Sabanci University
Faculty of Engineering and Natural Sciences

CS 308 Software Engineering
Online Store Project

As a course project, you will be developing a website for an online store, which interacts over the web with an application web server. The choice of technology is up to the project team.

The basic requirements for the system are given as follows:

1. The application shall present a number of products in categories and let users select and add the desired product/products to the shopping cart to purchase them.

2. 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 a clothing store of a brand, and so on.

3. The store has a limited stock of items. When the user selects a product, the number of items in 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, an order history page should allow the user to view the status as: processing, in-transit, and delivered. **(8%)**

4. Users can browse the online store and add products to their shopping carts without logging in to the system. They, however, should log in before placing an order and making a payment. Upon logging in, any products previously added to their cart should be retained. Once payment is made and confirmed by the (mock-up) banking entity, an invoice must be shown on the screen, and a PDF copy of the invoice should be emailed to the user. **(8%)**

5. Users should be able to leave comments and give ratings on the products they purchased. Additionally, products must be delivered before a user can rate and comment. 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. However, ratings are submitted directly without any manager's approval. **(8%)**

6. The application should have an attractive, easy-to-use, and professional-looking graphical user interface **(8%).**

7. 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, the user should not be able to add it to the shopping cart. **(8%)**

8. The user should be able to browse and purchase the products through the website. Additionally, the website should provide an admin interface for managerial tasks. **(8%)**

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

10. There are four types of basic roles in this system: customers, sales managers, product managers, and support agents.

11. 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 notifies the users whose wish list includes the discounted product about the discount. They shall also be able to view all the invoices in a given date range, print them, or save them as PDF files. Last but not least, they shall calculate the revenue and loss/profit between given dates and view a chart of it. For loss and profit calculations, the product cost can default to 50% of the sale price for convenience, or it can be specified by the product manager when adding the product. **(8%)**

12. The product managers shall add/remove products as well as product categories, and manage the stocks. Everything related to the stock shall be done by the product manager. The product manager is also in the role of the 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. The product manager can update the status of an order. Last but not least, the product managers shall approve or disapprove the comments. **(8%)**

13. The support agents should provide real-time assistance to customers through a live chat system integrated into the website. Customers can initiate a support conversation from any page, either as guests or logged-in users. If logged in, their profile, cart contents, and order history shall be automatically linked to the chat for context. Customers should be able to send text and/or attachable files like PDFs, images, and videos. Support agents shall have an interface to view and claim conversations from a queue of active customer chats. Respond in real-time with text and attachments. They should be able to access relevant customer details (previous orders, delivery status, wish list items) if the customer is logged in. **(13%)**

14. The customers shall view the products, search for the products, comment on the products, rate the products, include products in their wish lists, place new orders, cancel existing orders, and return previously purchased products. An order can only be cancelled if it is in “processing” status; similarly, it can only be refunded if it is in “delivered” status.

A customer has the following properties at the very least: ID, name, $taxID$, e-mail address, home address, and password. The customer should be able to view their name, email address, and home address on a profile page.
(8%)

15. A customer should enter his/her credit card information to purchase a product. Credit card verification and limit issues are out of scope for the project. **(2%)**

16. 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, provided the product has been delivered. The sales manager will evaluate the refund request, and upon receiving the product back to the store will authorize the refund. In case of refund approval, the product will be added back to stock, and the purchase price will be refunded to the customer's credit card or account. The customer must also be notified—ideally via email—of the refund approval and the refunded amount. 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. **(8%)**

17. Since the registration and payment process is sensitive in nature, your project development and programming should reflect the necessary amount of security awareness 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, invoices, and user accounts. **(2%)**

18. 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. **(3%)**

be aware that,* **(X%) *for the listed features above indicates their contribution in the grading*

Project Schedule

Week	Project Process
Week 4	Sprint 1 Planning and Sprint 1 Starts
Week 5	Sprint 2 Planning
Week 6	Sprint 1 Review & Retrospective and Sprint 2 Starts

Week 7	Sprint 3 Planning
Week 8	Sprint 2 Review & Retrospective and Sprint 3 Starts
Week 9	Sprint 4 Planning
Week 10	Sprint 3 Review & Retrospective and Sprint 4 Starts
Week 11	**Progress Demo**
Week 12	Sprint 5 Planning
Week 13	Sprint 4 Review & Retrospective and Sprint 5 Starts
Week 14	Sprint 5 Review & Retrospective
TBA	**Final Demos**

The table given above presents the project schedule. Note that we are using 2-week sprints, which indicates that we expect to see a new release of your systems every other week. Furthermore, the sprints are overlapping in the sense that in the middle of a sprint we hold the sprint planning meeting to plan for the subsequent sprint. All the SCRUM meetings indicated above (i.e., the sprint planning, review, and retrospective meetings) will be held during the lab hours. To this end, the TAs will schedule 30-minute online meetings for each team that they are responsible for. For the weeks where we have the sprint planning meetings, 30 minutes will be spent on the planning. For the weeks where we have the sprint review and retrospective meetings, the TA will join the last 15 minutes of the review meetings and the first 15 minutes of the retrospective meetings. Therefore, the teams should meet before their scheduled meeting times with the TAs (such that the TAs can join the meeting later) and stay in the meetings even after the TAs leave.

Note further that you will be demoing your project twice: once during the semester as a progress demo and once typically during and/or after the final exams as the final demo. You will be graded for both of them (see below for the grading criteria).

For **Progress Demo**, we will be expecting at least the following numbered features **1, 3, 4, 5, 7,** and **9** from the above requirements to be presented. However, for the final demo, all of the features listed above will be expected.

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 that you feel comfortable with. We, however, encourage you to consider frameworks that are well-documented, align with the project requirements, and are widely recognized for promoting the best practices in software development.

Also be aware that, in our experience, occasionally some teams experience that some of their members don't fulfill their assigned responsibilities and don't contribute as expected, which leads to negatively affecting the performance of the team. If your team experiences this or similar issues, it is the responsibility of the other team members to first communicate with those members to rectify the situation. However, if this does not work, the team should immediately inform the assigned TA and the instructor about the situation, so that corrective action(s) can be taken.

Technologies and Frameworks

Below is the list of suggested technologies that you may want to use in your projects. Note that this list is by no means comprehensive and you are free to choose other technologies as you sit fit.

Backend	NodeJS Django Laravel RubyOnRails Flask FastApi SpringBoot
Database	Mysql PostgreSQL Mongodb Firebase SQLite
Frontend Web	Angular ReactJS Bootstrap JQuery & JQueryUI Flutter VueJS

General Advice

Here is some general advice:

1. You can serve the website directly through the framework or provide a web API for frontend-backend interaction. 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 experience of your team members as well as the usability, the 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 to the project. If, for any reason, you feel that this is not the case, inform the respective TA and the course instructor 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.

Usage of AI

You are free to make use of AI tools while working on your projects. However, it is essential that you do so responsibly. You are expected to fully understand the work produced, ensure its correctness, and verify that it complies with the project requirements.

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%

For each demo, the grading will comprise on three parts:

- 60% from your demo presentation. The percentage points for each feature are given in the feature list above. For the progress demo, the percentages of the requested features will be used as weights, such that the total adds up to 100%.
- 20% of the grade will come from attendance and participation in SCRUM meetings, evaluated on a 0–2 scale for each meeting based on punctuality, contribution to discussions, and achievement of previously assigned goals.
- 20% from your development activities, which are further distributed as:

Demo Grading	**Points (20% of the demo grade)**
At least 5 commits per member per demo	20%
At least 25 new unit test cases per demo	20%
At least 5 new bug reports per demo	20%
At least 30 sprint backlog items per demo	40%