Leejin Kim

Richard Jullig

Introduction to Software Engineering

31 May 2022

Athena Project Reflection Essay

Being one of the people who like to read physical, hardcover books, I thought it would be extremely convenient if there is an app that can bookmark the book and save the notes I left for the impressive quotes. Therefore, I decided to contribute to building an Athena app, which can help users to keep track of the books they've read over the years. For this project, I had Sai Kambampati as product owner, and Nitya Bhupatiraju, Kevin Crawford, and Logan Thompson as team members. Its intention is to give the same technological features that EBook customers have to users who want to read the hardcover book instead. To build this app, our team used SwiftUI as a framework, Xcode as IDE, google books API as web API, Firebase as backend, and Github as Git repository. We had the frontend database that have a root tab view and the following 4 tabs representing 4 screens: home, search, notes, and profile. Each screen was in MVVM app architecture that contain model, vie, and view model architecture.

Our goal was to make Athena allow users to get back into the habit of reading, keep track of everything they had read and was going to read, search for books to add to a wish list and create notes for each book. In order to accomplish these goals, we set up six user stories: let users search for books by title and author, update progress on books being read by bookmarking them, create accounts, take notes for each book, categorize the books by already read, currently reading and want to read, and rank and review books they've read. The user stories were done by implementing the codes for different SwiftUI files representing 4 screens.

We used Github as a git repository to create the branch and SwiftUI files. In order to prevent any merging conflicts, we created a separate development branch and send a pull request to the product owner whenever we need to push codes. In order to distribute the workload equally, each of us worked on different SwiftUI files representing different screens.

Among those screens, I took charge of working on the home and notes screens. As my macOS does not support Xcode 13, I worked in pair programming with Nitya. We met up at least twice a week and went over the tasks on the Trello. Working in pairs helped us speed up the overall working process and helped us to figure out the difficulties we were facing easily. For sprint 1, we made onboarding screen 2 and set up the home page. For sprints 2 and 3, we worked on the notes view features such as adding a feature to create a new note, deleting, saving, or editing notes and linking saved notes with firebase as the backend. For sprint 4, we created the dropdown menu to sort the books the users have read by title and author. As a scrum master for sprint 4, I updated the tasks on Scrum Board, which were allowing users to review the books they've read and fixing any bugs such as log-out crashing bugs.

As a team, we had 3 scrum virtual meetings every week and one in-person meeting biweekly. Throughout the scrum meetings, we could hear each other's updates and progress being made and discuss any issues or impediments that need to be fixed by sharing feedback. Meeting in person as a group allowed us to help each other efficiently. Since the team members except for Sai, who is the product owner, were new to Swift programming language, we needed to ask any detailed debugging questions. If we had worked virtually on the project, we would have had a delay in our progress by debugging minor issues. We also used Trello as a Scrum board to keep track of the progress and distribute the tasks equally. Before every sprint, we had sprint planning meetings to write the high-level goal for the sprint and work out details of user stories. After each sprint, we wrote down the sprint report, which let us list the actions to stop,

start, or keep doing. Scrum master built the burn-up chart on the report and this burn-up chart visualizes the days that have the most work done. For most of the burn-up charts for each sprint, we have the most tasks during the in-person biweekly meeting, so we decided to keep having this meeting.

Though I didn't face any specific differences of opinion or argument as part of a team, I faced challenges in coding. At first, working on details such as layout, the color of the images, etc was challenging. As a solution, the team decided to stop working on the minute design details like spacing and colors since this is secondary to functionality and can always be fixed in later sprints. Also, figuring out how to link firebase to note view was also time-consuming but was a great opportunity to study how linking backend to frontend database work in general.

Though there were some challenges, I had valuable lessons throughout the project. I not only learned how to add app features using Swift language such as creating a progress bar and the dropdown menu for sorting, but also how to facilitate good teamwork and communication. During the sprint meeting, I brought up some ideas for improvement such as stopping having late-night meetings and starting scheduling more in-person group work sessions, since these are productive at immediately answering questions. Estimating the time for each task and setting the story points led us to assign tasks evenly. For any future app-building project, I would use test flight betas through automatic Xcode Cloud builds. Personally, I wished that we had more time to work on social features, rating books, and Niche UI forces. Thus, I am anticipating developing the app even after the course. I had so much fun working with enthusiastic team members who were willing to help each other and dedicate themselves to a project.