Sprint 1 Plan

Tasks to Complete:

1. Organize Artifacts: Create a folder named ARTIFACTS to store supplementary documents.
2. Sequence Diagram Creation: Develop the initial sequence diagram for the WARP project.
3. UML Diagram Development: Design UML diagrams for the visualization stub classes.
4. Update Documentation: Update the README file with tasks completed in Sprint 1.

Task Assignments and Workflow:

1. Katelyn will create the ARTIFACTS folder and collaborate with Teagan to design UML diagrams for visualization stub classes.
2. Manthan will create a project plan using Google Docs and share it with the team.
3. Entire Team will Meet to create the sequence diagram, use draw.io for collaborative editing and conduct two team meetings to refine the diagram together.
4. Teagan & Katelyn will develop UML diagrams for visualization stub classes.
5. Victoria & Molly will update the README file, include pointers to the files for grading, and provide a breakdown of task assignments and completions.

Sequence Diagram Instructions:

- Look at the -ra option in the main method of warp.java (line 147)
- Follow the code path back to warp and ultimately, the user to represent the interactions.
- These are the main connections to consider, User, Warp, VisualizationFactory, VisualizationImplementation, FileManager, ReliabilityVisualization,

WorkLoadDescription, WorkLoad, WarpInterface, VisualizationObject, Description, GUIVisualization.

Documentation:

- We will create a rough draft of our plans before starting the whole project
- Before each Sprint, we will refine plans and assign tasks.
- We will share and update the project plans document as tasks change and problems arise.

Quality Assurance (QA):

- We will work together at every stage of the way, so even if a task is assigned to someone, we all have shared responsibility to ensure its quality.
- At every stage of the process, as every item is updated, it is pushed and everyone has access to give feedback.
- Before submitting the assignment, everyone will take a look at every file and give a “green light” to affirm they agree with the contents of the graded files.

Sprint 2 Plan

Tasks to Complete:

1. Complete the Code: Implement the `ReliabilityVisualization` class for the WARP project.
2. JavaDoc Documentation: Write detailed JavaDoc comments for all attributes, methods, and constructors created in `ReliabilityVisualization`.
3. Testing: Develop and execute JUnit tests for the `ReliabilityVisualization` class and perform regression testing to ensure existing tests still pass.
4. Update UML Class Diagram: Revise the UML diagram to reflect the completed `ReliabilityVisualization` class.
5. Update Sequence Diagram: Incorporate changes from the `ReliabilityVisualization` class into the sequence diagram.
6. Update Documentation: Modify the `README` file with the tasks completed during Sprint2.

Task Assignments and Workflow:

- Code Development: Victoria & Manthan & Katelyn will collaborate to complete the implementation of the `ReliabilityVisualization` class.
    - Code Development will be the first task completed.
- JavaDoc Comments: Molly will write and review JavaDoc comments for clarity and completeness.
    - JavaDoc comments will be written directly after code is developed.
- Testing: Teagan & Manthan & Molly: Develop JUnit test cases for `ReliabilityVisualization` and conduct regression testing to verify the integrity of existing tests.
    - JUnit tests were developed following the conclusion of code development and JavaDoc comments.

- UML and Sequence Diagrams: Entire Team will update the UML class diagram to include all relevant attributes and methods of `ReliabilityVisualization`. We will also revise the sequence diagram to reflect the updates.
- README Updates: Katelyn & Teagan will edit the `README` file to summarize Sprint 2 tasks and their outcomes.

Guidelines for JavaDoc Comments:

- Attributes: Describe the purpose and expected values of each attribute.
- Methods: Explain the functionality, input parameters, and expected output for each method.
- Constructors: Document the purpose and initialization process for the class.
- JavaDoc comments will be written concluding the implementation of methods in ReliabilityVisualization class.

Testing Approach:

1. JUnit Tests: Create robust test cases covering all methods of the `ReliabilityVisualization` class and test edge cases and expected behaviors.
2. Regression Testing: Run existing tests to ensure changes do not introduce bugs.

Quality Assurance (QA):

- Collaborative Development: Use peer programming and reviews for coding and documentation tasks.
- Regular Updates: Push updates to the repository to facilitate team feedback.
- Final Review: Conduct a team-wide review of all deliverables before submission.

Sprint 3 Plan

- **Tasks to be Done**:
    1. Finish the *ReliabilityAnalysis* class in WARP. This will include implementing the getReliabilities method and adding other necessary methods (for example, setReliabilityHeaderRow, getReliabilityHeaderRow, getFinalReliabilityRow, and buildReliabilityTable).
    2. Add new JavaDoc comments for all methods, attributes, and constructors that are added to *ReliabilityAnalysis*.
    3. Add new JUnit tests for all methods that are added to *ReliabilityAnalysis*. Additionally, perform regression testing for already existing tests.
    4. Update UML class diagram for *ReliabilityAnalysis*.
    5. Update sequence diagram with changes made to *ReliabilityAnalysis*.
    6. Update the README with completed tasks for Sprint 3.
- **Task Assignments and Workflow:**
    1. Katelyn and Molly will write part of the new methods in ReliabilityAnalysis. This will include setReliabilityHeaderRow, getFinalReliabilityRow, getReliabilities, getReliabilityHeaderRow, and setReliabilities.
    2. Manthan will write the buildReliabilityTable method in ReliabilityAnalysis.
    3. Teagan will write JavaDoc comments for the new methods added to ReliabilityAnalysis.
    4. Victoria, Molly, and Katelyn will write Junit tests for the methods that are added in ReliabilityAnalysis.
    5. Manthan will update the UML diagram with changes made to ReliabilityAnalysis.
    6. Teagan will update the sequence diagram with changes to ReliabilityAnalysis.
    7. Victoria will update the README for Sprint 3 with details about what tasks each person completed.

Guidelines for JavaDoc Comments:

- Attributes: Describe the purpose and expected values of each attribute.

- Methods: Explain the functionality, input parameters, and expected output for each method.
- Constructors: Document the purpose and initialization process for the class.
- JavaDoc comments will be written concluding the implementation of methods in ReliabilityVisualization and ReliabilityAnalysis classes.

Testing Approach:

- Junit tests will be written for all methods that are added to ReliabilityAnalysis and ReliabilityVisualization during Sprint 3.
- Tests will be written to be robust and cover all edge cases for each method.
- Methods will undergo regression testing after runs to ensure thorough and effective testing.

Quality Assurance (QA):

- Collaborative Development: Use peer programming and reviews for coding and documentation tasks.
- TA/Instructor Collaboration: Regularly check in with TAs and instructor to clear up points of confusion and get feedback on work quality and efficiency.
- Regular Updates: Push updates to the repository to facilitate team feedback.
- Final Review: Conduct a team-wide review of all deliverables before submission.