

Vellore Institute of Technology School of Computer Science and Engineering

**Software Engineering**

**Course Code:** BCSE301L

**Class Number: SJT 604 Slot: G2 +TG2**

**PROJECT ON**

GOLDEN YEARS HUB

**ACKNOWLEDGEMENT**

I would like to express my deep appreciation and heartfelt gratitude to my supervisor, Prof.Yoganand S., from VIT University. Without his unwavering motivation and continuous encouragement, this research would not have been completed successfully.

I am sincerely grateful to the Chancellor of VIT University, Dr. G Viswanathan, the Vice Presidents, and the Vice Chancellor for their inspiring guidance and support. Their encouragement and provision of essential infrastructure and resources were instrumental in the progress of my research.

I would also like to extend my sincere thanks to Dr. N. Jaisankar, the Dean of the School of Computer Science and Engineering (SCOPE) at VIT University, for his kind words of support and encouragement throughout my research journey. Additionally, I am grateful to my ‘HOD’s’ and classmates for their support in various ways throughout my research work.

Furthermore, I wish to express my profound gratitude to my parents for their enduring sacrifices during my research. Their unwavering support and encouragement have been invaluable to me.

I am truly grateful to all those mentioned above for their contributions to my research, and I feel incredibly fortunate to have had their guidance and support throughout this endeavor.

Place – Vellore Date-

**Team Members**

**1.L PRANAY**

**2.S ATIF**

**3.M VARUN TEJA**

Signature of Supervisor

**Table of Contents:**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENTS** | **PAGE NUMBER** |
| **1.** | **INTRODUCTION** | 3 |
| **2.** | **ABSTRACT** | 4 |
| **3.** | **SRS** | 5 |
| **4.** | **DESIGN** | 6 |
| **5.** | **IMPLEMENTATION (CODE + SCREEN SHOTS)** | 11 |
| **6.** | **TESTING** | 20 |
| **7.** | **CONCLUSION** | 22 |
| **8.** | **REFERENCES** | 23 |

**INTRODUCTION:**

Elderly individuals living in old age homes often experience loneliness and social isolation, making it essential to provide them with digital tools for communication and engagement. **Elderly Connect** is a user-friendly web platform designed to help seniors stay connected with their families, caregivers, and peers.

The platform includes **video calling, health monitoring, medication reminders, interactive entertainment, and AI companionship**, ensuring that seniors can maintain social interactions and receive essential care. The interface is designed with **large fonts, simple navigation, and voice-assisted features**, making it accessible for elderly users with minimal technical knowledge.

By integrating technology into elderly care, **Elderly Connect** enhances the quality of life for senior citizens, providing them with companionship, security, and a sense of belonging in an increasingly digital world.

**ABSTRACT:**

**Elderly Connect is a digital platform designed to help senior citizens in old age homes stay connected with their families and caregivers. The system provides video calling, real-time health monitoring, medication reminders, social forums, and entertainment features to enhance the well-being of elderly individuals.**

**With an emphasis on ease of use and accessibility, the platform includes large fonts, voice assistance, and simple navigation to accommodate users with limited technical knowledge. Additionally, an AI-powered virtual companion provides engagement and emotional support, reducing loneliness and improving mental health.**

**The project aims to create a safe, interactive, and engaging environment where seniors can maintain social connections and receive essential care through technology. By integrating modern solutions with elderly-friendly design, Elderly Connect serves as a comprehensive digital companion for improving the lives of older adults.**

**SRS**

1. **Functional Requirements:**

✔ User Authentication – Secure login system for residents, caregivers, and family members.

✔ Video Calling – One-click video call feature to connect with family members and caregivers.

✔ Health Monitoring – Tracks vital signs like heart rate and blood pressure (if integrated with smart devices).

✔ Medication & Appointment Reminders – Automated notifications for medication schedules and doctor visits.

✔ Social Forum – A platform for seniors to interact with each other and share experiences.

✔ AI Chatbot Assistance – Virtual assistant for companionship and answering basic queries.

✔ Entertainment Module – Access to music, books, games, and religious content for engagement.

1. **Non-Functional Requirements:**

✔ Ease of Use – Simple and intuitive interface with large fonts and voice assistance.

✔ Security & Privacy – End-to-end encryption for calls and personal data protection.

✔ Scalability – The platform should support multiple users without performance issues.

✔ Cross-Platform Compatibility – Works on mobile, tablet, and desktop devices.

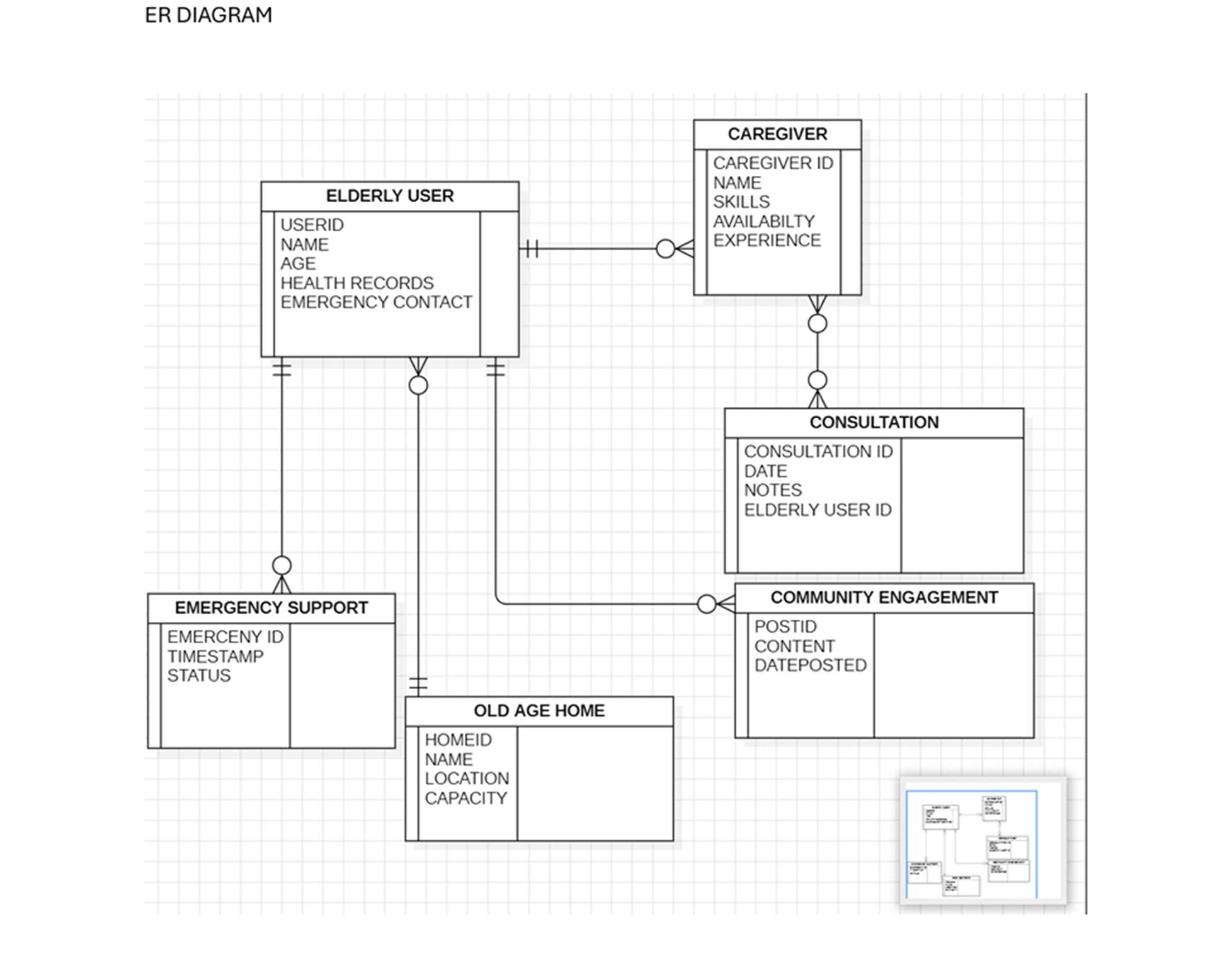
✔ Performance Efficiency – Quick response times for seamless user experience.

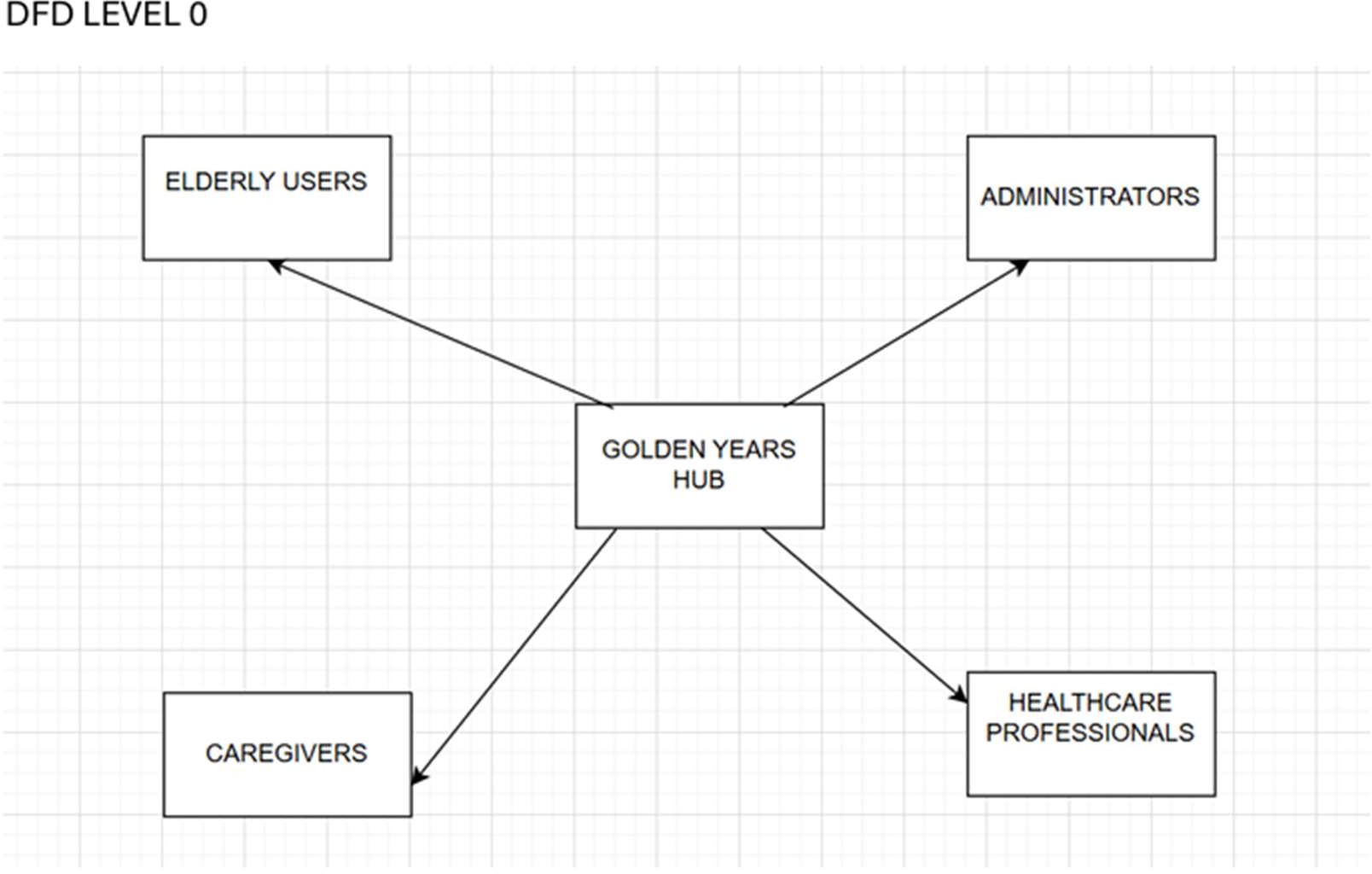
1. **Constraints:**

✔ Internet connectivity is required for real-time communication features.

✔ Smart devices (like wearables) are needed for advanced health tracking.

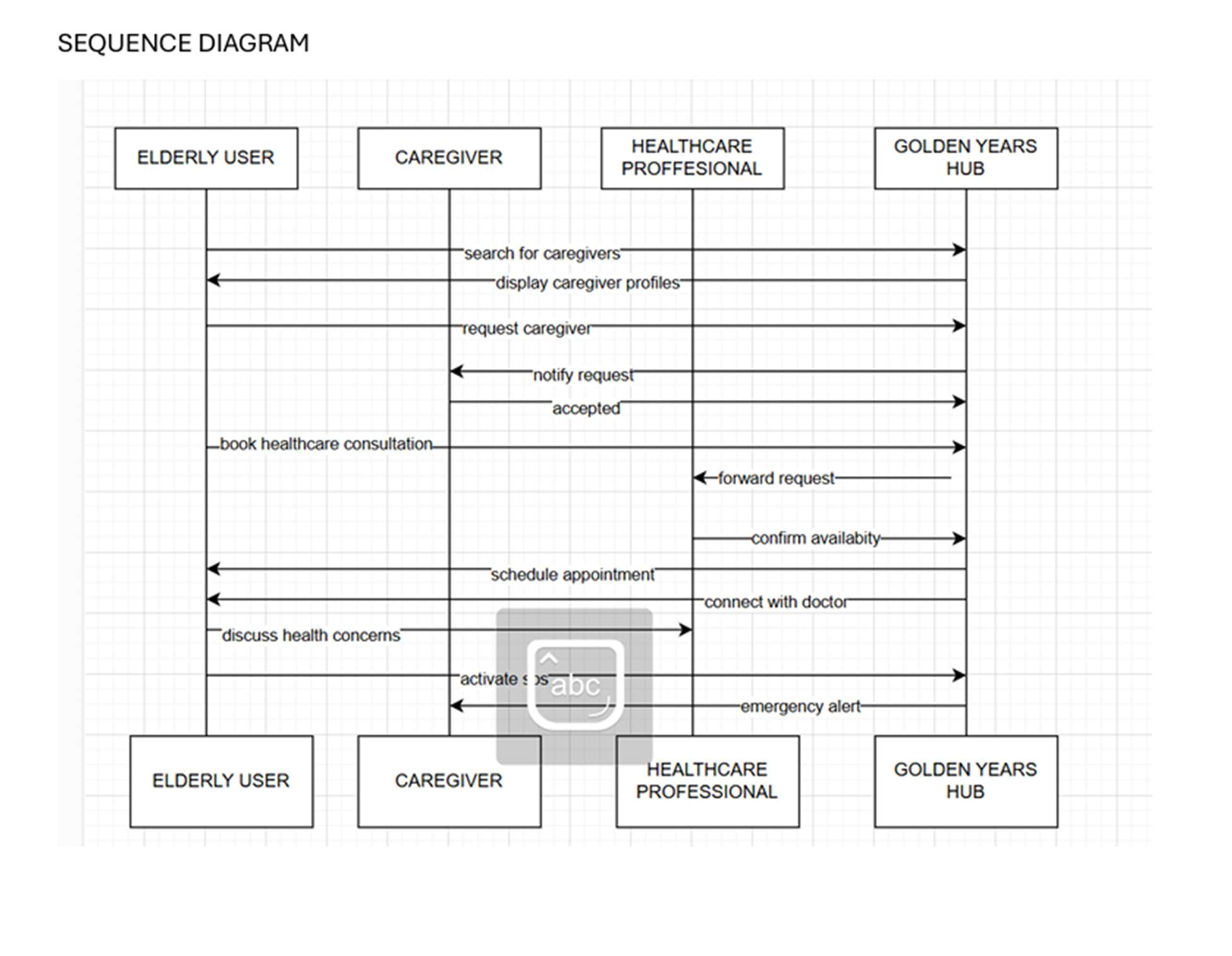
**Design**

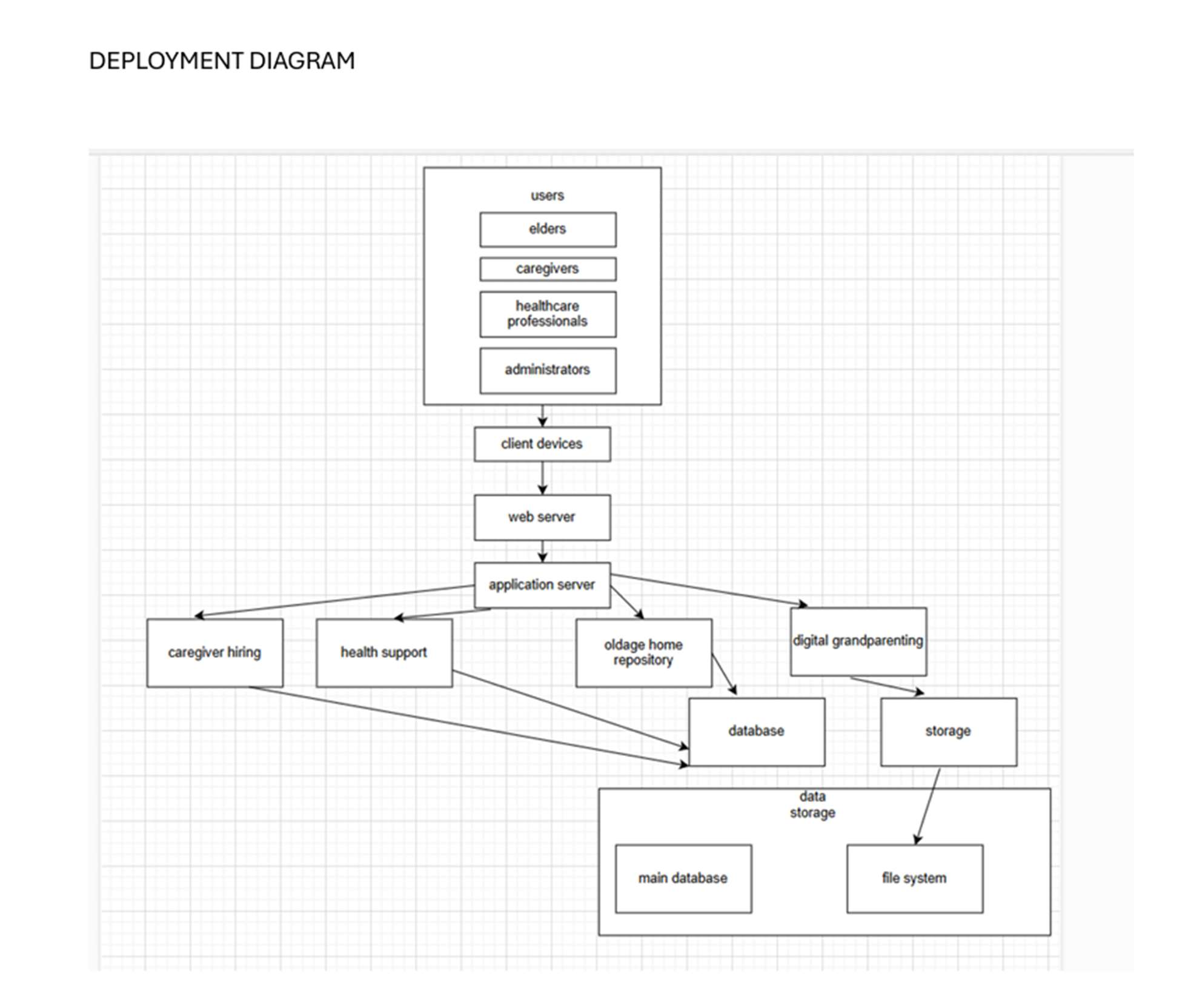


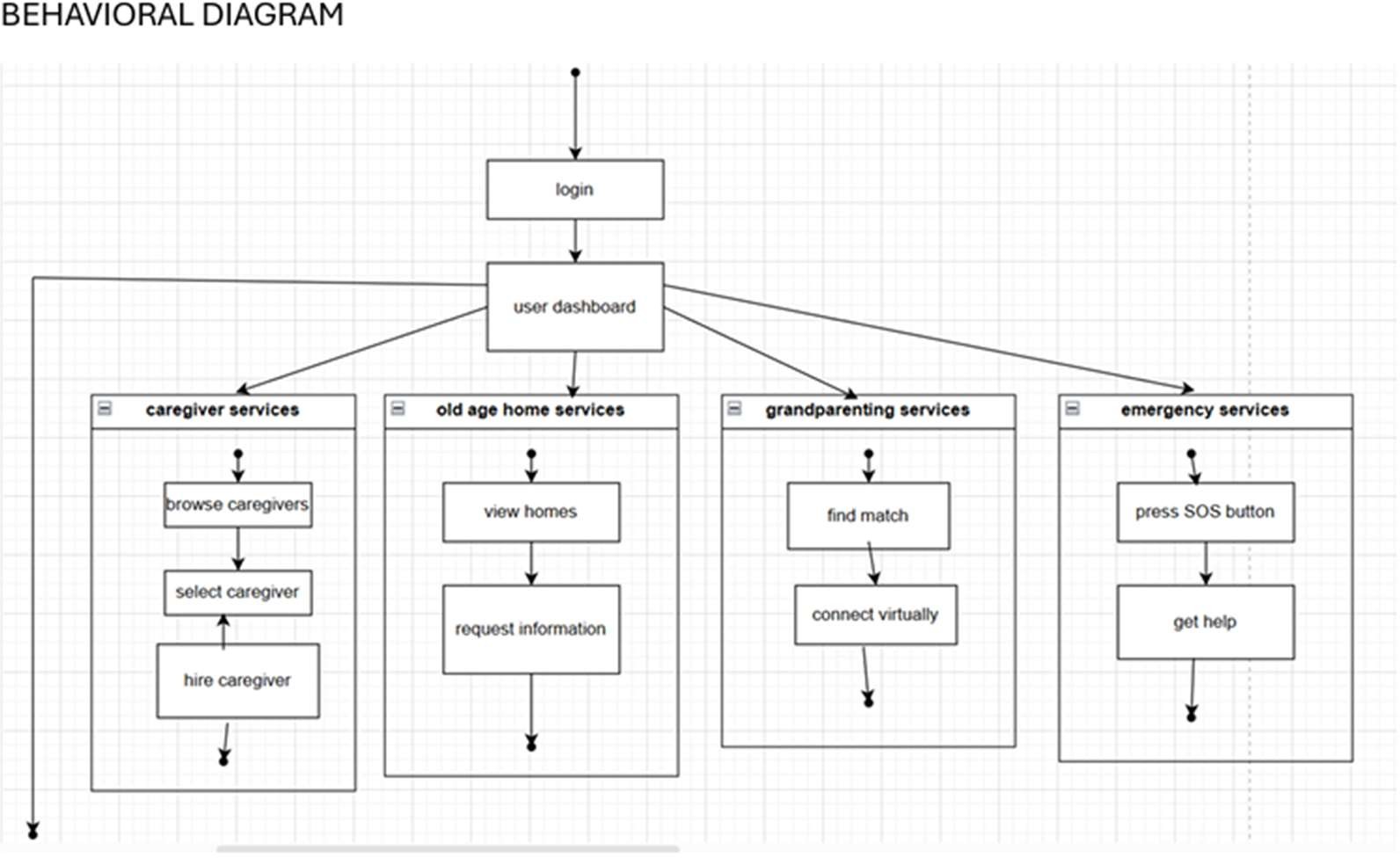


A diagram of a health care support

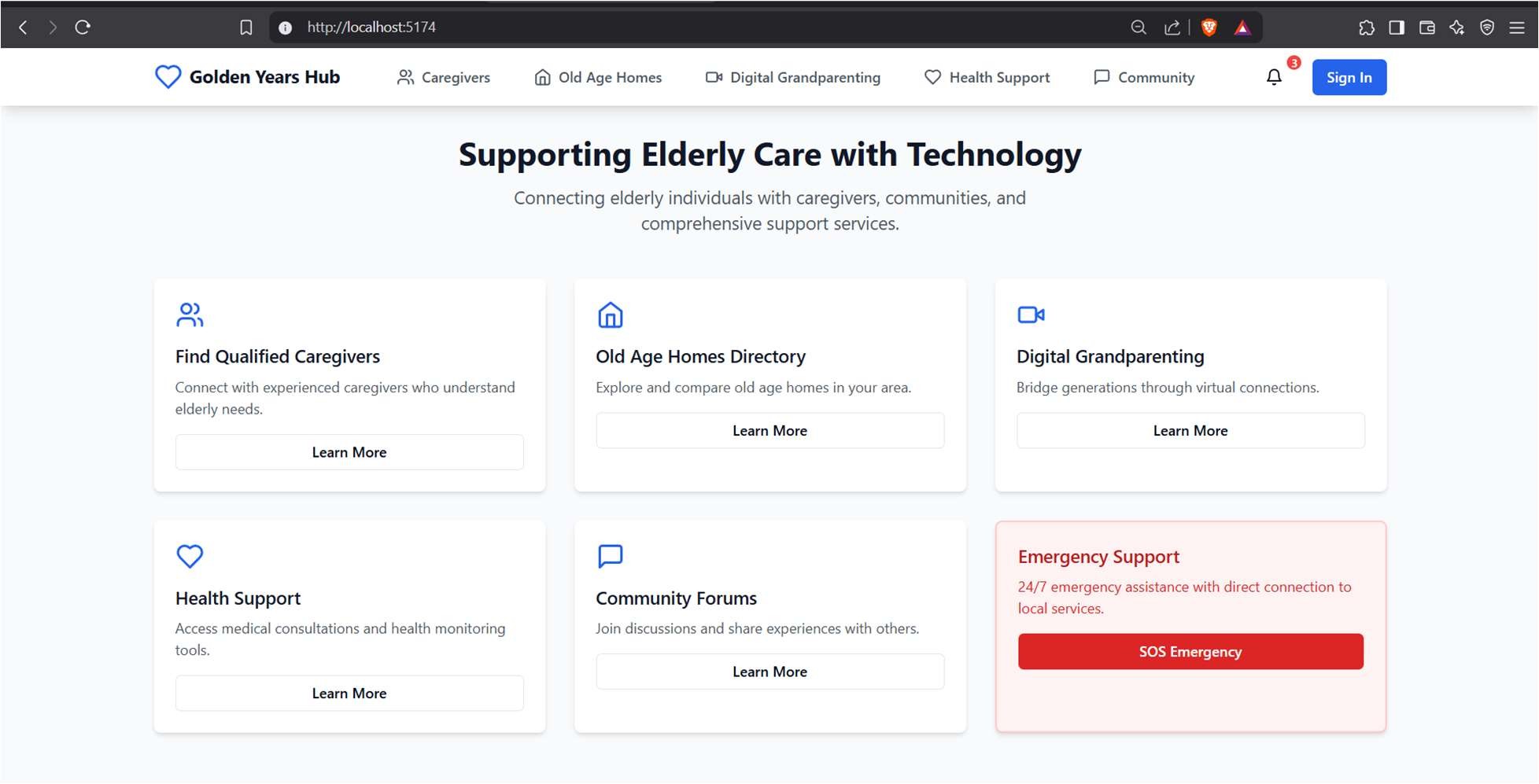
AI-generated content may be incorrect.







**UI/UX:**



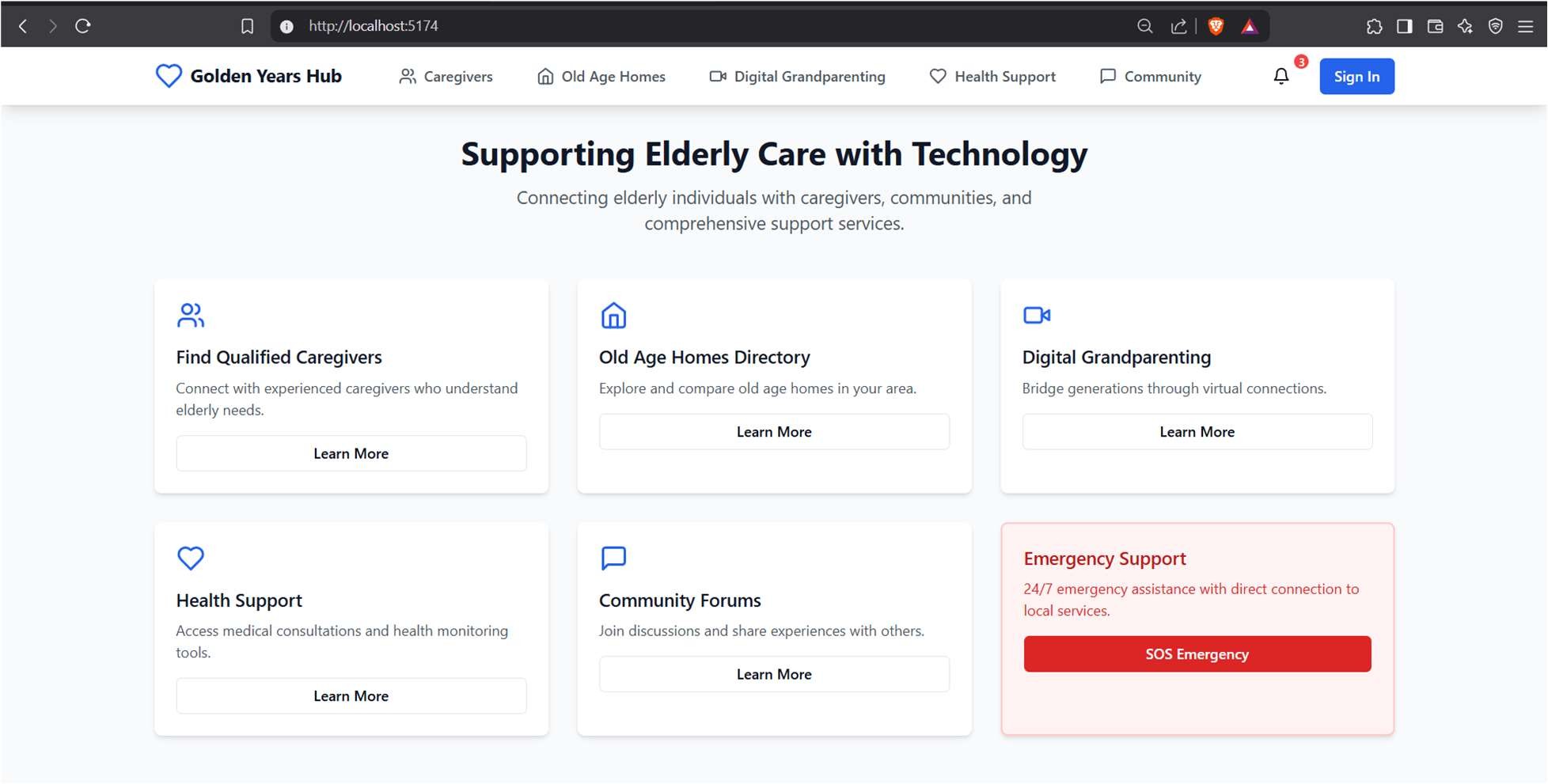
**ALGORITHM – CODE:**

Algorithm for "Golden Years Hub" React Website Development

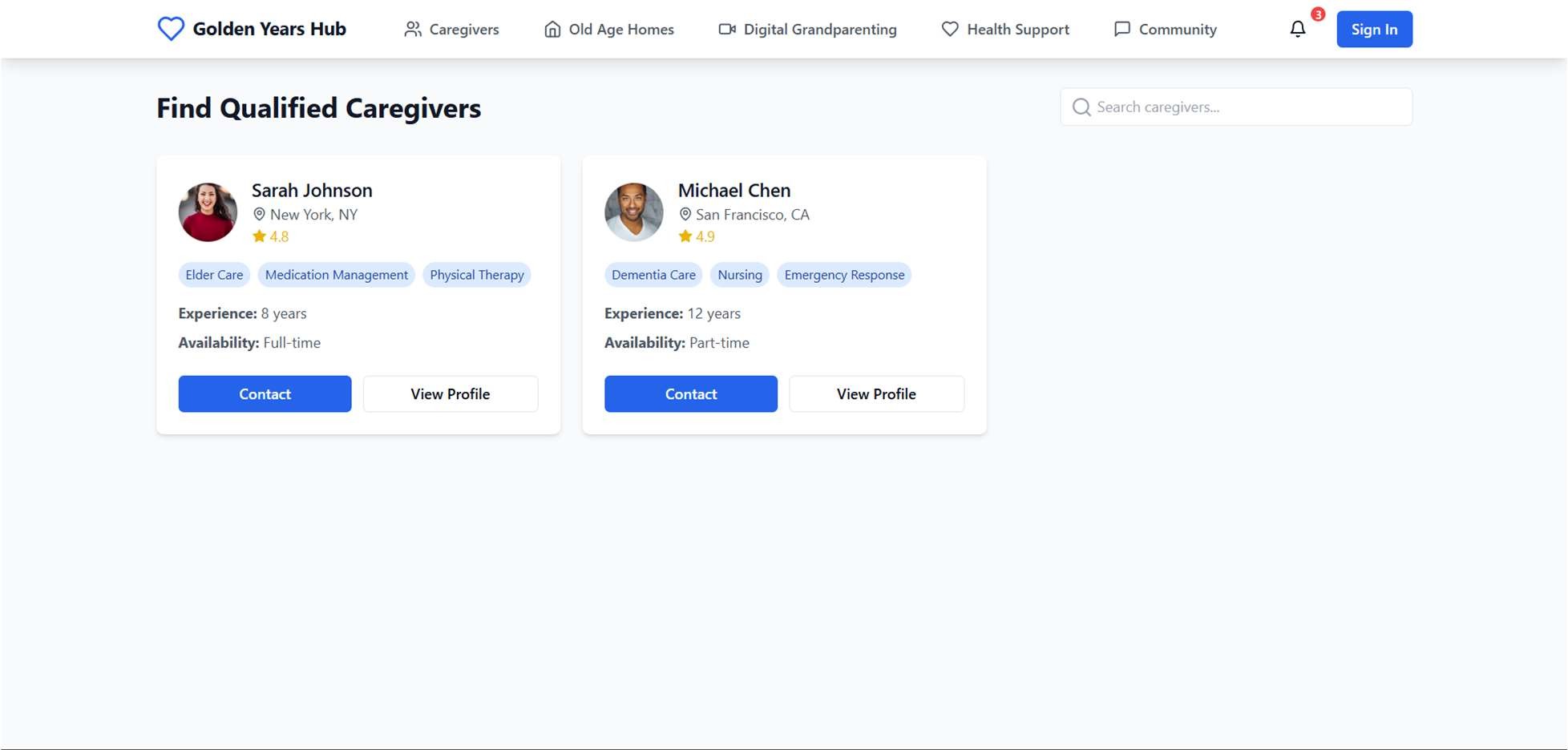
1. **Initialize the Project**
   * **Create a new React project using Vite or Next.js.**
   * **Install necessary dependencies: TailwindCSS, shadcn/ui, React Router, Axios, and Zustand for state management.**
2. **Set Up Project Structure**
   * **Organize the project with folders:**
     + **/components → Reusable UI components**
     + **/pages → Main application pages**
     + **/api → Backend API calls**
     + **/lib → Utility functions**
3. **Implement Navigation**
   * **Create a Navbar component with links to different sections:**
     + **Caregivers**
     + **Old Age Homes**
     + **Digital Grandparenting**
     + **Health Support**
     + **Community**
   * **Add a Sign-In button and notifications icon.**
4. **Design Homepage**
   * **Create a hero section with a heading and subheading.**
   * **Display feature cards for different services in a grid layout.**
   * **Highlight an "Emergency Support" section with an SOS button.**
5. **Develop Feature Pages**
   * **Caregivers Page: Display profiles of available caregivers.**
   * **Old Age Homes Page:**
     + **Connect to MongoDB.**
     + **Provide a form to add new old age homes to the database.**
     + **List existing old age homes.**
   * **Digital Grandparenting Page:**
     + **Implement a video calling feature using the ZegoCloud API.**
   * **Health Support Page: Offer medical consultation options.**
   * **Community Page: Enable discussions and forums.**
6. **Handle API Integrations**
   * **Set up a backend API using Express.js and MongoDB.**
7. **Ensure Responsive Design**
   * **Use Tailwind CSS for styling.**
   * **Make components mobile-friendly.**
8. **Deploy the Application**
   * **Build and optimize the app using npm run build.**
   * **Deploy on Vercel.**

IMPLEMENTATION: (it should include all functionalities/modules)

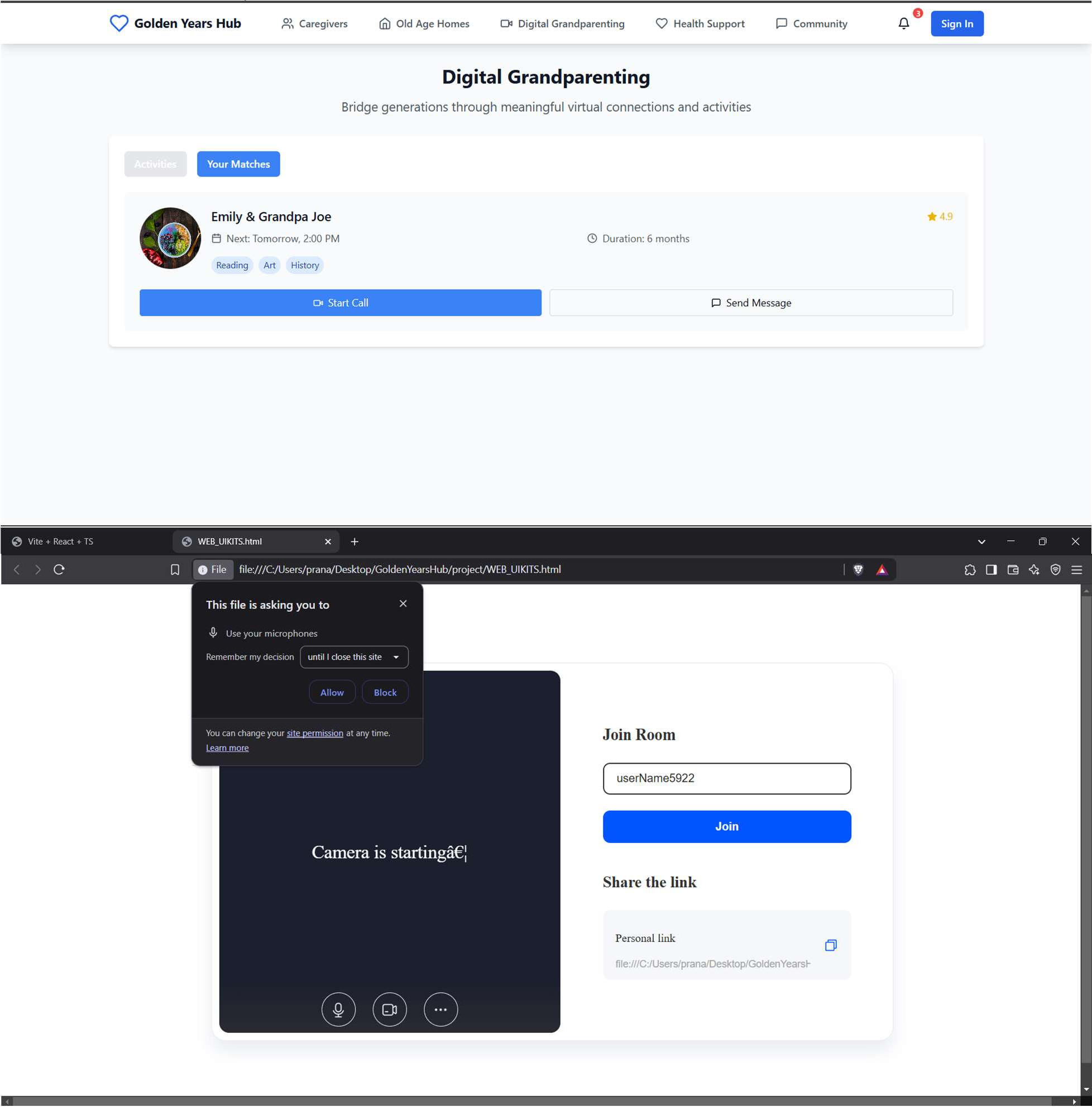
**HOME PAGE**

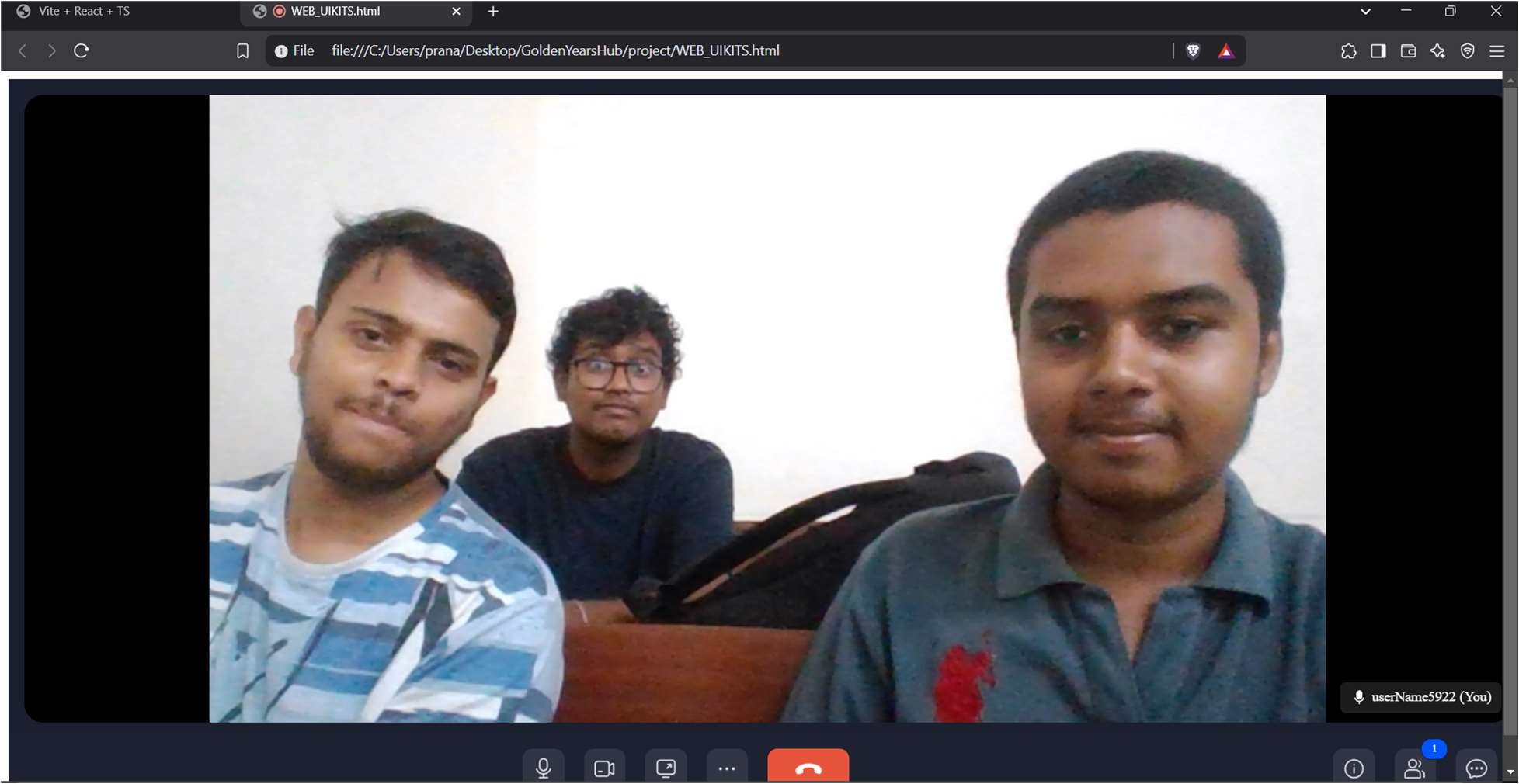


**CAREGIVERS PAGE**



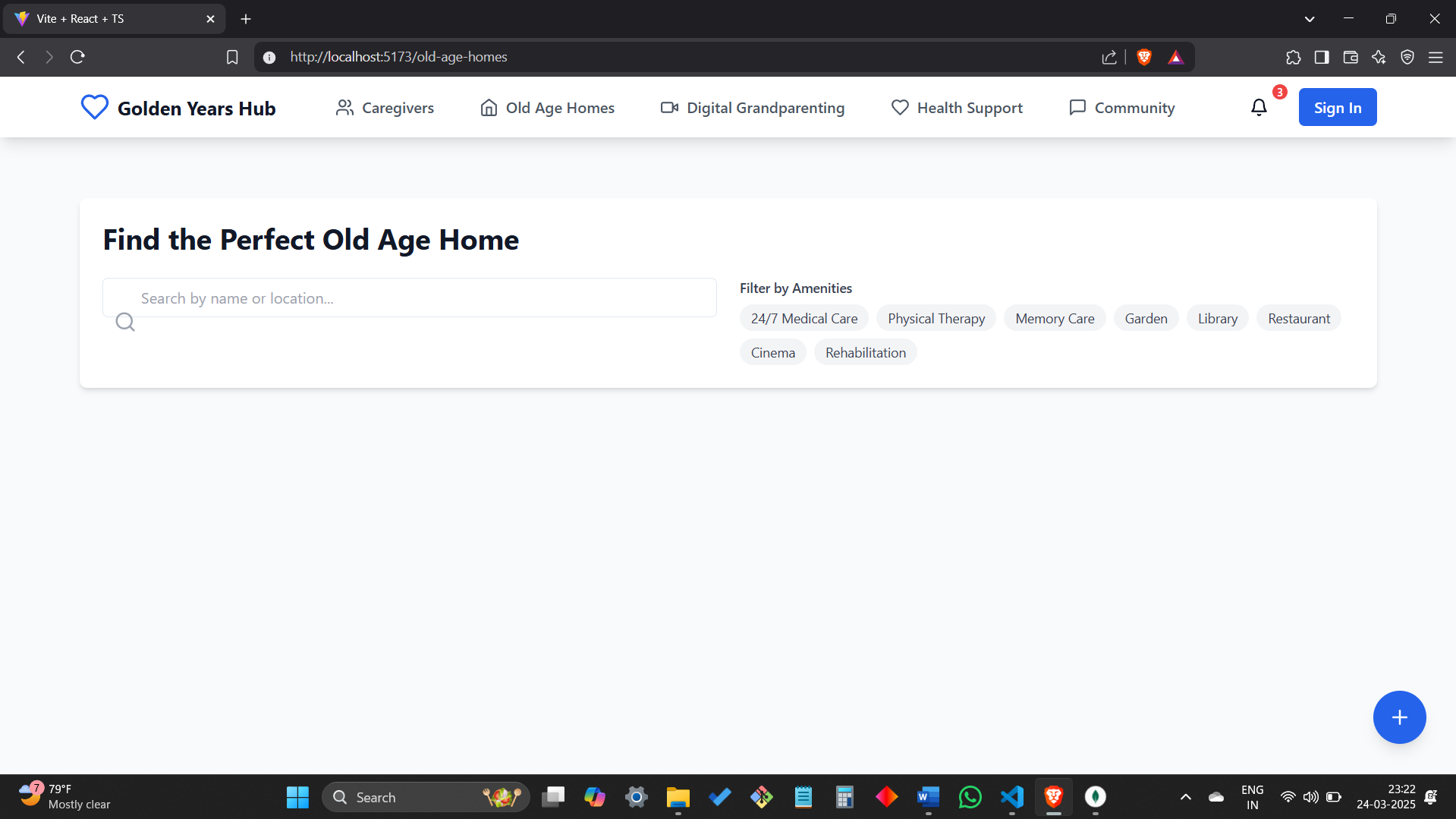
**DIGITAL GRANDPARENTING PAGE (VIDEO CALL FEATURE USING ZEGOCLOUD API )**

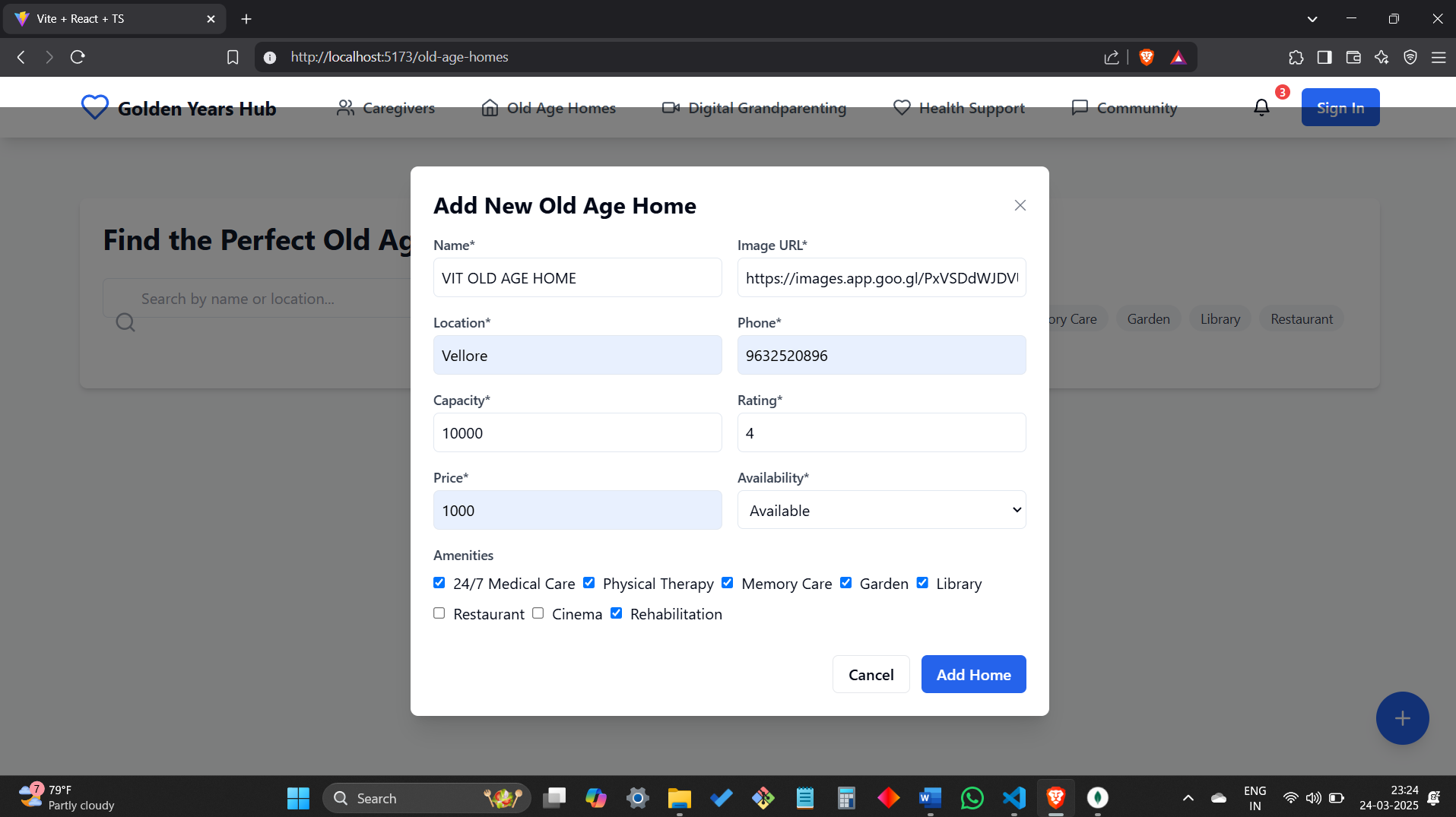




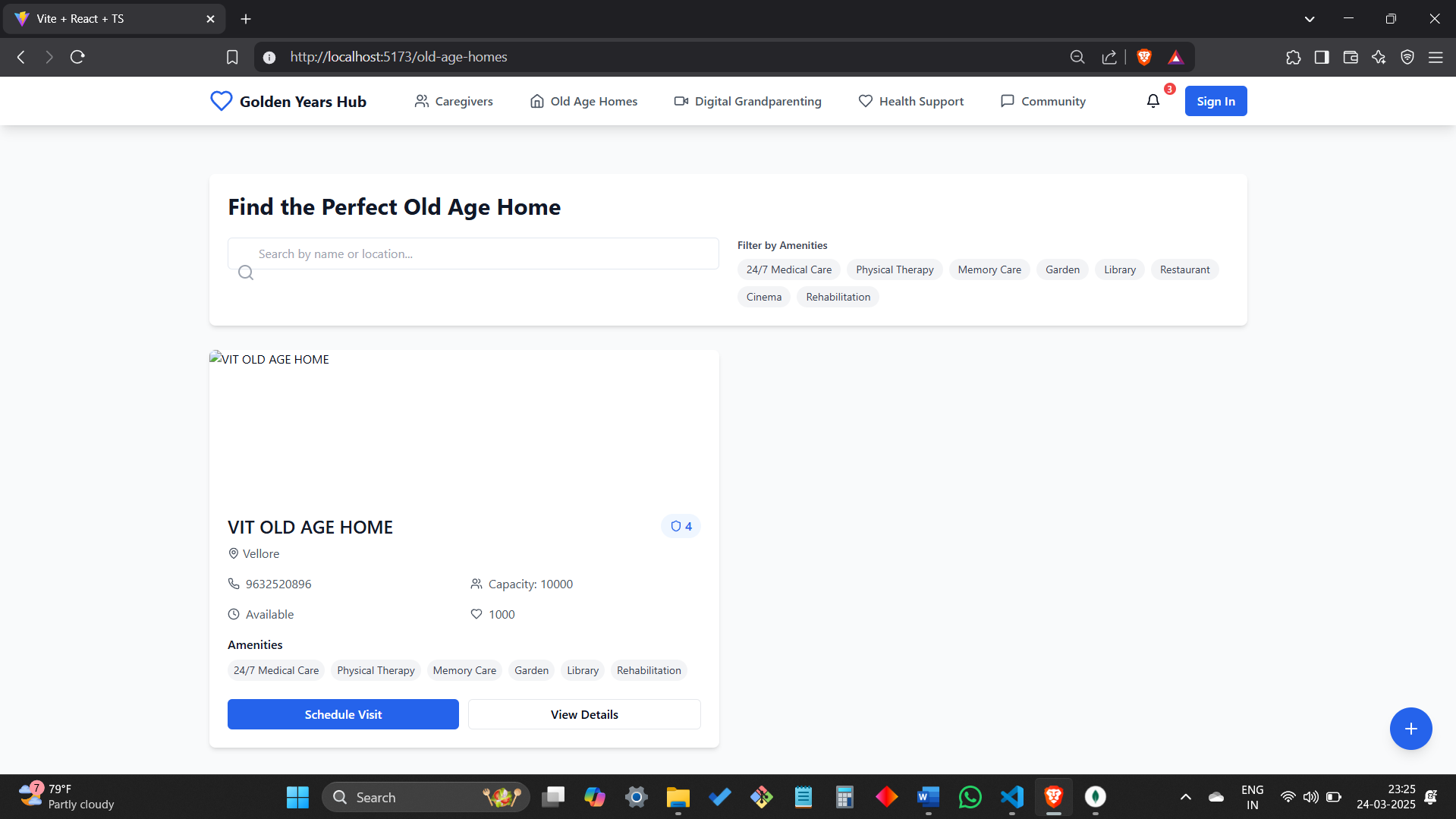
**OLD AGE HOME**

**(ADD NEW OLD AGE HOME FEATURE : STORES DETAILS OF OLD AGE HOME IN MONGODB DATABASE ON CLICKING + SYMBOL)**

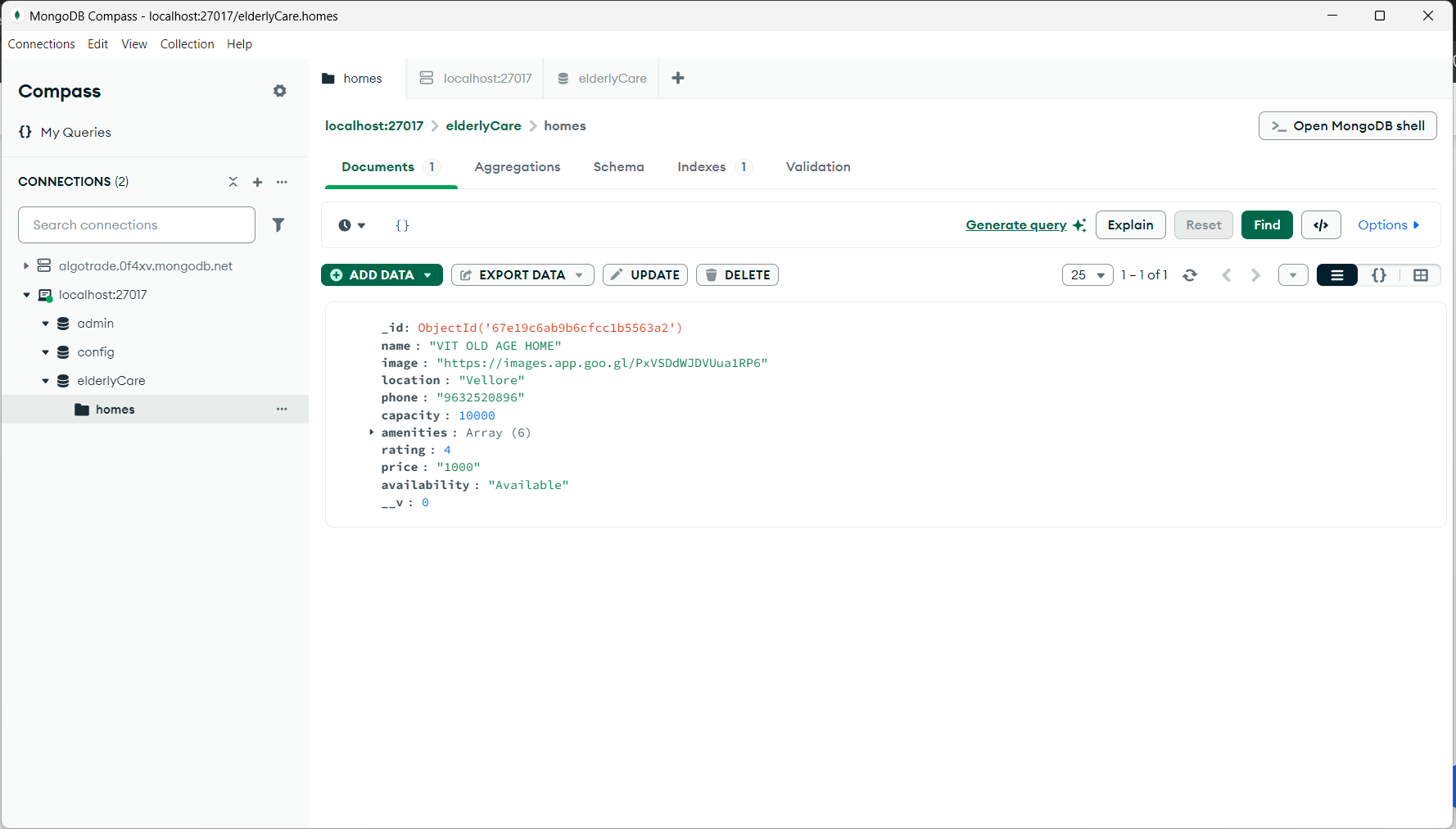


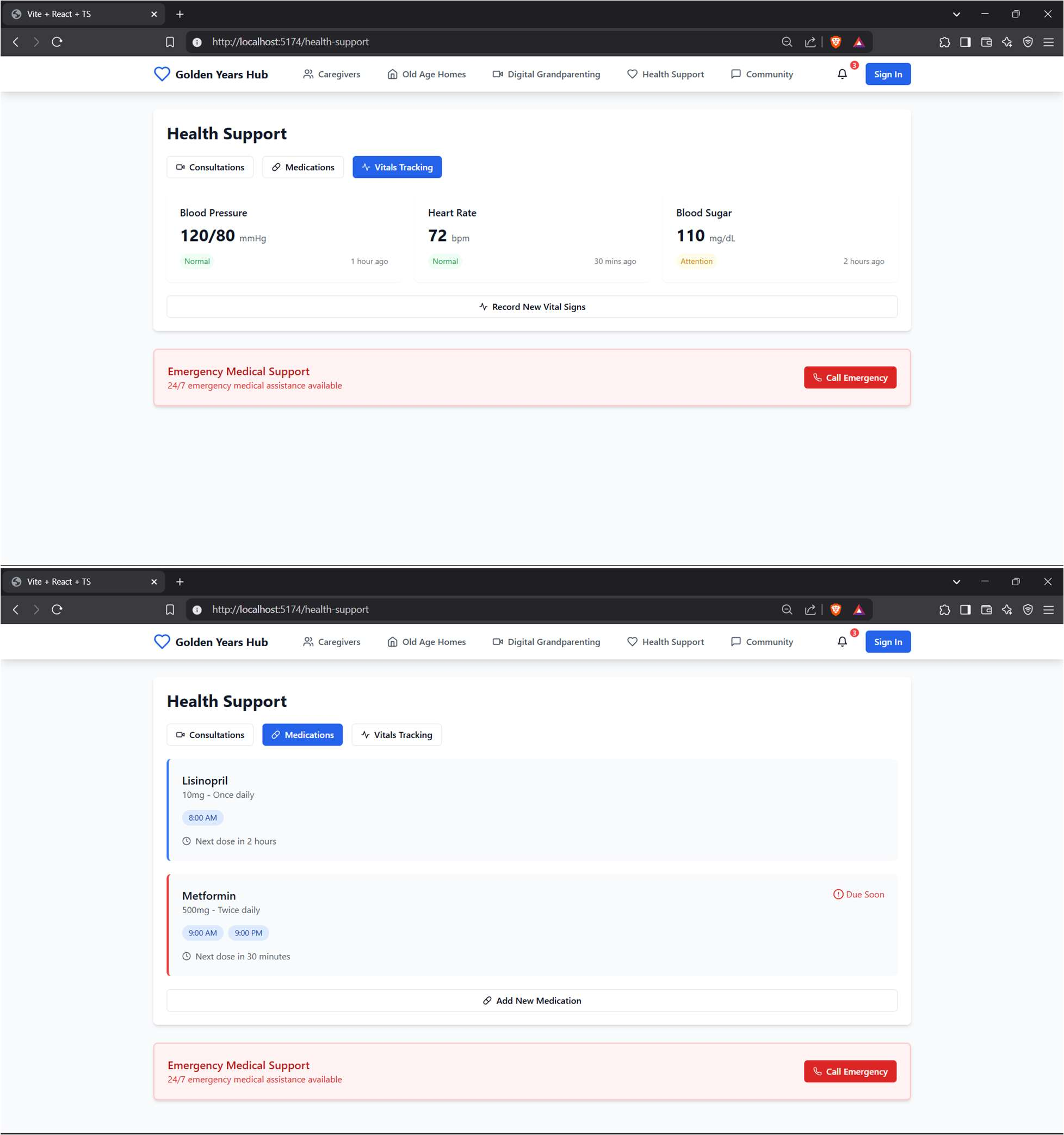


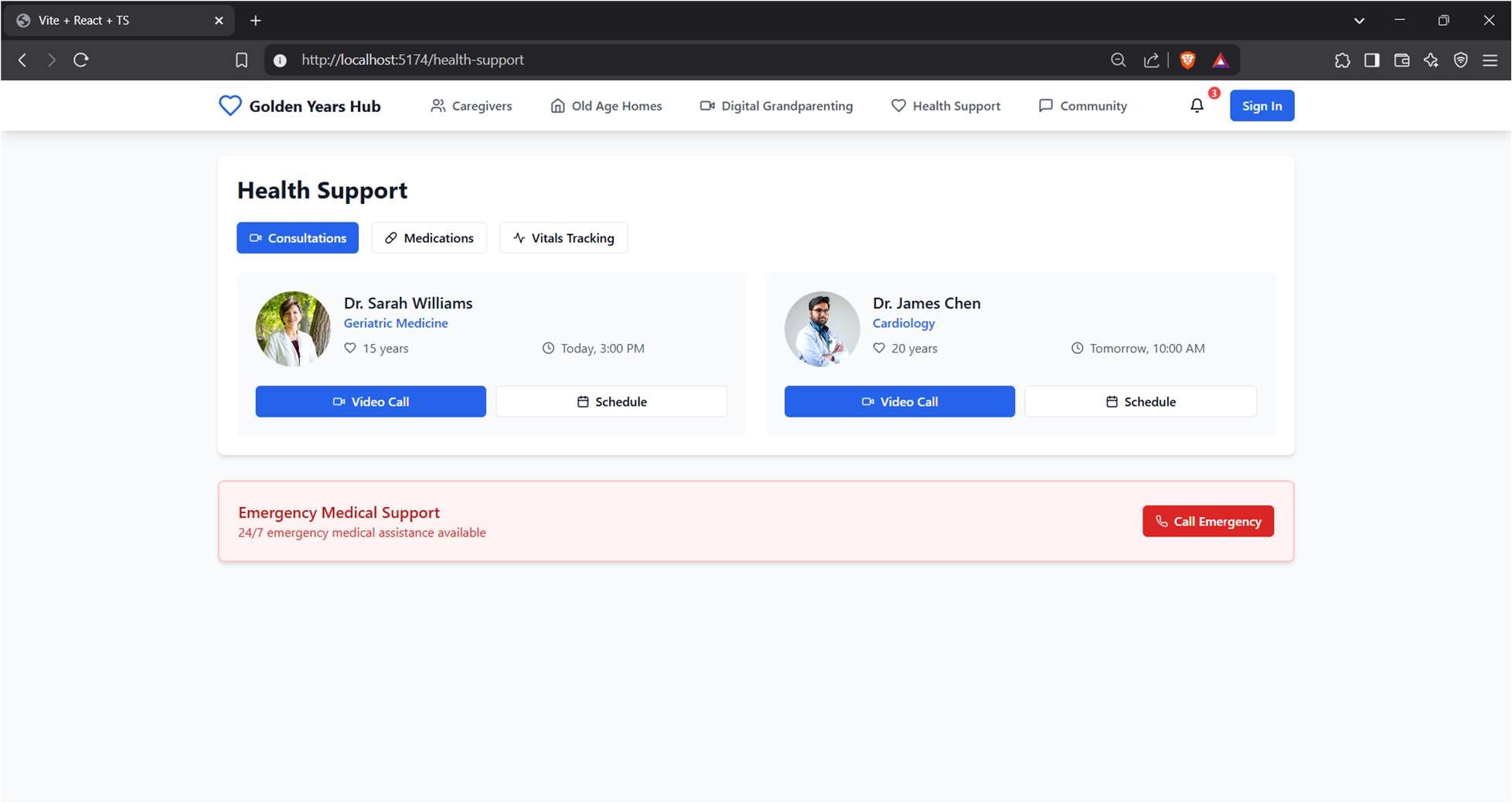
ADDED OLD AGE HOME TO THE DATABASE



**OLD AGE HOMES DATABASE / HOMES COLLECTION IN MONGODB COMPASS**



**HEALTH SUPPORT PAGE**



**CORE CODE FUNCTIONALITY FOR HANDLING VIDEO CALL API**

interface Home { id: string; name: string; location: string; phone: string;

rating: number; price: string; availability: string;

**}**

export function OldAgeHomes() {

const [homes, setHomes] = useState<Home[]>([]); const [searchTerm, setSearchTerm] = useState('');

const [selectedAmenities, setSelectedAmenities] = useState<string[]>([]); const [newHome, setNewHome] = useState<Home>({

id: '',

name: '',

location: '',

phone: '',

amenities: [], rating: 0, price: '',

availability: 'Available',

**});**

useEffect(() => { fetchHomes();

**}, []);**

const fetchHomes = async () => {

const response = await fetch('/api/homes'); const data = await response.json();

if (response.ok && data.success) setHomes(data.data);

**};**

const addHome = async () => {

const response = await fetch('/api/homes', { method: 'POST',

headers: { 'Content-Type': 'application/json' }, body: JSON.stringify(newHome),

**});**

const result = await response.json();

if (response.ok) setHomes([...homes, result.data]);

**};**

return (

<div>

<input type="text" value={searchTerm} onChange={(e) => setSearchTerm(e.target.value)} />

<button onClick={fetchHomes}>Refresh</button>

<button onClick={addHome}>Add Home</button>

<ul>

{homes

.filter((home) => home.name.toLowerCase().includes(searchTerm.toLowerCase())

**)**

.map((home) => (

<li key={home.id}>{home.name} - {home.location}</li>

**))}**

</ul>

</div>

**);**

**}**

**ADD A NEW HOME FEATURE**

**router.post('/', async (req, res) => {**

**const home = new Home({**

**name: req.body.name,**

**image: req.body.image,**

**location: req.body.location,**

**phone: req.body.phone,**

**capacity: req.body.capacity,**

**amenities: req.body.amenities,**

**rating: req.body.rating,**

**price: req.body.price,**

**availability: req.body.availability**

**});**

**try {**

**const newHome = await home.save();**

**res.status(201).json(newHome);**

**} catch (err) {**

**res.status(400).json({ message: err.message });**

**}**

**});**

**DATABASE SCHEMA**

**const homeSchema = new mongoose.Schema({**

**name: { type: String, required: true },**

**image: { type: String, required: true },**

**location: { type: String, required: true },**

**phone: { type: String, required: true },**

**capacity: { type: Number, required: true },**

**amenities: { type: [String], required: true },**

**rating: { type: Number, required: true },**

**price: { type: String, required: true },**

**availability: { type: String, required: true }**

**});**

**State management**

const [homes, setHomes] = useState<Home[]>([]);

const [searchTerm, setSearchTerm] = useState('');

const [selectedAmenities, setSelectedAmenities] = useState<string[]>([]);

Fetch data :

useEffect(() => {

fetch('http://localhost:5000/api/homes')

.then(res => res.json())

.then(data => setHomes(data));

}, []);

**Filter logic :**

const filteredHomes = homes.filter(home =>

(home.name.toLowerCase().includes(searchTerm.toLowerCase()) ||

home.location.toLowerCase().includes(searchTerm.toLowerCase())) &&

(selectedAmenities.length === 0 ||

selectedAmenities.every(a => home.amenities.includes(a)))

);

**Add new home :**

const handleAddHome = (homeData: Omit<Home, '\_id'>) => {

fetch('http://localhost:5000/api/homes', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

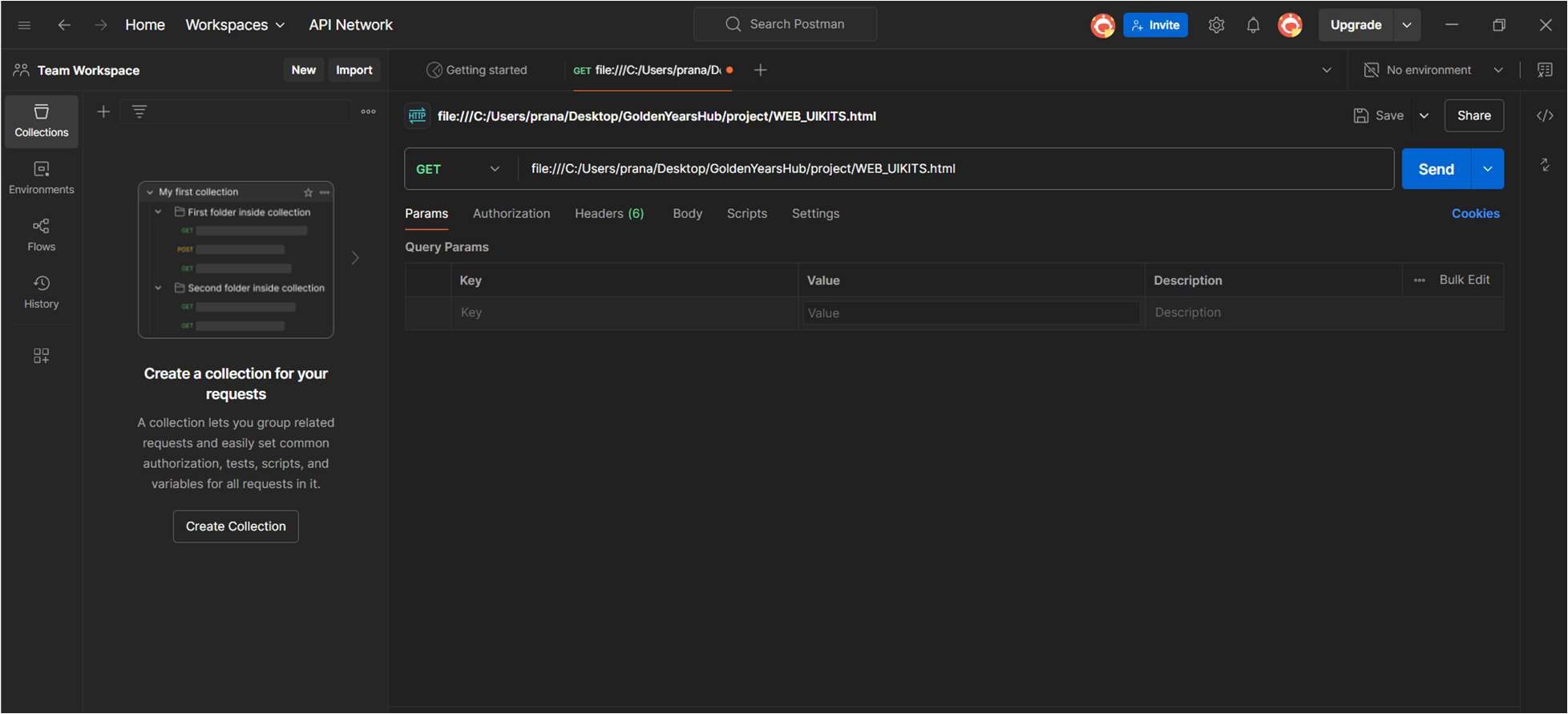
body: JSON.stringify(homeData)

})

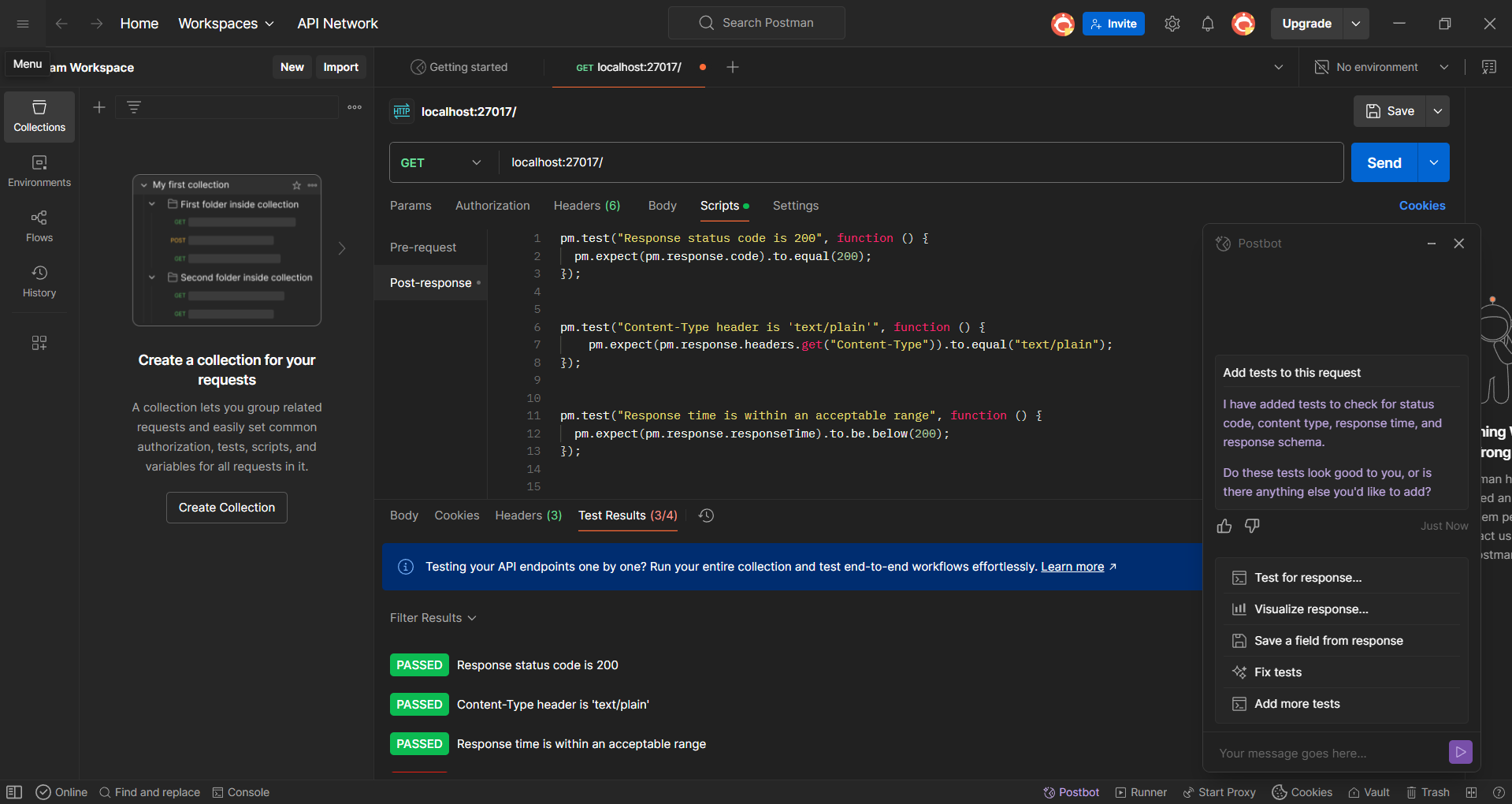
.then(res => res.json())

.then(newHome => setHomes(prev => [...prev, newHome]));

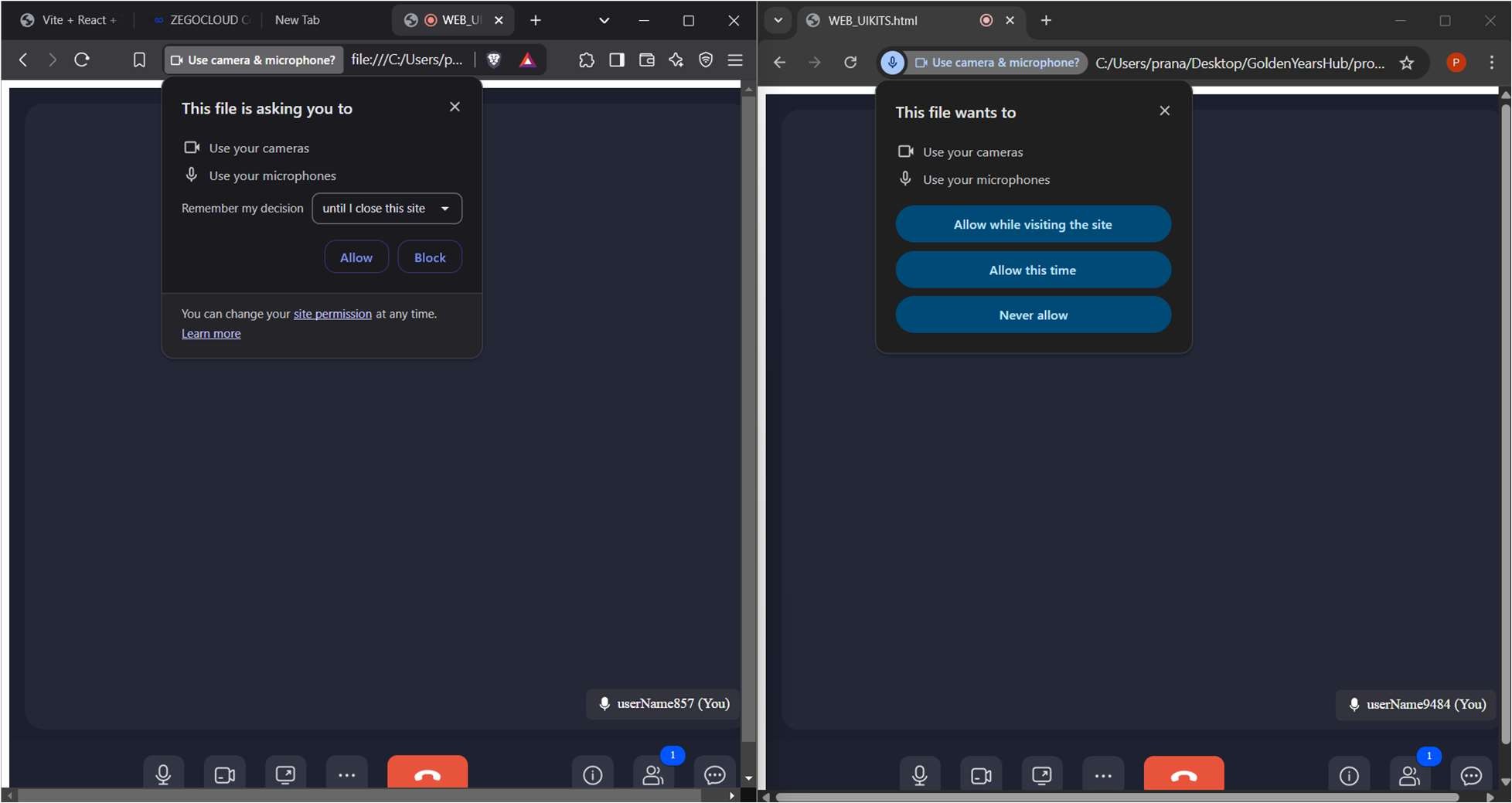
**TESTING (Test document as prepared in DA) POSTMAN ZEGOCLOUD API TESTING**



**POSTMAN MONGODB BACKEND API TESTING (PASSED TESTS 200 OK)**

****

**VIDEO CALL TESTING FOR 2 USERS**



**CONCLUSION:**

**Elderly Connect** successfully addresses the challenge of social isolation among senior citizens in old age homes by providing an easy-to-use digital platform for **communication, health monitoring, and entertainment**. With features like **video calling, AI companionship, medication reminders, and social engagement tools**, the platform enhances the quality of life for elderly individuals.

The system prioritizes **accessibility and security**, ensuring that seniors can use the platform with minimal assistance while keeping their data safe. By integrating **modern technology with elderly-friendly design**, **Elderly Connect** creates a sense of belonging and emotional support for senior citizens.

In the future, the platform can be expanded with **multi-language support, AI-driven health analysis, and personalized content recommendations**, further improving its usability and impact. **Elderly Connect is a step toward bridging the digital divide and fostering meaningful connections for the elderly community.**

**REFERENCES:**

1. **React Official Docs** – https://react.dev/
2. **Next.js Official Docs** – https://nextjs.org/docs
3. **MDN Web Docs (HTML, CSS, JavaScript)** – https://developer.mozilla.org/
4. **TailwindCSS Documentation** – https://tailwindcss.com/docs
5. **Stack Overflow (React Q&A)** – https://stackoverflow.com/questions/tagged/reactjs