



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 intends to bring your kitchen to the palm of your hand through an iOS device. As people who often forget, we recognize the need for services such as: seeing what's in your pantry at any time, anywhere; the ability to ask your voice assistant "Hey Siri, how much oregano do I have left in my pantry?"; the ability to see exactly what you have and are missing for a cooking recipe; and more. The following list is a set of features of which we intend MyKitchen app to have by the end of this project.

- See what's in your pantry at any time, anywhere through the use of a database that stores your food and recipes.
- See a list of ingredients you are missing for a given recipe, as well as the ingredients you do have, including the amount.
- Create smart shopping lists based on recipes you like or ingredients you often purchase.
- Connect to your smart assistants for purposes such as "Hey Siri, remove two apples from my pantry."
- Allow multiple users to manage your pantry, and a log that stores transactions from your pantry.
- Store your recipes in-app, to be available at any time. Additionally, you can view your recipes based on what you can make and what ingredients you are missing.
- Quickly log items into your pantry on MyKitchen by scanning the nutrition label, the barcode, or by manually entering the item.

Table Of Contents

Overview	1
Table Of Contents	2
Project Deliverables	2
Schedule / Timeline / Scope	3
Communication Plan	4
Tracking Plan	5
Meeting Notes	6
Risk Analysis	7
Risk Mitigation Plans	8
Project Assumptions	10
Performance Metrics	11
Technical Process Plan Section	13
Process Model	13
Technical Details - Tools and Techniques	14
iOS - Mobile	14
Approach 1: UIKit using Swift	14
Approach 2: SwiftUI	14
Approach 3: Mixing UIKit and SwiftUI	15
Database	15
Approach 1: Homebrewed Database	15
Approach 2: Core Data	15
Approach 3: Objects in UserDefaults	15
Approach 4: Realm	15
Approach 5: Firebase	16
Words on Realm and Firebase:	16
Signatures Page	17

Project Deliverables

The following is a list of all project deliverables.

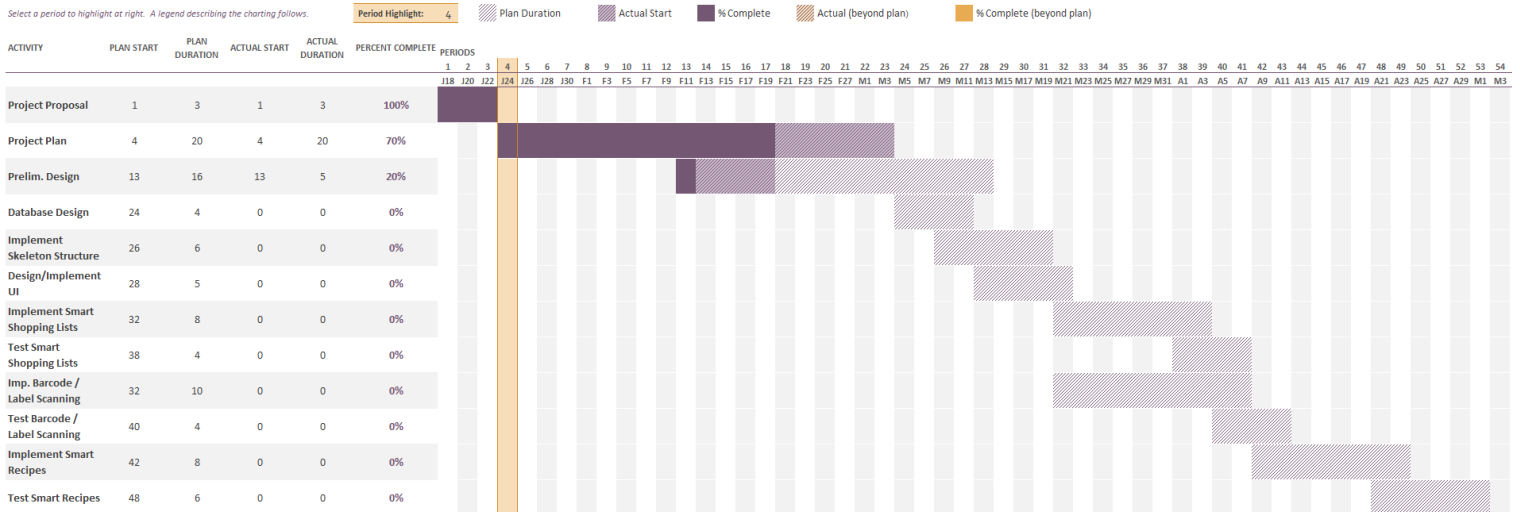
1. Deliver an app that allows users to track the items in their pantry, and use this information to provide various features described throughout this document.

- 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 will be used to track overall progress and will help identify project progress continuously.

MyKitchen Project Timeline



Task	Owner
Project Proposal (Internal)	Everyone
Project Plan	Everyone
Database Design	TBD
Implement Skeleton Structure	TBD
Design / Implement UI/UX	TBD
UI Review/Evaluation (Testing)	TBD
Implement Smart Shopping Lists	TBD

Test Smart Shopping List Feature	TBD
Implement Barcode / Label Scanning for entry into Database	TBD
Test Barcode Scanning Feature	TBD
Implement Smart Recipes	TBD
Test Smart Recipes Feature	TBD

Communication Plan

The following is the communication plan for our team. It covers type of communication, communication schedule, communication mechanism, who initiates, and the recipient.

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 will use Microsoft DevOps which includes Azure Boards and Azure Repos for individual and team work management, and version control. Specifically, we will utilize Azure Boards, a digital variation of the Kanban board, to track the progress of tasks in our project. As for coding, we will use Azure Repos (similar to 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. Given that development has not yet started and most of the team has little experience with iOS development, this is simply a rough estimate of what we would like to attempt to achieve with the project and will likely outlive the scope of this course. 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
1	Project Proposal (Internal)	100%	Feb 4th (Completed Feb 4th)
	Project Plan	95%	Feb 28th
	Backend Database Framework for Data Storage	0%	Mar 18th
	Implement Skeleton Structure	0%	Mar 25th
2	Design / Implement UI/UX	0%	Apr 1st* (And Ongoing)
	UI Review/Evaluation (Testing)	0%	Apr 1st - * (Ongoing)
	Implement Smart Shopping Lists	0%	Apr 8th
	Test Smart Shopping List Feature	0%	Apr 15th

	Implement Barcode / Label Scanning for entry into Database	0%	Apr 22nd
	Test Barcode Scanning Feature	0%	Apr 29th
3	Implement Smart Recipes ¹	0%	Apr 22nd
	Test Smart Recipes Feature	0%	Apr 29th
	Allow Multiple Users to Manage The Same Pantry	0%	TBD
	Recognize and Lookup Expiration Dates for Scanned Items	0%	TBD
	Final Report	0%	May 2nd

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

Meeting Notes

The following is a list of past meetings that have been scheduled. These details include what was discussed, and the next steps that are to be taken for the project. All meetings have and will continue to take place online. Meetings with our industry sponsor, Venky, will take place on MS Teams and team meetings with just the members will take place in Discord.

Meeting 1 - 4:00PM to 5:00PM (Friday, Jan 28th)

- Kickoff meeting with Venky and Dr. Veerasamy
 - Explained goal and scope of MyKitchen project
 - Discussed logistics of creating project documents with Venky and Dr. Veerasamy
 - Established rough timeline for planning and development phases as well as began investigating development tools to streamline product creation

Meeting 2 - 2:15PM to 3:30PM (Sunday, Jan 30th)

- Team meeting
 - Explored logistics of the project and discussed in varying degrees of depth the different features plans we would like to achieve for this project.

Meeting 3 - 6:00PM to 7:00PM (Thursday, Feb 3rd)

- Team meeting
 - Discussed internally what we would individually like to grasp from the fruition of this project to better align the scope with realistic goals of each member

- Drafted several outlines of a Project Plan and discussed logistics for combining them.

Meeting 4 - 12:30PM to 1:30PM (Friday, Feb 4th)

- Team meeting
 - Completed the project proposal.
 - Began transitioning project proposal content to project plan, discussing various additions that would be necessary to create a proper draft of said project plan

Meeting 5 - 7:00PM to 7:45PM (Monday, Feb 14th)

- Team meeting
 - Work on the ethics and project management assignments.
 - Work on the project plan.

Meeting 6 - 12:30pm to 1:30pm (Friday, Feb 18th)

- Team meeting
 - Completed the "Ethics Quiz" assignment and established remaining items necessary to complete for "Software Management" document, which were clarified in an email to Dr. Razo

Meeting 7 - 12:30pm to 1:30pm (Friday, Feb 25th)

- Team meeting
 - Reported the project progress.
 - Received feedback on the items in the project plan.
 - Delegated work for the next week.

Going forward, sponsor meetings will take place on Friday at 12:30PM via MS Teams. In addition to the sponsor meetings, the group has meetings via Discord every Wednesday at 7:00PM and every Friday following the sponsor meeting. Additional meetings take place as deemed necessary by a member of the group and according to reasonable availability of the other members.

Risk Analysis

Table 1 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 will be 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


Table 1. Risks associated with the myKitchen project.

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 Azure DevOps using the “Bug” and “Issue” work items. These work items will allow us to track impact, priority, hours spent and time to completion, as well as other metrics useful for tracking and handling bugs/issues. When a risk or software issue arises, a team member will create a bug or issue a work item to track the issue. 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. A task in Azure DevOps will be created to report statuses. 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 detail some expectations about either a tool or person, as per the nature of this entrepreneurial project. 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 pose 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 thoughts and much time has been spent on engineering the functional and nonfunctional requirements, so we expect that there will be not 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 have identified the following measures:

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 will aid us in ensuring project success and completion. In order to measure each metric, we have 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.

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

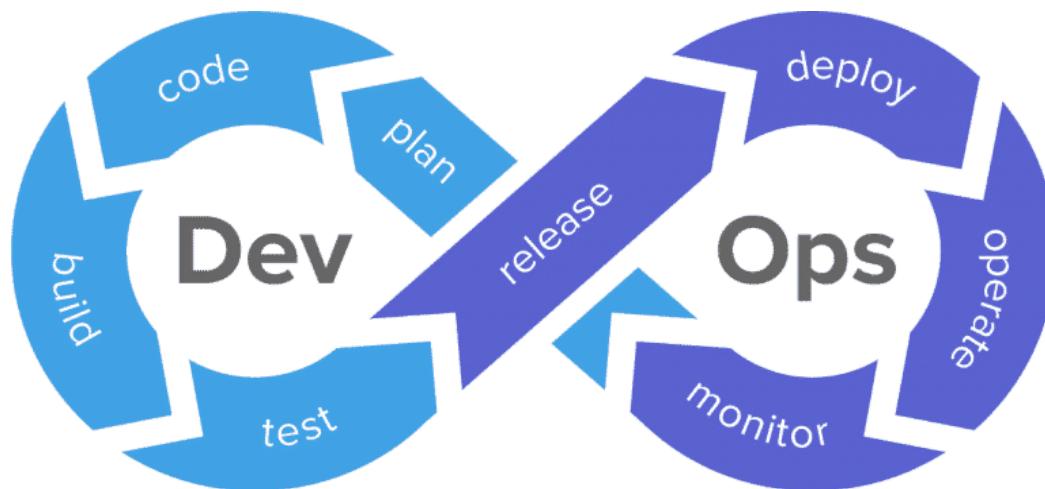
Metric 4: At each major feature testing release, we will create a testing environment with which to give users a platform to review the new featureset. Giving them categories such as "Ease of use", "Performance", "User Satisfaction", or other relevant categories to rate on a 1-5 scale, we can take their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

It is likely these metrics will change overtime as the project develops. To combat this, we will revisit this document throughout the project's lifetime to update this section. This way, we can always include the most relevant information at the time to ensure we have metrics that are helpful to the project at any given time.

Technical Process Plan Section

We will 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.

Process Model



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

For the development process model, we will use DevOps where everyone will be responsible for implementing and testing the feature to which they are 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.

Technical Details - Tools and Techniques

iOS - Mobile

The native iOS mobile app will be built using the iOS SDK running on the latest version of Xcode.

Approach 1: UIKit using Swift

This approach will utilize the long supported framework of **UIKit** and the Swift programming language. UIKit is the more advanced framework with deeper control and more customization options. Given the time frame of the project, and the great complexity offered by UIKit, this framework could present a somewhat steep initial learning curve for team members unfamiliar with iOS development. There are many complex components that may be necessary to set up right when starting a new app, such as `UIViewController` or `UISegmentedControl`. Simple primitives like these require a good understanding of the Model-View-Controller paradigm and for the developer to implement multiple methods right at start up.

Official Apple Tutorial of UIKit:

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

Approach 2: SwiftUI

SwiftUI is newer, simpler, and often easier for new developers to pick. However, due to those reasons, it is also more limited in terms of functionality. It is also a declarative (and usually reactive) language. Therefore, using this approach extensively would require 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. 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.

Official Apple Tutorial of SwiftUI:

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

Approach 3: Mixing UIKit and SwiftUI

It is possible to mix the use of **SwiftUI and UIKit** in the iOS SDK. UIKit UIViews can be embedded in SwiftUI Views and vice versa. Therefore we could make use of the SwiftUI framework to handle simpler UI archetypes like lists, while embedding UIKit UIViews into the app to handle more complex operations that interact with our object store (database). Regardless of which approach we start the app with, the mixed approach could be deployed to varying extents as-necessary.

Database

Approach 1: Homebrewed Database

It is possible to embed and manage a **mySQL database** in a UIKit app (this is made unreasonably difficult if SwiftUI is involved). This is the least supported by the iOS SDK among these approaches.

Approach 2: Core Data

Core Data is an object-oriented relational database framework offered by Apple for use in UIKit apps. One defines their objects in Core Data's proprietary format in addition to where the Swift classes are defined. There is a steep learning curve, but once set up, it can be treated as a fairly simple object store, even for fairly complicated relational databases.

Approach 3: Objects in UserDefaults


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.

Approach 4: Realm

Realm is an object-oriented MongoDB database that is directly defined via bindings on the Objective-C runtime by the Swift objects that will be stored in it. It has similar benefits (once



up and running) to Core Data, without the harsh initial learning curve. Like Core Data, it can handle more complicated relations and larger databases than would be a good idea to store in Defaults. Realm works well when fully local to the device, but can be hosted with a server component as well with no changes to existing code. It works with Java and Kotlin on Android as well as many web frameworks, so it would be easy to connect to a future web app and/or Android version.

Approach 5: Firebase

The **Firebase** platform is not just a database, but a broader platform geared towards mobile and web applications. It has support for easy integration with Google services, which could be advantageous to Phase 3 development. It offers a noSQL database (like Realm) with a cross-platform SDK that works with many frameworks and languages, both locally and remotely, so it would be easier to connect to a future web app and Android version without making any changes to the iOS code.

Unlike Realm, it does not provide bindings to Swift objects that can be used throughout development of the iOS app.

Firebase also supports authentication features, which could make it easier to support multi-user support for managing a single pantry. However, this would be relevant only in Phase 3 of the project, and the increased complication of starting with this approach may not be worth it in the early phases of development.

Words on Realm and Firebase:

One drawback to both of the cross-platform database solutions is that to start using them we must first understand and agree to their pricing structure in addition to setting up their SDKs in our app. They may remain free for the duration of development, but if our business grows quickly, we would rapidly find ourselves on the hook for ever-increasing SaaS fees.

Signatures Page



Elliott Gorman



Rowenna Switzer



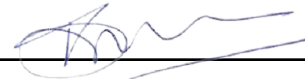
Garrett Griffiths



Thinh Le



Krish Suchdev



Venkata Vadlamani
(Project Sponsor)



Jey Veerasamy
(Supervisor)

DATE: 3/4/2022