Oregon State University
Ecampus

CS361: Assignment 11: Portfolio

Overview

Implement your Sprint 4 plan, then make a **video** showing your **entire individual project**.

Final requirements for your individual project:

- Reflects all 8 CSH
- Free of all 11 code smells
- Reflects your 3 quality attributes
- Calls a teammate's microservice programmatically

In addition, you'll create a **reference sheet** covering the main CS361 topics. The purpose of this is to (1) revisit the concepts you learned, (2) have a way to quickly refresh your memory in the future, (3) give you "talking points" for interviews or in case you ever need to explain software engineering concepts to others.

Instructions for Part 1 (of 2): Implementation & Video

Complete each item below by replacing the highlighted text (**Usability note**: double-click the text to select it).

You'll also need to create a **video**.

1. What were your **Sprint 4 user stories**? You'll need to implement these, and show how they work in the video.

	User story name (e.g., "Login")	**As a… I want to… so that… format**
1	Alex	As a novice investor I don't want to see any complicated technical data. I just want to see a simple one line answer that tells me if I should BUY/HOLD/SELL a stock.
2	Natasha	As an expert in stocks, I want to be able to see a trend of the technical indicators so that I can deep dive into each indicator.

2. What are your **quality attributes**? In your video, you'll need to show that the quality attributes are reflected. Remember that your quality attributes must include maintainability and usability.

maintainability
usability
Flexibility

3. Create a **video** showing your **individual project**.

   Requirements for video:
   - Shows each feature (including the two new features you implemented this Sprint)
   - Explains which features are new this Sprint
   - Explains where teammate's microservice is called
   - Explains how each of the **8 CSH** are reflected
   - Explains how each of your **3 quality attributes** are reflected
     - Maintainability is covered by scrolling through your code
     - Usability is covered by explaining how the CSH are reflected
   - **Scrolls slowly** through your individual project code, which should be free of all **11 code smells**

- **8 minutes or less**

Given your video cannot be more than 8 minutes, avoid going into unnecessary detail.

Link to my video: Video Link
Link to my gitHub release: gitHub Link

Instructions for Part 2 (of 2): Reference Sheet

Complete the reference sheet template on the next page by replacing the highlighted text (**Usability note**: double-click the text to select it).

Use your own words!

CS361 Reference Sheet

Concept	**What it is (1+ sentence)**	**When it's useful and/or NOT useful (1+ sentence)**
Agile	Agile is a software process model that is used to manage developing software projects.	Agile values the following: • Individuals and interactions • Working software • Customer collaboration • Responding to change I think Agile is popular in the tech world because of it ability to respond quickly to changes in the customer demands. I come from a different industry (chemical manufacturing) and there has never been a mention of Agile as a project management style. I think this is because project are much more rigid in the chemical manufacturing world and if specifications were constantly changing it would be difficult to manage a project.
Cognitive style heuristics	CSH are principles for interaction design that is used to find usability issues within a piece of software.	I think CSH can really improve the usability and inclusivity of your software by applying the eight principles outlined in our software. The only time I can think of that it may not

		be as useful is when you are developing a piece of software for yourself or a very small group of people that may want the software to behave in a very specific way. I think anytime you are designing a piece of software for a broad audience the CSH principles will help improve the usability and inclusivity of your programs.
Functional requirements	Specify the functionality that your piece of software must have to meet the customer's requirements.	I think functional requirements are almost always going to be the basis of your project. As a developer you need to listen to the feedback of your customer and provide them with the functionality they desire. If some functionality seems wrong or could be improved, the developer still needs to get the approval of the customer before proceeding.
GitHub	GitHub is an open-source collaboration tool that can be used to store your coding projects.	I think any time you are working on a team and have to have collaboration on the software between multiple team members it is good to have a central location to store all the files. In addition GitHub is great for managing branching and updates to various files by multiple team members. The only reason I wouldn't use this type of repository is if I didn't want to collaborate on a piece of software (ie. It is a standalone project or something).
Microservices architecture	Microservice architecture is a type of software development strategy where a program is run as collection of independently operating processes.	Microservice architecture is great when you want to minimize the risk of a complete software shutdown if one piece of code malfunctions, needs to be restarted, or needs to be updated. Because the process all run independently, assuming that microservice is not critical to the operation of the overall program, losing one service will not crash the bigger architecture. Microservice architecture is great for large projects where several APIs would all be operating at once.

		However, as Uber experienced, a company can go overboard on microservice and end up becoming less efficient if they create a microservice for every small task.
Non-functional requirements	Specifies the qualities that a piece of software should have to perform at the desired level.	I think any time you are developing a piece of software for a customer it can be very helpful to have a discussion about the non-functional requirements of the software. This will help manage the expectations of the customer and align the developer and the customer of the ”vision” of the software.
Paper prototyping	A low-cost way to get feedback on your interface design and begin discussing the outline of your design with your clients.	Great for initial discussions with the software client, this method of getting feedback on you UI design is a quick, low-cost way to present initial ideas and brainstorm with a client. I probably wouldn’t using paper prototyping to unveil a project to the CEO of a company though.
Quality attributes	Attributes that describe non-functional requirements for the software. Describes how good a piece of software is at certain tasks.	Quality attributes fit into the following categories: • Reliability • Efficiency • Integrity • Memorability • Flexibility • Interoperability • Reusability Having upfront discussions on the goals for each of these quality attributes with the client upfront can help the developer and the client stay on the same page as to the goals and how well the piece of software operates.
Scrum	Part of the Agile framework. Scrums are used to develop and sustain complex products.	Scrums work well for organizing and aligning team member’s goals during a sprint. Scrums can help teams stay on track during a sprint and the Scrum Master can help the overall project stay

		on course. I wouldn't use the Scrum framework to micromanage people's workloads.
Sequence diagrams	An interactive diagram showing how different participants collaborate	Use this when you want to have a visual aid to show people the relationships between two abstractions in the software. I think the most important part of sequence diagrams is to know your audience. If you need to give a high level diagram, present a high level diagram. If you need a lot more detail, consider other visualization tools.
Spikes	A quick deep-drill into various options to find the best option for the application.	Use a spike when you have several options (like which dashboard to use) and you want to find which one is best suited for your needs. Don't go into too much detail with each option, just enough to understand how well each option will suit your purpose.
Status updates	Routine communication between colleagues and/or stakeholders.	Status updates should be used when you feel like routine communication can help a team stay informed and on task. Don't use status updates for little things that could be communicated over email.
Task management system	A tool (Jira, etc.) that can be used to organize tasks and helps execute a sprint.	Task management systems are great for when you have a large amount of tasks that you need to complete and you need a system to keep track of them. Don't use task management systems to keep track of things like emails or small, personal to-do lists.
Team ground rules	Rules establish usually at the start of the formation of a team. Ground rules outline things like communication channels and appropriate responses to inaction, etc.	Team ground rules can be an effective way to make all member of a team pull their own weight. It is a great accountability tool. Don't use ground rules for small, unimportant teams.
User stories	A way to brainstorm and communicate functional requirements of a program.	Use a structure like… "Give that I am a {insert} I would like to see the program do {insert}". Using this method can help you identify what user's want out

		of a program and help you develop a better program. Don't over use user stories to where every decision requires going back to a user story.

Submission

PDF or Word format via Canvas. **Link to video** in the document.

There are two options for recording/uploading your video:

1. Record and upload following the instructions in Canvas > "Start Here - ReadMe First" > HOW TO: Create and Upload a Video

   OR

2. Record using the technique of your choice, upload to YouTube, set as Unlisted

Grading

You are responsible for satisfying all criteria listed in the Canvas rubric for this assignment. You will **NOT be able to revise** this assignment.

Questions?

Please ask via Ed so that others can benefit from the answer.