T305 - Mouthpiece

04/10/2023

EEL4914C

# Scholarship in Practice

## Decision Critiques:

### 1. Code structuring

What did we learn: Structuring our codebase for our app using an MVC design pattern helped us for the most part in knowing what to debug when bugs arose, not only that but also adding additional features. Using the concepts discussed in software engineering classes, we've split our codebase into Models which connect to our database, Views which are what the user experiences and sees, and the Controllers which connect UI actions from the views and build the necessary logic to send to the Models. We've learned that this design pattern does in fact work. Organizing our code in this way is one of the best decisions we've made in efficiently building our app.

Cause of suboptimality: Although MVC design patterns are very effective for scaling applications efficiently, towards the end of designing the app, while adding additional functionality, we've started to stray away from the ideal code structuring that we would've liked. For instance, adding geolocation functionality got a little cluttered and messy in the codebase thus making it hard to read and debug. All of this could've been avoided with better planning on code structuring.

### 2. Poor planning with project requirements

What did we learn: For one of our project requirements, we wanted to add tournament functionality within the app to provide a competitive factor that can be tested amongst the users in the app. Being ambitious and naive of the task at hand, once developing the app we soon began to realize that this functionality would be very difficult to implement with the allotted time and knowledge that we had with Swift and other technologies needed. A tournament manager is a project by itself. We've learnt that not all project requirements will be met like we would've hoped but we've tried our best in building the other bits of functionality with the resources we had.

Cause of suboptimality: The main reason why this didn't work out the way it should was because of our poor planning with time management. Allotting time for the back-end engineers to provide the framework and functionality behind the app for the front-end developers to polish up, would've been optimal. Going back, even structuring out how exactly the tournament functionality would've worked would've given us some idea on how to properly execute this task. Unfortunately, with how the project tasks were laid out, this would've been very difficult

to do. All in all, we should've had feasible requirements within our ability to perform the necessary functionality at first, applying project management skills from our Engineering Design Concepts class. Going over the team about how to implement these requirements would've been the next step. Then, with the project development stage that would've given us a good idea with the proper planning needed to get it done.

### 3. Poor implementation of Figma design into xcode

What did we learn: We would have liked to make our actual app look and function more like our Figma design. Our Figma design for the UI was something we as a group were really proud of but we could not get it implemented in xcode quick enough. Our app functioned like we had wanted and all buttons did what they were supposed to do but the look and feel wasn't there yet.

Cause of suboptimality: We could have tried doing one app page at a time. For example, maybe instead of having the whole app planned out and created on Figma as a design, we could have designed the home page, then coded the home page in xcode exactly as we have it on Figma. Then we could have designed the discover page and coded the discover page exactly as we have it on Figma and continue this for the rest of our app. This would have allowed us to have an app looking how we would like even if the whole app functionality isn't finished yet.

### 4. Too much time allocated for planning.

What we did learn: Having a well-crafted plan can be advantageous; nonetheless, excessive time allocation towards planning and learning may prove to be counterproductive. In our specific case, we dedicated an ample amount of time, the entire fall semester to be precise, towards acquiring proficiency in programming Swift through Udemy. This was a commendable effort aimed at ensuring that we possessed the requisite skills to develop the desired application without encountering any impediments. Regrettably, despite our preparations, roadblocks did manifest themselves.

Cause of suboptimality: One potential cause of suboptimality in our project was an unrealistic timeline for learning Swift programming and transitioning into app development. The decision to dedicate an entire semester to learning Swift on Udemy may have been well-intentioned, but it resulted in a significant delay in starting the app development process. This delay could have been exacerbated by a lack of prior experience in Swift programming and a failure to accurately estimate the amount of time needed to become proficient in the language. As a result, we may have set an unrealistic timeline for ourselves, leading to suboptimal outcomes in the project.

### 5. Task allocation imbalances and incompletion.

What we did learn: In the process of allocating tasks for project completion, we utilized the collaborative platform Microsoft Teams. This platform facilitated the efficient execution of daily scrum meetings, which enabled us to assign and monitor individual responsibilities. However, we encountered challenges in achieving optimal outcomes due to certain tasks remaining incomplete, and an uneven

distribution of tasks among team members. As a consequence, these shortcomings hindered our ability to meet the project deadlines in a timely and effective manner.

Causes of suboptimality: One potential cause of suboptimality in our project was an inadequate task allocation process, despite utilizing Microsoft Teams. The lack of a clear and effective strategy for assigning tasks and responsibilities may have resulted in some tasks being left incomplete. Additionally, the uneven distribution of tasks among team members may have led to some individuals being overburdened while others had insufficient workload. This suboptimal allocation of tasks could have ultimately impeded our ability to meet project deadlines in a timely and effective manner.