



Ways and Means Technology Pvt. Ltd.

GENERAL

Physiotherapy CRM — Technical Documentation

Living technical documentation maintained alongside development

Physiomates CRM · Ways and Means Technology Pvt. Ltd. · May 19, 2026

Table of Contents

1. Architecture & Infrastructure	
2. Database Schema	
3. API Reference	
4. Authentication & Authorization	
5. Business Logic	
6. Environment Configuration	
7. Deployment Guide	
8. Integrations	
9. Error Handling & Logging	
10. Testing Strategy	
11. Changelog	

1. Architecture & Infrastructure

Architecture & Infrastructure

High-Level Architecture

The system follows a **modular monolith** pattern — a single NestJS application organized into domain modules that communicate through well-defined internal interfaces.

Component Responsibilities

Frontend SPA (React 18 + Vite), Patient Portal (React 18 + Vite), Reverse Proxy (Nginx 1.25), API Server (NestJS 10), ORM (Prisma 5), Primary Database (PostgreSQL 15), Audit Database (MongoDB 7), Cache Layer (Redis 7), Process Manager (PM2), File Storage (Local filesystem).

Request Flow

1. Client sends HTTPS request to Nginx
2. Nginx terminates SSL, routes /api/* to NestJS, serves static files for /*
3. NestJS middleware chain: CORS → Rate Limit → Auth Guard → Validation Pipe → Controller
4. Controller delegates to Service layer (business logic)
5. Service uses Prisma for PostgreSQL queries, Mongoose for MongoDB audit writes
6. Response passes through interceptors (transform, logging) back to client

2. Database Schema

Database Schema

All business data is stored in **PostgreSQL 15** managed via **Prisma ORM**. Below are the core tables, their key columns, relationships, and indexes.

users

Column	Type	Constraints	Description
id	UUID	PK, DEFAULT gen_random_uuid()	Primary key
email	VARCHAR(255)	UNIQUE, NOT NULL	Login email
password_hash	VARCHAR(255)	NOT NULL	bcrypt hash
first_name	VARCHAR(100)	NOT NULL	First name
last_name	VARCHAR(100)	NOT NULL	Last name
role_id	UUID	FK → roles.id	User role
is_active	BOOLEAN	DEFAULT true	Account status
last_login_at	TIMESTAMPTZ	NULLABLE	Last login timestamp

Column	Type	Constraints	Description
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Updated timestamp

Indexes: `idx_users_email` (UNIQUE), `idx_users_role_id`

roles

Column	Type	Constraints	Description
id	UUID	PK	Primary key
name	VARCHAR(50)	UNIQUE, NOT NULL	ADMIN, THERAPIST, RECEPTIONIST, PATIENT
permissions	JSONB	DEFAULT '[]'	Array of permission strings
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

patients

Column	Type	Constraints	Description
id	UUID	PK	Primary key
user_id	UUID	FK → users.id, UNIQUE, NULLABLE	Portal login (optional)
patient_code	VARCHAR(20)	UNIQUE, NOT NULL	Auto-generated: PAT-00001
first_name	VARCHAR(100)	NOT NULL	First name
last_name	VARCHAR(100)	NOT NULL	Last name
email	VARCHAR(255)	NULLABLE	Contact email
phone	VARCHAR(20)	NULLABLE	Phone number
date_of_birth	DATE	NOT NULL	Date of birth
gender	VARCHAR(10)	NOT NULL	MALE, FEMALE, OTHER
address	TEXT	NULLABLE	Full address
medical_history	JSONB	DEFAULT '{}'	Conditions, allergies, medications
insurance_provider	VARCHAR(100)	NULLABLE	Insurance company
insurance_policy_no	VARCHAR(50)	NULLABLE	Policy number
status	VARCHAR(20)	DEFAULT 'ACTIVE'	ACTIVE, DISCHARGED, INACTIVE

Column	Type	Constraints	Description
primary_therapist_id	UUID	FK → therapists.id, NULLABLE	Assigned therapist
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Updated timestamp

Indexes: `idx_patients_code` (UNIQUE), `idx_patients_email`, `idx_patients_status`, `idx_patients_therapist`

therapists

Column	Type	Constraints	Description
id	UUID	PK	Primary key
user_id	UUID	FK → users.id, UNIQUE, NOT NULL	Linked user account
specializations	TEXT[]	DEFAULT '{}'	Array of specializations
license_number	VARCHAR(50)	UNIQUE, NOT NULL	Professional license
bio	TEXT	NULLABLE	Professional bio
hourly_rate	DECIMAL(10,2)	NOT NULL	Session rate
is_active	BOOLEAN	DEFAULT true	Active status
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

Indexes: `idx_therapists_user_id` (UNIQUE), `idx_therapists_license` (UNIQUE)

appointments

Column	Type	Constraints	Description
id	UUID	PK	Primary key
patient_id	UUID	FK → patients.id, NOT NULL	Patient
therapist_id	UUID	FK → therapists.id, NOT NULL	Therapist
treatment_plan_id	UUID	FK → treatment_plans.id, NULLABLE	Linked plan
start_time	TIMESTAMPTZ	NOT NULL	Appointment start
end_time	TIMESTAMPTZ	NOT NULL	Appointment end
status	VARCHAR(20)	DEFAULT 'SCHEDULED'	SCHEDULED, CONFIRMED, IN_PROGRESS, COMPLETED, CANCELLED, NO_SHOW

Column	Type	Constraints	Description
type	VARCHAR(30)	NOT NULL	INITIAL_ASSESSMENT, FOLLOW_UP, TREATMENT, REVIEW
notes	TEXT	NULLABLE	Session notes
google_event_id	VARCHAR(255)	NULLABLE	Google Calendar sync ID
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Updated timestamp

Indexes: `idx_appointments_patient`, `idx_appointments_therapist`, `idx_appointments_time` (start_time, end_time), `idx_appointments_status`

Constraint: `chk_appointment_time` — end_time > start_time

treatment_plans

Column	Type	Constraints	Description
id	UUID	PK	Primary key
patient_id	UUID	FK → patients.id, NOT NULL	Patient
therapist_id	UUID	FK → therapists.id, NOT NULL	Treating therapist
diagnosis	TEXT	NOT NULL	Primary diagnosis
goals	JSONB	DEFAULT '[]'	Treatment goals array
total_sessions	INTEGER	NOT NULL	Planned session count
completed_sessions	INTEGER	DEFAULT 0	Completed count
status	VARCHAR(20)	DEFAULT 'DRAFT'	DRAFT, ACTIVE, COMPLETED, ARCHIVED
start_date	DATE	NOT NULL	Plan start date
end_date	DATE	NULLABLE	Planned end date
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Updated timestamp

Indexes: `idx_tp_patient`, `idx_tp_therapist`, `idx_tp_status`

treatment_sessions

Column	Type	Constraints	Description
id	UUID	PK	Primary key

Column	Type	Constraints	Description
treatment_plan_id	UUID	FK → treatment_plans.id, NOT NULL	Parent plan
appointment_id	UUID	FK → appointments.id, UNIQUE, NULLABLE	Linked appointment
session_number	INTEGER	NOT NULL	Session sequence number
subjective_notes	TEXT	NULLABLE	Patient-reported symptoms (SOAP)
objective_notes	TEXT	NULLABLE	Therapist observations (SOAP)
assessment	TEXT	NULLABLE	Clinical assessment (SOAP)
plan_notes	TEXT	NULLABLE	Next steps (SOAP)
pain_level	INTEGER	CHECK (0-10)	Pain scale 0-10
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

exercise_library

Column	Type	Constraints	Description
id	UUID	PK	Primary key
name	VARCHAR(200)	NOT NULL	Exercise name
description	TEXT	NOT NULL	Instructions
body_region	VARCHAR(50)	NOT NULL	NECK, SHOULDER, BACK, KNEE, HIP, ANKLE, WRIST, FULL_BODY
difficulty	VARCHAR(20)	NOT NULL	BEGINNER, INTERMEDIATE, ADVANCED
video_url	VARCHAR(500)	NULLABLE	YouTube/Vimeo embed URL
image_url	VARCHAR(500)	NULLABLE	Illustration image
is_active	BOOLEAN	DEFAULT true	Active status
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

Indexes: `idx_exercises_body_region` , `idx_exercises_difficulty`

exercises (prescribed)

Column	Type	Constraints	Description
id	UUID	PK	Primary key
treatment_plan_id	UUID	FK → treatment_plans.id, NOT NULL	Parent plan

Column	Type	Constraints	Description
exercise_library_id	UUID	FK → exercise_library.id, NOT NULL	Library reference
sets	INTEGER	DEFAULT 3	Number of sets
reps	INTEGER	DEFAULT 10	Reps per set
hold_seconds	INTEGER	NULLABLE	Hold duration
frequency	VARCHAR(50)	NOT NULL	e.g., "2x daily", "3x weekly"
notes	TEXT	NULLABLE	Special instructions
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

invoices

Column	Type	Constraints	Description
id	UUID	PK	Primary key
invoice_number	VARCHAR(20)	UNIQUE, NOT NULL	Auto-generated: INV-00001
patient_id	UUID	FK → patients.id, NOT NULL	Billed patient
subtotal	DECIMAL(10,2)	NOT NULL	Pre-tax total
tax_amount	DECIMAL(10,2)	DEFAULT 0	Tax amount
total_amount	DECIMAL(10,2)	NOT NULL	Final total
paid_amount	DECIMAL(10,2)	DEFAULT 0	Amount paid so far
status	VARCHAR(20)	DEFAULT 'DRAFT'	DRAFT, SENT, PAID, PARTIALLY_PAID, OVERDUE, CANCELLED
due_date	DATE	NOT NULL	Payment due date
issued_date	DATE	NOT NULL	Invoice issue date
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp
updated_at	TIMESTAMPTZ	DEFAULT NOW()	Updated timestamp

Indexes: `idx_invoices_number` (UNIQUE), `idx_invoices_patient`, `idx_invoices_status`, `idx_invoices_due_date`

invoice_items

Column	Type	Constraints	Description
id	UUID	PK	Primary key
invoice_id	UUID	FK → invoices.id, NOT NULL	Parent invoice
appointment_id	UUID	FK → appointments.id, NULLABLE	Linked session
description	VARCHAR(255)	NOT NULL	Line item description
quantity	INTEGER	DEFAULT 1	Quantity
unit_price	DECIMAL(10,2)	NOT NULL	Price per unit
total	DECIMAL(10,2)	NOT NULL	Line total

payments

Column	Type	Constraints	Description
id	UUID	PK	Primary key
invoice_id	UUID	FK → invoices.id, NOT NULL	Linked invoice
amount	DECIMAL(10,2)	NOT NULL	Payment amount
method	VARCHAR(20)	NOT NULL	CASH, CARD, STRIPE, BANK_TRANSFER, INSURANCE
stripe_payment_id	VARCHAR(255)	NULLABLE	Stripe payment intent ID
status	VARCHAR(20)	DEFAULT 'COMPLETED'	COMPLETED, REFUNDED, FAILED
paid_at	TIMESTAMPTZ	DEFAULT NOW()	Payment timestamp
created_at	TIMESTAMPTZ	DEFAULT NOW()	Created timestamp

clinic_settings

Column	Type	Constraints	Description
id	UUID	PK	Primary key (singleton row)
clinic_name	VARCHAR(200)	NOT NULL	Clinic display name
address	TEXT	NULLABLE	Clinic address
phone	VARCHAR(20)	NULLABLE	Clinic phone
email	VARCHAR(255)	NULLABLE	Clinic email
working_hours	JSONB	NOT NULL	Per-day open/close times

Column	Type	Constraints	Description
appointment_duration_mins	INTEGER	DEFAULT 60	Default slot duration
tax_rate	DECIMAL(5,2)	DEFAULT 0	Tax percentage
currency	VARCHAR(3)	DEFAULT 'USD'	Currency code
updated_at	TIMESTAMP TZ	DEFAULT NOW()	Updated timestamp

Entity Relationship Summary

- users 1:1 therapists (via user_id)
- users 1:1 patients (via user_id, optional)
- patients 1:N appointments
- therapists 1:N appointments
- patients 1:N treatment_plans
- treatment_plans 1:N treatment_sessions
- treatment_plans 1:N exercises
- exercise_library 1:N exercises
- patients 1:N invoices
- invoices 1:N invoice_items
- invoices 1:N payments

3. API Reference

API Reference

All endpoints are prefixed with `/api/v1` . Authentication is via `Bearer` token in the `Authorization` header unless marked as Public.

Authentication

Method	Path	Description	Auth
POST	/api/v1/auth/login	Login with email/password, returns access + refresh tokens	Public
POST	/api/v1/auth/refresh	Refresh access token using refresh token	Public
POST	/api/v1/auth/logout	Invalidate refresh token, clear Redis session	Required
GET	/api/v1/auth/me	Get current user profile	Required
POST	/api/v1/auth/forgot-password	Send password reset email	Public
POST	/api/v1/auth/reset-password	Reset password with token	Public
PATCH	/api/v1/auth/change-password	Change password (authenticated)	Required

Patients

Method	Path	Description	Auth
GET	/api/v1/patients	List patients (paginated, filterable by status, therapist)	ADMIN, THERAPIST, RECEPTIONIST
GET	/api/v1/patients/:id	Get patient details with medical history	ADMIN, THERAPIST, RECEPTIONIST
POST	/api/v1/patients	Create new patient	ADMIN, RECEPTIONIST
PATCH	/api/v1/patients/:id	Update patient details	ADMIN, THERAPIST, RECEPTIONIST
DELETE	/api/v1/patients/:id	Soft-delete patient (set INACTIVE)	ADMIN
GET	/api/v1/patients/:id/appointments	Get patient's appointment history	ADMIN, THERAPIST, RECEPTIONIST
GET	/api/v1/patients/:id/treatment-plans	Get patient's treatment plans	ADMIN, THERAPIST
GET	/api/v1/patients/:id/invoices	Get patient's invoices	ADMIN, RECEPTIONIST
POST	/api/v1/patients/:id/discharge	Discharge patient (status → DISCHARGED)	ADMIN, THERAPIST

Appointments

Method	Path	Description	Auth
GET	/api/v1/appointments	List appointments (filterable by date range, therapist, patient, status)	ADMIN, THERAPIST, RECEPTIONIST
GET	/api/v1/appointments/:id	Get appointment details	ADMIN, THERAPIST, RECEPTIONIST
POST	/api/v1/appointments	Create appointment (validates conflicts)	ADMIN, THERAPIST, RECEPTIONIST
PATCH	/api/v1/appointments/:id	Update appointment (reschedule, add notes)	ADMIN, THERAPIST, RECEPTIONIST
PATCH	/api/v1/appointments/:id/status	Change status (CONFIRMED, COMPLETED, CANCELLED, NO_SHOW)	ADMIN, THERAPIST, RECEPTIONIST
GET	/api/v1/appointments/availability	Get available slots for a therapist on a date	ADMIN, THERAPIST, RECEPTIONIST
POST	/api/v1/appointments/:id/sync-calendar	Sync appointment to Google Calendar	ADMIN, THERAPIST

Treatment Plans

Method	Path	Description	Auth
GET	/api/v1/treatment-plans	List treatment plans (filterable by patient, therapist, status)	ADMIN, THERAPIST
GET	/api/v1/treatment-plans/:id	Get plan with sessions and exercises	ADMIN, THERAPIST
POST	/api/v1/treatment-plans	Create treatment plan	THERAPIST
PATCH	/api/v1/treatment-plans/:id	Update plan details	THERAPIST
PATCH	/api/v1/treatment-plans/:id/status	Change status (DRAFT → ACTIVE → COMPLETED → ARCHIVED)	THERAPIST
POST	/api/v1/treatment-plans/:id/sessions	Add treatment session (SOAP notes)	THERAPIST
POST	/api/v1/treatment-plans/:id/exercises	Prescribe exercises from library	THERAPIST
DELETE	/api/v1/treatment-plans/:id/exercises/:exerciseld	Remove prescribed exercise	THERAPIST

Exercises

Method	Path	Description	Auth
GET	/api/v1/exercises	List exercise library (filterable by body_region, difficulty)	ADMIN, THERAPIST
GET	/api/v1/exercises/:id	Get exercise details with video	ADMIN, THERAPIST, PATIENT
POST	/api/v1/exercises	Add exercise to library	ADMIN, THERAPIST
PATCH	/api/v1/exercises/:id	Update exercise	ADMIN, THERAPIST
DELETE	/api/v1/exercises/:id	Soft-delete exercise (set inactive)	ADMIN

Billing — Invoices

Method	Path	Description	Auth
GET	/api/v1/invoices	List invoices (filterable by patient, status, date range)	ADMIN, RECEPTIONIST
GET	/api/v1/invoices/:id	Get invoice with line items and payments	ADMIN, RECEPTIONIST
POST	/api/v1/invoices	Create invoice (manual or from completed sessions)	ADMIN, RECEPTIONIST

Method	Path	Description	Auth
POST	/api/v1/invoices/generate/:patientId	Auto-generate invoice from unbilled completed sessions	ADMIN, RECEPTIONIST
PATCH	/api/v1/invoices/:id	Update invoice (only DRAFT status)	ADMIN
POST	/api/v1/invoices/:id/send	Send invoice to patient via email	ADMIN, RECEPTIONIST
PATCH	/api/v1/invoices/:id/cancel	Cancel invoice	ADMIN

Billing — Payments

Method	Path	Description	Auth
POST	/api/v1/payments	Record payment against an invoice	ADMIN, RECEPTIONIST
GET	/api/v1/payments/:id	Get payment details	ADMIN, RECEPTIONIST
POST	/api/v1/payments/stripe/create-intent	Create Stripe payment intent for online payment	PATIENT
POST	/api/v1/payments/stripe/webhook	Stripe webhook handler (payment confirmation)	Public (Stripe signature)
POST	/api/v1/payments/:id/refund	Refund a payment	ADMIN

Therapists

Method	Path	Description	Auth
GET	/api/v1/therapists	List therapists (filterable by specialization, active status)	ADMIN, RECEPTIONIST
GET	/api/v1/therapists/:id	Get therapist profile with schedule	ADMIN, RECEPTIONIST
POST	/api/v1/therapists	Create therapist profile (linked to user)	ADMIN
PATCH	/api/v1/therapists/:id	Update therapist profile	ADMIN, THERAPIST (own)
GET	/api/v1/therapists/:id/schedule	Get therapist weekly availability	ADMIN, RECEPTIONIST
PUT	/api/v1/therapists/:id/schedule	Set therapist weekly availability	ADMIN, THERAPIST (own)
GET	/api/v1/therapists/:id/patients	Get therapist's assigned patients	ADMIN, THERAPIST (own)

Reports

Method	Path	Description	Auth
GET	/api/v1/reports/revenue	Revenue report (by period, therapist)	ADMIN
GET	/api/v1/reports/appointments	Appointment utilization report	ADMIN
GET	/api/v1/reports/patients	Patient demographics and outcomes	ADMIN
GET	/api/v1/reports/therapists	Therapist performance (sessions, revenue, ratings)	ADMIN
GET	/api/v1/reports/outstanding	Outstanding invoices / aging report	ADMIN

Standard Response Envelope

```
// Success
{
  "success": true,
  "data": { ... },
  "meta": { "page": 1, "limit": 20, "total": 150 }
}

// Error
{
  "success": false,
  "error": {
    "code": "API_003",
    "message": "Appointment conflict: therapist already booked for this time slot",
    "details": { "conflicting_appointment_id": "uuid" }
  }
}
```

4. Authentication & Authorization

Authentication & Authorization

Authentication Flow

The system uses **JWT-based authentication** with a dual-token strategy (access + refresh tokens).

1. User submits email/password to `POST /api/v1/auth/login`
2. Server validates credentials against bcrypt hash (12 salt rounds)
3. On success, server generates:
 - **Access Token** — short-lived (15 minutes), contains user ID, role, permissions
 - **Refresh Token** — long-lived (7 days), stored in Redis with session metadata
4. Client stores access token in memory, refresh token in httpOnly cookie
5. All API requests include `Authorization: Bearer <access_token>`
6. When access token expires, client calls `POST /api/v1/auth/refresh` with refresh token

7. Server validates refresh token against Redis, issues new token pair, rotates refresh token

Token Structure

```
// Access Token Payload
{
  "sub": "user-uuid",
  "email": "therapist@clinic.com",
  "role": "THERAPIST",
  "permissions": ["patients.read", "appointments.write", "treatment-plans.write"],
  "iat": 1700000000,
  "exp": 1700000900 // 15 min
}

// Refresh Token Payload
{
  "sub": "user-uuid",
  "jti": "unique-token-id",
  "iat": 1700000000,
  "exp": 1700604800 // 7 days
}
```

Role-Based Access Control (RBAC)

Role	Permissions
ADMIN	Full access — all CRUD operations, user management, reports, settings, billing
THERAPIST	Own patients, own appointments, treatment plans (CRUD), exercises, own schedule, session notes
RECEPTIONIST	All patients (read/create), all appointments (CRUD), invoices (read/create), payments (record)
PATIENT	Own profile, own appointments (read), own treatment plans (read), own exercises (read), own invoices (read/pay)

Permission Guards

Every controller method is decorated with permission guards:

```
@UseGuards(JwtAuthGuard, RolesGuard)
@Roles('ADMIN', 'THERAPIST')
@Permissions('patients.read')
@Get('patients')
async listPatients() { ... }
```

Session Management

- Active sessions stored in Redis with key pattern: `session:{userId}:{tokenId}`
- Each user can have max 5 concurrent sessions (oldest evicted)
- Logout invalidates specific session; "Logout All" clears all sessions for user

- Session TTL matches refresh token expiry (7 days)

Password Security

- Hashing: bcrypt with 12 salt rounds
- Minimum 8 characters, must include uppercase, lowercase, number, special character
- Password reset via time-limited token (1 hour expiry) sent to email
- Last 5 passwords stored (hashed) to prevent reuse

5. Business Logic

Business Logic

Appointment Conflict Detection

When creating or rescheduling an appointment, the system checks for conflicts:

```
// Conflict query: overlapping time range for same therapist
SELECT id FROM appointments
WHERE therapist_id = :therapistId
AND status NOT IN ('CANCELLED', 'NO_SHOW')
AND start_time < :newEndTime
AND end_time > :newStartTime;

// If any rows returned → APT_003 error (conflict)
```

- Buffer time: 0 minutes between appointments (configurable in clinic_settings)
- Patient double-booking is also checked (same patient, overlapping time)
- Cancelled and No-Show appointments are excluded from conflict checks

Treatment Plan Lifecycle

```
DRAFT → ACTIVE → COMPLETED → ARCHIVED

DRAFT:    Plan created, exercises prescribed, not yet started
ACTIVE:   Treatment in progress, sessions being recorded
COMPLETED: All planned sessions done OR therapist marks complete
ARCHIVED: Historical record, no further modifications allowed
```

Rules:

- Only THERAPIST can transition plan status
- Cannot skip states (DRAFT cannot go directly to COMPLETED)
- ACTIVE → COMPLETED auto-triggers when `completed_sessions == total_sessions`
- ARCHIVED plans are read-only — no edits, no new sessions
- A patient can have only 1 ACTIVE plan per therapist at a time

Invoice Generation

Invoices can be created manually or auto-generated from completed sessions:

1. System queries all COMPLETED appointments for a patient that are not yet billed (`invoice_items.appointment_id IS NULL`)
2. For each unbilled session, creates an `invoice_item` with:
 - Description: "Physiotherapy Session — {date} — {therapist name}"
 - Unit price: $\text{therapist's hourly_rate} \times (\text{session duration} / 60)$
3. Subtotal = sum of all line items
4. Tax = subtotal $\times$ `clinic_settings.tax_rate`
5. Total = subtotal + tax
6. Invoice number auto-generated: INV-{5-digit sequence}
7. Due date = `issued_date` + 30 days (configurable)

Payment Tracking

- **Full payment:** When `payment.amount == invoice.total_amount` → invoice status = PAID
- **Partial payment:** When `sum(payments) < total_amount` → invoice status = PARTIALLY_PAID, `paid_amount` updated
- **Overpayment:** Rejected — payment amount cannot exceed remaining balance
- **Refund:** Creates negative payment record, recalculates `paid_amount` and status
- **Overdue detection:** Cron job runs daily at midnight, marks SENT invoices past `due_date` as OVERDUE

Patient Discharge Workflow

1. Therapist or Admin initiates discharge via `POST /patients/:id/discharge`
2. System validates: no ACTIVE treatment plans (must be COMPLETED or ARCHIVED first)
3. System validates: no SCHEDULED or CONFIRMED future appointments
4. System validates: no outstanding unpaid invoices (warning, not blocking)
5. Patient status → DISCHARGED
6. Discharge summary auto-generated with treatment history
7. Patient portal access remains active (read-only) for 90 days

Therapist Schedule & Availability

- Each therapist defines weekly availability: per-day start/end times, break periods
- Availability slots calculated by subtracting existing appointments from available windows
- Slot duration from `clinic_settings.appointment_duration_mins` (default 60 min)
- Therapists can block specific dates (vacation, sick leave) via schedule exceptions

Exercise Prescription

- Exercises are prescribed from the `exercise_library` to a specific `treatment_plan`
- Each prescription includes: sets, reps, `hold_seconds`, frequency, custom notes
- Patients see prescribed exercises in their portal with video demonstrations
- Therapist can update prescription parameters without removing/re-adding

6. Environment Configuration

Environment Configuration

All configuration is managed via environment variables loaded from `.env` files. The backend uses `@nestjs/config` with Joi validation — the app will not start if required variables are missing.

Core Application

Variable	Required	Default	Description
<code>NODE_ENV</code>	No	development	Environment: development, staging, production
<code>APP_PORT</code>	No	3000	API server port
<code>APP_NAME</code>	No	PhysioCRM	Application name (used in emails, logs)
<code>FRONTEND_URL</code>	Yes	—	Frontend origin for CORS and email links
<code>API_PREFIX</code>	No	/api/v1	Global API route prefix

Database

Variable	Required	Default	Description
<code>DATABASE_URL</code>	Yes	—	PostgreSQL connection string: <code>postgresql://user:pass@host:5432/physiocrm</code>
<code>MONGODB_URI</code>	Yes	—	MongoDB connection string: <code>mongodb://host:27017/physiocrm_audit</code>
<code>REDIS_URL</code>	Yes	—	Redis connection string: <code>redis://host:6379/0</code>

Authentication

Variable	Required	Default	Description
<code>JWT_ACCESS_SECRET</code>	Yes	—	Secret for signing access tokens (min 32 chars)
<code>JWT_REFRESH_SECRET</code>	Yes	—	Secret for signing refresh tokens (min 32 chars)
<code>JWT_ACCESS_EXPIRY</code>	No	15m	Access token TTL
<code>JWT_REFRESH_EXPIRY</code>	No	7d	Refresh token TTL

Email (SMTP)

Variable	Required	Default	Description
<code>SMTP_HOST</code>	Yes*	—	SMTP server hostname
<code>SMTP_PORT</code>	No	587	SMTP port (587 for TLS, 465 for SSL)
<code>SMTP_USER</code>	Yes*	—	SMTP authentication username

Variable	Required	Default	Description
SMTP_PASS	Yes*	—	SMTP authentication password
SMTP_FROM_NAME	No	PhysioCRM	Sender display name
SMTP_FROM_EMAIL	No	noreply@physiocrm.com	Sender email address

* Required in production. In development, emails are logged to console.

Google Calendar Integration

Variable	Required	Default	Description
GOOGLE_CLIENT_ID	No	—	Google OAuth2 client ID
GOOGLE_CLIENT_SECRET	No	—	Google OAuth2 client secret
GOOGLE_REDIRECT_URI	No	—	OAuth2 callback URL

SMS Gateway

Variable	Required	Default	Description
SMS_PROVIDER	No	twilio	SMS provider: twilio, messagebird
SMS_ACCOUNT_SID	No	—	Twilio Account SID
SMS_AUTH_TOKEN	No	—	Twilio Auth Token
SMS_FROM_NUMBER	No	—	Sender phone number

Stripe Payments

Variable	Required	Default	Description
STRIPE_SECRET_KEY	No	—	Stripe secret API key
STRIPE_WEBHOOK_SECRET	No	—	Stripe webhook signing secret

File Storage

Variable	Required	Default	Description
UPLOAD_DIR	No	./uploads	Local file storage directory
MAX_FILE_SIZE_MB	No	10	Maximum upload file size in MB

Example .env File

```
# Core
NODE_ENV=production
APP_PORT=3000
FRONTEND_URL=https://physiocrm.example.com

# Database
DATABASE_URL=postgresql://physiocrm:<password>@localhost:5432/physiocrm
MONGODB_URI=mongodb://localhost:27017/physiocrm_audit
REDIS_URL=redis://localhost:6379/0

# Auth
JWT_ACCESS_SECRET=<access_secret>
JWT_REFRESH_SECRET=<refresh_secret>

# Email
SMTP_HOST=smtp.example.com
SMTP_PORT=587
SMTP_USER=<smtp_user>
SMTP_PASS=<smtp_password>

# Google Calendar (optional)
GOOGLE_CLIENT_ID=<google_client_id>
GOOGLE_CLIENT_SECRET=<google_client_secret>

# Stripe (optional)
STRIPE_SECRET_KEY=<stripe_key>
STRIPE_WEBHOOK_SECRET=<stripe_webhook_secret>
```

7. Deployment Guide

Deployment Guide

Prerequisites

- Ubuntu 22.04 LTS (or compatible Linux)
- Node.js 18+ (via nvm recommended)
- PostgreSQL 15
- MongoDB 7
- Redis 7
- Nginx 1.25+
- PM2 (installed globally: `npm install -g pm2`)

Step-by-Step Deployment

1. Clone Repository

```
git clone https://github.com/org/physiocrm.git /opt/physiocrm
```

```
cd /opt/physiocrm
```

2. Install Dependencies

```
# Backend
cd backend
npm ci --production

# Frontend
cd ../frontend
npm ci
```

3. Configure Environment

```
cp backend/.env.example backend/.env
# Edit backend/.env with production values (see Environment Configuration section)
nano backend/.env
```

4. Database Setup

```
# Run Prisma migrations
cd backend
npx prisma db push

# Verify schema
npx prisma db pull
```

5. Seed Initial Data

```
# Seeds: admin user, default roles (ADMIN, THERAPIST, RECEPTIONIST, PATIENT),
# clinic settings defaults, exercise library (50+ common exercises)
npx prisma db seed
```

Default admin credentials after seeding:

- Email: `admin@physiocrm.com`
- Password: `Admin@123` (must be changed on first login)

6. Build Frontend

```
cd frontend
echo "VITE_API_URL=https://physiocrm.example.com/api/v1" > .env.production
npm run build
# Output: frontend/dist/
```

7. Configure Nginx

```

server {
    listen 443 ssl http2;
    server_name physiocrm.example.com;

    ssl_certificate      /etc/ssl/certs/physiocrm.crt;
    ssl_certificate_key  /etc/ssl/private/physiocrm.key;

    # Frontend static files
    root /opt/physiocrm/frontend/dist;
    index index.html;

    # SPA fallback
    location / {
        try_files $uri $uri/ /index.html;
    }

    # API proxy
    location /api/ {
        proxy_pass http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # File uploads
    location /uploads/ {
        alias /opt/physiocrm/backend/uploads/;
        expires 30d;
        add_header Cache-Control "public, immutable";
    }

    # Security headers
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
}

```

8. Start with PM2

```

cd /opt/physiocrm/backend

# Start in cluster mode (2 instances)
pm2 start dist/main.js --name physiocrm-api -i 2

```

```
# Save PM2 process list
pm2 save

# Enable PM2 startup on boot
pm2 startup systemd
```

9. Verify Deployment

```
# Health check endpoint
curl -s https://physiocrm.example.com/api/v1/health | jq

# Expected response:
{
  "status": "ok",
  "uptime": 12.5,
  "postgresql": "connected",
  "mongodb": "connected",
  "redis": "connected",
  "version": "1.0.0"
}
```

Log Locations

Log	Location
Application logs	~/.pm2/logs/physiocrm-api-out.log
Error logs	~/.pm2/logs/physiocrm-api-error.log
Nginx access	/var/log/nginx/access.log
Nginx error	/var/log/nginx/error.log
Audit logs	MongoDB: physiocrm_audit.audit_logs collection

Common Operations

```
# Restart API
pm2 restart physiocrm-api

# View logs (live)
pm2 logs physiocrm-api

# Zero-downtime reload
pm2 reload physiocrm-api

# Run database migration after update
cd /opt/physiocrm/backend && npx prisma db push
```

8. Integrations

Integrations

Google Calendar — Appointment Sync

Therapists can link their Google Calendar to automatically sync appointments.

Aspect	Detail
Auth Method	OAuth2 with offline access (refresh tokens)
Scope	<code>https://www.googleapis.com/auth/calendar.events</code>
Sync Direction	One-way: PhysioCRM → Google Calendar
Trigger	On appointment create, update, cancel
Event Data	Title: "Physio: {Patient Name}", time, therapist as attendee

Flow:

1. Therapist clicks "Connect Google Calendar" in profile settings
2. OAuth2 consent screen → user grants calendar access
3. System stores encrypted refresh token per therapist
4. On appointment CRUD, BullMQ job queues calendar sync
5. Worker creates/updates/deletes Google Calendar event
6. `google_event_id` stored on appointment record for future updates

SMS Gateway — Appointment Reminders

Aspect	Detail
Provider	Twilio (primary), MessageBird (fallback)
Trigger	Cron job runs every hour, checks appointments in next 24 hours
Message	"Hi {firstName}, reminder: your physiotherapy appointment is tomorrow at {time} with {therapistName}. Reply CANCEL to cancel."
Opt-out	Patients can disable SMS in portal settings
Logging	All SMS attempts logged to MongoDB with status

Email — SMTP (Nodemailer)

Email Type	Trigger	Recipients
Appointment Confirmation	Appointment created	Patient
Appointment Reminder	24 hours before	Patient

Email Type	Trigger	Recipients
Appointment Cancellation	Appointment cancelled	Patient + Therapist
Invoice Sent	Invoice status → SENT	Patient
Payment Confirmation	Payment recorded	Patient
Password Reset	Forgot password	User
Welcome Email	Portal account created	Patient
Treatment Plan Shared	Plan shared	Patient

All emails use branded HTML templates with `{{variable}}` placeholders stored in `backend/src/templates/email/`.

Stripe — Online Payments

Aspect	Detail
Integration	Stripe Payment Intents API
Flow	Patient views invoice → "Pay Online" → Stripe Checkout → webhook confirms
Webhook Events	<code>payment_intent.succeeded</code> , <code>payment_intent.payment_failed</code>
Refunds	Admin panel → Stripe Refunds API
Security	Webhook signature verification, no card data stored locally

Exercise Video Hosting

Aspect	Detail
Platforms	YouTube (unlisted), Vimeo (private links)
Storage	External hosting; embed URLs in <code>exercise_library.video_url</code>
Display	Embedded iframe in exercise detail view
Fallback	Static image if no video available
Privacy	YouTube unlisted / Vimeo domain-restricted

9. Error Handling & Logging

Error Handling & Logging

Global Exception Filter

All exceptions are caught by a global NestJS exception filter returning a standardized error envelope:

```
{
  "success": false,
  "error": {
    "code": "APT_003",
    "message": "Appointment conflict detected",
    "details": { ... },
    "timestamp": "2024-01-15T10:30:00.000Z",
    "path": "/api/v1/appointments",
    "correlationId": "req-abc123-def456"
  }
}
```

HTTP status codes: 400 (validation), 401 (auth), 403 (forbidden), 404 (not found), 409 (conflict), 422 (business rule), 500 (internal).

Error Code Registry

Code Range	Module	Examples
AUTH_001 - AUTH_010	Authentication	AUTH_001: Invalid credentials, AUTH_002: Token expired, AUTH_003: Insufficient permissions, AUTH_004: Account disabled, AUTH_005: Refresh token invalid
PAT_001 - PAT_020	Patients	PAT_001: Patient not found, PAT_002: Duplicate email, PAT_003: Cannot discharge (active plans), PAT_004: Invalid status transition
APT_001 - APT_015	Appointments	APT_001: Not found, APT_002: Invalid time range, APT_003: Therapist conflict, APT_004: Patient conflict, APT_005: Past date, APT_006: Outside working hours
TRT_001 - TRT_010	Treatment Plans	TRT_001: Not found, TRT_002: Invalid status transition, TRT_003: Active plan exists, TRT_004: Archived (read-only)
BIL_001 - BIL_010	Billing	BIL_001: Invoice not found, BIL_002: Cannot edit non-draft, BIL_003: Payment exceeds balance, BIL_004: Already cancelled
SYS_001 - SYS_010	System	SYS_001: DB connection failed, SYS_002: Redis unavailable, SYS_003: File upload failed, SYS_005: Rate limit exceeded

Audit Logging (MongoDB)

```
// audit_logs collection
{
  _id: ObjectId,
  correlationId: "req-abc123",
  userId: "user-uuid",
  userEmail: "therapist@clinic.com",
  action: "UPDATE", // CREATE, READ, UPDATE, DELETE
  entity: "appointment",
  entityId: "appointment-uuid",
  changes: {
    before: { status: "SCHEDULED" },
  }
}
```

```
    after: { status: "COMPLETED" }
  },
  ipAddress: "192.168.1.100",
  userAgent: "Mozilla/5.0...",
  timestamp: ISODate("2024-01-15T10:30:00Z")
}
```

Retention: 2 years via MongoDB TTL index. Automated cleanup.

Structured Application Logging

- JSON structured logs with correlation ID, timestamp, level, context
- Levels: ERROR, WARN, LOG, DEBUG, VERBOSE
- Production: ERROR + WARN only (configurable via LOG_LEVEL)
- Correlation ID generated per request, propagated through all service calls

Rate Limiting

Endpoint	Limit	Window
POST /auth/login	5 attempts	15 min (per IP)
POST /auth/forgot-password	3 attempts	1 hour (per email)
POST /payments/stripe/webhook	100 requests	1 min (per IP)
All authenticated endpoints	100 requests	1 min (per user)

Implemented via @nestjs/throttler with Redis backing store.

10. Testing Strategy

Testing Strategy

Testing Pyramid

Layer	Tool	Scope	Coverage Target
Unit Tests	Jest	Services, utilities, guards, pipes	80%+ line coverage
Integration Tests	Jest + Prisma (test DB)	Repository layer, database queries	70%+ line coverage
E2E Tests	Jest + Supertest	Full API endpoint testing	All critical paths

Test Database

```
# Separate test database – never run tests against production
DATABASE_URL=postgresql://physiocrm_test:<password>@localhost:5432/physiocrm_test
```

```
# Before each test suite:
npx prisma db push --force-reset # Reset schema
npx prisma db seed # Seed test data
```

Test Data Seeding

The test seed creates:

- 1 Admin user, 2 Therapist users, 1 Receptionist user, 3 Patient users
- 5 Patients with varied statuses (ACTIVE, DISCHARGED)
- 2 Therapists with different specializations and schedules
- 10 Appointments across various statuses
- 2 Treatment plans (1 ACTIVE, 1 COMPLETED)
- 10 Exercises in the library
- 3 Invoices (DRAFT, SENT, PAID)
- Clinic settings with working hours

Key Test Scenarios

Appointment Conflict Detection

Scenario	Expected Result
Create appointment overlapping existing for same therapist	409 Conflict — APT_003
Create appointment overlapping existing for same patient	409 Conflict — APT_004
Create appointment overlapping CANCELLED appointment	201 Created (cancelled excluded)
Create appointment adjacent (no overlap) to existing	201 Created
Reschedule to conflicting time	409 Conflict — APT_003

Invoice Calculation

Scenario	Expected Result
Generate invoice from 3 completed sessions (60 min each, \$100/hr)	Subtotal: \$300, tax applied per clinic rate
Generate invoice with no unbilled sessions	422 — BIL_005
Record full payment on SENT invoice	Invoice status → PAID
Record partial payment	Invoice status → PARTIALLY_PAID, paid_amount updated
Record payment exceeding balance	422 — BIL_003

Treatment Plan State Transitions

Scenario	Expected Result
DRAFT → ACTIVE	200 OK
DRAFT → COMPLETED (skip)	422 — TRT_002
ACTIVE → COMPLETED (all sessions done)	200 OK
COMPLETED → ARCHIVED	200 OK
ARCHIVED → any (edit attempt)	422 — TRT_004
Create 2nd ACTIVE plan for same patient+therapist	409 — TRT_003

Auth Flow

Scenario	Expected Result
Login with valid credentials	200 — access + refresh tokens
Login with wrong password	401 — AUTH_001
Access protected route without token	401 — AUTH_002
Access admin route as THERAPIST	403 — AUTH_003
Refresh with valid refresh token	200 — new token pair
Refresh with expired token	401 — AUTH_005
Login with disabled account	401 — AUTH_004

Running Tests

```
# Unit tests
npm run test

# Unit tests with coverage
npm run test:cov

# E2E tests
npm run test:e2e

# Watch mode (development)
npm run test:watch
```

11. Changelog

Changelog

v1.0.0 — Initial Release

Release Date: 2026-05-03

First production release of the Physiotherapy CRM platform. All core modules implemented and tested.

Modules Included

Module	Features
Patient Management	Patient CRUD, medical history, insurance details, status management (ACTIVE/DISCHARGED/INACTIVE), discharge workflow with validation, patient code auto-generation (PAT-XXXXX)
Appointment Scheduling	Calendar-based scheduling, conflict detection (therapist + patient), status lifecycle (SCHEDULED → CONFIRMED → IN_PROGRESS → COMPLETED), availability slot calculation, Google Calendar sync, appointment types (Initial Assessment, Follow-up, Treatment, Review)
Treatment Plans	Plan creation with diagnosis and goals, session tracking with SOAP notes, pain level recording (0-10 scale), status lifecycle (DRAFT → ACTIVE → COMPLETED → ARCHIVED), one active plan per patient-therapist constraint
Exercise Library	Curated exercise database with body region tagging, difficulty levels (Beginner/Intermediate/Advanced), YouTube/Vimeo video embedding, exercise prescription to treatment plans with sets/ reps/frequency
Billing & Invoicing	Auto-invoice generation from completed sessions, manual invoice creation, line items with appointment linking, partial payment support, Stripe online payments, invoice status tracking (DRAFT → SENT → PAID/OVERDUE), overdue detection cron job
Therapist Management	Therapist profiles with specializations and license numbers, weekly availability schedule, schedule exceptions (vacation/sick leave), hourly rate configuration, workload tracking
Reports & Analytics	Revenue reports (by period, by therapist), appointment utilization metrics, patient demographics and outcomes, therapist performance dashboards, outstanding invoice aging report
Patient Portal	Self-service patient login, view upcoming appointments, access treatment plans and prescribed exercises (with videos), view and pay invoices online, update profile information

Infrastructure

- JWT authentication with access/refresh token strategy
- Role-based access control (ADMIN, THERAPIST, RECEPTIONIST, PATIENT)
- PostgreSQL 15 for business data with Prisma ORM
- MongoDB 7 for audit logging
- Redis 7 for sessions, caching, and BullMQ job queues
- Nginx reverse proxy with SSL termination
- PM2 process management with cluster mode
- Global exception filter with structured error codes
- Rate limiting on authentication endpoints
- SMTP email integration (appointment confirmations, invoice delivery, password reset)
- SMS reminders via Twilio (24-hour appointment reminders)
- Stripe payment integration

- Google Calendar sync for therapists

Known Limitations (v1.0.0)

- Google Calendar sync is one-way only (PhysioCRM → Google)
- File storage is local filesystem (S3 migration planned for v1.1)
- Reports are server-rendered (client-side charting planned for v1.1)
- SMS provider limited to Twilio (MessageBird support planned)
- No multi-clinic / multi-tenant support (single clinic only)