PROJECT Design Documentation

Team Information

- Team name: awesome blasters
- Team members
  - Thomas Garcia
  - Aagman Relan
  - Ivan Lin
  - JianZhuang Jiang

Executive Summary

Our website must allow signed in users to add products to their shopping cart and checkout said items utilizing our UI to navigate throughout the website.

The webiste we are creating should allow a user to sign up to a new account or log into an existing account to add a product our company is offering in their shopping cart and then proceed to purchase said product if they wish at checkout.

Purpose

Rochester Parkour is family run gym and fitness center situated in downtown Rochester. Our goal is to encourage individuals to partake in physical activties to better themselves physically, and mentally. We have been serving our community since 1980's. The target audience for this business is anyone who wants to work on themselves. This document describes the features and functionality of our newly developed Web application.

Glossary and Acronyms

Term	Definition
Visitor	A Person on the website who is not logged in
User	A person on the website who is logged in
Admin	This person can manage products and users of the website

Requirements

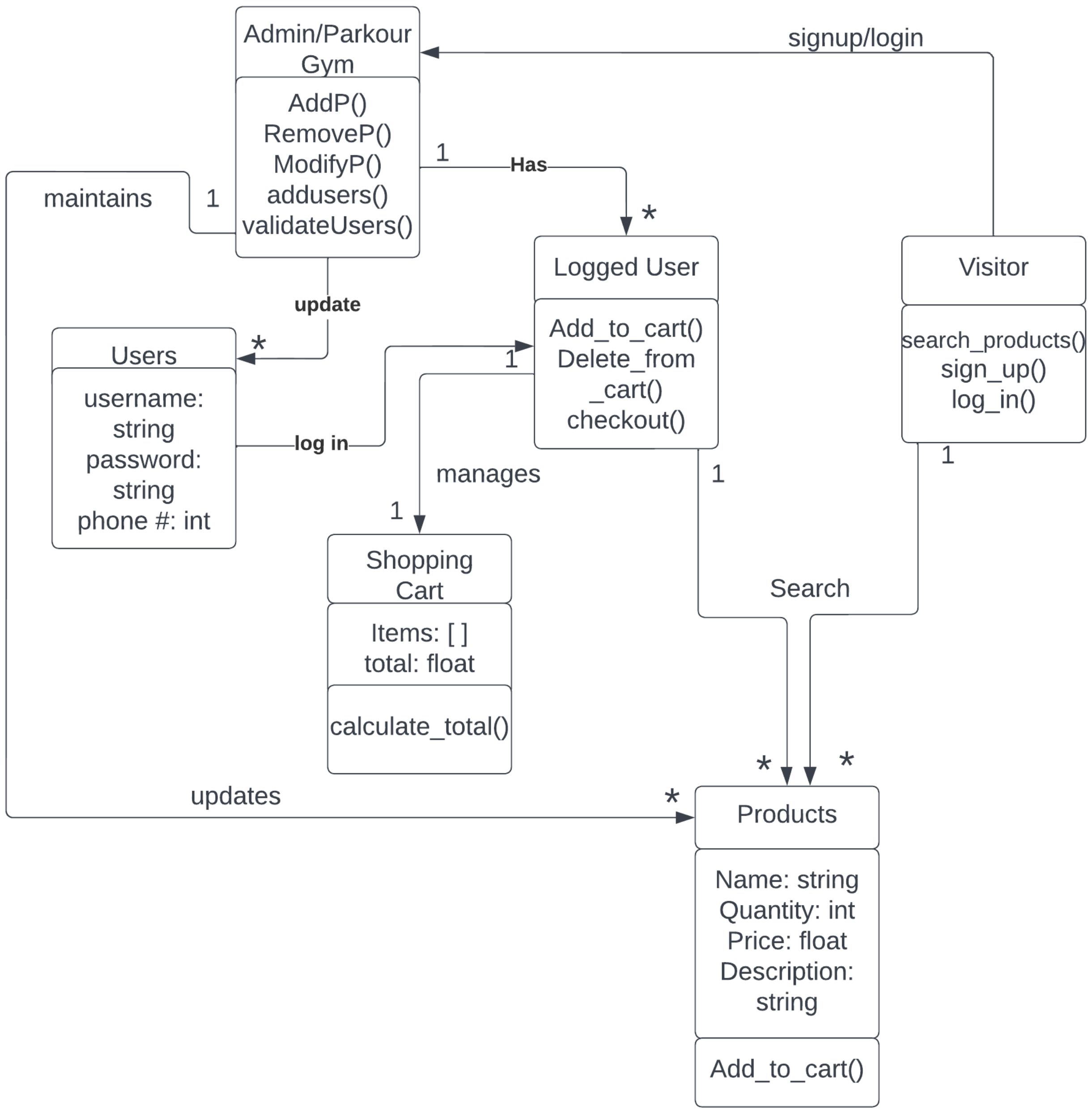
Definition of MVP

- The admin should be able to Add/Remove/Modify the products offered at the Estore
- Visitors can sign up and login as users of the website
- Users can add or remove products from their shopping cart.
- Users can checkout their shopping carts.

Roadmap of Enhancements

- User Rating: The user should be able to add ratings for the products and leave a review for the products if the user wants
- View Typed Password: Visitor should be able to view the typed password when signing.

Application Domain



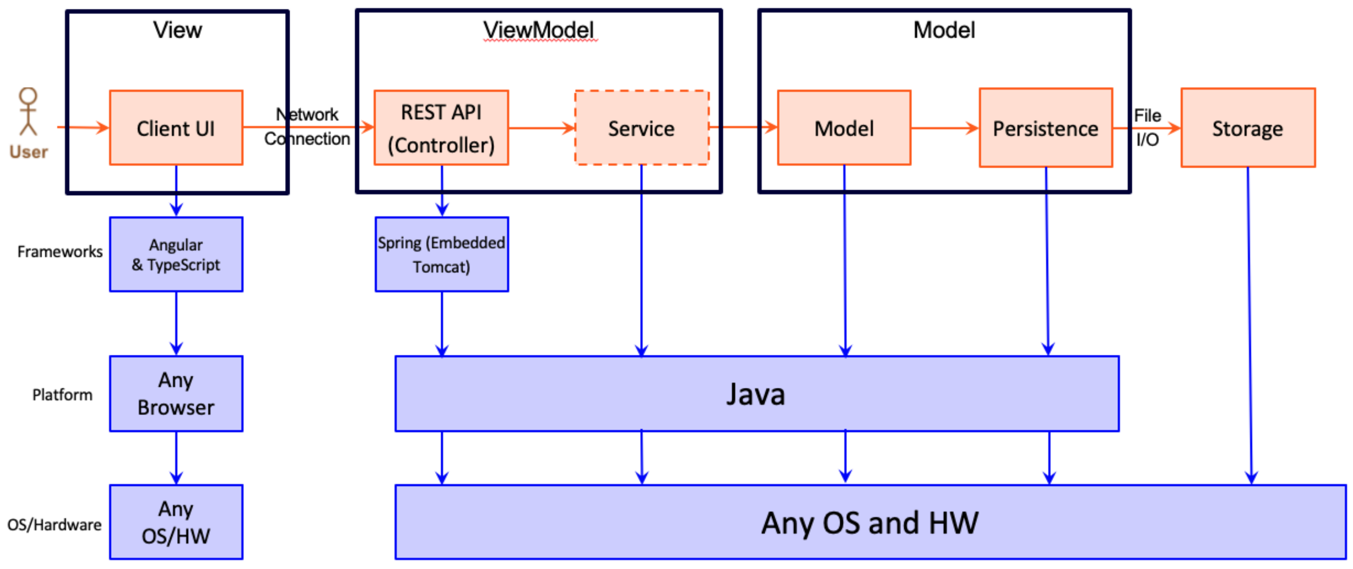
The Admin/Parkour gym maintains the Products that the store seliing. Admin/Parkour gym also validate visitors when they try to login or sign up. If the validation is successful they are either logged in as an existing user or a brand new user. The user has the ability search and add/remove products to/from their personal shoppping cart. Users can then checkout their shopping cart.

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.

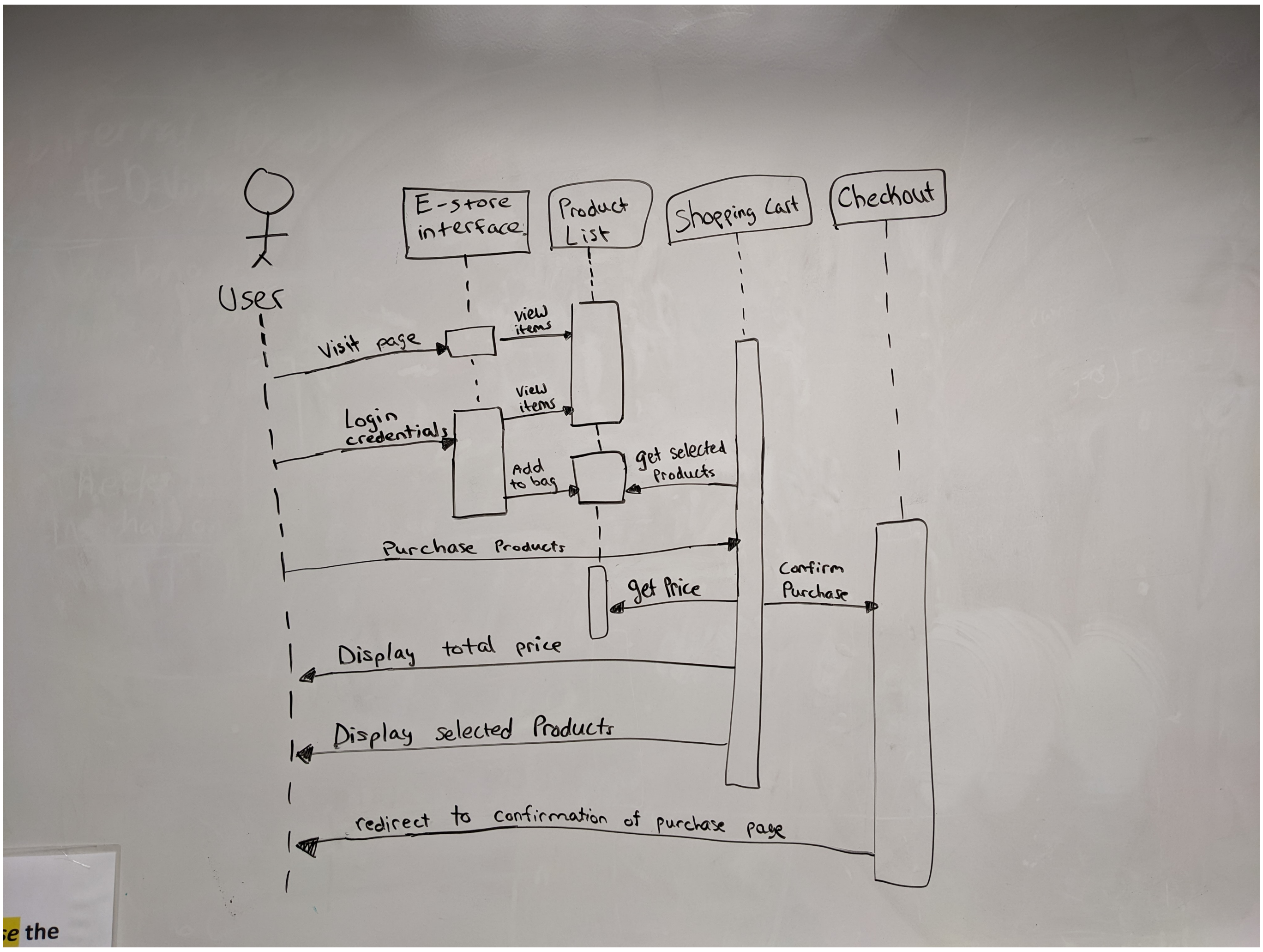
Overview of User Interface

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

> Once a user visits the home page of the e-store they are greeted with a big picture and in the middle of their screen and only have a navigation >bare to guide them throughout the website. Any user is able to view >products but in order to purchase them they need to be logged in, users >will be redirected to login/signup page if they try to add an item to >their shopping cart. The navigation bar makes traversing through the >e-store simple as everything is very straightfoward/ one dimensional.

View Tier

> There is an admin view which only the owner of the e-store can have access to via credentials and they have the abilities to add, update and remove items as they please. There is a customer view where they don't have the abilities to update/add/ and remove items such as the admin and the only thing they can do is login/signup and add items to their shopping cart only if they are logged in to then purchase.



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.*