EYE CLINIC MANAGEMENT SYSTEM

The Refraction Eye Prescription Dashboard

A comprehensive eye prescription management dashboard for optometrists to create, view, edit, and share patient prescription records.

Table of Contents

Overview

The Refraction is a part of a whole web-based eye clinic application designed for optometrists and eye care professionals to manage patient eye prescription data efficiently. It enables healthcare providers to store, analyze, and share prescription information with patients and optometrists while maintaining access control and data security.

Features

- **Patient Management**
  - Create, view, edit, and delete patient records
  - Store comprehensive eye prescription data including:
    - Spherical power (SPH)

        - Cylindrical power (CYL)
        - Axis
        - Addition power (ADD)
        - Pupillary distance (PD) single or dual measurements
    - Track demographics and visit history
    - Filter and sort patient lists
- **Data Import**
    - Parse prescription data from text files
    - Import patient information via uploads
    - Automatic extraction of prescription values
- **Prescription Communication**
    - Share prescriptions via email (Mailjet) and SMS (MessageBird)
    - Professional HTML-formatted emails
- **Doctor's Interface**
    - Secure PIN-based access
    - Clinical review of prescriptions
    - Add professional notes and recommendations
    - Digital signature for prescriptions
- **Security**
    - User authentication and role-based access
    - PIN verification for sensitive operations
    - Secure session management

Tech Stack

Frontend

- React (TypeScript)
- TanStack Query (Data fetching)
- Shadcn UI components (UI)
- Tailwind CSS (Styling)

- React (Form management and validation)

Backend

- Node.js with Express (Routing)
- Flask (Parsing and LLM integration)
- PostgreSQL (RDS)
- Passport.js (Authentication)
- RESTful API architecture (Frontend-Backend communication)

External Services

- Mailjet (Email delivery)
- MessageBird (SMS delivery)

Installation

Prerequisites

- Node.js (v16+)
- PostgreSQL database

Setup Steps

1. **Clone the repository**
2. git clone https://github.com/your-repo/the-refraction.git
3. cd the-refraction
4. **Install dependencies**
5. npm install
6. **Setup environment variables**
   Create a .env file with:
7. DATABASE_URL=postgresql://username:password@localhost:5432/refraction
8. MAILJET_API_KEY=your_mailjet_api_key
9. MAILJET_SECRET_KEY=your_mailjet_secret_key
10. MESSAGEBIRD_API_KEY=your_messagebird_api_key
11. **Initialize the database**
12. npm run db:push

13. **Start the development server**
14. npm run dev

Configuration

Email (Mailjet)

- Sign up at Mailjet
- Add MAILJET_API_KEY and MAILJET_SECRET_KEY to .env

SMS (MessageBird)

- Sign up at MessageBird
- Add MESSAGEBIRD_API_KEY to .env

Security PIN

- Default PIN: 1234
- Changeable through the application's security settings.

Usage

Authentication

- **Default Admin Credentials:**
  - Username: admin
  - Password: admin

Patient Management

- **Create** new patient records with full prescription details.
- **View** patients and search/filter the patient list.
- **Edit** patient details and prescriptions.
- **Delete** records (PIN verification required).

Doctor's Review

- Access secured by PIN.
- View prescriptions in clinical format.
- Add notes, recommendations, and sign prescriptions.
- Update status from pending to reviewed.

Communication

- **Email** prescriptions:
    - Send HTML-formatted emails.
- **SMS** prescriptions:
    - Send summarized prescriptions as text.

Data Import

- Upload text files containing prescriptions.
- Auto-parse and extract patient information.

API Endpoints

Authentication

- POST /api/login — Login
- POST /api/logout — Logout
- GET /api/user — Get current user info

Patients

- GET /api/patients — Get all patients
- GET /api/patients/:id — Get patient by ID
- POST /api/patients — Create patient
- PATCH /api/patients/:id — Update patient
- PATCH /api/patients/:id/status — Update patient status
- DELETE /api/patients/:id — Delete patient

Security

- GET /api/security-settings — Get security settings
- PATCH /api/security-settings — Update settings
- POST /api/verify-doctor — Verify doctor PIN
- GET /api/doctor-status — Check doctor status

Communication

- POST /api/email/send — Send prescription via email

Database Schema

Users

- id (Primary key)
- username
- password (hashed)

Patients

- id (Primary key)
- name
- phone
- email
- location
- dob (Date of birth)
- age
- examDate
- rightEye (JSON object: sph, cyl, axis, add)
- leftEye (JSON object: sph, cyl, axis, add)
- pdType ("single" or "dual")
- pd, pdOd, pdOs
- status ("pending" or "reviewed")
- createdAt: Date
- updatedAt: Date

Security Settings

- id (Primary key)
- security Pin (String)

Security

The application uses a session-based authentication system with Passport.js. Passwords are securely hashed using scrypt.

Authentication

- Session-based authentication using Passport.js
- Passwords securely hashed (scrypt)

Authorization

- Basic access: View records
- PIN verification: Required for doctor's operations

Data Protection

- Secure database connections
- Hashed passwords
- Session timeout policies
- PIN required for sensitive operations

Communication Features

Email (Mailjet)

- Professional HTML email templates
- Full prescription details
- Optional doctor's notes and signature
- Error handling and delivery confirmation

SMS (MessageBird)

- Short, patient-friendly summaries
- Supports international numbers
- Fallback simulations for development

Doctor's Review Interface

A secure, PIN-protected view allowing doctors to:

- Review prescriptions in a clinical format
- Add recommendations and personal notes
- Sign and update prescription statuses

Troubleshooting

Email Issues

- Verify Mailjet API keys
- Check recipient email format
- Review server logs

SMS Issues

- Verify MessageBird API key is correctly set.
- Ensure phone numbers have the correct format (country code).
- Check server logs for MessageBird error messages.

Database Issues

- Verify database (PostgreSQL) connection settings
- Ensure schema is initialized
- Check logs for errors

Authentication Issues

- Use default credentials.
- Check server logs if login fails
- Clear cookies if session issues occur