**COT6930 – Generative AI and Software Development Lifecycles**

**FINAL PROJECT**

**Group:** Genesis

**Members:** Neha Mohana ([n2024@fau.edu](mailto:n2024@fau.edu))

Smruthi Sharath Bejwadi ([sbejwadi2024@fau.edu](mailto:sbejwadi2024@fau.edu))

Tribhuvana Avvaru ([tavvaru2024@fau.edu](mailto:tavvaru2024@fau.edu))

**Solution Name:** AI Study Companion

1. **Solution Overview:**

The platform serves students during their study period by deploying Hugging Face Generative AI models at no cost.

Students who use the AI Study Companion Bot access easy ways to understand topics alongside note summarization functions for flashcard generation and quiz creation and free-form question responses.

**The solution enables users to access the following functions:**

* The system explains subjects while supplying examples for further understanding.
* The flashcards generation system provides users with flashcards of topic-related questions and corresponding answers.
* The application produces multiple-choice questions (MCQs) which include related answers in their output.
* The system produces clear summaries following the user's guidelines.
* The system provides a response mechanism which generates intelligent answers for academic inquiries from user-provided questions.

**The solution works by:**

* A user chooses between available functions which include Explain, Flashcards, Quiz,Summarize or Ask.
* The platform automatically creates elaborate set of prompts after users select options.
* The application utilizes the free model google/flan-t5-large on Hugging Face Inference API to receive prompts.
* The system produces proper and organized responses.
* The user interface of the Streamlit web app shows responses in an organized manner.

**Prompts:**

“Write a brief solution overview for an AI Study Companion that explains topics, creates flashcards and quizzes, summarizes notes, and answers questions using Hugging Face. Describe how users select a function, the system sends a prompt to the model, and shows the response.”

1. **Requirement Analysis**

**Prompt:**

"Create functional and non-functional requirements for a Hugging Face-powered AI study bot."

**Functional Requirements:**

| **Requirement Description** | **Priority** | **Complexity** | **Expected Outcomes** | **How to implement it** |
| --- | --- | --- | --- | --- |
| Explain a given topic clearly | High | Medium | Short, clear explanation with examples | Smart prompt to Hugging Face model |
| Create a multiple- choice quiz | High | Medium | MCQs with options and answers | Quiz prompt to model |
| Generate flashcards for a topic | High | Medium | flashcards formatted clearly | Flashcard prompt to model |
| Summarize provided study notes | High | Medium | Short, clear summary | Summarization prompt to model |
| Answer a free-style question | High | Medium | Friendly and detailed answer | Q&A prompt to model |
| Validate user input (empty check) | High | Low | Warn user if input missing | Input validation in Streamlit |
| Provide user-friendly web interface | High | Low | Easy navigation and usage | Streamlit app structure |
| Handle API errors gracefully | High | Medium | Display error message if API fails | Exception handling |
| Deploy app locally and online | Medium | Medium | Easy access through localhost / Hugging Face Space | Streamlit server / Hugging Face Space deployment |
| Display formatted and clean outputs | High | Low | Organized results for users | Streamlit markdown/text blocks |

**Non-Functional Requirements:**

| **Requirement Description** | **Priority** | **Complexity** | **Expected Outcomes** | **How to implement it** |
| --- | --- | --- | --- | --- |
| Quick response time (<20 seconds) | High | Medium | Minimize model latency | Optimize prompt size |
| System should handle invalid API responses | High | Low | No crashes | Try-except blocks |
| Simple and attractive UI design | High | Low | Better user experience | Streamlit layout components |
| Free access without subscription | High | Medium | No payment required to use app | Use Hugging Face free models |
| Maintain privacy and not save user inputs | High | Low | User trust | Do not store data on server |
| Ensure high accuracy of generated educational content | High | Medium | Minimize factual/model errors | Use prompt tuning, monitor outputs |
| Maintain compatibility across Windows, Mac, Linux | High | Medium | App works on all OS | Avoid OS-specific dependencies |
| Accessibility for visually impaired users | Medium | Medium | Supports screen readers, alt text | Streamlit accessibility features, TTS |
| Minimal system resource usage (<1GB RAM for small models) | Medium | Medium | Doesn’t slow down user’s device | Use flan-t5-small and limit concurrent runs |
| Secure handling of user data and privacy | High | Medium | No data leaks or unauthorized access | Never store sensitive data, secure .env files |
| No storage of user data without consent | High | Low | User privacy respected | Clearly state in README/policy |
| Allow easy installation and deployment | High | Low | Simple setup instructions | Provide clear README.md and requirements.txt |
| App must handle multiple users (if hosted online) | Medium | Medium | No data mix-ups, good concurrency | Use stateless design, session management |
| Open-source compliance | Medium | Low | All code and models used are open and shareable | Use only open-source models and code |
| Logging of errors for debugging (not user data) | Medium | Low | Helps troubleshoot app failures | Use Python logging, but no sensitive data |

1. **Solution Architecture:**

**Prompt:**

"Generate solution architecture diagram for a free Hugging Face AI chatbot."

**Solution:**

**A diagram of a company

AI-generated content may be incorrect.**

**Explanation:**

As students interact with several study support materials, our solution architecture for the AI Study Companion Bot is meant to provide them a perfect and interactive experience. The process begins as our flowchart depicts when a student enters their data into the user interface after selecting a function—such as explanation, flashcards, or summarizing. The system immediately checks the input; should it be empty, the learner should re-enter their request and get a clear warning. This guarantees processing of just relevant searches and a user-friendly interface.

The Prompt Builder Module generates a context-relevant prompt fit for the chosen usage once a valid input is provided. Based on this stimulus, the Hugging Face Inference API answers by using an artificial intelligence model. The motive of our design is Strong error handling; should the API fail or generate an error, the user is then notified and given a chance to try once more, therefore avoiding disruptions in the data flow. Should the artificial intelligence respond satisfactorially, data moves to the Response Formatter Module, which organizes and polishes the output for reading. The final result—an explanation, test, flashcards, or summary—showcases the student in a logical and practically useful form.

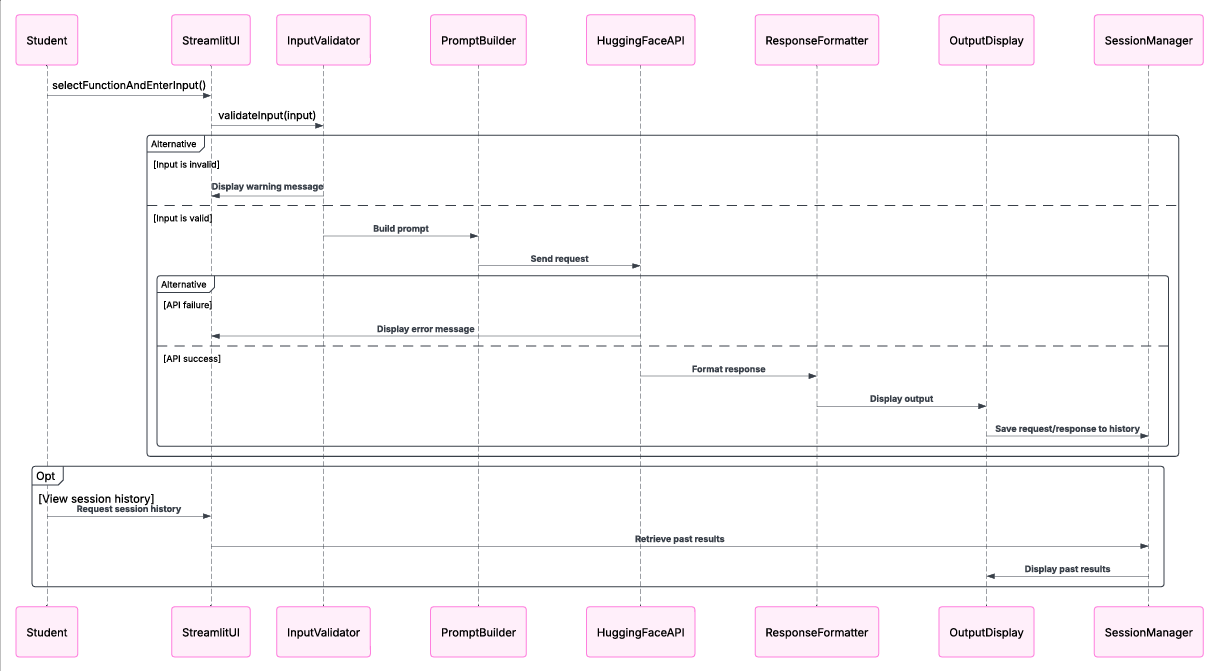
The Session History Manager improves user experience even more under oversight of all interactions and results. This app helps students to effectively and iteratively learn by letting them analyze their past searches and AI-generated outputs inside the same session. Stressing overall clear user feedback, dependable problem handling, and simple access to session history ensures that our design assures the AI Study Companion Bot performs as a full and effective academic aid.

1. **System Modelling:**

**Prompt:**

"Design sequence and class diagrams for a modular study companion bot."

* 1. **Sequence Diagram:**

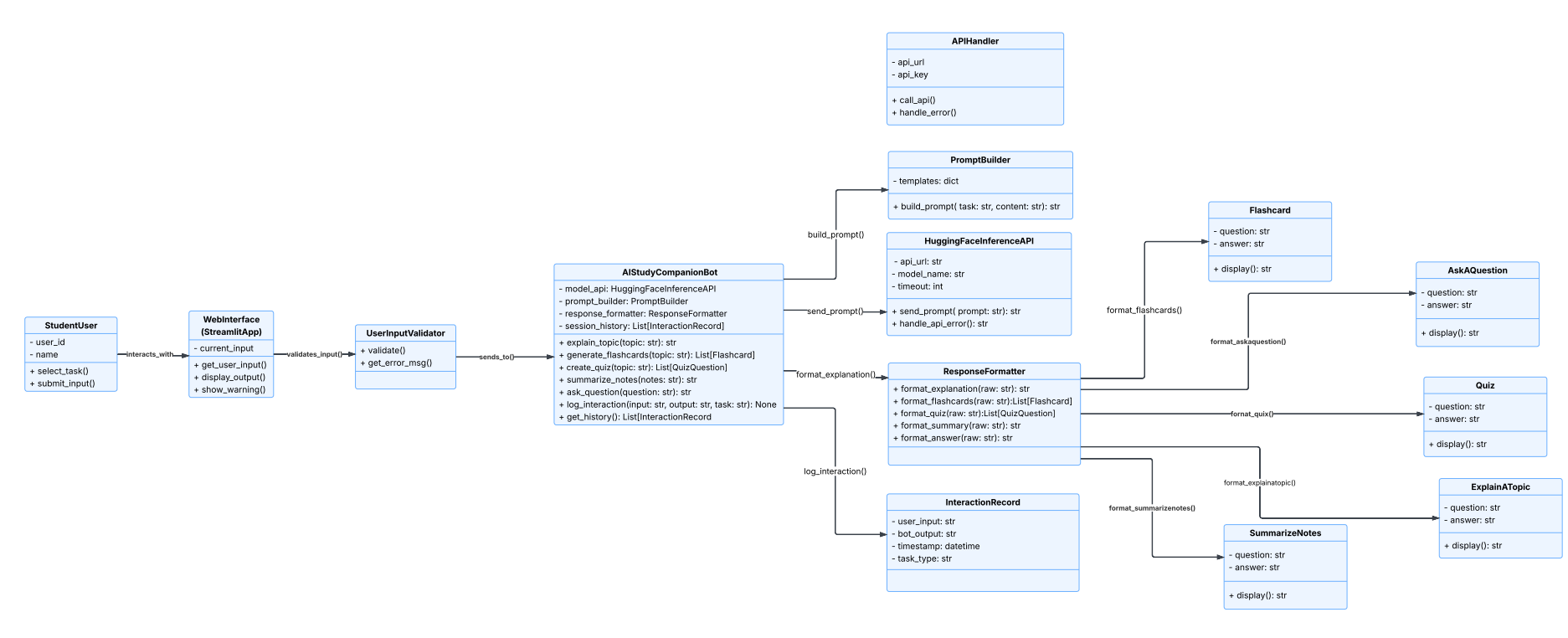


**Explanation:**

We wanted to show exactly in this sequence diagram how a typical user interaction moves across our AI Study Companion Bot system. It starts when the student select a function and enter their data into our StreamlitUI online interface. The Input Validator then instantly checks the input looking for any missing or erroneous data. Should the input prove to be invalid, we made sure the system immediately delivers a warning indication to the user so they would know what has to be fixed.

Once the input is valid, The Prompt Builder creates a custom prompt based on the chosen function. Receiving this prompt, our Hugging Face API module handles correspondence with the AI model. We included an alternative path in the graphic for API failures to show the user any problems rather than leave her confused. Should the API request be authorized, the raw artificial intelligence output is passed to the ResponseFormatter, which generates the response to ease reading and understanding. The output display presents to the user the completed work. Every request and response is also stored by the Session Manager so that students may examine their session record anytime they so like. We also presented the optional scenario whereby a student assesses their retrieved and displayed past performance. We want to keep the workflow simple and user-friendly generally while yet maintaining a record of all contacts and handling errors.

* 1. **Object Diagram OR Agent Architecture:**

****

**Explanation:**

For our AI Study Companion Bot project, we designed the system architecture so that the StudentUser, interacting with the program via the Web Interface built with Streamlit, starts the whole process. The WebInterface handles user inputs, result displays, and forwarding such inputs to the UserInputValidator for first validation. After validation, the input is sent to our main logic class, the AIStudyCompanionBot. After using the PromptBuilder to create exact prompts suited to the selected task, so structuring the remaining of the workflow, this main class calls the Hugging FaceInferenceAPI to generate responses from the AI model.

Once the model answers, our ResponseFormatter class arranges the unprocessed output into test- and flashcard-friendly study resources. Whereas the Flashcard class arranges the contents of every flashcard and quiz. We keep an InteractionRecord for every contact with a user so that we can keep track of what they're doing and give potential analytics tools access to that data. We also made an APIHandler to handle any calls from outside APIs. This made our code simpler and easier to maintain. Our plan is to make our system flexible, clean, and easy to change so that it can keep up with new developments.

1. **Prototype Implementation Plan**

* Rules: The model requires detailed prompts which function as guidelines to achieve proper output results.
* Prompt Templates: Tailored for explanation, flashcards, notes summarization, free Q&A.
* RAG: Not applied here — direct smart prompting.
* Artifacts: Python script (study\_companion\_app.py), Streamlit app.
* Multi-Agent: The system operates with a single AI assistant which manages all user tasks.
* GenAI Models Used: google/flan-t5-large (via Hugging Face Inference API).
* Model Parameters: The model operates with standard free API settings but contains small modifications to prompt administration.
* Datasets: The system does not depend on external datasets while it handles inputs from users in real-time to generate responses.

**Prompt:**

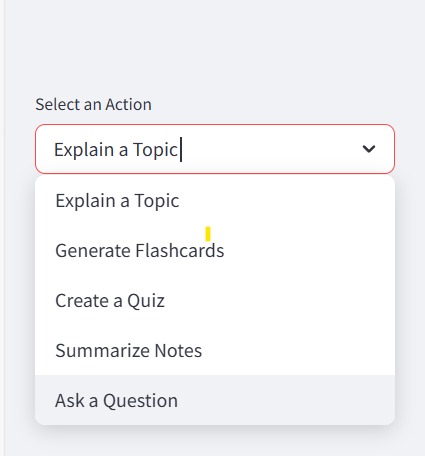
"Outline implementation plan for Hugging Face-based AI Study Assistant using free models."

1. **Experimental Prototype:**

**Link to GitHub**: https://github.com/NehaMohan59/StudyCompanionAI

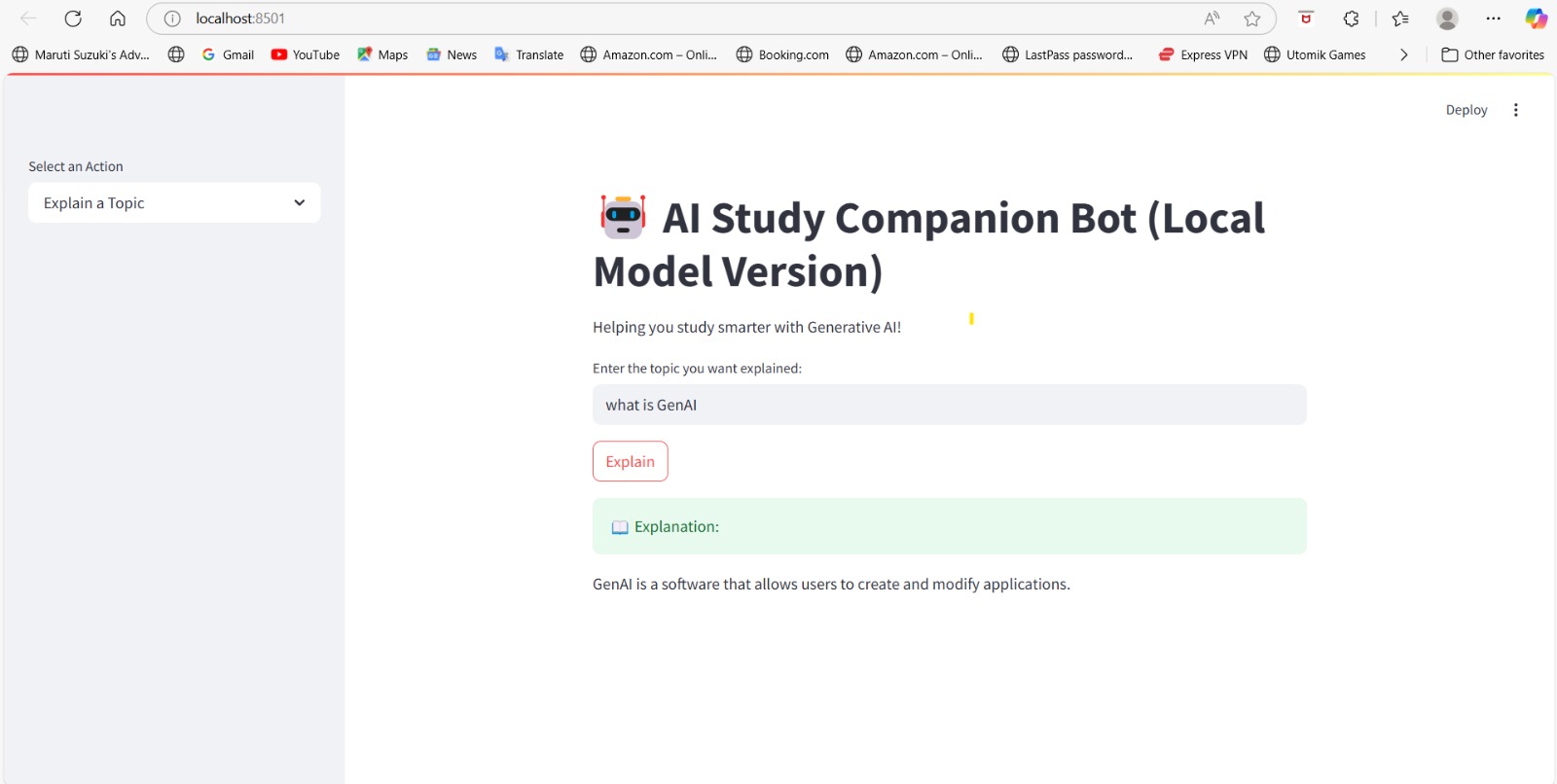
**Screenshots**:

This screenshot shows how many artificial intelligence-powered study tools our website shows students as a dropdown menu. Other choices among them include "Explain a Topic," "Generate Flashcards," "Summary Notes," “create a quiz” and "Ask a Question." We let customers quickly access the particular academic help they need for effective and tailored learning by grouping these options into a single, simple interface.

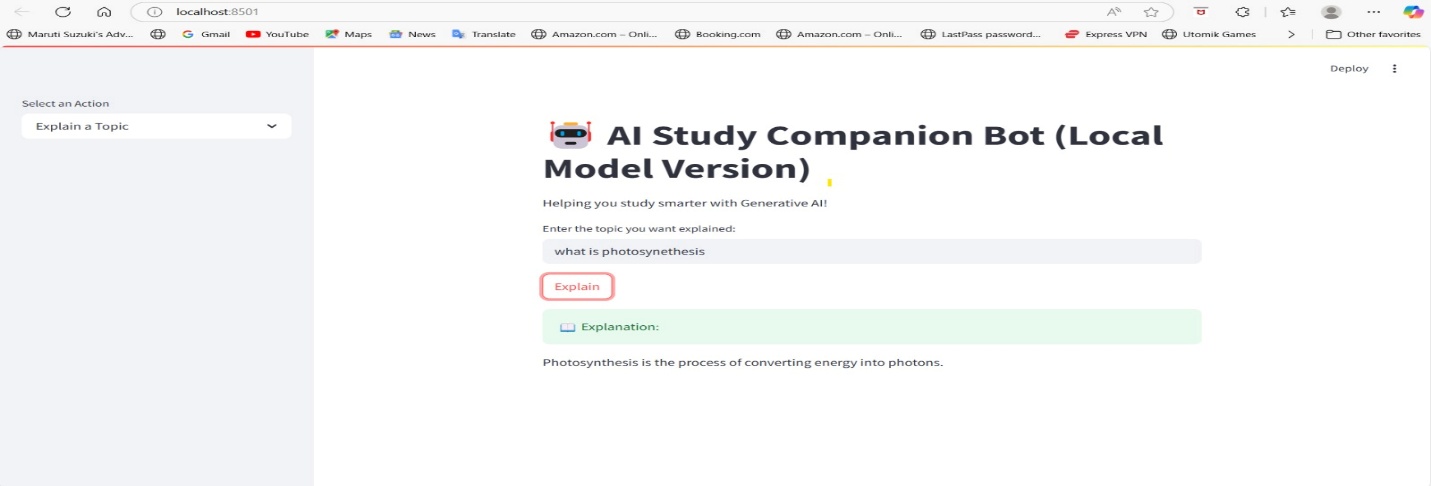


**Explain a Topic:**

This screenshot also shows our bot's features. Choosing the "Explain a Topic" option, we next entered "what is GenAI." Following a "Explore," click, the bot responded in the output box. This immediately supports the learning goals of the students since it displays how rapidly our system can respond for a variety of customer questions.

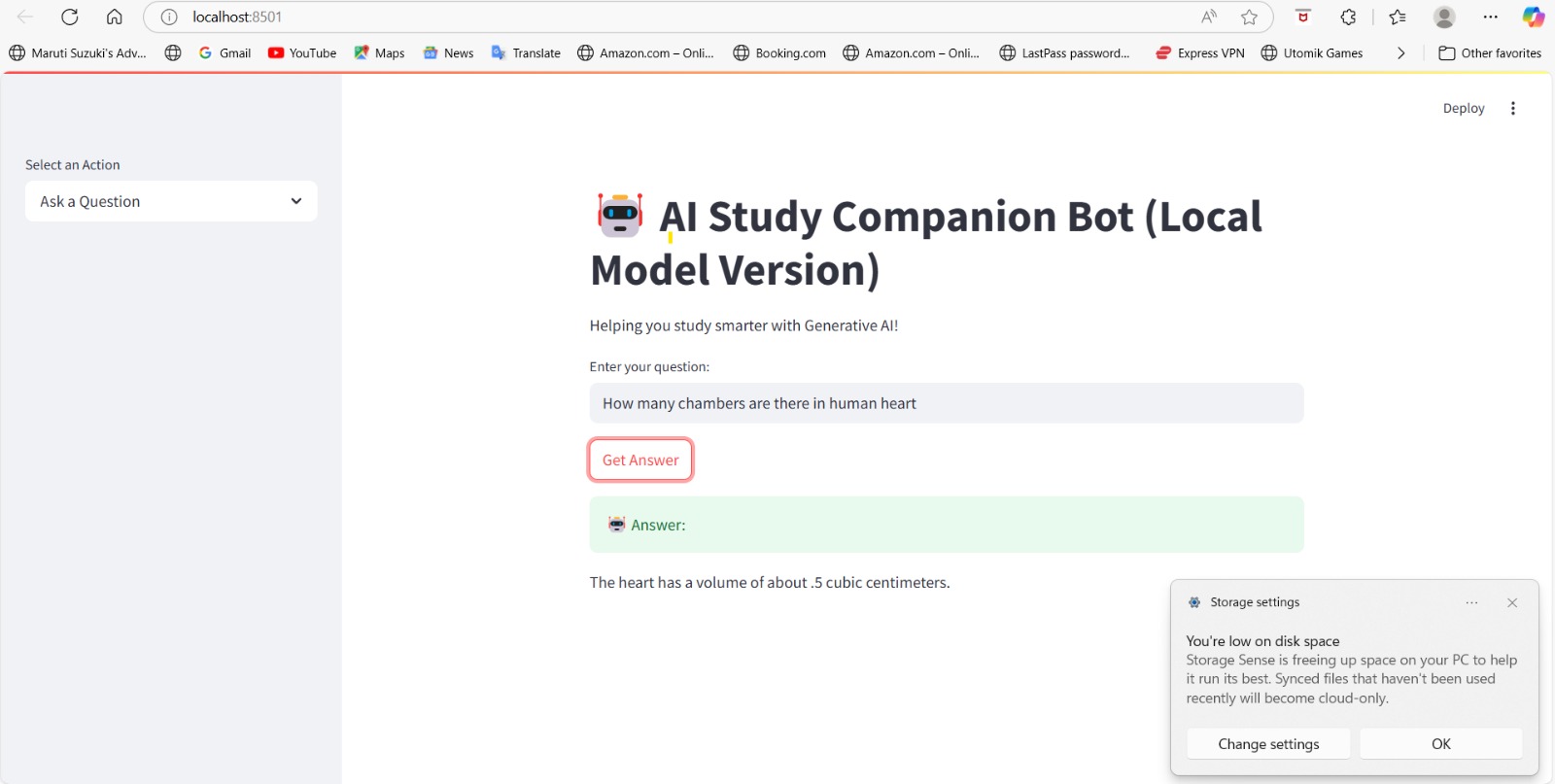


This screenshot shows our artificial intelligence study buddy bot in action. Here we place "what is photosynthesis" utilizing the "Explain a Topic" tool. Clicking the Explain button showed in the green output box from the bot a clear, concise explanation. This emphasizes how generative artificial intelligence helps our solution instantly, exactly answer student questions.

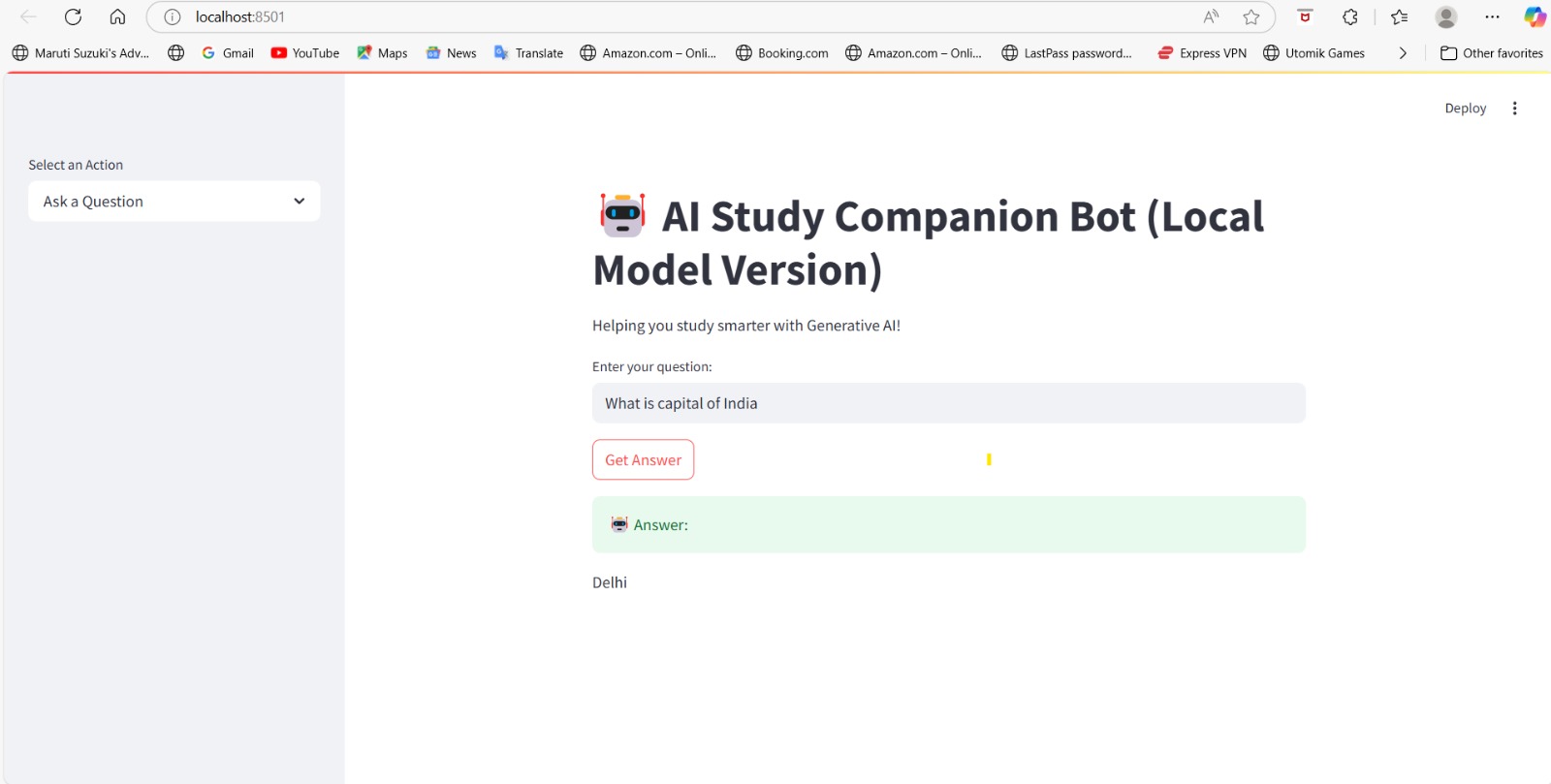


**Ask a Question :**

This screenshot's "Ask a Question" function lets us enter "how many chambers does a human heart have"? Clicking "Get Answers," the algorithm answered exactly—"Four". This example shows how fast our bot can respond to direct academic queries, therefore providing students with instant assistance for their benefit.

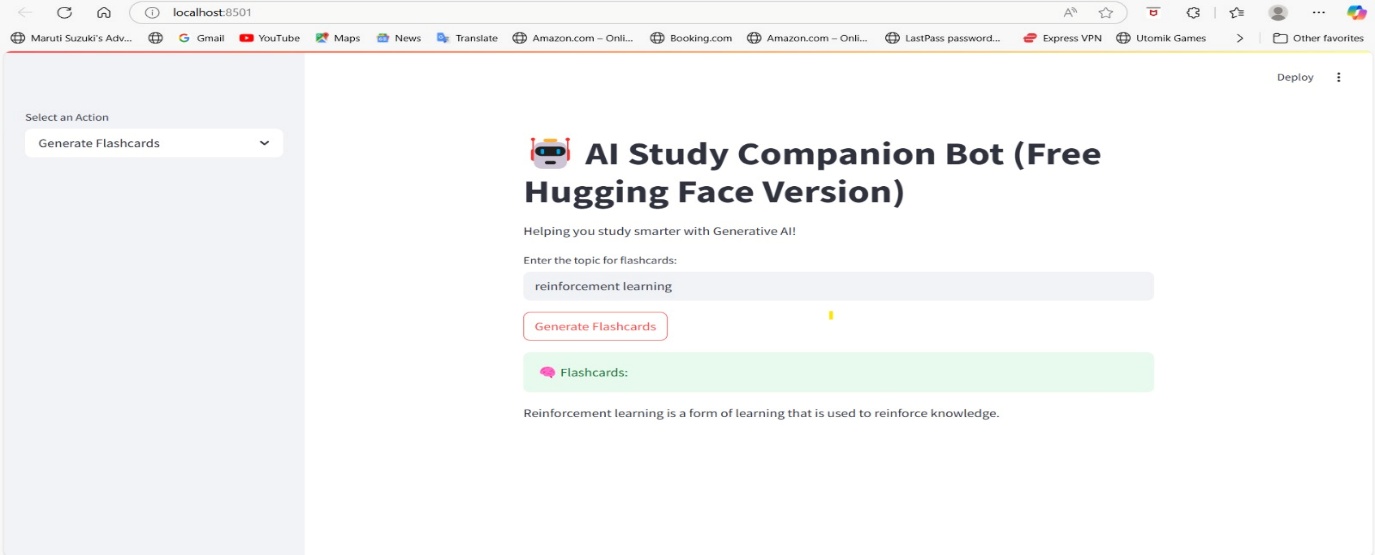


Here we typed "what is the capital of India," using the "Ask a Question" tool. Clicking "Get Answer," the computer responded correctly right away—"Delhi". For students looking for rapid information, this shows how easily the bot can respond to simple factual queries, which makes it a great tool.



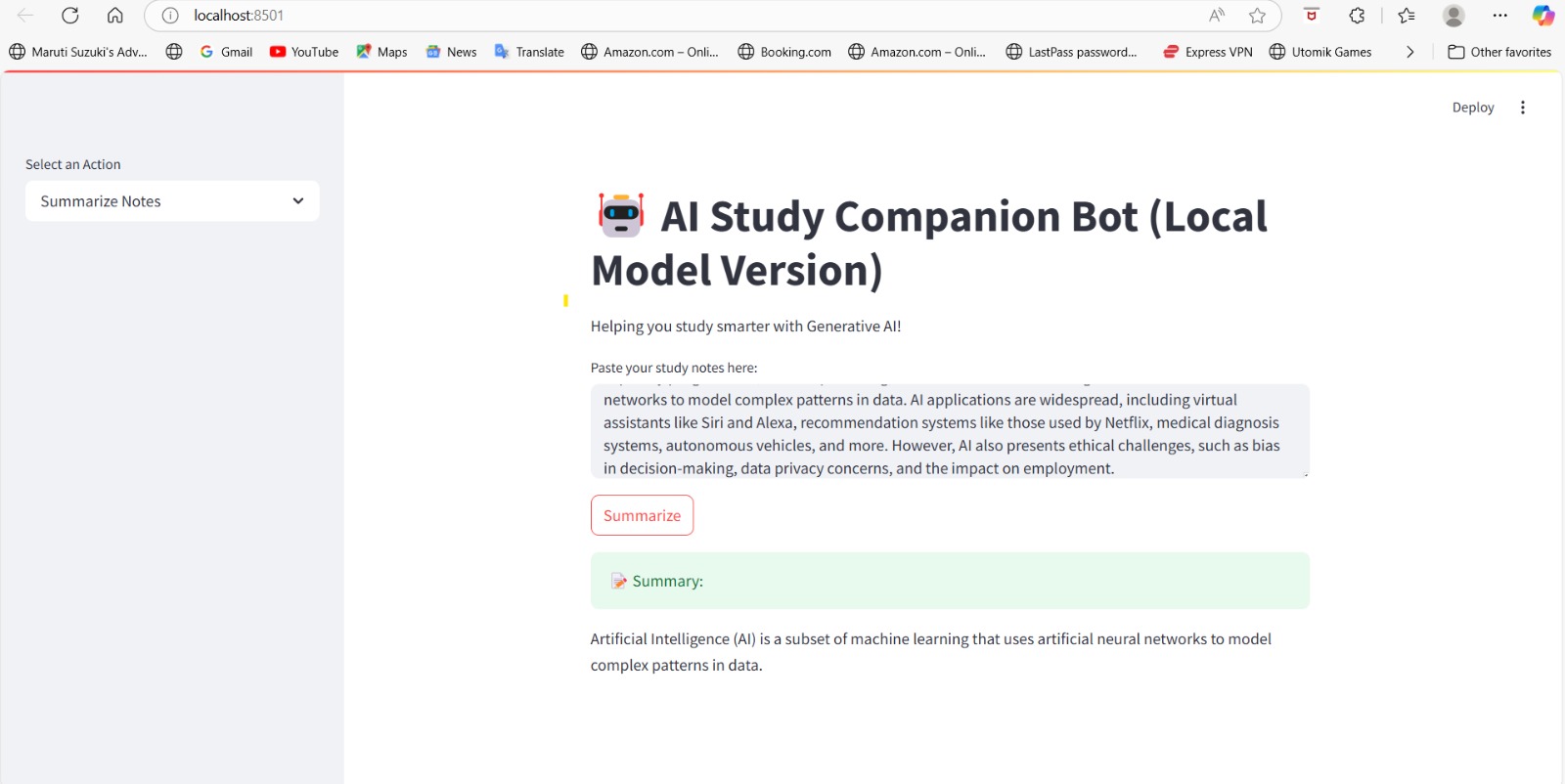
**Generate Flashcards:**

We chose the "Generate Flashcards" tool and put "reinforcement learning" as the subject in this screenshot. Clicking "Generate Flashcards," the bot created a flashcard outlining reinforcement learning. This shows how fast AI-powered study aids for difficult subjects let students generate them.



**Summary Notes:**

We pasted thorough notes on several forms of machine learning using the "Summary Notes" tool in this screenshot. The bot produced a brief summary after hitting the "Summary" button, proving how fast our approach lets students grasp and simplify difficult study content.



**Quiz:**

We entered "biology" as the topic using the "Create a Quiz" tool in this screenshot. Following a "Create Quiz" click, the bot produced a multiple-choice example quiz question. This shows how our technology might create practice tests on demand, allowing students to quickly assess their grasp of any topic.

