1. Installation of VS Code:
   * Describe the steps to download and install Visual Studio Code on Windows 11 operating system. Include any prerequisites that might be needed.

**To install VS, verify the following prerequisites:**

1. Make sure Windows 11 is your operating system.
2. Processor: 1.6 GHz or faster processor.
3. RAM of at least 1 GB.
4. An active internet connection is needed to download the VS Code installer from the official website.
5. Have administrator access.

**Steps of installing and setting up VS code**

Download the VS code from the official website.

Navigate to the Visual Studio Code download page by opening your web browser.

The "Download for Windows" button should be clicked. This will start the Windows setup for VS Code to download.

Once downloaded, locate the installer file (usually in your Downloads folder).

Double-click the installer file to start the installation process: Follow the installation set up instructions e.g accepting the license agreement.

Choose the destination location for installation.

Select additional tasks (such as adding VS Code to the PATH or creating a desktop shortcut).

Click "Install" to begin the installation.

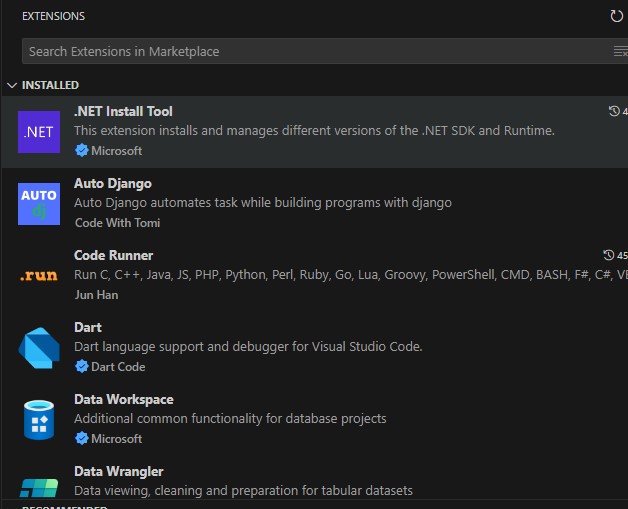
Launch VS Code:

After installation, you can choose to launch VS Code immediately by selecting the option during installation or by locating it in your Start Menu.

1. 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.

**Initial Configurations and Settings to be adjusted:**

1. **Settings:**
   * **Theme**: Choose a theme that is comfortable for your eyes. VS Code offers several built-in themes like Dark+, Light+, and Quiet Light. You can install additional themes from the VS Code marketplace (Ctrl+Shift+X).
   * **Editor Settings**: Customize the editor behavior, such as tab size, indentation, line wrapping, and line numbers. Modify these settings in the *settings.json* file or through the UI.
   * **Access Settings:** By going to *File > Preferences > Settings*
   * **Font :** Configure the preferred font family and font size, also choose a theme, *workbench.colorTheme.*
   * **Tab Size:** Set the number of spaces for tabs (*editor.tabSize*) based on your coding style (commonly 2 or 4 spaces).
   * **Line Numbers:** Enable line numbers for better navigation (*editor.lineNumbers*).
   * **Auto Save:** Choose whether to enable auto save *(files.autoSave)* based on your preference (e.g., *onWindowChange, afterDelay*).
2. **Important Extensions:**
   * + **Prettier - Code formatter**: For code formatting according to predefined rules.
     + **GitLens**: Enhances Git integration with inline Git blame annotations, repository/file history, etc.
     + **Bracket Pair Colorizer**: Colorizes matching brackets to improve code readability.
     + **Live Server**: Launches a local development server with live reload capability for web development.
     + **Debugger for Chrome**: Allows debugging JavaScript code in the Chrome browser.
     + **Code Spell Checker**: Checks spelling in your comments and strings.



1. 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.

Activity Bar

The Activity Bar offers rapid access to several panels and views within VS Code and is situated on the far left side of the window. It makes switching between different features and functionalities easy. Explorer, Search, Run & Debug, and Extensions are some of its components.

Side Bar

Adjacent to the Activity Bar, the Side Bar usually shows controls and details pertaining to the view or editor that is presently active. It offers more background information and useful tools related to your coding assignments. File Explorer, source control, and extensions are some of its constituent parts.

Editor Group

Code and text files can be written and edited in the Editor Group, which is made up of one or more editors. In the VS Code window, it takes up the center section. Tabbed interface, file navigation, and split view are some of its components.

Status Bar

The Status Bar, which is found at the bottom of the VS Code window, gives you rapid access to several settings and actions in addition to providing information about the status of your project and editor at the moment. Notification and Git Branch are two of its components.

1. 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 potent tool that lets you easily access a variety of commands and functions.

It can be accessed by pressing Ctrl+ Shift + P or going to view in the Menu bar and selecting Command Palette.

Here are examples of tasks you can perform using the Command Palette:

1. Opening Files and Folders: Type File: Open File to open a specific file, Type File: Open Folder to open a folder in the workspace.
2. Type Extensions: Install Extensions to browse and install new extensions from the VS Code Marketplace, Type Extensions: Show Installed Extensions to see a list of currently installed extensions.
3. Version Control: Type Git: Pull to pull changes from a remote repository, Type Git: Commit to commit staged changes to the local repository.
4. Searching and Navigating: Type Find in Files to search for specific text across your project.
5. Searching and Navigating: Type Find in Files to search for specific text across your project.
6. Searching and Navigating: Type Find in Files to search for specific text across your project.
7. 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.

**Enhanced VS code Functionality:** Extensions extend VS Code beyond its basic features by adding new capabilities and tools.

**Language Support:** They provide essential support for a variety of programming languages, including features like syntax highlighting, code completion, and linting.

**Debugging Tools:** Extensions offer support for debugging across different environments and frameworks, enhancing developers' ability to troubleshoot code.

**Productivity Tools:** They include tools for automating tasks, formatting code, managing snippets, and integrating with version control systems, boosting overall productivity.

**Customization:** Extensions empower users to personalize their coding environment with custom themes, icons, and tailoring the editor to individual preferences.

**Finding, Installing, and Managing Extensions:**

To find extensions, users can navigate to the Extensions view in VS Code (Ctrl+Shift+X), search by name or functionality, and explore featured or categorized extensions.

Installing extensions: click on an extension to view details and click "Install."

Once installed, extensions can be managed directly from the Extensions view, enabling users to enable, disable, update, or uninstall them as needed by clicking on them and navigating to options asking for management e.g uninstall, disable.

**Examples of Essential Extensions for Web Development:**

* **ESLint:** Provides real-time linting for JavaScript and TypeScript, ensuring code quality.
* **Prettier:** Automatically formats code based on predefined rules, promoting consistent coding styles.
* **Live Server:** Launches a local development server with live reload capability, ideal for web development.
* **Debugger for Chrome:** Facilitates debugging JavaScript in Chrome directly from VS Code.
* **Path Intellisense:** Autocompletes file names and paths, streamlining navigation within projects.
* **CSS Peek:** Allows peeking into CSS class definitions from HTML or JavaScript files, enhancing CSS editing efficiency.

1. 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:** To access the integrated terminal, press Ctrl + `(backtick). As an alternative, you can use the Command Palette (Ctrl + Shift + P) to type View: Toggle Integrated Terminal or navigate to the menu bar and select View > Terminal.

**Using the Integrated Terminal**: Once it's operational, the integrated terminal can be used in the same way as a standard terminal:

Use commands like *cd* to navigate directories.

Execute instructions like git clone, npm install, and so on.

Launch the development server with a command like npm start or python app.py.

Utilize debugging commands to see output right in Visual Studio Code.

**Smooth Integration:**

Because of the integrated terminal's close integration with VS Code, you can easily transition between editing and terminal duties without opening separate programs.

**Awareness of Context:**

It's simple to run commands in the context of your project because the terminal opens to the root of your current workspace by default.

**Direct Access to the Commands in VS Code:**

The integrated terminal allows you to execute VS Code commands directly, including code. to open files and code -g FILE:LINE to navigate to a specified line.

**Persistence of Workspace:**

For continuous development tasks, the terminal's ability to maintain its state between sessions—including its working directory, command history, and environment settings—can be useful.

**Increased Output:**

You may take advantage of the terminal shortcuts and features that come with VS Code, including integrated debugging, auto-completion, and inline problem indicators (such linting errors).

**Personalization and Expansion:**

With the help of extensions and settings (settings.json), you may customize the integrated terminal to meet your unique requirements (e.g., custom themes, key bindings).

1. 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**

Creating Files: Click on the Explorer icon in the Activity Bar (or press Ctrl + Shift + E) to open the File Explorer. Right-click in the Explorer area and select New File, or use the Command Palette (Ctrl + Shift + P) and type New File.

Creating Folders: Similarly, right-click in the Explorer area and select New Folder, or use the Command Palette and type New Folder.

**Opening Files and Folders**

Opening Files: Double-click on a file in the Explorer to open it in the editor. Use the Command Palette (Ctrl + Shift + P) and type File: Open File, then select the file to open.

Opening Folders: Click on File > Open Folder... in the menu bar, navigate to the folder you want to open, and click Select Folder. Alternatively, drag a folder into the VS Code window to open it.

**Managing Files and Folders**

Renaming and Deleting: Right-click on a file or folder in the Explorer and choose Rename or Delete. You can also use keyboard shortcuts (F2 for renaming, Delete for deleting). Moving and Copying: To move a file or folder, drag it to a new location within the Explorer.

Searching and Filtering: Use the search box at the top of the Explorer to filter files by name. Type Ctrl + Shift + F to search across all files in the current workspace. Navigating Between Files and Directories Efficiently.

File Navigation: Switching Between Open Files: Use Ctrl + Tab to cycle through open files or Ctrl + P to quickly open files by name. Navigating to Symbols: Use Ctrl + Shift + O to navigate to a symbol (e.g., functions, classes) within the current file.

**Directory Navigation:**

Explorer View: Use the Explorer in the Activity Bar to navigate through files and folders.

Breadcrumbs: Utilize the breadcrumbs navigation at the top of the editor to quickly navigate up the directory tree or switch between files.

Command Palette: Use the Command Palette (Ctrl + Shift + P) for quick access to commands related to file and folder management, such as creating new files/folders, renaming, deleting, etc.

1. Settings and Preferences:
   * Where can users find and customize settings in VS Code? Provide examples of how to change the theme, font size, and keybindings.

Users can find and customize settings in Visual Studio Code (VS Code) through the Settings UI or by directly editing the settings.json file.

**Examples of how to change the theme, font size, and keybindings**

**Theme**:Modify the "*workbench.colorTheme*" setting in *settings.json*, for example: "*workbench.colorTheme": "Dark+".*

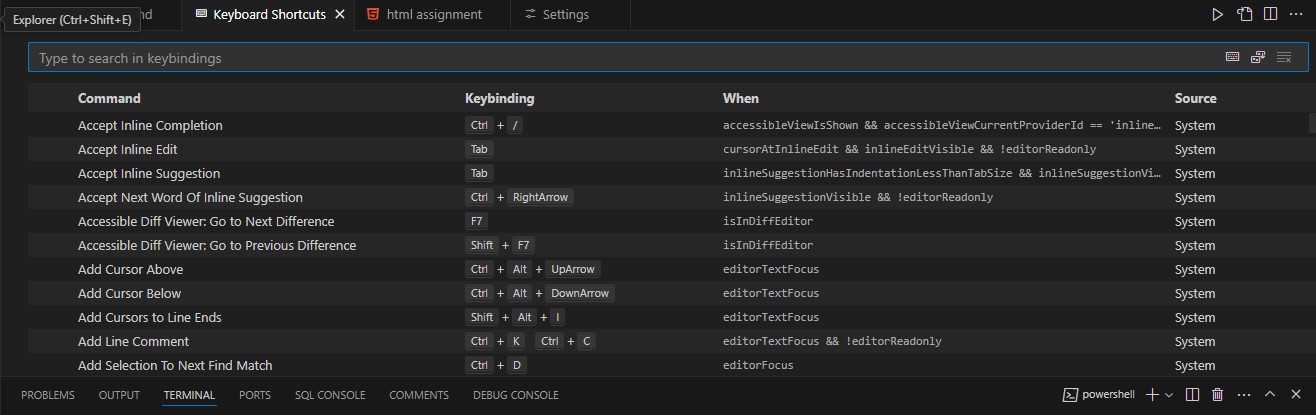
"*workbench.iconTheme*": "*material-icon-theme*",

**Font size:** Modify the "*editor.fontSize*" setting in *settings.json*, for example: "*editor.fontSize": 14*

**Keybindings:**

To customize keybindings in Visual Studio Code (VS Code), you have two main methods:

1. **Using the Command Palette:** Open the Command Palette (*Ctrl+Shift+P* and search for "Preferences: Open Keyboard Shortcuts". Navigate to the Keyboard Shortcuts tab, find the command you want to change, and edit its keybinding. Save your changes by pressing *Enter*.



1. **Editing keybindings.json File**: Press Ctrl+Shift+P (or Cmd+Shift+P) to open the Command Palette, then type "Preferences: Open Settings (JSON)".

Change the shortcut keys.JSON format is used in a file to specify unique keybindings for particular operations.

After making modifications, press Ctrl+S to save the keybindings.json file.

*Example:*

{"workbench.action.reloadWindow" }**,** { "key": "ctrl+shift+i", "command": "editor.action.formatDocument”

1. 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?
2. **Install Necessary Extensions:**
   * For example, install the "Python" extension for Python or "Debugger for Chrome" for JavaScript.
3. **Open Your Project:**
   * Open VS Code and navigate to **File** -> **Open Folder** to open your project folder.
4. **Create a Debug Configuration:**
   * Open the debug view (*Ctrl+Shift+D*) and click the gear icon to create a *launch.json* file.
   * Select the appropriate environment and customize the configuration as needed.
5. **Set Breakpoints:**
   * Click in the gutter next to the line numbers to set breakpoints in your code.
6. **Start Debugging:**
   * Click the green play button (Start Debugging) or press *F5.*

**Key Debugging Features in VS Code:**

1. **Breakpoints:** Pause execution at specific lines of code.
2. **Watch Expressions:** Monitor values of expressions as you step through the code.
3. **Call Stack:** View the sequence of function calls.
4. **Variable Inspection:** Inspect variables' values in the current scope.
5. **Step In/Out/Over:** Navigate through the code execution.
6. **Debug Console:** Execute arbitrary expressions and interact with the running program.
7. **Inline Values:** View variable values inline within the editor.
8. 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.

**Process of integrating Git with VS Code**

1. On your PC, open Visual Studio Code.
2. Launch or Duplicate a Repository**:** In VS Code, navigate to the project folder then go to View, thenTerminal.
3. To create a new Git repository, use git init; to clone an existing repository, use git clone <repository-url>.
4. Set Up User Information for Git: Press Ctrl+Shift+P to launch the command palette. To set up your Git user name and email, look up "Git: Set User Name" and "Git: Set User Email".

**Process of initializing a repository, making commits, and pushing changes to GitHub**

1. **Initializing a repository**

Launch Visual Studio Code and open your project folder.

Open Integrated Terminal: Open the terminal by navigating to **View** then **Terminal** or by pressing *Ctrl + `.*

Initialize Git by running this code: *git init*

1. **Making commits**

In the Source Control view (Ctrl+Shift+G), you'll see the files that have been changed.Click the + icon next to each file to stage them, or click the + icon at the top to stage all changes.

Commit Changes**:** Enter a commit message in the text box provided in the Source Control view. Click the checkmark icon to commit the staged changes, or run the following command in the terminal:

*git commit -m "Your commit message"*

1. **Pushing changes to GitHub**

After adding a remote repository type command *git push* to push commitments into the remote repository.

**Open the Source Control View**: Use the keyboard shortcut Ctrl+Shift+G, or click the Source Control button in the Activity Bar on the side of the window.

**References**

1. Class Notes and recordings
2. https://code.visualstudio.com/