Software Requirements Specification  
Scrum Manager

|  |  |
| --- | --- |
| Ryan Feeney | ryanq.feeney@ufl.edu |
| Jesse Martinez | kroronos@ufl.edu |
| Tony Strother | semphy@ufl.edu |
| Stephen Berkner | sberkner@ufl.edu |
| Kyle Collins | Kcoll196@ufl.edu |
| Bryan Fallin | fallinbryan@ufl.edu |

Table of Contents

[**1 INTRODUCTION** 3](#_Toc504338418)

[**1.1 Purpose** 3](#_Toc504338419)

[**1.2 Project Summary** 3](#_Toc504338420)

[**1.3 Target Platform(s) / Operating System(s) Supported** 3](#_Toc504338421)

[**1.4 Tools/APIs/Development Environment/Programming Languages** 3](#_Toc504338422)

[**2 SYSTEM DESCRIPTION** 3](#_Toc504338423)

[**2.1.1 (Example) Application Modes** 3](#_Toc504338424)

[**2.1.2 (Example) Database description** 3](#_Toc504338425)

[**2.1.3 (Example) Network description** 3](#_Toc504338426)

[**2.2 User Characteristics** 3](#_Toc504338427)

[**2.3 Constraints** 4](#_Toc504338428)

[**2.4 Wireframes** 4](#_Toc504338429)

[**3 FUNCTIONAL REQUIREMENTS** 4](#_Toc504338430)

[**3.1 Main Features** 4](#_Toc504338431)

[**3.2 Use Cases** 4](#_Toc504338432)

[**3.3 Use case diagrams** 4](#_Toc504338433)

[**4 EXTERNAL INTERFACE REQUIREMENTS** 4](#_Toc504338434)

[**4.1 User Interfaces** 4](#_Toc504338435)

[**4.2 Hardware Interfaces** 4](#_Toc504338436)

[**4.3 Software Interfaces** 4](#_Toc504338437)

[**4.4 Communications Interfaces** 5](#_Toc504338438)

# **1 INTRODUCTION**

### **1.1 Purpose**

SRS version 2.0 for Group16 Scrum Manager

### **1.2 Project Summary**

A project management interface based on the scrum development process.

### **1.3 Target Platform(s) / Operating System(s) Supported**

* Windows
* Linux
* Mac

### **1.4 Tools/APIs/Development Environment/Programming Languages**

* This project will by using Python 3.6 32bit, the included tkinter library, MySQLdb library, and the PyGitHub Library
  + <https://www.python.org/downloads/release/python-360/>
  + <https://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python>
  + https://github.com/PyGithub/PyGithub
* IDE will be Jetbrains PyCharm
  + https://www.jetbrains.com/pycharm/
* Version Control will be managed through GitHub
  + <https://github.com/CEN3031-group16/GroupProject>

# **2 SYSTEM DESCRIPTION**

This system will allow teams to visually manage development projects, track development progress, and communicate efficiently through an intuitive GUI

System will connect to a remote MySQL server @ 173.230.136.241 to store project data so that multiple users across different workstation and platforms can work on the same project with synchronized data. User PC must be connected to the internet for the application to function.

System will model the Scrum agile development process. Backlog items will be stored in the database as cards that will have foreign keys associated with an assigned user, a sprint cycle, and a link to explore code hosted on GitHub. Cards will have type ‘Epic’, ‘User Story’, ‘Feature’, ‘Bug’, ‘Re-Factor’, and ‘Complete’. Cards will be represented in code as distinct objects in a map hashed by a card’s unique table key.

Users will have privilege levels based on role, only admins can create new projects, new users, assign user roles and everything below. Scrum masters can finalize backlog items as complete, move cards back to the backlog for bugs or refactoring, initiate Sprints, assign cards to users, and lock a card to a dev user, and everything below. Dev users can assign cards to themselves, comment on cards, create new cards, submit cards for review, link cards to code, un-assign themselves from cards unless locked, and change their own password.

Sprint Cycles will be stored in the database with a unique key, start date, and due date. Assigning cards to sprint cycles will occur by populating a card sprint cycle field with the unique key. If a user is assigned to a card that is assigned to a sprint, that user is assigned to the sprint.

Comment will be stored in the database in a comment table with a unique identification key, a timestamp, and foreign keys that associated with users, cards, and sprints.

Users will be required to authenticate upon opening the application. Login credentials will not be cached locally.

Startup GUI will be a dashboard that displays current sprint details in the center, backlog on the left, list of all comments ordered by date on the right. Center display will be different based on user role.

### 

### **2.1.1 Application Modes**

Application will have user specific Mode

|  |  |
| --- | --- |
| Admin Mode | Can Create and delete projects and users |
| Scrum Master Mode | Can Assign User Roles |
| Developer Mode | Limited Admin Privileges |

### **2.1.2 Database description**

Remote MySQL server

Project\_Name Database

UserTable(Int:UserID, String:UserName, String:UserEmailAddress, String:UserPassword, String:UserRole)

CardTable(Int:CardType, Int:Priority, F\_Key:AssignedUserID, String:CardTitle, String:CardDescription, Date:DueDate, Date:AssignedDate, F\_Key:SprintID, String:httpLinkToCode, Int:Status )

SprintTable(Int:SprintID, Date:StartDate, Date:DueDate)

CommentTable(Int:CommentId, Date:CommentTimeStamp, F\_Key:CardID, F\_Key:UserID)

### **2.2 User Characteristics**

The typical user is a member of a development team utilizing the Scrum process for product development.

### **2.3 Constraints**

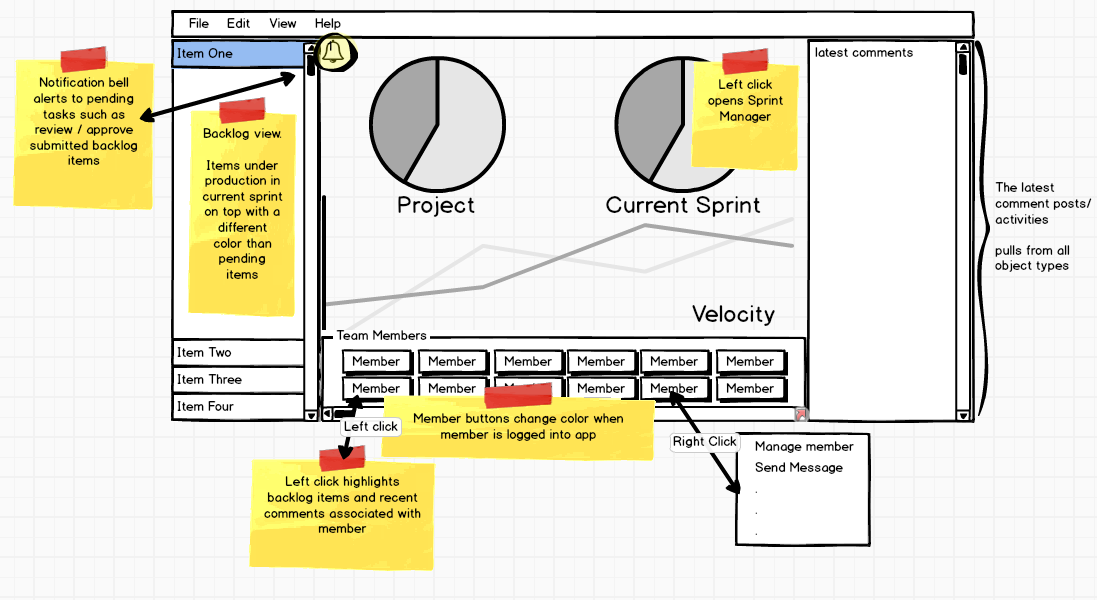
Application requires a pre-configured remote MySQL database

Linking code to a card will require a git repository that interfaces with a web server such as GitHub

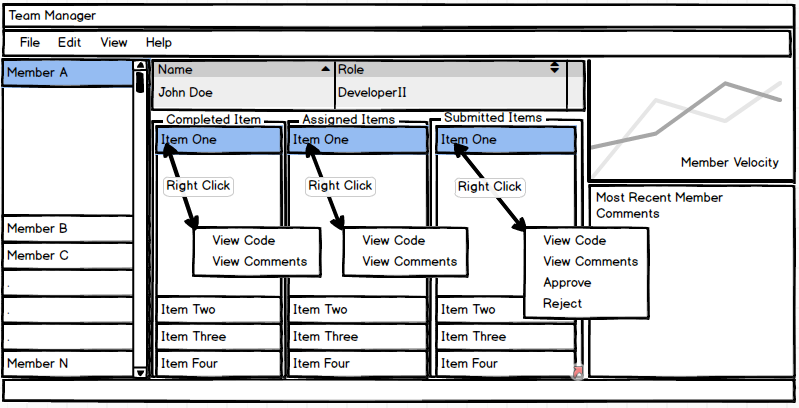
### **2.4 Wireframes**

Provide wireframes of major features to illustrate the overall structure/flow of the application. (At least 1 wireframe per person in the group).

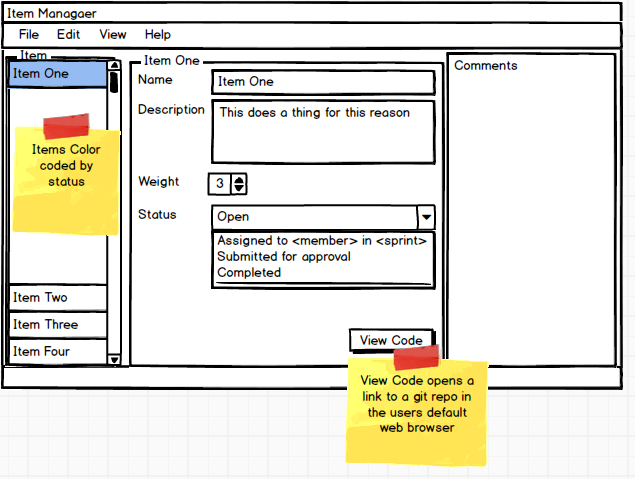
Scrum Master Dashboard



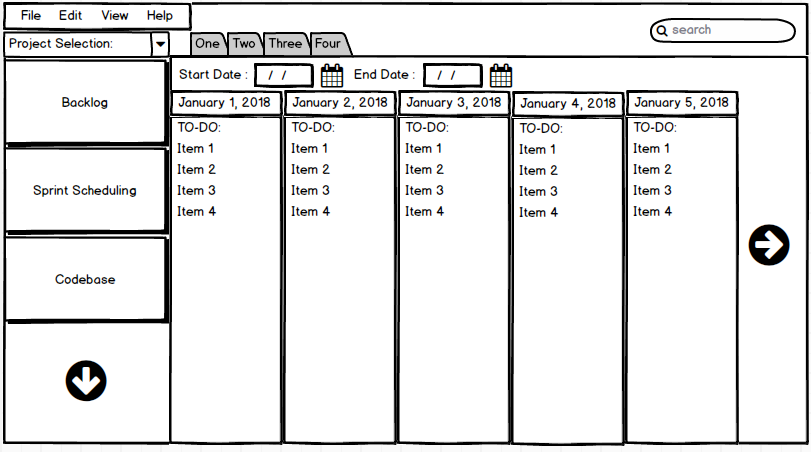
User Manger



Card Manager



Sprint Manager



# **3 FUNCTIONAL REQUIREMENTS**

### **3.1 Main Features**

* GUI
* Visual Backlog Queue
* Create Users
* Create Backlog Items
* Create Sprints with start dates and due dates
* Drag users to Backlog Items to assign tasks
* Drag Backlog Items to Sprints
* Auto create Sprints bases on Backlog priority Queue
* Add comments to Backlog Items or Sprints
* View Latest Code associated with Backlog Item From GitHub
* Connect to database
* Connect to Git repo

### **3.2 Use Cases**

Create a description of use cases for several of the features in your project.

|  |  |
| --- | --- |
| Create User Profile | |
| Primary Actor | Admin, User |
| Description | For a user to exist in the system, a new one must be created. This can be done by either an existing user with administrative privileges or through the application by any user in order to make their own account. The second action may only be done with an appropriate code, such as a product key or an invitation code from an existing administrator. |
| Precondition | If user is performing action, valid code must be submitted to system. |
| Trigger | User interacts with the new profile link on the login screen of the GUI or an admin interacts with the create a new user action within the application itself. |
| Postcondition | The data associated with a user is added to the data base. A notification regarding the success of this action is sent to whoever initiated it. |
| Main Success Scenario | A notification signaling the success of the action is sent to the user. A user may login to the newly created profile if they provide the appropriate information on login. |
| Extensions | A key for user creation should be able to be generated by profiles with sufficiently high privileges. Such a key may generate no profile of higher privileges than the profile which generated it. |

|  |  |
| --- | --- |
| Edit User Profile | |
| Primary Actor | Admin, User |
| Description | A user should be able to change certain details of their profile, whether it be their contact information or their name. Administrators should also be able to perform this action. |
| Precondition | A user exists. |
| Trigger | User clicks the edit profile button on their profile page. Administrator clicks the same button on the profile that they desire to change. |
| Postcondition | Information is altered in the database. Notification regarding the success of the action is sent to user. |
| Main Success Scenario | User profile is successfully changed. Success message is sent to user. |
| Extensions | When an administrator attempts to change specific fields a confirmation should be sent to the user whose profile they are trying to change, if the user neither accepts nor denies this change it will be applied after a certain period of time. |

|  |  |
| --- | --- |
| Create Project | |
| Primary Actor | Administrator |
| Description | An administrator should be able to create projects to which users may be assigned, sprints may be created, or and a backlog may be added to. |
| Precondition |  |
| Trigger | Administrator clicks the create project button. |
| Postcondition | Project is created, appropriate entry is created in database. Notification regarding success of the action is sent to user. Any users initially assigned to project also receive a notification. |
| Main Success Scenario | Project is created and visible to any users assigned to it. A success notification is sent to appropriate parties. |
| Extensions |  |

|  |  |
| --- | --- |
| Edit Project | |
| Primary Actor | Admin |
| Description | The details of a project should be able to be changed by an administrator. |
| Precondition | A project exists. |
| Trigger | Administrator clicks edit project button. |
| Postcondition | Details in the database entry are changed. Notification is sent to user. |
| Main Success Scenario | Success notification is sent to user. Changes made to databases are visible. |
| Extensions |  |

|  |  |
| --- | --- |
| Close Project | |
| Primary Actor | Administrator |
| Description | When a project is completed, or canceled a project should be able to be closed by an administrator. |
| Precondition | A project exists. |
| Trigger | Administrator clicks the close project button and confirms that the project ought to be closed. |
| Postcondition | All users are removed from project, all tasks removed and deleted, project is deleted from database. A notification is sent to all users assigned to project that it has been closed. |
| Main Success Scenario | Project is removed from database and is no longer visible to users. Users successfully received notification. Administrator who closed project received a success notification. |
| Extensions |  |

|  |  |
| --- | --- |
| Create a Task | |
| Primary Actor | User |
| Description | Any user may create a task. A task includes a short description of a task and may include other information such as a tag indicating what part of a project it relates to. The task must also be assigned to a project, usually determined by the context from which the user creates the tasks.  Users who are not in the context of any specific project would select which project the task is created for. |
| Precondition | A project exists for the task to be assigned to. |
| Trigger | User command issued by user. |
| Postcondition | Database is updated with appropriate information. User receives notification regarding status of task. |
| Main Success Scenario | Any user who is a part of the project can see that a task is now assigned to the project. User receives notification that task has been accomplished successfully. |
| Extensions | Any user who is not a part of a project cannot assign a task to that project without special permissions. |

|  |  |
| --- | --- |
| Reorder a Task | |
| Primary Actor | Scrum master |
| Description | A scrum master during the grooming process is responsible for prioritizing items in the backlog for ease of assignment for the next sprint.  A scrum master should be able to reorder existing tasks to reflect this process. |
| Precondition | More than one task exists in a single process. |
| Trigger | User command issued by a scrum master. |
| Postcondition | Database is updated with appropriate information. User receives notification regarding status of task. |
| Main Success Scenario | Any user who is a part of the project can see that the ordering of tasks in the appropriate view has changed. User receives notification that task has been accomplished successfully. |
| Extensions | If a scrum master attempts to change the ordering of a single task, nothing changes. The user receives a special message indicating that they performed an invalid action. |

|  |  |
| --- | --- |
| Assign a Task to a Sprint | |
| Primary Actor | Scrum master |
| Description | A scrum master may assign a task from the list of created tasks to a certain sprint. |
| Precondition | A task exists and there is both a sprint to assign it to. It is before the sprint start-date. |
| Trigger | User command issued by scrum manager. |
| Postcondition | Database is updated with appropriate information. Scrum master receives notification regarding status of task. |
| Main Success Scenario | In the context of a certain sprint, any user can see that the task has now been assign to the sprint, along with a timestamp regarding when it was added. Scrum master receives notification that task has been accomplished successfully. |
| Extensions | Scrum master receives an action failed notification if the sprint is already in progress.  This can be performed simultaneously with assign a task to a user. |

|  |  |
| --- | --- |
| Assign a Task to a User | |
| Primary Actor | Scrum master |
| Description | A scrum master may assign a task from the list of created tasks to a developer for the period of a certain sprint. |
| Precondition | A task exists and has already been assigned to a sprint. There is a user to assign the task to. |
| Trigger | User command issued by scrum manager. |
| Postcondition | Database is updated with appropriate information. Scrum master receives notification regarding status of task. |
| Main Success Scenario | In the context of a certain sprint, any user can see upon inspecting the task that it has been assigned to the correct user. The scrum master receives a notification that the task has been accomplished successfully. |
| Extensions | A user can volunteer for a task, but a scrum master must approve it.  Scrum master receives an action failed notification if the sprint is already in progress.  This can be performed simultaneously with assign a task to a sprint. |

|  |  |
| --- | --- |
| Submit Task for Review | |
| Primary Actor |  |
| Description |  |
| Precondition |  |
| Trigger |  |
| Postcondition |  |
| Main Success Scenario |  |
| Extensions |  |

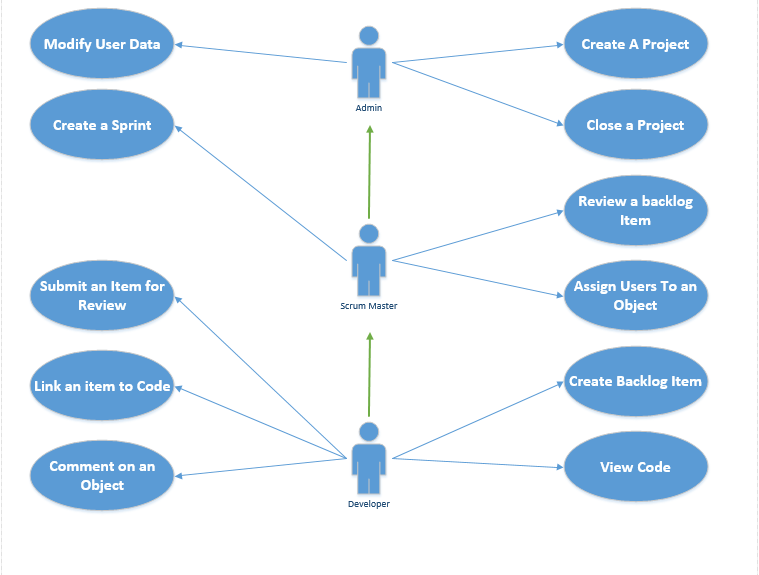
|  |  |
| --- | --- |
| Link an Item to Code | |
| Primary Actor | Developer |
| Description |  |
| Precondition |  |
| Trigger |  |
| Postcondition |  |
| Main Success Scenario |  |
| Extensions |  |

|  |  |
| --- | --- |
| View Code Linked to Item | |
| Primary Actor | Developer |
| Description |  |
| Precondition |  |
| Trigger |  |
| Postcondition |  |
| Main Success Scenario |  |
| Extensions |  |

|  |  |
| --- | --- |
| Create A New Sprint | |
| Primary Actor | Scrum master |
| Description |  |
| Precondition |  |
| Trigger |  |
| Postcondition |  |
| Main Success Scenario |  |
| Extensions |  |

|  |  |
| --- | --- |
| Edit Sprint | |
| Primary Actor | Scrum master |
| Description |  |
| Precondition |  |
| Trigger |  |
| Postcondition |  |
| Main Success Scenario |  |
| Extensions |  |

### **3.3 Use case diagrams**

Green Arrow indicates cascaded permissions. 

# **4 EXTERNAL INTERFACE REQUIREMENTS**

### **4.1 User Interfaces**

*Specify the logical characteristics of each interface between the software product and its users (e.g., required screen formats, report layouts, menu structures, or function keys).*

*Specify all the aspects of optimizing the interface with the person who must use the system (e.g., required functionality to provide long or short error messages). This could be a list of do’s and don’ts describing how the system will appear to the user.*

### **4.2 Hardware Interfaces**

*Specify the logical characteristics of each interface between the software product and the hardware components of the system. This includes configuration characteristics (e.g., number of ports, instruction sets), what devices are to be supported, and protocols.*

### **4.3 Software Interfaces**

*Specify the use of other required software products (e.g., a database or operating system), and interfaces with other application systems.*

*For each required software product, provide identification information including at least name, version number, and source.*