**ASSIGNMENT:**

**1.How I Downloaded and Installed Visual Studio Code on Windows 11**

First, I made sure my computer met the minimum system requirements and that I had

administrative rights, as these were necessary for the installation.

I started by opening my web browser and visiting the [Visual Studio Code official website](https://code.visualstudio.com/). On the homepage, I found the "Download for Windows" button and clicked it to download the installer.

Once the download was complete, I navigated to my Downloads folder and double-clicked the VSCodeSetup.exe file to run the installer. The installation process began, and I was presented with the license agreement. I read through the agreement, checked the box to accept it, and clicked "Next."

The installer then asked me to choose the installation location. I decided to go with the default location and clicked "Next" to proceed. Next, I was given the option to select additional tasks. I chose to create a desktop icon and add VS Code to my PATH, which would allow me to use it from the command line. I also enabled other context menu entries that I found useful. After making my selections, I clicked "Next."

I then clicked the "Install" button to begin the actual installation. The installer started copying the necessary files to my system. Once the installation was complete, I was given the option to launch Visual Studio Code immediately. I checked the "Launch Visual Studio Code" box and clicked "Finish."

**2. First-time Setup:**

After VS Code launched, I decided to install some extensions to enhance my coding experience. I clicked on the Extensions icon in the Activity Bar on the side of the window or used the shortcut Ctrl+Shift+X to open the Extensions view.

Next, I customized my VS Code settings to suit my workflow. I went to File > Preferences > Settings and adjusted various options to my liking.

Since I use Git, Python and Prettier, for version control, I set up Git, python and prettier integration by installing the their extension and configuring my Git , python and prettier settings.

Finally, I checked for any available updates to ensure I had the latest features and bug fixes. I did this by going to the Help menu and selecting "Check for Updates."

### 3. Main Components of the VS Code User Interface

When I first opened Visual Studio Code, I was struck by its clean and organized layout. The user interface is designed to be intuitive and efficient, making it easy to navigate through my coding projects.

ACTIVITY BAR

On the far left side of the window, I found the Activity Bar. This vertical bar is my gateway to different views and functionalities within VS Code. It has a series of icons, each representing a different section. For example, the Explorer icon lets me browse through my project’s files and folders, while the Search icon helps me find text within my files. There’s also the Source Control icon for managing version control with Git, the Run and Debug icon for handling my debugging tasks, and the Extensions icon that opens up the extension marketplace. This bar is incredibly handy for quickly switching between different tasks without getting lost.

SIDE BAR

Directly next to the Activity Bar is the Side Bar. The contents of this area change depending on which icon I select in the Activity Bar. When I’m in the Explorer view, it shows me a detailed directory structure of my project, making it easy to navigate through my files. If I switch to the Source Control view, it displays my changes, commits, and branches. In the Extensions view, it lists all the extensions I have installed and allows me to search for new ones. The Side Bar provides contextual information and controls, keeping everything I need within easy reach.

EDITOR GROUP

At the heart of the interface is the Editor Group. This central area is where I spend most of my time, writing and editing code. Each file I open appears as a tab at the top of the editor group, allowing me to switch between files effortlessly. I can also split the editor into multiple groups, enabling me to view and edit multiple files side by side. The syntax highlighting feature enhances readability by color-coding different elements of my code based on the file type, which is particularly useful for spotting errors and understanding complex code at a glance.

STATUS BAR

Finally, at the bottom of the window lies the Status Bar. This horizontal bar provides a wealth of information about my current workspace and open files. It shows details like the line and column number, file type, and encoding. The Status Bar also indicates the current Git branch and the status of my repository, which is crucial for version control. Additionally, it displays the number of errors and warnings in my code, helping me keep track of issues that need fixing. There are also various shortcuts available, such as changing the language mode, selecting a different terminal, and managing notifications, making it a powerful tool for quick actions.

**5. The Command Palette in Visual Studio Code**

One of the most powerful features I discovered in Visual Studio Code is the Command Palette. This tool provides quick access to a wide array of commands and functions within the editor, all from a single interface. It eliminates the need to navigate through multiple menus or remember complex keyboard shortcuts.

Accessing the Command Palette

To open the Command Palette, I simply press Ctrl + Shift + P on my keyboard. Alternatively, I can access it through the menu by going to View > Command Palette. When opened, the Command Palette presents a text input field where I can type commands or search for specific functions. This makes it incredibly easy to find and execute commands without breaking my workflow.

Common Tasks Performed Using the Command Palette

The versatility of the Command Palette is impressive, and here are some of the common tasks I often perform using it:

1. Opening Files:
   1. When I need to open a file quickly, I type >Open File in the Command Palette, and it lets me select the file I need from my workspace.
2. Running Debugging Commands:
   1. To start a debugging session, I simply type >Debug: Start Debugging. This command launches the debugger without me needing to navigate through the menus.
3. Git Commands:
   1. Managing my Git repositories is straightforward with the Command Palette. For instance, I type >Git: Clone to clone a repository or >Git: Commit to commit my changes.
4. Extension Management:
   1. Installing new extensions is easy. I type >Extensions: Install Extensions to open the Extensions view, where I can browse and install the tools I need.
5. Changing Language Mode:
   1. When I switch between different programming languages, I use the Command Palette to change the syntax highlighting and language support by typing >Change Language Mode.
6. Formatting Code:
   1. Keeping my code clean is important, so I frequently use >Format Document to format the entire document according to my settings. For selected portions of code, I type >Format Selection.
7. Searching Commands:
   1. The Command Palette is also great for finding commands quickly. By typing a few letters or keywords, such as >find, it shows me all related commands, saving me time.
8. Changing Themes:
   1. Personalizing my coding environment is easy with the Command Palette. Typing >Preferences: Color Theme lets me change the color theme of the editor.
9. Viewing Keyboard Shortcuts:
   1. To view and modify keyboard shortcuts, I type >Preferences: Open Keyboard Shortcuts. This command opens a comprehensive list of shortcuts available in VS Code.
10. Reloading the Window:
    1. Occasionally, I need to reload the VS Code window, especially after installing new extensions or changing settings. I type >Reload Window, and it refreshes the editor without closing my work.

**5.Extensions in VS Code**

Extensions play a crucial role in enhancing the functionality of Visual Studio Code, making it a versatile tool for developers across various programming disciplines. They allow users to customize their development environment to better suit their specific needs by adding new features, languages, debuggers, and tools.

To find, install, and manage extensions in VS Code, I start by opening the Extensions view. I can do this by clicking on the Extensions icon in the Activity Bar on the side of the VS Code window or by pressing Ctrl+Shift+X. This view provides a search bar where I can type keywords related to the functionality I need or the name of a specific extension.Once I've found the extension I'm interested in, I can install it by clicking the "Install" button next to the extension in the search results. After installation, the extension is immediately available for use. Managing extensions is straightforward; I can enable, disable, update, or uninstall extensions by clicking on the gear icon next to the extension name in the Extensions view and selecting the appropriate option.

For web development, several extensions are particularly useful.

1. The Live Server extension is invaluable for real-time previews of web pages as I develop, providing a local server environment with live reload capabilities.
2. Prettier helps in maintaining code style consistency by automatically formatting code, which is essential in collaborative projects.
3. ESLint assists in identifying and fixing code quality issues in JavaScript, ensuring adherence to coding standards.
4. The Debugger for Chrome extension allows me to debug JavaScript code directly in Google Chrome, which is especially useful for front-end development.
5. Path Intellisense offers auto-completion for file paths, enhancing navigation within projects.
6. Auto Rename Tag automatically renames paired HTML or XML tags, reducing the risk of mismatched tags.
7. Finally, GitLens extends the built-in Git capabilities of VS Code, providing detailed insights into code changes, authorship, and project history

**6.Integrated Terminal**

The integrated terminal in Visual Studio Code (VS Code) is a handy feature that allows me to run commands directly within the editor. This means I can stay in one place while working on my projects, without the need to switch between different applications.

Opening the Integrated Terminal

To open the integrated terminal in VS Code, I have a couple of options:

Menu Method: I can go to the top menu, click on View, and then select Terminal. There's also a keyboard shortcut for it, which is Ctrl+ (the backtick key, usually found just below the Escape key on most keyboards).

Command Palette: I can also open the Command Palette by pressing Ctrl+Shift+P, then typing "Terminal: New Terminal" and selecting it from the list.When the terminal opens, it appears at the bottom of the VS Code window, so I can keep an eye on my code and the output of my commands at the same time.

Using the Integrated Terminal

In the integrated terminal, I can run all kinds of commands just like I would in a regular terminal. This includes running scripts, compiling code, managing files, and using version control tools like Git. The terminal supports different shells, like PowerShell, Command Prompt, and various UNIX shells like Bash, depending on what's available on my system and my settings in VS Code.

Advantages of Using the Integrated Terminal

There are several reasons I find the integrated terminal in VS Code especially useful:

1. Convenience: It's super convenient to have the terminal right there in the editor. I can run my commands and see the output without having to leave VS Code, which saves me time and keeps me focused.
2. Context Awareness: The terminal knows where my project is because it opens in the directory of my current project in VS Code. This makes it easy to run commands that are specific to my project without having to navigate to the right folder manually.
3. Efficiency: I can easily copy and paste between the terminal and the editor. This is great for passing code snippets to scripts or feeding command arguments without interrupting my flow.
4. Unified Workflow: With the terminal integrated, I can use version control systems, run build scripts, and execute command-line tools all from within VS Code. This keeps everything in one place, which I find really efficient.
5. Customization: VS Code allows me to customize the terminal’s appearance and behavior, like adjusting the font size, choosing the shell path, and integrating with other tools and extensions. This means I can set up my environment exactly the way I like it.
6. Session Management: I can open multiple terminal tabs within VS Code, which makes it easy to manage different tasks or sessions simultaneously.

**7.File and Folder Management**

Managing files and folders in Visual Studio Code (VS Code) is designed to be intuitive, making it easier for me to organize my projects and navigate through them smoothly. Here’s how I handle creating, opening, and managing files and folders, along with some tips for navigating them efficiently:

Creating Files and Folders

1. New Files: If I need to create a new file, I simply right-click in the Explorer sidebar and choose New File, or I use the shortcut Ctrl+N. This opens up a new tab where I can start writing my code or adding content.
2. New Folders: To organize my project better, I might want to create a new folder. For this, I right-click in the Explorer sidebar and select New Folder. This helps keep related files together.
3. Opening Files and Folders
4. Opening Files: When I want to work on an existing file, I go to File > Open File... or press Ctrl+O. A dialog box pops up, letting me browse my file system to find the file I need.
5. Opening Folders: To open a whole project, I choose File > Open Folder... or press Ctrl+K Ctrl+O. This opens the folder and its contents in the Explorer sidebar, giving me a view of all the files in that project.

Managing Files and Folders

1. Renaming: If I need to rename a file or folder, I just right-click on it in the Explorer and select Rename, or I can click on the name and start typing a new one directly.
2. Deleting: Deleting files or folders is as simple as right-clicking on them and selecting Delete, though I usually get a prompt to confirm this action.
3. Moving or Copying: I can move files or folders by dragging them to a new location within the Explorer, or by using Cut and Paste commands from the context menu or keyboard shortcuts.

Navigating Between Files and Directories Efficiently

1. Quick Open: One of my favorite features is the Quick Open, activated by pressing Ctrl+P. It lets me type the name of the file I want to open, and it quickly searches through my project to find the right file, making it super fast to switch between files.
2. Explorer Sidebar: The Explorer sidebar shows me the folder structure of my project. I can click on any folder or file here to open it in the editor, making it easy to navigate through my project.
3. Breadcrumbs: At the top of the editor, breadcrumbs display the path to the currently open file. Clicking on any part of this trail allows me to quickly navigate to different sections of my project.
4. Search: If I'm looking for specific text or files, the search feature (Ctrl+Shift+F) is incredibly useful. It lets me find anything in my project without having to manually browse through the files.

**8.Settings and Preferences**

In Visual Studio Code (VS Code), customizing the editor to fit my preferences is straightforward, allowing me to tailor the environment to enhance my productivity and comfort. Here’s how I can find and adjust settings like the theme, font size, and keybindings on Windows:

Accessing Settings

To access the settings in VS Code:

Open Settings from the Menu: I can go to File > Preferences > Settings. This will open the Settings editor where I can adjust various settings according to my needs.

Using the Keyboard Shortcut: A quicker way to open the Settings editor is by pressing Ctrl+,. This shortcut brings up the settings directly.

Accessing Settings JSON Directly: For more advanced settings or manual edits, I can open the settings JSON file by clicking on the {} icon in the top-right corner of the Settings editor, switching to the JSON view where I can directly modify the settings file.

Customizing the Theme

Changing the Theme: To change the theme, I can search for "Color Theme" in the Settings editor search bar. This brings up the Color Theme settings, where I can click on the Color Theme dropdown menu. I can preview different themes by clicking on them, and I can select the one that best suits my preference. Alternatively, I can use the keyboard shortcut Ctrl+K Ctrl+T to open the theme picker directly.

Adjusting Font Size

Changing Font Size: In the Settings editor, I can search for "Font Size" to find the setting for adjusting the editor's font size. By default, this might be set to 14 pixels, but I can increase or decrease this value according to my comfort. This change affects only the size of the text in the editor, not in menus or other UI elements.

Customizing Keybindings

Modifying Keybindings: To change keybindings, I go to File > Preferences > Keyboard Shortcuts. This opens the Keyboard Shortcuts editor, where I can search for a specific command and click the pencil icon next to it to edit its keybinding. Alternatively, I can click on the plus icon to add a new keybinding.

Editing Keybindings JSON:

For more detailed customization, I can open the keybindings JSON file by clicking on the {} icon at the top-right corner of the Keyboard Shortcuts editor. This allows me to directly edit the JSON file where I can add, modify, or delete keybindings manually.

**9.Debugging in VS Code**

Setting up and starting debugging in Visual Studio Code (VS Code) is straightforward, providing a powerful environment for identifying and fixing issues in my code. Here's how I can set up a simple program for debugging, along with some key features that make the debugging process effective.

Setting Up a Simple Program for Debugging

First, I need to ensure that Visual Studio Code is installed on my system. If it's not, I can download it from the official website. Once installed, I open VS Code and load the folder containing the program I want to debug by choosing File > Open Folder....

Next, I select the file that contains my code and click to open it in the editor. To set up debugging, I click on the Run and Debug icon in the Activity Bar or press Ctrl+Shift+D. If this is my first time setting up debugging for a specific language, VS Code might prompt me to install the necessary debugger extension.

When prompted, I create a launch.json file, which stores the configuration needed for the debugger. I select the environment that matches my programming language, which helps VS Code understand how to run and debug my program.

With my configuration in place, I set breakpoints by clicking in the gutter next to the line numbers in the editor. These breakpoints are where execution will pause, allowing me to inspect the state of my application at those points.

To start debugging, I press the green play button in the Run and Debug panel or press F5. This starts my program in debug mode, and it will pause at the breakpoints I set, allowing me to step through the code.

Key Debugging Features in VS Code

Breakpoints: These are crucial as they allow me to pause execution at specific points in my code to check what's happening at that moment.

Call Stack: The call stack window shows me the sequence of function calls that led to the current point in execution, helping me understand the flow of my application.

Variables and Watches: These panels display the current values of variables in my program. I can also add expressions to the watch list to monitor specific values as I step through my code.

Debug Console: The debug console is a great tool for interacting with my program's runtime environment. I can evaluate expressions, run commands, and print outputs to get insights into how my program is behaving.

Step Over, Step Into, Step Out: These controls let me manage execution flow in my code. Stepping over lines lets me continue without diving into called functions, stepping into allows me to enter a function's execution, and stepping out exits the current function to the caller.

Watch Expressions: I can add expressions to the watch list, which continuously monitor specific values as I debug, helping me keep track of important data throughout the execution.

Call Stack Navigation: This feature allows me to navigate back through the sequence of function calls, which is useful for understanding the path my program execution has taken to reach the current state.

**10.Using Source Control**

Integrating Git with Visual Studio Code (VS Code) for version control is seamless and provides a powerful environment for managing source code changes. Here's how I can set up Git in VS Code, initialize a repository, make commits, and push changes to GitHub:

Setting Up Git in VS Code

Install Git: Before using Git in VS Code, I need to have Git installed on my computer. I can download and install Git from git-scm.com.

Configure Git: Once Git is installed, I should configure it with my user information. I can set my username and email in Git by opening a command prompt or terminal and running:

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

Open VS Code: Launch Visual Studio Code and open the folder containing my project files.

Open the Source Control Panel: I can access the Source Control view by clicking on the Source Control icon in the Activity Bar on the side of the window.

Initializing a Repository

Initialize the Repository: If my project is not yet a Git repository, I can initialize it by opening the Source Control panel and clicking on the Initialize Repository button. This will create a new Git repository in my project directory.

Confirm Initialization: After initializing, I can see the .git directory in my project folder, indicating that the repository has been set up.

Making Commits

Staging Changes: After making changes to my files, I need to stage these changes for commit. In the Source Control panel, I can see a list of modified files. I can stage the changes by hovering over the files and clicking the + icon, or by clicking Stage All Changes to stage all modified files.

Committing Changes: Once the changes are staged, I can write a commit message in the message box at the top of the Source Control panel and click the checkmark icon to commit the changes. Committing creates a snapshot of my files at this point in time.

Pushing Changes to GitHub

Connect to GitHub: To push changes to GitHub, I first need to link my local repository to a remote repository on GitHub. I can do this by going to GitHub, creating a new repository, and copying the repository URL.

Add Remote Repository: In the terminal within VS Code, I can set the remote URL for my repository using the command:

git remote add origin [URL of the repository]

Replace [URL of the repository] with the URL of the GitHub repository.

Push Changes: To push my committed changes to GitHub, I use the command:

git push -u origin master

This pushes my changes to the master branch of the remote repository on GitHub. The -u flag sets the upstream reference, so in future, I can simply use git push without specifying the branch.

Authenticate: If prompted, I will need to enter my GitHub username and password. For added security, GitHub may require a personal access token instead of a password.