AI Integration into ‘Capture the Flag’

Introduction

Outlined here will be the methods and reasoning behind the AI requirements of the module COMP2003. This will detail the AI model used along with how it fits the functions outlined in our project proposal

Hosting

This section of our solution will be completed using ‘Ollama’. This program allows you to run and interact with larger AI models locally. This allows you to either use these models inside the application itself or interact with it via an API. Due to the models being run locally, this means data is kept private

In this case, we are using Ollama to run the model that connects to the backend of our website via its API using Node. The frontend of our website then communicates to this backend to allow for the user input a prompt and have the AI’s response sent back and displayed.

Currently, Ollama is hosted on the HTTP API 127.0.0.1:11434. This should be kept in consideration when connecting to the backend.

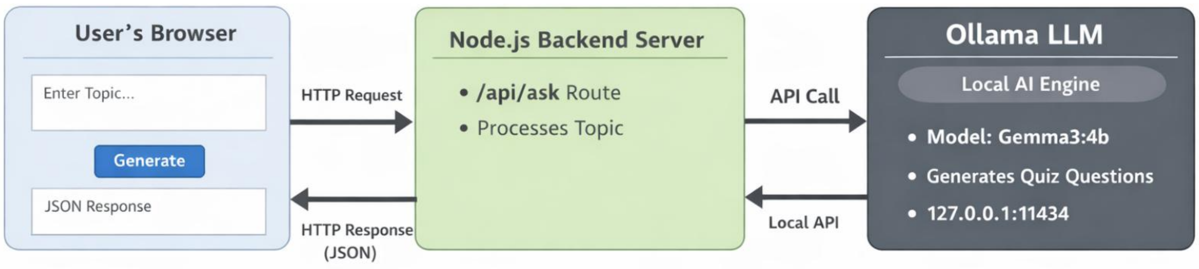

Figure 1 – AI website structure

AI model

I am currently using googles Gemma3. Gemma is a lightweight set of models built on Gemini technology. This can handle text and image processing – in over 140 native languages. This makes our website more available to other users across the globe (should it be shipped to do so). All model sizes excel in tasks like question answering, summarising and reasoning. It does all this using a very compact structure meaning it can run on a wide range of devices.

After some testing with Gemma, I ultimately decided that Gemma3-4b would be the most suitable. This is the mid-range solution of their models. I attempted to use Gemma3-1b initially but found its responses to be very slightly unsuitable due to a slight bit of inconsistency. These responses were almost perfect given the size of the model and could be suitable for running locally on almost any devices. I have found Gemma3-4b to be the best of the options – giving great, consistent responses whilst remaining on a small scale and providing very quick responses. Anything bigger than 4b would be insignificant due to the nature of the prompts it is needed for.

Prompts

This model is going to serve one very specific purpose. Generating questions and answers. To make results consistent and user friendly, a generic prompt 'template' will be used (within the backend of the website structure). Here, a user (in this case, a teacher) will simply input a topic that they would like to answer questions about and press generate. This will then produce a given number of multiple choice questions on said topic.

The output of this prompt is given in a JSON format so that these questions can be easily managed by the larger server structure and be saved on the user's device (locally) to be used at a later date.

Currently, the prompt is as shows:

```
const basePrompt = `
You are a teacher providing multiple choice questions to students.
Generate 8 questions on the following topic, along with:
- one correct answer
- three incorrect answers
Return output in valid JSON ONLY.

Topic: ${topic}
  `.trim();
```

This will need refining in the future to allow for a custom number of questions count and also proper formatting of the JSON structure for portability into the question structure in the greater website.