AI Assistant development

Abstract :

This report details the "AI Assistant Development" project, focusing on designing and implementing a basic AI Assistant. The project's objective was to showcase proficiency in crafting effective prompts and understanding how AI models respond to varied inputs. The assistant, built with a Flask backend and integrated with Google's Gemini 1.5 Flash API, performs tasks such as answering questions, summarizing text, generating creative content, and providing advice. A user-friendly web interface allows for task selection and query input, incorporating a feedback mechanism for continuous improvement. The project demonstrates the practical application of prompt engineering principles in creating a functional and responsive AI tool.

Project Overview:

Objective: The primary objective of this project was to design and implement a basic AI Assistant capable of performing various tasks based on user prompts. This endeavor aimed to demonstrate proficiency in crafting effective prompts and comprehending how different prompts influence AI model responses.

Project Requirements: The AI Assistant was developed to meet the following key requirements:

- **Functionality:** The assistant performs at least three distinct functions, including answering factual questions, summarizing given text, generating creative content (stories, poems, essays), and providing advice on specific topics.
- **Prompt Design:** For each function, at least three different prompts were designed to effectively guide the AI in generating desired outputs, varying in length, specificity, tone, style, complexity, and context.
- **User Interaction:** A user-friendly web interface was implemented, allowing users to input queries, select functions, and clearly view responses.
- **Feedback Loop:** A mechanism was included for users to provide feedback on responses with the aim of using this feedback to refine prompt design.

System Design and Implementation

The AI Assistant comprises a frontend web interface and a backend Flask application integrated with Google's Gemini API.

❖ **Frontend (User Interface)**

The frontend is built using HTML, CSS (Bootstrap for styling), and JavaScript (jQuery for AJAX calls).

- ✓ **Layout:** The interface is a chat-like structure with a header, a dynamic chat body to display messages, and a footer for input.
- ✓ **User Input:** Users interact via a text input field and a dropdown selector for choosing the assistant's function (Ask a Question, Summarize Text, Generate Creative Content, Get Advice). A "Send" button triggers the interaction.
- ✓ **Message Display:** User messages are displayed on the right with a blue background, and bot responses on the left with a light grey background.
- ✓ **Feedback Mechanism:** After each bot response, feedback buttons ("Yes"/"No") are appended, allowing users to rate the helpfulness of the response.

❖ **Backend :**

The backend, developed with Flask and integrated with the Google Generative AI library, manages the AI Assistant's core logic. The

GOOGLE_API_KEY is loaded to authenticate requests to the Gemini API, and the gemini-1.5-flash model is initialized to maintain conversation history.

Prompts are dynamically constructed based on the user-selected function type. Specific prompt templates are used for:

- ✓ **Asking a Question:** "Answer the following question clearly and concisely: {user_input}".
- ✓ **Summarizing Text:** "Summarize the following text, highlighting the main points: {user_input}".
- ✓ **Generating Creative Content:** "Generate creative content based on this idea: {user_input}. For example, write a short story, a poem, or a short essay.".
- ✓ **Providing Advice:** "Provide helpful advice on the following topic: {user_input}".

❖ **Challenges and Solutions**

During development, key challenges included:

- ✓ **Frontend-Backend Communication:** Ensuring smooth data flow between the JavaScript frontend and the Flask backend using AJAX requests and JSON payloads.

- ✓ **Effective Prompt Crafting:** Experimenting with various prompt formulations to consistently achieve desired outputs from the Gemini model for each function.
- ✓ **API Integration and Error Handling:** Properly configuring the Google Generative AI API and implementing basic error messages for failed responses.

These challenges were addressed through structured coding practices, careful prompt iteration, and logging to debug interactions between components.

Project Code :-

Frontend :

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>AI Assistant</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH
" crossorigin="anonymous">
   <style>
      body, html {
         height: 100%;
         margin: 0;
         padding: 0;
         font-family: Arial, sans-serif;
         background-color: #f1f1f1;
      }
      .chat {
         display: flex;
         flex-direction: column;
         height: 100vh;
         max-width: 600px;
         margin: auto;
         box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
         background-color: #ffffff;
      }
      .chat-header {
         padding: 15px;
         background-color: #007bff;
         color: white;
         text-align: center;
      }
```

```
.chat-body {
    flex-grow: 1;
    padding: 15px;
    overflow-y: auto;
    background-color: #f8f9fa;
    display: flex;
    flex-direction: column;
}
.chat-message {
    margin: 10px 0;
    padding: 10px;
    border-radius: 5px;
    width: fit-content;
    max-width: 80%;
}
.chat-message.user {
    background-color: #007bff;
    color: white;
    align-self: flex-end;
}
.chat-message.bot {
    background-color: #e2e3e5;
    color: black;
    align-self: flex-start;
}
.chat-footer {
    padding: 15px;
    background-color: #f8f9fa;
    display: flex;
    flex-wrap: wrap; /* Allows items to wrap to the next line */
    align-items: center;
    border-top: 1px solid #dee2e6;
    /* New: Align items to the end of the main axis when wrapped */
    justify-content: flex-end;
}
.chat-footer select, .chat-footer input {
    flex-grow: 1;
    margin-right: 10px;
    margin-bottom: 10px; /* Space between input/select and button */
}
.chat-footer button {
    margin-bottom: 10px; /* Space for the send button */
}
.feedback-section {
    margin-top: 5px;
```

```
            font-size: 0.9em;
            color: #555;
            display: flex;
            align-items: center;
        }
        .feedback-section button {
            margin-left: 5px;
            padding: 2px 8px;
            font-size: 0.8em;
        }
    </style>
</head>
<body>
    <div class="chat">
        <div class="chat-header">
            Sirisha's AI Assistant
        </div>
        <div class="chat-body" id="chat-body"></div>
        <div class="chat-footer">
            <select class="form-select mb-2" id="function-selector">
                <option value="Youtube" selected>Ask a Question</option>
                <option value="summarize_text">Summarize Text</option>
                <option value="creative_content">Generate Creative Content</option>
                <option value="provide_advice">Get Advice</option>
            </select>
            <input type="text" class="form-control" id="message-input" placeholder="Type
your query...">
            <button class="btn btn-primary" id="send-button">Send</button>
        </div>
    </div>

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>

    <script>
        $(document).ready(function() {
            let lastUserMessage = "";
            let lastBotResponse = "";

            function sendMessage() {
                var userMessage = $('#message-input').val().trim();
```

```
            var functionType = $('#function-selector').val();

            if (userMessage) {
                // Store for feedback
                lastUserMessage = userMessage;

                // Append user message to chat body
                $('#chat-body').append('<div class="chat-message user">' + userMessage +
'</div>');
                // Clear input field
                $('#message-input').val("");
                // Scroll to the bottom of the chat
                $('#chat-body').scrollTop($('#chat-body')[0].scrollHeight);

                // Send message to Flask backend
                $.ajax({
                    url: '/chat',
                    type: 'POST',
                    contentType: 'application/json',
                    data: JSON.stringify({
                        "message": userMessage,
                        "function_type": functionType
                    }),
                    success: function(response) {
                        var botResponse = response.response;
                        lastBotResponse = botResponse; // Store for feedback

                        // Append bot message to chat body
                        var botMessageDiv = $('<div class="chat-message
bot"></div>').text(botResponse);
                        $('#chat-body').append(botMessageDiv);

                        // Add feedback buttons
                        var feedbackSection = $('<div class="feedback-section">Was this helpful?
' +
                            '<button class="btn btn-sm btn-outline-success feedback-btn" data-
feedback="yes">Yes</button>' +
                            '<button class="btn btn-sm btn-outline-danger feedback-btn" data-
feedback="no">No</button>' +
                            '</div>');
                        botMessageDiv.append(feedbackSection);

                        // Attach click handlers to feedback buttons
                        feedbackSection.find('.feedback-btn').on('click', function() {
                            var feedback = $(this).data('feedback');
```

```
                    sendFeedback(feedback, lastUserMessage, lastBotResponse);
                    // Disable buttons after feedback
                    $(this).parent().html('Thanks for your feedback!');
                });

                // Scroll to the bottom of the chat after bot response
                $('#chat-body').scrollTop($('#chat-body')[0].scrollHeight);
            },
            error: function(xhr, status, error) {
                console.error("Error communicating with backend:", error);
                $('#chat-body').append('<div class="chat-message bot"
style="background-color: #dc3545; color: white;">Error: Could not get response.</div>');
                $('#chat-body').scrollTop($('#chat-body')[0].scrollHeight);
            }
        });
    }
}

function sendFeedback(feedback, userMsg, botResp) {
    $.ajax({
        url: '/feedback',
        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify({
            "feedback": feedback,
            "user_message": userMsg,
            "bot_response": botResp
        }),
        success: function(response) {
            console.log("Feedback sent:", response.status);
        },
        error: function(xhr, status, error) {
            console.error("Error sending feedback:", error);
        }
    });
}

// Event listener for send button click
$('#send-button').click(function() {
    sendMessage();
});

// Event listener for Enter key press on message input
$('#message-input').keypress(function(e) {
    if (e.which == 13) { // 13 is the keycode for Enter
```

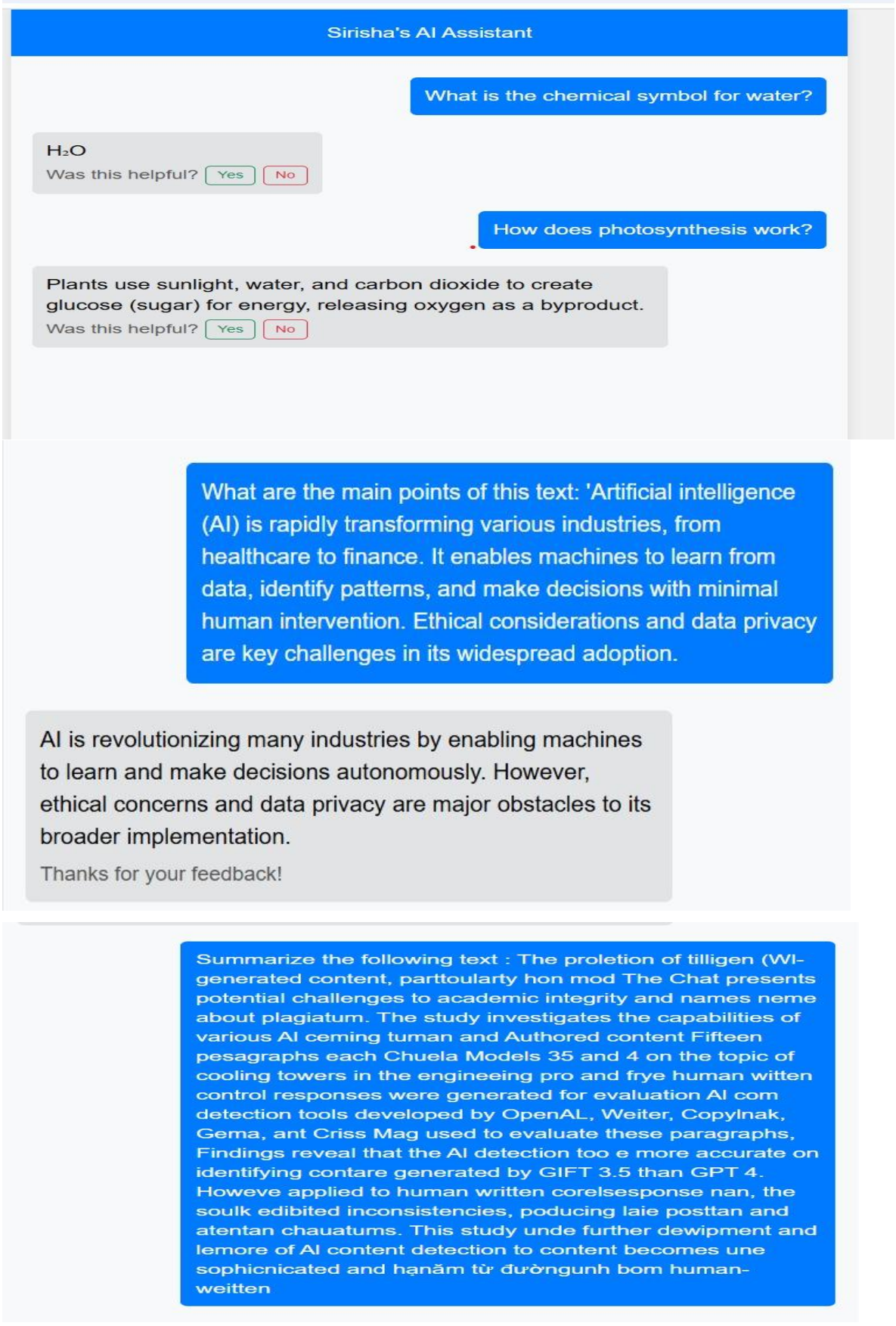
```
                sendMessage();
            }
        });
    });
  </script>
</body>
</html>
```

Backend :

```
from flask import Flask, render_template, request, jsonify
import google.generativeai as genai
import os

GOOGLE_API_KEY = "AIzaSyCa-YnjBmkSKJywMhyduIOujIck7IK3MKs"
if not GOOGLE_API_KEY:
    raise ValueError("GOOGLE_API_KEY is not set. Please provide your API key or
ensure it's in environment variables.")

try:
    genai.configure(api_key=GOOGLE_API_KEY)
        model = genai.GenerativeModel('gemini-1.5-flash')

    chat = model.start_chat(history=[])
    print("Gemini model and chat session configured successfully!")
except Exception as e:
    print(f"Error during Gemini API configuration or model initialization: {e}")

app = Flask(__name__)

@app.route('/')
def index():

    return render_template('index.html')

@app.route('/chat', methods=['POST'])
def chat_response():

        user_input = request.json.get('message')
    function_type = request.json.get('function_type')

    if not user_input or not function_type:
        print("Error: Missing message or function type in the request.")
        return jsonify({"error": "Missing message or function type"}), 400
       prompt = ""
    if function_type == "Youtube":
```

```
            prompt = f"Answer the following question clearly and concisely: {user_input}"
    elif function_type == "summarize_text":

        prompt = f"Summarize the following text, highlighting the main points:
{user_input}"
    elif function_type == "creative_content":

        prompt = f"Generate creative content based on this idea: {user_input}. For example,
write a short story, a poem, or a short essay."
    elif function_type == "provide_advice":
            prompt = f"Provide helpful advice on the following topic: {user_input}"
    else:
        return jsonify({"error": "Invalid function type selected"}), 400

    try:
        print(f"Received user message for {function_type}: '{user_input}'")
        print(f"Sending prompt to Gemini: '{prompt}'")

        response = chat.send_message(prompt).text
        print(f"Generated Gemini response: '{response[:100]}...'") # Log first 100 chars

            return jsonify({"response": response})
    except Exception as e:
            print(f"Error during Gemini API call: {e}")
            return jsonify({"error": str(e)}), 500

@app.route('/feedback', methods=['POST'])
def receive_feedback():

    data = request.json
    message_id = data.get('message_id')
    feedback = data.get('feedback') # 'yes' or 'no'
    user_message = data.get('user_message')
    bot_response = data.get('bot_response')
    print(f"--- FEEDBACK RECEIVED ---")
    print(f"User Message: '{user_message}'")
    print(f"Bot Response: '{bot_response}'")
    print(f"Feedback: {feedback.upper()} (Helpful)")
    print(f"-------------------------")

    return jsonify({"status": "Feedback received!"}), 200

if __name__ == '__main__':

    app.run(debug=True)
```

Output:


Sirisha's AI Assistant
What is the chemical symbol for water?
H₂O
Was this helpful? Yes No
How does photosynthesis work?
Plants use sunlight, water, and carbon dioxide to create glucose (sugar) for energy, releasing oxygen as a byproduct.
Was this helpful? Yes No
What are the main points of this text: 'Artificial intelligence (AI) is rapidly transforming various industries, from healthcare to finance. It enables machines to learn from data, identify patterns, and make decisions with minimal human intervention. Ethical considerations and data privacy are key challenges in its widespread adoption.
AI is revolutionizing many industries by enabling machines to learn and make decisions autonomously. However, ethical concerns and data privacy are major obstacles to its broader implementation.
Thanks for your feedback!
Summarize the following text : The proletion of tilligen (WI-generated content, parttoularty hon mod The Chat presents potential challenges to academic integrity and names neme about plagiatum. The study investigates the capabilities of various Al ceming tuman and Authored content Fifteen pesagraphs each Chuela Models 35 and 4 on the topic of cooling towers in the engineeing pro and frye human witten control responses were generated for evaluation Al com detection tools developed by OpenAL, Weiter, Copylnak, Gema, ant Criss Mag used to evaluate these paragraphs, Findings reveal that the Al detection too e more accurate on identifying contare generated by GIFT 3.5 than GPT 4. Howeve applied to human written corelsesponse nan, the soulk edibited inconsistencies, poducing laie posttan and atentan chauatums. This study unde further dewipment and lemore of Al content detection to content becomes une sophicnicated and hạnăm từ đườngunh bom human-weitten

This study examined the ability of AI detection tools to distinguish between AI-generated (GPT-3.5 and GPT-4) and human-written text on the topic of cooling towers. Results showed that AI detectors were more accurate at identifying GPT-3.5 generated content than GPT-4 content. Furthermore, the tools produced false positives when analyzing human-written text, highlighting the need for improved AI detection technology as AI-generated content becomes more sophisticated.

Thanks for your feedback!

Write a short poem about the ocean.

The ocean sighs, a turquoise breath, On shores of sand, a whispered death Of sun-bleached shells and secrets deep, Where coral castles softly sleep. A million blues, a shimmering haze, Reflecting skies in sunlit days, Then turns to steel, a stormy frown, As waves crash down, and mountains drown. Its heart, a mystery untold, Of creatures strange, both brave and bold, A silent song, a timeless grace, The ocean's vast and boundless space.

Was this helpful? Yes No

Generate a paragraph for an essay about the importance of reading

Reading is not merely the decoding of words on a page; it is a portal to empathy, understanding, and intellectual growth. Each book we consume expands our worldview, introducing us to new perspectives, cultures, and historical contexts that would otherwise remain inaccessible. Through the act of reading, we cultivate critical thinking skills, sharpening our ability to analyze information, form opinions, and engage in nuanced discussions. Furthermore, the immersive nature of reading fosters imagination and creativity, allowing us to escape the mundane and explore uncharted territories of the mind, enriching not only our intellect but also our emotional landscape. In a world increasingly reliant on fleeting digital interactions, the sustained engagement demanded by reading offers a crucial counterpoint, fostering deeper thought and a richer inner life.

Thanks for your feedback!

How can I improve my time management for academic tasks?

Improving academic time management requires a multi-pronged approach: **1. Planning & Prioritization:** * **Create a Master Schedule:** Use a planner, calendar app, or even a whiteboard to map out your semester, including deadlines, exams, and major assignments. Break down large assignments into smaller, manageable tasks. * **Prioritize Tasks:** Use methods like the Eisenhower Matrix (urgent/important) to focus on the most crucial tasks first. Don't get bogged down in less important details until the essentials are complete. * **Set Realistic Goals:** Don't try to do everything at once. Set achievable daily and weekly goals. Over-scheduling leads to burnout and inefficiency. * **Time Blocking:** Allocate specific time slots for studying particular subjects or completing specific tasks. Treat these blocks like appointments you can't miss. **2. Effective Study Habits:** * **Find Your Peak Performance Times:** Are you a morning person or a night owl?

How can I improve my communication skills?

Improving communication skills is a journey, not a destination. It requires consistent effort and self-awareness. Here's advice broken down into key areas: **1. Active Listening:** * **Focus on understanding:** Truly listen to what the other person is saying, both verbally and nonverbally. Avoid interrupting or formulating your response while they're still speaking. * **Ask clarifying questions:** Show your engagement by asking questions to ensure you understand their message fully. "So, if I understand correctly..." or "Can you elaborate on...?" are helpful phrases. * **Summarize and paraphrase:** Restate what you've heard to confirm your understanding and show the speaker you're paying attention. * **Observe nonverbal cues:** Pay attention to body language, tone of voice, and facial expressions to understand the unspoken message. **2. Clear and Concise Speaking:** * **Structure your thoughts:** Before speaking, take a moment to organize

Get Advice

Type your query...

Send