week_1_wa

ECOM stores, also known as e-commerce or electronic commerce stores, are online shops where products are sold on the internet. For example, Amazon, AliExpress, Tmall, and Flipkart are well-known ECOM businesses.

Businesses have been switching from “bricks to clicks” and becoming online-only stores or existing online and having a physical walk-in store. For example, Payless is a shoe store that closed all its physical stores in the US after declaring bankruptcy. Post bankruptcy, Payless made a comeback by becoming a fully online ECOM store. The company plans to open physical stores again in the US and still maintains physical stores abroad in other countries.

Choose a brick-mortar store and help this store transition from “bricks to clicks.” The store will be fully online and you will be managing its database, securing important data, and developing a mobile shopping application.

For this assignment,

1. What is the name of your store?

I was thinking about a name. I guess it is important. On the other hand, it should catch the attention, be easy to remember and probably have some meaning. Our online store name should comprehensively project the future. So, the name is going to be Futurama.

2. Decide what type of store this is. Is it a clothing store or maybe an online bookstore? Include all this information in your submission.

During the pandemic, we “thrilled” lockdowns, when everybody was sitting at home. Weather conditions like hurricanes or snow also require people to be in shelters. Physical stores as we see have a number of drawbacks.

Next question could be, how many customers will we serve? Is our store located only in town or does it have branches? Let’s choose we are going to sell books, like amazon does, or started with. To have many physical stores is too expensive, at least renting cost multiplying on number of locations. So, having an online store is a better option.

Other two questions are also quite important. Logistics and delivery. Where books will be stored, reordered and how many of them to keep to be available. What types of deliveries? By car, by bike or/and by drones?

3. Decide and justify your SDLC approach(es).

A couple of words regarding choosing a software development life cycle. Let's start with understanding what we want to achieve. Our goal is to deliver an online store on time (we would like to start selling our books or at least gather orders) with high quality, so that our customers will be satisfied. A store should be well tested and ready for production.

If we choose waterfall methodology it will require 100% planning and following everything. Our problem with that, until we plan and document every aspect of our store we could not make it working. Again, we would like to start selling books as fast as we can. More suitable method could be agile.

With that approach, our goal to plan is only for two weeks, including testing and deployment to production. It gives us the ability to get feedback from the customers quickly and change if needed. During the writing code we also add as part of the process writing unit tests, i.e follow TDD test driven development paradigm.

4. Apply the SDLC method(s) to this scenario.

First we will gather the requirements. Next, I will analyze them to understand and sort what is more important or can wait. Taking small chunks of these requirements we start to design. When it is done, we are going to write code with a test. If it is a database, we will create a schema, relationships and database type. From the beginning we should decide what our architecture will be, a monolith or microservices.

So, summing up. We will plan each spring, based on the customer's priorities or internal dependencies. Write code and test it. Each commit will automatically redeploy our services.

5. Decide which programming language(s) you are going to use.

Our architecture is microservices. Java or Go best bet. Another consideration is how easy we can hire specialists. Next, what are the available frameworks and libraries? Communities and support. Also an important aspect is maturity of chosen frameworks.

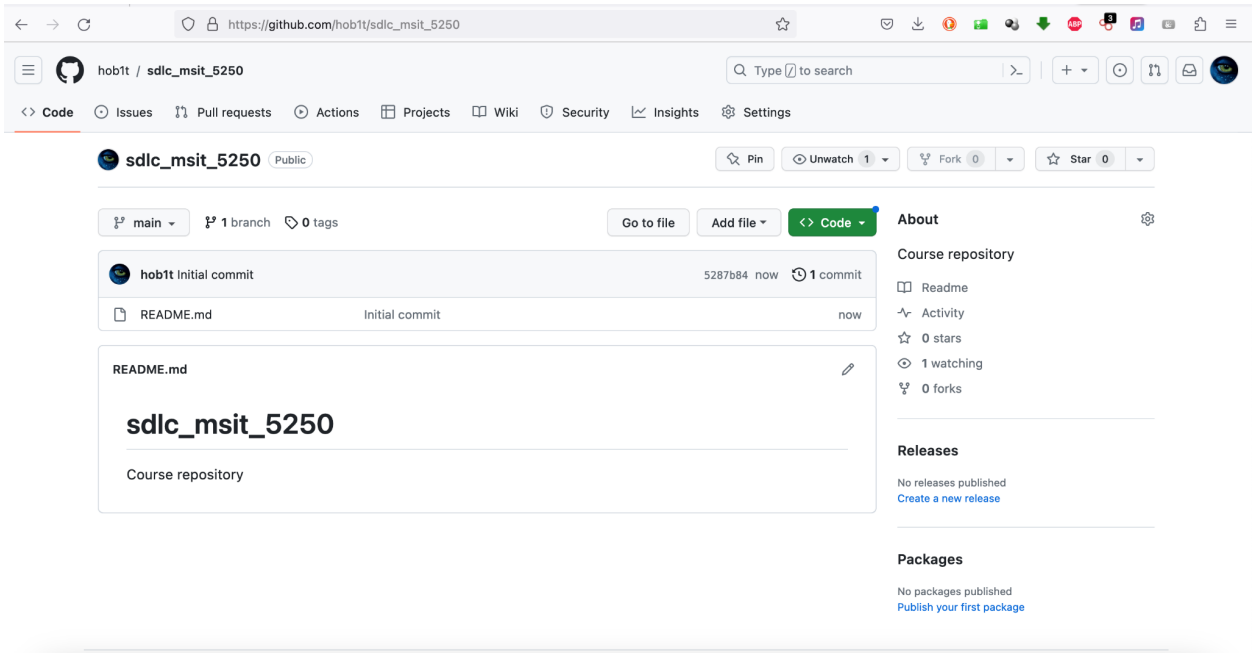
6. List all the requirements you will need to meet your goal (scenario). For example, securing customer data (credit card information, shipping and billing address, etc.), creating a mobile app, managing the retailer's database, and so on.

List of requirements:

1. UI user friendly, responsive, fast and pragmatic
2. Security - two factor authentication, rate limiter, gateway to avoid DDoS attack, avoiding sql injection, encrypt everything, use https, not guessable passwords
3. Availability - 24/7, SLA 9999
4. Scalability - our online store should be able to scale up or down depending on the number of customers.

5. Reliability - every service should be reliable, i.e. every request should get exactly what it should get.
6. Database - should be replicated, sharded, clustered and backuped. Backups every day + deltas every hour.
7. Monitoring and traceability.
8. CI/CD - Jenkins and argo.
9. Documentation - md files + git pages
10. Cloud provider - AWS/Google/Azure
11. Fraud prevention and detection - for example, making payments with stolen credit cards, or identity.

7. Your submission should include a screenshot of your GitHub account (created in a previous activity) and a document with your approach to this scenario.



References:

1. Tillie Dimentiou, (2022, November 7). Types of retails stores: What are they? https://www.eposnow.com/us/resources/types-of-retail-stores/
2. Security concerns by PCI https://listings.pcisecuritystandards.org/pdfs/Small_Merchant_Guide_to_Safe_Payments.pdf
3. Git repo: https://github.com/hob1t/sdlc_msit_5250

4.