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: Team Fore
- Team members
  - Jack McCarthy
  - Noah Lago
  - Swapnil Acharjee
  - Michael Passucci

Executive Summary

- The goal of this project is to create an online U-Fund Manager where a company's Needs are stored and users can fund those needs. the UI will represent a Food Bank using the UFund manager.

Purpose

> ***[Sprint 2 & 4]*** *Provide a very brief statement about the project and the most important user group and user goals.*

- Helper: the goal of a helper is to add needs from the cupboard to their funding basket and then proceed to checkout
- Manager: the goal of the manager is to maintain the cupboard, including removing, adding, and editing needs.

Glossary and Acronyms

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

Term	Definition
SPA	Single Page

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]*** *Provide a simple description of the Minimum Viable Product.*

- The MVP of this project is a UI that can allow both Helpers and Managers complete their tasks

MVP Features

> ***[Sprint 4]*** *Provide a list of top-level Epics and/or Stories of the MVP.*

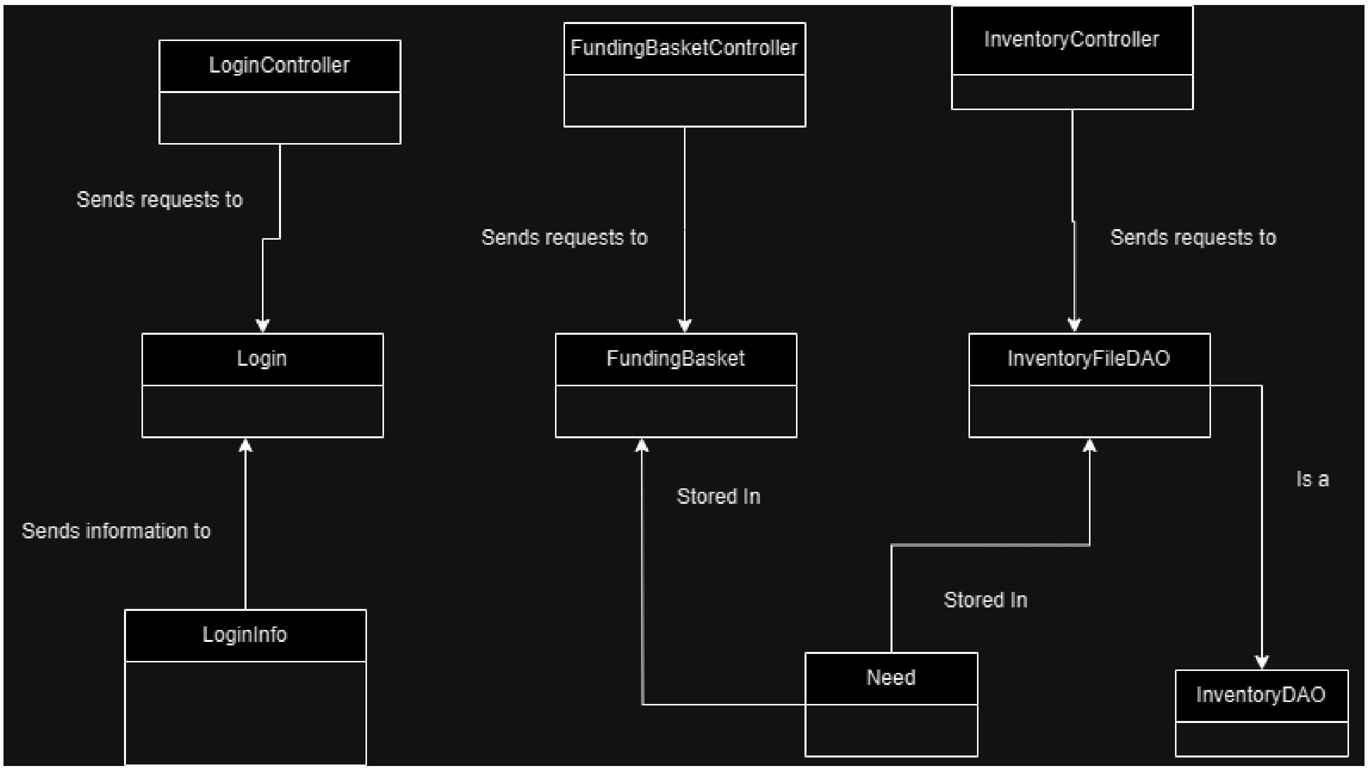
Enhancements

> ***[Sprint 4]*** *Describe what enhancements you have implemented for the project.*

- Enhancement 1: Full Login implementation
- Enhancement 2: User Registration
- Enhancement 3: Donation Matching Feature
- Enhancement 4: Error and Success Messages

Application Domain

This section describes the application domain.



> ***[Sprint 2 & 4]*** *Provide a high-level overview of the domain for this application. You can discuss the more important domain entities and their relationship to each other.*

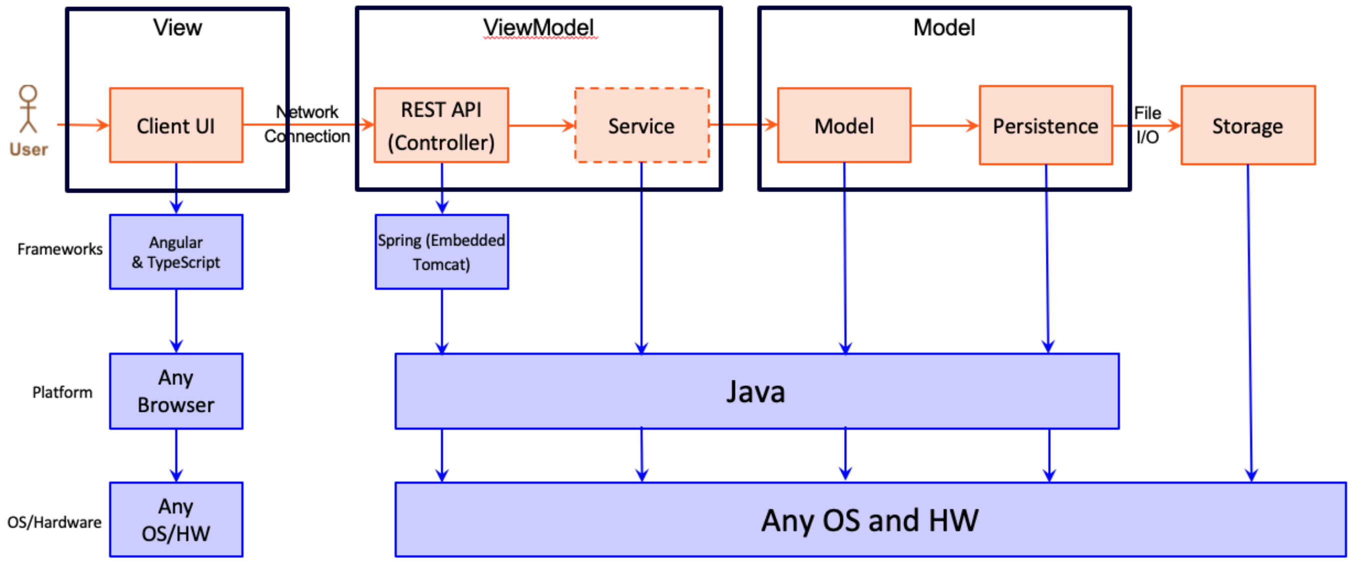
- The Need and LoginInfo classes are used to send information to their respective FileDaos
- Those FileDaos (FundingBasket, Inventory, and Login), then use that information to modify json objects and write to a file
- The three controller classes convert HTTP requests into formats that the FileDaos can understand

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*.)



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.

> *Provide a summary of the application's user interface. Describe, from the user's perspective, the flow of the pages in the web application.*

- In our UI a user must first navigate to the log in page ad enter their log in information. After logging in an admin will be able to view and edit the inventory of needs in the inventory page. After logging in a user can view needs and add needs to their checkout from the invenetory page and checkout from their basket 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]*** *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 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.*

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

Principle 1: Controller

- Our project features three controllers that are all used to link the our Sping API and out Angualar UI. Each controller is capable of processing requests from the Angualr UI and calling the correct function in the api tier.

Principle 2: Single Responsibility

- Each of our classes is only responsible for a specific task. For example our LoginContoller is only responsible for managing Login related requests while our FundingBasketController is only responsible for Funding Basket tasks instead of combining these two responsibilities into one giant Controller class.

Principle 3: Information Expert

- Classes in our system handle the tasks only for which they posses the required information. This is most apparent in our three controller classes. For example the Login controller handles all login requests because it is the only controller that has access to a Login object and therefore is the only controller with the information required to perform login operations

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.*

> ***[Sprint 4]*** *Discuss* ***future*** *refactoring and other design improvements your team would explore if the team had additional time.*

Testing

> *This section will provide information about the testing performed and the results of the testing.*

- 116 tests were run, all tests passed.

Acceptance Testing

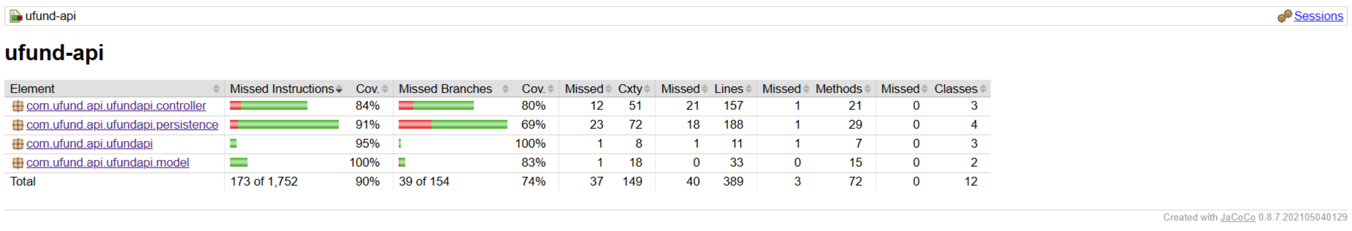
> ***[Sprint 2 & 4]*** *Report on the number of user stories that have passed all their acceptance criteria tests, the number that have some acceptance criteria tests failing, and the number of user stories that have not had any testing yet. Highlight the issues found during acceptance testing and if there are any concerns.*

- All users stories passed their acceptance criteria

Unit Testing and Code Coverage

> ***[Sprint 4]*** *Discuss your unit testing strategy. Report on the code coverage achieved from unit testing of the code base. Discuss the team's coverage targets, why you selected those values, and how well your code coverage met your targets.*

> ***[Sprint 2 & 4] Include images of your code coverage report.*** *If there are any anomalies, discuss those.*



ufund-api

Sessions

ufund-api

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.ufund.api.ufundapi.controller		84%		80%	12	51	21	157	1	21	0	3
com.ufund.api.ufundapi.persistence		91%		69%	23	72	18	188	1	29	0	4
com.ufund.api.ufundapi		95%		100%	1	8	1	11	1	7	0	3
com.ufund.api.ufundapi.model		100%		83%	1	18	0	33	0	15	0	2
Total	173 of 1,752	90%	39 of 154	74%	37	149	40	389	3	72	0	12

Created with JaCoCo 0.8.7.202105040129