PROJECT Design Documentation

> *The following template provides the headings for your Design Documentation. As you edit each section make sure you remove these commentary 'blockquotes'; the lines that start with a > character and appear in the generated PDF in italics.*

Team Information

- Team name: Sam and Friends
- Team members
  - Ashton Michelstein
  - Christian Morgado
  - Sam Ruan
  - Laurena Roberts

Executive Summary

We are creating a platform to help those who have been affected by natural disasters. An admin, called a U-Fund manager, can add needs to a cupboard. This cupboard acts as storage for the needs. A helper can log into the website and pick a need from the cupboard that they want to fund. They can add this need to their funding basket, where they can add funds to the need.

Purpose

> ***[Sprint 2 & 4]*** *In Sprint 2, we are trying to deliver the Minimum Viable Product. This MVP will have the minimal required functionality for the website. The helper should be able to add needs to their funding basket, and search for needs. The U-fund manager should be able to add and delete needs in the cupboard. They should not be able to view the funding basket. In Sprint 4, we have delivered the final product, with full functionality. The U-fund manager can edit and remove the cupboard, and the helpers can select needs from the cupboard to add to the funding basket. We have also cleaned up the UI.*

Glossary and Acronyms

> ***[Sprint 2 & 4]*** *Provide a table of terms and acronyms.*

Term	Definition
MVP	Minimum Viable Product
Helper	user other than admin
U-fund Manager	admin
Funding Basket	A page where needs can be funded by the helper

Requirements

This section describes the features of the application.

> *In this section you do not need to be exhaustive and list every story. Focus on top-level features from the Vision document and maybe Epics and critical Stories.*

Definition of MVP

> ***[Sprint 2 & 4]*** *The Minimum Viable Product is not focused on how the website looks, but rather the functionality of the website. Helpers will be able to pick needs from the cupboard and add them to the funding basket. The U-fund manager will be able to add, delete, and edit needs in the cupboard. The manager will not have access to the funding basket. All users will be able to log in with a username.*

MVP Features

> ***[Sprint 4]*** *Log in / Log out, Create Helper, Add Needs to Funding Basket, Create Funding Basket.*

Enhancements

> ***[Sprint 4]*** *For our two enhancements, we implemented full admin control over the dashboard and a feature that counts the total funds a helper has donated in their current session.*

Application Domain

This section describes the application domain.

group-domain-model-1

> ***[Sprint 2 & 4]*** *The most important domain entities in this project are the Helper, Cupboard, Funding Basket, Needs, and U-Fund Manager. The U-Fund manager will have full control of the cupboard and the needs placed in the cupboard. A helper can only access the cupboard and pick needs they want from the cupboard. They will pick a need and add it to the funding basket. The U-Fund Manager will not be able to see the funding basket.*

Architecture and Design

This section describes the application architecture.

Summary

The following Tiers/Layers model shows a high-level view of the webapp's architecture. **NOTE**: detailed diagrams are required in later sections of this document. (*When requested, replace this diagram with your* ***own*** *rendition and representations of sample classes of your system.*)

architecture-tiers-layers-1

The web application, is built using the Model–View–ViewModel (MVVM) architecture pattern.

The Model stores the application data objects including any functionality to provide persistance.

The View is the client-side SPA built with Angular utilizing HTML, CSS and TypeScript. The ViewModel provides RESTful APIs to the client (View) as well as any logic required to manipulate the data objects from the Model.

Both the ViewModel and Model are built using Java and Spring Framework. Details of the components within these tiers are supplied below.

Overview of User Interface

This section describes the web interface flow; this is how the user views and interacts with the web application.

> *Upon launching the website, the user is brought to the log in page. From there, the user can enter a username and press the log in button, which will take them to the dashboard. The dashboard contains a list of the top needs, as well as a search bar to search for needs. On the top of the page is a navigation bar with buttons for the Dashboard and Cupboard. The User's funding basket is displayed next to the cupboard on the Cupboard page. The user can get to any page on the website from anywhere. There is a log out button on the top right of the page, which, when pressed, will log the user out and bring them back to the login page.*

View Tier

> ***[Sprint 4]*** *Provide a summary of the View Tier UI of your architecture. Describe the types of components in the tier and describe their responsibilities. This should be a narrative description, i.e. it has a flow or "story line" that the reader can follow.*

> ***[Sprint 4]*** *You must provide at least* ***2 sequence diagrams*** *as is relevant to a particular aspects of the design that you are describing. (****For example****, in a shopping experience application you might create a sequence diagram of a customer searching for an item and adding to their cart.) As these can span multiple tiers, be sure to include an relevant HTTP requests from the client-side to the server-side to help illustrate the end-to-end flow.*

> ***[Sprint 4]*** *To adequately show your system, you will need to present the* ***class diagrams*** *where relevant in your design. Some additional tips:*
>
> - *Class diagrams only apply to the* ***ViewModel*** *and* ***Model*** *Tier*
> - *A single class diagram of the entire system will not be effective. You may start with one, but will be need to break it down into smaller sections to account for requirements of each of the Tier static models below.*
> - *Correct labeling of relationships with proper notation for the relationship type, multiplicities, and navigation information will be important.*
> - *Include other details such as attributes and method signatures that you think are needed to support the level of detail in your discussion.*

ViewModel Tier

> ***[Sprint 4]*** *Provide a summary of this tier of your architecture. This section will follow the same instructions that are given for the View Tier above.*

> *At appropriate places as part of this narrative provide* ***one*** *or more updated and* ***properly labeled*** *static models (UML class diagrams) with some details such as critical attributes and methods.*

Replace with your ViewModel Tier class diagram 1, etc.

Model Tier

> ***[Sprint 2, 3 & 4]*** *Our Model Tier consists of three files: Helper.java, Needs.java, and FundingBasket.java. These three files contain the methods for our objects. Helper.java has methods for getting the username, getting the funding basket, and adding and removing from the funding basket. Needs.java has methods for getting and setting the name, cost, quantity, and type of a need. FundingBasket.java has methods for getting the funding basket, adding a need, removing a need, and checking out, which removes all needs from the funding basket and adds to the total amount funded.*

> *At appropriate places as part of this narrative provide **one** or more updated and **properly labeled** static models (UML class diagrams) with some details such as critical attributes and methods.*

Replace with your Model Tier class diagram 1, etc.

OO Design Principles

> _**[Sprint 2, 3 & 4]** Will eventually address upto **4 key OO Principles** in your final design. Follow guidance in augmenting those completed in previous Sprints as indicated to you by instructor. Be sure to include any diagrams (or clearly refer to ones elsewhere in your Tier sections above) to support your claims.
>
> We are currently implementing Low Coupling, Dependency Inversion, Single Responsibility, and Controller in our code._

> ***[Sprint 3 & 4]*** *OO Design Principles should span across **all tiers.***

Static Code Analysis/Future Design Improvements

> _**[Sprint 4]** With the results from the Static Code Analysis exercise, **Identify 3-4** areas within your code that have been flagged by the Static Code Analysis Tool (SonarQube) and provide your analysis and recommendations.
> Include any relevant screenshot(s) with each area.
>
> When we tested our code with SonarQube, no problems emerged. We received an A in reliability (0 Bugs), an A in security (0 Vulnerabilities), an A in security seview (0 Security Hotspots), and an A in maintainability (42 code smells). If we had to improve upon certain areas, we would improve on our code smells. This would consist of remove commented out blocks of code. Additionally, SonarQube recommended that we invoke methods conditionally, so we could also implement this to decrease our code smells. SonarQube also recommended that we use built in formatting to make arguments, which is not a serious issue, but could be implemented to improve efficiency._ analysis

> ***[Sprint 4]*** *If we had more time, our team would try to get all tiers to 100% coverage instead of just over 90%. We would also try to get rid of all our code smells reported by SonarQube.*

Testing

> We used JUnit assserts to test each method in all three tiers of our code. We used expected and actual values combined with JUnit asserts to ensure that each of our methods was working as intended._

Acceptance Testing

> ***[Sprint 2 & 4]*** *Currently in Sprint 4, we have had 17 user stories pass their acceptance criteria. We only have user stories for our sprint 4 presentation left, but all our programming user stories have passed their acceptance criteria..*

Unit Testing and Code Coverage

> ***[Sprint 4]*** *Our unit testing strategy is to create a method, then create the unit test for it right after. We have consistently had above 90% coverage. Our teams coverage target was 90%, and we have exceeded that. Our persistence tier has 98%, our model tier has 92%, and our controller tier has 83%. Originally, we had out controller tier at 93%, but before the submission for sprint 3, we ran into merge conflicts and lost some unit tests. We plan to implement these tests to get above 90% again.*

> _**[Sprint 2 & 4] Include images of your code coverage report.** controller-tier persistence-tier model-tier