Project II: Team-Based Software Development

GitHub – Development Requirements

Overview

Congratulations! You have successfully completed the Proposal, Project Management and Requirements and Design phases of your new software product. Now it is time for the collaborative development phase.

Each of you has been assigned at least one unique module within your software project. These modules have been designed to be implemented in a standalone manner, eventually to be integrated together at the end into one application for release. During the development and integration phases your team is required to use GitHub as a collaborate space. This document defines the minimum functionality that is required to be used as part of your Project II activities and is setup as a "checklist" for you to manage and make sure all has been completed.

What to do

☐ Step#1: Setup GitHub Accounts

Each member of your group will require a GitHub account. If you are new to GitHub, please visit (https://github.com/) and create a new account. GitHub has plenty of resources to learn how to properly use the software, for example:

- **https://docs.github.com/en**
- **https://www.youtube.com/results?search_query=github**

I am not expecting you to be experts of GitHub by the end of this course. The idea here is to give you some experience with a different tool (something other than Azure, which you learned last semester) and a collaborative development space to work with. Remember, this course is about the collaboration, not the tool. That said, these are the features each group are required to use:

☐ Source Code Repository Management (check-in/check-out, version control)

☐ Branches (individual development, with peer review "Pull Request" functionality)

☐ Bug/Issue Tracking (Open, Close, and comment for traceability)

☐ Formal Release

☐ Step #2: Group Repository

Now that each of you have an account on GitHub, one of you must create a repository for the group project. My suggestion would be the assigned group leader.

☐ Your repository name must consist of your ***Section Number*** and ***Group Number***

☐ Using Visual Studio, create a basic "Hello World" application.

☐ Using either the built in GitHub capability within Visual Studio, or manually through GitHub.com or the GitHub desktop application, create a repository for this newly created "Hello World" program.

☐ Sync the newly created repository with your GitHub.com account, so that everybody can access it.

NOTE: The repository can be Public/Private, it is up to you.   Be aware, private repositories sometimes need paying subscriptions. If you find yourself in that situation, just make a free public repository.

☐ Step #3: Share the Repository

☐ The owner of the repository needs to invite all participants to collaborate with this repository.   This can be done by going to the "Settings → Manage Access" area of the repository and selecting the "Invite Teams or People" button.   Type in the username of your group members and send invitations.

☐ Make sure you include your professor. I can be reached on GitHub using the following username: ecoleshill

☐ Step #4: Create Branches

As mentioned above, each of you has been assigned at least one module to work on. The next step is to create branches for each of your modules.   You will then sync your local development environments with your branch(es) and perform your check-in/check-out source control management with your individual branch(es)

NOTE: If you are assigned to work on multiple modules, you can use individual branches or a single branch for all your development. It is up to you and your group to decide the best way to manage the development based on your Project Management Plan.

☐ Step #5: Bug/Issue Tracking (Development and Unit Testing Phase)

As you are developing your individual modules you will continue to collaboratively work through your MS Teams groups, having weekly meetings and discussing progress.   During this development time make sure you are using the Bug/Issue tracker in the project.   This can be done by logging into your GitHub.com account, selecting the projects repository and clicking on "Issues".   You should raise issues based on:

- New enhancements decided upon through group discussions
- Failures/bugs found during your Unit Level Testing

Make sure you are assigning tasks to the appropriate group member(s) and filling out the comments in the bugs/issues raised and not just opening/closing issues. You should have comments that state (as a minimum):

☐ Details about steps taken to reproduce and find the issue raised

☐ Details about the root cause of the problem

☐ Where in the source code the problem was found (filename(s), line numbers, etc…). Details that would help somebody else locate the area of the code in question

☐ Details about the solution implemented to solve the problem

☐ The version of the document/release this bug fix has been included in. This can be part of the comment that closes the bug/issue

☐ Step #6: Integration

If you have reached this stage you have two or more modules complete, with Unit Level testing done, ready to integrate. To integrate your modules, use the peer review functionality in GitHub by initiating “Pull Request”. This can be done using the following steps:

☐ Make sure your latest source code is checked in, and sync’ed with GitHub.com (i.e. your branch of code)

☐ Log into your GitHub.com account, select the groups project repository and go to your branch by selecting “Branches” from the “<> code” section of the repository.

☐ Click on the “New pull request” option for your branch.

☐ Follow the steps GitHub provides to review and integrate the code into the “Master” branch.

NOTE: At this point you will also need some “main()” code that will glue the modules together. You want to be able to compile and test. 😊

☐ Step #7: Bug/Issue Tracking (Integration & Testing Phase)

As you are performing your pull requests and integrating all the groups modules together you will ultimately find issues, things missing, etc... During this time make sure you are using the Bug/Issue tracker in the project. You should raise issues based on:

- New enhancements decided upon through group discussions
- Failures/bugs found during your Integration Testing

Make sure you are assigning tasks to the appropriate group member(s) and filling out the comments in the bugs/issues raised and not just opening/closing issues. You should have comments that state (as a minimum):

☐ Details about steps taken reproduce and/or find the issue raised

☐ Details about the root cause of the problem

☐ Where in the source code the problem was found (filename(s), line numbers, etc…). Details that would help somebody else locate the area of the code in question

☐ Details about the solution implemented to solve the problem

☐ The version of the document/release this bug fix has been included in. This can be part of the comment that closes the bug/issue

☐ Step #8: Formal Release

By this point all your modules are integrated together into a single application in your "Master" branch on GitHub. When you are confident that all your testing has been completed and successful, and your software is complete and ready for demonstration, you need to formally release a version of your software. This can be done using the following steps:

☐ Log into your GitHub.com account

☐ Go to your groups project repository and select "<> Code" in the menu

☐ On the right side of the screen you will see a grouping called "Releases", which contains a hyperlink "Create a new release". Click this link.

☐ Follow the GitHub.com instructions and formally release your source code for demonstration

What to Hand In

There is nothing to be handed in for this activity. How well you use GitHub for collaboration will be evaluated as part of the overall collaborative project mark. See the Rubrics in eConestoga for more details.