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](#_Toc505015901)

[1.1 Purpose 3](#_Toc505015902)

[1.2 Project Summary 3](#_Toc505015903)

[1.3 Target Platform(s) / Operating System(s) Supported 3](#_Toc505015904)

[1.4 Tools/APIs/Development Environment/Programming Languages 3](#_Toc505015905)

[2 SYSTEM DESCRIPTION 3](#_Toc505015906)

[2.1.1 Application Modes 4](#_Toc505015907)

[2.1.2 Database description 4](#_Toc505015908)

[2.2 User Characteristics 4](#_Toc505015909)

[2.3 Constraints 4](#_Toc505015910)

[2.4 Wireframes 5](#_Toc505015911)

[3 FUNCTIONAL REQUIREMENTS 10](#_Toc505015912)

[3.1 Main Features 10](#_Toc505015913)

[3.2 Use Cases 10](#_Toc505015914)

[3.3 Use case diagrams 17](#_Toc505015915)

[4 EXTERNAL INTERFACE REQUIREMENTS 18](#_Toc505015916)

[4.1 User Interfaces 18](#_Toc505015917)

[4.2 Hardware Interfaces 18](#_Toc505015918)

[4.3 Software Interfaces 18](#_Toc505015919)

[4.4 Hardware Requirements 19](#_Toc505015920)

[Figure 1 Dashboard ~ Feeney 5](#_Toc505015925)

[Figure 2. Scrum Master Dashboard ~ Fallin 5](#_Toc505015926)

[Figure 3 Project Manager ~ Martinez 6](#_Toc505015927)

[Figure 4 User Manger ~ Strother 6](#_Toc505015928)

[Figure 5 User Manager ~ Fallin 7](#_Toc505015929)

[Figure 6 Task Manager ~ Fallin 7](#_Toc505015930)

[Figure 7 Sprint View ~ Berkner 8](#_Toc505015931)

[Figure 8 Code Base View ~ Berkner 8](#_Toc505015932)

[Figure 9 Timeline View ~ Collins 9](#_Toc505015933)

[Figure 10 Use Case Diagram 17](#_Toc505015934)

[Figure 11 Application Topology 18](#_Toc505015935)

# **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
  + <https://www.python.org/downloads/release/python-360/>
  + <https://stackoverflow.com/questions/372885/how-do-i-connect-to-a-mysql-database-in-python>
* 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**

**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**

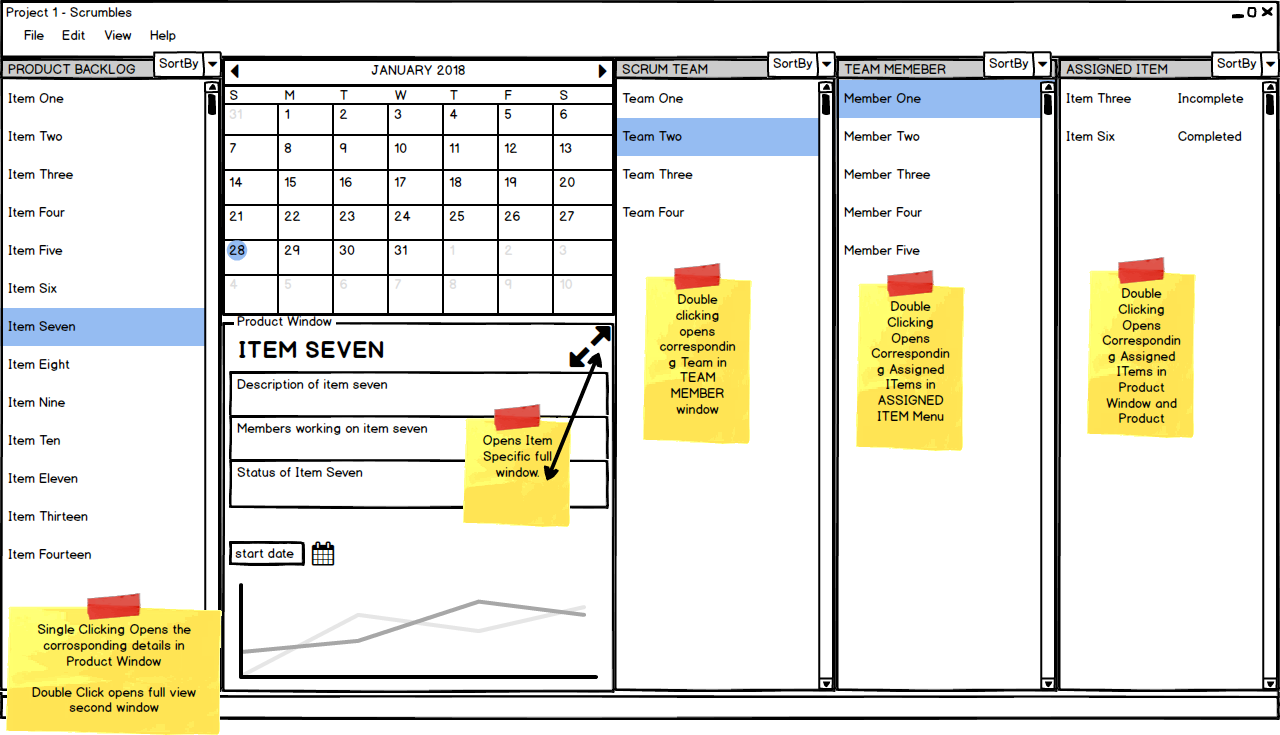


Figure 1 Dashboard ~ Feeney

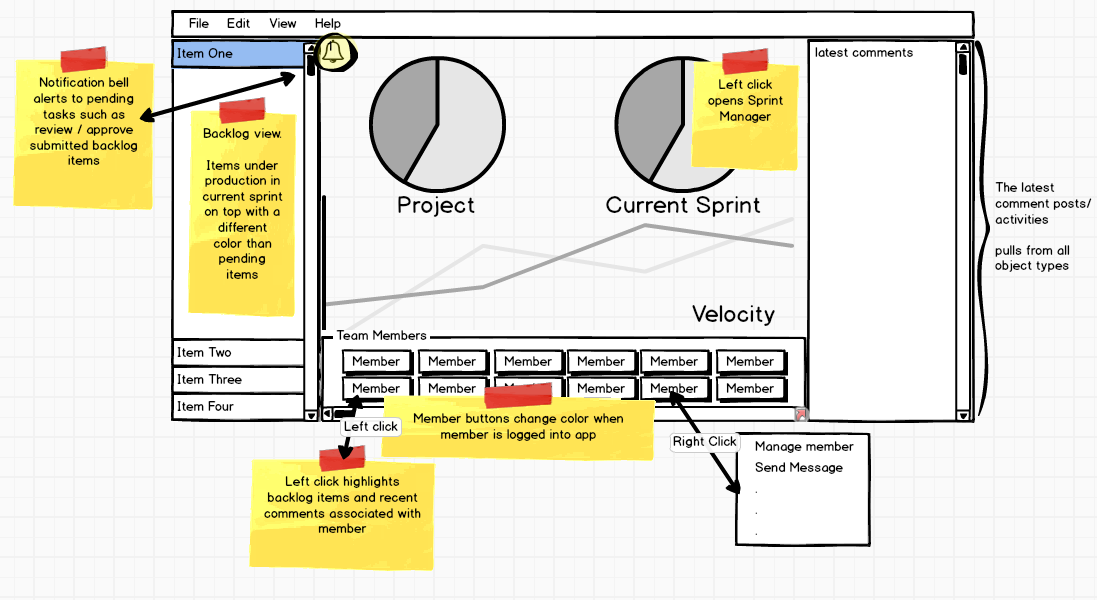


Figure 2. Scrum Master Dashboard ~ Fallin

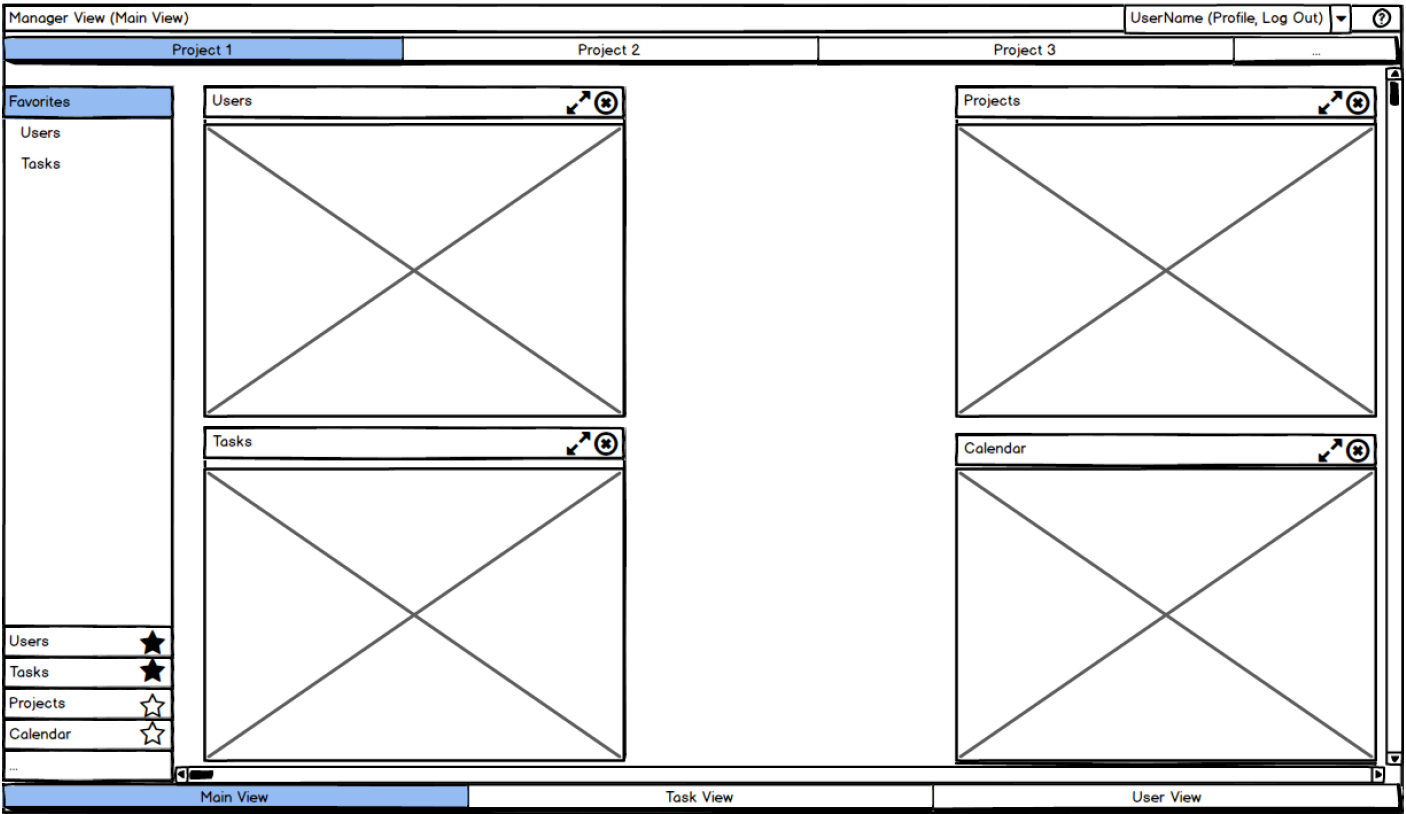


Figure 3 Project Manager ~ Martinez

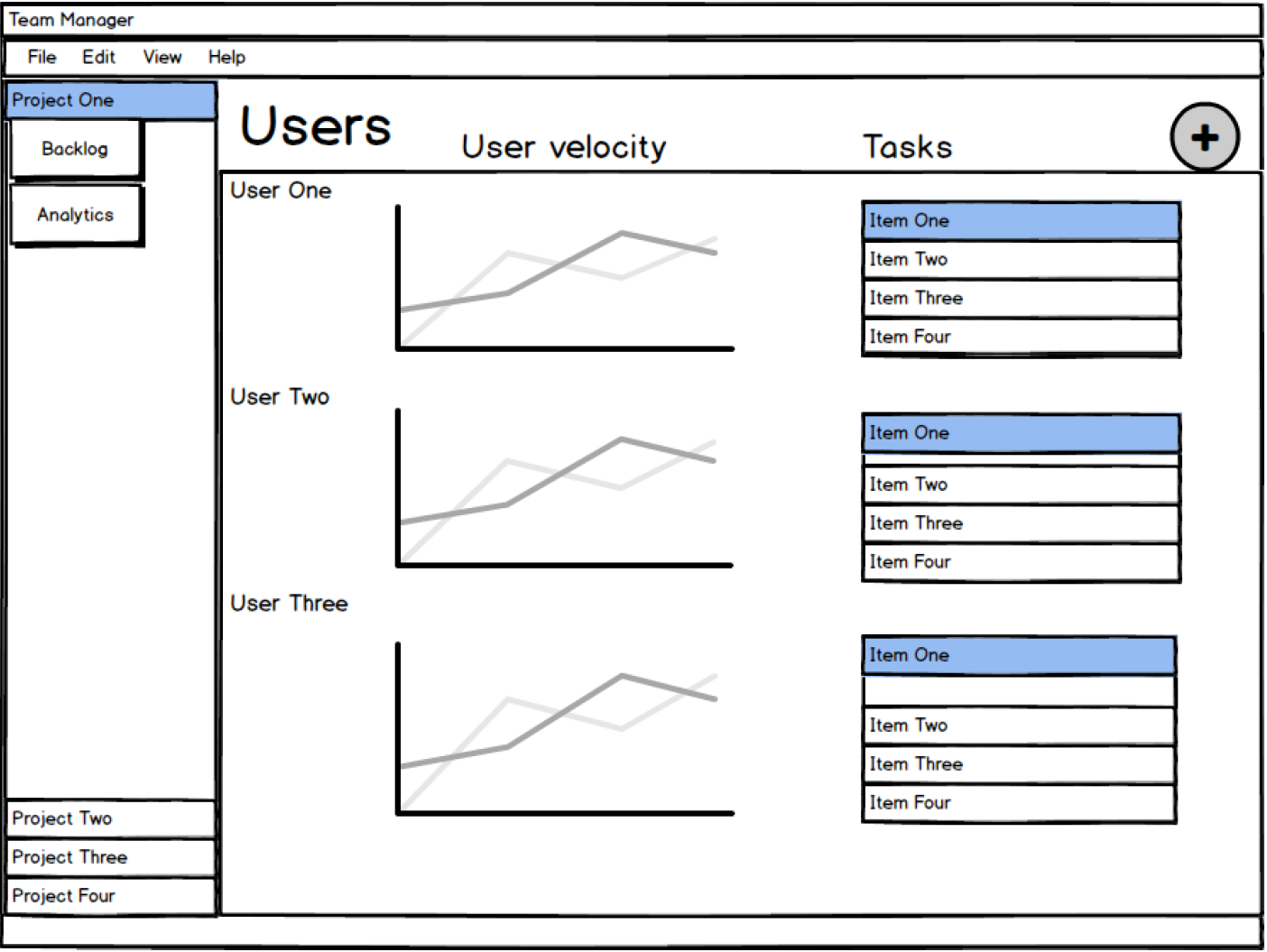


Figure 4 User Manger ~ Strother

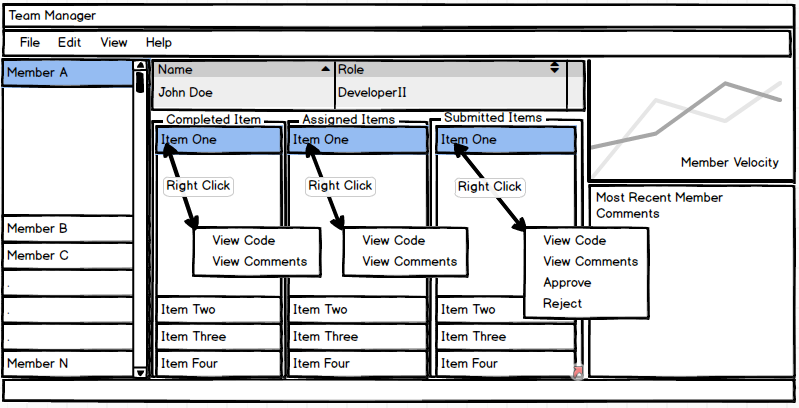


Figure 5 User Manager ~ Fallin

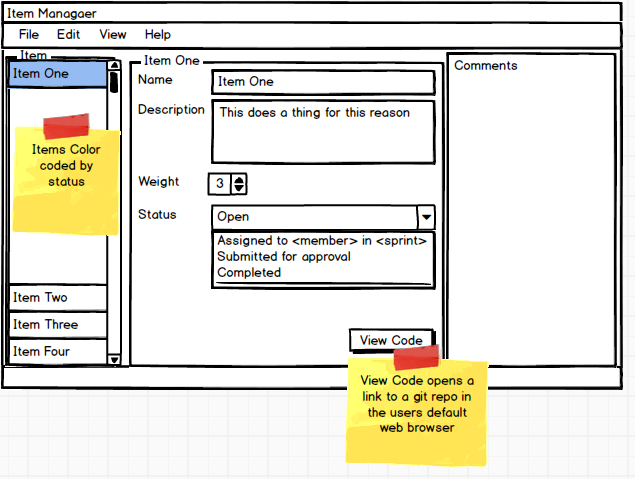


Figure 6 Task Manager ~ Fallin

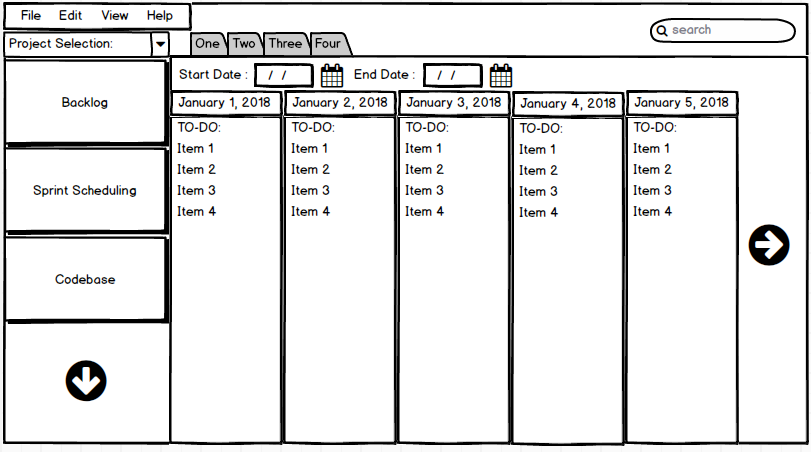


Figure 7 Sprint View ~ Berkner

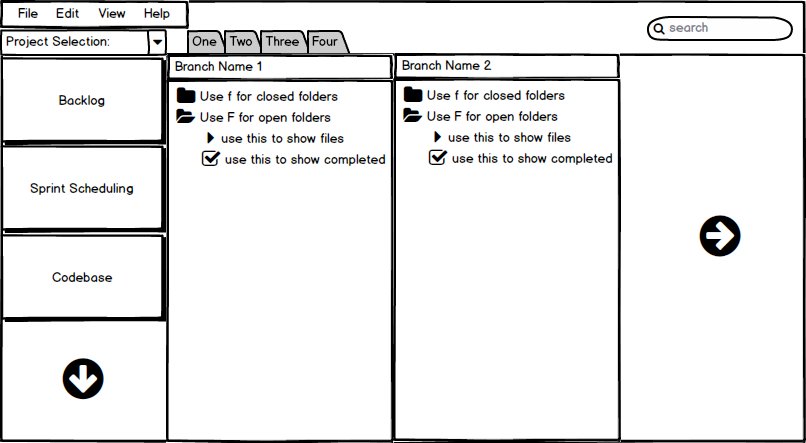


Figure 8 Code Base View ~ Berkner

# 

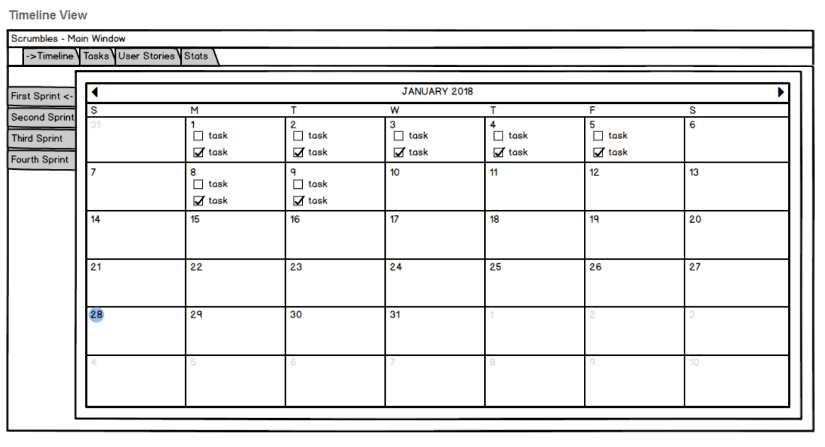


Figure 9 Timeline View ~ Collins

# **3 FUNCTIONAL REQUIREMENTS**

### **3.1 Main Features**

* GUI
* Visual Backlog Queue
* Ability to create Users
* Ability to create Backlog Items
* Ability to create Sprints with start dates and due dates
* Ability to drag users to Backlog Items to assign tasks
* Ability to drag Backlog Items to Sprints
* Automatically suggest Sprints based on Backlog priority Queue
* Add comments to Backlog Items or Sprints
* View Latest Code associated with Backlog Item via http link
* Data stored on external database accessible to multiple instances of application

### **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 | Administrator is logged into the application |
| 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 | Developer |
| Description | When a task is complete, a developer may click a button to submit the task to be reviewed by the Scrum Master |
| Precondition | Code linked to task or Comment attached to task |
| Trigger | Submit button clicked |
| Postcondition | Task status field is updated to reflect pending review in the database |
| Main Success Scenario | Task appears in the pending review column in the task manager view |
| Extensions | If there are no code links or comments the task action will fail alerting the user that a code must be linked or the task must have at least one comment |

|  |  |
| --- | --- |
| Link an Item to Code | |
| Primary Actor | Developer |
| Description | Developer can input a http address to relative code hosted on a server, ideally a git repo that interfaces with a web server such as GitHub |
| Precondition | User must be assigned to the task |
| Trigger | Address is pasted in a code link text box |
| Postcondition | Task field code link is updated in the database |
| Main Success Scenario | User can click ‘view code’ and a web browser launches on the link |
| Extensions | User can paste any text into the field, if the address is invalid, the user’s browser will handle the error |

|  |  |
| --- | --- |
| View Code Linked to Item | |
| Primary Actor | Developer |
| Description | A button that links an item to the code relevant to the item on github |
| Precondition | An item exists and code on github exists |
| Trigger | Developer clicks ‘View Code’ button in the Item Manager |
| Postcondition | Not applicable |
| Main Success Scenario | If the job works correctly, a browser window will open and the corresponding code on github will open |
| Extensions | If the developer wants to, s/he can open code in an editor instead of github. The code changed in the editor. |

|  |  |
| --- | --- |
| Create A New Sprint | |
| Primary Actor | Scrum master |
| Description | The Scrum Master may create a new sprint. A new sprint includes a list of backlogged items that are to be accomplished during the sprint. The sprint must be assigned to a project in the database. |
| Precondition | A project must be started and there must be a backlog of items to be completed for the project. Dates for the start and end of the sprint are decided and developers are assigned to items |
| Trigger | The scrum master configures and initiates the sprint |
| Postcondition | The database is updated with the items from the backlog and allocated to developers |
| Main Success Scenario | The success scenario is when every developer has a set of items assigned to them and the sprint has established dates. The Sprint Manager shows who is responsible for what item. |
| Extensions |  |

|  |  |
| --- | --- |
| Edit Sprint | |
| Primary Actor | Scrum master |
| Description | The sprint can be edited after it is started. Developers may leave the team, items may be abandoned, dates of the sprint may need to be altered and these changes need to be reflected in the database. |
| Precondition | A sprint has to be started already. |
| Trigger | The scrum master must decide to edit the sprint in the sprint manager by clicking on the edit button |
| Postcondition | Changes to the sprint are reflected in the database and a notification is sent to all members of the sprint that edits have been made. |
| Main Success Scenario | All edits to the sprint are altered in the database and all members of the sprint are notified of the change. |
| Extensions |  |

### **3.3 Use case diagrams**

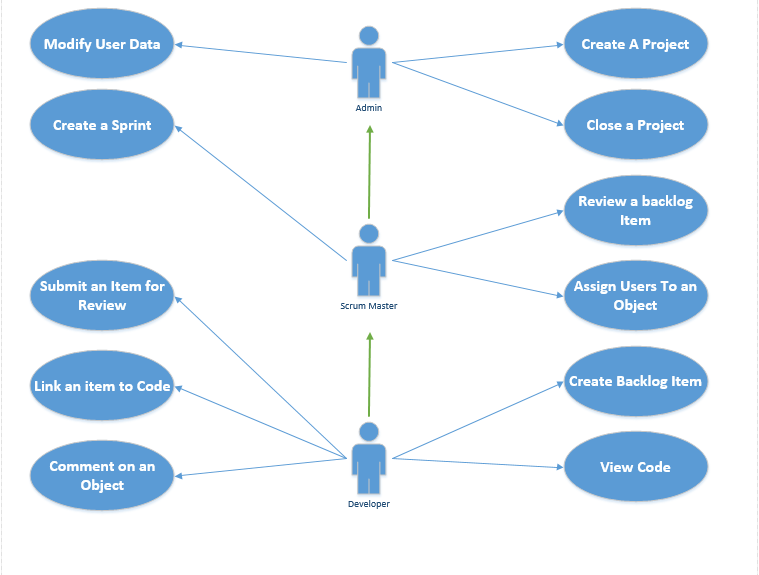
Green Arrow indicates cascaded permissions. 

Figure 10 Use Case Diagram

# **4 EXTERNAL INTERFACE REQUIREMENTS**

### **4.1 User Interfaces**

* Standard monitor
* Mouse
* Keyboard

### **4.2 Hardware Interfaces**

Application will utilize the Python interpreter as a layer between the application and hardware for cross platform compatibility. The user pc must physically be connected to the network on which the database resides. Installation technician will configure application with database IP information

If the database is hosted on an external network behind a firewall, either a VPN tunnel must be established or the firewall must be configured to allow the incoming packets from the application IP address on port 3306 and forward port 3306 to the IP of the appliance hosting the database.

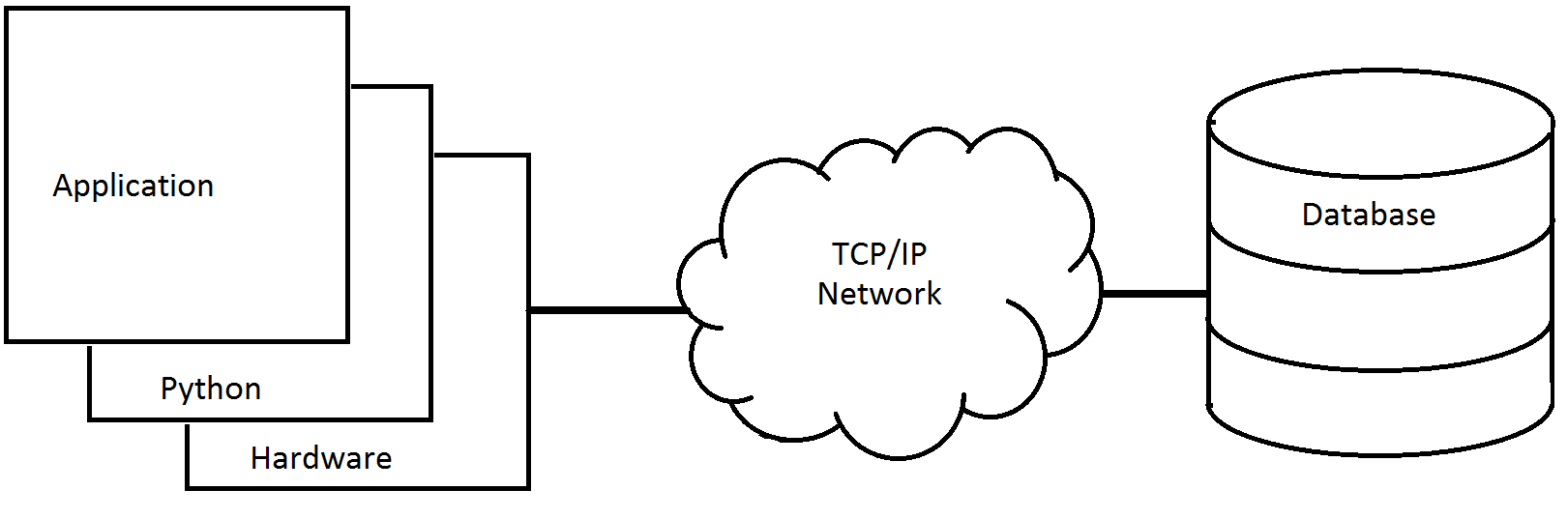


Figure 11 Application Topology

### **4.3 Software Interfaces**

* Python 3.6.4 is required to be installed on the user pc
  + https://www.python.org/downloads/
* Python MySQLdb libraries required
  + <python directory> /scripts/pip install mysqlclient
* Remote MySQL database version 5.7.20-0ubuntu0.17.10.1 (Ubuntu)
* MySQL database runs on Linux Ubuntu 17.10 (GNU/Linux 4.13.0-25-generic x86\_64)
* Application will interface with database over TCP/IP on port 33306

Supplementary information for MAC users: https://www.python.org/download/mac/tcltk/

| **Python Release** | **Installer Variant** | **macOS Release** | **Recommended Tcl/Tk** | **Alternate Tcl/Tk** | **Not Recommended** |
| --- | --- | --- | --- | --- | --- |
| [3.6.3](https://www.python.org/downloads/release/python-363/), [2.7.14](https://www.python.org/downloads/release/python-2714/) | 64-/32-bit | 10.13 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.12 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.11 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.10 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.9 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.8 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.7 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) | [Apple 8.5.9](https://www.python.org/download/mac/tcltk/#apple-8-5-9) |  |
| 10.6 | [ActiveTcl 8.5.18.0](https://www.python.org/download/mac/tcltk/#activetcl-8-5-18-0) |  | [Apple 8.5.7](https://www.python.org/download/mac/tcltk/#apple-8-5-7) |
| [2.7.14](https://www.python.org/downloads/release/python-2714/) | 32-bit-only | 10.5 | [ActiveTcl 8.4.20](https://www.python.org/download/mac/tcltk/#activetcl-8-4-20) | [Apple 8.4.7](https://www.python.org/download/mac/tcltk/#apple-8-4-7) |  |

### **4.4 Hardware Requirements**

* CPU: 1.2 Ghz processor
* RAM: 1 GB
* Network: 10/100 MBS Ethernet or 802.11n Wireless LAN