HealHive – Telehealth Application

Repository: Ankita-Gupta2004/HealHive
Event: Veersa Hackathon 2026

Team Name: HealHive

Student 1: Ankita, 22001002009

Student 2: Deepanshu, 22001001027

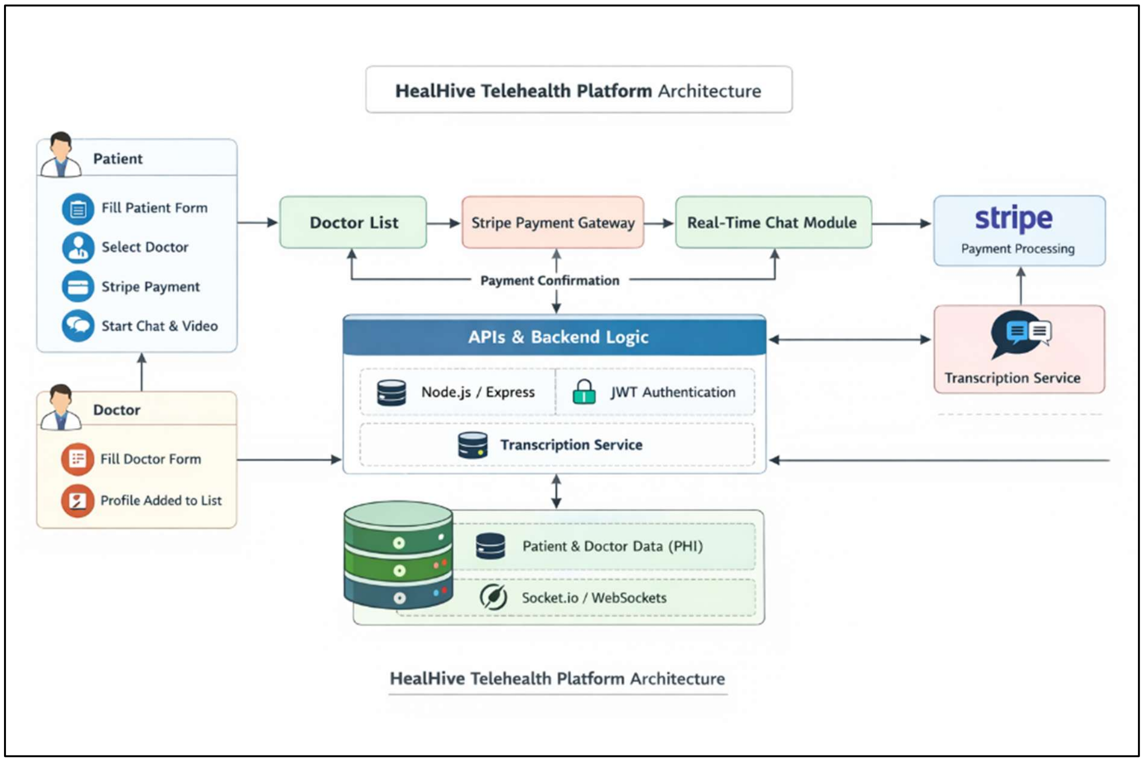
Student 3: Hansika, 22001001035

---

1. Overview

HealHive is a digitally enabled Telehealth platform designed to provide quick, affordable, and reliable healthcare access post-pandemic. The application supports secure patient–doctor interactions, online payments, real-time chat, and future-ready features like transcription to overcome dialect and accent challenges.

Architecture Diagram of application working



---

2. User Flow Summary

2.1 Patient Flow

1. User visits HealHive as a **Patient**
2. Fills **Patient Registration Form** (basic details + consultation specialty)
3. Views **Doctor List** based on specialty
4. Selects a **Registered Doctor**
5. Makes **Payment via Stripe**
6. **Real-time Chat** is enabled post-payment
7. Patient can view **Dashboard** (appointments, chat history, payment status, profile)

2.2 Doctor Flow

1. User visits HealHive as a **Doctor**
2. Fills **Doctor Registration Form** (qualification, specialty, availability)
3. Doctor profile is added to **Doctor List**
4. Doctor can chat with assigned patients
5. Doctor dashboard shows appointments and consultations

---

3. Identified Risks & Potential Blocking Issues

Area	Risk	Impact
Authentication	Unauthorized access	PHI data breach
Payment	Payment failure after doctor selection	Poor user experience
Chat	Chat enabled without payment	Revenue loss
Data Security	Improper storage of PHI	Compliance & legal risk
Scalability	Real-time chat lag	Poor consultation experience
Transcription	Accent mismatch	Miscommunication

---

4. Functional Requirements & Test Cases

FR-1: Patient Registration

Requirement: System must allow patients to register and submit basic health information.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR1-TC1	Valid patient registration	Fill all mandatory fields and submit	Patient registered successfully
FR1-TC2	Missing mandatory fields	Leave required field empty	Validation error shown

---

FR-2: Doctor Registration

Requirement: System must allow doctors to register and appear in doctor listings.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR2-TC1	Valid doctor registration	Fill form with valid details	Doctor profile added
FR2-TC2	Invalid specialization	Enter unsupported specialty	Error message displayed

---

FR-3: Doctor Listing & Selection

Requirement: Patients must see a filtered doctor list based on specialty.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR3-TC1	Specialty-based listing	Select specialty	Relevant doctors shown
FR3-TC2	No doctors available	Select rare specialty	Empty state shown

---

FR-4: Payment Integration (Stripe)

Requirement: Payment must be completed before consultation starts.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR4-TC1	Successful payment	Enter valid card details	Payment success
FR4-TC2	Failed payment	Enter invalid card	Payment failed message

Automated API Test (Example):

- Validate payment intent creation
- Verify payment status = succeeded

FR-5: Real-Time Chat

Requirement: Chat should activate only after successful payment.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR5-TC1	Chat after payment	Complete payment → Open chat	Chat enabled
FR5-TC2	Chat before payment	Try accessing chat	Access denied

FR-6: Dashboard

Requirement: Users must view appointments, history, and profile info.

Manual Test Cases:

TC ID	Scenario	Steps	Expected Result
FR6-TC1	Patient dashboard	Login as patient	Appointments visible
FR6-TC2	Doctor dashboard	Login as doctor	Patient list visible

---

FR-7: Data Privacy & Security

Requirement: All PHI data must be securely stored and accessed.

Test Cases:

- Verify HTTPS usage
- Unauthorized user cannot access chat history
- Firebase/DB rules restrict data access

FR-8: Transcription Service (Future Scope)

Requirement: System should support transcription to overcome accent/dialect issues.

Test Scenarios:

- Audio input → Text output accuracy
- Accent variation handling

---

5. Testing Strategy

HealHive follows a structured testing strategy in accordance with the Veersa Hackathon 2026 guidelines. To ensure functional correctness, backend reliability, and data privacy, we have adopted the following testing approaches:

- Approach A: Manual Test Cases (Documented)

- Approach C: Automated API Test Cases

5.1 Approach A: Manual Test Cases (Documented)

Manual testing is used to validate critical user workflows, UI behavior, real-time communication, and edge cases that directly impact user experience.

Test ID	Requirement ID	Test Scenario	Steps to Test	Expected Result
MTC-01	REQ-02	Doctor form validation	Submit the Doctor Registration Form with missing mandatory fields (e.g., Specialty)	System displays validation error and doctor profile is **not** added to the doctor list
MTC-02	REQ-03	Payment enforcement	Patient selects a doctor and closes the Stripe payment window before completing payment	The **Chat** button remains disabled/hidden on the patient dashboard
MTC-03	REQ-04	Real-time chat reliability	Send a message from Patient and check the Doctor's dashboard	Message appears instantly via **Socket.io/WebSockets** without page refresh
MTC-04	REQ-06	Accent & transcription handling	Start chat/video session and speak with a heavy accent	Transcription service renders text on screen with >90% accuracy

5.2 Approach C: Automated API Test Cases (Backend Reliability)

Automated API testing is performed to ensure backend stability, correct business logic execution, and strict enforcement of PHI data security.

1. Payment Verification API

- **Endpoint:** POST /api/payment/verify-session
- **Setup:** Use a Stripe Test Session ID
- **API Call:** Pass sessionId in the request body
- **Expected Result:**

```
{
  "status": "paid",
  "chatAccess": true
}
```

- **Purpose:** Ensures chat access is granted **only after successful payment**

---

2. Secure Dashboard Access

- **Endpoint:** GET /api/patient/dashboard
- **Setup:** Call the API using an expired JWT token
- **Expected Result:**
    - HTTP Status: **401 Unauthorized**
- **Purpose:** Validates authentication enforcement and **PHI data privacy protection**

---

3. Doctor Listing Logic

- **Endpoint:** GET /api/doctors/specialty?type=Cardiology
- **Setup:** Seed the database with doctors from multiple specialties
- **Expected Result:**
    - API returns **only Cardiology doctors**
    - No unrelated doctor profiles are included
- **Purpose:** Confirms accurate filtering and correct business logic

---

6. Tools & Technologies

- **Frontend:** React, Vite
- **Backend:** Node.js, Express
- **Database:** Firebase
- **Auth:** Firebase Authentication
- **Payment:** Stripe
- **Chat:** Real-time (WebSockets / Firebase)
- **Testing:** Postman
- **Tools:** GitHub
- **Deployment:** Firebase Hoisting (Frontend), Render (Backend)

---

7. Conclusion

HealHive effectively addresses the post-pandemic Telehealth challenges by providing secure consultations, seamless payments, and real-time communication. With proper testing and risk mitigation, the solution meets all problem statement requirements and is scalable for future enhancements.