17-313: Foundations of Software Engineering

Fall 2022 Midterm Exam

Michael Hilton, Rohan Padhye, Christopher Timperley, and Daye Nam

Name: _____

Andrew ID: _____

Instructions:

- Not including this cover sheet, your exam should have 13 pages in total: 11 pages of questions, and 1 page of appendix. Make sure you're not missing any pages. Write your full name and **Andrew ID** on at least this page, if not all others.
- Most questions in this exam refer to a scenario, described in the appendix. We encourage you to read it first. You may detach the appendix; the appendix will not be considered in grading, thus keep your answers on the question pages.
- Write concise, careful answers. Short and specific is much better than long, vague, and rambling, and grading will reflect this. You have enough time to write clear and readable responses. Bullet points and phrases are acceptable. Spend time to consider how best to present your answers, including citing examples to make your points more concretely.
- Clearly indicate and write your answers in the space provided below each problem. We cannot give you points for answers we cannot find or read.
- The exam has 6 multi-part questions with a maximum score of 80 points. The point value of each problem is indicated. We planned the exam to allocate approximately one minute per point.
- If you do not know the answer to a question, you may leave it blank and receive the indicated number of "No-nonsense" points (abbreviated *NN* for the rest of this exam). This may be an improvement over the zero (0) points that may be awarded an incorrect answer. The number of available *NN* points varies per question. If you start to answer a question and then change your mind, you may invoke this rule by crossing out your answer and prominently writing "LEAVING THIS BLANK." We will not read partial answers to questions on which you invoke this rule.
- You may consult one sheet of paper (8.5x11in, double sided, typed or handwritten) with notes. You may not use books, a calculator, cell phone, laptop, or any other electronic or wireless device.
- Good luck!

Question 1: Process and Planning (12 points)

Appendix A describes a scenario that will be referenced throughout the exam. You should familiarize yourself with the scenario before you continue.

(a) (4 points) *(1 NN)* The leadership of BobaFetch (Appendix A) has decided to create a public API which will let others use their fleet of Droids as a Service (DaaS). They will charge by the time, but also by the API call. This public API will not need to do anything with the droids that the current API cannot do, but it will need to instrument everything in order to charge the DaaS customers correctly. To ensure your team is making progress, you ask the team to define the intermediate milestones. Describe one good intermediate milestone for the subsystem.

(b) (4 points) *(1 NN)* The Boeing case study involved a discussion of how process and culture concerns led to software failures. Briefly describe one of those issues.

Question 1 continues...

(c) (4 points) *(1 NN)* What is one process intervention you could implement in your company to avoid the same kind of failure as in the previous question?

Writing below this line is permitted but discouraged.

Exam continues...

Question 2: Requirements (17 points)

(a) (9 points) *(1 NN)* You find that no one is evaluating risks for BobaFetch. Following the process you learned in your undergraduate Software Engineering course, you decide to follow the process of identifying risks, assessing them, and developing a mitigation strategy. Populate the table below with three risks with differing levels of severity.

	Description	Severity	Probability	Mitigation Strategy
Risk 1		HIGH		
Risk 2		MEDIUM		
Risk 3		LOW		

Question 2 continues...

(b) (8 points) *(2 NN, 1 per story)* One way to evaluate the quality of user stories is according to the INVEST principles (Independent, Negotiable, Valuable, Estimable, Small, Testable). For each of *estimable* and *testable*, write a user story that is generally sensible in the Boba Fetch scenario, but violates that principle. Afterward, briefly explain why this user story violates the principle and provide an improved version of the same user story. Use the common "As a [role], I want [function], so that [value]" template. NOTE: for each user story, you should write the user story targeting the prescribed role.

i. User story violating the *'estimable'* principle:

Briefly explain why it violates the principle:

Fixed version of the same user story:

ii. User story violating the *'testable'* principle:

Briefly explain why it violates the principle:

Fixed version of the same user story:

Question 3: Metrics and QA (16 points)

Upon inspecting the original BobaFetch implementation, you discover that the existing codebase is a real mess! You can't find any tests or documentation, and the source code is difficult to navigate and understand. Unsurprisingly, an expensive and embarrassing failure occurs in production: Certain customers are being charged for items that are never delivered! To make matters worse, the mobile app seems to crash if the customer tries to request a refund for those items.

(a) (4 points) *(1 NN)* Your CEO demands that no new work happens until customers are able to get their refunds. Describe what steps you could take to identify, understand, and address the root cause of this particular failure.

(b) (4 points) *(1 NN)* Describe and justify a plan for avoiding failures like this in the future.

(c) (4 points) *(1 NN)* As part of an effort to avoid similarly embarrassing failures in the future, management has tasked your team with writing tests for the original codebase. Given your limited time and resources, how do you plan to test the system?

Question 3 continues...

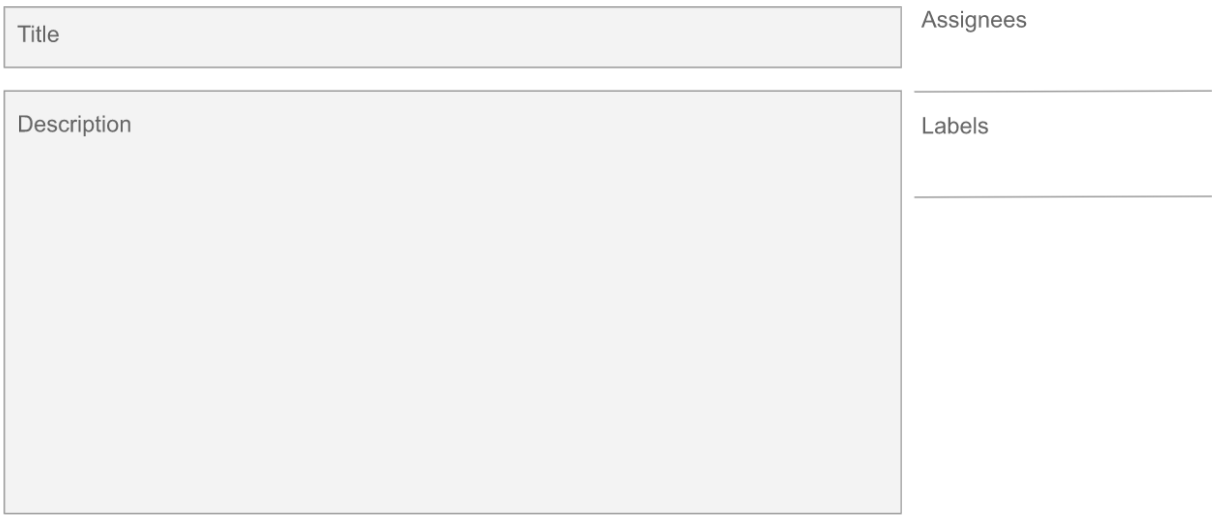
(d) (4 points) *(1 NN)* Due to a recent influx of cranky reviews, your manager now wants you to focus on improving the app's usability by 2×. In this discussion, your teammate argues that "overall usability" cannot be verified or measured using metrics and must instead be left to the intuition of the designer. Your teammate says "none of these options really capture the core idea of 'usability'; it's pointless!" Do you agree or disagree? Why or why not?

Writing below this line is permitted but discouraged.

Question 4: Communication (12 points)

BobaFetch recently hired many software engineers. Unfortunately, they do not seem to have taken 17-313, and keep submitting issues that are barely useful. As a senior software engineer at BobaFetch, you decided to take the initiative in writing an onboarding document for the new software engineers.

(a) (4 points) *(1 NN)* In the onboarding document, you want to provide an example of a bad issue that can highlight the need for a well-composed issue. Write an issue in the following form. You can refer to any hypothetical tasks or people, as long as they are in the context of BobaFetch. Your issue should have at least 4 ways it could be improved.



(b) (4 points) *(1 NN)* Explain why the above example needs improvements, explicitly discussing the 4 ways in which it could be improved. For each area, provide a concrete direction on how to improve.

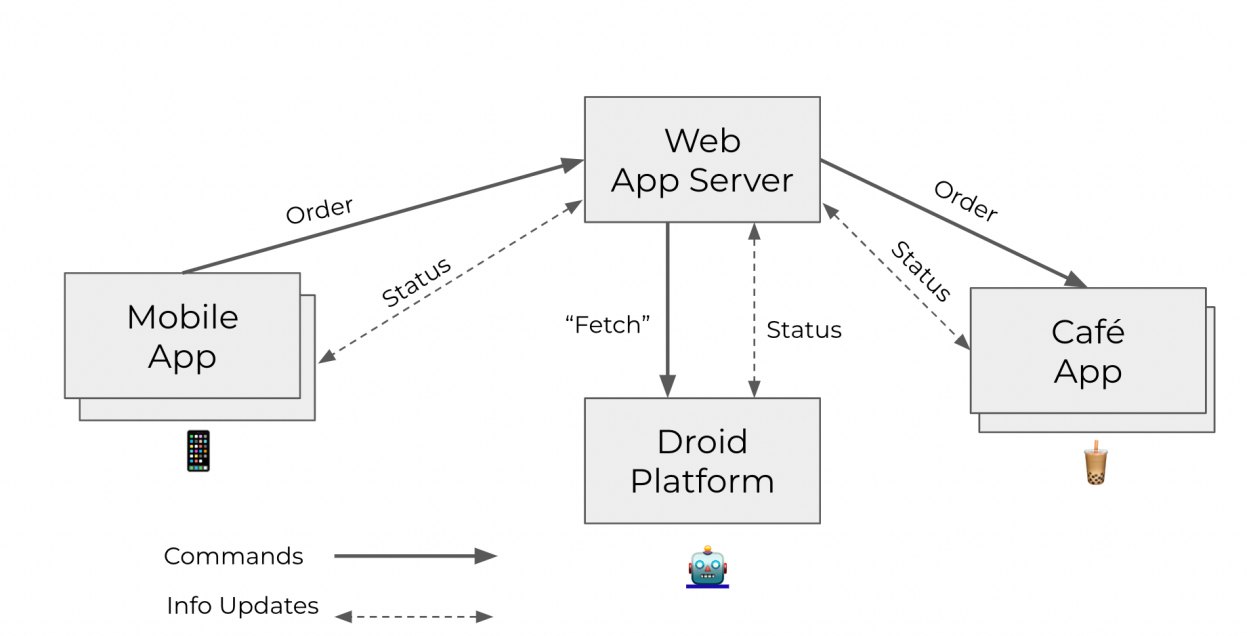
Question 4 continues...

(c) (4 points) *(1 NN)* Modify the bad issue example and write the improved version in the following template to compare with the bad example in the onboarding document.

Title	Assignees
Description	Labels

Question 5: Architecture (23 points)

The original implementation of BobaFetch has several software components: (a) the customer's mobile app, (b) a web app server that accepts orders and forwards them to cafes, (c) a point-of-sale app at the cafe that receives orders, and (d) the software that runs on BobaFetch's droids. One view of the overall architecture is shown below. Each component is a monolith running on a different device.



As the popularity of BobaFetch has grown, the system is bottlenecked by the limitation on the number of available droids. Pittsburgh being a bustling hub of droid research, there are many start-ups who own a fleet of advanced droids. Your CEO has recently struck a deal with several such droid companies to contract some of the deliveries to them. The pitch is that when a boba order is initiated by a customer, BobaFetch will broadcast the delivery details and a fee that it is willing to pay (called the *bounty*) for fulfilling the delivery. The fee is variable and can depend on several factors such as distance between café and customer, time since the order came in, the number of orders currently backlogged, etc. The first company to accept the fee is awarded the contract and must fulfill the delivery by dispatching a droid (called the *bounty hunter*).

In the meeting for planning this transition, you are asked for your opinion on architectural decisions. There are two options going forward:

- **Option 1:** Stick with the monolithic architecture, but manually add bounty hunter droids owned by other companies to the list of droids to contact when a customer order comes in.
- **Option 2:** Transition to a microservice-based architecture, with different services for tracking orders, customer payments, bounties, etc. Allow external bounty hunters to directly communicate with relevant microservices to accept orders.

Question 5 continues. . .

(a) (9 points) *(3 NN, 1 per quality attribute)* Architectural reasoning often involves tradeoffs, often with respect to quality attributes or non-functional properties. Identify three important quality attributes that might be affected, and describe (briefly) how each option affects it.

Quality	Effect of Option 1	Effect of Option 2

(b) Let's assume that you have taken the decision to transition to microservices.

i. (6 points) *(1 NN, must leave all parts blank)* Your existing application was developed monolithically. How would you transition to microservices? Do you recommend implementing new features as microservices and slowly transitioning, or re-writing the existing application from scratch right now? Give one argument in favor of each position and make a recommendation.

Argument for incremental transition:

Argument for full rewrite:

Recommendation (with brief justification in the context of the scenario):

Question 5 continues...

ii. (8 points) *(1 NN)* Draw an architecture diagram for a microservice-based instantiation of BobaFetch with the new entities in mind. We don't require a specific format for the diagram, but it should be clear and understandable.

End of questions.

Question 6: Bonus fun question (0 points)

(a) (0 points) Draw any picture you like.

Question 6 continues...

A Appendix: Scenario description

You work for an on-demand beverage delivery company called BobaFetch. Customers in the Pittsburgh area download the BobaFetch mobile app and use it to order bubble tea and other drinks from local cafés. They pay for their drinks on the app at order time. Once the beverages are prepared, they are home delivered to customers by droids (see below). The droids are clumsy but secure, and are universally loved by locals. BobaFetch owns a fleet of droids who can be dispatched to cafés as orders are received and then routed to customers' homes.

Figure 1: Artwork by David Hilton, age 11

Figure 2: Artwork by James Hilton, age 8

End of exam.