Journal Homepage: www.ijarpr.com ISSN: 3049-0103 (Online)

International Journal of Advance Research Publication and Reviews

Vol 03, Issue 11, pp 223-232, November, 2026

# Helyra: A Secure and Adaptive AI Health Companion for Family Activity, Medication and Emergency Support

***Dharshan M[1], Harish Vishal N[1], Harisakesan K[1] Dr. M. Deepa[2]***

[1]Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India
[2]Project Guide, Associate Professor, Department of Information Technology, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India
[harishkanagaraj862@gmail.com]

**ABSTRACT :**

People living with long-term conditions, and the family members who support them, usually depend on several disconnected tools for activity tracking, medication reminders, prescription records and emergency contact. This paper presents Helyra, a progressive web application that combines these functions in one adaptive and security-focused health companion. The system uses a client-server architecture with React, Vite and Tailwind CSS on the frontend and Node.js, Express.js and MongoDB Atlas on the backend. A multi-stage digital signal processing engine counts steps from accelerometer and gyroscope data while rejecting shakes, vehicle vibration and table movement. A bilingual (English and Tamil) voice assistant with spoken-spelling correction recognises health intents, gives SOS keywords priority and calls the Gemini API with automatic model fallback for open questions. Uploaded prescriptions are read by a document-understanding service constrained never to invent doctor instructions, and later questions are answered only from the document. Families link accounts with a unique family code and control six separate sharing permissions, with detailed medical information private by default. Security measures include Argon2id password hashing, TOTP and WebAuthn passkeys, HttpOnly session cookies, CSRF protection, rate limiting, user-scoped queries against BOLA/IDOR attacks, AES-256-GCM encryption and audit logging. An automated suite of 21 integration tests covers registration, lockout, two-factor authentication, session revocation, data isolation, sharing permissions and account deletion.

**Keywords:** Digital Health, Adaptive AI, Step Detection, Voice Assistant, Medical Document Understanding, Multi-Factor Authentication, Family Health Sharing

## Introduction

Helyra is a mobile-first web application that acts as an adaptive health companion for individuals and their families. It brings together daily activity tracking, hydration and sleep logging, medication scheduling, medical document understanding, a voice assistant, family sharing and an emergency SOS workflow. Its guiding idea, stated in the project's own data contracts, is "not just AI that responds, but AI that adapts": the interface, voice tone, text size and language follow each person's preferences rather than a fixed design.

### *Background and Motivation*

Smartphones contain the sensors needed to monitor everyday health, yet the resulting information is usually spread over a step counter, a reminder application, a gallery of prescription photographs and a messaging group with relatives. Older
adults and people with conditions such as diabetes, hypertension or heart disease need simple, reliable guidance, and their families need enough visibility to help without intruding on privacy. Health data is also among the most sensitive personal data, so weak authentication or careless sharing can cause real harm. Helyra was developed to provide one coherent product that is easy to use, supports Tamil as well as English, and treats security and consent as core requirements.

### *Problem Statement*

Existing consumer tools rarely combine accurate step counting on ordinary phones, medication adherence, understanding of doctor-issued documents, hands-free interaction and controlled family sharing. Prescriptions are hard to read and easy to misinterpret, voice interfaces often fail on health vocabulary and accented speech, and many applications store sensitive records with limited protection. Helyra addresses these gaps through a single secure platform that adapts to the user and shares information only with explicit permission.

### *Objectives of the Project*

- To build a progressive web application that records steps, distance, active minutes, calories, water intake and sleep for each user.
- To implement a sensor-based step detection engine that works without a native pedometer and rejects false steps.
- To provide a bilingual voice assistant that recognises health intents, corrects speech-to-text errors and treats emergency phrases with the highest priority.
- To read uploaded prescriptions and answer questions strictly from their content, without inventing instructions.
- To manage medication schedules and dose logs and to generate adaptive weekly or monthly health reports.
- To let families link accounts through a family code and share data under six individually controlled permissions.
- To provide an SOS workflow that alerts an emergency contact and family members.
- To protect accounts and records with strong password hashing, multi-factor authentication, session control, encryption and audit logging.

### *Existing and Proposed System*

In the existing situation a person uses a separate pedometer, a calendar reminder for tablets, paper prescriptions and phone calls to relatives. None of these tools understands the others, prescriptions cannot be queried, and sharing is either all or nothing. Voice features, where present, are generic and not tuned to medication names or Tamil speech.

Helyra is a single application in which every module reads from the same user-scoped data. The user records or automatically captures activity, schedules medicines, uploads prescriptions for structured extraction and talks to the assistant in English or Tamil. Family members linked by a family code see only the categories the owner has enabled, and the owner can trigger an SOS alert by button or by voice. All sensitive operations pass through an authenticated, rate-limited and audited API.

## Literature Review

Research and practice on digital health applications cover activity sensing, conversational interfaces, document understanding and privacy. Helyra brings these themes together in one product. The main themes drawn from the literature and from the technical documentation of the tools used are summarised below.

### *Smartphone Activity Tracking*

Phones count steps either through a dedicated hardware pedometer (Android step counter and detector sensors, or Apple Core Motion) or by processing raw accelerometer signals. Signal-processing approaches separate gravity from linear acceleration, smooth the signal, detect peaks and apply timing rules, but they are sensitive to threshold choice and to non-walking motion such as vehicle vibration. Browser-based applications face an additional limit because native pedometer interfaces are not exposed to web code on every platform, which motivates a layered strategy with a software fallback [10].

### *Voice Interaction in Health Applications*

Speech interfaces reduce the effort of data entry for older users and for people with limited vision or dexterity. Their weaknesses are recognition errors on medical terms, medication names and regional accents, and the cost of acting on a wrongly understood request. Practical designs therefore combine wake phrases, vocabulary correction and a conservative treatment of emergency intents [11].

### *Large Language Models for Medical Documents*

Large language models can extract structured facts from images and text of prescriptions and answer follow-up questions. They can also produce plausible but unsupported statements, so health applications need grounding rules, explicit “not mentioned” answers, structured output schemas and a clear statement that the system offers wellness guidance and not diagnosis [7].

### *Authentication, Authorisation and Data Protection*

Health records require stronger protection than ordinary accounts. Current guidance recommends memory-hard password hashing such as Argon2id [8], time-based one-time passwords [9], phishing-resistant WebAuthn credentials [12], protection against broken object-level authorisation and cross-site request forgery, and a documented list of web application risks [13]. Privacy is further supported by data minimisation, consent-based sharing and a user-initiated right to erasure.

### *Family and Caregiver Support*

Caregivers value timely awareness of missed medicines, low activity and emergencies, but constant monitoring can feel intrusive. Granular, owner-controlled sharing, with sensitive categories closed by default, balances safety and autonomy, and this principle is built into the sharing model of Helyra.

## Methodology and System Design

Development followed requirement analysis, system design, implementation, testing and deployment. The requirements came from three needs: reliable personal health tracking, safe and understandable use of medical documents, and secure family collaboration. The design stage produced the layered architecture of Fig. 1, the entity model and the REST interface, and each module was then implemented with the technologies listed in Table 1.

### *System Architecture*

Helyra uses a client-server architecture. The React 19 frontend [1], built with Vite [2] and styled with Tailwind CSS, runs as a progressive web application with a web manifest and a service worker that caches static assets and revalidates them in the background. The client reads device sensors, performs speech recognition and speech synthesis through browser APIs, and renders the dashboards. The Node.js [3] and Express.js [4] backend applies security headers, cookie parsing, session resolution and CSRF checks before routing requests to the authentication, health data, document and AI services. MongoDB Atlas [5] is the primary persistence layer for users, sessions, daily health records, medications, documents,
workouts, families and audit logs, while a Prisma schema [6] documents the same entity model and its relations. Medical files are written to an object-storage adapter that encrypts each file with AES-256-GCM.

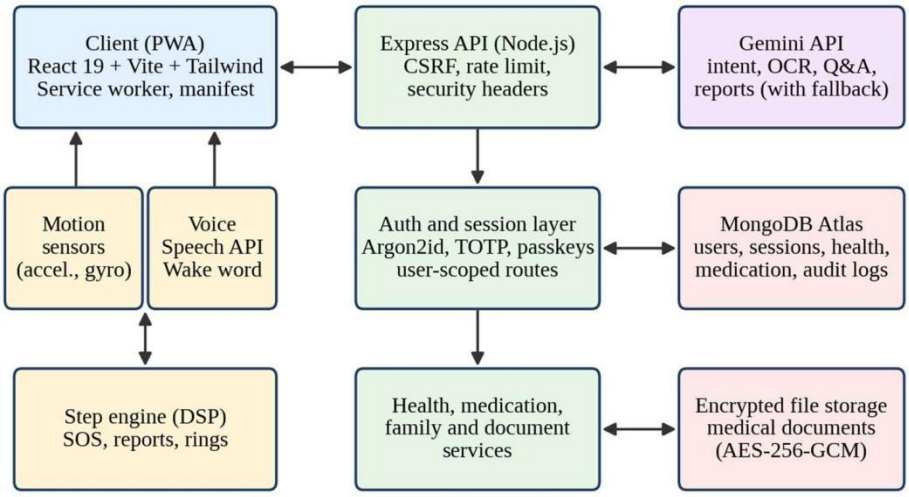



**Fig. 1 - Overall architecture of the Helyra platform**

**Table 1** - Technology stack of the Helyra platform.

| Layer | Technology | Purpose |
|---|---|---|
| Frontend | React 19, Vite, Tailwind CSS, Motion, Lucide | Responsive progressive web interface with activity rings, voice modal and dashboards; installable through a web manifest. |
| Backend | Node.js, Express.js, TypeScript | REST APIs, security middleware, session handling and orchestration of AI calls. |
| Database | MongoDB Atlas; Prisma schema | Document storage of users, sessions, health records, medications, documents, families and audit logs. |
| AI services | Google Gemini API (@google/genai) | Voice intent understanding, prescription extraction, document question answering and report generation, with model fallback. |
| Security | Argon2id, bcrypt, otpauth, SimpleWebAuthn, Node crypto | Password hashing, TOTP, passkeys, AES-256-GCM encryption and token hashing. |
| Reports | jsPDF | Export of adaptive health reports as PDF documents. |
| Testing and CI | Vitest, Supertest, GitHub Actions | Automated integration tests and build verification on every push. |

### *Users, Families and Sharing Control*

Every user owns a private profile that stores age, height, weight, medical conditions, allergies, goals and adaptive preferences such as interaction type (notification, voice or both), response length, voice tone, font size and high-contrast mode. Each account receives a unique family code, and another user can be linked to the family by entering that code. Access between linked users is governed by the six permissions in Table 2. The first four default to enabled, whereas detailed documents and private medical information default to disabled so that the most sensitive data is never shared unless the owner decides so.

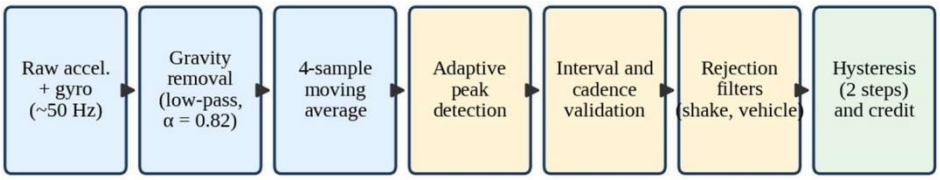
**Table 2 - Family sharing permissions and their default values.**

| Permission | Default | Data made visible to the linked member |
|---|---|---|
| Activity | On | Steps, distance, active minutes and calories. |
| Exercise | On | Workout sessions and routes. |
| Medication status | On | Whether scheduled doses were taken, not the clinical notes. |
| Water | On | Daily water intake against the goal. |
| Detailed documents | Off | Uploaded prescriptions and extracted content. |
| Private medical information | Off | Conditions, allergies and other private medical details. |

### *Step Detection Engine*

The step counter chooses the best available source in a fixed order: native hardware counter, platform motion framework, sensor fusion and finally a software DSP pipeline. When only the browser is available, the pipeline of Fig. 2 processes accelerometer samples at about 50 Hz. Gravity is estimated with a first-order low-pass filter and subtracted to obtain linear body acceleration, as given in equation (1), where $a_t$ is the raw acceleration sample, $g_t$ the gravity estimate and $\alpha = 0.82$.

$$g_t = \alpha \cdot g_{t-1} + (1 - \alpha) \cdot a_t , \qquad a_{lin} = a_t - g_t \tag{1}$$




*Accepted steps update daily totals, hourly buckets, distance (stride = 0.415 × height) and calories*

*Platform order: native hardware counter → Core Motion → sensor fusion → fallback DSP*

**Fig. 2 - Multi-stage step detection pipeline**

The linear signal is smoothed by a four-sample moving average, and peaks between 0.85 and 9.0 m/s² become step candidates. A candidate is accepted only if it follows the previous step by 230 to 1400 ms, if the change in interval stays within the cadence tolerance, if the gyroscope reading is below 5.5 rad/s (which rejects waving or twirling) and if the dominant vibration frequency is below 8 Hz (which rejects vehicles). Two consecutive rhythmic steps confirm a walking bout and are credited together. The parameters are collected in Table 3. Distance D in kilometres for n steps follows from the stride estimate in equation (2), with the user's height H in centimetres, and calories are estimated from walking MET values and body weight.

$$D = n \times (0.415 \times H) / 100000 \tag{2}$$

**Table 3 - Main parameters of the step detection engine.**

| Parameter | Value | Role |
|---|---|---|
| Gravity smoothing (α) | 0.82 | Low-pass estimate of the gravity vector. Moving- |
| average window | 4 samples | Removes sensor jitter without noticeable lag. |
| Peak threshold range | 0.85 to 9.0 m/s² | Accepts gentle walking, rejects drops and impacts. |

| Parameter | Value | Role |
|---|---|---|
| Step interval range | 230 to 1400 ms | Rejects bounce echoes and broken gait sequences. |
| Hysteresis buffer | 2 steps | Confirms a real walking bout before crediting steps. |
| Cadence variance tolerance | 0.55 | Allows natural speed changes and turns. |
| Gyroscope rejection | 5.5 rad/s | Rejects phone twirling and waving. |
| Vehicle vibration limit | 8 Hz | Rejects bus, car and train vibration. |

Accepted steps update the daily total and hourly buckets, which drive the activity rings, the hourly timeline, the streak and badge card and the workout cockpit with a cadence metronome. A baseline mechanism at midnight prevents double counting, and wake lock, the Visibility API and an audio keep-alive help tracking to continue while the application is in the background.

***Bilingual Voice Assistant and Intent Engine***

The assistant is activated by a wake phrase such as “Hey Helyra” or by the voice button. Speech-to-text output first passes through a correction engine that maps common phonetic errors to the intended health term, for example mis-heard words for water, medicine, tablet or the names of common drugs. The corrected query is sent to the intent service. The service first checks for emergency phrases in English and Tamil (such as SOS, help me, heart attack or fainted) and, on a match, returns the SOS action immediately without any model call. Other requests go through a fast keyword matcher for hydration, medication, activity and vital-sign questions, which answers in well under a second using the user’s own context. Only requests that do not match are sent to the Gemini API [7] with a JSON response schema. Responses are spoken through speech synthesis in en-US or ta-IN, and every response carries a short wellness notice.

***Medical Document Understanding***

When a user photographs or uploads a prescription, the file is stored in the encrypted object store and its content is sent to the analysis service. The prompt instructs the model never to invent medication names, dosages or instructions, to extract only what is clearly written, and to mark anything illegible as “[Unclear in document]”. The extracted facts are stored with the document. For later questions the answering service uses only that content and returns a structured result with the answer, a flag showing whether it is grounded in the document, the quoted section and a safety reminder. If the document does not contain the answer, the system says so and advises the user to consult the doctor.

***Medication, Reports and Emergency SOS***

Each medication stores its name, dosage, schedule times and notes, and a dose log records every scheduled dose with its status and the time taken. Adherence values feed the home dashboard and the reports. The report generator combines steps, water, sleep, calories, weight and adherence into a weekly or monthly summary with next-week focus points and lifestyle guidance in English or Tamil, and the result can be exported as a PDF. If the AI service is unavailable, a built-in fallback report is returned so that the screen never fails.

The SOS workflow can be started from a button or by voice. A short countdown lets the user cancel an accidental trigger; otherwise the request is dispatched with the trigger method and a location when available, the telephone dialer opens for the emergency contact, and linked family members are marked as notified. The server-side dispatch is currently simulated and returns an alert identifier, while SMS and push-notification provider modules are prepared for production use.

***Security Mechanisms***

Security follows the layered chain of Fig. 3. Passwords are hashed with Argon2id using 64 MiB of memory, three iterations and four lanes, with bcrypt (cost 12) as a fallback where native bindings are unavailable [8]. Registration
enforces a minimum length of 12 characters, a list of breached passwords and a check against the user’s own name and e-mail. Second-factor options are TOTP [9], whose secrets are encrypted with AES-256-GCM and backed by eight single-use recovery codes stored as hashes, and WebAuthn passkeys [12]. Sessions use HttpOnly, Secure and SameSite=Lax cookies with a seven-day sliding lifetime, can be inspected per device and are revoked on password change or “sign out everywhere”.

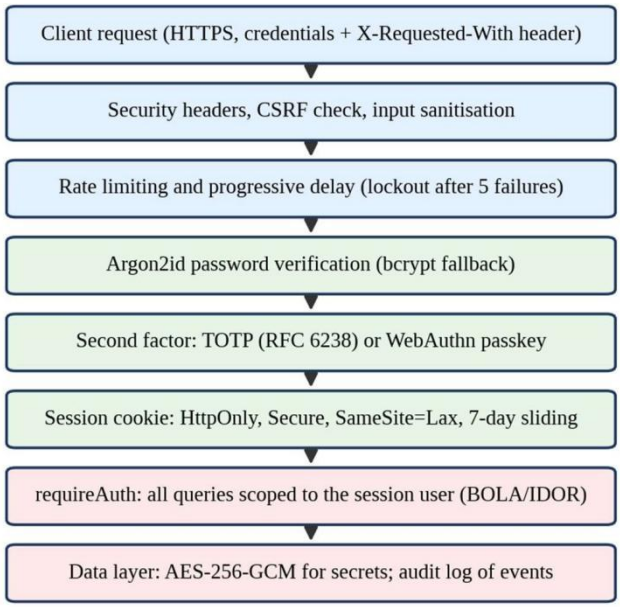



**Fig. 3 - Layered security chain applied to every request**

Brute-force protection returns one generic message for every login failure, adds progressive delay, enforces per-address and per-email rate limits and applies a temporary 15-minute lockout after five failed attempts. State-changing requests must carry a custom header to defeat CSRF. Every health, medication and document route reads the user identifier only from the validated session and scopes all queries to it, which prevents broken object-level authorisation (BOLA/IDOR) [13]. Response headers include Content-Security-Policy, Strict-Transport-Security, X-Content-Type-Options, X-Frame-Options, Referrer-Policy and Permissions-Policy. Security events such as login, logout, password change, two-factor changes, passkey changes, session revocation and account deletion are written to an audit log that excludes passwords, tokens and medical content. Users can erase their account after password confirmation, which removes their records and stored files. The main controls are summarised in Table 4.

**Table 4 - Summary of security controls in Helyra.**

| Area | Specification |
|---|---|
| Password hashing | Argon2id (m = 65,536 KiB, t = 3, p = 4); bcrypt cost 12 fallback. Password policy At least 12 characters; breached-password and personal-data checks. |
| Second factor | TOTP with encrypted secrets and 8 hashed recovery codes; WebAuthn passkeys. Sessions HttpOnly, Secure, SameSite=Lax; 7-day sliding window; per-device revocation. |
| Abuse control | Generic login error; progressive delay; 5 failures give a 15-minute lockout; per-route rate limits. |
| Authorisation | Server-side session check; all queries scoped to the session user. |
| Data protection | AES-256-GCM for secrets and stored medical files; hashed single-use tokens. |
| Accountability | Audit log of authentication and account events; user-initiated erasure. |

### *Testing and Deployment*

Backend behaviour is verified by an integration suite written with Vitest [14] and Supertest that exercises the real HTTP interface. Its 21 test cases are grouped in Table 5. A GitHub Actions workflow installs dependencies, runs the tests and builds the project on every push and pull request. The frontend is built with Vite as a progressive web application, and the backend is bundled with esbuild and started with Node.js, so the system can run locally or in a cloud environment with a MongoDB Atlas cluster.

**Table 5 -** Coverage of the automated integration test suite.

| Area | Tests | Behaviour verified |
|---|---|---|
| User registration | 3 | Argon2id-hashed registration, rejection of duplicate e-mail, creation of a second user for isolation tests. |
| Login and lockout | 4 | Successful login with session cookie, wrong-password rejection, lockout after five failures. |
| Two-factor authentication | 3 | Secret and recovery-code generation, enabling with a valid code, second-factor challenge at login. |
| Session management | 3 | Listing sessions, revoking one session, signing out all devices. |
| BOLA / IDOR protection | 4 | Creation of own records; other users cannot modify or delete medications or documents and read only their own data. |
| Family sharing | 1 | Granular permissions stored and enforced, medical details private by default. |
| Account deletion | 3 | Password required, wrong password rejected, complete purge of records and files. |

## Results and Discussion

This section presents the outcome of implementing Helyra. The screens in Figs. 4 to 6 show how activity tracking, medication, voice interaction, document understanding, reporting, family sharing, emergency support and account security work together.

### *Home Dashboard and Adaptive Experience*

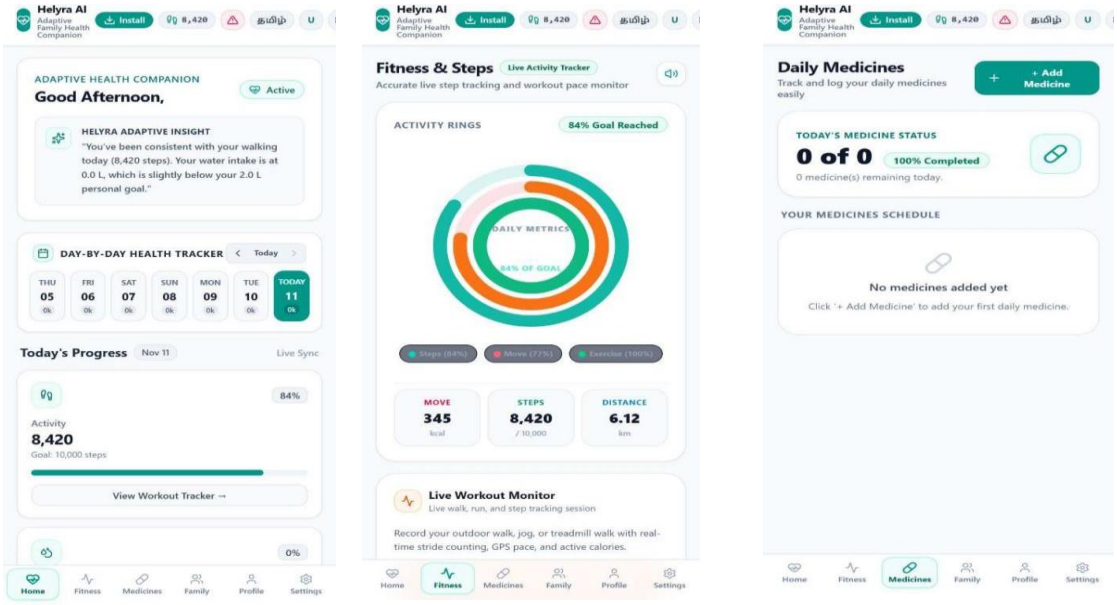
After sign-in the home dashboard summarises the day: steps against the goal, water intake, sleep, calories and medication progress (Fig. 4a). The language switch between English and Tamil changes the interface text and the spoken responses, and the adaptive preferences for voice tone, response length, large or extra-large fonts and high contrast change how information is presented, which supports older users.

### *Fitness Tracking*

The fitness screen shows three concentric activity rings for steps, calories and active minutes, an interactive hourly timeline, the streak and badges card and the workout cockpit (Fig. 4b). A separate debug screen exposes the sensor source in use, the motion state, the cadence and the counts of rejected false steps, so that the behaviour of the DSP pipeline can be inspected and tuned for different phones.

### *Medication Management*

The medicines screen lists each medication with its dosage and schedule and lets the user mark doses as taken (Fig. 4c). The adherence figure is reused in the dashboard, the voice answers and the generated reports, so that one record supports every view.




**(a) Home dashboard** **(b) Fitness tracker** **(c) Medicines**

**Fig. 4 - Home, fitness and medication screens of Helyra**

***Voice Assistant***

The voice modal shows the recognised speech, the corrected text and the spoken answer (Fig. 5a). Questions about water, medicines, steps and vital signs receive immediate, personalised answers, emergency phrases in either language open the SOS flow, and open questions receive a short model-generated reply. When the language model is unavailable the fast matcher and fallbacks still return a useful response.

***Medical Document Analysis***

On the documents screen a user captures or uploads a prescription, receives the extracted doctor, medicine, dosage and instruction details and can ask follow-up questions (Fig. 5b). Answers state whether they come from the document, and a question outside the document receives the "not mentioned" reply with advice to ask the doctor, which keeps the assistant from guessing.

***Adaptive Health Reports***

The reports screen produces the weekly or monthly summary with focus points for the coming week and exports it as a PDF (Fig. 5c).




**(a) Voice assistant** **(b) Document analysis** **(c) Health report**

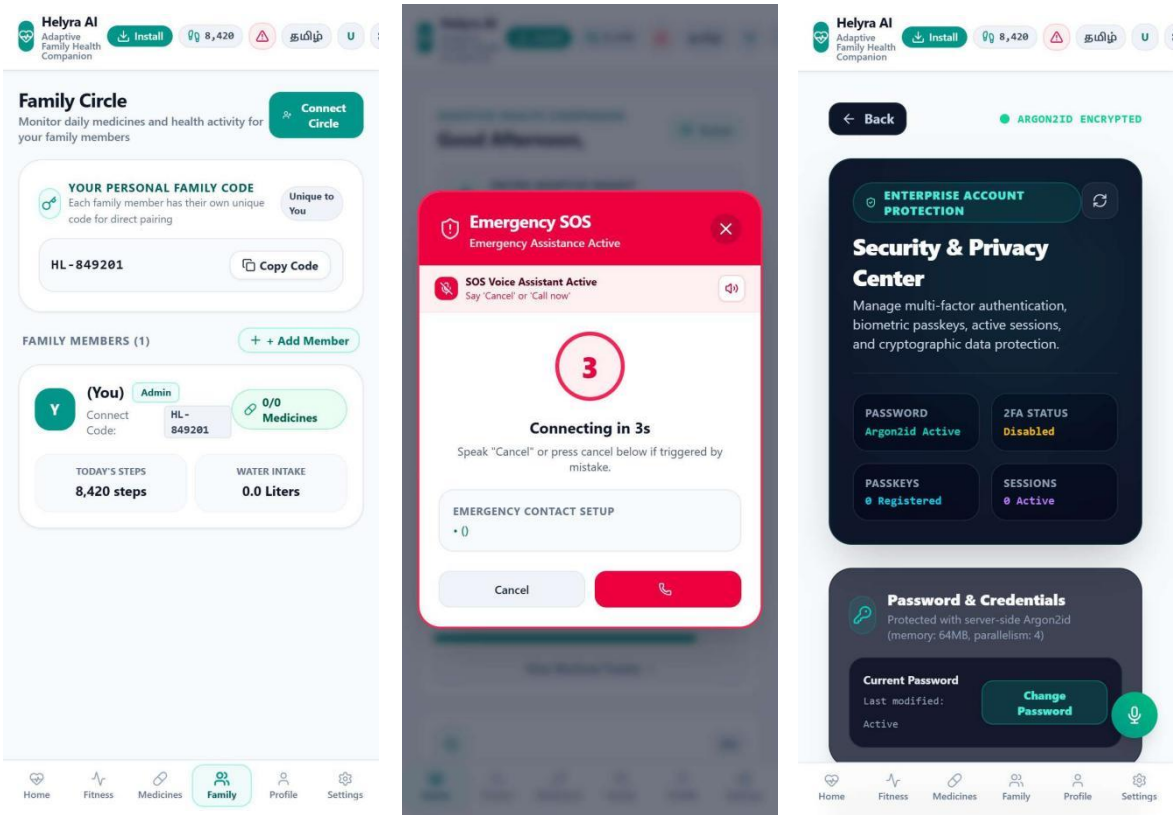
**Fig. 5 - Voice, document understanding and report screens**

***Family Sharing and Emergency SOS***

The family screen displays the user's family code, the linked members and the six sharing switches of Table 2 (Fig. 6a). The SOS dialog counts down, can be cancelled and then confirms dispatch and opens the dialer for the emergency contact (Fig. 6b). Because the sharing permissions are stored per owner and viewer pair and checked on the server, a viewer sees only the categories the owner has enabled.

***Account Security***

The security settings screen lets users enable TOTP, register passkeys, view active devices, sign out everywhere and delete the account (Fig. 6c).




**(a) Family sharing** **(b) SOS dispatch** **(c) Security settings**

**Fig. 6 - Family sharing, emergency SOS and security settings**

### *Discussion*

The implementation meets its objectives. Combining the modules around one user-scoped data model avoids duplicate entry and lets every screen, voice answer and report use the same information. The layered step engine allows tracking on phones without exposed pedometer interfaces, and its rejection rules reduce false steps from vehicles and handling. Treating emergency intents before any model call keeps the safety path fast and independent of network or model availability. The grounding rules in the document service reduce the risk of invented instructions, and the integration tests show that the main security promises, namely lockout, second factor, session revocation, data isolation, private-by-default sharing and erasure, are enforced through the real API.

Some limitations remain. Step-counting accuracy has not yet been measured against a reference device in a controlled study, and browser sensors depend on platform permissions. SOS dispatch is simulated at the API level. Fallback responses use illustrative default values when user context is missing, and AI-generated content is advisory and does not replace a doctor. These points define the next stage of work.

## Conclusion and Future Scope

### *Conclusion*

Helyra is an adaptive family health companion that unites activity tracking, medication management, prescription understanding, bilingual voice interaction, family sharing and emergency support in a single progressive web application. The combination of React, Vite, Node.js, Express.js, MongoDB Atlas and the Gemini API gives a modern full-stack design, while Argon2id hashing, multi-factor authentication, passkeys, user-scoped authorisation, encryption and audit logging protect sensitive health data. The project shows that adaptive, voice-enabled and privacy-respecting health support can be delivered with widely available web technologies.

### *Future Scope*

Helyra can be extended in the following directions:

- Controlled accuracy studies of the step engine against reference pedometers across devices, carrying positions and walking speeds.
- Production emergency dispatch through SMS, push notifications and live location sharing with delivery confirmation.

- Native Android and iOS builds that use hardware pedometers and background services directly.
- Integration with wearables and clinical vitals such as heart rate, blood pressure and glucose.
- More Indian languages for the interface and the voice assistant, and improved accent handling.
- Medication interaction and refill alerts, and doctor-facing summaries shared with consent.
- Independent security review, penetration testing and compliance mapping for health-data regulations.

***Acknowledgements***

The authors thank the management of Sri Shakthi Institute of Engineering and Technology, the Head of the Department, Dr. S. Prakash, the Project Co-Ordinator, Dr. T. Esther, and the Project Guide, Dr. M. Deepa, Department of Information Technology, for their guidance, support and encouragement throughout this project.

**REFERENCES**

**[1]** Meta Open Source. (n.d.). *React documentation*. https://react.dev/

**[2]** Vite. (n.d.). *Vite documentation*. https://vite.dev/

**[3]** OpenJS Foundation. (n.d.). *Node.js documentation*. https://nodejs.org/

**[4]** OpenJS Foundation. (n.d.). *Express.js documentation*. https://expressjs.com/

**[5]** MongoDB, Inc. (n.d.). *MongoDB Atlas documentation*. https://www.mongodb.com/docs/atlas/

**[6]** Prisma. (n.d.). *Prisma ORM documentation*. https://www.prisma.io/docs

**[7]** Google. (n.d.). *Gemini API documentation*. https://ai.google.dev/gemini-api/docs

**[8]** Biryukov, A., Dinu, D., Khovratovich, D., & Josefsson, S. (2021). *Argon2 memory-hard function for password hashing and proof-of-work applications (RFC 9106)*. IETF. https://www.rfc-editor.org/rfc/rfc9106

**[9]** M'Raihi, D., Machani, S., Pei, M., & Rydell, J. (2011). *TOTP: Time-based one-time password algorithm (RFC 6238)*. IETF. https://www.rfc-editor.org/rfc/rfc6238

**[10]** MDN Web Docs. (n.d.). *Sensor APIs*. Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/Sensor_APIs

**[11]** MDN Web Docs. (n.d.). *Web Speech API*. Mozilla. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

**[12]** World Wide Web Consortium. (2021). *Web Authentication: An API for accessing public key credentials, Level 2*. https://www.w3.org/TR/webauthn-2/

**[13]** OWASP Foundation. (n.d.). *OWASP Top Ten web application security risks*. https://owasp.org/www-project-top-ten/

**[14]** Vitest. (n.d.). *Vitest documentation*. https://vitest.dev/