SE Lab 5 Contents (Source Code Management – GitHub)

Source Code Management, GitHub, and Visual Studio Integration

Source Code Management

Source Code Management (SCM) is a critical aspect of software development that involves tracking and controlling changes to the source code. SCM systems provide a structured way to manage code versions, collaborate with team members, and maintain a history of changes. Key benefits of SCM include:

- **Version Control:** Keeps track of every modification to the code, allowing developers to revert to previous versions if needed.
- **Collaboration:** Enables multiple developers to work on the same project simultaneously without overwriting each other's changes.
- **History and Audit:** Maintains a detailed history of changes, including who made the changes and why, which is essential for auditing and troubleshooting.
- **Branching and Merging:** Allows developers to create branches for new features or bug fixes and merge them back into the main codebase once they are complete.

GitHub

GitHub is a web-based platform that uses Git, a distributed version control system, to provide SCM services. It is widely used by developers and organizations to host, manage, and collaborate on software projects. Key features of GitHub include:

- **Repositories:** Centralized locations where project files and their revision history are stored.
- **Pull Requests:** Mechanisms for proposing changes to the codebase, which can be reviewed, discussed, and merged by team members.
- **Issues and Project Management:** Tools for tracking bugs, feature requests, and project tasks, helping teams stay organized and on track.
- **Continuous Integration/Continuous Deployment (CI/CD):** Integration with CI/CD tools to automate testing and deployment processes.
- **Community and Collaboration:** A vast community of developers who contribute to open-source projects, share knowledge, and collaborate on code.

Visual Studio and GitHub Integration

Visual Studio, a powerful integrated development environment (IDE) from Microsoft, offers seamless integration with GitHub, enhancing the development workflow. Key capabilities of Visual Studio with GitHub integration include:

- **Built-in Git Support:** Visual Studio provides built-in support for Git, allowing developers to perform common Git operations such as cloning repositories, committing changes, and pushing to remote repositories directly from the IDE.
- **GitHub Extension:** The GitHub extension for Visual Studio adds additional features, such as creating and managing GitHub repositories, viewing pull requests, and collaborating with team members.
- **Code Review and Collaboration:** Developers can review code changes, leave comments, and collaborate on pull requests without leaving Visual Studio.
- **Branch Management:** Visual Studio makes it easy to create, switch, and merge branches, helping developers manage their workflow efficiently.
- **Integrated Terminal:** The integrated terminal in Visual Studio allows developers to run Git commands and scripts directly within the IDE.
- **Continuous Integration:** Integration with GitHub Actions and other CI/CD tools enables automated testing and deployment workflows.

Conclusion

Source Code Management, GitHub, and Visual Studio integration provide a robust framework for managing software development projects. By leveraging these tools, developers can collaborate more effectively, maintain a clean and organized codebase, and streamline their development processes. Whether you are working on a small personal project or a large enterprise application, understanding and utilizing these tools will significantly enhance your productivity and code quality

Exercise 1: Setting Up GitHub Repository for Windows Forms Application

Objective: Create a GitHub repository and manage the source code for the Windows Forms application.

Steps:

1. **Create a GitHub Account:**
    - If you don't already have a GitHub account, sign up at GitHub.
2. **Create a New Repository:**
    - Go to your GitHub profile and click on the Repositories tab.
    - Click the New button to create a new repository.
    - Name the repository WinFormsStudentApp.
    - Add a description (optional) and choose whether the repository should be public or private.
    - Check the box to initialize the repository with a README file. **(*) (see below)**
    - Click Create repository.
3. **Clone the Repository Locally:**
    - Open Visual Studio.
    - Go to File > Clone Repository.
    - Enter the URL of your GitHub repository and choose a local path to clone the repository.
    - Click Clone.
4. **Add Your Windows Forms Project:**
    - Open your Windows Forms project in Visual Studio.
    - Right-click on the solution in the Solution Explorer and select Add > Existing Project.
    - Navigate to the cloned repository folder and add your project.
5. **Commit and Push Changes:**
    - Go to the Git Changes window in Visual Studio.
    - Stage all changes by clicking the + button.
    - Enter a commit message, e.g., "Initial commit of Windows Forms Student App".
    - Click Commit All and then Push to push the changes to GitHub.

Explain (*)

When you create a new repository on GitHub, there's an option to "Check the box to initialize the repository with a README file." This means that GitHub will automatically create a README file in your new repository. Here's why this is useful:

Purpose of Initializing with a README File

1. Provides Basic Information:
    - A README file typically contains basic information about the project, such as its purpose, how to set it up, and how to use it. This is helpful for anyone who views your repository.
2. Creates the Initial Commit:
    - Initializing the repository with a README file creates the first commit in your repository. This is useful because it sets up the repository with a starting point.
3. Improves Repository Presentation:
    - The README file is displayed on the main page of your repository on GitHub, providing an immediate overview of the project.

How to Initialize with a README File

1. Create a New Repository:
    - Go to your GitHub profile and click on the Repositories tab.
    - Click the New button to create a new repository.
2. Fill in Repository Details:
    - Name your repository and add a description (optional).
    - Choose whether the repository should be public or private.
3. Check the Box:
    - Check the box labeled "Initialize this repository with a README".
    - This option is usually found under the repository details section.
4. Create Repository:
    - Click the Create repository button.

By checking this box, GitHub will automatically add a README file to your new repository, making it easier to get started and providing a clear starting point for your project

Exercise 2: Creating a Windows Forms Project Repository in Visual Studio and Mapping to GitHub

Objective: Create a Windows Forms project in Visual Studio, initialize a local Git repository, and then map it to a GitHub repository.

Steps:

1. **Create a New Windows Forms Project:**
    - Open Visual Studio.
    - Go to File > New > Project.
    - Select Windows Forms App (.NET Framework) from the list of templates.
    - Name your project WinFormsStudentApp and choose a location to save it.
    - Click Create.
2. **Initialize a Local Git Repository:**
    - In Visual Studio, go to View > Git Changes to open the Git Changes window.
    - Click the Initialize Repository button to create a local Git repository for your project.
3. **Create a GitHub Repository:**
    - Go to your GitHub profile and click on the Repositories tab.
    - Click the New button to create a new repository.
    - Name the repository WinFormsStudentApp.
    - Add a description (optional) and choose whether the repository should be public or private.
    - Do not initialize the repository with a README file, .gitignore, or license.
    - Click Create repository.
4. **Map the Local Repository to GitHub:**
    - In Visual Studio, go to the Git Changes window.
    - Click the Settings icon (gear icon) and select Repository Settings.
    - Under Remotes, click Add.
    - Enter origin as the remote name and paste the URL of your GitHub repository.
    - Click Save.
5. **Commit and Push Changes:**
    - Go to the Git Changes window in Visual Studio.
    - Stage all changes by clicking the + button.
    - Enter a commit message, e.g., "Initial commit of Windows Forms Student App".
    - Click Commit All and then Push to push the changes to GitHub.

Exercise 3: Two students to collaborate on a Windows Forms project using GitHub

These exercises will guide you through inviting a collaborator, cloning the project, making changes, committing those changes, and retrieving the latest updates.

Exercise 3a: Setting Up the GitHub Repository and Inviting a Collaborator

Objective: Create a GitHub repository for the Windows Forms project and invite another student as a collaborator.

Steps:

1. **Create a GitHub Repository:**
    - One student (Student A) should go to their GitHub profile and click on the Repositories tab.
    - Click the New button to create a new repository.
    - Name the repository WinFormsStudentApp (or other name)
    - Add a description (optional) and choose whether the repository should be public or private.
    - Check the box to initialize the repository with a README file.
    - Click Create repository.
2. **Invite a Collaborator:**
    - In the newly created repository, go to the Settings tab.
    - Click on Collaborators in the left sidebar.
    - Enter the GitHub username or email of the other student (Student B) and click Add collaborator.
    - Student B will receive an invitation to join the repository. They should accept the invitation.

Exercise 3b: Cloning the Repository and Setting Up the Project

Objective: Clone the repository locally and set up the Windows Forms project.

Steps:

1. **Clone the Repository (Student A and Student B):**
    - Both students should open Visual Studio.
    - Go to File > Clone Repository.

- Enter the URL of the GitHub repository and choose a local path to clone the repository.
- Click Clone.

2. **Create the Windows Forms Project (Student A):**
    - Student A should create a new Windows Forms project in Visual Studio.
    - Go to File > New > Project.
    - Select Windows Forms App (.NET Framework) from the list of templates.
    - Name the project WinFormsStudentApp and choose the cloned repository folder as the location.
    - Click Create.
3. **Commit and Push the Initial Project (Student A):**
    - Go to the Git Changes window in Visual Studio.
    - Stage all changes by clicking the + button.
    - Enter a commit message, e.g., "Initial commit of Windows Forms Student App".
    - Click Commit All and then Push to push the changes to GitHub.

Exercise 3c: Making Changes and Committing (Student B)

Objective: Clone the project, make changes to the form, and commit those changes.

Steps:

1. **Clone the Repository (Student B):**
    - Student B should clone the repository if they haven't already done so (see Exercise 2).
2. **Open the Project and Make Changes (Student B):**
    - Student B should open the cloned project in Visual Studio.
    - Open the main form (e.g., Form1.cs).
    - Drag a **Label** control and a **TextBox** control from the Toolbox to the form.
    - Set the text of the label to "Student Name:" and the name of the TextBox to txtStudentName.
3. **Commit and Push Changes (Student B):**
    - Go to the Git Changes window in Visual Studio.
    - Stage all changes by clicking the + button.
    - Enter a commit message, e.g., "Added Label and TextBox for Student Name".

- Click Commit All and then Push to push the changes to GitHub.

Exercise 3d: Retrieving the Latest Changes and Keeping Changes (Student A)

Objective: Retrieve the latest changes from GitHub and keep local changes.

Steps:

1. **Pull the Latest Changes (Student A):**
   - Student A should open the project in Visual Studio.
   - Go to the Git Changes window.
   - Click Pull to retrieve the latest changes from GitHub.
2. **Make Additional Changes (Student A):**
   - Student A should make additional changes to the form, such as adding another **Label** and **TextBox** for "Course Name".
   - Set the text of the new label to "Course Name:" and the name of the new TextBox to txtCourseName.
3. **Commit and Push Changes (Student A):**
   - Go to the Git Changes window in Visual Studio.
   - Stage all changes by clicking the + button.
   - Enter a commit message, e.g., "Added Label and TextBox for Course Name".
   - Click Commit All and then Push to push the changes to GitHub.

Exercise 3e: Resolving Conflicts (If Any)

Objective: Resolve any merge conflicts that may arise during collaboration.

Steps:

1. **Pull the Latest Changes (Student B):**
   - Student B should open the project in Visual Studio.
   - Go to the Git Changes window.
   - Click Pull to retrieve the latest changes from GitHub.
2. **Resolve Conflicts (If Any):**
   - If there are any merge conflicts, Visual Studio will prompt Student B to resolve them.
   - Open the conflicting files and manually merge the changes.

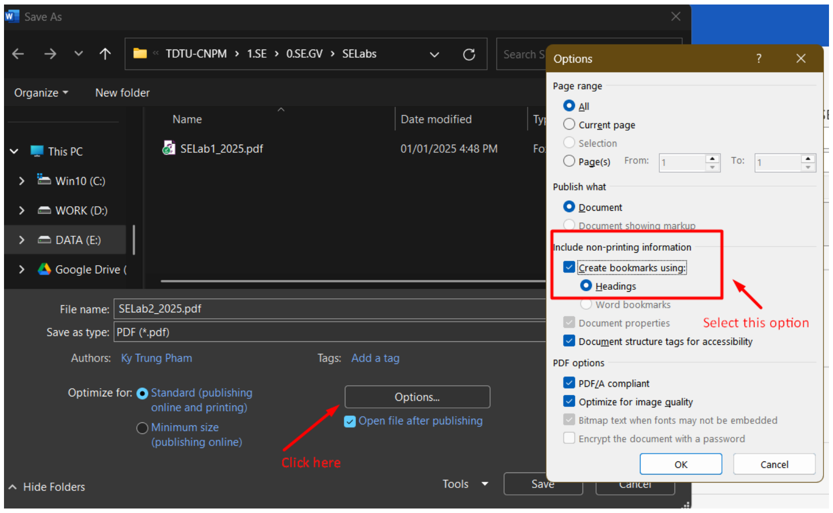
- After resolving conflicts, stage the resolved files, commit the changes, and push to GitHub.

Lab 5 Report Submission

- **Introduction:** Briefly describe the purpose of the exercises.
- **Exercises:** List each exercise with a brief description of what you did and learned for Class Activity included the screenshots of running App output.
- **Exercise 3:** Describe what you do with including the screenshots of the output for homework exercise. (Even exercise with Github over 2 students but each one should make your own report)
- **Conclusion:** Summarize your overall learning experience and any challenges you faced.

Submit 2 files on eLearning:

- StudentID_StudentName.zip (contain Windows Web Project & .sql file) for both Class Activity and Homework.
- StudentID_StudentName.pdf (require to create bookmarks)



SELab1_2025.pdf 01/01/2025 4:48 PM

File name: SELab2_2025.pdf
Save as type: PDF (*.pdf)
Authors: Ky Trung Pham Tags: Add a tag
Optimize for: [✓ Standard (publishing online and printing)] Minimum size (publishing online)
[✓ Open file after publishing]

Page range: [✓ All] Current page Selection Page(s) From: 1 To: 1
Publish what: [✓ Document] Document showing markup
Include non-printing information
[✓ Create bookmarks using:] [✓ Headings] Word bookmarks
Document properties
[✓ Document structure tags for accessibility]
PDF options
[✓ PDF/A compliant]
[✓ Optimize for image quality]
Bitmap text when fonts may not be embedded
Encrypt the document with a password

Notes: There may be a typo/mistake during creating this document. Please correct it by yourself.

Enjoy ☺ Email: tg_phamthaikytrung@tdtu.edu.vn