Group Project Assignments

Overview

The project will consist of three milestones. The milestone dates are listed on the course schedule. The first milestone is focused on getting you to plan out your graphical user interface and think about your use cases while developing a minimal prototype with basic functionality. The second milestone focuses on class design and developing a beta version of your web application with all its core functionality. The third milestone will take place during final exam week and will focus on a polished final product. The group with the best overall product will win the top award.

Tools

For diagrams, you can use LibreOffice Draw, LucidChart.com, or your preferred drawing tool. LibreOffice Draw is installed on all the PCs in CL46. LibreOffice is a free office suite that may be downloaded from https://www.libreoffice.org/download/download-libreoffice/.

Submission

All documentation should be submitted as a PDF (multiple PDFs are OK). Your web application should be submitted as Git commit ID.

Milestone 1

Deliverable	Grade Percentage	Description
GUI Visual Rendering	15%	A visual rendering of the primary GUI screen of your web application using a drawing tool.
GUI Screen Flow Diagram	15%	A screen flow diagram for you web application prototype that represents the logical flow of your GUI.
UML Use Case as Text	20%	A text description of the main success scenario of your web application prototype and include an appropriate extension to the main success scenario.
Web Application Prototype (alpha)	30%	A web application alpha version that contains the main code screen containing the "Fibonacci challenge". The web application should allow the user to submit a solution for the Fibonacci challenge and indicate pass or fail. There should also be a way for the user to reset the code window back to the initial state for the challenge. You will submit a Git commit ID for your code.
Unit Tests for Code Engine	15%	The Code Engine is the heart of the application that uses JShell. The Code Engine should pass unit tests for a Fibonacci challenge and a Binary search challenge.
Peer Contribution Report	5%	Identify how each teammate contributed to each of the deliverables for this project milestone.

* Refer to the "Introduction to Unified Modeling Language" slides for more details on UML and other types of diagrams.

Milestone 2

Deliverable	Grade Percentage	Description
GUI Visual Rendering (Revised)	5%	An updated visual rendering of the primary GUI screen of your web application using a drawing tool.
GUI Screen Flow Diagram (revised)	5%	An updated screen flow diagram for you web application prototype that represents the logical flow of your GUI.
UML Use Case as Text (Revised)	10%	An updated text description of the main success scenario of your web application prototype and include an appropriate extension to the main success scenario.
UML Class Diagram for Web Application	20%	A UML Class Diagram for all of the classes in your Web Application (excluding unit test classes). The UML diagram should include attributes and methods of the classes as well as relationships between classes.
Web Application Prototype (beta)	40%	A web application beta version that supports multiple users logging in, user sessions, a single module containing the four challenges from the unit tests, and being able to choose a challenge. The beta should also update the status for successfully completing a challenge and return the user to the module. The main code screen should allow the challenge to be run and reset. Running the code should show a pass/fail for the challenge and provide simple feedback on the fail. The user database should be populated with two test users. The user "alpha" with password "1234" who completed the Fibonacci challenge already and the user "beta" with password "5678" who has not completed any challenges yet. You will submit a Git commit ID for your code.
Unit Tests for Code Engine (milestone 1 tests and additional tests)	15%	The Code Engine is the heart of the application that uses JShell. The Code Engine should pass unit tests for a Fibonacci challenge and a Binary search challenge. The Code Engine should pass unit tests for the Selection Sort challenge and the List of letters of the alphabet starting at a particular letter challenge.
Peer Contribution Report	5%	Identify how each teammate contributed to each of the deliverables for this project milestone.

Milestone 3 (Final Exam)

Deliverable	Grade Percentage	Description
GUI Visual Rendering (Revised)	5%	An updated visual rendering of the primary GUI screen of your web application using a drawing tool.
GUI Screen Flow Diagram (revised)	5%	An updated screen flow diagram for you web application prototype that represents the logical flow of your GUI.
UML Use Case as Text (Revised)	10%	An updated text description of the main success scenario of your web application prototype and include an appropriate extension to the main success scenario.
UML Class Diagram for Web Application	10%	A UML Class Diagram for all of the classes in your Web Application (excluding unit test classes). The UML diagram should include attributes and methods of the classes as well as relationships between classes.
Web Application (final)	45%	The final version of the application should be focused on bug fixes and product quality improvements for the beta.
Unit Tests for Code Engine (milestone 1 and 2 tests)	10%	The Code Engine is the heart of the application that uses JShell. The Code Engine should pass unit tests for a Fibonacci challenge and a Binary search challenge. The Code Engine should pass unit tests for the Selection Sort challenge and the List of letters of the alphabet starting at a particular letter challenge. You will submit a Git commit ID for your code.
Peer Contribution Report	5%	Identify how each teammate contributed to each of the deliverables for this project milestone.
Instructor Top Product Award	10%	The instructor will consider both the presentation during the Final Exam period and the overall quality of the product to decide the “Top Product Award”. Only one group will receive this award.

Group Project Assignments

Groups

- Group 1 - TurleyTime
  - jcturley1@malone.edu
  - lgay1@malone.edu
  - ejmiller3@malone.edu
- Group 2 - ThreeFifths
  - mmthomas3@malone.edu
  - amdavis5@malone.edu
  - rgtimberlake1@malone.edu
- Group 3 - RedBlackPlaid
  - jcureton1@malone.edu
  - jlhicks2@malone.edu
  - jmrobertson1@malone.edu
- Group 4 - LaissezFaire
  - tjhouston1@malone.edu
  - jcmarapao1@malone.edu
  - gswoodland1@malone.edu

Git Instructions

The instructions that follow explain how to set up Git to access your group repository. If you use the same lab PC, you may have already completed some of these steps, so keep that in mind.

How to create a directory for your Git repository on a lab PC

1. In Windows 10, search for "Git Bash" application and run the application. The Git Bash window should appear. You should see a command prompt labeled "username@computername MINGW64~".
2. In the Git Bash window, type **pwd** and press enter. You should see the output "c/Users/yourusername". The pwd command shows the directory you are currently in.
3. Type **cd desktop** and press enter. The end of the prompt should change to "~/desktop". This changes the current directory to the sub-directory "desktop". You can use the pwd command again to observe the difference.
4. Type **mkdir repository** and press enter. This command will create a repository sub-directory. If you view the desktop, you will now see a repository folder.
5. Type **cd repository** and press enter. This command changes the directory to the "repository" sub-directory.

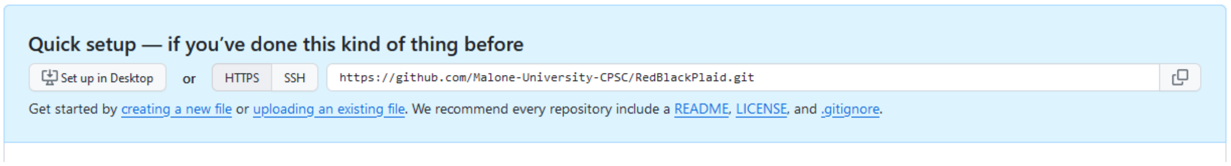
Group Project Assignments

Setting Global Parameters in Git Bash

1. In the Git Bash window, type **git--version** and press enter to see the version of Git.
2. Type **git config --global user.name yourgithubusername** and press enter. Please note that your GitHub username will be difference from your Malone username. This modifies the global configuration of Git Bash.
3. Type **git config --global user.email "yourmaloneemailaddress"** and press enter.

Cloning Your Group Repository

1. Open a web browser and go to **https://github.com/Malone-University-CPSC**. You will need to login to GitHub. You should see a list of your repositories. Click on the name of your group repository (same as group name).
2. Locate the quick setup information for your group repository. The screen shot shown below shows the quick setup information for group RedBlackPlaid.



Quick setup — if you've done this kind of thing before

https://github.com/Malone-University-CPSC/RedBlackPlaid.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

3. Copy the quick setup URL. For example, the quick setup URL in the screenshot would be https://github.com/Malone-University-CPSC/RedBlackPlaid.git .
4. In Git Bash, type the command
   a. **git clone yourquicksetupurlhere** and press enter.
5. A web browser window will open and you will need to login to GitHub and authorize the access. After authenticating and authorizing access, you should see output similar to below.

```
Cloning into 'YourGroupRepo'...

info: please complete authentication in your browser...

remote: Enumerating objects: 3, done.

remote: Counting objects: 100% (3/3), done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.
```

6. Git created a YourGroupRepo containing a local copy of the YourGroupRepo.
7. Type **git status** and press enter. You should see output similar to below. This means that the local main branch and the remote main branch are up to date (synched).

```
On branch main

Your branch is up to date with 'origin/main'.
```

Group Project Assignments

Git Tips

- Each student in your group may be working on your repository, so you need to practice Git hygiene.
- The **git status** command will tell you if you need to take a particular action. Always start with this command.
- If you add a new file OR update an existing file, you must stage the changes to your local Git repository using **git add filename**.
- If you delete an existing file, you must also stage the removal so Git stop tracking the file. Use the **git rm filename** to remove a file from the stage.
- After staging all your new or updated files, you need to commit your changes to your local Git repository using **git commit –m "Short message summarizing your changes"**.
- Once the changes are staged and committed to your local repository, you need to check if another team member has made changes at the remote repository using **git pull**.
- You may need to resolve conflicts. I encourage you to refer to "Introduction to Git Part 2" in Moodle or ask your instructor.
- Once all conflicts are resolved, use **git push** to push your changes to the remote repository.
- You can see the changes and current file in your repository by visiting **https://github.com/Malone-University-CPSC** in a web browser and viewing your repository.
- When you intend to make changes to the code, the best practice is to create a branch, make your changes in the branch, and then merge your branch back into the main for master branch. I encourage you to refer to "Introduction to Git Part 2" in Moodle or ask your instructor for more details.

Viewing Commit History in Git Bash

1. Type **git log** and press enter to view the commit history. Use the up/down arrow keys to view the list of commits and press **q** when you are ready to exit git log. You can see the Author and Date of each commit as well as the commit message. There is also the alphanumeric commit ID for each commit (e.g. commit bb8103dcac4f62bb45ef50f0e53c8113153e4cd4).
2. The commit ID represents a save point. You can view all source files for a specific commit ID in GitHub by opening a web browser and going to **https://github.com/Malone-University-CPSC/YourGroupRepo/tree/commitID**
3. For example, https://github.com/Malone-University-CPSC/YourGroupRepo/tree/bb8103dcac4f62bb45ef50f0e53c8113153e4cd4