**Core Principles for Our Use Case Design:**

**Zero investment**: Only free, open-source, or free tools with no required payment

**Web-based**: Everything runs in the browser — no native apps or paid APIs

**AI and smart features**: Lightweight, rule-based logic or free AI APIs

**Video conferencing**: Included via free embeddable platforms — no coding required

**1. Patient Registration**

**Solution**: Firebase Firestore + Firebase Auth (free tier)

**Goals**:

* Create secure patient profiles
* Store demographic and medical history
* Enable role-based access

**Workflow**:

* Patient or receptionist fills out registration form
* System validates and stores data in Firebase

**Success Scenario**:

* Patient profile is created and accessible to authorized users

**Extension Scenario**:

* Duplicate record detected → system prompts merge or alerts admin

### 2. ****Medical Record Update****

**Solution**: Firebase Firestore

**Goals**:

* Let physicians and nurses update patient data
* Maintain accurate, up-to-date health records

**Workflow**:

* Authorized user searches for patient
* Edits record and saves changes

**Success Scenario**:

* Record is updated and timestamped

**Extension Scenario**:

* Unauthorized access → system denies update and logs attempt

### 3. ****Access History via Telemedicine****

**Solution**: Jitsi Meet (embedded iframe)

**Goals**:

* Enable virtual consultations
* Share medical history during video calls

**Workflow**:

* Patient joins Jitsi Meet session via embedded link
* Physician accesses patient records during call

**Success Scenario**:

* Consultation completed with shared record access

**Extension Scenario**:

* Connection fails → system offers reschedule or offline access

### 4. ****QR Code-Based Record Sharing****

**Solution**: QRious.js or goqr.me API

**Goals**:

* Allow patients to share records instantly
* Simplify access for external providers

**Workflow**:

* Patient generates QR code from dashboard
* Provider scans code to view records

**Success Scenario**:

* Record accessed securely via QR

**Extension Scenario**:

* QR expired or invalid → system prompts regeneration

### 5. ****Cross-Organization Record Sharing****

**Solution**: Google Drive (link sharing) or Firebase Hosting

**Goals**:

* Share records with clinics, hospitals, labs
* Maintain patient control over shared data

**Workflow**:

* Patient selects records and destination
* System generates secure link and sends it

**Success Scenario**:

* Receiving organization accesses records successfully

**Extension Scenario**:

* Link broken or unauthorized → system alerts patient

### 6. ****AI-Powered Emergency Alerts****

**Solution**: JavaScript rule-based logic + EmailJS

**Goals**:

* Detect critical health issues
* Notify patients and optionally providers

**Workflow**:

* System scans records for emergency flags
* Sends alert via email or dashboard

**Success Scenario**:

* Patient receives timely alert and takes action

**Extension Scenario**:

* False alert → system logs and allows dismissal

### 7. ****Smart Reminders****

**Solution**: Browser notifications or EmailJS

**Goals**:

* Remind patients of appointments, medication, checkups
* Support preventive care

**Workflow**:

* Patient sets reminders via form
* System sends email or browser notifications

**Success Scenario**:

* Patient receives reminder and completes task

**Extension Scenario**:

* Reminder missed → system resends or escalates

### 8. ****Consent-Driven Data Sharing Dashboard****

**Solution**: Firebase Firestore + HTML dashboard

**Goals**:

* Let patients manage who can access their data
* Provide transparency and control

**Workflow**:

* Patient views sharing permissions
* Toggles access for organizations

**Success Scenario**:

* Permissions updated and logged

**Extension Scenario**:

* Unauthorized toggle attempt → system denies and logs

### 9. ****Manual Vitals Entry / Wearable Sync****

**Solution**: HTML form + optional Google Fit API

**Goals**:

* Allow patients to input health metrics
* Optionally sync with wearables

**Workflow**:

* Patient enters vitals manually or syncs device
* System stores and analyzes data

**Success Scenario**:

* Metrics saved and used for alerts or trends

**Extension Scenario**:

* Invalid data → system prompts correction

**Tech Integration Summary**

|  |  |
| --- | --- |
| Feature | Free Tool |
| Database & Auth | Firebase (free tier) |
| Hosting | Firebase Hosting / GitHub Pages |
| Video Conferencing | Jitsi Meet (iframe embed) |
| QR Code | QRious.js or goqr.me API |
| Email Alerts | EmailJS |
| Reminders | Browser notifications |
| Record Sharing | Google Drive / Firebase links |
| AI Logic | JavaScript rules or free OpenAI tier |

## Use Case Diagram

This visual diagram shows the actors and their interactions with the system using standard UML notation.

🎯 **Actors**:

* Patient
* Physician
* Receptionist
* External Organization
* System (AI logic)

🎯 **Use Cases**:

* Register Patient
* Update Medical Record
* Access History via Telemedicine
* Generate QR for Record Sharing
* Share Records with Other Organizations
* AI Emergency Alerts
* Smart Reminders
* Consent Dashboard
* Manual Vitals Entry / Wearable Sync

**Informal Use Case**

Use Case: Smart Reminders

**Actor**: Patient **Goal**: Receive timely reminders for appointments, medication, and checkups **Scenario**: The patient logs into the health dashboard and sets reminders for upcoming appointments and daily medication. The system stores these reminders and sends notifications via email or browser alerts. If a reminder is missed, the system resends it or escalates it with a follow-up message.

**Tech Used**:

* EmailJS (for email alerts)
* Browser notifications (for local alerts)
* Firebase (to store reminder data)

**Formal Use Case**

Use Case: Smart Reminders

|  |  |
| --- | --- |
| **Element** | Description |
| Use Case Name | Smart Reminders |
| Actor | Patient |
| Preconditions | Patient is logged in and has a valid profile |
| Main Flow | Patient logs into the system, **Patient navigates to the “Reminders” section,…** |

 Patient opens dashboard

 Selects “Set Reminder”

 Enters details (date, time, type)

 System stores reminder

 System sends notification at scheduled time | | **Alternative Flow** |

 Reminder missed → system resends or escalates

 Invalid input → system prompts correction | | **Postconditions** | Reminder is logged and patient is notified | | **Tech Integration** | EmailJS, browser notifications, Firebase Firestore |