Your Health Guide (دليلك الصحي)

A Progressive Web App (PWA) designed to transform healthcare management in Egypt by providing an intelligent, accessible digital health companion for patients with chronic conditions.

Features

- **AI-Powered Prescription Scanner**: Convert handwritten prescriptions to digital records with 95%+ accuracy
- **Smart Vitals Tracker**: Log and visualize health metrics over time
- **Voice-First Interface**: Egyptian Arabic voice command support
- **Offline-First Design**: Works without internet connectivity
- **Health Report Generator**: PDF reports for doctor visits
- **Emergency SOS System**: One-tap emergency alerts

Technology Stack

Backend

- **Framework**: Django with Django REST Framework
- **Database**: PostgreSQL with row-level security
- **Authentication**: JWT + OAuth 2.0
- **Cache**: Redis for session management
- **AI/ML**: Google Cloud Vision API, MedGemma via Vertex AI

Frontend

- **Framework**: Vanilla JavaScript (ES6+) with Vite
- **Styling**: Tailwind CSS with DaisyUI component library
- **PWA Features**: Service Workers, IndexedDB for offline support
- **Architecture**: Modular component-based structure

PROF

Quick Start

Prerequisites

- Python 3.10+
- Node.js 16+
- PostgreSQL (for production)
- Docker (optional)

Backend Setup

```
# Navigate to backend directory
cd backend
```

```
# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# .\venv\Scripts\activate  # Windows

# Install dependencies
pip install -r requirements.txt

# Run migrations
python manage.py makemigrations
python manage.py migrate

# Create superuser (optional)
python manage.py createsuperuser

# Run development server
python manage.py runserver
```

Frontend Setup

```
# Navigate to frontend directory
cd frontend

# Install dependencies
npm install

# Run development server with Vite
npm run dev

# Or build for production
npm run build
npm run preview
```

PROF

Full Stack Development

1. **Start Backend** (Terminal 1):

```
cd backend
source venv/bin/activate
python manage.py runserver
```

2. **Start Frontend** (Terminal 2):

```
cd frontend
npm run dev
```

3. **Access the Application**:
    - Frontend: http://localhost:5173 (Vite dev server)
    - Backend API: http://localhost:8000
    - Admin Panel: http://localhost:8000/admin

Docker Setup (Alternative)

```
# Build and run with Docker Compose
docker-compose up --build

# Access the application
# Frontend: http://localhost:8080
# Backend API: http://localhost:8000
# Admin Panel: http://localhost:8000/admin
```

Project Structure

```
your-health-guide/
├── backend/                  # Django REST API
│   ├── health_guide/        # Main Django project
│   │   ├── settings/        # Environment-specific settings
│   │   └── ...
│   ├── apps/                # Django applications
│   │   ├── authentication/ # User auth & JWT
│   │   ├── prescriptions/  # Prescription scanning
│   │   ├── vitals/          # Health metrics tracking
│   │   ├── reports/         # PDF generation
│   │   └── emergency/       # SOS functionality
│   └── requirements.txt
├── frontend/                # PWA frontend
│   ├── public/             # Static HTML files
│   │   ├── index.html      # Landing page
│   │   ├── dashboard.html # Main app interface
│   │   └── manifest.json  # PWA manifest
│   └── src/
│       ├── styles/         # Tailwind CSS
│       └── scripts/        # JavaScript modules
├── docs/                    # Documentation
├── docker-compose.yml      # Local development
├── Dockerfile             # Container configuration
└── README.md
```

API Endpoints

Authentication

- `POST /api/v1/auth/register/` - User registration

- `POST /api/v1/auth/login/` - User login
- `POST /api/v1/auth/token/refresh/` - Refresh JWT token
- `GET /api/v1/auth/profile/` - Get user profile

Prescriptions

- `GET /api/v1/prescriptions/` - List prescriptions
- `POST /api/v1/prescriptions/` - Upload prescription
- `GET /api/v1/prescriptions/{id}/` - Get prescription details

Vitals

- `GET /api/v1/vitals/` - List vital readings
- `POST /api/v1/vitals/` - Add vital reading
- `GET /api/v1/vitals/?type=blood_pressure` - Filter by type

Reports

- `GET /api/v1/reports/` - List health reports
- `POST /api/v1/reports/` - Generate report

Emergency

- `GET /api/v1/emergency/contacts/` - List emergency contacts
- `POST /api/v1/emergency/alert/` - Send emergency alert

Development

Backend Commands

```
# Run tests
python manage.py test

# Create new migration
python manage.py makemigrations

# Apply migrations
python manage.py migrate

# Create superuser
python manage.py createsuperuser

# Collect static files
python manage.py collectstatic
```

PROF

Frontend Commands

```
```

```
# Build CSS for development (with watch)
npm run build-css

# Build CSS for production (minified)
npm run build

# Serve frontend locally
npm run serve
```

Environment Variables

Create a `.env` file in the backend directory:

```
SECRET_KEY=your-secret-key-here
DEBUG=True
DATABASE_URL=postgresql://user:password@localhost:5432/health_guide
REDIS_URL=redis://localhost:6379/1
GOOGLE_CLOUD_PROJECT=your-gcp-project-id
TWILIO_ACCOUNT_SID=your-twilio-sid
TWILIO_AUTH_TOKEN=your-twilio-token
ALLOWED_HOSTS=localhost,127.0.0.1
CORS_ALLOWED_ORIGINS=http://localhost:8080,http://127.0.0.1:8080
```

Deployment

Google Cloud Platform

```
# Deploy to Cloud Run
gcloud run deploy --source . --platform managed --region us-central1

# Set up Cloud SQL database
gcloud sql instances create health-guide-db --database-
version=POSTGRES_13

# Configure environment variables
gcloud run services update health-guide --set-env-
vars="DATABASE_URL=..."
```

PROF

Contributing

1. Fork the repository
2. Create a feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Support

For support, email support@yourhealthguide.com or join our Slack channel.

Acknowledgments

- Google Cloud Platform for AI/ML services
- Tailwind CSS and DaisyUI for the beautiful UI
- Django and Django REST Framework for the robust backend
- The open-source community for the amazing tools and libraries