**Project**

**PROBLEM STATEMENT: Coding Chatbot GUI using Java**  
Develop a desktop-based chatbot application with a graphical user interface (GUI) that assists users with coding-related queries. The chatbot integrates with an external API (Google Gemini) to provide concise answers, including code examples where applicable.

**SOFTWARE AND HARDWARE REQUIREMENTS:**

* **Software:**
  1. **Frontend:** Java Swing (for GUI development).
  2. **Backend:** Java (core logic and API integration).
  3. **External API:** Google Gemini API for natural language processing and response generation.
  4. **Development Environment:** Any Java IDE (e.g., IntelliJ IDEA, Eclipse).
* **Hardware:**
  1. **Storage:** Local storage for the application executable and temporary data.
  2. **Devices:** Desktop or laptop computer with internet connectivity.

**THEORY:**

The **Coding Chatbot GUI** is a desktop application designed to assist programmers by answering coding-related questions in real time. Built using Java Swing for the user interface, it features a text area for chat display, an input field for user queries, and a "Send" button to submit questions. The application connects to the Google Gemini API to fetch responses, which are parsed and displayed with proper formatting (e.g., code blocks). This tool is ideal for beginners and experienced developers seeking quick coding assistance without leaving their development environment.

**CODES:**

Below is the complete Java code for the project:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.HttpURLConnection;

import java.net.URL;

import java.nio.charset.StandardCharsets;

public class GeminiCodingChatbotGUI {

private static final String API\_KEY = "ENTER\_ACTUAL\_API\_KEY"; // Replacewith your actual API key

private static final String API\_URL = "https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash:generateContent?key=" + API\_KEY;

private static final String SYSTEM\_PROMPT = "You are a coding assistant. Answer concisely with code examples where applicable.";

private JFrame frame;

private JTextArea chatArea;

private JTextField inputField;

private JButton sendButton;

public GeminiCodingChatbotGUI() {

frame = new JFrame("Coding Chatbot");

chatArea = new JTextArea(15, 40);

inputField = new JTextField(30);

sendButton = new JButton("Send");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new BorderLayout());

chatArea.setEditable(false);

chatArea.setLineWrap(true);

chatArea.setWrapStyleWord(true);

chatArea.setText("Welcome! Ask me coding questions (e.g., 'How do I write a for loop in Java?')\n");

frame.add(new JScrollPane(chatArea), BorderLayout.CENTER);

JPanel inputPanel = new JPanel();

inputPanel.add(new JLabel("You: "));

inputPanel.add(inputField);

inputPanel.add(sendButton);

frame.add(inputPanel, BorderLayout.SOUTH);

sendButton.addActionListener(e -> sendMessage());

inputField.addActionListener(e -> sendMessage());

frame.pack();

frame.setLocationRelativeTo(null);

frame.setVisible(true);

}

private void sendMessage() {

String userInput = inputField.getText().trim();

if (userInput.isEmpty()) return;

chatArea.append("You: " + userInput + "\n");

inputField.setText("");

String response = getGeminiResponse(userInput);

chatArea.append("Bot: " + response + "\n\n");

chatArea.setCaretPosition(chatArea.getDocument().getLength());

}

private String getGeminiResponse(String userInput) {

try {

String jsonPayload = "{\"contents\": [{\"parts\": [{\"text\": \"" + SYSTEM\_PROMPT + " " + userInput.replace("\"", "\\\"") + "\"}]}]}";

URL url = new URL(API\_URL);

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("POST");

conn.setRequestProperty("Content-Type", "application/json");

conn.setDoOutput(true);

try (OutputStream os = conn.getOutputStream()) {

byte[] input = jsonPayload.getBytes(StandardCharsets.UTF\_8);

os.write(input, 0, input.length);

}

int responseCode = conn.getResponseCode();

if (responseCode != HttpURLConnection.HTTP\_OK) {

String errorMsg = readErrorResponse(conn);

return "Error: HTTP " + responseCode + " - " + errorMsg;

}

StringBuilder response = new StringBuilder();

try (BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream(), StandardCharsets.UTF\_8))) {

String line;

while ((line = br.readLine()) != null) {

response.append(line);

}

}

return parseResponse(response.toString());

} catch (IOException e) {

return "Error: " + e.getMessage();

}

}

private String readErrorResponse(HttpURLConnection conn) {

try (BufferedReader br = new BufferedReader(new InputStreamReader(conn.getErrorStream(), StandardCharsets.UTF\_8))) {

StringBuilder error = new StringBuilder();

String line;

while ((line = br.readLine()) != null) {

error.append(line);

}

return error.toString();

} catch (IOException e) {

return "Unable to read error details";

}

}

private String parseResponse(String jsonResponse) {

try {

String textMarker = "\"text\": \"";

int textStart = jsonResponse.indexOf(textMarker);

if (textStart == -1) {

return "Error: No text field found\nRaw response: " + jsonResponse;

}

textStart += textMarker.length();

int textEnd = jsonResponse.indexOf("\"", textStart);

while (textEnd > textStart && jsonResponse.charAt(textEnd - 1) == '\\') {

textEnd = jsonResponse.indexOf("\"", textEnd + 1);

}

if (textEnd == -1) {

return "Error: Malformed text field\nRaw response: " + jsonResponse;

}

String text = jsonResponse.substring(textStart, textEnd)

.replace("\\n", "\n")

.replace("\\\"", "\"");

if (text.contains("```")) {

return formatCodeBlocks(text);

}

return text;

} catch (Exception e) {

return "Error parsing response: " + e.getMessage() + "\nRaw response: " + jsonResponse;

}

}

private String formatCodeBlocks(String text) {

String[] parts = text.split("```");

StringBuilder formatted = new StringBuilder();

boolean inCode = false;

for (String part : parts) {

if (inCode) {

formatted.append("\n[Code]\n").append(part.trim()).append("\n[End Code]\n");

} else {

formatted.append(part);

}

inCode = !inCode;

}

return formatted.toString();

}

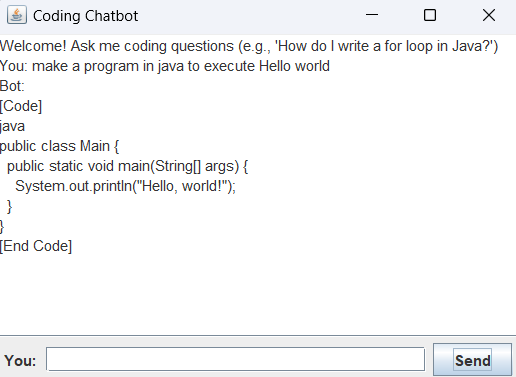
public static void main(String[] args) {

SwingUtilities.invokeLater(GeminiCodingChatbotGUI::new);

}

}

**SCREENSHOTS OF OUTPUT:**



**ADVANTAGES AND DISADVANTAGES:**

* **Advantages:**
  1. **Time-Saving:** Provides instant coding assistance without needing to search online.
  2. **Code Examples:** Delivers practical, concise code snippets for quick learning.
  3. **User-Friendly GUI:** Simple interface suitable for all skill levels.
  4. **Real-Time Interaction:** Immediate responses enhance productivity.
  5. **Offline Potential:** Could be adapted to work locally with cached responses.
* **Disadvantages:**
  1. **Internet Dependency:** Requires a stable connection to access the API.
  2. **API Costs:** Usage may incur costs depending on API limits.
  3. **Error Handling:** Limited robustness in parsing complex API responses.
  4. **Hardware Dependency:** Requires a computer with sufficient resources for Java runtime.
  5. **Security:** API key exposure in code poses a risk if not managed properly.

**FUTURE SCOPE:**

The **future scope** of the Coding Chatbot GUI can include:

1. **Local Database:** Store frequently asked questions and responses for offline use.
2. **Multi-Language Support:** Extend assistance to other programming languages beyond Java.
3. **Syntax Highlighting:** Enhance code display with colored syntax in the GUI.
4. **Voice Input:** Integrate voice recognition for hands-free operation.
5. **Cross-Platform:** Convert to a web or mobile app using frameworks like JavaFX or Android SDK.
6. **Customizable UI:** Allow users to adjust themes and layouts for better usability.

**CONCLUSION:**

The Coding Chatbot GUI offers a practical solution for developers seeking quick coding help. By leveraging Java Swing for the interface and the Gemini API for intelligent responses, it delivers a seamless experience. While currently limited by internet dependency and basic error handling, future enhancements like offline mode and advanced features can make it a robust tool for coders worldwide.