Vision Document

Planr, an Agile Project Planning Application

Version 2.0

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. Project Overview 3](#_Toc83632709)

[1.1 Purpose and Motivation 3](#_Toc83632710)

[1.2 Goals 3](#_Toc83632711)

[1.3 Risks 3](#_Toc83632712)

[1.4 Constraints 4](#_Toc83632713)

[2. Requirements Specification 4](#_Toc83632714)

[2.1 Functional Requirements 4](#_Toc83632715)

[2.1.1 Must allow input of developer resources 4](#_Toc83632716)

[2.1.2 Must allow input of project details 5](#_Toc83632717)

[2.1.3 Must allow input of algorithm variables 6](#_Toc83632718)

[2.1.4 Must output a roadmap schedule given the input 6](#_Toc83632719)

[2.2 Assumptions 6](#_Toc83632720)

# Project Overview

## 1.1 Purpose and Motivation

This document documents the vision and requirements for the Planr software project. The intended audience of this document is the supervisory committee members monitoring the project. This will be a “living” document and will, expectedly, go through several iterations which may be approved by the supervisory committee. The purpose of this document is to define the parameters and requirements specifications of the Planr software project.

The motivation behind developing the Planr application is manifold. I have spent several years as a Software Engineer at Garmin International writing desktop and mobile application software. In May of 2020 I was promoted to Team Lead – Software Engineering and had to take on more project planning tasks. The tools at my disposal are lacking in their ability to easily plan sprints with the features, engineering resources, and timelines I am working with. The Planr application is intended to aide in allowing a Project Manager to define features, schedules, and engineering resources to output an optimized schedule or roadmap.

## 1.2 Goals

The Planr application will be designed to allow a user to input several parameters that will be used to calculate and output an optimized sprint schedule. Planr will record data on individual engineer capabilities to help identify the best project schedule possible given the skillset of the resources. Working on a cross-platform mobile application with some engineers who are proficient on one platform, but not the other, and other engineers who are capable of working cross-platform creates complexity in sprint planning which Planr seeks to minimize. Planr will allow for the input of required skillsets for a particular project and feature, thus allowing the application to schedule individual engineers with their matches skillsets. After all data has been collected the Planr app will output a forward-looking sprint schedule which can also be used as a roadmap.

## 1.3 Risks

Providing accurate level of effort (LOE) estimates for projects can often be challenging. While the Planr application will allow users to input feature names and LOEs for project features, the accuracy of the Planr output is reliant upon accurate LOEs. There is a risk of inaccuracy if the user inputting the LOE values into the Planr application is not appropriately considering the disparity between engineer provided LOEs and the *actual* level of effort a particular feature took. Engineers may not always be concerned about the entire picture and can leave out details in their estimates like unit test estimates, quality assurance estimates, documentation, and integration testing.

Resource scheduling may be another risk in the Planr application output. The Planr application will allow for resource schedule input and will account for that in the projected schedule output. However, if the Planr user does not input the availability of the individual engineers on the projects, the application will not be able to provide an accurate schedule as part of the output.

The final potential risk that is worth calling out is that I will be doing this project in SwiftUI which I am not the most familiar with. I am proficient in traditional Swift using MVC and MVVM design patterns. My plan for mitigating this risk is to proof of concept the more complex calendar view with SwiftUI and if that ends up taking up too much time, I can shift to using MVC and traditional Swift.

## 1.4 Constraints

Due to my proficiency in Swift development and the macOS platform, the Planr application will be written in Swift to be used on macOS. Testing will be limited to an Apple computer running macOS.

# Requirements Specification

I will be capturing the respective requirements for the Planr application below.

## 2.1 Functional Requirements

### 2.1.1 Must allow input of developer resources

The input of developer resources is important in providing the pertinent information on how to schedule said resources. By allowing the user to input the developers and their proficiencies the planning algorithm can best schedule the appropriate resources.

#### 2.1.1.1 Developer first name

The user must be allowed to enter a developer’s first name as a String no more than 40 characters in length. This will be stored in a database with the remaining developer information. After all developer information input is complete the user will see all entered fields in the UI.

#### 2.1.1.2 Developer last name

The user must be allowed to enter a developer’s last name as a String no more than 40 characters in length. This will be stored in a database with the remaining developer information. After all developer information input is complete the user will see all entered fields in the UI.

#### 2.1.1.3 Platforms they are proficient in

The user must be allowed to designate the platforms that the user is proficient in. The platforms will be selectable from a collection of “chips” in the UI. The user must be allowed to select multiple platform proficiencies for a given developer. After all developer information input is complete the user will see all entered fields in the UI.

#### 2.1.1.4 Days they are available

The use must be allowed to choose the days of the week and the total number of hours per week that they are available to work. This will allow the scheduling algorithm to account for days that the developer has scheduled off. After all developer information input is complete the user will see all entered fields in the UI.

### 2.1.2 Must allow input of project details

The input of project details is important to provide the remaining parameters by which the scheduling algorithm will use to generate a schedule.

#### 2.1.2.1 Project features

Each project has a subset of features that make up the project itself. The user must be allowed to input the feature parameters.

#### 2.1.2.2 Feature platform

The user must be allowed to input the platform, or platforms, the respective feature, or features, will be developed on. This will allow the scheduling algorithm to decide which developer to put on which feature. The user will see the feature parameters displayed in the UI.

#### 2.1.2.3 Feature point estimate

The user must be allowed to input the point estimate for the respective features. This will allow the scheduling algorithm to determine which developers have the capacity to work on which feature and if the feature needs to be spread over multiple sprints. The user will see the feature parameters displayed in the UI.

#### 2.1.2.4 Feature Priority

The user must be allowed to input the priority of the features for the project. This will allow the scheduling algorithm to decide which feature to schedule with developers first. The user will see the feature parameters displayed in the UI.

### 

### 2.1.3 Must allow input of algorithm variables

#### 2.1.3.1 Sprint lengths

The user must be allowed to input the length of the sprint. The options for sprint lengths will be one-week sprint, two-week sprint, three-week sprint followed by a one-week sprint, and a four-week sprint. The user may only choose one option for the sprint length. After all project information input is complete the user will see all entered fields in the UI.

#### 2.1.3.2 Estimate padding

The user must be allowed to input a multiplier for estimates provided by development. Historically developers can be inaccurate when providing estimates. By allowing the user to input a multiplier for estimates the algorithm can more accurately provide a schedule. The padding multiplier must be a double with one decimal place accuracy between 0.0 and 3.0. After all project information input is complete the user will see all entered fields in the UI.

#### 2.1.3.3 Team Velocity

The user must be allowed to input an integer value for the team’s average velocity. It is within Agile principles to track *team* velocity, but not *individual* velocity. This number will be used help determine how many sprint points should be planned each sprint and how many points remain. After all project information input is complete the user will see all entered fields in the UI.

### 2.1.4 Must output a roadmap schedule given the input

The scheduling algorithm will take all valid inputs and generate a scheduled roadmap that displays the features each developer will work on for each sprint until each of the input features are all planned.

## 2.2 Assumptions

All input will be validated before scheduling algorithm runs. The user will be notified by error modal that input is invalid and instruct them to adjust the input to be within the valid parameters.