PROJECT Design Documentation

Team Information

- Team name: Name Undefined
- Team members:
  - Cole Johnson
  - Shandon Mith
  - Rhythm K C
  - Andrew Chang
  - Hunter Davonport

Executive Summary

This project is a semester long assignment that involves a team of 5 students in our case. In this project, the team will collaborate in creating and implementing a website that pertains a business where the admin and users are able to access and use. The end result of the project is the website fully functioning.

Purpose

In our case, our website revolves an interactive system where courses can be bought and accessed after purchasing. The most important user groups are the students. The goals of these users are to purchase and be able to access the content in the courses available in the website.

Glossary and Acronyms

Term	Definition
MVP	Minimum Viable Product = a list of stories/epics that are required to be implemented in the first release of product.

Requirements

This section describes the features of the application.

Definition of MVP

The Minimum Viable Product is a product where users are able to access and use a website where they are able to buy courses where they can later access bought courses to view their content.

MVP Features

List of Epics:

1. Add Courses to Cart An epic where students are able to add courses to cart, so they can purchase later.
2. Add/Update courses An epic where the admin is able to add, update, and remove courses.
3. Log In An epic where a user logs in as admin or student

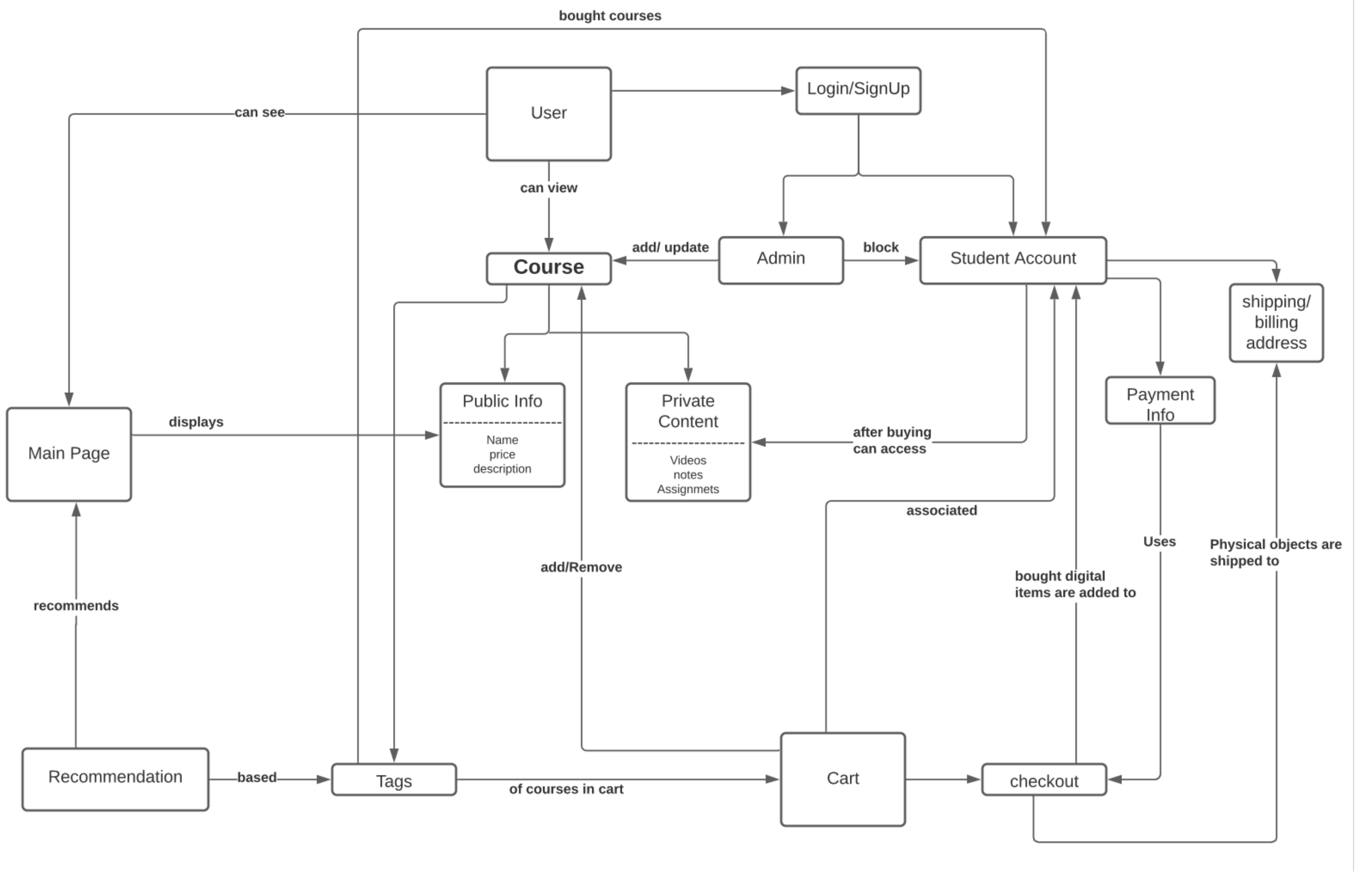
4. Course Access An epic where the three user groups each have their own purposes to access courses. For users, they can view course contents. For students, they can interact and view all of the contents of the courses they've purchased. For the admin, they can view and edit courses.
5. Recommendation An epic where all user groups can see recommended courses on the main page.
6. UI An epic where all user groups are able to access the website with its user interface.
7. Banning Users An epic where the admin is able to ban users from the website, either temporarily or permanently.
8. View Purchased Courses A specific epic in "Course Access" where students are able to view and interact all the contents in a purchased course.
9. Student Account An epic where students can manage their own account.

Roadmap of Enhancements

1. UI
2. Log In
3. Add/Update Courses
4. Add Courses to Cart
5. Course Access
6. View Purchased Courses
7. Recommendation
8. Student Account
9. Banning Users

Application Domain

This section describes the application domain.



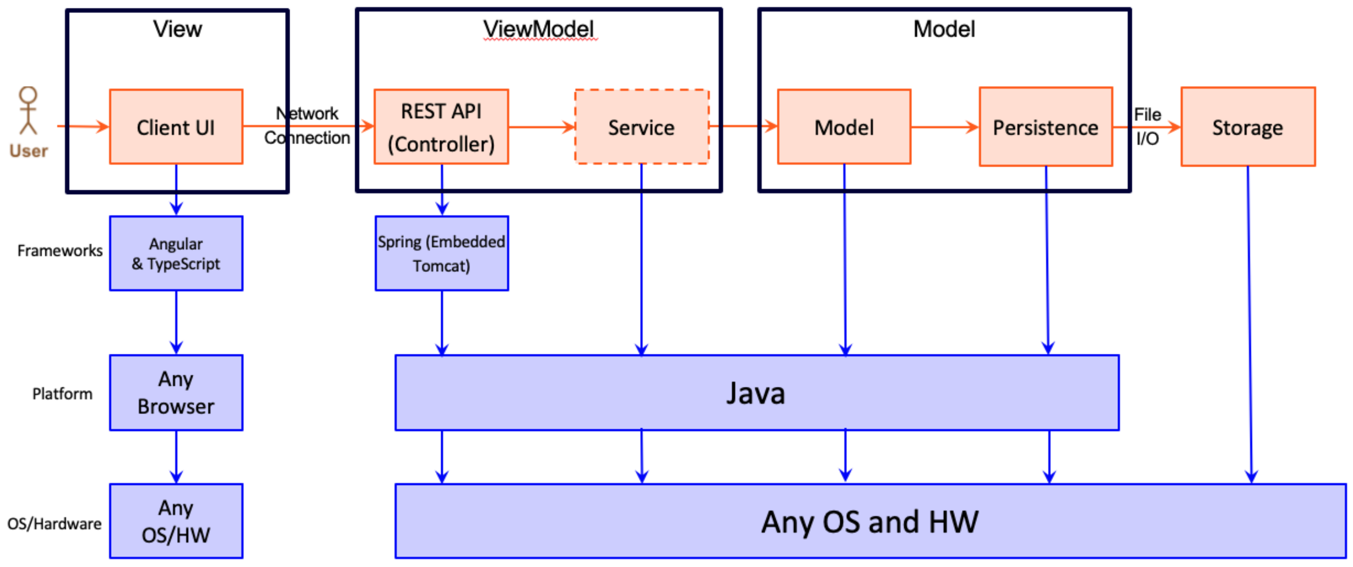
The main product in this website are the courses and in each course, they contain their own name, price, and description as their public info while their own videos, notes, and assignments are the private content of the courses. When a user enters the website, they first see the main page that contains courses. The user can view courses or login/sign up at this point. When a user signs in, they either log in as an admin or log in/sign up as a student. When an admin is logged in, they can add, update, or delete courses while also being able to ban students if needed. With students, they have their own information saved in their account, such as their payment info and shipping/billing address. When students are logged in, they can add courses to a cart and can later checkout to purchase courses they've added with their payment info. Once the students have bought their courses, they can access the private content of the courses. Every course has their unique set of tags according to the content and based on what the user has viewed and/or added to their cart, the website would recommend courses that would interest the user in the main page.

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 enters the website, they first enter the main page where they can see components they can interact with. Each component have their own specific purpose and when a user interacts with it, the Controllers manage the specific inputs. With the specific inputs inputted from the user, the Controllers would then determine how the inputs should be stored or managed.

View Tier

In the View Tier UI of our architecture, it consists of the visuals the user is able to see when they first enter the website. In the main page, the website shows a list of courses that users can view. At the top right, users can choose to touch a button to login/sign up. When a user has logged in as a student, they can add courses to the cart. When a student touches the cart, it leads to a page where it shows all the courses the student added.

ViewModel Tier

In the ViewModel Tier, we have the CourseController and the UserController. With these controllers, it helps the model decide what to do based on the inputs the user has put. For CourseController, it heps with features that consists of courses, such as editing and searching of courses. For UserController, it helps with features that consists of the user, such as creating and updating information on users.

Model Tier

In the Model Tier, we have the model and persistence layer. The model layer helps ensure that the classes and structures for the users and courses are correct. The persistence layer checks if the things that are stored and loaded are done properly.

Static Code Analysis/Design Improvements

With the way things are going, one improvement in this website is always to better optimize the code to be more concise and efficient.

Testing

Acceptance Testing

1. Sprint 1: All 7 stories have passed their acceptance criteria test.
2. Sprint 2: All 8 stories have passed their acceptance criteria test.

Unit Testing and Code Coverage

Our unit testing strategy comprises of testing 3 layers. The 3 layers are the persistence, model, and controller layers. Whenever we are unit testing, we always unit test one layer specifically that doesn't depend on another layer for testing. As we test through the layers separately, we are able to determine which layers are passed successfully and which are not. The code coverage achieved from unit testing of the code base is 99%. The team's coverage targets is everything the team has coded for this project, more specifically by the layers: persistence, model, and controller. For model, it's to ensure the classes and structures for users and courses are correct. For persistence, we check if what we store and load is done properly. For controller, we check if the way we access the data is done properly and we are able to prevent unauthorized access. Currently, our code coverage meets all of our targets except the main function due to the nature of the function not being able to be unit tested properly.