**Data Intelligence Low Code Inventory App for Personal and Corporate Settings**

**WasteDrop**

**Part 2 – Introduction and Design**

**Dr. Walker**

**Advanced Software Engineering**

**September 14, 2023**



Table of Contents

[Section 1 – Salient Characteristics 6](#_Toc149511469)

[Section 2 – Description of Salient Characteristics 6](#_Toc149511470)

[2.1 6](#_Toc149511471)

[2.2 7](#_Toc149511472)

[2.3 7](#_Toc149511473)

[Section 3 – Contextual Issues and Constraints 8](#_Toc149511474)

[3.1 8](#_Toc149511475)

[Section 4 – Proposed Schedule 9](#_Toc149511476)

[4.1.1 & 4.1.2 9](#_Toc149511477)

[(US001) Sign-up functionality 9](#_Toc149511478)

[(US002) Login Functionality 9](#_Toc149511479)

[(US003) Binz Creation 10](#_Toc149511480)

[(US004) Add Items to Binz Functionality 10](#_Toc149511481)

[(US005) “Consume” Binz Items Functionality 10](#_Toc149511482)

[(US006) UI enhancements 11](#_Toc149511483)

[(US007) Add Expiry Alerts 11](#_Toc149511484)

[(US008) User Personalized Alert Times 12](#_Toc149511485)

[(US009) Top Binz Items List 12](#_Toc149511486)

[(US010) Low Code User Friendly Query 13](#_Toc149511487)

[(US011) Logging Out 13](#_Toc149511488)

[(US012) Session State Handling 14](#_Toc149511489)

[(US0013) Create DB schema Design 14](#_Toc149511490)

[(US0014) Page-Switching Functionality 15](#_Toc149511491)

[(US015) “Use one” functionality for items 15](#_Toc149511492)

[(US016) “Use many” functionality for items 16](#_Toc149511493)

[(US017) Historical list 16](#_Toc149511494)

[(US018) Pre final release code tests 16](#_Toc149511495)

[(US019) Password Hashing 17](#_Toc149511496)

[(US020) Delete Binz 17](#_Toc149511497)

[4.2 – User Story Allocation 18](#_Toc149511498)

[4.3 – Sprint Period 18](#_Toc149511499)

[4.4 – Sprint Backlog Allocation 18](#_Toc149511500)

[4.5 – Burndown Chart 19](#_Toc149511501)

[Section 5 – Team & Roles 20](#_Toc149511502)

[Team Members 20](#_Toc149511503)

[Samuel Muvdi 20](#_Toc149511504)

[Ed 20](#_Toc149511505)

[Abel 20](#_Toc149511506)

[David Rice 21](#_Toc149511507)

[Ryan 21](#_Toc149511508)

[Section 6 – Project Design 22](#_Toc149511509)

[6.1 – Design Defined 22](#_Toc149511510)

[6.2 – Team Design Process 23](#_Toc149511511)

[6.2 – Graphical Design of WasteDrop 24](#_Toc149511512)

[Use Case Diagram 24](#_Toc149511513)

[Class Diagrams 25](#_Toc149511514)

[Sequence Diagram 26](#_Toc149511515)

[Section 7 – Rationale Management 27](#_Toc149511516)

[7.1 – Design Framework 27](#_Toc149511517)

[7.1.1 – Issue Defined 27](#_Toc149511518)

[7.1.2 – Guiding Criteria 28](#_Toc149511519)

[7.1.3 – Design Decision 28](#_Toc149511520)

[7.1.4 – Alternatives Considered 29](#_Toc149511521)

[7.2 – User Interface 30](#_Toc149511522)

[7.2.1 – Issue Defined 30](#_Toc149511523)

[7.2.2 – Guiding Criteria 30](#_Toc149511524)

[7.2.3 – Design Decision 31](#_Toc149511525)

[7.2.4 – Alternatives Considered 31](#_Toc149511526)

[Section 8 – Sources 32](#_Toc149511527)

Section 1 – Salient Characteristics

The salient characteristics of the customers will be individuals or companies looking to keep an organized and smart food inventory collection on their smart phones and computers where they can track all items from pantry items to freezer to fridge foods. This allows users to reduce food costs as well as have a positive environmental and social impact by reducing food waste which otherwise could have been donated or eaten before its expiration date. Our users will range from family households trying to manage multiple buyers putting in and taking out things from the fridge all the time to office managers who need to order large quantities of food for office events or just for their employees to restaurants, bakeries, catering companies, cafeterias, to food court venues and food trucks that just want to have a modern, visual, smart, and simple way of managing their food inventories without having to use the traditional paper and pencil.

# Section 2 – Description of Salient Characteristics

2.1

The background of WasteDrop stems from food waste, a common issue that is faced in many countries and areas around the world. According to the Food and Agriculture Organization of the United Nations, approximately one-third of all food produced for human consumption is lost or wasted (citation needed?). This significant waste can be partially attributed to consumers grappling with the management of expiration dates, as well as a lack of accessible information regarding optimal methods for storing and utilizing perishable items.

## 2.2

WasteDrop will stand out from other apps and tools in the market because we will be marketing and catering primarily to food inventories whether it’s dry, wet, or frozen food. Second, we will deploy our project online through the web and eventually as an app that can be downloaded and saved. Third, we will be using low-code user intelligence queries which will let the users get insights on their inventory without even having to know how to code. Finally, our app will implement sending notifications whenever food is expiring soon or otherwise according to the users’ preferences and settings.

## 2.3

WasteDrop is designed with the primary goal of substantially reducing food waste by assisting individuals in effectively handling their perishable items. Additionally, the application can act as a valuable tool for cost-conscious consumers, helping them cut unnecessary expenses. By offering essential insights, reminders concerning expiration dates, practical tips on food storage and usage, and streamlined inventory management, WasteDrop promotes cultural awareness and education, encouraging responsible food consumption practices. In this way, WasteDrop holds the potential to bring about a significant positive impact on both consumer behavior and the environmental repercussions linked to food waste.

# Section 3 – Contextual Issues and Constraints

## 3.1

Depending on user-provided input, printed expiration dates may pose potential issues. To enhance the application's efficacy, it may be necessary to establish a system for validating or cross-referencing expiration dates specific to various products. Additionally, compliance with data protection laws is a crucial external factor. Users should have confidence that their personal information and food inventory details are handled securely.

# Section 4 – Proposed Schedule

## 4.1.1 & 4.1.2

### (US001) Sign-up functionality

**Description**

As a user, I want to be able to sign up for an account on WasteDrop so that I may login and use the app.

**Criteria**

Users will be able to create accounts on WasteDrop, that will contain their first name, last name, email address, and a hashed password. This sign-up functionality will allow users to have accounts which they can then use to access WasteDrop.

### (US002) Login Functionality

**Description**

As a user, I want to be able to login to the account I created on WasteDrop and access their and only their accounts, so that I can access my own account

**Criteria**

Users will be able to login to the accounts they made on WasteDrop and retrieve their data.

### (US003) Binz Creation

**Description**

As a user, I want to be able to create a Binz (unit of storage for inventory, ex: fridge\_house, fridge\_garage, cabinet\_1) so that I can access my items.

**Criteria**

A user can create a Binz that is saved to their account, and they can access it freely.

### (US004) Add Items to Binz Functionality

**Description**

As a user, I want to be able to add items to my Binz so that I can track my inventory.

**Criteria**

Binz will have a feature where the user can add items with their information.

### (US005) “Consume” Binz Items Functionality

**Description**

As a user, I want to be able to “Consume” (delete from my binz) items in my Binz, so that items that no longer are in my real-life inventory can be removed from my Binz.

**Criteria**

Ability to take things off from Binz inventory if they were eaten or thrown away etc.…

### (US006) UI enhancements

**Description**

As a UI developer, I want users to be able to have a simple yet modern look for WasteDrop and be able to enjoy how the app looks so they are more inclined to continue using it.

**Criteria**

The app is beautified the most it can be so that users see it as more likeable and pretty.

### (US007) Add Expiry Alerts

**Description**

As a user, I want to be able to be alerted any time my Binz items are about to expire so that I may take proper action.

**Criteria**

WasteDrop lets users know when their items are about to expire (users add expiry dates using the sell by and best by dates on food, if for some reason the sell by or expiry dates are not accurate on the food, maybe it’s just the date the food should be sold by and not necessarily expired, the user should choose a given date to be the expiry date)

### (US008) User Personalized Alert Times

**Description**

As a user, I want to pick how many days before an item expires to be alerted that the item is expiring so that I can have enough time to take proper action.

**Criteria**

The user can pick how long before an item is about to expire to be notified that an item is expiring soon.

### (US009) Top Binz Items List

**Description**

As a user, I want my most added items to be added to a short list so that I may be able to see how many times I have added it and which items are most frequent in my Binz.

**Criteria**

Users have a top items list where they can see what items they added the most and how many times they have added it.

### (US010) Low Code User Friendly Query

**Description**

As a user, I want to be able to create my own searches (queries) on my Binz to get information on my items, like which items expire tomorrow, or how many items I have currently in my Binz, so that users who aren’t experienced in data analytics can still be able to search their Binz and find insights on it.

**Criteria**

Users will have the option to make no code / low code queries using drop down lists to get insights on their Binz data.

### (US011) Logging Out

**Description**

As a user, I want to be able to sign out of my account so that, if someone were to have access to the same device as me, they would not be able to access the information on my account without my login credentials.

**Criteria**

Users will have the option to log out of their account when they are finished using the app, closing their session and ensuring security of their account information.

### (US012) Session State Handling

**Description**

As a user, I do not want to have to repeatedly enter the same information to continue accessing my account. I want to have indefinite access to my account until if/when I choose to log out.

**Criteria**

Users’ account data will be stored while they use browse the application by maintaining the data on the server-side to persist across many requests from a particular client. Once the user closes their session, this temporary data is deleted.

### (US0013) Create DB schema Design

**Description**

As a developer, I want to be able to design a functional and normalized database schema so that the users can have a smooth experience when querying or inserting.

**Acceptance criteria**

A working and normalized database schema design is created.

### (US0014) Page-Switching Functionality

**Description**

As a user, I want to be able to switch pages whenever I log in so that I may access my items.

**Acceptance** **criteria**:

When users log in or need to access items, the switch page functionality works correctly.

### (US015) “Use one” functionality for items

**Description**

As a user, I want to be able to reduce my inventory quantity by 1 for a specific item each time by a button so that I may track updated quantity.

**Acceptance criteria**

A button will allow for the reduction of 1 for inventory quantity for item.

### (US016) “Use many” functionality for items

**Description**

As a user, I want to be able to reduce the count of my inventory quantity by a specified amount so that I may track my consumption and updated inventory.

**Acceptance criteria**

Users will be able to reduce inventory count quantity of an item by more than just 1

### (US017) Historical list

**Description**

As a user, I want to be able to see all items I ever added so that I may keep track of past inventory for analysis.

**Acceptance criteria**

Trigger moves consumed items to historical items table.

### (US018) Pre final release code tests

**Description**

As a developer, I want to do full app testing to pick up any bugs before the final release so that users wont experience any last-minute issues.

**Acceptance criteria**

No final bugs are found in last check before final code deployment on release.

### (US019) Password Hashing

**Description**

As a developer, I want to hash passwords before they are inserted into the database so that users’ data is protected.

**Acceptance Criteria**

User passwords are hashed before inserted to the database.

### (US020) Delete Binz

**Description**

As a user, I want to be able to delete Binz when I no longer need them.

**Acceptance Criteria**

Users have capability to delete a Binz which removes it from the database and their profile.

## 4.2 – User Story Allocation

Team developers will have their user stories assigned to them based on their past coding experiences, developing experiences, and what they have been able to learn through research this semester. Each developer will be assigned a reasonable amount of stories each sprint to ensure that the project moves smoothly and on time.

In the case that the project needs a new story for more functionalities or additions, a new story will be created for it and then will be added to the next sprint or started in the current sprint and if not completed, rolled over to the next sprint to complete. This will allow for all user stories to be tracked and the ones that are higher priority to be done as soon as possible.

## 4.3 – Sprint Period

Each sprint will last for 2 weeks. Each sprint will start on Wednesdays and end on Wednesdays. Sprints will be tracked via Trello.

## 4.4 – Sprint Backlog Allocation

The allocation of User Backlogs to sprints will be as follows:

* Sprint 1 9/13-9/27: (US001), (US002), (US011), (US013), (US014), (US019)
* Sprint 2 9/27-10/11: (US003), (US004), (US005), (US006), (US007)
* Sprint 3 10/11-10/25: (US008), (US009)
* Sprint 4 10/25-11/8: (US010), (US012), (US015)
* Sprint 5 11/8-11/22 (US016), (US017)
* Sprint 6 11/22-12/6 (US018)

## 4.5 – Burndown Chart

The burndown chart for the schedule that was outline in section 4.4 is shown below:

# Section 5 – Team & Roles

**Team #7 WasteDrop Developers**

## Team Members

Samuel Muvdi

* Lead developer
* BA
* Project Manager
* Oversee and develop code on full stack.
* Ensure data and Python code and scripts are working as expected.
* Create new user stories & approve tentative stories.

Ed

* Developer / Backend Data-Ops
* Assist with creation and testing of Python scripts and Postgres implementation.
* Ensure Python scripts are handling data as expected.

Abel

* Developer
* Assist with python coding to ensure functionalities are being properly created, implemented, and documented.
* Additionally, assist with testing of various components of the application.

### David Rice

* Scrum Master
* Maintaining Trello board and overseeing documentation of each rollout.
* Documenting project progress and completion of deliverables.

### Ryan

* Developer
* Help with python coding and making sure functionalities are being added and tested.

# Section 6 – Project Design

## 6.1 – Design Defined

WasteDrop was designed with as much efficacy as possible. We kept ease of use in mind, when creating the application’s front end. As such, we decided to employ a Python framework, as it was the simplest manner to bring WasteDrop to life. With it, the design is as follows;

The developers set up WasteDrop with the user’s experience in mind. Before the application was created, the developers created a database schema with a system in charge of managing user interfaces; namely, the “binz” feature. A user signs up and creates an account on WasteDrop, managed by the database held by Supabase. A user created binz is in charge of cataloging a unit of storage. Individual items can be aggregated to this singular binz. The user can also add expiry dates, quantity of items, record items consumed in these binz, app personal alerts and view top items within these binz. The system keeps track of all these changes, as well as whenever the user logs out of the account. The system keeps track of all the timestamps.

In creating a multi-page application such as this, we did not use Flask, despite its flexibility and other advantages. A study finds that Flask does not support multi-page functionality (Bahgat, 2023). We needed multiple pages for WasteDrop, considering the user must sign up for an account, then manage their individual Binz on a separate page. ReactJS’s ability for front-end library framework came from trial and error with Ionic. According to a study by AltexSoft, React shows better results [than Ionic], as apps are compiled into native (2019).

## 6.2 – Team Design Process

The idea behind WasteDrop came to during a discussion of what the team project should be centered around. After some consideration of the limitations and possibilities that could come of a Python project, we came to an agreement that a project such as this could be deliverable. We decided that our efforts would be best put to use for a design meant to manage everyday happenings, notably those more likely for humans to come across, something within the realm of Python’s capabilities, namely, tables and queries. As such, cataloging the kitchen and reducing wasted food items came to be.

Initially, we opted to use Ionic due to its flexibility and one of our member’s experiences with it. However, due to WasteDrop’s development and its status as a web-based application intended for scalability with mobile development, we wanted to minimize plugins. We then created the interface design using React.js and Streamlit, and the back-end architecture with Supabase for databases, our React.js libraries defined within the code. From then on, the app was developed during sprints, developing the boilerplate form of the front-end, then the back-end was enhanced.

## 6.2 – Graphical Design of WasteDrop

### Use Case Diagram

A diagram of a person's network

Description automatically generated

### Class Diagrams

Figure 1

A diagram of a computer

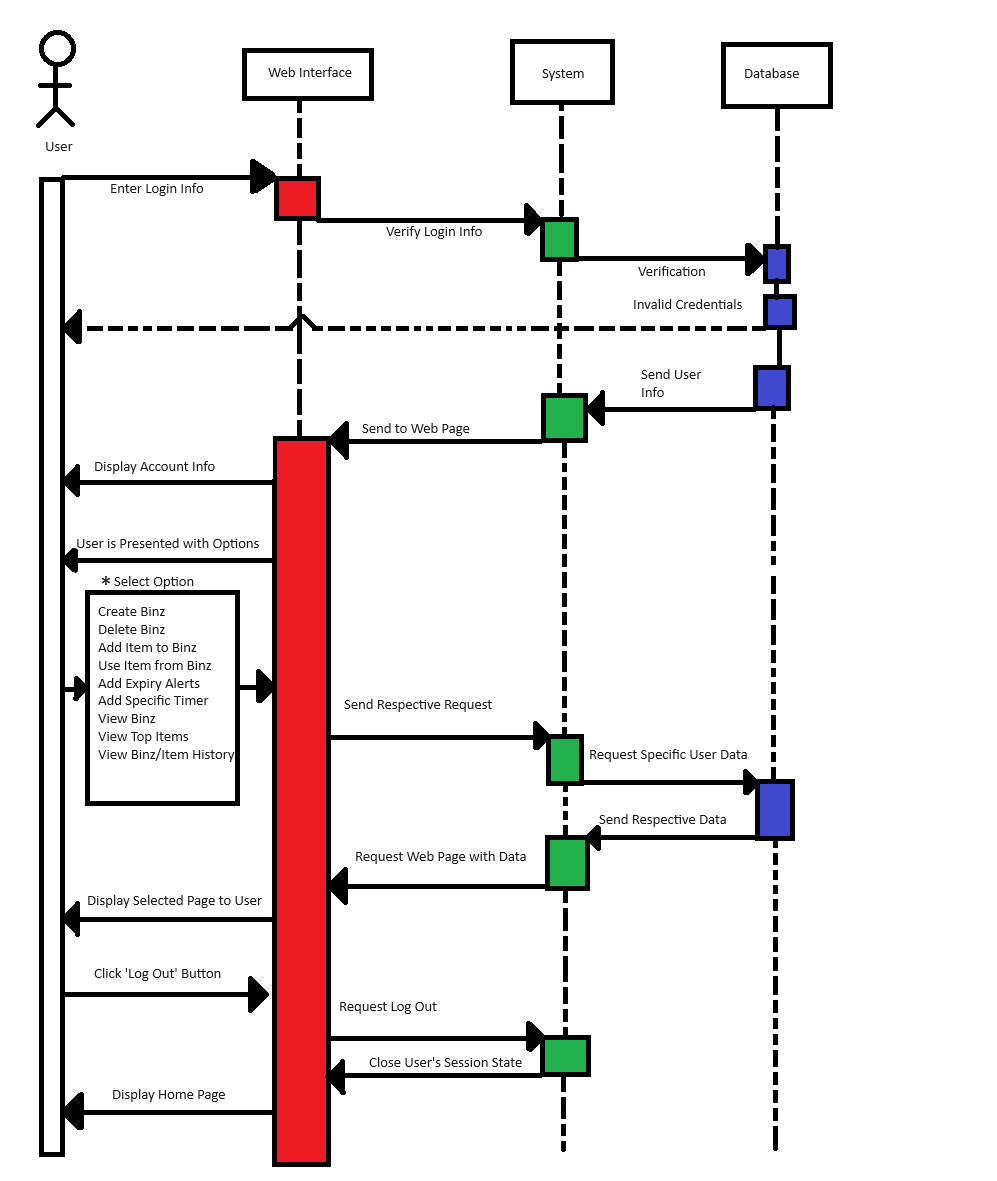
Description automatically generated with medium confidence

Figure 2

A diagram of a flowchart

Description automatically generated

### Sequence Diagram



\* For simplicity, this sequence diagram combines all options from the home page into one box, as they all follow the same generalized flow.

# Section 7 – Rationale Management

In this section, each major issue the team encountered will analyzed individually, providing context, the steps we took to find a solution, and the rationale for each situtation. Each individual issue will go in-depth on its specific definition, the guiding criteria used to determine what steps needed to be taken, the design decisions made that led to the solution, and any alternatives that were considered during the process. Additionally, discourse that occurred among the developers regarding each part of the design process, and each developer’s individual opinion(s), will be included.

## 7.1 – Design Framework

### 7.1.1 – Issue Defined

Throughout its lifetime, WasteDrop has encountered several issues regarding its design. At the project’s onset, when discussing what style of tech stack and the specific tools and technologies we would be using, the team had decided on using NodeJS, a popular open-source, asynchronous, event-driven JavaScript runtime used for developing web applications. In tandem with JavaScript, we had also decided to utilize ReactJS as a front-end library framework for the application’s development.

However, early development and testing revealed that support for these frameworks was not sufficient for our development needs, and the early versions of WasteDrop had to be scrapped, as there was no effective way to continue working with the current platforms. Ed and Samuel have experience working in multiple frameworks, and therefore discussed among the group what they had worked with, and what would be the best for achieving WasteDrop’s desired goals and overall purpose.

### 7.1.2 – Guiding Criteria

When discussing what framework to use for the project, the team considered multiple criteria: A simple yet powerful framework that provided easy and streamlined work was a priority, as not all members of the team had sufficient exposure web application development prior to the project. Having an easy-to-develop framework ensured that the learning curve was gentle, and that meaningful development would take place at a faster and more efficiently.

Continuing along the line of ease-of-use, the team also wanted to select a framework that was coded using a language that the entire team was familiar with. After some deliberation, we found that each member of the team had at least some experience with Python, so pinpointing a Python framework became our best target since it would provide an environment in which all members could contribute.

Furthermore, utilizing an open-source framework would allow the team to use the repertoire of online resources that exist for that framework. With many tools and other assets at our disposal, the issues that would likely be encountered during WasteDrop’s development would become significantly easier to manage and solve. Another huge benefit of using an open-source framework is that the team would not have to worry about monetarily budgeting the project’s development.

### 7.1.3 – Design Decision

With all this taken into consideration, the team selected a Python framework known as Streamlit. Streamlit is an open-source Python application framework built for data science web applications. The framework is targeted for data scientists with little web development experience, making it easy to learn and an ideal starting point for our team members with less experience.

Heavy discussion came up within the team because switching to a completely different framework meant that the overwhelming majority of our current work at the time would be scrapped. The team considered the long-term development of the project, and what each framework would mean for the design and ongoing development of WasteDrop. For example, Ed suggested an alternative open-source framework, and had described its many positive aspects and benefits. However, this particular framework did not utilize languages that the entire team knew well and were comfortable using. After it had been fully discussed among the team, the final decision was made to work with Streamlit as our web app development framework.

### 7.1.4 – Alternatives Considered

One of the strong contenders for a framework that was proposed by Ed was Ionic. Ionic is an open-source web software development kit (SDK) that is built upon the Angular framework and Apache Cordova. It is meant for our use case, in that it is built upon the idea of web development for those that are less experienced (Altexsoft, 2019). However, this was not chosen as better suited for mobile app development, and WasteDrop was to be a browser based web application.

Flask was another framework that was considered, as its main selling point was that it was easy to use and has an integrated unit testing system. This one was decided against as well, since it had no support for multi-page applications, and had no native object-relational mapping (Ahmed Bahgat, 2023).

## 7.2 – User Interface

### 7.2.1 – Issue Defined

Enterprise applications tend to be feature rich and offer a strong set of capabilities. This comes with the drawback that training to use these enterprise applications can take longer than a more user friendly, simplified application meant for the public. Conversely, applications developed specifically for personal use will have a very simple set of features that are easy to pick up but are lacking the depth that is sometimes needed for the individual’s use case.

This issue was a topic of debate in one of the meetings. Ryan and Abel were of opposing positions on the matter. Ryan argued that the application would make more sense for individual use, as the front-end simplicity of a personal application would be better suited for our application’s purpose. On the other hand, Abel believed that a more enterprise-focused application would allow us to expand greatly upon the feature set.

### 7.2.2 – Guiding Criteria

The decision for this design principle was made based on a few criteria: First, the team wanted to ensure that the project would remain in-scope for the target date towards the end of the semester. Going for too many features was projected to push our deadline beyond that of the allotted time we had, but going too simplistic would mean that the application would lack in its feature set, and fall short of our goals. Therefore, we needed to determine a balanced medium between the two.

The team also wanted to decide based on the overall interest in developing certain features. Samuel, David and Ryan preferred a more user-friendly interface, as it would prove a good challenge to make something that is user friendly while also trying not to restrict the users too heavily. Additionally, David noted that a user-friendly interface fosters efficient usage of the application, thus leading to a higher satisfaction rate among users and greater success overall.

### 7.2.3 – Design Decision

WasteDrop was then chosen to stand in the middle of these two design paradigms, making target audiences both enterprise use and individual use. This would allow our application to appeal to as many groups as possible, and it presented the team with unique challenges to learn from and grow. A careful balance of features vs UI polish would be kept in order to remain within scope of the project, and it satisfied the interests of everyone on the team. In this decision, we would ideally create an app that is simple enough to be suited for personal use, but also sufficiently powerful to be used in a commercial setting.

### 7.2.4 – Alternatives Considered

The alternatives in this situation were forgoing a simplistic user-interface for a more feature rich application, or having a trimmed down, slick application meant just for personal use. Both design paradigms offer positives and negatives, outlined by Ryan and Abel outlined in the sections above.

# Section 8 – Sources

1. (2019, May 21). *The Good and the Bad of Ionic Mobile Development*. AltexSoft. <https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-ionic-mobile-development/>
2. (2023, September 13). *Flask Vs Django* (A. Bahgat). Kinsta.com. <https://kinsta.com/blog/flask-vs-django/>