# **Infosys Internship 4.0 Project Documentation**

**Title: Project Documentation: VisiOCR**

**•Introduction:**

VISIOCR is a Django-based web application designed to extract text from images of PAN cards using Optical Character Recognition (OCR) technology. The primary objective of the project is to simplify the process of digitizing PAN card and Aadhar card information, making it easily accessible and usable for various applications. The significance of this project lies in its ability to automate the extraction of critical information, thereby reducing manual data entry errors and increasing efficiency.

**TEAM MEMBER**:

1. Aryan Gupta

2. Chikka Venkata Revathi

3. Dhairy Shrivastava

4. Harshith Kumar

5. Himanshu

6. Jishitha Gotham

7. Ketha Dhana Veera Chaitanya

8. Madugula Maha Lakshmi

9. Meddi Nagendra

#### **Project Goals:**

* Automate the visitor issuance process
* Reduce manual data entry errors
* Enhance security through QR code verification
* Improve efficiency in visitor management

**•Project Scope:**

The project focuses on extracting and displaying textual information from PAN and Aadhar card images uploaded by users. It includes

* Upload and processing of UIDAI (Aadhaar) or PAN card images
* Text extraction using OCR technology
* QR code generation for extracted information
* Secure storage of extracted information in MySQL database
* Visitor pass generation and download functionality
* Dashboard for visitor management
* Web interface for pass generation and management

The project does not cover:

• Verification of extracted information.

**• Handling of documents other than PAN and Aadhar cards.**

#### Limitations:

* Currently supports only UIDAI (Aadhaar) and PAN cards
* Requires clear, high-quality images for accurate OCR

### •Requirements:

#### Functional Requirements:

1. Image upload and processing
2. OCR-based text extraction
3. QR code generation
4. Database storage and retrieval
5. Visitor pass generation and download
6. QR code scanning for verification
7. Dashboard for visitor management

#### Non-functional Requirements:

1. User-friendly web interface
2. Secure data storage and handling
3. Efficient processing of images and data

### •Technical Stack:

#### Programming Languages:

* Python 3.9.13

#### Frameworks/Libraries:

* Django
* OpenCV
* Pillow
* pytesseract
* qrcode
* reportlab
* mysql-connector-python
* mysqlclient

#### Databases:

* MySQL

#### Tools/Platforms:

* Tesseract-OCR engine
* Visual Studio Code

### •Architecture/Design:

VisiOCR follows a Django MVT (Model-View-Template) architecture:

* Models: Define the data structure and interact with MySQL database
* Views: Handle the business logic and process user requests
* Templates: Render the user interface

#### Key components:

1. Image Capture Module
2. OCR Integration Module
3. Data Extraction Module
4. Data Integration Module
5. Visiting Pass Generation Module

#### Design Decisions:

* Use of Django for rapid development and robust web framework.
* Tesseract for OCR due to its reliability and open-source nature.
* QR code generation for secure and quick verification.

### Modules:

#### Image Capture Module

* Handles image upload from users
* Validates image format and quality
* Prepares images for OCR processing

#### OCR Integration Module

* Integrates Tesseract OCR engine
* Processes uploaded images to extract text

#### Data Extraction Module

* Parses OCR output
* Identifies and extracts relevant information (name, ID number, date of birth, etc.)
* Validates extracted data

#### Data Integration Module

* Stores extracted information in the MySQL database
* Manages data retrieval and updates

#### Visiting Pass Generation Module

* Generates QR codes with extracted information
* Creates downloadable visitor pass documents
* Manages pass expiration and validation

### •Development:

The project was developed using Python and Django framework. Key development practices include:

* Following Django's best practices and coding standards
* Modular design for easy maintenance and scalability
* Integration of external libraries for OCR and QR code generation

#### Challenges encountered:

* Ensuring accurate OCR results with varying image qualities
* Optimizing performance for concurrent user requests
* Securely handling sensitive user data

### •Testing:

#### Testing approach:

* Unit tests for individual components
* Integration tests for module interactions
* End-to-end tests for complete user flows

### • Deployment:

1. Clone the repository:



1. Install Tesseract-OCR:

a) Download from <https://sourceforge.net/projects/tesseract-ocr.mirror>

b) Add the tesseract-ocr.exe file to the environment variables

1. Create and activate virtual environment

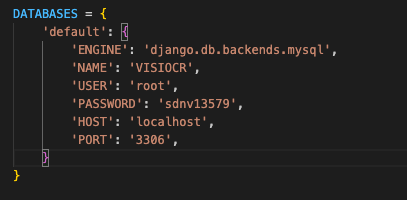


1. Install the required packages :



1. Configure MySQL:

* Create a new database named VISIOCR.
* Update the database settings in OCR\_DJango/settings.py:



1. Apply database migrations:



1. Start the development server :

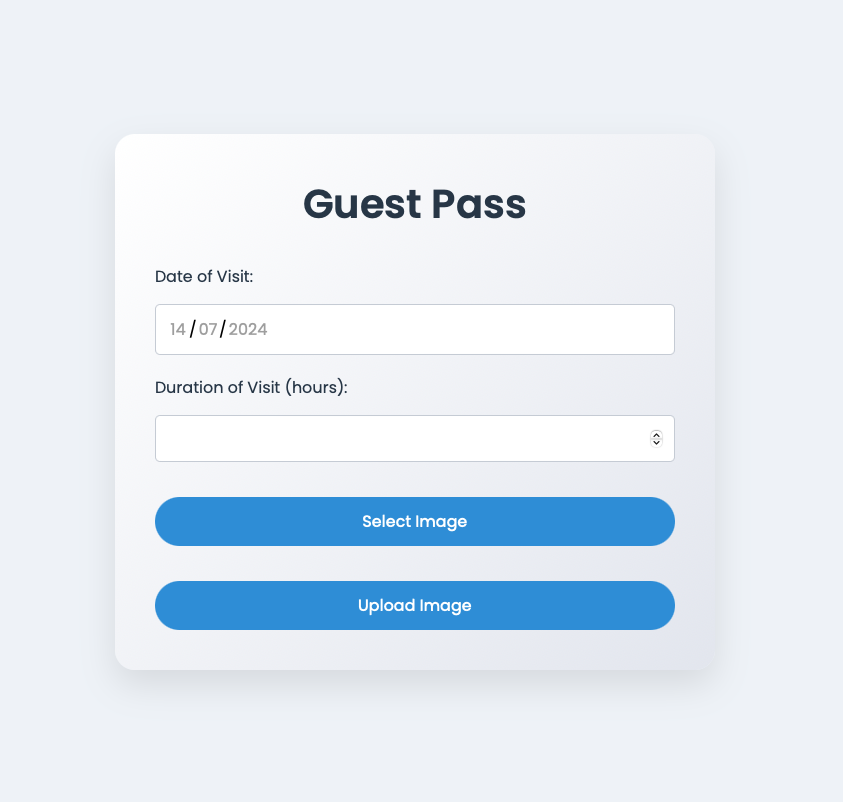


1. Access the application at  <http://127.0.0.1:8000/>

### •User Guide:

#### How to upload ID Card:

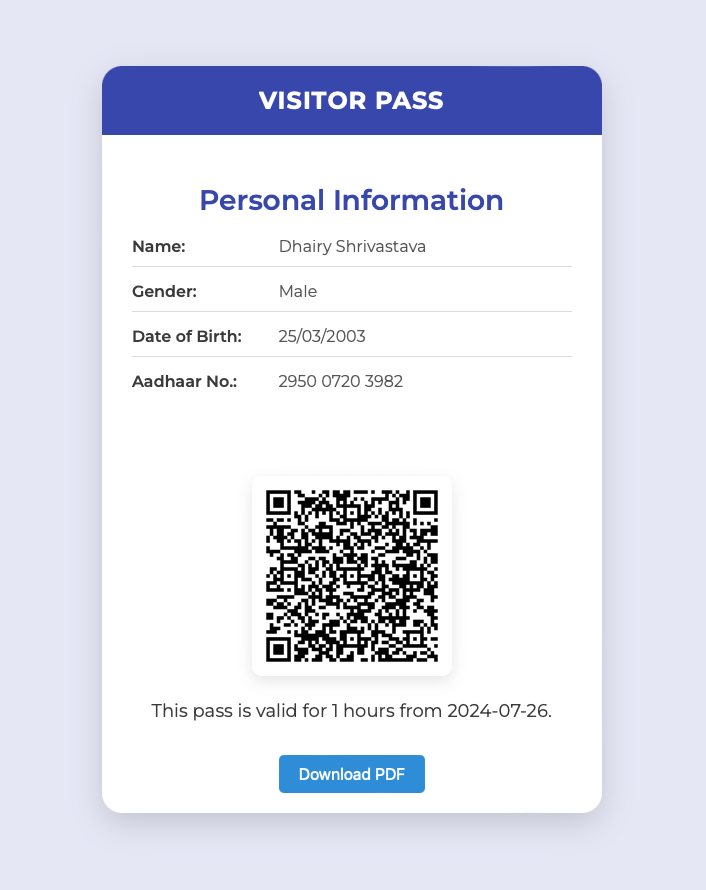
1. Access the project through <http://127.0.0.1:8000/>
2. Fill all the details in the form
3. Click on Choose file option
4. Upload the UIDAI (Aadhaar) or PAN Card image



#### How to Generate Pass:

1. Access the project through <http://127.0.0.1:8000/>
2. Upload the UIDAI (Aadhaar) or PAN Card image
3. Click on Generate Pass

It will generate visitor pass.



#### How to Scan QR Code:

1. Access the project through <http://127.0.0.1:8000/>
2. Upload the QR Code image or scan it through camera
3. Verified information will be displayed

#### Troubleshooting Tips:

* Ensure the uploaded image is clear and readable.
* Check the database connection if data retrieval fails.

### •Conclusion:

#### Summary of Outcomes:

* The VisiOCR project successfully automates the visitor pass generation process, leveraging OCR technology to extract information from identification documents.
* The system enhances efficiency and accuracy in visitor management.

#### Lessons Learned:

* Importance of clear and high-quality images for OCR accuracy.
* Effective use of Django for rapid web application development.

#### Areas for Improvement:

* Mobile support for on-the-go pass generation
* Integration with access control systems
* Support for additional ID card types
* Advanced analytics and reporting features

### \* Appendices:

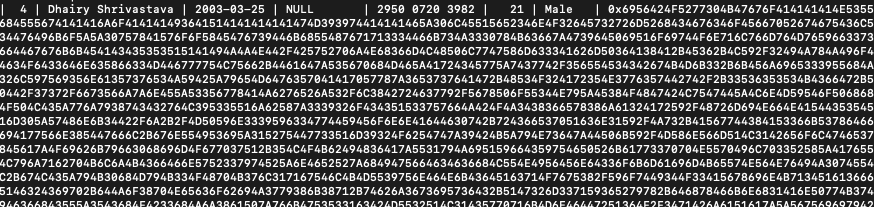
### Requirements.txt file:

* numpy
* opencv-python
* Pillow
* pytesseract
* ftfy
* django
* djangorestframework
* django-cors-headers
* django-environ
* django-extensions
* mysqlclient
* mysql-connector-python
* qrcode
* reportlab

### Database Schema (home\_visitorpass):

Table structure:

1. id: Primary key, auto-incrementing integer.
2. date\_of\_visiting: Date of the visit (date).
3. duration\_of\_visiting: Duration of the visit in hours (integer).
4. qr\_code\_image: Path to the generated QR code image file (varchar).
5. date\_of\_birth: Date of birth from Aadhaar card (date, nullable).
6. gender: Gender from card (varchar, nullable).
7. aadhaar\_number: Aadhaar number (varchar, nullable).
8. pan\_date\_of\_birth: Date of birth from PAN card (date, nullable).
9. name: Name from card (varchar, nullable).
10. pan\_number: PAN number (varchar, nullable).



**Sample Code :**

**Views.py**

import cv2

import numpy as np

import pytesseract

from datetime import datetime, timedelta

from django.shortcuts import render

from django.http import HttpResponse

from django.views.decorators.csrf import csrf\_exempt

import re

from django.template.loader import get\_template

from xhtml2pdf import pisa

import qrcode

import base64

from io import BytesIO

import mysql.connector

from mysql.connector import Error

from django.template.loader import render\_to\_string

import logging

import json

logging.basicConfig(level=logging.DEBUG)

def home(request):

return render(request, 'ocr\_app/home.html')

def preprocess\_image(image):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

processed\_image = cv2.threshold(gray, 0, 255, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU)[1]

return processed\_image

def extract\_info(image):

processed\_image = preprocess\_image(image)

text = pytesseract.image\_to\_string(processed\_image)

name, birth\_date, pan\_number, aadhaar\_number, gender = parse\_text(text)

return name, birth\_date, pan\_number, aadhaar\_number, gender

def parse\_text(text):

name = ""

birth\_date = ""

pan\_number = ""

aadhaar\_number = ""

gender = ""

all\_text\_list = re.split(r'[\n]', text)

text\_list = [i for i in all\_text\_list if i.strip() != ""]

pan\_pattern = r'[A-Z]{5}[0-9]{4}[A-Z]{1}'

pan\_match = re.search(pan\_pattern, text)

if pan\_match:

pan\_number = pan\_match.group(0).strip()

aadhar\_pattern = r'\d{4}\s\d{4}\s\d{4}'

aadhar\_match = re.search(aadhar\_pattern, text)

if aadhar\_match:

aadhaar\_number = aadhar\_match.group(0).strip()

if any(word in text.lower() for word in ["male", "female"]):

name, birth\_date, gender = extract\_aadhar\_info(text\_list)

else:

name, birth\_date, gender = extract\_pan\_info(text)

return name, birth\_date, pan\_number, aadhaar\_number, gender

def extract\_aadhar\_info(text\_list):

user\_dob = ""

user\_name = ""

user\_gender = ""

aadhar\_dob\_pat = r'(YoB|YOB:|DOB:|DOB|AOB)'

gender\_pat = r'\b(?:male|female|transgender|other)\b'

date\_ele = ""

index = None

for idx, line in enumerate(text\_list):

if re.search(aadhar\_dob\_pat, line):

index = re.search(aadhar\_dob\_pat, line).span()[1]

date\_ele = line

dob\_idx = idx

break

if index is not None:

date\_str = ''.join(char for char in date\_ele[index:] if re.match(r'\d|/', char))

user\_dob = date\_str

user\_name = text\_list[dob\_idx - 1]

name\_match = re.search(r'([A-Z][a-zA-Z\s]+)', user\_name)

if name\_match:

name = name\_match.group(0).strip()

else:

name = ""

for line in text\_list:

gender\_match = re.search(gender\_pat, line, re.IGNORECASE)

if gender\_match:

user\_gender = gender\_match.group(0).capitalize()

break

return name, user\_dob, user\_gender

else:

return "", "", ""

def extract\_pan\_info(text):

pancard\_name = ""

user\_gender = ""

name\_patterns = [

r'Name\s\*\n([A-Z\s]+)',

]

gender\_pat = r'\b(?:male|female|transgender|other)\b'

for pattern in name\_patterns:

name\_match = re.search(pattern, text)

if name\_match:

matched\_name = name\_match.group(1).strip().replace('\n', ' ')

pancard\_name = matched\_name

break

dob\_match = re.search(r'(\d{2}/\d{2}/\d{4})', text, re.IGNORECASE)

if dob\_match:

birth\_date = dob\_match.group(0).strip()

else:

birth\_date = ""

gender\_match = re.search(gender\_pat, text, re.IGNORECASE)

if gender\_match:

user\_gender = gender\_match.group(0).capitalize()

return pancard\_name, birth\_date, user\_gender

def create\_connection():

try:

connection = mysql.connector.connect(

host='localhost',

database='VISIOCR',

user='root',

password='sdnv13579'

)

return connection

except Error as e:

logging.error("Error while connecting to MySQL: %s", e)

return None

def create\_table(connection):

try:

if connection.is\_connected():

cursor = connection.cursor()

cursor.execute("""

CREATE TABLE IF NOT EXISTS extracted\_data (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(255),

birth\_date DATE,

pan\_number VARCHAR(10),

aadhaar\_number VARCHAR(70),

age INT,

gender VARCHAR(10),

qr\_code\_image LONGBLOB

)

""")

connection.commit()

logging.debug("Table 'extracted\_data' created successfully")

cursor.close()

except Error as e:

logging.error("Error while creating table: %s", e)

def insert\_data(connection, name, birth\_date, pan\_number, aadhaar\_number, gender, qr\_code\_image\_data, age):

try:

if connection.is\_connected():

cursor = connection.cursor()

sanitized\_name = name.replace("'", "''")

if birth\_date:

birth\_date = datetime.strptime(birth\_date, "%d/%m/%Y").strftime("%Y-%m-%d")

else:

birth\_date = None

query = """

INSERT INTO extracted\_data (name, birth\_date, pan\_number, aadhaar\_number, gender, qr\_code\_image, age)

VALUES (%s, %s, %s, %s, %s, %s, %s)

"""

values = (

sanitized\_name,

birth\_date,

pan\_number if pan\_number else None,

aadhaar\_number if aadhaar\_number else None,

gender if gender else None,

qr\_code\_image\_data if qr\_code\_image\_data else None,

age

)

cursor.execute(query, values)

connection.commit()

logging.debug("Record inserted successfully: Name=%s, Birth Date=%s, PAN Number=%s, Aadhaar Number=%s, Gender=%s", sanitized\_name, birth\_date, pan\_number, aadhaar\_number, gender)

cursor.close()

except Error as e:

logging.error("Error while inserting data into table: %s", e)

def process\_image(image):

try:

name, birth\_date, pan\_number, aadhaar\_number, gender = extract\_info(image)

logging.debug("Extracted Info: Name=%s, Birth Date=%s, PAN Number=%s, Aadhaar Number=%s, Gender=%s", name, birth\_date, pan\_number, aadhaar\_number, gender)

if not name or not birth\_date:

logging.error("Failed to extract valid name or birth date from the image.")

return "", "", "", "", "", "", None

connection = create\_connection()

if not connection:

logging.error("Failed to establish a database connection.")

return name, birth\_date, "", pan\_number, aadhaar\_number, gender, None

try:

create\_table(connection)

data = {

"name": name,

"birth\_date": birth\_date,

"pan\_number": pan\_number,

"aadhaar\_number": aadhaar\_number,

"gender": gender

}

qr\_code\_image\_data, expiration\_time = create\_qr\_code(data)

birth\_date\_obj = datetime.strptime(birth\_date, "%d/%m/%Y")

age = (datetime.now() - birth\_date\_obj).days // 365

insert\_data(connection, name, birth\_date, pan\_number, aadhaar\_number, gender, qr\_code\_image\_data, age)

except Exception as e:

logging.error("Error processing image: %s", e)

return name, birth\_date, "", pan\_number, aadhaar\_number, gender, None

finally:

if connection and connection.is\_connected():

connection.close()

logging.debug("MySQL connection is closed")

return name, birth\_date, qr\_code\_image\_data, pan\_number, aadhaar\_number, gender, expiration\_time

except Exception as e:

logging.error("An unexpected error occurred: %s", e)

return "", "", "", "", "", "", None

def create\_qr\_code(data, expiration\_hours=2):

try:

expiration\_time = datetime.now() + timedelta(hours=expiration\_hours)

data['expiration\_time'] = expiration\_time.strftime('%Y-%m-%d %H:%M:%S')

qr = qrcode.QRCode(

version=1,

error\_correction=qrcode.constants.ERROR\_CORRECT\_L,

box\_size=10,

border=4,

)

qr.add\_data(json.dumps(data))

qr.make(fit=True)

img = qr.make\_image(fill='black', back\_color='white')

buffered = BytesIO()

img.save(buffered, format="PNG")

qr\_code\_image\_data = base64.b64encode(buffered.getvalue()).decode('utf-8')

return qr\_code\_image\_data, expiration\_time

except Exception as e:

logging.error("Failed to create QR code: %s", e)

return "", None

@csrf\_exempt

def upload\_image(request):

if request.method == 'POST' and request.FILES.get('image'):

image\_file = request.FILES['image']

image\_data = np.frombuffer(image\_file.read(), np.uint8)

image = cv2.imdecode(image\_data, cv2.IMREAD\_COLOR)

name, birth\_date, qr\_code\_image\_data, pan\_number, aadhaar\_number, gender, expiration\_time = process\_image(image)

visit\_date = request.POST.get('visit\_date')

duration = request.POST.get('duration')

# Calculate age from birth\_date

if birth\_date:

birth\_date\_obj = datetime.strptime(birth\_date, "%d/%m/%Y")

age = (datetime.now() - birth\_date\_obj).days // 365

else:

age = ""

context = {

'name': name,

'birth\_date': birth\_date,

'qr\_code\_image\_data': qr\_code\_image\_data,

'pan\_number': pan\_number,

'aadhaar\_number': aadhaar\_number,

'gender': gender,

'visit\_date': visit\_date,

'duration': duration,

'age': age,

'expiration\_time': expiration\_time

}

return render(request, 'ocr\_app/result.html', context)

return render(request, 'ocr\_app/home.html')

@csrf\_exempt

def download\_pdf(request):

template\_path = 'ocr\_app/pdf\_template.html'

context = {

'name': request.POST.get('name'),

'birth\_date': request.POST.get('birth\_date'),

'age': request.POST.get('age'),

'pan\_number': request.POST.get('pan\_number'),

'aadhaar\_number': request.POST.get('aadhaar\_number'),

'gender': request.POST.get('gender'),

'visit\_date': request.POST.get('visit\_date'),

'duration': request.POST.get('duration'),

}

# Render the template as a string

html = render\_to\_string(template\_path, context)

# Create a PDF

response = HttpResponse(content\_type='application/pdf')

response['Content-Disposition'] = 'attachment; filename="visitor\_pass.pdf"'

pisa\_status = pisa.CreatePDF(

html, dest=response

)

# If PDF creation fails, return an error message

if pisa\_status.err:

return HttpResponse('We had some errors <pre>' + html + '</pre>')

return response

**urls.py**

from django.urls import path

from . import views

urlpatterns = [

path('', views.home, name='home'),

path('upload/', views.upload\_image, name='upload\_image'),

path('download/', views.download\_pdf, name='download\_pdf'),

]

**Home.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Guest Pass</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

body {

display: flex;

justify-content: center;

align-items: center;

background-color: #f0f4f8;

min-height: 100vh;

width: 100%;

padding: 20px;

}

.container {

width: 100%;

max-width: 600px;

background: linear-gradient(135deg, #ffffff, #e6e9f0);

border-radius: 20px;

box-shadow: 0 15px 35px rgba(0, 0, 0, 0.1);

padding: 40px;

text-align: left;

}

h1 {

font-size: 2.5rem;

color: #2c3e50;

margin-bottom: 30px;

text-align: center;

font-weight: 600;

}

.id-card {

background-color: #ffffff;

border-radius: 15px;

padding: 20px;

margin-bottom: 30px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.05);

}

.id-card p {

margin: 10px 0;

font-size: 1rem;

color: #34495e;

}

.id-card p b {

color: #2c3e50;

font-weight: 600;

}

.qr-code {

text-align: center;

margin-top: 30px;

}

.qr-code img {

max-width: 200px;

border-radius: 10px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

}

.btn {

background-color: #3498db;

color: #ffffff;

padding: 12px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

width: 100%;

font-size: 1rem;

margin-top: 15px;

cursor: pointer;

border: none;

border-radius: 30px;

transition: background-color 0.3s ease, transform 0.3s ease;

}

.btn:hover {

background-color: #2980b9;

transform: translateY(-2px);

}

.file-upload input[type="file"] {

display: none;

}

#file-name {

display: block;

margin-top: 10px;

font-size: 0.9rem;

color: #7f8c8d;

}

.form-group {

margin-bottom: 20px;

}

label {

display: block;

font-size: 1rem;

color: #2c3e50;

margin-bottom: 10px;

}

input[type="date"],

input[type="number"] {

width: 100%;

padding: 10px;

font-size: 1rem;

border-radius: 5px;

border: 1px solid #ccd1d9;

margin-top: 5px;

}

</style>

</head>

<body>

<div class="container">

<h1>Guest Pass</h1>

<div class="file-upload">

<form action="{% url 'upload\_image' %}" method="POST" enctype="multipart/form-data">

{% csrf\_token %}

<div class="form-group">

<label for="visit\_date">Date of Visit:</label>

<input type="date" name="visit\_date" id="visit\_date" required>

</div>

<div class="form-group">

<label for="duration">Duration of Visit (hours):</label>

<input type="number" name="duration" id="duration" required>

</div>

<label class="btn">

Select Image

<input type="file" name="image" accept="image/\*" id="image" onchange="displayFileName()">

</label>

<span id="file-name"></span>

<button type="submit" class="btn">Upload Image</button>

</form>

</div>

{% if name %}

<form action="{% url 'download\_pdf' %}" method="POST">

{% csrf\_token %}

<input type="hidden" name="name" value="{{ name }}">

<input type="hidden" name="birth\_date" value="{{ birth\_date }}">

<input type="hidden" name="age" value="{{ age }}">

<input type="hidden" name="pan\_number" value="{{ pan\_number }}">

<input type="hidden" name="aadhaar\_number" value="{{ aadhaar\_number }}">

<input type="hidden" name="gender" value="{{ gender }}">

<input type="hidden" name="visit\_date" value="{{ visit\_date }}">

<input type="hidden" name="duration" value="{{ duration }}">

<button type="submit" class="btn">Generate PDF</button>

</form>

{% endif %}

</div>

<script>

function displayFileName() {

const input = document.getElementById('image');

const fileName = input.files[0] ? input.files[0].name : '';

document.getElementById('file-name').innerText = fileName;

}

</script>

</body>

</html>

**pdf\_template.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Visitor Pass</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.info {

display: flex;

justify-content: space-between;

margin-bottom: 10px;

max-width: 400px;

}

.info strong {

flex-basis: 50%;

text-align: right;

padding-right: 10px;

font-weight: bold;

}

.info span {

flex-basis: 50%;

text-align: left;

}

.info-container {

margin: 0 auto;

max-width: 600px;

}

h1 {

text-align: center;

}

</style>

</head>

<body>

<div class="info-container">

<h1>Visitor Pass</h1>

<div class="info">

<strong>Name:</strong>

<span>{{ name }}</span>

</div>

<br>

<div class="info">

<strong>Date of Birth:</strong>

<span>{{ birth\_date }}</span>

</div>

<br>

<div class="info">

<strong>Gender:</strong>

<span>{{ gender }}</span>

</div>

<br>

<div class="info">

<strong>Age:</strong>

<span>{{ age }}</span>

</div>

<br>

{% if pan\_number %}

<div class="info">

<strong>PAN Number:</strong>

<span>{{ pan\_number }}</span>

</div>

<br>

{% endif %}

<!-- <div class="info">

<strong>Date of Visiting:</strong>

<span>{{ date\_of\_visiting }}</span>

</div>

<br>

<div class="info">

<strong>Duration of Visiting:</strong>

<span>{{ duration\_of\_visiting }}</span>

</div> -->

<!-- <br> -->

<div class="info">

<strong>Aadhaar Number:</strong>

<span>{{ aadhaar\_number }}</span>

</div>

<br>

</div>

</body>

</html>

**result.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Guest Access Card</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Montserrat:wght@400;600;700&display=swap');

body {

font-family: 'Montserrat', sans-serif;

margin: 0;

padding: 0;

background-color: #e8eaf6;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

}

.container {

width: 90%;

max-width: 500px;

background-color: #ffffff;

border-radius: 20px;

box-shadow: 0 10px 30px rgba(0, 0, 0, 0.15);

overflow: hidden;

padding-bottom: 20px;

}

.header {

text-align: center;

padding: 20px 0;

background-color: #3f51b5;

color: white;

}

.header h2 {

margin: 0;

font-size: 24px;

font-weight: 700;

text-transform: uppercase;

letter-spacing: 1px;

}

.guest-info {

padding: 30px;

}

.guest-info h1 {

margin-bottom: 20px;

font-size: 28px;

color: #3f51b5;

text-align: center;

font-weight: 600;

}

.info-item {

margin: 15px 0;

display: flex;

align-items: center;

border-bottom: 1px solid #e0e0e0;

padding-bottom: 10px;

}

.info-item:last-child {

border-bottom: none;

}

.info-item strong {

flex: 1;

font-weight: 600;

font-size: 16px;

color: #424242;

}

.info-item span {

flex: 2;

font-size: 16px;

color: #616161;

}

.qr-code {

text-align: center;

margin-top: 30px;

}

.qr-code img {

max-width: 200px;

border-radius: 10px;

box-shadow: 0 5px 15px rgba(0, 0, 0, 0.1);

}

.validity {

text-align: center;

margin-top: 20px;

font-size: 18px;

color: #424242;

}

.btn {

background-color: #3498db;

color: #ffffff;

padding: 10px 20px;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin-top: 15px;

cursor: pointer;

border: none;

border-radius: 5px;

transition: background-color 0.3s ease, transform 0.3s ease;

}

.btn:hover {

background-color: #2980b9;

transform: translateY(-2px);

}

</style>

</head>

<body>

<div class="container">

<div class="header">

<h2>Visitor Pass</h2>

</div>

<div class="guest-info">

<h1>Personal Information</h1>

<div class="info-item">

<strong>Name:</strong>

<span>{{ name }}</span>

</div>

<div class="info-item">

<strong>Gender:</strong>

<span>{{ gender }}</span>

</div>

<div class="info-item">

<strong>Date of Birth:</strong>

<span>{{ birth\_date }}</span>

</div>

{% if pan\_number %}

<div class="info-item">

<strong>PAN Number:</strong>

<span>{{ pan\_number }}</span>

</div>

{% endif %}

{% if aadhaar\_number %}

<div class="info-item">

<strong>Aadhaar No.:</strong>

<span>{{ aadhaar\_number }}</span>

</div>

{% endif %}

</div>

<div class="qr-code">

<img src="data:image/png;base64,{{ qr\_code\_image\_data }}" alt="QR Code">

</div>

<div class="validity">

<p>This pass is valid for {{ duration }} hours from {{ visit\_date }}.</p>

</div>

<div style="text-align: center;">

<form action="{% url 'download\_pdf' %}" method="POST">

{% csrf\_token %}

<input type="hidden" name="name" value="{{ name }}">

<input type="hidden" name="birth\_date" value="{{ birth\_date }}">

<input type="hidden" name="age" value="{{ age }}">

<input type="hidden" name="pan\_number" value="{{ pan\_number }}">

<input type="hidden" name="aadhaar\_number" value="{{ aadhaar\_number }}">

<input type="hidden" name="gender" value="{{ gender }}">

<input type="hidden" name="visit\_date" value="{{ visit\_date }}">

<input type="hidden" name="duration" value="{{ duration }}">

<input type="hidden" name="qr\_code\_image\_data" value="{{ qr\_code\_image\_data }}">

<button type="submit" class="btn">Download PDF</button>

</form>

</div>

</div>

</body>

</html>

**User Interface (UI):**

