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: TEAMMUSIC
- Team members
  - Aaron Santos
  - Darian Cheung
  - Ryan Haver
  - Victor Oliveira
  - Spencer Randolph

Executive Summary

- Our project is an estore specializing in the sale of music products such as albums and music-playing devices.

Purpose

- The purpose of this project is to connect the store owner with consumers by creating an estore that allows consumers to purchase products.

Glossary and Acronyms

> *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.*

- As an owner, I want to be able to add and remove inventory
- As a customer, I want to able to add and remove items from my shopping cart

Definition of MVP

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

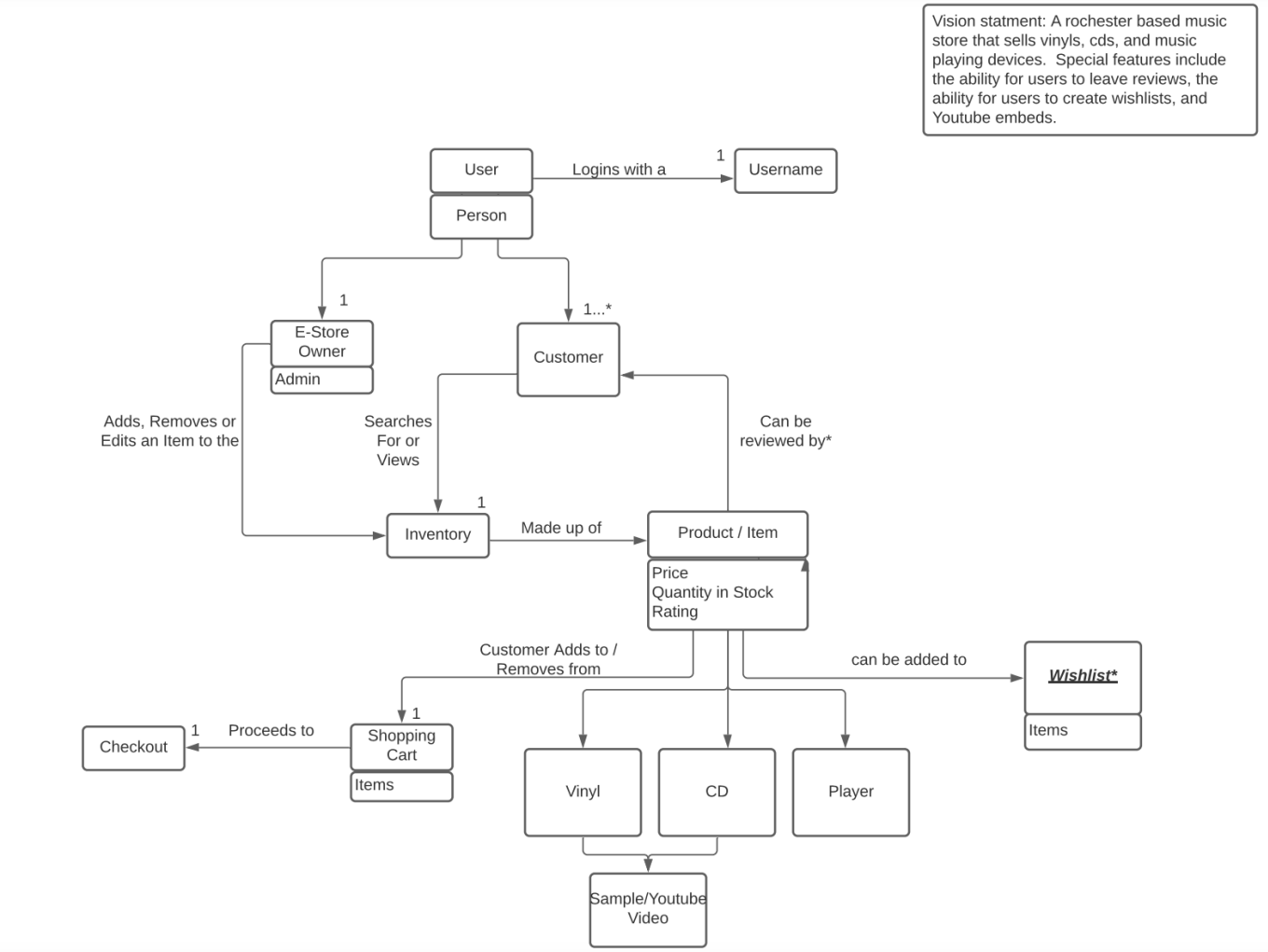
MVP Features

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

Roadmap of Enhancements

> *Provide a list of top-level features in the order you plan to consider them.*

Application Domain



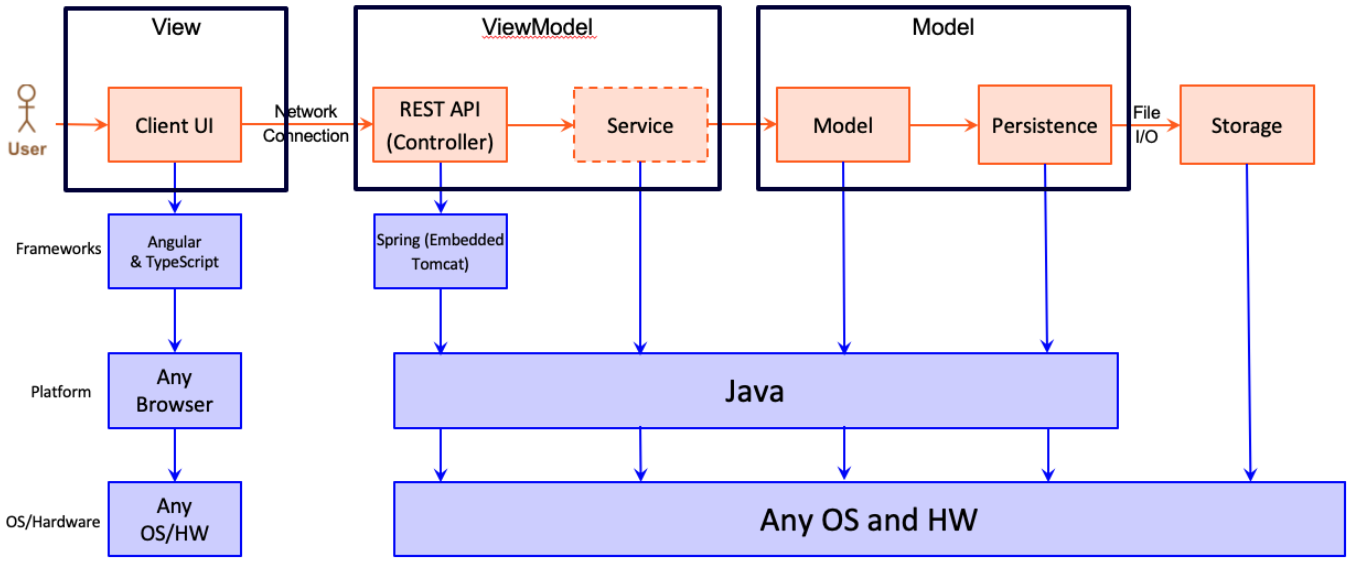
> The domain of the application is retail sales. The application is designed around the User class. The User contains basic information like username, password etc. The user is either a customer or an administrator. The user can view products and add them to their shopping cart if they are logged in. The administrator can alter the product list and change prices.

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.



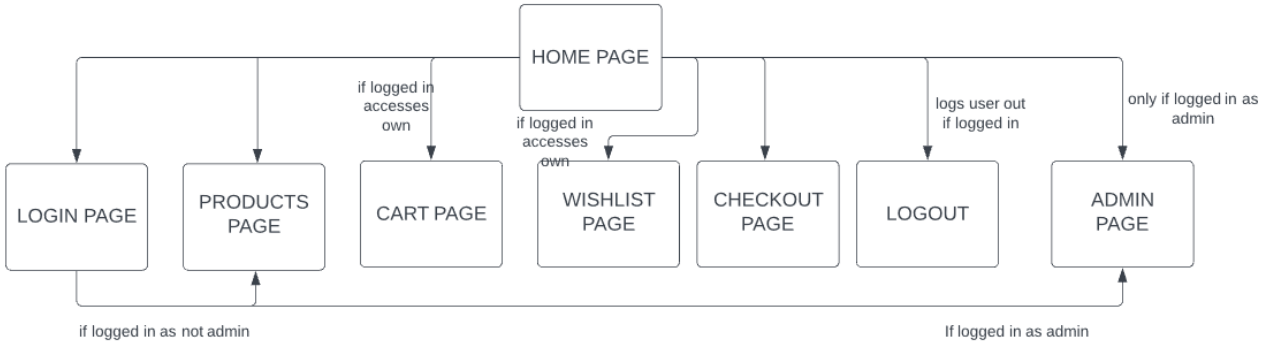
The e-store web application, is built using the Model–View–ViewModel (MVVM) architecture pattern.

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



> The user starts on the Home page which has links to all other pages. These links are accessible on all other pages. If the user click login page they are taken to a page where they can enter login information for an account and create a new account. If the user logins with a non-admin account they are taken to the products page, if they are logged in with an admin account they are taken to the admin page. If the user clicks the products page they are taken to page that lists all the products and has the functionality to add a product to the user's cart or wish-lit if they are logged in. If the user clicks the cart page they are taken to a page that displays their cart if they are logged in. If the user clicks the wish-list page they are taken to a page that displays their wish-lit if they are logged in. If the user clicks the checkout page, the products in their cart are listed and a button is displayed that has the functionality to simulate a purchase by removing the products from the user's cart. If the

> user clicks logout, they are logged out of their account if they are logged in. If they user clicks the admin page they are taken to a page that has the ability to delete a product and create new products. The admin page is only accessible if the user is logged in as admin.

View Tier

> *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.*

> *You must also provide 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. Be sure to include an relevant HTTP reuqests from the client-side to the server-side to help illustrate the end-to-end flow.*

ViewModel Tier

> *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 static models (UML class diagrams) with some details such as critical attributes and methods.*

Model Tier

> *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 static models (UML class diagrams) with some details such as critical attributes and methods.*

Static Code Analysis/Design Improvements

> *Discuss design improvements that you would make if the project were to continue. These improvement should be based on your direct analysis of where there are problems in the code base which could be addressed with design changes, and describe those suggested design improvements.*

> *With the results from the Static Code Analysis exercise, discuss the resulting issues/metrics measurements along with your analysis and recommendations for further improvements. Where relevant, include screenshots from the tool and/or corresponding source code that was flagged.*

Testing

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

Acceptance Testing

> *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

> *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. If there are any anomalies, discuss those.*