PROJECT Design Documentation

Team Information

- Team name: name
- Team members
    - Finn Saunders
    - Justin Lin
    - Tashi Tseten
    - PJ Esterly

Executive Summary

This is a summary of the project.

This project is an e-store application with minimal functionality, security, and presistance.

Purpose

The project is an e-store application that will allow the customer(s) to shop for certain sports balls with the goal of having an easy shopping expereince and the ability to log out and login without their data disappearing.

Glossary and Acronyms

Term	Definition
SPA	Single Page
DAO	Data Access Object
API	Application Programming Interface
HTML	Hypertext Markup Language

Requirements

This section describes the features of the application.

-Login Page -Unique Admin and User Pages -Inventory Control -Product Search -Product Management -Shopping Cart

Definition of MVP

The MVP will have the following functionality: minimal authentucation for users (owner and custmoer and only usernames), customer functionality including viewing products, a shopping cart, and a checkout, inventory management, and data persistance.

MVP Features

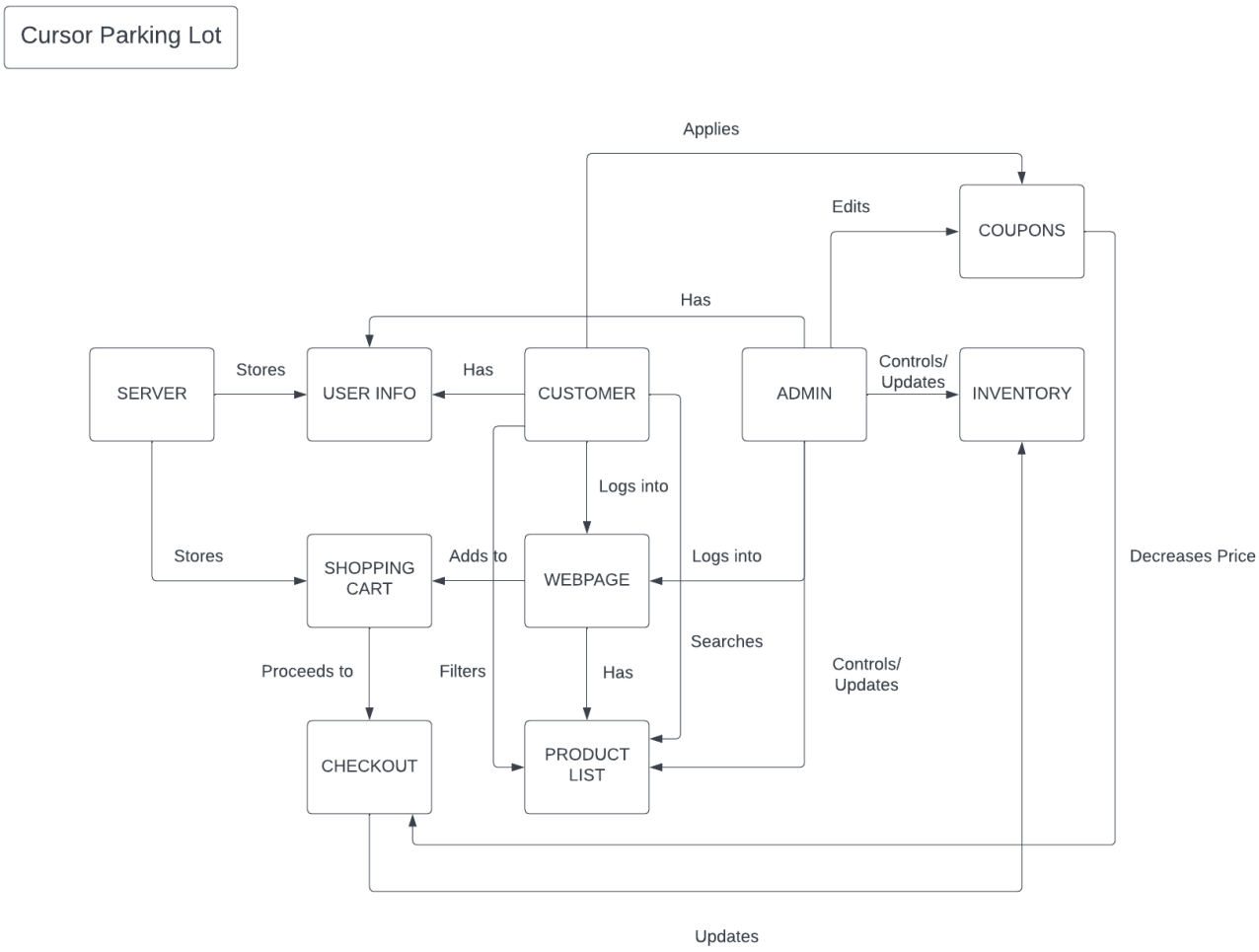
-Inventory -Add -Remove -Modify -Login -Owner view -Customer veiw -Username -Shopping Cart -Add -Remove -Checkout -Browse -View products -Search products

Roadmap of Enhancements

-Inventory Management -Browse -Login -Shopping Cart -10% #1 -10% #2

Application Domain

This section describes the application domain.



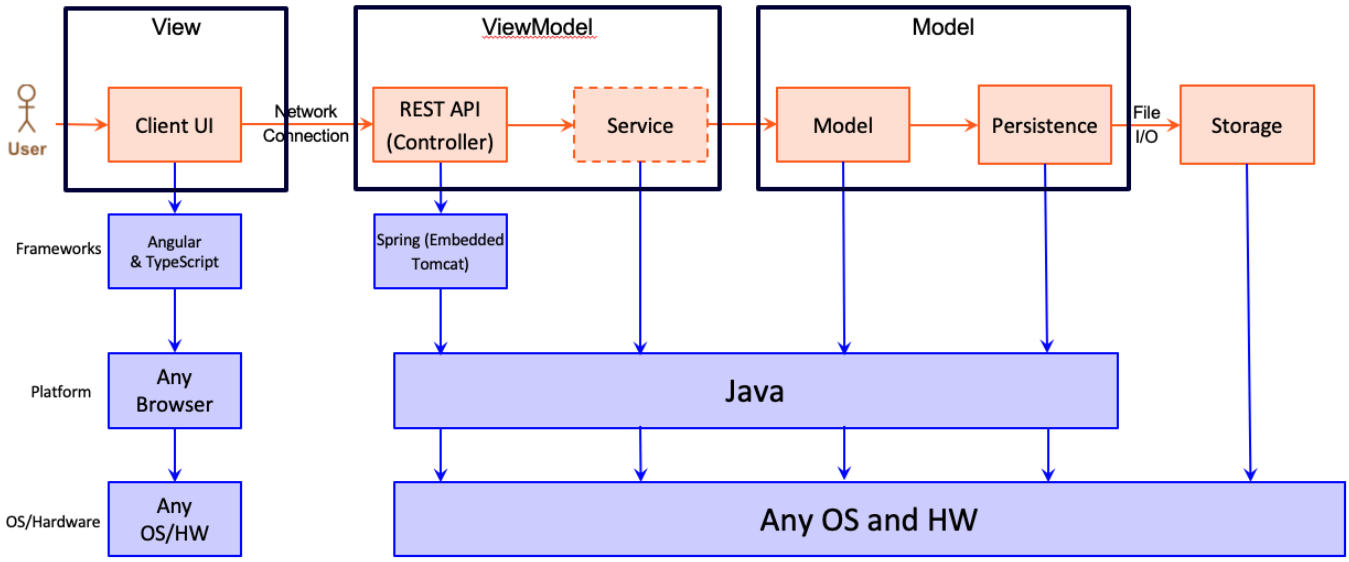
The domain for this application consists of the user, owner, inventory, product browsablilty, and login function as the large features for this application. The domain model helps to illustrate how these features interact with each other and includes some smaller sub-features and how they interact with each other and the large features.

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

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

> *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.*