CLC - Capstone Project

Lython: An AI Python Tutor

Richard Pointing

College of Science, Engineering, & Technology, Grand Canyon University

CST-590: Capstone

Amr Elchouemi

April 30th, 2024

Capstone Project

Project Overview and Project Objectives

Background: For high schoolers with little computer literacy, computer programming classes are often incredibly challenging. When having to learn a language on top of learning to use a keyboard more effectively, it can become a lot of work. Often I find myself unable to help all my children during a class period, and feel like I'm leaving some students behind because of how much time it takes to explain aspects of the problem individually to each student.

Objectives: To create an application with a user-friendly interface that connects with an AI framework to help address my students issues during programming assignments.

Challenges: Maintaining intellectual integrity when answering questions, and being able to teach concepts without giving away answers.

Benefits and Opportunities: A tool students will be able to use during class time that will help them solve Python programming problems if I'm not available to help them.

Project Scope

1. The scope of the project falls within user interface design, AI capabilities, and data collection.
2. The project will be done mostly on the creation of a working and efficient user-interface, followed up by the creation of accurate prompts for the provided questions. The research portion would be found within the data poured into the AI framework, allowing it to learn what to say and generate appropriate examples and feedback.

Work Breakdown Structure

ID	Task	Dependencies	Status	Effort Hours	Cost	Start Date	Planned Completion	Estimate to Completion	Actual Completion	Resource
1	Creation of the User Interface	Flask	Complete	6		3/1	3/15		3/14	
2	Connection to AI framework	OpenAI	Complete	1	$10	3/1	3/6		3/3	
3	Chat History Page	Flask	Complete	2		3/21	3/25		4/2	
4	Testing and Debugging	Flask	Complete	1		3/21	4/1		4/15	
5	Launch to Production	PyPly	Complete			3/21	4/1		4/15	

Project Completion

Project Completion Criteria
1 - Student can ask for help on a specific problem without being told explicitly how to solve it.
2 - Student can submit code and get feedback on potential errors, with further explanation on why it's wrong.
3 - Student can ask for help understanding a term or function, and receive examples that are relevant.

Assumptions and Constraints					
ID	Description	Comments	Type	Status	Date Entered
1	Constraint - AI being too correct	Students using AI to solve their homework for them without any effort	Constraint	Dealt with	
2					

Project Controls

Risk Management				
Event Risk	**Risk Probability (high, medium, low)**	**Risk Impact**	**Risk Mitigation**	**Contingency Plan**
What is the risk?	What is the probability?	What is the impact if the risk occurs?	What can be done to minimize the risk?	What can be done to minimize the impact of the risk?
The AI gives too much information away	Decently likely	Breaks academic integrity	More constraint on AI feedback	Extensive testing and debugging before release

Project Schedule

1. Creation of User Interface - 30 days
2. Connection to AI/prompt creation - 5 days
3. Implementation of AI into UI - 30 days
4. Testing and Debugging - 30 days

Issue Log

Issues Log								
ID	Issue Description	Project Impact	Action Plan/Resolution	Owner	Importance	Date Entered	Date to Review	Date Resolved
1	What is the issue?	How will this impact scope, schedule & cost?	How do you intend to deal with this issue?	Who manages the issue?				
2	When I refresh the page, it re-queries the question and stores it in the database	Takes up a lot of time to refresh the page, and if done by many users the database will fill up quick	Need to look into it, I could use a function to call the users last stored message in the textarea so they still have the text there when the page refreshes, I just need to learn how to force the function to not start on page refresh.	My issue				
3	ChatResponses ID being null	Difficult to implement further functionality to chat history page	Not sure, I have it set to serial right now, so maybe I'll manually write out not null auto increment instead.					3/28
4	Response_time not being displayable	Makes time of question asked not viewable	For some reason, the data type of the response_time (datetime) breaks every time it is displayed					

Requirements Analysis

Use Cases

Largely useful when teaching large classes when you can't always ensure each student gets adequate teaching 1 on 1 time.

Technical Requirements

User interface designed with either HTML & CSS or Python (Flask), OpenAI.

System Logical Model

User either accesses or logs into application -> user asks AI question -> AI responds based on query design and past experience from data collected -> responds with generated answer.

Reports

All history for the user is stored in a SQLite database. Accuracy could be measured by user-answered tests after an answer is given, but the integrity of these responses is sometimes questionable.

Screen Definitions and Layouts

A few screens built with HTML/CSS for logging in, registering, using the chat, and seeing your history.

Security

If a user-profile system becomes integrated at some point, the integrity of each user's personal information will have to be ensured, as well as their queries. Also, since it's a chat interface, ensuring no hacking methods like SQL-injection are capable.

Design Plan Summary

1. The project itself is to create an application students can use to help them with their Python homework without giving away too much of the answer. To achieve this, I'll have to create an interaction between the user and an AI framework trained to respond to the user's questions. I will also have to create a front-end where the user can interact with the AI.

The development as it currently stands, will require;

1) Connection between AI and UI
2) Make and tweak prompts that will generate expected answers
3) Creating the user interface (UI)
4) Testing the integration between the two (AI and UI)
5) Testing the AI

Overview of Design Concepts

I had originally planned to not allow users to insert their own code. This was because I had an idea of what the questions they would need help with would look like. However, I thought that the more dynamic approach would improve the longevity of the product, and also allow for more accurate answers in case the question is irrelevant to a specific problem or not.

The current design sees an integration with OpenAI's API to connect to their Chat GPT version 3.5-Turbo. This saves me the time of having to train the AI in NLP, and allows for more flexibility with user questions. The only thing for me is to sit down and create a prompt that will be entered before the users, to help field the results and ensure there isn't any workaround to just get the answer they are looking for. Currently, the chatbot tells the user general tips and tricks about the concept they have questions on, centered around the 6 fundamental concepts of programming.

I also implemented a feature using LangChain, that uses Retrieval Augmented Generation (RAG) to allow for more personalized responses. Using LangChain's web scraper on some problems that my students have done for homework, I was able to use OpenAI's model to feed it embedded data from the scraping, and respond with information something like ChatGPT wouldn't be able to answer or know.

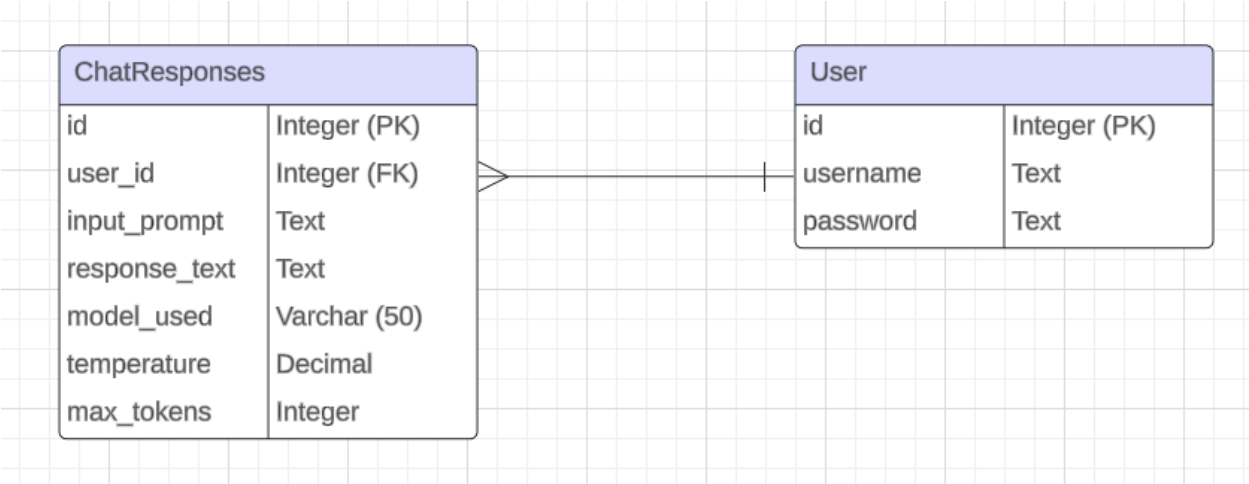
Deliverable Acceptance Log

ID	Deliverable Description	Comments
1	AI Framework	OpenAI API
2	User Interface	HTML/CSS
3	Integration of AI into UI	Python/Flask
4	RAG Feature	LangChain

Detailed Solution Architecture

1. The application should be usable alongside a student performing their coding assignments. Since the students work in either a web-based IDE or on the local machine's IDE's, I think the easiest might be a web-based application. The student should be able to seamlessly open their programming assignment, be able to write and edit their code, and pull up the AI helper to work alongside them.

2.



3. The web page has a basic log-in/log-out system that tracks the user's username, password, and a separate table for the generated responses. Once the user is signed-in, they can ask the AI a question. The AI sends the question back to the screen, as well as saves the response in the SQL table.

4. The security portion I believe links nicely with the concept of not giving away too much of an answer. The security should really be validation that the user isn't attempting to exploit the AI to

just give it the answer. However, this would only be an issue that would arise after advancing onto allowing more loose user-input. Also, passwords are hashed and stored this way.

Hardware and Software Technologies

1 - AI Framework (OpenAI, LangChain)
2 - User Interface (Web Application built with PHP, HTML, etc.)
3 - Database (To store user and output data, SQLite)

Mapping of Functional Requirements

The current functional requirement has the user ask a question and store it inside the user_question variable. The question is passed to the ask_python_question function in the openai_doc file after the user clicks “Ask.” Once the question is passed, it’s appended to the end of a larger prompt I generated to help filter the question and ensure there isn’t any cheating. The response is returned and stored inside of bot_response, which is displayed automatically after being generated. The bot_response is stored in a SQLite database, alongside the user upon creation.

When updated, the web scraping feature will embed and store the questions in a vector database, that with OpenAI and LangChain can be deciphered to be used to respond to questions.

Source Code Listing

https://github.com/mr-pointing/Lython_GCU

Implementation Plan

My implementation plan is as follows;

1) Get project on GitHub
    a) Accomplished
2) Get app to store and remember users
    a) Accomplished
3) Get app to store chat history/make it accessible
    a) Accomplished
4) Get app to web scrape and use LangChain
    a) Accomplished
5) Measure/Review the data to see how accurate
    a) Find out some quantifiable measure to see if response is good enough
6) Design
    a) Accomplished
7) Creation of User Guide
8) Launch

Current Project Status: The project is currently finished. The main functions are complete and work. Below are my current functioning test cases.

Video Presentation Link: https://youtu.be/D9-pGLFJ85I

Test Case: init_db_command			
Priority: High			
Module: Database			
Test Objective: Ensure the database can be created and the ‘init-db’ command can be called			
Step	**Detail**	**Expected Results**	**Problem/Issues**
1	Recorder class made with called attribute to act as a flag for init_db	Recorder class made without issue	None
2	Define a fake function that sets called to true to act like the actual init_db	Used with monkeypatch	None
3	Use monkeypatch to replace the original init_db function with our new fake function	Our fake database command is called instead of init_db	None
4	Runner is used to call the init-db command	The fake init_db function is called instead of the actual init_db	None

Test Case: get_close_db
Priority: High
Module: Database
Test Objective: Ensure the database can be opened and closed

Step	Detail	Expected Results	Problem/Issues
1	Call the get_db() function from db.py and checks connection	get_db() to run without issue	None
2	Testing application context/connection closed by ‘SELECT 1’	Exception should be captured since the database should be closed	None

Test Case: test_register			
Priority: High			
Module: Authorization			
Test Objective: Make sure the user can be registered			
Step	**Detail**	**Expected Results**	**Problem/Issues**
1	Test a get request using code 200	Get request response works	None
2	Testing a post request	Inserting a new user and password into the fake database	None

3	Assert we are logged in by checking the headers location	Should be in /auth/login	None
4	Uses app context to grab the user that was just inserted to make sure it works	The db execution isn't None	None

Test Case: register_validate_input			
Priority: Medium			
Module: Authorization			
Test Objective: Validate the required inputs for Register			
Step	**Detail**	**Expected Results**	**Problem/Issues**
1	Set up three different test cases to use	One without a username, one without a password, and one with a username that already exists	None
2	Test each against the register function	Each specific error should be raised depending on the case	None

Test Case: test_login			
Priority: High			
Module: Authorization			
Test Objective: Test the log-in functionality			
Step	**Detail**	**Expected Results**	**Problem/Issues**
1	Test a get request using code 200	Get request response works	None

2	Perform a log-in	Log-in is successful	Fails: This is currently the only failing test. I am not sure exactly why this test fails; I think it might have to do the Location being set to something other than "/"
3	Check the session and global username	User should exist and log in	Fail

Test Case: test_login_validate_input			
Priority: High			
Module: Authorization			
Test Objective: Test the log-in functionality			
Step	**Detail**	**Expected Results**	**Problem/Issues**
1	Set up three different test cases to use	One without an invalid username, and one without an invalid password	None
2	Test each against the register function	Each specific error should be raised depending on the case	None