Dermatology Management System

A full-featured clinical and dermatology management system designed to digitize and streamline clinic operations, including patient management, appointments, diagnosis, inventory, billing, and reporting.

Overview

This project provides a complete solution for managing a dermatology clinic with clear role separation between Doctors and Receptionists.
It follows real clinic workflows where receptionists handle bookings and payments, while doctors handle diagnosis and treatment.

Technology Stack

- Backend: Django + Django REST Framework
- Frontend: React Native (Expo)
- Database: SQLite

System Architecture

The system follows a client–server architecture:

- Backend (Django):
    - Business logic and data validation
    - Authentication and authorization
    - Database management
    - Reporting and analytics
    - REST API endpoints
- Frontend (React Native):
    - Doctor and Receptionist dashboards
    - Patient and appointment management
    - Inventory and billing UI
    - Data visualization and reports

All communication is done using JSON over HTTP via APIs.

Project Structure

Backend

```
backend-1/
├── clinic/
│   ├── models.py
│   ├── serializers.py
│   ├── views.py
│   ├── urls.py
│   └── migrations/
├── clinical_backend/
├── db.sqlite3
├── requirements.txt
└── seed_data.py
```

Frontend

```
DermaSkincareApp/
├── app/
├── components/
├── context/
├── doctor/
├── receptionist/
├── assets/
├── hooks/
├── src/api/
└── types/
```

Backend Features (Django)

Authentication

- Login for doctors and receptionists
- Role-based access control

Patient Management

- Add, edit, and view patients
- Store personal and medical information
- Track allergies, conditions, and surgeries

Appointment Management

- Book appointments
- Status tracking: booked, checked-in
- Link appointments to doctors and services

Service Management

- Laser and diagnosis services
- Price, duration, and category support

Medical Records & Diagnosis

- Store diagnosis notes per visit
- Link records to appointments
- Maintain full patient history

Medication & Prescriptions

- Medication management
- Prescriptions linked to medical records
- Track dosage and duration

Treatment Plans & Sessions

- Create treatment plans
- Track individual treatment sessions
- Suitable for laser and beauty services

Inventory Management

- Inventory items with expiry dates and minimum stock
- Stock transactions: add, use, expired
- Low stock and expiring item alerts

Billing & Invoices

- Generate invoices per appointment
- Track total amount, discounts, and payment method
- Store payment status

Patient Files

- Upload and store documents, photos, and lab results

Reports & Analytics

- Appointments statistics
- Inventory usage
- Revenue and patient analytics

Backend Models

Patient
Doctor
Receptionist
Service
Appointment
AppointmentService
Invoice
InvoiceItem
MedicalRecord
Medication
Prescription
PrescriptionMedication
TreatmentPlan
TreatmentSession
Allergy
PatientAllergy
MedicalCondition
PatientMedicalCondition
SurgeryType
PatientSurgery
InventoryItem
StockTransaction
PatientFile
LaserSession

See `backend-1/clinic/models.py` for full field definitions.

API Endpoints

Core

```
/login/
/patients/
/doctors/
/receptionists/
/services/
/appointments/
```

Medical & Treatment

```
/medical-records/
/medications/
/prescriptions/
/treatment-plans/
/treatment-sessions/
```

Medical History

```
/allergies/
/patient-allergies/
/medical-conditions/
/patient-medical-conditions/
/surgery-types/
/patient-surgeries/
```

Inventory & Billing

```
/inventory/
/stock-transactions/
/invoices/
```

Reports

```
/reports/analytics/
/reports/appointments/
/reports/inventory/
/reports/patients/
```

Backend Setup

```
cd backend-1
pip install -r requirements.txt
python manage.py migrate
python manage.py runserver
```

Optional demo data:

```
python seed_data.py
```

Backend runs at:

```
http://localhost:8000/
```

Frontend Features (React Native / Expo)

Authentication

- Login screen
- Context-based authentication state

Receptionist Screens

- Dashboard with appointment statistics and calendar
- Add, edit, and manage patients
- Book and manage appointments
- Invoice creation and payment tracking
- Inventory management with alerts

Doctor Screens

- Dashboard with today's sessions
- Patient history and visit tracking
- Diagnosis and prescription management
- Inventory usage
- Reports and analytics

Shared Screens

- Home tab navigation
- Role-based layouts
- Patient quick view modal

Key Components

- DiagnosisTemplatePills
- PatientInfoBar
- PrescriptionTable
- ReusablePhotoUploader
- ServiceTabs
- InventoryTable
- InventoryCharts
- Sidebar and DoctorSidebar

Backend–Frontend Integration

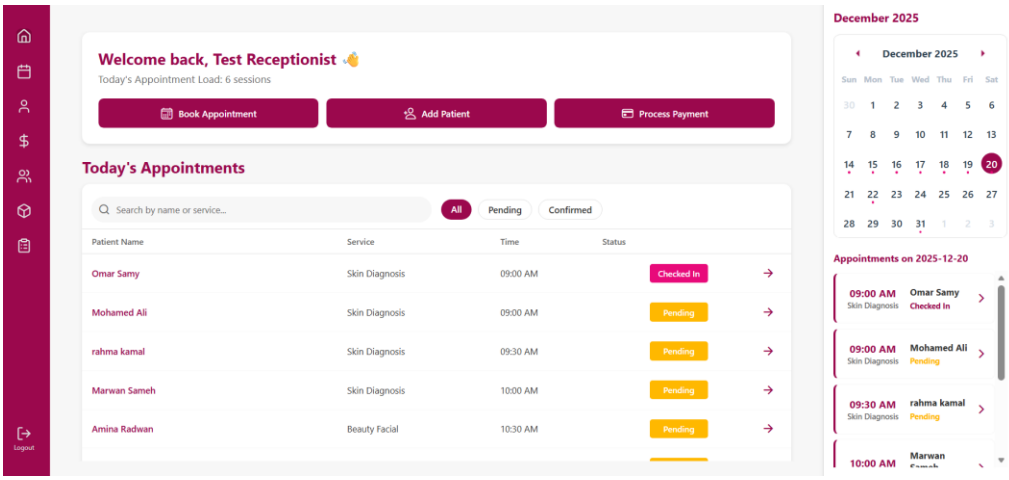
- API communication
- JSON data exchange
- Frontend performs all operations through backend endpoints
- Backend must be running before using the frontend

Frontend Setup

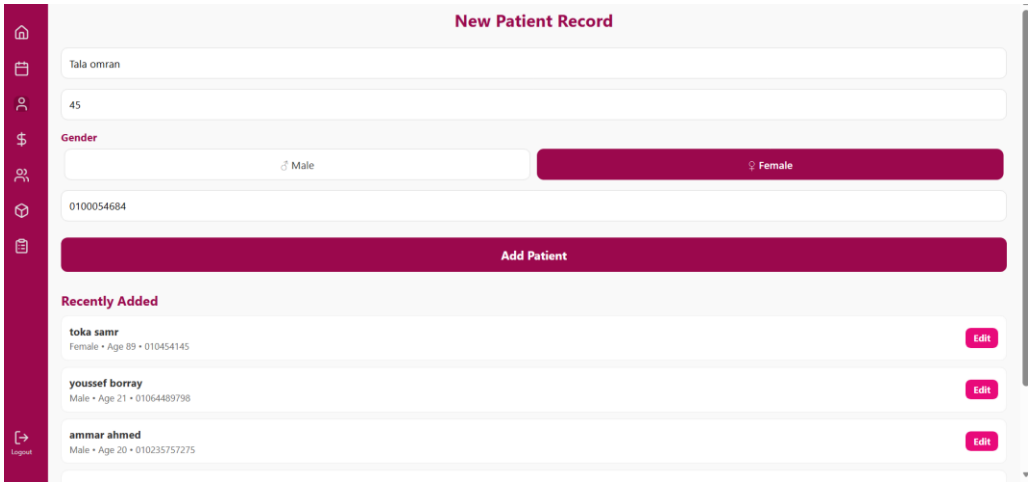
```
cd DermaSkincareApp
npm install
npx expo start
```

Screenshots

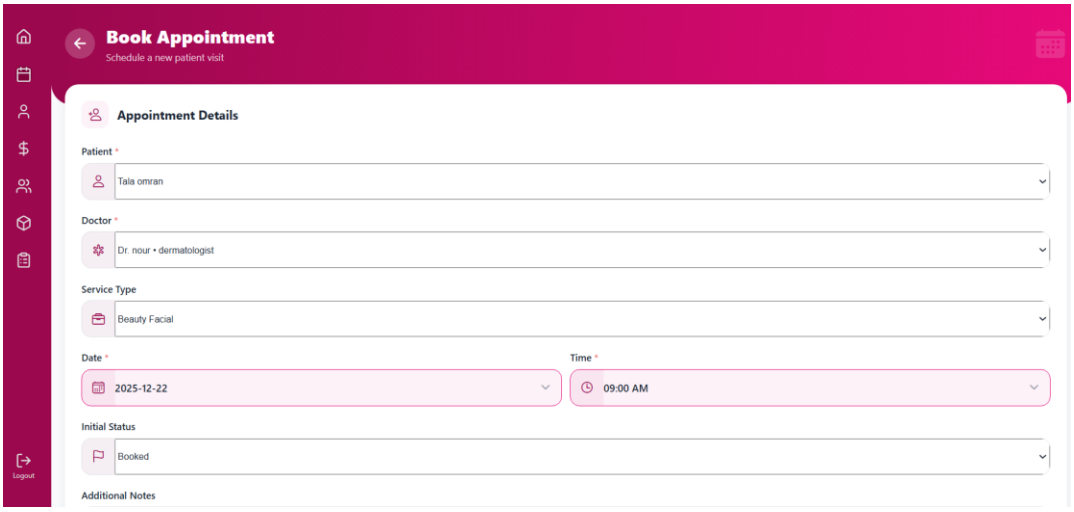
Receptionist Dashboard



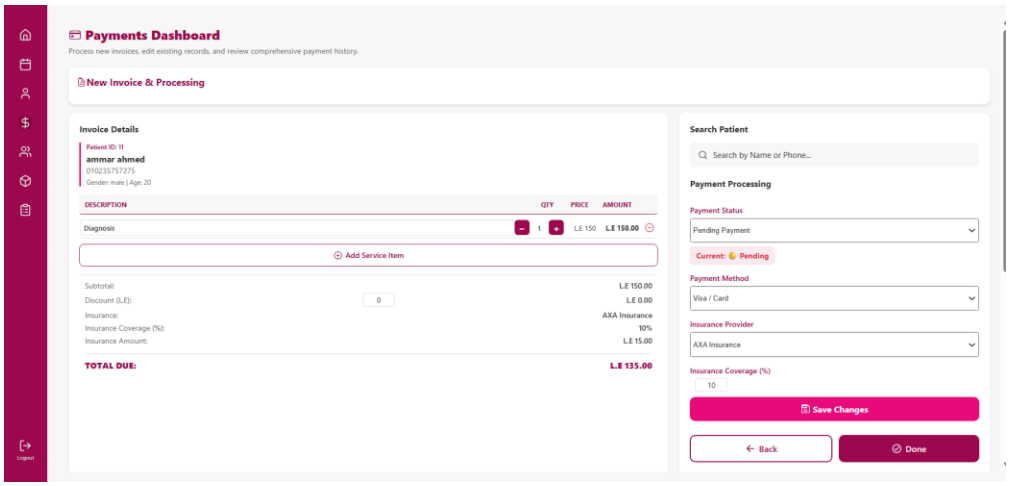
New Patient



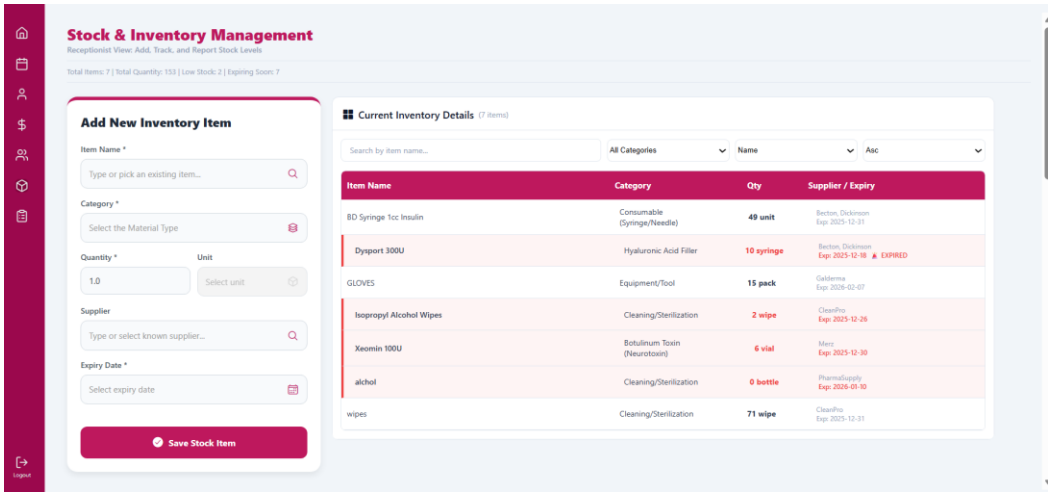
Appointment Booking



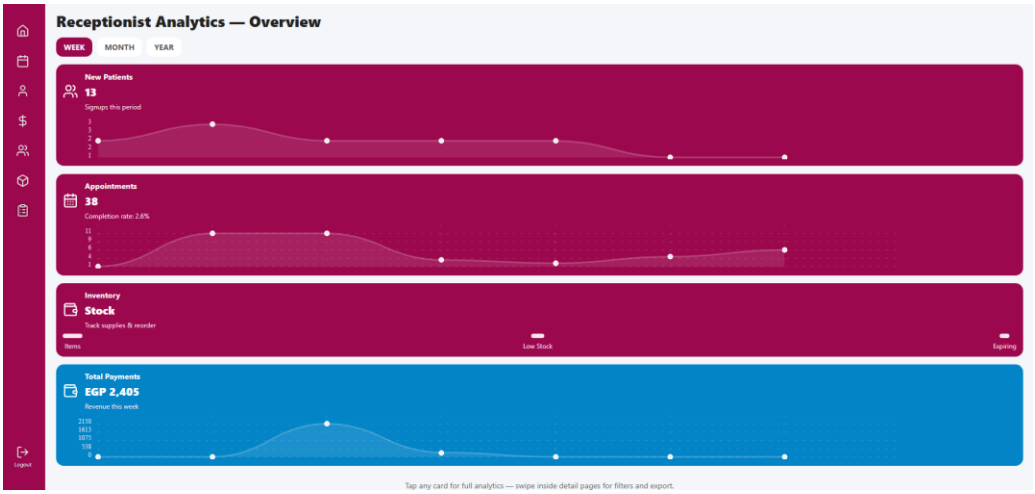
Payments



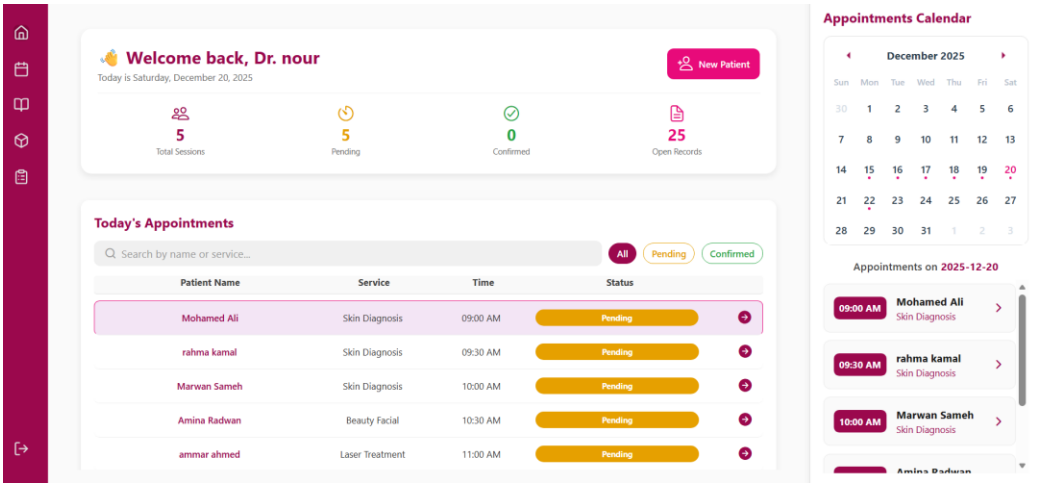
Receptionist Inventory



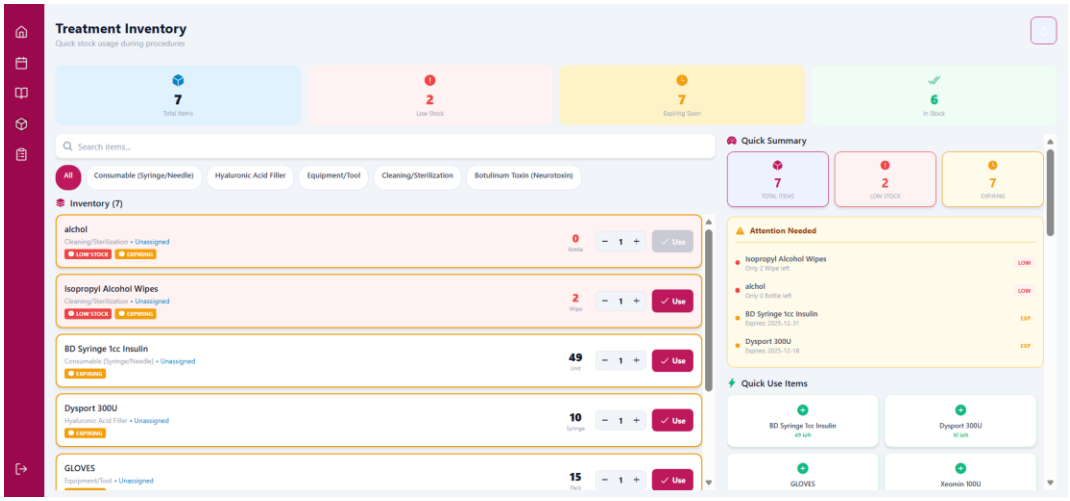
Receptionist Analytics



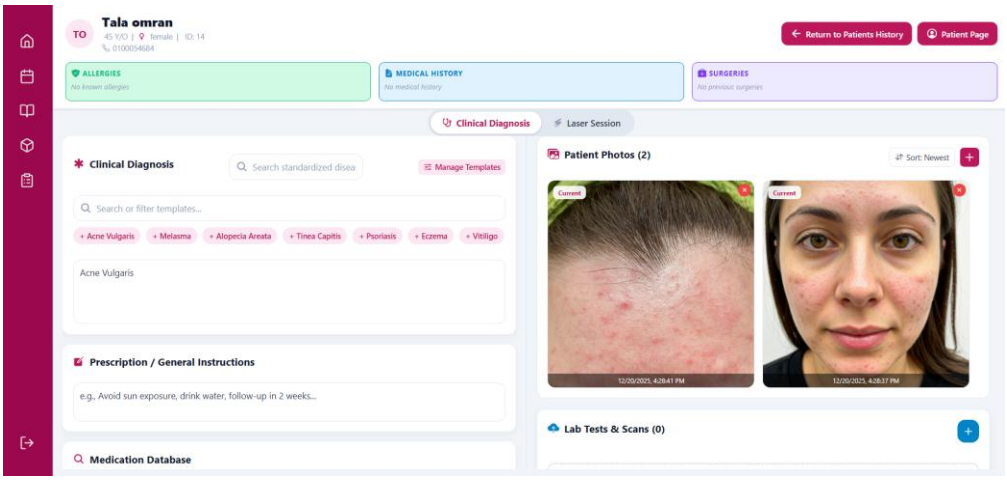
Doctor Dashboard

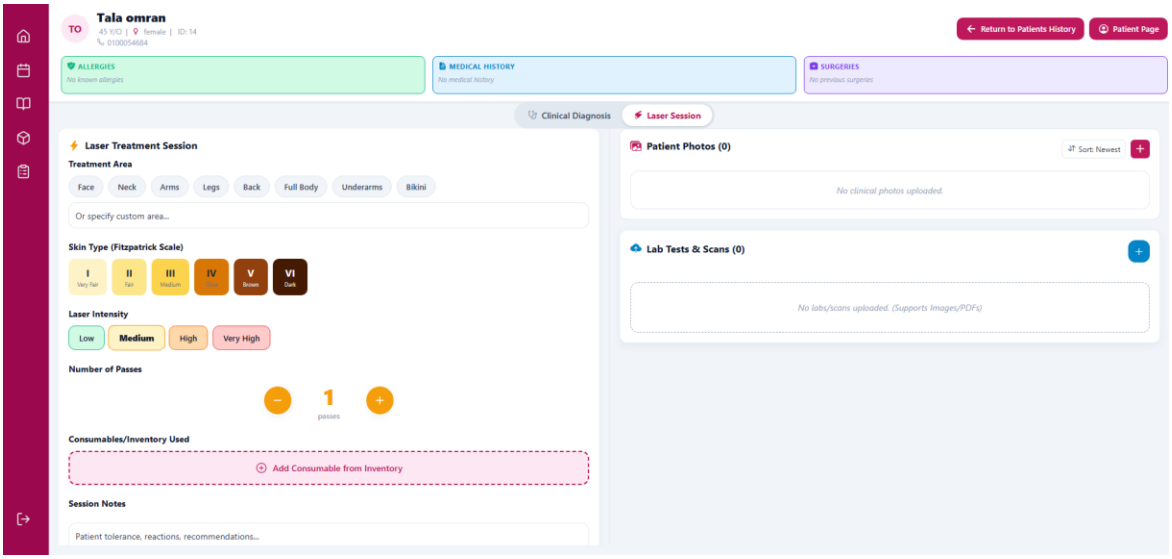


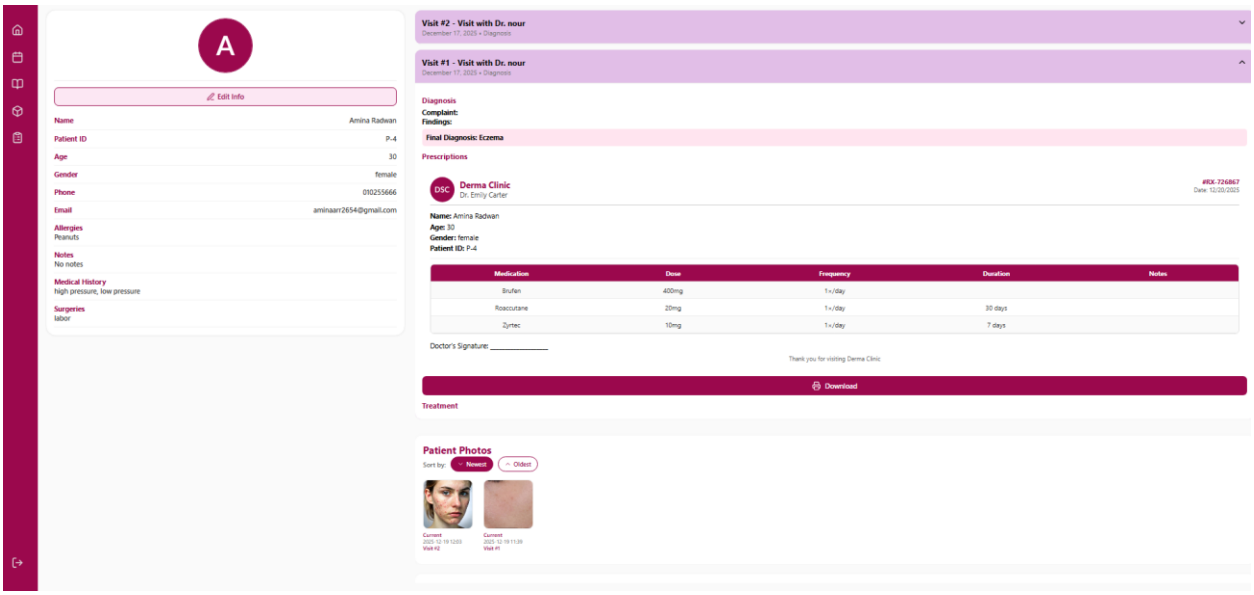
Inventory Management



Diagnosis & Prescription







Contribution Guidelines

1. Fork the repository
2. Create a new branch for your feature or fix
3. Follow coding standards and naming conventions
4. Add tests where applicable
5. Submit a clear pull request

Notes & Future Enhancements

- Patient portal
- Before/after image comparison
- SMS or WhatsApp notifications
- Advanced analytics dashboards
- PostgreSQL deployment
- Cloud file storage

Developed By

Biomedical Informatics Students – Nile University

- Nour Sameh
- Sarah Gado
- Nadeen Khaled
- Mohamed Hussien
- Youssef Adel
- Menna Adel

Focused on AI, Healthcare Security, and Clinical Software Systems.

Conclusion

This system models real dermatology clinic workflows:

- Receptionist manages bookings and payments
- Doctor handles diagnosis and medical decisions
- All clinic data is centralized, structured, and secure