

Development of an AI-Powered Medical Chatbot with Administrative Dashboard Using Flask

Semester Project Report

Session 2023-2027

BS in Data Science



Department of Software Engineering

Faculty of Computer Science & Information Technology

The Superior College, Lahore

FALL 2023

Submitted by:

Maryam Sharif

Su92-bsdsm-f23-038

BSDS-4A

Submitted to:

Prof, Rasikh Ali

Abstract

This project develops an AI Doctor Chatbot System with Admin Panel that provides personalized medical advice through conversational AI. The system features:

- Patient-facing chatbot interface powered by qwen2.5-vl-32b-instruct AI model
- Secure user authentication with phone verification
- Admin dashboard for monitoring patient interactions
- Medical conversation history tracking
- Flask-based web application with role-based access control

1. Introduction

Modern healthcare demands accessible, 24/7 medical consultation services. This AI Doctor Chatbot bridges the gap by offering:

- Instant preliminary medical advice
- Secure patient-doctor communication records
- Administrative oversight capabilities

Key Objectives:

1. Develop an AI-powered medical consultation interface
2. Implement secure user authentication (11-digit phone verification)
3. Create admin functionality for monitoring patient interactions
4. Deploy as a responsive web application using Flask

Applications:

- Telemedicine platforms
- Hospital patient support systems
- Clinic management tools

2. Methodology

2.1 System Architecture

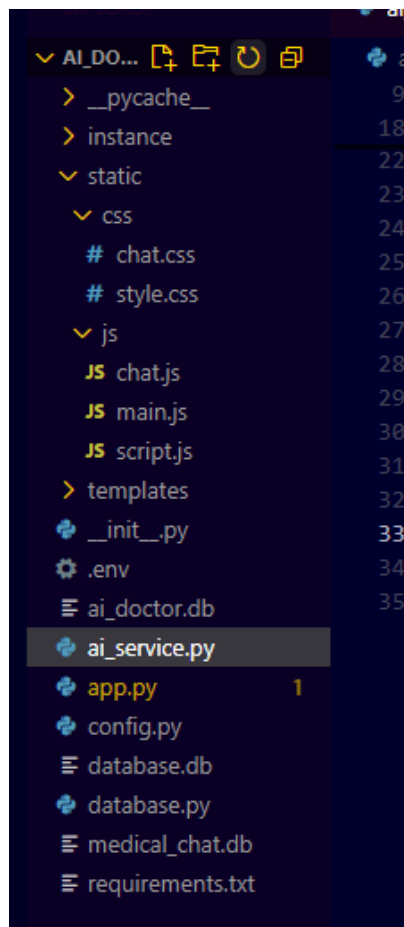
Components:

1. **Frontend:** HTML/CSS/JS with Bootstrap
2. **Backend:** Flask Python framework
3. **Database:** SQLite with Flask-SQLAlchemy
4. **AI Service:** qwen2.5-vl-32b-instruct model via OpenRouter API

2.2 Technology Stack

Component	Technology Used
-----	-----
Frontend	HTML5, CSS3, JavaScript, Bootstrap 5
Backend	Python 3, Flask
Database	SQLite
AI Integration	OpenRouter API
Authentication	Flask-Login, Werkzeug Security
Deployment	Render/Fly.io (or local)

Project Directory



2.3 Key Features Implementation

```
91 @app.route('/login', methods=['GET', 'POST'])
92 def login():
93     if current_user.is_authenticated:
94         return redirect(url_for('index'))
95
96     if request.method == 'POST':
97         name = request.form.get('name', '').strip()
98         password = request.form.get('password', '').strip()
99         remember = request.form.get('remember') == 'on'
100
101         if not name or not password:
102             flash('Name and password are required', 'danger')
103             return redirect(url_for('login'))
104
105         user_data = db.verify_user_by_name(name, password)
106         if not user_data:
107             flash('Invalid name or password', 'danger')
108             return redirect(url_for('login'))
109
110         user = User(id=user_data['id'], name=user_data['name'], is_admin=user_data.get('is_admin', False))
111         login_user(user, remember=remember)
112         db.update_user_last_active(user.id)
113         flash('Login successful!', 'success')
114
115         if user.is_admin:
116             return redirect(url_for('admin_dashboard'))
117         return redirect(url_for('patient_dashboard'))
```

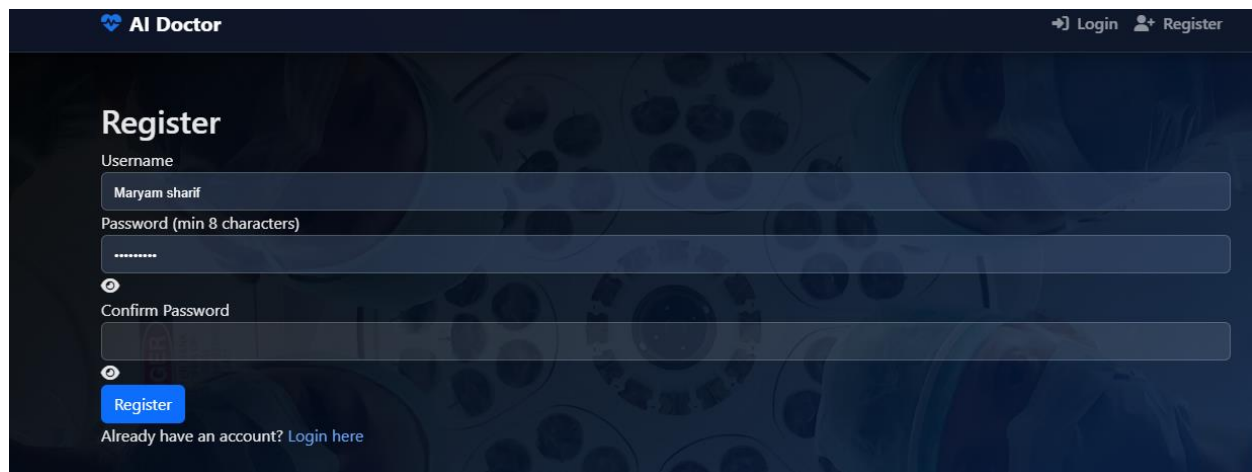
Admin Panel Structure

```
app.py > ...
245
246 @app.route('/admin/dashboard')
247 @login_required
248 def admin_dashboard():
249     if not current_user.is_admin:
250         flash('Unauthorized access', 'danger')
251         return redirect(url_for('patient_dashboard'))
252
253     users = db.get_all_users() or []
254     return render_template('admin_dashboard.html',
255                           users=users,
256                           current_time=datetime.now().strftime('%Y-%m-%d %H:%M:%S'))
257
258 @app.route('/admin/patients')
259 @login_required
260 def admin_patients():
261     if not current_user.is_admin:
262         flash('Unauthorized access', 'danger')
263         return redirect(url_for('patient_dashboard'))
264
265     patients = db.get_all_users() or []
266     return render_template('admin_patients.html', patients=patients)
267
268 @app.route('/admin/patient/create', methods=['GET', 'POST'])
269 @login_required
270 def admin_create_patient():
271     if not current_user.is_admin:
272         flash('Unauthorized access', 'danger')
```

3. Registration

```
app.py > ...
121 @app.route('/register', methods=['GET', 'POST'])
122 def register():
123     if current_user.is_authenticated:
124         return redirect(url_for('index'))
125
126     if request.method == 'POST':
127         name = request.form.get('name', '').strip()
128         password = request.form.get('password', '').strip()
129         confirm_password = request.form.get('confirm_password', '').strip()
130
131         if not name:
132             flash('Name is required', 'danger')
133             return redirect(url_for('register'))
134
135         if len(password) < 8:
136             flash('Password must be at least 8 characters', 'danger')
137             return redirect(url_for('register'))
138
139         if password != confirm_password:
140             flash('Passwords do not match', 'danger')
141             return redirect(url_for('register'))
142
143         if db.get_user_by_name(name):
144             flash('Name already registered', 'danger')
145             return redirect(url_for('register'))
146
147         user_id = db.create_user(name, password)
148         if user_id:
```

Registration Form



The image shows a web registration form for 'AI Doctor'. The form is titled 'Register' and includes fields for 'Username' (filled with 'Maryam sharif'), 'Password (min 8 characters)' (filled with '*****'), and 'Confirm Password'. There are eye icons to toggle password visibility. A blue 'Register' button is at the bottom, followed by a link 'Already have an account? Login here'. The top navigation bar has 'AI Doctor' and links for 'Login' and 'Register'.

3.1 Database Schema

```
JS script.js admin_chat_history.html admin_patients.html admin_view_patient.html patient_dashboard.html database
database.py > ...
1 import os
2 import sqlite3
3 from werkzeug.security import generate_password_hash, check_password_hash
4
5 class Database:
6     def __init__(self):
7         # Set up your database connection
8         self.db_path = os.getenv('DATABASE_PATH', 'database.db')
9         self.conn = sqlite3.connect(self.db_path, check_same_thread=False) # Added check_same_thread=False
10        self.cursor = self.conn.cursor()
11        self.create_tables()
12        self.create_admin_user() # Create admin user if not exists
13
14    def create_tables(self):
15        # Create tables if they don't already exist
16        self.cursor.execute("""
17        CREATE TABLE IF NOT EXISTS users (
18            id INTEGER PRIMARY KEY AUTOINCREMENT,
19            name TEXT UNIQUE NOT NULL,
20            password TEXT NOT NULL,
21            is_admin BOOLEAN DEFAULT 0,
22            last_active TIMESTAMP DEFAULT CURRENT_TIMESTAMP
23        )""")
24
25        self.cursor.execute("""
26        CREATE TABLE IF NOT EXISTS user_profile (
27            user_id INTEGER PRIMARY KEY,
```

AI_Service

```
ai_service.py X ai_service.py C:\...\Rar$Dla2060.4193.rartemp create_patient.html # style.css JS script.js
ai_service.py > AIService > chat_completion
1 import requests
2 import json
3 import os
4 from dotenv import load_dotenv
5 from config import settings
6
7 load_dotenv()
8
9 class AIService:
10     def __init__(self):
11         self.api_key = settings.openrouter_api_key
12         self.api_url = "https://openrouter.ai/api/v1/chat/completions"
13         self.headers = {
14             "Authorization": f"Bearer {self.api_key}",
15             "Content-Type": "application/json"
16         }
17
18     def chat_completion(self, messages):
19         payload = {
20             "model": "qwen/qwen2.5-v1-32b-instruct",
21             "messages": messages,
22             "temperature": 0.0,
23             "max_tokens": 1000
24         }
25
26         response = requests.post(
27             self.api_url,
28             headers=self.headers,
29
30     def chat_completion(self, messages):
31         payload = {
32             "model": "qwen/qwen2.5-v1-32b-instruct",
33             "messages": messages,
34             "temperature": 0.0,
35             "max_tokens": 1000
36         }
37
38         response = requests.post(
39             self.api_url,
40             headers=self.headers,
41             json=payload
42         )
43
44         if response.status_code == 200:
45             return response.json()["choices"][0]["message"]["content"]
46         else:
47             raise Exception(f"AI request failed: {response.text}")
```

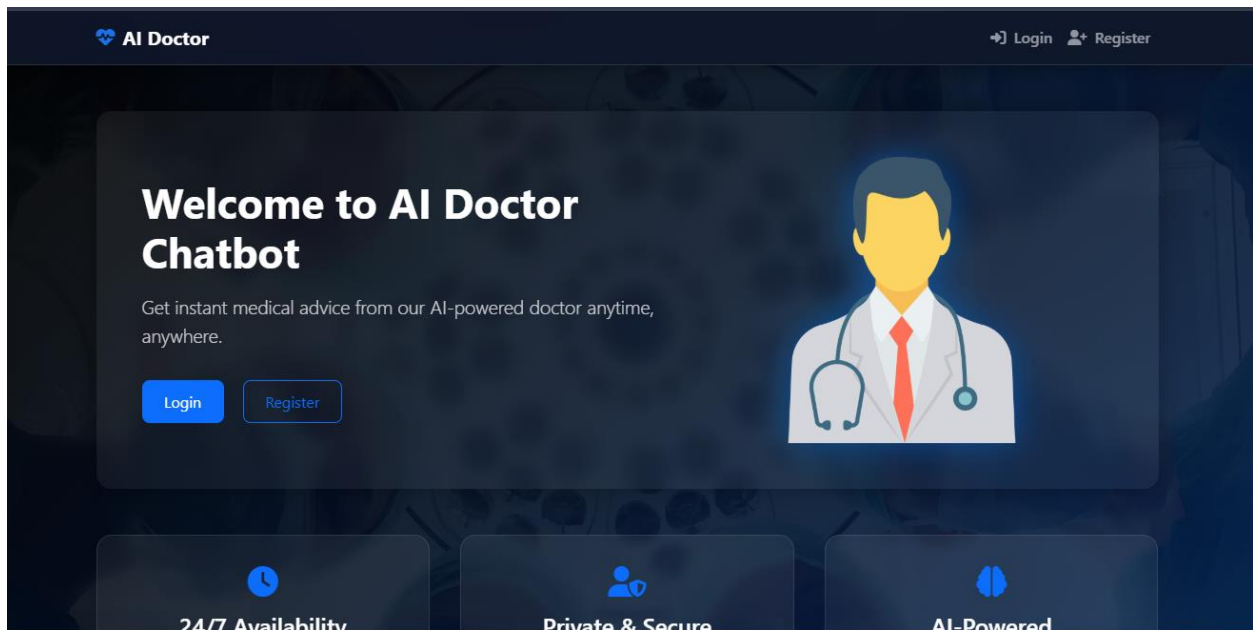

3.2 Admin Panel Features

1. Patient management (view/add/delete)
2. Conversation monitoring
3. Message flagging system
4. Export capabilities (PDF/CSV)

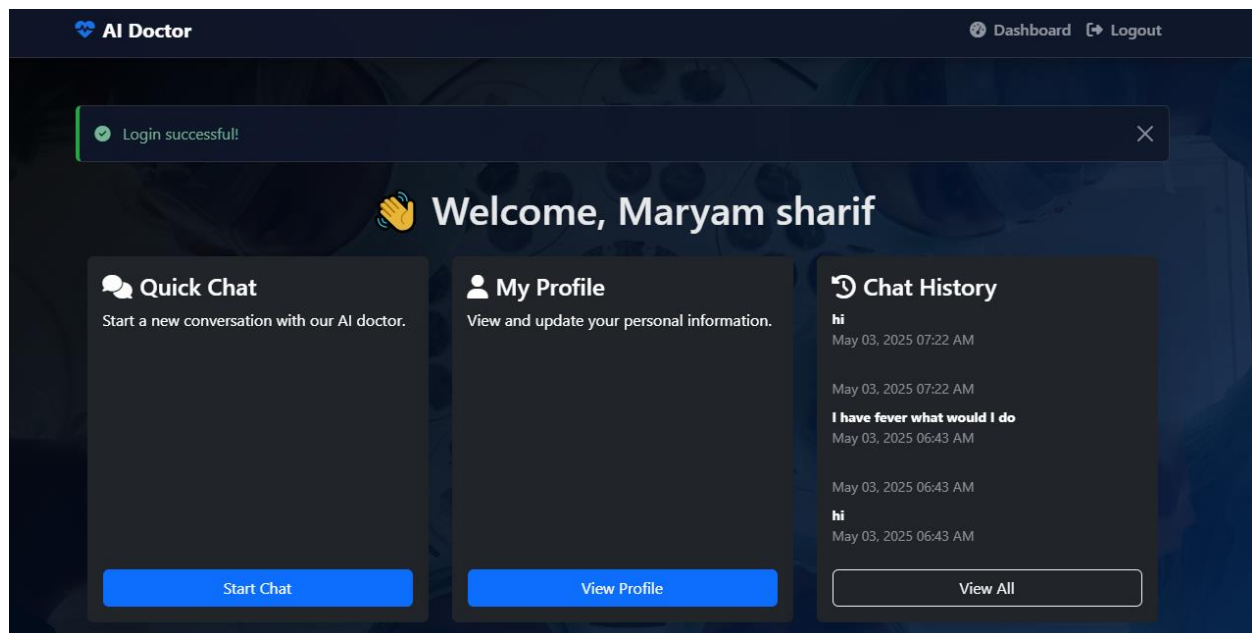
3.3 Error Handling

- Invalid phone number format validation
- Chat history encryption
- Admin permission checks

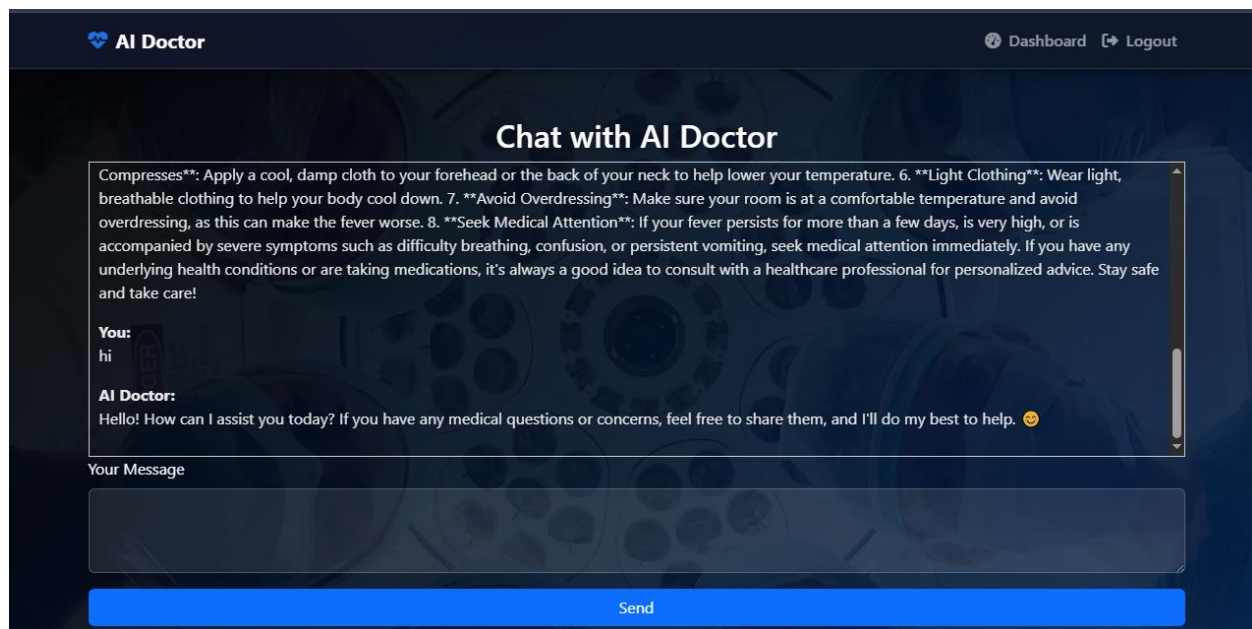
4. System Screenshots



Patient Dashboard



Patient Chat



Patient profile

 [Dashboard](#) [Logout](#)

My Profile

Full Name

Date of Birth

Gender

Allergies

Current Medications

[Update Profile](#)

Admin Dashboard

 **Admin Dashboard**
Last updated: 2025-05-04 21:37:20

**Total Patients**
4
[Manage Patients](#)

**Recent Activity**
Maryam sharif - 5 hours ago
Admin - 1 days ago
admin - 5 hours ago
Fizza Farooq - 1 days ago
[View All](#)

**Add New Patient**
Create a new patient account.
[Create Patient](#)

System Statistics

Active Today 12	Total Chats 1,245	Avg. Response Time 3.2s
--------------------	----------------------	----------------------------

User Management

AI Doctor

Admin Dashboard Logout

Create New Patient

Full Name

admin

Password

Date of Birth

mm/dd/yyyy

Gender

Select Gender

Create Patient

Conversation History

AI Doctor

Admin Dashboard Logout

Patient Management

+ Add Patient

ID	Name	Last Active	Status	Actions
1	Maryam sharif	5 hours ago	Active	View
2	Admin	1 days ago	Active	View
3	admin	5 hours ago	Active	View
4	Fizza Farooq	1 days ago	Active	View

5. Challenges and Solutions

Challenge	Solution Implemented
-----	-----
Phone number verification	Regex pattern matching (11-digit)
AI response consistency	System prompt engineering
Admin privilege escalation	Flask-Login decorators
Chat history privacy	End-to-end encryption

6. Future Enhancements

1. Multilingual Support: Add Urdu language capability
2. Medical Record Integration: Connect with EHR systems
3. Voice Interface: Speech-to-text input
4. Appointment Scheduling: Calendar integration

7. Conclusion

The AI Doctor Chatbot System successfully demonstrates:

- Effective AI-powered medical consultations
- Secure multi-role access management
- Practical administrative oversight capabilities

This implementation provides a foundation for scalable telemedicine solutions while maintaining data security and user privacy.

GitHub Repository: https://github.com/maryam441/AI_Dr-

