PROJECT PLANNING

Team Members:
Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

Table of Contents

1. Introductory Questions

- What tools, if any, will you use for your task board?
  - ZenHub
- What tools, if any, will you use for your burn-down chart?
  - ZenHub
- Who will maintain the burn-down chart? How?
  - ZenHub will automatically update the burn-down chart according to the number of completed tasks. Mike Petrella is responsible for making sure that the burn-down chart is created correctly for each sprint.
- What is every team member's role?
  - Maintaining burn-down chart: Mike Petrella
  - Proofreading: Mark Viola
  - Master Branch Organizer: ZiXian Chen
  - Meetings Manager: Hyeong Ook Yu
  - Deadline Reminder: Michael Liu
- What tools, if any, will you use for communication?
  - Facebook Messenger
  - Phone call/ texting
  - Discord/Skype
- When do you plan to meet in person?
  - Tuesday 2pm - 4pm
  - Wednesday 12pm - 2pm
- How will you use your repository on GitHub?
  - Use branching to deal with potential conflicts
  - ZenHub has GitHub integration to allow viewing of the task board and burndown chart
- Which machines will be used to development by each team member? E.g., the lab machine, a Linux laptop, a Windows home computer, etc.
  - Personal Mac/Windows computers

2. Product Backlog

2.1 Plugin Support

Priority: High
Cost: 12 Story Points

As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programming language that the application can use to extract data from other catalogs.

Tasks:

1. Build/find a class library to make it easier to deal with web requests.
2. Create plugin interface.
3. Create a XML parser that will take a dictionary and output a XML format text, with fields in the same order as dictionary.
4. Create serializer class and add plugin loading functionality
5. Create two default plugins for exoplanet.eu and NASA Exoplanet Archive

---

2.2 Standardizing Units

Priority: High
Cost: 8 Story Points

As Hanno (an administrator), I want each plugin to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.

Tasks:

1. Find/build a unit conversion library
   - Create a way of making general conversion (i.e., specify original unit and convert to whatever is defined in the settings)

2.3 Scheduling a Check for Updates

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want the application to check the other exoplanet databases it is tracking, for new updates, on a specified interval (default is daily at 11:59pm).

Tasks:

1. Create a server framework that will run the plugins with threads
2. Have server constantly check system time

2.4 User Permissions

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Tasks:

1. Create GitHub API management class, and add oauth2 login functionality
2. Add ability to check if user is a collaborator of the repo

---

2.5 Application Making Pull Requests

Priority: High
Cost: 20 Story Points

As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.

Tasks:

1. Add repo management functionality to GitHub API class
2. Add the ability to make pull requests to GitHub API class
3. Integrate GitHub API into a server so that pull-request are made with each new change
4. Add XML file reading to serializer
5. Add the ability to update a XML file with changes to serializer

---

2.6 Accepting/Rejecting Pending Pull Requests

Priority: High
Cost: 5 Story Points

As Irving (a user), I want to be able to accept, reject, or amend pull requests made by the application through a user interface by pressing a button on a pop-up window.

Tasks:

1. Add the ability to accept, reject, and change pull requests
2. Add pull-request to GUI
    - Add viewing
    - Add accepting/denying
    - Add sorting to display
    - Add editing

2.7 Source of the Update

Priority: Medium
Cost: 2 Story Points

As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull request in the database.

Tasks:

1. Create a field that shows all of the publications related to a specific system. Shown when clicking on the system from the list of altered systems.
2. Make it mandatory for plugins to extract publication data

2.8 See History

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to see the history of pull requests, on a user interface.

Tasks:

1. Create a button on the Home screen that allows the user to go into the History screen.
2. Create a History screen that displays a list of the 20 most recent pull requests made by the GitHub bot.

2.9 Edit Pending Pull Requests

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to be able to edit a pending pull request through a user interface by pressing an edit button when shown a list of new changes.

Tasks:

1. Create an edit button that opens an interface that allows the user to edit a specified XML file.
2. Update the user's current commit to the one that was edited using the interface.

2.10 History of Updates

Priority: Medium
Cost: 4 Story Points

As Irving (a user), I want to see the planets that have been most recently added/edited by the application in the OEC, using the user interface.

Tasks:

1. Use GitHub API and get the most recent changes
2. Format and parse data from the API and display it in the GUI

2.11 Force a Check for Updates

Priority: Medium
Cost: 3 Story Points

As Hanno (an administrator), I want to be able to force the application to run even if it is not scheduled to check the databases.

Tasks:

1. Create a button in the Home screen that allows the application to check the tracked databases for updates.

2.12 Creating a User Interface

Priority: Medium
Cost: 49 Story Points

As Irving (a user), I want to be able to interact with the application using a user interface.

Tasks:

1. Research UI implementation
2. Create basic UI application
3. Add GitHub login UI
4. Add force check button
5. Add a status checker
6. Add pull request viewer to UI
7. Add plugin management to UI
8. Add settings editor to UI

2.13 Difference Between Updates

Priority: Low
Cost: 25 Story Points

As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.

Tasks:

1. Add repo checking to GUI
    - Add the ability to see recent changes compared to the master branch
        - Create an interactable list of all recent changes
        - Create a GUI that will display changes of the XML files

---

2.14 Settings Form

Priority: Low
Cost: 15 Story Points

As Hanno (an administrator), I want to be able to change time controls, bot account settings, and conversion units, in a settings form in the application.

Tasks:

1. Use C# settings designer to create a settings form and maintain user settings
2. Add timed update controls to user settings
3. Add GitHub bot user account specification to user settings
4. Add a list of possible conversion units to user settings
5. Integrate user settings into the server with on demand changing
6. Add user settings to GUI

---

3. Release Plan

Sprint 1:

- As Hanno (an administrator), I want each plugin I've created to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.
- As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programing language that the application can use to extract data from other catalogs.
- As Hanno (an administrator), I want the application to check other exoplanet databases for new changes on a specified interval (default is daily at 11:59pm).
- As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Sprint 2:

- As Hanno (an administrator), I want the program to make pull requests to the OEC database when it finds changes in the databases that the application is monitoring, using an automated GitHub user account.
- As Irving (a user), I want to be able to accept or reject pull requests made by the application through a user interface by pressing a button on a pop-up window. (**Only the backend half of this story**)
- As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull-request in the database. (**Only the backend half of this story**)
- As Irving (an user), I want to be able to edit a pending pull request through a user interface by pressing an edit button when shown a list of new changes.
- As Irving (a user), I want to see the history of pull requests, on a user interface. (**Only the backend half of this story**)

Sprint 3:

- As Irving (a user), I want to see the planets that have been most recently added/edited by the application in the OEC, using the user interface.
- As Irving (a user), I want to be able to interact with the application using a user interface. (**Only half of this story**)
- As Hanno (an administrator), I want to be able to force the application to run even if it is not scheduled to check the databases.

Sprint 4:

- As Irving (a user), I want to be able to interact with the application using a user interface. (**Other half of this story**)
- As Irving (a user), I want to see the history of pull requests, on a user interface.

Sprint 5:

- As Hanno (an administrator), I want the website source of the new changes to be a comment in the pull-request in the database.
- As Irving (a user), I want to be able to click on a newly updated system from a list in the UI and have lines highlighted, showcasing which parts in the XML file have changed.
    - The lines that were changed will stand out from the ones that didn't. OR be stored in a separate text file and will be displayed using a simple table from the UI. The text file will contain the line number and changes for the lines that were changed.
- As Hanno (an administrator), I want to be able to change time controls, bot account settings, conversion units in a settings form in the application. (**Half**)

Sprint 6:

- As Hanno (an administrator), I want to be able to change time controls, bot account settings, conversion units in a settings form in the application. (**Half**)
- Finish leftovers and testing

4. Sprint Plan

4.1 Sprint Backlog

For the first sprint (10/17/2016 – 10/23/2016), the user stories that will be completed are the users stories found in 2.1, 2.2, 2.3, 2.4. Each sprint will have a sprint velocity of 30 story points and will be 7 days long, starting on the Monday of every week. The developers at Spazio have estimated that they can each complete 6 story points per week. The following, details the user stories being completed and their corresponding tasks.

4.1.1 Plugin Support

Priority: High
Cost: 12 Story Points

As Hanno (an administrator), I want plugin API support for the application, where I can create, add, modify plugins written in a programming language that the application can use to extract data from other catalogs.

Task 1. Build/find a class library to make it easier to deal with web requests. **(2.5 story points)**
Task 2. Create plugin interface. **(1 story point)**
Task 3. Create a XML parser that will take a dictionary and output a XML format text, with fields in the same order as dictionary. **(2 story points)**
Task 4. Create serializer class and add plugin loading functionality. **(2.5 story points)**
Task 5. Create two default plugins for exoplanet.eu and NASA Exoplanet Archive. **(4 story points)**

4.1.2 Standardizing Units

Priority: High
Cost: 8 Story Points

As Hanno (an administrator), I want each plugin to be able to standardize units of measurements found in its catalogue to one that I specify in the settings of the application.

Task 6. Find/build a unit conversion library. **(8 story points)**
- Create a way of making general conversion (i.e., specify original unit and convert to whatever is defined in the settings)

4.1.3 Scheduling a Check for Updates

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want the application to check the other exoplanet databases it is tracking, for new updates, on a specified interval (default is daily at 11:59pm).

Task 7. Create a server framework that will run the plugins with threads. **(4 story points)**
Task 8. Have server constantly check system time. **(1 story point)**

4.1.4 User Permissions

Priority: High
Cost: 5 Story Points

As Hanno (an administrator), I want to use GitHub repo collaborators as a way of distinguishing which user has access to use the application (manage pull-requests, run the server).

Task 9. Create GitHub API management class, and add oauth2 login functionality. **(4 story points)**
Task 10. Add ability to check if user is a collaborator of the repo. **(1 story point)**

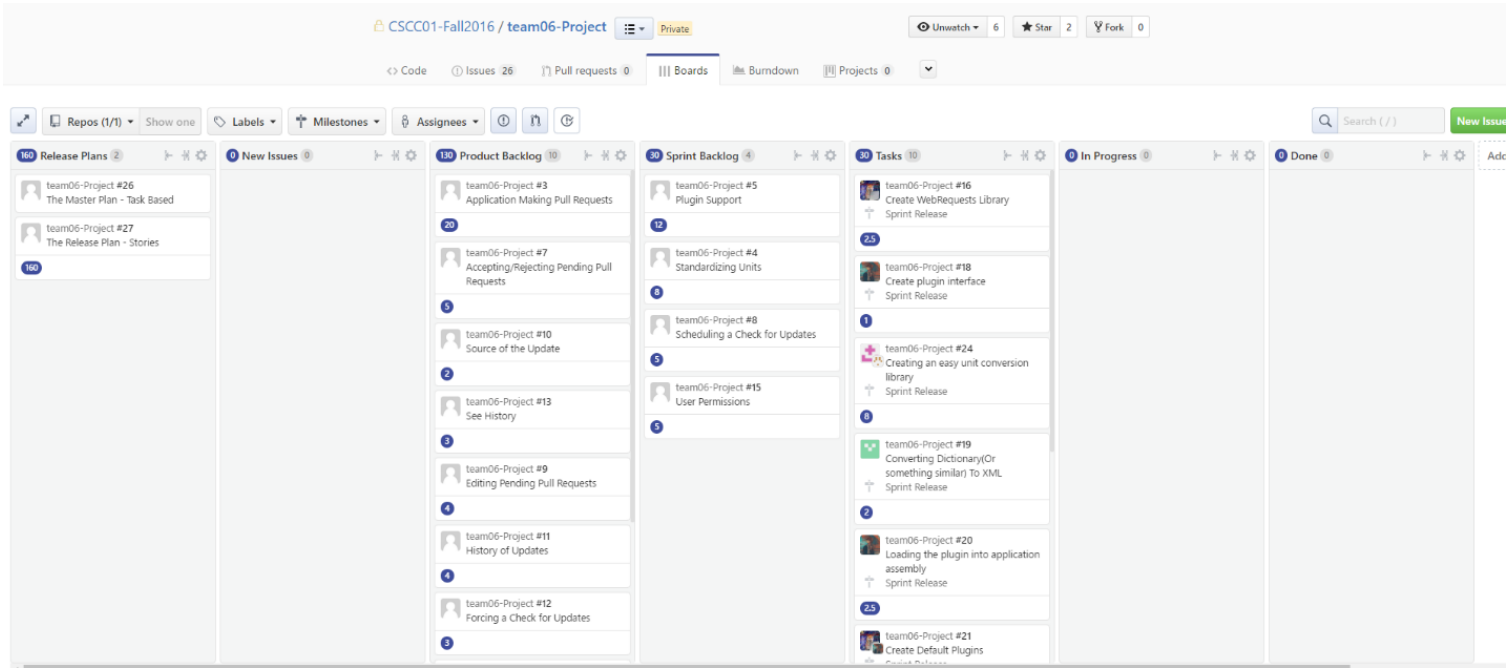
4.2 Detailed Plan

	Oct 17	Oct 18	Oct 19	Oct 20	Oct 21	Oct 22	Oct 23
Task 1		2.5					
Task 2		1					
Task 3			2				
Task 4				2.5			
Task 5				2.5	1.5		
Task 6			6 + 2				
Task 7					4		
Task 8						1	
Task 9					4		
Task 10						1	

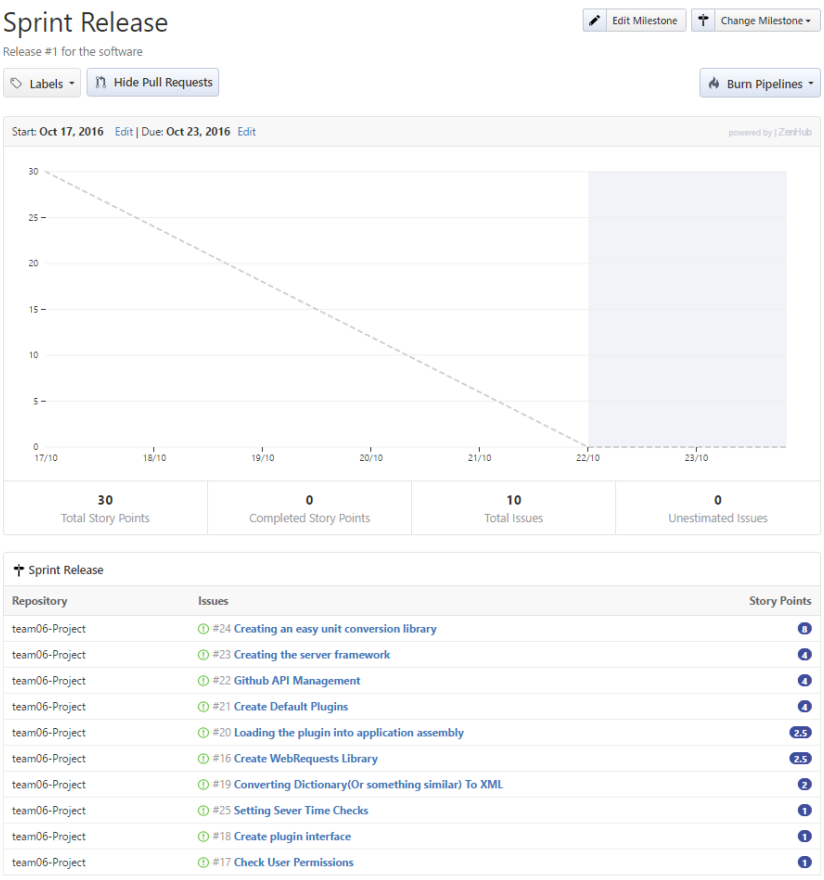
Legend:

Mark Viola
Mike Petrella
Hyeong Ook Yu
ZiXian Chen
Michael Liu

4.3 Task Board Snapshot



4.4 Burn-Down Chart Snapshot



5. System Components

5.1 Plugin Manager/Interface

The plugins are the modular components that allow for the interaction between the various exoplanet databases and the application. Creating a plugin interface allows the addition of new exoplanet databases to be fairly simple since functions and their return values will be defined, so as long as those conditions are met, then the plugin can seamlessly interact with the application. Furthermore, the plugin manager allows for interaction between the plugins themselves.

5.2 Server

The server is where the application will run indefinitely. Periodically (as a defined interval), the application will check to see if there are updates to any of the exoplanet databases its tracking, and will send pull requests accordingly.

5.3 User Settings

The user settings are recognized by the application and will determine how it operates. In particular, the user settings allows the administrator to be able to change time controls, bot account settings, and conversion units.

5.4 GitHub API Handler

The GitHub API Handler is responsible for being the interaction between GitHub and the application. The administrator will set the user permissions on GitHub, but to be able to use the application, the user will need to log into GitHub so that their permissions can be verified.

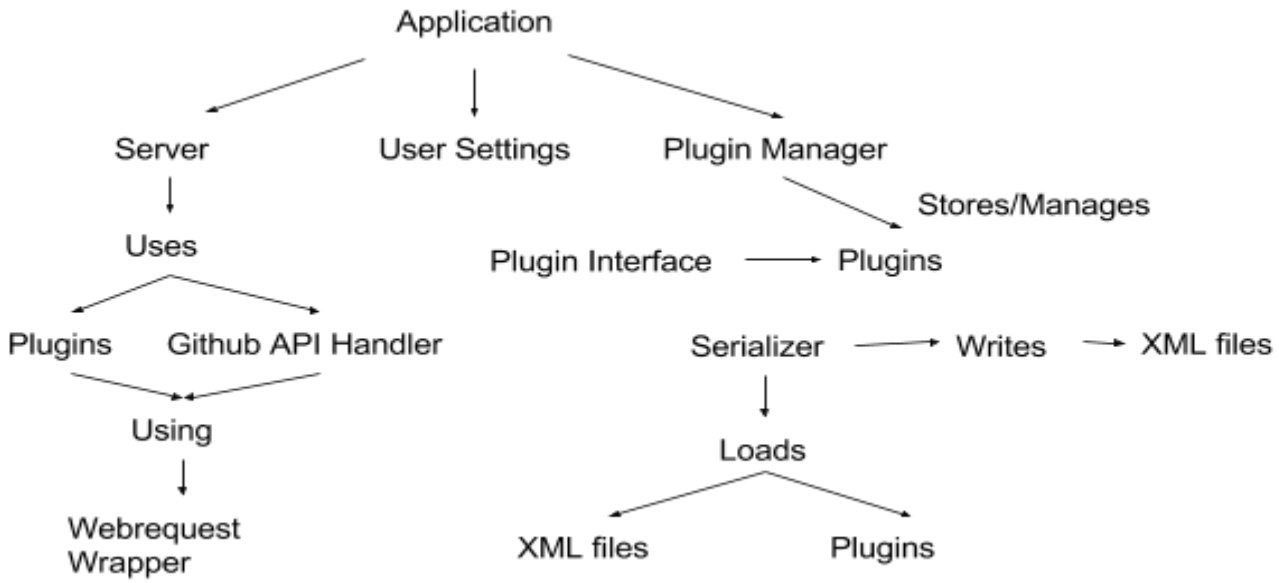
5.5 Web Request Wrapper

The web request wrapper is a class that allows for much easier interaction with making web requests.

5.6 Serializer

The serializer allows the application to communicate with XML files. The serializer will primarily be used to write object instances (such as planet systems, stars, and planets) as XML files so that the can be compared to the XML files found in the OEC. Furthermore, the XML files generated and found in the OEC can get loaded as object instances using the serializer.

5.7 System Components Diagram



6. Sprint Results

6.1 Project Progression

The project has seen great improvement from deliverable 2 to its current state. The user stories in the product backlog have been refined to be more specific so that time costs can be much easier to identify. Additionally, specific tasks have been detailed for each of these user stories. The largest update is the creation of an executable program that can perform web requests to retrieve recent updates in the tracked exoplanet databases (Exoplanet.EU and NASA Exoplanet Archive).

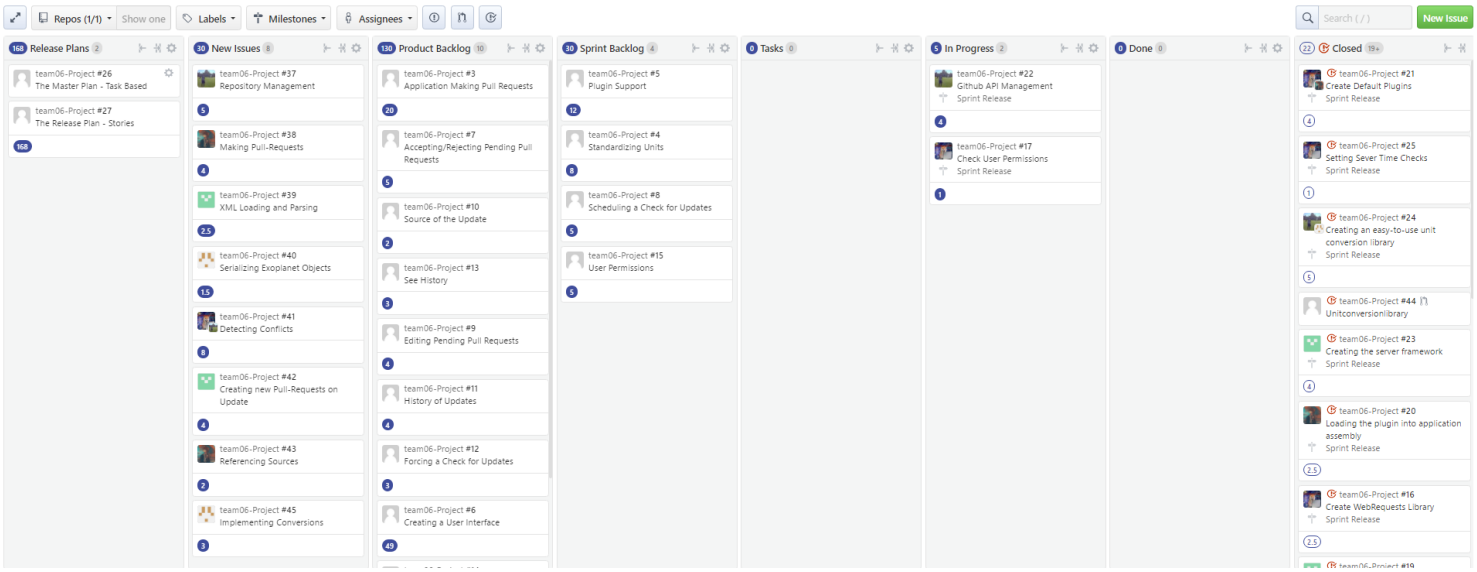
6.2 Project Velocity

Projected project velocity: 30 Story points

Actual project velocity: 22 Story points

The actual project velocity was much lower than the team expected. This was primarily due to the fact that Spazio team member had additional commitments (such as midterms and assignments). The team may have been too optimistic about their goals for this sprint, given this particular week's workload, and they will take more careful consideration in project planning under similar circumstances.

6.3 Post-Sprint Task Board Snapshot



6.4 Post-Sprint Burn-Down Chart Snapshot

6.5 Conclusion

Upon concluding the first sprint, team Spazio has determined that it has been fairly successful, especially considering the conditions they have faced (high workload from other commitments). The priority of the first sprint was to get a working executable created so that team can have a base to build up on, and they have achieved this. The current build of the program has plugin framework support to allow for easier implementation and use of plugins for the exoplanet databases, and there is functionality to allow for actually retrieving the data from these exoplanet database.