Milestone: Testing

OVERVIEW

During this milestone, you will:

- Create a test plan for a card game
- Create stubs and tests for Models using state testing.
- Create integration tests for Controller using mocking.
- Collaborate with other teams to develop the final test suite.

SPRINTS

The milestone will be completed in one 1-week spring and one 2-week sprint. Sprint activities are completed as a team.

Sprint 1

1. Tag the current version of the repostoy with the tag `initial_design`
    a. Either:
        i. Use the git command `git tag -a initial_design 9fcbt` where 9fcbtis the commit GUID.
        ii. Add the tag in GitLab:
            1. Go to `Repository --> Commits`
            2. Find the commit id on the right side.
            3. Write down the number (or do a copy)
            4. Go to `Repository --> Tags`
            5. Create a new tag (button in top right)
            6. Enter the tag name.
            7. Click on the drop down in `Create from`
            8. Enter (or paste) the commit id
            9. Click `Create tag`
2. Remove the previous design documents.
3. Prepare your game's repository by:
    a. Create the following directories
        i. `src`
        ii. `test`
        iii. `include`
        iv. `docs`
            1. `code`
            2. `design`
                a. Move class and sequence diagram files from the Master repository to this directory.
            3. `requirements`
                a. Move use case files to this directory.
    b. Copy the following files from the `Project Files` repository
        i. `Makefile`
        ii. `CPPLINT.cfg`

iii. `.gitlab-ci.yml`
iv. `docs/code/doxyfile`

c. Update the `Makefile`:
   i. Change the project name to match your game.
   ii. Add `-lgmock` to `LINKFLAGS`
d. Update the `.gitlab-ci.yml` file to match the updated `Makefile`.

4. Examine the class diagrams of the game.
   a. If you find a design issue, create an issue in the Master repository so the problem can be noted and discussed by members of the other teams. Label the issues with the "design" and "bug" so they can easily be found.
   b. The teams will decide if the design issue is major or minor. Possible solutions include:
      i. Adjusting the test plan accordingly without changing the overall design.
      ii. Adopting another team's design (or a variant of it) as the master design.
5. Create a test plan for the game.
   a. All public methods of `Controller(s)` and `Model(s)` are to be tested, either directly (`Deck::draw()` is called) or indirectly (e.g. `Deck::getCard()` is used by `Deck::draw()`)
   b. The test scaffolding will replace the `View` classes. No tests need to be written for these.
   c. Commit the test plan to `test/TestPlan.pdf`
6. Create header files for the classes.
   a. Comment the headers so that `Doxygen` can be used to extract the documentation for the classes, methods and attributes.
7. The Scrum Masters examine the teams' test plans and header files to create a `Test Plan.pdf` and set of game header files in the `Master` repository on a `testing` branch for the game.
   a. Other team members may attend the meeting but should be "Chickens," not "Pigs."
8. Create a pull request in the game's `Master` repository from the `testing` branch to the `main` branch.
   a. Place the test plan in a `test` directory.
   b. Place the header files in an `include` directory.

Sprint Grading

The sprint will be graded on the following criteria:

1. Individual Teams:
   a. Test plan [10 marks]
   b. Documented header files [10 marks]
2. Game Teams (all teams for a game share this grade) [10 marks – Completed/Not Completed]:
   a. Master test plan
   b. Master header files

Sprint 2

1. Write stubs for the game's methods (i.e. empty or return default values).
2. Implement the tests for the `Model` classes.
   a. Each `Model` class is to have a corresponding test suite in a separate file (e.g. `TestDeck.cpp`) in the test directory.

   b. The tests are to fail as these classes will not have an implementation (i.e. stubs). Use `DISABLED_` to disable these tests once you have a failing test.
3. Implement tests for the use cases using mocking.
   a. Create mocks of the dependencies (i.e. `Model` and `View` classes)
4. Implement the `Controller` class(es) to pass the test(s).
5. Select one team member to be a Scrum Master to meet with the other Scrum Masters for the same game.
6. The Scrum Masters examine the team's tests to create a master test suite for the game.
   a. Other team members may attend the meeting but should be "Chickens," not "Pigs."
   b. Problems or omissions regarding the game's design are likely discovered or discussed. Scrum Masters are to revise the design appropriately and delegate among the teams the following:
      i. Updating the class diagram.
      ii. Updating the sequence diagram(s).
      iii. Creating a working project for the master repository.
7. Create a pull request in the game's `Master` repository from the `testing` branch to the `main` branch.

Sprint Grading

1. Individual Teams:
   a. Stubs of the `Model` and `View` classes [10 marks]
   b. Tests of the Model classes [10 marks]
   c. Test of the Controller class [15 marks]

a. Game Teams (all teams for a game share this grade):
   d. Updated class diagram [5 marks]
   e. Updated sequence diagram(s) [5 marks]
   f. Master testing suite [5 marks]

MILESTONE DELIVERABLES

Sprint 1	Test plan Header files
Sprint 2	Unit tests for Models and Controller Updated design diagrams