

PROJECT Design Documentation

Team Information

- Team name: Crocky's Cohort
- Team members
 - Alex Vernes
 - Holden Lalumiere
 - Stevie Alvarez
 - Isaac McKinney
 - Tylin Hartman

Executive Summary

Purpose

The project will be a website of an online e-store called "Crocky's Crochet", where various crochet products are sold. Customers can browse and purchase products and the administrator can manage the inventory through the website.

Glossary and Acronyms

Term	Definition
MVP	Minimum Viable Product

Requirements

Definition of MVP

- A Customer can login with a username into the e-store. Customers can browse and search for products and also add those products to their shopping cart. When that customer logs in later, the products that were in their shopping cart from a previous login are still there.
- The Admin can login with a special Admin-only username into the e-store. The Admin can edit, add, and remove and generally manage the inventory of the e-store.

MVP Features

- Customer can login
- Customer can browse products
- Customer can search products
- Customer can add products to their shopping cart
- Admin can login
- Admin can manage the inventory of the e-store

Roadmap of Enhancements

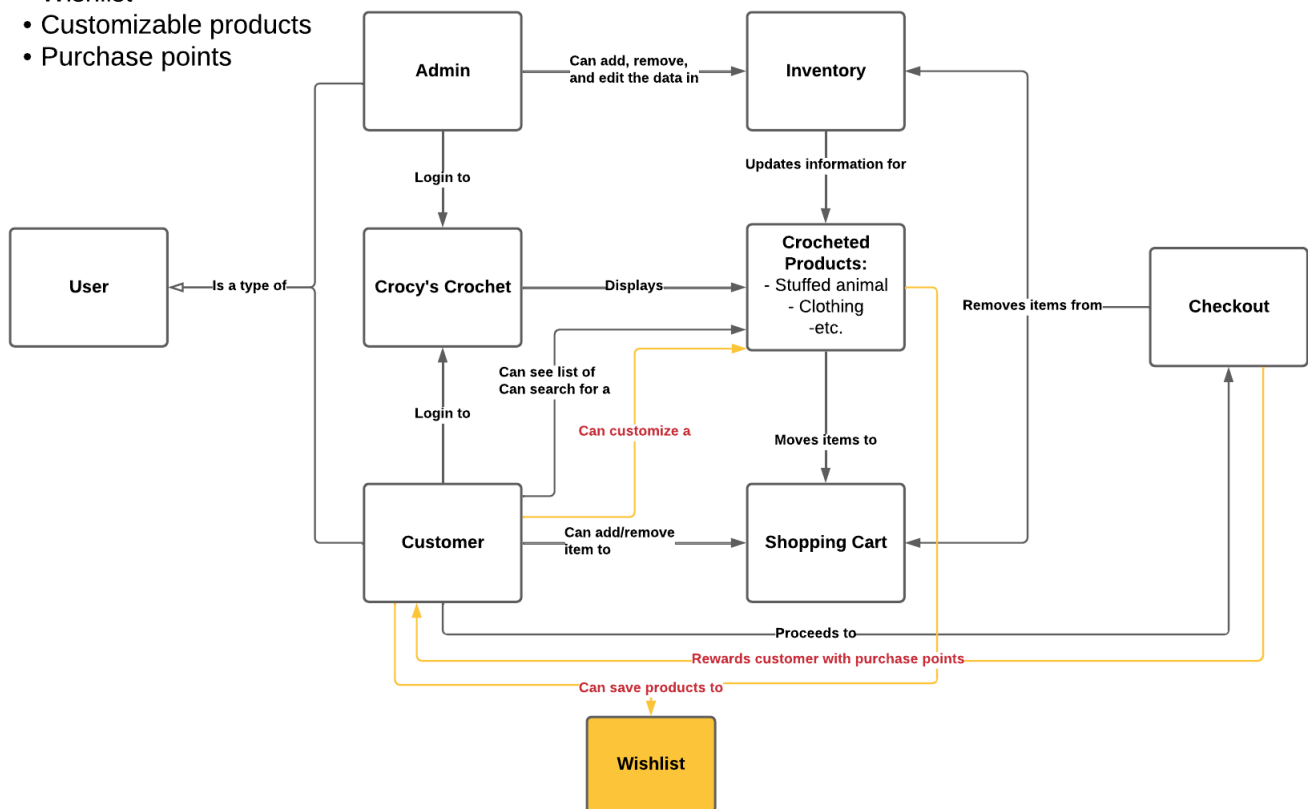
1. Customer can browse products

2. Customer can search products
3. Admin can manage the inventory of the e-store
4. Customer can login
5. Admin can login
6. Customer can add products to their shopping cart

Application Domain

Potential 10% Features:
(10% features marked with
yellow lines and red text)

- Wishlist
- Customizable products
- Purchase points



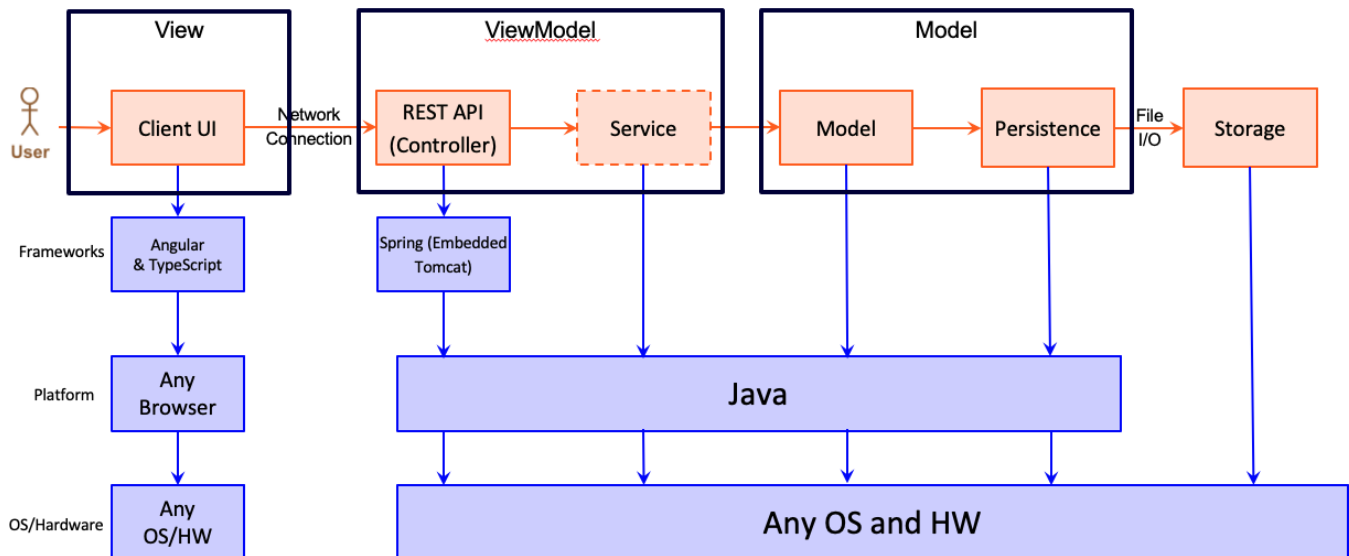
In the domain model, firstly, there are two kinds of Users that can login to Crocky's Crochet, an Admin and a Customer. Crocky's Crochet displays for Users the products that are in the inventory. The Admin can manage those products that are in inventory. A Customer can browse the products displayed by Crocky's Crochet and can add products to their shopping cart. A Customer can proceed to the checkout with the products in their shopping cart, making the purchase complete.

Architecture and Design

The files will be split into two sections, the e-store UI and the e-store API. The e-store UI contains all components that are part of every webpage for the e-store. The e-store API is split further into three subsections, controller, model, and persistence. Controller mediates communication between the front-end and the back-end in terms of data stored. Model provides a platform for viewing and modelling the data stored. Persistence provides a method of manipulating the data stored.

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

View Tier

Customers open the webpage to a generic product search page, displaying the all products currently in the inventory. The Customer can click "login" and be taken to the login page, where they can either login with an existing username or create a new account. Now that the Customer is logged in, when clicking in a product, the option to add that product to the cart displays alongside other information about that product. The Customer can add products to their shopping cart. The Customer can also remove the item or proceed to checkout, where it clears all products in the cart. When the Admin logs in, the same display of products is shown alongside the "New Product" button. When the Admin clicks on a product, they can edit the information for that product or delete the product.

ViewModel Tier

Model Tier









Static Code Analysis/Design Improvements

Testing

Acceptance Testing

Unit Testing and Code Coverage

estore-api

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.estore.api.estoreapi		88%		n/a	1	4	2	7	1	4	0	2
com.estore.api.estoreapi.persistence		100%		100%	0	43	0	120	0	31	0	3
com.estore.api.estoreapi.controller		100%		100%	0	26	0	107	0	20	0	3
com.estore.api.estoreapi.model		100%		100%	0	38	0	82	0	30	0	3
Total	5 of 1,473	99%	0 of 52	100%	1	111	2	316	1	85	0	11

After each collection of files were written, such that it is ensured that they will not be edited later, a team member or two were assigned to write unit tests for those files. The code coverage is 99% complete with the exception of the main method. We did not expect to get that much code coverage in this sprint, but with efficient code writing we were able to go beyond the expected 90%.