### ASSIGNMENT 2: INSTALLATION AND NAVIGATION OF VISUAL STUDIO CODE

### NAME: LAMECK KIPSANG

### EMAIL: [lameck.cs16@gmail.com](mailto:lameck.cs16@gmail.com)

### 1.Installation of VS code:

Describe the steps to download and install vs code on windows 11 operating system. Include any prerequisites that might be needed.

### Prerequisites

1. **Operating System**: Ensure you are using Windows 11.
2. **Internet Connection**: Required for downloading VS Code.

### Steps to Download and Install VS Code

1. **Download Visual Studio Code**:
   * Open your web browser and go to the official Visual Studio Code website: [Visual Studio Code](https://code.visualstudio.com/).
   * Click on the "Download for Windows" button. This will start downloading the installer file (typically named something like VSCodeSetup-x64-1.x.x.exe).
2. **Run the Installer**:
   * Once the download is complete, navigate to your downloads folder and double-click the installer file to run it.
   * If prompted by the User Account Control (UAC), click "Yes" to allow the installer to make changes to your device.
3. **Install Visual Studio Code**:
   * The setup wizard will open. Click "Next" to proceed.
   * Read and accept the license agreement by checking the "I accept the agreement" box, then click "Next".
   * Choose the installation location or leave it at the default path (e.g., C:\Program Files\Microsoft VS Code\). Click "Next".
   * Choose whether to create a Start Menu folder or not, then click "Next".
   * Select additional tasks you want to perform. It's recommended to check the following options for easier use:
     + Add "Open with Code" action to Windows Explorer file context menu.
     + Add "Open with Code" action to Windows Explorer directory context menu.
     + Register Code as an editor for supported file types.
     + Add to PATH (required to use VS Code from the command line).
   * Click "Next" and then "Install" to begin the installation process.
4. **Finish Installation**:
   * Once the installation is complete, you can choose to launch VS Code immediately by checking the "Launch Visual Studio Code" box.
   * Click "Finish" to complete the installation process.
5. **Launch and Set Up VS Code**:
   * If you didn’t choose to launch VS Code immediately, you can open it later from the Start Menu or by searching for "Visual Studio Code" in the Windows search bar.
   * Upon first launch, you may be prompted to install recommended extensions. You can choose to install them or skip this step for now.
6. **Update VS Code (Optional)**:
   * VS Code periodically checks for updates and will notify you if an update is available. You can manually check for updates by going to Help > Check for Updates.

### Additional Steps (Optional)

* **Install Extensions**: To enhance the functionality of VS Code, you can install extensions for various programming languages, tools, and frameworks. Go to the Extensions view (Ctrl+Shift+X) and search for the desired extensions.
* **Customize Settings**: VS Code allows extensive customization. You can adjust settings by navigating to File > Preferences > Settings or using the command palette (Ctrl+Shift+P) and searching for settings.

### 2. First-time Setup:

After installing VS code what initial configurations and settings should be adjusted for an optimal coding environment?Mention any important settings or extensions.

**Important Settings**

1. **User and Workspace Settings**:
   * Open the settings panel by pressing Ctrl+, or navigating to File > Preferences > Settings.
   * Adjust settings to your preference. For example:
     + **Theme**: Choose a color theme that is comfortable for you. (Color Theme in the settings or Ctrl+K Ctrl+T).
     + **Font Size**: Adjust the editor font size. (Editor: Font Size).
     + **Tab Size**: Set the number of spaces for a tab. (Editor: Tab Size).
     + **Word Wrap**: Enable word wrap to avoid horizontal scrolling. (Editor: Word Wrap).
2. **Auto Save**:
   * Enable auto-save to avoid losing changes. (Files: Auto Save).
3. **Format on Save**:
   * Enable format on save to automatically format code according to your settings. (Editor: Format On Save).
4. **Line Numbers**:
   * Display line numbers for easier code navigation. (Editor: Line Numbers).
5. **Bracket Pair Colorization**:
   * Enable bracket pair colorization for better visibility of matching brackets. (Editor: Bracket Pair Colorization).

**Key Extensions**

1. **Language Support**:
   * **Python**: Microsoft’s Python extension for linting, debugging, and IntelliSense.
   * **JavaScript/TypeScript**: Built-in support, but consider additional extensions like ESLint.
   * **C/C++**: Microsoft’s C/C++ extension for IntelliSense, debugging, and code browsing.
   * **HTML/CSS**: Built-in support, with additional extensions like Live Server for a live preview.
2. **Code Linting and Formatting**:
   * **ESLint**: Integrates ESLint into VS Code for JavaScript and TypeScript.
   * **Prettier**: A code formatter that supports many languages.
3. **Version Control**:
   * **GitLens**: Enhances Git capabilities in VS Code.
4. **IntelliSense and Code Snippets**:
   * **JavaScript (ES6) code snippets**: Adds snippets for JavaScript ES6.
   * **Python Docstring Generator**: Automatically generates docstrings for Python.
5. **Docker**:
   * Provides tools for developing and deploying containerized applications.
6. **Remote Development**:
   * **Remote - WSL**: Enables you to use the Windows Subsystem for Linux as your full-time development environment.
   * **Remote - SSH**: Connect to a remote machine via SSH and work with its file system.
7. **Debugging**:
   * **Debugger for Chrome**: Debug JavaScript code running in the Chrome browser.
   * **Python**: Includes a debugger for Python.
8. **Themes and Icons**:
   * **Material Icon Theme**: Adds icons to the file explorer.
   * **One Dark Pro**: A popular dark theme for VS Code.

**Additional Configurations**

1. **Keyboard Shortcuts**:
   * Customize keyboard shortcuts to increase productivity. Navigate to File > Preferences > Keyboard Shortcuts or press Ctrl+K Ctrl+S.
2. **Sync Settings**:
   * Use Settings Sync to keep your settings, extensions, and keyboard shortcuts synced across multiple machines. (Settings Sync: Turn On).
3. **Terminal Integration**:
   * Customize the integrated terminal settings. For example, set the default shell to your preferred terminal (PowerShell, Command Prompt, Git Bash, etc.). (Terminal: Integrated: Shell).

### 3. User Interface Overview

Explain the main components of the VS code user interface. Identify and describe the purpose of the Activity Bar, Side Bar, Editor Group and Status Bar.

**Main Components of the VS Code User Interface**

1. **Activity Bar**:
   * **Location**: The vertical bar on the far left of the window.
   * **Purpose**: Provides access to different views and functions such as the Explorer, Search, Source Control, Run and Debug, Extensions, and additional views depending on installed extensions.
   * **Details**:
     + **Explorer**: File and folder navigation within your project.
     + **Search**: Find and replace functionality across files.
     + **Source Control**: Integration with version control systems like Git.
     + **Run and Debug**: Access to debugging configurations and controls.
     + **Extensions**: Manage and install extensions to enhance VS Code.
2. **Side Bar**:
   * **Location**: The vertical pane adjacent to the Activity Bar on the left side of the window.
   * **Purpose**: Displays different views and tools selected from the Activity Bar. It can show the file explorer, search results, source control changes, debugging information, and more.
   * **Details**:
     + **Explorer View**: Shows the folder and file structure of your workspace.
     + **Source Control View**: Displays changes, commit history, and provides tools for version control.
     + **Debug View**: Displays debugging tools, variables, call stacks, and watch expressions.
     + **Extensions View**: Lists installed extensions and allows for managing and searching for new ones.
3. **Editor Group**:
   * **Location**: The central area of the window where you write and edit code.
   * **Purpose**: Hosts one or more editor tabs where files can be viewed and edited. Supports multiple editor groups side by side, allowing for split-screen editing.
   * **Details**:
     + **Tabs**: Each open file is represented by a tab. You can open multiple tabs for different files.
     + **Split View**: Allows you to divide the editor into multiple groups to view and edit files side by side.
     + **Editor Layout**: Customize how editor groups are arranged.
4. **Status Bar**:
   * **Location**: The horizontal bar at the bottom of the window.
   * **Purpose**: Provides information about the current state of the editor and workspace. It displays details like line and column numbers, the current language mode, encoding, Git branch, and more.
   * **Details**:
     + **Line and Column Indicator**: Shows the current cursor position.
     + **Language Mode**: Indicates the programming language of the current file.
     + **Git Branch**: Displays the current Git branch if the workspace is under version control.
     + **Notifications**: Shows messages and alerts from extensions and VS Code itself.
     + **Quick Actions**: Provides quick access to common actions and settings.

### 4.Command Palette:

**What is the command palette in VS Code and how can it be accessed?Provide examples of common tasks that can be performed using the Command Palette.**

The Command Palette in Visual Studio Code (VS Code) is a powerful feature that provides quick access to a wide range of commands and settings. It allows you to execute commands, navigate to files, and perform various tasks without having to navigate through menus and toolbars.

**Accessing the Command Palette**

The Command Palette can be accessed in the following ways:

* **Keyboard Shortcut**: Press Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
* **Menu**: Go to the top menu and select View > Command Palette.

**Common Tasks Performed Using the Command Palette**

Here are some examples of common tasks that can be performed using the Command Palette:

1. **Opening and Navigating Files**:
   * **Quick Open**: Type Ctrl+P (or just start typing after opening the Command Palette) to quickly open files by name.
   * **Go to Symbol**: Type @ followed by the symbol name to navigate to a specific symbol within the file.
2. **Running Commands**:
   * **Toggle Terminal**: Type > Toggle Integrated Terminal to show or hide the integrated terminal.
   * **Format Document**: Type > Format Document to format the entire document according to the configured formatter.
   * **Change Language Mode**: Type > Change Language Mode to change the syntax highlighting and language features for the current file.
3. **Source Control**:
   * **Git: Commit**: Type > Git: Commit to commit changes with a message.
   * **Git: Pull**: Type > Git: Pull to pull the latest changes from the remote repository.
4. **Extension Management**:
   * **Install Extensions**: Type > Extensions: Install Extensions to open the Extensions view and search for new extensions.
   * **Disable Extensions**: Type > Extensions: Disable to disable a specific extension.
5. **Debugging**:
   * **Start Debugging**: Type > Debug: Start Debugging to start a debugging session.
   * **Add Configuration**: Type > Debug: Add Configuration to add or edit launch configurations for debugging.
6. **Settings and Preferences**:
   * **Open Settings**: Type > Preferences: Open Settings to open the settings editor.
   * **Configure User Snippets**: Type > Preferences: Configure User Snippets to create or edit code snippets for specific languages.
7. **Development Tasks**:
   * **Run Task**: Type > Tasks: Run Task to execute predefined tasks from the tasks.json file.
   * **Create New File**: Type > File: New File to create a new untitled file.

### 5. Extensions in VS code

**Discuss the role of extensions in VS code. How can users find, install, and manage extensions? Provide examples of essential extensions for web development.**

### Role of Extensions in VS Code

1. **Language Support**: Add syntax highlighting, IntelliSense, and other features for different programming languages.
2. **Debugging**: Integrate debugging tools for various languages and frameworks.
3. **Productivity**: Enhance productivity with tools for code formatting, linting, snippets, and task runners.
4. **Source Control**: Improve version control integration with additional features and services.
5. **Customization**: Customize the look and feel of the editor with themes and icon packs.
6. **Collaboration**: Enable real-time collaboration with tools like Live Share.

### Finding, Installing, and Managing Extensions

Finding Extensions

1. **Extensions View**: Access the Extensions view by clicking the Extensions icon in the Activity Bar or by pressing Ctrl+Shift+X (Windows/Linux) or Cmd+Shift+X (Mac).
2. **Search**: Use the search bar in the Extensions view to find extensions by name or functionality.
3. **Marketplace**: Browse extensions on the [Visual Studio Code Marketplace](https://marketplace.visualstudio.com/VSCode).

Installing Extensions

1. **From Extensions View**:
   * Find the extension you want to install.
   * Click the Install button next to the extension.
2. **From Marketplace**:
   * Visit the extension page on the Marketplace.
   * Click the Install button, which will prompt VS Code to open and install the extension.

Managing Extensions

1. **Enable/Disable**:
   * Open the Extensions view.
   * Find the installed extension.
   * Click the gear icon and select Enable or Disable.
2. **Uninstall**:
   * Open the Extensions view.
   * Find the installed extension.
   * Click the gear icon and select Uninstall.
3. **Update**:
   * Extensions usually update automatically, but you can manually check for updates by clicking the gear icon in the Extensions view and selecting Check for Updates.

### Essential Extensions for Web Development

1. **Language and Framework Support**:
   * **ESLint**: Integrates ESLint into VS Code for JavaScript and TypeScript linting.
   * **Prettier - Code Formatter**: Formats code consistently according to Prettier rules.
   * **Python**: Provides rich support for Python, including IntelliSense, linting, and debugging.
   * **HTML, CSS, and JavaScript**: Built-in support, often enhanced by additional extensions like HTML CSS Support.
2. **Productivity**:
   * **Live Server**: Launches a local development server with live reload feature for static and dynamic pages.
   * **Path Intellisense**: Autocompletes file names and paths.
   * **Auto Close Tag**: Automatically closes HTML/XML tags.
   * **Auto Rename Tag**: Automatically renames paired HTML/XML tags.
3. **Version Control**:
   * **GitLens**: Enhances Git capabilities, provides insights into code authorship, and more.
   * **Git History**: Provides a visual history of Git repositories.
4. **Debugging**:
   * **Debugger for Chrome**: Debug JavaScript code running in the Google Chrome browser directly from VS Code.
   * **Debugger for Firefox**: Similar to the Chrome debugger but for Firefox.
5. **Theming and Customization**:
   * **One Dark Pro**: A popular dark theme for VS Code.
   * **Material Icon Theme**: Adds a comprehensive set of icons for better visual navigation.

### **6**. Integrated Terminal

**Describe how to open and use the integrated terminal in VS code. What are the advantages of using the integrated terminal compared to an external terminal.**

### Opening the Integrated Terminal

1. **Using the Menu**:
   * Go to the top menu and select Terminal > New Terminal.
2. **Using the Keyboard Shortcut**:
   * Press Ctrl+ (Windows/Linux) or Cmd+ (Mac).
3. **Using the Command Palette**:
   * Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
   * Type Terminal: Create New Integrated Terminal and press Enter.

**Using the Integrated Terminal**

1. **Basic Commands**:
   * Once the terminal is open, you can use it just like any other command-line interface. Type your commands and press Enter to execute them.
2. **Multiple Terminals**:
   * You can open multiple terminal instances by clicking the + icon in the terminal panel or using the Ctrl+Shift+ (Windows/Linux) or Cmd+Shift+ (Mac) shortcut. Each terminal can run different tasks or commands.
3. **Switching Terminals**:
   * Switch between open terminals using the drop-down menu at the top right of the terminal panel or the keyboard shortcuts Ctrl+PageDown and Ctrl+PageUp (Windows/Linux) or Cmd+ (Mac).
4. **Splitting Terminals**:
   * Split the terminal view to run multiple commands side by side by clicking the split icon in the terminal panel.
5. **Terminal Configuration**:
   * Customize the terminal settings by navigating to File > Preferences > Settings and searching for "terminal". You can configure the default shell, font size, cursor style, and more.

**Advantages of Using the Integrated Terminal Compared to an External Terminal**

1. **Seamless Workflow**:
   * The integrated terminal allows you to stay within the VS Code environment, reducing context switching and increasing productivity. You can easily run scripts, execute commands, and view outputs without leaving the editor.
2. **Project Context**:
   * The integrated terminal opens in the context of your current workspace, meaning it starts in the root directory of your project. This makes it easier to run project-specific commands and scripts without needing to navigate to the project directory manually.
3. **Integrated Tools**:
   * You can take advantage of VS Code’s features, such as multi-cursor editing, IntelliSense, and version control, while working in the terminal. For example, you can quickly open files from the terminal by holding Ctrl (Windows/Linux) or Cmd (Mac) and clicking on file paths.
4. **Consistency Across Platforms**:
   * The integrated terminal behaves consistently across different operating systems, providing a unified experience whether you are on Windows, macOS, or Linux.
5. **Extension Integration**:
   * Extensions can enhance the functionality of the integrated terminal, adding features like task running, terminal multiplexing, and more. This allows for a more powerful and customized terminal experience within VS Code.
6. **Customization**:
   * The integrated terminal can be customized to suit your preferences, including changing the default shell (e.g., Bash, PowerShell, Command Prompt), adjusting the appearance, and configuring key bindings.

### 7. File and Folder Management

Explain how to create, open, and manage files and folders in VS code. How can users navigate between different files and directories efficiently?

**Creating Files and Folders**

1. **Creating Files**:
   * **Using the Explorer View**:
     + Open the Explorer view by clicking the Explorer icon in the Activity Bar or pressing Ctrl+Shift+E.
     + Right-click on the folder where you want to create the new file and select New File.
     + Enter the name for the new file and press Enter.
   * **Using the Command Palette**:
     + Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
     + Type File: New File and press Enter.
     + Save the new untitled file by providing a name and location.
2. **Creating Folders**:
   * **Using the Explorer View**:
     + Open the Explorer view by clicking the Explorer icon in the Activity Bar or pressing Ctrl+Shift+E.
     + Right-click on the parent directory where you want to create the new folder and select New Folder.
     + Enter the name for the new folder and press Enter.

**Opening Files and Folders**

1. **Opening Files**:
   * **Using the Explorer View**:
     + Navigate through the directory tree in the Explorer view and double-click on the file you want to open.
   * **Using Quick Open**:
     + Press Ctrl+P (Windows/Linux) or Cmd+P (Mac) to open the Quick Open panel.
     + Start typing the name of the file you want to open and select it from the list.
2. **Opening Folders**:
   * **Using the Menu**:
     + Go to File > Open Folder and browse to the folder you want to open.
   * **Using the Command Palette**:
     + Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
     + Type File: Open Folder and press Enter.

**Managing Files and Folders**

1. **Renaming Files and Folders**:
   * In the Explorer view, right-click on the file or folder you want to rename and select Rename.
   * Enter the new name and press Enter.
2. **Deleting Files and Folders**:
   * In the Explorer view, right-click on the file or folder you want to delete and select Delete.
   * Confirm the deletion when prompted.
3. **Moving Files and Folders**:
   * Drag and drop files or folders within the Explorer view to move them to a new location.

**Navigating Between Files and Directories Efficiently**

1. **Quick Open**:
   * Press Ctrl+P (Windows/Linux) or Cmd+P (Mac) to quickly open the Quick Open panel.
   * Start typing the name of the file or directory to quickly navigate to it.
2. **Go to Definition**:
   * Use Ctrl+Click (Windows/Linux) or Cmd+Click (Mac) on a function or variable to jump to its definition within the code.
3. **Explorer View**:
   * Use the Explorer view (Ctrl+Shift+E) to navigate the directory structure visually.
   * You can expand and collapse folders to quickly navigate through the hierarchy.
4. **Breadcrumbs**:
   * Enable breadcrumbs by navigating to View > Show Breadcrumbs or using the command View: Toggle Breadcrumbs from the Command Palette.
   * Breadcrumbs provide a navigation path at the top of the editor, allowing you to click and navigate to parent directories and files.
5. **File Tabs**:
   * Open files are represented as tabs at the top of the editor. Click on a tab to switch to that file.
   * Use Ctrl+Tab (Windows/Linux) or Cmd+Tab (Mac) to cycle through open tabs.
6. **Keyboard Shortcuts**:
   * Use Ctrl+Shift+O (Windows/Linux) or Cmd+Shift+O (Mac) to navigate to a symbol within the current file.
   * Use Ctrl+T (Windows/Linux) or Cmd+T (Mac) to search for symbols across all files in the workspace.

### 8. Settings and Preferences:

Where can users find and customize settings in VS code? Provide examples of how to change the theme, font size, and key bindings.

### Accessing Settings

1. **Via the Menu**:
   * Go to File > Preferences > Settings (Windows/Linux) or Code > Preferences > Settings (Mac).
2. **Via Keyboard Shortcut**:
   * Press Ctrl+, (Windows/Linux) or Cmd+, (Mac).
3. **Via Command Palette**:
   * Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
   * Type Preferences: Open Settings and select it.

### Customizing Settings

Settings can be edited in the settings UI or directly in the settings.json file. Here are examples of how to customize specific settings:

Changing the Theme

1. **Using the Settings UI**:
   * Open the settings as described above.
   * In the search bar at the top, type theme.
   * Under Color Theme, select your preferred theme from the dropdown list.
2. **Using the Command Palette**:
   * Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
   * Type Preferences: Color Theme and press Enter.
   * Use the arrow keys to navigate through the list of available themes and press Enter to select one.

Changing the Font Size

1. **Using the Settings UI**:
   * Open the settings as described above.
   * In the search bar at the top, type font size.
   * Find Editor: Font Size and set your desired font size.
2. **Using the settings.json File**:
   * Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
   * Type Preferences: Open Settings (JSON) and select it.
   * Add or modify the following line:

json

"editor.fontSize": 14

* + Save the file.

Changing Key Bindings

1. **Using the Keybindings UI**:
   * Open the settings as described above.
   * Go to File > Preferences > Keyboard Shortcuts or press Ctrl+K Ctrl+S.
   * In the search bar, type the command you want to change (e.g., toggle terminal).
   * Click on the pencil icon next to the command and press the new key combination you want to assign.
2. **Using the keybindings.json File**:
   * Open the Command Palette by pressing Ctrl+Shift+P (Windows/Linux) or Cmd+Shift+P (Mac).
   * Type Preferences: Open Keyboard Shortcuts (JSON) and select it.
   * Add or modify a key binding entry. For example, to change the key binding for the integrated terminal:

json

{

"key": "ctrl+`",

"command": "workbench.action.terminal.toggleTerminal"

}

* + Save the file.

### 9. Debugging in VS code:

Outline the steps to set up and start debugging a simple program in VS code. What are some key debugging features available in VS code?

### Steps to Set Up and Start Debugging a Simple Program

1. Install Necessary Extensions

Depending on the programming language you are using, you might need to install relevant extensions. For example:

* **Python**: Install the "Python" extension.
* **JavaScript/Node.js**: The built-in support is usually sufficient, but "ESLint" and other helpful extensions can be installed.
* **C/C++**: Install the "C/C++" extension.

2. Open or Create Your Project

1. **Open VS Code**.
2. **Open Folder**: Go to File > Open Folder and select the folder containing your project files.

3. Create a Simple Program

Create a new file in your project. For example, a simple Python program:

python

# hello.py

print("Hello, world!")

Or a simple JavaScript program:

javascript

Copy code

// hello.js

console.log("Hello, world!");

4. Set Up Debug Configuration

1. **Open the Run and Debug View**: Click on the Run icon in the Activity Bar on the side of the window or press Ctrl+Shift+D.
2. **Create Launch Configuration**: Click the gear icon to create a launch.json file, which will configure how the debugger should start your program.
   * For Python, a simple launch.json might look like this:

json

{

"version": "0.2.0",

"configurations": [

{

"name": "Python: Current File",

"type": "python",

"request": "launch",

"program": "${file}",

"console": "integratedTerminal"

}

]

}

* + For Node.js, a simple launch.json might look like this:

json

"version": "0.2.0",

"configurations": [

{

"type": "node",

"request": "launch",

"name": "Launch Program",

"skipFiles": ["<node\_internals>/\*\*"],

"program": "${file}"

}

]

}

5. Set Breakpoints

1. **Open the file** you want to debug.
2. **Set Breakpoints**: Click in the gutter to the left of the line numbers to set a breakpoint. A red dot will appear indicating the breakpoint.

6. Start Debugging

1. **Run the Debugger**: Go to the Run and Debug view, select your configuration, and click the green play button (Start Debugging) or press F5.
2. The debugger will start, and execution will pause at your breakpoints, allowing you to inspect the state of your application.

### Key Debugging Features in VS Code

1. **Breakpoints**:
   * Set, enable, disable, and remove breakpoints by clicking in the editor gutter.
   * Conditional breakpoints can be set to pause execution only when certain conditions are met.
2. **Watch Expressions**:
   * Add expressions to the Watch panel to monitor their values as you step through the code.
3. **Call Stack**:
   * View the call stack to see the sequence of function calls that led to the current point in the execution.
4. **Variable Inspection**:
   * Inspect variables in the Variables panel to see their current values.
   * Hover over variables in the editor to see their values inline.
5. **Step Controls**:
   * **Continue (F5)**: Resume execution until the next breakpoint.
   * **Step Over (F10)**: Execute the next line of code but do not step into functions.
   * **Step Into (F11)**: Step into functions to debug line-by-line.
   * **Step Out (Shift+F11)**: Step out of the current function and pause.
6. **Integrated Terminal**:
   * The integrated terminal can be used to run commands and view outputs without leaving the VS Code environment.
7. **Debug Console**:
   * Evaluate expressions and execute commands in the context of the debug session.

### 10.Using Source Control

How can users integrate Git with VS code for version control? Describe the process of initializing a repository, making commits, and pushing changes to Github.

### Integrating Git with VS Code

1. Install Git

Before integrating Git with VS Code, ensure you have Git installed on your system. You can download and install Git from [git-scm.com](https://git-scm.com/).

2. Open VS Code and Set Up Git

1. **Open your project in VS Code**:
   * Launch VS Code and open the folder containing your project by selecting File > Open Folder.
2. **Verify Git Installation**:
   * Open the integrated terminal by pressing Ctrl+ (Windows/Linux) or Cmd+ (Mac).
   * Type git --version to check if Git is installed and accessible from VS Code.

### Initializing a Git Repository

1. **Initialize the Repository**:
   * Open the Source Control view by clicking the Source Control icon in the Activity Bar or pressing Ctrl+Shift+G.
   * If your project folder is not already a Git repository, you'll see an Initialize Repository button. Click it to initialize a Git repository in your project folder.
2. **Verify Initialization**:
   * After initializing, you should see a list of your project files with a U (untracked) status icon.

### Making Commits

1. **Stage Changes**:
   * Open the Source Control view (Ctrl+Shift+G).
   * You'll see a list of changes in your project. Hover over a file and click the + icon to stage individual changes, or click the + icon next to Changes to stage all changes.
2. **Commit Changes**:
   * Once changes are staged, enter a commit message in the input box at the top of the Source Control view.
   * 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](https://github.com/) and log in to your account.
   * Click the + icon in the top-right corner and select New repository.
   * Fill in the repository name, description (optional), and other details. Click Create repository.
2. **Add Remote Repository**:
   * Copy the URL of the newly created GitHub repository.
   * In VS Code, open the integrated terminal (Ctrl+ or Cmd+).
   * Add the remote repository by running:

bash

git remote add origin <repository-url>

Replace <repository-url> with the URL you copied from GitHub.

1. **Push Changes to GitHub**:
   * Push your local commits to GitHub by running:

bash

git push -u origin master

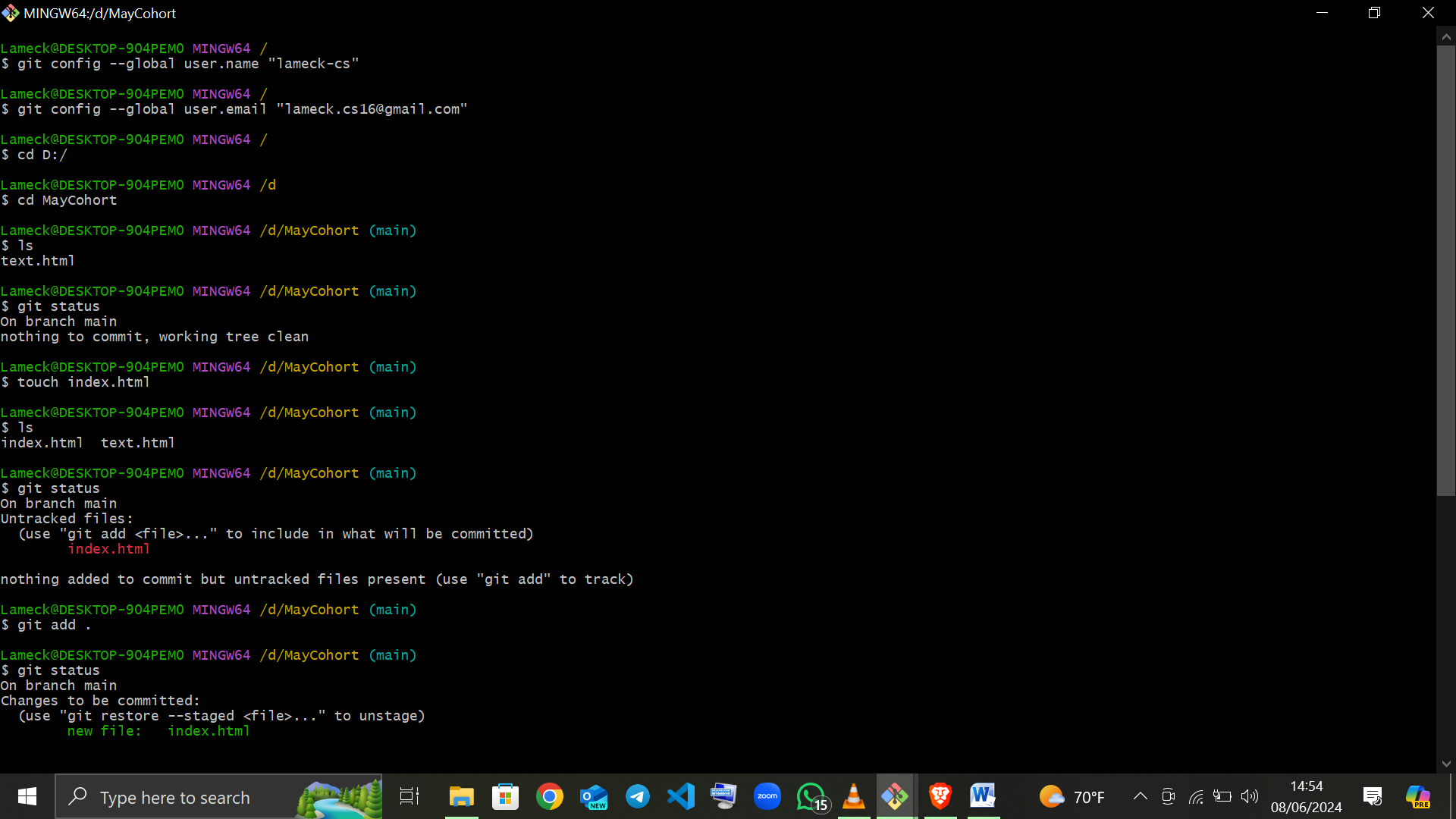
This command pushes your changes to the master branch of the remote repository. If you're using a different branch, replace master with your branch name.

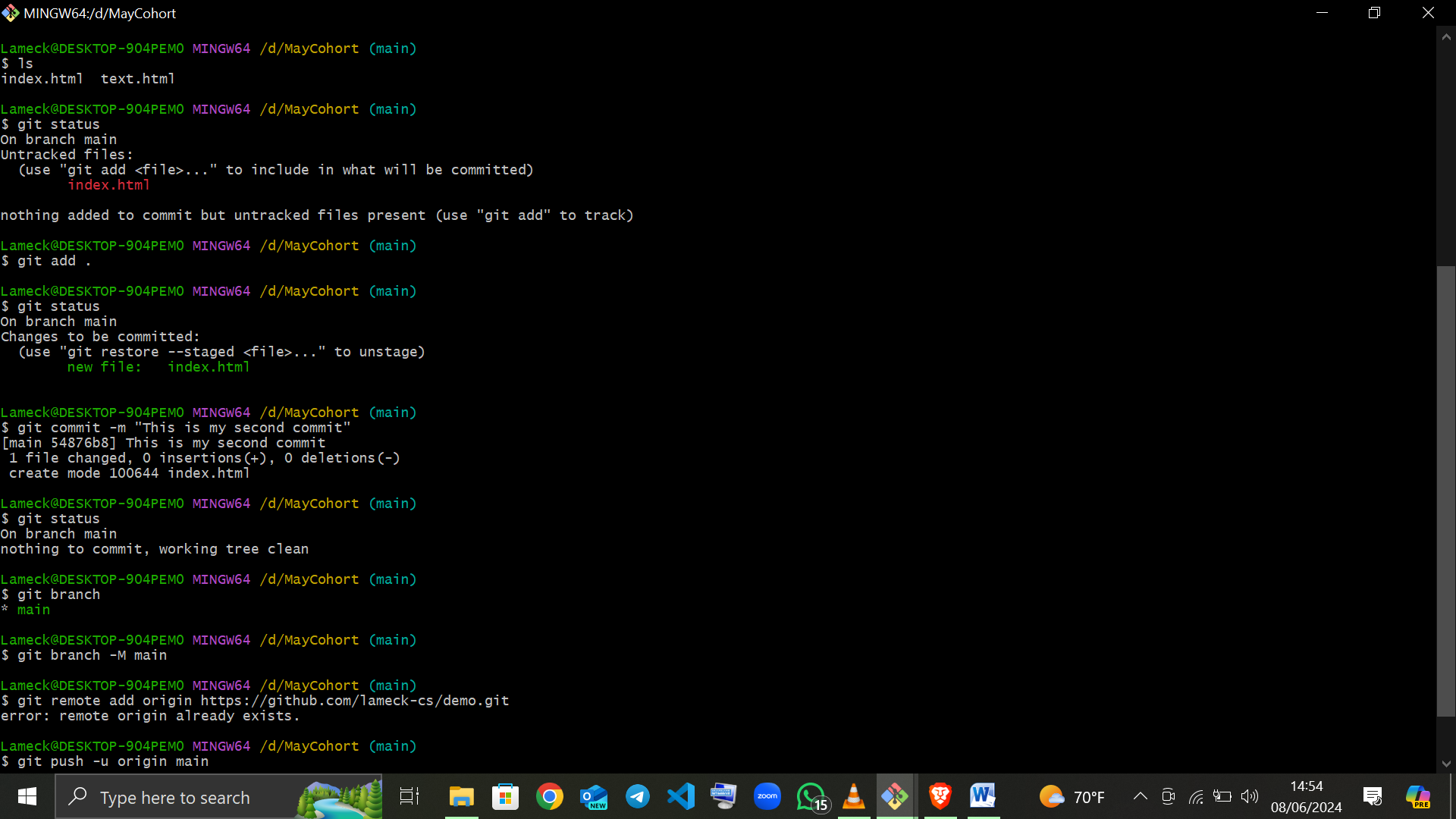
### Summary of Key Commands

* **Initialize a Repository**: git init
* **Stage Changes**: git add <file-name> or git add . (for all changes)
* **Commit Changes**: git commit -m "commit message"
* **Add Remote Repository**: git remote add origin <repository-url>
* **Push Changes**: git push -u origin master

### Key Features of Git Integration in VS Code

1. **Source Control View**: Provides an overview of changes, staged files, and allows for easy committing.
2. **Integrated Terminal**: Allows running Git commands directly within VS Code.
3. **GitLens Extension**: Enhances Git capabilities in VS Code by providing features like blame annotations, commit history, and more.
4. **Visual Diff**: View diffs between file versions directly in the editor.
5. **Branch Management**: Create, switch, and manage branches from within VS Code.





References And Sources:

The steps provided for integrating Git with Visual Studio Code (VS Code) for version control are based on standard practices widely documented in official VS Code documentation, Git documentation, and various tutorials available online.

Some of the answers are also from Chatgpt and AI chatbot: GEMINI