing and testing the feature to which they were assigned.

iOS - Mobile

- Pantry List View
- Pantry Detail View
- Barcode Scanning Items Into Pantry
- Barcode Scanning Data Lookup*
- Siri Integration

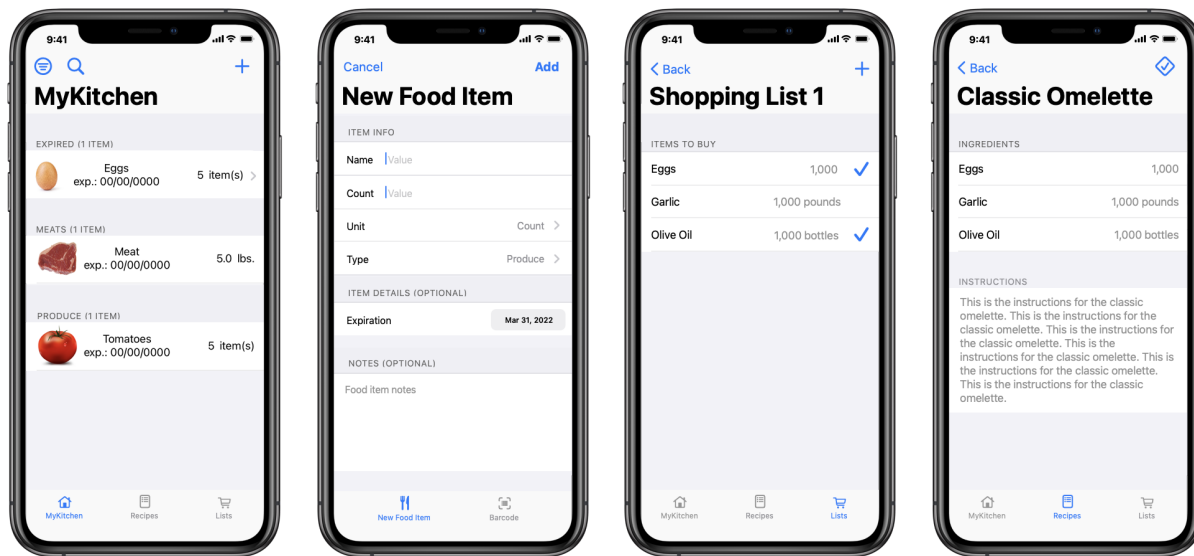
Database

- Table to Store Pantries
- Table to Store Ingredient Information
- Secure User Authentication Data Storage
- Multi-User Support for Sharing Pantries*

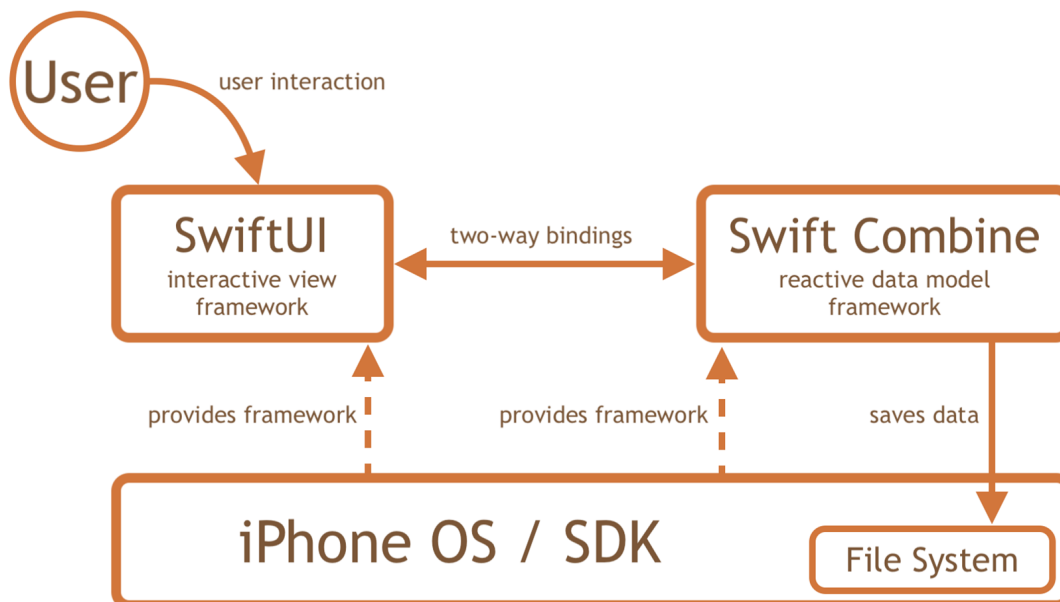
* Stretch goals.

Mock Designs

The following are snippets of the originally intended design of MyKitchen. These are mock designs and were made before any development work started, so these views were developed without any functionality.



Technical Details - Tools and Techniques



iOS - Mobile

The native iOS mobile app was built using the SwiftUI and UIKit frameworks from the iOS SDK running on the latest version of Xcode.

SwiftUI is newer, simpler, and is easier for developers that are new to iOS development to pick up. However, due to those reasons, it is also more limited in terms of functionality. It is also a declarative (and usually reactive) language. This required our team, most of whom have experience almost exclusively in imperative languages, to become accustomed to new programming paradigms. Starting up a new app with simple primitives like Views and Lists is much easier in the SwiftUI framework, with little-to-no configuration needed. In addition, it allowed team members to become better acquainted with separating views and models, which is one of the most important best practices in iOS development.

However, we also made use of **UIKit**, which is the more advanced framework with deeper control and more customization options, where SwiftUI was not possible. More specifically, we implemented UIKit UIViews embedded within SwiftUI Views in the barcode scanning functionality of the app. While UIKit is a long supported framework within iOS development, given the time frame of the project, this framework presented a somewhat steep initial learning curve for team members unfamiliar with iOS development. High complexity configurations are admittedly harder to achieve than in UIKit, and can require deep understanding of SwiftUI and its particular declarative and reactive quirks. However, in the case for the barcode scanning feature, the full functionality that we needed to pull that off was not yet supported in the current iteration of SwiftUI, so we had no choice but to utilize UIKit.

Official Apple Tutorial of SwiftUI:

<https://developer.apple.com/tutorials/app-dev-training/#swiftui-essentials>

Official Apple Tutorial of UIKit:

<https://developer.apple.com/tutorials/app-dev-training/#uikit-essentials>

iOS - Database

In both UIKit and SwiftUI apps there is a default key-value object store provided by the system. When an app is quit by the system, the app should ensure that its objects are stored outside of main memory so that it can start again later without data loss.

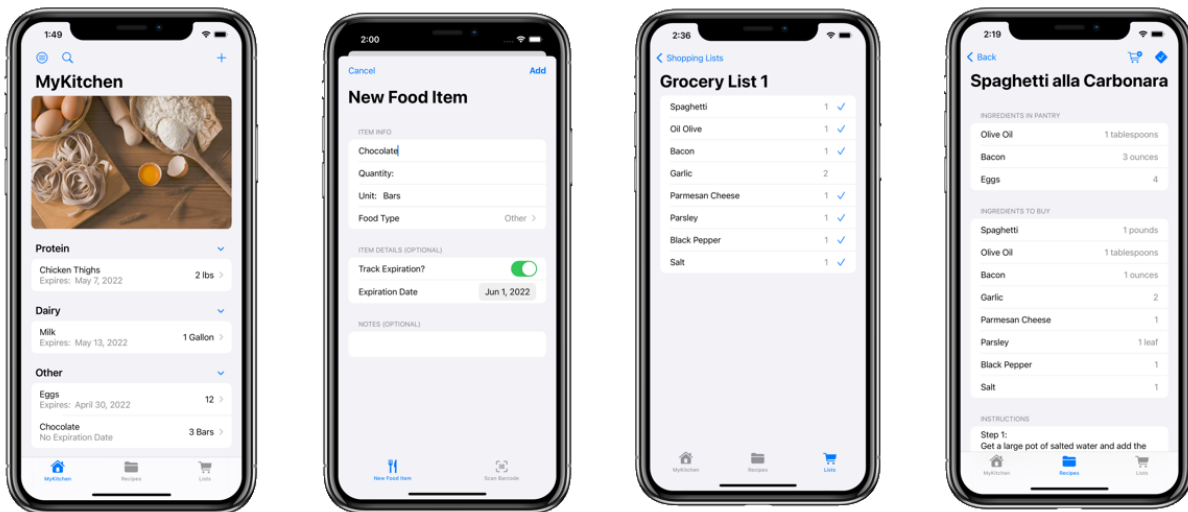
UserDefaults is offered as a key-value object store meant to be used for small-medium amounts of data (filling it with gigabytes of data will unacceptably slow down app launch).

Our databases, which will only contain kitchen inventory and a grocery list, should be sufficiently small to be acceptably stored in UserDefaults.

When using the SwiftUI framework (as in iOS approaches 2 or 3), asking UserDefaults to store an object is as easy as marking it with the @AppStorage decorator.

Final Implementation Designs

The following are snippets of the current functionality (but not of design) of MyKitchen. We slightly adjusted from the original planned design where necessary, but the core functionality and design remains the same. Starting from the left, we have the: pantry view of MyKitchen, new Food Item view of MyKitchen, shopping list view of MyKitchen, and finally the recipe detail view of MyKitchen.




Impact and Security

Impact

MyKitchen will have a great impact on its users. The use cases of the app have far reach, especially for users that have trouble deciding what food to make based on what they have, or for those who have trouble grocery shopping. MyKitchen aims to make the use and management of a kitchen much simpler. People who struggle with this or who would like to make that process easier can benefit from MyKitchen.

Being able to choose recipes and understand what needs to be added or used will help to save a great deal of time in planning meals and outings centered around meals. The time saved through the MyKitchen app will likely vary greatly from user to user, but saving any time at all while offering a permanent increase in accessibility to home cooking would have



an overwhelmingly positive lasting impact on the user. Users will be able to save time by planning grocery lists based on recipes as these items can be exported to a list directly. Additionally, users wanting to find something to cook will have an easier time doing so using the smart recipes feature, allowing them to see what they have and do not have for a recipe.

Additionally, preparing a meal yourself and understanding what is in that meal is almost always healthier than eating out at a restaurant or fast food establishment, and is most certainly more nutritious than snacking. Nutrition aside, having more structured eating habits does a great deal of good for a person's physical and mental health, which would be another positive effect that the MyKitchen app could have on a user.


With the smart recipe and expiration tracking features, users will be able to save food and money by letting less food go to waste. Furthermore, home cooking and personal meal preparation are significantly cheaper alternatives to buying similar food at restaurants. That said, when many people consider what to cook, there are often a number of small factors such as not having a certain ingredient or having something unknowingly expire that can prevent someone from cooking. The MyKitchen app aims to resolve this by having these small factors be preventable through the management of a user's pantry. This enables them to save money and avoid accidental waste, which is beneficial for the environment as well.

Additionally, users will be able to quickly know if they have any particular item. This is beneficial because it saves the user from buying too much, or not buying something they need, without having to ask another person if possible. Future work for MyKitchen includes shareable shopping lists and pantries, where these features will also provide users an easy way of knowing what friends have, making gatherings requiring these details easier.

Security

The MyKitchen team sees the security and privacy of our users as a very important issue. We have already considered both throughout the design of our app and in thinking about the future development of the app. In the app's current state, there is no personally identifiable information (let alone SPII) that is collected from the user. The app is, however, entrusted with user generated content. Provided the app is being used as intended, that content will only pertain to the contents of their pantry. However, user-generated data, even if not personally identifiable, should of course be treated with care.

In the current version of the MyKitchen app, user-generated data never leaves the phone. We have no server-assisted functionality nor third party dependencies that could possibly access user data. We do have plans for future work to enable users to make accounts and share data between multiple accounts, where security will be a concern. In that future



work, we will make use of modern encryption algorithms to ensure user data is encrypted both in transit and at rest on any external server.

Furthermore, user data sitting at rest on the phone is protected from other apps by the operating system. iOS gives each app its own view of the file system and other OS facilities. This environment is called a sandbox, and it is the responsibility of the operating system to ensure that apps may not access each others' sandboxes. Except in specific cases (such as multiple apps created by the same company, like Google's suite of apps), the operating system guarantees that apps can't even tell which other apps are installed on the phone, let alone access their part of the file system. Any breach to iOS sandboxing would be a major breach in the security of the iOS platform as a whole, and the platform would likely be patched very quickly after disclosure.

Lastly, all iOS devices capable of installing the MyKitchen app will have a fully encrypted file system and a trusted platform module, which present a major roadblock outside actors, even if they have physical access to the device, provided that the device is locked. Apple's platform guidelines recommend that data that is known to be sensitive (like a password or SSN) should still be encrypted within the app's sandbox using the cryptography libraries provided by the operating system, but in the current state of the app we believe this extra precaution is unnecessary.

Individual Assessment

Elliott Gorman

Prior to the development phase, I helped the team meet project deliverables and worked to make sure the team was on track. I helped codify the MyKitchen app design, and, along with my other team members, helped bolster the team's vision for the project.

During development, I implemented a number of features and continuously worked on improving the MyKitchen app's design. I added the ability to allow users to add new food items manually, the ability to create new shopping lists and add items to shopping lists, the ability to search items in the MyKitchen pantry, the ability to create new recipes, and improved the UI/UX of the MyKitchen pantry, the recipe views. I also helped fix a multitude of bugs that were caused by my additions and others to the app, as well as bugs that arose during the integration between team member's additions.

Garrett Griffiths

For this project, I handled a majority of group coordination and helped everyone to seamlessly make decisions during discussions so as to more efficiently establish plans and visions for the project. I organized the majority of meetings and gave great care to any deadlines, internal or external, to minimize potential issues concerning deliverable submission or falling short of our schedule.

Additionally, I tried to handle or lead as many of the non-programming aspects of this course as I could (written deliverables, documenting meetings, external communication, etc) to allow those that were working primarily on development to be able to do so without interruption.

Before the start of development, I worked with the team to come up with a concise data model and preliminary views upon which to build the MyKitchen app, as well as organizing deliverables such as the Project Plan, Project Proposal, and other CS 4485 assignments.

After development began, I continued to perform clerical and document-centric duties, as well as helping guide the creative direction/ vision for the MyKitchen App. I helped with the conceptual design and direction of the UI and Views as a whole, and helped with some implementation of the Shopping List feature.

Krish Suchdev

Before development, I primarily worked with the team to finalize the details of our idea. I also worked closely with Elliott to develop the mocks for the design of our app.

In the early stages of development, I created the initial Xcode project for us to develop the app. In the setup of the project, I developed the tab bar view to separate the 3 primary features of the app and organized the file system of our Xcode project in this way as well. As many other team members already individually developed features outside of the project, I often took charge of merging many of these features, which often had merge conflicts, to allow them to work together. I also helped develop the general UI/UX of the app and made many adjustments to various views to provide more consistency across the app.

Later on in development, I took charge of the recipe detail view, which displays the instructions and ingredients needed for each recipe, and the functionality behind the check ingredients feature, which checks the users' pantry to identify which ingredients they do and don't have for a recipe. This feature was also integrated with the shopping list to allow users to export ingredients they don't have into a new shopping list. I also worked closely with Thinh to develop the food view to group foods by their food type (produce, protein, etc). Having prior experience with Swift, I often helped out team members and showed

them best practices as they developed other technical features over the course of this project.

Rowenna Switzer

In initial development, I created the original implementation of the data model and its persistence/storage methods, which taken together form a database. I also created the initial version of the new food item view, which would be iterated on by Elliott later. Later in development I would heavily modify some of Garrett's initial work on the shopping list tab to create the meta shopping list view (which shows the list of your shopping lists), the shopping list view, and the detail view for shopping list items. I also cleaned up a lot of state bugs in the codebase that involved handling the translation of nil values in the database to views. To aid this, I created new extensions on the Swift Combine two way bindings in order to properly translate nil values to and from a default value for safe access and update by views. Throughout development, I often took the role of SwiftUI expert, and was largely in charge of the database. So much of my work involved ensuring that state was safely and elegantly tied into the app's views so that updates from the views are properly propagated to the database, and data from the database is safely propagated back to the views.

Thinh Le

Since this is an entrepreneurial project, we didn't really have to spend a lot of time on the requirements engineering part of the development. An idea of what the app would be like had been thought out, so my main responsibility prior to the development phase was to ask for clarifications on the parts of the project that I didn't understand. Beside that, I also attended weekly meetings and worked on group assignments. During the development, I was tasked with developing the Pantry view, Item detail view, and the Recipe view.

Issues and Lessons Learned

Elliott Gorman

The main issue I had when working on this project was developing in iOS. I have played around with a multitude of platforms and technologies, but my passion and strong skills reside in embedded programming, including C and assembly. Learning SwiftUI was difficult at first because of the restrictions Swift imposes upon programmers. These restrictions have made more sense as I've learned more of SwiftUI; however, they caused me much strife at the beginning of the development phase. Fortunately, I have played around with app development for Android with Kotlin, and with Flutter for iOS and Android. My prior experience with these technologies aided me due to the similarities in structure between

the languages. Drawing upon these past experiences helped me understand SwiftUI concepts more easily.


In addition to the aforementioned, I worked through official Apple SwiftUI tutorials, and constantly referenced other sources to learn more about SwiftUI. When it came time for implementation, I struggled at first, but over the course of the development cycle, I've become comfortable with SwiftUI and am able to implement features much quicker. Now that we are concluding the project, I feel comfortable enough to say that I can program in SwiftUI! I still have a lot to learn if in the future I pursue more SwiftUI, and I may or may not actually do so, but I'm happy and have enjoyed learning this new technology.

Concerning learning from past courses and how they have helped me with this project, the important note to make is that this project largely differs in two ways from other similar semester-long projects: 1. Development work follows the Agile methodology closely due to the short life-span of the project; 2. The project requires teams to engage in project management. These are software engineering skills, and therefore knowledge from CS 3345 was immensely helpful. The terminology and ideas learned in CS 3345 applied directly to this project, and that made understanding the project deliverables and goals more clear.

Personally, I feel as if the goal of CS project is to provide students the opportunity to work in an environment that mimics industry work. I believe that the course does a good job at achieving this goal. However, I think there should be more opportunities for theoretical CS work, and not just software engineering related work. Many computer science majors will seek software engineering work after graduation, so while this does align with many students, it does not align with computer science at its core. I believe that having opportunities for students to develop mini-theses or similar work as a part of CS project would improve the course experience for some students. As an undergraduate pursuing a Masters degree and planning on doing a thesis, I would have loved the opportunity to explore that path of computer science more, rather than doing more software engineering related work.

Garrett Griffiths

The huge issue I had to manage for this project was insufficient hardware. We decided fairly early on to develop the MyKitchen App for iOS, as it would allow us to initially reach a larger potential market and made the most sense to begin with given that two of our members had experience in app design previously. Unfortunately, I was the only one in the team that did not own a macOS system, which was required to be able to develop in the chosen language. I was unable to obtain a MacBook from the university and thus reached out to my family and managed to procure a mid-2012 macOS laptop that met the minimum requirements to use SwiftUI. Unfortunately, SwiftUI is an incredibly young language and thus has undergone rapid updates, and the MacBook I received was incompatible with the version we were using for the MyKitchen App. I finally decided to simply purchase a current



macOS device to be able to help with development, but that solution did not come for some time. As a result, I was unable to reasonably participate in actual development for the majority of the semester, so I took it upon myself to make up for that by handling as much of all other facets of this project as my time would allow. As a result, the team was able to develop more efficiently and I handled the majority of documentation, planning, and external communication in their stead.


Over the course of this project, the primary things I've learned involve how to properly deal with hardware/software requirements and version control (as a result of the macOS issue and compatibility problems I had), as well as how to approach managing a team to better produce our intended results. Because I was not able to focus on development due to the aforementioned issues, I was able to focus on and learn how to manage the non-developmental aspects of this industry.

Regarding a concept from a previous class that proved useful, I would have to mention the documentation and planning concepts taken from CS 3354 Software Engineering. The tools learned in this class, and just the broader concept of planning out a project, proved quite useful in ensuring that the project proceeded seamlessly. A major item I learned this term that helped the project succeed would have to be how to be decisive. Through my role in this project and through our meetings with Venky, I learned the value of being truly decisive, as it helps to keep the conversation on track while still allowing for all members to voice their concerns.

Regarding what can be done to improve the course, I would recommend allowing more time and possibly holding discussions whenever deadlines are involved. I believe that the course was handled well and I had a good experience going through this project with my team, but we had a great deal of confusion on the logistics of many assignments, as well as some concerns with what exactly the assignment asked for. As such, spending more time to elaborate on assignments would mitigate confusion, and holding a discussion during class time or some other medium would allow the class the opportunity to have most of their questions answered, instead of having to email the professors questions regarding each assignment

Krish Suchdev

While I had many years of prior iOS development experience, I had little experience working with the SwiftUI framework. By design, SwiftUI was built to be simplistic enough for new programmers to pick up, yet complex enough to be used in large-scale production iOS apps. It's more of a declarative language and not something that I was used to working with, so it was tricky for me to make that shift in mindset. SwiftUI is deemed to be the future of iOS development and I always meant to pursue that avenue, so this project was the perfect opportunity to work on it for me and my team members. I would say I am



definitely very passionate about iOS development and it was a real privilege to be able to share that passion and work with my fellow teammates with it for this project.

One incredibly valuable skill I learned in this project was constructively building off of the ideas of others. As with any long-term project, we did not agree on every decision. I've always believed in staying overly positive when working with others and tending to avoid conflict at all costs, but this often leads to me keeping my ideas to myself. I'm naturally not the kind of person to always voice my disagreements and opinions, but working on this project forced me to develop some communication skills and I've found it's much better to build off of people's ideas rather than staying silent. I feel like I could've contributed more in this way, but I've become more comfortable continuing to do this now that I've started to develop this skill. Regardless, I think individual and group communication is a very important skill for a long-term group project like this one. It's probably impossible to not improve this skill over this course since there is so much planning and designing involved even before development starts.

As for my previous courses taken at UTD, I've found the software engineering class (CS 3354) and the software project planning class (SE 4381) incredibly useful. Both classes taught me the software project lifecycle and the importance of developing a plan. For our project, we intended to follow an agile-like approach with week-long sprints and I think all of us understood this relatively well because this was taught in at least one class we all took.

Overall, this course was well structured and I would say one of the most relevant classes in regards to real-world industry experience. As someone who had multiple internships throughout my undergraduate career, I definitely see the parallels between this course and software engineering in the industry that I don't see in many of the other classes I didn't mention already. I believe soft skills are a major underrated skill to have in software engineering and I think that this is one of the few courses at UTD that actually discusses this. While I think this is great, I would recommend this course to go further. I think this course could take a more direct approach at giving suggestions on how students should be communicating with their team and even mandating specific meetings for teams. Meetings that are common in agile teams, such as standup and team-building exercises, are very crucial in the industry. I think our team did an excellent job with this, but if it hadn't been for the internships I've had so far, I would be lost at this. Another suggestion I would give is to move quicker at the beginning of the semester in terms of team forming and project aligning. I like the fact that this course is divided into 2 major parts of requirements/design and then implementation, but I felt more time in the development portion would have been more helpful and realistic. The amount of work our team needed to put in the project was uneven over the course of the semester and it definitely felt more rushed towards the end.


Rowenna Switzer

An issue I personally faced earlier in the project was to do with how and when to speak in meetings with our project sponsor. A week or two before spring break, I was confronted by Garrett about it as an issue I needed to fix. I believe I incorporated that feedback well, and Garrett confirmed recently that he was impressed with the improvement as well. It didn't cause any other problems within the group for the rest of the semester. I, along with the rest of the group, began receiving feedback from our project mentor with minor pointers for how better to speak about the project development with external parties like him. That feedback is greatly appreciated on my part, and I believe that altogether the feedback I've gotten on how best to speak in those sorts of engagements is my most important takeaway from this project.

Later during development we encountered significant merge conflicts due to some duplicate views being developed (in order to implement new features). Some were developed without existing clean relations to state, while some were simple views I'd created with minimal features early on for the sole reason of having clear and complete relations to state. We called a meeting to discuss how we wanted to move forward. At that meeting, I explained my thinking on how the features from the more complicated views could be moved into the views with a clear relation to state, while I believe the converse would've been much more difficult, as managing state in SwiftUI is a subtle and difficult art. The rest of the group agreed, and in the oncoming days there was a team effort to merge view functionality from duplicate files into the files that already had clear relations to state. The team kept in close contact, discussing what to keep and what to move in various files. I was most involved in cleaning up state variables used in view functionality, including places where there were subtle nil-referencing errors.

Among courses I've previously taken at UTD, I believe the Software Engineering course (CS 3354) was by far the most helpful. It is only because of that course that I did not feel lost in the first half of this semester doing so much project planning.

Overall, I found this course to have fairly minimal structure. Through the knowledge and team dynamics present in our group, we were successful at running our own standups and doing our own project management and generally being self-sufficient, and I believe we would be similarly successful in a similarly unstructured environment, like a Hackathon. This has been the best group I've ever participated in for a school project. The lack of guardrails in this course has not bothered me because of how well the semester has gone with this group. I do think that the need to put forth a project plan of some sort before development begins is good, but there were only 6 weeks after spring break, and some parts of the plan can only be properly developed when some work has been done. I believe starting the development time earlier, and perhaps asking for a second revision of the plan after spring break, could have been beneficial. I have already had multiple internships in



industry which trusted me to be a competent engineer. Each was an incredible experience, and they may have robbed this course of a significant amount of its purpose, as very little of this was new or different for me. I still had a good experience taking this course, though. And I have learned from it, though for me it is less to do with real development or the software engineering process, and instead mostly to do with communicating with others in specific social settings.

Thinh Le

For the majority of us, this is the first time we get to develop an iOS app, so we encountered a lot of problems along the way. For me, the biggest issue I faced was to learn to manage view states (things like Property Wrappers). Another major issue was to make a view have the look and feel of an iOS app and not a web app. Fortunately, we had 2 experienced iOS developers in our team. I received a lot of help as well as shared experience from them with regard to the technical issue.

As for nontechnical issues, I have learned a lot from Venky and other team members. We talked about how to refer to someone in our group without giving a specific name. We discussed work experience (since I have never had any internship before, I asked them to share their experience), etc.

I would say what I have learned in ECS3390 has helped me a lot in this course. Specifically, I learned about how to create posters and reports and final reports as well as presentations.

I think this course is perfect and doesn't have to be improved in any way. One thing I really much like about this course is that we are allowed a lot of time to work on the project in the second half of the semester. I imagine it would be quite difficult to simultaneously attend lectures weekly and work on the project.

Future Work

The MyKitchen app was originally conceived by team member Elliott Gorman and further discussed with additional members of the team upon their addition to the group. Given that this project is entrepreneurial by nature, we've had plans to take development beyond the scope of this semester since the formation of the team. Given these plans for the future and the inevitable intent to commercialize the MyKitchen app, we will not be providing the source code for the app in the interest of protecting the viability of that commercialization in the future. For any questions regarding this decision, please contact the owner of the repository, Elliott Gorman (elliott.gorman@utdallas.edu), for details.

Given our intent to take this project beyond the scope of the semester, there are an unknown number of features planned, so this section cannot be given a complete

response. Overall, our intention with any addition to this app is to improve the usability and user experience of the app by allowing users to interact with it as efficiently as possible, as we understand that the true marketability of this app as a solution to the food management issue requires that it be as efficient and effective as possible.

A few of the features that are currently planned or being discussed as useful additions to the MyKitchen app:

- Multi-user access to a shared pantry through the implementation of an Account system
 - Expanding this feature to allow pantry “snapshots”, or a viewable state of a pantry that one could send to a friend so they would know what to buy for a dinner party
 - Addition of a transaction log
- Support for voice assistants such as Siri or Google, to allow hands-free simple commands to add something to a shopping list or remove some quantity of an item from your pantry without needing to open the app
- Support for kitchen utensils and hardware (pots, pans, knives) to be stored, shopped for, and used in procedures for recipes
- Implementation of scanning features to more efficiently add items to a pantry:
 - Scanning of nutrition labels for more detailed info or as a simple alternative method to populate your pantry
 - Scanning of receipts to bulk-add items after a shopping visit, saving time compared to individually adding each item
 - Importing grocery lists from various grocery stores to bulk-add items
- Integrating and adding support for various grocery APIs to allow users to export a shopping list to an online shopping cart for a grocery store, or vice-versa

Ethics Discussion

The methods by which this project was planned, managed, and completed all consist of responsible practices employed by reasonable people. Content written in various documentation was all sourced from ourselves, and was formatted occasionally using examples or templates as a foundation. All code was written from scratch: those in the team that did not previously know SwiftUI used the expertise of those that did as well as online resources to first build demonstrations, knowledge of which was then used to help structure the app itself. Many features that we considered for development, both in the scope of this semester and in our future work, would be trivial implementations using implementations found online, but were postponed deliberately until we had the time and



skill to implement them ourselves in the interest of the reliability of our code and the integrity of our project.

User security is the only other section where ethics would be reasonably discussed: the methods by which iOS handles security as outlined in the “Impact and Security” section, as well as the fact that we do not have any form of data collection or online component of our app means that security and user privacy are thus handled ethically and need not be discussed at this time. Going forward, however, we do have plans to include online components and thus an ethical discussion could be brought forth. To that end, we will be ensuring that all user data is handled securely and according to all laws and IEEE standards.

Appendix

Useful Sources

The following is a list of sources that may be helpful references when working on MyKitchen.

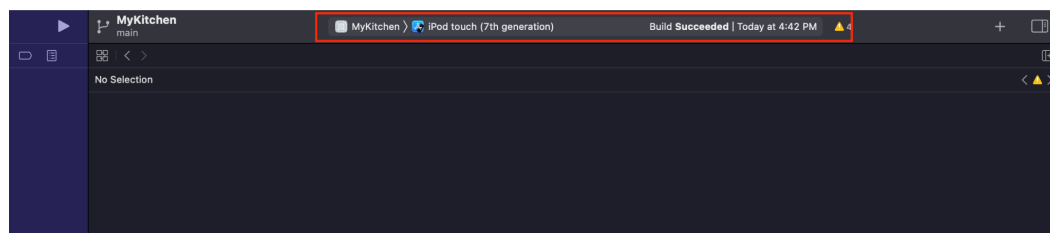
1. Apple Swift documentation: <https://developer.apple.com/documentation/swift>
2. Apple SwiftUI documentation: <https://developer.apple.com/documentation/SwiftUI>
3. Apple UIKit documentation: <https://developer.apple.com/tutorials/app-dev-training/#uikit-essentials>
4. Xcode debugging: <https://developer.apple.com/documentation/xcode>

Building, Running, and Testing MyKitchen

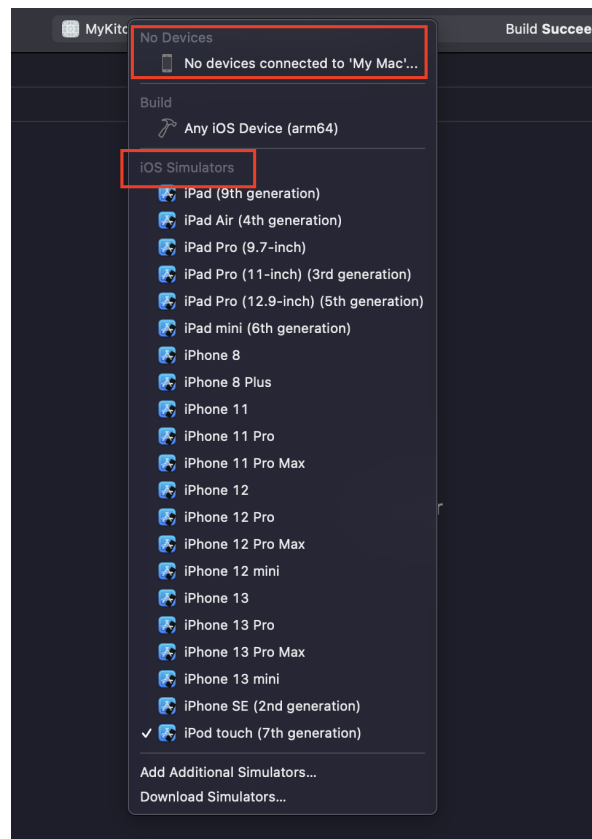
MyKitchen has been written in Xcode 13.2.1 using SwiftUI. The Git repository in Azure DevOps contains the source code for MyKitchen, as well as the Xcode project necessary to build, test, and run MyKitchen. To run the project, simply clone the repository and open the MyKitchen Xcode project, "MyKitchen.xcodeproj", with Xcode.

To build MyKitchen, you need either the iPhone simulator included with Xcode or an iPhone with at least iOS version 15. In most cases you will run the project on the simulator. To build and run the project, the following steps are provided.

1. At the top of the Xcode IDE is a bar that displays how the project will be built. By default, Xcode will use a simulator and the device to simulate is selectable by clicking on this bar, as shown in the image below.

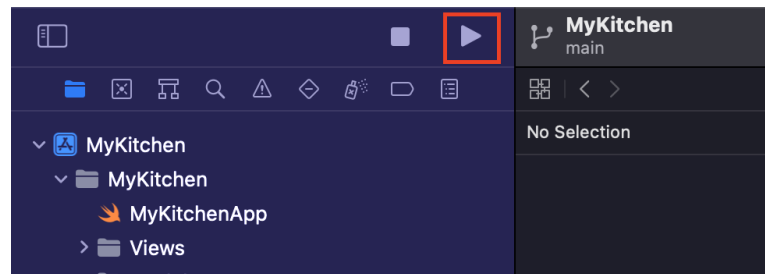


- a.
2. Click on the drop-down and select the appropriate device. At the top of this menu you will have the option to select a physical iPhone should one be connected to your device running Xcode. Select your preferred method of running the project.



a.

3. After choosing your preferred method to run the project, click the run button at the top of the Xcode IDE, or by clicking "Product → Run".



a.

To test and debug the MyKitchen project, it is suggested to review Apple's documentation for debugging Xcode projects: <https://developer.apple.com/documentation/xcode>.

Signatures Page



Elliott Gorman



Rowenna Switzer



Garrett Griffiths



Thinh Le



Krish Suchdev



Venkata Vadlamani
(Project Sponsor)



Jey Veerasamy
(Supervisor)

DATE: 5/6/2022