Rayat Shikshan Sanstha's

Karmaveer Bhaurao Patil College of Engineering, Satara

Project Based Learning

Class: Second Year **Div:** A(Pearl)

Semester: Fourth

Academic Year: 2024-25

Course with code: Python Programming (BTCOL406)

PBL Topic Name: Smart Calculator - Using voice input

Group Members:

PRN	Full Name
23062701242053	Prem Rajendra Rajput
23062701242062	Palak Bharat Shah
23062701242068	Omkar Rajendra Mane

Faculty Name and Designation: Ms. Priyanka Salunkhe-(Assistant Professor)

Project Title: Smart Calculator - Using Voice Input

Objective:

- Voice-Enabled User Input
- Accurate Mathematical Evaluation
- Voice-Based Output
- History Tracking with Database

Theory:

1. Imports: Required Modules:

- Flask: Web framework to create the app.
- render_template: Loads the HTML file (like index.html).
- request: To get data sent from the user (form inputs).
- send_file: Used to send the generated audio file back.
- speech_recognition: (Optional here) used for converting speech to text.
- `gTTS`: Google Text-to-Speech, converts text to audio.
- `BytesIO`: Creates an in-memory file (like a temporary MP3).
- `mysql.connector`: Connects Python to your MySQL database.
- `datetime`: Handles date and time for logging history.
- `math`: Gives access to functions like `sin`, `cos`, `sqrt`, etc.

2. Flask App Setup:

- This creates your Flask app instance.

3. MySQL Database Connection:

- Connects to MySQL database calculator_db using root user.
- cursor is used to execute SQL queries like insert, select, etc.

4. Degree-Based Trigonometry:

- Converts degree to radians before applying sin().
- Converts degree to radians before applying cos().
- Converts degree to radians before applying tan().

5. **Allowed Math Functions (for Safe Eval):**

- Collects safe functions from math module (ignores private stuff).
- Adds more functions manually (like `sin`, `pi`, `round`) including the degree-based trig functions.

6. Preprocess Voice Expression:

- Converts spoken math like:
  - "five plus six" → "5 + 6"
  - "square root" → "sqrt"
- Replaces words with actual math symbols so eval() can understand.

7. Evaluate Safely:

- eval() runs the math expression safely using only allowed functions.
- Prevents dangerous code like os.system('rm -rf /')

8. Text to Speech Route /speak:

- Route that takes text and returns spoken audio using gTTS.
- Used to speak result aloud in your frontend.
- If no text is provided, it gives a 400 (bad request) error.

9. Main Route /:

- The homepage of your web app.
- Handles both form submission (POST) and normal load (GET).

10. POST: Evaluate Expression and Save:

- Gets input from user form.
- Prepares and safely evaluates the math expression.
- Converts result to string and rounds float values.
- Stores the input, result, and timestamp into the MySQL history table.

11. Error Handling:

- If expression is invalid (like 5 **), catches and shows the error.

12. Fetch Recent History:

- Loads the last 5 calculations.
- Formats each history row into a dictionary.

13. Render HTML Page:

- Sends all result and history data to your HTML page to display.

14. Run the Flask App:

- Runs the app in development mode with auto-reload on code changes.

Source code:
1] Frontend (Index.html)

```
<!DOCTYPE html>
<head>
  <title>Smart Scientific Calculator</title>
  <style>
    * {
     box-sizing: border-box;
    }
   body {
     margin: 0;
     font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
     background: linear-gradient(145deg, #1e1e2f, #252545);
     color: #fff;
     display: flex;
     justify-content: center;
     align-items: flex-start;
     min-height: 100vh;
     padding: 2rem;
    }



    .container {
     max-width: 900px;
     width: 100%;
     background: #2e2e44;
     border-radius: 15px;
     box-shadow: 0 0 40px rgba(0, 0, 0, 0.3);
     display: flex;
     overflow: hidden;
     flex-wrap: wrap;
    }

    /* Left panel: calculator buttons */
    .calc-panel {
     flex: 1 1 350px;
     background: #3a3a5c;
     padding: 1.5rem;
     display: grid;
     grid-template-columns: repeat(5, 1fr);
     gap: 12px;
     border-right: 2px solid #00bcd4;
    }
    button.calc-btn {
     background: #44476b;
     border: none;
     border-radius: 10px;
     color: white;
     font-size: 1.15rem;
     padding: 15px 0;
     cursor: pointer;
     transition: background-color 0.3s ease;
```

```
  user-select: none;
}

button.calc-btn.operator {
  background: #00bcd4;
  color: #121212;
  font-weight: 700;
}

button.calc-btn:hover {
  background-color: #00acc1;
}

/* Right panel: voice, results, history */
.control-panel {
  flex: 1 1 350px;
  padding: 1.5rem 2rem;
  display: flex;
  flex-direction: column;
  justify-content: flex-start;
}



h1 {
  margin: 0 0 1rem 0;
  font-weight: 700;
  font-size: 2rem;
  color: #00bcd4;
  user-select: none;
}

form {
  margin-bottom: 1rem;
}

button#voiceBtn {
  width: 100%;
  background-color: #00bcd4;
  border: none;
  color: #121212;
  font-weight: 700;
  font-size: 1.1rem;
  padding: 15px;
  border-radius: 10px;
  cursor: pointer;
  transition: background-color 0.3s ease;
  margin-bottom: 1rem;
}

button#voiceBtn:hover {
  background-color: #0097a7;
}
```

```
.toggle {
  display: flex;
  align-items: center;
  margin-bottom: 1.5rem;
  user-select: none;
}

.toggle label {
  margin-right: 10px;
  font-weight: 600;
  font-size: 1rem;
}
.toggle input[type="checkbox"] {
  display: none;
}
.slider {
  width: 50px;
  height: 25px;
  background-color: #ccc;
  border-radius: 25px;
  position: relative;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.slider::before {
  content: "";
  position: absolute;
  width: 21px;
  height: 21px;
  border-radius: 50%;
  background-color: #fff;
  top: 2px;
  left: 2px;
  transition: 0.3s;
}

input:checked + .slider {
  background-color: #00bcd4;
}

input:checked + .slider::before {
  transform: translateX(25px);
}

.output {
  background: #202035;
  padding: 1rem;
  border-radius: 10px;
  font-size: 1.2rem;
  min-height: 60px;
  user-select: text;
  margin-bottom: 1.5rem;
  word-wrap: break-word;
```

```
}

/* History table */
.history {
  background: #202035;
  padding: 1rem;
  border-radius: 10px;
  max-height: 200px;
  overflow-y: auto;
  font-size: 0.9rem;
  display: none;
}

.history h3 {
  margin-top: 0;
  margin-bottom: 0.8rem;
  font-weight: 600;
}


table {
  width: 100%;
  border-collapse: collapse;
  color: #fff;
}

table thead tr {
  border-bottom: 2px solid #00bcd4;
}

table th, table td {
  text-align: left;
  padding: 8px;
}

table tbody tr {
  border-bottom: 1px solid #44476b;
}

table tbody tr:last-child {
  border-bottom: none;
}

table td {
  max-width: 200px;
  white-space: nowrap;
  overflow: hidden;
  text-overflow: ellipsis;
}

/* Show History button */
#toggleHistoryBtn {
  background: #00bcd4;
  border: none;
```

```
      color: #121212;
      font-weight: 700;
      font-size: 1rem;
      padding: 10px 15px;
      border-radius: 10px;
      cursor: pointer;
      margin-bottom: 1rem;
      width: 150px;
      user-select: none;
    }

    #toggleHistoryBtn:hover {
      background-color: #0097a7;
    }

    /* Responsive */
    @media (max-width: 750px) {
      .container {
        flex-direction: column;
      }
      .calc-panel,
      .control-panel {
        flex: 1 1 100%;
        border-right: none;
        margin-bottom: 2rem;
      }
    }
  </style>
</head>
<body>
  <div class="container">
    <!-- Left: Calculator Buttons -->
    <div class="calc-panel" id="buttonPanel">
      <button class="calc-btn operator" data-value="(">(</button>
      <button class="calc-btn operator" data-value=")">)</button>
      <button class="calc-btn operator" data-value="pi">π</button>
      <button class="calc-btn operator" data-value="e">e</button>
      <button class="calc-btn operator" data-value="clear">C</button>

      <button class="calc-btn" data-value="7">7</button>
      <button class="calc-btn" data-value="8">8</button>
      <button class="calc-btn" data-value="9">9</button>
      <button class="calc-btn operator" data-value="/">÷</button>
      <button class="calc-btn operator" data-value="**">^</button>

      <button class="calc-btn" data-value="4">4</button>
      <button class="calc-btn" data-value="5">5</button>
      <button class="calc-btn" data-value="6">6</button>
      <button class="calc-btn operator" data-value="*">×</button>
      <button class="calc-btn operator" data-value="sqrt(">√</button>

      <button class="calc-btn" data-value="1">1</button>
      <button class="calc-btn" data-value="2">2</button>
      <button class="calc-btn" data-value="3">3</button>
```

```
  <button class="calc-btn operator" data-value="-">−</button>
  <button class="calc-btn operator" data-value="log10(">log</button>

  <button class="calc-btn" data-value="0">0</button>
  <button class="calc-btn" data-value=".">.</button>
  <button class="calc-btn operator" data-value="sin(">sin</button>
  <button class="calc-btn operator" data-value="cos(">cos</button>
  <button class="calc-btn operator" data-value="+">+</button>

  <button class="calc-btn operator" data-value="tan(">tan</button>
  <button class="calc-btn operator" data-value="log(">ln</button>
  <button class="calc-btn operator" data-value="abs(">abs</button>
  <button class="calc-btn operator" data-value="round(">round</button>
  <button class="calc-btn" data-value="=">=</button>
</div>

<!-- Right: Voice input, output, history -->
<div class="control-panel">
  <h1>🔊 Smart Scientific Calculator</h1>

  <form method="POST" id="voiceForm">
    <input type="hidden" id="expressionInput" name="expression" />
    <input type="hidden" id="voiceInputFlag" name="voice_input" value="false" />
    <button id="voiceBtn" type="button">🎙️ Speak Expression</button>

    <div class="toggle">
      <label for="voiceToggle">Voice Output</label>
      <input type="checkbox" id="voiceToggle" name="voice_enabled" checked />
      <label class="slider" for="voiceToggle"></label>
    </div>
  </form>

  <div class="output" id="outputArea" aria-live="polite">
    {% if error %}
      <div style="color: #ff6666;">❌ {{ error }}</div>
    {% else %}
      {{ result if result else "Results show here..." }}
    {% endif %}
  </div>

  <button id="toggleHistoryBtn">Show History</button>

  <div class="history" id="historySection">
    <h3>🕘 Recent History</h3>
    {% if history %}
    <table>
      <thead>
        <tr>
          <th>Expression</th>
          <th>Result</th>
          <th>Timestamp</th>
        </tr>
      </thead>
      <tbody>
```

```
          {% for h in history %}
          <tr>
            <td title="{{ h.expression }}">{{ h.expression }}</td>
            <td>{{ h.result }}</td>
            <td>{{ h.created_at }}</td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
      {% else %}
        <p>No history yet.</p>
      {% endif %}
    </div>
  </div>
</div>

<script>
  const voiceBtn = document.getElementById('voiceBtn');
  const expressionInput = document.getElementById('expressionInput');
  const voiceForm = document.getElementById('voiceForm');
  const voiceInputFlag = document.getElementById('voiceInputFlag');
  const outputArea = document.getElementById('outputArea');
  const voiceToggle = document.getElementById('voiceToggle');
  const buttonPanel = document.getElementById('buttonPanel');

  let expression = "";

  // Handle calculator button clicks
  buttonPanel.addEventListener('click', (e) => {
    if (e.target.classList.contains('calc-btn')) {
      const value = e.target.getAttribute('data-value');
      if (value === "clear") {
        expression = "";
      } else if (value === "=") {
        expressionInput.value = expression;
        voiceInputFlag.value = "false";
        voiceForm.submit();
        return;
      } else {
        expression += value;
      }
      outputArea.textContent = expression || "Results show here...";
    }
  });

  //  Handle keyboard input
  document.addEventListener("keydown", function (event) {
    const key = event.key;

    if (!isNaN(key) || "+-*/().".includes(key)) {
      expression += key;
    } else if (key === "Enter") {
      expressionInput.value = expression;
      voiceInputFlag.value = "false";
```

```
    voiceForm.submit();
    return;
  } else if (key === "Backspace") {
    expression = expression.slice(0, -1);
  } else {
    return;
  }

  outputArea.textContent = expression || "Results show here...";
});

// Voice Input
voiceBtn.addEventListener('click', () => {
  const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  if (!SpeechRecognition) {
    alert("Voice recognition is not supported in this browser. Please use Google Chrome.");
    return;
  }

  const recognition = new SpeechRecognition();
  recognition.lang = 'en-US';
  recognition.interimResults = false;
  recognition.maxAlternatives = 1;

  voiceBtn.disabled = true;
  voiceBtn.innerText = " 🎤 Listening...";

  recognition.start();

  recognition.onresult = (event) => {
    const transcript = event.results[0][0].transcript;
    expressionInput.value = transcript;
    voiceInputFlag.value = "true";
    voiceForm.submit();
  };

  recognition.onerror = (event) => {
    console.error("Voice error:", event.error);
    alert("Error: " + event.error + "\nPlease try again.");
  };

  recognition.onend = () => {
    voiceBtn.disabled = false;
    voiceBtn.innerText = " 🎙️ Speak Expression";
  };
});

// 🗣️ Speak result
window.addEventListener("DOMContentLoaded", () => {
  const resultText = outputArea.innerText.trim();
  if (resultText && resultText !== "Results show here..." && voiceToggle.checked) {
    const synth = window.speechSynthesis;
    const utter = new SpeechSynthesisUtterance(resultText);
    utter.lang = "en-US";
```

```
        synth.speak(utter);
      }
    });
  </script>
</body>
</html>
```

2] Backend (app.py)

```
from flask import Flask, render_template, request, send_file
import speech_recognition as sr
from gtts import gTTS
from io import BytesIO
import mysql.connector
import datetime
import math

app = Flask(__name__)

# MySQL DB connection
conn = mysql.connector.connect(
    host='localhost',
    user='root',
    password='WJ28@krhps',
    database='calculator_db'
)
cursor = conn.cursor()

# Degree-based trig functions
def sin_deg(x):
    return math.sin(math.radians(x))

def cos_deg(x):
    return math.cos(math.radians(x))

def tan_deg(x):
    return math.tan(math.radians(x))

# Allowed math functions with degree trig replacements
allowed_names = {k: v for k, v in math.__dict__.items() if not k.startswith("__")}
allowed_names.update({
    "abs": abs,
    "round": round,
    "pi": math.pi,
    "e": math.e,
    "sin": sin_deg,
    "cos": cos_deg,
    "tan": tan_deg,
})

# Preprocessing voice input
def preprocess(expression):
    expression = expression.lower()
    expression = expression.replace("plus", "+").replace("minus", "-")
```

```
    expression = expression.replace("into", "*").replace("times", "*")
    expression = expression.replace("divide", "/").replace("divided by", "/")
    expression = expression.replace("power", "**")
    expression = expression.replace("square root", "sqrt")
    expression = expression.replace("log", "log10").replace("ln", "log")
    return expression

# Safe evaluation
def safe_eval(expr):
    return eval(expr, {"__builtins__": None}, allowed_names)

# Route for speaking result
@app.route("/speak")
def speak_api():
    text = request.args.get("text", "")
    if not text:
        return "No text", 400
    tts = gTTS(text=text, lang='en')
    mp3_fp = BytesIO()
    tts.write_to_fp(mp3_fp)
    mp3_fp.seek(0)
    return send_file(mp3_fp, mimetype="audio/mp3")

# Main route
@app.route("/", methods=["GET", "POST"])
def index():
    result = expression = error = None
    history_data = []

    if request.method == "POST":
        expression = request.form.get("expression", "").strip()

        try:
            if expression:
                raw_expression = preprocess(expression)
                eval_result = safe_eval(raw_expression)

                # Convert result to string
                if isinstance(eval_result, float):
                    result = str(round(eval_result, 8))
                else:
                    result = str(eval_result)

                # Save to DB
                cursor.execute(
                    "INSERT INTO history (expression, result, created_at) VALUES (%s, %s, %s)",
                    (expression, result, datetime.datetime.now())
                )
                conn.commit()

        except Exception as e:
            error = f"Error: {str(e)}"
    # Load recent history
    cursor.execute("SELECT * FROM history ORDER BY created_at DESC LIMIT 5")
```

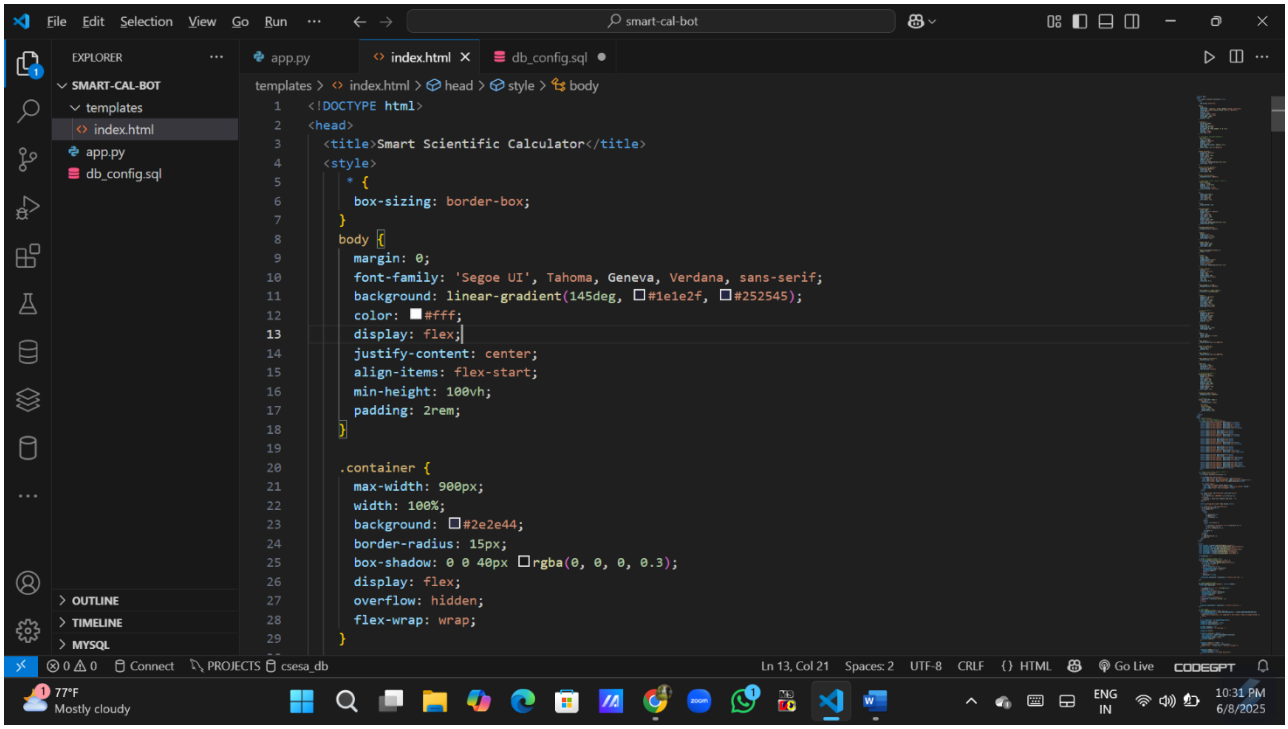
```
    rows = cursor.fetchall()
    for row in rows:
        history_data.append({
            'id': row[0],
            'expression': row[1],
            'result': row[2],
            'created_at': row[3].strftime("%Y-%m-%d %H:%M:%S")
        })

    return render_template("index.html", result=result, expression=expression, error=error,
history=history_data)

if __name__ == "__main__":
    app.run(debug=True)
```

3] MYSQL Database (db_config.sql)

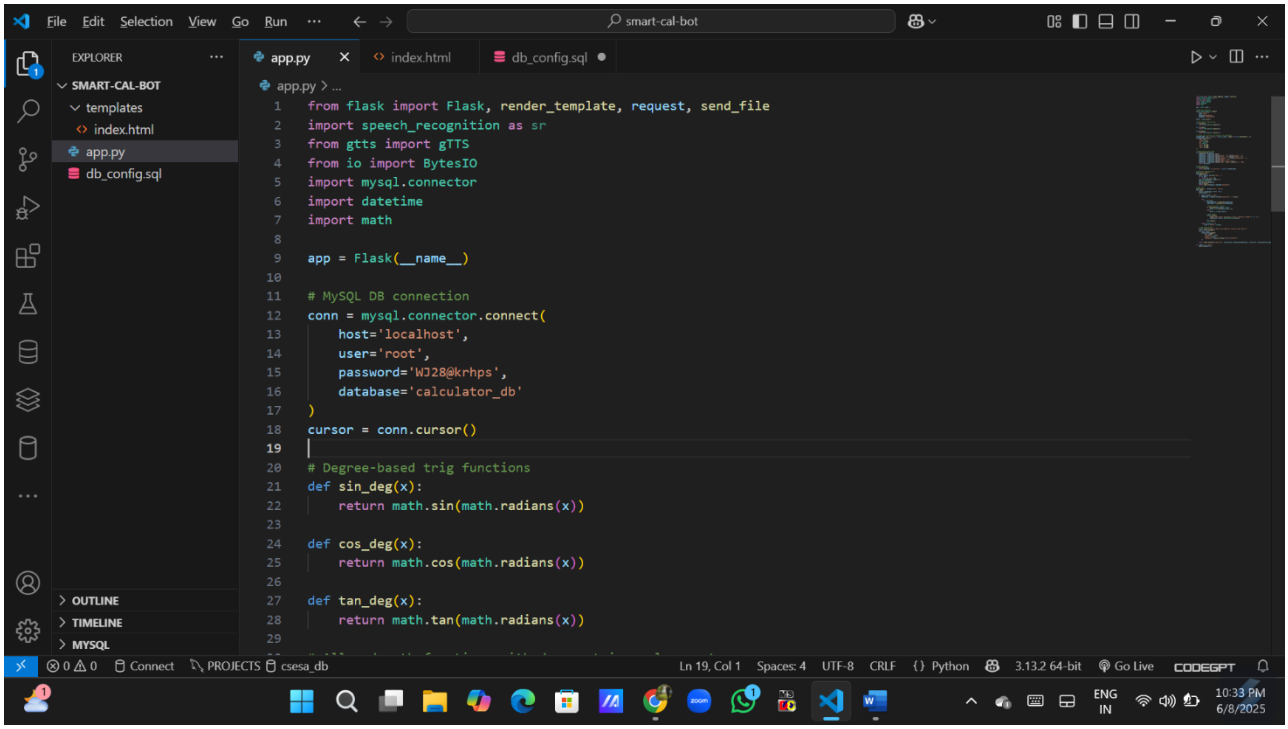
```
CREATE DATABASE IF NOT EXISTS calculator_db;
USE calculator_db;

CREATE TABLE IF NOT EXISTS history (
    id INT AUTO_INCREMENT PRIMARY KEY,
    expression TEXT NOT NULL,
    result TEXT NOT NULL,
    created_at DATETIME NOT NULL
);
```

Screenshots:

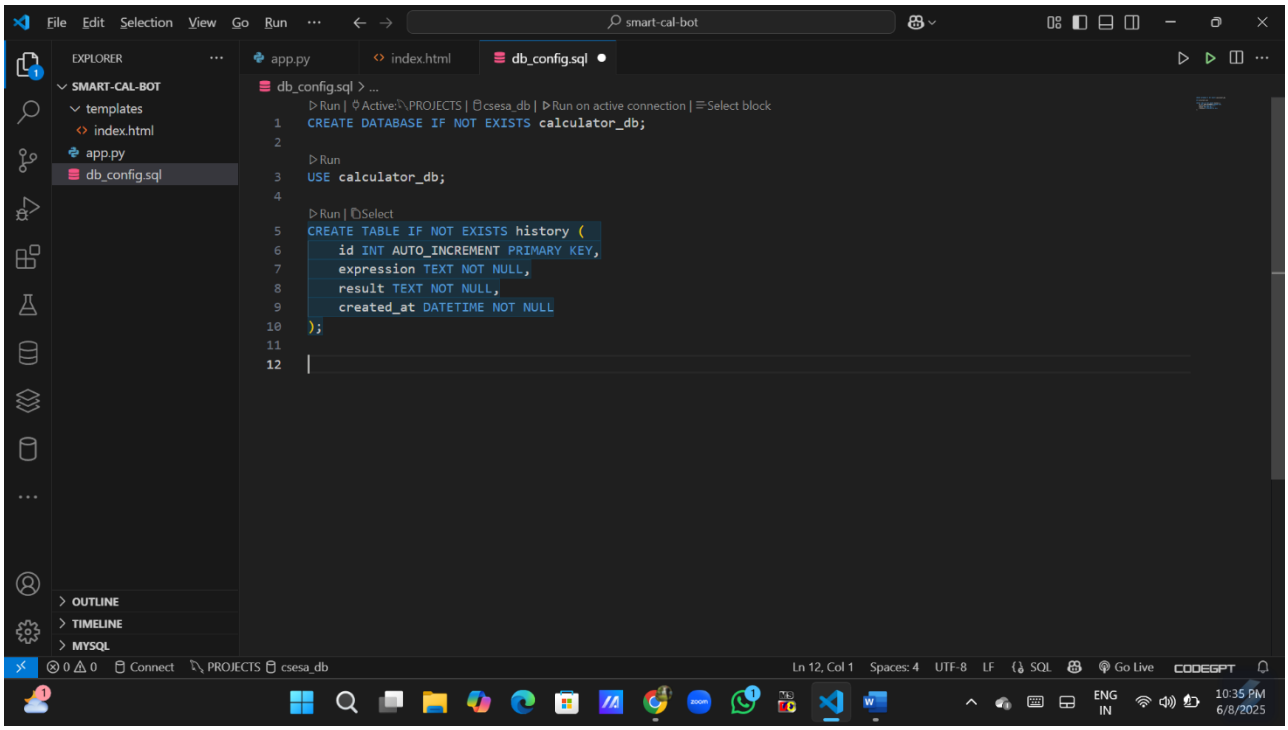
1] Frontend:



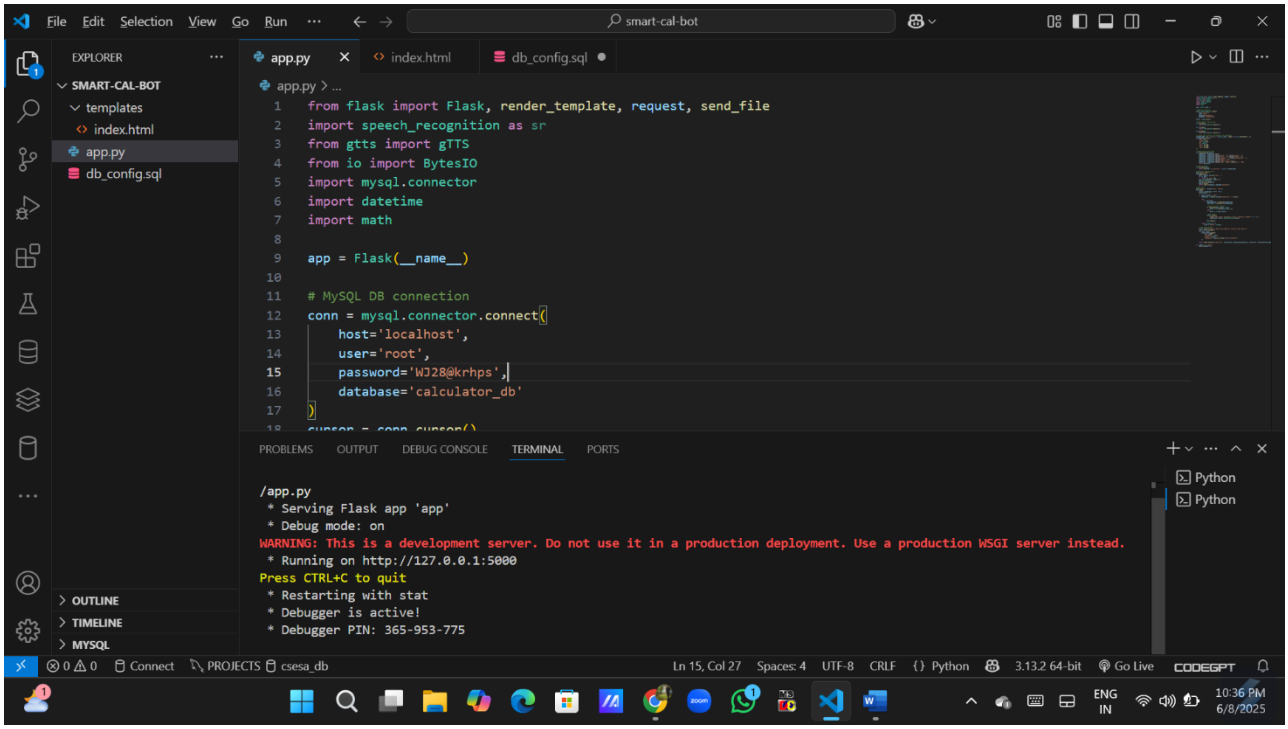
2] Backend:

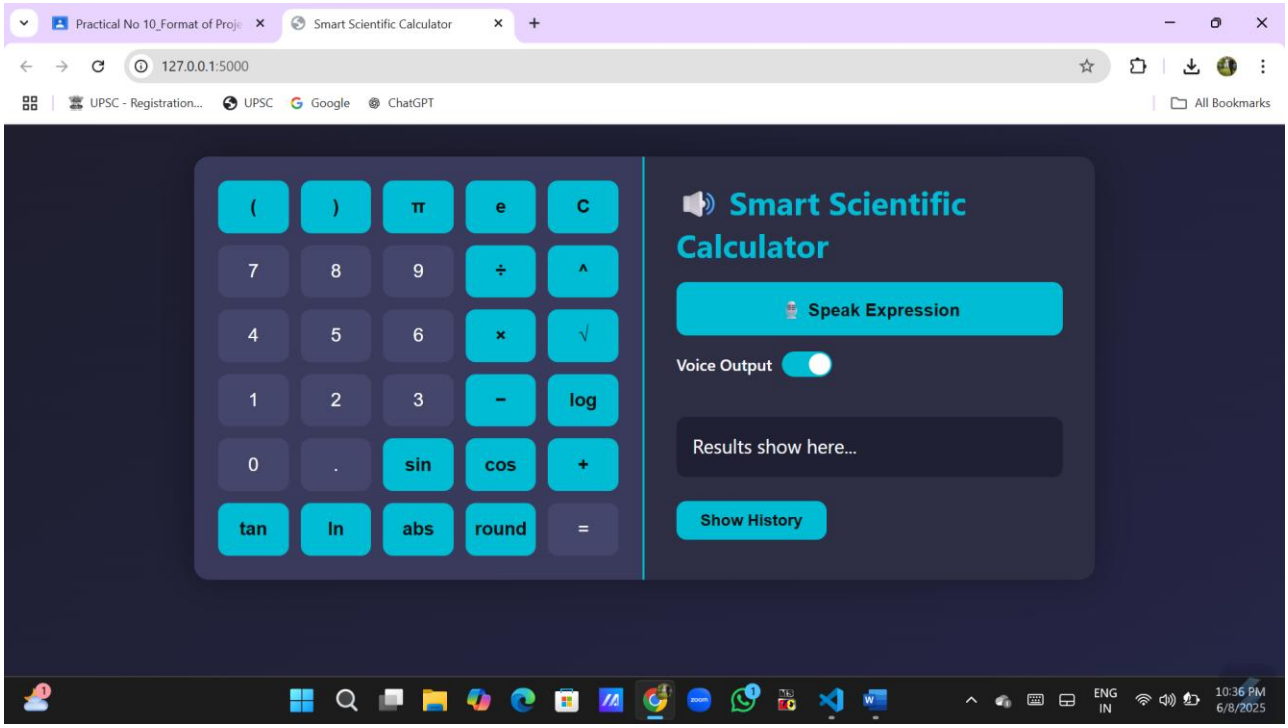


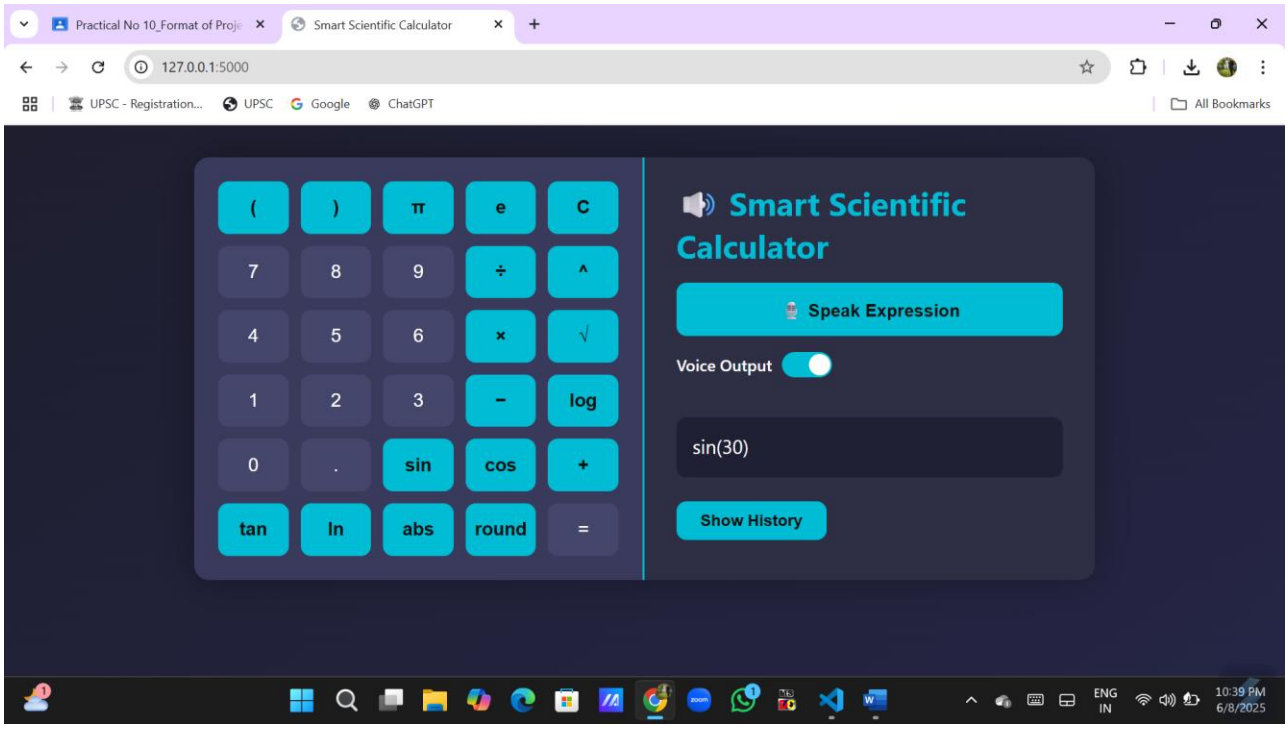
3] Database:

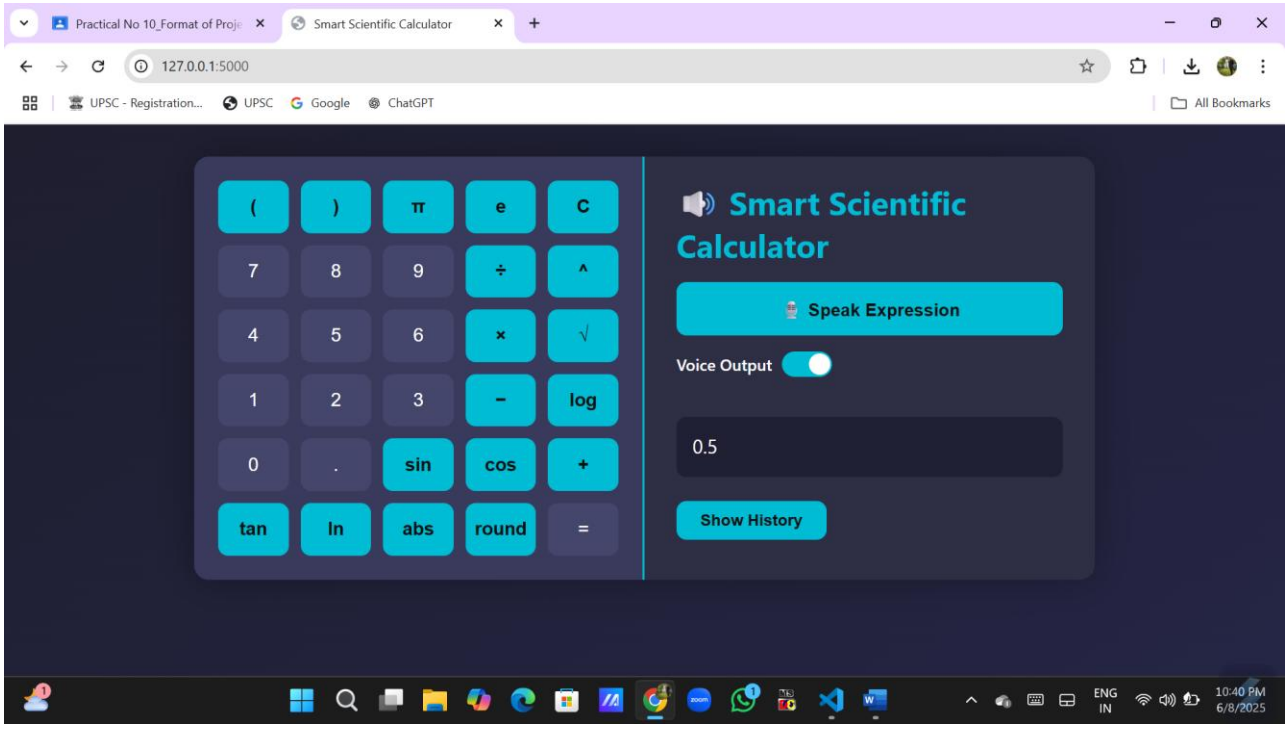


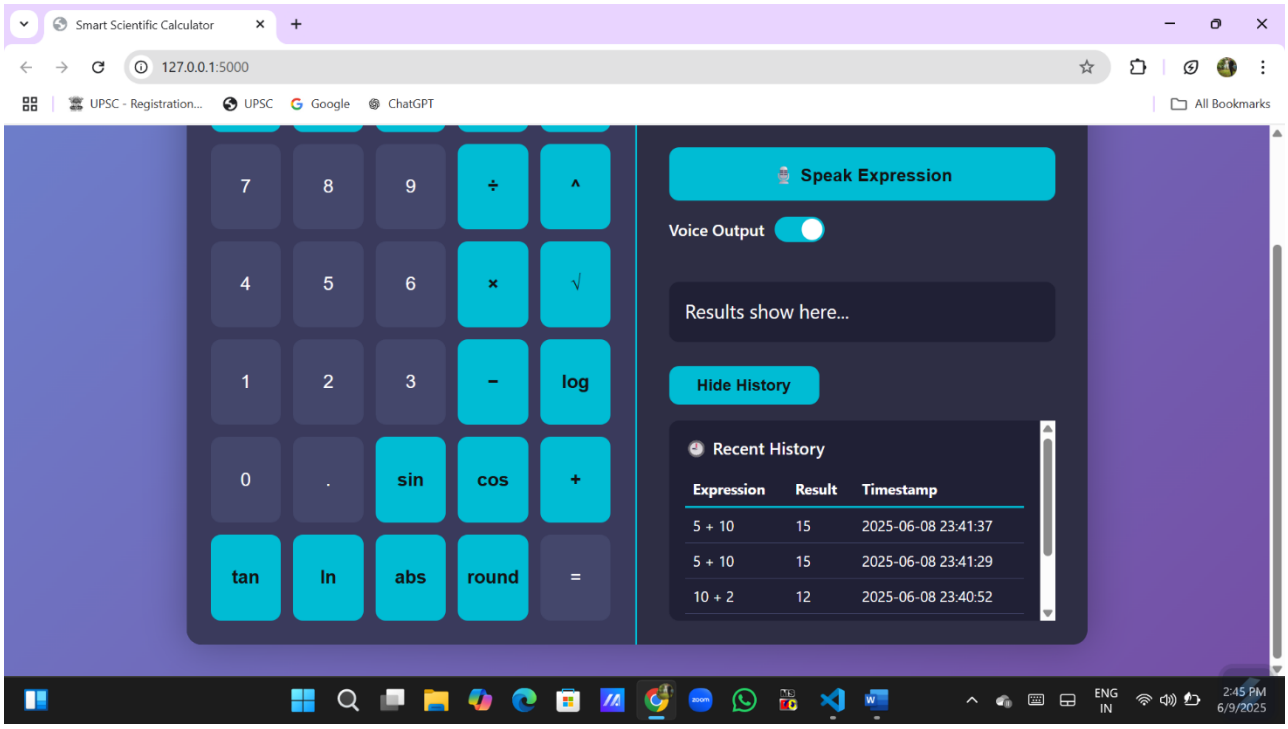
4] Working:


File Edit Selection View Go Run
smart-cal-bot
EXPLORER
SMART-CAL-BOT
templates
index.html
app.py
db_config.sql
app.py
index.html
db_config.sql
from flask import Flask, render_template, request, send_file
import speech_recognition as sr
from gtts import gTTS
from io import BytesIO
import mysql.connector
import datetime
import math
app = Flask(__name__)
# MySQL DB connection
conn = mysql.connector.connect(
host='localhost',
user='root',
password='WJ28@krhps',
database='calculator_db'
)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
/app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 365-953-775
Python
Python
OUTLINE
TIMELINE
MYSQL
Ln 15, Col 27 Spaces: 4 UTF-8 CRLF Python 3.13.2 64-bit Go Live CODEGPT
ENG IN
10:36 PM 6/8/2025


Practical No 10_Format of Proje
Smart Scientific Calculator
127.0.0.1:5000
UPSC - Registration...
UPSC
Google
ChatGPT
All Bookmarks
( ) π e C
7 8 9 ÷ ^
4 5 6 × √
1 2 3 − log
0 . sin cos +
tan ln abs round =
Smart Scientific Calculator
Speak Expression
Voice Output
Results show here...
Show History
ENG IN
10:36 PM 6/8/2025


Practical No 10_Format of Proje
Smart Scientific Calculator
127.0.0.1:5000
UPSC - Registration...
UPSC
Google
ChatGPT
All Bookmarks
( ) π e C
7 8 9 ÷ ^
4 5 6 × √
1 2 3 − log
0 . sin cos +
tan ln abs round =
Smart Scientific Calculator
Speak Expression
Voice Output
sin(30)
Show History
ENG IN
10:39 PM 6/8/2025


Practical No 10_Format of Proje
Smart Scientific Calculator
127.0.0.1:5000
UPSC - Registration...
UPSC
Google
ChatGPT
All Bookmarks
( ) π e C
7 8 9 ÷ ^
4 5 6 × √
1 2 3 − log
0 . sin cos +
tan ln abs round =
Smart Scientific Calculator
Speak Expression
Voice Output
0.5
Show History
ENG IN
10:40 PM 6/8/2025



Real Time Application:

1. **Voice-Controlled Smart Assistants (e.g., Alexa, Google Assistant)**
   - Smart calculators can be integrated into voice assistants to help users solve math problems hands-free.
   - Users can say: *"Hey Google, what is the square root of 225?"* and get instant answers.
2. **Educational Platforms & E-Learning Tools**
   - Can be used in learning apps to help students quickly perform calculations and understand math concepts using voice.
   - Makes the learning process more interactive and accessible for kids and visually impaired learners.
3. **Accessibility for Visually Impaired Users**
   - Helps blind or visually impaired users to perform complex calculations using just their voice without needing to type.
   - The voice output helps them hear the results clearly and interact naturally.
4. **Voice-Based Data Entry in Finance and Engineering**
   - Financial analysts or engineers can quickly calculate interest, returns, or formulas while working, without using the keyboard.
   - Speeds up workflows in accounting, research, and engineering projects.

Learning Outcome:

1. **Understanding of Flask Web Framework:**
   - Learned how to create routes, handle form data, and render templates using Flask for building web applications.
2. **Integration of Speech Technologies:**
   - Gained practical experience using SpeechRecognition and gTTS libraries to process voice input and generate voice output in real-time.
3. **Working with MySQL Database:**
   - Understood how to connect a Python backend with a MySQL database, perform CRUD operations, and store/retrieve user calculation history efficiently.
4. **Safe Expression Evaluation using eval():**
   - Learned how to securely evaluate user-provided mathematical expressions by restricting function access and preventing code injection using a custom allowed_names dictionary.
5. **Frontend-Backend Integration:**
   - Understood how to connect HTML forms with backend Python logic, pass data between them using Jinja templating, and display dynamic results and history on a web page.

Sign of Subject Incharge with Date: