

Full Stack Development Lab Instructions  
ENSF381: Lab01

Getting Started with Web Development

Created by: Mahdi Jaberzadeh Ansari (He/Him)  
Schulich School of Engineering  
University of Calgary  
Calgary, Canada  
mahdi.ansari1@ucalgary.ca

Week 2, January 15/17, 2024

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Forming groups . . . . .	1
1.2	Before submission . . . . .	1
1.3	Academic Misconduct/Plagiarism . . . . .	2
1.4	Marking Scheme . . . . .	2
1.5	Complains . . . . .	2
<b>2</b>	<b>Exercise A (Preparing Your Web Development Environment)</b>	<b>4</b>
2.1	Objectives . . . . .	4
2.2	Prerequisites . . . . .	4
2.3	Part 1: Installing Visual Studio Code . . . . .	4
2.4	Part 2: Installing Essential Extensions . . . . .	4
2.5	Part 3: Configuring Your Environment . . . . .	5
2.6	Part 4: Creating Your First Project . . . . .	6
2.7	Questions . . . . .	7
2.8	Conclusion . . . . .	7
<b>3</b>	<b>Exercise B (Web Functionality Explanation)</b>	<b>7</b>
<b>4</b>	<b>Exercise C (Web Basics)</b>	<b>7</b>
<b>5</b>	<b>Exercise D (Git and GitHub)</b>	<b>7</b>
5.1	Introduction to Git . . . . .	7
5.2	Getting Started with Git . . . . .	8
5.2.1	Download and Install Git . . . . .	8
5.2.2	Set Up Git . . . . .	8
5.2.3	Configuring Git . . . . .	8
5.2.4	Checking Your Configuration . . . . .	8
5.2.5	Scope of Configuration . . . . .	8
5.2.6	Why It's Important . . . . .	8
5.3	Understanding Git Concepts . . . . .	9
5.3.1	Git Snapshots . . . . .	9
5.3.2	Local vs. Remote Repositories . . . . .	9
5.3.3	Branches in Git . . . . .	10
5.3.4	Transition from 'master' to 'main' . . . . .	10
5.4	Using Git for a Project . . . . .	10
5.4.1	Creating a New Repository . . . . .	10
5.4.2	Making Changes . . . . .	10
5.4.3	Branching with Git . . . . .	10
5.5	Collaborating with GitHub . . . . .	10
5.6	Resources for Further Learning . . . . .	12
5.7	Basic Git Commands . . . . .	12
5.8	Questions . . . . .	12
5.9	Conclusion . . . . .	12

# 1 Introduction

## 1.1 Forming groups

- In this lab session, you can't work with a partner.

## 1.2 Before submission

- For most of the labs, you will receive a DOCX file that you need to fill out the gaps. Make sure you have this file to fill out.
- All your work should be submitted as a single file in PDF format. For instructions about how to provide your lab reports, study the posted document on the D2L called [How to Hand in Your Lab assignment](#).
- Please note that if it is group work, only one team member must submit the solution in D2L. For ease of transferring your marks, please mention the group member's name and UCID in the description window of the submission form.
- If you have been asked to write code (HTML, CSS, JS, etc.), make sure the following information appears at the top of your code:
  - File Name
  - Assignment and exercise number
  - Your names in alphabetic order
  - Submission Date:

Here is an example for CSS and JS files:

```
/*
=====
Name       : lab2_exe_A.css
Assignment : Lab 2 Exercise A
Author(s)  : Mahdi Ansari, William Arthur Philip Louis
Submission : May 21, 2030
Description : A CSS file for decorating X form
=====
*/
```

- Some exercises in this lab and future labs will not be marked. Please do not skip them, because these exercises are as important as the others in learning the course material.
- In courses like ENSF381, some students skip directly to the exercises that involve writing code, skipping sections such as “Read This First,” or postponing the diagram-drawing until later. That's a bad idea for several reasons:
  - “Read This First” sections normally explain some technical or syntax details that may help you solve the problem or may provide you with some hints.
  - Some lab exercises may ask you to draw a diagram, and most of the students prefer to hand-draw them. In these cases, you need to scan your diagram with a scanner or an appropriate device, such as your mobile phone, and insert the scanned picture of your diagram into your PDF file (the lab report). A possible mobile app to scan your documents is Microsoft Lens, which you can install on your mobile device for free. Please make sure your diagram is clear and readable; otherwise, you may either lose marks or it could be impossible for TAs to mark it at all.
  - Also, it is better to use the [Draw.io](#) tool if you need to draw any diagram.

- Drawing diagrams is an important part of learning how to visualize data transfer between modules and so on. If you do diagram-drawing exercises at the last minute, you won't learn the material very well. If you do the diagrams first, you may find it easier to understand the code-writing exercises, so you may be able to finish them more quickly.

- **Due Dates:**

- You must submit your solution until 11:59 p.m. on the same day that you have the lab session.
- Submissions until 24 hours after the due date get a maximum of 50% of the mark, and after 24 hours, they will not be evaluated and get 0.

### 1.3 Academic Misconduct/Plagiarism

- Ask for help, but don't copy.
  - You can get help from lab instructor(s), TAs, or even your classmates as long as you do not copy other people's work.
  - If we realize that even a small portion of your work is a copy from a classmate, both parties (the donor and the receiver of the work) will be subject to academic misconduct (plagiarism). More importantly, if you give exercise solutions to a friend, you are not doing him or her a favor, as he or she will never learn that topic and will pay off for this mistake during the exam or quiz. So, please do not put yourself and your friend in a position of academic misconduct.
  - You can use ChatGPT, but please note that it may provide similar answers for others too, or even the wrong answers. For example, it has been shown that AI can hallucinate, proposing the use of libraries that do not actually exist <sup>1</sup>. So, we recommend that you imagine ChatGPT as an advanced search engine, not a solution provider.
  - In order to find out who is abusing these kinds of tools, we will eventually push you toward the incorrect responses that ChatGPT might produce. In that case, you might have failed for the final mark and be reported to administration.
  - If we ask you to investigate something, don't forget to mention the source of your information. Reporting without reference can lead to a zero mark even by providing a correct answer.

### 1.4 Marking Scheme

- You should not submit anything for the exercises that are not marked.
- In Table 1, you can find the marking scheme for each exercise.

Table 1: Marking scheme

Exercise	Marks
A	5 marks
B	10 marks
C	10 marks
D	25 marks
Total	50 marks

### 1.5 Complains

- Your grades will be posted one week following the submission date, which means they will be accessible at the subsequent lab meeting.

---

<sup>1</sup><https://perma.cc/UQS5-3BBP>

- Normally, the grades for individual labs are assessed by a distinct TA for each lab and section. Kindly refrain from contacting all TAs. If you have any concerns regarding your grades, please direct an email to the TA responsible for that specific lab.

## 2 Exercise A (Preparing Your Web Development Environment)

### 2.1 Objectives

In this lab session, you will set up your web development environment by installing **Visual Studio Code (VS Code)** and the necessary extensions. This setup will equip you with the tools needed for efficient web development in future lab sessions and assignments.

### 2.2 Prerequisites

1. A computer with internet access.
2. Basic understanding of working with file systems and operating systems.

### 2.3 Part 1: Installing Visual Studio Code

#### 1. Download VS Code:

- Go to the [Visual Studio Code website](#).
- Click on the download link appropriate for your operating system ([Windows](#), [Mac OS](#), [Linux Debian/Ubuntu](#) or [Linux Fedora/Red Hat/SUSE](#)).

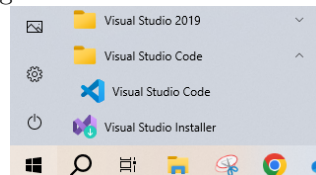
#### 2. Install VS Code:

- Execute the downloaded file and follow the installation prompts.
- Accept the license agreement and choose your desired installation settings.

#### 3. Launch VS Code:

- Once installed, open Visual Studio Code. For example, in Windows OS, you must click on the Visual Studio Code menu in the start menu, as shown in Figure 1.

Figure 1: VSC in the start menu



- After a successful run, you must be able to see the VSC editor, as shown in Figure 2.
- Most of Gen Z like dark themes; however, if you don't like them, you can change them by going to *File > Preferences > Theme > Color Theme*. When you press on the 'Color Theme' menu item, a window will open, and you can select a light theme.

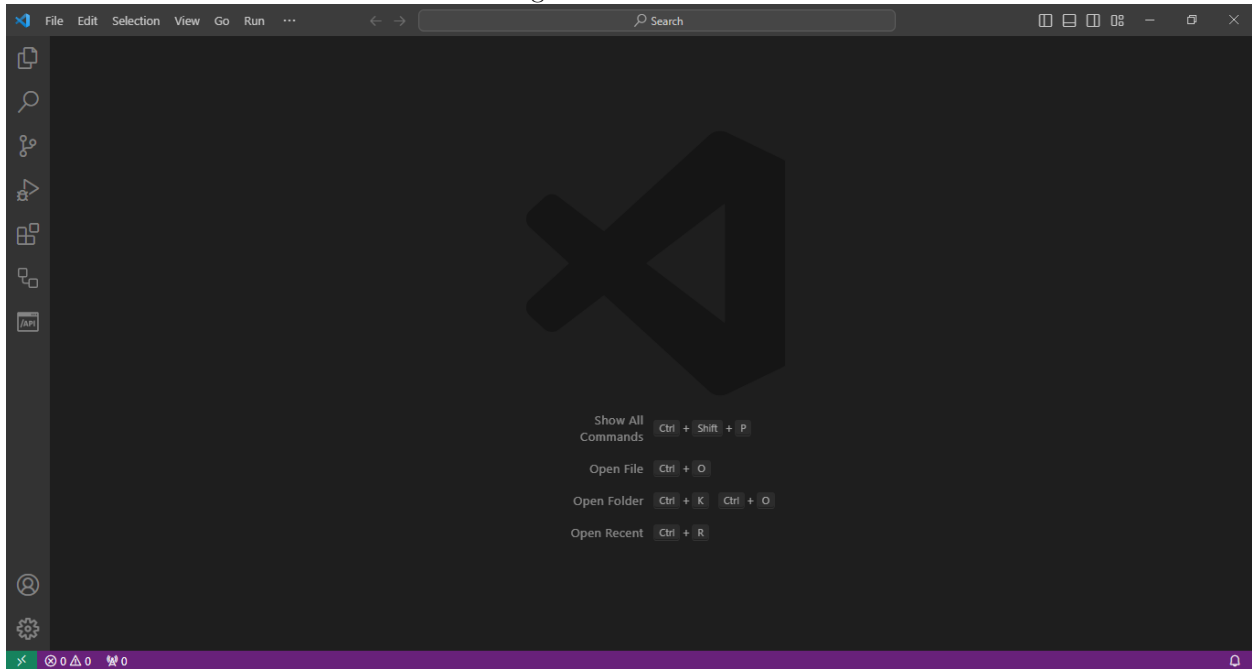
### 2.4 Part 2: Installing Essential Extensions

#### 1. Accessing the Extension Marketplace:

- In VS Code, click on the Extensions icon or press *Ctrl+Shift+X* in the Activity Bar on the side of the window.
- This opens the Extensions view, where you can search for extensions.

#### 2. Install Live Server:

Figure 2: VSC editor



- In the Extensions view, search for “Live Server.”
- Click on the install button next to the Live Server extension which has been made by Ritwick Dey.
- This extension allows you to view your webpages live as you develop them.

### 3. Install Prettier - Code Formatter:

- Search for “Prettier - Code Formatter” in the Extensions view.
- Click install. Prettier helps in automatically formatting your code for better readability.

### 4. Install ESLint:

- Search for “ESLint.”
- Click install. ESLint helps in identifying and reporting on patterns found in ECMAScript/-JavaScript code, which is useful for maintaining code quality.

### 5. (Optional) Install Other Useful Extensions:

- You may also consider other extensions like **Bracket Pair Colorizer**, **GitLens**, or language-specific extensions depending on your development needs. We might also ask you to install more extensions in the future. For example, you can install a bunch of extensions by installing **Web Dev Extensions**; however, please note that unnecessary extensions might lead to a slower editor.

## 2.5 Part 3: Configuring Your Environment

### 1. Setting Up Prettier:

- Go to *File > Preferences > Settings*.
- Search for “Prettier” and customize the settings as per your requirements, or leave the settings as they are.

- For example, it is recommended to tick “indent lines with tabs”; it causes Prettier to use tabs instead of spaces.

## 2. Configuring ESLint:

- Go to the settings and search for “ESLint.”
- Adjust the ESLint settings to fit your coding standards and rules, or leave the settings as they are.

## 2.6 Part 4: Creating Your First Project

### 1. Create a New Folder:

- Create a new folder on your computer where you will store your web development projects.

### 2. Open the Folder in VS Code:

- In VS Code, go to *File > Open Folder* and select your newly created folder.
- It is better to have an “ENSF381” folder inside your home address and add some folders that reflect the lab and exercise. For example, I made “C:\Users\mahdi\ENSF381\Lab1\EXEA.2.6.3”.

### 3. Create Your First HTML File:

- Right-click in the Explorer pane in VS Code and choose New File.
- Name it `index.html` and write some basic HTML code as follows:

```

<!--
=====
Name      : index.html
Assignment : Lab 1 Exercise A
Author(s)  : Mahdi Ansari, William Arthur Philip Louis
Submission : May 21, 2030
Description : A simple HTML file.
=====
-->
<!DOCTYPE html>
<html>
<head>
    <title>My Simple HTML Page</title>
</head>
<body>
    <h1>Welcome to My Webpage</h1>
    <p>This is a simple HTML document.</p>
</body>
</html>

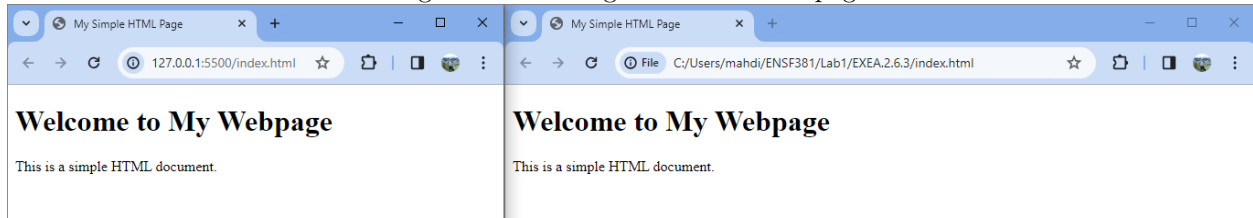
```

### 4. Testing Live Server:

- Right-click on your HTML file in the Explorer pane and select “Open with Live Server” to see your webpage in action.
- The result must be shown in your default web browser, as shown in the left part of Figure 3.
- Right-click once more on the HTML file and select “Reveal in File Explorer”; it must show a file explorer with the address where the “index.html” is placed. Now right-click on the file inside the file explorer and open the file with a browser.
- The result must be shown in your default web browser, as shown in the right part of Figure 3.



Figure 3: Browsing the first HTML page



## 2.7 Questions

1. Why is the content of the address bar in the two browsing methods different? Write your answer in the designated space on the answer file.

## 2.8 Conclusion

**Congratulations!** You have successfully set up your web development environment with Visual Studio Code and essential extensions. You are now ready to start building web applications. Remember to explore additional extensions and settings in VS Code to further enhance your development experience.

## 3 Exercise B (Web Functionality Explanation)

1. *URL vs. URI:*
  - (a) Research and write a comparison between a URL and a URI, highlighting their unique purposes and features.
  - (b) Name each part of this URL: **`https://auth.example.com/site1?name=Alex&num=123`**
2. *IP Address:*
  - (a) What is the relationship between IP addresses and website domains?
  - (b) What is the IP address of [ucalgary.ca](https://ucalgary.ca)?
  - (c) How many domains exist on the IP address that [ucalgary.ca](https://ucalgary.ca) is served?

## 4 Exercise C (Web Basics)

1. *HTML vs. HTTP:* Write a paragraph explaining the differences between HTML and HTTP.
2. *Web Browsers Overview:* Use 'most popular browsers' keywords in Google and report the current browser market shares. Include at least the ten most popular web browsers. Different sites might report different numbers as they have different resources but in general they report the same order of browsers in most cases. This is one of the questions that you must provide references for your answer.

## 5 Exercise D (Git and GitHub)

### 5.1 Introduction to Git

**Read This First:** Git is a distributed version control system that allows you to track changes in your code over time. Unlike centralized version control systems, Git gives every developer a local copy of the entire development history, and changes are copied from one such repository to another. These data models allow Git to manage a project's history as a series of snapshots.

## 5.2 Getting Started with Git

### 5.2.1 Download and Install Git

- Visit the [official Git website](#) and download the version for your operating system.
- Follow the installation instructions in the installer window.

### 5.2.2 Set Up Git

- Open your terminal or command prompt.
- Configure your user name and email with the ‘git config’ command if you know how, and jump to section 5.7 or follow the following sub section.

### 5.2.3 Configuring Git

Open your CLI and follow these steps:

#### 1. Set Your Username:

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

Replace "Your Name" with your actual name. This name will be attached to your commits and tags.

#### 2. Set Your Email Address:

```
git config --global user.email "your_email@example.com"
```

Replace "your\_email@example.com" with your email address. Use the same email address associated with your GitHub, GitLab, Bitbucket, or other Git hosting service accounts if you plan to push repositories to these services.

### 5.2.4 Checking Your Configuration

- To check if your configuration was successful, use:

```
git config --list
```

- This command will list all Git configurations, and you should see your `user.name` and `user.email` settings listed.

### 5.2.5 Scope of Configuration

- The `--global` flag in the commands sets your username and email for all repositories on your system under your user account.
- If you need to set a different username or email for a specific repository, navigate to that repository's directory in your CLI and run the same commands without the `--global` flag. This sets the configuration only for that particular repository.

### 5.2.6 Why It's Important

- These configurations are critical because every Git commit uses this information, and it's important for accountability and collaboration in team environments.
- Commits you make to repositories hosted on services like GitHub, GitLab, or Bitbucket, will show up under your name and email as specified in these configurations.

Remember, if you're contributing to open-source projects or using public repositories, the username and email you configure here will be publicly visible on those platforms.

## 5.3 Understanding Git Concepts

### 5.3.1 Git Snapshots

Git creates ‘snapshots’ of the repository at each commit. Unlike other systems that track differences in files, Git records the entire content of all files.

Figure 4: Subversion

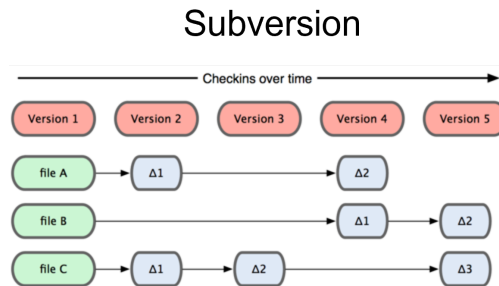
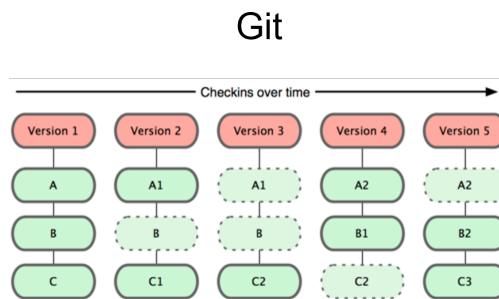


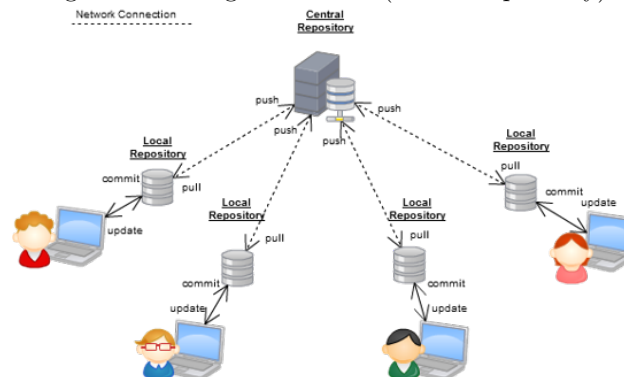
Figure 5: Git



### 5.3.2 Local vs. Remote Repositories

Local repositories are on your machine, allowing you to work offline. Remote repositories are hosted on servers like GitHub for collaboration. The schema for local and remote repositories has been shown in figure 6.

Figure 6: Having remote Git (central repository)



### 5.3.3 Branches in Git

Branches are used to develop features isolated from each other. The ‘main’ branch is the default branch in most cases.

### 5.3.4 Transition from ‘master’ to ‘main’

The Git community has moved from using ‘master’ to ‘main’ as the default branch name for new repositories, reflecting a more inclusive language.

## 5.4 Using Git for a Project

### 5.4.1 Creating a New Repository

- Open a terminal or a command line.
- Navigate into the directory that you made in section 2.6 , and run ‘git init’.

### 5.4.2 Making Changes

- Make some changes in your ‘index.html’ file, use ‘git add’ to stage it, and use ‘git commit’ to commit.

### 5.4.3 Branching with Git

- Create, switch, and merge branches with ‘git branch’, ‘git checkout’, and ‘git merge’.

## 5.5 Collaborating with GitHub


- Set up a GitHub account and create an empty remote repository, like what I did in Figure 7.

Figure 7: Repository creation in GitHub

Search or jump to... Pull requests Issues Codespaces Marketplace Explore

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner  Repository name

Great repository names are short and memorable. Need inspiration? How about [verbose-waddle?](#)

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template:

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

License:

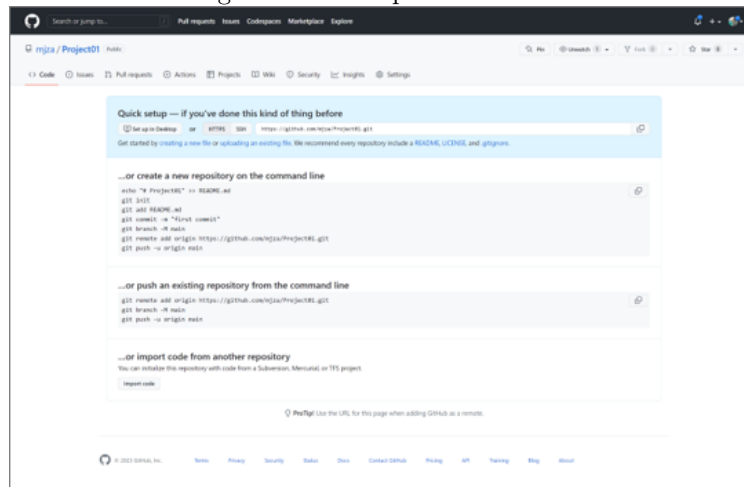
① You are creating a public repository in your personal account.

[Create repository](#)

© 2023 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact GitHub](#) [Pricing](#) [API](#) [Training](#) [Blog](#) [About](#)

- Figure 8 shows a sample repository before pushing to it. In such a situation, it shows the commands that you need to use for pushing and the URL of your brand-new repository.

Figure 8: Before push in GitHub



- Set the current **local** branch to main (in terminal).

```
git branch -M main
```

- Add the remote repository.

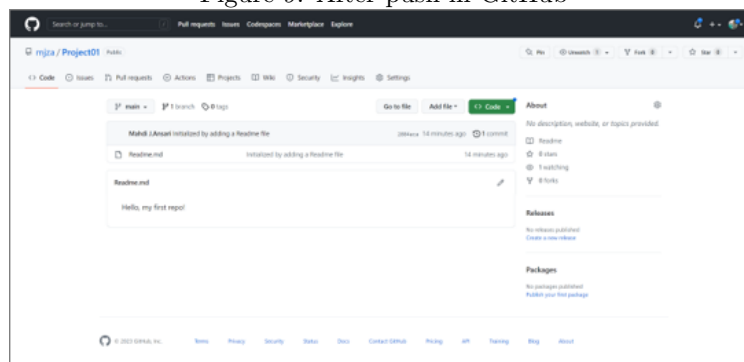
```
git remote add origin https://url...
```

- Push the latest changes to the remote repository.

```
git push -u origin main
```

- Figure 9 shows a sample repository after pushing to it. If you refresh your repository after pushing, you must be able to see the 'index.html' file in it.

Figure 9: After push in GitHub



## 5.6 Resources for Further Learning

- Refer to the Pro Git Book, GitHub Learning Lab.
- Also, [Learn Git Branching](#) is a good tutorial for learning Git.

## 5.7 Basic Git Commands

You need to learn these commands by heart.

- **git init:** Initialize a new Git repository.
- **git add [file]:** Add files to the staging area.
- **git commit:** Commit your changes.
- **git clone [url]:** Clone a repository.
- **git status:** Check the status of your repository.
- **git push:** Push changes to a remote repository.
- **git pull:** Update your local repository.

## 5.8 Questions

1. Make a screenshot from the GitHub repository that you created. It must clearly show the URL of your repository.
2. Copy and paste the URL of your repository in the answer file, and make sure your repository is public.

## 5.9 Conclusion

**Congratulations!** You have finished the first lab. Don't forget to save your DOCX file as a PDF file and only submit the PDF file.