Healthcare Report Documentry Content Management System Application Development

Creating a healthcare report documentary Content Management System (CMS) for Windows 10 can involve building a more specialized desktop application with functionalities tailored for healthcare document management, such as uploading, viewing, editing, and managing healthcare reports.

Here’s a basic outline and source code for a simple healthcare CMS desktop application using **Python**, **Tkinter** for the GUI, and **SQLite** for the database, specifically built for Windows 10. This project will allow a user to:

 Log in to the system.

 Upload healthcare reports.

 View, edit, and delete reports.

 Export reports as PDF.

Project Structure

A screenshot of a computer

Description automatically generated

healthcare\_cms/

│

├── app.py

├── templates/

│ ├── home.html

│ ├── add\_report.html

│ ├── edit\_report.html

│ └── report\_list.html

├── reports.db

├── reports/

│ └── (uploaded reports are stored here)

└── static/

└── styles.css

**Step 1: Python Libraries**

Make sure you have the following libraries installed. You can install them using pip:

A screen shot of a computer

Description automatically generated

pip install flask

pip install flask\_sqlalchemy

pip install reportlab # For generating PDFs

Step 2: app.py - Main Application: A computer screen shot of a black screen

Description automatically generated

import os

from flask import Flask, render\_template, request, redirect, url\_for, flash, send\_from\_directory

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.utils import secure\_filename

from reportlab.lib.pagesizes import letter

from reportlab.pdfgen import canvas

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'your\_secret\_key'

app.config['SQLALCHEMY\_DATABASE\_URI'] = 'sqlite:///reports.db'

app.config['UPLOAD\_FOLDER'] = 'reports'

db = SQLAlchemy(app)

# Define the Report model

class Report(db.Model):

id = db.Column(db.Integer, primary\_key=True)

patient\_name = db.Column(db.String(100), nullable=False)

doctor\_name = db.Column(db.String(100), nullable=False)

diagnosis = db.Column(db.String(200), nullable=False)

report\_file = db.Column(db.String(200), nullable=True)

# Home page to display all reports

@app.route('/')

def home():

reports = Report.query.all()

return render\_template('home.html', reports=reports)

# Upload a new healthcare report

@app.route('/add\_report', methods=['GET', 'POST'])

def add\_report():

if request.method == 'POST':

patient\_name = request.form['patient\_name']

doctor\_name = request.form['doctor\_name']

diagnosis = request.form['diagnosis']

report\_file = request.files['report\_file']

if report\_file:

filename = secure\_filename(report\_file.filename)

report\_file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

new\_report = Report(patient\_name=patient\_name, doctor\_name=doctor\_name, diagnosis=diagnosis, report\_file=filename)

else:

new\_report = Report(patient\_name=patient\_name, doctor\_name=doctor\_name, diagnosis=diagnosis)

db.session.add(new\_report)

db.session.commit()

flash('Report added successfully')

return redirect(url\_for('home'))

return render\_template('add\_report.html')

# Edit an existing report

@app.route('/edit\_report/<int:report\_id>', methods=['GET', 'POST'])

def edit\_report(report\_id):

report = Report.query.get\_or\_404(report\_id)

if request.method == 'POST':

report.patient\_name = request.form['patient\_name']

report.doctor\_name = request.form['doctor\_name']

report.diagnosis = request.form['diagnosis']

if 'report\_file' in request.files:

report\_file = request.files['report\_file']

if report\_file:

filename = secure\_filename(report\_file.filename)

report\_file.save(os.path.join(app.config['UPLOAD\_FOLDER'], filename))

report.report\_file = filename

db.session.commit()

flash('Report updated successfully')

return redirect(url\_for('home'))

return render\_template('edit\_report.html', report=report)

# Delete a report

@app.route('/delete\_report/<int:report\_id>')

def delete\_report(report\_id):

report = Report.query.get\_or\_404(report\_id)

if report.report\_file:

os.remove(os.path.join(app.config['UPLOAD\_FOLDER'], report.report\_file))

db.session.delete(report)

db.session.commit()

flash('Report deleted successfully')

return redirect(url\_for('home'))

# View report as PDF

@app.route('/generate\_pdf/<int:report\_id>')

def generate\_pdf(report\_id):

report = Report.query.get\_or\_404(report\_id)

pdf\_file\_path = f'reports/{report.patient\_name}\_report.pdf'

# Generate PDF using ReportLab

c = canvas.Canvas(pdf\_file\_path, pagesize=letter)

c.drawString(100, 750, f"Patient Name: {report.patient\_name}")

c.drawString(100, 730, f"Doctor Name: {report.doctor\_name}")

c.drawString(100, 710, f"Diagnosis: {report.diagnosis}")

c.save()

return send\_from\_directory(directory='reports', filename=f'{report.patient\_name}\_report.pdf')

# Run the application

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Step 3: HTML Templates**

1. **home.html**

A screenshot of a computer program

Description automatically generated

<!DOCTYPE html>

<html>

<head>

<title>Healthcare Reports</title>

</head>

<body>

<h1>Healthcare Reports</h1>

<a href="{{ url\_for('add\_report') }}">Add New Report</a>

<table>

<tr>

<th>Patient Name</th>

<th>Doctor Name</th>

<th>Diagnosis</th>

<th>Actions</th>

</tr>

{% for report in reports %}

<tr>

<td>{{ report.patient\_name }}</td>

<td>{{ report.doctor\_name }}</td>

<td>{{ report.diagnosis }}</td>

<td>

<a href="{{ url\_for('edit\_report', report\_id=report.id) }}">Edit</a> |

<a href="{{ url\_for('delete\_report', report\_id=report.id) }}" onclick="return confirm('Are you sure?')">Delete</a> |

<a href="{{ url\_for('generate\_pdf', report\_id=report.id) }}">View as PDF</a>

</td>

</tr>

{% endfor %}

</table>

</body>

</html>

1. add\_report.html

A screenshot of a computer program

Description automatically generated

<!DOCTYPE html>

<html>

<head>

<title>Add Report</title>

</head>

<body>

<h1>Add Healthcare Report</h1>

<form method="POST" enctype="multipart/form-data">

<label for="patient\_name">Patient Name:</label>

<input type="text" name="patient\_name" required>

<br>

<label for="doctor\_name">Doctor Name:</label>

<input type="text" name="doctor\_name" required>

<br>

<label for="diagnosis">Diagnosis:</label>

<input type="text" name="diagnosis" required>

<br>

<label for="report\_file">Upload Report:</label>

<input type="file" name="report\_file">

<br>

<button type="submit">Add Report</button>

</form>

</body>

</html>

**Step 4: Running the Application**

1. Create the database and tables:

A black rectangular object with a black border

Description automatically generated

from app import db

db.create\_all()

1. Start the Flask application:

A black rectangular object with a black stripe

Description automatically generated

python app.py

Access the application by opening a browser and navigating to <http://127.0.0.1:5000/>.

**Features**

1. **Healthcare Reports CRUD:**
   * Add new healthcare reports.
   * Edit existing reports.
   * Delete reports.
   * Upload additional report documents as files.
   * Generate reports as PDFs.
2. **PDF Generation:**
   * Uses ReportLab to create a simple PDF version of the report that can be viewed or downloaded.
3. **Security Considerations:**
   * Ensure proper handling of uploaded files to avoid malicious uploads. You might want to restrict the file types or sanitize the filenames.
   * Implement more robust user authentication and authorization in a real-world system.

This is a foundational structure that can be expanded with more features such as report search, advanced user roles (doctors, nurses, administrators), or integration with external medical databases.