Hyperiondev

Build Your Brand 05

Building Your Tech Portfolio

Visit our website

Take note!

Important - By **30 March 2024**, you should record an **invite to an interview** (if seeking employment) **or** a self-employment declaration (if seeking to be self-employed) at **www.hyperiondev.com/outcome5**. Additionally, you should record an offer of a **job** (if seeking employment) or evidence of obtaining **new work/contracts** (if self-employed or seeking to be) at **www.hyperiondev.com/finaloutcome5** by **23 September 2024**. Please record these milestones as soon as you reach them.

A **limited** number of co-certifications will be awarded based on your achievement of all **four criteria** for successful completion of this skills bootcamp.

Introduction

This is the fifth task in the series of "Build Your Brand" tasks. This task focuses on helping you to showcase your newly acquired development skills to peers, potential clients, and employers. In this task, you will push some of the code that you have written to GitHub. Your GitHub repository is a place where you can share your code to demonstrate your skills. This will become an important component of your developer portfolio.

BUILDING YOUR PROFESSIONAL BRAND

In a previous task, we talked about building your professional brand. Many tools help build a professional brand online, but to showcase your skills as a developer, few are more important than Github.

WHAT IS GITHUB?

Git is the foundation of many services that work on version control. The most popular and widely used of them all is GitHub. GitHub is an online Git repository hosting service. GitHub offers all of the functionality of Git and a lot more. While Git is a command-line tool, GitHub provides a web-based graphical interface. It offers access control and many features that assist with collaboration, such as wikis and basic task management tools for all projects.

GitHub is not just a project-hosting service, it is also a large social networking site for developers and programmers. Each user on GitHub has a profile, showing their past work and contributions that they have made to other projects. GitHub allows users to follow each other, subscribe to updates for projects, like them by giving them a star rating, etc.

Each project hosted on GitHub will have its own repository. Anyone can sign up for an account on GitHub and create their own repositories. They can then invite other GitHub users to collaborate on their project. You can even host websites for free directly from your repository!

GIT COMMANDS

You may already have encountered some of these key Git commands that you will be using in this task. If you feel comfortable with Git commands and what they do, you may skip ahead to how to sync a local Git Repository with a remote repository on Github.

INITIALISING A REPOSITORY

To create a new local repository, you have to initialise it using the **`init`** command. To do this, firstly open your terminal (or command prompt if you are using Windows) and go to your project's directory. To change your current directory, you use the **`cd`** (change directory) command followed by the pathname of the directory you wish to access.

After you have navigated to your project's directory, enter the following command:

```
git init
```

This creates a new, hidden subdirectory called .git in your project directory. This is where Git stores necessary repository files, such as its database and configuration information, so that you can track your project.

ADDING A NEW FILE TO THE REPOSITORY

Now that your repository has been initialised, you can add new files to your project using the `git add` command.

Assume that you have set up a project at `/Users/user/your_repository` and that you have created a new file called **newFile.py**. For the sake of this task, we will continue to refer to newFile.py, but if you'd like to use a .js file that includes JavaScript, you are welcome to do so. To add **newFile.py** to the repository staging area, you would need to enter the following into your terminal or command prompt:

```
cd /Users/user/your_repository
git add newFile.py
```

CHECKING THE STATUS OF YOUR FILES

Files can either exist in a tracked state or in an untracked state in your working directory. Tracked files are files that were in the last snapshot, while untracked files are any files in your working directory that were not in your last snapshot and are not currently in the staging area. We use the `git status` command to determine which files are in which state.

Using the `git add` command begins tracking a new file. If you run the `git status` command after you have added **newFile.py**, you should see the following code, showing that **newFile.py** is now tracked:

```
git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   newFile.py
```

You can tell that **newFile.py** is staged because it is under the "Changes to be committed" heading.

COMMITTING YOUR CHANGES

You should now be ready to commit your staged snapshot to the project history using the commit command. If you have edited any files and have not run `git add` on them, they will not go into the commit. To commit your changes, enter the following:

```
git commit -m "added new file newFile.py"
```

The message after the `-m` flag inside the quotation marks is known as a commit message. Every commit needs a meaningful commit message. This makes it easier for other people who might be working on the project (or even for yourself later on) to understand what modifications you have made. Your commit message should be short and descriptive and you should write one for every commit you make.

VIEWING THE CHANGE HISTORY

Git saves every commit that is ever made in the course of your project. To see your repository or change history over time, you need to use the `git log` command. Running the `git log` command shows you a list of changes in reverse chronological order, meaning that the most recent commit will be shown first. The `git log` command displays the commit hash (which is a long string of letters and numbers that serves as a unique ID for that particular commit), the author's name and email, the date written, and the commit message.

Below is an example of what you might see if you run git log:

```
git log
commit a9ca2c9f4e1e0061075aa47cbb97201a43b0f66f
Author: HyperionDev Student <hyperiondevstudent@gmail.com>
Date: Mon Sep 8 6:49:17 2017 +0200
Initial commit.
```

There are a large variety of options to the `git log` command that enable you to customise or filter what you would like to see. One extremely useful option is

`--pretty` which changes the format of the log output. The `oneline` option is one of the prebuilt options available for you to use in conjunction with `--pretty`. This option displays the commit hash and commit message on a single line. This is particularly useful if you have many commits.

Below is an example of what you might see if you run `git log --pretty=oneline`:

```
git log --pretty=oneline
A9ca2c9f4e1e0061075aa47cbb97201a43b0f66f  Initial commit.
```

For the full set of options, you can run `git help log` from your terminal or command prompt or take a look at the reference documentation.

SYNCING YOUR LOCAL AND REMOTE REPOSITORIES

Now that you understand how to create and add files to a local Git repository you should also learn how to **push** your repository from the command line to a remote GitHub repository. Have a look at this **excellent video tutorial** that explains how you can do exactly that.

Here is a summary of the command line prompts when you "`push an existing repository from the command line`":

```
git remote add origin https://github.com/[REPO-OWNER]/[REPO-NAME]
git branch -M main
git push -u origin main
```

Take note that if you would like added security through the use of Access Tokens please read the next optional section on Access Tokens and note the syntactic difference when syncing your repositories.

ACCESS TOKENS

In GitHub, an Access Token is an alternative to using passwords for authentication when using the GitHub API or, in our case, the command line. The purpose of these tokens are to provide added security and define which types of actions can be performed based on the "scope" of that token.

Take note that the use of Access Tokens is entirely optional. You are welcome to follow the steps below to generate an access token for your account and learn more about the username and password associated with your access token.

Generate an access token by following these steps:

1. Login to **GitHub**.
2. Click on the drop down arrow next to your profile picture and click on "**Settings**".
3. In your settings window, scroll down and click on "**Developer settings**"
4. Next, click on the "**Personal access token**" drop down selection.
5. Click on "**Tokens (classic)**".

   You'll notice there's also the option to select "**Fine Grained Tokens**".
   Both types of personal access tokens differ in terms of their capabilities and the level of access they grant to users.

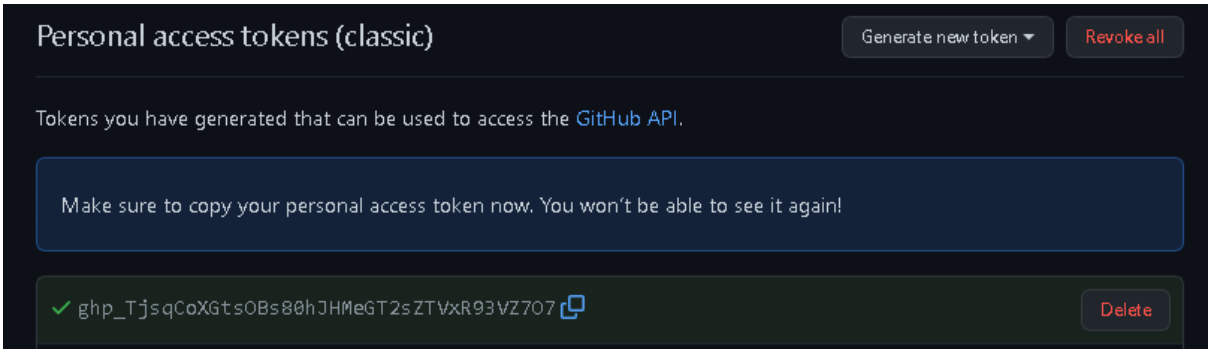
   Tokens (classic) provide full access to the account or organisation associated with the token, including the ability to read, write, and delete all types of resources, such as repositories, issues, and pull requests. They are designed for scripts and other automated processes that require full access to the account or organisation.
   Fine Grained Tokens, on the other hand, are a newer type of personal access token that provides more control over the level of access granted to users, enabling access to be granted to specific resources without giving full access to the entire account or organisation.

   Currently it's simpler and faster to use a Classic token which is why we will be selecting this option instead of a Fine Grained Token.

6. Now click on the drop down selection that says "**Generate new token**".
7. Click on "**Generate new token (classic)**".
8. GitHub might ask you for your password or to verify your login with GitHub mobile depending on the security measures you've enabled.
9. Confirm your access by entering your password or using your preferred method.
10. You should now be able to start creating an **Access Token**. You can use the "Note" section to call the token anything you'd like. The expiration will set how long the token will be valid. Please note that after your token expires you will no longer be able to use it and will need to generate a new one!

11. You will need to select all the applicable scopes for your token. For example, if you'd like full control of the repository using this access key, simply ensure you check the "*repo*" checkbox.
12. Finally scroll down and click on the "Generate token" button at the bottom. The full access token will be displayed on the next screen.

    Here is an example:



13. **Please ensure you copy your access token! If you intend to reuse it multiple times while it's active, ensure you store it in a secure location such as a credential store or password manager!**

    Once you have your token copied, ensure you paste it as your "**password**" when prompted to back on your original Git request.

Entering your username and password in the command prompt:

When it's time to push your local repository to your remote GitHub repository, you may occasionally be prompted for your **username** and **password** when performing specific actions. **Take note** that this is not always the case and that you may not be prompted to authenticate every time you push your repositories to a remote location.

Before moving ahead, the important thing to note here when pushing your repositories is that while your "**username**" in this context will need to match your actual "**username**" on GitHub, your "**password**" will need to be replaced with the **Access Token** that was generated.

There are two options available when entering your username and password / token credentials into the command prompt:

Option 1 - add the token directly into the URL:

```
git remote add origin https://[TOKEN]@github.com/[REPO-OWNER]/[REPO-NAME]
```

```
git branch -M main
git push -u origin main
```

Option 2 - enter the username and password separately:

```
git push https://github.com/[REPO-OWNER]/[REPO-NAME]
Username: <username>
Password: <token>
```

You can also review alternative methods for managing your access tokens in the **official GitHub documentation**.

Extra resource

Have a look at the Git cheat sheet in your Dropbox for more useful Git commands.

GITHUB AND YOUR DEVELOPER PORTFOLIO

As repeatedly stated, a **developer portfolio** (a collection of online programs that you have developed) allows you to demonstrate your skills rather than just telling people about them.

GitHub provides one of the most industry-recognised ways of sharing your code with others, including peers, prospective employers, or clients. A well-organised and documented GitHub repository can serve as a core component of a developer portfolio.

Even before seeing your work, prospective employers may also be impressed with the fact that you have experience in working with Git and Github.

README.MD FILES

When you add your code to GitHub, you can and should create README files. A README file is usually the first file that anyone interested in your code will look at. This file should describe your code. It should tell the reader what the project does, why the project is useful, who maintains and contributes to the project, and how a user can get your code to work.

A README file is essential for all software projects and learning to write clear, easy-to-read and detailed README files is an essential skill.

According to **this GitHub guide**, README files should contain the following:

- The project name.
- A clear, short, and to the point description of your project. Describe the importance of your project, and what it does.
- A table of contents to allow other people to quickly navigate especially long or detailed READMEs.
- An installation section which tells other users how to install your project locally.
- A usage section that instructs others on how to use your project after they've installed it. Include screenshots of your project in action. Keep in mind that if you add screenshots to your readme file, you will have to push those image files into your repository too.
- A section for credits which highlights and links to the authors of your project if the project has been created by more than one person.

README files have a `.md` extension, which stands for **m**ark**d**own. Markdown is a syntax that lets you style text. If you write text in a program like MS Word, you usually use the toolbar to select appropriate options to style your text (e.g. make certain text bold, underlined, or formatted in another way). When creating markdown files, you style your text using keywords and characters instead. For example, if you want to italicise text, you would surround the text with asterisks, like this: `In this paragraph *this text would be in italics*.`

Below is a summary of **Markdown syntax** taken from this **GitHub Guide**:

Headers	Headings are indicated with hash symbols that range from H1 to H6
	`# This is the biggest heading you get, used for Main Titles`
	`## This is a slightly smaller heading, used for Subtitles`
	`###### This is the smallest heading`

Emphasis	`*This text will be italics*`
	`_This will also be italics_`
	`**This text will be bold**`
	`__This will also be bold__`
	`_You **can** combine them_`

Unordered Lists	`* Item 1`
	`* Item 2`
	`  * Item 2a`
	`  * Item 2b`

Ordered Lists	`1. Item 1`
	`2. Item 2`
	`3. Item 3`
	`  a. Item 3a`
	`  b. Item 3b`

Images	`![alt text](image url)` Example: ![GitHub Logo](/images/logo.png)

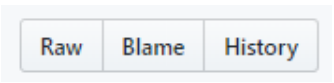
Links	`[link text](link url)`
	Example: [GitHub](http://github.com)
	OR use the shortcut syntax which doesn't specify the link text:
	`<link>`
	Example: <http://github.com> will automatically display as http://github.com

Blockquotes	`> This is a block quote. It can span over multiple lines.`
	`> It will display the quote with a grey line on the left-hand side.`
	Example:
	As Kanye West said:
	> We're living the future so
	> the present is our past.

A note from our coding mentor

Ridhaa

*To see an example of a README file, go **here**. Notice how the README file is rendered in the browser. Now click on "Raw" to see the Markdown for this file.*



*For more information about markdown, see the markdown cheatsheet (additional reading) provided by GitHub **here**.*

Take note!

A reminder of the important dates.

By **30 March 2024**, you should record an **invite to an interview** (if seeking employment) **or** a self-employment declaration (if seeking to be self-employed) at **www.hyperiondev.com/outcome5**. Please record this milestone as soon as you reach it. You can record any of the following:

- Joining an apprenticeship programme that utilises some of the knowledge obtained in your bootcamp.
- Obtaining a paid work opportunity for a duration of at least 12 weeks that utilises some of the knowledge obtained in your bootcamp.
- Obtaining a full-time job that utilises some of the knowledge obtained in your bootcamp and is with a new employer or your current employer (e.g. a promotion or extension of role).

Additionally, by **23 September 2024**, you should record an offer of a **job** (if seeking employment) or evidence of obtaining **new work/contracts** (if self-employed or seeking to be) at **www.hyperiondev.com/finaloutcome5**. Please record this milestone as soon as you reach it. You can record any of the following:

- An offer of a new job or apprenticeship that utilises part or all of the skills acquired in your bootcamp. You don't need to accept the offer to fulfil this requirement.
- New contracts or opportunities you obtained that utilise the new skills acquired through the bootcamp, which may include self-employed contract work or starting your own company.

Remember to keep an open mind and **explore various opportunities** to help you practise your interviewing skills and broaden your engagement with the tech sector. If you receive an invitation to a job or apprenticeship interview, please remember to fill out **www.hyperiondev/outcome5 before attending** the interview.

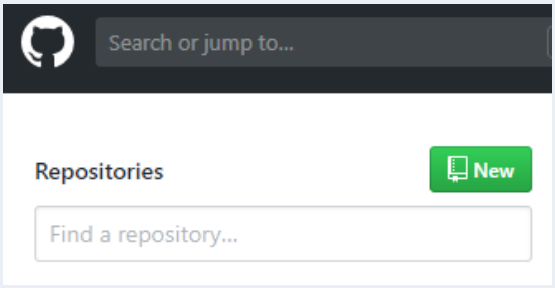
Co-certification: A **limited** number of co-certifications will be awarded based on your achievement of all **four criteria** for successful completion of this skills bootcamp.

Compulsory Task 1

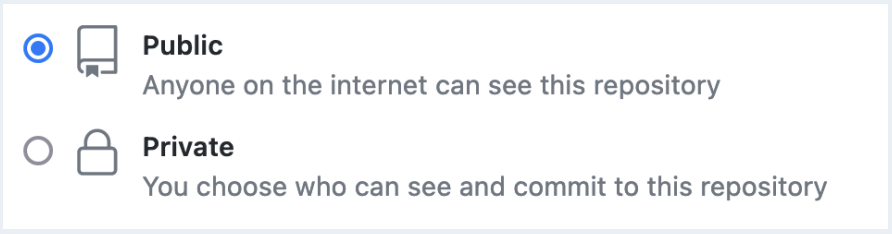
First, create a Google doc with a uniquely identifiable filename that includes your name and email address and a task identifier for this task (**BYB5**). For example, if your name was John Smith and your email address was john_smith@gmail.com, your filename would be **John Smith - john_smith@gmail.com - BYB5**. As you progress through this Compulsory Task you will fill your answers into this Google doc, which you will save as a PDF and upload to your Dropbox at the end.

Now follow these steps:

- Login to GitHub using the account you created in a previous task.
- Create a new repository by selecting the 'New' button as shown in the image below.



- Name the repository 'byb_project' and make sure that it is public.



- Next, create an empty folder called **byb_project** on your local machine.
- Open your terminal or command prompt and then change directory (`cd`) to your newly created folder.
- Enter the `git init` command to initialise your new repository.
- Enter the `git status` command and make a note of what you see. You should have a clean working directory.
- Create a new file in the **byb_project** folder called **helloWorld.py** or **helloWorld.js** (if you'd like to write in JavaScript) and write a program that prints out the message "Hello World!".

- Run the `git status` command again. You should now see that your **helloWorld.py** file is untracked.

- Enter the `git add` command followed by **helloWorld.py** to start tracking your new file.

- Once again, run the `git status` command. You should now see that your **helloWorld.py** file is tracked and staged to be committed

- Now that it is tracked, let us change the file **helloWorld.py**. Change the message printed out by the program to "Git is Awesome!"

- Run `git status` again. You should see that **helloWorld.py** appears under a section called "Changes not staged for commit". This means that the file is tracked but has been modified and not yet staged.

- To stage your file, simply run `git add helloWorld.py` again.

- If you run `git status` again you should see that it is once again staged for your next commit.

- You can now commit your changes by running the `git commit -m` command. Remember to enter a suitable commit message after the `-m` switch.

- Running the `git status` command should now show a clean working directory once again.

- Push the repository on your local machine to the remote repository on GitHub by following these steps:
    - Open your terminal or command prompt. Change directory (`cd`) into the **byb_project** folder you created.
    - **Add your remote repository** using the following command:

      `git remote add [remote-name] [url]`
      e.g. `git remote add origin https://github.com/HyperionDev/byb_project.git`.

      Now you can use the short name (e.g. **origin**) on the command line in lieu of the whole URL. The URL will be indicated under the heading shown below once you have created your repository on GitHub.

```
...or push an existing repository from the command line
```

- Push your local repository to your remote repository using the following command:
  `git push -u [remote-name] master`
  E.g. `git push -u origin master`

- Make the repository **public** and put a link to it into your Google answers doc (help for this is available **here**). Make it clear that the link is for Compulsory Task 1.
- Once a code reviewer has marked this task as complete (and not before!), you can delete the repository that you have created here since it doesn't store any meaningful application code. Help for this is available **here**.

Compulsory Task 2

Follow these steps:

- Create another Github repository. Entitle this: **finalCapstone**.

- Push the last Capstone Project that you created to this remote repository (and any previous ones you'd like to include).

- Add a detailed README file for each project that you have pushed to GitHub. A reminder that README files should contain the following:
  - The project name.
  - A clear, short, and to the point description of your project. Describe the importance of your project, and what it does.
  - A table of contents to allow other people to quickly navigate especially long or detailed READMEs.
  - An installation section that tells other users how to install your project locally.
  - A usage section that instructs others on how to use your project after they've installed it. Include screenshots of your project in action.
  - A section for credits that highlights and links to the authors of your project if the project has been created by more than one person.

- Make the repository public and put a link to it in your Google answer doc, making it clear that this time the link is for Compulsory Task 2.

Compulsory Task 3

Follow these steps:

- Create a new repository on your GitHub account with the repo name spelt **exactly** the same as your username.
- Create a README.md file inside this newly created repo.
- Use this README.md file to create a landing page for your GitHub account that is attractively styled by using the **GitHub Styling Guide**. This README page will automatically display on your main account page for future recruiters to see.
- Feel free also to add images. This might require research on the usage of the <img> or <picture> tag. You can read more about picture tags **here**.
- Add the link to your **GitHub page** into the "Featured" section of your Linkedin page. (Click on Featured, the "+" sign, then "Add a link", paste the GitHub account link, complete the details, and "Save"). Share a link to your LinkedIn profile in your Google doc.
- Add your GitHub link to your personal CV or résumé with your other relevant links.
- Take a screenshot of your change to your CV or résumé and paste it into your Google answers doc.

Useful Resource

- As mentioned in the previous Build Your Brand task, we highly recommend you look for a text called "Cracking the coding interview" by Gayle Laakmann McDowell, which offers valuable guidance and practise for tech interviews. This can be **purchased from Amazon**, but may also be available in your local public library or in other places online.

Compulsory Task 4

Follow these steps, using the updated version of your CV from Compulsory Task 3. As usual, the instructions below differ slightly for job seekers and self-employed students. Read them carefully and follow those that fit the category you are in.

JOB SEEKERS

Time to apply for some more roles! You can follow these steps, which should be familiar by now:

- Once again, consider the sort of roles you might like to apply for and the sort of companies you might want to work for. Find another 5 junior tech roles currently being advertised - **different from those you have previously applied for** - ensuring you consider **a range** of employers - large (1000+ employees) and small (100-500 employees) companies as well as tech scale-ups (up to 100 employees), paid internships and apprenticeships, or companies that source, hire, and train (SHT is the keyword often used in job ads - these companies hire talent with the intention of providing extra training to ensure the employee can hit the ground running).

  Remember that we specifically request that you search for and apply to a broad range of organisations. We find this to be best practice for a number of reasons: it pushes students out of the comfort zone of just applying to work for the major brands they're already familiar with; it helps students to broaden their knowledge of which companies form part of the tech hiring landscape, particularly small to medium enterprises (SMEs); and, it ensures that we meet the DfE requirement to address the needs of the broader, especially SME, tech market. Keep in mind that our team will review your submitted data and create relationships with your identified companies wherever possible. We already have relationships with many larger tech companies, so focussing on companies of different sizes and profiles will also help to diversify your application profile and increase your chances of success.
- **For each of the roles** you have identified, again identify the recruiting or hiring managers (possibly the Talent or HR people at each company) and their contact details if possible (**email addresses for at least 2**) and **LinkedIn profile URLs** (**for all**, mandatory).
- **Create and submit applications** to these 5 opportunities. Also, **reach out** to the contacts you have identified in whatever way possible (email, LinkedIn direct message, phone call, etc.), drawing their attention to your application to their company's role.
- Remember to track your applications in the **Job Interviews/Pitch Opportunities Tracker** we provided a template for earlier in this task, and add a link to this into your Google answers doc.

SELF-EMPLOYED STUDENTS

- Once again, consider companies and business contacts that you would pitch your services or products to. Note the contact and other details of 5 such companies and contacts.

- **Create and submit pitches** to these 5 opportunities. Remember to track your pitches in the **Job Interviews/Pitch Opportunities Tracker** we provided a template for earlier in this task, and add a link to this into your Google answers doc.

ALL STUDENTS

- Fill in the details of the contact people you have identified into **this form**. Take screenshots of the form showing all the data you have filled in, and paste them into your Google answers doc so that the mentor marking your work can see that you have completed this part of the task.
- Ensure your Google doc contains the answers to Compulsory Tasks 1, 2, 3, and 4, and then save your document as a PDF (using menu options File -> Download -> PDF) and upload the PDF file to your Dropbox.

- **Remember, if you secure an interview, it is absolutely vital that you notify us** via **hyperiondev.com/outcome5**.

Rate us

Share your thoughts

HyperionDev strives to provide internationally excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.