Full Stack Development Lab Instructions
ENSF381: Lab04

Responsive Layout and Flex

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

Week 5, February 5/7, 2024

Contents

1 Introduction

1.1 Objectives

Lab 4 focuses on several key aspects of web development. You will learn to manage Git repositories directly in your code editor, understand responsive layout, and learn more advanced CSS techniques like Flex.

1.2 Prerequisites

1. Basic understanding of computer operations.
2. A text editor (such as Notepad++, Visual Studio Code, or Sublime Text).
3. A web browser (Chrome, Firefox, Safari, etc.) to view your HTML file.
4. GitHub account and an installed Git client.

1.3 Forming groups

- In this lab session, you **MUST** work with a partner (groups of three or more are not allowed).
- The main goal of working in a group is to learn:
  - how to do teamwork
  - how to not be a single player
  - how to not be bossing
  - how to play for team
  - how to tolerate others
  - how to behave with colleagues
  - how to form a winner team
  - and ...
- Working with a partner usually gives you the opportunity to discuss some of the details of the topic and learn from each other. Also, it will give you the opportunity to practice one of the popular methods of program development called pair programming. In this method, which is normally associated with the "Agile Software Development" technique, two programmers normally work together on the same workstation (you may consider a Zoom session for this purpose). While one partner, the driver, writes the code, the other partner, acting as an observer, looks over his or her shoulder, making sure the syntax and solution logic are correct. Partners should switch roles frequently in such a way that both have equivalent opportunities to practice both roles. **Please note that you MUST switch roles for each exercise.**
- When you have to work with a partner:
  - Choose a partner that can either increase your knowledge or transfer your knowledge. (i.e., do not find a person with the same programming skill level!)
  - Please submit only one lab report with both names. Submitting two lab reports with the same content will be considered copying and plagiarism.

1.4 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         : lab4_exe_C.css
Assignment   : Lab 4 Exercise C
Author(s)    : Mahdi Ansari, William Arthur Philip Louis
Submission   : May 21, 2030
Description  : Responsive layout.
=============================================================
*/
```

- 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.

1.5 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 [1]. 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.

1.6 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	10 marks
B	15 marks
C	15 marks
D	10 marks
Total	50 marks

1.7 Complains

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

[1] https://perma.cc/UQS5-3BBP

2 Exercise A (Creating a repository)

2.1 Initialize the Project

Like last time, we are going to clone an existing project from a remote repository. Cloning is particularly useful when you want to create a local copy of a remote repository to work on. Follow these steps:

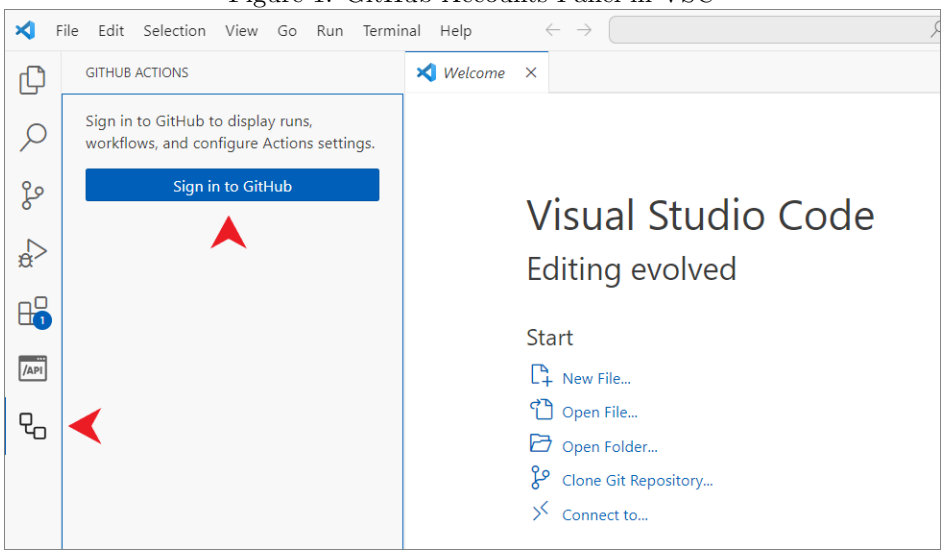
- Create a new repository on GitHub named `Lab4`. Remember to:
  - Check the box for `Add a README file`. This file serves as an introduction and overview of your project.
  - Select `MIT License` for your project's license. This is a permissive license that allows others considerable freedom with your code while still crediting you.
- After creating the repository, it should contain two files: a `README.md` and a `LICENSE` file. The README.md is where you can write about your project, and the LICENSE file contains the license text.

2.2 Using Visual Studio Code for Git Operations

This section focuses on using Visual Studio Code (VSC), a powerful editor that includes native Git integration, as an alternative to command-line Git operations. This provides an intuitive and user-friendly way to manage repositories on Windows, MacOS, and Linux.

1. **Configuring Git in Visual Studio Code:**
   - In VS Code, the GitHub integration is provided by the GitHub Pull Requests and Issues extension. You need to install Git (which you have done in Lab 1) and this extension if your VSC does not already have it.
   - In VSC go to the `GitHub Accounts Panel`, as shown in Figure 1, which is accessible from the `Activity Bar` on the side of the window.

Figure 1: GitHub Accounts Panel in VSC

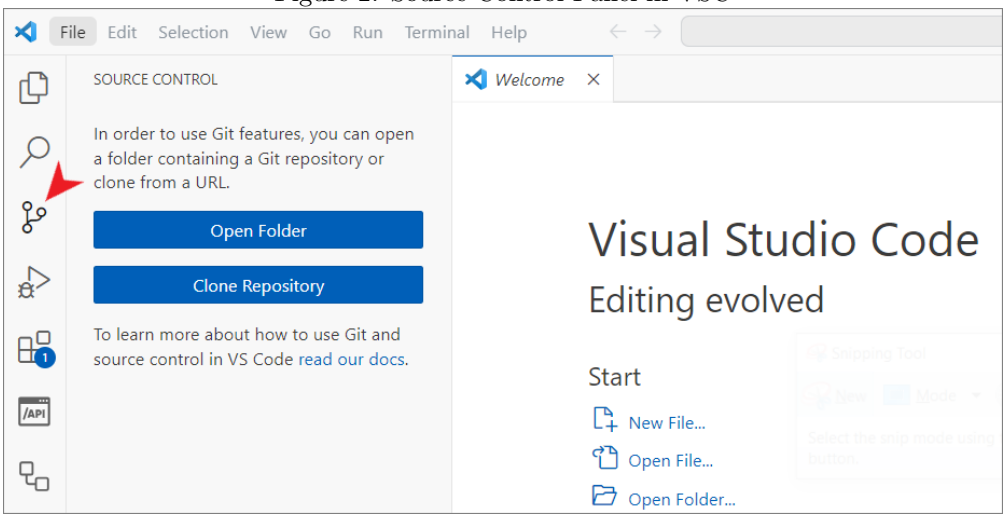


- Then, you will need to sign in to your GitHub account. Follow the prompts to authenticate with GitHub in the browser and return to VS Code.
- If you need to learn more about this extension, check out here then.

2. **Cloning a Repository using Visual Studio Code:**

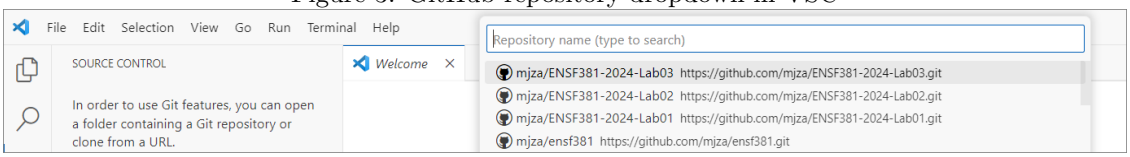
   - In VSC, open the `Command Palette` (Ctrl+Shift+P or Cmd+Shift+P) and type 'Git: Clone'. Then enter the URL of your GitHub repository. Or you can easily use the `Clone Repository` button in the`Source Control Panel` (Ctrl+Shift+G or Cmd+Shift+G), as shown in Figure 2.

Figure 2: Source Control Panel in VSC



   - From the GitHub repository dropdown, you can filter and pick the repository you have recently created for `Lab4` to clone locally.

Figure 3: GitHub repository dropdown in VSC



   - Select a local path for where to clone the repository, and VSC will begin the cloning process. It is better that you select the address ~`/ENSF381/Lab4` as the destination on your local machine.
   - Once the cloning is complete, you can open the folder of the cloned repository in VSC and start working with your files.

3. **Pushing Changes to GitHub:**

   - After making changes to your files, go to the Source Control panel in VSC.
   - Stage your changes by clicking on the '`+`' icon next to the changed files. Write a commit message, and then commit the changes using the checkmark icon (✓) or the `Commit` button.
   - To push the committed changes to GitHub, click on the '`...`' menu in the `Source Control Panel` and select `Push`.
   - VSC will handle the process of pushing your changes to the remote repository on GitHub.

2.3 Copy/Paste initial materials

1. Download the lab content from D2L and unzip the materials.

2. Copy and paste materials related to this lab into the `Lab4` folder. Please note that you must only copy and paste web-related materials (e.g., *.html or *.css file; **no *.docx file**). Also, if there is an `images` folder, copy and paste it with all its contents.

3. Commit and push the initial contents to GitHub, using VSC with this comment: 'Add initial contents'.

2.4 Inviting Collaborators to Your GitHub Repository

Collaborating with others on a project is a key aspect of software development. GitHub allows you to add collaborators to your repository. Here's how you can invite your colleagues to the repository you created in the previous steps:

- First, log in to your GitHub account and navigate to the repository you want to share. In this case, it's the `Lab4` repository.
- On your repository's page, locate the `Settings` tab near the top of the page and click on it. This will take you to the repository settings.
- In the settings menu, on the left side of the page, you will find a section called `Manage access`. Click on it.
- On the `Manage access` page, you will see a button labeled `Invite a collaborator`. Click on this button.
- GitHub will prompt you to enter the GitHub username or email address of the person you want to invite. Enter the username or email of your colleague.
- Once you enter the username or email, GitHub will suggest the matching account. Click on the account to select it.
- After selecting the collaborator, you can set the role for them. For most cases, the `Write` access level is sufficient. This allows them to push changes to the repository but doesn't allow actions like deleting the repository.
- Click on `Add collaborator` to send the invitation. Your colleague will receive an email from GitHub with a link to accept the invitation.
- Inform your collaborator to check their email and accept the invitation to collaborate on your repository.

Once your colleague accepts the invitation, they will have access to the repository and can start collaborating with you on the project. This process is crucial for teamwork in software development, allowing multiple people to work together seamlessly on the same codebase. Ask your colleges to clone the repository on their local machines with SourceTree.

Please note: *In the subsequent sections, each exercise should be completed by a single individual. It is imperative to rotate the role of the developer. This means that if Person X performs the steps in Exercise B, then Exercise C must be undertaken by Person Y, and this pattern should continue accordingly. We will check the commits on GitHub to give you marks.*

2.5 Deliverable

1. Add the address of your GitHub repository to your answer sheet file. Please make sure the repository is public and do not delete it until the end of the semester.

2. Click on the `...` menu in the `Source Control Panel` and select `Source Control Repositories` from the `View`. Make a screenshot from the list of repositories and add it to your answer sheet file.

3 Exercise B (Introduction to Responsive Web Design Exercise)

3.1 Introduction

In this exercise, you will be working with a basic HTML structure that includes various elements such as a header, footer, main content area, and news items. Your task is to write CSS to make a webpage layout responsive. Responsive web design is crucial in today's web development as it ensures that web content looks good and is usable on all devices, from desktops to tablets and smartphones.

3.2 Lab Activities

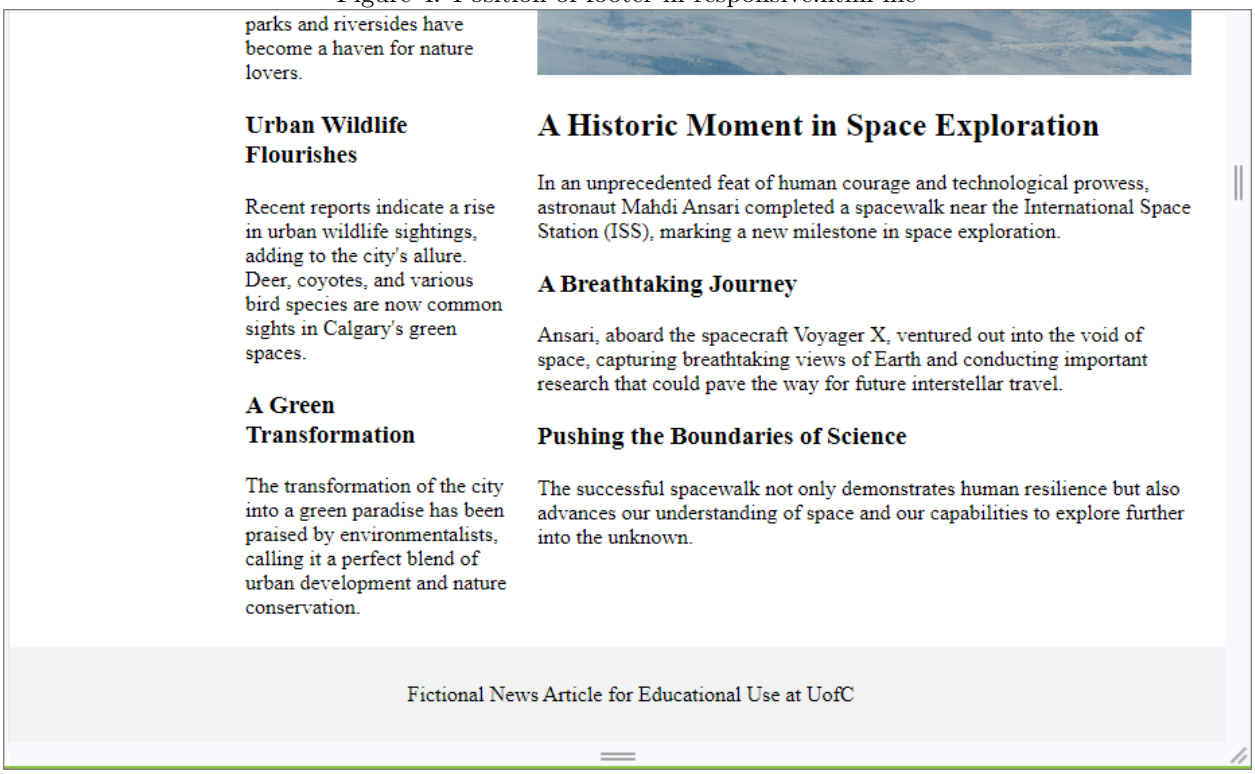
1. Before starting to code, please read this section entirely.
2. Open the ~/`ENSF381/Lab4` folder in Visual Studio Code (VSC).
3. Run the `responsive.html` file using "Open with Live Server". The HTML provided is structured with div elements representing different sections of a webpage. You will find classes like *.container*, *.header*, *.footer*, *.main*, *.news-item*, *.row*, *.large*, *.medium*, and *.small*. Each of these classes is meant to style corresponding sections of the webpage. The div elements with *.row* class contain news items of varying importance, denoted by *.large*, *.medium*, and *.small* classes.
4. **For this exercise, you are not allowed to change any code inside the <`body`> element or the <`body`> element itself. You must only change the codes inside the <`style`> element in the <`header`>.**
5. **For this exercise, you are not allowed to use Flex.**
6. Your goals for this exercise are:
   - **Responsive Layout:** Implement CSS to ensure the layout is responsive. Use media queries to adjust the layout for at least three different screen sizes: desktops, tablets, and mobile phones.
   - **Fluid Images:** Make sure that images within news items are responsive and adjust to the screen size without distortion.
   - **Menu Styling:** Link an external CSS file for the menu (`menu.css`) and style the menu to be responsive. The menu should be easy to navigate on all devices.
   - **Gap Management:** Manage the `.gap` class to make sure that there is always space at the bottom of the page, preventing content from being hidden by the footer. Figure 4 shows the expected position of the footer in relation to the page contents. As it can be seen, all contents can be read.
   - **Visual Debugging:** Temporarily added borders are there to help you visualize the layout. You are expected to remove these borders in your final submission.
7. We expected you to write some CSS codes to provide the following design:
   (a) Run the `ExerciseB.gif` to see what we expect as a responsive design.
   (b) As it can be seen in `ExerciseB.gif`, the elements have different designs for different screen size.
   (c) We already provided our expectations for screen sizes. Screens with a width bigger than 769 pixels, screens with a width between 769 and 480 pixels, and screens with a maximum width of 480 pixels. Look at the `@media` CSS blocks.
   (d) In desktops (i.e., width>768px), we expected to have an article with 100% width on top as the large item and 30% small and 70% medium items side by side under it.
   (e) In tablets (i.e., 769px>width>480px), we expected to have an article with 100% width on top as the large item and 40% small and 60% medium items side by side under it. Also, the menu must be lined under the header, and the main body must occupy the whole width.

(f) In mobiles (i.e., width<480px), we expected to have all articles with 100% width. They must sit in three different rows. The condition for menu and main classes is the same as for tablets; however, make the h1 headers' font size 18 pixels.

8. Change my name in the header of the medium item to your name. Commit and push to GitHub.

9. Utilize Google Chrome's Developer Tools to assess your webpage's mobile responsiveness. Open the tools and switch to the mobile device simulator. Examine how your page appears in various screen sizes. Is the layout and content appropriately adjusted for a smaller screen? Make a gif image from your final solution and name it 'ExerciseB.gif'. Don't forget to remove borders before making the video. Commit and push the 'ExerciseB.gif' file to your GitHub repository.

Figure 4: Position of footer in responsive.html file



3.3 Deliverable

1. Copy and paste the content of the <style> element of your final answer into your answer sheet file.

4 Exercise C (Working with Flex)

Your task is to write CSS to make a responsive webpage layout, but this time with the Flex technique. The webpage is more or less similar to the previous exercise, with a few changes.

4.1 Lab Activities

1. Before starting to code, please read this section entirely.
2. Open the ∼/ENSF381/Lab4 folder in Visual Studio Code (VSC).
3. Run the `flex.html` file using "Open with Live Server". Almost all tags were kept from the previous exercise. Please note the role of the now `.body` class in this webpage.
4. **For this exercise, you are not allowed to change any code inside the <body> element or the <body> element itself. You must only change the codes inside the last <style> element in the <header>.**
5. **For this exercise, you must use Flex.**
6. Your goals for this exercise are:
   - **Responsive Layout:** Implement CSS to ensure the layout is responsive. Use media queries to adjust the layout for at least three different screen sizes: desktops, tablets, and mobile phones.
   - **Fluid Images:** Make sure that images within news items are responsive and adjust to the screen size without distortion.
   - **Menu Styling:** The menu style has been changed. You don't need to change them.
   - **Gap Management:** There is no `.gap` class any more as we use Flex, and the way we manage the footer changed a bit.
   - **Visual Debugging:** Temporarily added borders are there to help you visualize the layout. You are expected to remove these borders in your final submission.
7. We expected you to write some CSS codes to provide the following design:
   (a) Run the `ExerciseC.gif` to see what we expect as a responsive design.
   (b) As it can be seen in `ExerciseC.gif`, the elements have different designs for different screen size.
   (c) We already provided our expectations for screen sizes. Screens with a width bigger than 769 pixels, screens with a width between 769 and 480 pixels, and screens with a maximum width of 480 pixels. Look at the `@media` CSS blocks.
   (d) In desktops (i.e., width>768px), we expected to have an article with 100% width on top as the large item and 30% small and 70% medium items side by side under it.
   (e) In tablets (i.e., 769px>width>480px), we expected to have an article with 100% width on top as the large item and 40% small and 60% medium items side by side under it. Also, the menu must be lined under the header, and the main body must occupy the whole width.
   (f) In mobiles (i.e., width<480px), we expected to have all articles with 100% width. They must sit in three different rows. The condition for menu and main classes is the same as for tablets; however, make the h1 headers' font size 18 pixels.
8. Change my name in the header of the medium item to your name. Commit and push to GitHub.
9. Utilize Google Chrome's Developer Tools to assess your webpage's mobile responsiveness. Open the tools and switch to the mobile device simulator. Examine how your page appears in various screen sizes. Is the layout and content appropriately adjusted for a smaller screen? Make a gif image from your final solution and name it 'ExerciseC.gif'. Don't forget to remove borders before making the video. Commit and push the 'ExerciseC.gif' file to your GitHub repository.

4.2 Deliverable

1. Copy and paste the content of the last <style> element of your final answer into your answer sheet file.

5 Exercise D (Editing and Pushing a README.md to GitHub)

In this exercise, you will become familiar with Markdown (MD) formatting and the process of committing and pushing changes to a README.md file on GitHub. You are required to edit the README.md file in your project's repository to include specific content using Markdown formatting. Your final task is to commit and push these changes to GitHub.

5.1 Exercise Overview

1. Open the ~/ENSF381/Lab4 folder in Visual Studio Code (VSC).
2. Open the README.md file in the VSC editor. GitHub has created the file for you, so you only need to edit it.
3. Delete all its default content and make the following changes using Markdown syntax:
   (a) Headers:
      - Add a main header titled **Project Overview**.
      - Add a sub-header titled **Team Members**.
      - Add a sub-header titled **Screenshots**.
      - Use # for main headers, ## for sub-headers.

   (b) Lists:
      - Under **Project Overview**, add a bullet list briefly describing your project.
      - For bullet lists, use - or *.

   (c) Table
      - Under **Team Members**, create a table with one column: **Name**.
      - List the names of two partners in the project.

   (d) Example:

```
| Name   |
|--------|
| X Y    |
| W Z    |
```

   (e) Add images:
      - You already pushed 'ExerciseB.gif' and 'ExerciseC.gif' files to your repository.
      - Add these two images under **Screenshots** using `![Alt text](URL to your image)` syntax. Use a suitable sub-subheader using ###. For example:

```
### Exercise B
![Exercise B final output](./ExerciseB.gif)
```

4. Commit and push changes to GitHub.

5.2 Deliverable

1. Capture a screenshot of the rendering of your README.md file on GitHub. Include this screenshot in your answer sheet file.