Web Development Lab 3 - Building Web Apps

CS 1301 - Intro to Computing - Fall 2025

Important

- Part I Due Date: **Friday November 14th, 11:59 PM**.
- Part II Due Date: **Monday December 1st, 11:59 PM**
- This is a team assignment, and there will be a peer evaluation and live demo after the due date of the lab. We reserve the right to deduct letter grades from poor peer evaluations.
- Your Lab03 grade will be determined by a group demo on **Monday December 1st**.
- **This assignment is meant to be a self-guided exploration of web development. You are encouraged to search resources independently before seeking TA help.**
  We really want this to be a project representative of what you learned all semester so you can show it off and have one more project under your belt and on your portfolio

Purpose

In this lab, you will build a cohesive web application using the `streamlit` Python libary, which is a Python module that can easily transform Python code into interactive web applications, dash-boards, or reports that can be hosted online. You will build on top of the skills you acquired from Lab01 and Lab02.

As a brief summary, your web application will contain the following pages:

- A home page to breifly explain the purpose of your app
- An analytical or creative app that processes data fetched from a web API of your choice.
- An incorporation of a pre-trained LLM (large language model, like ChatGPT) to further process and analysis the data from your web API.
- An incorperation of Google Gemini to make a Chatbot specialized in specific knowledge from your chosen API

Submission Overview

When you finish this assignment, you will need to zip all of your files into a compressed folder and submit to Gradescope under the correct assignment before the deadline (See Submission Process for more information). **Please double check your submission by re-downloading your submission from Gradescope to ensure that it is correct.** Please un-zip your downloaded file (see

Starter Files) before you double check, since your webpage may not properly load inside of a compressed folder.

CS 1301 is not responsible for incorrect or missing submissions to Gradescope past the deadline of this lab. An incorrect or missing submission will receive no credit and is not eligible for a regrade request. PLEASE double check!

Explanation

You will build and deploy your streamlit web app through the following phases:

- **Part I - Phase 1**: You will set up a multi-page structure for your web app, including a home page and links to all of your other sub-pages. Then you will deploy your app online, so it can be accessed outside of your local computer.
- **Part I - Phase 2**: You will build an analytical, creative, or functional page that will analyze and display data fetched from a web API of your choice. You will also create an interactable/dynamic chart with data from the API.
- **Part II - Phase 3**: You will use the Google Gemini LLM API (similar to ChatGPT) to analyze and process your web API data further, outputting creative LLM-generated messages
- **Part II - Phase 4**: You will develop a chatbot to interact with the LLM that has knowledge about your selected API and has a "memory" of the ongoing conversation

An example of a fully-functioning example of a Lab03 submission and web application can be found here: https://cs1301webdevlab03.streamlit.app/. We recommend spending time exploring it!

Lab Rules

You may **not** use any kind of generators or website/app builders other than streamlit itself.

All code written in this assignment must be written by you. While we encourage the use of Google to reference streamlit properties, any **large sections** of code taken from another source are not permitted (including the projects listed on Streamlit's website).

Submissions violating any of these rules will receive an automatic 0 and a report to the Office of Student Integrity.

Getting Started

To start this lab, you will need the starter files provided on Canvas as `Lab03.zip`. Since streamlit is a python library you can use IDLE or an editor of your choice to modify the template files and create your project.

Starter Files

To start this lab, download the `Lab03.zip` file from Canvas. Extract the files by right clicking the `.zip` and clicking "Extract All" (for Windows) or double-click the file on Mac. Once you extract the files, you should see the following file structure:

```
Lab03
    - requirements.txt # File for all modules that have to be installed
    - Images (folder) # Container for your images
    - Home_Page.py #File for the home page of your app
```

Use these files as a base to start of your lab. Make sure that all images you add from your computer are inside the **images** folder shown above. **All files related to your lab must be in this Lab03 folder.**

Streamlit

What is Streamlit?

Streamlit is an open-source Python library that is used for creating web applications with minimal effort. It is specifically designed for data scientists, engineers, and developers to quickly turn data scripts into shareable web apps. Streamlit allows you to build interactive web applications using Python code, and you don't need extensive web development experience to get started.

Installing Streamlit

You should have already installed Streamlit to complete Lab02, but if you have not installed Streamlit yet, please refer to the installing Python handout on Canvas in Files > Handouts > 01 - Python Installation and Verification Handout.

Lab03 is still completable if you are unable to install Streamlit to your computer. You will host your web application online, which will be the only way you will be able to execute your Streamlit code if you are unable to install Streamlit.

Running Streamlit Scripts

To run your code on streamlit, you must execute your script directly from the command line using Command Prompt (Windows) or Terminal (Mac). Launch the Command Prompt/Terminal application on your computer.

Let's say your files are saved in your documents in the CS 1301 folder in your computer. First, you need to move command line into that folder (also called a directory) by typing the following line into your Command Prompt or Terminal:

- `cd Documents\CS 1301` (Or wherever your Lab03 files are located)

Run the script by typing the following line into the Command Prompt or Terminal and pressing `ENTER`, replacing `script_name.py` with the name of the file you want to execute, e.g. `Profile.py`. Make sure your script name is in quotes.

- `streamlit run "script_name.py"`

To exit the `streamlit` appliation, click onto the Command Prompt or Terminal and type the following on your keyboard:

- `ctrl` + `c`

Additional Resources

Streamlit's official documentation is the most comprehensive and authoritative source for learning Streamlit. It provides in-depth guides, tutorials, and examples, making it the perfect starting point for beginners and advanced users. Documentation

Phase 1a: Multi-Page Application Set Up

Just like in the previous lab, set up the file structure to seamlessly integrate all of your pages into one, cohesive web application.

The goal is to have one home page that has navigational links to the other pages of your web application. Re-visit the Canvas modules for Lab02 to get a refresher on how to do this!

`Home_Page.py` is given to you in the Web Development Lab03 starter files, but you must customize the page with the following requirements:

1. Fill out your team number, section, and team members.
2. Write a quick description for all of your pages, in the form:

```
1. **Page Name**: Description
2. **Page Name**: Description
```

```
3. **Page Name**: Description
4. **Page Name**: Description
```

Other than these requirements, feel free to customize your home page with anything you like! Be sure to test your multi-page web app (which just includes your home page and two portfolios for now) before moving on.

Phase 1b: App Deployment

Before we move on to building the other pages of the lab, deploy your in-progress web application on streamlit's website so it can be viewed online. Previously, you had all of your code on your personal computer. Deploying your app has the following benefits:

1. You can host your web application online so you can share it with others (including recruiters, friends, and family).
2. You can collaborate more efficiently with your team by writing and editing code hosted in the cloud instead of on your computer.

Visit the Phase 3 Canvas Pages in the Web Development Lab03 Module for a tutorial on how to deploy your web app. You will be able to collaborate online using Visual Studio Code Live Share for the remainder of this lab, which will make it easier to collaborate on code virtually.

Be sure to check the Canvas modules for more information

At the end of the lab, you will submit a link to your deployed web application *and* a zipped folder of your code to Gradescope.

Phase 2: Web API Analysis

In Phase 2 of this lab, build a streamlit page by analyzing, visualizing, or presenting data fetched from an external web API. You have full creative freedom to make your project content about whatever you want, as long as the content is school appropriate. Your goal for this part of the lab should be to create a **functional** app that serves the purpose of your choice. Feel free to look at the Canvas Module. Your page must meet the following requirements:

- Must be linked to the home page using the file structure described in Phase 2.
- Must have a clear and meaningful topic or purpose.
- Must incorporate the analysis or presentation of data from an **external web API** into the application. You may display relevant data from your API of choice, or provide functionality related to your API. For example, a weather API can be used to display location-specific weather data, or a finance API can be used to display real-time stock price information.

    - You must analyze and process data received from your API by yourself in code. You may **not** use an LLM to assist with any processing or analyzing of the data (see Phase 5 for that!). You **can not** use the Pokemon API (the same API as the example).
- Must allow at least 2 user inputs or interactions that **affect the app's behavior** (e.g., text input, sliders, select boxes, etc.). Buttons, sliders, etc. with no purpose will not be counted.
- Must contain at least one dynamic visual representation (e.g: images, graphs, plots) that **changes depending on the user inputs**.
    - For example, images can be displayed that depend on what is retrieved from the API, or graphs can be displayed from API data.
- At least one **Dynamic/Interactable** Chart or graph displaying relevant information from your API
- If you use modules that are not part of the Python Standard Library (`math`, `random`, etc.), you **must** list out all of the modules that must be installed for your app to work in `requirements.txt`, one module per line. Streamlit is already included in the `requirements.txt` file that is provided.
    - Do not include the `os` module here, since `os` is a module that is a part of the Python Standard Library.

These requirements are the minimum. Feel free to add anything else. **Be creative!** You will use the same web API for further data processing and analysis in Phase 3 and 4 of this lab, so choose an API with sufficient content!

Example Phase 2 Web App Pages

- Display real-time weather data.
- Analyze and visualize real-time stock/market data.
- Query real-time sports data to display an updated leaderboard or season standings.
- Anything functional relating to your major:
    - Protein folding visualizer for BME students.
    - Astronomical data analyzer for Physics students.
    - Literature text analyzer for LMC students.

There are so many web APIs out there, so go explore and find an API that peaks your interest! For more inspiration, check the Canvas Modules!

Phase 3: Further Data Processing Using an LLM

Now that you have finished a rudimentary analysis of the data fetched from your web API in Phase 2, we will now utilize the power of LLMs to revolutionize the processing, analysis, and presentation of your web API data.

An LLM, or Large Language Model, is a very large deep learning model that is capable of analyzing and generating langauge inputs and outputs to perform a large variety of tasks. In only the past few years, LLMs have taken the AI revolution by storm and are now fully integrated in the lives of many people in the United States. You may have used LLMs like ChatGPT throughout your studies here at Georgia Tech, and you may be familiar with the capabilities of ChatGPT, which include content processing, code generation, problem solving, and critical thinking.

In the final two phases of this lab, you will be interacting with Google Gemini, an LLM developed by Google with a free API. You will use Google Gemini to further process your web API data to either create a fully functioning, specified chatbot or generate functional text from the analysis of your web API data. In other words, you are going to incorporate a ChatGPT-like model directly into your page!

Integrating a live LLM into your page isn't as hard as you think. Visit the Phase 3 Canvas Pages in the Web Development Lab03 Module for a detailed tutorial on how to interact with the Google Gemini LLM API. The module also contains a detailed example of a sample Phase 3 web page that integrates the LLM with data from Phase 2.

Your Phase 3 web page must meet the following requirements:

- Must be a **separate** web page from your Phase 2 web page.
- Must be linked to the home page using the file structure described in Phase 1
- Must use the **same** web API as Phase 2.
- Must allow at least 2 user inputs or interactions that alter the web API's data that is fed into the LLM (e.g., text input, sliders, select boxes, etc.). Buttons, sliders, etc. with no purpose will not be counted.
    - For example, if you are analyzing sports data, the user can input two baseball sports teams to feed the specific sports teams' data into the LLM.
    - Or, if you are analyzing weather data, the user can input a city and a day in the future to fetch the weather forecast for that city on that specific day.
    - Your data pre-processing and data inputs must be **different** than the data inputs used in Phase 4.
- Must use the Google Gemini API to generate functional content. You must:
    - Generate specialized text that pertains to the web API data fed in. For the sports API example presented above, the LLM can generate a news article comparing the strengths and weaknesses from the data of the two inputted baseball teams. For the weather example, the LLM can generate a script for a weatherman presenting the weather forecast for the inputted city and day.

Phase 4: AI Chatbot Implementation

Build another page that contains a chatbot that can sustain a intelligent and functional conversational using the data passed into the LLM. For the sports example, the chatbot can answer questions regarding the team composition for a user unfamiliar with the team roster. For the weather example, the chatbot can help the user decide whether an activity they are planning to do is feasible under the predicted weather conditions.

Hint: Check the Canvas modules for help on how to implement this!

- Must include error handling (Try/Except) for errors that may be raised by the LLM. This may include hitting rate limits due to submitting too many LLM requests at once, or accidentally generating inappropriate content. Your app must never crash due to errors raised by the Google Gemini LLM API.

Please do not be overwhelmed if you are intimidated by these requirements. Visit the Canvas Pages for more details on how to actually specialize the Google Gemini LLM API to your specific web API data.

You may have noticed that ChatGPT is not helpful for questions asked about real-time data, since the text on which ChatGPT is trained ends on September 2021. This is a wonderful opportunity to build a chatbot and generate useful content using real-time data (from your web API); something that ChatGPT can't currently do!

Extra Credit: Extra functions + Deploying Your App

You may receive up to **10 bonus points** on this lab (allowing for a max grade of 110/100 on Lab03) for the following separate requirements:

Requirement 1: Layouts and Containers (5 points)

You will receive 5 points of extra credit if you implement the use of **Containers / Layers** in Streamlit to make your website look more clean!
These points will be given by your TA, so make sure you do this in a non-trivial way!
Herehttps://docs.streamlit.io/develop/api-reference/layout is a link to read more about them!

Requirement 2: Resume Description or LinkedIn Post Submission (5 points)

Complete **one** of the following assignments for 5 points of extra credit:

- Craft a detailed description of this project, highlighting the key programming skills and specific learning outcomes you have achieved (fetching web APIs, building web apps, working with LLMs). This description should be tailored for inclusion in your resume, showcasing your hands-on experience and the technical competencies you have developed. For guidance on crafting an effective resume, we recommend consulting the Georgia Tech Career Center's resource guide, available at https://career.gatech.edu/resumes/.

- **OR** share your project experience on LinkedIn. Create a post that not only showcases images from your project but also highlights the skills you have honed and the insights you have gained throughout the lab. This is an excellent opportunity to engage with your network and demonstrate your practical expertise in a real-world application.

Submit either your resume project description or a link to your LinkedIn post on Gradescope for extra credit (one per team).

Submission Process

How to Submit

Part I & II Due Date: Friday November 14th, 11:59 PM.**

- Submit the Zip File with Phase I and Phase II to Gradescope.
- Submit the link to your deployed website

Part III & IV Due Date: Monday December 1st, 11:59 PM

- Submit the Zip File with Phase III and Phase IV to Gradescope.
- Submit the link to your deployed website

Lab03 will be submitted through Gradescope. You will need to submit all files used in Lab03 (all images and python files) and a link to your deployed streamlit web application. To do this, compress the entire Lab03 folder that you are working in into a `.zip` file. Follow the instructions below:

1. Right click your Lab03 folder:
    - On Windows, click **Send To > Compressed (zip) folder**
    - On Mac, click **Compress "Lab03"**
2. This will create a zip folder on your computer. Rename the zip folder so that it has the following format (replace the below with your first and last name):

    ```
    {team_member_last_name_1}-{team_member_last_name_2}.zip
    ```

3. Upload this `.zip` file to the Web Development Lab03 assignment on **Gradescope.** To double check that you did this correctly, please download your submission and run your app to ensure everything works and looks correct.

Please put all images that you used in your website in this Lab03 folder. Otherwise, your grader will not be able to see images that you included.

Grading Rubric

Requirement	Points
Phase 2: API Implementation Page	35
Phase 3: Gemini + API Page	25
Phase 4: Gemini Chatbot Page	25
Demo Questions	15
Extra Credit	10

The grading rubric for Web Development Lab03 is **deliverable based** and will be assessed with a live TA-led demo. The demo will assess the functionality of your web app and require you to answer a series of comprehension questions that may test code functionality, streamlit comprehension, and ethical considerations. A future announcement with demo details will be published as the deadline for Lab03 approaches.

Final Total (with extra credit): 110/100 points