SOFTWARE ENGINEERING ASSIGNMENT5

1. INSTALLATION OF VS CODE

Here are the steps to download and install Visual Studio Code on a Windows 11 operating system:

PREREQUISITES

1. Operating System: Ensure that you are running Windows 11.

2. User Permissions: You need to have administrative rights on your computer to install new software.

Steps to Download and Install Visual Studio Code

1. Download Visual Studio Code:

- Open a web browser and go to the [Visual Studio Code download page](https://code.visualstudio.com/Download).
- Click on the "Windows" button to download the installer.

2. Run the Installer:

- Once the download is complete, locate the downloaded file (`VSCodeUserSetup{1.90.2 }.exe`) in your Downloads folder or the specified location.
- Doubleclick the installer file to run it.

3. Installation Process:

- When the Visual Studio Code Setup window appears, click "Next".
- Read and accept the license agreement by clicking "I accept the agreement", then click "Next".
- Choose the destination location where you want to install Visual Studio Code. The default location is usually fine. Click "Next".
- Select additional tasks, such as creating a desktop icon, adding "Open with Code" actions to the context menu, and registering Code as an editor for supported file types. These options are useful for quick access and integration. Click "Next".
- Review your installation settings and click "Install" to start the installation.

4. Complete the Installation:

- The installation process will take a few moments. Once completed, you can choose to launch Visual Studio Code immediately by selecting the "Launch Visual Studio Code" checkbox.
- Click "Finish" to exit the installer.

5. First Launch:

- If you didn't choose to launch Visual Studio Code immediately, you can start it from the Start Menu or the desktop shortcut if you created one.
- On the first launch, you may be prompted to select a theme and install recommended extensions.

2. FIRSTTIME SETUP

- **Initial Configurations and Settings**
  a) Theme and Appearance:

  Choose a theme that suits your preference. Go to File > Preferences > Color Theme (or press Ctrl+K Ctrl+T). Popular themes include "Dark+ (default dark)", "Light+ (default light)", and "Monokai".

  b) Languages Support:
- **Python**: Install the Python extension by Microsoft for rich Python support.

- **HTML/CSS**: Builtin support, but you can enhance it with extensions like Live Server for realtime preview.

  c) Version Control:

- **GitLens:** Enhance Git capabilities within VS Code.
- **GitHub Pull Requests and Issues**: Manage pull requests and issues directly from VS Code.

3. USER INTERFACE OVERVIEW:

Visual Studio Code (VS Code) has a wellorganized and intuitive user interface designed to help you navigate and manage your development tasks efficiently. Here are the main components of the VS Code user interface:

1. Activity Bar

- Location: Left side of the window

- Purpose: The Activity Bar provides quick access to different views and features of VS Code. It contains icons for the most common views and activities, including:

- Explorer: Displays your project files and folders.
- Search: Allows you to search across your project.
- Source Control: Integrates with version control systems like Git.
- Run and Debug: Manage and run your debug configurations.
- Extensions: Access and manage extensions to enhance functionality.

- You can customize the Activity Bar by rearranging or hiding icons according to your preferences.

2. Side Bar

- Location: Right next to the Activity Bar

- Purpose: The Side Bar shows different panels based on the icon selected in the Activity Bar. Each panel provides detailed tools and information related to its function. For example:

- Explorer Panel: Displays a tree view of your project files and directories.
- Source Control Panel: Shows changes, branches, and repositories for version control.
- Search Panel: Allows for projectwide search and replace operations.
- Extensions Panel: Lists installed extensions and allows you to browse and install new ones.

3. Editor Group

- Location: Central area of the window

- Purpose: The Editor Group is where you write and edit your code. You can open multiple files simultaneously, and they will be displayed in tabs within the Editor Group. Key features include:

- Tabs: Each open file is represented by a tab.
- Split Editors: You can split the editor into multiple groups to view and edit files side by side. Rightclick on a tab and select "Split" to open the file in a new editor group.
- Syntax Highlighting: Automatically highlights syntax for different programming languages.
- IntelliSense: Provides code completion and suggestions.

4. Status Bar

- Location: Bottom of the window

- Purpose: The Status Bar provides information about the current state of the editor and context about the file you are working on. It displays useful information such as:

- Current Branch: Shows the active Git branch.
- Line and Column Number: Indicates the cursor position in the file.
- Encoding and Language Mode: Displays the file's character encoding and language mode.
- Errors and Warnings: Shows a count of errors and warnings in the file.
- Live Server Status: If using the Live Server extension, it shows the server status.

You can interact with many items in the Status Bar by clicking on them to perform actions or change settings.

4. COMMAND PALETTE:

The Command Palette in Visual Studio Code (VS Code) is a powerful feature that allows you to access and execute various commands quickly. It provides a quick way to search for and run commands without having to navigate through menus or remember complex keyboard shortcuts.

Accessing the Command Palette

You can open the Command Palette in two ways:

1. Keyboard Shortcut: Press `` `*Ctrl+Shift+P*` `` (Windows/Linux) or `` `*Cmd+Shift+P*` `` (Mac).
2. Menu: Go to `View > Command Palette...`.

Using the Command Palette

Once the Command Palette is open, you can start typing the name of the command you want to execute. The palette will filter and display matching commands in realtime. You can then select the desired command by clicking on it or navigating with the arrow keys and pressing `Enter`.

Examples of Common Tasks Performed Using the Command Palette

1. Change Theme:
   - Open the Command Palette and type `theme`.
   - Select `Preferences: Color Theme`.
   - Choose from the list of available themes.

2. Install Extensions:
   - Open the Command Palette and type `install`.
   - Select `Extensions: Install Extensions`.
   - Search for the desired extension and install it.

3. Open Settings:
   - Open the Command Palette and type `settings`.
   - Select `Preferences: Open Settings (UI)` to open the settings in the user interface, or `Preferences: Open Settings (JSON)` to open the settings in JSON format.

4. View Git History:
   - Open the Command Palette and type `git log`.
   - Select `Git: View History`.

5. Toggle Integrated Terminal:
   - Open the Command Palette and type `terminal`.
   - Select `View: Toggle Terminal` to show or hide the integrated terminal.

6. Run Code:
   - Open the Command Palette and type `run`.

- Select `Run Code` (requires the Code Runner extension) to execute the current file.

7. Format Document:
   - Open the Command Palette and type `format`.
   - Select `Format Document` to automatically format the current file according to the configured formatter.

8. Open File by Name:
   - Open the Command Palette and type `open`.
   - Select `File: Open File` and type the name of the file you want to open.

9. Reload Window:
   - Open the Command Palette and type `reload`.
   - Select `Developer: Reload Window` to restart VS Code.

10. Change Language Mode:
    - Open the Command Palette and type `language`.
    - Select `Change Language Mode` and choose the desired programming language.

The Command Palette is an essential tool for increasing productivity and efficiency in VS Code. It provides quick access to a vast array of commands, making it easy to perform tasks without leaving the editor.

5. EXTENSIONS IN VS CODE:

Extensions in Visual Studio Code (VS Code) play a crucial role in enhancing the functionality and productivity of the editor. They allow users to customize and extend the capabilities of VS Code to fit their specific development needs. Extensions can provide language support, debugging tools, linters, formatters, themes, and more.

Finding, Installing, and Managing Extensions

Finding Extensions

1. Marketplace:
   - Open the Extensions view by clicking the Extensions icon in the Activity Bar on the side of the window or by pressing `Ctrl+Shift+X` (Windows/Linux) or `Cmd+Shift+X` (Mac).
   - In the Extensions view, you can search for extensions using the search bar.

2. Command Palette:
   - Open the Command Palette by pressing `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac).
   - Type `Extensions: Install Extensions` and select it to open the Extensions view.

Installing Extensions

1. From the Extensions View:
   Search for the extension you want to install.
   Click the `Install` button next to the extension.

2. Using Command Palette:
   Open the Command Palette and type `ext install` followed by the extension name. For example, to install the Prettier extension, you can type `ext install esbenp.prettiervscode`.

Managing Extensions

1. Disable or Enable Extensions:
   In the Extensions view, find the installed extension you want to disable or enable.
   Click the `Disable` or `Enable` button.

2. Uninstall Extensions:
   In the Extensions view, find the installed extension you want to uninstall.
   Click the `Uninstall` button.

3. Update Extensions:
   If updates are available for installed extensions, you will see an update button. Click `Update` to install the latest version.

Essential Extensions for Web Development

1. Prettier Code Formatter:
   - Description: An opinionated code formatter that supports many languages.
   - Purpose: Automatically formats your code to ensure a consistent style.
   - Installation: `ext install esbenp.prettiervscode`.

2. ESLint:
   - Description: Integrates ESLint into VS Code.

- Purpose: Provides JavaScript and TypeScript linting to help catch errors and enforce coding standards.
- Installation: `ext install dbaeumer.vscodeeslint`.

3. Live Server:
   - Description: Launches a local development server with live reload.
   - Purpose: Automatically reloads the browser when files are saved, making it easy to see changes in realtime.
   - Installation: `ext install ritwickdey.LiveServer`.

4. Debugger for Chrome:
   - Description: Debug your JavaScript code in the Google Chrome browser.
   - Purpose: Allows you to set breakpoints, step through code, and inspect variables directly within VS Code.
   - Installation: `ext install msjsdiag.debuggerforchrome`.

5. HTML CSS Support:
   - Description: Provides CSS class name completion for the HTML class attribute.
   - Purpose: Enhances HTML development by providing CSS class name suggestions.
   - Installation: `ext install ecmel.vscodehtmlcss`.

6. Path Intellisense:
   - Description: Autocompletes filenames in the current workspace.
   - Purpose: Helps you quickly navigate and reference files within your project.
   - Installation: `ext install christiankohler.pathintellisense`.

7. IntelliSense for CSS class names in HTML:
   - Description: Provides CSS class name completion for the HTML class attribute based on the defined CSS.
   - Purpose: Makes it easier to reference CSS classes while writing HTML.
   - Installation: `ext install Zignd.htmlcssclasscompletion`.

8. npm:
   - Description: npm support for VS Code.
   - Purpose: Adds npm scripts support, running npm commands, and detecting installed npm packages.
   - Installation: `ext install eg2.vscodenpmscript`.

Extensions significantly enhance the development experience in VS Code by providing tools and features tailored to specific tasks and technologies. By carefully selecting and managing extensions, developers can create a highly customized and efficient coding environment.

6. INTEGRATED TERMINAL:

The integrated terminal in Visual Studio Code (VS Code) provides a convenient way to run commandline tasks within the editor, which helps streamline your development workflow. Here's how to open and use the integrated terminal and the advantages it offers over an external terminal.

Opening the Integrated Terminal

1. Keyboard Shortcut:
   Press `Ctrl+` (Windows/Linux) or `Cmd+` (Mac) to open the integrated terminal.

2. Command Palette:
   Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac) to open the Command Palette.
   Type `Toggle Terminal` and select `View: Toggle Terminal`.

3. Menu:
   Go to `View > Terminal` from the top menu.

Using the Integrated Terminal

1. Basic Operations:
   - Create New Terminal: Click the `+` icon in the terminal tab bar or use the shortcut `Ctrl+Shift+` (Windows/Linux) or `Cmd+Shift+` (Mac).
   - Switch Between Terminals: Click on the terminal tab you want to switch to or use `Ctrl+PageUp/PageDown` (Windows/Linux) or `Cmd+Option+` (Mac).
   - Kill Terminal: Click the trash can icon or use the shortcut `Ctrl+Shift+W` (Windows/Linux) or `Cmd+Shift+W` (Mac).

2. Customization:
   - Change Shell: To change the default shell, go to `File > Preferences > Settings` (or `Ctrl+,`), search for `terminal.integrated.shell`, and set it to your preferred shell (e.g., bash, PowerShell, zsh).
   - Split Terminal: Click the split terminal icon to split the terminal horizontally. This allows you to run multiple command lines simultaneously.

3. Running Commands:
You can run any command in the integrated terminal just as you would in an external terminal. For example, `npm install` to install dependencies or `git status` to check the status of your git repository.

Advantages of Using the Integrated Terminal

1. Convenience and Integration:
    - Single Interface: Access your terminal without leaving the VS Code interface, keeping your workflow focused and reducing the need to switch between windows.
    - Workspace Context: The integrated terminal opens in the context of your workspace, so you don’t need to navigate to the project directory manually.

2. Enhanced Productivity:
    - SidebySide Editing: Easily view and edit code sidebyside with terminal output, which is helpful for debugging and running tests.
    - Terminal Shortcuts: Utilize keyboard shortcuts to quickly open, close, and switch between terminals, speeding up your workflow.

3. Customization and Theming:
    - Consistent Look and Feel: The terminal can inherit the VS Code theme and settings, providing a consistent and visually appealing interface.
    - Shell Customization: Customize the terminal to use your preferred shell and configuration directly within VS Code.

4. MultiTasking:
    - Multiple Terminals: Open multiple terminal instances and easily switch between them. This is particularly useful for running different tasks simultaneously, such as running a server in one terminal and running tests in another.
    - Split Terminal: Split the terminal to view and interact with multiple command lines at once.

5. Extension Support:

Extension Integration: Some VS Code extensions can interact directly with the integrated terminal, providing features like automated task running, terminal output parsing, and more.

The integrated terminal in VS Code offers numerous advantages over using an external terminal, including better integration with the editor, enhanced productivity, customization options, and support for multiple tasks. By using the integrated terminal, you can streamline your development process and maintain a more efficient and organized workflow.

7. FILE AND FOLDER MANAGEMENT:

In Visual Studio Code (VS Code), managing files and folders efficiently is crucial for an effective development workflow. Here’s how you can create, open, and manage files and folders, as well as navigate between them.

Creating Files and Folders

1. Using the Explorer View:
   - Open the Explorer View: Click the Explorer icon in the Activity Bar or press `Ctrl+Shift+E`.
   - Create a New File:
   - Rightclick on the folder where you want to create the file.
   - Select `New File`.
   - Enter the file name and press `Enter`.
   - Alternatively, you can click the `New File` icon at the top of the Explorer view.
   - Create a New Folder:
   - Rightclick on the folder where you want to create the new folder.
   - Select `New Folder`.
   - Enter the folder name and press `Enter`.
   - Alternatively, you can click the `New Folder` icon at the top of the Explorer view.

2. **Using the Command Palette:**
   - Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac) to open the Command Palette.
   - Type `New File` or `New Folder` and select the respective command.
   - Enter the name for the new file or folder and press `Enter`.

Opening Files and Folders

1. Using the Explorer View:
    - Open a File: Doubleclick the file you want to open in the Explorer view.
    - Open a Folder: Rightclick on the folder and select `Open Folder`.

2. Using the Command Palette:
    - Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac) to open the Command Palette.
    - Type `Open File` to open a specific file, or `Open Folder` to open a folder.
    - Navigate to the desired file or folder and select it.

3. Drag and Drop:

You can also drag and drop files or folders from your file system into the Explorer view in VS Code.

Managing Files and Folders

1. Rename:
    - Rightclick on the file or folder you want to rename.
    - Select `Rename` and enter the new name.
    - Press `Enter` to confirm the change.

2. Move:
    - Drag the file or folder to the new location within the Explorer view.
    - You can also cut (`Ctrl+X`) and paste (`Ctrl+V`) files or folders to move them.

3. Delete:
    - Rightclick on the file or folder you want to delete.
    - Select `Delete` and confirm the deletion.

Navigating Between Files and Directories Efficiently

1. Quick Open:
    - Press `Ctrl+P` (Windows/Linux) or `Cmd+P` (Mac) to open the Quick Open feature.
    - Start typing the name of the file you want to open. VS Code will filter and display matching files in realtime.
    - Select the file from the list to open it.

2. File Explorer:
    - Use the Explorer view to navigate through your project directories.

- Click on folders to expand and collapse them, and doubleclick files to open them.

3. Breadcrumbs:
   - Enable breadcrumbs by clicking the breadcrumbs icon at the top of the editor or by pressing `Ctrl+Shift+.`.
   - Use the breadcrumbs to navigate through the file's directory structure and quickly switch to other files or folders.

4. Go to Definition:
   - Rightclick on a function, variable, or class, and select `Go to Definition` to jump to its definition.
   - Use `F12` to go to the definition or `Ctrl+Click` on the item.

5. Open Editors:
   - View and manage open files in the `Open Editors` section at the top of the Explorer view.
   - Switch between open files by clicking on their names.

6. Tabs:
   - Open files are displayed in tabs at the top of the editor.
   - Click on a tab to switch to that file.
   - Rightclick on a tab to see options like `Close`, `Close Others`, and `Close to the Right`.

7. Navigate Back and Forward:
   - Use `Ctrl+` (Windows/Linux) or `Cmd+` (Mac) to navigate back to the previous location.
   - Use `Ctrl+Shift+` (Windows/Linux) or `Cmd+Shift+` (Mac) to navigate forward.

8. Split Editor:

Split the editor to view and edit multiple files side by side by rightclicking on a tab and selecting `Split Right` or `Split Down`.

By using these features and techniques, you can efficiently create, open, manage, and navigate files and folders in VS Code, enhancing your overall productivity and development workflow.

8. SETTINGS AND PREFERENCES:

In Visual Studio Code (VS Code), users can find and customize settings to tailor the editor to their preferences. These settings include appearance, font size, keybindings, and more. Here's how you can access and modify these settings:

Accessing Settings

1. Using the Settings GUI:
   - Go to `File > Preferences > Settings` (Windows/Linux) or `Code > Preferences > Settings` (Mac).
   - Alternatively, press `Ctrl+,` (Windows/Linux) or `Cmd+,` (Mac).

2. Using the Command Palette:
   - Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac) to open the Command Palette.
   - Type `Preferences: Open Settings` and select it.
3. Settings File:
   - For more advanced customization, you can directly edit the `settings.json` file.
   - Open the Command Palette and type `Preferences: Open Settings (JSON)`.

Customizing Settings

Changing the Theme

1. Using the Settings GUI:
   - Open the Settings GUI as described above.
   - In the search bar, type `Color Theme`.
   - Click on `Color Theme` and choose from the list of available themes.

2. Using the Command Palette:
   - Open the Command Palette.
   - Type `Color Theme` and select `Preferences: Color Theme`.
   - Use the arrow keys to navigate through the list and press `Enter` to select a theme.

Changing the Font Size

1. Using the Settings GUI:
   - Open the Settings GUI.
   - In the search bar, type `Font Size`.
   - Look for `Editor: Font Size` and enter your desired font size.

2. Using the Settings File:
   Open the `settings.json` file.
   Add or modify the following line:

```json
"editor.fontSize": 14
```

   Replace `14` with your preferred font size.

Changing Keybindings

1. Using the Keybindings GUI:
   - Go to `File > Preferences > Keyboard Shortcuts` (Windows/Linux) or `Code > Preferences > Keyboard Shortcuts` (Mac).
   - Alternatively, press `Ctrl+K Ctrl+S`.

2. Using the Command Palette:
   - Open the Command Palette.
   - Type `Preferences: Open Keyboard Shortcuts`.

3. Editing Keybindings:
   - In the Keyboard Shortcuts editor, you can search for a command and see its current keybinding.
   - To change a keybinding, click on the pencil icon next to the command.
   - Press the new key combination you want to assign and press `Enter`.
   - To remove a keybinding, click on the `X` icon next to the command.

4. Using the Keybindings File:
   You can also edit keybindings directly in the `keybindings.json` file.
   Open the Command Palette and type `Preferences: Open Keyboard Shortcuts (JSON)`.
   Add or modify keybindings in the following format:

```json
[
  {
    "key": "ctrl+alt+n",
    "command": "workbench.action.files.newUntitledFile"
  },
  {
    "key": "ctrl+shift+x",
    "command": "workbench.view.extension"
  }
```

]

Replace `"ctrl+alt+n"` and `"ctrl+shift+x"` with your desired key combinations, and the `"command"` values with the respective commands.

By using these methods, you can customize VS Code settings to fit your workflow and preferences. Whether changing the theme, adjusting the font size, or modifying keybindings, VS Code offers flexible and powerful ways to personalize your development environment.

9. DEBUGGING IN VS CODE:

Setting up and starting debugging in Visual Studio Code (VS Code) involves a few steps. Here's a stepbystep guide to set up and debug a simple program, along with an overview of key debugging features available in VS Code.

Steps to Set Up and Start Debugging

1. Install VS Code:
Make sure you have Visual Studio Code installed. You can download it from [here](https://code.visualstudio.com/).

2. Install Necessary Extensions:
Depending on the programming language you are using, you may need to install relevant extensions. For example, for Python, install the Python extension by Microsoft.

3. Open Your Project:
Open your project folder in VS Code by going to `File > Open Folder` and selecting your project directory.

4. Create or Open a Program File:
Create a new file or open an existing file with your program code. For example, if you are writing a simple Python program, create a file named `app.py`.

5. Write a Simple Program:
Write a simple program. For example, in `app.py`:
```python
def greet(name):
    return f"Hello, {name}!"

if __name__ == "__main__":
    name = input("Enter your name: ")
    print(greet(name))
```

6. Configure the Debugger:
    - Go to the Debug view by clicking the Debug icon in the Activity Bar on the side of the VS Code window or by pressing `Ctrl+Shift+D`.
    - Click on the gear icon (`Configure or Fix 'launch.json'`) to create a `launch.json` file if it doesn't exist. Select the appropriate environment for your language, e.g., "Python File" for Python.

    For Python, the `launch.json` file may look something like this:
    ```json
    {
        "version": "0.2.0",
        "configurations": [
            {
                "name": "Python: Current File",
                "type": "python",
                "request": "launch",
                "program": "${file}",
                "console": "integratedTerminal"
            }
        ]
    }
    ```

7. Set Breakpoints:
    - Set breakpoints in your code by clicking in the gutter to the left of the line numbers where you want to pause execution.

8. Start Debugging:
   Click the green play button in the Debug view or press `F5` to start debugging.

Key Debugging Features in VS Code

1. Breakpoints:
    - Set breakpoints by clicking in the left gutter next to the line number.
    - Conditional breakpoints can be set by rightclicking the breakpoint and adding a condition.

2. Watch:
    - Add variables to the Watch panel to monitor their values over time.
    - Rightclick on a variable in the editor and select `Add to Watch`.

3. Call Stack:
   View the call stack to see the sequence of function calls that led to the current point of execution.

4. Variables:
    - Inspect and modify variables in the Variables panel.
    - Hover over variables in the editor to see their current values.

5. Step Controls:
    - Use the step controls in the Debug toolbar to control execution:
    - Continue (`F5`): Resume execution until the next breakpoint.
    - Step Over (`F10`): Execute the next line of code, but don't step into functions.
    - Step Into (`F11`): Step into the function call.
    - Step Out (`Shift+F11`): Step out of the current function.

6. Integrated Terminal:
   Use the integrated terminal to run commands and see output directly within VS Code.

7. Debug Console:
   Use the Debug Console to evaluate expressions and execute commands in the context of the paused debug session.

8. Exception Handling:
   Configure exception handling to break on specific exceptions. This can be done in the `launch.json` file or through the UI.

Example: Debugging a Simple Python Program

1. Open VS Code and create a file named `app.py`.
2. Write the following Python code in `app.py`:
   ```python
   def greet(name):
       return f"Hello, {name}!"

   if __name__ == "__main__":
       name = input("Enter your name: ")
       print(greet(name))
   ```
3. Go to the Debug view (`Ctrl+Shift+D`) and click the gear icon to create a `launch.json` file.
4. Select "Python File" when prompted.
5. Set a breakpoint by clicking in the left gutter next to the `print(greet(name))` line.
6. Start debugging by pressing `F5`.
7. The program will pause at the breakpoint. You can inspect variables, step through the code, and use other debugging features.

By following these steps and using the described features, you can effectively debug your code in VS Code, making it easier to identify and fix issues.

10. USING SOURCE CONTROL:

Integrating Git with Visual Studio Code (VS Code) for version control is straightforward and enhances your workflow by providing seamless Git operations within the editor. Here's a stepbystep guide on how to initialize a repository, make commits, and push changes to GitHub.

Prerequisites

Install Git on your system. You can download it from [here](https://gitscm.com/).
Ensure you have a GitHub account.

Integrating Git with VS Code

1. Open Your Project in VS Code:
   - Open the project folder in VS Code by going to `File > Open Folder` and selecting your project directory.

Initializing a Repository

1. Open the Source Control View:
   - Click the Source Control icon in the Activity Bar on the side of the VS Code window or press `Ctrl+Shift+G`.

2. Initialize the Repository:
   - Click on the `Initialize Repository` button. This will create a new Git repository in your project folder.
   - Alternatively, you can open the terminal in VS Code by pressing `` Ctrl+` `` and run the following command:

```sh
git init
```

Making Commits

1. Stage Changes:
   - Open the Source Control view.
   - You will see a list of files that have changes. Hover over the files you want to stage and click the `+` icon next to them to stage the changes.
   - Alternatively, you can stage all changes by clicking on the `+` icon at the top of the changes list.

2. Commit Changes:
   - After staging the changes, you will see the staged changes section.
   - Enter a commit message in the message box above the changes list.
   - Click the checkmark icon (✓) or press `Ctrl+Enter` to commit the changes.

Pushing Changes to GitHub

1. Create a Repository on GitHub:
   Go to GitHub and create a new repository. Do not initialize it with a README, .gitignore, or license since you'll be pushing an existing repository.

2. Add Remote Repository:
   Copy the repository URL from GitHub.
   Open the terminal in VS Code (`` Ctrl+` ``) and run the following command:
   ```sh
   git remote add origin https://github.com/yourusername/yourrepository.git
   ```
   Replace `https://github.com/yourusername/yourrepository.git` with your actual repository URL.

3. Push Changes:
   Push your commits to GitHub by running the following command in the terminal:
   ```sh
   git push u origin master
   ```
   If you're using the main branch instead of master, use:
   ```sh
   git push u origin main
   ```
   You may be prompted to enter your GitHub username and password.

Additional Git Operations in VS Code

1. Pulling Changes:
   To pull changes from the remote repository, click the ellipsis (...) in the Source Control view and select `Pull`, or run the following command in the terminal:
   ```sh
   git pull
   ```

2. Branch Management:
   - You can create, switch, and manage branches within VS Code.
   - Click on the current branch name in the bottomleft corner of the status bar to switch branches.

- To create a new branch, open the Command Palette (`Ctrl+Shift+P`), type `Git: Create Branch`, and follow the prompts.

3. Viewing History and Diffs:
   - You can view the commit history and diffs by clicking on the commits in the Source Control view.
   - To see the history of a specific file, rightclick the file in the Explorer view and select `Git: View File History`.

Summary

By following these steps, you can effectively integrate Git with VS Code for version control. Initializing a repository, making commits, and pushing changes to GitHub are essential tasks that can be easily managed within the VS Code interface. Additionally, VS Code provides robust tools for branch management, viewing history, and handling other Git operations, making it a powerful environment for version control.

References:

- Copilot
- www.google.com
- Chatgpt
- https://git-scm.com/book/en/v2/Git-Basics-Working-with-Remotes
- https://docs.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh