Healthify Project Specification Document: Architecture, Database, and UI/UX Design with FHIR and Supabase Integration

1. Executive Summary

This document provides a comprehensive project specification for Healthify, a digital health platform designed to revolutionize healthcare delivery by offering accessible, integrated, and continuous telemedicine services. It outlines Healthify's overarching vision, its detailed technical architecture leveraging a microservices approach, and a strategic adoption of a FHIR-native data modeling paradigm. The integration of Supabase for managed PostgreSQL and authentication services is strategically incorporated to enhance backend capabilities and streamline development. The platform's core components, encompassing distinct user interface portals (Patient, Provider, Admin) and a robust microservices layer, are meticulously detailed. Emphasis is placed on the critical role of Fast Healthcare Interoperability Resources (FHIR) Release 4 (R4) in achieving broad interoperability and stringent regulatory compliance, particularly with HIPAA. This architectural and design evolution is anticipated to yield significant benefits, including improved data exchange capabilities with external healthcare systems, fortified security protocols, accelerated development cycles, and a scalable, future-proof foundation poised for continuous expansion and adaptation within the evolving digital health landscape.

2. Introduction to Healthify

2.1 Platform Vision, Scope, and Problem Definition

Healthify is envisioned as a versatile cross-platform telemedicine application, accessible via both mobile and web interfaces. It is dedicated to delivering a comprehensive suite of healthcare services spanning chronic and non-urgent acute care, proactive preventive services, and essential health education. The platform's foundational mission is to address critical deficiencies prevalent within Sri Lanka's current healthcare infrastructure, characterized by fragmented systems, protracted waiting times for consultations, elevated consultation fees, and an underdeveloped focus on preventive health initiatives.

The strategic scope of Healthify encompasses the provision of a holistic telehealth platform, prioritizing accessibility, continuity of care, and preventive services. Its Minimum Viable Product (MVP) integrates essential functionalities, including comprehensive Electronic Health Records (EHR) management, secure video and chat consultations, efficient e-prescribing capabilities, an intuitive appointment scheduling system, robust billing functionalities, and a distinctive subscription-based "Ever Care" Follow-Up Program. All core functionalities are meticulously engineered upon FHIR standards to ensure rapid deployment, seamless data exchange, and future interoperability. The platform will feature dedicated mobile and web applications tailored for patients, healthcare providers, and administrators, facilitating secure consultations, advanced EHR security, streamlined e-prescriptions, flexible pricing models, extensive health education resources, and an integrated health tracker. Healthify's unique "Ever Care" program specifically targets vulnerable populations, such as aged parents, who often lack consistent health monitoring, aiming to provide continuous follow-up care through virtual consultations and home visits.

The prevailing healthcare challenges in Sri Lanka, marked by a cure-focused system, extended waiting periods, prohibitive consultation fees, and limited appointment availability, underscore the necessity for Healthify's intervention. The platform endeavors to establish an affordable, organized, and prevention-driven healthcare ecosystem, empowering Sri Lankans to proactively manage their well-being. The primary audience for Healthify includes patients seeking diverse healthcare services, individuals with existing conditions requiring targeted follow-up and home visit services, health-conscious adults, individuals residing abroad who manage care for loved ones in Sri Lanka, those desiring accessible digital health records, and individuals seeking immediate telemedicine consultations for non-urgent conditions. Healthcare providers, encompassing general practitioners and specialists, represent another key audience, requiring tools for virtual consultations, e-prescribing, and patient monitoring. Therapists, particularly mental health counselors, necessitate secure communication channels and progress note management. Lastly, administrators and CEOs are crucial stakeholders, responsible for platform management, provider onboarding, billing, content updates, and ensuring regulatory compliance.

The explicit commitment to building Healthify upon FHIR standards embodies a foundational strategic imperative. By adopting FHIR from its inception, Healthify strategically positions itself as an integral, interconnected entity within the broader healthcare ecosystem. This forward-thinking approach to interoperability is expected to yield substantial advantages, including a significant reduction in future integration costs, accelerated opportunities for partnerships with other healthcare providers such as pharmacies, laboratories, or other Electronic Medical Record systems, and a considerable competitive advantage by ensuring frictionless data exchange. This proactive stance implicitly prepares the platform for potential future national or regional health information exchange initiatives, thereby ensuring its long-term relevance, adaptability, and scalability within the evolving digital health landscape.

2.2 Document Purpose and Audience

This document serves as a comprehensive project specification, providing a detailed description of the requirements for the Healthify system. It explicates system constraints, defines interfaces, and offers a complete declaration for the development of the system. The primary purpose is to guide the development team in constructing a robust, compliant, and scalable digital health platform.

The intended audience for this specification includes Lead Software Architects, Project Managers, Frontend Development Teams, Backend Development Teams, Database Administrators, Quality Assurance Engineers, and all key stakeholders involved in the technical implementation and oversight of the Healthify project.

3. High-Level System Architecture

3.1 Overview of Microservices Architecture

Healthify operates on a robust microservices architecture, logically segmented into three primary layers: the User Interface Layer, the API Gateway, and the Microservices Layer. This distributed architectural paradigm inherently supports scalability, enabling independent development, deployment, and scaling of individual services, which allows for greater agility in development and maintenance.

While a microservices architecture aligns effectively with FHIR's granular resource model, which defines "building blocks" and "atoms" of health information exchange, this distributed approach introduces a critical challenge in maintaining a unified FHIR context and ensuring referential integrity for FHIR `Reference` elements that span across different service boundaries. For instance, an `Appointment` resource residing within the Appointment Service may need to reference a `Patient` resource managed by the Identity Service, or an `Encounter` resource in the EHR Service might reference a `Practitioner` from the IAM Service. To manage and overcome this architectural complexity, careful consideration is given to how these inter-service FHIR references are correctly resolved and how complex queries that involve multiple FHIR resource types (e.g., retrieving all appointments for a patient with a specific chronic condition) can be executed efficiently across service boundaries. A robust API Gateway that possesses an understanding of FHIR resource types, coupled with a shared FHIR server layer or a sophisticated distributed data management strategy, is essential.

3.2 User Interface Layer (Patient, Provider, Admin Portals)

The User Interface Layer comprises three distinct portals, each tailored to the specific needs of its user base. The **Patient Portal** is implemented as a React Native-based Mobile Web App (PWA), providing patients with capabilities for appointment scheduling, access to Electronic Health Records (EHR), secure messaging, prescription refills, and health education resources. The **Provider Portal** offers a React Native-based interface for doctors and therapists, enabling them to efficiently manage appointments, issue prescriptions, and access relevant patient data during consultations. The **Admin Portal** is a React Native-based dashboard designed for administrators to oversee and manage users, content, billing operations, and system analytics.

The underlying technologies for this layer include React Native and React.js, augmented with Tailwind CSS to ensure a responsive user interface that adapts seamlessly across various screen sizes. For real-time communication, WebRTC (utilizing Jitsi Meet or LiveKit) is integrated to facilitate video consultations, while WebSockets are employed to enable real-time chat functionalities.

The consistent use of React Native for all three portals implies a strategic approach aimed at maximizing code reusability and establishing a unified technology stack. To achieve distinct yet cohesive user experiences for such diverse user roles—patients, clinical professionals, and administrators—within a shared framework, the UI/UX design will be highly adaptive. Each portal will be meticulously tailored to its specific user class, optimizing for their unique workflows and information needs through detailed wireframing and user flow analysis.

3.3 API Gateway

The API Gateway serves as the singular entry point for all client requests directed towards the Healthify system. Its primary functions encompass routing incoming requests to the appropriate microservices, handling comprehensive authentication and authorization processes, and enforcing rate limiting to manage system load effectively.

Key features of the API Gateway include robust Authentication/Authorization capabilities, which integrate seamlessly with a custom Identity Service to ensure secure user access through OAuth 2.0 and JSON Web Tokens (JWT). It also performs critical Request Validation, ensuring that all incoming requests comply with custom FHIR APIs. This involves not only basic schema validation but also validating incoming requests against defined FHIR profiles (e.g., US Core Profile or Healthify's custom profiles) and potentially transforming data to or from internal representations. Such a capability ensures data integrity and consistency, crucial for interoperability. The API Gateway could also play a pivotal role in FHIR resource aggregation or decomposition for complex queries that span multiple microservices, acting as a facade that presents a unified FHIR interface to clients even if internal services are still in transition to full FHIR-nativeness. Furthermore, the gateway implements Rate Limiting mechanisms to manage the load, particularly for non-subscribed users, thereby maintaining system stability and performance. Technology options for the API Gateway include AWS API Gateway or the open-source Kong API Gateway.

3.4 Microservices Layer

The Microservices Layer comprises several independent services, each dedicated to a distinct functional domain within the Healthify platform:

- **Identity and Access Management (IAM) Service:** This service is responsible for custom user registration, login, and profile management for all user types, including patients, providers, and administrators. It implements OAuth 2.0 and SMART-on-FHIR for healthcare-specific authentication, enforces Role-Based Access Control (RBAC), and integrates comprehensive audit logging for HIPAA compliance using FHIR `AuditEvent` and `Provenance` resources. The service is built using Node.js/Express for the backend and Passport.js for OAuth support, with data stored in PostgreSQL, where sensitive fields are encrypted with AES-256.

- **Electronic Health Records (EHR) Service:** Central to Healthify, this service manages the custom storage and management of patient EHRs, including demographics, conditions, allergies, and medications, in a FHIR-compliant format. It supports standard CRUD (Create, Read, Update, Delete) operations, FHIR resource search, and secure file uploads, with provisions for future integration with wearable devices. The technology stack includes Node.js/Express, FHIR libraries (e.g., node-fhir-server-core) for compliance, and PostgreSQL with JSONB support for FHIR resources, encrypted with AES-256. This service will explicitly manage FHIR resources such as `Patient`, `Condition`, `AllergyIntolerance`, `Observation`, `Procedure`, `DiagnosticReport`, and `DocumentReference`.
- **Appointment Scheduling Service:** This service manages appointment booking, rescheduling, and cancellation processes. It provides calendar views, automated reminders, and handles home visit requests, which are subject to administrative review. It is developed with Node.js/Express and integrates with calendar syncing tools such as iCal, storing appointment data in PostgreSQL. This service will primarily manage `Appointment`, `Schedule`, `Slot`, and `ServiceRequest` FHIR resources.
- **Prescription Management Service:** This service enables healthcare providers (doctors) to generate, digitally sign, and deliver e-prescriptions as PDFs. It enforces access restrictions post-issuance and ensures prescriptions are accurately mapped to specific patients. The service utilizes Node.js/Express with PDF generation libraries (e.g., pdfkit) and stores prescription records in PostgreSQL. This service will explicitly use the `MedicationRequest` FHIR resource for comprehensive medication order management, covering all necessary details like dosage, frequency, and patient instructions.
- **Communication Service:** This service facilitates secure, real-time messaging and video consultations. It supports file and image sharing, with size restrictions that may vary based on subscription tiers. The service employs Node.js with WebSockets for chat and WebRTC (Jitsi Meet or LiveKit) for video, storing message data in PostgreSQL. This service will manage the `Communication` FHIR resource for all message exchanges.
- **Billing and Payment Service:** This service manages both pay-per-consultation and subscription-based payments, encompassing Vital Starter, Boost, and Pro tiers. It supports a "Consult Now, Pay Later" option for subscribers and integrates with established payment gateways like Stripe and PayHere. The service is built with Node.js/Express, utilizing Stripe SDK and PayHere API, and stores transaction records in PostgreSQL. This service will utilize `ChargeItem` and `Invoice` FHIR resources for billing artifacts.
- **Content Management Service:** This service is responsible for managing health education resources, including articles, FAQs, and videos, with robust categorization and search functionalities. Patients are able to bookmark their preferred resources for future reference. It uses Node.js/Express with a CMS library (e.g., Strapi) and stores content in PostgreSQL. This service will primarily leverage `DocumentReference` and `Media`

FHIR resources for content management, along with `Questionnaire` and `QuestionnaireResponse` for interactive elements.

- **Analytics and Reporting Service:** This service generates comprehensive reports on user activity, appointment statistics, and EHR insights. It supports the export of reports in various common formats such as PDF and CSV. The service is developed with Node.js/Express, incorporates reporting tools (e.g., Chart.js), and stores analytics data in PostgreSQL, while primarily querying and aggregating data from the FHIR-compliant resources across other microservices. `AuditEvent` resources will be crucial for user activity logs and compliance reporting.
- **Ever Care Program Service:** This unique service manages personalized Non-Communicable Disease (NCD) screenings, scheduled virtual/home visits, and monitors health parameters, providing tailored preventive care content. It uses Node.js/Express and stores its data in PostgreSQL. This service will manage `CarePlan`, `Observation`, `Questionnaire`, and `QuestionnaireResponse` FHIR resources.

3.5 Data Storage Strategy

The data storage strategy for Healthify is diversified to optimize for different data types and access patterns, ensuring compliance and performance.

- **FHIR Resource Storage (JSONB in PostgreSQL):** Clinical data, which aligns directly with FHIR resources (e.g., `Patient`, `Condition`, `Observation`, `Encounter`, `MedicationRequest`), will be stored as JSONB objects within PostgreSQL. This provides the flexibility to store the full FHIR resource structure, including arrays and nested objects, while leveraging PostgreSQL's indexing capabilities for efficient querying of specific FHIR elements. AES-256 encryption for sensitive fields will continue to be a critical security measure.
- **Relational Data (PostgreSQL):** Non-FHIR-specific metadata, lookup tables, and data that are purely administrative or internal to Healthify's business logic (e.g., user authentication details, subscription tiers, payment transaction IDs, content categories) will reside in traditional relational tables within PostgreSQL. This optimizes performance for operations that do not require FHIR's semantic richness.
- **File Storage (AWS S3):** Large binary files such as lab reports, imaging scans, and e-prescriptions (PDFs) will be stored in AWS S3, complete with appropriate access controls. Metadata about these files, crucial for their discoverability and context within the clinical record, will be captured in FHIR `DocumentReference` resources, stored in the JSONB FHIR database.
- **Caching (Redis):** Redis will be used for caching frequently accessed, non-persistent data, such as provider availability, to enhance response times and overall system performance.

All external-facing APIs will strictly adhere to FHIR RESTful API principles, supporting standard CRUD operations on FHIR resources and leveraging FHIR's standardized search parameters.

The API Gateway will be enhanced to validate incoming and outgoing data against FHIR profiles. Internal microservice communication will also prioritize FHIR resource exchange where applicable.

4. Functional Requirements Specification

The functional requirements for Healthify are delineated across four distinct user classes, ensuring that the platform caters to the specific needs and interactions of each group.

4.1 User Class: Non-Subscribed Members (Healthify Guests)

Non-subscribed members, often referred to as guests, have access to a foundational set of functionalities:

- **Patient Registration and Profile Management:** The system facilitates new patient registration using email/phone and password, with optional social login. It validates and verifies registration details. Registered patients can securely log in and manage their profile information, including personal details, medical history, and insurance.
- **Appointment Scheduling and Management:** Patients can search and select healthcare providers based on specialization, availability, and consultation mode (video, chat, home visit). The system provides a clear calendar view of available slots. Patients receive instant appointment confirmations and reminders. They can reschedule or cancel appointments (subject to policy). Payments for consultations are processed on a pay-per-consultation basis via integrated payment gateways.
- **Electronic Health Records (EHR) Management:** The system securely stores and manages patient EHRs, including comprehensive medical history, diagnoses, treatments, lab results, and prescriptions. Patients can manually add certain health parameters (e.g., daily vitals, symptoms) to their EHR.
- **Prescription Refills:** Patients can view existing prescriptions and request refills directly through the portal.
- **Secure Messaging and Communication:** Patients and healthcare providers can communicate securely through the portal, supporting real-time messaging and instant notifications. Attachments and images are permitted, subject to specific file size restrictions, and the number of texts for non-members may be limited.
- **Payment Processing and Billing:** Non-subscribed patients can make payments for services prior to confirming an appointment. The system securely processes online payments utilizing standard payment gateways.
- **Information Resources:** The portal provides a comprehensive repository of informative articles, FAQs, and health tips. This content is categorized, easily searchable, and patients can bookmark their preferred resources.

4.2 User Class: Subscribed Members (Healthify Fam - Vital Starter, Boost, Pro)

All functionalities of Non-Subscribed Members, plus:

- **Patient Registration and Profile Management:** This functionality mirrors that provided to non-subscribed members, allowing for registration, login, and comprehensive profile updates.
- **Appointment Scheduling and Management:** In addition to core appointment features, subscribed patients have the option for "consult now and pay later" and are allocated a limited number of free consultations based on their specific subscription tier.
- **Electronic Health Records (EHR) Management:** Beyond secure storage and management of EHRs, the system supports the integration of external systems, such as wearable devices, for importing and exporting EHR data, enhancing the comprehensiveness of patient records.
- **Prescription Refills:** Similar to non-subscribed members, patients can view and request prescription refills through the portal.
- **Secure Messaging and Communication:** Secure messaging and real-time communication are provided, with attachment and image size restrictions varying based on the user's subscription plan, offering greater flexibility for higher tiers. Subscribed members can also view their complete past chat history.
- **Payment Processing and Billing:** Subscribed patients can make payments for their monthly subscriptions, with the system securely processing online payments via standard payment gateways. The system will manage tiered feature access and dynamically display features or upgrade prompts based on the user's subscription status.
- **Information Resources:** This functionality is consistent with that offered to non-subscribed members, providing access to categorized, searchable, and bookmarkable health articles, FAQs, and tips.
- **Subscription Plan Monitoring:** Subscribed patients can easily view the remaining days of their current subscription plan, providing transparency regarding their membership status.
- **Exclusive Follow-up Care Program ("Ever Care"):** This exclusive program allows subscribed patients to engage in personalized Noncommunicable Disease (NCD) screening programs. It includes scheduled virtual consultations and home visits by doctors, access to a dedicated Preventive Care/Health Advice Portal, and the ability to monitor major health parameters through the system, managed via FHIR `CarePlan` resources.

4.3 User Class: Healthcare Providers

Healthcare providers, including doctors and therapists, are afforded specific functionalities to manage their clinical practice effectively:

- **Login to the System and Profile Management:** The system sends login credentials to providers upon account creation by an administrator. Registered providers can securely log in and update their profile information, including personal details, working history, insurance, and professional achievements.

- **Appointment Management:** Healthcare providers can accept, reschedule, or cancel appointments and receive instant notifications. They can update their available time slots and adjust appointment details (e.g., time per appointment, max patients). Critically, providers can view a patient's medical history and reports *during* a consultation. After the consultation concludes, strict security measures dictate that providers *lose access* to the patient's full history and past records, controlled by fine-grained, context-aware access control mechanisms.
- **Prescription Management:** (Applicable only to doctors) Providers can issue prescriptions at the conclusion of a consultation, which are automatically mapped to the relevant patient as `MedicationRequest` FHIR resources. After a prescription is issued, providers lose direct access to it from the general patient record. Doctors maintain an "issue prescription access point" from their portal, allowing them to access and issue prescriptions anytime, but only to patients assigned to them. The system provides options to add a digital signature to prescriptions for authenticity and generates them as PDFs.
- **Secure Messaging and Communication:** Patients and healthcare providers can communicate securely, supporting real-time messaging and instant notifications. Attachments and images are permitted, subject to file size restrictions. Providers can view their past chat history with patients.
- **Payment Processing and Billing:** Healthcare providers are granted access to the billing and payment history for each patient they have consulted.
- **Analytics and Reporting:** The system generates comprehensive analytics and reports specifically for healthcare providers, encompassing patient demographics, appointment statistics, and EHR insights. These reports are accessible through a user-friendly interface and can be exported in common formats.

4.4 User Class: Healthify Admin

The Healthify Administrator possesses comprehensive control and oversight functionalities:

- **Provider Account Creation and User Management:** The administrator creates user accounts for healthcare providers, grants appropriate system access, and manages user roles and permissions. They can deactivate user accounts for terms violations and manage all user accounts within the system.
- **Payment Oversight and Billing:** The administrator monitors all billing transactions and payment processing activities, including managing integrations with payment gateways and overseeing comprehensive financial reporting.
- **Appointment Oversight:** The administrator has access to view all scheduled appointments across the entire system and can intervene in scheduling conflicts or errors when necessary.

- **Content Management:** The administrator manages the repository of informative articles, FAQs, and health tips, with the ability to add, edit, categorize, or remove content within the information resources section.
- **System Monitoring and Reporting:** The administrator has access to system-wide analytics and reporting tools, enabling them to generate usage reports, user activity logs, and system performance metrics.
- **Audit and Compliance:** The administrator can access comprehensive audit logs to track user activities and system changes. This function is critical for ensuring compliance with relevant regulations, such as HIPAA, by leveraging FHIR `AuditEvent` and `Provenance` resources to capture granular details for auditability and forensics.

5. Non-Functional Requirements and System Constraints

5.1 Security and HIPAA Compliance

Security is paramount for Healthify, especially given the sensitive nature of healthcare data. The platform will employ industry-standard encryption and security measures to safeguard patient data from unauthorized access. This includes the implementation of Transport Layer Security (TLS) for data in transit and AES-256 encryption for data at rest. User authentication will be robust, featuring strong password policies and, optionally, additional authentication factors. The Identity Service will manage authentication using OAuth 2.0/JWT and SMART-on-FHIR, which are well-aligned with healthcare security frameworks. Access controls will be strictly enforced through granular Role-Based Access Control (RBAC) managed by the Identity Service, ensuring that only authorized users can view and modify patient data, including context-aware restrictions like post-consultation data access for providers.

A critical component of Healthify's security posture is comprehensive audit logging. Every action that touches sensitive data within the Electronic Health Records (EHR) and Identity services will be meticulously logged in a secure, tamper-proof audit trail. This will be achieved by leveraging FHIR `AuditEvent` and `Provenance` resources. The `AuditEvent` resource will capture granular details such as the type and subtype of event (e.g., user login, data access, resource creation/modification/deletion), the action performed (e.g., C, R, U, D for CRUD operations), occurred and recorded timestamps, the outcome (success/failure), the agent (who performed the action, referencing `Practitioner`, `Patient`, `Device`, or `Organization`), the source (where the event originated), and the entity (what FHIR resource or data object was affected). The `Provenance` resource will complement `AuditEvent` by recording detailed metadata about the history and origin of a specific FHIR resource instance, including who made changes, when, and what changes were made. This dual approach ensures both system-level security monitoring and resource-level data integrity tracking, fulfilling strict HIPAA requirements. Regular security audits and penetration testing will be conducted to proactively identify and address vulnerabilities.

5.2 Performance, Reliability, Usability, and Compatibility

- **Performance Requirements:** The system must be capable of handling a large number of simultaneous users without experiencing significant performance degradation. Response times for critical operations, such as appointment scheduling and EHR access, must consistently meet acceptable benchmarks (<500ms for API responses, <3 seconds for page loads). Performance will be optimized through efficient JSONB indexing for frequently queried FHIR elements, continued use of Redis for caching, asynchronous processing for non-real-time operations, and future consideration for database sharding or partitioning as data volume grows.
- **Reliability:** Healthify is designed for high availability, with a target of 99.9% uptime for core services. Minimal downtime will be allocated only for scheduled maintenance and updates. The system will incorporate robust backup and disaster recovery mechanisms to safeguard against data loss and ensure business continuity.
- **Usability:** The user interface will be mobile-responsive, user-friendly, and accessible to users with varying levels of technical expertise. Clear instructions and tooltips will be provided where necessary to guide users through different features and processes, enhancing the overall user experience. Adherence to WCAG 2.1 AA standards for users with disabilities will be a core consideration.
- **Compatibility:** The system will be compatible with major modern web browsers (Chrome, Firefox, Safari, Edge) and will ensure consistent functionality across different mobile devices. Furthermore, the system is designed to support integration with external systems such as pharmacy systems and electronic medical record systems, fostering broader interoperability.

Constraints: The development and operation of the Healthify platform are subject to specific constraints. The healthcare portal must rigorously comply with all relevant healthcare regulations and standards, most notably the Health Insurance Portability and Accountability Act (HIPAA) in the United States. Additionally, the project development must adhere strictly to the allocated budget and established timeline.

6. Database Design and FHIR Compliance

6.1 Current Data Landscape Overview

Healthify's current data landscape is structured as a "Healthify Database Cluster," logically segmented into distinct databases, each serving a specific microservice. These include `Identity_DB`, `Billing_DB`, `EHR_DB`, `Communication_DB`, `Appointment_DB`, `Analytics_DB`, `Content_DB`, and `Evercare_DB`. This provides a modular foundation for data management.

6.2 Proposed FHIR-Compliant Hybrid Data Persistence Model

To achieve full FHIR compliance and maximize interoperability, a strategic evolution of Healthify's database architecture and data module design is essential. The fundamental shift involves treating FHIR resources as the primary data model for all clinical and administrative data where a corresponding FHIR resource exists. This "FHIR-native" approach simplifies data transformation, reduces potential for data loss or misinterpretation, and inherently supports interoperability from the ground up. For elements not directly covered by base FHIR resources, standard FHIR profiling and extensions will be utilized to maintain compliance.

Healthify's existing use of PostgreSQL with JSONB for FHIR resources provides a strong foundation. The proposed strategy refines this into a hybrid model:

- **FHIR Resource Storage (JSONB in PostgreSQL):** Clinical data, which aligns directly with FHIR resources (e.g., `Patient`, `Condition`, `Observation`, `Encounter`, `MedicationRequest`), will be stored as JSONB objects within PostgreSQL. This provides the flexibility to store the full FHIR resource structure, including arrays and nested objects, while still leveraging PostgreSQL's indexing capabilities for efficient querying of specific FHIR elements. AES-256 encryption for sensitive fields will continue to be a critical security measure.
- **Relational Data (PostgreSQL):** Non-FHIR-specific metadata, lookup tables, and data that are purely administrative or internal to Healthify's business logic (e.g., user authentication details, subscription tiers, payment transaction IDs, content categories) will reside in traditional relational tables within PostgreSQL. This optimizes performance for operations that do not require FHIR's semantic richness.
- **File Storage (AWS S3):** Large binary files such as lab reports, imaging scans, and e-prescriptions (PDFs) will be stored in AWS S3, complete with appropriate access controls. Metadata about these files, crucial for their discoverability and context within the clinical record, will be captured in FHIR `DocumentReference` resources, stored in the JSONB FHIR database.
- **Caching (Redis):** Redis will be used for caching frequently accessed, non-persistent data, such as provider availability, to improve response times.

All external-facing APIs will strictly adhere to FHIR RESTful API principles, supporting standard CRUD operations on FHIR resources and leveraging FHIR's standardized search parameters. The API Gateway will be enhanced to validate incoming and outgoing data against FHIR profiles. Internal microservice communication will also prioritize FHIR resource exchange where applicable. HIPAA compliance is paramount, enforced through end-to-end encryption, granular Role-Based Access Control (RBAC), and comprehensive audit logging using FHIR `AuditEvent` and `Provenance` resources.

6.3 Detailed FHIR Resource Mapping for Each Service

The following section details the proposed FHIR resource mappings for each of Healthify's core microservices, highlighting how existing data points will be transformed and enriched to align with FHIR R4 standards.

- **6.3.1 Identity and Access Management (IAM) Service:**

    - **User Management:** The `Patient` FHIR resource will store all patient demographics and contact information. The `Practitioner` FHIR resource will represent doctors and therapists, capturing their personal details, contact information, and qualifications. The `PractitionerRole` FHIR resource is crucial for defining specific roles, linking a `Practitioner` to an `Organization`, specifying location, and defining service types, managing active status and engagement periods. For caregivers, the `RelatedPerson` resource can represent their relationship to a `Patient`. The `Organization` FHIR resource will represent Healthify and affiliated entities.
    - **Roles and Permissions (RBAC):** Internal `role_id` and `role_name` will be managed as attributes, mapped to FHIR-compliant `PractitionerRole` instances and `Patient` profiles. Access control will be enforced at the API Gateway and microservice level.
    - **Authentication and Authorization:** The custom Identity Service's OAuth 2.0 and SMART-on-FHIR implementation aligns with FHIR's security framework, ensuring secure user access.
    - **Audit Logging:** Robust audit logging will utilize the FHIR `AuditEvent` resource, capturing event type, action, timestamps, outcome, agent, source, and affected entity. The `Provenance` resource will complement `AuditEvent` by recording detailed metadata about resource history.
- **6.3.2 Electronic Health Records (EHR) Service:**
    - The EHR Service will consume and manage `Patient` resources, ensuring all clinical data is linked to a valid `Patient.id`.
    - `Condition` FHIR resource will record patient conditions, problems, and diagnoses, including `clinicalStatus`, `verificationStatus`, `category`, `severity`, `code`, `bodySite`, `subject`, `encounter`, `onset`, and `abatement`.
    - `AllergyIntolerance` FHIR resource will capture allergies and intolerances, including `patient`, `clinicalStatus`, `verificationStatus`, `code`, `category`, `criticality`, `reaction` details, and `encounter`.
    - `Observation` FHIR resource is highly versatile for vital signs, lab results, and health parameters (manual or wearable integration), including `subject`, `status`, `category`, `code`, `value`, `effective`, and `performer`.
    - `Procedure` FHIR resource will record actions performed on or for a patient, including `subject`, `status`, `code`, `encounter`, `performed`, `performer`, `location`, `reasonCode`, and `outcome`.
    - `DiagnosticReport` FHIR resource will represent findings from diagnostic tests, including `subject`, `status`, `code`, `encounter`, `issued` date, `performer`, `result` (linking to `Observation`), `media`, and `conclusion`.
    - `DocumentReference` FHIR resource will manage metadata for all uploaded files (e.g., lab reports, prescriptions), capturing `subject`, `type`, `category`,

date, author, custodian, securityLabel, and content details (including S3 URL).

- **6.3.3 Appointment Scheduling Service:**
  - Appointment FHIR resource will manage all bookings, including identifier, status, cancelationReason, serviceCategory, serviceType, appointmentType, reasonCode, priority, description, start and end times, minutesDuration, created timestamp, comment, patientInstruction, basedOn (referencing ServiceRequest), and a detailed participant list.
  - Schedule and Slot FHIR resources will manage provider availability. A Schedule represents a grouping of Slots for a HealthcareService, linking to Practitioner, Location, and Organization. Slot resources represent free time periods within a Schedule.
  - ServiceRequest FHIR resource will represent patient requests for appointments or other services, including subject, status, intent, category, priority, code, occurrence, and requester.
- **6.3.4 Prescription Management Service:**
  - The MedicationRequest FHIR resource will manage all e-prescriptions. This resource covers all types of medication orders and includes identifier, status, intent, priority, medication (the drug), subject (referencing Patient), encounter (referencing Encounter), authoredOn (date of prescription), requester (the prescribing Practitioner), dosageInstruction, and dispenseRequest (details for pharmacy). This resource is designed to generalize across inpatient and outpatient settings, ensuring comprehensive medication management.
- **6.3.5 Communication Service:**
  - The Communication FHIR resource will represent secure, real-time messages and notifications exchanged between patients and providers. It includes identifier, status, category, medium (e.g., email, chat), subject, recipient, sender, sent and received timestamps, reasonCode, and payload (message content, including attachments).
- **6.3.6 Billing and Payment Service:**
  - ChargeItem FHIR resource will describe the provision of healthcare products or services for a patient, including identifier, status, code, subject, context, occurrence, performer, performingOrganization, quantity, priceOverride, and account.
  - Invoice FHIR resource will represent the billing artifact, including identifier, status, type, subject, recipient, date, issuer, lineItem (linking to ChargeItem), totalNet, and totalGross.
- **6.3.7 Content Management Service:**

    - `DocumentReference` can be used for articles, FAQs, and videos as static documents, storing metadata with the actual content in S3. `category` and `type` elements will be profiled for "health education article," "FAQ," or "video."
    - The `Media` FHIR resource (in R4) can represent video or audio content with an S3 link.
    - `Questionnaire` and `QuestionnaireResponse` resources can be used for interactive health education, such as quizzes or risk assessments.
- **6.3.8 Ever Care Program Service:**
    - The `CarePlan` FHIR resource is ideal for managing personalized NCD screenings, scheduled virtual/home visits, and preventive care content. It includes `subject`, `status`, `intent`, `category`, `period`, `author`, `contributor`, `careTeam`, `addresses` (linking to `Condition`), `goal`, and `activity`.
    - NCD Screenings will be modeled using `Questionnaire` for risk assessments and `QuestionnaireResponse` for patient answers, with results recorded as `Observation` resources.
    - Health Parameter Monitoring for manual entry and wearable integration will primarily leverage the `Observation` FHIR resource.
- **6.3.9 Analytics and Reporting Service:**
    - Instead of directly storing raw analytics data in separate tables, the Analytics Service will query and aggregate data from the FHIR-compliant resources across other microservices. This ensures reports on user activity, appointment statistics, and EHR insights are derived directly from the canonical FHIR data. `AuditEvent` resources will be crucial for user activity logs and compliance reporting.

Table 6.3.1: FHIR Resource Mapping per Service

Microservice	**Key Functionality**	**Mapped FHIR Resource(s)**	**Key FHIR Elements**	**Purpose/Description**
Identity & Access Management	User Management, RBAC, Auth	`Patient`, `Practitioner`, `PractitionerRole`, `RelatedPerson`, `Organization`,	`id`, `name`, `address`, `telecom`, `gender`, `birthDate`, `active`, `period`,	Manages all user identities, roles, and access. Ensures secure, compliant authentication and detailed audit

		`AuditEvent`, `Provenance`	`type`, `action`, `agent`, `entity`	trails for sensitive actions.
Electronic Health Records	EHR Storage & Management	`Patient`, `Condition`, `AllergyIntolerance`, `Observation`, `Procedure`, `DiagnosticReport`, `DocumentReference`	`subject`, `status`, `code`, `value`, `effective`, `encounter`, `category`, `content` (`url`)	Stores and manages all patient clinical data in a FHIR-compliant, semantically rich format, including documents and lab results.
Appointment Scheduling	Appointment Booking, Slots	`Appointment`, `Schedule`, `Slot`, `ServiceRequest`	`status`, `start`, `end`, `participant`, `actor`, `identifier`, `code`	Manages all aspects of appointment booking, provider availability, and patient service requests.
Prescription Management	E-Prescription Generation	`MedicationRequest`	`identifier`, `status`, `medication`, `subject`, `requester`, `dosageInstruction`, `authoredOn`	Handles the creation, signing, and delivery of e-prescriptions, ensuring comprehensive medication order management.
Communication	Secure Messaging, Video	`Communication`	`identifier`, `status`, `sender`, `recipient`,	Facilitates secure, real-time text and video communication between patients

			`payload`, `sent`, `received`	and providers, including file sharing.
Billing & Payment	Payments, Invoicing	`ChargeItem`, `Invoice`	`identifier`, `status`, `subject`, `context`, `totalGross`, `lineItem`	Manages all financial transactions, including pay-per-consultation and subscriptions, and generates billing artifacts.
Content Management	Health Education Content	`DocumentReference`, `Media`, `Questionnaire`, `QuestionnaireResponse`	`subject`, `type`, `category`, `content` (`url`), `item`, `answer`	Manages and delivers health education resources, including articles, videos, and interactive quizzes.
Ever Care Program	Preventive Care, Monitoring	`CarePlan`, `Observation`, `Questionnaire`, `QuestionnaireResponse`	`subject`, `status`, `intent`, `activity`, `goal`, `code`, `value`	Manages personalized care plans for NCDs, scheduled visits, and health parameter monitoring.
Analytics & Reporting	Data Insights, Logs	Aggregates from all FHIR resources, `AuditEvent`	`type`, `action`, `agent`, `entity`, `timestamp`	Generates reports and analytics by querying canonical FHIR data, ensuring compliance

logging for all user activities.

6.5 Entity-Relationship Diagrams (Conceptual and Logical)

To effectively visualize the proposed FHIR-compliant database design, two levels of Entity-Relationship Diagrams (ERDs) will be developed:

- **Conceptual ERD:** This high-level diagram will illustrate the primary FHIR resources (e.g., `Patient`, `Practitioner`, `Encounter`, `Appointment`, `CarePlan`, `MedicationRequest`) and their logical relationships. Crucially, it will emphasize the FHIR `Reference` model (`resourceType/id`) rather than traditional relational foreign key constraints. This diagram will visually demonstrate how, for instance, a `Patient` resource logically references `Condition`, `Observation`, and `Encounter` resources, reflecting the interconnected nature of FHIR data without implying direct relational joins.
- **Logical ERD:** More detailed diagrams will be provided for key microservices, such as the EHR Service and Appointment Service. These diagrams will illustrate the hybrid persistence model by showing how JSONB fields within PostgreSQL tables will store the full FHIR resources, and how traditional relational tables will store auxiliary data (e.g., internal IDs, lookup values, subscription tiers that do not have direct FHIR resource equivalents). This will provide a clearer picture of the physical storage structure and how it accommodates the FHIR data model.

Since FHIR employs logical "references" (`resourceType/id`) rather than rigid relational foreign key constraints, traditional ERDs, which are typically focused on depicting explicit foreign key relationships, might not fully capture the nuanced nature of FHIR's data model. To accurately convey the proposed FHIR-compliant design, the ERDs must explicitly visualize these logical references and the JSONB storage of FHIR resources. For example, instead of a direct foreign key line from an `Appointment` table to a `Patient` table, the ERD would show an `Appointment` resource (stored as JSONB) containing a `patient` element that holds a `Reference` to a `Patient` resource's ID. For JSONB storage, the ERD should depict a table with a `jsonb` column dedicated to the FHIR resource, and potentially separate columns for frequently indexed FHIR elements or associated metadata. These specialized ERDs will be crucial for developers to understand the fundamental shift in data modeling philosophy. They will serve as a visual guide for implementing the FHIR-native approach, helping to avoid common pitfalls of attempting to force FHIR's flexible, graph-like data structure into a purely relational mindset.

7. Supabase Integration Strategy

7.1 Rationale for Supabase Adoption

The integration of Supabase into the Healthify architecture is driven by a specific user requirement for its database and authentication capabilities. Supabase is built upon PostgreSQL, which is Healthify's established database choice. This inherent compatibility ensures minimal friction for integrating existing data persistence strategies and simplifies any migration efforts for relational data. As a Backend-as-a-Service (BaaS) platform, Supabase offers several compelling benefits, including out-of-the-box real-time capabilities, robust authentication services, and an integrated GraphQL/REST API layer. These features have the potential to significantly accelerate development for specific microservices or the rapid prototyping of new functionalities.

While Supabase offers substantial benefits for rapid development due to its integrated features, its nature as a comprehensive "monolithic BaaS" presents a nuanced challenge when integrated into Healthify's existing microservices architecture. The core principle of a microservices architecture is to promote autonomy and independent deployment of services. Directly leveraging Supabase's auto-generated APIs for all functionalities might inadvertently bypass the existing API Gateway and the specialized business logic embedded within Healthify's custom microservices, potentially compromising the autonomy and unique capabilities of services like IAM or Billing. Therefore, the integration strategy must carefully balance the agility and development speed offered by Supabase with the overarching need to maintain microservice autonomy and ensure comprehensive FHIR compliance across the entire platform. This implies a strategic, rather than wholesale, adoption of Supabase.

7.2 Supabase for Database and Realtime Features

Supabase is well-positioned to serve as the managed PostgreSQL instance for Healthify, accommodating both the flexible FHIR JSONB storage and traditional relational tables. This strategic choice offloads significant database management overhead, allowing the development team to focus on core application logic. Healthify will leverage Supabase's real-time subscriptions for functionalities that demand instant updates, such as the seamless delivery of chat messages within the Communication Service or live updates to provider availability in the Appointment Scheduling Service. This capability can either complement or augment the existing WebSockets implementation. Each microservice, such as the Communication Service or the Appointment Service, would establish direct connections to the Supabase PostgreSQL instance, ensuring efficient data interaction. Data for FHIR resources will continue to be stored as JSONB objects within this managed PostgreSQL environment.

7.3 Supabase for Authentication and Row Level Security

Supabase Auth offers a robust solution for managing user registration, login, and session management, supporting various authentication methods and integrating directly with PostgreSQL. This provides a solid foundation for user identity management. Crucially, Supabase's Row Level Security (RLS) can be configured directly on PostgreSQL tables. This feature is highly relevant for enforcing granular access control at the database level, which is vital for HIPAA compliance and for implementing strict rules such as restricting provider access to patient data post-consultation. The existing Identity and Access Management (IAM) Service, built with Node.js/Express and Passport.js for OAuth 2.0/SMART-on-FHIR, will integrate with Supabase Auth. In this model, Supabase would handle the underlying user table and core authentication flows, while the IAM service will wrap this functionality, ensuring adherence to SMART-on-FHIR compliance standards and managing the higher-level Role-Based Access Control (RBAC) logic on top of Supabase's RLS.

7.4 Integration with Existing Microservices and FHIR Strategy

The integration of Supabase is designed to complement Healthify's existing microservices architecture and FHIR strategy. Supabase provides a managed PostgreSQL instance, aligning perfectly with Healthify's current database choice. For the authentication layer, Supabase's authentication capabilities can serve as the underlying identity provider, with Healthify's custom Identity Service building upon this foundation to incorporate the SMART-on-FHIR layer and specific RBAC logic. Supabase's PostgreSQL database fully supports JSONB, making it entirely compatible with the proposed FHIR resource storage strategy. The API Gateway will continue to function as the single entry point, routing requests to the appropriate microservices. These microservices will interact with Supabase's PostgreSQL directly or via its client libraries, while the external API will consistently remain FHIR RESTful. Maintaining data consistency between data stored in Supabase (especially authentication and user profiles) and the FHIR `Patient` and `Practitioner` resources managed by the IAM service will be a continuous focus.

8. Low-Level Architecture Details

8.1 Detailed Technology Stack

The Healthify platform leverages a comprehensive and modern technology stack across its various layers and services:

Table 8.1: Comprehensive Technology Stack

Layer/Service	Technology	Primary Role/Purpose
Frontend	React Native, React.js	Building cross-platform user interfaces
	Tailwind CSS	Responsive UI design and styling

Backend (Microservices)	Node.js/Express	Core backend framework for all microservices
Database	PostgreSQL	Primary data store, supporting relational and JSONB for FHIR resources
Database Management/Auth	Supabase	Managed PostgreSQL, Authentication, Row Level Security (RLS), Real-time features
Real-time Communication	WebRTC (Jitsi Meet, LiveKit)	Video consultations
	WebSockets	Real-time chat functionality
	Supabase Realtime	Complementary real-time features for specific use cases
API Gateway	AWS API Gateway or Kong API Gateway	Single entry point, request routing, authentication, rate limiting
Authentication Libraries	Passport.js	OAuth support (integrated with Supabase Auth)
FHIR Libraries	node-fhir-server-core	FHIR compliance for EHR service, custom FHIR server implementation
	HAPI FHIR (potential)	Open-source FHIR server consideration

PDF Generation	pdfkit	Generating e-prescriptions as PDFs
Content Management	Strapi	Headless CMS for health education resources
Reporting	Chart.js	Generating analytics and reports
Payment Gateways	Stripe SDK, PayHere API	Processing online payments and subscriptions
Calendar Sync	iCal	Calendar synchronization for appointments
File Storage	AWS S3	Scalable storage for uploaded files (e.g., lab reports, prescriptions)
Caching	Redis	In-memory data store for frequently accessed data
CI/CD	GitHub Actions, Jenkins	Continuous Integration/Continuous Deployment pipelines
Version Control	Git	Source code management and collaboration
Hosting	AWS, Azure, Vercel	Cloud infrastructure for application deployment

8.2 API Design Principles (FHIR RESTful APIs)

All external-facing APIs within Healthify will strictly adhere to FHIR RESTful API principles. This commitment ensures that the platform supports standard CRUD (Create, Read, Update, Delete) operations on FHIR resources, facilitating seamless data interaction. The APIs will leverage FHIR's standardized search parameters for efficient querying of data, enabling robust and flexible data retrieval. Both the API Gateway and individual microservices will be configured to validate incoming and outgoing data against defined FHIR profiles, ensuring data integrity and compliance with established healthcare data standards. Furthermore, internal microservice communication will prioritize FHIR resource exchange where applicable, fostering a consistent data language across the entire platform. A clear API versioning strategy will be implemented to manage future updates to the FHIR standard and accommodate evolving requirements.

8.3 Key Data Flow Diagrams

Data Flow Diagrams (DFDs) are essential for translating the abstract architectural components and FHIR resource mappings into concrete, actionable sequences of operations. These diagrams visually represent the flow of information through the system for critical user journeys, revealing potential bottlenecks or complex interactions between microservices and FHIR resources that might not be immediately obvious from static architecture diagrams. They are crucial for understanding the operational logic, identifying potential points of failure, optimizing message queues, ensuring data consistency across service boundaries, and validating the design against functional requirements. DFDs serve as a vital bridge between high-level architecture and low-level implementation, particularly for a complex distributed system handling sensitive healthcare data.

Key DFDs to be developed include:

- **User Registration & Login Flow:** This diagram will illustrate the step-by-step process of a new user registering, detailing how credentials are managed by the Identity and Access Management (IAM) Service (and Supabase Auth), and how subsequent login requests are securely handled via the API Gateway.
- **Appointment Booking Flow:** This DFD will detail the complete sequence from a patient initiating an appointment request, through the Appointment Scheduling Service, the provider's confirmation process, and the delivery of automated notifications.
- **E-Prescription Workflow:** This diagram will trace the flow from a healthcare provider issuing a prescription via the Prescription Management Service, through its secure storage within the EHR Service (leveraging `DocumentReference`), to the patient's secure access of the prescription.
- **Video Consultation Flow:** This DFD will outline the setup and execution of a video call, detailing the interactions within the Communication Service and the integration of WebRTC technology.
- **Payment Processing Flow:** This diagram will trace a payment from its initiation by the patient, through the Billing and Payment Service, its integration with external payment gateways like Stripe or PayHere, and the subsequent generation of an invoice.

9. User Experience (UX) and User Interface (UI) Specifications

9.1 Responsive Web Application Design Principles

The Healthify web application will adhere to robust responsive design principles to ensure an optimal user experience across a diverse range of devices and screen sizes. A **mobile-first approach** will guide the design process, prioritizing the experience for smaller screens and then scaling up to tablets and desktops. The implementation will utilize **flexible grids and layouts**, leveraging frameworks like Tailwind CSS to dynamically adapt content and components. **Fluid images and media** handling will be implemented to prevent overflow and ensure fast loading times. Specific **breakpoints** will be defined for different device categories (e.g., mobile, tablet, small desktop, large desktop) to optimize layouts and component visibility for each context. **Accessibility (A11y)** will be a core consideration, ensuring the UI is usable by individuals with varying levels of technical expertise and potential disabilities, including provisions for keyboard navigation, proper ARIA attributes, and sufficient color contrast. Finally, the UI will be optimized for **performance**, minimizing asset sizes and leveraging caching mechanisms to ensure fast loading times and smooth, responsive interactions.

9.2 UI Component Library and Design System

To ensure consistency, efficiency, and scalability in UI development, Healthify will establish a comprehensive UI component library and a robust design system. This will involve the development of **standardized components** (e.g., buttons, forms, navigation elements, cards, modals) that are reusable across all portals (Patient, Provider, Admin). This standardization will accelerate development and ensure a cohesive look and feel. A comprehensive **design system** will be established, encompassing guidelines for typography, color palette, iconography, spacing, and interaction patterns. This system will maintain brand identity and ensure user experience coherence across the entire platform. Furthermore, consideration will be given to **theming** options, such as light/dark modes, and potential for custom branding to support future partnerships.

9.3 Wireframes (Key Screens for Each Portal)

Wireframes are a critical component for translating abstract functional requirements into concrete visual layouts. They force a detailed consideration of user interaction paths and information hierarchy, which is essential for ensuring both usability and mobile responsiveness. Without wireframes, the development team lacks a visual blueprint for how these functional requirements will manifest in the user interface. Wireframes act as low-fidelity visual representations of the UI, depicting layout, content arrangement, and basic interaction. They directly inform how users will accomplish tasks (user flows) and how the application will adapt to different screen sizes (responsiveness). They are essential for validating the design against functional and usability requirements *before* high-fidelity design or coding commences. Including detailed wireframes in the specification reduces ambiguity, minimizes rework during development, and ensures that the final product aligns with user expectations and business goals. They represent a crucial step in translating the "what" (functional requirements) into the "how" (visual design and interaction).

Here are the detailed textual descriptions of key wireframes for each portal, designed with responsiveness in mind:

9.3.1 Patient Portal Wireframes

1. Login/Registration Screen:

- **Layout:** Centered layout. On mobile, it will be a single column. On tablet/desktop, it can be a two-column layout with a brand image/illustration on one side and the form on the other.
- **Components:**
    - **Header:** Healthify Logo prominently displayed.
    - **Login Form:** Email/Phone input field, Password input field (with toggle visibility icon), "Forgot Password?" link.
    - **Call-to-Action:** "Login" button (primary).
    - **Social Login:** Buttons for "Continue with Google," "Continue with Facebook."
    - **Registration Link:** "Don't have an account? Register Now."
    - **Responsiveness:** Input fields will be `w-full` on mobile, scaling to `w-3/4` or `w-1/2` on larger screens. Buttons will use `py-3 px-6` for good tap targets.

2. Patient Dashboard Screen:

- **Layout:**
    - **Mobile:** Top navigation (hamburger menu, notifications, profile icon), followed by a vertical stack of content widgets. A bottom navigation bar will provide quick access to key sections (Home, Appointments, EHR, Messages).
    - **Tablet/Desktop:** Left-hand sidebar navigation, top header (logo, search, notifications, profile), and a main content area with a grid-based layout for various widgets.
- **Components:**
    - **Navigation:** Top bar with logo, search icon, notification bell, user profile avatar. Left sidebar/Bottom navigation with icons and labels for Home, Appointments, My EHR, Messages, Health Tips, Subscription.
    - **Upcoming Appointments Widget:** Card displaying next appointment details (Provider, Time, Type). "View All Appointments" button.
    - **Health Summary Widget:** Snapshot of key health metrics (e.g., latest BP reading, weight, medication adherence). "View Full EHR" button.
    - **Messages Widget:** Recent unread messages. "View All Messages" button.
    - **Health Tips/Articles Widget:** Carousel or list of trending health articles. "Explore All Health Resources" button.

    - **Subscription Status Widget (Subscribed Members):** Displays current plan, remaining free consultations, and renewal date. "Manage Subscription" button.
    - **Action Buttons:** Prominent "Book New Appointment" button.
    - **Responsiveness:** Widgets will stack vertically on mobile, transition to 2-column then 3-column grids on tablet/desktop. Navigation will switch between bottom bar (mobile) and sidebar (desktop).

3. Appointment Booking Screen (Multi-Step Wizard):

- **Layout:** A guided wizard interface, ensuring clear progress indication. Each step occupies the full width on mobile, with a wider, centered container on tablet/desktop.
- **Components:**
    - **Progress Indicator:** Stepper (e.g., dots or numbered steps) at the top: "1. Select Service > 2. Choose Provider > 3. Select Time > 4. Confirm & Pay."
    - **Step 1: Select Service:** Radio buttons or cards for "Video Consultation," "Chat Consultation," "Home Visit." "Next" button.
    - **Step 2: Choose Provider:** Search bar for providers, filter options (specialty, language), list/grid of provider cards (Name, Specialty, Rating, Availability snippet). "View Profile" button on each card.
    - **Step 3: Select Time:** Calendar component (interactive, showing available slots highlighted). Time slot selection list below calendar. "Back" and "Next" buttons.
    - **Step 4: Confirm & Pay:** Summary of appointment details (service, provider, date, time, cost). Payment method selection (Credit Card, Mobile Wallet). Input fields for payment details. "Pay & Confirm" button (Non-subscribers) or "Confirm Appointment" (Subscribers with "Pay Later").
    - **Responsiveness:** Form elements stretch full width on mobile. Calendar component will adapt its size, perhaps showing fewer days on mobile or requiring horizontal scroll.

4. My EHR Screen:

- **Layout:** Tabbed interface.
    - **Mobile:** Tabs at the top, content below. Could also use an accordion for sections.
    - **Tablet/Desktop:** Left-hand navigation (tabs) with content on the right.
- **Components:**
    - **Tabs:** Demographics, Conditions, Medications, Lab Results, Procedures, Allergies, Documents, Vitals.
    - **Content Area:**
        - Each tab will display data in a clear, readable format (e.g., lists, tables).
        - **Conditions:** List of diagnosed conditions with date of diagnosis, status. "Add New Condition" button.
        - **Medications:** List of current and past medications (Name, Dosage, Frequency, Start/End Date). "Request Refill" button next to each active prescription.

        - **Lab Results:** List of lab tests, with date, test name, and "View Report" button (to open PDF/image from S3).
        - **Vitals:** Chart displaying trends for BP, heart rate, weight, etc. "Add New Vital" button.
        - **Documents:** List of uploaded documents (`DocumentReference`) with date, type, and "View/Download" button.
    - **Responsiveness:** Tables will convert to stacked cards or horizontal scrolling on mobile. Charts will be responsive to container size.

5. Secure Messaging Interface:

- **Layout:**
    - **Mobile:** Single pane, showing either chat list or active chat. Transitions between views.
    - **Tablet/Desktop:** Two-pane layout: left panel for chat list (recent conversations), right panel for active chat window.
- **Components:**
    - **Chat List (Left Panel / Mobile List View):** List of contacts/conversations (Provider Name, Last Message Snippet, Unread Count, Last Active Time). Search bar.
    - **Active Chat Window (Right Panel / Mobile Detailed View):**
        - **Header:** Contact name/avatar, "Video Call" icon (WebRTC), "Phone Call" icon (if applicable), "i" (info) icon.
        - **Message Display Area:** Scrollable area showing messages, aligned left for recipient, right for user. Support for text, emojis, image/file previews. Timestamps for message groups.
        - **Input Area:** Text input field, "Attach File" icon (paperclip), "Send" button.
    - **Responsiveness:** Chat list collapses or becomes a separate screen on mobile. Message display area adjusts text wrapping and image sizes.

9.3.2 Provider Portal Wireframes

1. Login Screen:

- **Layout:** Identical to Patient Login but with "Provider Login" title.
- **Components:** Email/Phone, Password, "Login" button, "Forgot Password?" link.
- **Responsiveness:** Same as Patient Login.

2. Provider Dashboard Screen:

- **Layout:**
    - **Mobile:** Top navigation (hamburger, notifications, profile), vertically stacked widgets. Bottom navigation for quick access (Home, Appointments, Patients, Messages).

    - **Tablet/Desktop:** Left-hand sidebar navigation, top header (logo, search, notifications, profile), main content area with a dashboard view.
- **Components:**
    - **Navigation:** Similar to Patient, but with Provider-specific links (e.g., My Schedule, Patient List, Prescriptions, Analytics).
    - **Upcoming Appointments Widget:** List of next few appointments (Patient Name, Time, Type). "Start Consult" button (active 5 mins before).
    - **Pending Tasks Widget:** New messages, refill requests, home visit requests pending review.
    - **Availability Status Toggle:** Quick switch to set availability (e.g., "Available," "On Break," "Offline").
    - **Quick Actions:** "View Patient List," "Issue Prescription."
    - **Analytics Summary Widget:** Snapshot of consultations completed, earnings.
    - **Responsiveness:** Similar to Patient Dashboard, adapting widget layout.

3. Appointment Management Screen:

- **Layout:** Calendar view with integrated list of appointments.
    - **Mobile:** Month/Week view selector, daily appointment list scrollable below.
    - **Tablet/Desktop:** Larger calendar grid (month/week view) on one side, detailed daily list on the other.
- **Components:**
    - **Calendar:** Displays days with appointments highlighted. Allows navigation between days/weeks/months.
    - **Appointment List:** For selected day: (Patient Name, Time, Type, Status).
    - **Action Buttons per Appointment:** "Accept," "Reschedule," "Cancel," "Start Consultation."
    - **Availability Editor:** Button/panel to set/edit time slots for specific days. Form with start/end times, breaks, max patients.
    - **Notifications:** Real-time toast notifications for new appointment requests or changes.
    - **Responsiveness:** Calendar adapts, list items might become cards on mobile.

4. Patient Consultation Screen:

- **Layout:** Optimized for concurrent video and data access.
    - **Mobile:** Video call dominates, with collapsible/swipeable overlay for EHR/notes.
    - **Tablet/Desktop:** Split-screen layout. Left: Video call window (WebRTC). Right: Tabbed interface for patient EHR and consultation notes.
- **Components:**
    - **Video Call Area:** Self-view, patient view. Mute, video on/off, end call buttons. Connection status indicator.
    - **Patient EHR/Notes Area:**
        - **Tabs:** Patient Summary, Conditions, Medications, Lab Results, Notes.
        - **Patient Summary:** Key demographics, current medications, allergies.

        - **Notes:** Free-text area for capturing consultation notes. Auto-save functionality.
        - **Lab Results:** Viewable in-app.
        - **Prescription Button:** Prominent "Issue Prescription" button, leading to a dedicated prescription form.
    - **Responsiveness:** Dynamic resizing of video and data panels. Collapsible elements for mobile to maximize screen space.

5. Prescription Management Screen (for Doctors):

- **Layout:** Form-based for prescription creation. Full width on mobile, centered/wider on desktop.
- **Components:**
    - **Patient Selector:** Dropdown/search for assigned patients (pre-filled if accessed from consultation).
    - **Medication Search & Add:** Search bar for medications (with auto-suggest). "Add Medication" button.
    - **Medication List (Dynamically Added):** Each added medication: Drug Name, Dosage Input, Frequency Input (dropdowns), Route Input, Duration Input, Quantity, Instructions (text area).
    - **Digital Signature Pad/Upload:** Area for provider to add/confirm digital signature.
    - **Preview Button:** To show PDF preview.
    - **Call-to-Action:** "Issue Prescription" button (generates PDF, saves `MedicationRequest`).
    - **Responsiveness:** Input fields stack vertically on mobile. Medication list items are well-spaced and readable.

9.3.3 Admin Portal Wireframes

1. Login Screen:

- **Layout:** Identical to other logins but clearly labeled "Admin Login."
- **Components:** Email/Phone, Password, "Login" button, "Forgot Password?" link.
- **Responsiveness:** Same as other login screens.

2. Admin Dashboard Screen:

- **Layout:**
    - **Mobile:** Top navigation, vertically stacked summary cards.
    - **Tablet/Desktop:** Left-hand sidebar navigation, top header, main content area with overview cards and quick links.
- **Components:**
    - **Navigation:** Sidebar with links for Dashboard, User Management, Content Management, Billing, Reports, Audit Logs, Settings.

    - **Summary Cards:** Total Users, Active Providers, Total Appointments, Revenue Summary, Pending Approvals (e.g., new providers).
    - **Recent Activity Feed:** Latest registrations, recent consultations, payment alerts.
    - **Quick Links:** "Create New Provider Account," "Manage Content," "View All Payments."
    - **System Health Indicators:** Basic metrics like API uptime, database status.
    - **Responsiveness:** Cards will reflow and stack based on screen size.

3. User Management Screen:

- **Layout:** Tabbed interface or filters for user types, with a searchable, sortable table.
- **Components:**
    - **User Type Filters:** Tabs/buttons for "All Users," "Patients," "Providers," "Admins."
    - **Search and Filter Bar:** For user name, email, ID, status.
    - **User Table:** Columns for Name, Email, Role, Status (Active/Inactive), Last Login, Actions.
    - **Action Buttons per User:** "View Profile," "Edit User," "Deactivate/Activate," "Reset Password."
    - **"Create New Provider" Button:** Leads to a new form.
    - **Responsiveness:** Table will likely require horizontal scrolling or transform into a list of cards on mobile.

4. Content Management Screen:

- **Layout:** Two-pane or single-pane based on screen size.
- **Components:**
    - **Content List (Left Panel/Main View):** Search and filter options. List of articles/FAQs/videos (Title, Category, Author, Last Updated, Status (Draft/Published)).
    - **Action Buttons per Content Item:** "Edit," "Preview," "Publish/Unpublish," "Delete."
    - **"Add New Content" Button:** Leads to a rich text editor/form.
    - **Content Editor (Right Panel/Modal/Separate Screen):**
        - **Form Fields:** Title, Category (dropdown), Content (Rich Text Editor), Media Upload (for images/videos), Status (Draft/Published), SEO fields.
        - **Save/Publish/Cancel Buttons.**
    - **Responsiveness:** Content list might become a vertical list of cards on mobile. Editor will be full-screen on mobile.

9.4 User Flows (Detailed Critical Paths)

User flows are vital for validating the end-to-end user journeys across the complex microservices architecture. They highlight how different services interact to support a single user goal, ensuring a seamless and intuitive experience, especially for critical paths involving sensitive data and compliance. Without defined user flows, individual microservices might be developed in isolation, potentially leading to disjointed user experiences or missed integration points. User flows map the sequential steps a user takes to complete a specific task, illustrating decision points, system responses, and transitions between screens or services. For Healthify, this is particularly critical for ensuring that complex processes, such as a "provider losing access to patient history post-consultation", are handled correctly and compliantly across the IAM and EHR services. User flows serve as a blueprint for both UI/UX design and backend development, ensuring that the system effectively supports the intended user journeys efficiently and compliantly. They are essential for identifying edge cases, potential errors, and opportunities for optimization within the user experience.

Here are the detailed steps for critical user flows:

9.4.1 User Flow: New Patient Onboarding & First Appointment (Non-Subscribed Member)

Objective: A new user registers for Healthify and successfully books their first pay-per-consultation appointment.

1. **User (Patient) Accesses Healthify Web Application.**
   - *System:* Displays Login/Registration screen.
2. **User Clicks "Register Now."**
   - *System:* Navigates to the Registration Form.
3. **User Enters Email, Password, Basic Profile Information (Name, Date of Birth, Gender), and Accepts Terms & Conditions.**
   - *System:* Validates input. Sends registration data to Identity Service.
   - *Identity Service:* Creates `users` record in PostgreSQL (Supabase Auth). Creates `Patient` FHIR resource (JSONB) and stores `Patient.id` in `users` table.
4. **User Clicks "Register."**
   - *System:* Displays "Registration Successful! Please Verify Your Email" message. Sends verification email.
5. **User Verifies Email (Clicks Link in Email).**
   - *System:* Identity Service confirms email, activates user account. User is redirected to Login screen.
6. **User Logs In with New Credentials.**
   - *System:* Identity Service authenticates user. Generates JWT. Redirects to Patient Dashboard.
7. **User (on Dashboard) Clicks "Book New Appointment."**
   - *System:* Navigates to "Select Service" step of Appointment Booking wizard.
8. **User Selects "Video Consultation" (or other service).**
   - *System:* Updates wizard progress. Displays "Choose Provider" step.
9. **User Searches/Filters for a Provider (e.g., "Dermatologist").**
   - *System:* Queries Appointment Service for available providers based on criteria. Displays list of provider cards.
10. **User Selects a Provider and Clicks "View Profile" (Optional).**
    - *System:* Displays Provider's detailed profile.
11. **User Clicks "Select Provider" (or equivalent on profile/card).**

    - *System:* Updates wizard progress. Displays "Select Time" step with calendar of provider's available slots (`Schedule`, `Slot` FHIR resources).

12. **User Selects a Date and Time Slot.**
    - *System:* Temporarily reserves slot. Displays "Confirm & Pay" step with appointment summary and cost.
13. **User Reviews Summary and Selects "Credit Card" Payment Method.**
    - *System:* Displays credit card input form.
14. **User Enters Credit Card Details and Clicks "Pay & Confirm."**
    - *System:* Billing Service initiates payment with Stripe/PayHere.
    - *Billing Service:* Records pending payment transaction.
15. **Payment Gateway Processes Payment.**
    - *System:* Receives payment confirmation/failure from gateway.
    - *Billing Service:* Updates payment status.
    - *Appointment Service:* If successful, confirms appointment, updates `Appointment` FHIR resource status. Sends confirmation to patient and provider.
16. **User Receives "Appointment Confirmed" Notification (In-app, Email, SMS).**
    - *System:* Displays confirmation screen. User can return to Dashboard.
17. **(At Appointment Time) User Joins Video Consultation.**
    - *System:* Communication Service initiates WebRTC session.

9.4.2 User Flow: Provider Consultation & E-Prescription

Objective: A doctor conducts a video consultation and issues an e-prescription to the patient.

1. **Provider Logs In to Provider Portal.**
    - *System:* Identity Service authenticates provider. Redirects to Provider Dashboard.
2. **Provider (on Dashboard) Sees "Upcoming Appointments" Widget.**
    - *System:* Displays list of scheduled appointments for the day, including patient names.
3. **5 Minutes Before Appointment, "Start Consult" Button Becomes Active for the Specific Appointment.**
    - *System:* Appointment Service updates appointment status for provider's view.
4. **Provider Clicks "Start Consult."**
    - *System:* Navigates to Patient Consultation screen. Communication Service initiates WebRTC session. EHR Service prepares to provide patient's historical EHR data *for the duration of the consult only*.
5. **Provider and Patient Connect via Video Call.**
    - *System:* WebRTC connection established. Communication Service manages call state.
6. **Provider Accesses Patient's EHR (via Tabs/Panels on Consult Screen).**
    - *System:* EHR Service provides real-time access to `Patient`, `Condition`, `Observation`, `MedicationStatement`, `DocumentReference` etc. via secure API calls.

7. **Provider Reviews Patient History and Takes Consultation Notes.**
   - *System:* Notes are saved automatically/periodically in a temporary or draft format, linked to the `Encounter` resource.
8. **Provider Determines a Prescription is Needed and Clicks "Issue Prescription."**
   - *System:* Navigates to Prescription Management form, pre-populated with patient and encounter ID.
9. **Provider Searches for Medication, Adds to List, Specifies Dosage, Frequency, Instructions, etc.**
   - *System:* Displays medication details. Validation on input.
10. **Provider Adds Digital Signature (or confirms pre-configured signature).**
    - *System:* Validates signature.
11. **Provider Clicks "Issue Prescription."**
    - *System:* Prescription Management Service creates a `MedicationRequest` FHIR resource. Generates a PDF of the prescription. EHR Service stores the PDF as a `DocumentReference` in AWS S3, linked to the `Patient` and `Encounter`.
    - *Prescription Management Service:* Sends confirmation to patient. Records prescription issuance.
12. **Provider Returns to Consultation Screen and Clicks "End Call."**
    - *System:* Communication Service terminates WebRTC session.
13. **(Post-Consultation) System Processes Access Restriction.**
    - *System:* IAM Service (via RLS in Supabase/PostgreSQL) revokes the provider's *direct read access* to the patient's full historical EHR and the newly issued prescription (except through the "issue prescription access point" for their assigned patients). An `AuditEvent` is recorded detailing the end of the consult and change in access context.

9.4.3 User Flow: Subscribed Member Ever Care Journey

Objective: A subscribed member enrolls in an NCD screening, monitors health parameters, and has a follow-up virtual visit.

1. **User (Subscribed Patient) Logs In to Patient Portal.**
   - *System:* Redirects to Dashboard.
2. **User Navigates to "Ever Care Program" Section.**
   - *System:* Displays available NCD screening programs (e.g., Diabetes Risk Assessment).
3. **User Selects a Screening Program (e.g., "Diabetes Risk Assessment") and Clicks "Start."**
   - *System:* Content Management Service loads `Questionnaire` FHIR resource. Displays the assessment questions.
4. **User Completes the Questionnaire.**

- *System:* User's answers are captured as a `QuestionnaireResponse` FHIR resource. Ever Care Program Service processes responses, calculates risk score.

5. **Ever Care Program Service Analyzes Score and Suggests a `CarePlan`.**
   - *System:* Displays initial `CarePlan` recommendations (e.g., "High Risk, recommended monthly virtual consults and daily BP monitoring"). Creates `CarePlan` FHIR resource.
6. **User Accepts the Recommended `CarePlan`.**
   - *System:* `CarePlan` status updated to "active." Automated tasks (`ServiceRequest`) for virtual consults and monitoring are added to the plan.
7. **User (Daily/Weekly) Logs Health Parameters (e.g., Blood Pressure, Glucose) Manually.**
   - *System:* User inputs data on "My EHR" screen. EHR Service creates `Observation` FHIR resources.
8. **Ever Care Program Service Monitors `Observation` Data against `CarePlan` Goals.**
   - *System:* If anomalies detected or follow-up due, flags for review.
9. **Automated System Schedules a Virtual Follow-up Consult (per `CarePlan` activity).**
   - *System:* Appointment Service schedules an `Appointment` based on `ServiceRequest` from `CarePlan`. Patient and Provider receive notifications.
10. **(At Consult Time) Patient Joins Virtual Consult.**
    - *System:* Communication Service initiates WebRTC. Provider accesses patient's `CarePlan` and `Observation` data during consult.
11. **Provider Reviews Progress, Updates `CarePlan` (e.g., adjusts goals, adds new activities).**
    - *System:* EHR Service updates `Observation` and `CarePlan` FHIR resources. An `AuditEvent` is logged for the consultation.
12. **Patient Continues Monitoring and Receiving Tailored Health Content.**
    - *System:* Content Management Service provides relevant `DocumentReference` and `Media` resources based on `CarePlan`.

9.4.4 User Flow: Admin User Management & Content Update

Objective: An administrator creates a new provider account and then publishes a new health article.

1. **Admin Logs In to Admin Portal.**
   - *System:* Identity Service authenticates admin. Redirects to Admin Dashboard.
2. **Admin (on Dashboard) Clicks "User Management" in Navigation.**
   - *System:* Navigates to User Management screen, displaying a table of all users.
3. **Admin Clicks "Create New Provider" Button.**
   - *System:* Displays "Create New Provider Account" form.

4. **Admin Enters Provider Details (Name, Email, Specialization, Contact, etc.) and Sets Initial Password.**
   - *System:* Validates input. Sends data to Identity Service.
   - *Identity Service:* Creates `users` record in PostgreSQL (Supabase Auth). Creates `Practitioner` and `PractitionerRole` FHIR resources (JSONB). Generates initial login credentials.
5. **Admin Clicks "Create Provider Account."**
   - *System:* Identity Service processes creation. Sends credentials to provider (e.g., via email). Displays "Provider Account Created" confirmation.
6. **Admin Navigates to "Content Management" in Navigation.**
   - *System:* Displays Content Management screen, showing a list of existing articles.
7. **Admin Clicks "Add New Content" Button.**
   - *System:* Displays Rich Text Editor/Content Creation form.
8. **Admin Enters Article Title, Selects Category, Adds Content (text, images), and Sets Status to "Draft."**
   - *System:* Content Management Service saves article content and metadata as a `DocumentReference` FHIR resource (JSONB), potentially storing large media in S3.
9. **Admin Clicks "Save Draft."**
   - *System:* Content Management Service saves changes. Article appears in "Draft" status in the content list.
10. **Admin Reviews Draft Article and Clicks "Edit" Again.**
    - *System:* Opens the article in the editor.
11. **Admin Sets Status to "Published" and Clicks "Save & Publish."**
    - *System:* Content Management Service updates the `DocumentReference` FHIR resource status to "current" or "published." Makes the content available to Patient Portal.
    - *Analytics Service:* Potentially logs an `AuditEvent` for content publication.
12. **(Optional) Admin Navigates to Patient Portal to Verify Published Article.**
    - *System:* Patient Portal displays the newly published article under "Information Resources."

10. Identification and Integration of Missed Components

During the analysis of the user query and provided materials, several components were identified as either missing from the initial specifications or requiring further integration and elaboration to ensure a comprehensive and compliant project specification.

10.1 Comprehensive Audit Logging (FHIR AuditEvent, Provenance)

Gap Identified: While "Audit logging for HIPAA compliance" is explicitly noted, the initial `Analytics_DB` metrics table lacks the comprehensive, standardized structure necessary for robust audit trails, particularly when compared to FHIR `AuditEvent` and `Provenance` resources. The existing structure is insufficient for the granular, standardized audit records required by HIPAA.

Solution: Healthify will implement robust audit logging by leveraging FHIR `AuditEvent` and `Provenance` resources. Every action that touches sensitive data within the EHR and Identity services will generate an `AuditEvent`. This resource will capture the event type, action performed, the agent (who performed the action), the source of the event, the affected entity (the FHIR resource or data object), and precise timestamps for when the event occurred and was recorded, along with its outcome. The `Provenance` resource will complement `AuditEvent` by recording detailed metadata about the history and origin of a specific FHIR resource instance, including who made changes, when, and what specific changes were made. This ensures accountability, facilitates compliance audits, and provides a tamper-proof record, significantly reducing legal and operational risks associated with handling Protected Health Information (PHI).

10.2 FHIR Profiling and Extensions Strategy

Gap Identified: The FHIR standard is vast, and achieving a "fully" compliant server is acknowledged as a significant undertaking. Healthify needs to clearly define which specific FHIR resources, profiles, and search capabilities it will support to manage this complexity effectively. Without a clear profiling strategy, custom data elements might be stored in a non-standard way, hindering interoperability and future upgrades.

Solution: Healthify will develop a clear strategy for FHIR profiling and extensions. This involves:

- **Defining Core Profiles:** Creating custom FHIR profiles for specific Healthify use cases that are not fully covered by base FHIR resources. Examples include specific Non-Communicable Disease (NCD) screening data elements or unique subscription details.
- **Judicious Use of Extensions:** For data elements that do not fit into standard FHIR resource fields, FHIR extensions will be used judiciously. These extensions will be formally defined and documented to ensure they are understood by other systems that might interact with Healthify's data. Particular attention will be paid to modifier extensions, which can alter the meaning of a resource.
- **Leveraging Existing Profiles:** Where applicable, Healthify will consider adopting existing, widely used profiles, such as the US Core Profile, as a starting point. This approach can reduce development effort and enhance broader interoperability with established healthcare systems.

This strategy ensures that Healthify's FHIR implementation is both compliant with the standard and practical for meeting specific business requirements without sacrificing the benefits of standardization. It also provides clear documentation for future integrations.

10.3 Data Migration and Transformation Plan

Gap Identified: The current database design is based on a traditional relational model, while the proposed architecture shifts towards a FHIR-compliant JSONB structure within PostgreSQL. This fundamental change necessitates a significant and carefully planned data migration effort from existing data to the new structure without loss or corruption.

Solution: A meticulous, phased data migration strategy will be implemented. This process will include:

- **Data Mapping:** Detailed mapping documents will be created, outlining precisely how each field in the existing relational tables translates to specific FHIR resource elements. This will include careful consideration for `CodeableConcept` and `Reference` types, which are fundamental to FHIR's semantic richness.
- **Transformation Scripts:** Robust Extract, Transform, Load (ETL) scripts will be developed to automate the conversion of existing data into the new FHIR resource format. These scripts will handle data type conversions, cardinality requirements, and the creation of new FHIR resources (e.g., separating generic user entries into distinct `Patient` and `Practitioner` resources).
- **Validation:** Thorough validation of the transformed data against FHIR schemas and Healthify's newly defined profiles will be conducted to ensure accuracy and compliance.
- **Phased Migration:** A phased approach to data migration will be adopted, potentially starting with non-critical data or new user onboarding, to minimize disruption to existing services and users.

This comprehensive plan minimizes business disruption, ensures the accuracy and integrity of historical data, and provides a clear roadmap for transitioning to the new FHIR-compliant system.

10.4 FHIR Server Selection Strategy

Gap Identified: The Healthify team has considered various options for FHIR servers, including managed services like Google Cloud Healthcare API and Azure API for FHIR, as well as open-source alternatives such as HAPI FHIR. A definitive strategic decision is required regarding the optimal FHIR server solution. The choice of FHIR server significantly impacts development effort, operational costs, and the speed of achieving compliance and interoperability.

Solution: A thorough evaluation will be conducted to weigh the trade-offs between managed FHIR services and open-source FHIR servers:

- **Managed FHIR Services (e.g., Google Cloud Healthcare API, Azure API for FHIR):** These services offer out-of-the-box FHIR APIs, inherent scalability, robust security, and compliance features. They significantly reduce development and operational overhead by handling the complexities of FHIR resource persistence, versioning, and search. This option is generally recommended for platforms aiming for faster time-to-market and reduced maintenance burden.

- **Open Source FHIR Servers (e.g., HAPI FHIR, node-fhir-server-core):** While offering greater control and customization, open-source solutions demand significant internal expertise for deployment, ongoing maintenance, scaling, and ensuring continuous compliance. Given that building a "fully" compliant server is a substantial effort, this path requires a considerable and sustained commitment of resources for development, security patching, and performance tuning to match the capabilities of managed services.

The final decision will carefully weigh these factors against Healthify's specific business objectives, resource availability, and desired speed of deployment. For rapid deployment and robust compliance, managed services often provide a more efficient path to FHIR compliance and interoperability.

10.5 Other Identified Gaps and Solutions

Beyond the major components discussed above, several other elements were identified as requiring explicit inclusion or further detail:

- **Wireframes and User Flows:** These were explicitly requested in the user query but were not provided in the research materials. They will be conceptualized and specified in detail within Section 9.3 (Wireframes) and Section 9.4 (User Flows) of this document.
- **Detailed Entity-Relationship Diagrams (ERDs):** Also explicitly requested in the user query but absent from the research. These will be conceptualized and specified within Section 6.5 (Entity-Relationship Diagrams) to visually represent the FHIR-compliant data model.
- **Prescription Module Schema:** As noted in the current database review, the specific table schema for the Prescription Management Service was missing. The solution involves fully defining its schema based on the `MedicationRequest` FHIR resource, as detailed in Section 6.4.4.
- **Business Model and Monetization Strategy:** The Product Requirements Document (PRD) outlines Healthify's revenue streams, including a Pay-Per-Consultation Model for non-members and Tier-based Subscription Plans (Vital Starter, Boost, Pro) for members. This crucial information will be referenced within the relevant sections, particularly concerning the Billing and Payment Service.
- **Continuous Compliance and Interoperability:** Recognizing that FHIR is an evolving standard, Healthify will implement strategies for continuous compliance. This includes planning for future FHIR version upgrades (e.g., from R4 to R5), actively monitoring new FHIR updates and implementation guides, engaging in interoperability testing with other FHIR-compliant systems, and participating in the broader FHIR community to leverage shared knowledge and contribute to the standard's evolution.
- **Security Audits:** Regular security audits and penetration testing will be conducted to proactively identify and address vulnerabilities within the platform.

The following table summarizes all identified gaps and their proposed solutions:

Table 10.1: Identified Gaps and Proposed Solutions

Gap/Discrepancy	Impact	Proposed Solution	Section in Document
Insufficient Audit Logging Schema	Hinders HIPAA compliance, accountability, and external audits	Implement FHIR `AuditEvent` and `Provenance` resources	10.1, 5.1, 6.4.1, 6.4.9
Lack of Defined FHIR Profiling Strategy	Limits interoperability, increases integration complexity	Define custom FHIR profiles, judicious use of extensions, leverage existing profiles	10.2, 6.3
Need for Data Migration Plan (Relational to FHIR)	Risk of data loss, service disruption during transition	Detailed data mapping, ETL scripts, validation, phased migration	10.3
Undecided FHIR Server Strategy	Impacts development effort, operational cost, time-to-market	Evaluate managed vs. open-source FHIR services based on project goals	10.4
Missing Wireframes	Ambiguity in UI/UX, potential rework, usability issues	Conceptualize and specify key screen layouts for each portal	9.3
Missing User Flows	Disjointed user journeys, missed integration points	Detail critical end-to-end user paths across services	9.4

Missing Prescription Module Database Schema	Incomplete data model, potential friction with external systems	Define schema based on `MedicationRequest` FHIR resource	6.4.4, 3.4
Need for Continuous Compliance & Interoperability	Risk of obsolescence, integration challenges with evolving standards	Plan for FHIR version upgrades, monitor updates, interoperability testing	10.5
Need for Regular Security Audits	Potential vulnerabilities, compliance risks	Conduct regular security audits and penetration testing	10.5

11. Conclusion and Recommendations

The Healthify platform's ambition to provide comprehensive digital health services, particularly within the context of Sri Lanka's evolving healthcare infrastructure, is significantly bolstered by its foundational commitment to FHIR standards. The existing microservices architecture provides a suitable framework for FHIR adoption, as FHIR's resource-centric model aligns naturally with distributed service design. However, the current database schemas, largely rooted in a traditional relational approach, exhibit notable discrepancies in granularity, semantic richness, and entity modeling when compared to FHIR's detailed specifications. This mismatch necessitates a deliberate and structured transition to a FHIR-native data model.

The proposed hybrid data persistence strategy, leveraging PostgreSQL's JSONB capabilities for FHIR resources, alongside traditional relational tables for auxiliary data, and AWS S3 for file storage, offers a robust and flexible solution. This approach allows Healthify to store clinical data in its canonical FHIR form, directly supporting the platform's strategic objective of enhanced interoperability and reduced future integration costs. The detailed mapping of Healthify's core functionalities to specific FHIR resources—such as `Patient`, `Practitioner`, `Appointment`, `Encounter`, `MedicationRequest`, `Observation`, `Condition`, `DiagnosticReport`, `DocumentReference`, `Communication`, `ChargeItem`, `Invoice`, and `CarePlan`—provides a clear pathway for data transformation and module design. A critical aspect of this transformation involves moving beyond generic data storage to embrace the semantic depth of FHIR. For instance, transforming the simplistic `encounters` and `appointments` tables into their rich FHIR `Encounter` and `Appointment` counterparts will enable Healthify to capture a far more comprehensive and clinically meaningful representation of patient interactions. Similarly, adopting distinct FHIR resources for `Patient`, `Practitioner`, and

`PractitionerRole` will resolve current user identity modeling inconsistencies, facilitating more precise access control and data sharing. The explicit adoption of FHIR `AuditEvent` and `Provenance` resources for audit logging is not merely a compliance measure but a strategic enhancement that provides detailed, standardized, and tamper-proof records of all sensitive data interactions, crucial for both regulatory adherence and operational transparency. The integration of Supabase for database management and authentication further streamlines development while maintaining compatibility with the PostgreSQL-based FHIR strategy.

Strategic Recommendations:

Based on the comprehensive analysis, the following strategic recommendations are put forth to guide the successful development and deployment of Healthify:

- **Prioritize FHIR-Native Development:** Adopt a "FHIR-first" mindset for all new data module development and refactoring efforts. Design internal data structures to directly mirror FHIR resources, minimizing the need for complex translation layers.
- **Strategic FHIR Server Selection:** Evaluate managed FHIR services (e.g., Google Cloud Healthcare API, Azure API for FHIR) against open-source alternatives (like HAPI FHIR or building upon `node-fhir-server-core`) based on total cost of ownership, development velocity, and long-term maintenance capabilities. For rapid deployment and reduced operational burden, managed services are generally advantageous.
- **Invest in Comprehensive Profiling:** Develop and rigorously document custom FHIR profiles and extensions for Healthify-specific use cases that are not fully covered by base FHIR resources. This ensures both compliance and the ability to capture unique business requirements.
- **Implement Robust Data Governance:** Establish clear data governance policies for FHIR resource creation, update, and deletion. This includes defining data ownership across microservices and ensuring referential integrity for cross-service FHIR `Reference` elements.
- **Enhance Audit and Security Frameworks:** Fully integrate FHIR `AuditEvent` and `Provenance` resources into the security and compliance framework. Ensure that every sensitive data interaction generates a detailed, immutable audit record, reinforcing HIPAA compliance and providing granular accountability.
- **Meticulous Phased Migration:** Plan a thorough, phased data migration from existing relational schemas to the new FHIR-compliant structures. This should include comprehensive data mapping, automated transformation scripts, and rigorous validation to ensure data integrity and minimize service disruption.
- **Continuous Learning and Adaptation:** Recognize that FHIR is an evolving standard. Allocate resources for continuous learning, monitoring new FHIR releases and implementation guides, and participating in the FHIR community to ensure Healthify remains at the forefront of health interoperability.

Next Steps:

To transition from this specification to implementation, immediate actions are required. These include detailed design sessions for wireframes and Entity-Relationship Diagrams, a technology spike for Supabase integration to validate its specific use cases within the microservices architecture, and comprehensive data mapping workshops to finalize the transformation logic for existing data. By systematically addressing these architectural and design recommendations, Healthify can establish a highly interoperable, scalable, and compliant digital health platform, well-positioned to meet current healthcare needs and adapt to future demands.