

Infosys Internship 4.0 Project Documentation

Made by – Altamash Faraz

Title: Project Documentation: visiOCR

Introduction

Brief overview of the project, its objectives, and its significance:

VisiOCR is an innovative project aimed at automating the process of extracting and managing visitor information using Optical Character Recognition (OCR) technology. The project involves the extraction of data from visitor documents, generating dynamic QR codes for visitor passes, and integrating the system with a database for efficient management. The primary objective is to streamline visitor management processes, enhance data accuracy, and improve security measures.

Team members :

- Altamash Faraz

Project Scope

Define the boundaries of the project:

- **Included:**
 - Extraction of text data from images of visitor documents.
 - Generation of dynamic QR codes for visitor passes.
 - Database integration for storing and managing visitor data.
 - Responsive web interface for visitor management and pass generation.
- **Not Included:**
 - Advanced image processing techniques beyond basic OCR.
 - Mobile application development.
 - Integration with external visitor management systems.

Limitations or constraints considered during development:

- Accuracy of OCR depends on the quality of the input images.
- Limited to specific types of documents (e.g., Aadhaar cards, PAN cards).
- Scalability issues with a large number of simultaneous visitors.

Requirements

Functional Requirements:

- Ability to upload images of visitor documents.
- Extract and parse relevant information from the uploaded images.
- Generate dynamic QR codes containing visitor details.
- Store visitor information in the database.
- Provide a web interface for viewing and managing visitor data.
- Allow administrators to download visitor passes as PDFs.

Non-functional Requirements:

- High accuracy in OCR extraction.
- Secure storage and handling of visitor data.
- Responsive and user-friendly web interface.
- Reliable performance under peak loads.

User Stories/Use Cases:

- **Visitor Uploads Document:**
 - As a visitor, I want to upload an image of my document so that my information can be extracted and a pass can be generated.
- **Admin Manages Visitors:**
 - As an admin, I want to view and manage visitor data to ensure the smooth operation of the visitor management system.
- **Generate QR Code:**
 - As a system, I want to generate a dynamic QR code for each visitor pass to streamline the check-in process.

Technical Stack

- **Programming Languages:**

- Python
- JavaScript

- **Frameworks/Libraries:**

- Django
- OpenCV (cv2)
- pytesseract
- qrcode
- xhtml2pdf

- **Databases:**

- MySQL

- **Tools/Platforms:**

- GitHub
- PyCharm
- XAMPP
- phpMyAdmin

Architecture/Design

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

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

Key components:Image Capture Module

1. OCR Integration Module
2. Data Extraction Module
3. Data Integration Module

4. 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

Technologies and frameworks used:

- Django for backend development.
- OpenCV and Pytesseract for OCR processing.
- qrcode library for generating QR codes.
- MySQL for database management.
- HTML, CSS, JavaScript for frontend development.

Overview of coding standards and best practices:

- Followed PEP 8 guidelines for Python code.
- Used Django's built-in security features to protect against common vulnerabilities.
- Employed version control (Git) to manage codebase changes and collaboration.

Challenges encountered:

- **OCR Accuracy:** Ensuring high accuracy in text extraction required preprocessing steps to enhance image quality.
- **QR Code Generation:** Managing dynamic QR codes and ensuring they remain valid for the required duration.
- **Database Integration:** Efficiently storing and querying large amounts of visitor data.

Testing

Testing approach:

- **Unit Tests:** Conducted on individual modules to ensure they function correctly.
- **Integration Tests:** Verified the interactions between different components (e.g., OCR module and database).
- **System Tests:** Tested the complete system to ensure all requirements are met.

Results of the testing phase:

- Successfully identified and resolved several bugs related to OCR extraction.
- Improved the performance and reliability of the QR code generation process.
- Ensured the system could handle peak loads without significant performance degradation.

Deployment

Deployment process:

- **Development Environment:** Used XAMPP for local testing and development.
- **Production Environment:** Deployed on a web server with MySQL database support.
- **Automation:** Created deployment scripts to automate the setup and configuration of the application.

Instructions for deployment:

1. Clone the repository:

```
git clone https://github.com/Springboard-Internship-2024/VisiOCR_May_2024.git
```

2. Install dependencies:

```
pip install -r requirements.txt
```

3. Set up the database:

Configure MySQL and import the database schema.

4. Run migrations:

```
python manage.py migrate
```

5. Start the server:

```
python manage.py runserver
```

User Guide

Instructions for Using the Application

1. Setup and Configuration

Prerequisites:

- Python 3.x
- Django 3.x
- MySQL server
- Tesseract OCR

Steps to Set Up the Project:

5. Clone the Repository

```
git clone https://github.com/Springboard-Internship-2024/VisiOCR_May_2024.git
cd VisiOCR_May_2024
```

6. Create a Virtual Environment

```
python -m venv venv
`venv\Scripts\activate`
```

7. Install Required Packages

```
pip install -r requirements.txt
```

8. Configure MySQL Database

- Create a MySQL database named VISIOCR.
- Update the DATABASES setting in settings.py with your MySQL credentials.

```
DATABASES = {
    'default': {
        'ENGINE':
'django.db.backends.mysql',
        'NAME': 'VISIOCR',
        'USER': 'your_username',
        'PASSWORD': 'your_password',
```

```
        'HOST': 'localhost',  
        'PORT': '3306',  
    }  
}
```

9. Run Migrations

```
python manage.py makemigrations  
python manage.py migrate
```

10. Start the Django Development Server

```
python manage.py runserver
```

2. Using the Application

Accessing the Application:

- Open your web browser and navigate to <http://127.0.0.1:8000/>.

Steps for Uploading an Image and Generating a Visiting Pass:

2. Home Page

- Navigate to the home page of the application.
- Click on the "Upload Image" button to go to the image upload page.

3. Upload Image Page

- On the upload page, fill out the form with the required details:
 - Select the image file to be uploaded.
 - Enter the visit date.
 - Specify the duration of the visit.
- Click on the "Submit" button to upload the image and process it.

4. Processing the Image

- The application will process the uploaded image, extracting the necessary information using the OCR engine.
- The extracted data will be stored in the MySQL database.
- A QR code will be generated containing the extracted information and other relevant details.

5. Viewing the Generated QR Code

- After the image is processed, you will be redirected to a page displaying the generated QR code.

- The page will also display the visitor's details extracted from the image.

6. Downloading the Visitor Pass

- On the QR code page, there will be an option to download the visitor pass as a PDF document.
- Click on the "Download Visitor Pass" button to download the PDF.

Troubleshooting Tips for Common Issues:

1. Database Connection Errors

- Ensure that your MySQL server is running.
- Double-check the database credentials in `settings.py`.

2. Tesseract OCR Errors

- Make sure Tesseract is installed and its path is correctly set in the environment variables.

3. Image Upload Issues

- Verify that the uploaded image is in a supported format (e.g., JPEG, PNG).
- Ensure the image file size is within acceptable limits.

4. QR Code Generation Problems

- Check if the QR code library is correctly installed.
- Ensure the extracted data is valid and correctly formatted.

Conclusion

Summary of the Project's Outcomes and Achievements:

The VisiOCR project successfully achieved its primary objectives of streamlining the visitor management process through the implementation of OCR technology and QR code generation. Key outcomes include:

- **Efficient Image Processing and Data Extraction:** Implemented a robust system for capturing and processing images, extracting relevant data using Tesseract OCR.
- **Secure Data Management:** Established a secure MySQL database to store and manage visitor information, ensuring data integrity and accessibility.
- **User-Friendly Interface:** Developed an intuitive web interface for users to upload images, generate visitor passes, and download them as PDFs.
- **QR Code Generation:** Enabled quick and secure verification of visitor information through dynamically generated QR codes embedded in the visitor passes.

- **Seamless Integration:** Achieved smooth integration of various modules, including image capture, OCR processing, data extraction, and QR code generation, providing a cohesive user experience.

Reflections on Lessons Learned and Areas for Improvement:

Lessons Learned:

- **Importance of Image Quality:** The accuracy of OCR processing heavily depends on the quality of the uploaded images. Ensuring high-quality images significantly improves data extraction accuracy.
- **Scalability Considerations:** Designing the system with scalability in mind is crucial. Handling larger volumes of data and concurrent user requests efficiently requires careful planning and optimization.
- **User Feedback:** Regular feedback from users is invaluable for identifying usability issues and areas for enhancement. Iterative development and continuous user testing can lead to a more refined and user-friendly application.

Areas for Improvement:

- **Enhanced Error Handling:** Implement more comprehensive error handling and validation mechanisms to manage unexpected inputs and system errors gracefully.
- **Mobile Optimization:** Develop a mobile-friendly version of the application to cater to users accessing the platform from smartphones and tablets.
- **Advanced Data Analytics:** Integrate advanced data analytics tools to provide insights and reports on visitor trends, enhancing decision-making processes for administrators.
- **Additional Document Types:** Expand the system's capability to process and recognize a wider range of identification documents to increase its versatility and applicability.
- **Performance Optimization:** Continuously monitor and optimize the performance of the system to ensure quick response times and efficient resource utilization.

In conclusion, the VisiOCR project demonstrated the potential of combining OCR technology with web-based applications to create an efficient and secure visitor management system. By reflecting on the lessons learned and focusing on areas for improvement, future iterations of the project can further enhance its functionality and user experience.

Appendices

Sample Code Snippets

Django Model for Visitor:

```
from django.db import models

class Visitor(models.Model):
    name = models.CharField(max_length=100)
    image = models.ImageField(upload_to='visitor_images/')
    visit_date = models.DateTimeField()
    duration = models.IntegerField()
    qr_code_image_data = models.TextField()

    def __str__(self):
        return self.name
```

Django Form for Visitor:

```
from django import forms

class VisitorForm(forms.Form):
    name = forms.CharField(max_length=100)
    image = forms.ImageField()
    visit_date = forms.DateTimeField()
    duration = forms.IntegerField()
```

View for Image Upload and QR Code Generation:

```
import qrcode
from io import BytesIO
import base64
from django.shortcuts import render
from .forms import VisitorForm
from .models import Visitor

def upload_image(request):
    if request.method == 'POST':
        form = VisitorForm(request.POST, request.FILES)
        if form.is_valid():
            visitor = Visitor(
                name=form.cleaned_data['name'],
                image=form.cleaned_data['image'],
                visit_date=form.cleaned_data['visit_date'],
                duration=form.cleaned_data['duration']
            )
            visitor.save()

            qr = qrcode.QRCode(
```

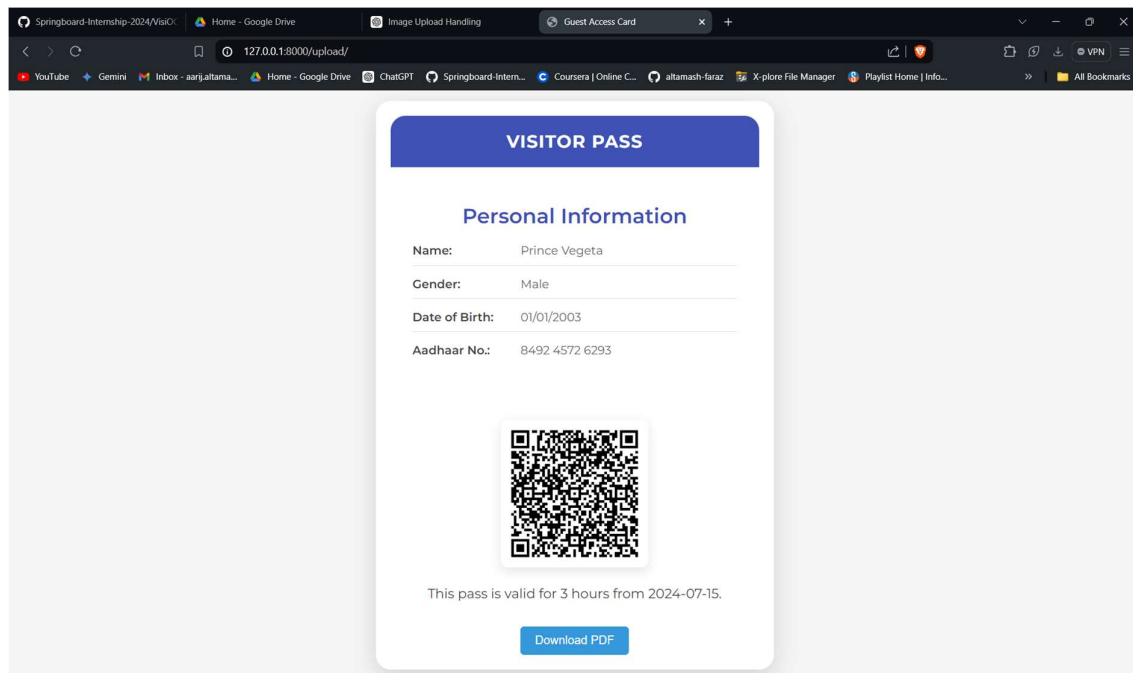
```

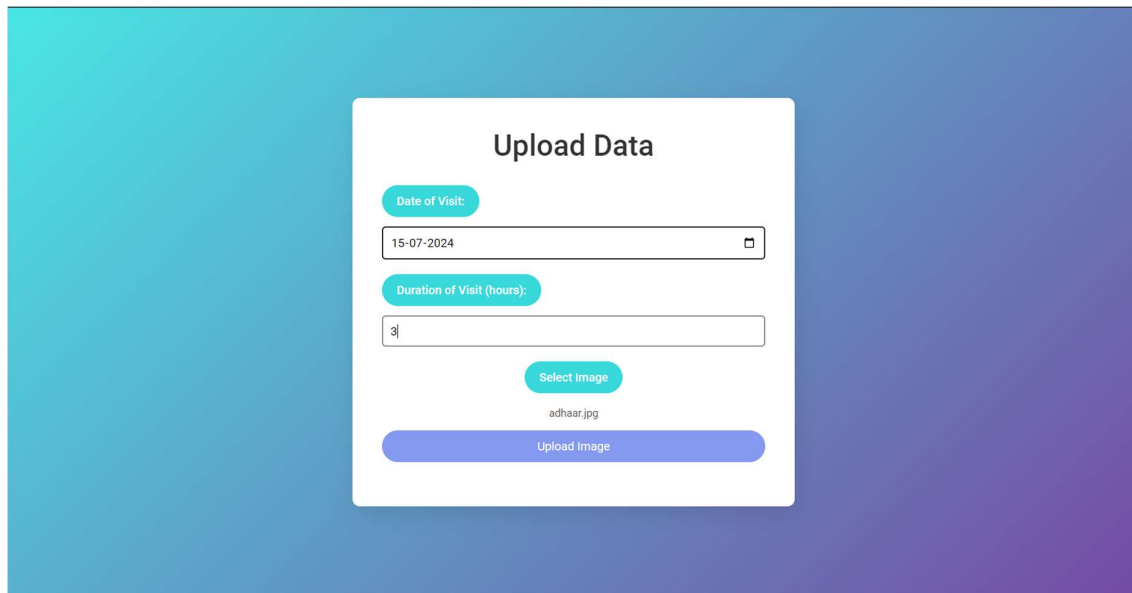
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=10,
        border=4,
    )
    qr_data = f"Name: {visitor.name}, Visit Date:
{visitor.visit_date}, Duration: {visitor.duration} hours"
    qr.add_data(qr_data)
    qr.make(fit=True)

    img=qr.make_image(fill='black',back_color='white')
    buffered = BytesIO()
    img.save(buffered, format="PNG")

img_str=base64.b64encode(buffered.getvalue()).decode()
    visitor.qr_code_image_data = img_str
    visitor.save()
    return render(request, 'pdf_template.html',
{'visitor': visitor})
    else:
        form = VisitorForm()
        return render(request, 'upload.html', {'form': form})

```





The screenshot shows a web interface with a white central card on a blue-to-purple gradient background. The card is titled 'Upload Data'. It contains two teal-colored labels: 'Date of Visit:' and 'Duration of Visit (hours):'. Below the first label is a date input field showing '15-07-2024' with a calendar icon. Below the second label is a text input field containing the number '3'. Under these fields is a teal 'Select Image' button. Below that, the filename 'adhaar.jpg' is displayed. At the bottom of the card is a wide, rounded purple button labeled 'Upload Image'.

Requirements File (requirements.txt):

```
Django==3.2.9
mysql-connector-python==8.0.27
pytesseract==0.3.8
opencv-python==4.5.3.56
xhtml2pdf==0.2.5
qrcode[pil]==7.3.1
Pillow==8.3.1
```

Research References

1. **Django Documentation:** The official Django documentation was extensively used for understanding Django models, forms, views, and deployment strategies.

(<https://docs.djangoproject.com/>)

2. **Tesseract OCR Documentation:** The Tesseract OCR documentation provided insights into configuring and using the OCR engine for text extraction from images.

(<https://github.com/tesseract-ocr/tesseract>)

3. **qrcode Library Documentation:** This resource was crucial for generating QR codes and integrating them into the Django project.

(<https://github.com/lincolnloop/python-qrcode>)

4. **MySQL Documentation:** Used for setting up and managing the MySQL database for storing visitor information.

(<https://dev.mysql.com/doc/>)

By integrating these elements, the VisiOCR project documentation provides a comprehensive overview of the project's development, implementation, and usage, ensuring it meets the specified format and detail requirements.