HealOfy

Self Help Application

CS 837 - Healthcare Application Development

International Institute of Information Technology, Bangalore

under guidance of

Prof. T.K.Srikanth

T.A. Aman Kumar Verma

Team Members:
MT2023037: Aparajita Kumari
MT2023096: Nandini Yadav
MT2023052: Nikita Mishra
MT2023179: Shreya Chavan

Video Link:
https://drive.google.com/file/d/1AiOeu1R2hp8N0c1VMbn0zaijpgNoeXXS/view

Webapp video demo link;
https://drive.google.com/file/d/1zWxfOVCEyuI4eyqnvcsDGlxK3anpm5Kr/vie

Contents

Abstract / Executive Summary

‘Healofy’: Your Comprehensive Digital Healthcare Companion .It is a comprehensive online mental health platform designed to promote mental well-being, facilitate education, and ensure timely access to professional help. It incorporates a Q&A forum, curated resources, self-assessment tools, and direct connections with mental health professionals. Emphasizing accessibility and multilingual support, Healofy integrates automated screening for urgent situations, enhancing user safety and engagement.

Introduction

Healofy is a cutting-edge digital health platform designed to empower individuals with comprehensive healthcare solutions. Featuring a robust Q&A forum, extensive resource library, self-assessment tools, and direct access to professional expertise, Healofy is your one-stop-shop for all your healthcare needs.

The Q&A forum allows users to post questions and receive responses from a team of trained responders, ensuring accurate and reliable information. With the ability to flag inappropriate content, the forum maintains a safe and constructive environment for all users.

Healofy's resource library offers a vast collection of educational videos, articles, and materials to enhance your understanding of various health topics. Expertly curated, these resources provide valuable insights and guidance to support your well-being.

Delve into Healofy's self-assessment and self-help tools, such as the GAD7 and PHQ9 assessments, to gain deeper insights into your mental and emotional health. These intuitive tools empower you to take an active role in managing your own healthcare.

Healofy's commitment to accessibility is evident through its multilingual capabilities, catering to diverse user needs. Moreover, the platform's automated screening of posts for emergency conditions or harmful content reinforces its dedication to ensuring a safe and secure environment for all..

Key Features of this applications includes:

- Secured Data Access and Storage.
- Data Access on Demand.
- Privacy of patient data maintained by following authorization access levels.
- Efficient Architecture for consent management.
- Administering patient’s mental health through robust assessment tools and providing tools to help them feel better..
- Log Management for each access and special attention was taken to Serious Scenarios.

Project Scope & Requirements

Theme: Self Help App Monitored by Therapist

Objective: The objective of this platform appears to be to create an online community and resource hub focused on mental health support and education.

Background:
1. Increasing Mental Health Awareness: Recognizing the growing importance of mental health awareness and support due to rising rates of mental health disorders and societal recognition of their impact.
2. Addressing Stigma: Acknowledging the persistent stigma associated with mental health issues, prompting the need for a safe and supportive online space where individuals can seek information and support anonymously.
3. Expanding Access to Resources: Understanding the limited access to mental health services in certain regions or communities, necessitating an accessible platform that transcends geographical barriers.
4. Harnessing Technology for Good: Leveraging the power of technology to democratize mental health resources, making information, tools, and professional support readily available to a wide audience.
5. Empowering Individuals: Recognizing the importance of empowering individuals to take charge of their mental well-being through self-assessment tools, educational materials, and connection with mental health professionals.
6. Complementing Traditional Services: Complementing traditional mental health services by providing an additional avenue for support, especially for those who may be hesitant to seek in-person assistance.
7. Crisis Intervention: Responding to the need for immediate crisis intervention and emergency support, ensuring that individuals in distress have access to timely assistance.

Requirements:
1. User-Friendly Interface: Design an intuitive interface for easy navigation and accessibility.
2. Quality Content Management: Curate and maintain accurate, evidence-based mental health resources.
3. Community Engagement Tools: Enable user registration, discussion forums, and effective content moderation.
4. Expert Interaction: Implement live chat for users to connect with mental health professionals.
5. Self-Assessment Features: Include validated assessment tools with result interpretation.
6. Data Security and Privacy Compliance: Ensure adherence to data protection regulations and provide anonymity options.

.

Project Methodology

1. **Requirements gathering:** The design of the mental health support platform focuses on creating an intuitive user interface for easy navigation. It includes a visually engaging forum for user interaction and moderation. The platform integrates self-assessment tools seamlessly and offers chat-based professional support and emergency features. Accessibility, multi-lingual support, and automated content screening enhance safety and inclusivity.

2. **Design:** Designing the mental health support platform involves creating a user-friendly interface with intuitive navigation. The platform will feature a visually appealing forum for user engagement and moderation. Seamless integration of self-assessment tools is a priority, alongside chat-based professional support and emergency features.

Front-end application development:- Using ReactJs

A. ReactJS is a popular JavaScript library for building user interfaces. It is used to create interactive and dynamic web applications, and is widely adopted by web developers due to its performance, flexibility, and large community support. In order to develop a frontend using ReactJS, we followed these steps:

B. Setup development environment: To get started with ReactJS development, First we, set up our development environment. This typically involves installing Node.js, which is a JavaScript runtime environment, and a package manager like npm or yarn. Then we use these tools to create a new ReactJS project.

C. Create user interface: After our development environment is set up, we started creating our user interface using ReactJS components. ReactJS components are reusable building blocks that encapsulate functionality and can be combined to create complex user interfaces. We used existing ReactJS components from third-party libraries.

D. Manage state: ReactJS uses a concept called "state" to manage data in our application. State refers to any data that changes over time, such as user input or server responses. We used ReactJS hooks to manage state in our application.

E. Handle user interactions: ReactJS provides a declarative way to handle user interactions, such as button clicks or form submissions. We used event handlers to respond to user interactions and update the state of our application.

F. Connect to a backend: To create a fully-functional web application, We connect our frontend to a backend server. ReactJS can communicate with a backend server through RESTful APIs, which allow us to send and receive data between our frontend and backend.

.

Back-end development:- Using Java Spring Boot,Spring JPA

a. Java Spring Boot is a popular framework for building web applications and microservices in Java. It provides a lot of features that make it easy to develop, deploy, and maintain web applications, including support for building RESTful APIs, integration with popular databases and message queues, and a robust testing framework.

b. JPA (Java Persistence API) is a standard API for object-relational mapping in Java. It allows developers to map Java objects to relational database tables, and provides a high-level interface for querying and manipulating data. JPA is often used in conjunction with Spring Boot to build database-backed web applications.

c. To develop backend code using Java Spring Boot and JPA, we started by creating a new Spring Boot project using a tool like Maven. We then added dependencies for Spring Data JPA,Spring web, MySql, Spring security which provides a set of high-level abstractions for working with databases in Spring Boot applications.

d. Then we defined our database schema using JPA annotations on Java classes. These annotations tell JPA how to map our Java objects to database tables and columns, and provide a way to specify relationships between entities.

e. Next, we defined our JPA repositories, which were Spring Data interfaces that provide a set of standard CRUD (create, read, update, delete) operations for working with our entities. Spring Data JPA provides a lot of built-in functionality for querying and manipulating data, so we can usually write very concise and expressive code to work with our database.

f. Finally, we defined our RESTful endpoints using Spring MVC, which provides a way to map HTTP requests to controller methods.Then it will call a service package where we write our backend logic. We use our JPA repositories to read and write data from our database, and return JSON responses to our clients.

DataBase:- MySql..

a. For this system we required one database
   a) Self-Help Database
b. In the Self-help database, we store registered doctor, registered patient, appointment details, posts, comments, chats between patient and doctor, questionnaire responses and Q&A responses.

Development: We developed the software components of the system using an agile methodology. We developed the user interface and the back-end logic to manage consents, including consent requests, approvals, and revocations.

Testing: We conducted functional testing, integration testing, and user acceptance testing to ensure that the system is working correctly and meets the requirements.

Important features of different Roles

Patient Entity:

- Ability to view posts in the post forum.
- Capability to create posts in the post forum.
- Option to comment on posts in the post forum.
- Facility to ask questions in the Q&A forum.
- Ability to book appointments with preferred doctors.
- Capability to chat with doctors.
- Option to flag inappropriate posts and comments.
- Capability to edit their profile.
- Opportunity to answer quizzes to determine severity of illness.

Doctor Entity:

- Creation, viewing, and commenting on posts.
- Ability to flag posts or comments.
- Acceptance of appointment requests from patients.
- Checking patient profiles.
- Replying to questions asked by users.
- Ability to chat with patients after accepting appointment requests.

Admin Entity:

- Supervising the usage of the application.
- Assigning moderators to manage specific areas.
- Reviewing patient profiles.
- Accessing doctor profiles.
- Disabling doctor-patient interactions if necessary.
- Monitoring posts and Q&A forums like any regular user.
- Accessing chats between patients and doctors.
- Ability to flag posts, questions, and answers in the forums.

Moderator Entity:

- Viewing posts and Q&A forums like any regular user.
- Authority to disable posts, comments, questions, and answers flagged as inappropriate.

t

API Listing/ Description

Title: Self help App-HealOfy
Description: Spring boot API-documentation :

1. Update User Endpoint (/api/user/update/{id}):
   a. HTTP Method: PUT
   b. Operation ID: updateUserPatient
   c. Parameters: id (path parameter, integer): The ID of the user to be updated.
   d. Request Body: Expects JSON data (application/json) conforming to the UserDto schema.
   e. Responses: HTTP 200 (OK) response with a string content type (*/*).

2. Get Replies Endpoint (/api/questionPost/{postId}/replies/{id}):
   a. HTTP Method: GET
   b. Parameters:
      i) postId (path parameter, string/integer): The ID of the question post.
      ii) id (path parameter, string/integer): The ID of the reply.
   c. Responses: This endpoint retrieves replies related to a specific post and reply ID.

3. Get Reply by ID (/api/reply/{postId}/{id})
   a) HTTP Method: GET
   b)Tags: reply-controller
   c) Operation ID: getReplyById
   d) Parameters:
      i) postId (path parameter, integer): ID of the post containing the reply.
      ii) id (path parameter, integer): ID of the reply to retrieve.
   e) Responses: HTTP 200 (OK) response with a JSON response body conforming to ReplyDto schema.

4.Update Reply by ID (/api/reply/{postId}/{id})
   a) HTTP Method: PUT
   b) Tags: reply-controller
   c) Operation ID: updateCommentById
   d) Parameters:
      i) postId (path parameter, integer): ID of the post containing the reply.
      ii) id (path parameter, integer): ID of the reply to update.
   e) Request Body: Expects JSON data (application/json) conforming to the ReplyDto schema.

f) Responses: HTTP 200 (OK) response with a JSON response body conforming to ReplyDto schema.

5. Get Post by ID (/api/questionPost/{id})
    a) HTTP Method: GET
    b) Tags: question-post-controller
    c) Operation ID: getPostById
    d) Parameters: id (path parameter, integer): ID of the post to retrieve.
    e) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionPostDto schema.

6. Update Post by ID (/api/questionPost/{id})
    a) HTTP Method: PUT
    b) Tags: question-post-controller
    c) Operation ID: updatePost
    d) Parameters: id (path parameter, integer): ID of the post to update.
    e) Request Body: Expects JSON data (application/json) conforming to the QuestionPostDto schema.
    f) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionPostDto schema.

7. Get Question by ID (/api/question/{id})
    a) HTTP Method: GET
    b) Tags: questionnaire-ques-controller
    c) Operation ID: getQuestionById
    d) Parameters: id (path parameter, integer): ID of the question to retrieve.
    e) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionnaireQuesDto schema

8. Update Question by ID (/api/question/{id})
    a) HTTP Method: PUT
    b) Tags: questionnaire-ques-controller
    c) Operation ID: updatePost_1
    d) Parameters: id (path parameter, integer): ID of the question to update.
    e) Request Body: Expects JSON data (application/json) conforming to the QuestionnaireQuesDto schema.
    f) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionnaireQuesDto schema

9. Get Comment by ID (/api/posts/{postId}/comments/{id})
    a) HTTP Method: GET
    b) Tags: comment-controller
    c) Operation ID: getCommentById
    d) Parameters:
        i) postId (path parameter, integer): ID of the post containing the comment.

ii) id (path parameter, integer): ID of the comment to retrieve.

e) Responses: HTTP 200 (OK) response with a JSON response body conforming to CommentDto scheme

10. Update Comment by ID (/api/posts/{postId}/comments/{id})
    a) HTTP Method: PUT
    b) Tags: comment-controller
    c) Operation ID: updateCommentById_1
    d) Parameters:
       i) postId (path parameter, integer): ID of the post containing the comment.
       ii) id (path parameter, integer): ID of the comment to update.
    e) Request Body:Expects JSON data (application/json) conforming to the CommentDto schema.
    f) Responses: HTTP 200 (OK) response with a JSON response body conforming to CommentDto schema.

11. Delete Comment by ID (/api/posts/{postId}/comments/{id})
    a) HTTP Method: DELETE
    b) Tags: comment-controller
    c) Operation ID: deleteComment
    d) Parameters:
       i) postId (path parameter, integer): ID of the post containing the comment.
       ii) id (path parameter, integer): ID of the comment to delete.
    e) Responses: HTTP 200 (OK) response.

12. Get Post by ID (/api/posts/post/{id})
    a) HTTP Method: GET
    b) Tags: post-controller
    c) Operation ID: getPostById_1
    d) Parameters: id (path parameter, integer): ID of the post to retrieve.
    e) Responses: HTTP 200 (OK) response with a JSON response body conforming to PostDto schema

13. Update Post by ID (/api/posts/post/{id})
    a) HTTP Method: PUT
    b) Tags: post-controller
    c) Operation ID: updatePost_2
    d) Parameters: id (path parameter, integer): ID of the post to update.
    e) Request Body: Expects JSON data (application/json) conforming to the PostDto schema.
    f) Responses: HTTP 200 (OK) response with a JSON response body conforming to PostDto schema.

14. Get Patient by ID (/api/patient/{id})

a) HTTP Method: GET
b) Tags: patient-controller
c) Operation ID: getPatientById
d) Parameters: id (path parameter, integer): ID of the patient to retrieve.
e) Responses: HTTP 200 (OK) response with a JSON response body conforming to PatientDto schema.

15. Update Patient by ID (/api/patient/{id})

a) HTTP Method: PUT
b) Tags: patient-controller
c) Operation ID: updatePatient
d) Parameters: id (path parameter, integer): ID of the patient to update.
e) Request Body: Expects JSON data (application/json) conforming to the PatientDto schema.
f) Responses: HTTP 200 (OK) response with a JSON response body conforming to PatientDto schema.

16. Get Doctor by ID (/api/doctor/{id})

a) HTTP Method: GET
b) Tags: doctor-controller
c) Operation ID: getDoctorById
d) Parameters: id (path parameter, integer): ID of the doctor to retrieve.
e) Responses: HTTP 200 (OK) response with a JSON response body conforming to DoctorRegisterDto schema.

17. Update Doctor by ID (/api/doctor/{id})

a) HTTP Method: PUT
b) Tags: doctor-controller
c) Operation ID: updateDoctor
d) Parameters: id (path parameter, integer): ID of the doctor to update.
e) Request Body: Expects JSON data (application/json) conforming to the DoctorByAdminDto schema.
f) Responses: HTTP 200 (OK) response with a JSON response body conforming to DoctorByAdminDto schema.

18. Update User as Doctor (/api/doctor/update/{id})

a) HTTP Method: PUT
b) Tags: user-controller
c) Operation ID: updateUserDoctor
d) Parameters: id (path parameter, integer): ID of the user (doctor) to update.
e) Request Body: Expects JSON data (application/json) conforming to the UserDto schema.
f) Responses: HTTP 200 (OK) response with a string response.

19. Update Appointment (/api/appointment/{appointList}/{id})

a) HTTP Method: PUT
b) Tags: appointment-controller
c) Operation ID: updateAppointment
d) Parameters:
   i) appointList (path parameter, integer): Appointment list ID.
   ii) id (path parameter, integer): ID of the appointment to update.
e) Request Body: Expects JSON data (application/json) conforming to the AppointmentDto schema.
f) Responses: HTTP 200 (OK) response with a JSON response body conforming to AppointmentDto schema.

20. Create Questionnaire Response (/api/{qid}/createResponse)

a) HTTP Method: POST
b) Tags: questionnaire-res-controller
c) Operation ID: createResponse
d) Parameters: qid (path parameter, integer): ID of the questionnaire.
e) Request Body: Expects JSON data (application/json) conforming to the QuestionnaireResponseDto schema.
f) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionnaireResponseDto schema.

21. Create Chat (/api/{appointmentId}/{userId}/createChat)

a) HTTP Method: POST
b) Tags: chat-controller
c) Operation ID: createChat
d) Parameters:
   i) appointmentId (path parameter, integer): ID of the appointment.
   ii) userId (path parameter, integer): ID of the user (participant).
e) Request Body: Expects JSON data (application/json) conforming to the ChatDto schema.
f) Responses: HTTP 200 (OK) response.

22. Create Reply for Question Post (/api/questionPost/{postId}/createReply)

1) HTTP Method: POST
2) Tags: reply-controller
3) Operation ID: createComment
4) Parameters: postId (path parameter, integer): ID of the question post to which the reply will be created.
5) Request Body: Expects JSON data (application/json) conforming to the ReplyDto schema.
6) Responses: HTTP 200 (OK) response with a JSON response body conforming to ReplyDto schema.

23. Create Question Post (/api/questionPost/createQuestion)

1) HTTP Method: POST
2) Tags: question-post-controller
3) Operation ID: createQuestionPost

4) Request Body: Expects JSON data (application/json) conforming to the QuestionPostDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionPostDto schema.

24. Create Comment for Post (/api/posts/{postId}/createComment)
1) HTTP Method: POST
2) Tags: comment-controller
3) Operation ID: createComment_1
4) Parameters: postId (path parameter, integer): ID of the post to which the comment will be created.
5) Request Body: Expects JSON data (application/json) conforming to the CommentDto schema.
6) Responses: HTTP 200 (OK) response with a JSON response body conforming to CommentDto schema.

25. Create Post (/api/posts/createPost)
1) HTTP Method: POST
2) Tags: post-controller
3) Operation ID: createpost
4) Request Body: Expects JSON data (application/json) conforming to the PostDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to PostDto schema.

26. Create Question (/api/createQuestion)
1) HTTP Method: POST
2) Tags: questionnaire-ques-controller
3) Operation ID: createQuestion
4) Request Body: Expects JSON data (application/json) conforming to the QuestionnaireQuesDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to QuestionnaireQuesDto schema.

27. Create Patient (/api/createPatient)
1) HTTP Method: POST
2) Tags: patient-controller
3) Operation ID: createPatient
4) Request Body: Expects JSON data (application/json) conforming to the PatientDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to PatientDto schema.

28. Create Doctor (/api/createDoctor)
1) HTTP Method: POST
2) Tags: doctor-controller
3) Operation ID: createDoctor

4) Request Body: Expects JSON data (application/json) conforming to the DoctorRegisterDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to DoctorRegisterDto schema.

29. Create Appointment (/api/createAppointment/{appointList})

1) HTTP Method: POST
2) Tags: appointment-controller
3) Operation ID: createappointment
4) Parameters: appointList (path parameter, integer): ID of the appointment list.
5) Request Body: Expects JSON data (application/json) conforming to the AppointmentDto schema.
6) Responses: HTTP 200 (OK) response with a JSON response body conforming to AppointmentDto schema.

30. User Authentication - Login (/api/auth/login) and Signin (/api/auth/signin)

1) HTTP Method: POST
2) Tags: auth-controller
3) Operation IDs: login, login_1
4) Request Body: Expects JSON data (application/json) conforming to the LoginDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to JWTAuthResponse schema.

31. Doctor Authentication - Login (/api/auth/doctor/login) and Signin (/api/auth/doctor/signin)

1) HTTP Method: POST
2) Tags: auth-controller
3) Operation IDs: loginDoctor, loginDoctor_1
4) Request Body: Expects JSON data (application/json) conforming to the DoctorLoginDto schema.
5) Responses: HTTP 200 (OK) response with a JSON response body conforming to JWTAuthResponse schema.

32. User and Doctor Registration (/api/auth/register, /api/auth/signup, /api/auth/doctor/register, /api/auth/doctor/signup)

1) HTTP Method: POST
2) Tags: auth-controller
3) Operation IDs: register, register_1, doctorRegister, doctorRegister_1
4) Request Body: Expects JSON data (application/json) conforming to the respective registration DTO schemas.
5) Responses: HTTP 200 (OK) response with a string response.

33. Get Questionnaire Response by ID (/api/{qId}/responses/{id})

1) HTTP Method: GET
2) Tags: questionnaire-res-controller
3) Operation ID: getResponseById
4) Parameters:
    a) qId (path parameter, integer): ID of the questionnaire.

        b) id (path parameter, integer): ID of the response.
    5) Responses: HTTP 200 (OK) response.

34. /api/{appointmentId}/chats/{id}
    1) GET
        a) Description: Retrieves a chat by its unique id within a specific appointmentId.
        b) Parameters:
            i) appointmentId (path, integer): The ID of the appointment.
            ii) id (path, integer): The ID of the chat.
        c) Responses: 200 OK: Successful response with the requested chat data.

35. /api/{appointmentId}/allChats
    1) GET
        a) Description: Retrieves all chats associated with a specific appointmentId.
        b) Parameters: appointmentId (path, integer): The ID of the appointment.
        c) Responses: 200 OK: Successful response with an array of chat data.

36. /api/user/{username}
    1) GET
        a) Description: Retrieves user details by their username.
        b) Parameters: username (path, string): The username of the user.
        c) Responses: 200 OK: Successful response with user data.

37. /api/user/patient/{email}
    1) GET
        a) Description: Retrieves patient details by their email.
        b) Parameters: email (path, string): The email address of the patient.
        c) Responses: 200 OK: Successful response with patient data.

38. /api/user/doctor/{username}
    1) GET
        a) Description: Retrieves doctor details by their username.
        b) Parameters: username (path, string): The username of the doctor.
        c) Responses: 200 OK: Successful response with doctor data.

39. /api/role/{id}
    1) GET
        a) Description: Retrieves role details by id.
        b) Parameters: id (path, integer): The ID of the role.
        c) Responses: 200 OK: Successful response with role data.

40. /api/questionPost/{postId}/allReplies
    1) GET
        a) Description: Retrieves all replies to a specific question post (postId).
        b) Parameters: postId (path, integer): The ID of the question post.
        c) Responses: 200 OK: Successful response with an array of reply data.

41. /api/questionPost/allQuestions

1) GET
    a) Description: Retrieves all question posts.
    b) Parameters:
        i) pageNo (query, integer, optional): Page number for pagination (default: 0).
        ii) pageSize (query, integer, optional): Number of items per page (default: 10).
        iii) sortBy (query, string, optional): Field to sort by (default: id).
        iv) sortDir (query, string, optional): Sorting direction (default: desc).
    c) Responses: 200 OK: Successful response with an array of question post data.

42. /api/posts/{qId}/allResponses
1) GET
    a) Description: Retrieves all responses to a specific question (qId).
    b) Parameters: None
    c) Responses: 200 OK: Successful response with an array of response data.

43. /api/posts/{postId}/allComments
1) GET
    a) Description: Retrieves all comments associated with a specific post (postId).
    b) Parameters: postId (path, integer): The ID of the post.
    c) Responses: 200 OK: Successful response with an array of comment data.

44. /api/posts/allPosts
1) GET
    a) Description: Retrieves all posts.
    b) Parameters:
        i) pageNo (query, integer, optional): Page number for pagination (default: 0).
        ii) pageSize (query, integer, optional): Number of items per page (default: 10).
        iii) sortBy (query, string, optional): Field to sort by (default: id).
        iv) sortDir (query, string, optional): Sorting direction (default: desc).
    c) Responses: 200 OK: Successful response with an array of post data.

45. /api/moderator/allDoctors
1) GET
    a) Description: Retrieves all doctors for moderator/admin purposes.
    b) Parameters: None
    c) Responses: 200 OK: Successful response with an array of doctor data.

46. /api/appointmentId/{id}
1) GET
    a) Description: Retrieves an appointment by its id.
    b) Parameters: id (path, integer): The ID of the appointment.
    c) Responses: 200 OK: Successful response with appointment data.

47. /api/appointment/patient/{patientId}
1) GET
    a) Description: Retrieves all appointments associated with a specific patient (patientId).
    b) Parameters: patientId (path, integer): The ID of the patient.
    c) Responses: 200 OK: Successful response with an array of appointment data.

48. /api/appointment/doctorId/{doctorId}

   1) GET

      a) Description: Retrieves all appointments associated with a specific doctor (doctorId).
      b) Parameters: doctorId (path, integer): The ID of the doctor.
      c) Responses: 200 OK: Successful response with an array of appointment data.

49. /api/appointment/doctor/{id}

   1) GET

      a) Description: Retrieves all appointments associated with a specific doctor (id).
      b) Parameters: id (path, integer): The ID of the doctor.
      c) Responses: 200 OK: Successful response with an array of appointment data.

50. /api/allQuestions

   1) GET

      a) Description: Retrieves all questions.
      b) Parameters: None
      c) Responses: 200 OK: Successful response with an array of question data.

51. /api/questionnaire-ques-controller/getAllQuestions

   1) GET

      a) Description: Retrieves all questionnaire questions.
      b) Responses: 200 OK: Successful response with an array of QuestionnaireQuesDto.

52. /api/allPatients

   2) GET

      a) Description: Retrieves all patients.
      b) Responses: 200 OK: Successful response with an array of PatientDto.

53. /api/allDoctors

   3) GET

      a) Description: Retrieves all doctors from the perspective of a patient.
      b) Responses: 200 OK: Successful response with an array of DoctorPatientDto.

54. /api/allAppointments

   4) GET

      a) Description: Retrieves all appointments.
      b) Responses: 200 OK: Successful response with an array of AppointmentDto.

55. /api/admins/doctor/{username}

   5) GET

      a) Description: Retrieves doctor details by username for administrative purposes.
      b) Parameters: username (path, string): The username of the doctor.
      c) Responses: 200 OK: Successful response with DoctorByAdminDto.

56. /api/admin/doctor/{id}

   6) GET

      a) Description: Retrieves doctor details by id for administrative purposes.
      b) Parameters: id (path, integer): The ID of the doctor.

c) Responses: 200 OK: Successful response with DoctorByAdminDto.

57. /api/user/doctor/{id}

7) DELETE
    a) Description: Deletes a doctor user by id.
    b) Parameters: id (path, integer): The ID of the doctor user to delete.
    c) Responses: 200 OK: Successful response with a confirmation message.

58. /api/appointment/{id}

8) DELETE
    a) Description: Deletes an appointment by id.
    b) Parameters: id (path, integer): The ID of the appointment to delete.
    c) Responses: 200 OK: Successful response with a confirmation message.

Schema Definitions:

QuestionnaireQuesDto

- Properties:
    - question (string): The questionnaire question.
    - weightage (integer): The weightage assigned to the question.
    - responses (array of QuestionnaireResponseDto): Responses associated with the question.
    - id (integer): Unique identifier for the question.

PatientDto

- Properties: (similar to DoctorPatientDto)

DoctorPatientDto

- Properties:
    - firstName (string): Doctor's first name.
    - lastName (string): Doctor's last name.
    - age (integer): Doctor's age.
    - exp (string): Doctor's experience.
    - licenseId (string): Doctor's license ID.
    - username (string): Doctor's username.
    - email (string): Doctor's email address.
    - id (integer): Unique identifier for the doctor.

AppointmentDto

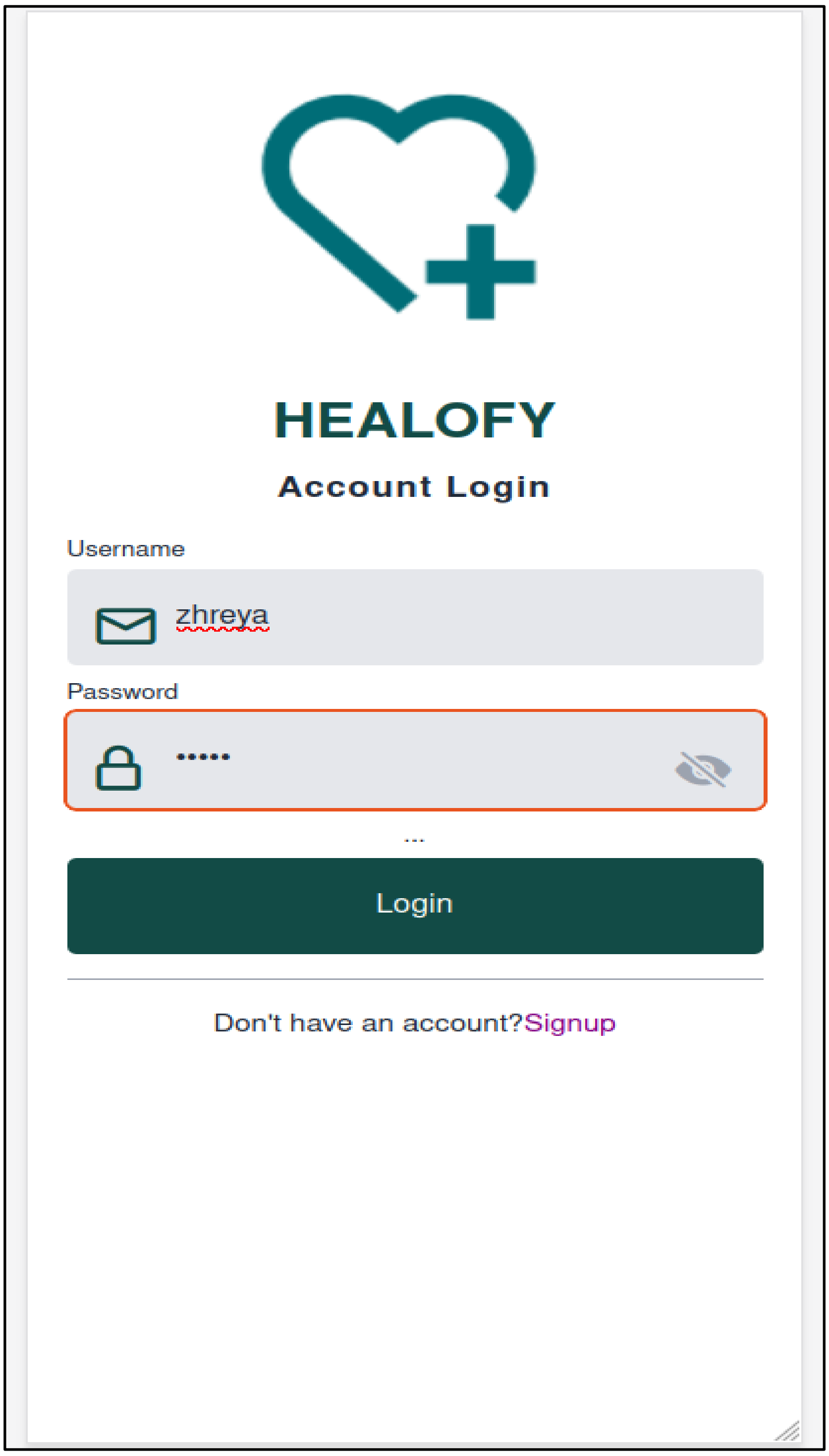
- Properties: (similar to Appointment)

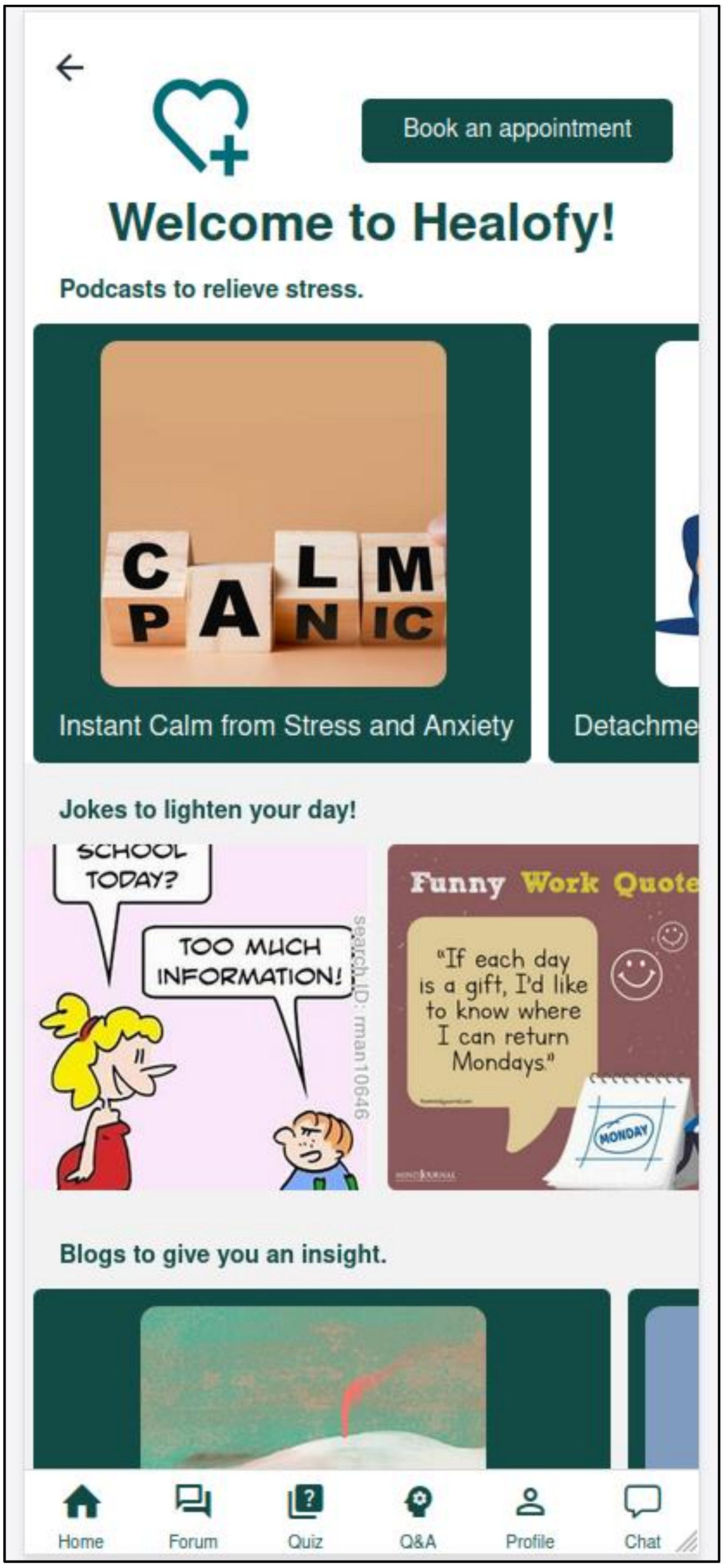
DoctorByAdminDto

- Properties: (similar to DoctorPatientDto with additional admin-related properties.

User Interface & User Experience

We have designed UI/UX using Figma at Design Stage of Sammati Software Development Life Cycle View
As we proceeded further made changes to our design and ended up the following frontend user





Book an appointment

Welcome to Healofy!

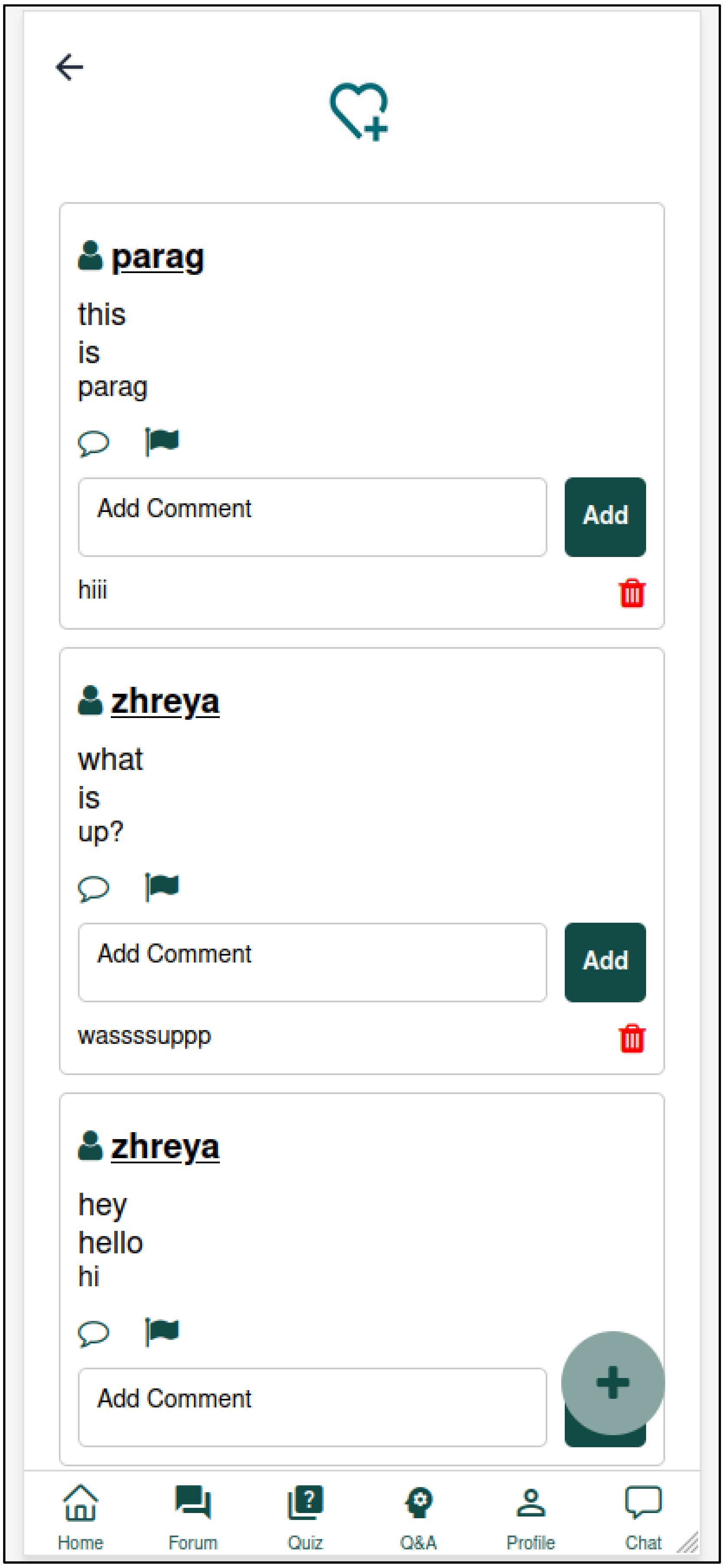
Podcasts to relieve stress.

Instant Calm from Stress and Anxiety

Detachme

Jokes to lighten your day!

Blogs to give you an insight.



parag

this
is
parag

Add Comment

Add

hiii

zhreya

what
is
up?

Add Comment

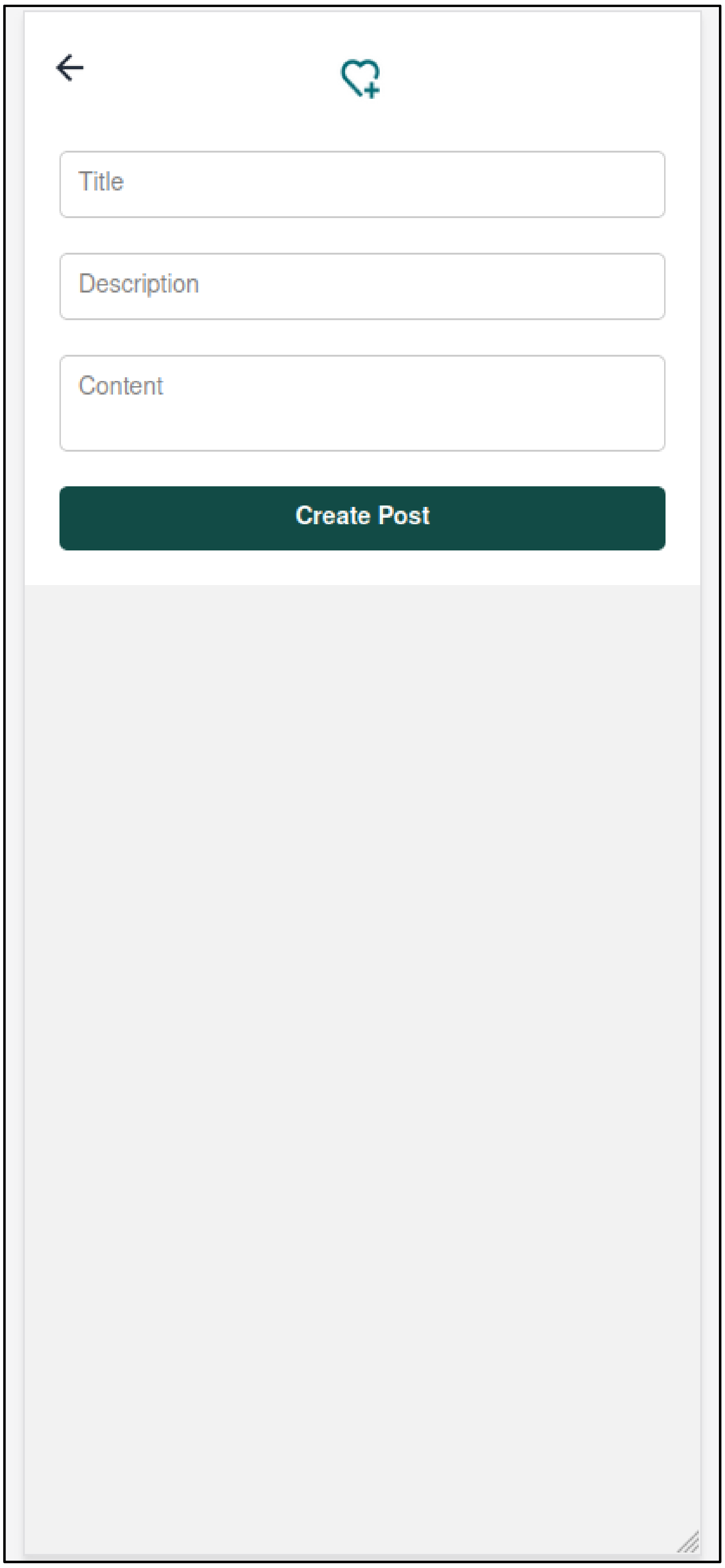
Add

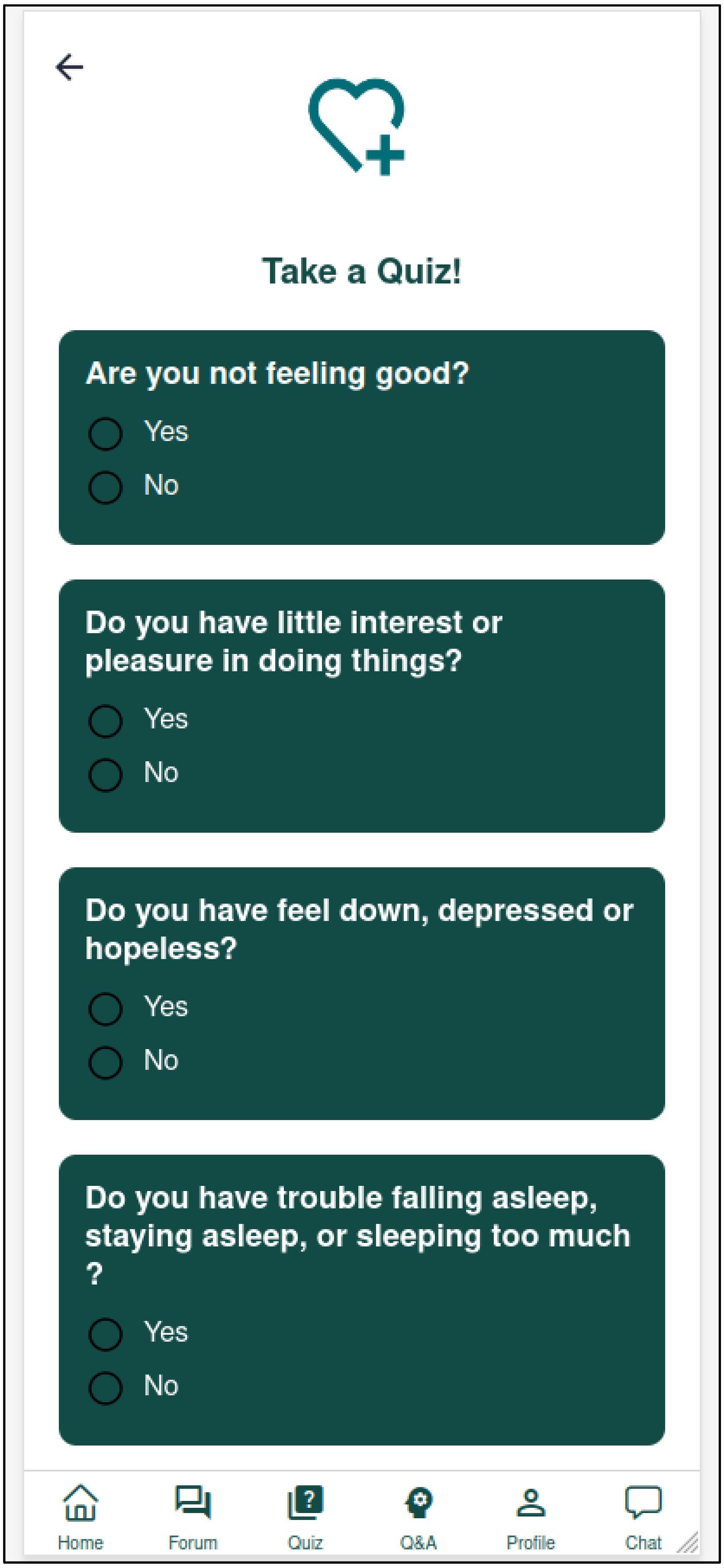
wassssuppp

zhreya

hey
hello
hi

Add Comment


Title
Description
Content
Create Post



Take a Quiz!

Are you not feeling good?

- ○ Yes
- ○ No

Do you have little interest or pleasure in doing things?

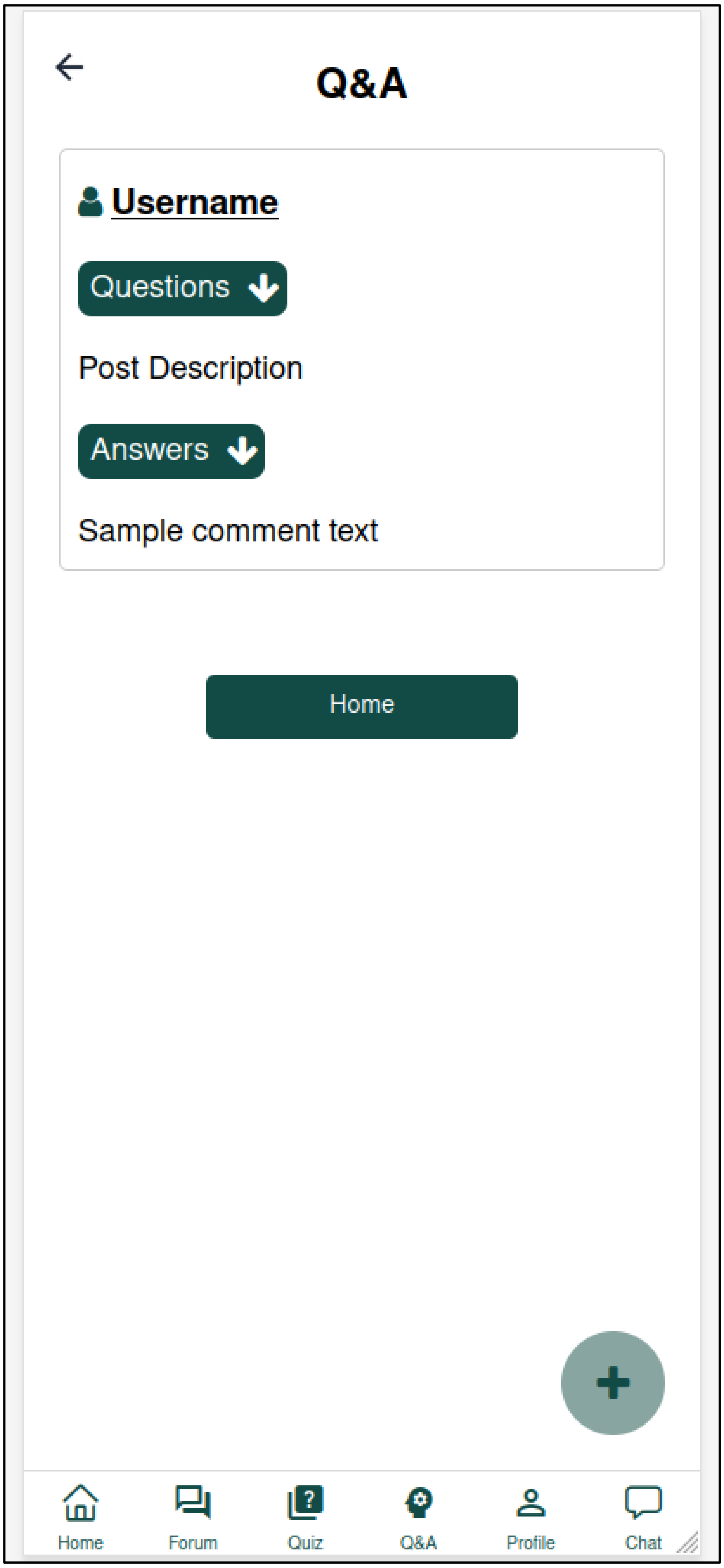
- ○ Yes
- ○ No

Do you have feel down, depressed or hopeless?

- ○ Yes
- ○ No

Do you have trouble falling asleep, staying asleep, or sleeping too much ?

- ○ Yes
- ○ No



Q&A

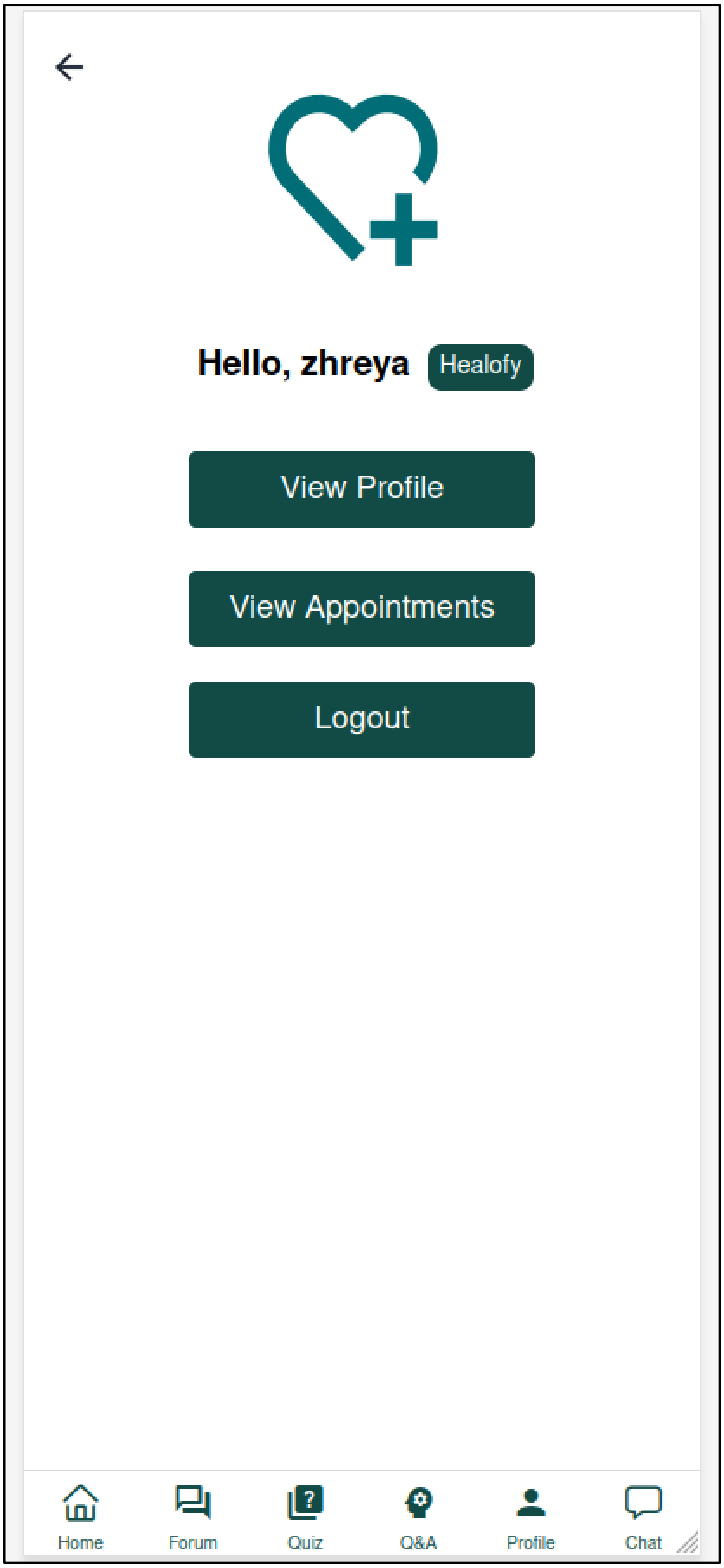
Username

Questions

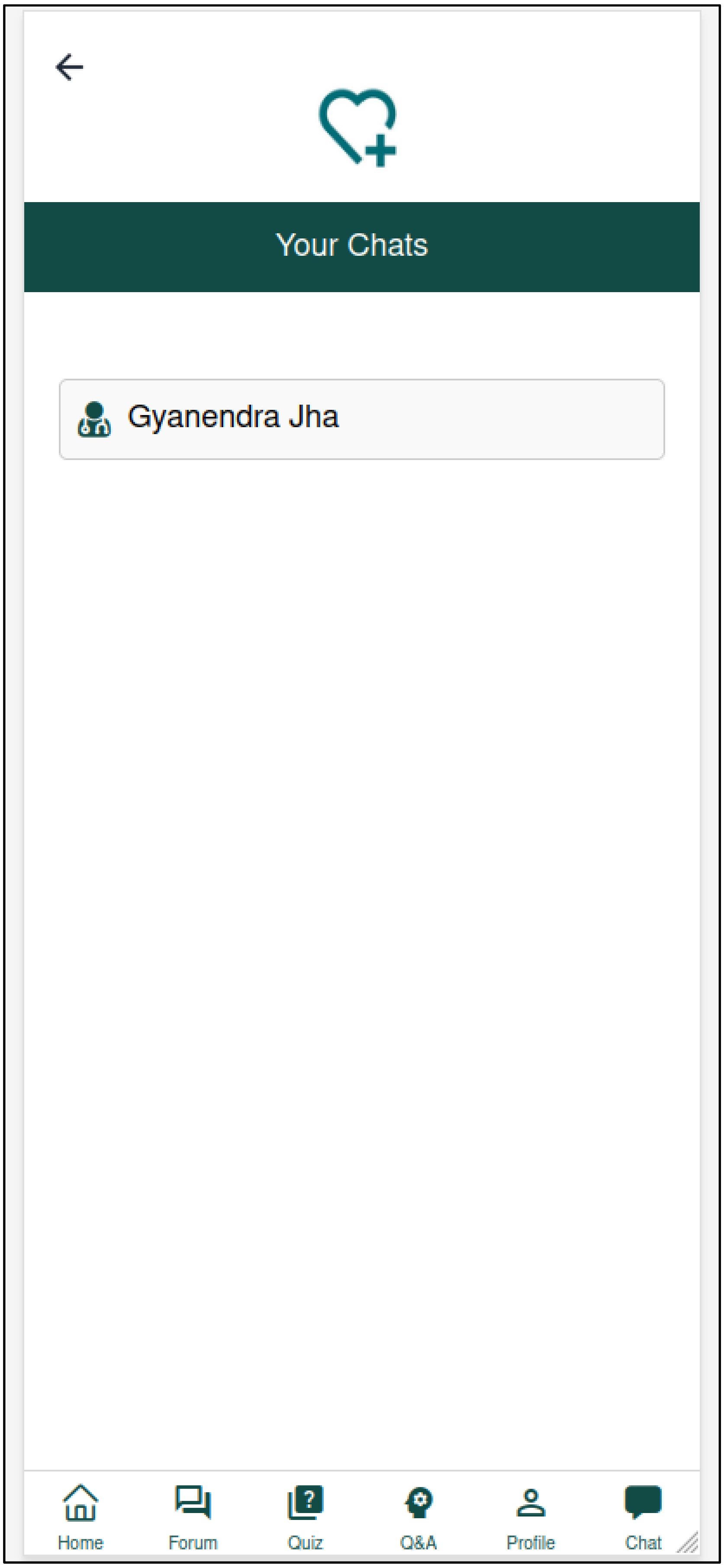
Post Description

Answers

Sample comment text

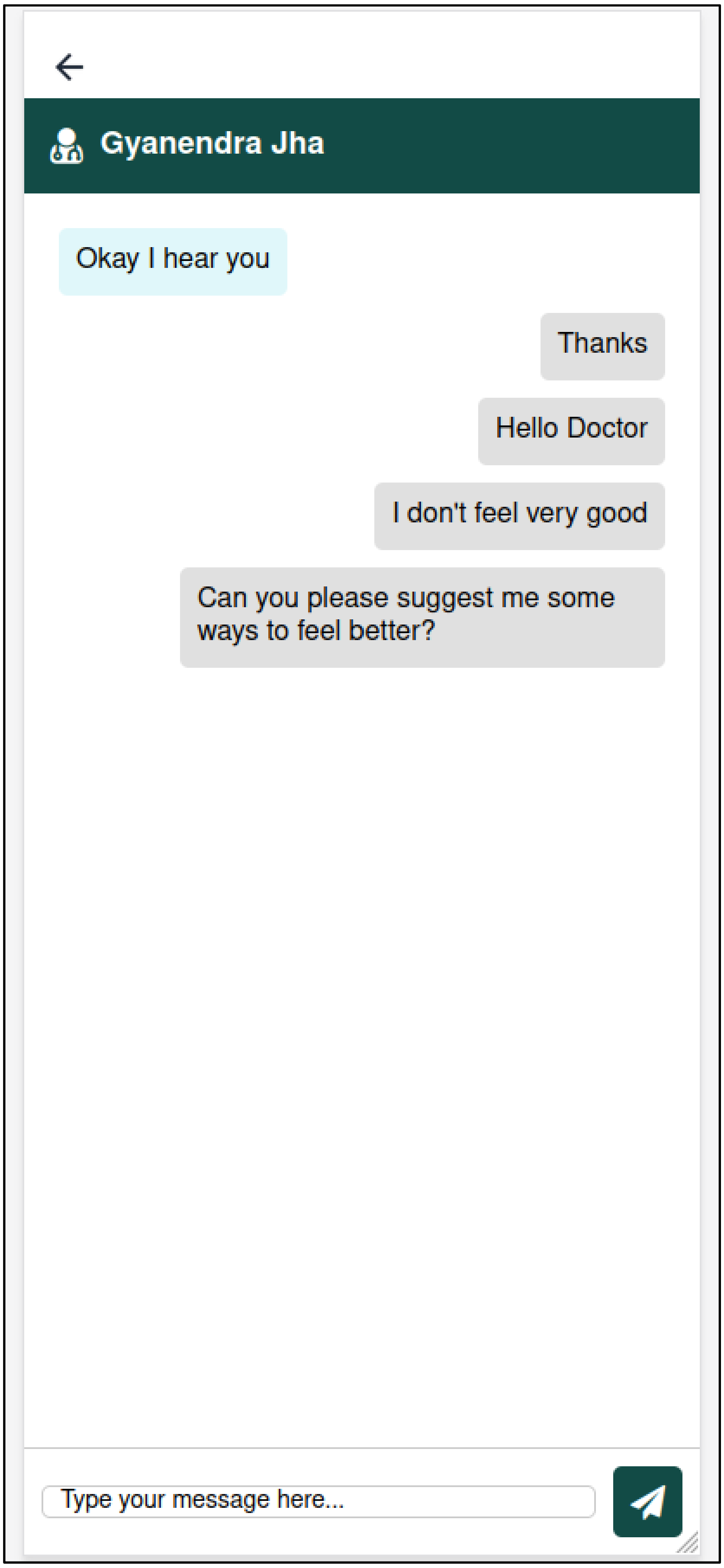


Hello, zhreya
Healofy
View Profile
View Appointments
Logout
Home
Forum
Quiz
Q&A
Profile
Chat



Your Chats

Gyanendra Jha



Gyanendra Jha

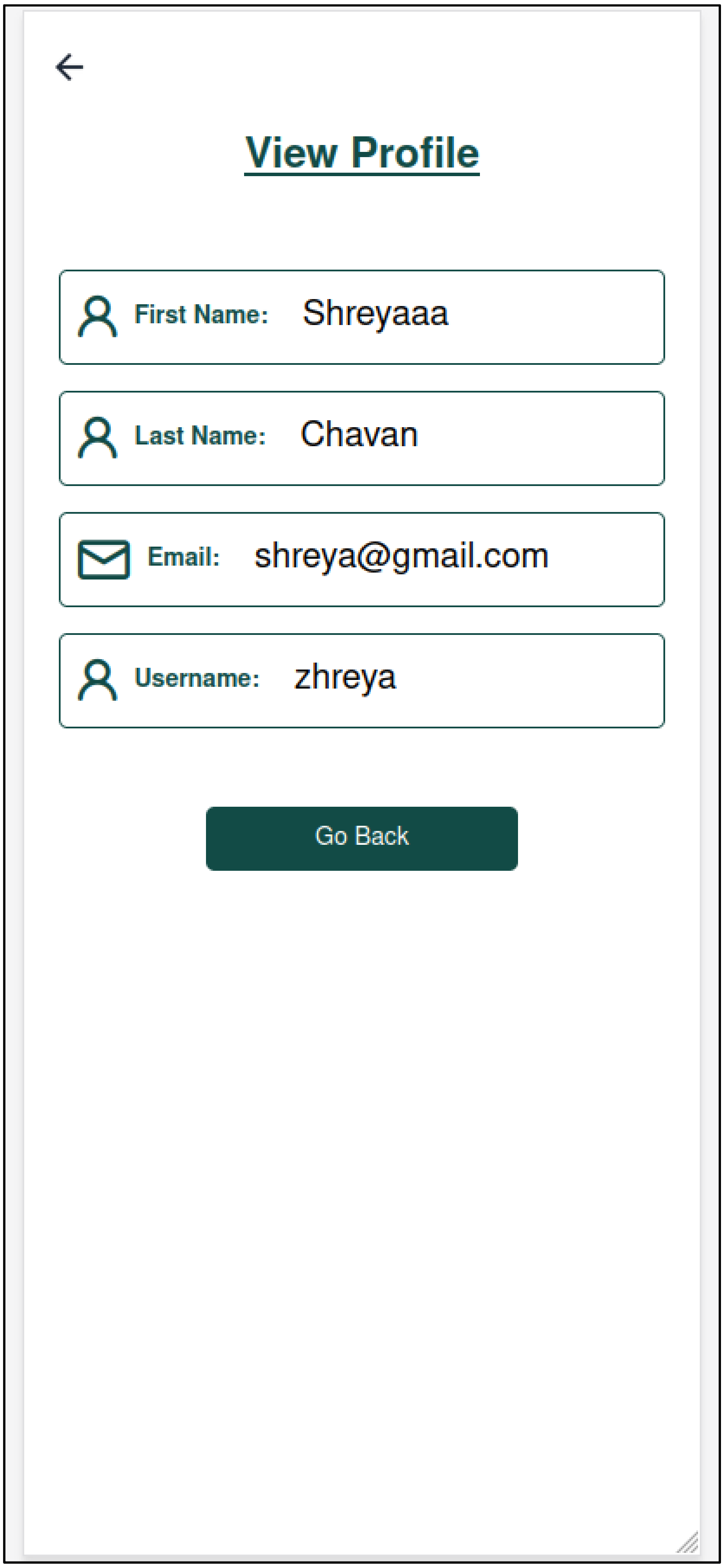
Okay I hear you

Thanks

Hello Doctor

I don't feel very good

Can you please suggest me some ways to feel better?



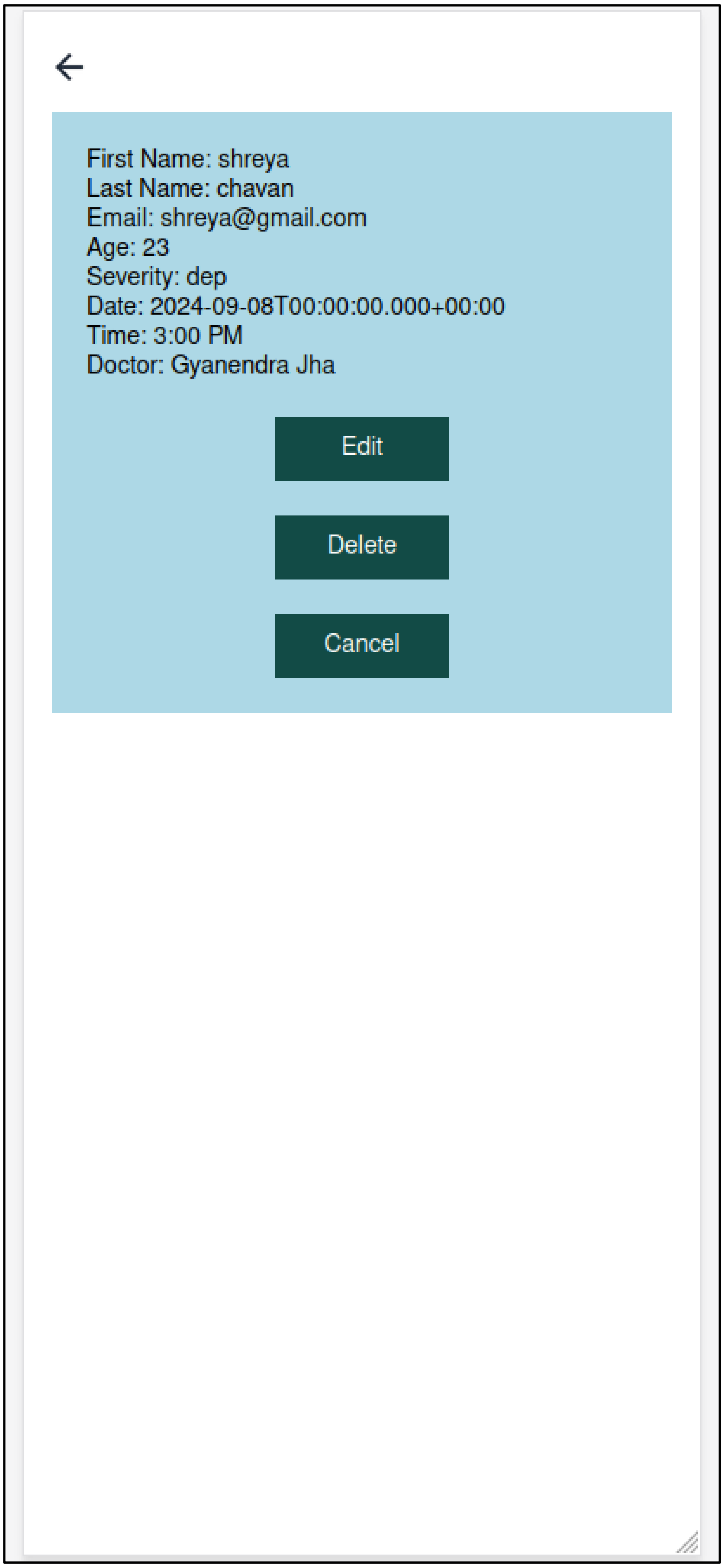
View Profile

First Name: Shreyaaa

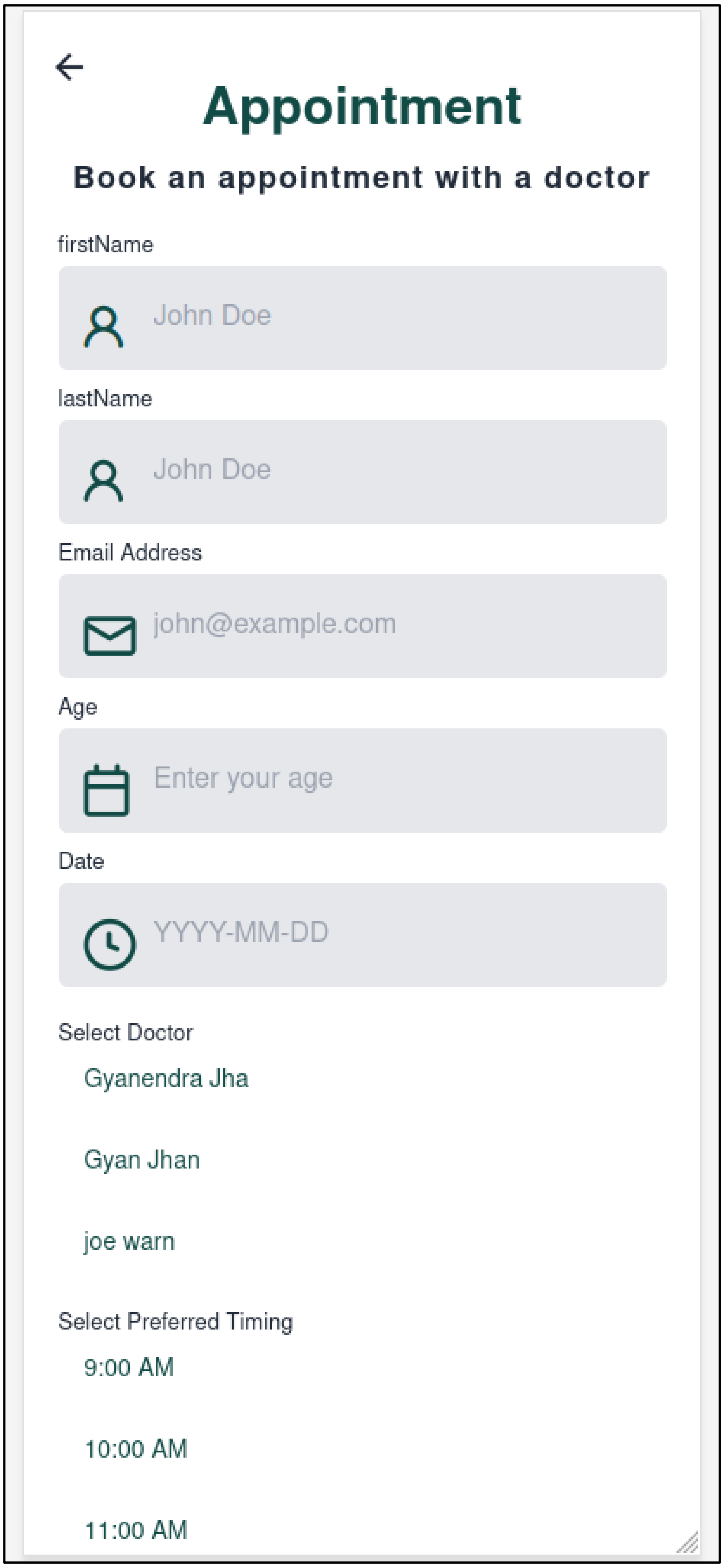
Last Name: Chavan

Email: shreya@gmail.com

Username: zhreya



First Name: shreya
Last Name: chavan
Email: shreya@gmail.com
Age: 23
Severity: dep
Date: 2024-09-08T00:00:00.000+00:00
Time: 3:00 PM
Doctor: Gyanendra Jha



Appointment

Book an appointment with a doctor

firstName

John Doe

lastName

John Doe

Email Address

john@example.com

Age

Enter your age

Date

YYYY-MM-DD

Select Doctor

Gyanendra Jha

Gyan Jhan

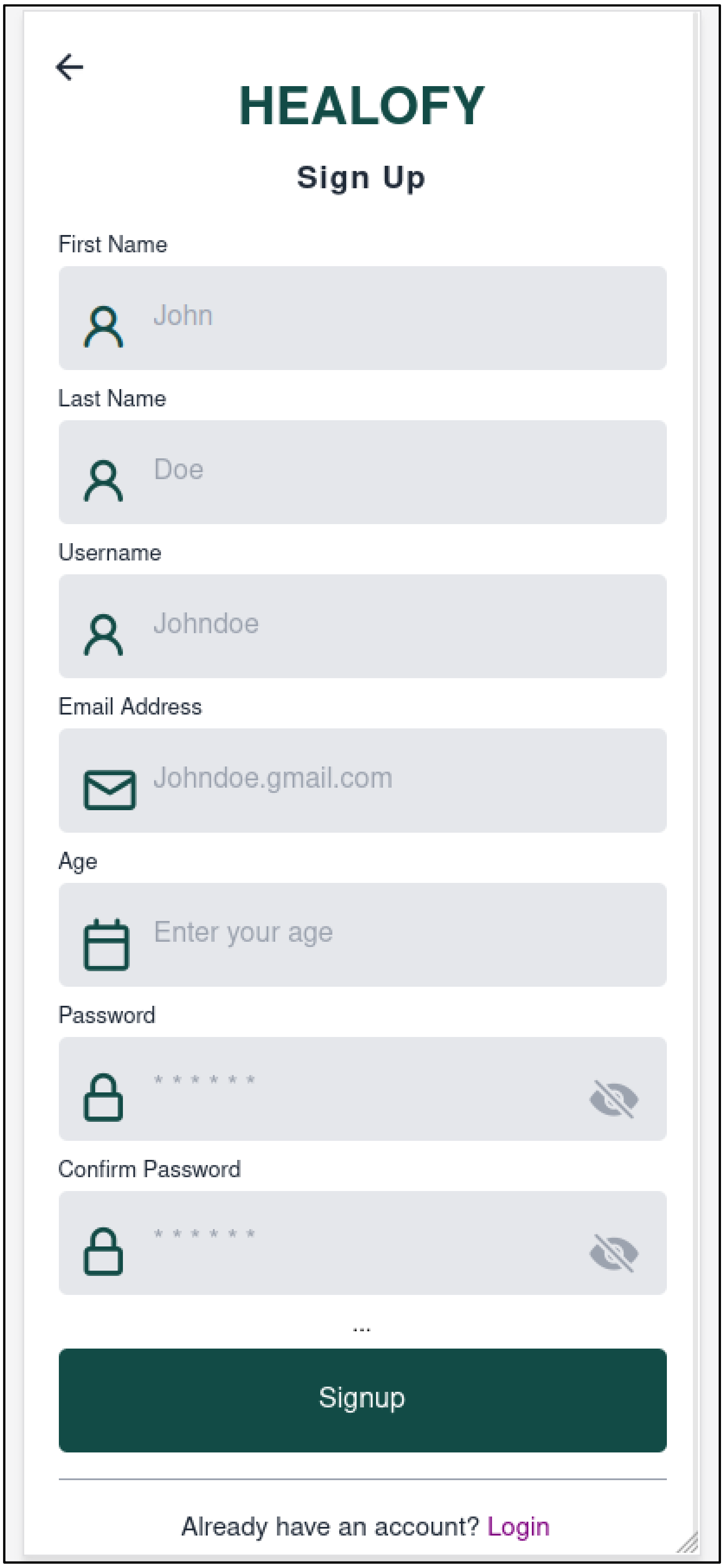
joe warn

Select Preferred Timing

9:00 AM

10:00 AM

11:00 AM



HEALOFY

Sign Up

First Name

John

Last Name

Doe

Username

Johndoe

Email Address

Johndoe.gmail.com

Age

Enter your age

Password

* * * * * *

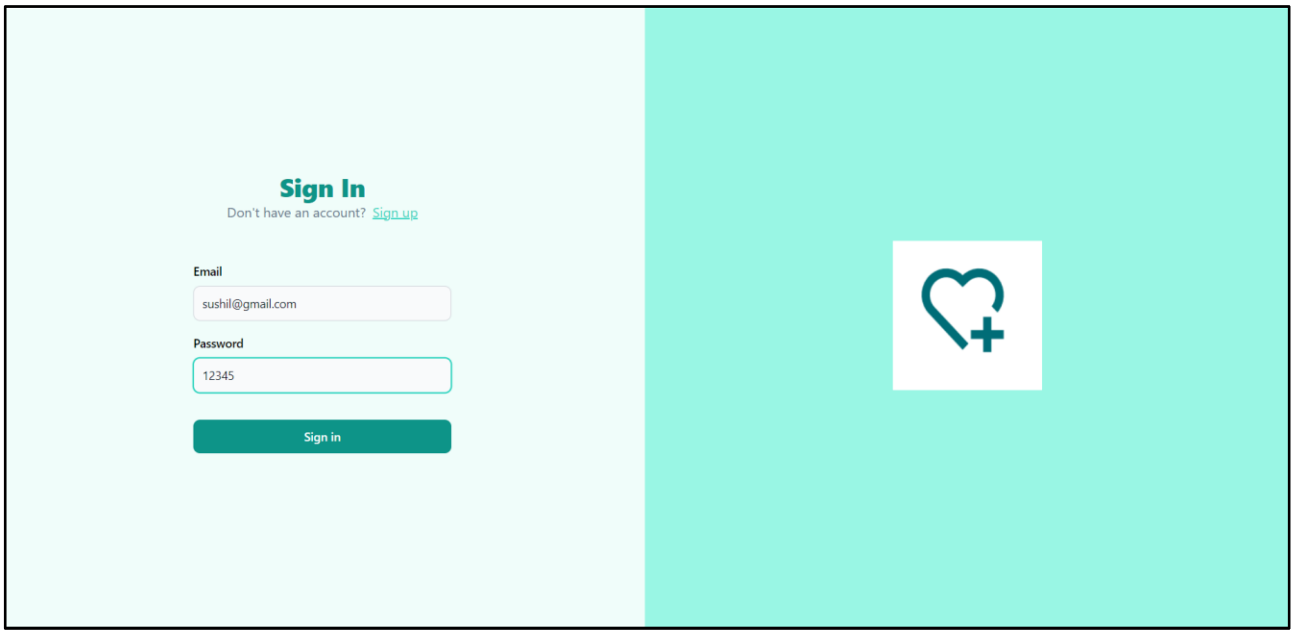
Confirm Password

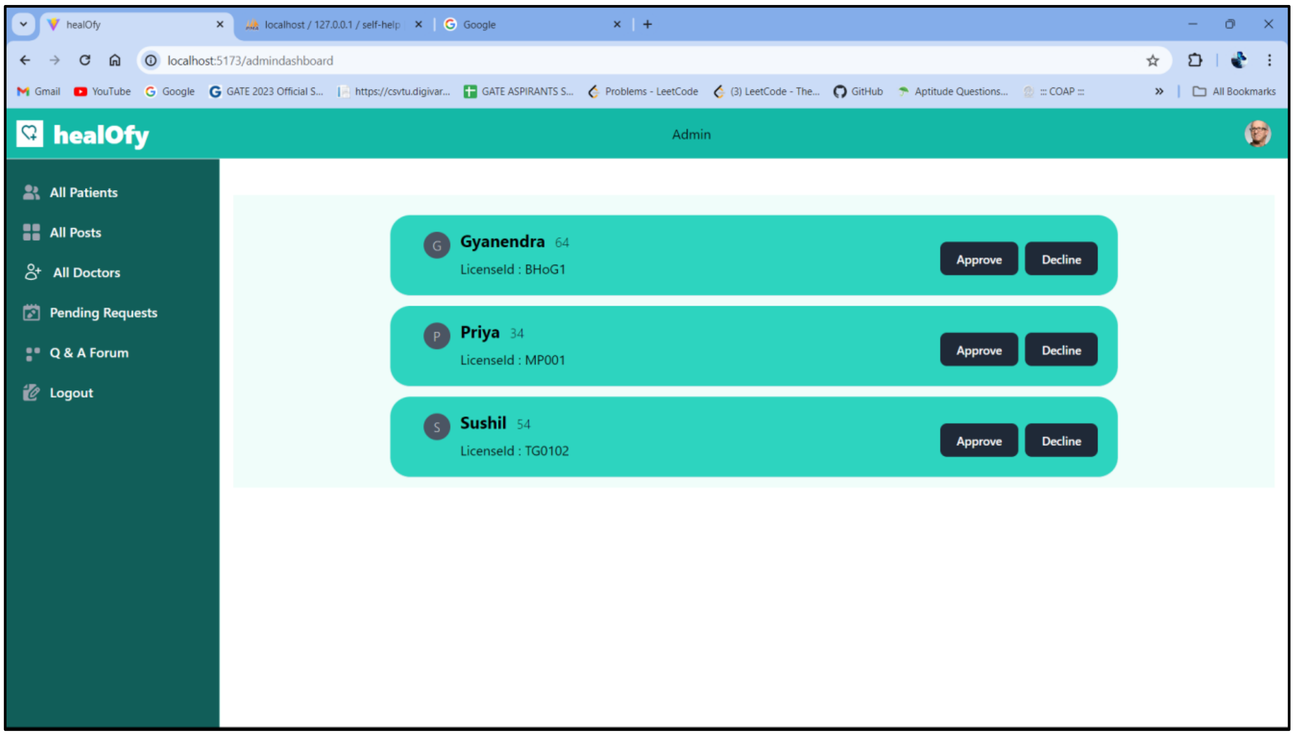
* * * * * *

...

Signup

Already have an account? Login

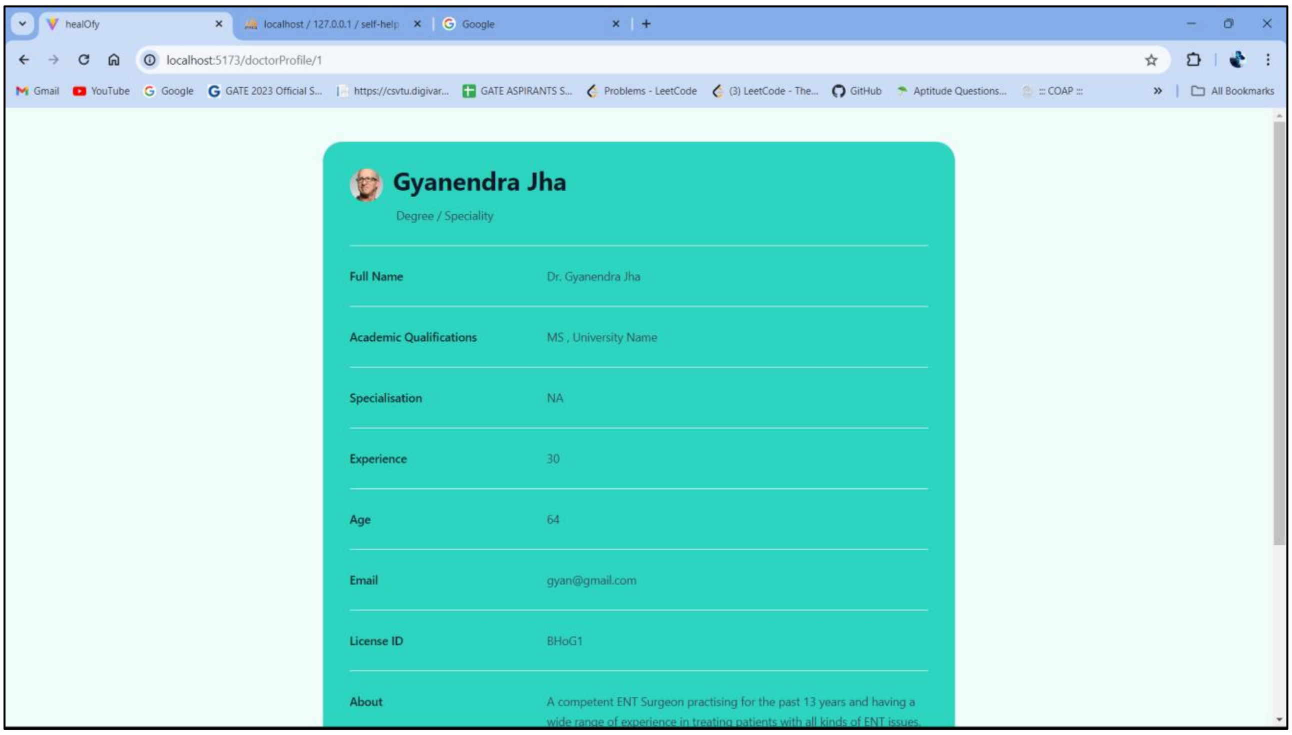




Gyanendra 64
LicenseId : BHoG1
Approve Decline

Priya 34
LicenseId : MP001
Approve Decline

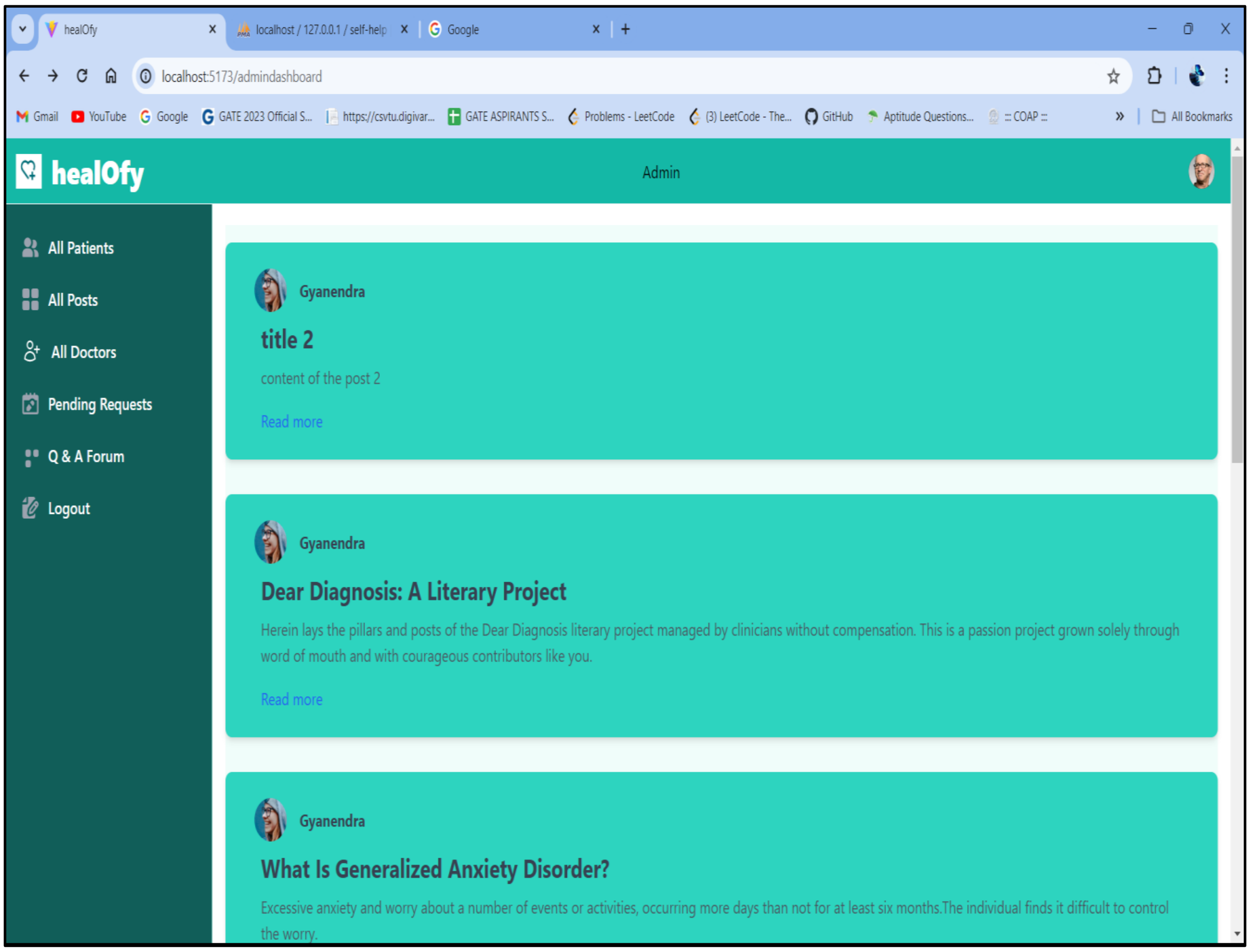
Sushil 54
LicenseId : TG0102
Approve Decline



Gyanendra Jha

Degree / Speciality

Full Name	Dr. Gyanendra Jha
Academic Qualifications	MS , University Name
Specialisation	NA
Experience	30
Age	64
Email	gyan@gmail.com
License ID	BHoG1
About	A competent ENT Surgeon practising for the past 13 years and having a wide range of experience in treating patients with all kinds of ENT issues.



Gyanendra

title 2

content of the post 2

Read more

Gyanendra

Dear Diagnosis: A Literary Project

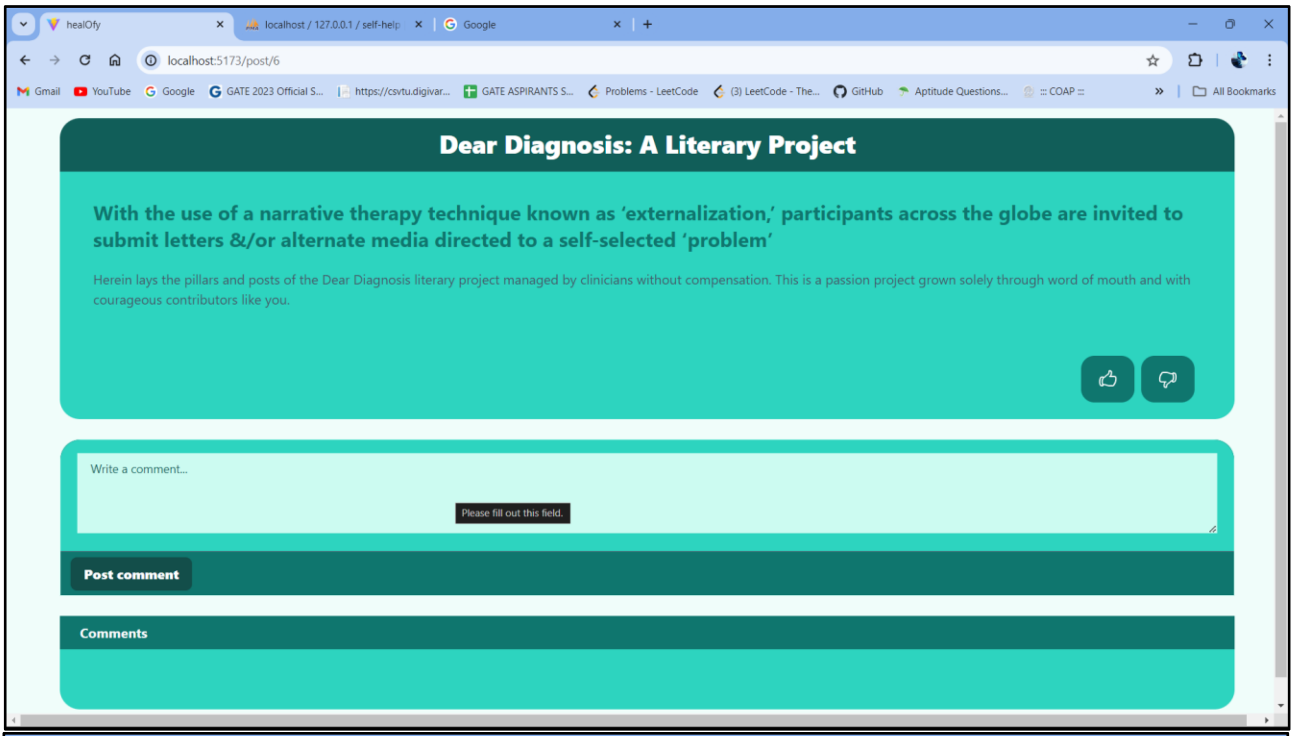
Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Read more

Gyanendra

What Is Generalized Anxiety Disorder?

Excessive anxiety and worry about a number of events or activities, occurring more days than not for at least six months.The individual finds it difficult to control the worry.

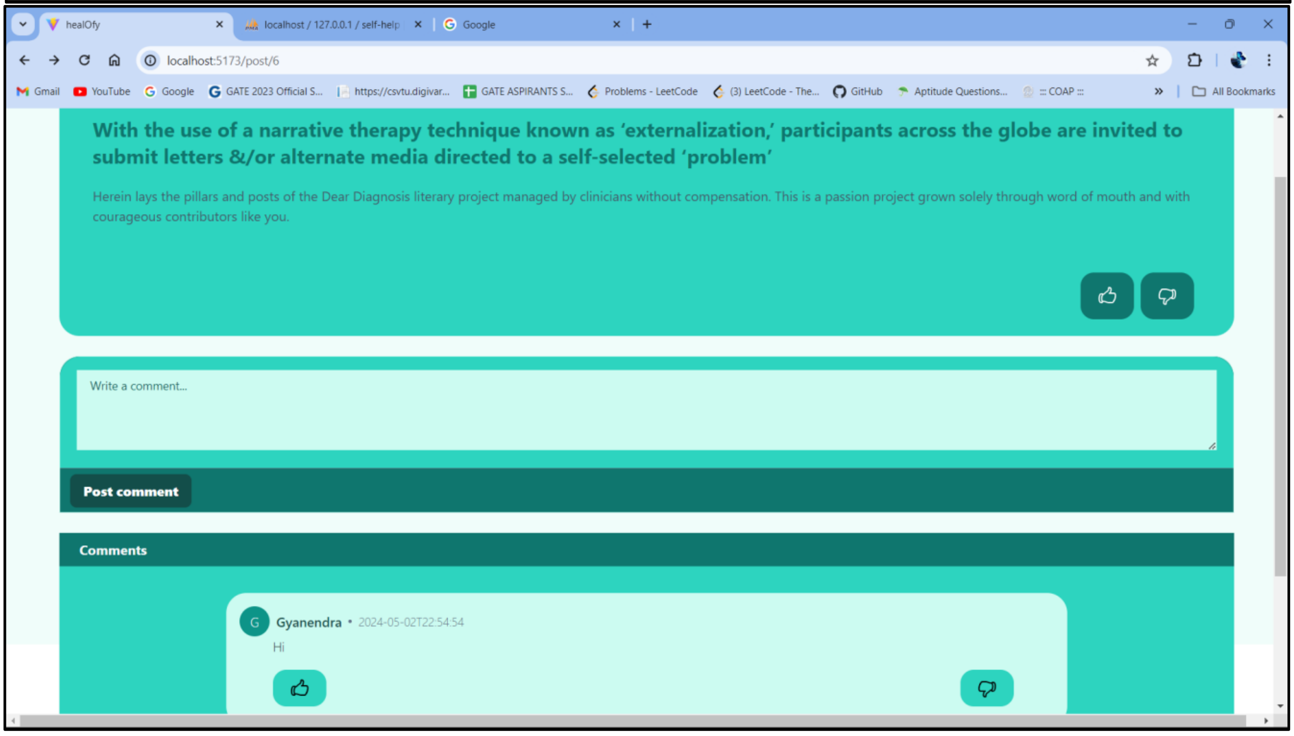

healOfy
localhost / 127.0.0.1 / self-help
Google
localhost:5173/post/6
Gmail YouTube Google GATE 2023 Official S... https://csvtu.digivar... GATE ASPIRANTS S... Problems - LeetCode (3) LeetCode - The... GitHub Aptitude Questions... :: COAP :: All Bookmarks

Dear Diagnosis: A Literary Project

With the use of a narrative therapy technique known as 'externalization,' participants across the globe are invited to submit letters &/or alternate media directed to a self-selected 'problem'

Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Write a comment...
Please fill out this field.
Post comment
Comments

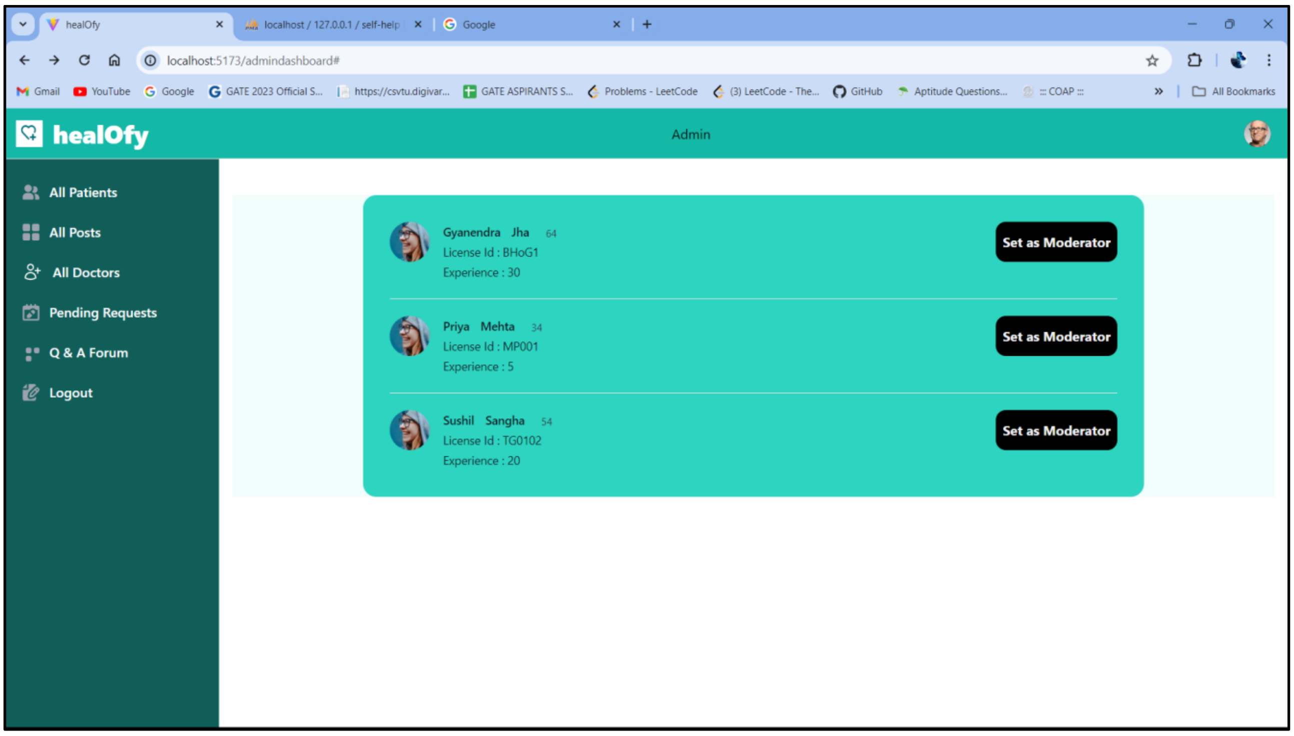

healOfy
localhost / 127.0.0.1 / self-help
Google
localhost:5173/post/6
Gmail YouTube Google GATE 2023 Official S... https://csvtu.digivar... GATE ASPIRANTS S... Problems - LeetCode (3) LeetCode - The... GitHub Aptitude Questions... :: COAP :: All Bookmarks

With the use of a narrative therapy technique known as 'externalization,' participants across the globe are invited to submit letters &/or alternate media directed to a self-selected 'problem'

Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

G Gyanendra • 2024-05-02T22:54:54

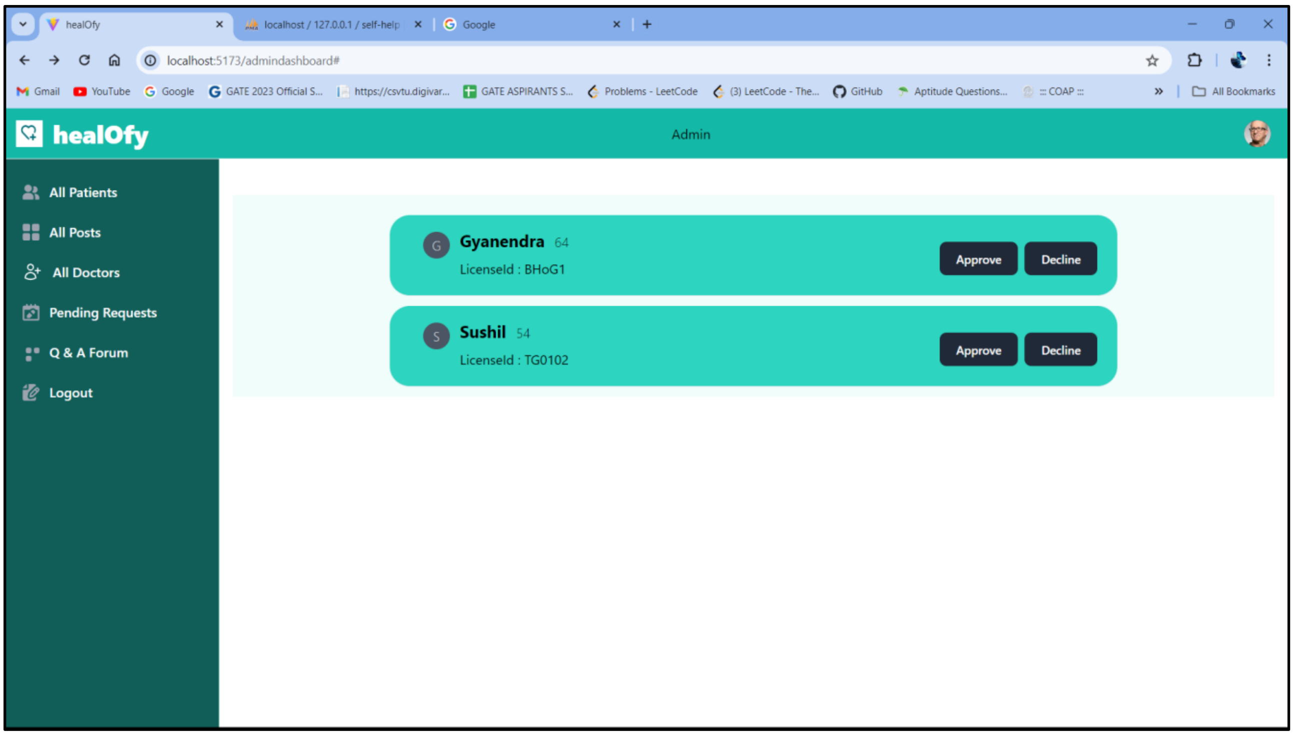
Hi



Gyanendra Jha 64
License Id : BHoG1
Experience : 30

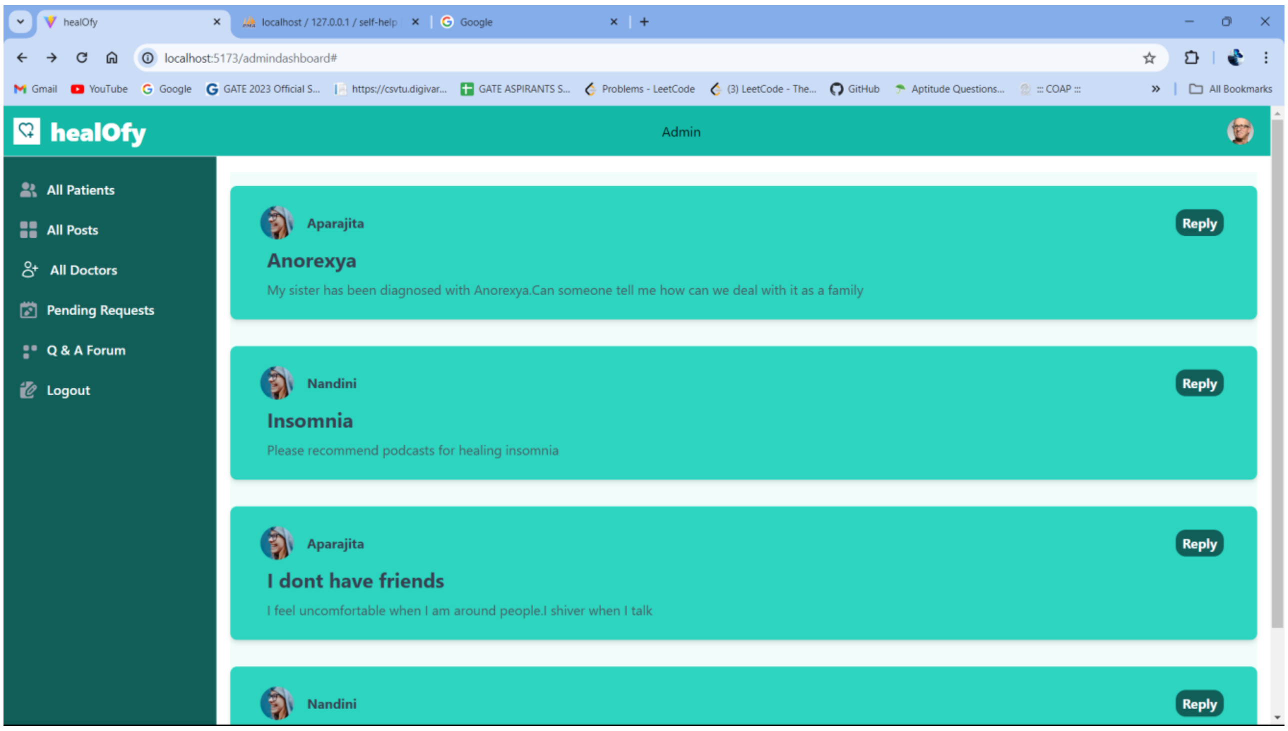
Priya Mehta 34
License Id : MP001
Experience : 5

Sushil Sangha 54
License Id : TG0102
Experience : 20



Gyanendra 64
LicenseId : BHoG1

Sushil 54
LicenseId : TG0102



Aparajita

Anorexya

My sister has been diagnosed with Anorexya.Can someone tell me how can we deal with it as a family

Reply

Nandini

Insomnia

Please recommend podcasts for healing insomnia

Reply

Aparajita

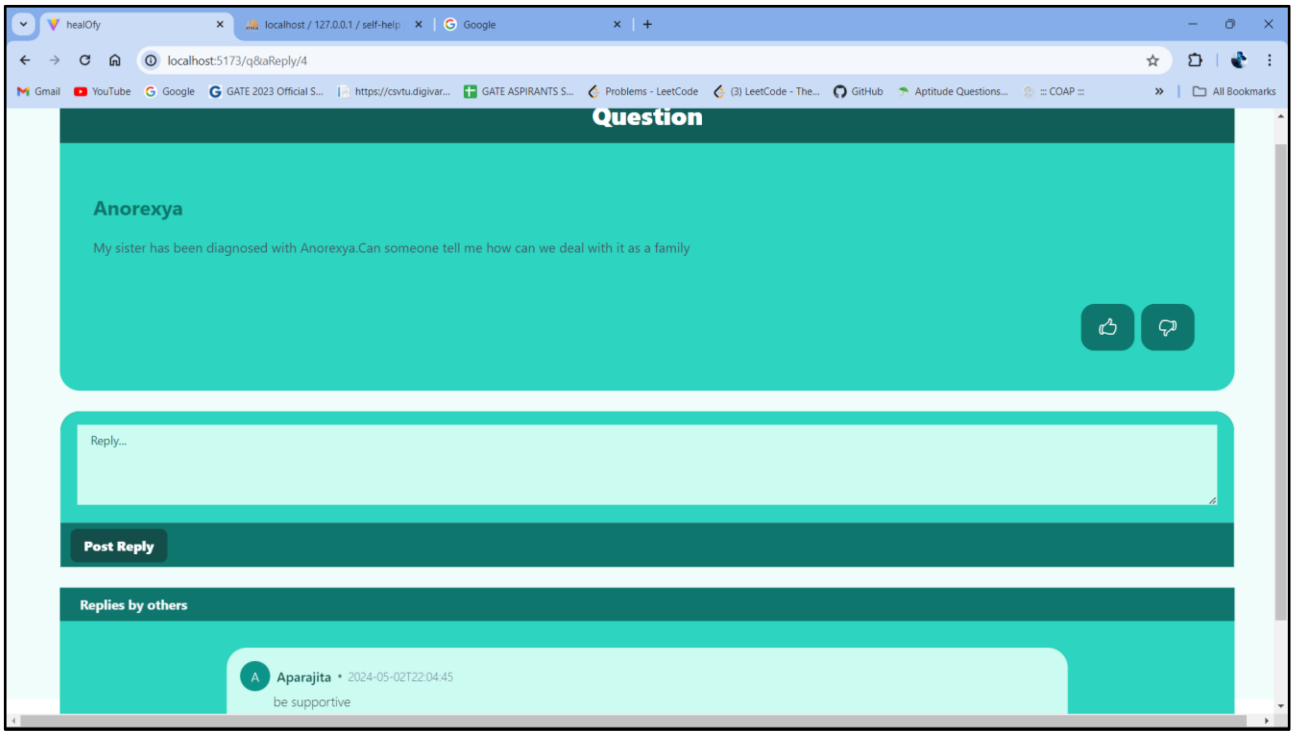
I dont have friends

I feel uncomfortable when I am around people.I shiver when I talk

Reply

Nandini

Reply



Question

Anorexya

My sister has been diagnosed with Anorexya.Can someone tell me how can we deal with it as a family

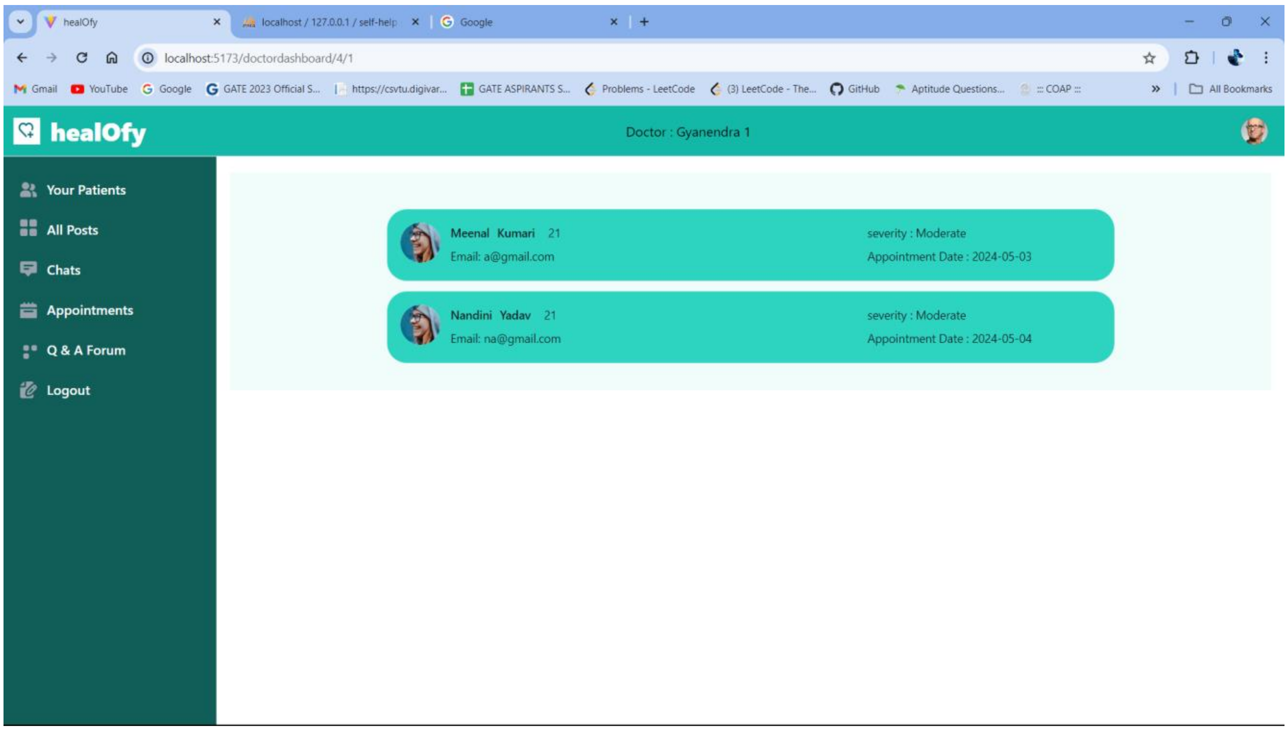
Reply...

Post Reply

Replies by others

Aparajita • 2024-05-02T22:04:45

be supportive


localhost:5173/doctordashboard/4/1

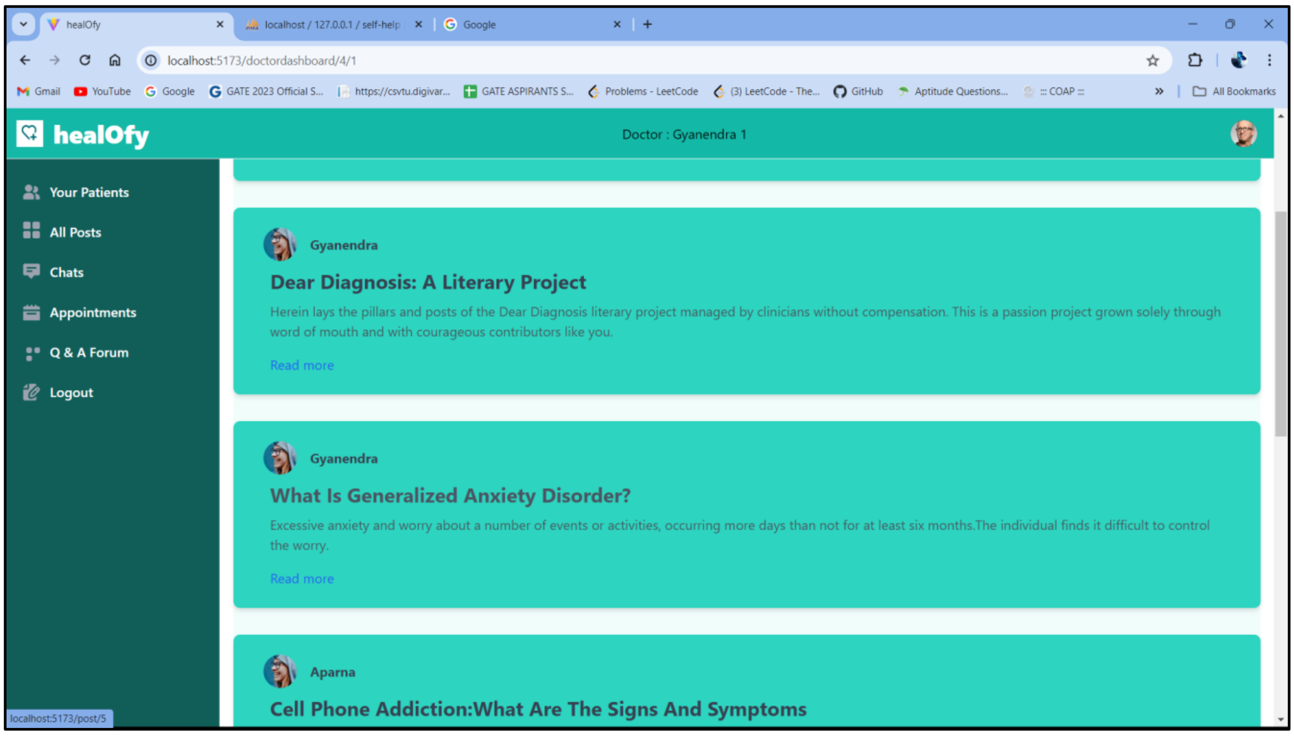
healOfy

Doctor : Gyanendra 1

- Your Patients
- All Posts
- Chats
- Appointments
- Q & A Forum
- Logout

Meenal Kumari 21
Email: a@gmail.com
severity : Moderate
Appointment Date : 2024-05-03

Nandini Yadav 21
Email: na@gmail.com
severity : Moderate
Appointment Date : 2024-05-04


localhost:5173/doctordashboard/4/1

healOfy

Doctor : Gyanendra 1

- Your Patients
- All Posts
- Chats
- Appointments
- Q & A Forum
- Logout

Gyanendra

Dear Diagnosis: A Literary Project

Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Read more

Gyanendra

What Is Generalized Anxiety Disorder?

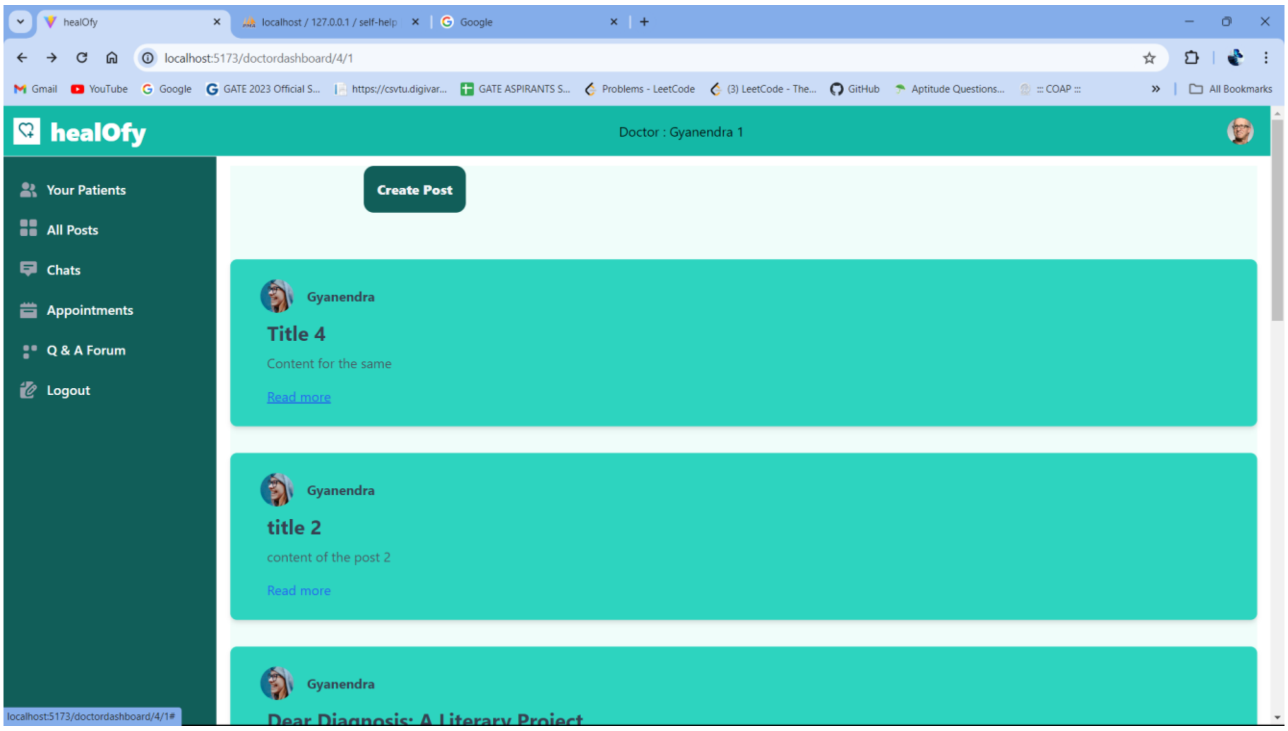
Excessive anxiety and worry about a number of events or activities, occurring more days than not for at least six months.The individual finds it difficult to control the worry.

Read more

Aparna

Cell Phone Addiction:What Are The Signs And Symptoms

localhost:5173/post/5



Gyanendra

Title 4

Content for the same

Read more

Gyanendra

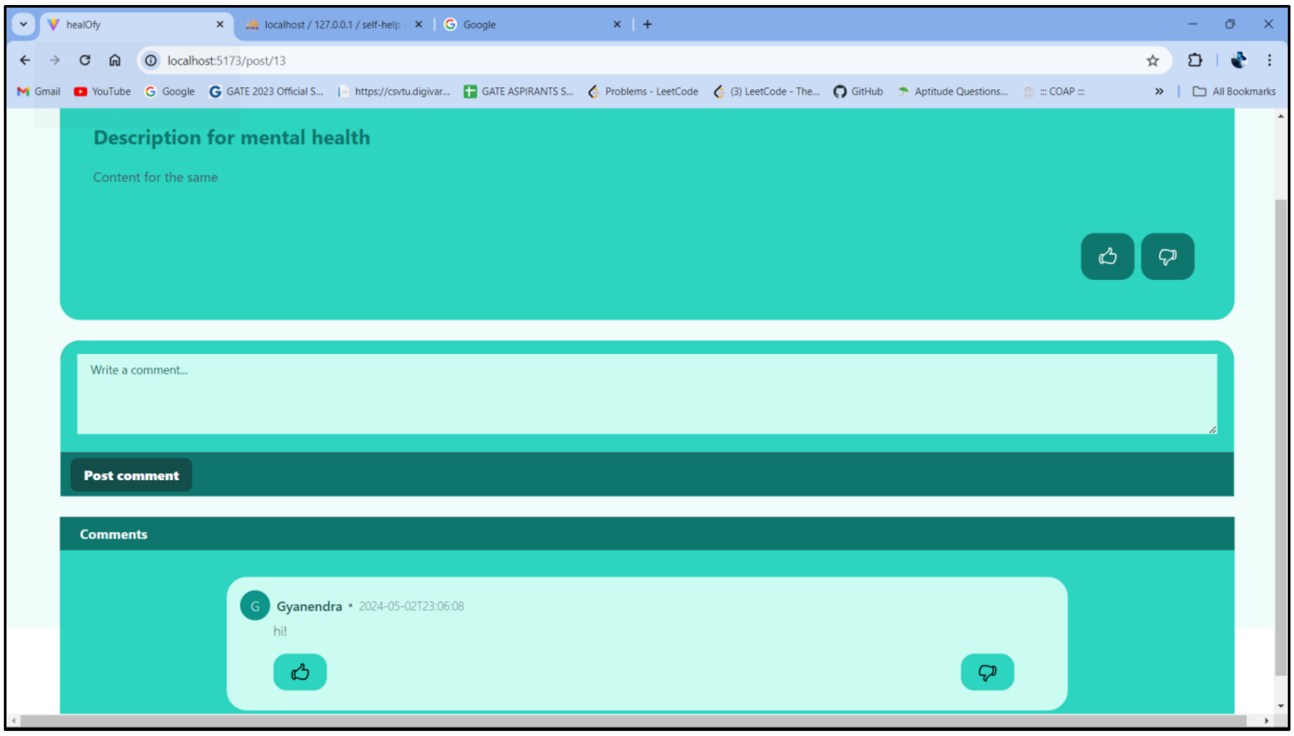
title 2

content of the post 2

Read more

Gyanendra

Dear Diagnosis: A Literary Project

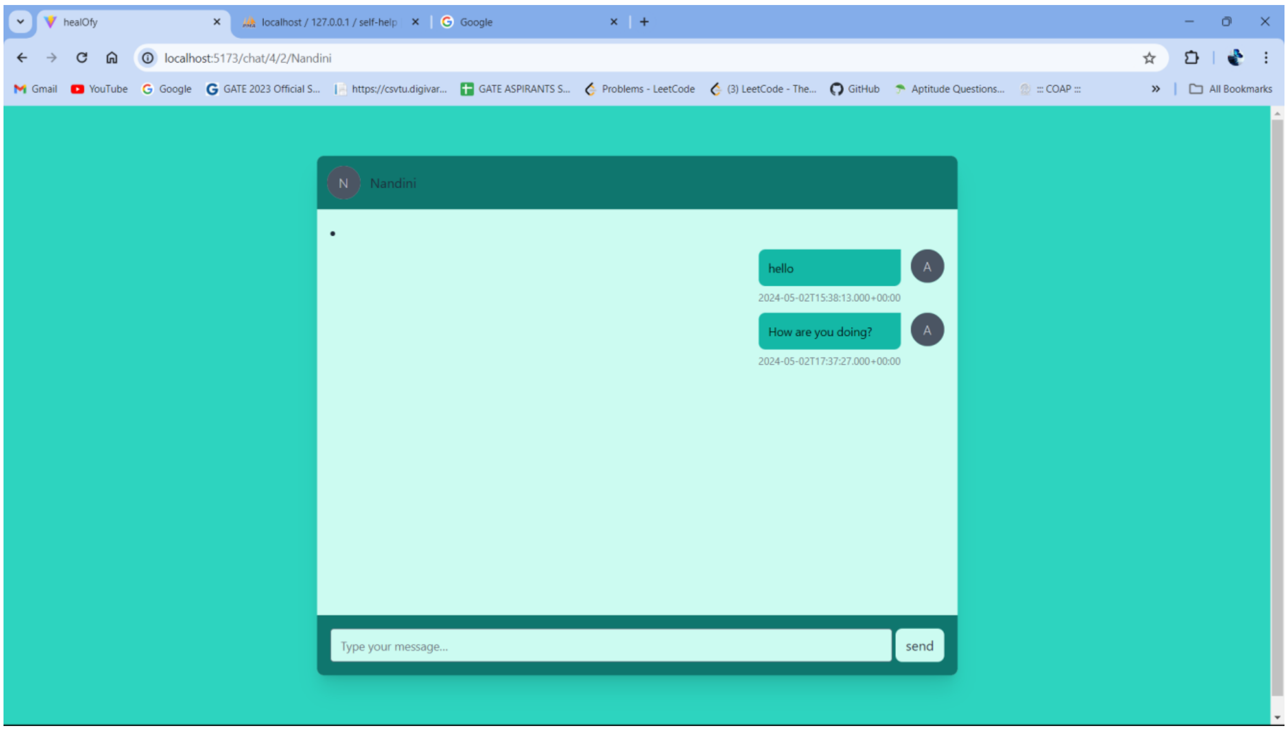


Description for mental health

Content for the same

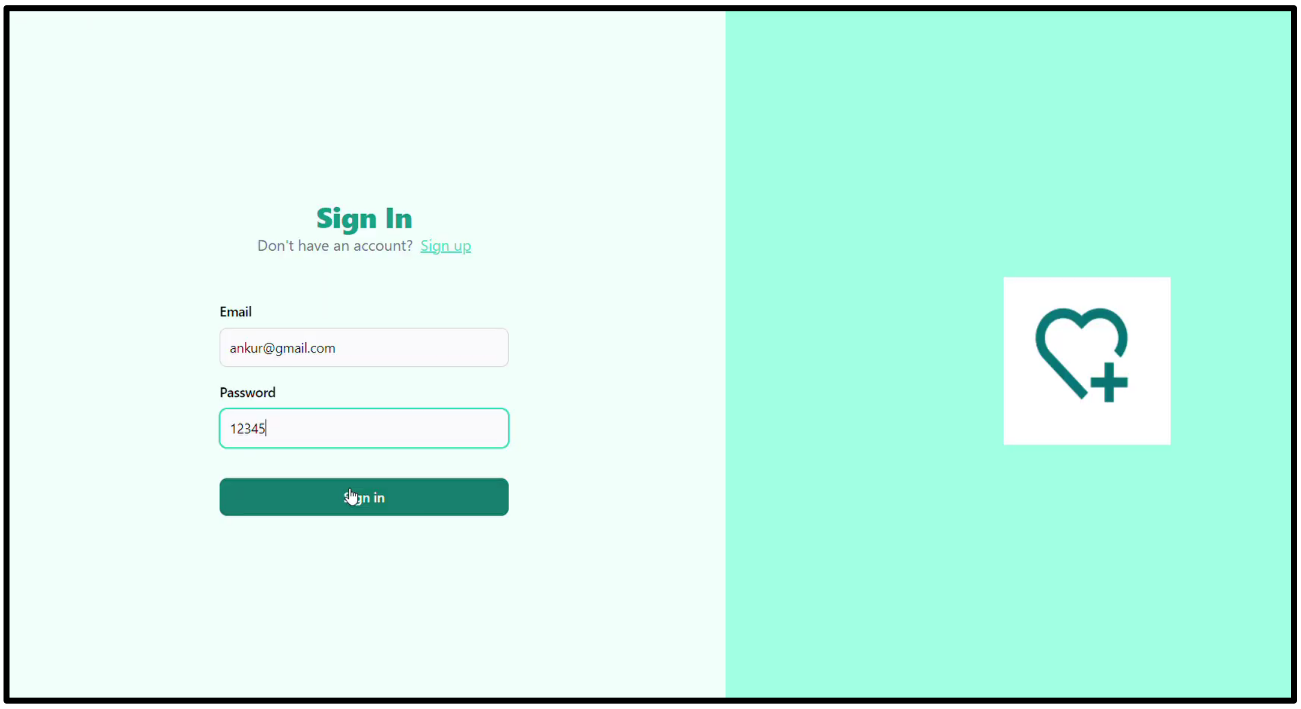
Gyanendra • 2024-05-02T23:06:08

hi!



hello
2024-05-02T15:38:13.000+00:00

How are you doing?
2024-05-02T17:37:27.000+00:00

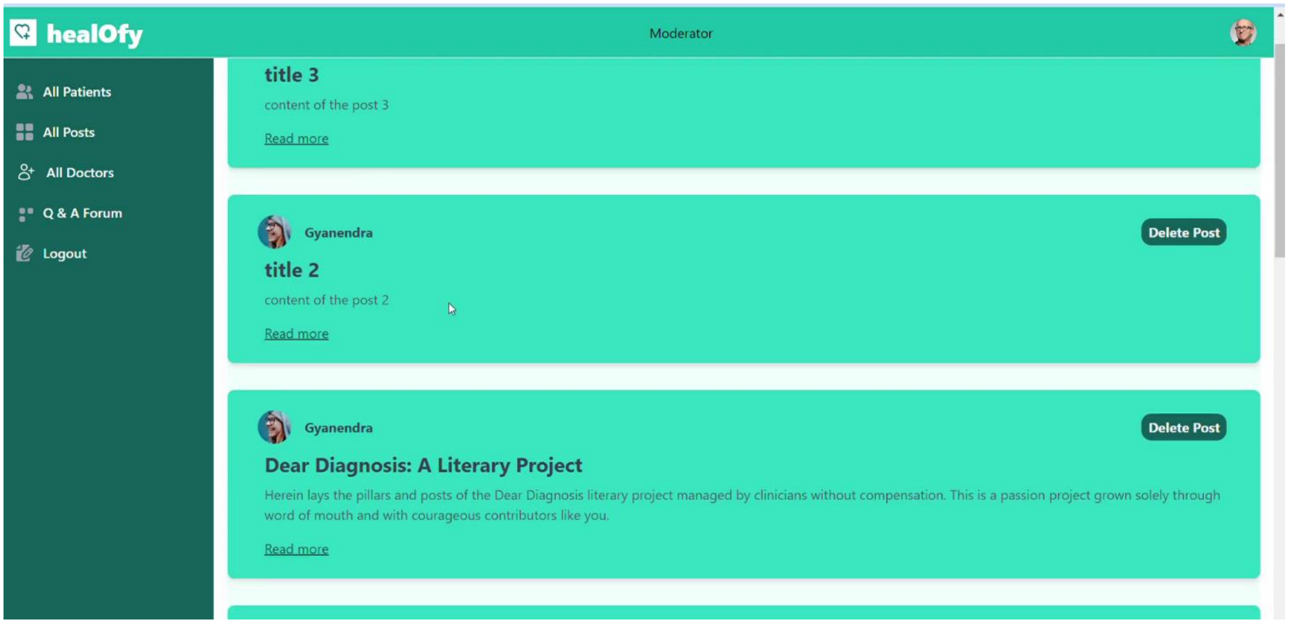


Sign In

Don't have an account? Sign up

Email: ankur@gmail.com

Password: 12345



title 3

content of the post 3

Read more

Gyanendra

Delete Post

title 2

content of the post 2

Read more

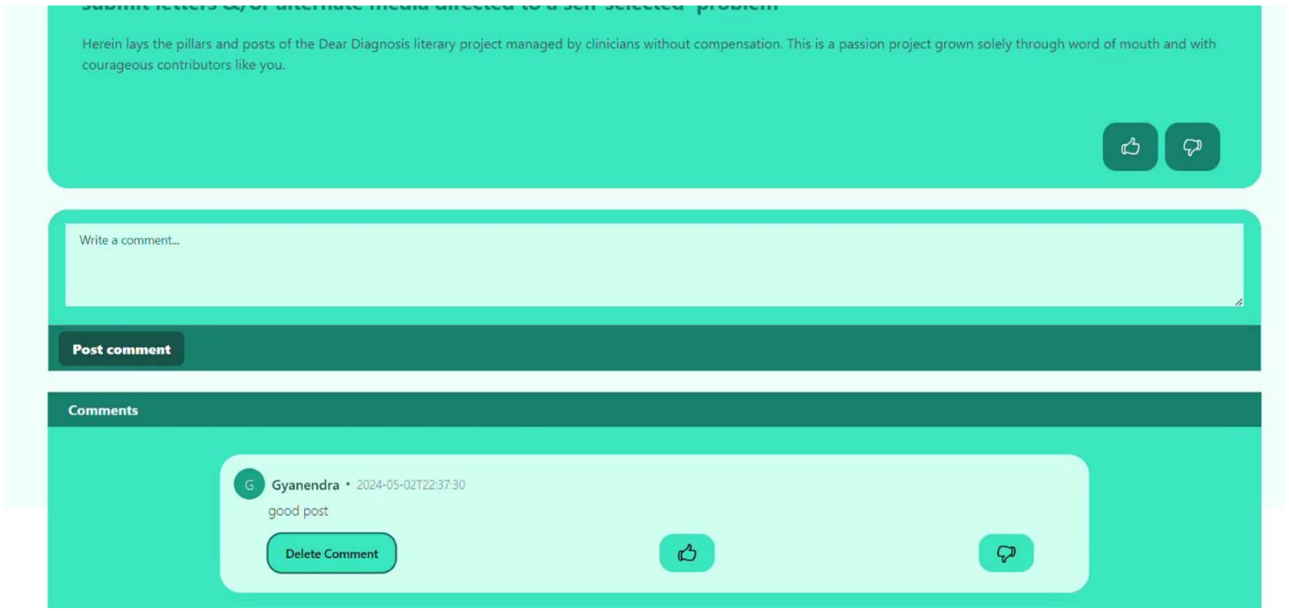
Gyanendra

Delete Post

Dear Diagnosis: A Literary Project

Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Read more

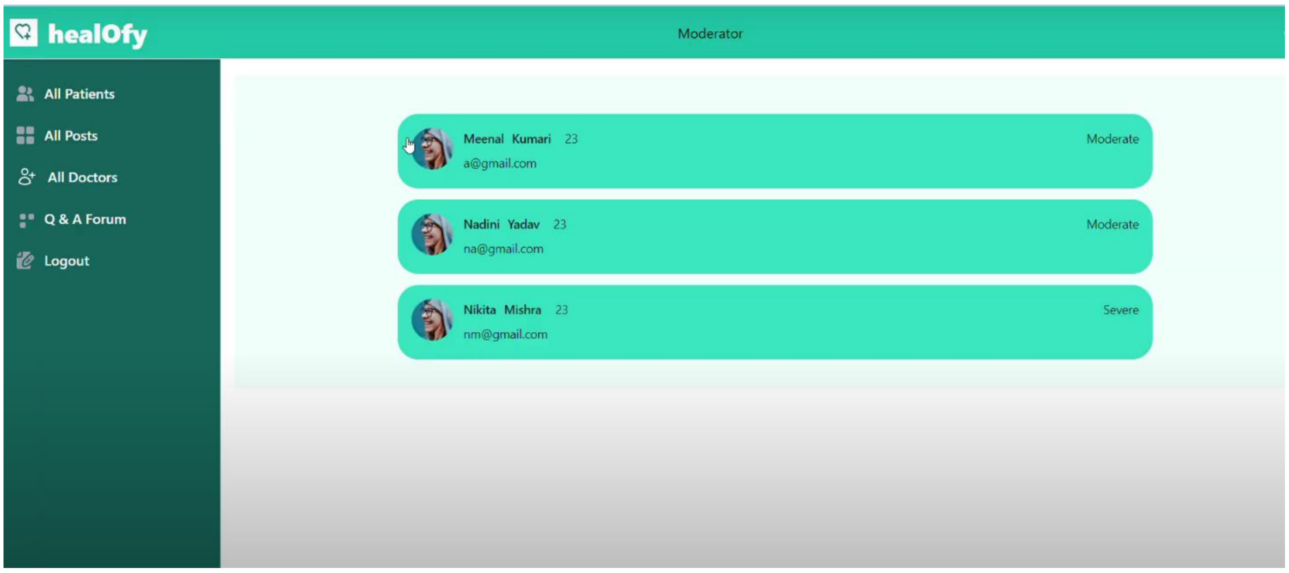


Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Write a comment...

Gyanendra • 2024-05-02T22:37:30

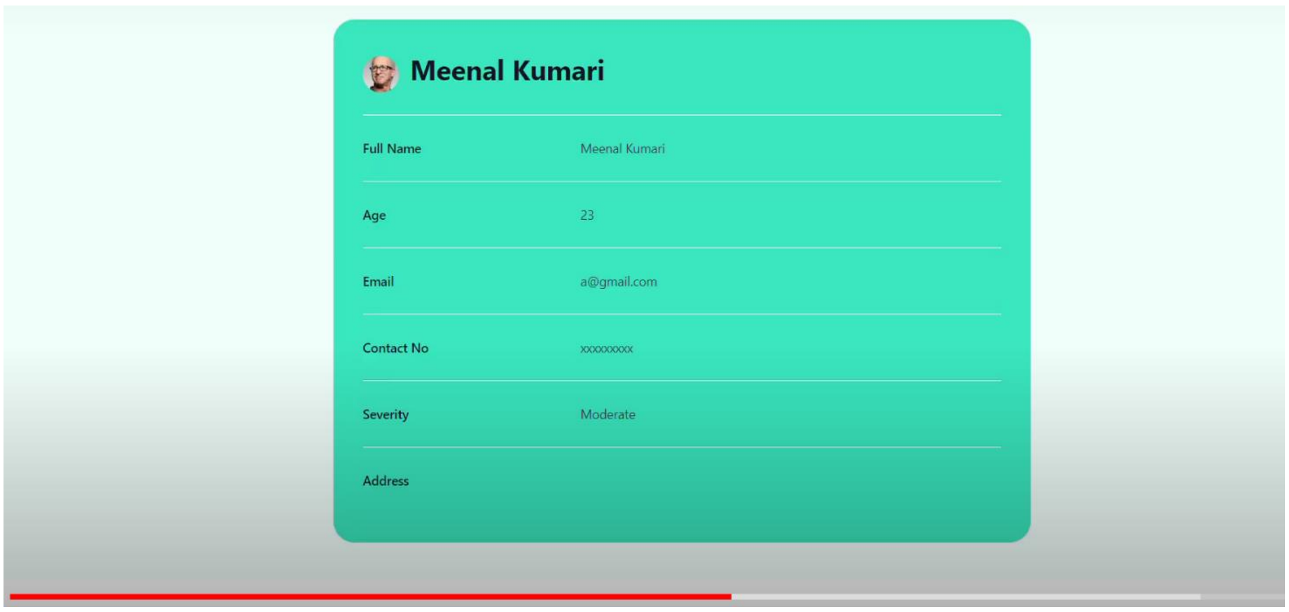
good post



Meenal Kumari 23 Moderate
a@gmail.com

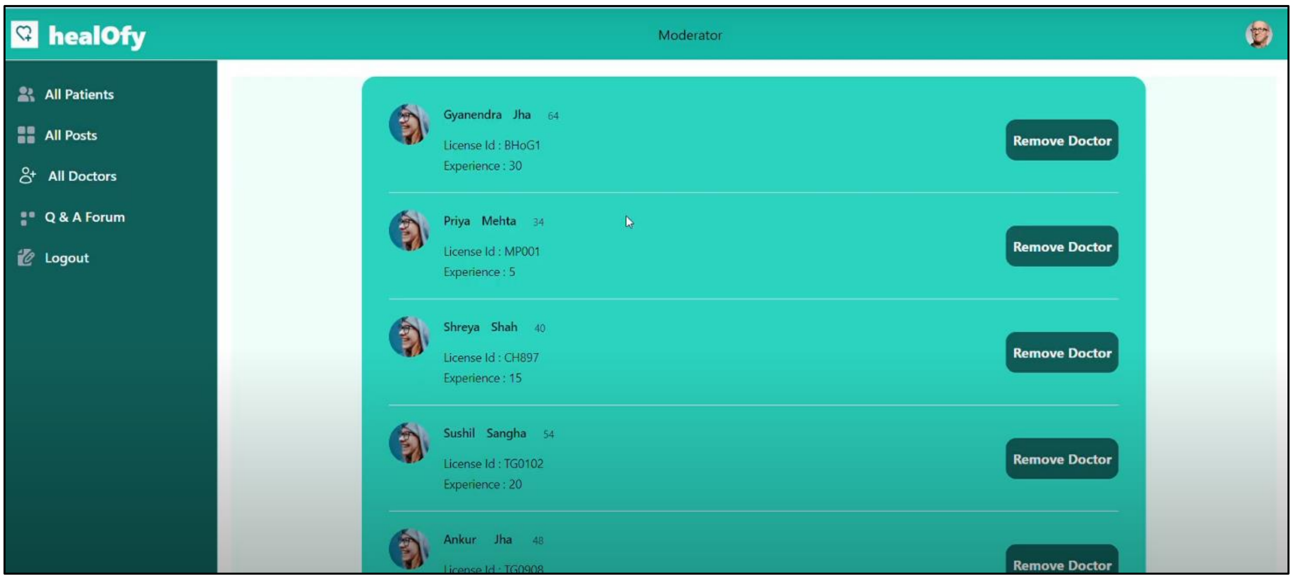
Nadini Yadav 23 Moderate
na@gmail.com

Nikita Mishra 23 Severe
nm@gmail.com



Meenal Kumari

Full Name	Meenal Kumari
Age	23
Email	a@gmail.com
Contact No	xxxxxxxxx
Severity	Moderate
Address	



healOfy   Moderator

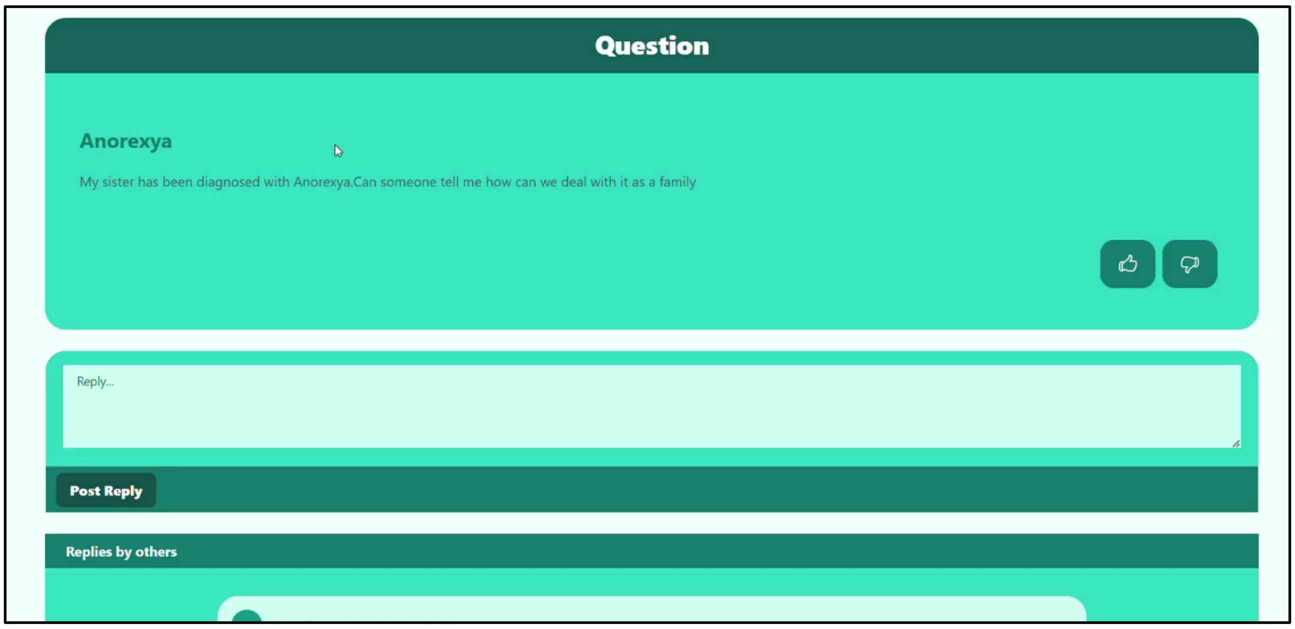
Gyanendra Jha 64
License Id : BHoG1
Experience : 30
Remove Doctor

Priya Mehta 34
License Id : MP001
Experience : 5
Remove Doctor

Shreya Shah 40
License Id : CH897
Experience : 15
Remove Doctor

Sushil Sangha 54
License Id : TG0102
Experience : 20
Remove Doctor

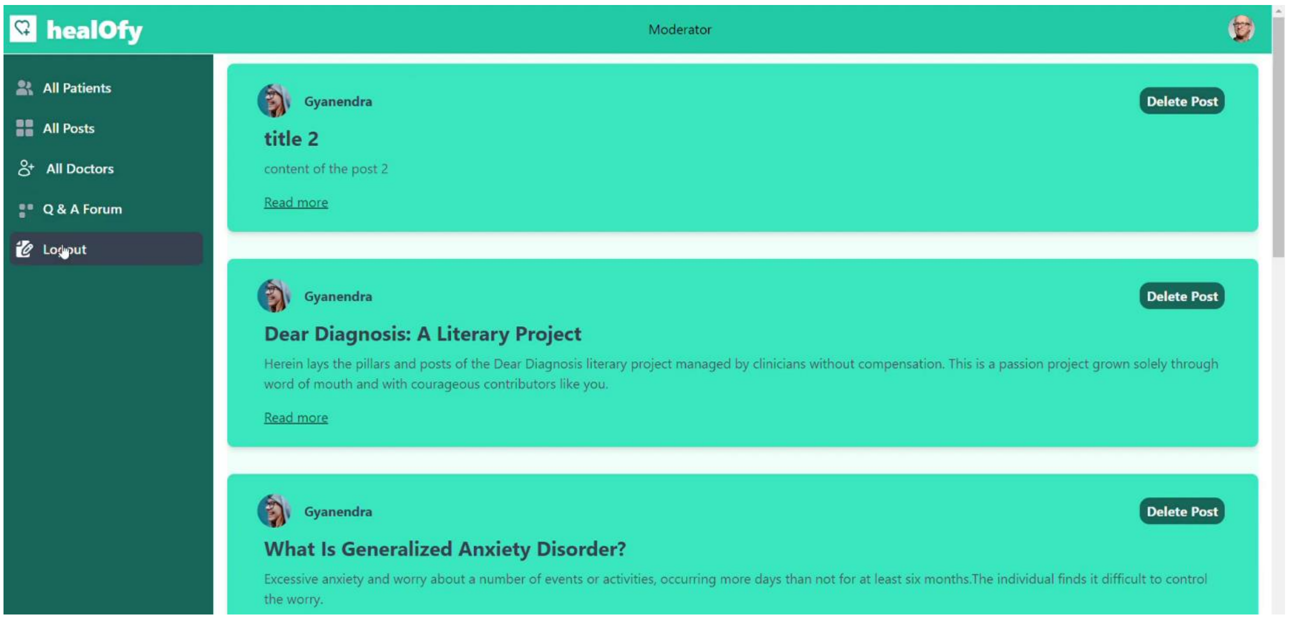
Ankur Jha 48
License Id : TG0908
Remove Doctor



Question

Anorexya

My sister has been diagnosed with Anorexya.Can someone tell me how can we deal with it as a family



Gyanendra

title 2

content of the post 2

Gyanendra

Dear Diagnosis: A Literary Project

Herein lays the pillars and posts of the Dear Diagnosis literary project managed by clinicians without compensation. This is a passion project grown solely through word of mouth and with courageous contributors like you.

Gyanendra

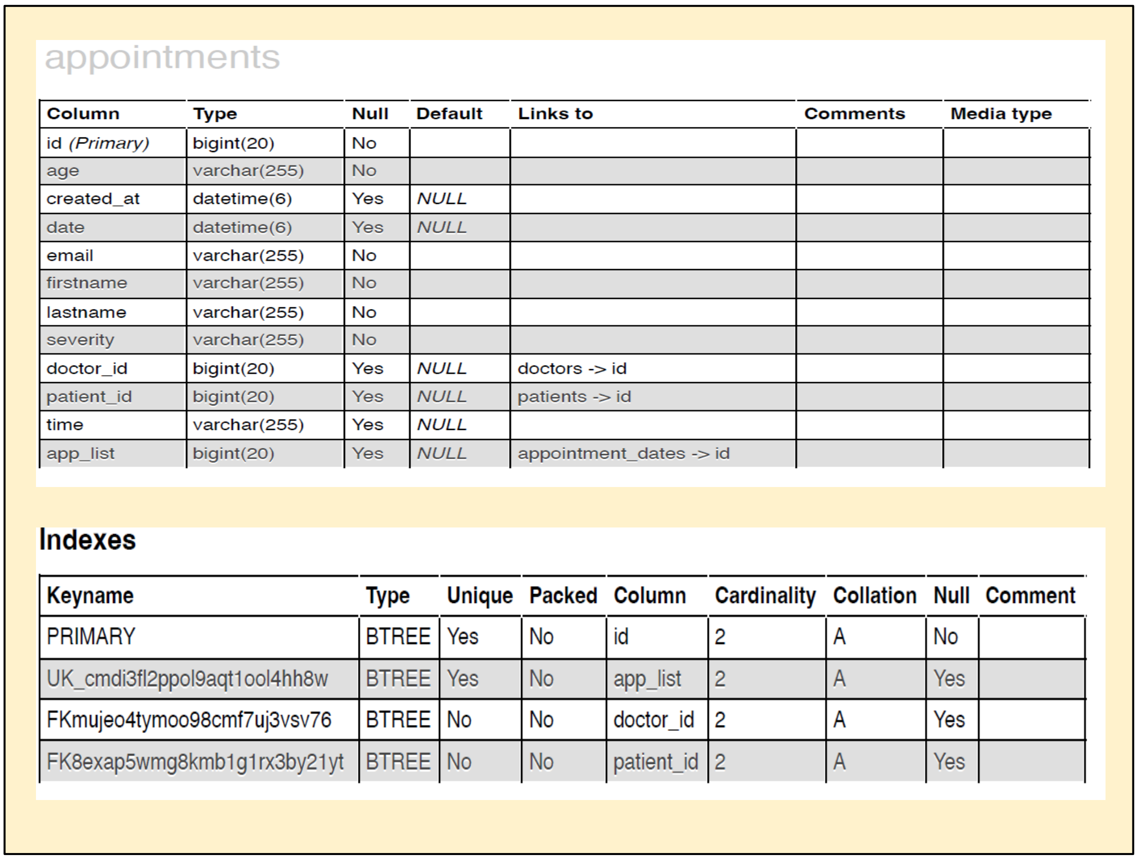
What Is Generalized Anxiety Disorder?

Excessive anxiety and worry about a number of events or activities, occurring more days than not for at least six months.The individual finds it difficult to control the worry.

Database Design, Data Management

This Architecture contains a DataBase named Self-Help .

1. **Self-Help Database:** This database schema, likely for a self-help or healthcare application, includes several tables such as appointments, chats, users, roles, and more. Each table contains specific columns defining attributes like appointment details, user information,moderator and admin roles. The schema is designed to support functionalities such as appointment scheduling, user management, and facilitating communication between users and professionals. Detailed indexes are defined to optimize data retrieval and ensure efficient database operations. If you have specific inquiries about this schema or its implementation, please let me know!



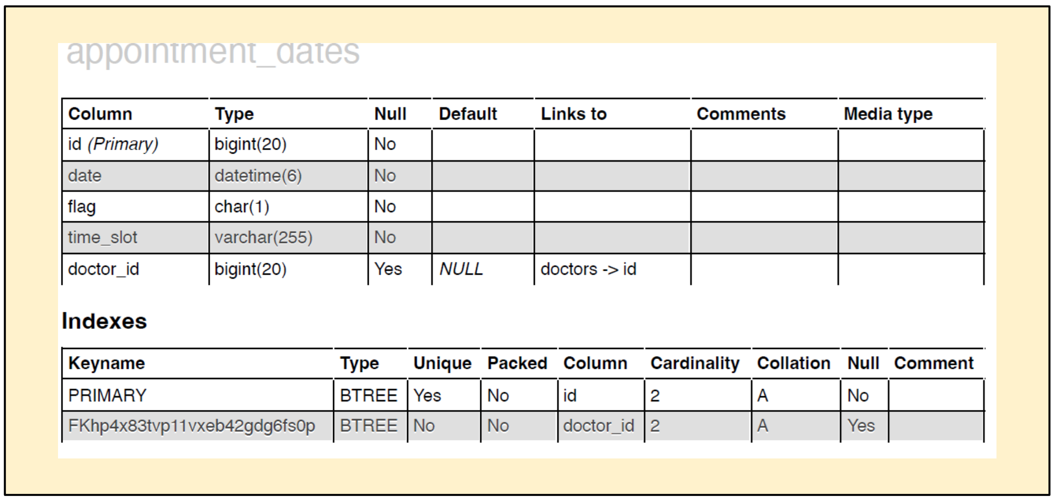
appointments

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
age	varchar(255)	No				
created_at	datetime(6)	Yes	*NULL*			
date	datetime(6)	Yes	*NULL*			
email	varchar(255)	No				
firstname	varchar(255)	No				
lastname	varchar(255)	No				
severity	varchar(255)	No				
doctor_id	bigint(20)	Yes	*NULL*	doctors -> id		
patient_id	bigint(20)	Yes	*NULL*	patients -> id		
time	varchar(255)	Yes	*NULL*			
app_list	bigint(20)	Yes	*NULL*	appointment_dates -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	
UK_cmdi3fl2ppol9aqt1ool4hh8w	BTREE	Yes	No	app_list	2	A	Yes	
FKmujeo4tymoo98cmf7uj3vsv76	BTREE	No	No	doctor_id	2	A	Yes	
FK8exap5wmg8kmb1g1rx3by21yt	BTREE	No	No	patient_id	2	A	Yes	

Fig. appointments table

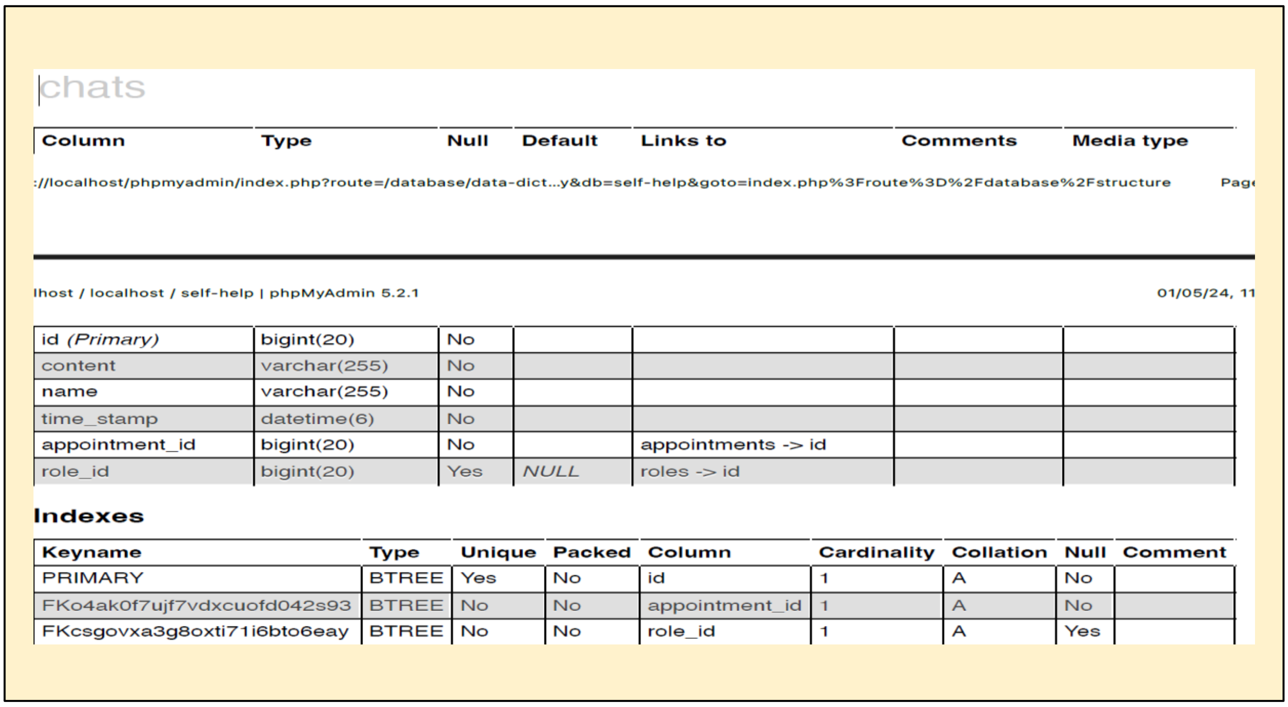

appointment_dates

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
date	datetime(6)	No				
flag	char(1)	No				
time_slot	varchar(255)	No				
doctor_id	bigint(20)	Yes	*NULL*	doctors -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	
FKhp4x83tvp11vxeb42gdg6fs0p	BTREE	No	No	doctor_id	2	A	Yes	

Fig. Appointments_date Table


chats

:/localhost/phpmyadmin/index.php?route=/database/data-dict...y&db=self-help&goto=index.php%3Froute%3D%2Fdatabase%2Fstructure Page

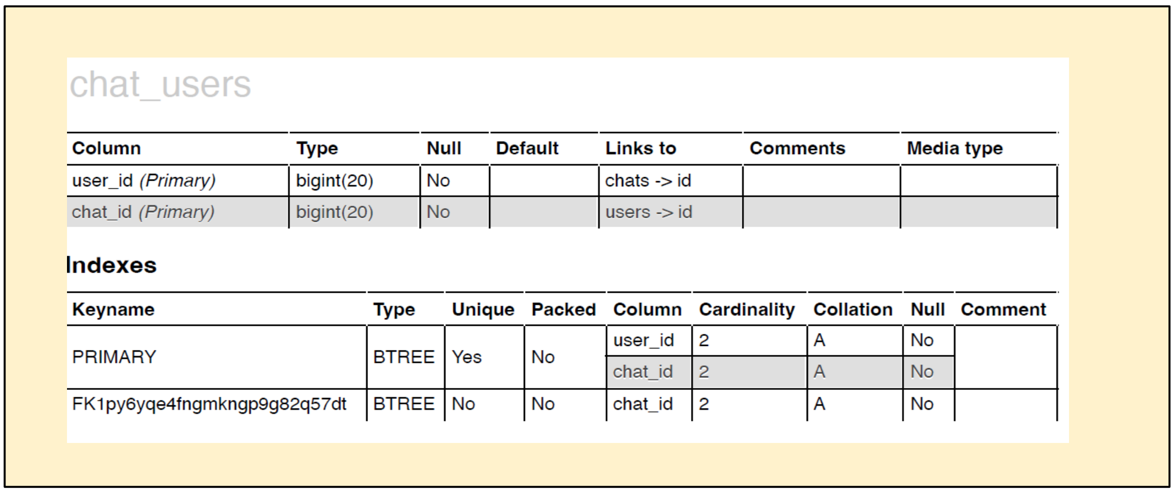
lhost / localhost / self-help | phpMyAdmin 5.2.1 01/05/24, 11

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
content	varchar(255)	No				
name	varchar(255)	No				
time_stamp	datetime(6)	No				
appointment_id	bigint(20)	No		appointments -> id		
role_id	bigint(20)	Yes	*NULL*	roles -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	
FKo4ak0f7ujf7vdxcuofd042s93	BTREE	No	No	appointment_id	1	A	No	
FKcsgovxa3g8oxti71i6bto6eay	BTREE	No	No	role_id	1	A	Yes	

Fig.chats table

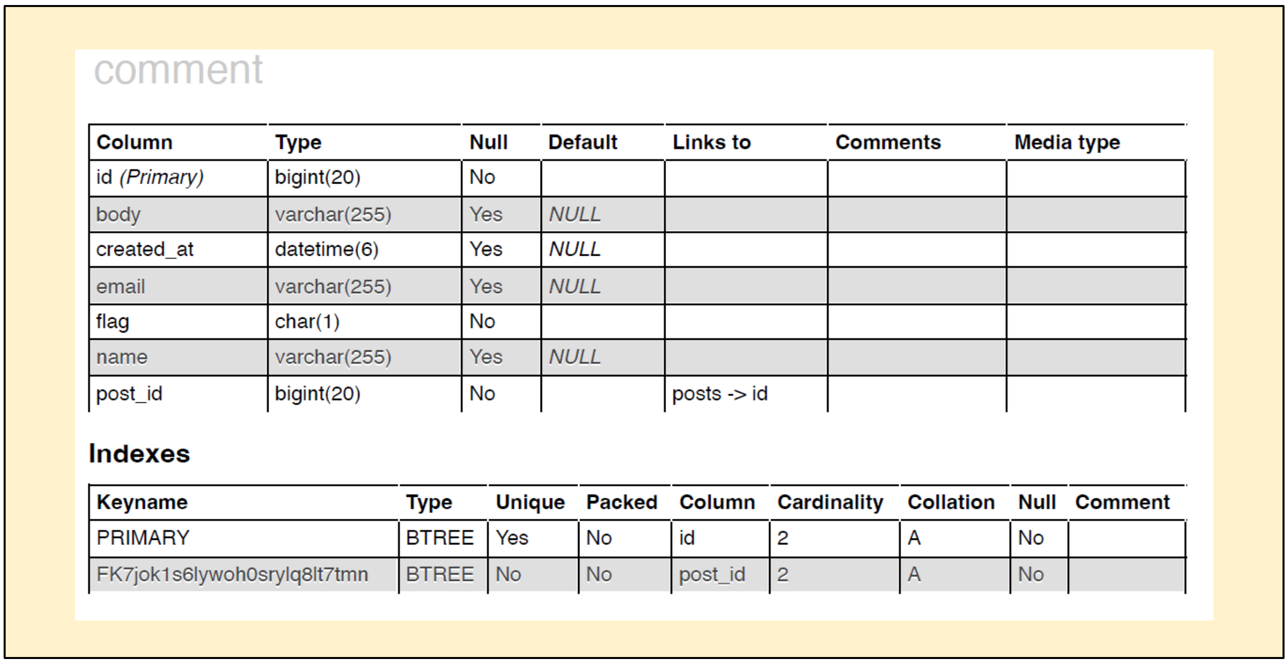

chat_users

Column	Type	Null	Default	Links to	Comments	Media type
user_id *(Primary)*	bigint(20)	No		chats -> id		
chat_id *(Primary)*	bigint(20)	No		users -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	user_id	2	A	No	
				chat_id	2	A	No	
FK1py6yqe4fngmkngp9g82q57dt	BTREE	No	No	chat_id	2	A	No	

Fig. chats_users Table



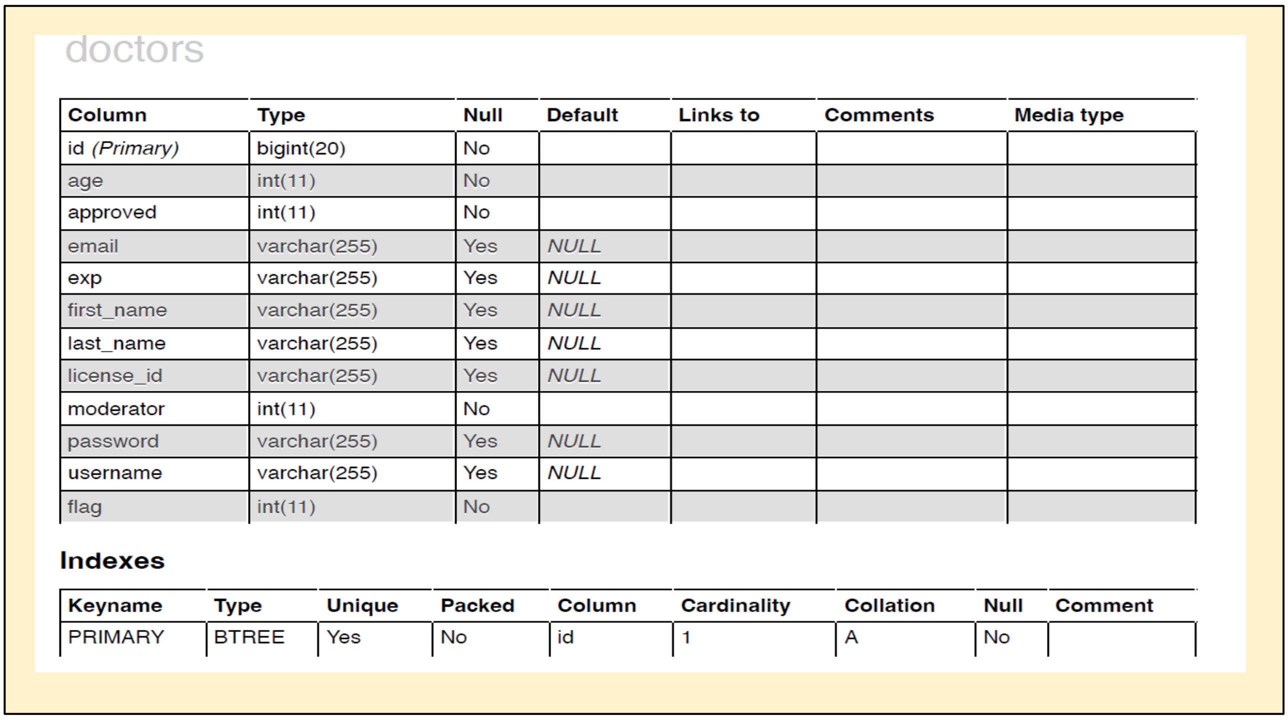
comment

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
body	varchar(255)	Yes	*NULL*			
created_at	datetime(6)	Yes	*NULL*			
email	varchar(255)	Yes	*NULL*			
flag	char(1)	No				
name	varchar(255)	Yes	*NULL*			
post_id	bigint(20)	No		posts -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	
FK7jok1s6lywoh0srylq8lt7tmn	BTREE	No	No	post_id	2	A	No	

Fig.comment Table



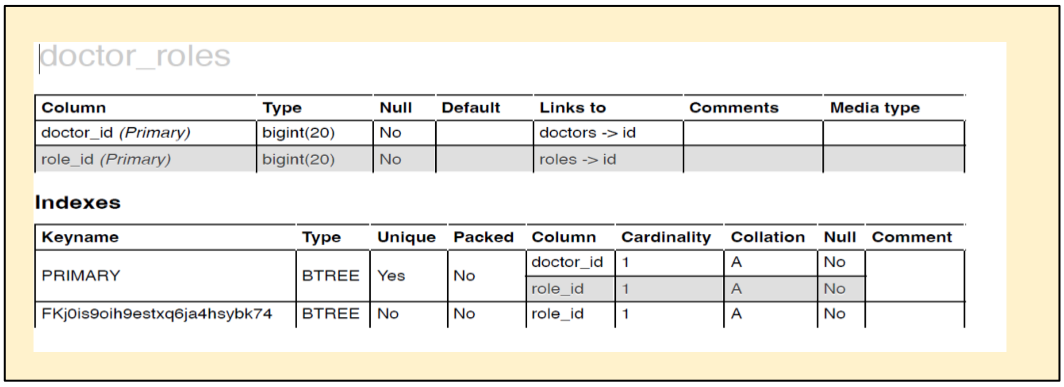
doctors

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
age	int(11)	No				
approved	int(11)	No				
email	varchar(255)	Yes	*NULL*			
exp	varchar(255)	Yes	*NULL*			
first_name	varchar(255)	Yes	*NULL*			
last_name	varchar(255)	Yes	*NULL*			
license_id	varchar(255)	Yes	*NULL*			
moderator	int(11)	No				
password	varchar(255)	Yes	*NULL*			
username	varchar(255)	Yes	*NULL*			
flag	int(11)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	

Fig.doctors table



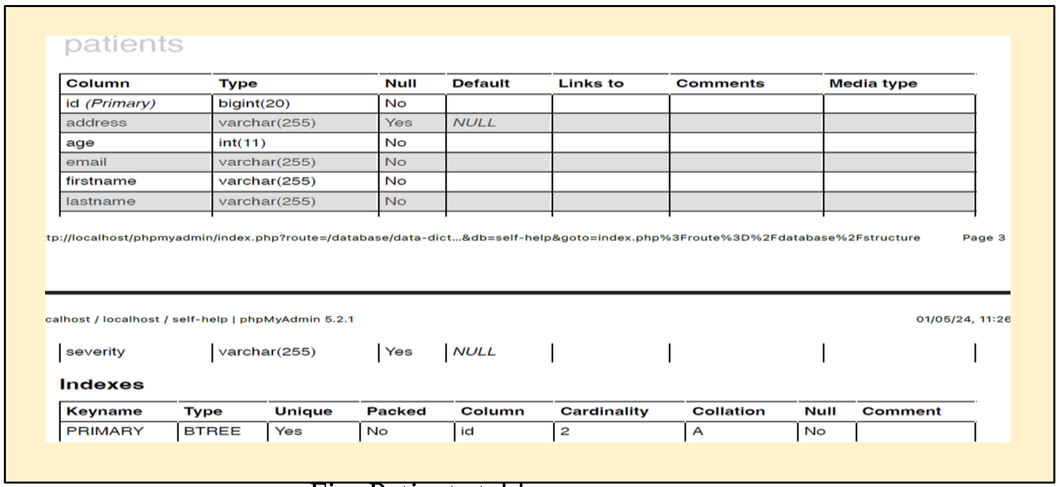
doctor_roles

Column	Type	Null	Default	Links to	Comments	Media type
doctor_id *(Primary)*	bigint(20)	No		doctors -> id		
role_id *(Primary)*	bigint(20)	No		roles -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	doctor_id	1	A	No	
				role_id	1	A	No	
FKj0is9oih9estxq6ja4hsybk74	BTREE	No	No	role_id	1	A	No	

Fig. doctor_roles



patients

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
address	varchar(255)	Yes	*NULL*			
age	int(11)	No				
email	varchar(255)	No				
firstname	varchar(255)	No				
lastname	varchar(255)	No				
severity	varchar(255)	Yes	*NULL*			

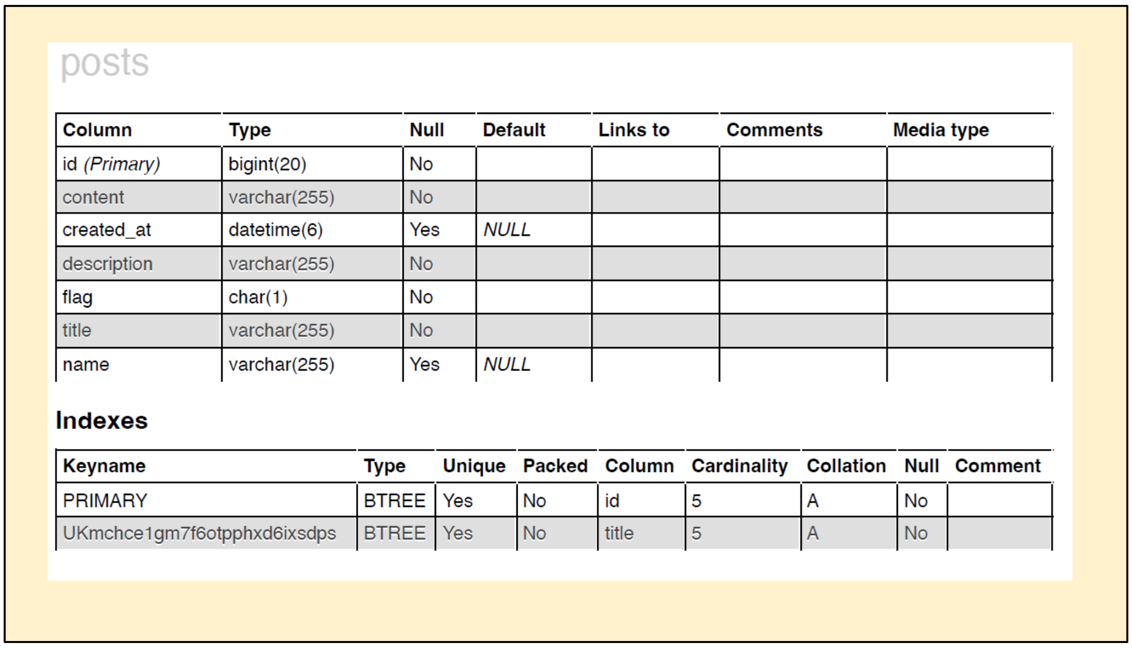
tp://localhost/phpmyadmin/index.php?route=/database/data-dict...&db=self-help&goto=index.php%3Froute%3D%2Fdatabase%2Fstructure Page 3

calhost / localhost / self-help | phpMyAdmin 5.2.1 01/05/24, 11:26

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	

Fig. Patients table



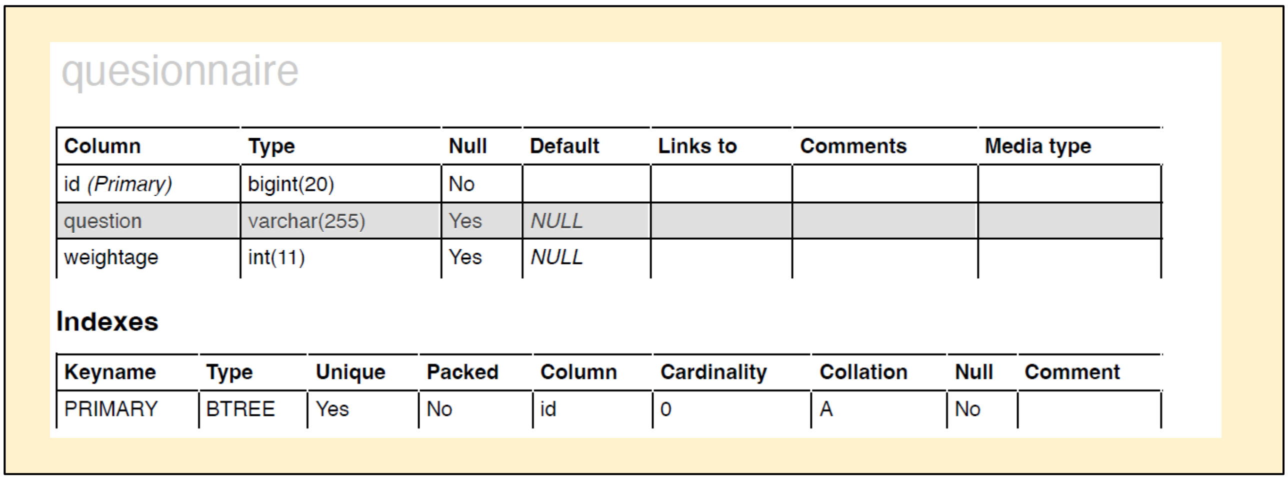
posts

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
content	varchar(255)	No				
created_at	datetime(6)	Yes	*NULL*			
description	varchar(255)	No				
flag	char(1)	No				
title	varchar(255)	No				
name	varchar(255)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	5	A	No	
UKmchce1gm7f6otpphxd6ixsdps	BTREE	Yes	No	title	5	A	No	

Fig.Posts table



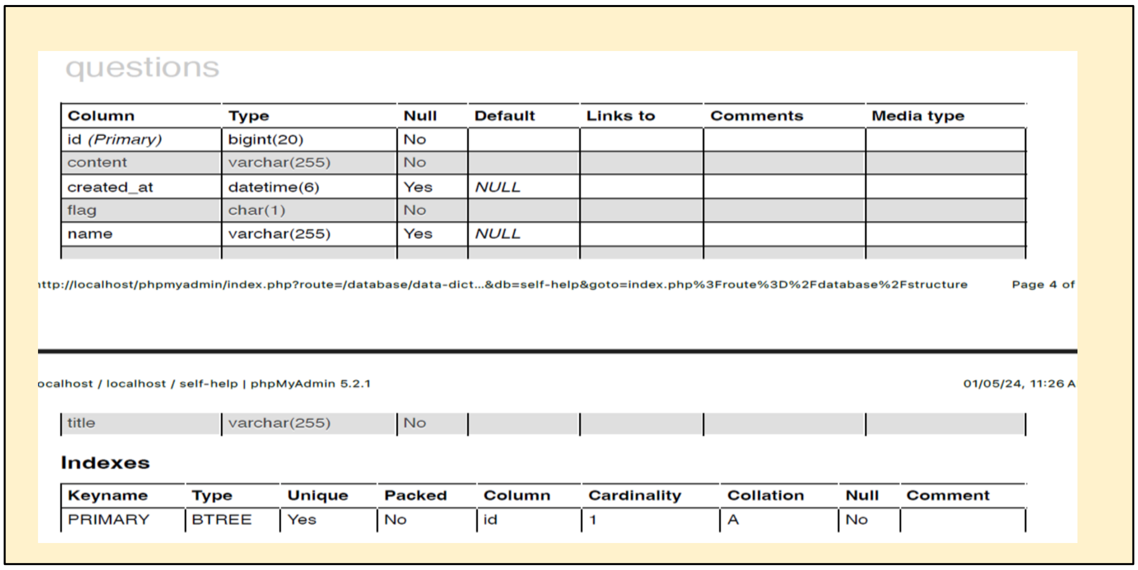
quesionnaire

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
question	varchar(255)	Yes	*NULL*			
weightage	int(11)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	

Fig. questionnaire table


questions

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
content	varchar(255)	No				
created_at	datetime(6)	Yes	*NULL*			
flag	char(1)	No				
name	varchar(255)	Yes	*NULL*			
title	varchar(255)	No				

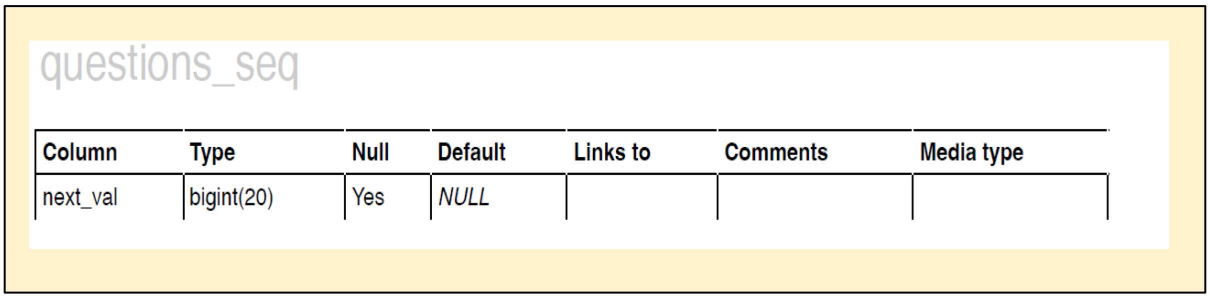
ıttp://localhost/phpmyadmin/index.php?route=/database/data-dict...&db=self-help&goto=index.php%3Froute%3D%2Fdatabase%2Fstructure Page 4 of

ocalhost / localhost / self-help | phpMyAdmin 5.2.1 01/05/24, 11:26 A

Indexes

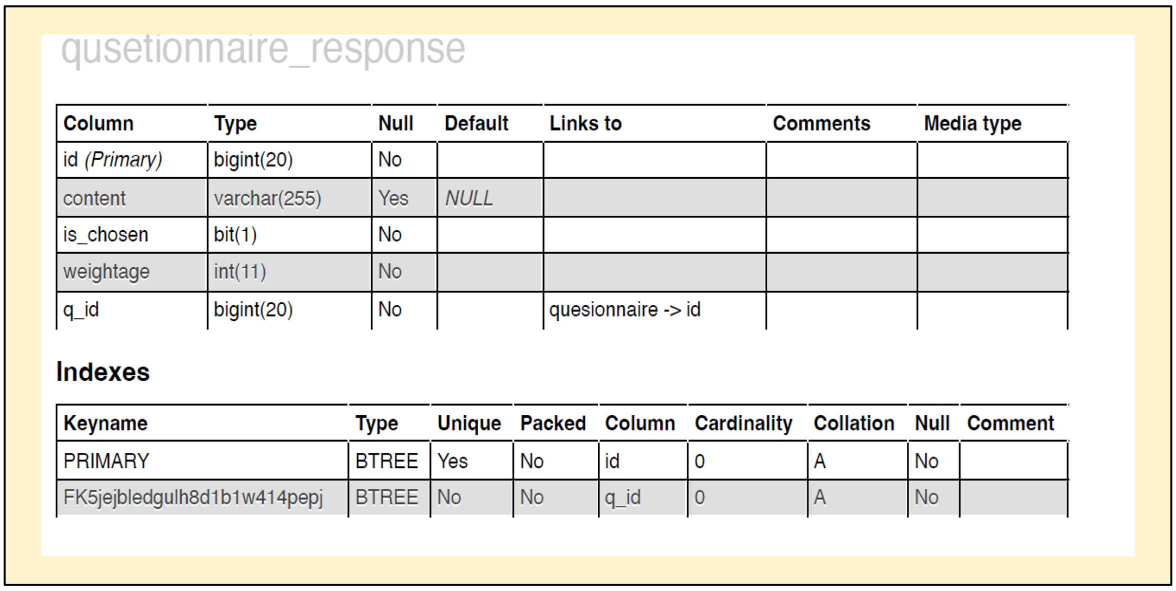
Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	1	A	No	

fig. questions table


questions_seq

Column	Type	Null	Default	Links to	Comments	Media type
next_val	bigint(20)	Yes	*NULL*			

fig. questions_seq

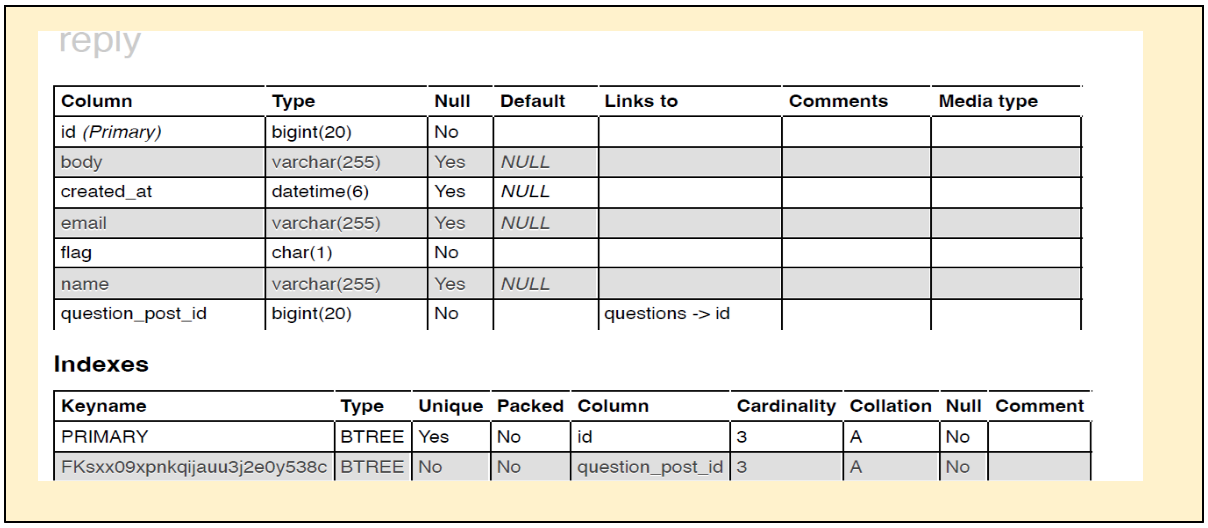

qusetionnaire_response

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
content	varchar(255)	Yes	*NULL*			
is_chosen	bit(1)	No				
weightage	int(11)	No				
q_id	bigint(20)	No		quesionnaire -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	0	A	No	
FK5jejbledgulh8d1b1w414pepj	BTREE	No	No	q_id	0	A	No	

fig.questionnaire_response table



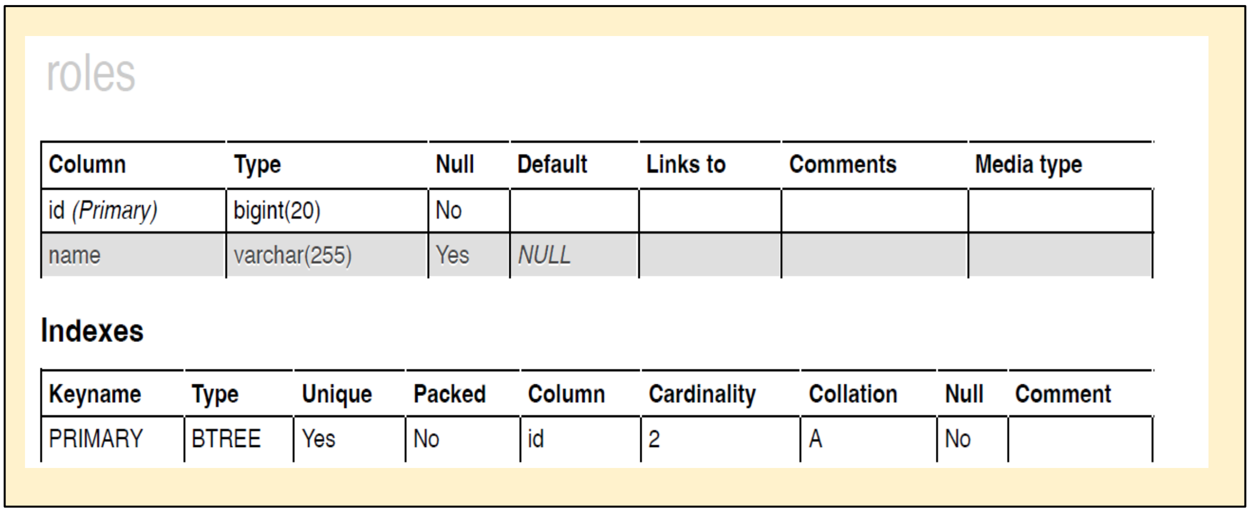
reply

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
body	varchar(255)	Yes	*NULL*			
created_at	datetime(6)	Yes	*NULL*			
email	varchar(255)	Yes	*NULL*			
flag	char(1)	No				
name	varchar(255)	Yes	*NULL*			
question_post_id	bigint(20)	No		questions -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	3	A	No	
FKsxx09xpnkqijauu3j2e0y538c	BTREE	No	No	question_post_id	3	A	No	

fig.Reply table



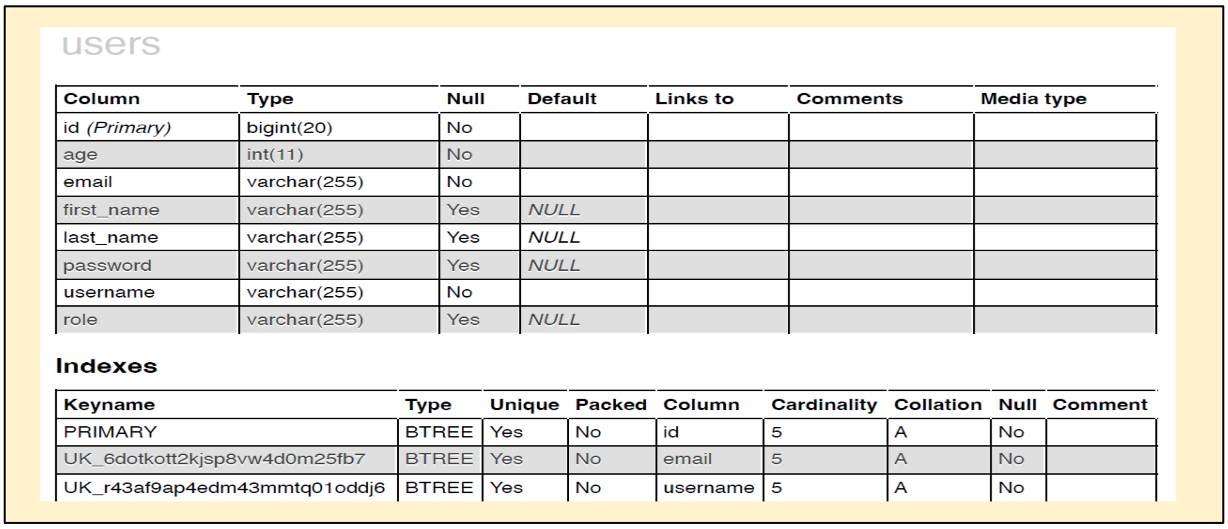
roles

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
name	varchar(255)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	2	A	No	

fig. roles table



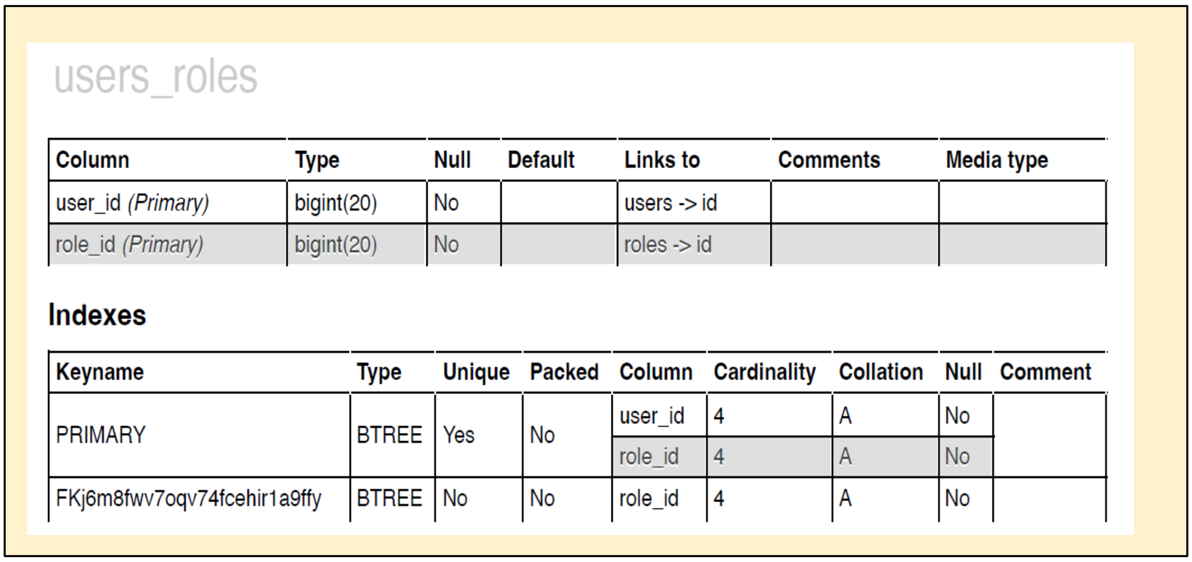
users

Column	Type	Null	Default	Links to	Comments	Media type
id *(Primary)*	bigint(20)	No				
age	int(11)	No				
email	varchar(255)	No				
first_name	varchar(255)	Yes	*NULL*			
last_name	varchar(255)	Yes	*NULL*			
password	varchar(255)	Yes	*NULL*			
username	varchar(255)	No				
role	varchar(255)	Yes	*NULL*			

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	id	5	A	No	
UK_6dotkott2kjsp8vw4d0m25fb7	BTREE	Yes	No	email	5	A	No	
UK_r43af9ap4edm43mmtq01oddj6	BTREE	Yes	No	username	5	A	No	

fig. Users table



users_roles

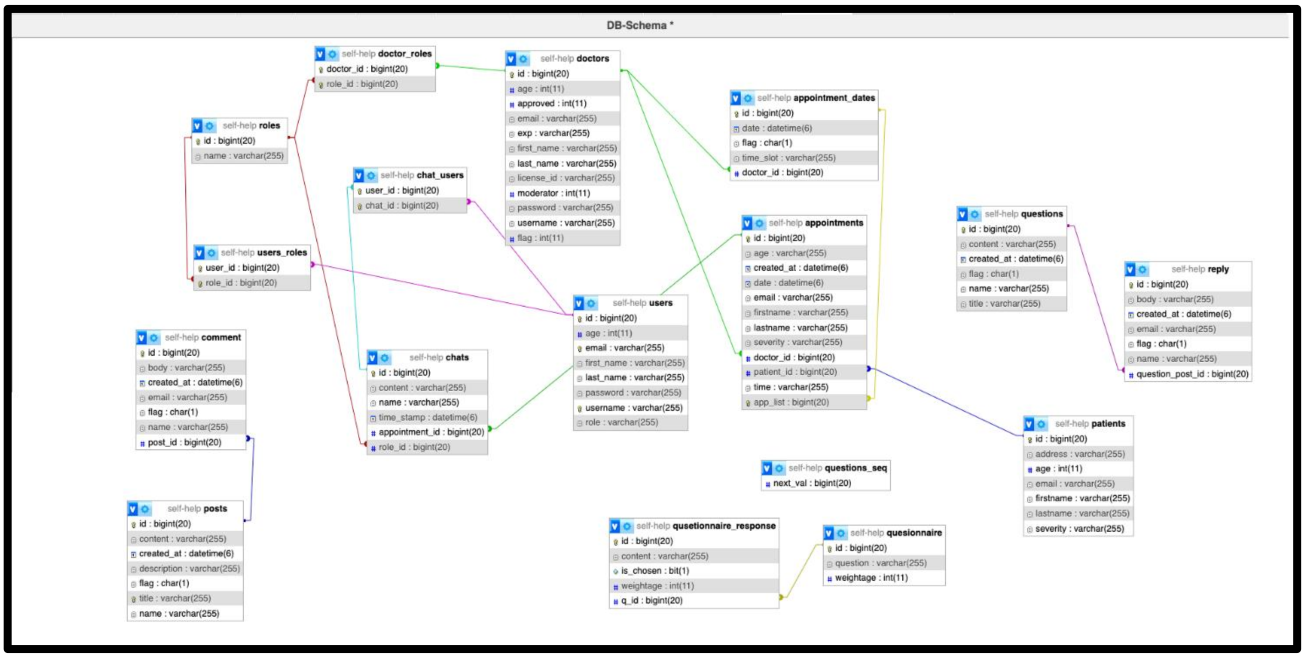
Column	Type	Null	Default	Links to	Comments	Media type
user_id *(Primary)*	bigint(20)	No		users -> id		
role_id *(Primary)*	bigint(20)	No		roles -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	user_id	4	A	No	
				role_id	4	A	No	
FKj6m8fwv7oqv74fcehir1a9ffy	BTREE	No	No	role_id	4	A	No	

fig. User_roles table

DATABASE DESIGN:



Security

<u>JWT :</u> **JWT (JSON Web Token)** is a popular method for implementing authentication in web applications. It is an open standard for securely transmitting information between parties as a JSON object.

The JWT consists of a header that specifies the algorithm used to sign the token, a payload that contains claims about an entity (usually the user) and additional data, and a signature that is generated by taking the header and payload, signing them with a secret key, and appending the resulting signature to the token.

To secure our endpoints(API) from unauthorised access, we use JWT. When a user logs in to our system, our server creates a token, which is then sent to the user. The user includes the JWT token in subsequent requests to the server, and the server verifies the authenticity of the JWT by checking its signature and the claims in the payload. This way, JWT protects our API from unauthorised access by unauthenticated users.

<u>Role-Based Authentication:</u>Role-based authentication is a security model that restricts access to resources based on the roles of individual users.

The project consists of four types of roles:
1. USER (Patient)
2. DOCTOR
3. ADMIN
4. MODERATOR

All these roles only have the privilege to access their respective API. Limiting access to resources based on user roles can minimize the risk of data breaches and other security threats.

<u>Server to Server security:</u> Our project implements server-to-server security through the use of a global token issued for each server-to-server communication. To establish communication between servers, authentication with the token is required. This ensures that only authorized servers can access specific resources or services and prevents unauthorized access or tampering through the encryption of the token.

Patient Verification JWT Token:We have created a token for **healOfy** server to access patient server apis.This token is also 7 days long and stored in **application.properties file** of the selfhelp server.

ADMIN Verification JWT Token:We have created tokens for selfhelp server to access every admin servers apis.These tokens are 7 day long and stored in selfhelp database.

DOCTOR AND MODERATOR Verification JWT Token: We have created tokens for selfhelp server to access every doctor servers apis.These tokens are 7 day long and stored in selfhelp database