Open Exoplanet Catalogue

Deliverable 4: Overview

By: Team 5

OEC Table of Contents

Additional Persona

1.1 Assistant Ross Geller

- 24 years, male.
- Graduate student at University of Toronto studying Astrophysics.
- Works part time (2 days per week) for Professor Cooper.
- Currently enrolled in 3 courses, and also involved in the Astronomy and Space Exploration Society (UTASX) at U of T.
- Uses Facebook, twitter on a daily basis.
- Good computer skills - capable of using computer software such as Microsoft Office and Google Drop box.
- Has not been exposed to highly technical software such as GitHub.
- Carries personal windows laptop, sometimes uses his friend's MacBook, but have never heard of Linux or Unix operating systems.
- Prefers intuitive software.
- Hardworking when involved in Astrophysics.
- Becomes frustrated when caught in technical difficulties such as a printer connection issue (i.e. no patience for technology)
- Capable of following instructions; Prefers tutorials for software/tools compared to learning by trial and error.

Product Backlog

Priority Scale (Low 1 - 5 High)
Cost in Story Points (1 Story Point = 1 Developer Hour)

2.1 Download Data from Other Catalogues

User Story ID: 4.1
Priority: 5
Cost: 3

As Prof. Copper, I want to be able to download csv data from other catalogues from a given URL; so that it can then be used to generate updates.

2.2 Only Fetch Data fields in OEC

User Story ID: 4.2
Priority: 5
Cost: 2

As Prof. Cooper, I want to fetch data from other catalogues only if its data field exists in the OEC (i.e. there should not be an update notification if another catalogue updated its value for "Chance of Living Organisms" in Planet X, and the OEC does not have a corresponding XML data value to "Chance of Living Organisms."

2.3 Convert Data from Catalogues into XML

User Story ID: 4.3

Priority: 5

Cost: 6

As Prof. Cooper, I want to automatically generate XML system pages (one system per XML page) for the OEC for data taken from the NASA Exoplanet Archive and Exoplanet.eu.

2.4 Merge Updates and Changes via Pull on Git

User Story ID: 2.9

Priority: 5

Cost: 15

As Prof. Cooper, I want to be able to merge the updates/changes into the OEC via a pull request on GitHub. The program should generate a separate pull request for each XML system page generated.

2.5 Pull Request Notifications

User Story ID: 2.3

Priority: 5

Cost: 3

As Prof. Cooper I would like to be notified via pull requests if an update has been generated (new XML system page created) from monitored catalogues so that I can choose whether or not to update the existing catalogue.

2.6 Daily Update Check

User Story ID: 2.4

Priority: 5

Cost: 3

As Prof. Cooper I would like the system to be able to check for updates daily.

2.7 Convert Data into Standard Units Used by OEC

User Story ID: 2.7

Priority: 5

Cost: 5

As Prof. Cooper, I want updates for the OEC to automatically convert data from the Nasa Exoplanet Archive and Exoplanet.eu into the standard units of measurement used by the OEC.

2.8 No Duplicate Updates

User Story ID: 2.6

Priority: 4

Cost: 8

As Prof. Cooper, I only want to be notified ONCE when discrepancies exist between other catalogues and the OEC (and the other catalogue was updated more recently than the OEC) - unless the other catalogue's data containing the discrepancy is updated again later.

2.9 Manually Initiate Update Check

User Story ID: 2.5
Priority: 3
Cost: 2

As Prof. Cooper I would like to be able to manually initiate an update check via terminal command.

2.10 OEC Commit Messages

User Story ID: 2.8
Priority: 3
Cost: 3

As Prof. Cooper, I want updates for the OEC to contain commit messages that specify the reference URL for each update (i.e. the URL for that planet in either the Nasa Exoplanet Archive or Exoplanet.eu), and contain a hash tag identifying that the update was automatically generated by the program.

2.11 Data Value Change Threshold

User Story ID: 2.10
Priority: 3
Cost: 4

As Prof. Cooper, I want the program to not generate updates if the only differences in data values are below certain predefined thresholds (to be provided by client/specialist TA and do not include changes below the currently displayed number of decimals).

2.12 First Synchronization

User Story ID: 2.11
Priority: 2
Cost: 5

As Prof. Cooper, I want the first run of the program to check all planets in the other catalogues and create pull requests for all available updates for planets that have been updated more recently in the other catalogues than in the OEC. The first run should set a benchmark for future runs (i.e. monitor the dates that the planets were last updated in the other catalogues, so that no future updates are generated if the date of last update has not changed.)

2.13 Delete Catalogue

User Story ID: 2.2

Priority: 1
Cost: 2
As Prof. Cooper I would like an option to remove catalogues that are currently being monitored via a terminal command, so that they are no longer monitored for updates.

2.14 Specify Catalogues to Monitor

User Story ID: 2.1
Priority: 1
Cost: 3
As Prof. Cooper I would like to be able to specify URL's to be monitored for updates (specifically the Nasa Exoplanet Archive and Exoplanet.eu). I would also like to be able to monitor additional URL's in future but understand that I will need to go into the python code and provide a mapping between the data in the other catalogue and the XML files in the OEC.

2.15 Handle Human Error while Updating Data

User Story ID: 2.13
Priority: 1
Cost: 15
As Prof. Cooper, I want the program to try and identify data in other catalogues containing possible typos/human error (hypothetical example being "Kepler-16 b" is "Kepler 16 b") and attempt to match it with the corresponding XML page in the OEC. In the given example this would mean creating a pull request for "Kepler-16 b" (OEC name) instead of "Kepler 16 b" (Nasa name).

2.16 Git Tutorial

User Story ID: 2.14
Priority: 1
Cost: 1
As Rose Geller (Assistant of the Professor), I want a step-by-step instruction for basic operations of GitHub such as to clone, pull, merge, etc. so that I can use the program if needed.

Release Plan

Sprint Duration: 7 Days (including weekends).

3.1 Sprint 1 (Deliverable 3)

Oct 17 - Oct 23
The initial release 0.1 will contain basic program functionality (core functions upon which all future functions will be based). Release 0.1 will be able to download csv files from a given URL, and given a mapping of csv columns to XML tags will be able to

generate a well-formatted XML page in the style used by the OEC (1 system per page) for each exoplanet in the csv file.

User Stories Implemented:

4.1 Download Data from Other Catalogues

4.2 Only Fetch Data fields In OEC

4.3 Convert Data from Catalogues into XML

3.2 Sprint 2

Oct 24 - Oct 30

Release 0.2

User Stories Implemented:

2.9 Merge Updates and Changes via Pull on Git

2.3 Pull Request Notifications

3.3 Sprint 3

Oct 31 - Nov 6

Release 0.3

User Stories Implemented:

2.7 Convert Data into Standard Units Used by OEC

2.4 Daily Update Check

3.4 Sprint 4 (Deliverable 4)

Nov 7 - Nov 13

Release 0.4

User Stories Implemented:

2.12 Handle Alias while Updating Data

2.6 No Duplicate Updates

3.5 Sprint 5

Nov 14 - Nov 20

Release 0.5

User Stories Implemented:

2.10 Data Value Change Threshold

2.5 Manually Initiate Update Check

2.8 OEC Commit Messages

3.6 Sprint 6 (Deliverable 5)

Nov 21 - Nov 27

Release 1.0

User Stories Implemented:

2.11 First Synchronization

2.1 Specify Catalogues to Monitor
2.2 Delete Catalogue
2.13 Handle Human Error while Updating Data

3.7 Sprint 7

Release 1.1
Nov 28 – Dec 1(4 Days)
User Stories Implemented:
2.14 Tutorial

Our Release Plan has slightly changed as we have added a Sprint 7 to implement a newly added user story (2.14).

Evidence of System Validation

4.1 Description and Evidence of System Validation Activities

Our team have ensured that we have built the “right product” by sending emails to our client (Professor Hanno) and also attending his office hours to seek answers regarding system design and technical implementation. Each feature that was implemented throughout the deliverable 3 and 4 closely matches the requirements provided by Professor Hanno. For example, we have implemented update notifications via Git pull requests and also kept the installation process of our software to be as simple as possible (as specified by Prof. Hanno). As a result, our team have verified that the software we are currently developing meets the needs and expectations of our client.

Deliverable 4 Overview

5.1 Brief Overview of Project from Deliverable 3 to Deliverable 4

Compared to having only one sprint in Deliverable 3, the three sprints in Deliverable 4 means that our team had more time to implement our software. In this Deliverable, many more features were implemented. At the end of Deliverable 3, our software was only capable of downloading the csv files from the catalogues (NASA and Exoplanet) and creating basic xml files for each planet. However, by the end of Deliverable 4, our software is now able to merge updates, create pull requests, create xml files with converted units, etc. In Deliverable 4, our team was able to successfully implement most of the core features for our software.

5.2 How does the work done for deliverable 4 differ from the work done for deliverable 3, both in terms of progress and end result?

The work done for deliverable 3 is mostly intended to get the team started in implementing the software. At the end of deliverable 3, our software is only capable of generating csv files and creating corresponding xml for those planets. However, the work completed in deliverable 4 implements most of the core features required in the software. In terms of progress, our team was able to get more work done during deliverable 4 mainly because deliverable 4 consists of three sprints whereas deliverable 3 only consists of one sprint. In addition, the time period that deliverable 3 (sprint one) took place, most of our team members were in the middle of midterms and therefor had to focus their time in review/studying for those midterms. On the other hand, our team members were finishing midterms by the second week of deliverable 4 (sprint three) and had more time to work on the project. By the end of deliverable 4, our software now meets the basic standards of the OEC Synchronizer with functionalities to merge updates to the xml files (of the client) and creating pull requests for these updates. This is a large improvement compared to the end result at the end of deliverable 3 where the software is only capable of downloading csv files (from NASA and Exoplanet) and creating basic xml files with the downloaded csv files.