# MEDLYGO

Non-Emergency Hospital Patient Booking
Web Application for Public Hospitals in Ghana

## TECHNICAL DOCUMENTATION

Version 1.0
January 2026

medlygo.com

# Table of Contents

# 1. Executive Summary

MedlyGo is a modern, cloud-native web application designed to revolutionise non-emergency healthcare appointment scheduling for public hospitals in Ghana. This technical documentation outlines the architecture, technology choices, and implementation strategy for building a secure, scalable, and user-centric platform.

The system leverages cutting-edge technologies, including Next.js for the frontend, Node.js with Express for the backend, and Supabase for database hosting, all deployed on a modern infrastructure stack comprising Vercel for application hosting and Cloudflare for DNS management and security.

## 1.1 Key Objectives

- Reduce patient waiting times through efficient digital scheduling
- Decrease no-show rates via automated SMS and email reminders
- Optimise hospital resource allocation with real-time analytics
- Provide an AI-powered assistant for intelligent scheduling support
- Ensure compliance with the Ghana Data Protection Act 2012 (Act 843)
- Support multilingual interfaces (English, Twi, Ga, Ewe, Hausa)

# 2. Technology Stack

The technology stack has been carefully selected to balance modern development practices, cost-effectiveness, and suitability for the Ghanaian healthcare context.

## 2.1 Frontend Technologies

| Technology | Version | Purpose |
|---|---|---|
| Next.js | 14.x (App Router) | React framework with SSR/SSG |
| TypeScript | 5.x | Type-safe development |
| Tailwind CSS | 3.x | Utility-first CSS framework |
| shadcn/ui | Latest | Accessible UI components |
| React Query | 5.x (TanStack) | Server state management |
| Zustand | 4.x | Client state management |

## 2.2 Backend Technologies

| Technology | Version | Purpose |
|---|---|---|
| Node.js | 20.x LTS | JavaScript runtime |
| Supabase | Latest | PostgreSQL + Auth + Realtime |
| Prisma | 5.x | Type-safe ORM |
| Upstash Redis | Serverless | Caching and rate limiting |

## 2.3 Infrastructure Stack

| Service | Provider | Justification |
|---|---|---|
| App Hosting | **Vercel** | Optimised for Next.js, global CDN |
| Database | **Supabase** | PostgreSQL, built-in auth, realtime, RLS |
| DNS / CDN | **Cloudflare** | DDoS protection, WAF, SSL, edge network |
| Domain | **Namecheap** | medlygo.com registered |
| Email | **Resend** | Modern email API, React Email support |

| | | |
|---|---|---|
| SMS | **Hubtel / Twilio** | Local Ghana coverage + global fallback |

# 3. System Architecture

## 3.1 High-Level Architecture

MedlyGo follows a modern three-tier architecture with clear separation of concerns:

**PRESENTATION LAYER**

Patient Portal | Provider Dashboard | Admin Console | AI Chat Widget

(Next.js 14 + React + Tailwind CSS + shadcn/ui)

**APPLICATION LAYER**

Auth Service | Appointment Service | Notification Service | AI Agent Service

(Vercel Serverless Functions + Edge Functions)

**DATA LAYER**

PostgreSQL | Supabase Auth | Realtime Subscriptions | File Storage

(Supabase with Row-Level Security)

## 3.2 Component Architecture

The application follows a feature-based folder structure optimised for Next.js 14 App Router:

| Directory | Purpose |
|---|---|
| **app/** | Next.js App Router - pages and API routes |
| (auth)/ | Authentication pages (login, signup, reset) |
| (patient)/ | Patient portal routes (booking, history, profile) |
| (provider)/ | Provider dashboard routes (schedule, patients) |
| (admin)/ | Admin console routes (analytics, settings) |
| api/ | API route handlers (REST endpoints) |
| **components/** | Reusable UI components (buttons, forms, cards) |
| **lib/** | Utilities, configurations, Supabase client |
| **hooks/** | Custom React hooks (useAuth, useAppointments) |
| **services/** | API service functions and business logic |
| **stores/** | Zustand state stores (user, UI, notifications) |
| **types/** | TypeScript type definitions and interfaces |

| | |
|---|---|
| `locales/` | i18n translation files (en, tw, ga, ee, ha) |
| `emails/` | React Email templates for Resend |

# 4. Database Design

The database schema is designed using PostgreSQL hosted on Supabase, leveraging row-level security (RLS) for data isolation and security.

## 4.1 Entity Relationship Overview

| Entity | Description |
|---|---|
| `users` | Core user profiles linked to Supabase Auth (role, preferences) |
| `patients` | Patient-specific data (Ghana Card ID, medical history ref) |
| `providers` | Healthcare provider profiles (specialisation, credentials) |
| `hospitals` | Healthcare facility info (location, services offered) |
| `departments` | Hospital departments (General Medicine, Pediatrics, etc.) |
| `schedules` | Provider availability schedules with time slots config |
| `appointments` | Core appointment records (status tracking, notes) |
| `notifications` | Notification logs for SMS and email reminders |
| `feedback` | Patient feedback and ratings after appointments |

## 4.2 Supabase Row-Level Security

RLS policies ensure data isolation and access control at the database level:

- Patients can only view and modify their own appointments
- Providers can view appointments assigned to them
- Admins have full access within their hospital scope
- Service role bypasses RLS for background jobs

# 5. AI Agent Architecture

The AI agent is a core differentiator for MedlyGo, providing intelligent conversational assistance for appointment booking, rescheduling, and general healthcare inquiries.

## 5.1 Recommended Technology Stack

| Component | Technology | Rationale |
|---|---|---|
| LLM Provider | **Claude 3.5 Sonnet** | Best reasoning, safety features, healthcare-appropriate |
| Framework | **LangChain + LangGraph** | Industry standard, complex workflow support |
| Observability | **LangSmith** | Audit trails, PII redaction, compliance monitoring |
| Vector Store | **Supabase pgvector** | Native PostgreSQL, no extra infrastructure |
| Chat UI | **Vercel AI SDK** | Streaming responses, React hooks, type-safe |

## 5.2 Agent Capabilities

### Appointment Management

- Book new appointments based on patient preferences and availability
- Reschedule existing appointments with conflict detection
- Cancel appointments with appropriate notice handling
- Provide appointment status and history summaries

### Healthcare Guidance

- Recommend appropriate department/specialist based on symptoms
- Provide pre-visit preparation instructions
- Triage urgency assessment (with disclaimers for emergencies)

### Multilingual Support

English (primary), Twi (Akan), Ga, Ewe, and Hausa

## 5.3 Agent Workflow Design

The agent follows a ReAct (Reasoning + Acting) pattern with human-in-the-loop for critical actions:

| | | | |
|---|---|---|---|
| User Message<br>Language Detection | Intent Recognition<br>Route to Handler | Call Tools<br>Confirm Actions | Format Response<br>Send Notifications |

### Workflow Routes

| | |
|---|---|
| **Booking Flow** | Check availability → Confirm with user → Execute booking → Send confirmation |
| **Query Flow** | Fetch data from database → Format response → Return to user |
| **FAQ Flow** | RAG retrieval from knowledge base → Generate contextual answer |

## 5.4 Tool Definitions

| Tool Name | Description |
|---|---|
| `check_availability` | Query available time slots for provider/department/date |
| `book_appointment` | Create a new appointment after user confirmation |
| `reschedule_appointment` | Modify the existing appointment to a new time slot |
| `cancel_appointment` | Cancel appointment with reason tracking |
| `get_patient_appointments` | Retrieve appointment history for the current patient |
| `search_providers` | Search healthcare providers by specialty or name |
| `get_preparation_info` | Retrieve pre-visit preparation instructions |

# 6. Security and Compliance

## 6.1 Data Protection Compliance

MedlyGo complies with the Ghana Data Protection Act 2012 (Act 843) as the primary legal framework, while adopting HIPAA and GDPR as international benchmarks.

### Key Compliance Requirements

- Explicit consent collection before processing personal health information
- Data minimisation - collect only necessary information
- Purpose limitation - use data only for stated purposes
- Right to access, rectification, and erasure
- Breach notification within 72 hours

## 6.2 Security Architecture

| Layer | Implementation |
|---|---|
| Authentication | Supabase Auth (email/password, phone OTP, Google OAuth) |
| MFA | Required for provider and admin accounts via Supabase MFA |
| Authorization | Role-Based Access Control (RBAC) with Supabase RLS policies |
| Encryption (Transit) | TLS 1.3 enforced via Cloudflare and Vercel |
| Encryption (Rest) | AES-256 encryption for database (Supabase managed) |
| Session Mgmt | JWT tokens (1-hour access, 7-day refresh) |

## 6.3 Infrastructure Security

- Cloudflare WAF for web application firewall protection
- DDoS protection via Cloudflare automatic mitigation
- Rate limiting via Upstash Redis at the API layer
- Security headers enforced via Next.js middleware
- Comprehensive audit logging for all data access

# 7. Deployment and Infrastructure

## 7.1 Domain and DNS Configuration

The domain medlygo.com is registered with Namecheap and configured to use Cloudflare as the authoritative DNS provider.

| Type | Name | Value | Purpose |
|---|---|---|---|
| CNAME | @ | `cname.vercel-dns.com` | Root domain |
| CNAME | www | `cname.vercel-dns.com` | WWW redirect |
| TXT | @ | (Vercel verification) | Verification |

## 7.2 Vercel Deployment

- Production branch: main
- Preview deployments: All pull requests
- Build command: npm run build
- Framework preset: Next.js
- Node.js version: 20.x

## 7.3 Environment Variables

Required environment variables organised by service (stored securely in Vercel):

| Variable | Description |
|---|---|
| **SUPABASE** | |
| `NEXT_PUBLIC_SUPABASE_URL` | Supabase project URL |
| `NEXT_PUBLIC_SUPABASE_ANON_KEY` | Supabase anonymous/public key |
| `SUPABASE_SERVICE_ROLE_KEY` | Supabase service role key (server-side only) |
| **AI AGENT** | |
| `ANTHROPIC_API_KEY` | Claude API key for AI agent |
| `LANGCHAIN_API_KEY` | LangSmith API key for observability |
| `LANGCHAIN_PROJECT` | LangSmith project name |
| **NOTIFICATIONS** | |
| `RESEND_API_KEY` | Resend API key for email |
| `HUBTEL_CLIENT_ID` | Hubtel SMS client ID |

| | |
|---|---|
| HUBTEL_CLIENT_SECRET | Hubtel SMS client secret |
| TWILIO_ACCOUNT_SID | Twilio account SID (fallback) |
| TWILIO_AUTH_TOKEN | Twilio auth token (fallback) |
| **CACHING** | |
| UPSTASH_REDIS_REST_URL | Upstash Redis REST URL |
| UPSTASH_REDIS_REST_TOKEN | Upstash Redis REST token |
| **APPLICATION** | |
| NEXT_PUBLIC_APP_URL | https://medlygo.com |
| NODE_ENV | production |

# 8. Notification Services

## 8.1 SMS Notifications

SMS notifications are critical for appointment reminders in Ghana, where mobile phone penetration is high.

### Primary: Hubtel SMS API

- Ghana-based provider with excellent local network coverage
- Competitive pricing for local SMS delivery
- Supports sender ID customisation (MEDLYGO)

### Fallback: Twilio

- Global reliability and redundancy
- Excellent API documentation
- Webhook support for delivery receipts

## 8.2 Email Notifications (Resend)

Resend is a modern email API designed for developers, with native support for React Email templates.

### Why Resend

- Modern developer experience with excellent TypeScript support
- React Email integration for building email templates as React components
- High deliverability with dedicated IPs available
- Simple pricing with generous free tier (3,000 emails/month)
- Built-in analytics and tracking

## 8.3 Notification Schedule

| Trigger | Channel | Content |
|---|---|---|
| Booking confirmed | SMS + Email | Confirmation with details and reference number |
| 48 hours before | SMS | Reminder with reschedule/cancel options |
| 24 hours before | SMS + Email | Final reminder with preparation instructions |
| 2 hours before | SMS | Short reminder with location/directions |
| After appointment | Email | Feedback request and next steps |

# 9. Development Roadmap

## 9.1 Phase 1: Foundation (Week 1)

- Initialise Next.js 14 project with TypeScript
- Set up a Supabase project with a database schema
- Configure Cloudflare DNS and Vercel deployment
- Implement authentication flow (email, phone, Google)
- Set up CI/CD pipeline with GitHub Actions

## 9.2 Phase 2: Core Features (Week 2-3)

- Hospital and department selection interface
- Provider availability calendar with real-time updates
- Appointment booking, modification, and cancellation
- Patient profile and appointment history
- Provider dashboard with schedule management
- Patient check-in functionality

## 9.3 Phase 3: Notifications and AI (Week 4-5)

- SMS integration with Hubtel/Twilio
- Email templates with Resend + React Email
- Scheduled reminder jobs via Vercel Cron
- LangChain agent setup with Claude integration
- Chat widget integration in patient portal
- Multilingual support configuration

## 9.4 Phase 4: Admin and Analytics (Week 5-6)

- Admin console for hospital management
- Analytics dashboard with key metrics
- Patient feedback system
- Reporting and export functionality

## 9.5 Phase 5: Testing and Launch (Weeks 7-8)

- Comprehensive testing (unit, integration, E2E)
- Security audit and penetration testing
- Performance optimisation
- User acceptance testing with a pilot hospital
- Staff training and documentation

- Production launch

# 11. Appendices

## A. API Endpoint Reference

| Method | Endpoint | Description |
|---|---|---|
| **AUTHENTICATION** | | |
| POST | `/api/auth/signup` | Create a new account |
| POST | `/api/auth/login` | Authenticate user |
| **APPOINTMENTS** | | |
| GET | `/api/appointments` | List user appointments |
| POST | `/api/appointments` | Create appointment |
| PATCH | `/api/appointments/:id` | Update appointment |
| DELETE | `/api/appointments/:id` | Cancel appointment |
| **AI AGENT** | | |
| POST | `/api/chat` | Send a message to the AI agent |

## B. Glossary

- RLS: Row-Level Security - database access control at the row level
- SSR: Server-Side Rendering - generating HTML on the server
- JWT: JSON Web Token - secure token for authentication
- RBAC: Role-Based Access Control - a permission system based on user roles
- CDN: Content Delivery Network - a distributed system for fast content delivery
- WAF: Web Application Firewall - protects against web exploits
- ORM: Object-Relational Mapping - database abstraction layer
- FHIR: Fast Healthcare Interoperability Resources - healthcare data exchange standard