



# Healthcare Application Development

## Project: Tele-Consulting Platform

Instructor: Prof. T K Srikanth

Teaching Assistant: Ashish Singhal

**Team: 23**

**Team Members:**


- |                     |                        |                         |
|---------------------|------------------------|-------------------------|
| 1. Akash Anand(009) | 2. Sachin singh(094)   | 3. Sudhanshu Kumar(119) |
| 4. Yash Tiwari(136) | 5. Dhruv Kharkwal(158) |                         |

## API's Implemented in Project

### Patient

The **Patient controller** has several endpoints that handle different requests.

1. **Post `"/join-queue/{patientId}/{specialization}"`**: This endpoint allows a patient to join the queue of a particular specialization. It takes in the `patientId` and `specialization` as path variables, and `roomId` as a query parameter, and returns a boolean value indicating whether the operation was successful or not.
2. **Post `"/left-queue/{patientId}"`**: This endpoint allows a patient to leave the queue. It takes in the `patientId` as a path variable and updates the status of the patient in the queue.
3. **Get `"/role"`**: This endpoint allows retrieving the role of a patient by their phone number. It takes in the `phoneNumber` as a query parameter and returns a string value indicating the role of the patient.
4. **Get `"/get-name/{patientId}"`**: This endpoint allows retrieving the name of a patient by their ID. It takes in the `patientId` as a path variable and returns the name of the patient as a string.
5. **Get `"/get-history-patient/{patientId}"`**: This endpoint allows retrieving the consultation history of a patient. It takes in the `patientId` as a path



variable and returns a list of ConsultationModel objects, which contain information about the consultation, such as the patient name, doctor name, date, and time.

**6. Get `"/get-patient/{patientId}"`:** A GET endpoint that returns the details of a specific patient identified by their patientId. The response contains a PatientModel object that includes the patient's name, age, gender, medical history, phone number, and status in the queue.

**7. Get `"/patient-list/phone-number"`:** A GET endpoint that takes a phone number as a parameter and returns a list of all patients associated with that phone number. The response includes a list of PatientModel objects, each containing the patient's name, age, gender, medical history, phone number, and status in the queue.

**8. Get `"/prescription/{prescriptionId}"`:** This endpoint allows the user to download a prescription in the form of a PDF file. The endpoint accepts a prescriptionId as a parameter and then uses this to search for the prescription in the database using the prescriptionService. If the prescription is found, the fileService is used to generate a PDF file of the prescription. The byte array of the PDF file is returned as the response body

**9. Post `"/appointment/{patientId}"`:** This endpoint is used to create a new appointment for a given patient. The patient ID is passed in the URL path, and the appointment details (date, symptoms, description, specialization) are passed in the request body as an AppointmentModel object. The endpoint returns the ID of the created appointment as a Long value.


**10. Get `"/prescription-list/{patientId}"`:** This endpoint is used to retrieve a list of prescriptions for a given patient. The patient ID is passed in the URL path. The endpoint returns a list of PrescriptionModel objects, which contain the prescription details (prescription ID, date, dosage, duration, medical finding, medicine name, and doctor name).

**11. Post `"/add-patient"`:** This endpoint is used to add a new patient to the system. The patient details (name, age, gender, medical history, phone number) are passed in the request body as a PatientModel object. The endpoint adds the patient to the system and returns the same PatientModel object in the response body.

## Doctor

The **Doctor controller** has several endpoints that handle different requests.

- 1. Post `"/add"`** - POST request to add a new doctor to the system.
- 2. Get `"/queue-size/{doctorId}"`** - GET request to retrieve the number of patients in the queue for a particular doctor based on specialization.
- 3. Get `"/role"`** - GET request to retrieve the role of a doctor given their phone number.
- 4. Patch `"/update-doctor/{id}"`** - PATCH request to update the attributes of a particular doctor.



**5. Post `"/consultation/{doctorId}"`** - POST request to start a consultation with the next patient in the queue for a particular doctor.

**6. Get `"/get-history/{phoneNumber}"`** - GET request to retrieve the consultation history of a doctor given their phone number.

**7. Get `"/get-appointment-details/{patientId}"`** - GET request to retrieve the details of the latest appointment of a patient.

**8. Get `"/view"`** - GET request to view all doctors in the system.

**9. Get `"/get-name/{doctorId}"`** - GET request to retrieve the name of a doctor given their doctorId.

**10. Get `"/doctor-by-contact/{phoneNumber}"`** - GET request to retrieve a DoctorModel object for a doctor given their phone number.

**13. Post `"/add/prescription/{patientId}/{doctorId}"`**: This endpoint is used to add a new prescription for a patient. The patient ID and doctor ID are provided as path parameters, and the details of the new prescription are provided in the request body as a PrescriptionModel object. The method returns a ResponseEntity<Boolean> object indicating whether the operation was successful.

**14. GET `/prescription-list/{doctorId}`**: This endpoint is used to retrieve a list of all prescriptions issued by a specific doctor. The doctor ID is provided as a path parameter, and the method returns a ResponseEntity<List<PrescriptionModel>> object containing a list of PrescriptionModel objects, each representing a prescription issued by the specified doctor.

## General APIs

**1. Post `/authenticate`:** This is a POST endpoint that handles user authentication. It takes a `JwtRequest` object as a request body, which contains the phone number entered by the user. The `authenticationManager` object then attempts to authenticate the user with the provided credentials. If the authentication is successful, a JSON Web Token (JWT) is generated using the `jwtUtility` object, which contains the user details and is returned in the `JwtResponse` object. If the authentication fails due to invalid credentials, a `BadCredentialsException` is thrown and an exception is returned with an `"INVALID_CREDENTIALS"` message.

**2. Post `/authenticate/add`** The purpose of this endpoint is to add a new patient to the system. The API expects a request body in the form of a `PatientModel` object, which contains information about the patient including their name, age, gender, medical history, and phone number.

At first, the API checks if any patients already exist in the system with the same phone number as the patient being added. If such patients exist, the API returns a bad request status.

If no patients exist with the same phone number, the API builds a new `Patient` object using the information provided in the `PatientModel` object from the request body. The role of the patient is set as `"ROLE_PATIENT"` and their `statusQueue` is set to `"NO"`.

## Various Entities in the Project

### 1. Patient

It contains:

{ phoneNumber, patientName, age, gender, medicalHistory, statusQueue, role, prescriptionList }

### 2. Doctor

It contains:

{ phoneNumber, doctorName, age, gender, emailId, Specialization, role, isAvailable, prescriptionList }

### 3. Appointment

It contains:

{ date, symptoms, description, specialization, patient }

### 4. Consultation

It contains: { date, time, patient, doctor }

### 5. Prescription

It contains:

{ date, medicalFinding, medicalName, dosage, duration, patient, doctor }

## Functionalities and Features of the Application

Patient sign ups into the application

Patient takes an appointment specifying the symptoms, description of the issue.

Patient selects the specialist(physician, cardiologist, etc) he/she wants to meet and proceed further.

He/she joins the room and waits till the doctor accepts the call.

The Doctor can see the number of patients waiting in the queue based on his/her specialization.


The functionality is implemented in such a way that whoever comes first, gets first consultation with the doctor.

When the doctor has patients in the queue he/she can accept the call and video call starts between the two parties.

Doctors can view the symptoms and description that the patient has entered before entering the room for prior knowledge.

The functionality is implemented in such a way that a doctor joins the call only when there are patients waiting in the queue, which





ensures that the time of the doctor is considered crucial and application focuses making it easy for the doctor to accept the call.

The details of consultation between the patient and doctor are recorded and stored in the database in the consultation table.

The doctor fills the prescription for the patient which is stored in the database in the prescription table. The patient can then download the prescription given to him by the respective doctor.

Patient can view all of his prescriptions given to him by the respective doctors in the future.

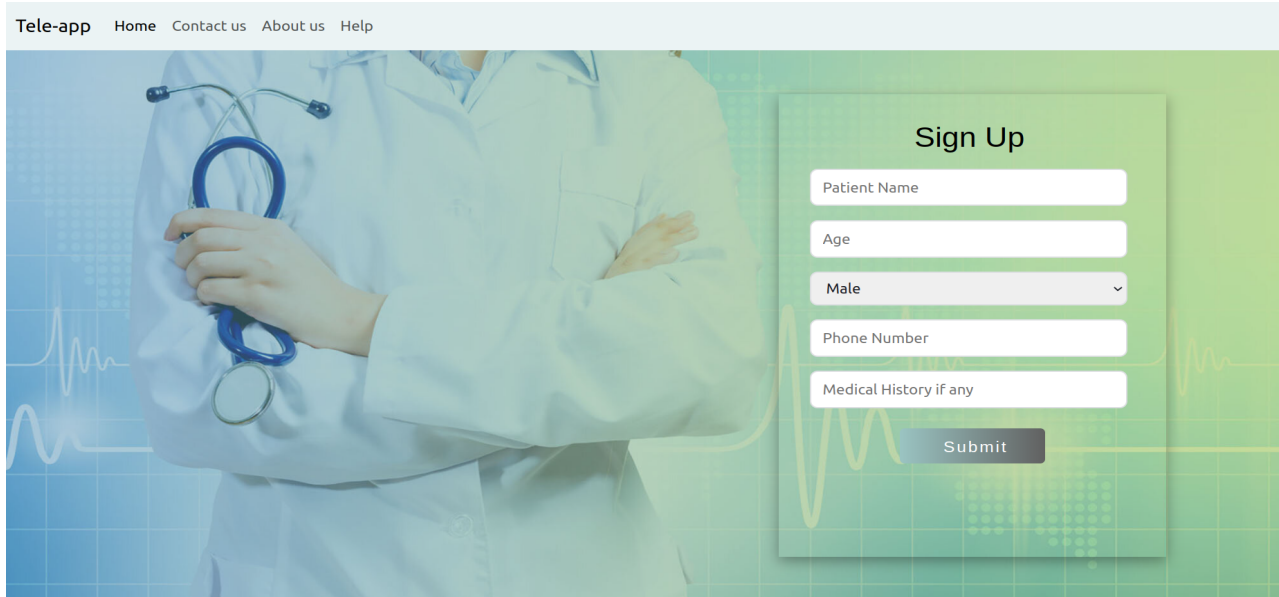
Doctor can view his/her consultations that he has done with all the patients and the prescriptions he has given.

Using a single number multiple patients can sign up as a family or a group of people may be using a single contact number this functionality is implemented by associating the multiple patients with a single contact number.

Individual patient can view their all consultations and download their prescriptions.

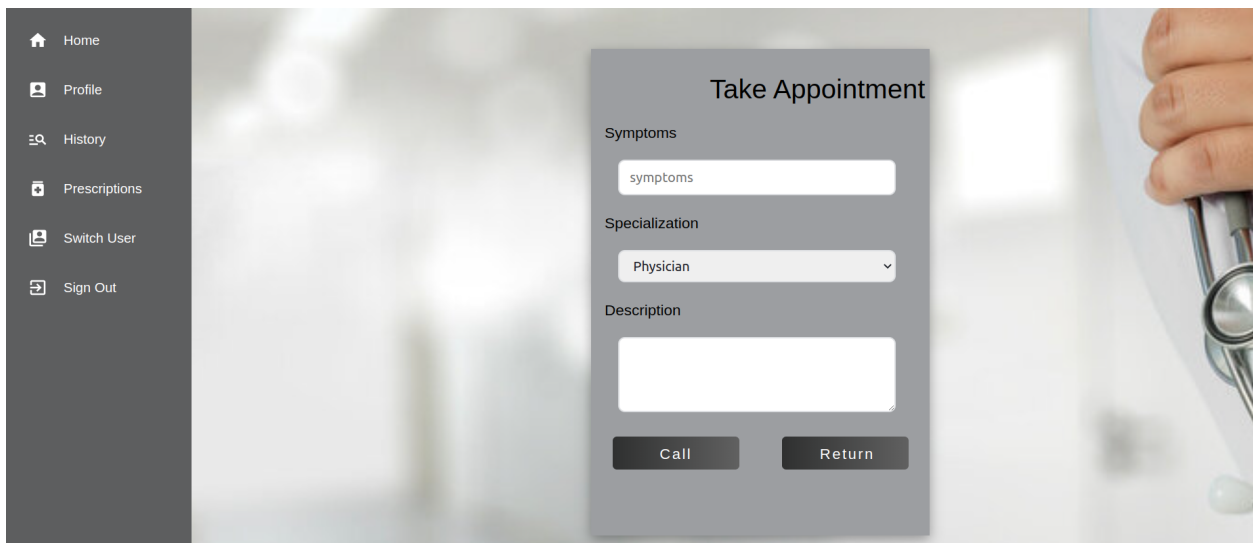
## Various Screenshots

### 1. Sign up page for the patient.



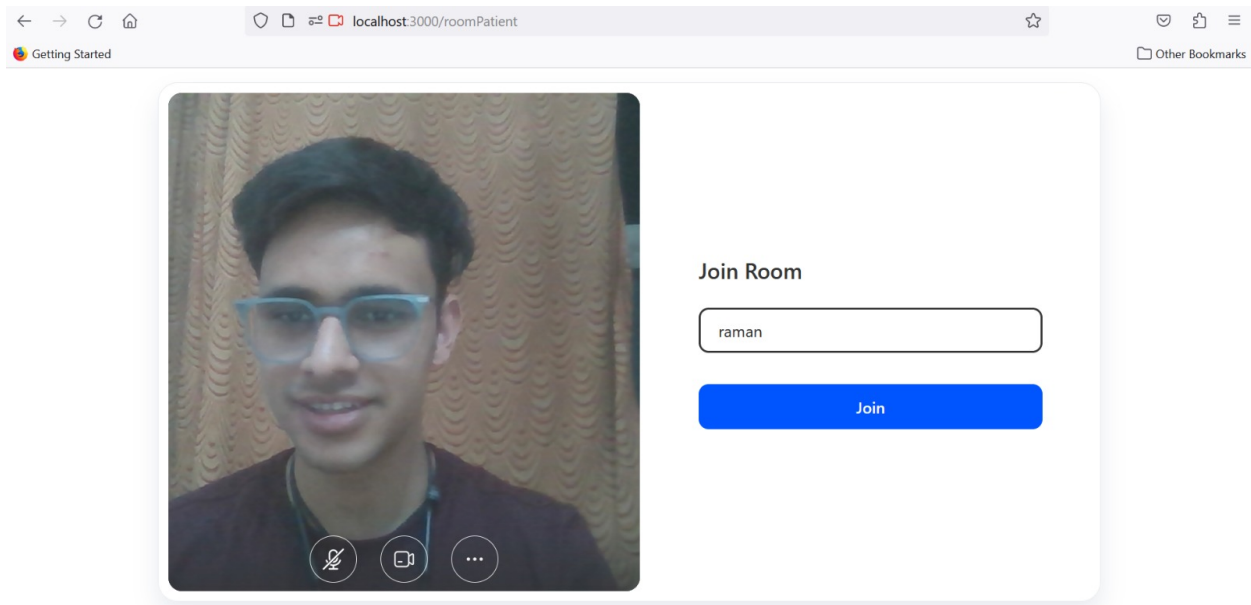
The screenshot shows a web application interface for a patient sign-up page. At the top, there is a navigation bar with links: "Tele-app", "Home", "Contact us", "About us", and "Help". The main content area features a background image of a doctor in a white coat with a stethoscope. Overlaid on this is a "Sign Up" form with the following fields: "Patient Name", "Age", "Male" (a dropdown menu with a downward arrow), "Phone Number", and "Medical History if any". A "Submit" button is located at the bottom right of the form.

### 2. Patient takes the appointment filling the symptoms, description and selecting the Specialist he/she wants to meet.

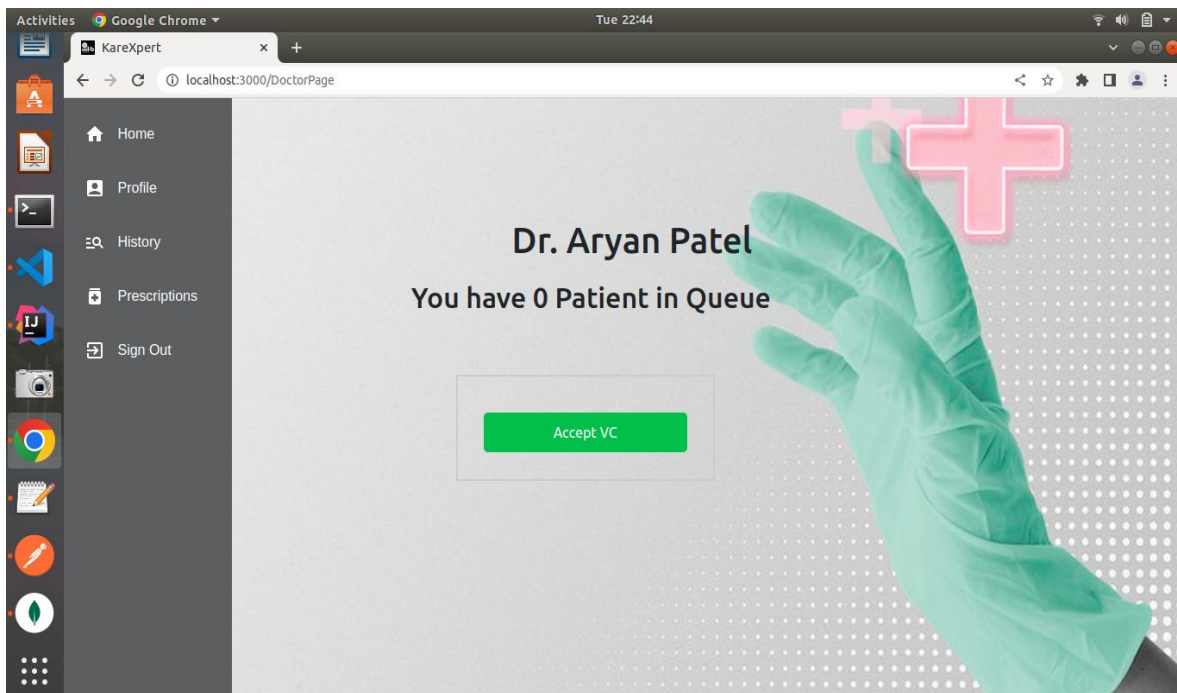


The screenshot shows a web application interface for a patient appointment page. On the left, there is a dark sidebar with navigation links: "Home", "Profile", "History", "Prescriptions", "Switch User", and "Sign Out". The main content area features a background image of a doctor's hand holding a stethoscope. Overlaid on this is a "Take Appointment" form with the following fields: "Symptoms" (a text input field containing the word "symptoms"), "Specialization" (a dropdown menu with "Physician" selected), and "Description" (a text input field). At the bottom of the form, there are two buttons: "Call" and "Return".

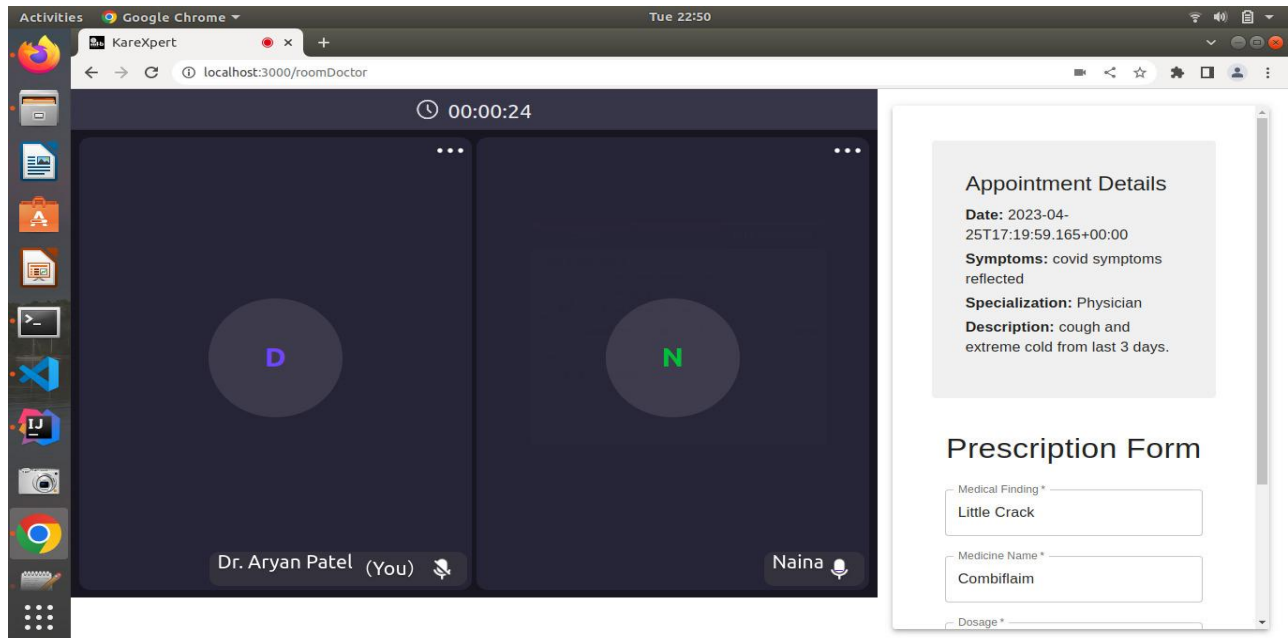
3. Patient join the call after taking appointment.



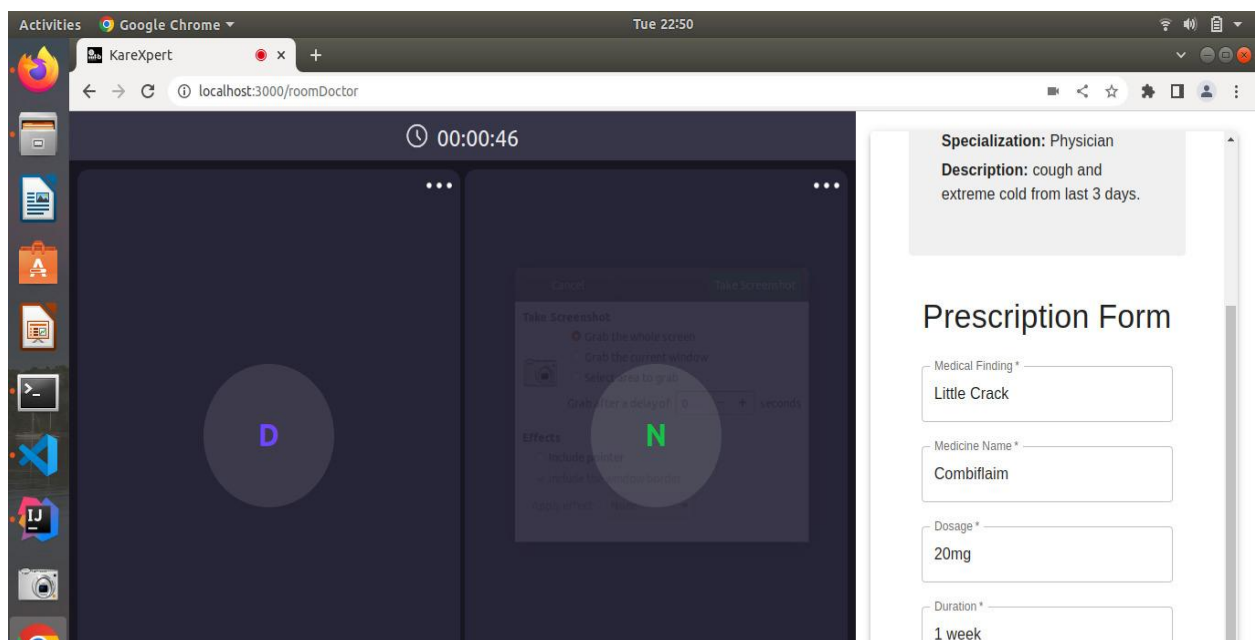
4 Doctor dashboard which displays the number of patients waiting in the queue.



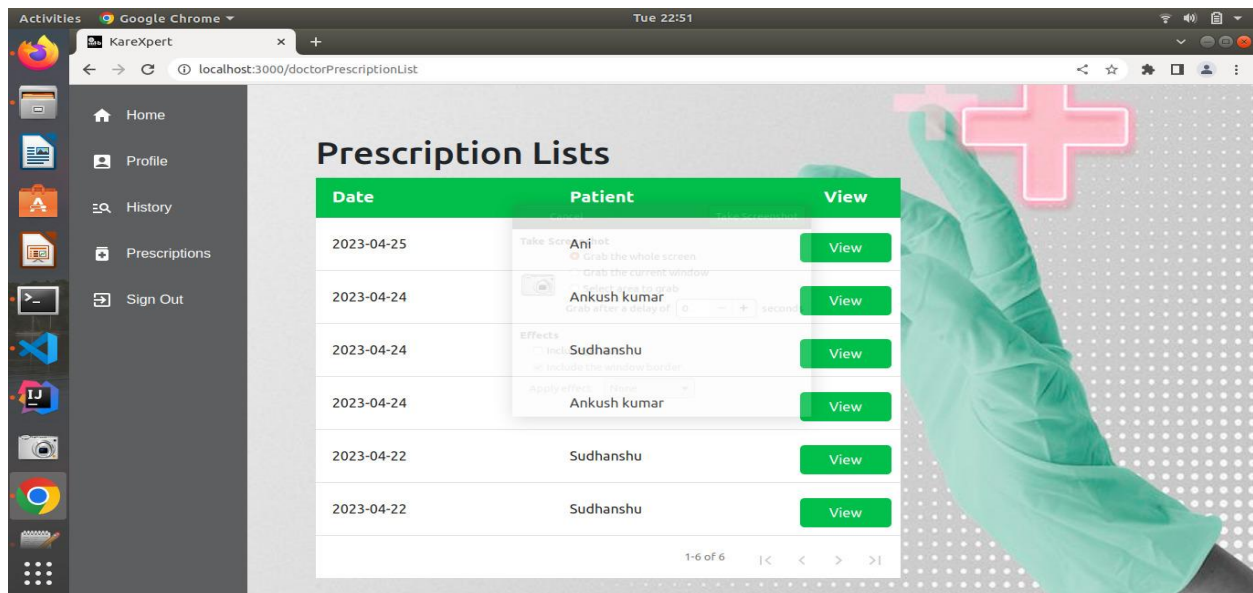
5. Doctor accepts the call when a patient join the call, the appointment details of the patient are displayed in the screen of doctor for reference.



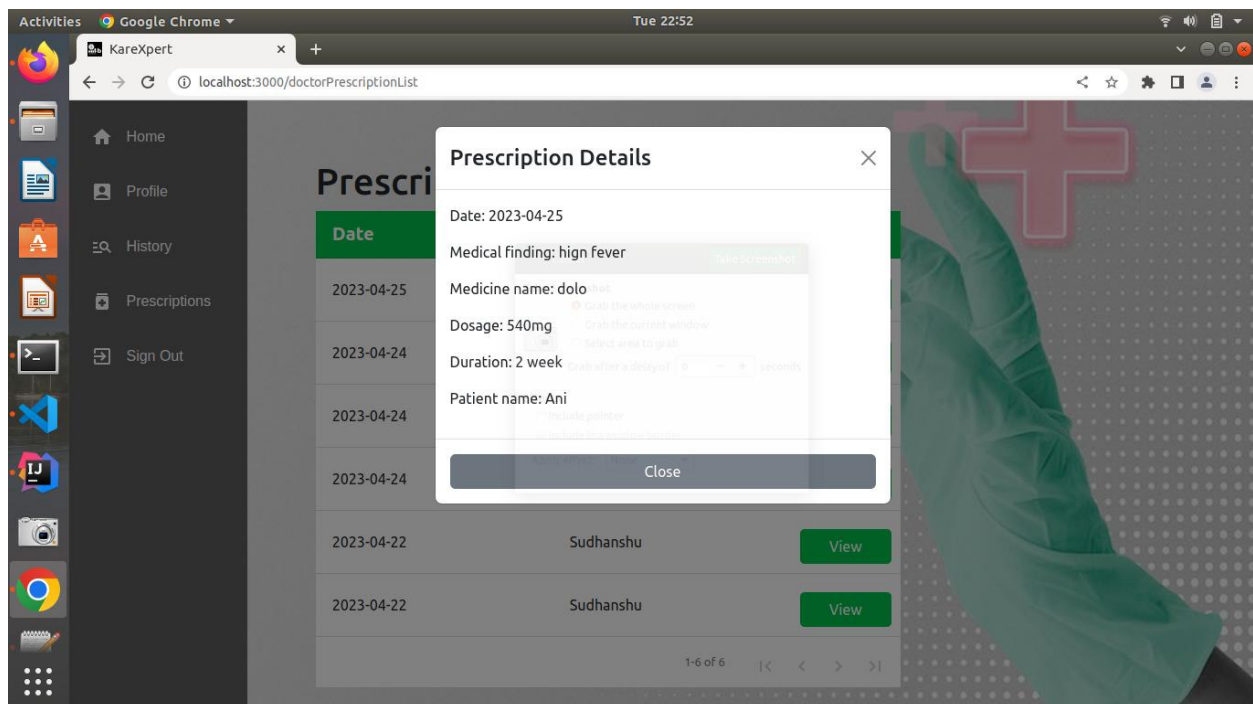
6. Prescription form that is filled by the doctor for the based on the consultation.



## 7. Doctor can view his last consultations and view prescription he has given



## 8. Prescription details doctor can see on his end.



## Contribution of team members:

### 1. Front-end designed and coded by:

Sudhanshu Kumar and Yash Tiwari

### 2. Backend design and coded by:

Akash Anand, Sachin Singh and Dhruv Kharkwal.

## Links

### Frontend Link

<https://github.com/Skchauhan07/KareXpert-iiitb>

### Backend Link

<https://github.com/akashanand842/tele-backend>

### Video Demo Link

[https://drive.google.com/file/d/1iTxjncjjLbwugln7WCkwth\\_sLNzp6Qow/view?usp=sharing](https://drive.google.com/file/d/1iTxjncjjLbwugln7WCkwth_sLNzp6Qow/view?usp=sharing)

