Project Evaluation

Planr, an Agile Project Planning Application

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of Master of Software Engineering

Prepared by

Michael Blakeman

CIS 895 – MSE Project

Kansas State University

Table of Contents

[1. Introduction 3](#_Toc88225900)

[2. Lessons Learned 3](#_Toc88225901)

[3. Project Effort Breakdown 3](#_Toc88225902)

[4. General Takeaways 3](#_Toc88225903)

# Introduction

This document will go over the lessons learned, project effort, and other general takeaways.

# Lessons Learned

There were many lessons learned when planning and developing the Planr application. I found that it worked out quite well to have the data models defined up front. Once all the data models were defined it was easy for me to design the planning algorithm to work with them. I also found that it was incredibly helpful to work off a UX document from the very beginning. This is typically how my team does development and it was very effective to have the UI and UX defined such that I could create the data models and visualize how the UI components could be broken down into reusable components.

SwiftUI had a slightly learning curve but once I did a couple of hours of research and proofs of concept, I was able to pick it up rapidly. It was nice to be able to have a live preview of the UI I was implementing and displaying dummy data was paramount to making the UI come together as quickly as it did. The UI bindings in SwiftUI are also pretty “magical”. Being able to reflect change in the UI with bindings just worked with minimal code to hook them up.

I discovered during the development process that the Realm documentation was lacking in many ways. There have also been some major updates to Realm and their SDK documentation didn’t reflect the updates to their APIs nor did they effectively provide good example documentation for updated SwiftUI versions as well.

Going along with more negative lessons learned, navigation within SwiftUI for macOS was unintuitive and convoluted. The framework setup for macOS page navigation for SwiftUI forced developers into requiring a split view with a navigation bar on the left-hand-side. This did not fit the aesthetic of the application I was wanting to create so I had to implement an open-sourced solution to aide in simple page-to-page navigation. Once I pulled in that small library it worked as I would expect it to.

I also learned many things about myself behaviorally. As it turns out I am prone to being very distracted with two main things during development. Firstly, I will spend inordinate amounts of time trying to refactor things to be the most efficient use of code, or play “code golf”. I would find myself trying to refactor items while I was actively developing a section of code. Secondly, I would get distracted on UI polish that may not have been the most effective use of time. The priority of completing code work and features should have taken precedence but there were times where I there would be an imbalance of effort placed on making a reusable, aesthetically pleasing, UI component instead of completing a feature in code.

# 3. Project Effort Breakdown

|  |  |  |  |
| --- | --- | --- | --- |
| Activity Type | Project Time (Hours) | Quantity | Productivity |
| Source Code | 59.5 Hours | 2650 SLOC | 44.5 SLOC/Hour |
| Documentation / Design | 38 Hours | 14 Documents | 0.37 Documents/Hour |
| Research / Testing | 10 Hours | - | - |
|  | 107.5 Hours |  |  |

# Shape Description automatically generated

# 4. General Takeaways

Overall, this project was very fun to work on. My goal was to “develop an application to aide in project planning and roadmap development for software projects.” I think that I achieved my goal. Prior to having this application, I was manually using several tools to come up with a roadmap for stakeholders. During the manual process I had for roadmap development I would not have come up with as efficient roadmap plans as my Planr application would. It does a good job of abstractly taking in resources and assigning them out based on feature priority. It fills in work blocks in ways that I would not have thought to do, but ultimately makes it more efficient. I plan on wrapping up the TODO items and deploying this to my peers at Garmin to help them in resource and roadmap planning as well.