Final Project Report

CM3070 Final Project

Application Name:

Health Mate Application

Health Mate

Student Name:

Shudhanshu Gunjal

Table of Contents

Introduction

Health Mate: Telemedicine Service for Underserved Communities

A Non-Profit Web Application

Hello and welcome to my dream, my project – HealthMate, a concept born from a confluence of compassion and innovation, aimed at addressing the glaring disparities in healthcare access faced by underserved communities. This initiative is not merely a technological solution but a mission to redefine the landscape of healthcare accessibility. HealthMate stands as a testament to the power of digital platforms in bridging the vast divide between urban and rural healthcare services, offering a comprehensive, web-based application designed with the user's needs at the forefront.

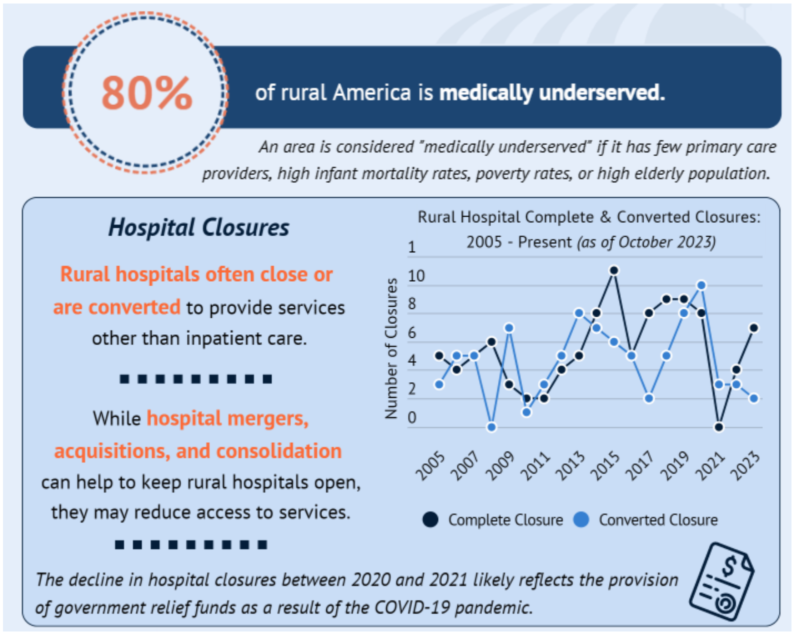
This non-profit initiative is built upon the foundational pillars of accessibility, data security, and collaborative healthcare. Through HealthMate, I aspire to dismantle the barriers that hinder individuals in remote areas from obtaining timely and effective medical advice and care. The platform enables users to effortlessly manage their medical records and connect with healthcare professionals in real-time, symbolizing a step forward in our collective journey toward equitable healthcare for all.

Why HealthMate? Understanding the Need for Change

The inception of HealthMate was propelled by the stark realities that characterize the current healthcare landscape, particularly in rural settings:

1. Exploring the Healthcare Crisis in Rural America:

- Research reveals that approximately **80%** of rural regions are medically underserved.
- The physician-to-population ratio in these areas starkly contrasts with urban centers, standing at **39.8 per 100,000** compared to **53.3 per 100,000**.
- The looming threat of rural hospital closures, with **25%** at risk, poses a significant challenge to healthcare access in these communities.
- The prevalence of premature death in rural areas exceeds that in urban areas by nearly **20%**, underscoring a critical need for intervention.
- Economic barriers further compound these issues, with **36%** of adults in rural locales forgoing necessary healthcare due to cost constraints.



80% of rural America is **medically underserved.**

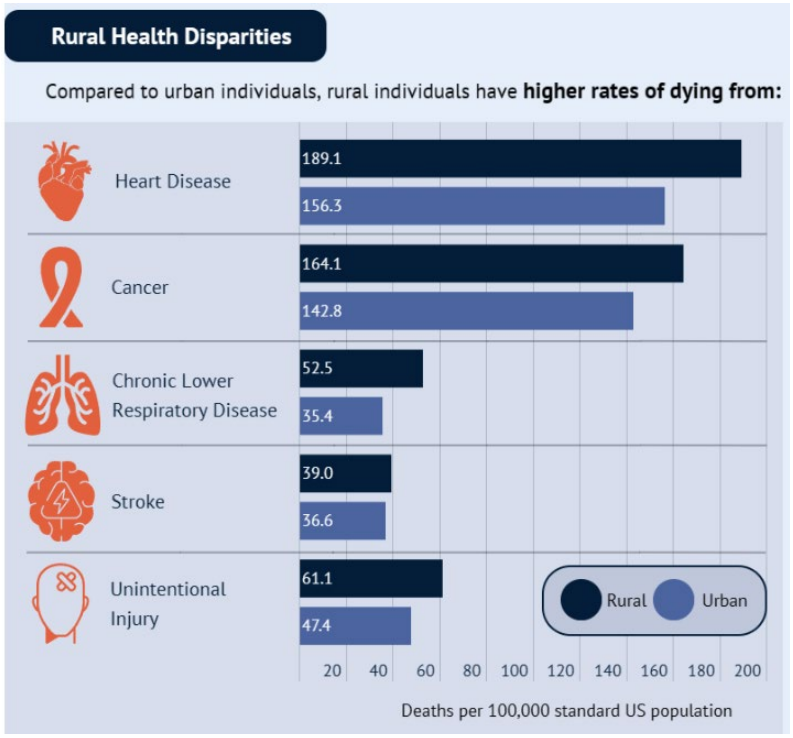
An area is considered "medically underserved" if it has few primary care providers, high infant mortality rates, poverty rates, or high elderly population.

Hospital Closures

Rural hospitals often close or are converted to provide services other than inpatient care.

While **hospital mergers, acquisitions, and consolidation** can help to keep rural hospitals open, they may reduce access to services.

The decline in hospital closures between 2020 and 2021 likely reflects the provision of government relief funds as a result of the COVID-19 pandemic.



Rural Health Disparities

Compared to urban individuals, rural individuals have **higher rates of dying from:**

2023. Rural Health: Addressing Barriers to care. *NIHCM*. Retrieved from https://nihcm.org/publications/rural-health-addressing-barriers-to-care

These statistics not only highlight the urgency of the situation but also fuel my commitment to spearheading a solution that can significantly mitigate these challenges.

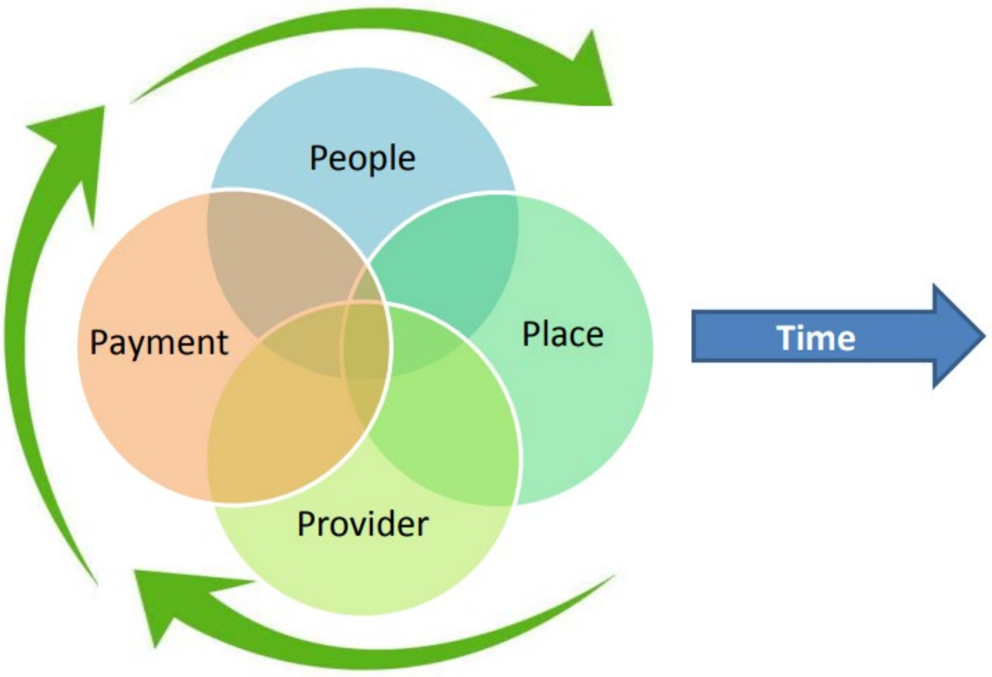
2) Addressing Barriers to Healthcare Access: My Mission with HealthMate:

The barriers encompassing healthcare access are multifaceted, ranging from structural and financial to personal and cultural. Each presents unique challenges that require innovative solutions.

Structural: Health care plan, shortage of health care providers, waiting time.

Financial: Uninsured, Underinsured, Absent coverage for certain conditions.

Personal and Cultural: Unable to travel, unable to communicate, disrespectful provider



2014. Access to Rural Health Care – A Literature Review and New Synthesis. Rural Policy Research Institute. Retrieved from https://rupri.org/wp-content/uploads/20140810-Access-to-Rural-Health-Care-A-Literature-Review-and-New-Synthesis.-RUPRI-Health-Panel.-August-2014-1.pdf

3) The Role of Telemedicine During and Beyond the Pandemic:

The COVID-19 pandemic highlighted the importance of telemedicine. It's a powerful tool for keeping communities safe while ensuring continuous care, a principle that's at the core of HealthMate.

Distinguishing HealthMate: Insights into the Healthcare Ecosystem

In the quest to develop HealthMate, it became imperative to critically analyze existing digital healthcare solutions. Platforms like the NHS in the UK and MyChart by Epic have set commendable precedents in the realm of digital healthcare. However, their limitations, such as geographical constraints and navigation complexities, highlight the need for a more inclusive and user-friendly solution. HealthMate is conceived with the vision of transcending these limitations, offering a platform that is universally accessible and intuitively designed.

Crafting HealthMate: Aspirations and Objectives

Empowering Through Innovation:

The design philosophy behind HealthMate emphasizes empowerment through innovation. The platform is engineered to be both robust and user-centric, enabling secure access to medical records, facilitating seamless appointment scheduling, and providing reliable real-time medical advice. This approach is reflective of my aspiration to deliver a solution that not only meets the immediate needs of users but also fosters long-term engagement and trust.

Guaranteeing Accessibility: A Commitment to Inclusivity:

Recognizing that health concerns are not confined to conventional schedules, HealthMate is designed to offer 24/7 accessibility. This commitment to inclusivity ensures that users can access vital health information and services at their convenience, embodying the principle that effective healthcare should be a right, not a privilege.

The Advantages of Joining the HealthMate Community

- **Facilitating Personalized Health Management:** HealthMate leverages advanced data analytics to offer personalized health insights, advocating for a proactive and preventive approach to health management. This not only enhances the user's ability to make informed decisions about their health but also contributes to the broader goal of improving public health outcomes.
- **Streamlining Healthcare Coordination:** By simplifying the healthcare management process, HealthMate significantly reduces the logistical burdens associated with healthcare access. This efficiency is vital for users in rural areas, where such challenges are most pronounced.
- **Enhancing Connectivity and Security:** The platform ensures enhanced connectivity with healthcare professionals, facilitated by a secure and confidential environment. This aspect of HealthMate is crucial in building trust and ensuring user privacy and data security.

In conclusion, HealthMate is more than just a digital platform; it is a vehicle for change in the realm of healthcare accessibility. By integrating advanced technology with a deep understanding of the needs of underserved communities, I aim to catalyze a shift towards more equitable healthcare provision. As we embark on this journey together, I invite you to join me in transforming the vision of HealthMate into a reality, paving the way for a future where quality healthcare is accessible to all, irrespective of their geographical or economic circumstances.

Literature Reviews

INTRODUCTION

Telemedicine is the use of information and communication technologies to deliver health care services remotely, such as diagnosis, consultation, treatment, and education. Telemedicine has the potential to improve access, quality, and efficiency of health care, especially for underserved populations who face barriers such as distance, cost, transportation, and availability of providers. However, telemedicine also poses challenges and limitations, such as technical, legal, ethical, and cultural issues, as well as the need for evidence-based practice and evaluation.

This project aims to develop a non-profit web application that provides telemedicine service for underserved communities, focusing on primary care and chronic disease management. The web application will allow users to connect with healthcare providers via video, audio, or text chat, and to access health information and resources.

This literature will review the previous work and academic literature related to telemedicine in primary care, with a focus on the following aspects:

- The benefits and challenges of telemedicine for underserved communities
- The design and implementation of telemedicine web applications
- The evaluation and assessment of telemedicine interventions

LITERATURE REVIEWS

Rural Health: Addressing Barriers to Care [1]

The article, titled "Rural Health: Addressing Barriers to Care", was published by the National Institute for Health Care Management (NIHCM) Foundation on October 25, 2023 [1]. It is an infographic that explores the state of rural health and healthcare access in the US, including the challenges and solutions to improve the health and well-being of rural Americans.

According to the article, rural Americans face significant barriers to health care, such as:

- High rate of underserved rural community: Rural Americans make up 20% of the US population, but 80% of rural America is medically underserved.
- High rates of uninsurance: 16.6% of rural residents are uninsured.
- Long travel distances: 20% of rural residents live long distances from the nearest hospital, and 60% of rural counties are classified as health professional shortage areas.
- Shortage of health care providers: There are only 39.8 physicians per 100,000 rural residents, compared to 53.3 per 100,000 urban residents.

These barriers contribute to poorer health outcomes and shorter life expectancies among rural Americans, who have higher rates of chronic conditions, such as obesity, diabetes, heart disease, and cancer, than urban Americans. Rural Americans also have higher rates of mortality from unintentional injuries, suicide, and opioid overdose.

The article also discusses how telemedicine, can help address some of these barriers and improve health outcomes for rural Americans. Some of the benefits of telehealth include:

- Telemedicine can increase access to primary care, specialty care, mental health care, and emergency care for rural patients.

- Telemedicine can reduce travel time and costs, improve patient satisfaction and engagement, and enhance care coordination and quality for rural patients.
- Telemedicine can support rural healthcare providers by expanding their reach, reducing their isolation, and providing them with education and training opportunities.

However, telemedicine also faces some challenges and limitations in rural areas, such as:

- Telemedicine requires adequate broadband infrastructure, equipment, and technical support, which may not be available or affordable in some rural areas.
- Telemedicine may encounter regulatory, reimbursement, and licensing barriers that vary by state and payer.
- Telemedicine may not be able to replace some aspects of in-person care, such as physical examinations, procedures, and laboratory tests.
- Telemedicine may raise some concerns about privacy, security, and quality of care, especially for vulnerable populations.

This article contributes to my project by providing an overview of the underserved communities' health situation, the barriers and benefits of telemedicine, and the policy recommendations and best practices to promote telemedicine. The article also points out the gaps in the current healthcare system for rural communities, such as the necessity to travel longer distances for medical care and a shortage of healthcare providers. My project aims to fill these gaps by providing accessible and quality healthcare services through a non-profit web application.

Telemedicine application in patients with chronic disease: a systematic review and meta-analysis [2]

The literature "Telemedicine application in patients with chronic disease: a systematic review and meta-analysis" published in BMC Medical Informatics and Decision-Making aims to review and analyze the effect of telemedicine on the management of chronic diseases such as hypertension, diabetes, and rheumatoid arthritis using a systematic review and meta-analysis.

- Telemedicine has been widely used for long-term care and self-management in patients with chronic disease.
- The results of the systematic review indicated that telemedicine consultation and telemonitoring are the most commonly used intervention methods.
- The results of the systematic review indicated that telemedicine consultation and telemonitoring are the most commonly used intervention methods. Telemedicine is helpful for improving self-management in patients with rheumatoid arthritis.
- The results of the meta-analysis showed patients' index of glycosylated hemoglobin (HbA1c) improved after 12 months of intervention (MD = -0.84; 95% CI = -1.53, -0.16;

Z = 2.42; P = 0.02), and no significant differences in fasting blood glucose (FBG) levels were observed after 6 months of intervention (MD = -0.35; 95% CI = -0.75, 0.06; Z = 1.69; P = 0.09). The results also showed that systolic blood pressure (MD = -6.71; 95% CI = -11.40, -2.02; Z = 2.81; P = 0.005) was reduced after 6 months of intervention The results also showed that systolic blood pressure was reduced after 6 months of intervention.

- Telemedicine had a positive effect on the management of diabetes, hypertension, and rheumatoid arthritis, especially when telemedicine consultation and telemonitoring method were used. When telemedicine was used as a disease management tool for patients with diabetes, the optimal intervention time is 12 months. Telemedicine improved the systolic blood pressure in hypertensive patients while also reducing negative emotions and enhancing medication adherence in rheumatoid arthritis patients.
- The authors reviewed several methods for assessing the effect of telemedicine on the management of chronic diseases such as hypertension, diabetes, and rheumatoid arthritis. They found that telemedicine consultation and telemonitoring are the most commonly used intervention methods. Telemedicine had a positive effect on the management of diabetes, hypertension, and rheumatoid arthritis, especially when telemedicine consultation and telemonitoring method were used.

The authors reviewed several methods for assessing the effect of telemedicine on the management of chronic diseases such as hypertension, diabetes, and rheumatoid arthritis. They found that telemedicine consultation and telemonitoring are the most commonly used intervention methods. Telemedicine had a positive effect on the management of diabetes, hypertension, and rheumatoid arthritis, especially when telemedicine consultation and telemonitoring method were used. The authors concluded that telemedicine is helpful for improving self-management in patients with rheumatoid arthritis and that it can be used as a disease management tool for patients with diabetes.

In conclusion, telemedicine is a promising tool for managing chronic diseases such as hypertension, diabetes, and rheumatoid arthritis. Telemedicine consultation and telemonitoring are the most commonly used intervention methods, and they have a positive effect on the management of these chronic diseases. Telemedicine has the potential to improve self-management in patients with rheumatoid arthritis and enhance medication adherence while reducing negative emotions. The findings of this study suggest that telemedicine can be used as a disease management tool for patients with chronic diseases.

RESEARCH STUDIES

The Empirical Foundations of Telemedicine Interventions in Primary Care [3]

The article “The Empirical Foundations of Telemedicine Interventions in Primary Care” by Rashid L Bashshur, Joel D Howell, Elizabeth A Krupinski, Kathryn M Harms, Noura Bashshur and Charles R Doarn is a systematic review of the scientific evidence for the merits of telemedicine interventions in primary care. The article examines the feasibility, acceptance, outcomes, and cost of telemedicine in primary care, based on 86 studies published from 2005 to 2015. The review found that telemedicine improved access to care, especially for rural and remote areas, and for specialty care, such as mental health, dermatology, and cardiology. The review also found that telemedicine improved the quality of care, as measured by clinical outcomes, patient satisfaction, and adherence.

SIMILAR PROJECTS

Doxy.me [4]

This is a web-based platform that allows healthcare providers to communicate with patients via text, audio, or video, without requiring any software installation or registration. Doxy.me is HIPAA, GDPR, and SOC2 compliant, and offers free, professional, and clinic plans. Doxy.me claims to be the world's most loved telemedicine solution, with over one million providers and 8 billion minutes of telemedicine delivered. Doxy.me is similar to my project in terms of using a web-based platform, but differs in terms of targeting a broader market, not specifically focusing on underserved communities, it does not include medical records and charging fees for some plans and features.

NHS [5]

This is the publicly funded health care system in the United Kingdom, which provides a comprehensive range of health services, free at the point of delivery, for people ordinarily resident in the UK. NHS also offers telemedicine services, such as online consultations, video calls, and audio calls, to improve access, convenience, and efficiency of care. NHS has been using telemedicine to cope with the COVID-19 pandemic, as well as to address the needs of rural and remote areas, chronic conditions, and mental health. NHS is similar to my project in terms of providing telemedicine services for underserved communities but differs in the targeting market it is specifically designed for United Kingdom residents, not for other countries.

Mend [6]

This is another web-based platform that enables healthcare providers to offer telemedicine and digital patient intake services. Mend claims to have the highest connection rate, patient satisfaction, and no-show reduction in the industry. Mend also integrates with various electronic health records (EHRs), scheduling systems, and payment gateways. Mend offers different pricing plans based on the number of providers and features. Mend is similar to my project in terms of using a web-based platform and integrating with other systems, but differs in terms of targeting a broader market, not specifically focusing on underserved communities, and charging fees for all plans and features.

MyChart [7]

This is a secure, online health management tool that connects patients to portions of their personal medical records and insurance information. MyChart is used by various healthcare organizations in the US and abroad and allows patients to communicate with their care team, review test results, medications, and immunization history, schedule and manage appointments, get price estimates, view and pay bills, securely share medical records, and connect with other health care providers. MyChart is similar to my project in terms of providing online health management and communication but differs in terms of being a patient portal, not a telemedicine service and requiring patients to have an account with a participating healthcare organization.

TECHNIQUES AND METHODS

I plan to use the following techniques and methods to achieve our goals:

- Web development: I will use Next.js a React.js based framework that will create the front end of my web application which will be an interface for the users (patients and providers) to interact with telemedicine service.
- WebRTC: I will use WebRTC, a web technology that enables real-time communication between browsers, to implement the video and audio chat functionality of my web application. WebRTC allows peer-to-peer connection between users, without requiring any plugins or third-party servers. WebRTC also supports encryption, authentication, and data transfer, which are essential for ensuring the security and privacy of the telemedicine service.
- Django: I will use Django for the backend as it is a feature rich framework including authentication, admin panel, ORMs, scalable, secure, and we can add new features if needed.
- Database: I will use Postgres as a database and Redis for Realtime chat which is in a memory data store that offers extremely fast read and write operations.
- Server: I will use the Hetzner server with the Ubuntu operating system and Docker to build, test, and deploy the application.

Project Design

INTRODUCTION

This project design outlines the development of a non-profit web application dedicated to providing telemedicine services to underserved communities. The design is informed by an extensive literature review and analysis of existing telemedicine solutions, with a focus on addressing the unique needs of rural and remote populations who face significant barriers to healthcare access.

OVERVIEW OF THE PROJECT

The primary objective of this project is to create an accessible, user-friendly telemedicine platform tailored for underserved communities. This web application aims to bridge the gap in healthcare access due to challenges such as distance, cost, and availability of healthcare providers. By leveraging digital technologies, the platform will facilitate remote health care services including diagnosis, consultation, treatment, and education through various modes of communication like video, audio, and text chat.

Project Template

The chosen template for this project is a **Non-Profit Web Application.**

Domain and Users

- **Domain:** The domain of this project is telemedicine, with a specific focus on primary care and chronic disease management.
- **Users:** The primary target users are individuals in underserved communities, particularly in rural areas, who face significant barriers to accessing healthcare. This includes people dealing with issues like long travel distances to healthcare facilities, high costs of medical care, and a lack of available healthcare providers. Secondary users include healthcare professionals who will utilize the platform to offer remote consultations and services to these communities.

DESIGN JUSTIFICATION

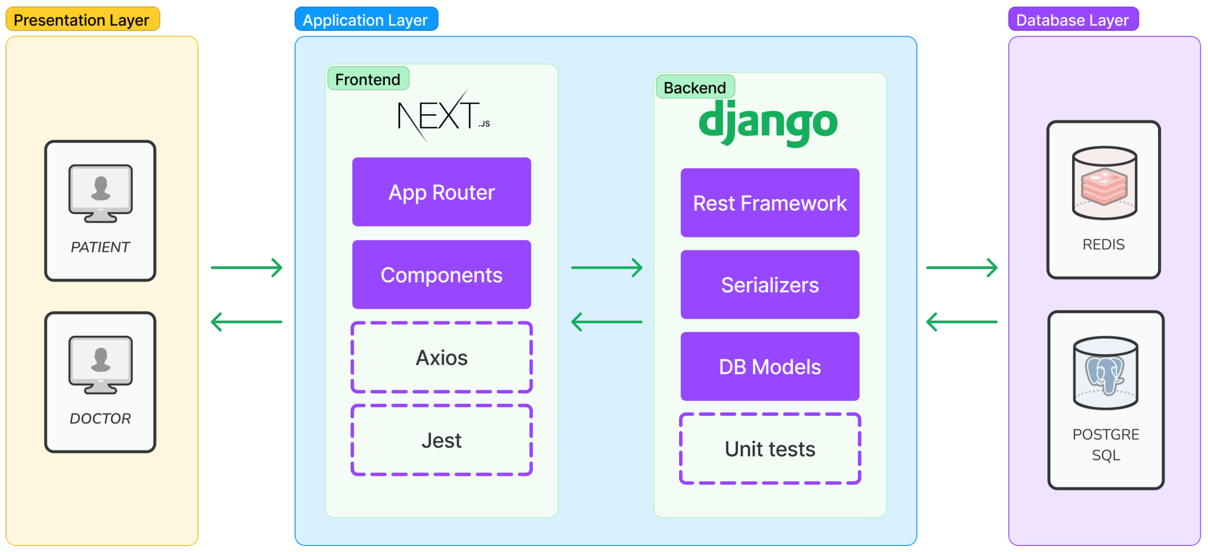
The design choices for this project are driven by the specific needs of its users and the requirements of the domain.

- **Accessibility and Simplicity:** Recognizing that many users may have limited technical expertise or access to high-end devices, the interface is designed to be straightforward and intuitive. This ensures that the application is accessible to a wider audience, including those with minimal experience in using digital platforms.
- **Privacy and Security:** Given the sensitive nature of health-related data, the application adheres to stringent privacy and security standards such as HIPAA compliance. This is crucial for building trust with the users and ensuring that their personal and medical information is securely handled.
- **Real-time Communication:** The implementation of WebRTC technology facilitates reliable and real-time communication between patients and providers. This feature is essential for delivering effective remote consultations, diagnoses, and treatment plans.
- **Resource Center:** A dedicated section within the application provides users with access to a wealth of health information and resources. This feature is designed to educate users about various health conditions, treatments, and preventive care measures, thereby empowering them with knowledge to make informed health decisions.

STRUCTURE OF THE PROJECT

The project is structured into several key components to ensure a comprehensive telemedicine solution:

- **Web Pages:** The application will consist of multiple web pages, including a user dashboard for patients, a provider portal for healthcare professionals, a scheduling system for appointments, and a resource center for health information.
- **API:** The application programming interface (API) will handle the exchange of data between the front-end user interface and the back-end server, ensuring seamless interaction and functionality.
- **Database Models:** The application will utilize PostgreSQL for storing structured data such as user profiles, appointment details, and medical records. Additionally, Redis will be employed to manage real-time chat sessions, offering fast read and write operations essential for instant messaging features.
- **Architecture:** The front-end of the application will be developed using Next.js, a React.js based framework that provides a robust environment for building interactive user interfaces. The back-end will be powered by Django, a high-level Python web framework that ensures rapid development and clean, pragmatic design. For real-time communication, WebRTC technology will be implemented to allow peer-to-peer connections between users for video and audio chats.



TECHNOLOGIES AND METHODS

- **Front-End Development:** Next.js is selected for its efficiency in rendering dynamic web pages and its scalability, making it ideal for handling high user traffic typical in telemedicine applications.
- **Back-End Development:** Django is chosen for its ability to handle complex data models and its strong security features, which are critical for protecting patient data.
- **Real-Time Communication:** WebRTC is the keystone technology for enabling live audio and video communication, crucial for real-time consultations between patients and healthcare providers.
- **Database Management:** PostgreSQL will manage structured data storage, ensuring data integrity and security. Redis will be used for its high performance in managing real-time chat data.
- **Server and Deployment:** The application will be hosted on a Hetzner server utilizing Ubuntu as the operating system. Docker will be employed for building, testing, and deploying the application, ensuring a consistent environment across development, staging, and production stages.

WORK PLAN AND GANTT CHART

A detailed Gantt chart outlines the project timeline, spanning from the 1th to the 21th week. The plan is structured into distinct phases:

Weeks 1-5 (Discovery Phase):

Initial phase focusing on understanding project requirements and selecting appropriate templates. This includes pitching the project idea and preparing both the project proposal and literature review.

Weeks 6-10 (Design Phase):

Concentrated on planning and designing the project, preparing a prototype design for the app, and developing key functions. This phase concludes with preparing a preliminary report.

Weeks 11-15 (Development Phase Part 1):

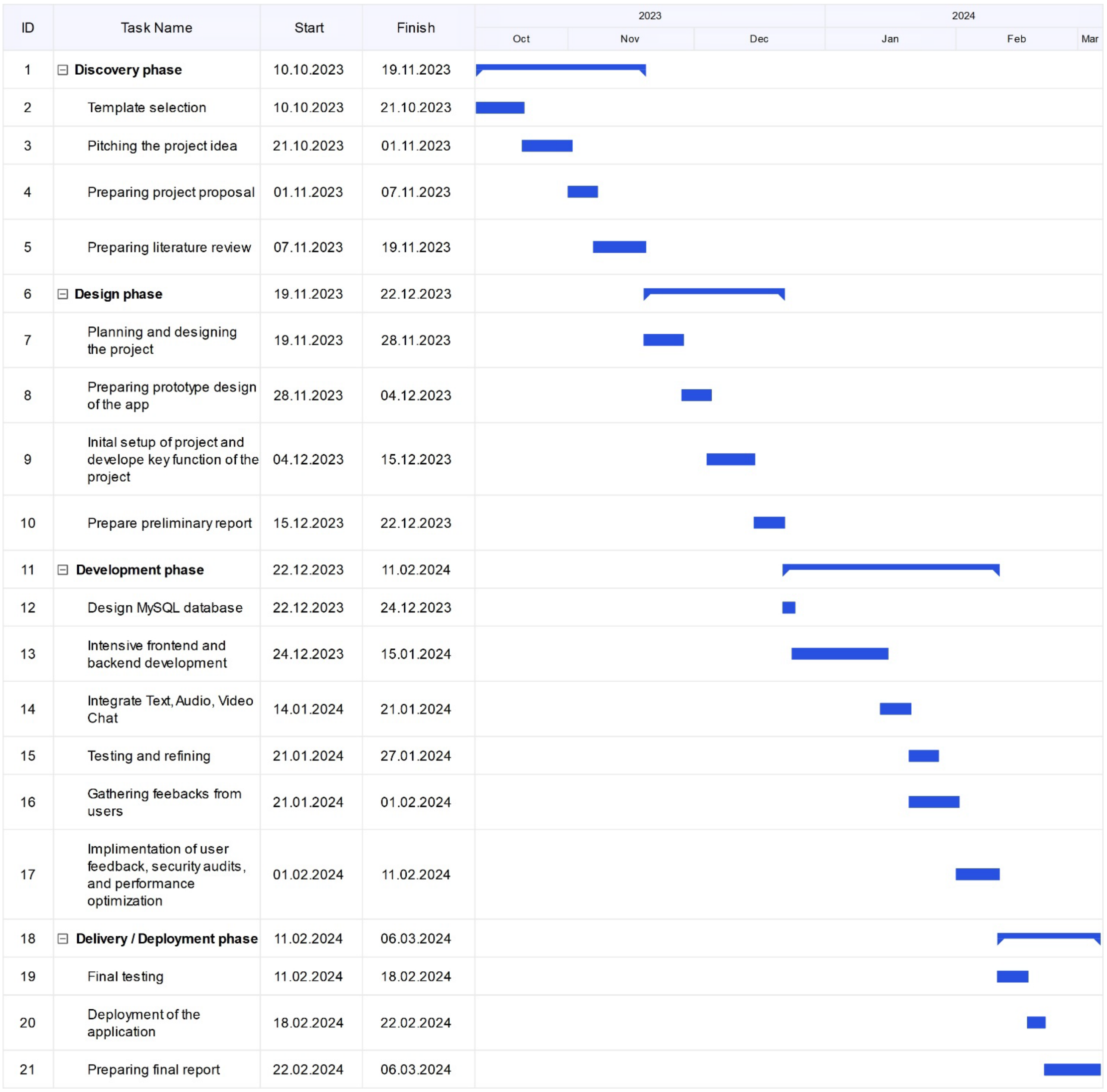
Begins with designing the PostgreSQL database and intensive front-end and back-end development. It includes integrating text, audio, and video chat features, followed by testing and refining these components.

Weeks 16-18 (Development Phase Part 2):

Focused on gathering feedback from users, implementing feedback, conducting security audits, and optimizing performance to enhance the overall project.

Weeks 19-21 (Delivery/Deployment Phase):

Final testing of the application, deploying it, and preparing the final report. This phase ensures the project is ready for launch and meets all the predefined objectives and quality standards.



ID	Task Name	Start	Finish
1	⊟ **Discovery phase**	10.10.2023	19.11.2023
2	Template selection	10.10.2023	21.10.2023
3	Pitching the project idea	21.10.2023	01.11.2023
4	Preparing project proposal	01.11.2023	07.11.2023
5	Preparing literature review	07.11.2023	19.11.2023
6	⊟ **Design phase**	19.11.2023	22.12.2023
7	Planning and designing the project	19.11.2023	28.11.2023
8	Preparing prototype design of the app	28.11.2023	04.12.2023
9	Inital setup of project and develope key function of the project	04.12.2023	15.12.2023
10	Prepare preliminary report	15.12.2023	22.12.2023
11	⊟ **Development phase**	22.12.2023	11.02.2024
12	Design MySQL database	22.12.2023	24.12.2023
13	Intensive frontend and backend development	24.12.2023	15.01.2024
14	Integrate Text, Audio, Video Chat	14.01.2024	21.01.2024
15	Testing and refining	21.01.2024	27.01.2024
16	Gathering feebacks from users	21.01.2024	01.02.2024
17	Implimentation of user feedback, security audits, and performance optimization	01.02.2024	11.02.2024
18	⊟ **Delivery / Deployment phase**	11.02.2024	06.03.2024
19	Final testing	11.02.2024	18.02.2024
20	Deployment of the application	18.02.2024	22.02.2024
21	Preparing final report	22.02.2024	06.03.2024

IMPLEMENTATION

The HealthMate project, a comprehensive health management system, integrates various functionalities to provide a seamless experience for managing health appointments, medical records, and real-time communications. This section delineates the technical implementation of the project, focusing on the key components that underpin its operation: authentication mechanisms, video calling features, database structuring with serializers, Redis configuration, state management using Zustand, overall methodology, conversation consumers, appointment scheduling, and medical records management.

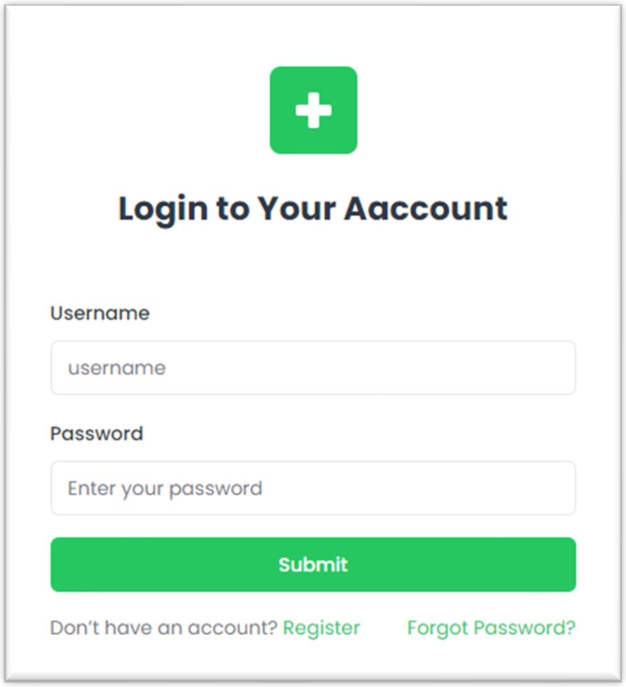
1) Authentication Mechanism

Authentication serves as the gateway to the HealthMate system, ensuring that access is securely managed and user identities are accurately maintained. The project employs a robust authentication system built on Django's built-in authentication framework for the backend, complemented by a React-based frontend that facilitates user interaction through a friendly UI.

2) User Registration and Login

User registration is the initial step in accessing the HealthMate services, requiring users to provide essential details such as username, email, and password. This process is handled by a dedicated registration view in Django, which validates the input data and creates a new user record in the system's database. The corresponding React component, RegisterForm, captures the user input through a form interface and communicates with the backend via an API call to register the user.

Upon successful registration, users proceed to login, a process managed by the LoginForm component in the frontend. This component collects the user's credentials and submits them to the backend, where Django's authentication system verifies the credentials against the stored user data. Upon successful authentication, the system grants access to the user, establishing a session that persists across the application.



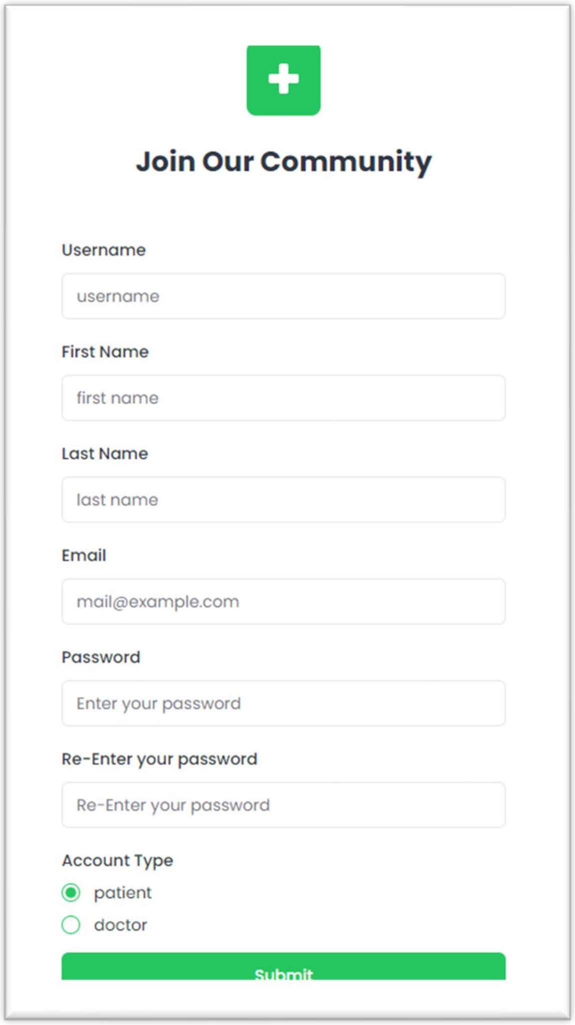
Login to Your Aaccount

Username

username

Password

Enter your password



Join Our Community

Username

username

First Name

first name

Last Name

last name

Email

mail@example.com

Password

Enter your password

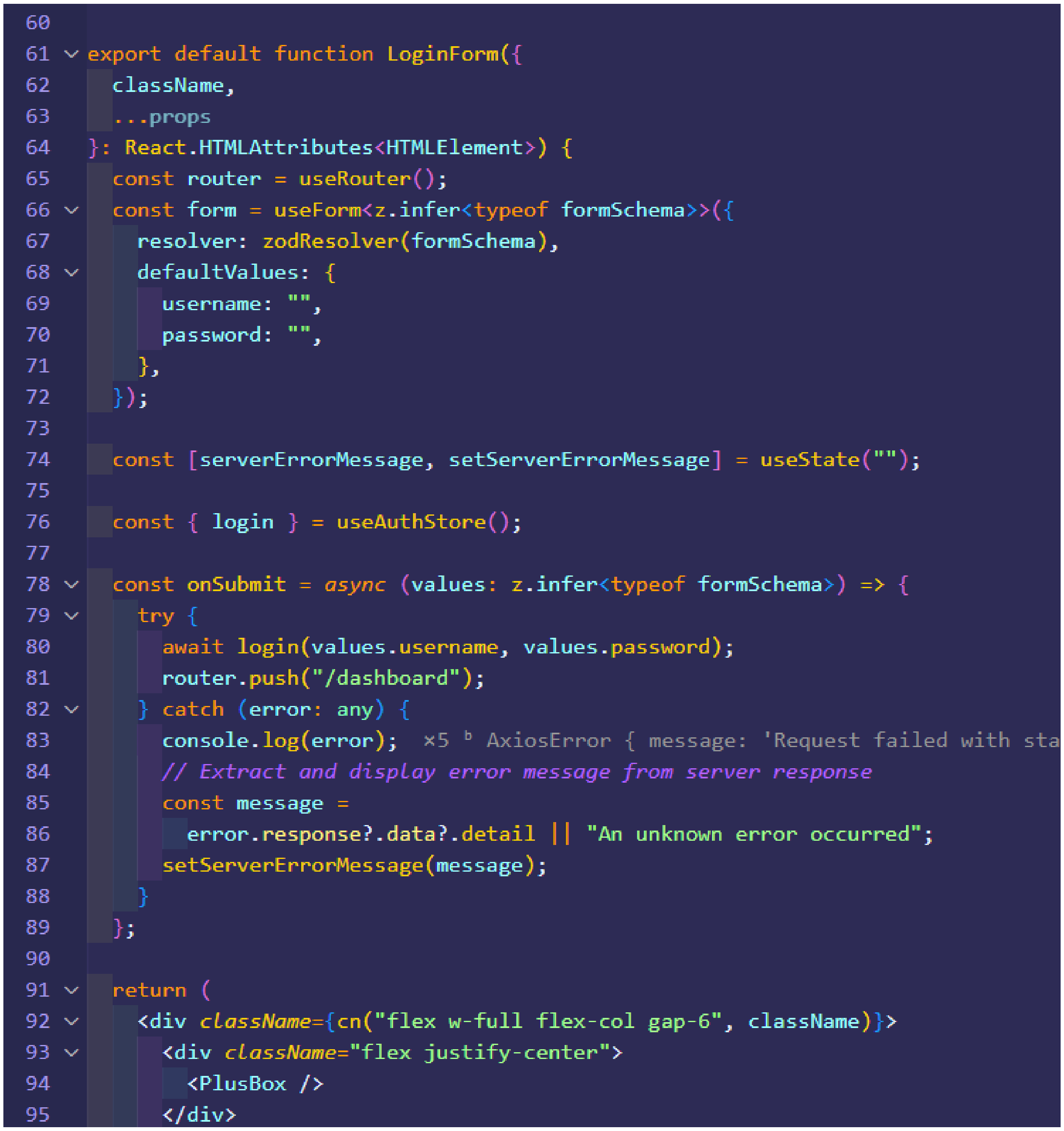
Re-Enter your password

Re-Enter your password

Account Type

[✓ patient]

doctor



```
export default function LoginForm({
  className,
  ...props
}: React.HTMLAttributes<HTMLElement>) {
  const router = useRouter();
  const form = useForm<z.infer<typeof formSchema>>({
    resolver: zodResolver(formSchema),
    defaultValues: {
      username: "",
      password: "",
    },
  });

  const [serverErrorMessage, setServerErrorMessage] = useState("");

  const { login } = useAuthStore();

  const onSubmit = async (values: z.infer<typeof formSchema>) => {
    try {
      await login(values.username, values.password);
      router.push("/dashboard");
    } catch (error: any) {
      console.log(error);  ×5 ᵇ AxiosError { message: 'Request failed with sta
      // Extract and display error message from server response
      const message =
        error.response?.data?.detail || "An unknown error occurred";
      setServerErrorMessage(message);
    }
  };

  return (
    <div className={cn("flex w-full flex-col gap-6", className)}>
      <div className="flex justify-center">
        <PlusBox />
      </div>
```

Password Management and Security

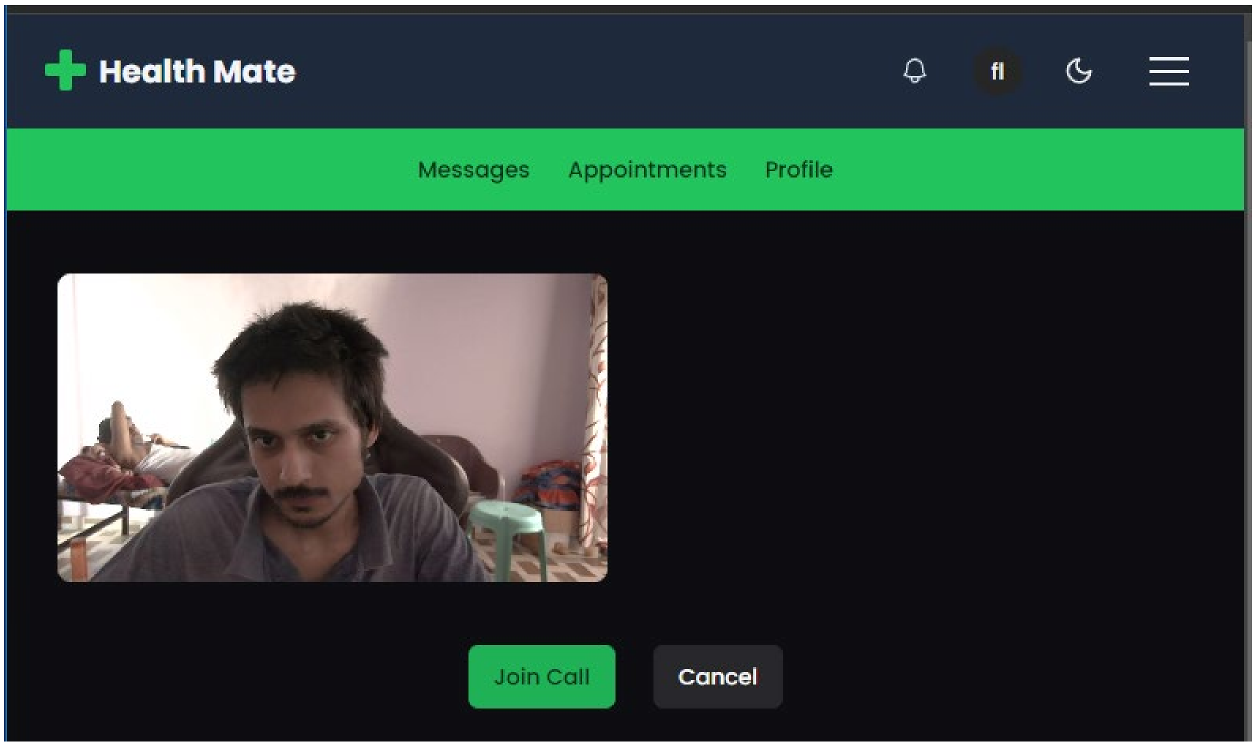
In addition to user registration and login, HealthMate incorporates advanced password management features, including password reset and account activation mechanisms. These functionalities are crucial for maintaining user security and ensuring that access to the system is restricted to authorized individuals only.

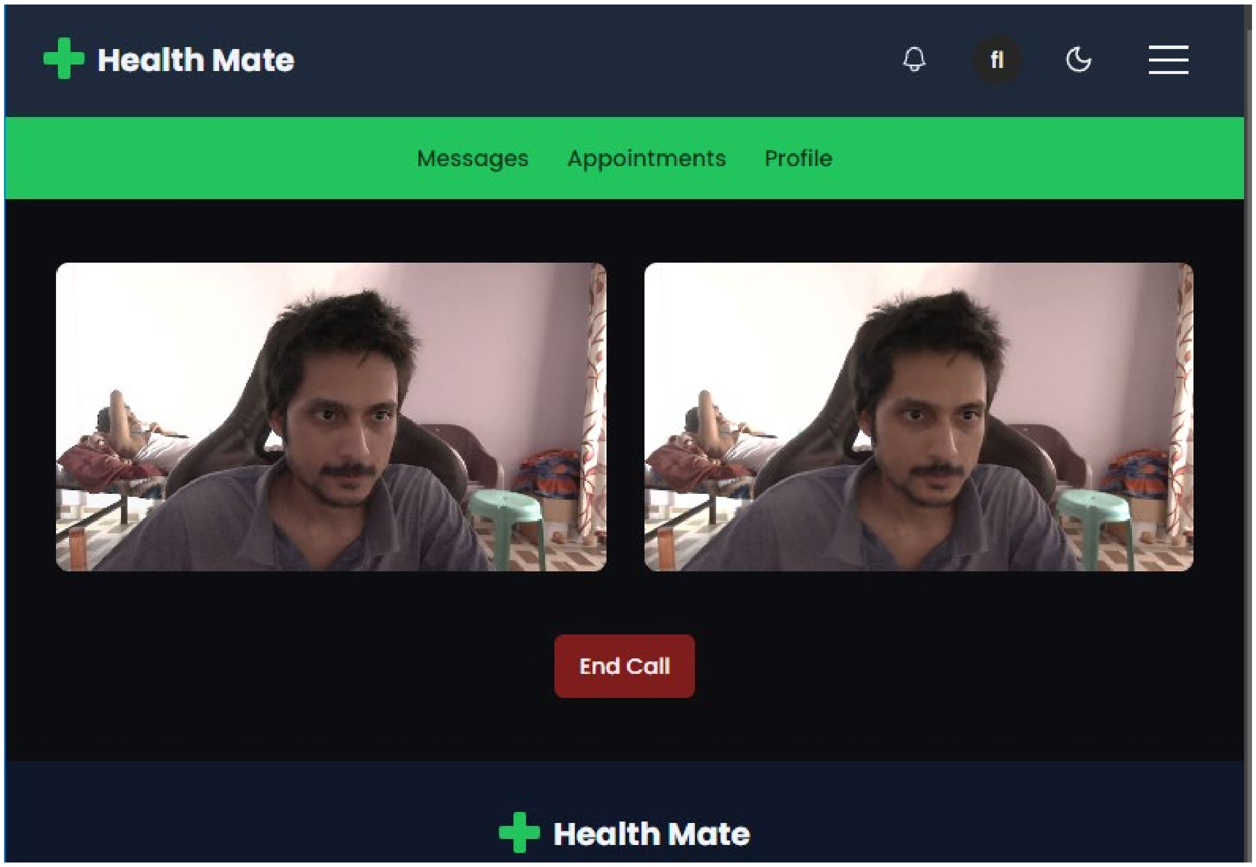
- **Password Reset**: This feature allows users to recover their accounts through a secure process. By submitting their registered email, users receive a password reset link, enabling them to set a new password without compromising their account security.
- **Account Activation**: To further enhance security, HealthMate employs an account activation step post-registration. The **activate** endpoint sends an activation link to the user's email, which must be clicked to activate the account, thereby verifying the email address.

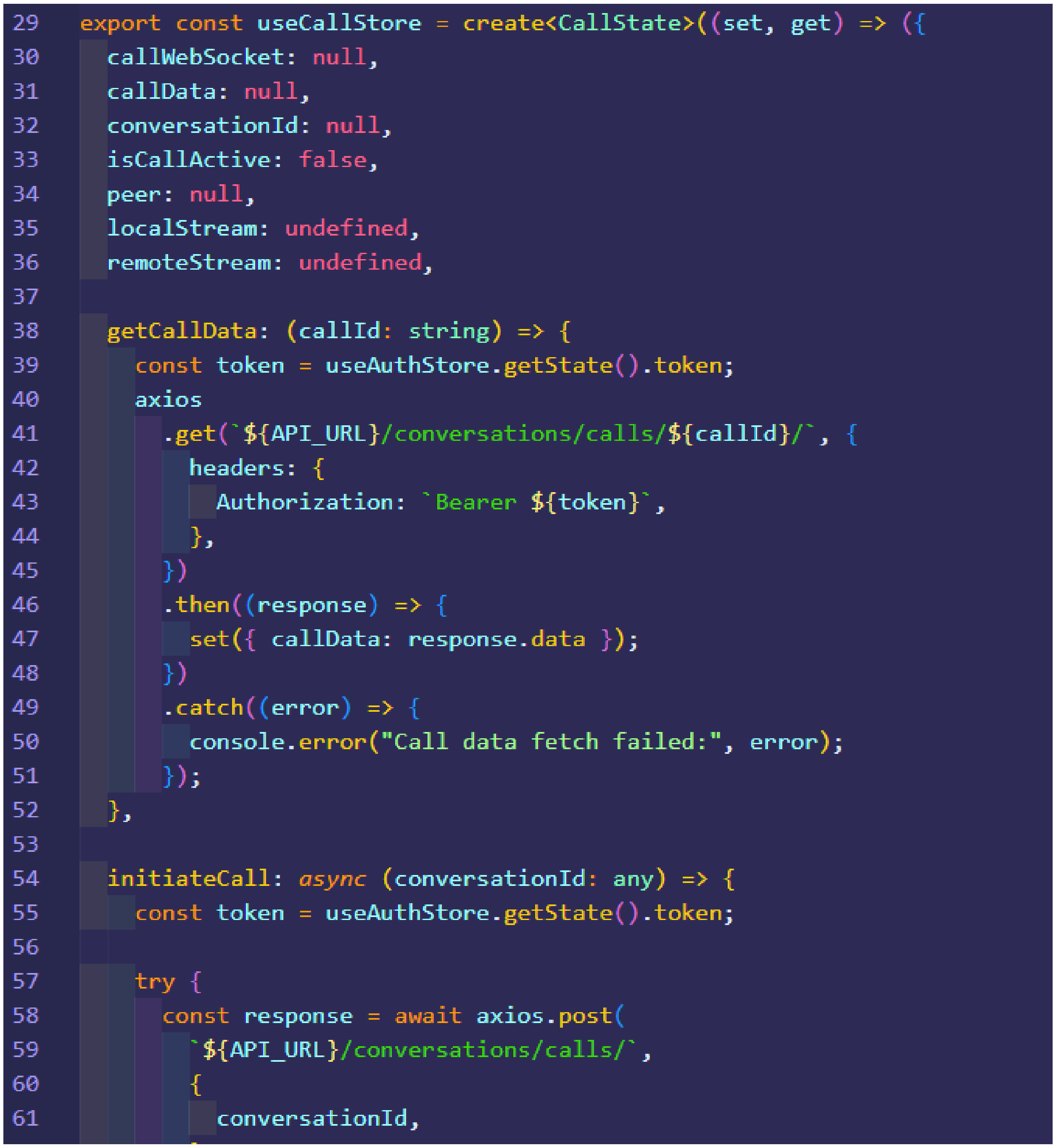
3) Video Calling Feature:

A pivotal feature of HealthMate is its real-time video calling capability, which facilitates direct communication between patients and healthcare providers. This feature leverages the WebRTC protocol for peer-to-peer communication, with Django Channels acting as the signaling server to coordinate the connection establishment between users.

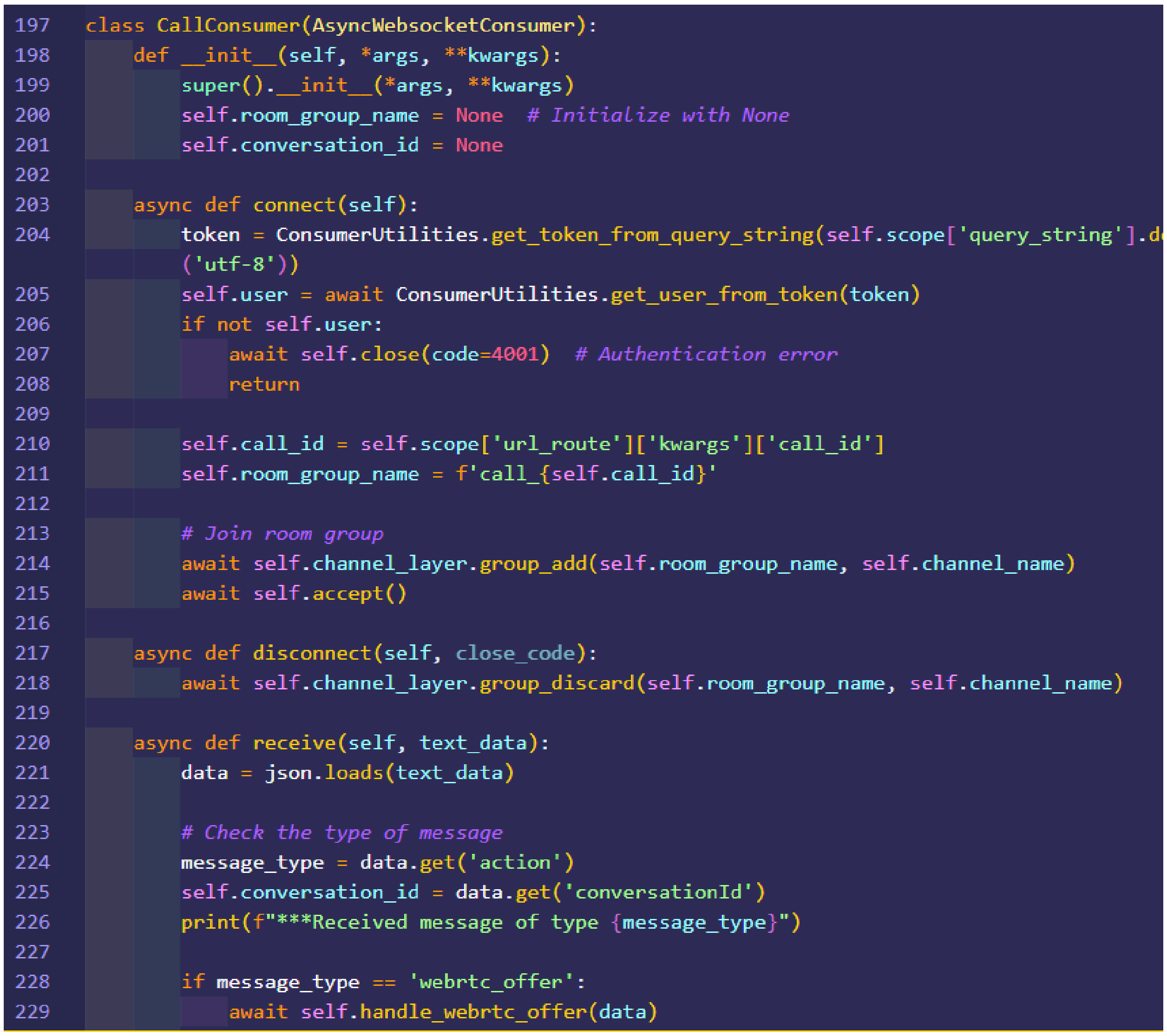
The implementation of video calling is orchestrated through a combination of frontend components and backend consumers. The frontend component, CallPage, initiates the video call setup by capturing user actions and communicating with the backend via WebSocket connections. The backend, specifically the CallConsumer in Django Channels, handles signaling messages between call participants, managing offer and answer messages, ICE candidates, and other necessary signaling data to establish the WebRTC connection.




Health Mate
fl
Messages
Appointments
Profile
End Call
Health Mate



```
export const useCallStore = create<CallState>((set, get) => ({
  callWebSocket: null,
  callData: null,
  conversationId: null,
  isCallActive: false,
  peer: null,
  localStream: undefined,
  remoteStream: undefined,

  getCallData: (callId: string) => {
    const token = useAuthStore.getState().token;
    axios
      .get(`${API_URL}/conversations/calls/${callId}/`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((response) => {
        set({ callData: response.data });
      })
      .catch((error) => {
        console.error("Call data fetch failed:", error);
      });
  },

  initiateCall: async (conversationId: any) => {
    const token = useAuthStore.getState().token;

    try {
      const response = await axios.post(
        `${API_URL}/conversations/calls/`,
        {
          conversationId,
```



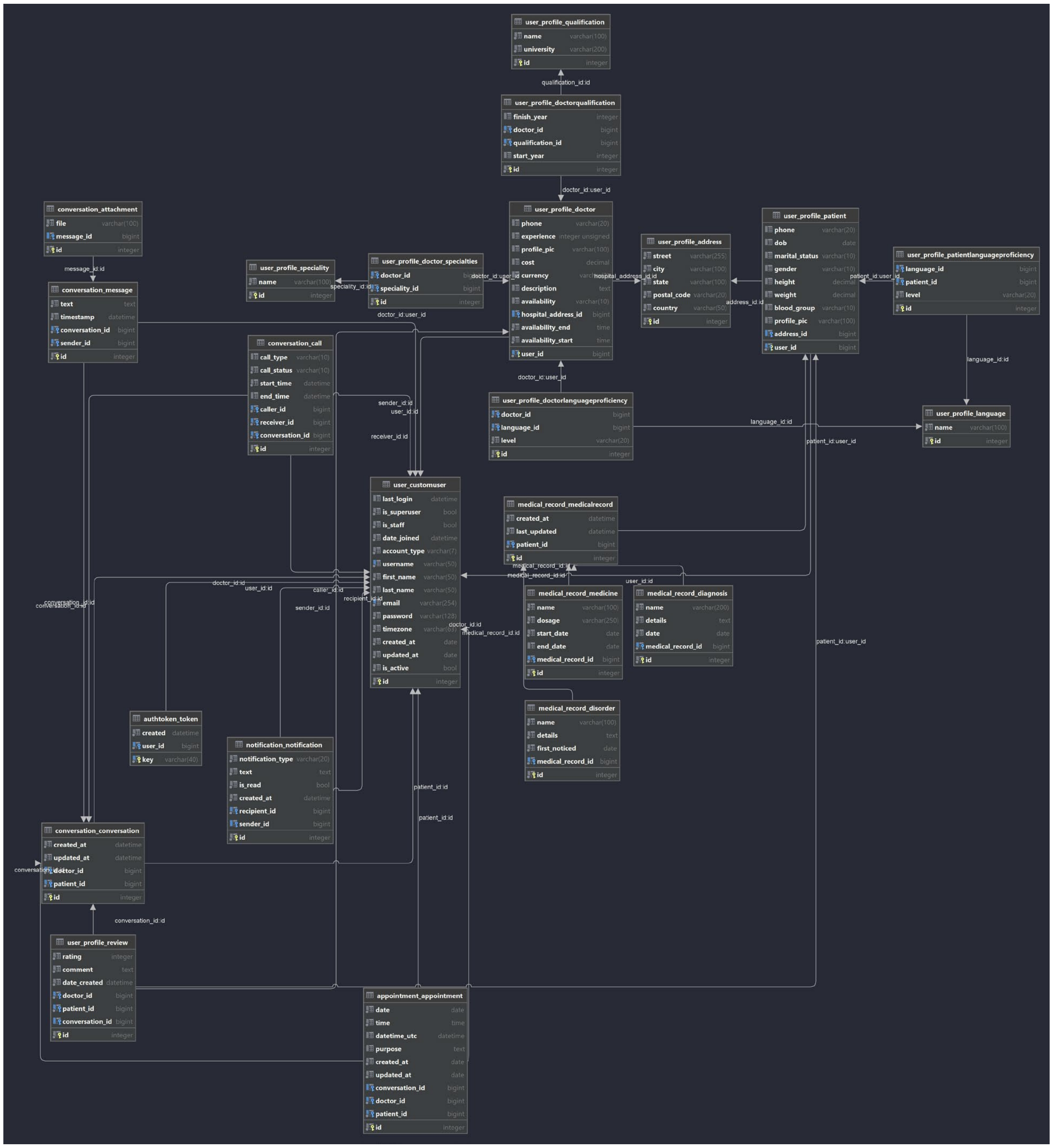
```python
class CallConsumer(AsyncWebsocketConsumer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.room_group_name = None  # Initialize with None
        self.conversation_id = None

    async def connect(self):
        token = ConsumerUtilities.get_token_from_query_string(self.scope['query_string'].d
        ('utf-8'))
        self.user = await ConsumerUtilities.get_user_from_token(token)
        if not self.user:
            await self.close(code=4001)  # Authentication error
            return

        self.call_id = self.scope['url_route']['kwargs']['call_id']
        self.room_group_name = f'call_{self.call_id}'

        # Join room group
        await self.channel_layer.group_add(self.room_group_name, self.channel_name)
        await self.accept()

    async def disconnect(self, close_code):
        await self.channel_layer.group_discard(self.room_group_name, self.channel_name)

    async def receive(self, text_data):
        data = json.loads(text_data)

        # Check the type of message
        message_type = data.get('action')
        self.conversation_id = data.get('conversationId')
        print(f"***Received message of type {message_type}")

        if message_type == 'webrtc_offer':
            await self.handle_webrtc_offer(data)
```

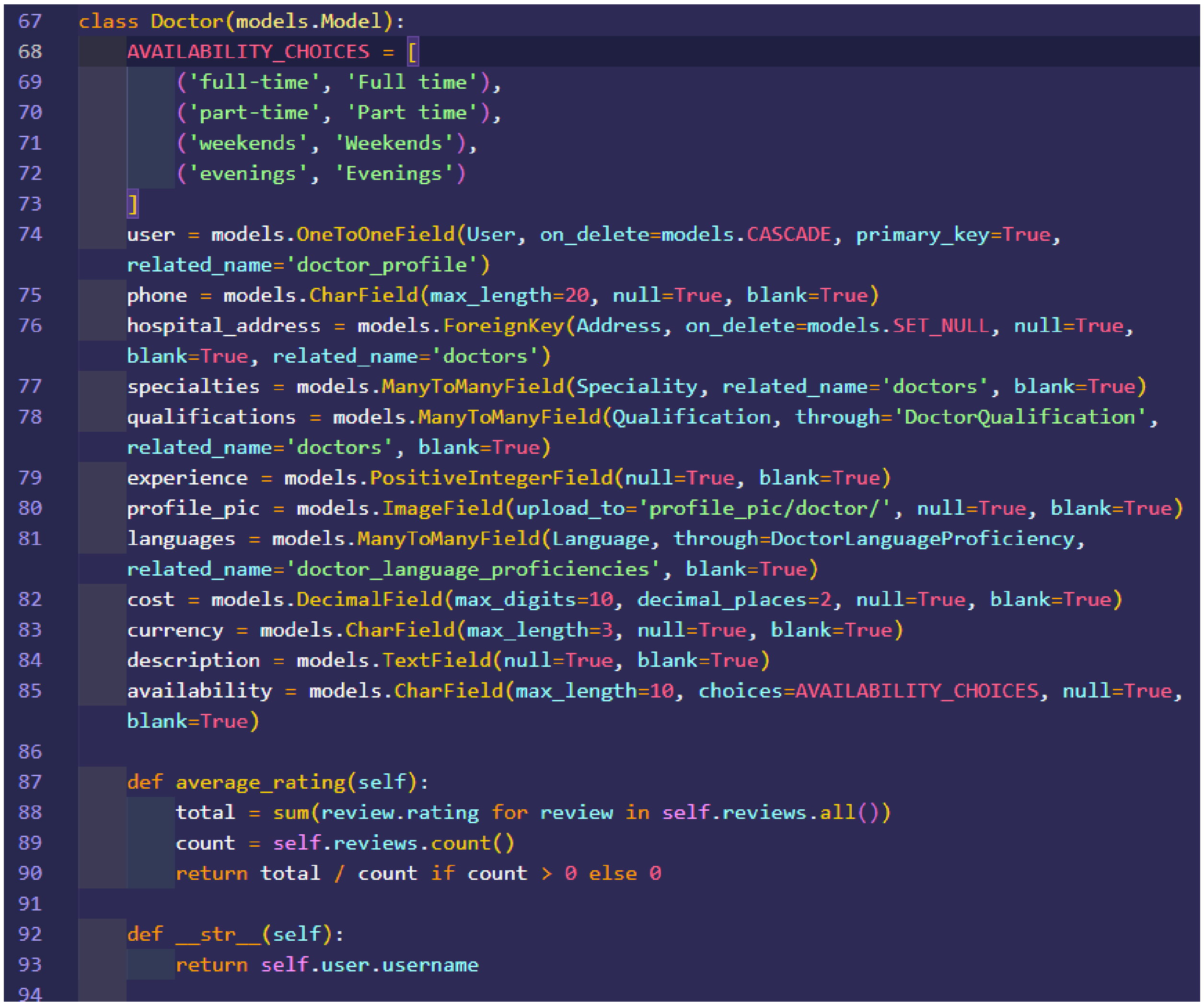
4) Database Structure and Serializers

The backbone of HealthMate's data management is its relational database, designed to store and organize data related to users, appointments, medical records, and other entities. The Django Object-Relational Mapping (ORM) framework is utilized to define the database schema through models, which are then manipulated through Django views and serializers to interact with the frontend.

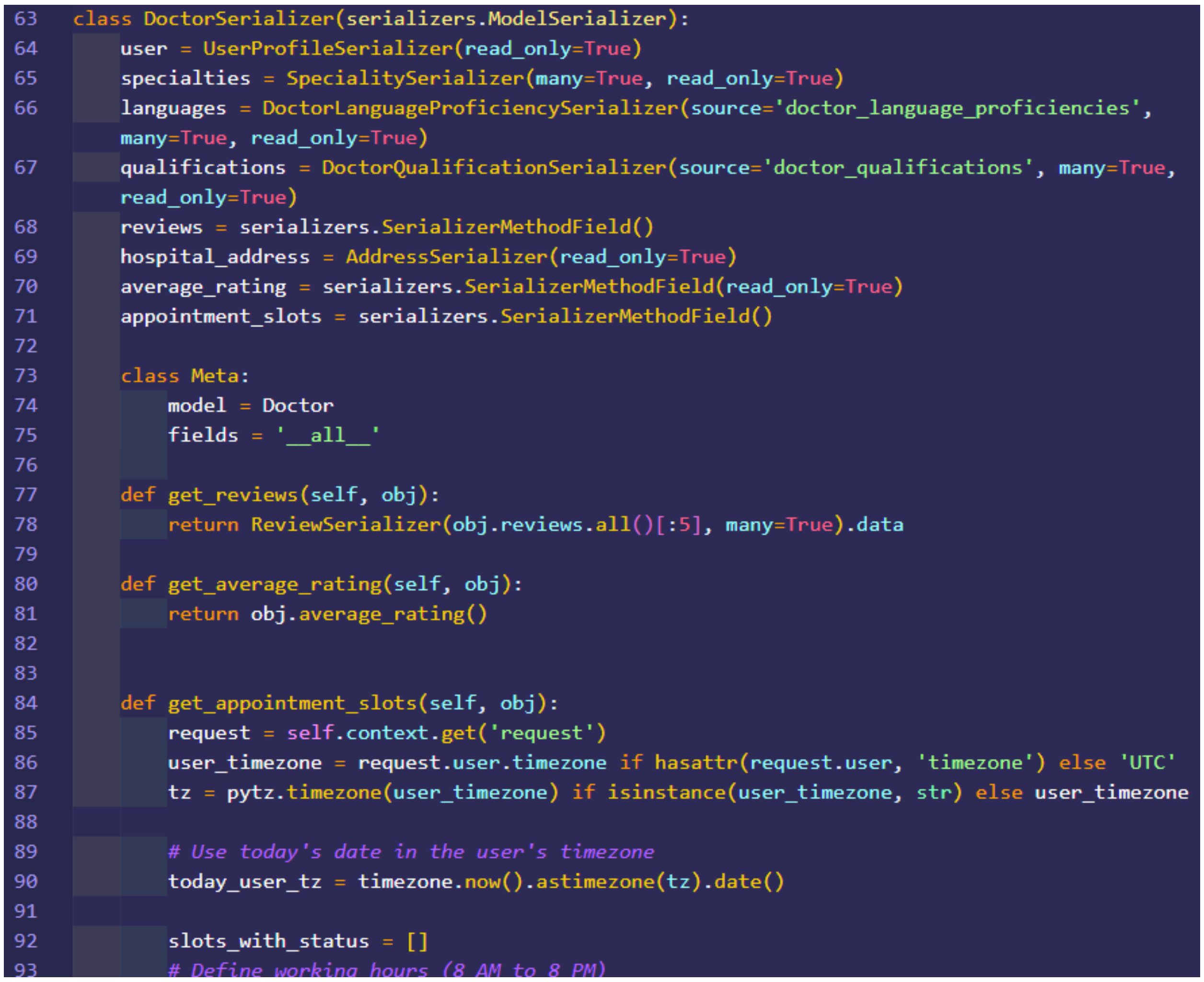
Models and Serializers

Models such as User, Appointment, and MedicalRecord define the structure of the database tables, specifying the fields and relationships between different entities. Serializers translate these models into JSON format, facilitating the exchange of data between the server and the client. This approach ensures a clear separation of concerns, with models focusing on data integrity and serializers on data representation.

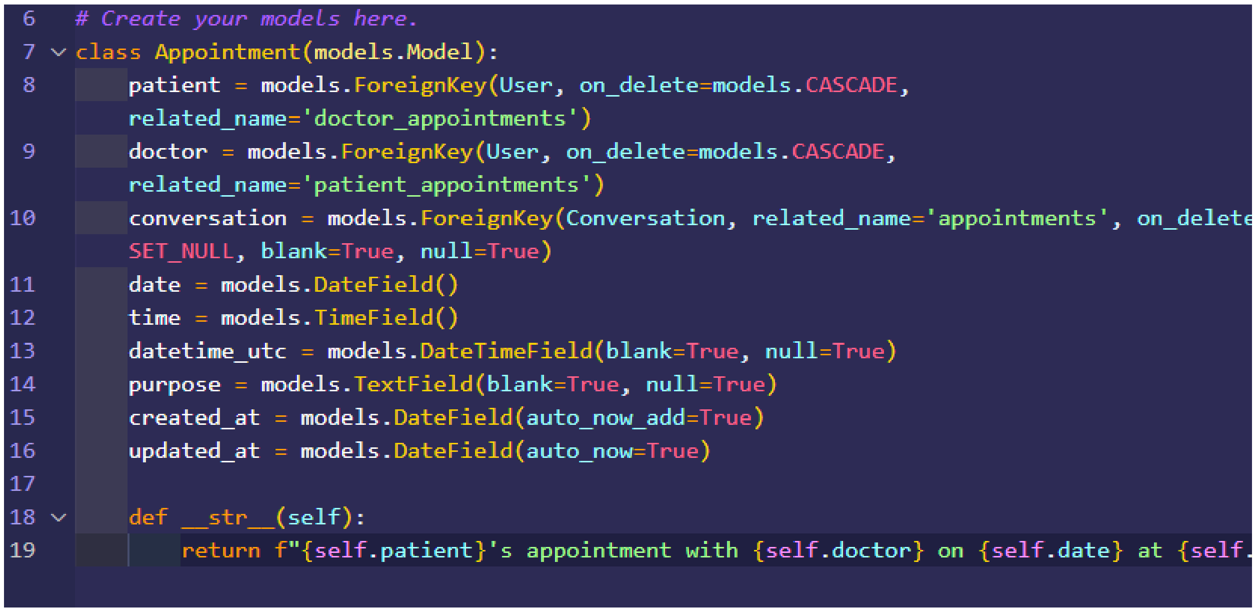

user_profile_qualification
name varchar(100)
university varchar(200)
id integer
qualification_id:id
user_profile_doctorqualification
finish_year integer
doctor_id bigint
qualification_id bigint
start_year integer
id integer
doctor_id:user_id
conversation_attachment
file varchar(100)
message_id bigint
id integer
message_id:id
conversation_message
text text
timestamp datetime
conversation_id bigint
sender_id bigint
id integer
user_profile_speciality
name varchar(100)
id integer
specialty_id:id
user_profile_doctor_specialties
doctor_id bigint
speciality_id bigint
id integer
doctor_id:user_id
user_profile_doctor
phone varchar(20)
experience integer unsigned
profile_pic varchar(100)
cost decimal
currency varchar
description text
availability varchar(10)
hospital_address_id bigint
availability_end time
availability_start time
user_id bigint
hospital_address_id:id
user_profile_address
street varchar(255)
city varchar(100)
state varchar(100)
postal_code varchar(20)
country varchar(50)
id integer
address_id:id
user_profile_patient
phone varchar(20)
dob date
marital_status varchar(10)
gender varchar(10)
height decimal
weight decimal
blood_group varchar(10)
profile_pic varchar(100)
address_id bigint
user_id bigint
patient_id:user_id
user_profile_patientlanguageproficiency
language_id bigint
patient_id bigint
level varchar(20)
id integer
language_id:id
user_profile_language
name varchar(100)
id integer
conversation_call
call_type varchar(10)
call_status varchar(10)
start_time datetime
end_time datetime
caller_id bigint
receiver_id bigint
conversation_id bigint
id integer
sender_id:id
user_id:id
receiver_id:id
doctor_id:user_id
user_profile_doctorlanguageproficiency
doctor_id bigint
language_id bigint
level varchar(20)
id integer
language_id:id
patient_id:user_id
user_customuser
last_login datetime
is_superuser bool
is_staff bool
date_joined datetime
account_type varchar(7)
username varchar(50)
first_name varchar(50)
last_name varchar(50)
email varchar(254)
password varchar(128)
timezone varchar(63)
created_at date
updated_at date
is_active bool
id integer
medical_record_medicalrecord
created_at datetime
last_updated datetime
patient_id bigint
id integer
medical_record_id:id
medical_record_id:id
user_id:id
medical_record_medicine
name varchar(100)
dosage varchar(250)
start_date date
end_date date
medical_record_id bigint
id integer
medical_record_diagnosis
name varchar(200)
details text
date date
medical_record_id bigint
id integer
doctor_id:id
user_id:id
caller_id:id
recipient_id:id
sender_id:id
doctor_id:id
medical_record_id:id
conversation_id:id
conversation_id:id
patient_id:user_id
medical_record_disorder
name varchar(100)
details text
first_noticed date
medical_record_id bigint
id integer
authtoken_token
created datetime
user_id bigint
key varchar(40)
notification_notification
notification_type varchar(20)
text text
is_read bool
created_at datetime
recipient_id bigint
sender_id bigint
id integer
patient_id:id
patient_id:id
conversation_conversation
created_at datetime
updated_at datetime
doctor_id bigint
patient_id bigint
id integer
conversation_id:id
conversation_id:id
user_profile_review
rating integer
comment text
date_created datetime
doctor_id bigint
patient_id bigint
conversation_id bigint
id integer
appointment_appointment
date date
time time
datetime_utc datetime
purpose text
created_at date
updated_at date
conversation_id bigint
doctor_id bigint
patient_id bigint
id integer



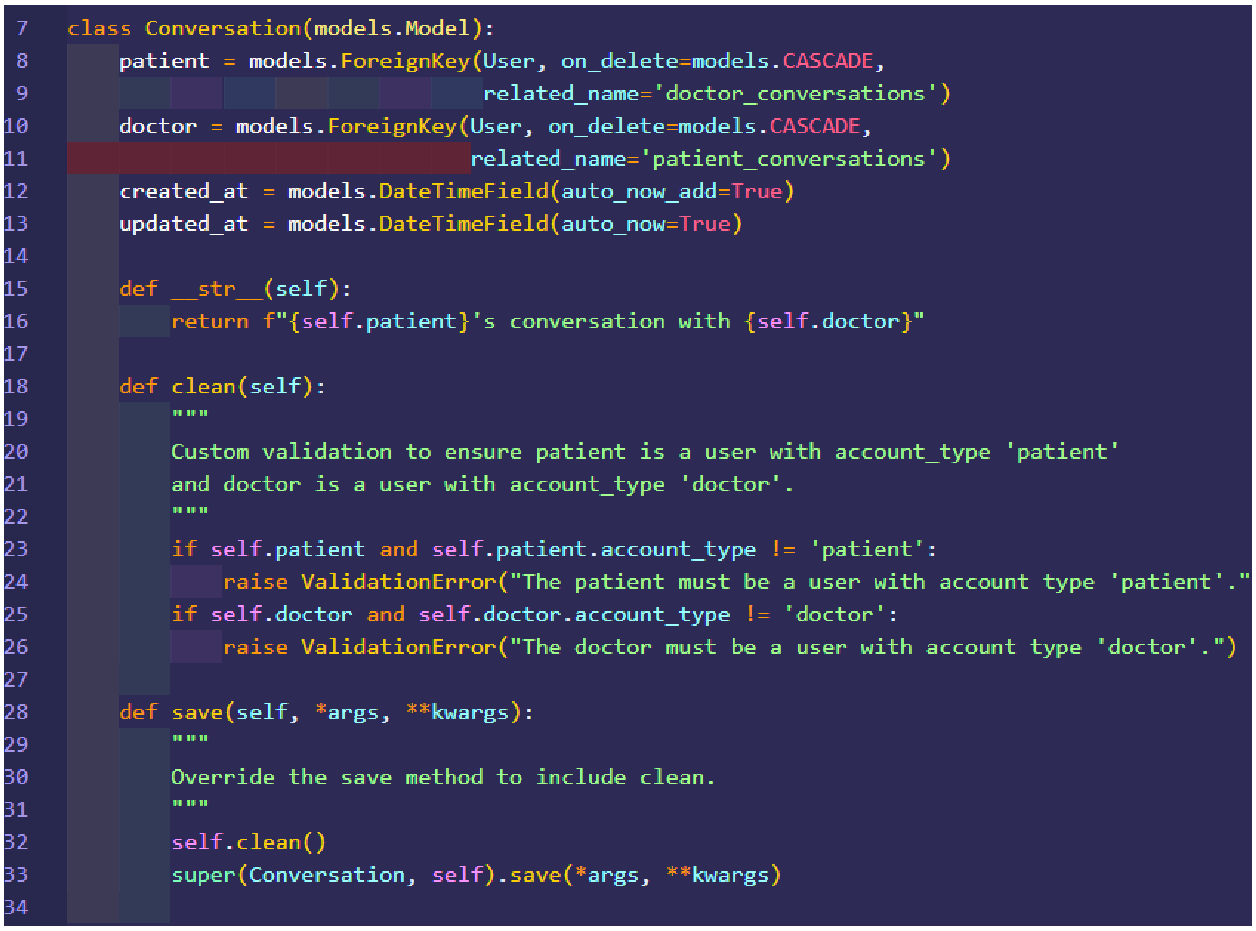
```python
class Doctor(models.Model):
    AVAILABILITY_CHOICES = [
        ('full-time', 'Full time'),
        ('part-time', 'Part time'),
        ('weekends', 'Weekends'),
        ('evenings', 'Evenings')
    ]
    user = models.OneToOneField(User, on_delete=models.CASCADE, primary_key=True,
    related_name='doctor_profile')
    phone = models.CharField(max_length=20, null=True, blank=True)
    hospital_address = models.ForeignKey(Address, on_delete=models.SET_NULL, null=True,
    blank=True, related_name='doctors')
    specialties = models.ManyToManyField(Speciality, related_name='doctors', blank=True)
    qualifications = models.ManyToManyField(Qualification, through='DoctorQualification',
    related_name='doctors', blank=True)
    experience = models.PositiveIntegerField(null=True, blank=True)
    profile_pic = models.ImageField(upload_to='profile_pic/doctor/', null=True, blank=True)
    languages = models.ManyToManyField(Language, through=DoctorLanguageProficiency,
    related_name='doctor_language_proficiencies', blank=True)
    cost = models.DecimalField(max_digits=10, decimal_places=2, null=True, blank=True)
    currency = models.CharField(max_length=3, null=True, blank=True)
    description = models.TextField(null=True, blank=True)
    availability = models.CharField(max_length=10, choices=AVAILABILITY_CHOICES, null=True,
    blank=True)

    def average_rating(self):
        total = sum(review.rating for review in self.reviews.all())
        count = self.reviews.count()
        return total / count if count > 0 else 0

    def __str__(self):
        return self.user.username

```



```python
class DoctorSerializer(serializers.ModelSerializer):
    user = UserProfileSerializer(read_only=True)
    specialties = SpecialitySerializer(many=True, read_only=True)
    languages = DoctorLanguageProficiencySerializer(source='doctor_language_proficiencies',
    many=True, read_only=True)
    qualifications = DoctorQualificationSerializer(source='doctor_qualifications', many=True,
    read_only=True)
    reviews = serializers.SerializerMethodField()
    hospital_address = AddressSerializer(read_only=True)
    average_rating = serializers.SerializerMethodField(read_only=True)
    appointment_slots = serializers.SerializerMethodField()

    class Meta:
        model = Doctor
        fields = '__all__'

    def get_reviews(self, obj):
        return ReviewSerializer(obj.reviews.all()[:5], many=True).data

    def get_average_rating(self, obj):
        return obj.average_rating()


    def get_appointment_slots(self, obj):
        request = self.context.get('request')
        user_timezone = request.user.timezone if hasattr(request.user, 'timezone') else 'UTC'
        tz = pytz.timezone(user_timezone) if isinstance(user_timezone, str) else user_timezone

        # Use today's date in the user's timezone
        today_user_tz = timezone.now().astimezone(tz).date()

        slots_with_status = []
        # Define working hours (8 AM to 8 PM)
```



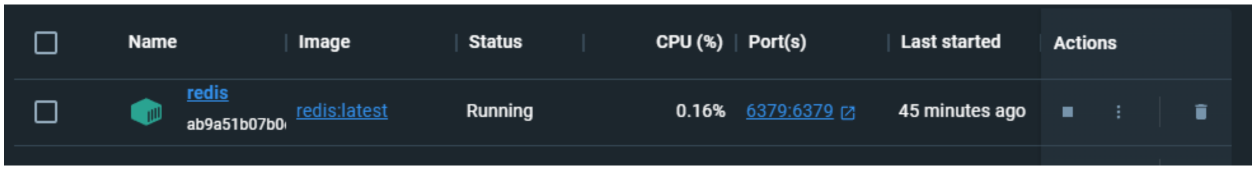
```python
# Create your models here.
class Appointment(models.Model):
    patient = models.ForeignKey(User, on_delete=models.CASCADE,
    related_name='doctor_appointments')
    doctor = models.ForeignKey(User, on_delete=models.CASCADE,
    related_name='patient_appointments')
    conversation = models.ForeignKey(Conversation, related_name='appointments', on_delete
    SET_NULL, blank=True, null=True)
    date = models.DateField()
    time = models.TimeField()
    datetime_utc = models.DateTimeField(blank=True, null=True)
    purpose = models.TextField(blank=True, null=True)
    created_at = models.DateField(auto_now_add=True)
    updated_at = models.DateField(auto_now=True)

    def __str__(self):
        return f"{self.patient}'s appointment with {self.doctor} on {self.date} at {self.
```



```python
class Conversation(models.Model):
    patient = models.ForeignKey(User, on_delete=models.CASCADE,
                                related_name='doctor_conversations')
    doctor = models.ForeignKey(User, on_delete=models.CASCADE,
                               related_name='patient_conversations')
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return f"{self.patient}'s conversation with {self.doctor}"

    def clean(self):
        """
        Custom validation to ensure patient is a user with account_type 'patient'
        and doctor is a user with account_type 'doctor'.
        """
        if self.patient and self.patient.account_type != 'patient':
            raise ValidationError("The patient must be a user with account type 'patient'."
        if self.doctor and self.doctor.account_type != 'doctor':
            raise ValidationError("The doctor must be a user with account type 'doctor'.")

    def save(self, *args, **kwargs):
        """
        Override the save method to include clean.
        """
        self.clean()
        super(Conversation, self).save(*args, **kwargs)
```

5) Redis Configuration

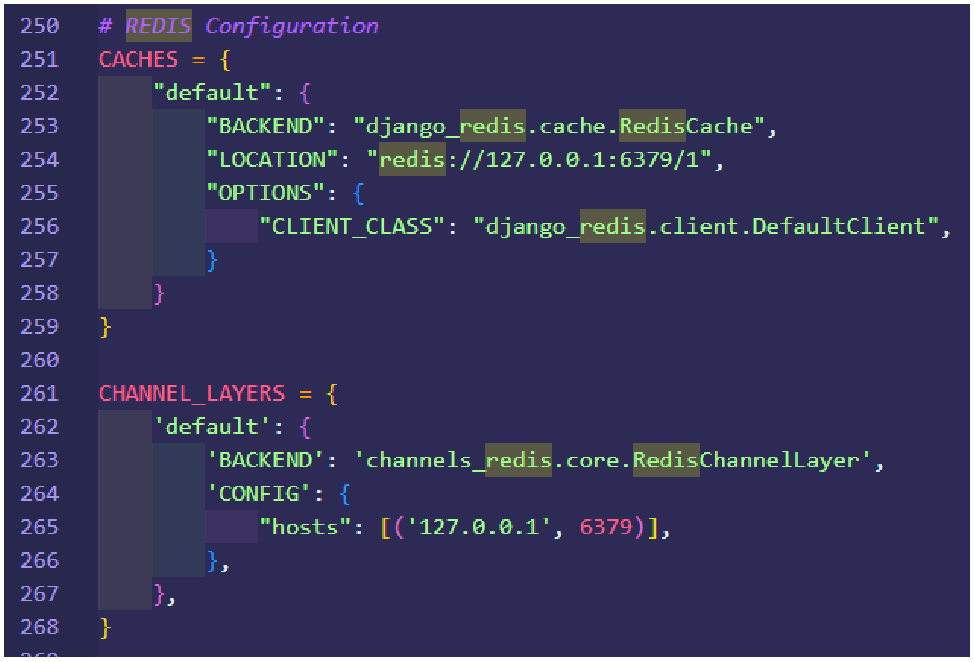
Redis plays a dual role in the HealthMate project, serving as both a cache to enhance performance and a message broker for Django Channels. This setup enables efficient data retrieval and real-time features such as notifications and chat.



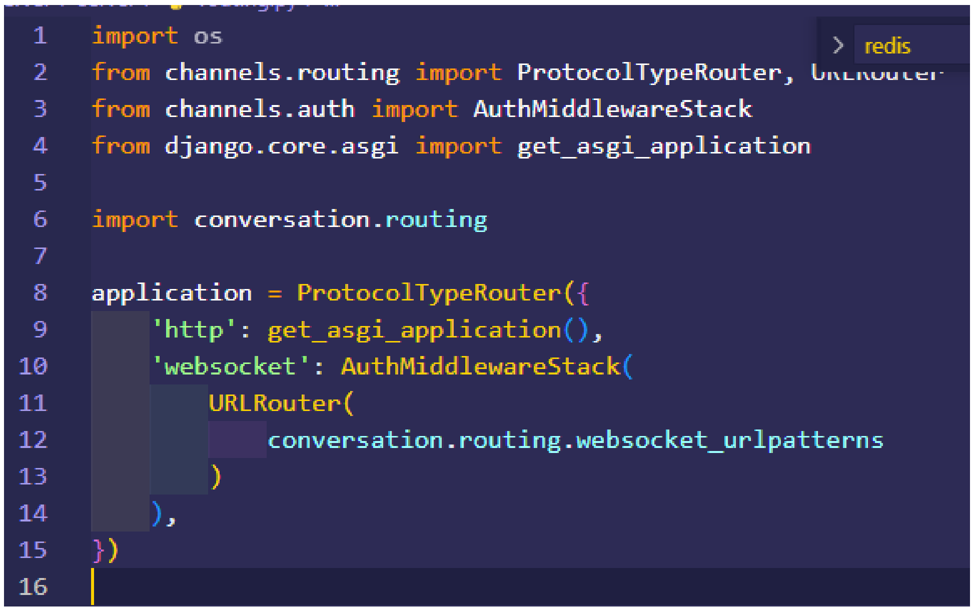
Name	Image	Status	CPU (%)	Port(s)	Last started	Actions
redis ab9a51b07b0	redis:latest	Running	0.16%	6379:6379	45 minutes ago	

Cache and Message Broker

As a cache, Redis stores frequently accessed data, reducing the load on the database and speeding up response times for user requests. As a message broker, it facilitates the communication between WebSocket consumers in Django Channels, ensuring messages are efficiently passed between users in real-time applications.



```python
# REDIS Configuration
CACHES = {
    "default": {
        "BACKEND": "django_redis.cache.RedisCache",
        "LOCATION": "redis://127.0.0.1:6379/1",
        "OPTIONS": {
            "CLIENT_CLASS": "django_redis.client.DefaultClient",
        }
    }
}

CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
            "hosts": [('127.0.0.1', 6379)],
        },
    },
}
```



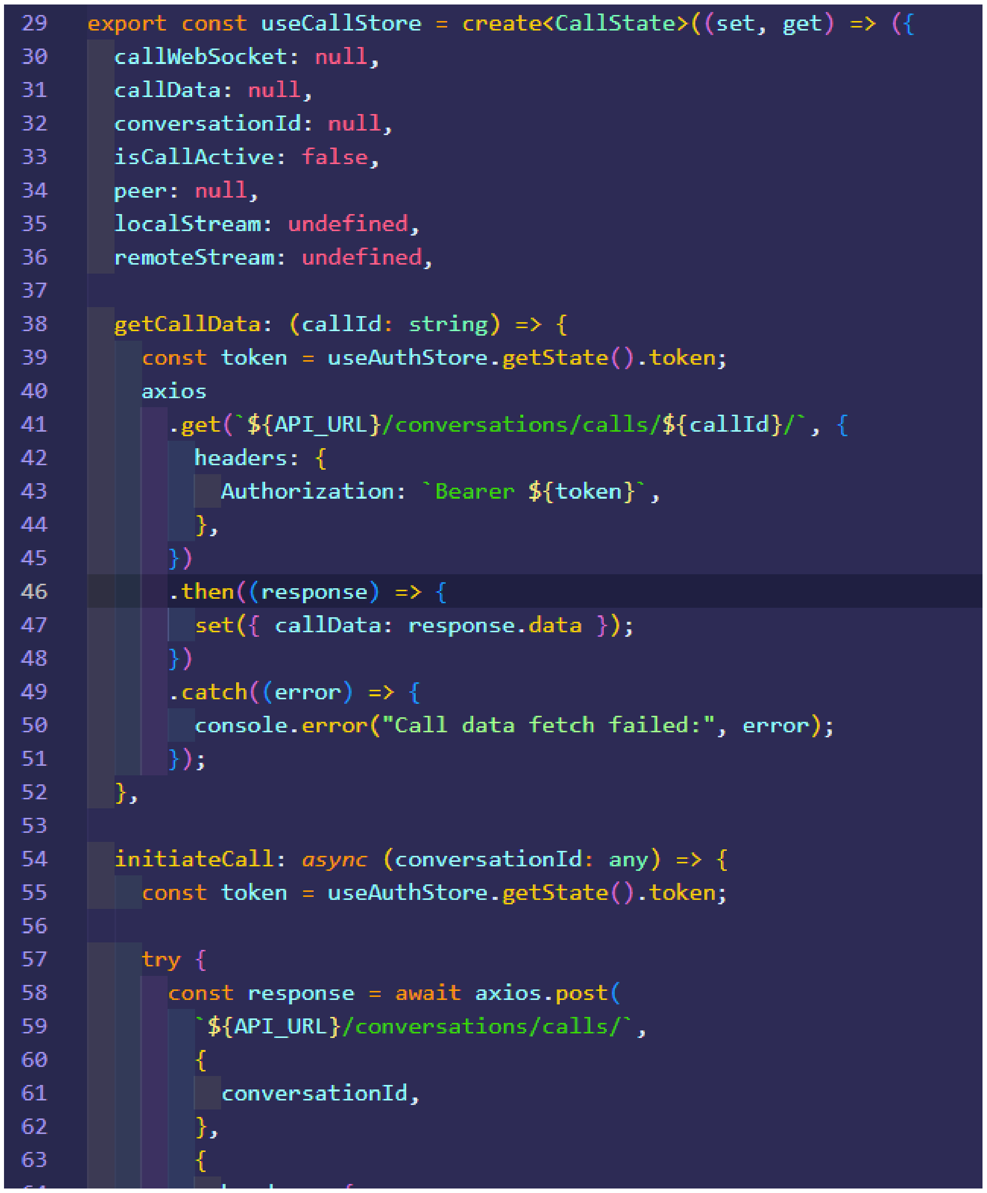
```python
import os
from channels.routing import ProtocolTypeRouter, URLRouter
from channels.auth import AuthMiddlewareStack
from django.core.asgi import get_asgi_application

import conversation.routing

application = ProtocolTypeRouter({
    'http': get_asgi_application(),
    'websocket': AuthMiddlewareStack(
        URLRouter(
            conversation.routing.websocket_urlpatterns
        )
    ),
})
```

6) State Management with Zustand

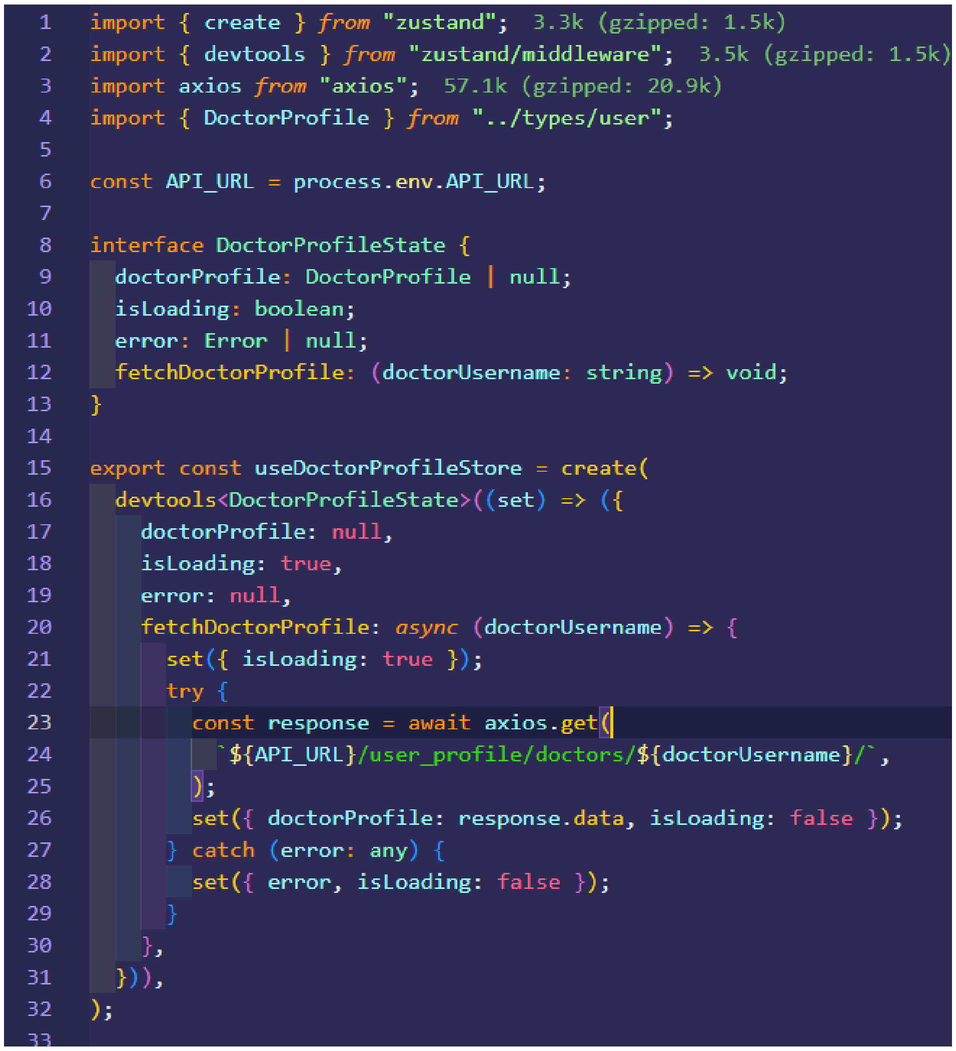
State management is crucial in maintaining the responsiveness and consistency of the HealthMate application. Zustand, a state management library for React, is employed to manage the application's state, including user authentication status, appointment details, and chat messages.

Zustand Stores

Zustand stores, such as useAuthStore, useAppointmentStore, and useMessageStore, encapsulate different aspects of the application's state. Each store provides a set of actions and selectors to update and retrieve the state, enabling components to react dynamically to state changes and ensure a seamless user experience.



```
export const useCallStore = create<CallState>((set, get) => ({
  callWebSocket: null,
  callData: null,
  conversationId: null,
  isCallActive: false,
  peer: null,
  localStream: undefined,
  remoteStream: undefined,

  getCallData: (callId: string) => {
    const token = useAuthStore.getState().token;
    axios
      .get(`${API_URL}/conversations/calls/${callId}/`, {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      })
      .then((response) => {
        set({ callData: response.data });
      })
      .catch((error) => {
        console.error("Call data fetch failed:", error);
      });
  },

  initiateCall: async (conversationId: any) => {
    const token = useAuthStore.getState().token;

    try {
      const response = await axios.post(
        `${API_URL}/conversations/calls/`,
        {
          conversationId,
        },
        {
```



```
import { create } from "zustand";  3.3k (gzipped: 1.5k)
import { devtools } from "zustand/middleware";  3.5k (gzipped: 1.5k)
import axios from "axios";  57.1k (gzipped: 20.9k)
import { DoctorProfile } from "../types/user";

const API_URL = process.env.API_URL;

interface DoctorProfileState {
  doctorProfile: DoctorProfile | null;
  isLoading: boolean;
  error: Error | null;
  fetchDoctorProfile: (doctorUsername: string) => void;
}

export const useDoctorProfileStore = create(
  devtools<DoctorProfileState>((set) => ({
    doctorProfile: null,
    isLoading: true,
    error: null,
    fetchDoctorProfile: async (doctorUsername) => {
      set({ isLoading: true });
      try {
        const response = await axios.get(
          `${API_URL}/user_profile/doctors/${doctorUsername}/`,
        );
        set({ doctorProfile: response.data, isLoading: false });
      } catch (error: any) {
        set({ error, isLoading: false });
      }
    },
  })),
);
```

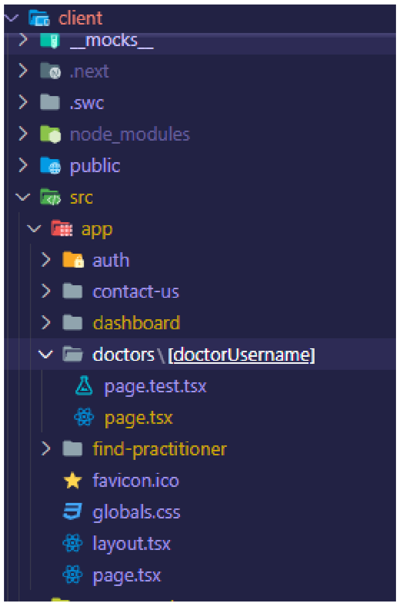
Dynamic State Updates

The dynamic nature of HealthMate's application state, managed through Zustand, allows for real-time updates across the application. This is particularly evident in features such as the messaging system, where Zustand stores like useMessageStore (client/src/store/useMessageStore.ts) enable instant messaging and notifications, reflecting new messages and conversation updates immediately in the UI.

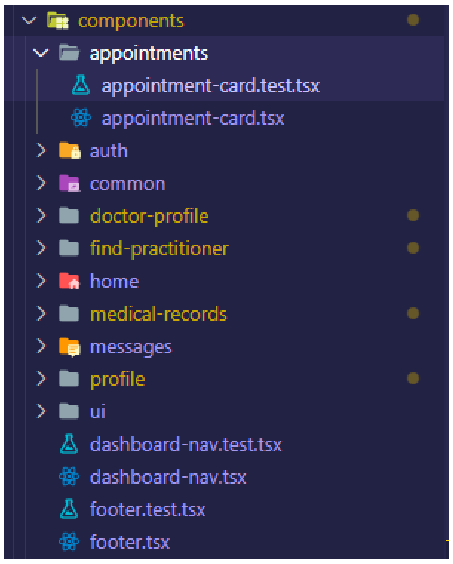
7) Methodology

The HealthMate project adheres to a modular development methodology, emphasizing separation of concerns and component-based architecture. This approach facilitates the development of reusable components and modules, streamlining the development process and enhancing maintainability.

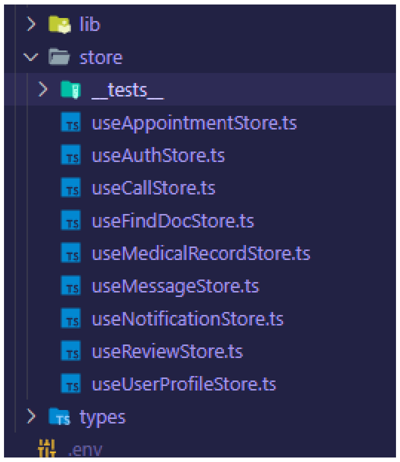
The project is structured into distinct modules for authentication, appointments, medical records, and real-time communication. Each module encapsulates specific functionalities, with clear interfaces for interaction with other parts of the application. This modular architecture simplifies the complexity of the system, making it easier to understand, develop, and test.



```
client
  __mocks__
  .next
  .swc
  node_modules
  public
  src
    app
      auth
      contact-us
      dashboard
      doctors \ [doctorUsername]
        page.test.tsx
        page.tsx
      find-practitioner
      favicon.ico
      globals.css
      layout.tsx
      page.tsx
```



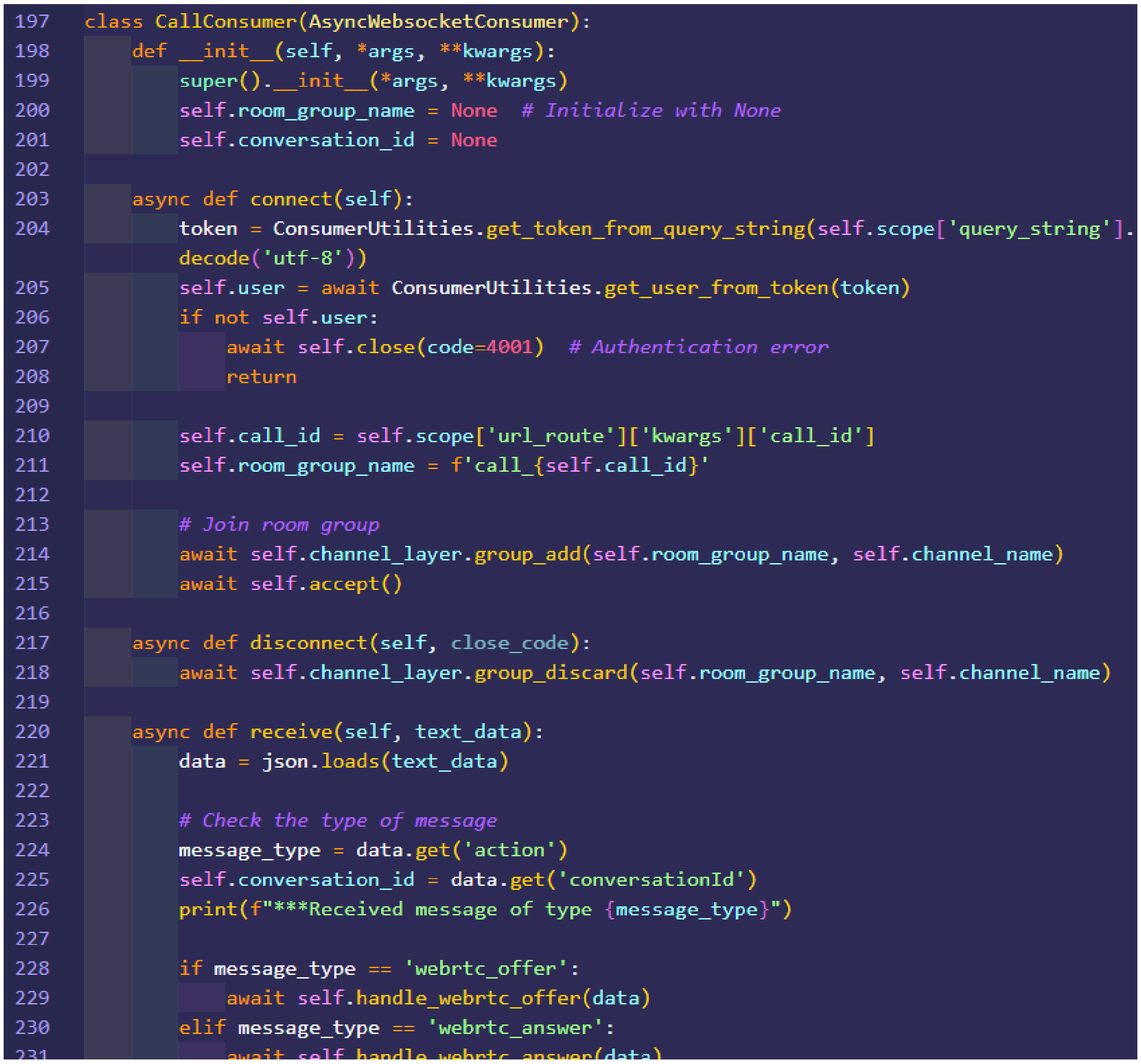
```
components
  appointments
    appointment-card.test.tsx
    appointment-card.tsx
  auth
  common
  doctor-profile
  find-practitioner
  home
  medical-records
  messages
  profile
  ui
  dashboard-nav.test.tsx
  dashboard-nav.tsx
  footer.test.tsx
  footer.tsx
```



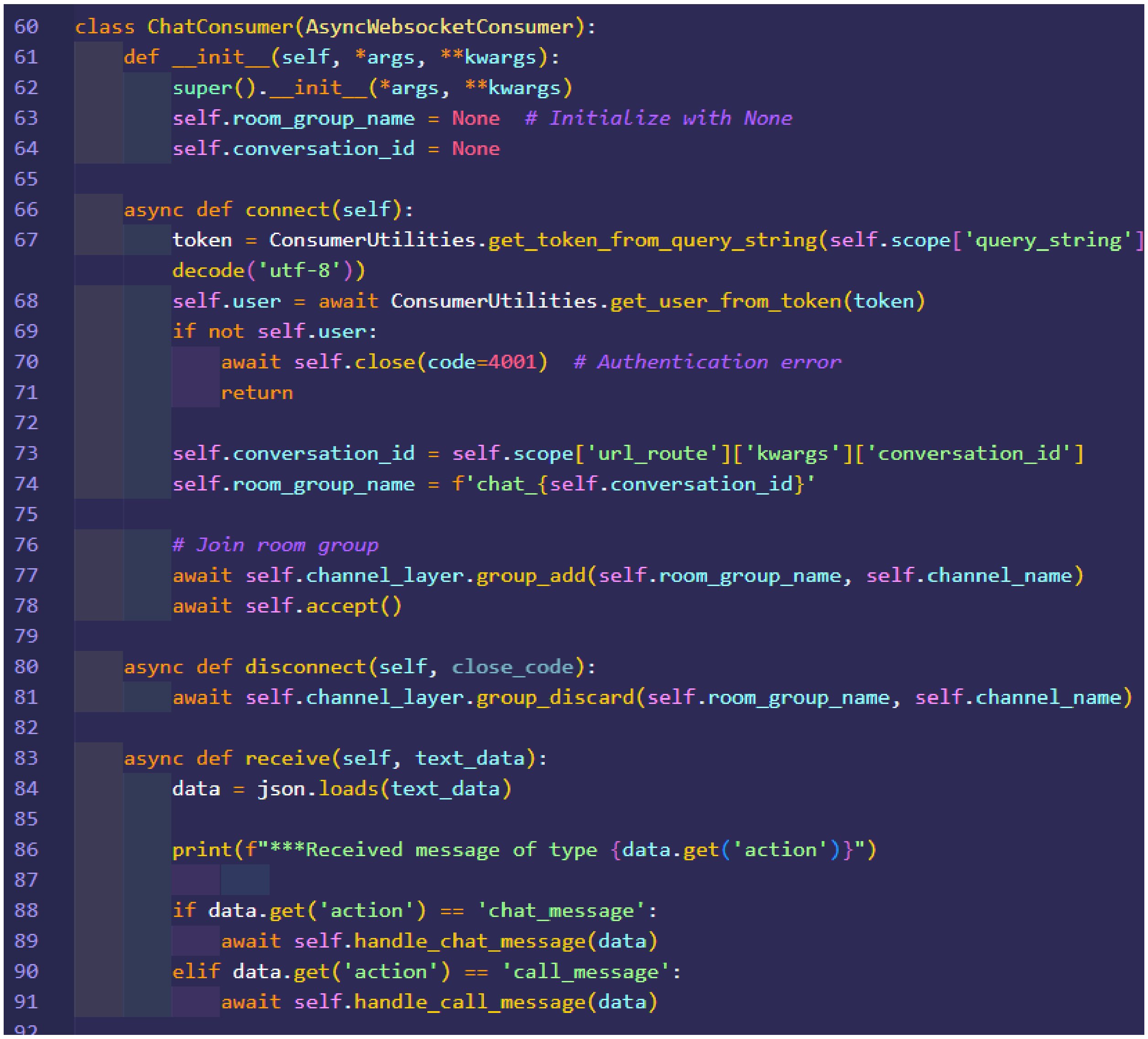
```
lib
store
  __tests__
  useAppointmentStore.ts
  useAuthStore.ts
  useCallStore.ts
  useFindDocStore.ts
  useMedicalRecordStore.ts
  useMessageStore.ts
  useNotificationStore.ts
  useReviewStore.ts
  useUserProfileStore.ts
types
.env
```

8) Conversation Consumers

Conversation consumers, part of the Django Channels framework, handle real-time communication features such as chat and notifications. These consumers manage WebSocket connections, allowing for the real-time exchange of messages between users.



```python
class CallConsumer(AsyncWebsocketConsumer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.room_group_name = None  # Initialize with None
        self.conversation_id = None

    async def connect(self):
        token = ConsumerUtilities.get_token_from_query_string(self.scope['query_string'].decode('utf-8'))
        self.user = await ConsumerUtilities.get_user_from_token(token)
        if not self.user:
            await self.close(code=4001)  # Authentication error
            return

        self.call_id = self.scope['url_route']['kwargs']['call_id']
        self.room_group_name = f'call_{self.call_id}'

        # Join room group
        await self.channel_layer.group_add(self.room_group_name, self.channel_name)
        await self.accept()

    async def disconnect(self, close_code):
        await self.channel_layer.group_discard(self.room_group_name, self.channel_name)

    async def receive(self, text_data):
        data = json.loads(text_data)

        # Check the type of message
        message_type = data.get('action')
        self.conversation_id = data.get('conversationId')
        print(f"***Received message of type {message_type}")

        if message_type == 'webrtc_offer':
            await self.handle_webrtc_offer(data)
        elif message_type == 'webrtc_answer':
            await self.handle_webrtc_answer(data)
```

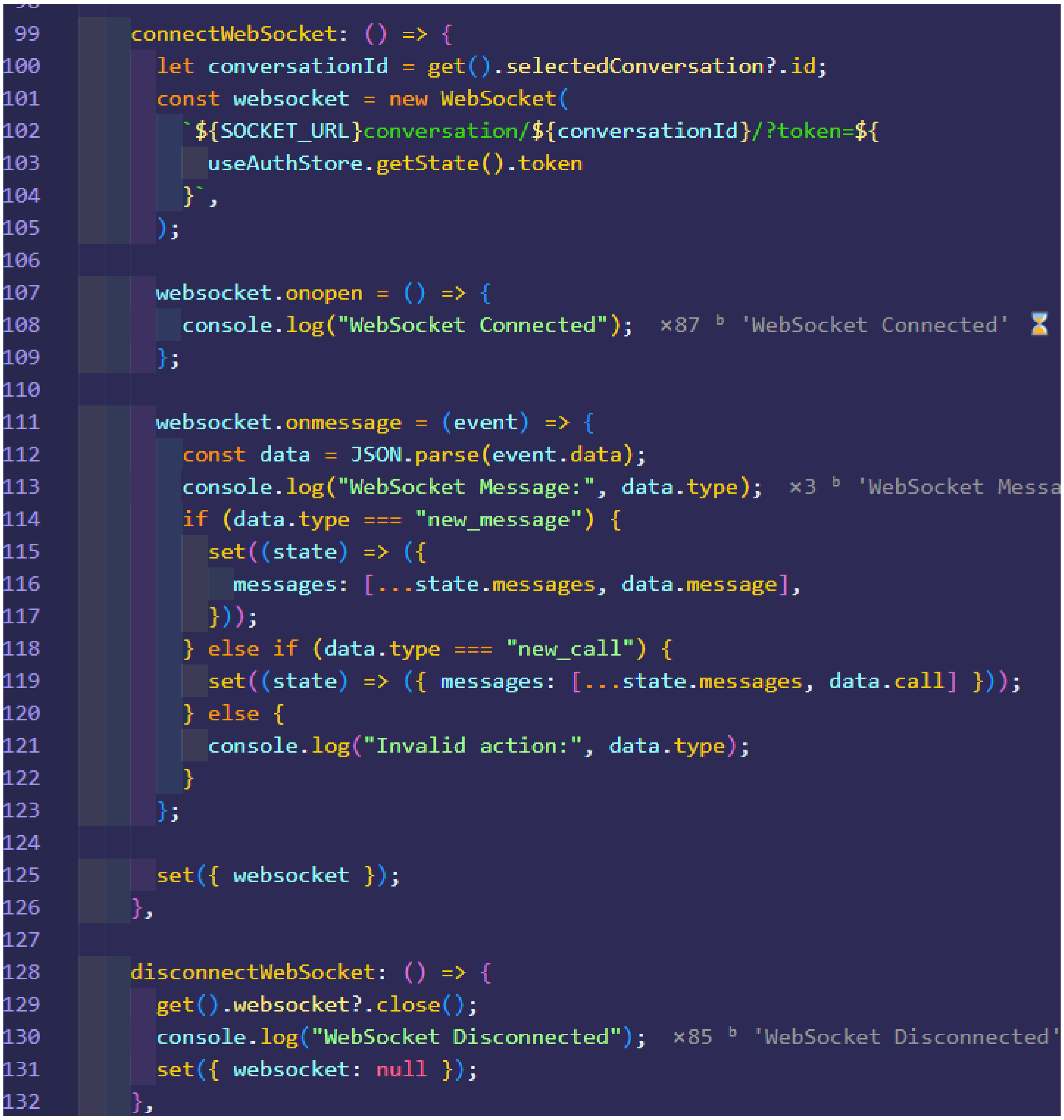

```python
class ChatConsumer(AsyncWebsocketConsumer):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.room_group_name = None  # Initialize with None
        self.conversation_id = None

    async def connect(self):
        token = ConsumerUtilities.get_token_from_query_string(self.scope['query_string']
        decode('utf-8'))
        self.user = await ConsumerUtilities.get_user_from_token(token)
        if not self.user:
            await self.close(code=4001)  # Authentication error
            return

        self.conversation_id = self.scope['url_route']['kwargs']['conversation_id']
        self.room_group_name = f'chat_{self.conversation_id}'

        # Join room group
        await self.channel_layer.group_add(self.room_group_name, self.channel_name)
        await self.accept()

    async def disconnect(self, close_code):
        await self.channel_layer.group_discard(self.room_group_name, self.channel_name)

    async def receive(self, text_data):
        data = json.loads(text_data)

        print(f"***Received message of type {data.get('action')}")

        if data.get('action') == 'chat_message':
            await self.handle_chat_message(data)
        elif data.get('action') == 'call_message':
            await self.handle_call_message(data)
```

Real-time Communication

The implementation of conversation consumers involves defining asynchronous methods to handle events such as connection establishment, message reception, and disconnection. These methods facilitate the real-time interaction between users, ensuring that messages are promptly delivered and displayed in the user interface.

WebSocket Management

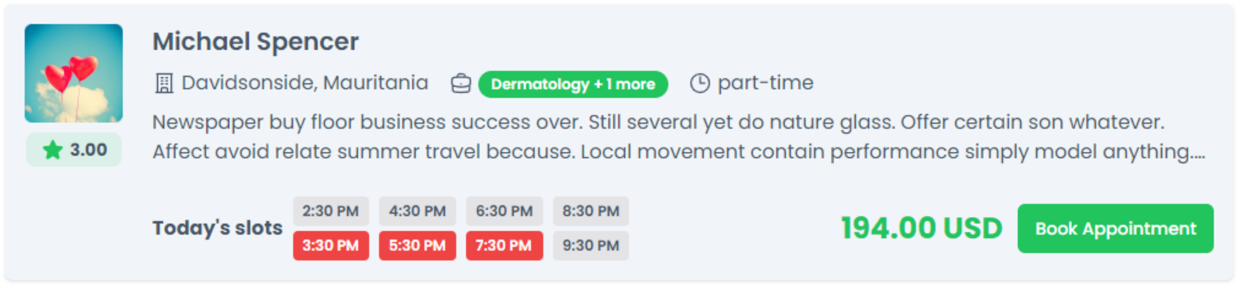
The management of WebSocket connections is a critical aspect of HealthMate's real-time communication capabilities. The ConversationConsumer (server/conversation/consumers.py) implements efficient connection handling, message broadcasting, and disconnection logic, ensuring a stable and responsive chat feature.



```
connectWebSocket: () => {
  let conversationId = get().selectedConversation?.id;
  const websocket = new WebSocket(
    `${SOCKET_URL}conversation/${conversationId}/?token=${
      useAuthStore.getState().token
    }`,
  );

  websocket.onopen = () => {
    console.log("WebSocket Connected");  ×87 ᵇ 'WebSocket Connected' ⌛
  };

  websocket.onmessage = (event) => {
    const data = JSON.parse(event.data);
    console.log("WebSocket Message:", data.type);  ×3 ᵇ 'WebSocket Messa
    if (data.type === "new_message") {
      set((state) => ({
        messages: [...state.messages, data.message],
      }));
    } else if (data.type === "new_call") {
      set((state) => ({ messages: [...state.messages, data.call] }));
    } else {
      console.log("Invalid action:", data.type);
    }
  };

  set({ websocket });
},

disconnectWebSocket: () => {
  get().websocket?.close();
  console.log("WebSocket Disconnected");  ×85 ᵇ 'WebSocket Disconnected'
  set({ websocket: null });
},
```

9) Appointment Scheduling

Appointment scheduling is a core functionality of HealthMate, enabling users to book, view, and manage healthcare appointments. The system integrates a comprehensive scheduling interface, backed by a robust backend that handles appointment data.


Michael Spencer

Davidsonside, Mauritania · Dermatology + 1 more · part-time

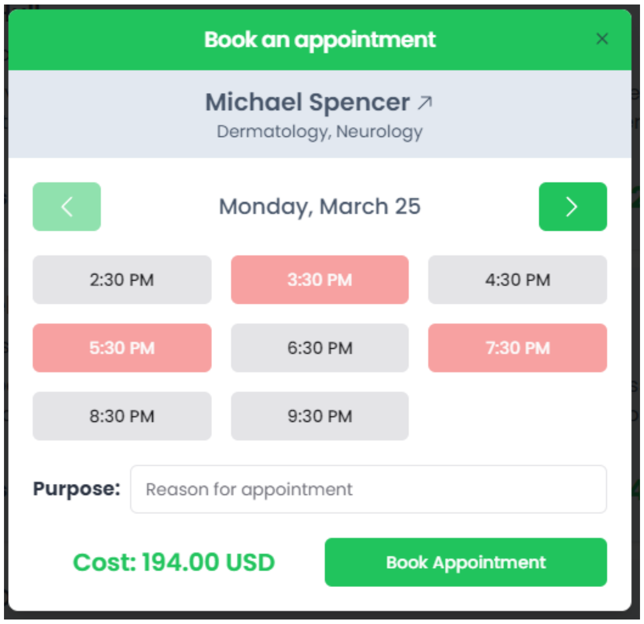
Newspaper buy floor business success over. Still several yet do nature glass. Offer certain son whatever. Affect avoid relate summer travel because. Local movement contain performance simply model anything....

3.00

Today's slots: 2:30 PM, 4:30 PM, 6:30 PM, 8:30 PM, 3:30 PM, 5:30 PM, 7:30 PM, 9:30 PM

194.00 USD

Book Appointment


Book an appointment

Michael Spencer

Dermatology, Neurology

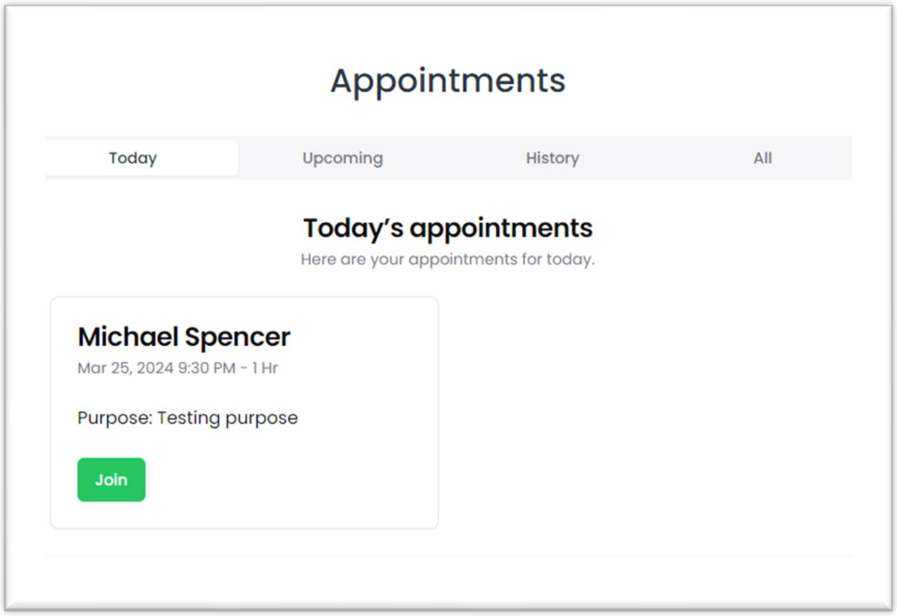
Monday, March 25

2:30 PM · 3:30 PM · 4:30 PM · 5:30 PM · 6:30 PM · 7:30 PM · 8:30 PM · 9:30 PM

Purpose: Reason for appointment

Cost: 194.00 USD

Book Appointment


Appointments

Today · Upcoming · History · All

Today's appointments

Here are your appointments for today.

Michael Spencer

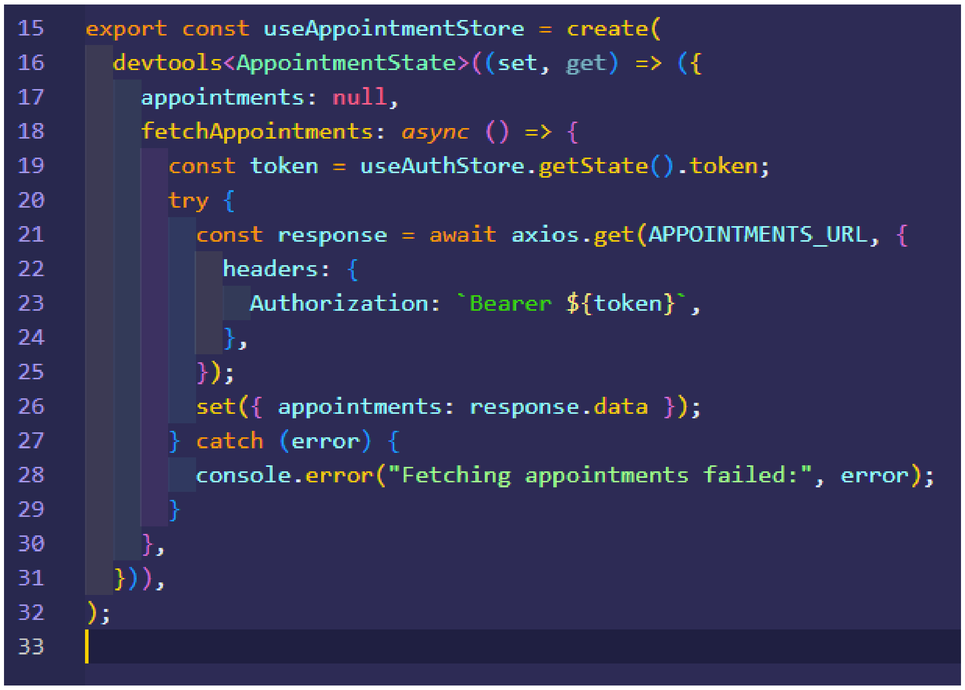
Mar 25, 2024 9:30 PM - 1 Hr

Purpose: Testing purpose

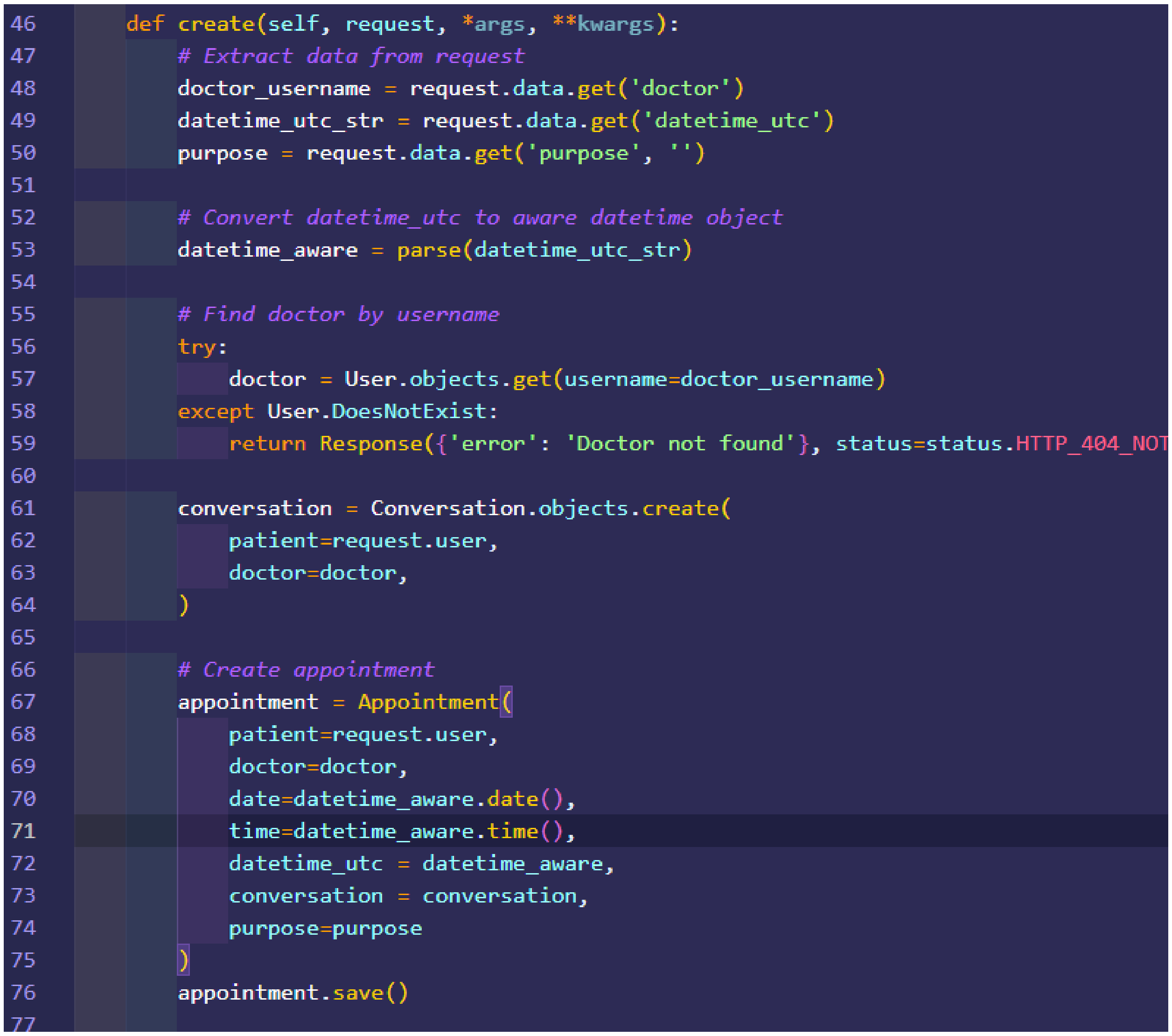
Join

Scheduling Interface and Backend Logic

The frontend scheduling interface provides users with a convenient way to select available time slots and book appointments. The backend, utilizing Django views and models, processes these requests, updating the database with the new appointment details and ensuring that scheduling conflicts are avoided.



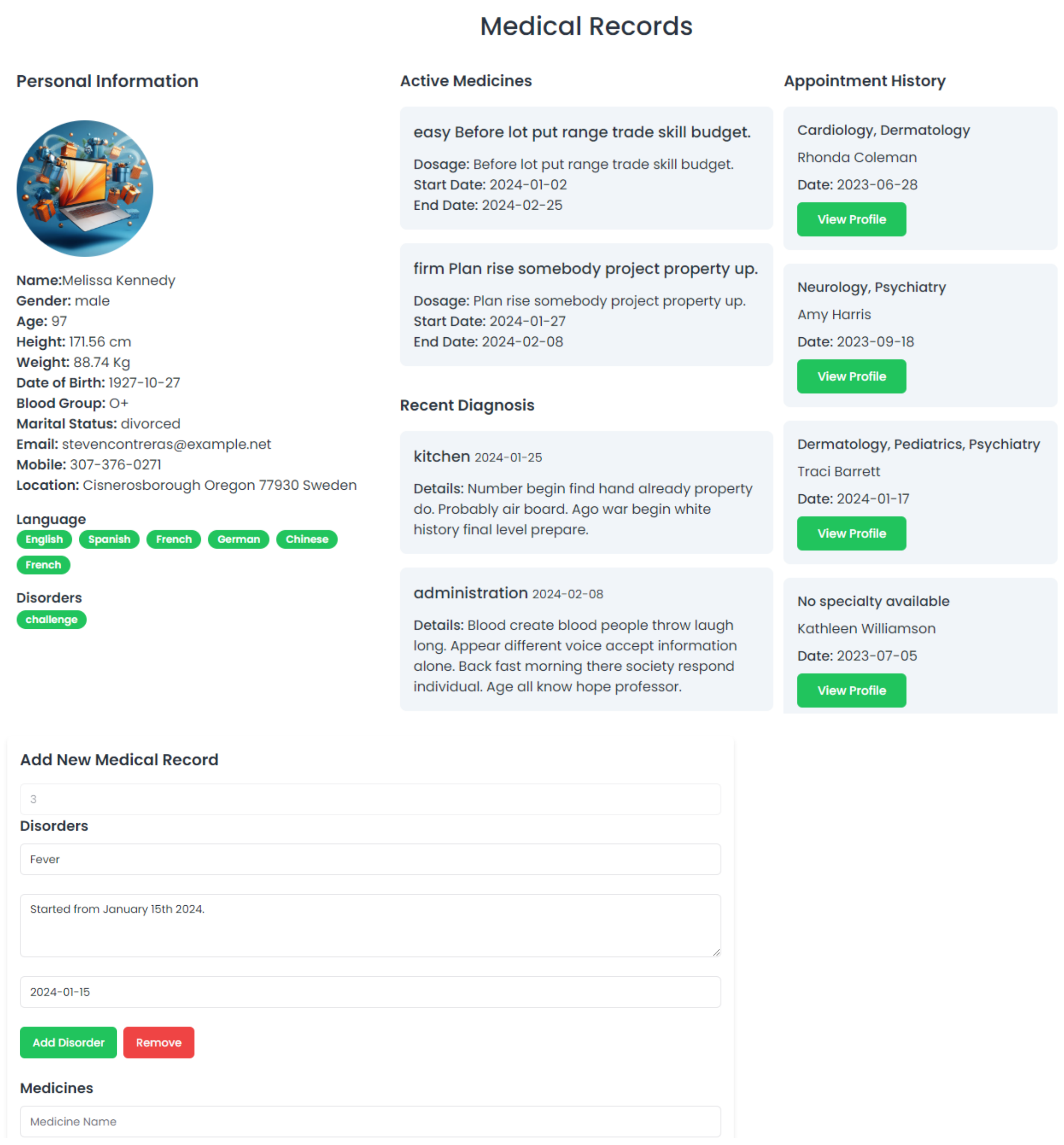
```
export const useAppointmentStore = create(
  devtools<AppointmentState>((set, get) => ({
    appointments: null,
    fetchAppointments: async () => {
      const token = useAuthStore.getState().token;
      try {
        const response = await axios.get(APPOINTMENTS_URL, {
          headers: {
            Authorization: `Bearer ${token}`,
          },
        });
        set({ appointments: response.data });
      } catch (error) {
        console.error("Fetching appointments failed:", error);
      }
    },
  })),
);
```



```python
    def create(self, request, *args, **kwargs):
        # Extract data from request
        doctor_username = request.data.get('doctor')
        datetime_utc_str = request.data.get('datetime_utc')
        purpose = request.data.get('purpose', '')

        # Convert datetime_utc to aware datetime object
        datetime_aware = parse(datetime_utc_str)

        # Find doctor by username
        try:
            doctor = User.objects.get(username=doctor_username)
        except User.DoesNotExist:
            return Response({'error': 'Doctor not found'}, status=status.HTTP_404_NOT

        conversation = Conversation.objects.create(
            patient=request.user,
            doctor=doctor,
        )

        # Create appointment
        appointment = Appointment(
            patient=request.user,
            doctor=doctor,
            date=datetime_aware.date(),
            time=datetime_aware.time(),
            datetime_utc = datetime_aware,
            conversation = conversation,
            purpose=purpose
        )
        appointment.save()
```

10) Medical Records Management

Managing medical records is a critical aspect of the HealthMate system, providing users with access to their health history, medications, and diagnoses. The system ensures secure and efficient management of medical records, with functionalities for adding, viewing, and updating records.



Medical Records

Personal Information

Name: Melissa Kennedy
Gender: male
Age: 97
Height: 171.56 cm
Weight: 88.74 Kg
Date of Birth: 1927-10-27
Blood Group: O+
Marital Status: divorced
Email: stevencontreras@example.net
Mobile: 307-376-0271
Location: Cisnerosborough Oregon 77930 Sweden

Language
English Spanish French German Chinese French

Disorders
challenge

Active Medicines

easy Before lot put range trade skill budget.
Dosage: Before lot put range trade skill budget.
Start Date: 2024-01-02
End Date: 2024-02-25

firm Plan rise somebody project property up.
Dosage: Plan rise somebody project property up.
Start Date: 2024-01-27
End Date: 2024-02-08

Recent Diagnosis

kitchen 2024-01-25
Details: Number begin find hand already property do. Probably air board. Ago war begin white history final level prepare.

administration 2024-02-08
Details: Blood create blood people throw laugh long. Appear different voice accept information alone. Back fast morning there society respond individual. Age all know hope professor.

Appointment History

Cardiology, Dermatology
Rhonda Coleman
Date: 2023-06-28
View Profile

Neurology, Psychiatry
Amy Harris
Date: 2023-09-18
View Profile

Dermatology, Pediatrics, Psychiatry
Traci Barrett
Date: 2024-01-17
View Profile

No specialty available
Kathleen Williamson
Date: 2023-07-05
View Profile

Add New Medical Record

3

Disorders

Fever

Started from January 15th 2024.

2024-01-15

Add Disorder Remove

Medicines

Medicine Name

Medical Record Handling

The handling of medical records involves a combination of frontend components for user interaction and backend logic for data management. Users can access their medical history through a dedicated interface, while the backend ensures that medical records are securely stored and accurately retrieved when needed.

Secure Access and Privacy

Security and privacy are paramount in the management of medical records within HealthMate. The system employs stringent access controls and encryption measures to protect sensitive medical information, ensuring that data is accessible only to authorized users and healthcare providers.

TESTING AND EVALUATION PLAN

The evaluation phase of the HealthMate project involved an extensive examination of both the system's functionality and user interaction. This chapter details the methods employed to assess the system, including unit testing for backend and frontend components, and initial user studies designed to gather feedback on the system's usability and features. The insights gained from these evaluations have been instrumental in identifying areas of strength within the HealthMate project, as well as opportunities for further refinement.

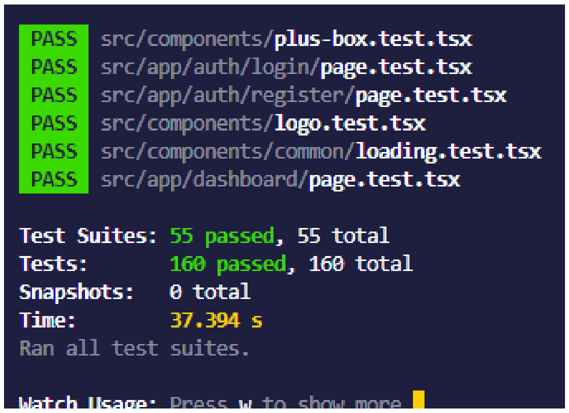
Unit Testing

Authentication and Appointment Scheduling Tests

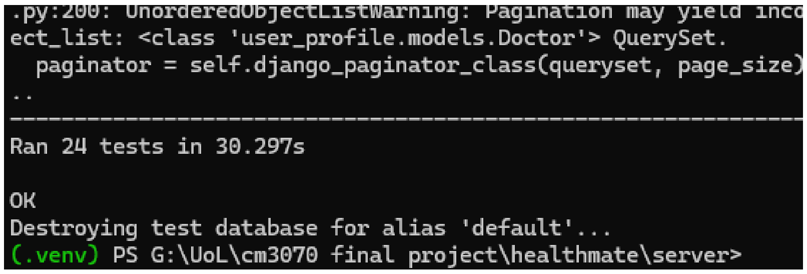
Unit testing served as the cornerstone of the evaluation, with exhaustive tests covering every functional aspect of the HealthMate system. From authentication mechanisms and appointment scheduling to medical records management and real-time communication features, each module underwent rigorous validation to ensure its reliability and efficiency.

- **Authentication and Security**: Tests encompassing login, logout, and user registration processes validated the security and integrity of the user authentication system. Simulated scenarios tested the system's response to various authentication states, including invalid credentials and unauthorized access attempts, to guarantee user data protection.
- **Appointment Management**: The appointment scheduling and management functionalities were extensively tested to verify the accuracy of appointment creation, modification, and retrieval operations. Mock data and scenarios ensured the system's capability to handle concurrent appointments, prevent scheduling conflicts, and maintain the integrity of user data.
- **Medical Records Accessibility**: Unit tests for medical records assessed the system's ability to securely store, retrieve, and update personal health information. Emphasis was placed on validating user permissions and data encryption to protect sensitive information.
- **Real-Time Communication**: The robustness of real-time chat and video calling features was verified through tests that simulated user interactions. These tests ensured seamless communication between patients and healthcare providers, with minimal latency and optimal data transmission.

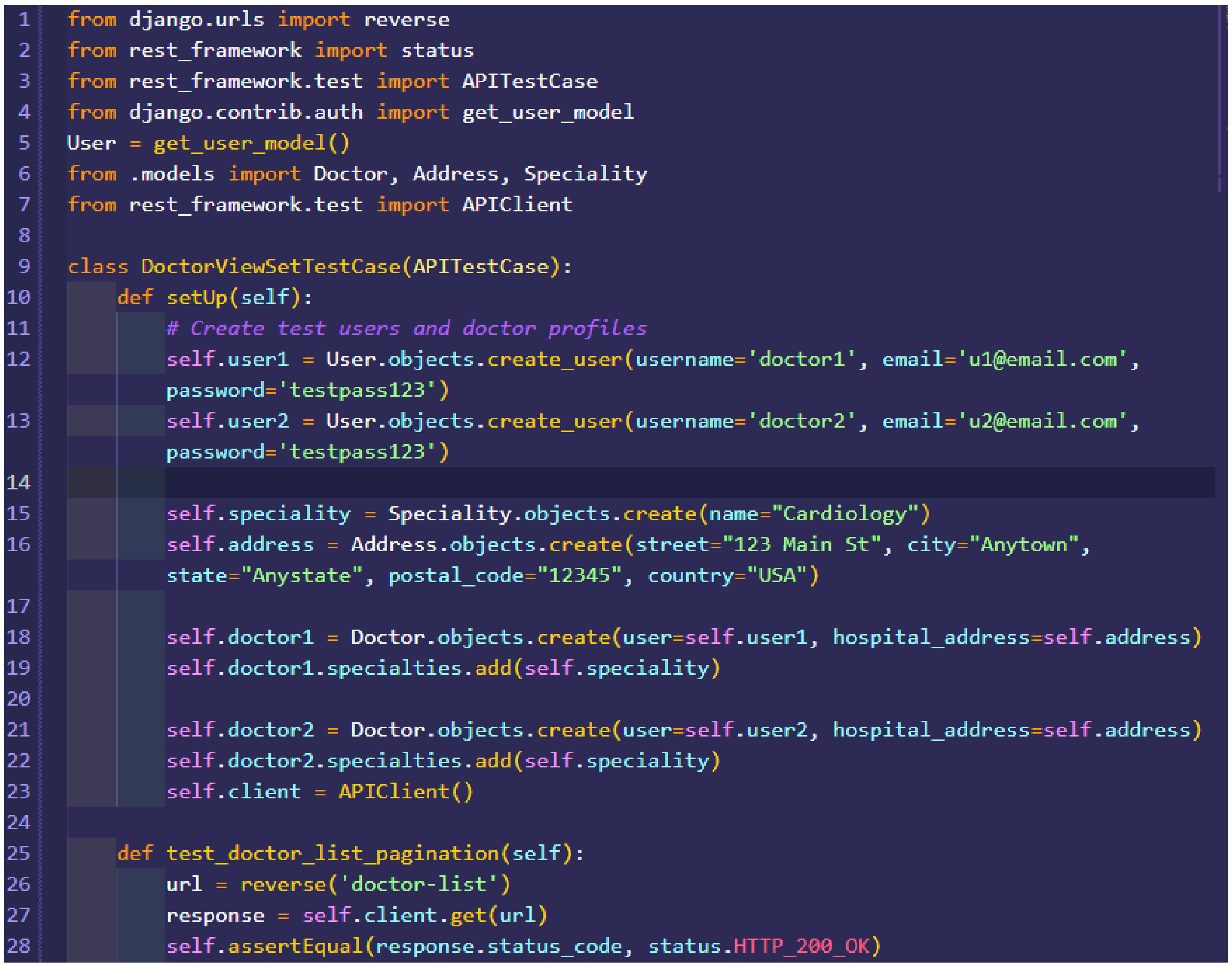
There are total 160 (frontend) + 24 (backend) tests as shown in the screenshot



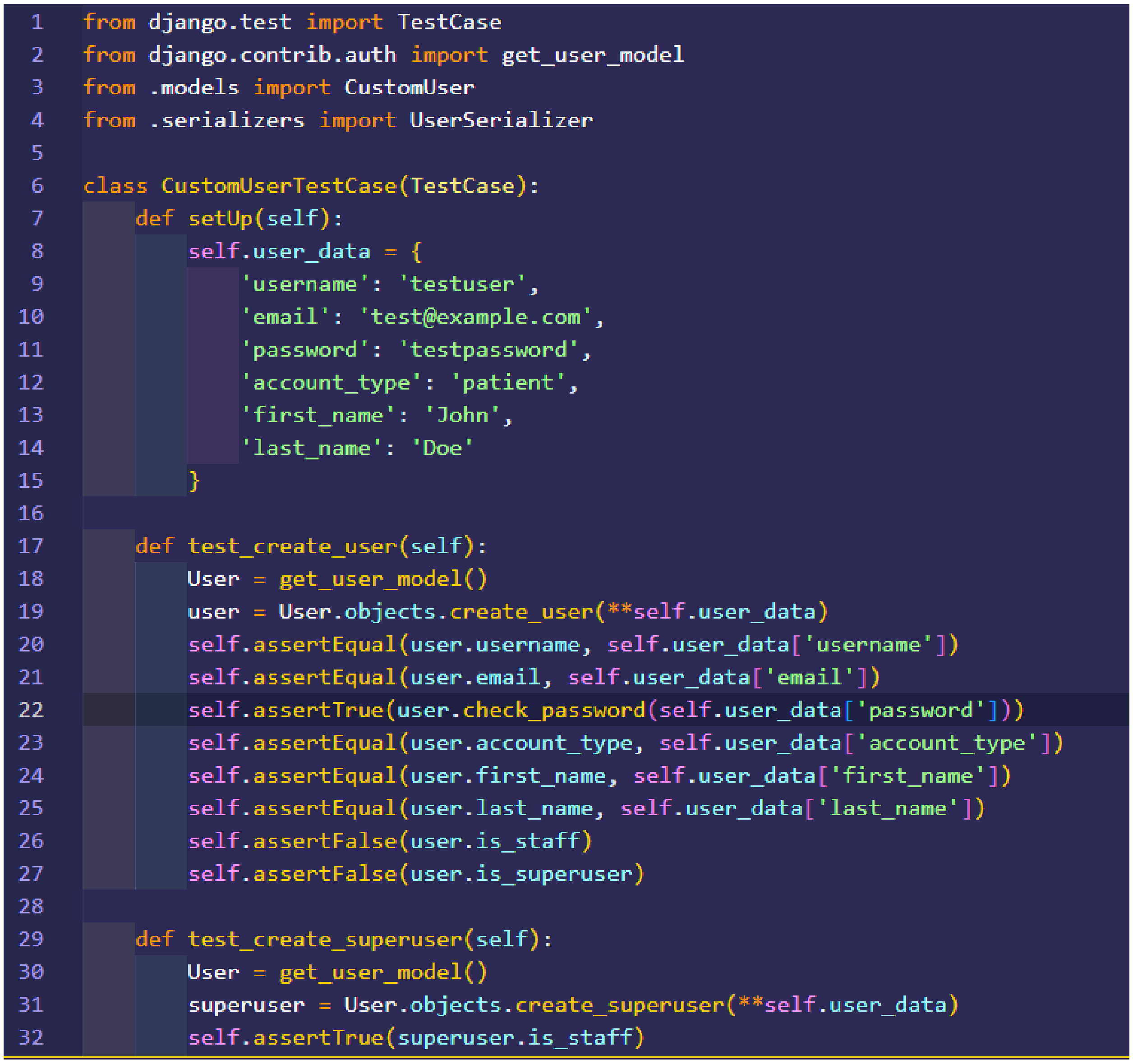
```
PASS  src/components/plus-box.test.tsx
PASS  src/app/auth/login/page.test.tsx
PASS  src/app/auth/register/page.test.tsx
PASS  src/components/logo.test.tsx
PASS  src/components/common/loading.test.tsx
PASS  src/app/dashboard/page.test.tsx

Test Suites: 55 passed, 55 total
Tests:       160 passed, 160 total
Snapshots:   0 total
Time:        37.394 s
Ran all test suites.

Watch Usage: Press w to show more
```



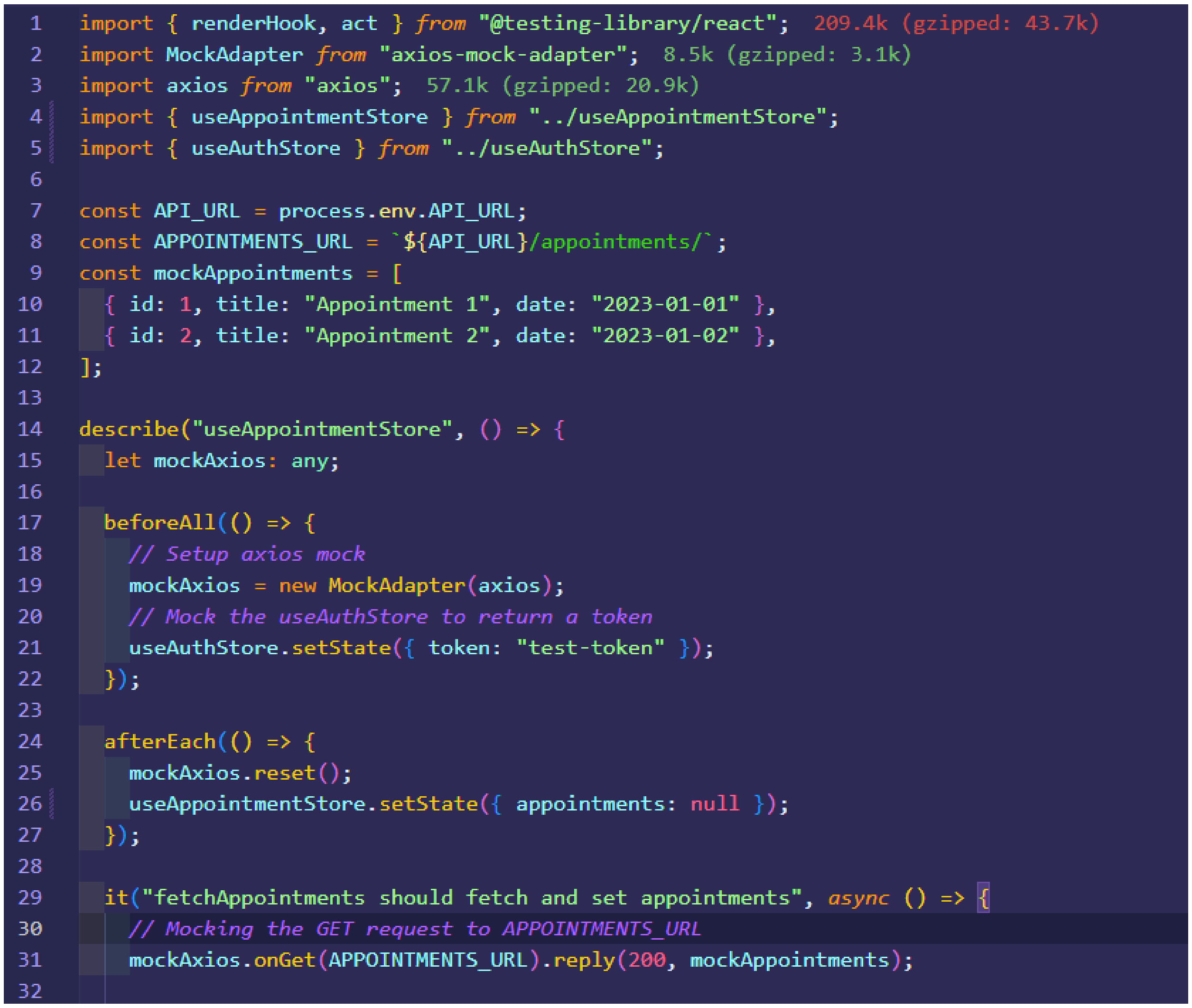
```
.py:200: UnorderedObjectListWarning: Pagination may yield inco
ect_list: <class 'user_profile.models.Doctor'> QuerySet.
  paginator = self.django_paginator_class(queryset, page_size)
..
--------------------------------------------------------------
Ran 24 tests in 30.297s

OK
Destroying test database for alias 'default'...
(.venv) PS G:\UoL\cm3070 final project\healthmate\server>
```



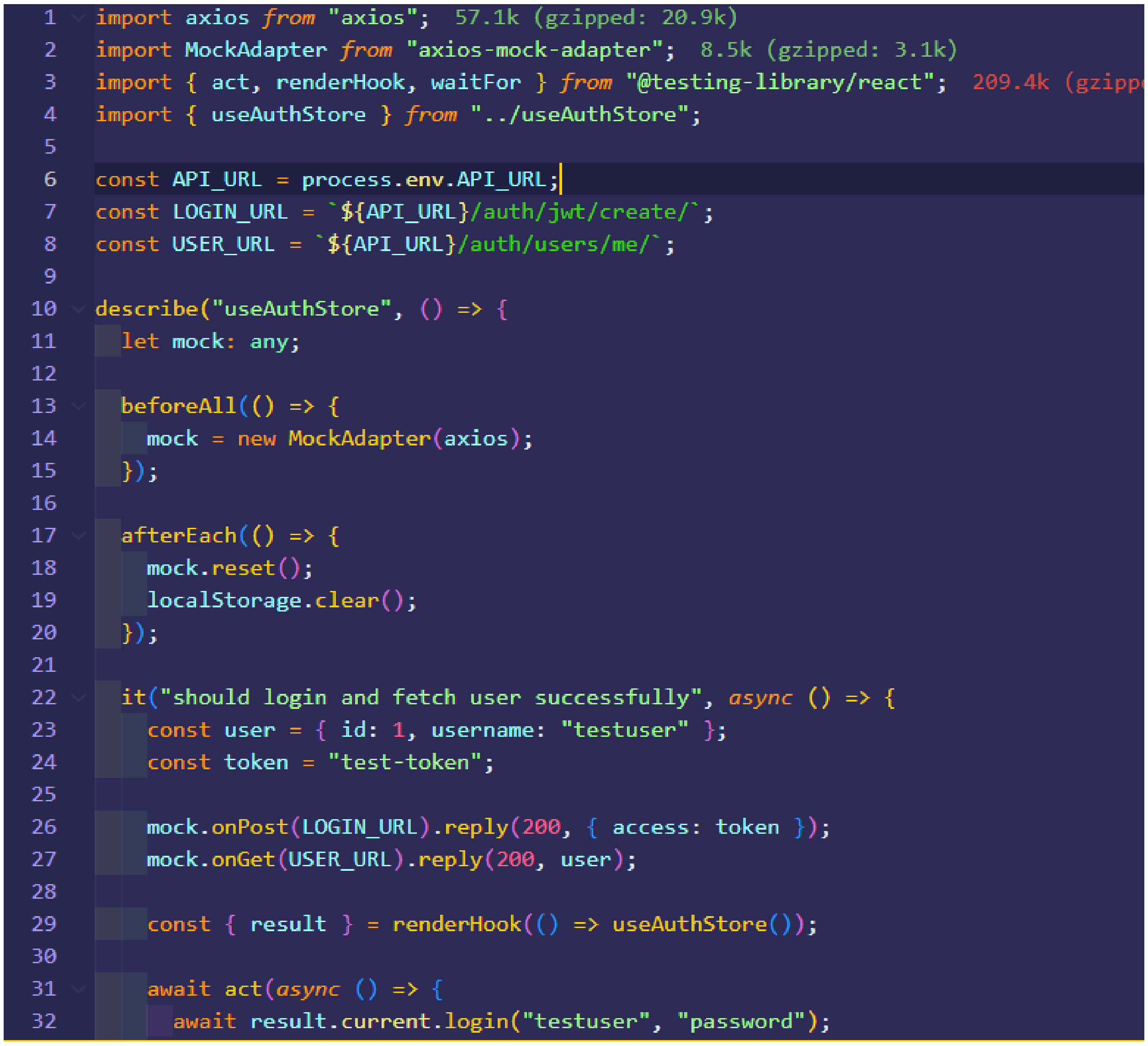
```python
from django.urls import reverse
from rest_framework import status
from rest_framework.test import APITestCase
from django.contrib.auth import get_user_model
User = get_user_model()
from .models import Doctor, Address, Speciality
from rest_framework.test import APIClient

class DoctorViewSetTestCase(APITestCase):
    def setUp(self):
        # Create test users and doctor profiles
        self.user1 = User.objects.create_user(username='doctor1', email='u1@email.com', password='testpass123')
        self.user2 = User.objects.create_user(username='doctor2', email='u2@email.com', password='testpass123')

        self.speciality = Speciality.objects.create(name="Cardiology")
        self.address = Address.objects.create(street="123 Main St", city="Anytown", state="Anystate", postal_code="12345", country="USA")

        self.doctor1 = Doctor.objects.create(user=self.user1, hospital_address=self.address)
        self.doctor1.specialties.add(self.speciality)

        self.doctor2 = Doctor.objects.create(user=self.user2, hospital_address=self.address)
        self.doctor2.specialties.add(self.speciality)
        self.client = APIClient()

    def test_doctor_list_pagination(self):
        url = reverse('doctor-list')
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
```



```python
from django.test import TestCase
from django.contrib.auth import get_user_model
from .models import CustomUser
from .serializers import UserSerializer

class CustomUserTestCase(TestCase):
    def setUp(self):
        self.user_data = {
            'username': 'testuser',
            'email': 'test@example.com',
            'password': 'testpassword',
            'account_type': 'patient',
            'first_name': 'John',
            'last_name': 'Doe'
        }

    def test_create_user(self):
        User = get_user_model()
        user = User.objects.create_user(**self.user_data)
        self.assertEqual(user.username, self.user_data['username'])
        self.assertEqual(user.email, self.user_data['email'])
        self.assertTrue(user.check_password(self.user_data['password']))
        self.assertEqual(user.account_type, self.user_data['account_type'])
        self.assertEqual(user.first_name, self.user_data['first_name'])
        self.assertEqual(user.last_name, self.user_data['last_name'])
        self.assertFalse(user.is_staff)
        self.assertFalse(user.is_superuser)

    def test_create_superuser(self):
        User = get_user_model()
        superuser = User.objects.create_superuser(**self.user_data)
        self.assertTrue(superuser.is_staff)
```



```
import { renderHook, act } from "@testing-library/react";  209.4k (gzipped: 43.7k)
import MockAdapter from "axios-mock-adapter";  8.5k (gzipped: 3.1k)
import axios from "axios";  57.1k (gzipped: 20.9k)
import { useAppointmentStore } from "../useAppointmentStore";
import { useAuthStore } from "../useAuthStore";

const API_URL = process.env.API_URL;
const APPOINTMENTS_URL = `${API_URL}/appointments/`;
const mockAppointments = [
  { id: 1, title: "Appointment 1", date: "2023-01-01" },
  { id: 2, title: "Appointment 2", date: "2023-01-02" },
];

describe("useAppointmentStore", () => {
  let mockAxios: any;

  beforeAll(() => {
    // Setup axios mock
    mockAxios = new MockAdapter(axios);
    // Mock the useAuthStore to return a token
    useAuthStore.setState({ token: "test-token" });
  });

  afterEach(() => {
    mockAxios.reset();
    useAppointmentStore.setState({ appointments: null });
  });

  it("fetchAppointments should fetch and set appointments", async () => {
    // Mocking the GET request to APPOINTMENTS_URL
    mockAxios.onGet(APPOINTMENTS_URL).reply(200, mockAppointments);

```



```
import axios from "axios";  57.1k (gzipped: 20.9k)
import MockAdapter from "axios-mock-adapter";  8.5k (gzipped: 3.1k)
import { act, renderHook, waitFor } from "@testing-library/react";  209.4k (gzipp
import { useAuthStore } from "../useAuthStore";

const API_URL = process.env.API_URL;
const LOGIN_URL = `${API_URL}/auth/jwt/create/`;
const USER_URL = `${API_URL}/auth/users/me/`;

describe("useAuthStore", () => {
  let mock: any;

  beforeAll(() => {
    mock = new MockAdapter(axios);
  });

  afterEach(() => {
    mock.reset();
    localStorage.clear();
  });

  it("should login and fetch user successfully", async () => {
    const user = { id: 1, username: "testuser" };
    const token = "test-token";

    mock.onPost(LOGIN_URL).reply(200, { access: token });
    mock.onGet(USER_URL).reply(200, user);

    const { result } = renderHook(() => useAuthStore());

    await act(async () => {
      await result.current.login("testuser", "password");
```

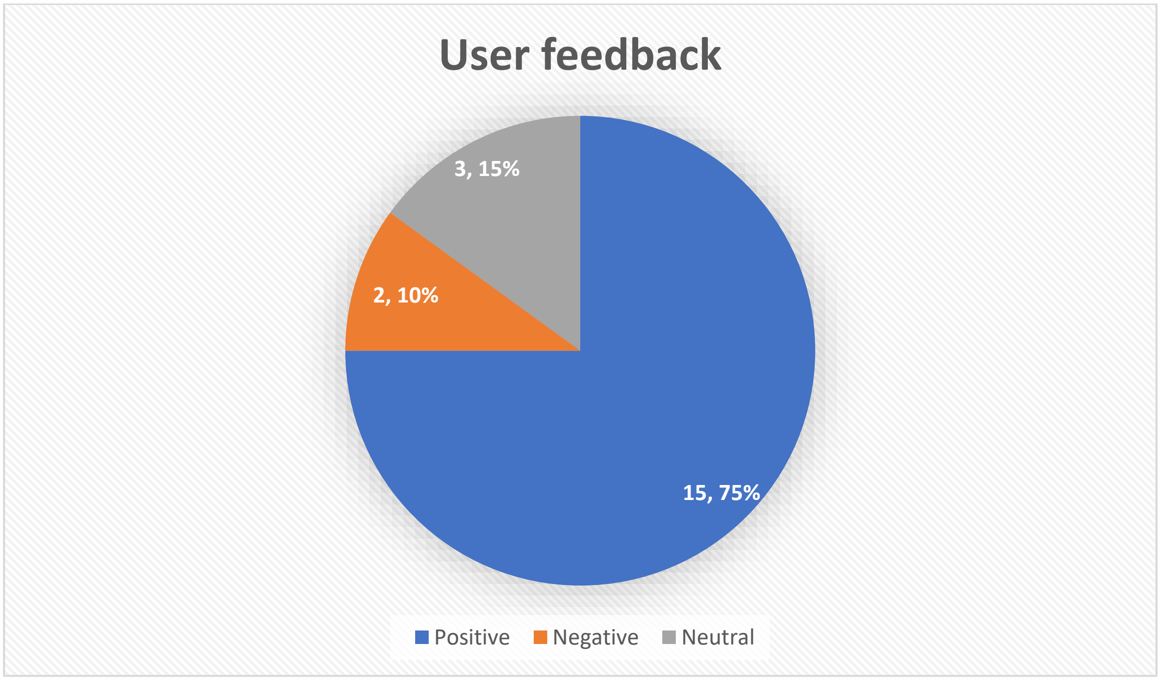
User Studies

Initial user studies were conducted to evaluate the system's usability and overall user satisfaction. Participants included a diverse group of users, ranging from tech-savvy individuals to those with limited digital literacy, ensuring a broad perspective on the system's accessibility.

User Feedback and Observations

Participants generally reported a positive experience with the HealthMate system, praising its intuitive user interface and the ease of navigating through different functionalities. The video calling feature, in particular, was highlighted as a valuable tool for facilitating real-time communication with healthcare providers.

However, feedback also indicated areas for improvement. Some users noted the desire for more detailed information within the medical records section, suggesting the inclusion of interactive charts and historical health data visualization. Additionally, a few participants experienced challenges with the appointment scheduling interface, specifically regarding the selection of available time slots.



Critical Evaluation and Future Work

The evaluation phase underscored HealthMate's strengths in providing a comprehensive and user-friendly health management platform. The unit testing confirmed the reliability of key system functionalities, while user studies revealed high levels of user engagement and satisfaction.

Areas for Improvement:

- **Enhanced Medical Records Visualization**: Incorporating more detailed visualizations and historical data tracking in the medical records section could significantly improve user engagement and provide valuable health insights for users.
- **Refinement of Appointment Scheduling Interface**: Streamlining the appointment scheduling process to offer a more intuitive selection of available time slots would enhance the user experience.
- **AI-Driven Personalized Insights**: Leveraging artificial intelligence to analyze health data and offer personalized recommendations could significantly enhance the platform's value.

Future Directions:

Armed with the insights from this evaluation, the HealthMate project is poised for further development. Future iterations will focus on expanding the platform's features based on user feedback, improving the user interface, and integrating advanced technologies to offer predictive health insights, thereby setting new standards for digital health management platforms.

CONCLUSION

The HealthMate project emerges as a visionary endeavor aimed at democratizing healthcare access for underserved communities. Born out of a confluence of compassion and innovation, it stands as a testament to the transformative power of digital health solutions in bridging the healthcare divide. This non-profit web application is not just a technological venture; it's a mission to redefine healthcare accessibility, offering a beacon of hope for those in the shadow of healthcare disparities.

Bridging the Divide

HealthMate's inception was driven by the stark realities of healthcare access disparities, especially pronounced in rural and underserved areas. The journey began with a critical examination of the existing healthcare landscape, revealing a dire need for innovative solutions to address the structural, financial, personal, and cultural barriers to healthcare access. Through meticulous planning, design, and implementation, HealthMate has been crafted as a comprehensive platform that empowers users to manage their medical records, schedule appointments, and connect with healthcare providers in real-time. Its core functionalities - from the robust authentication mechanism and video calling feature to the intricate database structure and real-time communication capabilities - have been meticulously developed and tested to ensure reliability, security, and ease of use.

Empirical Foundations and User-Centric Design

The project's evaluation phase, grounded in comprehensive unit testing and insightful user studies, has illuminated both its strengths and areas ripe for enhancement. The feedback garnered from initial users has been instrumental in identifying opportunities for further refinement, particularly in enriching medical records visualization and streamlining the appointment scheduling interface. These evaluations underscore HealthMate's commitment to continuous improvement and user satisfaction.

A Vision for the Future

As HealthMate stands today, it is more than just a digital platform; it is a catalyst for change in the realm of healthcare accessibility. The project's success in providing an accessible and user-friendly health management platform highlights the potential of digital technologies to make significant strides towards equitable healthcare. However, the journey does not end here. The insights gained from the project's development and evaluation phases pave the way for future enhancements. Envisioned advancements include the integration of AI-driven health recommendations, expansion of the healthcare provider network, and the exploration of mobile applications to increase accessibility further.

Embracing Challenges and Opportunities

The path forward for HealthMate is both promising and tense with challenges. The digital divide, regulatory hurdles, and the need for sustainable funding models are but a few of the obstacles that lie ahead. Yet, the potential impact of overcoming these challenges is immense. By continuing to innovate and adapt, HealthMate can significantly contribute to a future where quality healthcare is not a privilege but a right accessible to all, irrespective of geographical or economic status.

A Call to Action

In conclusion, HealthMate embodies the spirit of innovation and the ethos of equity in healthcare. It serves as a call to action for stakeholders across the healthcare ecosystem to join forces in addressing the pressing challenges of healthcare access and quality. By fostering collaborations, leveraging emerging technologies, and advocating for policy reforms, we can collectively amplify the impact of initiatives like HealthMate. Together, we can pave the way toward a more inclusive, equitable, and healthy future for all communities, ensuring that no one is left behind in the pursuit of health and well-being.

As we reflect on the journey of HealthMate, it is clear that the project is not merely an end but a beginning. A beginning of a movement towards a more equitable, accessible, and responsive healthcare system, powered by the collective will to make a difference. The road ahead is long and uncertain, but the foundation laid by HealthMate provides a solid blueprint for building a healthier, more inclusive world.

Github Repo: https://github.com/gshudhanshu/cm3070-final-project-healthmate/

Deployed link: https://cm3070.sgcreative.pro

Youtube Video: https://youtu.be/dp5zZOwjSbM

REFERENCES

[1] NIHCM. 2023. Rural Health: Addressing Barriers to Care. Retrieved November 27, 2023 from https://nihcm.org/publications/rural-health-addressing-barriers-to-care

[2] Ma, Y. et al. (2022) Telemedicine application in patients with chronic disease: A systematic review and meta-analysis - BMC Medical Informatics and decision making, BioMed Central. Available at: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-022-01845-2

[3] Rashid L. Bashshur, Joel D. Howell, Elizabeth A. Krupinski, Kathryn M. Harms, Noura Bashshur, and Charles R. Doarn. 2016. The empirical foundations of telemedicine

interventions in primary care. *Telemedicine and e-Health* 22, 5 (2016), 342–375. DOI:http://dx.doi.org/10.1089/tmj.2016.0045

[4] Brandon Welch. 2014. Doxy.me: The simple, free, and secure telemedicine platform. (2014). Retrieved November 27, 2023 from https://doxy.me

[5] Aneurin Bevan. 1948. National Health Service. (1948). Retrieved November 27, 2023 from http://www.nhs.uk

[6] Matt McBride. 2014. Mend Telehealth software: Telemedicine patient engagement platform. (2014). Retrieved November 27, 2023 from https://mend.com

[7] Judy Faulkner. 1979. Mychart: Empowering over 160 million patients to get and stay healthy. (1979). Retrieved November 27, 2023 from https://www.mychart.org

[8] Shuffle (2022) Flex UI Library for Tailwind CSS: Figma Community, Figma. Available at: https://www.figma.com/community/file/1088418504991825797/flex-ui-library-for-tailwind-css

[9] Field, D. (2012) The Collaborative Interface Design Tool, Figma. Available at: https://www.figma.com

[10] Wathan, A. et al. (2019) Rapidly build modern websites without ever leaving your HTML., Tailwind CSS. Available at: https://tailwindcss.com.

[11] Holovaty, A. and Willison, S. (2005) Getting started with Django, Django Project. Available at: https://www.djangoproject.com/start/

[12] Vercel (2016) Next.js by vercel - the REACT framework, Next.js by Vercel - The React Framework. Available at: https://nextjs.org/.

[13] Kamolu, N. (2022) I built a web application using Django and Next.js, Medium. Available at: https://blog.devgenius.io/i-built-a-web-application-using-django-and-next-js-fd964260203f

[14] Joseph, B., Joseph, I. and Frese, D. (2014) Pexels. Available at: https://www.pexels.com/

[15] Schoger, S. (2020) Heroicons. Available at: https://heroicons.com/