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: Better Monkeys
- Team members
  - Lucie Lim
  - Dara Prak
  - Jaden Seaton
  - Robert Huang
  - Adam Pang

Executive Summary

A snack e-store where users can browse, purchase, and rate snacks. They can also view their purchase history just in case they loved a snack but dont remember the name.

Purpose

The purpose behind the project is to learn about the development process of a website from scratch. In this case we learned the basics for a e-store website with functions for the owner and customers, such as adding new products, changing info of the product, browsing products, and updating customers shopping cart.

Glossary and Acronyms

> *Provide a table of terms and acronyms.*

Term	**Definition**
SPA	Single Page

Requirements

This section describes the features of the application.

User authentication: Depending on username and password the user will have certain responsibilities, and features they are able to access. The owner will be able to add, remove and change info of a product. Customers will see their subscription, reviews, and browse snacks.

Shopping cart: Users will be able to add and remove stuff from their cart, and update it to their satisfaction.

Rating System: Users will be able to create a review of a snack that would show others how good a snack is worth buying. They are able to see the average rating of a snack along with individual ratings of others.

Order History: users will be able to see their order history for every snack purchase they made. They can see the date, time and what snacks they bought.

Definition of MVP

A system where a user can make a snack, search for a specific snack, search for snacks given a word of the snack, delete a snack, get all snacks, and update details of a snack.

MVP Features

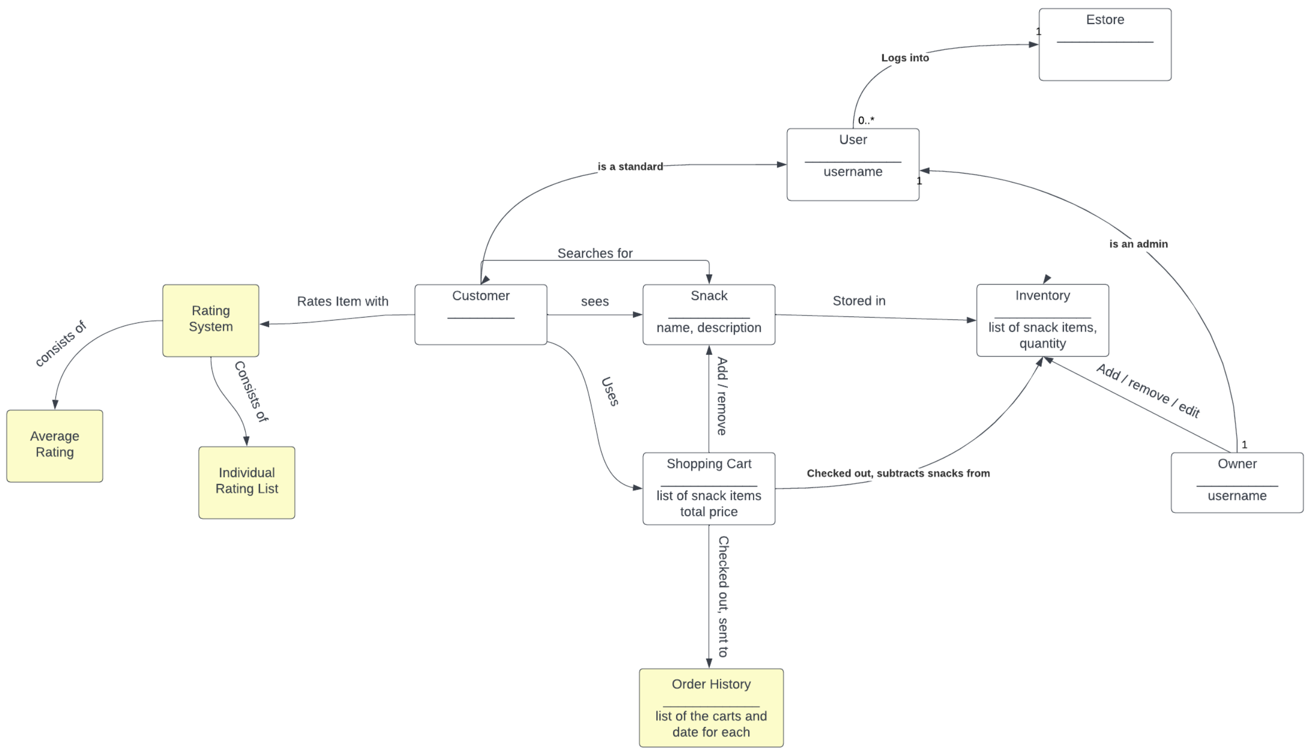
- Delete a single snack
- Update a single snack
- Create a new snack
- Get entire inventory
- Search for a snack
- Get a single snack

Roadmap of Enhancements

- Order history
- Ratings & reviews for snacks

Application Domain

This section describes the application domain.



A Rochester snack store owner who sells snacks like chips, drinks, baked goods

Enhancements to include:

- **Rating Model:** Rating model for different snacks, so each Buyer can submit a rating 1-5 and the average rating is displayed for each snack.
- **Order History:** When a Buyer checks out their cart, the contents of their cart at the time is saved along with the time of checkout and it can be viewed on an order history page.

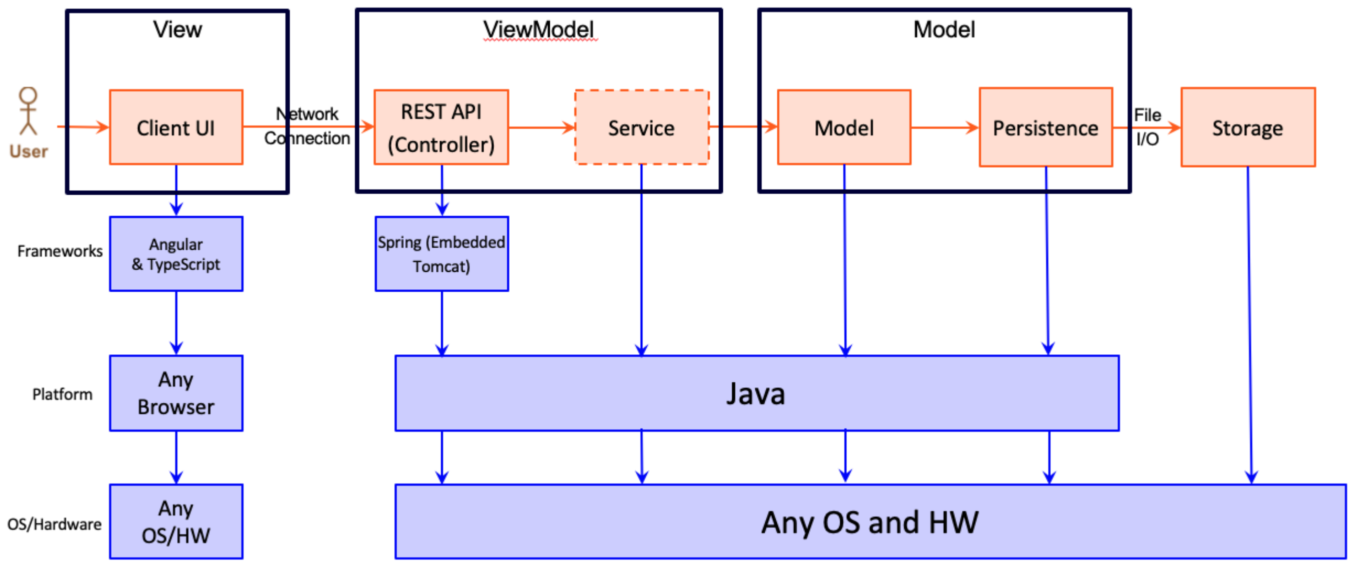
This is a domain model of our snack e-store, it shows the relations between the user, and the features of our website such as the shopping cart, where the customer can add and remove items from the cart. The model also displays the relationship between the owner and the inventory, adding, removing, and updating snacks. Also it shows new features such as the subscription where the customer can edit a subscription and change how frequently they can recieve snacks.

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.

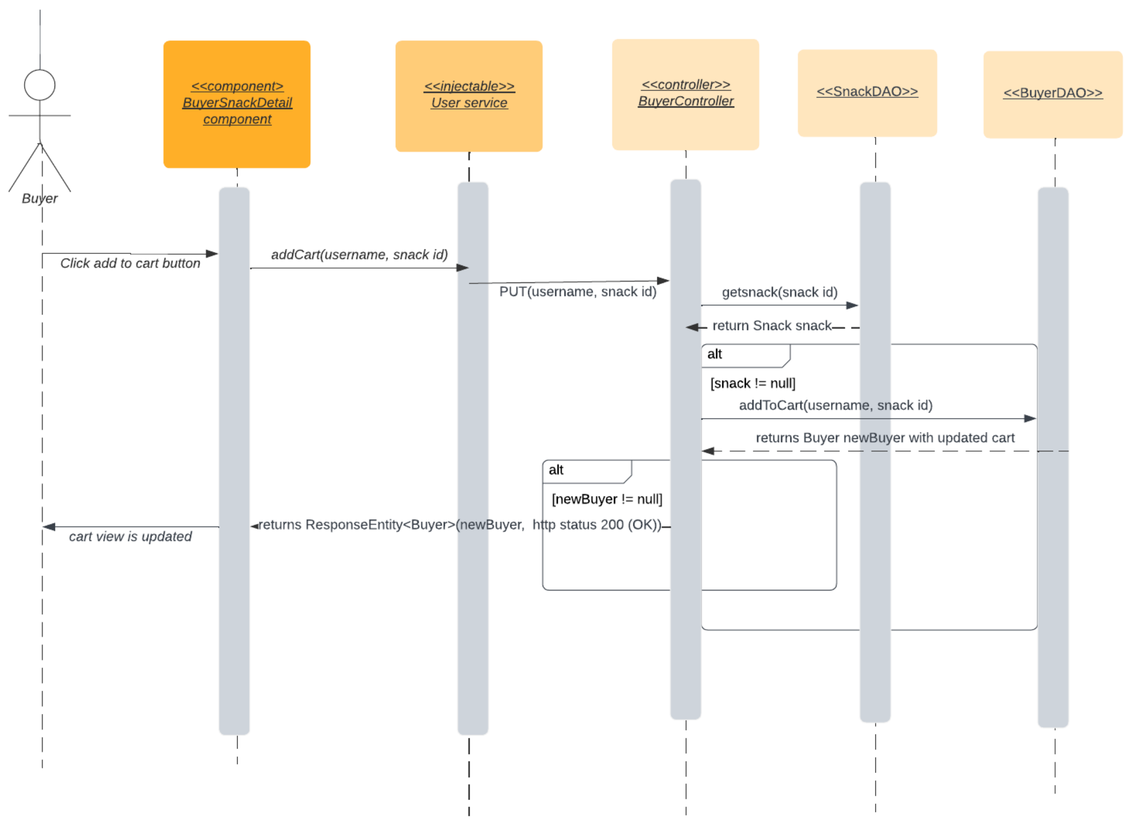
When a user first encounters our website they are presented with 3 buttons: inventory, catalog, and login. When they click on inventory they have access to the owners actions such as updating a snack, adding a new snack, and even deleting it. When they choose the catalog they are presented with snacks, they can search but cannot update them, and they can add items to their shopping cart. When they click login they are given 3 fields where they can login as the owner, a customer, or register for the website.

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.*

Successfully Add to Cart Sequence Diagram



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

Our Buyer implementation files have not been covered yet, and should the project continue, it would be in our best interest to focus on the buyer files and testing each individual function. Testing these functions would allow us to continue developing the shopping cart files and implement an organized method for handling our data persistence for buyers. Design improvements would lie primarily in the shopping cart files, as our code analysis shows that we have malfunctioning methods that need to be revised.

- We can focus on improving our shopping cart by implementing simpler data structures and elaborating in the future into a more abstract class.

> *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

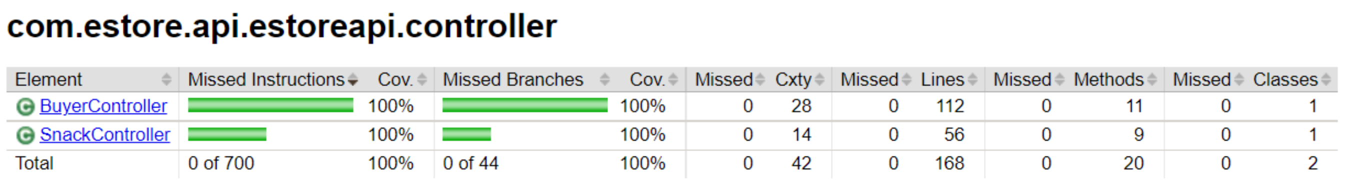
> *This section will provide information about the testing performed and the results of the testing.*

Acceptance Testing

Passed All Criteria: 27 Passed Some Criteria: 0 Passed No Criteria: 2 The main issue with acceptance criteria testing is that we haven't completed the frontend of the 10% features for out product. There is 0 passed com criteria because we split up the frontend and backend parts of features and right now only the front end of the 10% features havent been completed so there is 0 for some criteria passed. On the backend, a lot of our stories passed all the criteria.

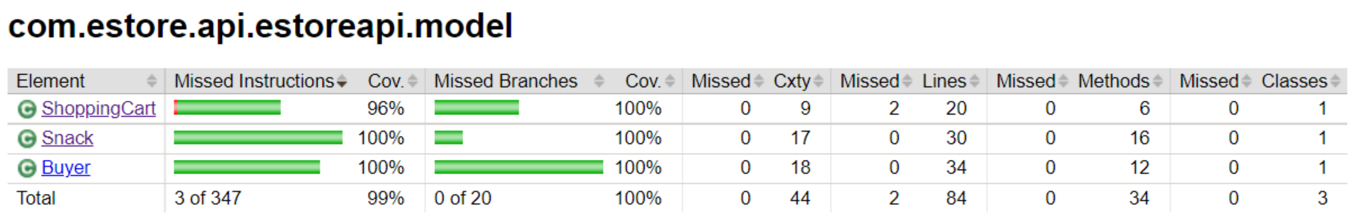
Unit Testing and Code Coverage

Our unit testing strategy is primarily in creating a fake database of snacks and using that fake database to assert various functions and conditional statements. The code coverage achieved from unit testing our snack files have shown to be above 90% covered, where the final 10% lie in additional functions yet to be covered and will be our primary focus for code coverage.



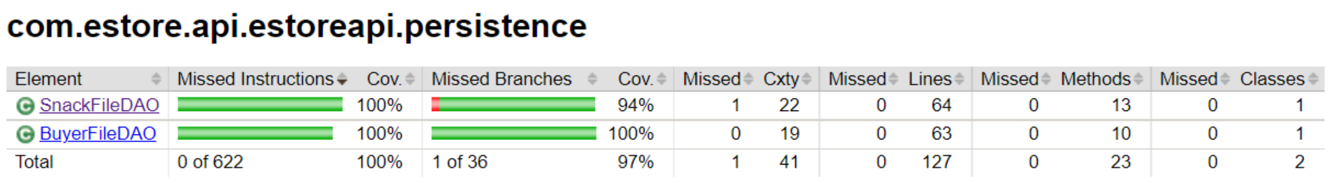
com.estore.api.estoreapi.controller

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
BuyerController		100%		100%	0	28	0	112	0	11	0	1
SnackController		100%		100%	0	14	0	56	0	9	0	1
Total	0 of 700	100%	0 of 44	100%	0	42	0	168	0	20	0	2



com.estore.api.estoreapi.model

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ShoppingCart		96%		100%	0	9	2	20	0	6	0	1
Snack		100%		100%	0	17	0	30	0	16	0	1
Buyer		100%		100%	0	18	0	34	0	12	0	1
Total	3 of 347	99%	0 of 20	100%	0	44	2	84	0	34	0	3



com.estore.api.estoreapi.persistence

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
SnackFileDAO		100%		94%	1	22	0	64	0	13	0	1
BuyerFileDAO		100%		100%	0	19	0	63	0	10	0	1
Total	0 of 622	100%	1 of 36	97%	1	41	0	127	0	23	0	2

The teams code coverage is finished with the mvp and 10% for the backend.