

```

import os
import base64
import re
from datetime import datetime
from flask import Flask, render_template, request, redirect, url_for,
flash, send_from_directory
from werkzeug.utils import secure_filename
from dotenv import load_dotenv
import cv2
from gtts import gTTS
import pyttsx3
from openai import OpenAI
from db_config import get_db_connection

# Load environment
load_dotenv()
openai_api_key = os.getenv("OPENAI_API_KEY")
client = OpenAI(api_key=openai_api_key)

# Flask setup
app = Flask(__name__)
app.secret_key = 'supersecretkey'
UPLOAD_FOLDER = 'static/uploads'
AUDIO_FOLDER = 'static/audio'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(AUDIO_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['AUDIO_FOLDER'] = AUDIO_FOLDER

# Text-to-speech engine setup
engine = pyttsx3.init()
engine.setProperty('rate', 150)

def extract_plate_number(image_path: str) -> str:
    with open(image_path, "rb") as img_file:
        image_bytes = img_file.read()
        image_base64 = base64.b64encode(image_bytes).decode('utf-8')

    response = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": "Extract and return only the vehicle number from this image. Do not include any extra text."},
                    {"type": "image_url", "image_url": {"url": "data:image/jpeg;base64," + image_base64}}
                ]
            }
        ],
        max_tokens=50
    )
    extracted_text = response.choices[0].message.content.strip()
    plate_match = re.search(r'[A-Z]{2}\d{1,2}[A-Z]{1,2}\d{3,4}', extracted_text.replace(" ", "").upper())
    return plate_match.group(0) if plate_match else "UNKNOWN"

def announce_desktop(truck_number):
    message = f"Truck number {truck_number.upper()} please come inside the gate."

```

```

        engine.say(message)
        engine.runAndWait()

def generate_browser_audio(truck_number):
    message = f"Truck number {truck_number.upper()} please come inside the
gate. Truck number {truck_number.upper()} कृपया गेट के अंदर आइए।"
    tts = gTTS(text=message, lang='hi')
    filename = f"{truck_number.upper()}_announcement.mp3"
    filepath = os.path.join(app.config['AUDIO_FOLDER'], filename)
    tts.save(filepath)
    return filename

@app.route('/capture')
def capture_plate():
    cap = cv2.VideoCapture("http://192.168.31.143:4747/video")
    ret, frame = cap.read()
    cap.release()

    if ret:
        filename = f"plate_{datetime.now().strftime('%Y%m%d_%H%M%S')}.jpg"
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        cv2.imwrite(filepath, frame)
        return redirect(url_for('upload', captured=filename))
    else:
        flash("Failed to capture image")
        return redirect(url_for('upload'))

@app.route('/', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        captured_filename = request.form.get('captured_plate_filename')
        license_img = request.files.get('license')
        challan = request.files.get('challan')

        if not (captured_filename and license_img and challan):
            flash("All fields are required.")
            return redirect(url_for('upload'))

        plate_path = os.path.join(app.config['UPLOAD_FOLDER'],
        captured_filename)
        license_path = os.path.join(app.config['UPLOAD_FOLDER'],
        secure_filename(license_img.filename))
        challan_path = os.path.join(app.config['UPLOAD_FOLDER'],
        secure_filename(challan.filename))
        license_img.save(license_path)
        challan.save(challan_path)

        try:
            truck_number = extract_plate_number(plate_path)
        except Exception as e:
            flash(f"Error extracting plate number: {e}")
            return redirect(url_for('upload'))

        conn = get_db_connection()
        cur = conn.cursor()
        cur.execute('
                    INSERT INTO trucks (truck_number, license_path, challan_path,
plate_path, status)
                    VALUES (%s, %s, %s, %s, %s)
                    ', (truck_number, license_path, challan_path, plate_path,
'Queued'))
    
```

```

        conn.commit()
        cur.close()
        conn.close()

        flash(f'Truck {truck_number} registered successfully!')
        return redirect(url_for('upload'))

    return render_template('upload.html',
                           captured=request.args.get('captured'))

@app.route('/gate', methods=['GET', 'POST'])
def gate():
    conn = get_db_connection()
    cur = conn.cursor()

    if request.method == 'POST':
        cur.execute("SELECT id, truck_number FROM trucks WHERE status = "
                   "'Queued' ORDER BY id ASC LIMIT 1")
        truck = cur.fetchone()
        if truck:
            cur.execute("UPDATE trucks SET status = 'Entered' WHERE id = "
                       "%s", (truck[0],))
            conn.commit()
            announce_desktop(truck[1])
            audio_filename = generate_browser_audio(truck[1])
            flash(f"Truck {truck[1]} entry announced.")
            return redirect(url_for('gate', audio=audio_filename))

        cur.execute("SELECT id, truck_number, status FROM trucks WHERE status = "
                   "'Queued' ORDER BY id ASC")
        queued_trucks = cur.fetchall()
        cur.close()
        conn.close()

    return render_template('gate_entry.html', queue=queued_trucks,
                           audio=request.args.get('audio'))

@app.route('/static/audio/<truck_number>.mp3')
def play_audio(truck_number):
    filename = f"{truck_number.upper()}_announcement.mp3"
    return send_from_directory(app.config['AUDIO_FOLDER'], filename)

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

```

#### HTML OF GATE\_ENTRY.HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Gate Entry Queue</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-5">
    <h2 class="mb-4">Gate Entry Queue</h2>
    {% with messages = get_flashed_messages() %}
    {% if messages %}
        {% for message in messages %}
            <div class="alert alert-info">{{ message }}</div>

```

```

{ % endfor %}
{ % endif %}
{ % endwith %}

{ % if queue %}
    <table class="table table-striped">
        <thead>
            <tr>
                <th>Truck ID</th>
                <th>Truck Number</th>
                <th>Status</th>
                <th>Action</th>
            </tr>
        </thead>
        <tbody>
            { % for truck in queue %}
                <tr>
                    <td>{{ truck[0] }}</td>
                    <td>{{ truck[1] }}</td>
                    <td>{{ truck[2] }}</td>
                    <td>
                        <form method="POST" style="display:inline-block;">
                            <button type="submit" class="btn btn-success btn-sm">Announce</button>
                        </form>
                        { % set audio_file = 'static/audio/' +
truck[1]|replace(' ', '_') + '_announcement.mp3' %}
                            <audio id="audio-{{ truck[0] }}" src="{{ url_for('static', filename='audio/' + truck[1]|replace(' ', '_') + '_announcement.mp3') }}"></audio>
                            <button class="btn btn-outline-secondary btn-sm" onclick="document.getElementById('audio-{{ truck[0] }}').play()">▶
Play</button>
                        </td>
                    </tr>
                { % endfor %}
            </tbody>
        </table>
{ % else %}
    <p class="text-muted">No trucks in the queue.</p>
{ % endif %}

<a href="{{ url_for('upload') }}" class="btn btn-secondary mt-4">←
Back to Upload Page</a>
</body>
</html>

```

**UPLOAD HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Upload Truck Documents</title>
  <link
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
    rel="stylesheet">
</head>
<body class="container mt-5">
  <h2 class="mb-4">Upload Truck Documents</h2>
```

```

<!-- Flash Messages -->
{%
    with messages = get_flashed_messages() %
}
{%
    if messages %
        {% for message in messages %}
            <div class="alert alert-info">{{ message }}</div>
        {% endfor %}
    {% endif %}
{%
    endwith %
}

<!-- Captured Image Notification -->
{%
    if captured %
        <div class="alert alert-success">
            Captured Image: {{ captured }}
        </div>
        <div class="mb-3">
            <label>Captured Plate Image:</label><br>
            
        </div>
    {% endif %}
}

<form method="POST" enctype="multipart/form-data">
    {%
        if captured %
            <input type="hidden" name="captured_plate_filename" value="{{ captured }}">
        {% else %}
            <input type="hidden" name="captured_plate_filename" value="{{ request.args.get('captured', '') }}">
        {% endif %}
    }

    <div class="mb-3">
        <label class="form-label">Truck Plate Image:</label>
        <a href="{{ url_for('capture_plate') }}" class="btn btn-outline-primary btn-sm ms-2"> Capture from Camera</a>
    </div>

    <div class="mb-3">
        <label for="license" class="form-label">Driver License:</label>
        <input type="file" name="license" class="form-control" required>
    </div>

    <div class="mb-3">
        <label for="challan" class="form-label">Challan Document:</label>
        <input type="file" name="challan" class="form-control" required>
    </div>

    <button type="submit" class="btn btn-primary">Upload & Register</button>
</form>

<hr>
<a href="{{ url_for('gate') }}" class="btn btn-secondary mt-3">Go to Gate Entry Screen</a>

</body>
</html>

```

## DB\_CONFIG.PY FOR DATABASE CONNECTION

```
import mysql.connector

def get_db_connection():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="Satya@7735",
        database="truckdb"
    )
```

## .ENV FOR CALLING OPENAI THROUGH APIKEYMETHOD

```
OPENAI_API_KEY=sk-proj-
vXNUYUCzCDIfUzKs4HrH9qj9OKBeySSjoR2HmMT5voSqbgEt6DWdfo-28t1DROvYFTqcQNZ-
7T3B1bkFJTbEqMq3knmD57zP3MD_yS0J401G_3-aaT1dLmm6ueEQ2BPCtx1xKr6Bbc0-
iqxc9UL6D3HbvIA
```

## REQUIREMEMNTS.TXT FOR IMPORTANT IMPORTS FILE

```
Flask==2.3.2
opencv-python==4.9.0.80
pytesseract==0.3.10
mysql-connector-python==8.3.0
Werkzeug==2.3.6
```

## Upload Truck Documents

Truck Plate Image:

Driver License:

No file chosen

Challan Document:

No file chosen

---

## Gate Entry Queue

```
{% with messages = get_flashed_messages() %} {% if messages %} {% for message in messages %}
```

```
    {{ message }}
```

```
{% endfor %} {% endif %} {% endwith %} {% if queue %} {% for truck in queue %} {% endfor %}
```

### Truck

Truck ID	Number	Status	Action
----------	--------	--------	--------

{{ truck[0] }}	{{ truck[1] }}	Announce	{% set audio_file = 'static/audio/' + truck[1] replace(' ', '_') + '_announcement.mp3' %} {% if audio_file %}  Play Audio {% endif %}
	{{ truck[2] }}		

```
{% else %}
```

No trucks in the queue.

```
{% endif %}
```

 Back to Upload Page