

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: Bobby Fanclub (team 2)
- Team members
 - Dakota Fedor
 - Ryan Symons
 - Hiro Takeuchi
 - Caity Kurutz
 - Sahil Patel

Executive Summary

We are designing and developing a musical instrument online store, where you can browse and purchase musical instruments of your choice.

Purpose

[Sprint 2 & 4] For sprint 2, we as a group are providing basic skeleton functionality for the website.

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.

Definition of MVP

[Sprint 2 & 4] At a minimum, the website should provide minimal authentication for customers and owners/administrators (logging in and out). Customers should be able to see the products, search for specific products, as well as add or removing these products to a shopping cart. Finally, customers should be able to check out the items in their shopping cart for purchase. Owners should be able to edit the data of their inventory to correctly align with their current stock. The server should be able to handle requests made by customers and update what is viewed on the website accordingly, such as when an item is purchased, all customers should be able to see the change in stock.

MVP Features

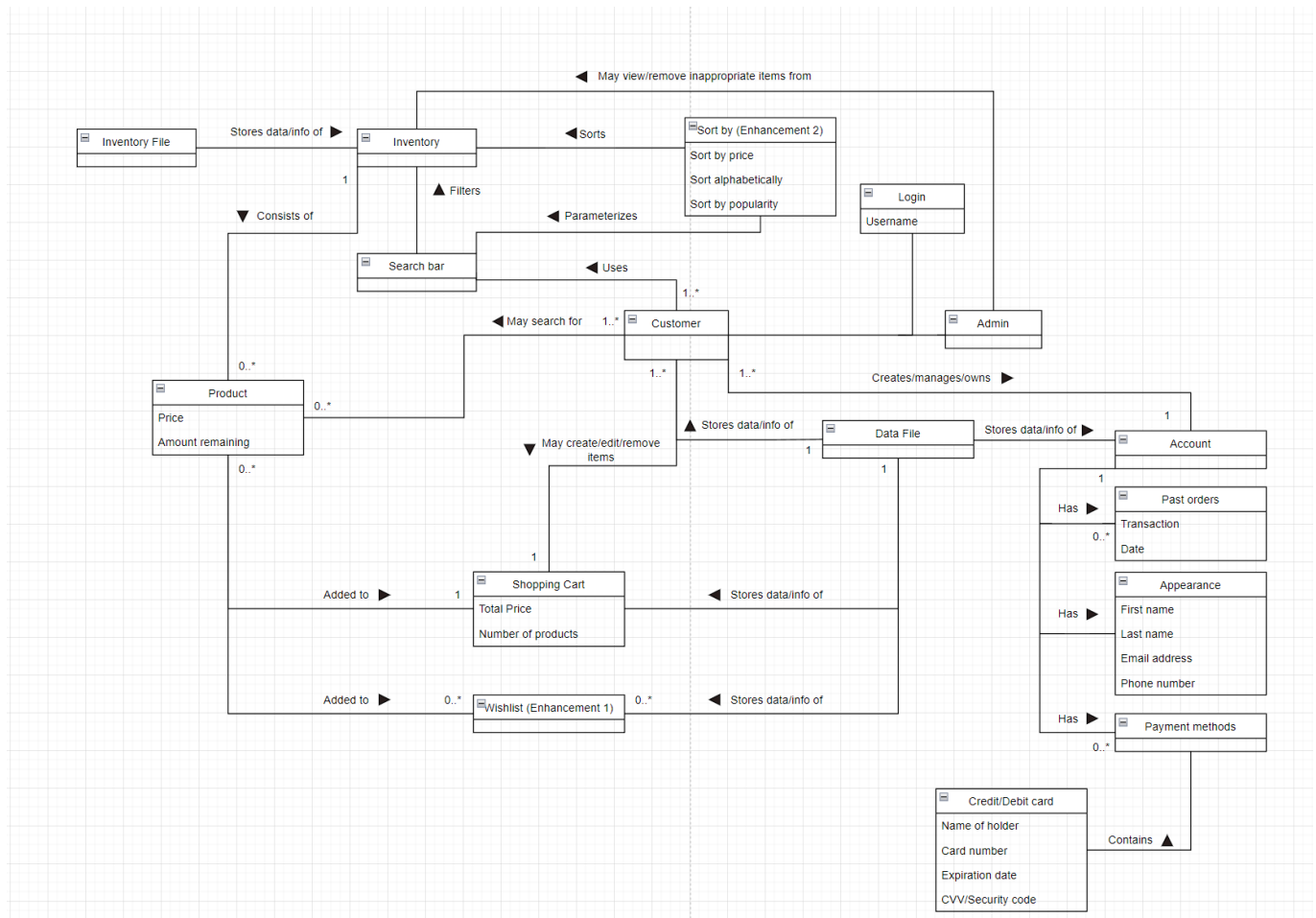
[Sprint 4] Provide a list of top-level Epics and/or Stories of the MVP.

Enhancements

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

Application Domain

This section describes the application domain.



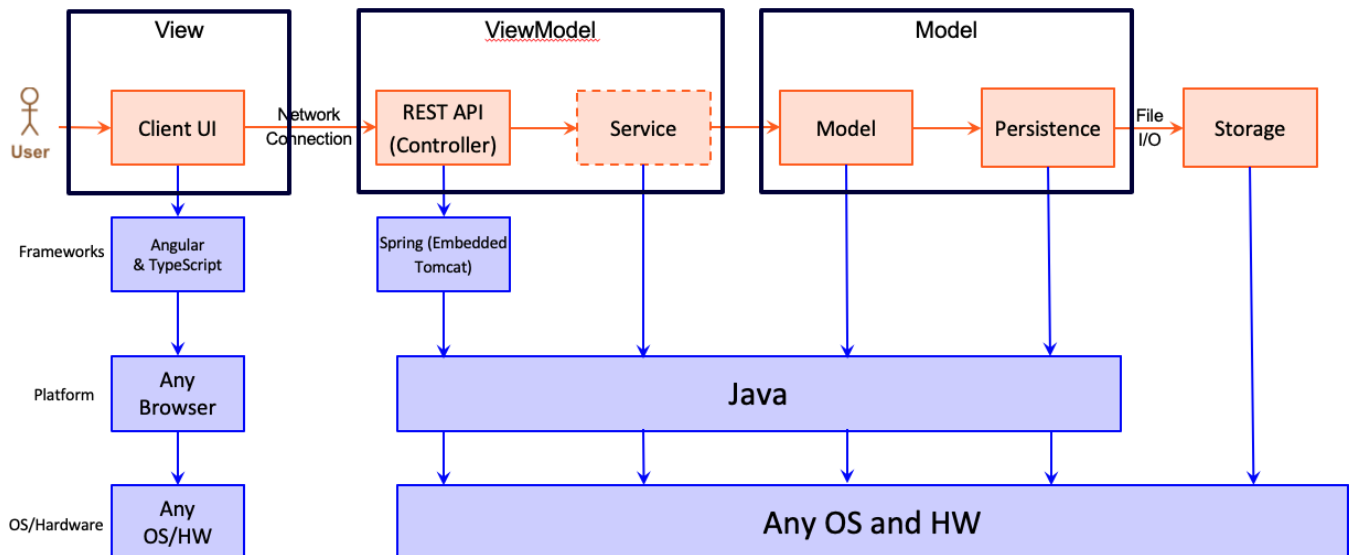
[Sprint 2 & 4] Given a login, you should be able to sign in as either a customer or an admin(owner). As an admin, you may view the inventory and make changes to it as you see fit. As a customer, you should also be able to view the inventory, search for products using a search bar (which will contain premade sort by components), and add products to your shopping cart or wishlist. Inventory and customer data will be stored in there own separate files, with the customer data file storing account information where you can view past orders, general user information, and payment methods.

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.



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

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

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 e-store 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.

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 e-store 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] Discuss at least **4 key OO Principles** in your current design. This should be taken from your work in "Adherence to Architecture and Design Principles" that you have completed in a previous Sprint. 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**.

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.

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.

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.