Release Planning Document

COSC4P02

SOFTWARE ENGINEERING II

WINTER 2024

NASER EZZATI-JIVAN

JANUARY 26TH, 2024

Alexandre Reuillon
ar16sv@brocku.ca
6197834

Balkaran Sidhu
bs19xt@brocku.ca
6837322

Lance Brown
lb17sy@brocku.ca
6445944

Manroop Singh Rakhra
mr19mq@brocku.ca
6857551

Marylina Aka Beyeh
lb20sm@brocku.ca
7056260

Nicole Chang
du20lm@brocku.ca
7201841

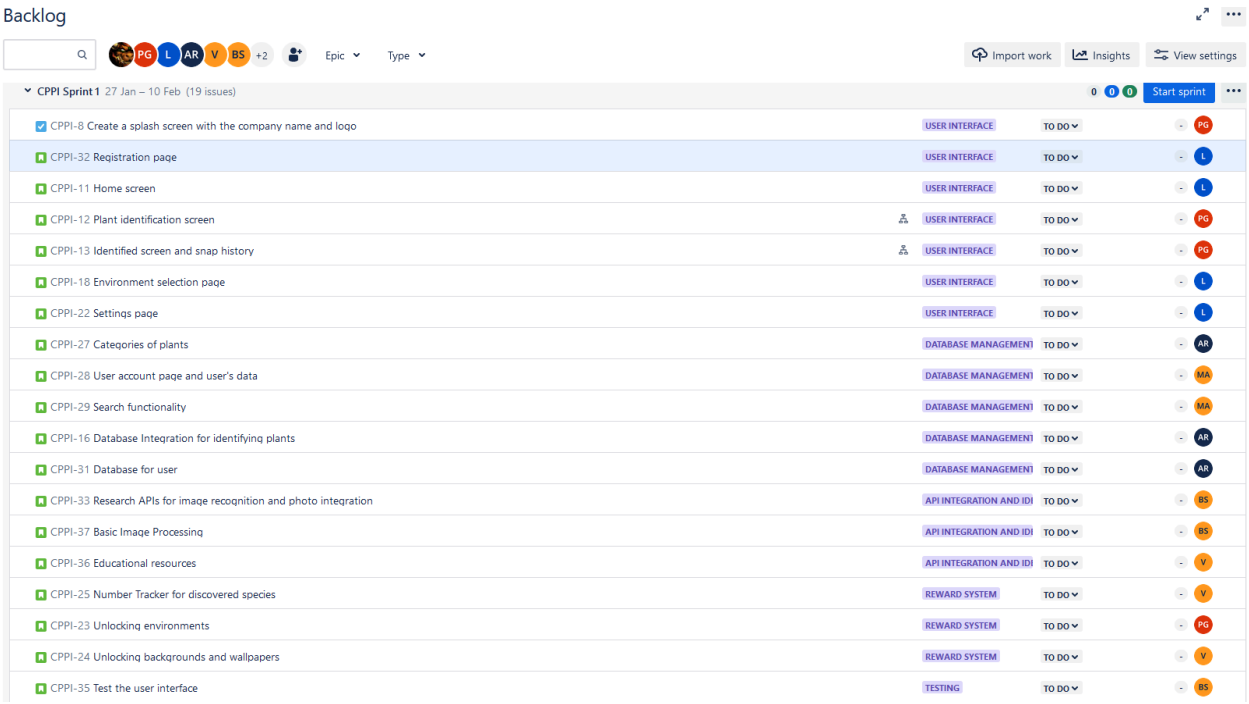
Parneet Gill
pg19uy@brocku.ca
6854640

Vishistha Sharma
vs19rq@brocku.ca
6855944

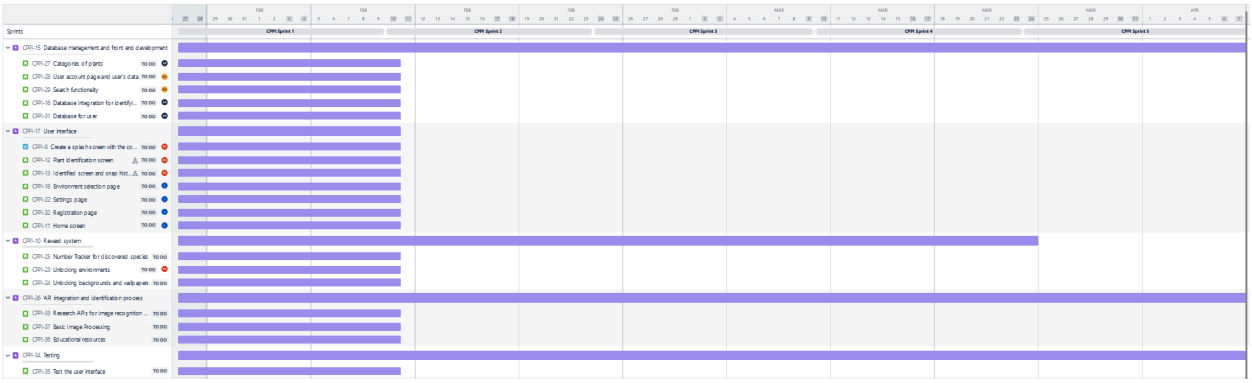
Tools used for product management:

For our project, we decided to use Jira to generate our product backlogs, user stories, epics, tasks/subtasks, and sprint backlogs. Jira is integrated with Github and will be used for our product management.

Backlog of general features and user stories of the application:



Timeline for our upcoming sprint:



Note: Since we are applying an agile scrum method that is both incremental and iterative, requirements and backlogs may be changed and additional features may be implemented throughout the sprint process. Our main goal is to produce a working 'skeleton' code with a functioning user interface for the first sprint and future sprints. That is, we want to produce a working prototype with important functions for the first sprint, then refine the important features – or add more important features if needed – to our backlogs for each future sprint. Not only this incremental and iterative approach lowers risk of overall project failure, job satisfaction is increased for our team who can see early results of their work and see how much has been implemented. It also encourages more rapid delivery and deployment of useful software to the customer. Our method will help us elicit requirements for later increments and receive quick feedback on system performance. We produced a general representation and structure of the app through epics. For now, sprint one is focused on developing the core and basic features of the app. We will test our features for each sprint and record and/or resolve any issues encountered.

Link to our Jira backlogs/user stories and details:
https://cosc4p02plantidproject.atlassian.net/jira/software/projects/CPPI/boards/1/backlog

Architectural design: In terms of software architectural design, our application requires database models, user input and actions, and integration with a presentable, interactive interface. Therefore, we will be following the MVC design pattern for this project.

Functional Requirements:

- The system shall allow users to register by creating a username, password, and valid email address
- The system shall allow users to recover their account by using their email to reset their password
- The system shall allow logged in users to change their account's email address
- The system shall allow users to identify plants by uploading photos of them
- The system should allow users to decide whether to allow notifications for system events/updates or not
- The system shall record the number of plants discovered by the user, as well as which plants exactly have been found so far
- The system shall contain a library of plants species that can be searched through by the user
- Users shall be provided with basic information about the identified plant species.
- The identified plant names should be displayed clearly.
- Users should be able to access more detailed information about the identified plant (country of origin with world map, etc.)
- The system should provide a virtual environment to showcase discovered plant species

- The system should allow the user to unlock cosmetic items after milestones for the number of identified plants have been met (wallpapers, backgrounds, and appearances for their virtual environment)
- The system should have two separate databases for user specific data and plant data
- The system shall display a splash screen containing the Tree Hugger Co. logo when the user access it
- The system shall identify/flag dangerous plants in a reasonable amount of time
- The application shall implement rate limiting
- The application shall implement sanitization for user input to protect data stored on the server

Non-Functional Requirements:

- The system shall not provide false or misleading information that may be harmful to the general public.
- The system's web pages should be easy to read/navigate
- The system shall use webpages that have environmental themed colour schemes
- Reliability: The app should gracefully handle errors or unexpected situations through error messages to the users.
- The application should be easy to access.
- The application should be compatible with a range of devices.
- The visuals, backgrounds, wallpapers, and environments should be aesthetically pleasing.
- The system should be able to handle a growing number of users and an expanding database/content library.
- The code should be easy to maintain and flexible to new changes.
- The loading time for unlocking and displaying new backgrounds, wallpapers, and environments should be optimized for a smooth user experience.

Domain Requirements:

- The system shall flag dangerous plants, keeping in line with the poisonous plants listed in the Canadian Government's "Poisonous Plants of Canada" document, as well as the known poisonous plants located in other countries the system will be available in