



Ways and Means Technology Pvt. Ltd.

ARCHITECTURE

Physiomates CRM

May 19, 2026

System Components (8)

Name	Type	Technology	Description	Connections
Automation Framework	SERVICE	Selenium , Java	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s,	• OS — I • Windows — II • Linux — III • Unix — IV
Frontend SPA	FRONTEND	React + Vite + Tailwind CSS	Single-page application serving the admin dashboard, lead management, scheduling, and product sales UI. Uses Redux Toolkit for state management, React Router for navigation, React Hook Form + Zod for form handling and validation.	• Backend API — REST API calls
Express API Server	SERVICE	Node.js, Express	Node.js Express server (server.js) — entry point. Configures Helmet, CORS, rate limiting (200/15min), mongo-sanitize, morgan logging, JSON parsing (50mb). Mounts all routes under /api/v1/. Health check at /health. Startup: connectDB → seedSuperAdmin → initJobs → listen on port 6667.	—
Backend API	SERVICE	Node.js + Express	RESTful API server handling authentication, lead management, email automation, scheduling, products, meetings, and transcripts. Follows Controller → Service → Repository pattern. All endpoints prefixed with /api/v1/.	• MongoDB — Mongoose ODM • Email Service — Nodemailer • Frontend SPA — JSON responses
MongoDB Database	DATABASE	MongoDB, Mongoose	Remote MongoDB instance via Mongoose ODM. 15 collections. All models use baseSchema with soft delete (softDelete: boolean), timestamps (createdAt/updatedAt), createdBy/updatedBy audit fields. Indexes on frequently queried fields. Connection in config/db.js with auto-reconnect error handling.	—
MongoDB	DATABASE	MongoDB (Remote Instance)	Remote MongoDB instance storing all application data including users, leads, email logs, products, meetings, and transcripts. Accessed via Mongoose ODM with schema validation. Hosted at 163.227.92.149:27017.	• Backend API — Mongoose queries
Email Service	EXTERNAL_API	Nodemailer + SMTP	Email automation engine handling welcome emails, questionnaire links, and template-based notifications. Uses {{variable}} placeholder system. Falls back to console logging in dev mode when SMTP is not configured.	• SMTP Server — SMTP protocol • Backend API — Email triggers
SMTP Server	EXTERNAL_API	External SMTP Provider	External SMTP server for outbound email delivery. Configurable via environment variables (SMTP_HOST, SMTP_PORT, SMTP_USER, SMTP_PASS).	• Email Service — SMTP relay

Architecture Decisions (4)

ADR-001: Controller-Service-Repository layered architecture [ACCEPTED]

Context: The backend serves multiple domains (auth, leads, emails, scheduling, products, meetings, transcripts) and needs a consistent, scalable code organization pattern.

Decision: Adopt Controller → Service → Repository layered architecture with module-based folder structure. Each module encapsulates its own routes, controller, service, repository, and model.

Consequences: Clear separation of concerns makes the codebase maintainable and testable. New modules (scheduling, products, meetings) follow the same pattern. Trade-off: more boilerplate per module compared to a simpler flat structure.

ADR-002: Nodemailer with template-based email automation [ACCEPTED]

Context: The CRM requires automated email workflows — welcome emails on lead creation, questionnaire links, and admin-triggered resends. Emails must be branded HTML with dynamic content.

Decision: Use Nodemailer with configurable SMTP transport and a {{variable}} template placeholder system. Fall back to console logging when SMTP is not configured for development environments.

Consequences: Automated lead nurturing reduces manual follow-up. Template system enables non-technical users to customize emails. Console fallback in dev mode prevents SMTP dependency during development. Trade-off: email deliverability depends on SMTP provider configuration.

ADR-003: JWT dual-token authentication with RBAC [ACCEPTED]

Context: The application needs authentication supporting multiple roles (Super Admin, Admin, Counselor, Lead) with both admin dashboard and public form access.

Decision: Use JWT with dual-token strategy — short-lived access tokens (15m) for API auth and long-lived refresh tokens (7d) for session continuity. Role-based access control enforced via middleware.

Consequences: Stateless auth scales horizontally without session storage. Short-lived access tokens (15m) limit exposure window. Refresh tokens (7d) provide seamless UX. Trade-off: token revocation requires additional logic (e.g., blacklist).

ADR-004: Use MongoDB over relational database for primary storage [ACCEPTED]

Context: The CRM needs to store leads, users, email logs, products, meetings, and transcripts with varying schemas. Lead data includes dynamic fields from public forms and questionnaires.

Decision: Use MongoDB as the primary database with Mongoose ODM for schema validation. MongoDB's document model fits the CRM's varied data shapes and allows schema evolution without migrations.

Consequences: Flexible schema allows rapid iteration on lead/email/product models. No complex migrations needed. Trade-off: no native joins — must use aggregation pipelines or application-level joins for cross-collection queries.