Sabanci University

Faculty of Engineering and Natural Sciences

CS 308 Software Engineering
Online Store Project

As a course project, you will be developing an online store, consisting of a website and a mobile application, which interact over the web to an application web-server. The essential details encompassing requirements are given as follows:

- The website and accompanying mobile app will present a number of products in categories and let users select and add the desired product/products to the shopping cart to purchase them.

- You are free to design your store for any kind of product that your team prefers. For instance, it can be an electronics appliance store such as *teknosa.com* or clothing store of a brand and so on.

- The store has a limited stock of items, when the user selects a product, the number of items in the stock must be shown. When the shopping is done, that product should be decreased from the stock and the order for delivery processing should be forwarded to the delivery department, which will process the order for shipping. During order processing the user should be able to see the status as: *processing*, *in-transit*, and *delivered*.

- Users can browse the online store and add products to their shopping carts without logged in to the system. They, however, should login before placing an order and making payment. Once payment is made and confirmed by the banking entity, an invoice must be shown on the screen and a pdf copy of the invoice should be emailed to the user.

- Users should be able to make comments and give ratings to the products. The ratings typically are between 1 and 5 stars or 1 and 10 points. The comments should be approved by the product manager before they become visible.

- The overall user interface of the website and mobile app should have an attractive and professional-looking design, so that people should find the site appealing and trust to buy products. This is not just a website or an app, but the shop window of your store.

- The user should be able to search products depending on their names or descriptions. Additionally, the user should be able to sort products depending on their price or

popularity. If a product is out-of-stock the product must still be searchable, however it should mention out-of-stock and the user should not be able to add it to the shopping cart.

- The user should be able to browse and purchase the products either through website or mobile application. However, the managerial operations of the store are not mandated to be done through the mobile app; nonetheless the main website should provide an admin interface for such managerial tasks. A user can choose to continue shopping by switching between website and mobile app without any data loss.

- A product should have the following properties at the least: ID, name, model, number, description, quantity in stocks, price, warranty status, and distributor information.

- There are three types of basic roles in this system; customers, sales managers, and product managers.

- The sales managers are responsible for setting the prices of the products. They shall set a discount on the selected items. When the discount rate and the products are given, the system automatically sets the new price and notify the registered users about the discount. They shall also view all the invoices in a given date range, can print them or save them as “pdf” files. Last but not least, they shall calculate the revenue and loss/profit in between given dates and view a chart of it.

- The product managers shall add/remove products and manage the stocks; everything related to stock shall be done by the product manager. The product manager is also in the role of delivery department since it controls the stock. This means, the product manager shall view the invoices, products to be delivered, and the corresponding addresses for delivery. A delivery list has the following properties: delivery ID, customer ID, product ID, quantity, total price, delivery address, and a field showing whether the delivery has been completed or not. Last but not least, the product managers shall approve or disapprove the comments.

- The customers shall view the products, search for the products, comment on the products, rate the products, place new orders, cancel existing orders, and return previously purchased products. A customer has the following properties at the very least: ID, name, tax ID, e-mail address, home address, and password.

- A customer should enter his/her credit card information to purchase a product. Credit card verification and limit issues are out of scope of the project.

- A customer shall also be able to selectively return a product and ask for a refund. In such a case, the customer will select an already purchased product from his/her order history within 30 days of purchase. The sales manager will evaluate the refund request and upon receiving the product back to the store will authorize the refund. The product will be added back to the stock and the purchased price will be refunded to the

customer's account. Moreover, if the product was bought during a discount campaign and the customer chooses to return the product after the campaign ends, the refunded amount will be the same as the time of its purchase with the discount applied.

- Since the registration and payment process is sensitive in nature; your project development and programming should reflect the necessary amount of security aware-ONLINE-ness and defensive programming. The various user roles have their own security privileges and they should not be mixed. Whatever your method of information storage (databases, XML files etc.) is, sensitive information should be kept encrypted, so that it's not easily compromised.

- Note that sensitive information includes the following at the very least: user passwords, credit card information, the invoices, and the user accounts. Needless to say, your software is expected to run smoothly and not to display any unexpected behavior while functioning within its normal parameters. Additionally, since this system will serve a potentially large number of users, you should keep concurrency in mind: Your system should be able to handle multiple users of various roles working on it at the same time and retain its normal functionality under such circumstances.

Project Schedule

Date	Project Process
18.03.2020	Sprint 1 Planning and Sprint 1 Starts
25.03.2020	Sprint 2 Planning
01.04.2020	Sprint 1 Review & Retrospective and Sprint 2 Starts
08.04.2021	Sprint 3 Planning
15.04.2021	Sprint 2 Review & Retrospective and Sprint 3 Starts
22.04.2021	Sprint 4 Planning
29.04.2021	**Progress Demo**
06.05.2021	Sprint 3 Review & Retrospective and Sprint 4 Starts
13.05.2021	Sprint 5 Planning
20.05.2021	Sprint 4 Review & Retrospective and Sprint 5 Starts
27.05.2021	Sprint 5 Review & Retrospective

	Final Demos

The table given above presents the project schedule. Note that you will be demoing your project twice; once during the semester as a progress demo and once during the finals as the final demo. You will be graded for both of them.

At the end of the semester, you are expected to deliver a professional-looking, ready-to-use product. Therefore you should learn/use the optimal frameworks for the implementation. You are free to choose any framework, which you feel comfortable with. We, however, encourage you to chose OOP based frameworks, such as C# based *dotNET-Core*, Java based *PlayWithJava*, Python based *Django*, JavaScript based *Node.JS*, etc.

To this end, you will be provided with short pre-recorded lectures about the following technologies:

- PlayWithJava for web applications
- C# with .Net Core for web applications
- VueJS for front-end development
- ReactJS for front-end development
- Essential Server-side web programming, including HTML/HTTP, RESTFUL API, and JSON/XML.
- React Native for cross platform app development
- Apache Cordova for hybrid cross platform mobile apps.

Here are some advices:

1. The project starts in the 4th week of classes. You can provide either a web API based interface to your web frontend and mobile app, or the framework can serve the main website and provide the api for the mobile app. Nonetheless, the chosen backend (server-side) will connect with a database for data storage, handle concurrent http requests and serve web based resources.

2. We strongly suggest you familiarize yourself with the different tools/frameworks that can be used in your projects before starting the implementation. Make sure that you and your team are ready and well versed for their assigned tasks. To decide on the tools/frameworks to be used, take into account the usability, documentation, and the popularity of the framework. Keep in mind that changing a backend server-side framework during the development process will be costly!

3. Teams must ensure that each team member is consistently contributing the project. If, for any reason, you feel that this is not the case, inform the course instructor and the respective TA immediately, so that necessary precautions can be taken in time.

4. Keep in mind the milestones and the amount of resources you have for the project, so that you can plan your development efforts accordingly. We generally suggest that you stay ahead or in-sync with the schedule given above. Note that falling behind the specified schedule, typically contributes to the technical debt you owe, which (especially for short projects) can accumulate quite fast, resulting in failed projects. You should especially keep an eye on the demo dates and work accordingly.

5. If you happen to use the Python-based *Django* framework, please refrain from developing your admin panel of the project with the built-in admin interface provided by this framework as this doesn't require any development effort. We expect you to develop your own admin interface, as this carries a portion of the project grade. If you violate this constraint, you may not receive any points for the admin interface implementation.

Project Grading

The following tables present the **tentative** project grading scheme:

Project Grading	**Points (overall effect in the final course grade)**
Progress Demo	20%
Final Demo	30%

Demo Grading	**Points (20% of the demo grade)**
At least 5 commits per member per demo	18
At least 25 new unit test cases per demo	18
At least 5 new issue reports per demo	18
At least 15 product backlog items per demo	18
At least 30 sprint backlog items per demo	18
Attendance in the SCRUM meetings	10