System Architecture 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. Introduction 5](#_Toc88569348)

[2. Class and Method Descriptions 5](#_Toc88569349)

[2.1 Project 5](#_Toc88569350)

[2.2 Engineer 5](#_Toc88569351)

[2.3 PlanrFeature 6](#_Toc88569352)

[2.3.1 UnplannedFeature 6](#_Toc88569353)

[2.3.2 PlannedFeature 6](#_Toc88569354)

[2.4 Sprint 7](#_Toc88569355)

[2.5 WorkBlock 7](#_Toc88569356)

[2.6 Platform 8](#_Toc88569357)

[2.7 Roadmap 8](#_Toc88569358)

[2.8 PlanningCoordinator 8](#_Toc88569359)

[2.9 Views 8](#_Toc88569360)

[2.9.1 ProjectRoadmapView 9](#_Toc88569361)

[2.9.2 WorkBlockView 10](#_Toc88569362)

[2.9.3 PlanrSettingsView 10](#_Toc88569363)

[2.9.4 AppSettingsGroupView 11](#_Toc88569364)

[2.9.5 NewProjectView, RootView, WelcomeView 12](#_Toc88569365)

[2.9.6 Add\* Views 13](#_Toc88569367)

[2.9.7 Summary Views 15](#_Toc88569368)

[2.9.8 MultiDatePickerView 16](#_Toc88569369)

[2.9.10 PlannedSprintView 16](#_Toc88569370)

[2.9.11 PlatformSelectionView 17](#_Toc88569371)

[2.9.12 PlatformSprintBlockView 17](#_Toc88569372)

[2.10 ViewModels 17](#_Toc88569373)

[2.11 PlatformSelectionModel 18](#_Toc88569375)

# Introduction

This is the architecture design document for the Planr application. This document provides a reference to the design of the application with class and sequence diagrams. Each class will have a brief description about the function of the class and its methods.

# Class and Method Descriptions

## 2.1 Project

Timeline

Description automatically generated

This class is a representation of an unplanned body of work containing Engineers and UnplanedFeatures. It is the top-level object that gets passed into the PlanningCoordinator to plan and used to generate the Roadmap. This has methods to add and remove UnplannedFeatures and Engineers.

## 2.2 Engineer

Diagram

Description automatically generated

This class is a representation of an Engineer for planning purposes. The constructor requires a first name, last name, a collection of platforms the Engineer is proficient in, and a collection of dates that the Engineer is unavailable to work. This can be constructed with a RealmEngineer object as well.

The RealmEngineer class has to conform to Realm’s Object protocol in order to be serialized to the Realm database. The only notable difference is that Realm objects have their own collections. This has a List of Dates instead of an Array of Dates.

## 2.3 PlanrFeature

Diagram

Description automatically generated with medium confidence

This is an interface (or protocol in Swift) that both the UnplannedFeature and PlannedFeature classes implement. This includes the name of the feature, a summary of the feature, the platforms that apply, an effort estimate (per platform), a priority value from 0-1000, a Boolean value for if concurrency is allowed for the feature, a color to display on the UI, and a feature id GUID to identify the feature.

### 2.3.1 UnplannedFeature

This class conforms to the PlanrFeature protocol and represents an unplanned feature. A collection of these are passed into the Project contstructor to be used for planning. This has a public method to update the name of the feature and another public method to update the priority of the feature.

### 2.3.2 PlannedFeature

A PlannedFeature has a collection of WorkBlocks that is initially empty. As WorkBlocks are planned they get assigned to a PlannedFeature via the assignWorkBlock() method. There is a convenient util method called nextUnassignedWorkBlockForPlatform(platform:) that returns the next WorkBlock that is not assigned for the passed in Platform.

## 2.4 Sprint

Text

Description automatically generated

This structure represents an Agile Sprint. The constructor takes in a collection of WorkBlocks, the points remaining in the sprint, and a date range of the start and end dates of the sprint. There is a public method to add a WorkBlock to the sprint and a public method to output the information about the sprint in the console for debugging purposes. There is a private member variable that holds the initial sprint points to allow for calculating the sprint points remaining. There is a util method, workBlocksForPlatform(\_:), that returns all WorkBlocks for the passed in Platform. The printSprintInfo() is a debug method for showing information about the sprint. The decrementPointsRemaining(by:) is a private method to reduce the pointsRemaining member variable.

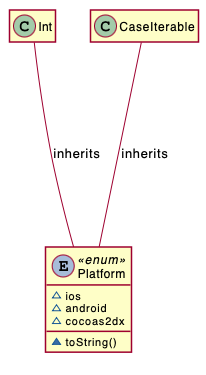
## 2.5 WorkBlock

Timeline

Description automatically generated

This class is a representation of the block of work inside a sprint. It can be initialized either with an already PlannedFeature, point value of the work block, the platform it’s associated with and the id of the sprint it will be a part of, or with the alternative constructor that has the feature name and summary instead.

## 2.6 Platform



This enum is used to indicate what platform to use for the given associated struct or class. This is used throughout the application and is a property on several classes. It inherits from the Int and CaseIterable classes and has a helper method toString() to return the String value of the enum.

## 2.7 Roadmap

Diagram, text, application

Description automatically generated

This struct represents a planned timeline of sprints. The constructor takes in a collection of Sprint instances and has a public method for printing the roadmap information to console.

## 2.8 PlanningCoordinator

Text

Description automatically generated

This struct is used to plan the Roadmap. The constructor takes in the Project object which is used to populate the remaining member variables. The plan() public method is used to plan the project and will output a Roadmap object that will be used to display on the UI.

## 2.9 Views

### 2.9.1 ProjectRoadmapView

Diagram

Description automatically generated with low confidence A picture containing graphical user interface

Description automatically generated

This struct, as with the remainder of the views, inherits from the View base class. To construct this you need to pass in the String of the projectName and give it a provided Roadmap object. This also has a private method to get a placeholder view for the month until more work has been done to implement the horizontal scrolling month view.

### 2.9.2 WorkBlockView

Diagram

Description automatically generated with low confidence Graphical user interface, text, application, chat or text message

Description automatically generated

This struct is a reusable view that displays the information from a given WorkBlock. It is constructed with a WorkBlock object.

### 2.9.3 PlanrSettingsView

A picture containing diagram

Description automatically generated Graphical user interface, text, application

Description automatically generated

This struct is a container view that holds an instance of the AppSettingsGroupView.

### 2.9.4 AppSettingsGroupView

Diagram

Description automatically generated Graphical user interface, text, application, email

Description automatically generated

This view is a reusable view that is contained inside of the NewProjectView and the PlanrSettingsView. The fields on are property wrappers of the AppStorage struct which allows for the setting/getting of the values to write directly to UserDefaults.

### 2.9.5 NewProjectView, RootView, WelcomeView

### Diagram Description automatically generated Graphical user interface, text, application Description automatically generated Graphical user interface, application Description automatically generated

The WelcomeView is the first view within the application. Clicking the “Start New Project” button navigates to the NewProjectView via the NavigationStack EnvironmentObject. The NewProjectView utilizes the NewProjectViewModel, which is an ObservedObject member variable to automatically validate the project name and store the project start date.

### 2.9.6 Add\* Views

Diagram

Description automatically generated

Graphical user interface, text, application

Description automatically generated

The AddEngineerView is constructed with a Project. It uses the AddEngineerViewModel member variable to process automatic validation of the first and last name input text fields as well as creating the Engineer (and RealmEngineer) objects. The PlatformSelectionViewModel is used in conjunction with the PlatformSelectionModel to indicate if any of the platforms have been selected and binds the switch values to a property automatically. Finally, the MultiDatePickerViewModel is used to get the selected values from the date picker.

Graphical user interface, text, application

Description automatically generated

The AddFeatureView is used for creating an UnplannedFeature object. This uses the AddFeatureViewModel to validate the input fields on the fly, store the input values, and ultimately create the UnplannedFeature object. Like the AddEngineerView, this view also uses the PlatformSelectionViewModel.

Graphical user interface, text, application

Description automatically generated

The AddProjectEngineersView enables the user to create Engineer objects and add them to the project. The “Add Engineer” button navigates to the AddEngineerView. This has a List view that contains an updated list of EngineerSummaryViews. Continue is enabled when the list of Engineers is not empty.

Graphical user interface, text, application

Description automatically generated

Similar to the AddProjectEngineersView this view contains a List of FeatureSummaryViews. The “View Roadmap” button is disabled until the list of UnplannedFeatures is not empty. Clicking the “Add Feature” button navigates to the AddFeatureView.

### 2.9.7 Summary Views

Diagram

Description automatically generated Graphical user interface, text, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

These views are the reusable views that are constructed with Engineers and Feature objects respectively. They are used by their container views to display the information provided from the contstructors.

### 2.9.8 MultiDatePickerView

Diagram

Description automatically generated with medium confidence Calendar

Description automatically generated

This is a container view for an external date picker solution. The MultiDatePickerViewModel contains the collection of chosen Dates. This is contained within the AddEngineerView.

### 2.9.10 PlannedSprintView

Diagram

Description automatically generated Graphical user interface, application

Description automatically generated

This view is a reusable view constructed with a given platform and a collection of Sprint objects. This view contains a collection of PlatformSprintBlockViews inside of an HStack. Each PlatformSprintBlockView contains a collection of WorkBlockViews.

### 2.9.11 PlatformSelectionView

Timeline

Description automatically generated with low confidence Graphical user interface, text, application, chat or text message

Description automatically generated

This view in conjunction with the PlatformSelectionViewModel allows for either a Feature or Engineer to have a Platform or Platforms associated with them. The view model stores the active values and allows the caller to check if any of the items are enabled.

### 2.9.12 PlatformSprintBlockView

Diagram

Description automatically generated Graphical user interface, text, application

Description automatically generated

This view is constructed with a DateInterval object to display the date range of the given Sprint. It also requires a collection of WorkBlocks and an Integer value of the remaining points in the sprint. There are private methods to get the date range view and the points remaining view.

## 2.10 ViewModels

## Diagram Description automatically generated

Each ViewModel is used by it’s associated View to store values in bound properties to the UI to automatically update the UI. Each of these inherit from the ObservableObject class which allows for the SwiftUI views to automatically observe changes and reflect them in the UI.

## 2.11 PlatformSelectionModel

Timeline

Description automatically generated

This struct is used to hold Toggle button values.