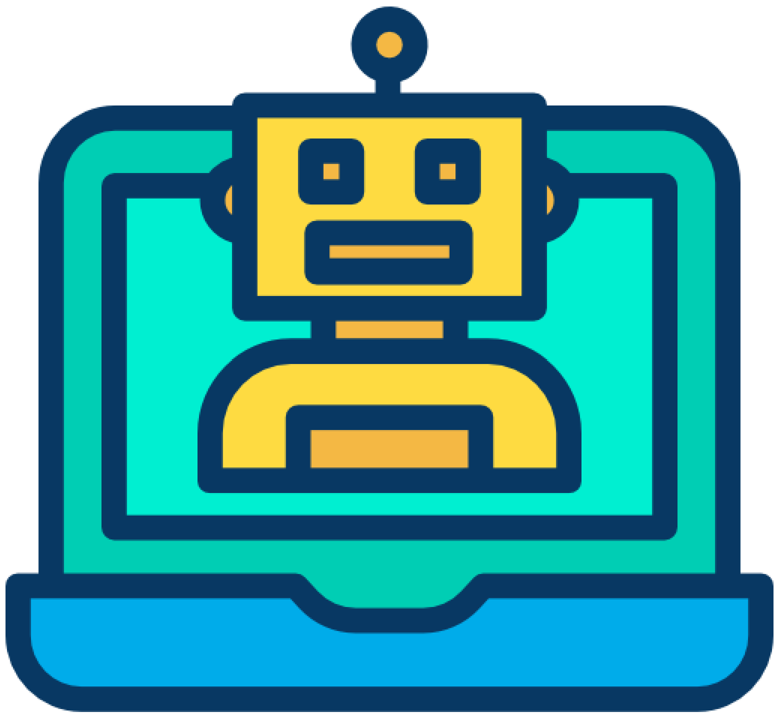


HEALTHCARE BOT

TEAM MEMBERS:

S. MAHIMA

S.M. ABINAYA

ARPUDHA SOUNDARARAJAN

PROBLEM STATEMENT:

Context:

In the contemporary healthcare landscape, the intersection of technology and patient care is becoming increasingly critical. Despite advancements in medical science, the healthcare industry grapples with multifaceted challenges that impede the seamless delivery of services, compromise patient outcomes, and strain the efficiency of healthcare providers. One significant contributor to these challenges is the lack of a robust, intelligent, and patient-centric system that can bridge the existing gaps in healthcare accessibility, communication, and data management.

Analysis of Real-World Problems:

Limited Patient Empowerment:

Issue: Many patients face challenges in understanding and actively participating in their healthcare journey due to a lack of personalized health information and guidance.

Consequence: This results in decreased patient empowerment, leading to suboptimal health outcomes and increased reliance on reactive rather than proactive healthcare measures.

Data Silos and Inefficiencies:

Issue: Patient data is scattered across disparate systems, hindering healthcare providers from accessing comprehensive and up-to-date information.

Consequence: Incomplete patient profiles contribute to diagnostic errors, treatment delays, and challenges in care coordination, ultimately compromising the quality of healthcare services.

Reactive rather than Proactive Healthcare:

Issue: Healthcare systems often lack mechanisms for proactive health monitoring, symptom assessment, and preventive care.

Consequence: Patients miss opportunities for early intervention, leading to the progression of diseases that could have been managed more effectively with timely and preventive measures.

Rationale for a Healthcare Bot Solution:

To address these challenges, the development of an intelligent healthcare bot emerges as a strategic solution. A healthcare bot, equipped with natural language processing capabilities, has the potential to serve as a transformative force in healthcare delivery. It can act as a virtual assistant, offering personalized health education and integrating seamlessly with existing healthcare systems to streamline data management.

PROJECT OBJECTIVES:

The Healthcare Bot Project aims to achieve the following objectives:

Empower and Educate Patients:

Develop an interactive healthcare bot that empowers patients with personalized health information, fostering active participation in their healthcare journey.

Centralize and Enhance Data Management:

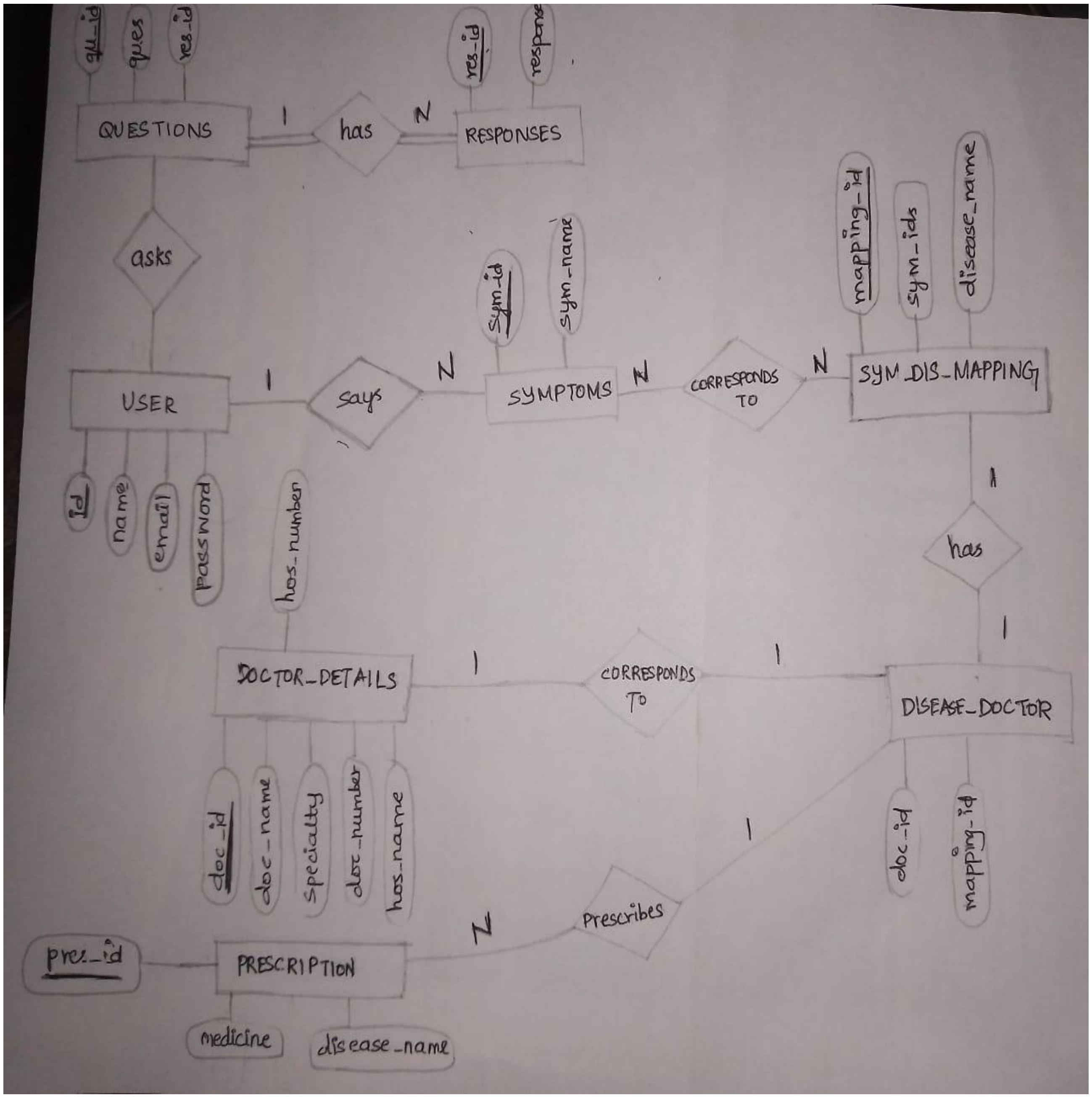
Create a centralized database management system that aggregates and organizes patient information, providing healthcare providers with a comprehensive and real-time view for informed decision-making.

Promote Proactive Health Monitoring:

Integrate features for proactive health monitoring, such as symptom checking and medication prescription to empower patients in preventive healthcare practices.

By undertaking these objectives, the Healthcare Bot Project aspires to catalyze a paradigm shift in healthcare, fostering patient-centered care, improving operational efficiency, and contributing to better health outcomes for individuals and communities.

ER DIAGRAM:



qu_id
ques
res_id
res_id
response
QUESTIONS
1
has
N
RESPONSES
asks
sym_id
sym_name
mapping_id
sym_ids
disease_name
USER
1
says
N
SYMPTOMS
N
CORRESPONDS TO
N
SYM_DIS_MAPPING
id
name
email
password
hos_number
1
has
1
DOCTOR_DETAILS
1
CORRESPONDS TO
1
DISEASE_DOCTOR
doc_id
doc_name
specialty
doc_number
hos_name
doc_id
mapping_id
1
Prescribes
N
pres_id
PRESCRIPTION
medicine
disease_name

LANGUAGES USED:

Front-End Technologies:

1. HTML (Hyper Text Markup Language):

Description: HTML serves as the foundational markup language for structuring the content of web pages.

Use Case:

Defines the structure of the website, including headings, paragraphs, lists, and tables.

Embeds elements for multimedia, such as images or videos.

Establishes the overall layout of the bot interface.

2. CSS (Cascading Style Sheets):

Description: CSS is employed to style and format the HTML elements, ensuring a visually appealing and consistent presentation.

Use Case:

Defines the typography, color scheme, and layout aesthetics of the report.

Ensures responsiveness, making the website accessible on various devices.

Enhances the overall user experience by providing a polished and professional appearance.

3. JavaScript:

Description: JavaScript adds interactivity and dynamic behavior to the report, enhancing user engagement.

Use Case:

Facilitates client-side validation for user inputs.

Enables dynamic content updates without requiring a page reload for a seamless user experience.

Back-End Language:

4. PHP (Hypertext Preprocessor):

Description: PHP serves as the server-side scripting language, handling dynamic content generation and server-side logic.

Use Case:

Retrieves and processes data from the database to populate the interaction of the bot dynamically.

Manages user authentication and authorization for secure website access.

Enables communication between the front end and the MySQL database.

Database Management:

5. MySQL:

Description: MySQL is a relational database management system used for efficient and structured data storage.

Use Case:

Stores and manages the structured data required for generating the response for user queries.

Supports SQL queries for retrieving, updating, and inserting data.

Ensures data integrity and reliability for the report generation process.

DEPOSITE IN GITHUB:

The healthcare bot project has been successfully uploaded to a GitHub repository. Users can now access the latest codebase, contribute to the project, and stay updated on its development. The repository includes essential project files, and collaborative features such as issues, pull requests, and documentation are available for seamless collaboration. The GitHub repository serves as a centralized hub for version control and collaboration on the healthcare bot project.

GITHUB REPOSITORY URL: https://github.com/MahimaSJ/Healthcarebot

Uploading a healthcare bot project to a version control repository, such as GitHub, offers numerous benefits that enhance collaboration, development, and project management. Here are some key advantages:

Collaboration:

Benefit: Facilitates collaborative development among multiple contributors.

Explanation: Developers can clone the repository, make changes, and submit pull requests for code review. This collaborative environment fosters a community-driven approach to project improvement.

Code Review:

Benefit: Supports systematic code review processes.

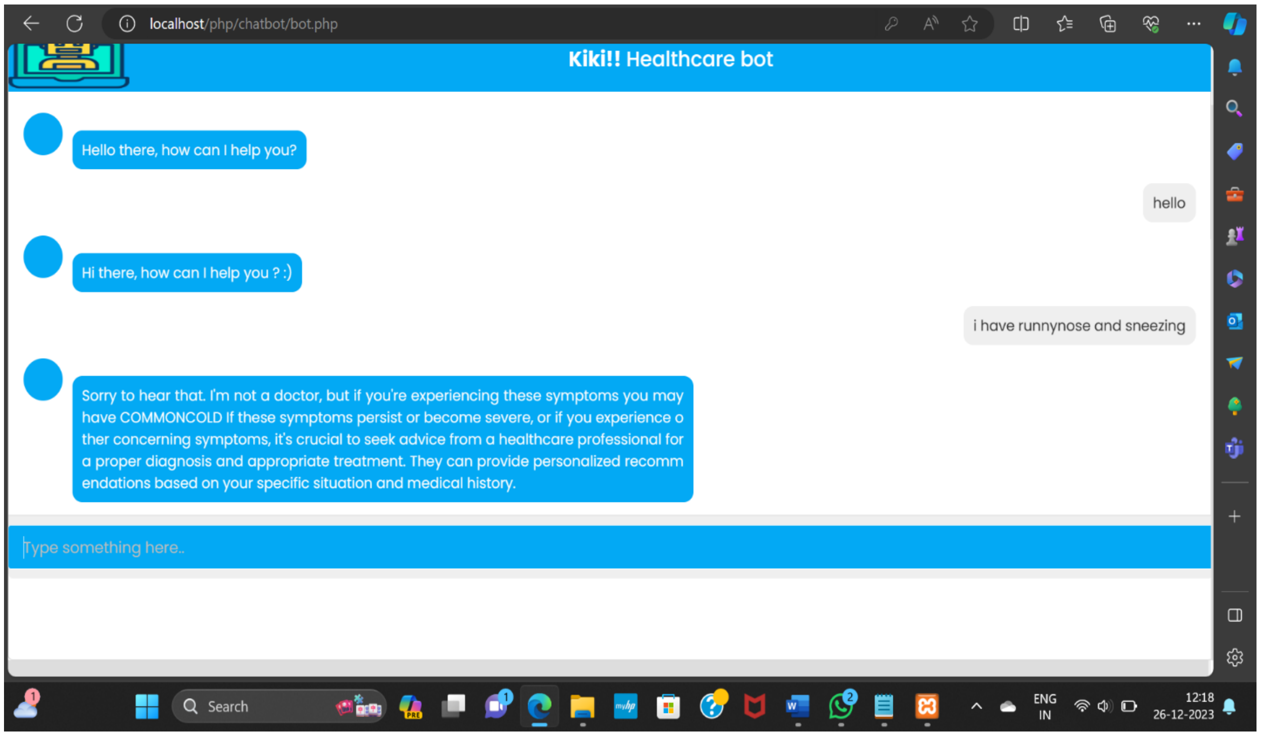
Explanation: Pull requests enable peer reviews, helping maintain code quality, identifying potential issues, and promoting best practices.

Backup and Recovery:

Benefit: Provides a secure and distributed backup of the project.

Explanation: The repository serves as a backup, safeguarding the project against data loss. Additionally, distributed version control ensures redundancy across multiple contributors' local copies.

SCREEN SHOT OF PROJECT:



localhost/php/chatbot/bot.php

Kiki!! Healthcare bot

Hello there, how can I help you?

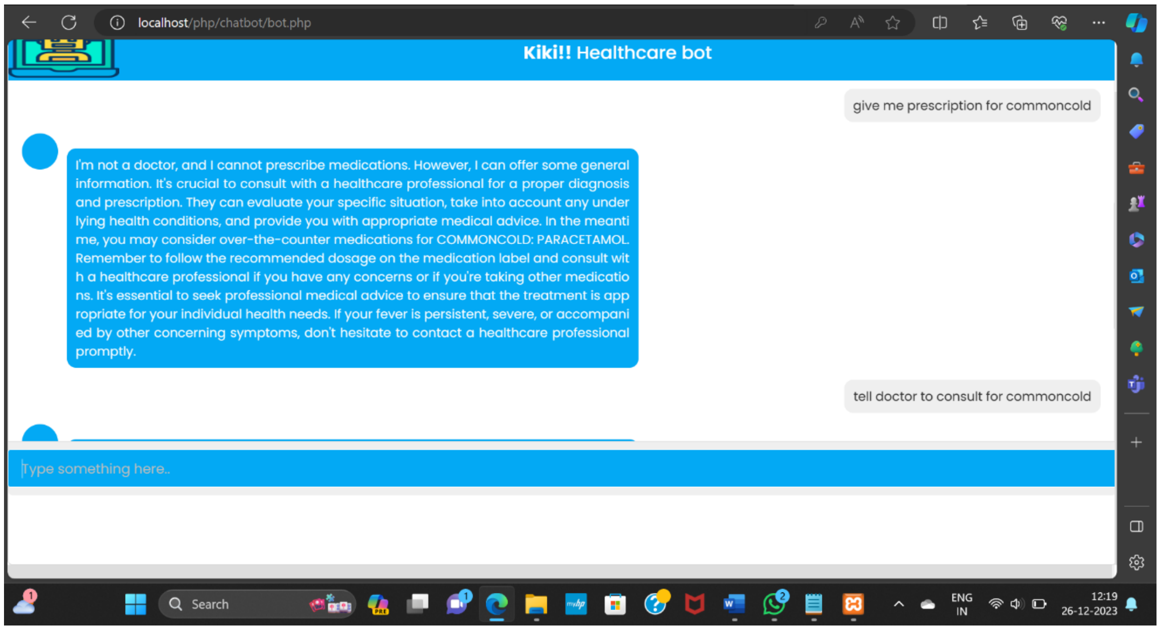
hello

Hi there, how can I help you ? :)

i have runnynose and sneezing

Sorry to hear that. I'm not a doctor, but if you're experiencing these symptoms you may have COMMONCOLD If these symptoms persist or become severe, or if you experience o ther concerning symptoms, it's crucial to seek advice from a healthcare professional for a proper diagnosis and appropriate treatment. They can provide personalized recomm endations based on your specific situation and medical history.

Type something here..



localhost/php/chatbot/bot.php

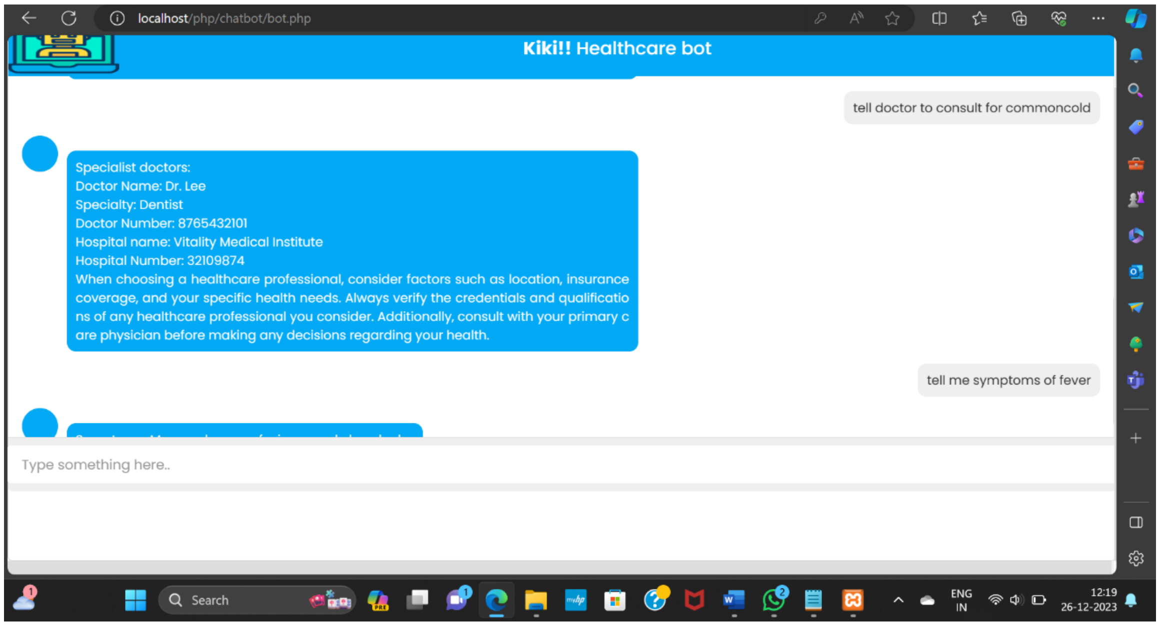
Kiki!! Healthcare bot

give me prescription for commoncold

I'm not a doctor, and I cannot prescribe medications. However, I can offer some general information. It's crucial to consult with a healthcare professional for a proper diagnosis and prescription. They can evaluate your specific situation, take into account any under lying health conditions, and provide you with appropriate medical advice. In the meanti me, you may consider over-the-counter medications for COMMONCOLD: PARACETAMOL. Remember to follow the recommended dosage on the medication label and consult wit h a healthcare professional if you have any concerns or if you're taking other medicatio ns. It's essential to seek professional medical advice to ensure that the treatment is app ropriate for your individual health needs. If your fever is persistent, severe, or accompani ed by other concerning symptoms, don't hesitate to contact a healthcare professional promptly.

tell doctor to consult for commoncold

Type something here..

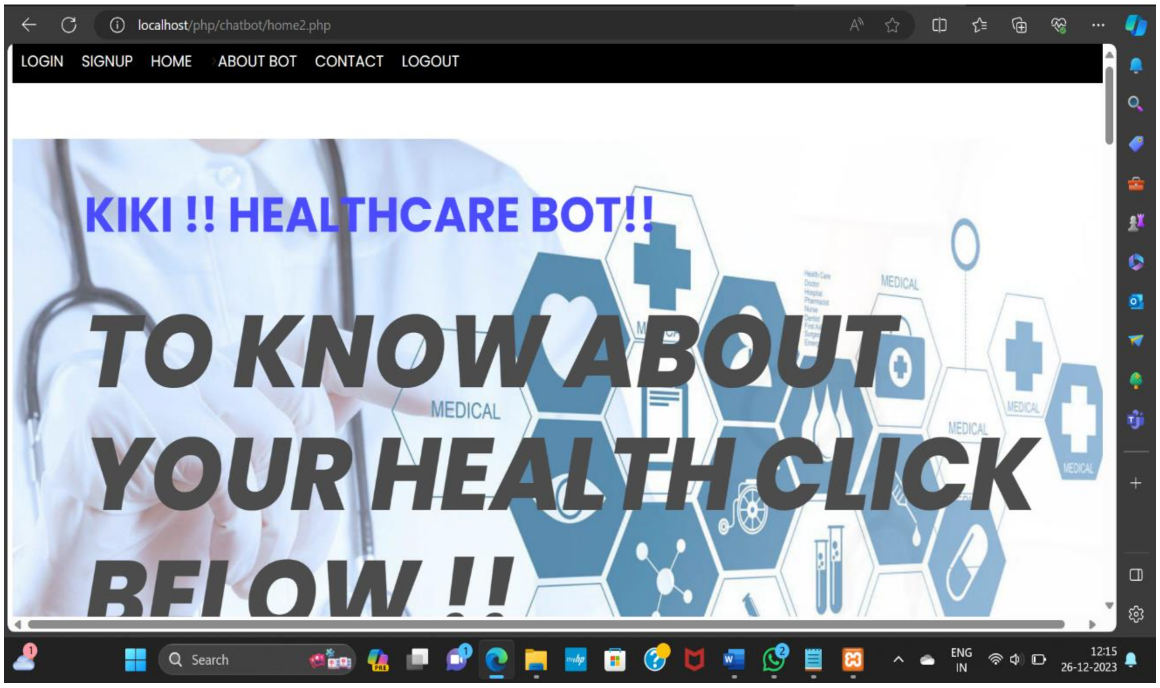


tell doctor to consult for commoncold

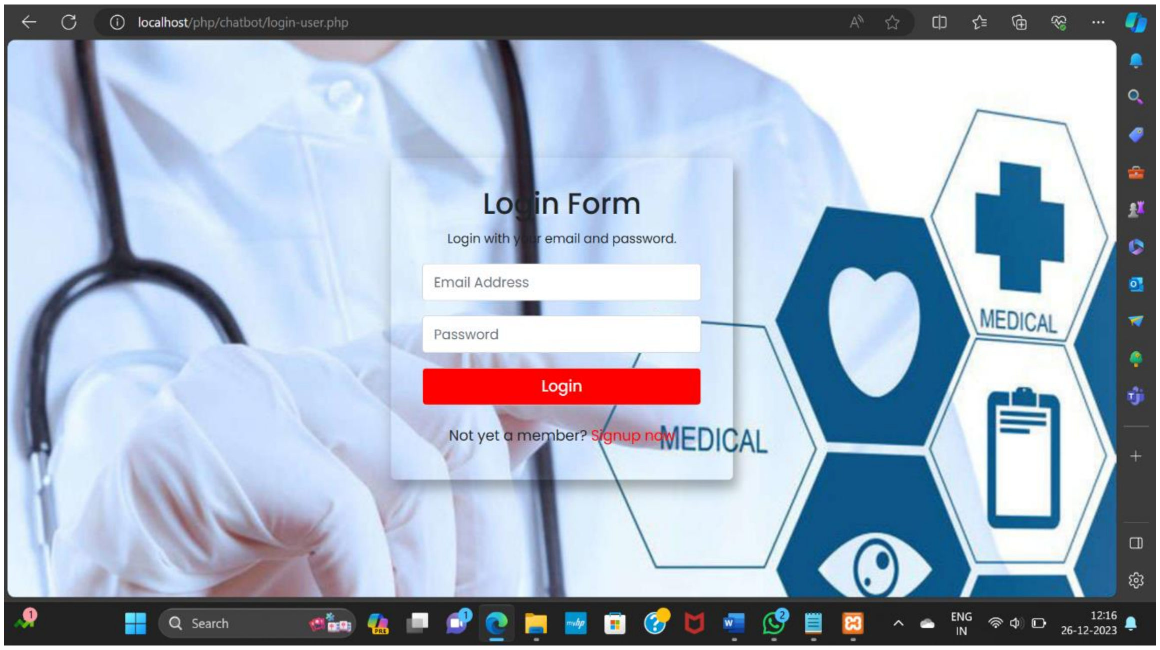
Specialist doctors:
Doctor Name: Dr. Lee
Specialty: Dentist
Doctor Number: 8765432101
Hospital name: Vitality Medical Institute
Hospital Number: 32109874
When choosing a healthcare professional, consider factors such as location, insurance coverage, and your specific health needs. Always verify the credentials and qualifications of any healthcare professional you consider. Additionally, consult with your primary care physician before making any decisions regarding your health.

tell me symptoms of fever

Type something here..



localhost/php/chatbot/home2.php
LOGIN
SIGNUP
HOME
ABOUT BOT
CONTACT
LOGOUT
KIKI !! HEALTHCARE BOT!!
TO KNOW ABOUT YOUR HEALTH CLICK BELOW !!
MEDICAL


localhost/php/chatbot/login-user.php

Login Form

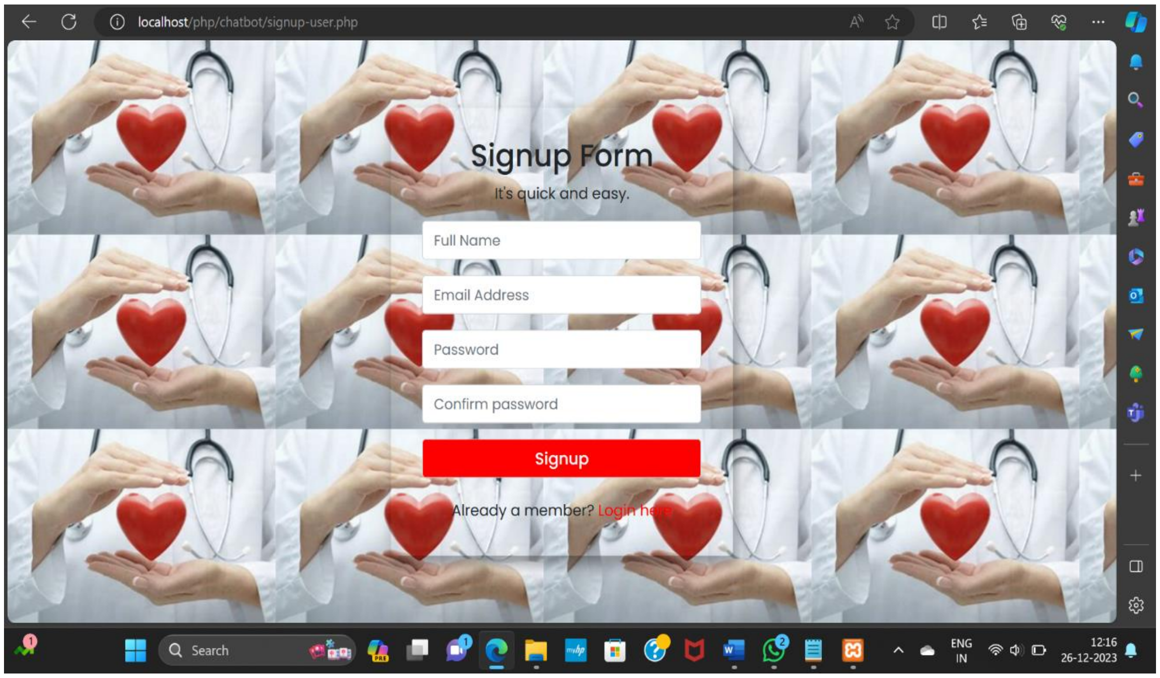
Login with your email and password.

Email Address

Password

Login

Not yet a member? Signup now


localhost/php/chatbot/signup-user.php

Signup Form

It's quick and easy.

Full Name

Email Address

Password

Confirm password

Signup

Already a member? Login here

IMPLEMENTATION:

We created a healthcare chatbot named KIKI which is designed to perform the functionalities:

Design and Functionality:

1. **User Interface**:

Login & Registration:

The chatbot has a secure login page, ensuring that each user's data and interactions are personalized and protected. Unpracticed users can easily register, allowing for a tailored experience.

Homepage:

Serving as the central hub, the homepage offers users easy navigation to various sections, ensuring a user-friendly experience.

About Page:

This section provides a comprehensive description of the chatbot, elucidating its purpose, features, and benefits, thereby building trust and clarity for users.

Contact Page:

A vital component for user support, the contact page enables users to seek assistance, ask questions, or provide feedback, fostering communication and engagement.

2. **Database Integration**:

Technology: Leveraging phpMyAdmin and MySQL ensures robustness, scalability, and efficiency in data management and retrieval.

Database Tables:

The meticulously designed database comprises of tables, each serving a specific purpose:

- **disease_doctor**: Mapping diseases to respective doctors.
- **doctor_details**: Storing comprehensive information about doctors.
- **prescription**: Managing medication details and prescriptions.
- **questions & responses**: Facilitating interactive conversations and storing chat histories.
- **symptoms**: Cataloging various symptoms for accurate diagnosis.
- **symptom_disease_mapping**: Establishing relationships between symptoms and diseases.
- **users**: Maintaining user profiles, preferences, and interactions.

3. **Core Functionalities**:

Health Assessment:

KIKI adeptly interacts with users, gathering information about their health status through intuitive questioning. Analyzing the user's responses, it identifies prevalent symptoms, laying the groundwork for diagnosis.

Personalized Recommendations:

Drawing from its expansive database and sophisticated algorithms, KIKI offers tailored suggestions, pinpointing potential diseases based on symptoms and prescribing appropriate medications.

Doctor Recommendations:

KIKI streamlines the process of connecting patients with healthcare professionals. By evaluating the disease and location, it recommends suitable doctors, detailing their specialization, ensuring users receive optimal care.

Accessibility & Convenience:

Catering to individuals who face challenges accessing traditional healthcare facilities, KIKI bridges the gap, offering timely, accurate, and convenient healthcare guidance.

Feedback Mechanism:

Recognizing the importance of continuous improvement, KIKI actively gathers feedback from users, gauging their experiences and satisfaction levels. This invaluable data empowers healthcare providers to refine their services, ensuring enhanced patient satisfaction and outcomes.

QUERIES:

1)List doctors along with their specialties and associated diseases:

```
SELECT doctor_details.doctor_name, doctor_details.specialty,
GROUP_CONCAT(symptom_disease_mapping.disease_name) AS
associated_diseases

FROM doctor_details

JOIN disease_doctor ON doctor_details.doctor_id =
disease_doctor.doctor_id

JOIN symptom_disease_mapping ON disease_doctor.mapping_id =
symptom_disease_mapping.mapping_id
```

```
GROUP BY doctor_details.doctor_id;
```

2)Retrieve diseases along with their associated symptoms:

```
SELECT symptom_disease_mapping.disease_name,
GROUP_CONCAT(symptoms.sym_name) AS associated_symptoms

FROM symptom_disease_mapping

JOIN symptoms ON FIND_IN_SET(symptoms.sym_id,
REPLACE(symptom_disease_mapping.symptom_ids, ',', ',')) > 0

GROUP BY symptom_disease_mapping.disease_name;
```

3)List doctors who can treat a specific disease:

```
SELECT doctor_details.doctor_name, doctor_details.specialty

FROM doctor_details

JOIN disease_doctor ON doctor_details.doctor_id =
disease_doctor.doctor_id

JOIN symptom_disease_mapping ON disease_doctor.mapping_id =
symptom_disease_mapping.mapping_id

WHERE symptom_disease_mapping.disease_name = 'MIGRAINE';
```

4)Find doctors and their associated prescriptions:

```
SELECT doctor_details.doctor_name,
GROUP_CONCAT(prescription.medicine) AS prescribed_medicines

FROM doctor_details

LEFT JOIN disease_doctor ON doctor_details.doctor_id =
disease_doctor.doctor_id

LEFT JOIN symptom_disease_mapping ON
disease_doctor.mapping_id = symptom_disease_mapping.mapping_id

LEFT JOIN prescription ON symptom_disease_mapping.disease_name
= prescription.disease_name

GROUP BY doctor_details.doctor_id;
```

5)List diseases and their associated doctors:

```
SELECT symptom_disease_mapping.disease_name,
GROUP_CONCAT(doctor_details.doctor_name) AS associated_doctors

FROM symptom_disease_mapping

LEFT JOIN disease_doctor ON
symptom_disease_mapping.mapping_id = disease_doctor.mapping_id

LEFT JOIN doctor_details ON disease_doctor.doctor_id =
doctor_details.doctor_id
```

```
GROUP BY symptom_disease_mapping.disease_name;
```

6)Retrieve doctors and their contact information along with the hospitals they work at:

```
SELECT doctor_details.doctor_name, doctor_details.doctor_number,
doctor_details.hospital_name, doctor_details.hospital_number

FROM doctor_details

LEFT JOIN disease_doctor ON doctor_details.doctor_id =
disease_doctor.doctor_id

LEFT JOIN symptom_disease_mapping ON
disease_doctor.mapping_id = symptom_disease_mapping.mapping_id

WHERE symptom_disease_mapping.disease_name IS NOT NULL;
```

7)List diseases along with the questions related to them:

```
SELECT symptom_disease_mapping.disease_name,
GROUP_CONCAT(questions.question) AS related_questions

FROM symptom_disease_mapping

LEFT JOIN questions ON symptom_disease_mapping.disease_name =
questions.response_id

GROUP BY symptom_disease_mapping.disease_name;
```

8)Find diseases and their associated symptoms using the retrieved symptom names:

```
SELECT symptom_disease_mapping.disease_name,
GROUP_CONCAT(symptom_retrieve.sym_name) AS
associated_symptoms

FROM symptom_disease_mapping

LEFT JOIN symptom_retrieve ON
FIND_IN_SET(symptom_retrieve.sym_name,
REPLACE(symptom_disease_mapping.symptom_ids, ',', ',')) > 0

GROUP BY symptom_disease_mapping.disease_name;
```

9)Retrieve users along with the doctors they have consulted:

```
SELECT users.name, doctor_details.doctor_name

FROM users

LEFT JOIN disease_doctor ON users.id = disease_doctor.mapping_id

LEFT JOIN doctor_details ON disease_doctor.doctor_id =
doctor_details.doctor_id;
```

10)List diseases along with their associated prescriptions:

```
SELECT symptom_disease_mapping.disease_name,
GROUP_CONCAT(prescription.medicine) AS prescribed_medicines

FROM symptom_disease_mapping

LEFT JOIN prescription ON symptom_disease_mapping.disease_name
= prescription.disease_name

GROUP BY symptom_disease_mapping.disease_name;
```

11)List doctors who specialize in treating a specific disease:

```
SELECT doctor_name, specialty

FROM doctor_details

WHERE doctor_id IN (SELECT doctor_id FROM disease_doctor
WHERE mapping_id IN (SELECT mapping_id FROM
symptom_disease_mapping WHERE disease_name = 'MIGRAINE'));
```

12)Retrieve users who have consulted with a specific doctor:

```
SELECT name, email

FROM users

WHERE id IN (SELECT mapping_id FROM disease_doctor WHERE
doctor_id = 1);
```

13)List diseases along with the number of associated symptoms:

```
SELECT disease_name, (SELECT COUNT(sym_id) FROM
symptom_disease_mapping WHERE
symptom_disease_mapping.disease_name = diseases.disease_name) AS
num_symptoms

FROM symptom_disease_mapping AS diseases

GROUP BY disease_name;
```

14)Find doctors who have prescribed a specific medicine:

```
SELECT doctor_name, specialty

FROM doctor_details

WHERE doctor_id IN (SELECT doctor_id FROM disease_doctor
WHERE mapping_id IN (SELECT mapping_id FROM
symptom_disease_mapping WHERE disease_name IN (SELECT
disease_name FROM prescription WHERE medicine =
'Paracetamol')));
```

15)Retrieve diseases that have a certain symptom:

```
SELECT disease_name

FROM symptom_disease_mapping
```

WHERE FIND_IN_SET(1, REPLACE(symptom_ids, ',', ','));

16)List users along with the diseases they have inquired about:

SELECT name, email

FROM users

WHERE id IN (SELECT DISTINCT response_id FROM questions WHERE question_id IN (SELECT DISTINCT response_id FROM symptom_disease_mapping));

17)Find doctors who work in a specific hospital:

SELECT doctor_name, specialty

FROM doctor_details

WHERE hospital_name = 'Golden Health Hospital';

18)Retrieve diseases that have more than a certain number of associated symptoms:

SELECT disease_name

FROM symptom_disease_mapping

```
WHERE (SELECT COUNT(sym_id) FROM symptom_retrieve
WHERE FIND_IN_SET(sym_retrieve.sym_name,
REPLACE(symptom_ids, ',', ','))) > 2;
```

19)List users who have not consulted with any doctors:

```
SELECT name, email
FROM users
WHERE id NOT IN (SELECT DISTINCT mapping_id FROM
disease_doctor);
```

20)Find diseases that have a certain symptom and are treated by a specific doctor:

```
SELECT disease_name
FROM symptom_disease_mapping
WHERE FIND_IN_SET(2, REPLACE(symptom_ids, ',', ',')) AND
mapping_id IN (SELECT mapping_id FROM disease_doctor WHERE
doctor_id = 3);
```

21)Procedure to Retrieve Doctor Information:

```
CREATE PROCEDURE get_doctor_info(IN p_doctor_id INT)
BEGIN
   SELECT *
   FROM doctor_details
   WHERE doctor_id = p_doctor_id;
END;
```

22)Procedure to List Diseases for a Symptom:

```
CREATE PROCEDURE get_diseases_for_symptom(IN p_symptom_id
INT)
BEGIN
   SELECT disease_name
   FROM symptom_disease_mapping
   WHERE FIND_IN_SET(p_symptom_id, REPLACE(symptom_ids,
',', ''));
END;
```

23)Procedure to Update User Password:

```
CREATE PROCEDURE update_user_password(IN p_user_id INT, IN
p_new_password VARCHAR(100))
BEGIN
```

```
    UPDATE users
    SET password = p_new_password
    WHERE id = p_user_id;
END;
```

24)Procedure to Generate Prescription Report:

```
CREATE PROCEDURE generate_prescription_report()
BEGIN
    SELECT users.name AS patient_name, doctor_details.doctor_name, prescription.medicine
    FROM users
    JOIN disease_doctor ON users.id = disease_doctor.mapping_id
    JOIN doctor_details ON disease_doctor.doctor_id = doctor_details.doctor_id
    JOIN prescription ON symptom_disease_mapping.disease_name = prescription.disease_name;
END;
```

25)Procedure to Delete User and Associated Data:

```
CREATE PROCEDURE delete_user_and_data(IN p_user_id INT)
BEGIN
    DELETE FROM disease_doctor WHERE mapping_id = p_user_id;
    DELETE FROM users WHERE id = p_user_id;
END;
```

25)

```
CREATE TRIGGER check_prescription_validity
BEFORE INSERT ON disease_doctor
FOR EACH ROW
BEGIN
    IF NEW.doctor_id NOT IN (SELECT doctor_id FROM prescription) THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid prescription: Medicine not found';
    END IF;
END;
```

26)

```
CREATE TRIGGER user_login_audit
AFTER INSERT ON users
FOR EACH ROW
BEGIN
```

```
    INSERT INTO user_login_audit (user_id, login_time)
    VALUES (NEW.id, CURRENT_TIMESTAMP);
END;
```

27)

```
CREATE TRIGGER update_disease_count
AFTER INSERT ON symptom_disease_mapping
FOR EACH ROW
BEGIN
    UPDATE disease_count
    SET count = count + 1
    WHERE disease_name = NEW.disease_name;
END;
```