A

Major Project Synopsis on

SwasthyaSaarthi

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P)

In Partial fulfillment for the award of degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

By

RITIK KUMAR, 0818CS221165

VIKAS CHOURASIYA, 0818CS221223

VINAY PATIDAR, 0818CS221224

YASH JAIN, 0818CS221230

Under the Guidance of

SMRITI JAIN

Assistant Professor

INDORE INSTITUTE OF SCIENCE & TECHNOLOGY, INDORE

PITHAMPUR ROAD, OPPOSITE IIM, RAU, INDORE 453331, MP.

Approved by AICTE, New Delhi, affiliated to RGPV, Bhopal, Recognized by UGC under Section 2(f)

Jul-Dec-2025

INDORE INSTITUTE OF SCIENCE & TECHNOLOGY, INDORE

PITHAMPUR ROAD, OPPOSITE IIM, RAU, INDORE 453331, MP.

Approved by AICTE, New Delhi, affiliated to RGPV, Bhopal, Recognized by UGC under Section 2(f)

DECLARATION

We hereby declare that the work, which is being presented in this dissertation entitled **"SwasthyaSaarthi"** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering**, is an authentic record of work carried out by us.

The matter embodied in this report has not been submitted by us for the award of any other degree.

RITIK KUMAR
0818CS221165

VIKAS CHOURASIYA
0818CS221223

VINAY PATIDAR
0818CS221224

YASH JAIN
0818CS221230

INDORE INSTITUTE OF SCIENCE & TECHNOLOGY, INDORE

PITHAMPUR ROAD, OPPOSITE IIM, RAU, INDORE 453331, MP.

Approved by AICTE, New Delhi, affiliated to RGPV, Bhopal, Recognized by UGC under Section 2(f)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **"……..…**

……..”

submitted to RGPV, Bhopal (M.P.) in the Department of **…….**

……by

Mr…...……Enrollment number.……

Mr…...……Enrollment number.……

Mr…...……Enrollment number.……

Mr…...……Enrollment number.……

in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **…….....**

during the academic year **…….**

PROJECT GUIDE **HEAD OF THE DEPARTMENT**

PRINCIPAL

ABSTRACT

SwasthyaSaarthi is a lightweight, AI-assisted digital health platform that builds a practical "digital twin" for each patient to enable early risk detection and proactive care in low-resource settings. Patients and community health workers capture vitals, symptoms, medications, and lifestyle logs through a simple web/mobile interface with offline-first data entry and background sync. Data flows through a Spring Boot API with JWT-based authentication and role-based access control to a relational database that stores user profiles, encounters, attachments, and time-series measurements using an extensible twin schema. An agentic AI and rules engine performs anomaly and trend detection, computes a Cardio-Renal Risk Score (CRRS), and triggers configurable alerts with escalation to caregivers via SMS or push notifications. Dashboards for patients and health workers visualize trends, insights, CRRS tiers, adherence status, and outreach queues, enabling timely interventions. The methodology is privacy-by-design (encryption, audit logs, consent), and phased: start with validated clinical thresholds and simulations, then incrementally incorporate ML models trained on public datasets. Targeting prevalent chronic conditions such as diabetes and hypertension, the system seeks to reduce missed deterioration events by converting routine community-collected data into actionable guidance. Evaluation will track alert precision/recall, time-to-intervention, user satisfaction, and feasibility of deployment in rural environments, including bandwidth resilience and multilingual messaging.

ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to everyone who has supported us throughout the journey of completing our minor project.

First and foremost, we would like to thank the Head of the Department, **Dr. Richa Gupta**, for their encouragement, guidance, and providing us with all the necessary facilities to carry out this project.

We extend our sincere gratitude to our Project Coordinator, **Mr. Rakesh Jain** for their continuous support, timely advice, and valuable feedback. We are deeply indebted to our Project Guide, **Mrs. Smriti Jain**, for their expert guidance, valuable suggestions, and support throughout this project.

Thank you all for making this project a meaningful and enriching experience.

Ritik Kumar
0818CS221165

Vikas Chaurasiya
0818CS221223

Vinay Patidar
0818CS221224

Yash Jain
0818CS221230

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	DESCRIPTION	PAGE NO.
1	AI	Artificial Intelligence	01
2	API	Application Programming Interface	01
6	ICMR	Indian Council of Medical Research	01
8	CKD	Chronic Kidney Disease	02
3	JWT	Json Web Token	03
4	CRRS	Cardio-Renal Risk Score	03
5	SMS	Short Message Services	04
7	RBAC	Role Based Access Control	13

I N D E X

CHAPTER **PAGE NO**

CHAPTER 1

INTRODUCTION

1.1 Background

India is confronting a rapidly escalating burden of non-communicable diseases (NCDs), with Type 2 Diabetes and Hypertension at the forefront of the crisis as of 2025. The ICMR–INDIAB nationwide study, published in The Lancet Diabetes & Endocrinology, estimated that 101 million Indians have diabetes [1] and an additional 136 million have prediabetes based on 2021 data, reflecting much higher prevalence than earlier estimates. The same research program also reported that approximately 35.5% of India's population about 315 million people live with hypertension[1] / [4], underscoring the scale of cardio-metabolic risk nationwide. This dual epidemic is compounded by frequent co-morbidity, with multiple Indian and international analyses documenting substantial overlap between diabetes, hypertension, and dyslipidemia in clinical populations, reinforcing the need for holistic cardio-metabolic management rather than siloed disease control.

Despite the scale of disease, control rates remain critically low, revealing a persistent "management gap" between diagnosis and effective day-to-day self-management. A large cross-sectional analysis of Indian adults found that among those with hypertension, only about one in three are diagnosed, fewer than one in five receive treatment, and approximately one in twelve achieve blood pressure control [4] / [5], with significant state-level variation necessitating decentralized solutions. Systematic reviews similarly report poor hypertension control in India, with only a minority of treated patients attaining target blood pressure, indicating gaps in sustained care and adherence across settings. These gaps are clinically consequential, as uncontrolled diabetes and hypertension drive preventable complications including chronic kidney disease, cardiovascular events, neuropathy, and premature mortality placing mounting strain on a health system historically oriented toward acute, episodic care rather than continuous chronic-disease management.

The magnitude and complexity of India's cardio-metabolic burden highlight the imperative to shift from reactive treatment to proactive, preventive, and patient-guided care models that integrate risk factor control, multi-morbidity management, and sustained engagement. Evidence from the ICMR–INDIAB program and related national surveys provides a robust epidemiological foundation to reframe care toward integrated risk stratification, simplified protocols, and longitudinal monitoring to close the diagnosis–treatment–control gaps at scale. In this context, solutions that combine accessible data capture, multilingual support, risk communication, and continuous guidance are well aligned with the evolving national priority to curb the NCD epidemic and reduce downstream cardio-renal complications.

1.2 Problem Statement

India's diabetes–hypertension "dual epidemic" is marked by very high prevalence but low control, creating a persistent management gap between diagnosis and day-to-day self-care that current systems fail to close. While ~101 million Indians live with diabetes and ~136 million with prediabetes, an estimated 315 million live with hypertension [1] / [4], illustrating a massive cardio-metabolic risk pool requiring continuous management rather than episodic care. Yet along the hypertension care cascade, only about 1 in 3 are diagnosed, <1 in 5 treated, and ~1 in 12 controlled, with large within-state variations evidence that population-level programs have not translated into consistent individual-level control. Systematic reviews likewise place pooled hypertension control in India at ~17.5% through 2020 [5], confirming chronically poor control despite expanding initiatives. Existing mHealth tools often act as passive loggers and are not optimized for India's multilingual, data-sparse contexts, leaving patients without timely, personalized, and actionable guidance to avert complications like CKD and cardiovascular events[3] / [4].

- Very high prevalence but low control of diabetes and hypertension.
- Large gaps across the care cascade: low diagnosis, lower treatment, and very low control.

- Day-to-day self-management gap between clinic visits leading to poor adherence.
- Existing mHealth tools are mostly passive, not personalized, and not optimized for multilingual/low-literacy contexts.
- Lack of continuous, actionable guidance and timely alerts for patients and caregivers.
- Episodic, acute-care system not suited for continuous chronic disease management.
- High risk of preventable complications (CKD, CVD) due to uncontrolled conditions.
- Significant state/district-level heterogeneity requiring localized, adaptive solutions.

1.2.1 Concept of a Digital Twin in Healthcare

In the context of this project, a Digital Twin is a structured, data-driven representation of an individual's current and historical health profile, continuously updated through periodic inputs from patients, caregivers, and health workers. Unlike fictional or 3D human clones, this digital twin is a functional health model composed of vitals (e.g., blood pressure, heart rate, oxygen saturation, body temperature), medical history, lifestyle factors, and environmental context. The digital twin serves as a continuously evolving repository for clinical and behavioural information, enabling proactive risk assessment, health simulations, and personalised recommendations. It is updated in near real-time where feasible, or via weekly/monthly manual inputs in low-connectivity environments. The twin supports both individual care journeys and population-level analytics in alignment[9]/[10].

1.3 Purpose/Objective

- **To design** a proactive, multilingual digital health guardian tailored for Type 2 diabetes and hypertension patients in India.
- **To develop** an explainable Cardio-Renal Risk Score (CRRS) engine that translates vitals, symptoms, and lifestyle data into actionable risk insights.
- **To implement** a timely nudging and alerting system that improves daily self-management and adherence to treatment plans.
- **To integrate** privacy-by-design principles to ensure secure handling of patient data and compliance with relevant health regulations.
- **To enable** scalable interoperability with national health programs and clinical pathways for boarder adoption.

1.4 Solution Approach

- Digital twin and low-friction data capture: Create a patient "digital twin" using periodic vitals (glucose, BP, HR), symptoms, medications, and lifestyle inputs captured via multilingual UI and voice to fit India's heterogeneous literacy and access landscape.
- Agentic guidance loop: Continuously analyze trends against clinical thresholds to trigger contextual nudges, adherence reminders, and escalation to caregivers for safety events, addressing the care gaps highlighted in India's hypertension cascade evidence.
- Explainable risk signal: Provide a daily, explainable cardio-renal risk score derived from deterministic rules initially (and extensible to ML later), making long-term risk tangible and improvable for patients.

- Engagement mechanics: Use points, streaks, and tailored "quests" to sustain participation beyond 3–6 months, a period where many digital interventions see declining effect, thereby supporting control gains targeted by IHCI and NP-NCD.

1.5 Scope of the Project

1.5.1 In scope (prototype/MVP)

- Web application (desktop/mobile web) for onboarding, daily logs, dashboards, risk visualization, and multilingual (e.g., English/Hindi) support aligned to India's epidemiology and control priorities.
- Voice-first logging and NLU for vitals and symptoms to reduce friction in routine self-management.
- Agentic rules-based coaching, alerts, and caregiver notifications anchored to thresholds consistent with national control goals and the documented treatment-to-control gaps.
- Deterministic risk engine with transparent drivers (vitals, adherence, lifestyle), allowing auditability and progressive validation against control outcomes.
- Privacy, consent, RBAC, encryption, and audit trails to align with programmatic integration prospects (e.g., NP-NCD/AB-HWC settings).

1.5.2 Out of scope (initial release)

- Native mobile apps, clinician dashboards/EHR integrations, continuous sensor streams, and ML-based personalized predictions; these will be evaluated after validating engagement and control improvements against baseline benchmarks from Indian literature.

1.6 Existing System & Limitations

- Public health programs: India's IHCI and NP-NCD have expanded standardized treatment protocols and follow-up capacity, enrolling millions; however, national control remains low in population studies, indicating challenges in adherence, continuity, and decentralized execution at scale.
- Control gap evidence: A pooled analysis shows hypertension control at ~17.5% (2001–2020) [5] , while district-level data indicate only ~1 in 12 hypertensives have controlled BP [4] / [5], underscoring systemic barriers to sustained control beyond initial diagnosis/treatment.
- Epidemiological scale and complexity: The ICMR-INDIAB study documents massive burdens across diabetes (101M), prediabetes (136M), and hypertension (~35.5% prevalence, ~315M people) [1] / [4], along with dyslipidemia and obesity, reinforcing the need for integrated, multi-risk management rather than siloed disease apps.
- Digital tool gaps: Many consumer health apps act as passive trackers, lack multilingual/voice accessibility, and provide limited, non-adaptive feedback, which is misaligned with India's care cascade deficits and heterogeneous literacy and access realities highlighted by national studies.

SwasthyaSarthi addresses these limitations by delivering an explainable, proactive, multilingual guidance layer that complements public programs, focuses on sustained behavior change, and targets documented diagnosis–treatment–control gaps with practical, patient-side interventions.

CHAPTER 2

METHODOLOGY

2.1 Proposed Methodology

2.1.1 User-centered design and iterative prototyping

- Start with problem discovery from literature and brief stakeholder interviews (patients/caregivers/primary care providers) to validate pain points: low-friction logging needs, multilingual support, adherence gaps, and need for timely guidance.
- Build a clickable prototype to test information architecture, risk visualization (CRRS), and voice-first flows in English/Hindi before engineering.

2.1.2 Data model and digital twin

- Define core entities: User, Conditions, Medications, Vitals (BP, glucose, HR), Lifestyle (activity, diet), Symptoms, CRRS Score, Events (alerts, nudges), Consent/Caregiver.
- Normalize time-series storage for vitals and CRRS; ensure auditability and explainability fields (reason codes for risk delta).

2.1.3 Rules-driven agentic guidance (phase 1)

- Implement a transparent rules engine mapping thresholds and trends to actions:
  - Safety thresholds (e.g., BP > 180/110, glucose < 70 or > 300).
  - Trend-based nudges (e.g., rising 7-day average SBP).
  - Adherence checks (missed logs/meds).

- Multilingual message templates with severity levels and escalation paths.

2.1.4 Explainable CRRS (Cardio-Renal Risk Score)

- Composite score updated daily using deterministic weighted inputs:
  - Vitals control bands (BP, fasting/post-prandial glucose).
  - Adherence (medication, logging).
  - Lifestyle (steps/activity proxy, diet tags if captured).
  - Symptoms flags.
- Provide driver breakdown and "what changed today" explanations.

2.1.5 Engagement and behavior design

- Streaks, badges, points, and short quests (e.g., 7-day BP logging streak).
- Gentle escalation: app notifications → SMS to patient → optional caregiver alert for safety events.

2.1.6 Privacy, security, and compliance foundation

- Consent capture, role-based access, encryption at rest and in transit, audit logs.
- Data minimization and purpose limitation aligned with DPDPA principles.

2.1.7 Evaluation and learning loop

- Define success metrics: weekly active users (WAU/MAU), 30/60/90-day retention, median logging days/week, proportion with controlled BP/glucose bands at 12 weeks, time-to-escalation for safety events, and user satisfaction (CSAT/NPS).
- Instrument analytics and A/B test nudges and engagement mechanics.

2.1.8 Digital Twin Component Table

Component	Description	Feasibility (MVP Stage)
Health Data	Vitals, symptoms, medical history, lifestyle logs, environment	Manual entry via app/web
Input Sources	Mobile/web forms,	Rural: manual entry; Urban: device sync
AI Engine	Rules-based detection, gradual ML-upgrade for predictions	Rule-first using open datasets
Simulation Module	Predict outcomes from lifestyle/medication changes	Clinical rule sets initially
Real-time Updates	Sync on demand or scheduled intervals	Weekly acceptable in pilot

Table: 2.1.1 Digital Twin Component Table

2.2 Process Model Adopted

2.2.1 Hybrid Agile with iterative releases

- Sprint cadence: 2-week sprints with clear acceptance criteria and demo.
- Continuous feedback: incorporate user feedback each sprint into backlog.
- Risk-first sequencing: implement safety-critical thresholds, consent, and alerts early.

- Staged development phases

2.2.2 Discovery and specification (2–3 weeks)

- Confirm user journeys, finalize MVP scope, draft rules and CRRS v1 spec.

2.2.3 Architecture and data design (1–2 weeks)

- Define services, APIs, database schema, message queues, and security controls.

2.2.4 MVP build (6–8 weeks)

- Frontend: onboarding, multilingual logging, dashboards, CRRS visualization.
- Backend: auth, vitals/lifestyle APIs, rules engine, notifications, caregiver flows.
- AI/NLU: voice-to-structured logging (phase 1 with constrained intents).

2.2.5 Pilot and refinement (4–6 weeks)

- Small cohort pilot, bug fixes, UX polish, template tuning, metric baselines.

2.2.6 Hardening and scale readiness (2–4 weeks)

- Load testing, security checks, monitoring/observability, CI/CD improvements.

2.2.7 Quality assurance

- Test strategy: unit tests for rules/CRRS math, integration tests for API and queue flows, end-to-end user journeys, i18n validation, and negative tests for safety logic.
- Documentation: API contracts, configuration of thresholds, message catalogs, and data dictionary.

2.3 Planning and Scheduling

2.3.1 Milestones and timeline (indicative 16–20 weeks)

M1: Requirements & UX prototype (Week 1–3)

- Deliverables: user flows, wireframes, content templates, CRRS v1 spec.
- Metrics readiness plan and data governance checklist.

M2: Architecture & foundations (Week 4–5)

- Deliverables: DB schema, service skeletons, auth/consent, logging scaffolds.

M3: Core MVP build (Week 6–11)

- Frontend: onboarding, multilingual forms, vitals/symptoms logging, dashboards.
- Backend: rules engine v1, CRRS engine v1, notifications, caregiver module.
- Voice/NLU: intent extraction for BP/glucose entries in EN/HIN.
- Security: encryption, RBAC, audit logs; basic observability.

M4: Pilot (Week 12–15)

- Onboard pilot users, collect feedback, fix issues, tune thresholds and messages.
- Track retention, logging frequency, safety event responsiveness.

M5: Hardening & Release Candidate (Week 16–18/20)

- Load/security testing, error budget/SLA targets, CI/CD pipelines, documentation.

2.3.2 Resource plan

- Team: 1 product manager, 1 UX/UI, 2 frontend, 2 backend, 1 data/ML-NLU, 1 QA, 1 DevOps/security (shared).
- Tools: issue tracker, design system, CI/CD, logging/monitoring, feature flags, and analytics.

2.3.3 Risk management

- Data quality/engagement risk: mitigate with voice-first logging, reminders, simple flows; monitor early retention.
- Safety event handling: strict testing and staged rollout for alerts/escalation; on-call rotation.
- Multilingual accuracy: start with limited languages/templates; expand after validation.
- Scope creep: adhere to MVP must-haves; use feature flags to defer nice-to-haves.

2.3.4 Dependencies and assumptions

- SMS/notification provider availability and DLT registration compliance.
- Browser-based access suffices for MVP; native apps deferred.
- Users possess basic BP/glucose measurement access; app does not replace medical advice.

This methodology ensures SwasthyaSarthi is built iteratively, with safety, explainability, and engagement at its core, and is evaluated against clear control and adherence outcomes before expanding features or scale.

2.3.5 Project Timeline Table

Week	Milestone	Key Deliverables
1–3	M1: Requirements & UX Prototype	- User flows - Wireframes - Content templates - CRRS v1 specification - Metrics readiness plan - Data governance checklist
4–5	M2: Architecture & Foundations	- Database schema - Service skeletons - Authentication & consent module - Logging scaffolds
6–11	M3: Core MVP Build	- Frontend: Onboarding, multilingual forms, vitals/symptoms logging, dashboards - Backend: Rules engine v1, CRRS engine v1, notifications, caregiver module - Voice/NLU: Intent extraction for BP/glucose entries (EN/HIN) - Security: Encryption, RBAC, audit logs, basic observability

12–15	M4: Pilot	- Onboard pilot users - Collect feedback - Fix issues - Tune thresholds and messages - Track retention, logging frequency, safety event responsiveness
16–18/20	M5: Hardening & Release Candidate	- Load & security testing - Error budget/SLA targets - CI/CD pipelines - Documentation

Table: 2.3.5 Project Timeline

2.3.6 Gantt Chart

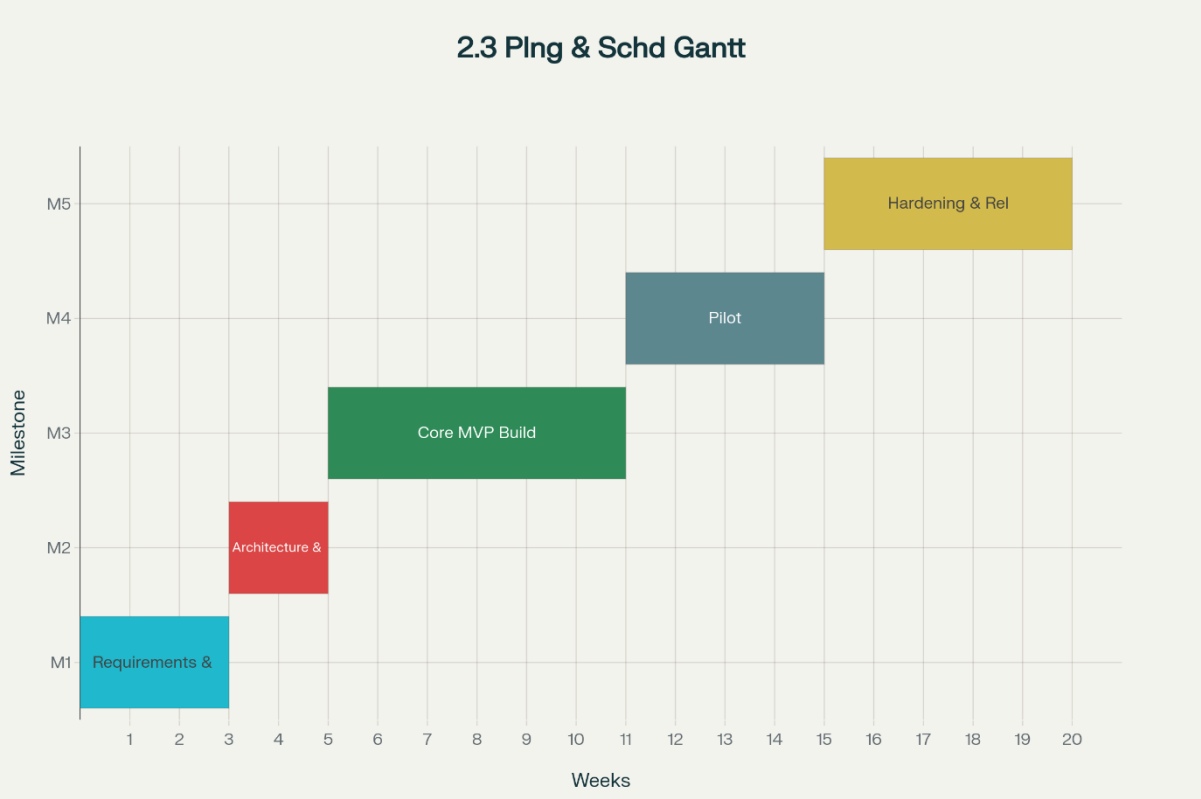

Fig. : 2.3.6 Gantt Chart

2.3.7 Data Model Design (JSON) Formate

{

"name": "Ramesh Kumar",

"age": 45,

"gender": "Male",

"existingConditions": ["Diabetes"],

"vitals": {

"bp": "140/90",

"spo2": 94,

"16eartrate": 88

},

"symptoms": ["fatigue", "blurred vision"],

"environment": "Dusty village",

"lifestyle": "Sedentary"

}

2.3.8 System Architecture Diagram

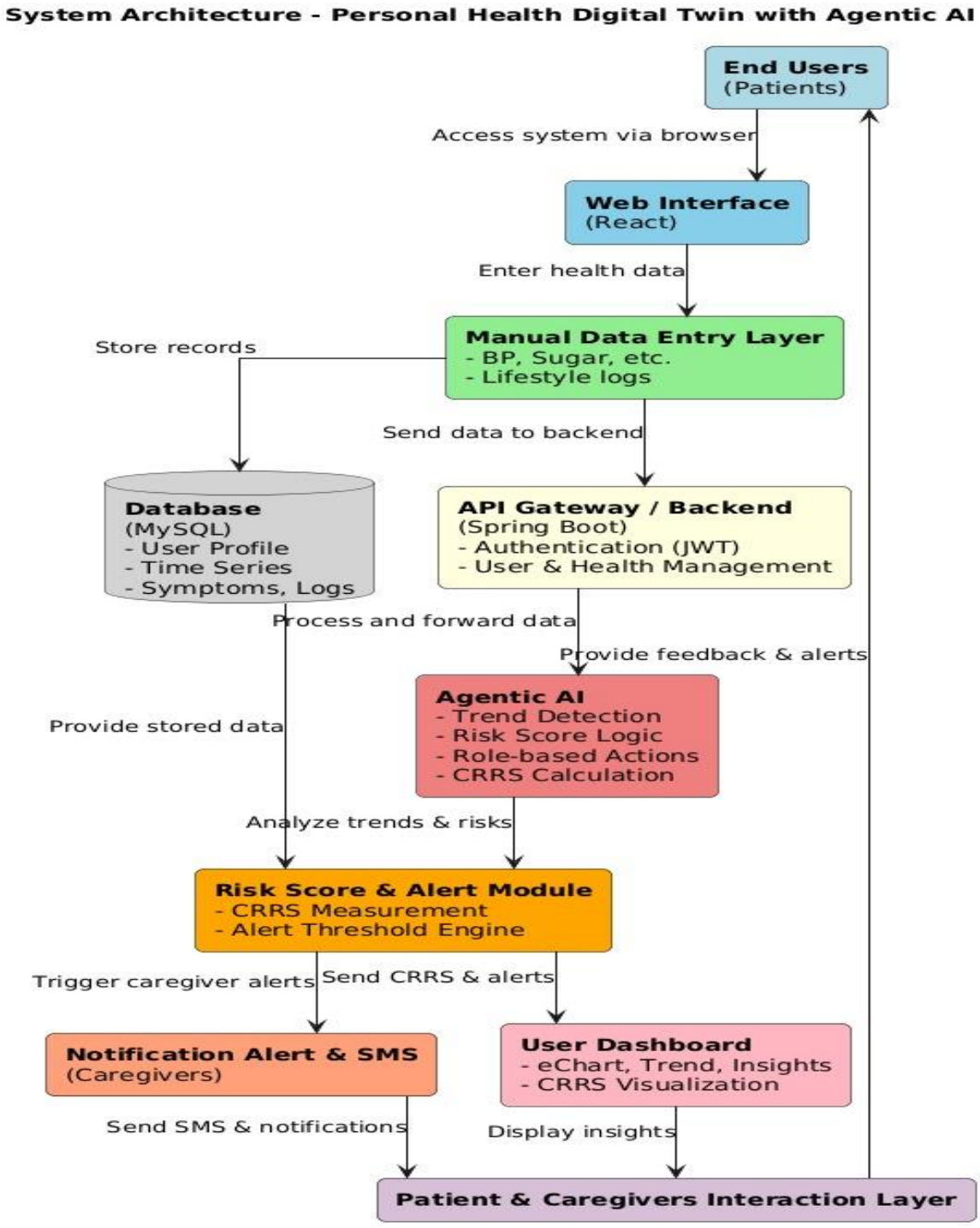

Fig. 2.3.4

CHAPTER 3

REQUIREMENTS AND ANALYSIS

3.1 Problem Definition

SwasthyaSarthi addresses the persistent, last-mile management gap for adults living with Type 2 Diabetes and Hypertension in India by:

- Enabling low-friction, multilingual, voice-assisted logging of vitals, symptoms, and adherence.
- Providing timely, personalized, explainable guidance and alerts based on trends and thresholds.
- Translating daily behavior and vitals into an understandable cardio-renal risk signal to motivate action.
- Sustaining engagement with behavior design (streaks, points, quests) and caregiver loops for safety.
- Ensuring privacy and consent management aligned with Indian data protection norms.

The target outcomes are improved adherence, higher time-in-control for BP and glucose, faster response to safety events, and sustained engagement over 12+ weeks.

3.2 Feasibility Study

3.2.1 Technical Feasibility

Baseline strengths and constraints:

- Team strengths: Java, Spring, Spring Boot, SQL databases. This suits core APIs, domain logic, and transactional workflows.

- Architecture approach: Service-oriented with a strong Java/Spring core, complemented by specialized services where needed (e.g., Python/FastAPI for AI/NLU). Use asynchronous messaging for alerts and background processing.
- Data profile: Mix of transactional data (user, consent, caregiver, configuration) and time-series vitals/CRRS; requires optimized storage for append-heavy writes and trend queries.

Recommended stack (leveraging existing expertise, plus targeted additions):

- Backend (Core): Spring Boot (Java 21+), Spring WebFlux (for reactive endpoints where beneficial), Spring Data JPA for transactional entities, and Spring Security for auth/RBAC.
- Rules and guidance: A rules engine layer in Java (e.g., Drools or a custom rules module with well-defined YAML/JSON rule packs) for transparency and easy iteration of thresholds, trends, and escalations.
- Time-series data: PostgreSQL + TimescaleDB extension for efficient time-series storage and queries (rolling averages, trend windows) while keeping SQL familiarity.
- Messaging: RabbitMQ or Apache Kafka for event-driven alerts, CRRS recomputation, and notification pipelines.
- AI/NLU and voice:
    - Phase 1: Intent and entity extraction for vitals via a lightweight Python service (FastAPI) using pre-trained ASR/NLU APIs where available; integrate with Java core over REST or gRPC.
    - Phase 2+: Optional on-device/offline ASR exploration or fine-tuned NLU for domain terms; retain Python for faster experimentation and libraries.
- Frontend: React + MUI with i18n libraries (e.g., react-i18next) for multilingual UI; PWA setup for installable web app and offline caching where safe.

- Notifications: Provider SDKs for SMS/push, DLT compliance for India; templating managed in backend with localization.
- Security: TLS1.3, AES-256 at rest (cloud KMS), JWT with refresh tokens, RBAC, consent artifacts, audit logs, rate limiting, secrets management.
- Observability: OpenTelemetry, centralized logs, metrics, alerts, SLOs for critical flows (alerts, CRRS updates).

Feasibility by feature:

- Multilingual, voice-first logging: Feasible with React+i18n and ASR/NLU via Python microservice; low risk if scoped to constrained intents initially.
- Explainable CRRS (deterministic v1): Highly feasible in Java with auditable drivers and reason codes; later extendable to ML without breaking interfaces.
- Real-time nudges/alerts and caregiver escalation: Feasible with event-driven pipeline and rules engine; requires rigorous testing for safety and false positives.
- Privacy and consent: Feasible with Spring Security, auditable consent records, and data minimization patterns; requires careful design.

Risks and Mitigations:

- ASR accuracy across languages/dialects – *Risk:* Variations in pronunciation and code-mixing may reduce speech recognition accuracy. *Mitigation:* Implement support for limited languages initially, use guided prompts, and provide confirmation screens with fallback to manual entry.
- Data quality and engagement – *Risk:* Incomplete or inconsistent user inputs may affect accuracy and adoption. *Mitigation:* Incorporate guided data-entry flows, send reminders, use streak-based incentives, and monitor retention to iteratively improve content.

- Complexity creep in rules – *Risk:* Rapid expansion of rule sets may make system unmanageable. *Mitigation:* Maintain version-controlled rule packs, apply configuration flags, adopt staged rollouts, and develop a simulation mode for offline testing.
- Time-series query performance – *Risk:* Large-scale historical data may slow analytics. *Mitigation:* Leverage TimescaleDB continuous aggregates, optimize indexing, and archive raw data periodically while ensuring accessibility for audits.

Conclusion: Technically feasible with a Java/Spring core, SQL familiarity preserved via PostgreSQL/TimescaleDB, and selective Python services for AI/NLU where the ecosystem is stronger.

3.2.2 Economic Feasibility

Assumptions for an MVP (6–9 months horizon):

- Team Composition**:** Engage a balanced team including a Project Manager (0.5–1 FTE), UX/UI Designer (1), Frontend Developers (2), Backend Developers (2), Data/ML–NLU Specialist (0.5–1), QA Engineer (1), and DevOps/Security Engineer (0.5 shared).
- Cloud Infrastructure Costs**:** Deploy using managed services such as PostgreSQL with TimescaleDB, container orchestration (Kubernetes/ECS), and scalable storage/bandwidth. Adopt autoscaling policies and reserved instances post-stabilization to minimize recurring expenses.
- Third-Party Service Costs**:** Utilize ASR/NLU APIs billed per minute, DLT-compliant SMS gateways with per-message fees, and cost-effective email/push notification services.
- Cost Optimization Measures: Leverage open-source components where feasible, implement usage monitoring, and negotiate volume-based discounts with service providers as the user base grows

Cost control strategies:

- Use open-source core (PostgreSQL, Timescale community, RabbitMQ) where viable; upgrade to managed services as usage grows.
- Start with API-based ASR to avoid up-front model training; negotiate volume discounts later.
- Feature-flag ML features; only scale GPU/ML infra when validated.
- Optimize SMS usage with in-app notifications and digest messages where appropriate.

Value and ROI rationale:

- Measurable outcomes: Improvements in control bands, adherence, and reduced safety events can justify institutional partnerships (clinics, employers, payers).
- Modular rollout: Patient-only MVP, then optional clinician views or program integrations, aligning spend with demonstrated impact.

Conclusion: Economically viable for an MVP with controlled scope, open-source leverage, and API-based AI services; scalable cost structure tied to value realization.

3.2.3 Operational Feasibility

People and process:

- Fit with team skills: Java/Spring centricity ensures high development velocity for core features; Python is contained to AI/NLU pieces with clear interfaces.
- Support and operations: On-call rotation for alerts pipeline; runbooks for incident response; SLOs for CRRS latency and notification delivery.
- Content operations: Centralized localization and message catalogs; versioned clinical thresholds; governance for rule changes with sign-off.

- Compliance and data protection: Consent workflows, data minimization, purpose limitation; clear privacy notices; PII/PHI handling policies; regular security reviews.

Deployment and scale:

- Environments: Dev → Staging → Prod with blue/green or canary releases for safety features.
- Monitoring: Health checks, queue backlogs, notification delivery success, CRRS compute times; dashboards for operational visibility.
- User onboarding: Simple flows; educational tooltips; caregiver addition; consent and privacy acknowledgment.
- Partnerships: Optional later integrations with public programs or provider networks; maintain decoupled architecture (API-first).

Conclusion: Operationally feasible with standard DevOps and product operations; key is disciplined governance of clinical rules, localization, and safety escalations.

3.3 Functional Requirements

3.3.1 User management and onboarding

- Register/login with email/phone; JWT-based sessions; password reset/OTP.
- Capture demographics, diagnoses (T2D, HTN), baseline vitals, medications, target ranges.
- Consent management and optional caregiver linking with scoped permissions.

3.3.2 Multilingual, low-friction data capture

- Log vitals: BP (SBP/DBP), glucose (FPG/PPG/RBG), HR, weight.
- Log medications (dose, time, adherence), lifestyle (activity proxy/steps), and symptoms.

- Voice-first logging workflow (EN/HIN v1) with confirmation screens; fallback to forms.
- Backdated entries (bounded), edit/correct entries with audit trail.

3.3.3 Risk computation and explainability

- Daily CRRS (Cardio-Renal Risk Score) computation with deterministic rules.
- Driver breakdown (vitals, adherence, lifestyle, symptoms) and change explanations ("why score changed today").

3.3.4 Guidance, nudges, and safety

- Rules-based nudges (missed logs, rising trends), reminders, and habit-building quests.
- Safety detection and escalation (e.g., SBP ≥ 180 or DBP ≥ 110; glucose < 70 or > 300).
- Multilingual message templates with severity levels; caregiver alerts for high-severity events.

3.3.5 Dashboards and insights

- Personal dashboard showing trends (7/14/30-day), CRRS trajectory, and achievements.
- Insights feed with contextual tips mapped to recent trends and adherence patterns.

3.3.6 Notifications

- In-app notifications; SMS for critical alerts and selected reminders; push (via PWA) where available.
- Delivery status tracking and retry logic.

3.3.7 Data governance and security

- Role-based access (user, caregiver, admin), least-privilege access.
- Audit logs for authentication, data edits, rule executions, and notifications.

3.3.8 Admin and operations

- Manage localization catalogs and rule versions.
- View system health (queues, notification success), feature flags, and configuration.

3.4 Non-Functional Requirements

3.4.1 Performance and scalability

- P95 API latency: < 300 ms for standard reads/writes; < 1 s for CRRS read.
- CRRS daily batch/near-real-time updates within 2 minutes of relevant event.
- Horizontally scalable stateless services; time-series optimized reads.

3.4.2 Reliability and availability

- Target availability: 99.5% for MVP; graceful degradation if AI service is down.
- Idempotent processing for events; at-least-once delivery with deduplication.

3.4.3 Security and privacy

- TLS 1.3, AES-256 at rest (KMS), OWASP controls, rate limiting, secrets management.
- GDPR/DPDPA-aligned consent, data minimization, purpose limitation.
- PHI/PII segregation; field-level encryption for sensitive attributes.

3.4.4 Compliance and auditability

- Verifiable consent artifacts; audit logs immutable and retained per policy.
- Configurable data retention and deletion workflows.

3.4.5 Internationalization and accessibility

- Multilingual UI (EN/HIN v1), RTL readiness later; WCAG AA where feasible.
- Simple language and icons; supportive confirmations for ASR-driven inputs.

3.4.6 Observability

- Centralized logging, metrics, traces (OpenTelemetry).
- SLOs for notification latency, CRRS freshness, and alert pipeline success rate.

3.4.7 Maintainability and extensibility

- Modular architecture with clear contracts (REST/gRPC); versioned APIs/rules.
- Infrastructure as code; CI/CD with automated tests and quality gates.

3.4.8 Usability and engagement

- Onboarding completion rate > 80%; median 4+ logging days/week by week 4.
- Clear error messages; offline-friendly PWA caching for non-sensitive assets.

3.5 Technical Requirements

3.5.1 Hardware Requirements

3.5.1.1 Server-side (cloud or on-prem; indicative for MVP scale)

- 2–3 application nodes for Java/Spring Boot (each: 2–4 vCPU, 4–8 GB RAM).

- 1 Python AI/NLU service node (2 vCPU, 4–8 GB RAM) or use managed ASR API.
- PostgreSQL/TimescaleDB: managed instance with 2–4 vCPU, 8–16 GB RAM, SSD storage (100–200 GB to start), PITR enabled.
- Message broker (RabbitMQ/Kafka): 2–3 node cluster (2 vCPU, 4 GB RAM each) or managed.
- Caching layer (optional for later scale): Redis (2 vCPU, 4 GB RAM).
- Storage for logs/metrics (e.g., managed ELK/CloudWatch/Prometheus stack).

3.5.1.2 Client-side

- Modern smartphones/PCs capable of running a PWA-enabled browser (Chrome, Edge, Safari, Firefox).
- Microphone access for voice logging.

3.5.2 Software Requirements

3.5.2.1 Backend

- Java 17+, Spring Boot (Web/WebFlux, Data JPA, Security, Validation, Actuator).
- PostgreSQL 14+ with TimescaleDB extension for time-series.
- Rules engine: Drools or custom rules module with YAML/JSON rule packs.
- Messaging: RabbitMQ or Kafka client libraries.
- Notification integrations: SMS gateway SDKs (DLT-compliant), email, web push.
- Auth: JWT, refresh tokens; Spring Security with RBAC.

- Observability: OpenTelemetry SDK, log aggregation, metrics (Prometheus/Grafana or cloud-native).

3.5.2.2 AI/NLU and voice

- Python 3.10+, FastAPI for microservice.
- ASR/NLU via APIs initially (e.g., domain-agnostic cloud ASR) with custom post-processing.
- Entity extraction for vitals; language detection; deterministic slot-filling and confirmation.

3.5.2.3 Frontend

- React 18+, TypeScript, MUI, react-query/RTK Query, react-i18next for i18n.
- PWA configuration; form validation; charting library for trends.

3.5.2.4 DevOps and tooling

- Containerization with Docker; orchestration via Kubernetes/ECS.
- CI/CD (GitHub Actions/GitLab CI); IaC (Terraform).
- Secrets management (Vault/cloud KMS), SAST/DAST scanners, dependency audit.

3.5.2.5 Security and compliance

- Key rotation, token blacklisting, device/session management.
- Data retention tooling and deletion workflows.

3.5.3 Network Requirements

3.5.3.1 External connectivity

- Stable internet for client devices: minimum 3G/4G; typical payloads small (<100 KB per request).
- Outbound secure connectivity from services to:
  - SMS/notification providers (HTTPS).
  - ASR/NLU API endpoints (HTTPS) if using managed services.
  - Email/push services.

3.5.3.2 Internal service networking

- Encrypted intra-cluster communication (mTLS where supported).
- Private VPC subnets for DB and brokers; security groups/firewalls restricting access.
- API gateway or ingress controller with rate limiting and WAF rules.

3.5.3.3 Performance targets (network-related)

- Notification dispatch latency: < 30 seconds P95 from event to delivery request.
- CRRS update propagation: < 120 seconds from relevant input event.

3.5.3.4 Compliance and resilience

- DLT registration for SMS templates and headers in India.
- Redundant paths for notification providers (primary/secondary).
- Backoff and retry with circuit breakers for upstream dependencies.

Chapter 4
Technology Stack

4.1 Front-End Technology

4.1.1 React 18

- Rationale: Mature component ecosystem, strong typing for complex health data forms and Digital Twin visualizations; PWA-ready for installable app behavior on low-end devices.

4.1.2 UI Layer: MUI (Material UI)

- Rationale: Accelerated delivery of accessible, consistent UI; sensible defaults for responsiveness and WCAG-aligned components.

4.1.3 Internationalization: react-i18next

- Rationale: Robust multilingual support for India's contexts; runtime language switching, ICU message formats; critical for an engaging twin-guided experience.

4.1.4 State/Data: React Query (or RTK Query)

- Rationale: Cache-first data fetching with auto-refetch, background sync for time-series dashboards and CRRS freshness.

4.1.5 Visualization

- Trend charts and Digital Twin panels (vitals time-series, risk drivers, "what changed today"); leverage canvas/SVG-based charting for performance on mobile.

4.1.6 PWA Enhancements

- Installable on Android/desktop; offline asset caching for non-sensitive resources; guarded offline forms only if privacy constraints are satisfied.

How it supports the Digital Twin

- Presents longitudinal time-series of vitals and behaviors, aligned with digital twin principles where the "virtual replica" continuously reflects the physical entity and supports simulation/prediction views.
- Multilingual, voice-first logging flows reduce friction to keep the twin synchronized with the physical state, supporting the bidirectional loop core to digital twins.

4.2 Back-End Technology

4.2.1 Core Services: Java 21+, Spring Boot

- Modules: Spring Web/WebFlux, Spring Data JPA, Spring Security, Validation, Actuator.
- Rationale: Team expertise; reliable foundation for domain logic, transactions, and secure APIs.

4.2.2 Rules and Guidance Service

- Option A: Drools (BRMS) in Java for transparent, versioned clinical thresholds, trend rules, and escalation logic.
- Option B: Custom YAML/JSON rules compiled into Java evaluators for simpler ops and deterministic behavior.
- Supports OODA "Decide/Act" with auditable mappings from observations/trends to actions.

4.2.3 Agentic AI Orchestrator (OODA loop)

- Observe: subscribe to events (new logs, anomaly detections).
- Orient: contextualize with user baseline, recent trends, thresholds, and constraints.

- Decide: pick nudge/alert, simulate options, apply governance; can call ML/LLM where needed.
- Act: post in-app notification, queue SMS/push, trigger caregiver escalation; feed results back into loop.
- Architecture mirrors agentic AI patterns grounded in OODA to convert reactive tools into proactive systems.

4.2.4 Eventing and Workflows: RabbitMQ

- Rationale: Decouple ingestion, CRRS recompute, alerting, notifications, and caregiver workflows; backpressure and replay for safety-critical pipelines.

4.2.5 AI/NLU Microservice (Voice and Structured Extraction)

- Python 3.10+ with FastAPI
- Rationale: Python's ASR/NLU ecosystem accelerates prototyping; clean REST/gRPC contract to Java core.
- Phase 1: Use managed ASR; deterministic entity extraction; confirmation UX to mitigate ASR errors.
- Phase 2: Domain-tuned NLU; optional offline/on-device models for resilience.

4.2.6 Notifications

- SMS (DLT-compliant), email, and web push; templated, localized, severity-tagged; deliverability metrics and retries.

4.2.7 Security and Compliance

- TLS1.3, AES-256 at rest (KMS), RBAC, consent artifacts, PHI/PII segregation, immutable audit trails; rate-limiting, WAF.

Why this suits a Digital Twin + Agentic AI

- Digital twin demands bidirectional flow: incoming observations update the virtual replica; decisions/actions flow back to the patient—this is central to formal DT definitions and the 5Is (individualized, interconnected, interactive, informative, impactful).
- Agentic AI requires continuous, autonomous OODA cycling with governance and feedback—supported by rules/queues and optional ML/LLM augmentation.

4.3 Database

4.3.1 Primary Store: PostgreSQL 14+ with TimescaleDB extension

4.3.1.1 Schema split:

- Transactional: users, consents, caregivers, medications, configurations.
- Time-series: vitals, symptoms, adherence, lifestyle metrics, CRRS snapshots, and risk drivers.

4.3.1.2TimescaleDB advantages:

- SQL-native development on a high-performance time-series engine (hypertables, chunking, compression, retention).
- Time-window queries, continuous aggregates for rolling averages and anomaly detection; proven suitability for healthcare monitoring use cases.

Why it fits: Digital twin requires efficient longitudinal analytics and frequent updates, while staying in a familiar SQL world for the Spring team.

4.3.2 Secondary/Supporting

- Redis (optional): caching of user baselines, rule packs, and content catalogs to lower latency.

- Object storage: encrypted storage for voice snippets if retention is justified; otherwise discard after transcription to minimize PHI surface.

Digital Twin Alignment

- The DT's "connection" to the physical patient is realized via efficient ingestion and query of time-series and state snapshots, a core element in DT literature.
- Continuous aggregates and hypertables enable near-real-time orientation and decision stages without bespoke TSDB skills, preserving team velocity.

4.4 APIs and Other Tools

4.4.1 API Design

- REST-first (JSON), with clear resource contracts and versioning;
- Core APIs: Auth/Consent, Profile/Baseline, Vitals/Symptoms/Lifestyle, Medications/Adherence, CRRS and explanations, Nudges/Alerts/Notifications, Caregiver, Localization, Rulepacks.
- Event APIs: Publish/subscribe topics for new measurements, anomaly detections, CRRS changes, and escalation events.

4.4.2 Agentic AI Interfaces

- Observation bus: events from data capture and analytics feed the Observe stage.
- Orientation services: recent windows, baselines, thresholds; simulation endpoints for "what-if" evaluation.
- Decision/Action: rules/LLM endpoints gated by governance; notification and caregiver connectors.
- OODA structure follows established agentic AI practice to achieve proactive, adaptive behavior.

4.4.3 Voice/ASR and NLU

- Ingestion: Browser captures audio; sent to ASR (managed API) via AI service.
- Post-processing: Deterministic slot-filling for vitals, language detection, confidence gating; human-readable confirmations to keep the twin accurate.
- This keeps the DT "interconnected" and "interactive," enabling continuous updates from real-world signals.

4.4.4 DevOps/Platform

- Containers: Docker; Orchestration: Kubernetes/ECS; IaC: Terraform.
- CI/CD: GitHub Actions/GitLab CI; quality gates, SAST/DAST, dependency scanning.
- Observability: OpenTelemetry traces/metrics/logs; dashboards for queue lag, CRRS latency, notification success—vital for safety loops.
- Data lifecycle: retention/compression policies in TimescaleDB; backups and PITR.

4.4.5 Governance and Safety

- Rule versioning and audit: every nudge/alert ties back to rule version and evidence windows for explainability.
- Safeguards: rate limits, duplicate detection, and caregiver escalation throttles; sandbox/simulation mode for rule testing before promotion.

How these tools serve the Digital Twin and Agentic AI mandate

- A digital twin in healthcare explicitly requires a physical entity, its virtual replica, and a live connection that enables two-way impact—sustained by our ingestion APIs, time-series store, rules, and notifications.

- Agentic AI with OODA delivers proactive, context-aware guidance: Observe (events), Orient (context windows, thresholds), Decide (rule/LLM), Act (nudge/alert/caregiver), then learn from outcomes—improving adaptability over time

Chapter 5:

Conclusion, Limitations and Future Work

5.1 Conclusion

SwasthyaSarthi is designed to bridge the last-mile management gap in India's dual epidemic of Type 2 Diabetes and Hypertension by combining a high-fidelity Digital Twin with an agentic AI guidance loop to deliver continuous, explainable, and multilingual support for day-to-day self-management. By centering the architecture on streaming observations (vitals, adherence, lifestyle), near-real-time orientation (baselines, rolling trends), transparent decisioning (rules first, ML-ready), and timely action (nudges, alerts, caregiver escalation), the system directly targets low diagnosis–treatment–control rates and the associated cardio-renal risk that population studies continue to document at scale. The chosen stack—Spring/Spring Boot core with PostgreSQL+TimescaleDB, complemented by a Python microservice for ASR/NLU balances team strengths with best-fit components for time-series analytics and voice-first logging central to keeping the Digital Twin synchronized. This foundation positions SwasthyaSarthi to measurably improve adherence, increase time-in-control for BP and glucose, and reduce delays in responding to safety events, while maintaining privacy, auditability, and operational robustness aligned to India's regulatory and infrastructural realities.

5.2 Limitations/Constraints

- Data quality and sparsity: Reliance on user-entered or intermittent device data can introduce gaps, noise, and bias; confirmations, guided flows, and fallback form entry mitigate but do not eliminate this constraint.
- ASR/NLU variability: Speech recognition accuracy varies across accents, code-mixing, and environments; scoped intents, confidence gating, and human-readable confirmations are required safeguards.

- Rules-first scope: The initial deterministic Cardio-Renal Risk Score (CRRS) and guidance rules prioritize explainability over model complexity; predictive performance is intentionally conservative until sufficient validated data supports ML upgrades.
- Clinical boundaries: The system augments but does not replace medical care; emergency detection is threshold-based and errs toward caution, which may increase false positives and user alert fatigue if not tuned carefully.
- Language coverage: Initial multilingual support is limited and will expand iteratively; nuanced local idioms and health literacy levels require ongoing content operations and testing.
- Connectivity and device constraints: PWA-first design minimizes friction, but low bandwidth and older devices may still affect voice logging and media-heavy features.
- Integration scope: Early phases do not include EHR/AB-HWC integration or clinician dashboards, which limits bidirectional clinician oversight until later phases.
- Compliance evolution: Data protection regulations and telecom/DLT rules evolve; maintaining compliance requires continuous monitoring, documentation, and periodic audits.

5.3 Future Work

5.3.1 ML-enhanced risk and personalization:

- Upgrade CRRS with validated ML models for individualized risk trajectories and event prediction, while retaining explainability via feature attributions and rule overlays.
- Add adaptive coaching policies using reinforcement learning within strict guardrails and human-in-the-loop governance.

5.3.2 Expanded signals and devices:

- Integrate affordable connected BP/glucose devices and wearables for higher-fidelity, lower-friction data capture; support intermittent offline buffering.
- Introduce meal photo logging with lightweight, privacy-preserving nutrition estimation.

5.3.3 Clinical and program integrations:

- Build clinician dashboard, secure messaging, and summary exports; integrate with public programs (e.g., IHCI/NP-NCD) and health information exchanges via standardized APIs.
- Add referral workflows and pharmacy linkages for adherence support.

5.3.4 Content and languages at scale:

- Extend supported languages and dialect-sensitive templates; co-design content with patient groups to enhance cultural fit and readability.
- Gamified care plans (“quests”) co-created with clinicians for comorbidity pathways.

5.3.5 Safety, governance, and validation:

- Formalize a Rules & Models Governance Board; implement shadow testing, canary rules, and post-deployment drift/impact monitoring.
- Conduct prospective cohort pilots measuring control rates, retention, and safety event responsiveness; publish results.

5.3.6 Agentic AI maturity:

- Introduce scenario simulation (“what-if”) at the edge of the Digital Twin for user-facing risk communication and planning.
- Add outcome-aware feedback loops that adjust message timing and channel based on user responsiveness.

References

[1] The Lancet Diabetes & Endocrinology (ScienceDirect landing). (2023). Metabolic non-communicable disease health report of India (ICMR INDIAB). https://www.thelancet.com/journals/landia/article/PIIS2213-8587(23)00119-5/fulltext

[2] BBC News. (2023, June 9). Lancet study: More than 100 million people in India diabetic. https://www.bbc.com/news/world-asia-india-65852551

[3] National Medical Journal of India. (2024, Feb 3). Lessons Learnt from the ICMR–INDIAB Study. https://nmji.in/lessons-learnt-from-the-icmrindiab-study/

[4] JAMA Network Open. (2023, Oct 2). Hypertension Diagnosis, Treatment, and Control in India, 2019–2021. https://jamanetwork.com/journals/jamanetworkopen/fullarticle/2810984

[5] Singh, S., et al. (2023). Hypertension Control Cascade and Regional Performance in India: A Repeated Cross-Sectional Analysis (2015–2021). Cureus. https://pmc.ncbi.nlm.nih.gov/articles/PMC10042544/

[6] WHO India. (2022, June 2). India Hypertension Control Initiative: a high-impact and low-cost solution. https://www.who.int/india/news-room/detail/02-06-2022-india-hypertension-control-initiative--a-high-impact-and-low-cost-solution

[7] Press Information Bureau, Government of India. (2023, Sept 9). Symposium on "Recent Trends and Siddha Management..." (ICMR–INDIAB figures referenced). https://www.pib.gov.in/PressReleasePage.aspx?PRID=1955800

[8] Press Information Bureau, Government of India. (2023). National programme updates and IHCI/NP-NCD context. https://www.pib.gov.in/PressReleasePage.aspx?PRID=1944600

[9] Vallée, A., et al. (2023). Digital twin for healthcare systems: opportunities and challenges. Frontiers in Digital Health. https://www.frontiersin.org/journals/digital-health/articles/10.3389/fdgth.2023.1253050/full

[10] Vallée, A., et al. (2023). Digital twin for healthcare systems. PMC version. https://pmc.ncbi.nlm.nih.gov/articles/PMC10513171/

[11] Nature npj Digital Medicine. (2024). Agentic AI and healthcare decision support (context for agent loops). https://www.nature.com/articles/s41746-024-01073-0

[12] ScienceDirect Review. (2024). The status quo and future prospects of digital twins for healthcare. https://www.sciencedirect.com/science/article/pii/S2950489924000423

[13] Cureus PDF. (2023). Hypertension Control Cascade and Regional Performance in India (NFHS-4 vs NFHS-5). https://www.cureus.com/articles/138955-hypertension-control-cascade-and-regional-performance-in-india-a-repeated-cross-sectional-analysis-2015-2021.pdf

[14] SMS Gateway Center. (n.d.). DLT SMS in India: registration and compliance overview. https://www.smsgatewaycenter.com/dlt-sms/