Setting Up GitHub Instructional Guide

1. Definitions & Lingo.....1

2. GitHub Profile Set-up.....2

3. GitHub Desktop Set-up.....3

4. IDE Set-up (VS Code).....4

5. Creating a Repository.....6

6. Updating a Repository.....7

7. Templates & Open Source.....8

8. Exploration: GitHub Pages Deployment.....9

1. Definitions & Lingo

Before starting, familiarize yourself with the vocabulary of version control.

IDE (Integrated Development Environment): A software application that bundles developer tools, like a code editor, debugger, and compiler, into a single interface (e.g. VS Code, Cursor, Google Antigravity).

Version Control (Git): A system that records changes to a file or set of files over time so that you can recall specific versions later.

Repository (Repo): The fundamental unit of GitHub. It is a folder that contains all your project files and the revision history for each file.

Remote vs. Local:

- **Remote:** The version of your repository hosted on the internet (GitHub.com).
- **Local:** The version of your repository stored on your physical computer's hard drive.

Clone: The action of downloading a Remote repository to your Local machine to create a connected copy.

Commit: A snapshot of your local files at a specific point in time. You must write a "Commit Message" to explain what changed.

Push: Uploading your local repository content (commits) to a remote repository.

Pull: Fetching, downloading, and altering content from a remote repository and immediately updating the local repository to match that content.

Pull Request (PR): A formal proposal to merge code changes from one branch of a repository into another.

Branch: A parallel version of a repository. It is contained within the repository but does not affect the primary (usually called `main`) branch until you merge it.

Merge: Combines changes from one branch (like a feature branch) into another (like the main branch).

Terminal (Command Line): A text-based interface used to give instructions to your computer. In VS Code, this is built-in.

Workspace: In VS Code, this refers to the root folder of your project, including all its specific settings and configurations.

Markdown (.md): A lightweight markup language used to format text (like headings, bolding, and lists) in plain-text files.

2. GitHub Profile Set-up

A. Account Creation

1. Navigate to GitHub.com and click **Sign Up**.
2. Follow the prompts. Use your CSU-provided or personal email (you can add or change emails later if desired) and verify.

B. Profile Customization

1. Click your avatar in the top-right corner and select **Settings**.
2. In the "Public Profile" section, enter your Name, a brief Bio (e.g., "Engineering Student at University"), and upload a photo.

C. [Optional] Exploration: The Special Profile README

This creates a unique content section on your main profile page.

1. Create a new repository (see Section 5 for steps).
2. **Crucial Step:** Name the repository **exactly the same** as your GitHub username. (e.g., If your username is JaneDoe, the repo name must be JaneDoe).
3. Ensure the repository is **Public**.
4. Check the box **Initialize this repository with a README**.
5. Once created, click the **Pencil Icon** (Edit) on the README.md file.
6. **Editing:** Use Markdown to add text.
    - Headings: Use # for large text.
    - Images: Use the format ![Alt Text](url-to-image).
    - Links: Use [Link Text](URL).
7. Scroll down to "Commit changes" to save.

✨ Try It Now: Create your profile repository and add a single sentence or emoji to the README. Navigate to your main profile page to see it appear!

3. GitHub Desktop Set-up

A. Installation

1. Download GitHub Desktop.
2. **Sign In:** Launch the app and sign in with your GitHub browser account.
3. **Configure Git:** The app will ask for a name and email. Use the **same email address** associated with your GitHub account to ensure your contributions are attributed correctly.

B. What You Can Do with GitHub Desktop:

- **Clone Repositories** – Download any repository from GitHub to your local machine with just a few clicks.
- **View Changes** – See exactly what files you've modified with a visual diff viewer.
- **Commit & Push** – Stage your changes, write commit messages, and push to GitHub without touching the command line.
- **Branch Management** – Create, switch, and merge branches with an intuitive interface.

C. Connecting to VS Code:

GitHub Desktop works seamlessly with VS Code, the IDE you will be setting up in Section 4 (IDE Set-up). After cloning a repository (Section 6):

1. Right-click on the repository in GitHub Desktop
2. Select "Open in Visual Studio Code"
3. Start editing your files!

Any changes you make will automatically appear in GitHub Desktop for committing.

4. IDE Set-up (VS Code)

Configuring your Integrated Development Environment. We will be using Visual Studio Code (VS Code), a free, open-source, lightweight code editor developed by Microsoft.

A. Installation

1. Go to the Visual Studio Code Download Page.
2. Download and run the installer for your OS.

B. Orientation & Navigation

- **Activity Bar (Far Left):** The narrow strip with icons for Explorer (files), Search, Source Control (Git), and Extensions.
- **Command Palette:** The most powerful tool in VS Code.
    - **Shortcut:** `Ctrl + Shift + P` (Windows) or `Cmd + Shift + P` (Mac).
    - **Usage:** Type commands here to do almost anything.
- **Terminal:** Toggle the built-in command line with `Ctrl + ~` (tilde).

C. *[Optional] Exploration: Essential Extensions*

Click the Extensions icon (squares) in the Activity Bar and install:

1. **Live Server** (by Ritwick Dey): Launches a local development server to view webpages.
2. **Prettier** (by Prettier): Automatically formats code.

D. *[Optional] Exploration: AI-Assistance & Prompting*

1. Setting up GitHub Copilot (free version) (Recommended)

1. **Install:** In the VS Code Extensions tab, search for and install **GitHub Copilot**.
2. **Auth:** A popup will appear asking you to sign in to GitHub. Click **Allow** and sign in with your account.
3. **Verify:** Look for the small Copilot icon (a little alien/pilot face) in the bottom-right corner of the window. If it spins or has a line through it, click it to check your status.
4. **Usage:** Start typing code, and ghost text will appear. Press `Tab` to accept suggestions. You can also ask Copilot what you want to make, and it will write code for you!
    - *Docs:* GitHub Copilot in VS Code Overview
5. **Want more from Copilot?** As a student, you can get free access to Copilot Pro via GitHub Education. See how to apply.

2. Other AI Coding Tools

- **Claude Code:** You can install extensions like "Claude Dev" to use Anthropic's Claude models directly in your editor. Claude Code in VS Code Docs
- **Gemini Code Assist:** Google's AI assistant plugin for VS Code. Gemini Code Assist Docs

3. AI Prompting Best-Practices

When asking AI to generate code, specificity is key. Use the R.T.R.O. framework:

- **Role:** Who is the AI? (e.g., "Expert React Developer", "Data Science Tutor")
- **Task:** What exactly do you want it to do? (e.g., "Write a function", "Debug this error")
- **Requirements (Context):** What constraints exist? (e.g., "Use Python 3.9", "No external libraries", "Explain it simply")
- **Output:** How should the answer look? (e.g., "Code block only", "Step-by-step list")

Example Prompt:

> *"Act as a **Front-End Engineering Instructor** (Role). **Write a CSS rule** to center a div horizontally and vertically (Task). You must use **Flexbox** and ensure it works on mobile screens (Requirements). Provide the **code snippet followed by a one-sentence explanation** (Output)."*

E. *[Optional] Exploration: Themes & Icons*

1. Download themes and icon packs of choice from the extension marketplace
2. Open Command Palette (Ctrl/Cmd + Shift + P).
3. Type "Preferences: Color Theme" and press Enter.
4. Use arrow keys to preview and select a color scheme.
5. Do the same for "Preferences: File Icon Theme"

5. Creating a Repository

Initializing a project the "Remote-First" way.

1. Log in to GitHub.com.
2. Select the **+** dropdown (top-right) and click **New repository**.
3. **Name:** Use a specific, short name (e.g., `mental-model-1`).
4. **Public/Private:** Select visibility (public for this course).
5. **Initialize with:**
    - **Add a README file:** Check this box.
    - **Add .gitignore:** Select a template (e.g., "macOS", "Node", or "Python"). This prevents uploading hidden system files.
6. Click **Create repository**.

✨ Try It Now: Create a "Sandbox" repository right now just to practice. You can delete it later!

6. Updating a Repository

The core cycle: Clone, Edit, Commit, Push.

A. Cloning (Remote to Local)

On your GitHub repo page, click the green **Code** button. Then you can use GitHub, VS Code, or both to download the repo to your device:

- **For GitHub Desktop:** Click "Open with GitHub Desktop." Choose the local file path for the repository folder.
- **For VS Code:** Two methods –
  - Open a new window, click “Clone Git Repository”, and follow the prompts, **OR**
  - Copy the HTTPS URL shown after clicking the Code button. In VS Code, open the Command Palette, type "Git: Clone", paste the URL, and select a folder.
- **For GitHub Desktop → VS Code:** follow the “Open with GitHub Desktop” prompts to create the local folder, then open this folder in VS Code.

B. The Edit Cycle

- **Using GitHub Desktop:**
  1. Edit files in your local folder using any text/file.
  2. Open GitHub Desktop and verify changes are listed.
  3. **Commit:** Type a summary and click **Commit to main**.
  4. **Push:** Click **Push origin** (top toolbar).
- **Using VS Code (Source Control):**
  1. **Open as Workspace:** Go to **File > Open Folder...** and select your repository folder. This ensures VS Code treats the folder as a distinct workspace with its own Git context.
  2. Make changes and save (or enable Auto Save).
  3. Click the **Source Control** icon (branch logo).
  4. **Stage:** Hover over the file name and click the **+** icon (or skip this to stage all at once).
  5. **Commit:** Type a message and click **Commit**.
  6. **Sync:** Click **Sync Changes** to Push to your remote repository.

✨ Try It Now: In your Sandbox repo, create a text file named hello.txt. Add the text "Hello World," save it, and push it to GitHub using either method above. Verify you can see it on the website!

7. Templates & Open Source

A. Using a Template

1. Navigate to a Template Repository URL.
2. Click **Use this template** > **Create a new repository.**
3. Name your new repo and click **Create.**
4. You now have an independent remote copy of the repo to Clone and edit.

B. Forking & Pull Requests (Open Source Workflow)

1. **Fork:** Click **Fork** on the original repo (on GitHub browser) to create a separate copy of the repo on your account (your “fork”).
2. **Clone:** Clone *your fork* (remote) to your computer (local).
3. **Edit & Push:** Make changes and push to *your fork* (your copy of the original repo).
4. **Pull Request (PR):**
    a. Go to the *Original Repository*,
    b. Click **Pull Requests**, then **New Pull Request**
    c. Select "Compare across forks" and choose your fork
    d. Write a description of your changes and submit!

C. Fork vs. Template:

A Fork maintains a connection to the original repo, allowing you to submit changes back. A Template creates a completely independent copy.

8. Exploration: GitHub Pages Deployment

Turn your repository into a live website! You can deploy using **GitHub Actions** (Modern/Recommended) or from a Branch (Classic). Ask Copilot in VS Code to help you with this if it seems intimidating!

Option A: Deploy via GitHub Actions (Recommended) This method is more robust and allows you to customize how your site is built.

1. Navigate to your repository on GitHub.
2. Click **Settings** (Gear icon in the top toolbar).
3. In the left sidebar, click **Pages**.
4. Under **Build and deployment** > **Source**, select **GitHub Actions** from the dropdown menu.
5. GitHub will analyze your repo and suggest a workflow. You can also ask Copilot in VS Code for help configuring your files.
    - For the class template (HTML/CSS), look for the **Static HTML** card and click **Configure**.
    - For custom React apps, you may need a specific Node.js workflow.
6. **Review the YAML:** An editor will open showing a `.yml` file. You typically do not need to edit this unless you have a complex site.
7. Click **Commit changes...** (green button, top right).
8. **Monitor the Build:** Click the **Actions** tab in your repository. You will see a workflow running (yellow circle). When it turns green, your site is live!

Option B: Deploy from Branch (Classic) Best for very simple, static HTML files without any build steps.

1. Go to **Settings > Pages**.
2. Under **Source**, select **Deploy from a branch**.
3. Under **Branch**, select `main` (or `master`) and ensure the folder is `/ (root)`.
4. Click **Save**.