Project Part 1

This is an individual assignment.

The rest of the project will be completed as a team, but this first part is completed individually

In this part of the project, you will become familiar with the idea of using a framework to build a web application. You will choose a web framework, build an app, and set up docker compose for deployment.

Objective 1: Choosing a Framework and Docker Set Up

You must use a web framework for the project (As opposed to the homework where you effectively build your own framework). You are expected to find and study documentation to learn how to use your framework of choice as it will not be covered in lecture.

You may choose from the following approved frameworks:

- Flask / Python
- Express / Node.js
- Django / Python
- gin / go
- Play / Java;Scala
- Koa / Node.js
- FastAPI / Python

If you would like to use a framework that is not in this list, let Jesse know and it will be considered for approval. If approved, it will be added to this list.

You may change your framework throughout the semester. If you have chosen your team, it's recommended that you all choose the same framework for part 1 so you all get practice with it before working as a team. If you don't choose the same framework, you can pick one as a team when you start part 2 and switch to that framework.

Once you choose a framework, build an app that uses the framework and hosts *anything* at the root path.

Docker: Once you choose a framework, set up your project using Docker and docker compose. Your app should run on local port 8080.

Testing Procedure

1. Start your server using "docker compose up"
2. Open a browser and navigate to http://localhost:8080/
3. Ensure that anything [other than an error] appears in the browser
4. Check the code to ensure that an approved framework is being used (No credit will be given on any of the 3 objectives if the code uses a TCP socket like the HW code does)

Objective 2: Build an App

Build an app. You have full freedom in what you build for this objective as long as it contains all of the following that must be hosted by your server:

- HTML hosted at the root path
- CSS hosted at a separate path
- JavaScript hosted at a separate path
- An image

All of these parts must be hosted by your server (eg. You must serve the image from your server using your framework of choice. If you add a full url as the src of an img tag, you will not earn credit for this objective and you will miss out on an application objective.)

All files must be served with the correct MIME type and the X-Content-Type-Options: nosniff header must be set.

This app can be very simple and you will not be graded on aesthetics. We will only check if you have the 4 required parts.

Testing Procedure

1. Start your server using "docker compose up"
2. With the network tab open, navigate to http://localhost:8080/ in a browser
3. Ensure that web page appears in the browser
4. In the network tab, check that at least 4 HTTP requests were sent to load the page containing HTML, CSS, JavaScript, and an image all in separate requests
5. Verify that the "X-Content-Type-Options: nosniff" header is set on each response
6. Check to ensure that each of the 4 parts had some impact on the loaded page (eg. the CSS and JavaScript must do something/anything)

Objective 3: Visit Counting Cookie

Note: This objective is identical to HW1 LO3. The intent is for you to see the difference between building features from scratch and using a framework. This will be a common theme throughout the project.

Use a cookie to count the number of times a user has visited the path "/visit-counter". When a user first requests this page, set a cookie to 1 to track the number of times they visited this page. If the cookie is already set (Subsequent visits), read the cookie to check the number of times the user visited, increment this value by 1, then set the cookie again with the incremented value.

Only set/update the cookie on requests for the path "/visit-counter".

The cookie must have an expiration time of 1 hour or longer. It cannot be a session cookie.

The response for this page must contain the current number of visits. This can be served as plain text and can be as simple as just serving the number as your response.

Testing Procedure

1. Start your server with "docker compose up"
2. Open a browser and navigate to http://localhost:8080/visit-counter
3. Verify that the page displays a "1"
4. Refresh the page
5. Verify that the page displays a "2"
6. Close the browser window
7. Open a new window of the same browser and navigate to http://localhost:8080/visit-counter
8. Verify that the page displays a "3"
9. Open a different browser (eg. If Chrome was used in the previous steps, use Firefox here and vice-versa) and navigate to http://localhost:8080/visit-counter
10. Verify that the page displays a "1"
11. In the browser, find the cookie storing the number of visit and change the value to 100
12. Refresh the page
13. Verify that the page displays a "101"

Submission

Submit all files for your server to AutoLab in a .**zip** file (A .rar or .tar file is not a .zip file!). Be sure to include:

- A docker-compose file in the root directory that exposes your app on port 8080

- All of the static files you need to serve in the public directory (HTML/CSS/JavaScript/images)

It is **strongly** recommended that you download and test your submission after submitting. To do this, download your zip file into a new directory, unzip your zip file, enter the directory where the files were unzipped, run docker-compose up, then navigate to localhost:8080 in your browser. This simulates exactly what the TAs will do during grading.

If you have any Docker or docker-compose issues during grading, your grade for each objective may be limited to a 1/3.

Grading

Each objective will be scored on a 0-3 scale as follows:

3 (Complete)	Clearly correct. Following the testing procedure results in all expected behavior
2 (Complete)	Mostly correct, but with some minor issues. Following the testing procedure does not give the exact expected results
1 (Incomplete)	Clearly incorrect, but an honest attempt was made to complete the objective. Following the testing procedure gives completely incorrect results or no results at all. This includes issues running Docker or docker-compose even if the code for the objective is correct
0 (Incomplete)	No attempt to complete the objective or violation of the assignment (Ex. Using an HTTP library) -or- a security risk was found while testing the objective

Note that for your final grade there is no difference between a 2 and 3, or a 0 and a 1. The numeric score is meant to give you more feedback on your work.

3	Objective Complete
2	Objective Complete
1	Objective Not Complete
0	Objective Not Complete

For each completed objective, you will earn 2 application objectives if you complete it before the project part 1 deadline and 1 application objective if you complete it before the project part 2 deadline.

Recall that you must complete all 3 of these objectives by the last day of classes to pass CSE312.

Team Formation

Starting with part 2, you will work on a team of 4-6 members. If you have chosen a team of 4-6 members, use this form to let me know. If you do not fill out this form before the part 1 deadline, you will be placed on random teams.

Team meetings start the week of the part 1 deadline (week 5). You must fill out the team meeting and eval form after each meeting. Failure to do so may result in a loss of application objectives in parts 2-4.