**Software Requirements Specifications**

**For Monkey Manager**

**Version 1.0**

**Nick Aubuchon**

**Michael Bernardy**

**Brittany Dibble**

**Gregory Farinelli**

**Jesus Hernandez**

**Shane Lopez**

**Angelo Mendoza**

**Benjamin Siegel**

**1. Introduction**

**1.1 Purpose**

The document provides an explanation for the design, implementation, and use

cases for the Monkey Manager application.

**1.2 Background**

Monkey Manager organizes a user’s programming project by implementing their project into a kanban board. A user creates a new project, adds tasks to each project and keeps track of due dates. By keeping track of each task, it allows the user to see the progress of each task to completion.

**1.2 Definition, Acronyms and Abbreviations**

No software specifics.

**2. System Requirements**

**2.1 GUI**

Upon opening the program, the user will first see an empty board with preselected categories for the columns: To Do, Doing, Testing, and Done. On the top right there is a button for delete project. Directly to the left is the Save All button. Directly to the left of save all, is Save. Under Delete Projects, is a New Tasks button. On the bottom left is where the New Project button is located. On the bottom portion of the gui is a description for the currently selected task, including: Name of the task, due date, created date, and description.

When a new project is selected the project will populate on the left side of the gui.

When new task is selected, a popup window will display allowing the user to enter the Name, Due Date, Description, and Status.

When a new task is created, the task will populate on the main screen under the designated status.

**2.2 Actor(s)**

Programmers- Students and programmers who are structuring code and require a form of organization.

**2.3 Basic Use Cases**

2.3.1 Open Program

2.3.1.1 System creates new project or opens an existing saved projects

2.3.2 Create New Project

2.3.2.1 User opens program

2.3.2.2 User selects New Project Button

2.3.2.3 System creates new project object with name “New Project”

2.3.2.4 System updates project list to show the new project

2.3.3 Open Existing Project

2.3.3.1 User selects project on the list of projects

2.3.3.2 System refreshes board and displays tasks assosiated with that project

2.3.4 User Creates Task

2.3.4.1 User selects New Task button

2.3.4.2 System opens Dialog Box for task information insertion: Name, Due Date, Description, Status.

2.3.4.3 User writes task Name, selects task Due Date, writes in task Description, and selects the task Status from the drop down menu.

2.3.4.4 System creates project card and will populate on the board under designated status column.

2.3.5 User Rearranges Task

2.3.5.1 User edits the task Status

2.3.5.2 System reassigns task location to new task Status

2.3.6 User Edits Task

2.3.6.1 User clicks on Task

2.3.6.2 System shows Task Detail View

2.3.6.3 User click on edit button in Detail View

2.3.6.4 System opens dialog with editable fields

2.3.6.5 User submits changes

2.3.6.6 System updates card on the board

2.3.7 User Deletes Task

2.3.6.1 User clicks on Task

2.3.6.2 System shows Task Detail View

2.3.6.3 User clicks on Delete Task Button in Detail View

2.3.6.4 System removes task card from board

2.3.8 User Saves Project

2.3.8.1 Saves Currently Selected Project

2.3.8.1.1 User clicks the Save button

2.3.8.1.2 Currently selected project is saved

2.3.8.2 Save All Projects

2.3.8.2.1 User clicks the Save All button

2.3.8.2.2 All open projects are saved

2.3.8 System synchronizes with database

Classes:

MAINWINDOW CLASS

Connects the backend to the front end GUI of the main window. All buttons, labels, lists, and the task kanban board are controlled by the MainWindow class.

PROJECT CLASS

Contains information on a project object including a list of tasks associated with the project, as well as the methods to manipulate said information.

PROJECTLISTMODEL CLASS

Connects the projects vector stored in ProjectUtils to the frontend listview. Changes made on the frontend in the listview will be applied to the backend with this class. Changes to the backend will emit a signal so that the frontend knows that it needs to change.

PROJECTUTILS CLASS

The ProjectUtils class is the class that handles all of our project storing and retrieving. It is a singleton that keeps an instance of a project and can open other projects. It has functions to open and close a task as well as set and get an open project

TASK CLASS

The Task class stores data that is entered in the TaskWidget class. Task class stores data for the due date, created date, description, id, status, and the task number of a task that the user creates.

TASKDIALOG CLASS

The TaskDialog class is for the task input gui. The intended purpose of this class is to take a user input for all of the task details, including: Name, due date, description, and status. Once the input has been accepted, the data will be sent to the task class.

TASKWIDGET CLASS

The TaskWidget class is for the task card on the kanban board. It connects the data from the tasks vector, stored within the currently open project, to a front end widget to be displayed to the user. The data that is displayed includes: Name and due date. The data is pulled from the task class.

Other Files

CONSTANTS

This file contains global constants, such as error codes, for access anywhere in the program. There are no functions or non-constant variables in this file.

JSON UTILS

This file contains functions for storing and reading program data into a JSON file. The file saves each Project in a JSON file along with their attributes. This includes the vector of Task objects associated with the Project and each of the Task objects attributes. This information is then read and loaded into the program on the next start of the application.