Version Control System and Collaboration Deliverable

Team: Yasmin Shahed, Brandon Coleman, Rohan Sagar

Repository Architecture Overview:

Broader View:

```
cutter/
├── docs/          # Cutter Documentation
├── team_docs/  # Our docx, pdf files for progress purposes (Also added to Confluence)
├── rizin/         # Rizin submodule
├── docker/        # Cutter Containerization
└── src/           # Cutter Source Code
```

In depth View:

```
src/
├── common/        # Contains common functions like Colors, Decompiler, Worker, etc.
├── core/          # Core application (Major components that bring the app together)
├── dialogs/       # All Popup Dialogs
├── fonts/         # Font files (.ttf format)
├── img/           # Relevant Icons (Cutter logo etc.)
├── menus/         # Context Menus (when user right clicks)
├── plugins/       # Sample plugin files + Plugin manager
├── python/        # Python cutter file
├── themes/        # Theme configuration
├── tools/         # Secondary tools (dialogs) like Find, Search
├── translations/  # Language support files
├── widgets/       # Cutter Widgets
└── Cutter-Application.cpp, Main.cpp   # Starter Files
```

Branching Strategy:

For this project, we will be using GIT for branch development. The project will start with creating a fork off of cutter's 'dev' branch to have our own repository to develop and contribute to Cutter.

The 'dev' branch will be the main integration branch. There will be 3 feature branches stemming from this: issue-1, issue-2, issue-3 for each of the three Cutter issues we picked.

For each issue, we will subdivide it into different issues/tasks that each teammate will be assigned to, which will be in their own branch. Code reviews/pull requests will be merged to their respective feature branch. More than one person may work on a single sub-issue if needed.

Review and Merge Policy:

For the code review policy, as this is a 3 person project, we plan to do a simple method of review. The developer will notify the other two members that their code is ready for review or the reviewers will be notified by a Discord Bot that a commit has been made. Upon which the reviewers will review the code, The reviewers will give a response back in roughly 72 hours. The developer should also be reviewing the code as the reviewers are. During review, the reviewers must follow Cutter's testing procedure (https://cutter.re/docs/contributing/code/release-procedure.html#basic-testing-procedure). Also all feedback messages and commits must have how the developer/reviewer tested the code meant to be reviewed.

- **Approval Count:** 3, The developer and both reviewers must approve of the code before it is merged.
- **Merge Window:** After the code has been approved and reviewed, it can be merged into the issue branch it stems from as soon as possible.
- **Merge Target:** The various subissue branches will merge into the primary issues branches (issue-1, issue-2, issue-3). The issue branches will then merge into the 'develop' branch before being sent to Cutter's 'stable' branch to be reviewed by Cutter devs and to create a release request.
- **Proposal Structure:**
    - For merging into Cutter, you must follow their template:
        - Have read the guidelines for contributing
        - Followed the projects coding style
        - Updated the documentation with relevant information
    - Then you must give a detailed description of the issue being solved
    - Then you must give the "Test plan" which consists of images showing your solution working
    - Lastly, you must say which of the issues is now closed by your solution

Team specific review policy:

- *Keep the scope of your commits small so it's easier for your teammates to review: focus on one specific task for each commit and pull request.*
- Branch protections with linear commit history requirement, aim for a clean commit history.
- *One commit per pull request/code review*
- *When creating a PR/code review, link the existing issue and make sure there is a description of what you added and why you added it.*
- *Do your best to review all code reviews your teammates publish. If you see something in the code that blocks you from wanting to approve it, comment with your reasons why. If you need additional explanation for the code, ask*
- *Include what you did to test your code in the code review description.*
- *Regularly communicate with the team to avoid merge conflicts or other problems*