Maintenance Plan

Jonathan Lavoie, Mason Cacheino, Nooh Alavi, Rahif Haffeez, Shawn Xiao

1. Issue Tracking

Issues will be tracked by using GitHub Issues.

a. Issue Creation:

- **Reporting**: Any identified issue should be reported through creating a new issue in the GitHub repository with the bug label and respective priority label. Use the following template to ensure that information is consistent:
    - Template:
        - ➢ Description of the problem
        - ➢ Steps to reproduce the problem
        - ➢ Expected vs. actual output
        - ➢ Any screenshots/logs (if applicable)
        - ➢ Environment details (browser, os, etc)
- **Enhancements:** Any feature should be reported through creating a new issue in the GitHub repository with the enhancement label. Use the following template to ensure that the information is consistent:
    - Template:
        - ➢ Specify enhancement
        - ➢ Description of the feature
        - ➢ Why this feature should be added and how it benefit the user
        - ➢ Similar programs with the feature (if applicable)
- **Improvements:** Any feature should be reported through creating a new issue in the GitHub repository with the feature-request label. Use the following template to ensure that the information is consistent:
    - Template:
        - ➢ Specify improvement
        - ➢ Description of the improvement
        - ➢ How this improvement will benefit the user
- **GitHub Repository Label List:**
    - Bug: Feature isn't working
    - Priority Low: Bug is a low priority
    - Priority High: Bug is a high priority
    - Enhancement: Optimization and improvement of new feature
    - Feature Request: Request a new feature
    - Duplication: Issue already exists

    - Help Wanted: Extra help needed for an issue
    - Invalid: Issue that doesn't fit the template criteria
    - Question: Information request about something
    - Won't Fix: Issue does not seem important

b. **Assignment:**

- Depending on the issue, be sure to assign the correct team members in what they are mainly responsible for. If the issue is dealt with multiple components, assign multiple team members responsible in those components to handle it.

2. Bug Handling

a. **Bug Fix workflow**
    - **Categorize:** First the bug will be categorized into the labels by the user making the report and the components affected are given to assignees that are responsible for that component.
    - **Reproduce the Bug:** The team members then reproduce the bug and determine where it is in the codebase.
    - **Develop a Fix:** Once the bug has been located, develop the fix for it, if necessary, write unit tests/integration tests to make it easier.
    - **Pull Request (PR):** Once the bug has been fixed, send out a PR with the change.
    - **Review of PR:** Other team members can read the PR and determine if the fix is sufficient.
    - **Merge:** Merge the PR with the main branch.
    - **Release:** When pushed to main, wait for the release cycle.

b. **Feature Workflow**
    - **Issue Created:** First the issue is created under the label "feature-request"
    - **Evaluate the Request:** Team members will evaluate the feature request and communicate with each other to see if the feature should be added.
    - **Design and Plan:** If the feature is confirmed, design and planning of how the feature will be implemented will start.
        - This will include who is working on it and where it's located in the codebase, etc.
    - **Develop:** Once a sufficient enough plan has been made development will begin, if necessary, unit and integration testing will also begin.
    - **Pull Request (PR):** Once the feature has been implemented, send out a PR.
    - **Review of PR:** Review the PR and determine if the implementation of the feature is sufficient.
    - **Merge:** Merge the PR with the main branch.
    - **Release:** When pushed to main, wait for the release cycle.

c. **Improvement Workflow**
   - **Issue Created:** First the issue is created with the label “enhancement”
   - **Evaluate the Request:** Team members will then evaluate the optimization request and communicate to see if it is a viable request and who should handle it.
   - **Develop:** Team members who are responsible for the development will implement the feature, making sure there are no regressions to the implementation.
   - **Pull Request (PR):** A PR request is made with the changes
   - **Merge:** The PR gets merged with the main branch
   - **Release**: When pushed to main, wait for the release cycle.

3. Release Cycle

a. **Versioning for GitHub**
   - Semantic Versioning. (major.minor.patch) and tagging each release with a version number.
     - Major Releases: Changes that aren’t backwards compatible
     - Minior Releases: Adding new features and backwards compatible enhancements.
     - Patch Releases: For bug fixes and small improvements that don’t affect the core functionality

b. **GitHub Actions**
   - Everytime a PR is merged into the main branch, automatic testing should occur to verify that the application is running correctly.
   - After successfully passing tests, the app should be deployed to the production environment.

c. **Hotfixes**
   - If a critical bug is found post-release, create a hotfix branch, fix it, and deploy as a patch release.