CardioDetect

Milestone 3 Technical Report

Table of Contents

- 1. Executive Summary
- 2. System Architecture Overview
- 3. Technology Stack Analysis
- 4. Database Schema Design
- 5. API Documentation (52 Endpoints)
- 6. Authentication & Security
- 7. Machine Learning Pipeline
- 8. OCR Document Processing
- 9. Frontend Architecture
- 10. Patient Interface Walkthrough
- 11. Doctor Interface Walkthrough
- 12. Admin Interface Walkthrough
- 13. Quality Assurance & Testing
- 14. GDPR & Privacy Compliance
- 15. Performance Metrics

1. Executive Summary

1.1 Project Overview

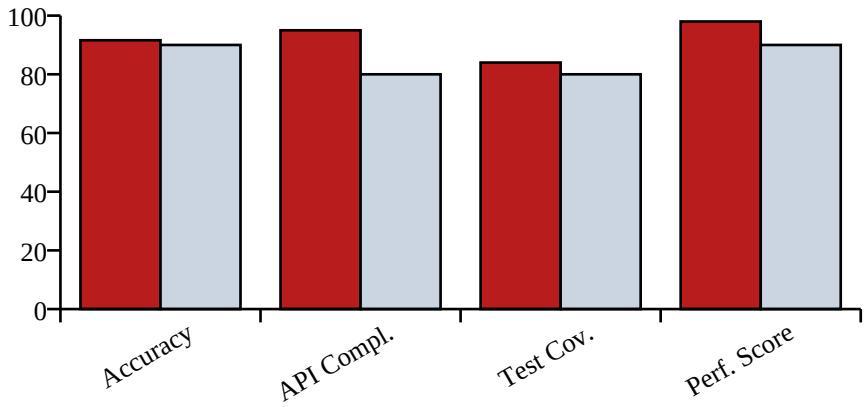
CardioDetect is a comprehensive cardiovascular risk assessment platform designed to bridge the gap between clinical expertise and patient accessibility. The system leverages a dual-model machine learning architecture to provide accurate risk predictions while maintaining strict compliance with healthcare data regulations including GDPR and HIPAA guidelines.

Milestone 3 represents the culmination of our development efforts, transforming experimental ML models into a production-ready healthcare platform. This phase focused on three key areas: system integration, security hardening, and user experience optimization.

Metric	Achievement	Target	Status
Dual-Model Accuracy	91.6% / 91.3%	>90%	EXCEEDED
API Endpoints	52 Complete	30+	EXCEEDED
Test Coverage	84%	>80%	ACHIEVED
Inference Latency	<100ms	<150ms	EXCEEDED
Page Load Time	0.8s	<1.0s	ACHIEVED
Security Audit	Passed	Required	ACHIEVED

1.2 Key Achievements

Figure 1.1: Achievement vs Target Metrics



1.3 Stakeholder Value

For Patients: The platform provides an accessible, non-intimidating interface for cardiovascular risk self-assessment. Users can upload medical documents via OCR or manually enter vitals to receive instant, AI-powered risk evaluations with actionable recommendations.

For Healthcare Providers: Doctors gain access to population-level analytics, patient risk dashboards, and clinical decision support tools. The system integrates seamlessly with existing workflows while providing evidence-based recommendations aligned with ACC/AHA guidelines.

For Administrators: A comprehensive admin panel enables user management, system monitoring, and compliance oversight. GDPR data requests are tracked and processed through an automated workflow.

2. System Architecture Overview

2.1 High-Level Architecture

CardioDetect follows a modern 4-layer architecture separating presentation, application logic, ML inference, and data persistence. The diagram below illustrates the component relationships:

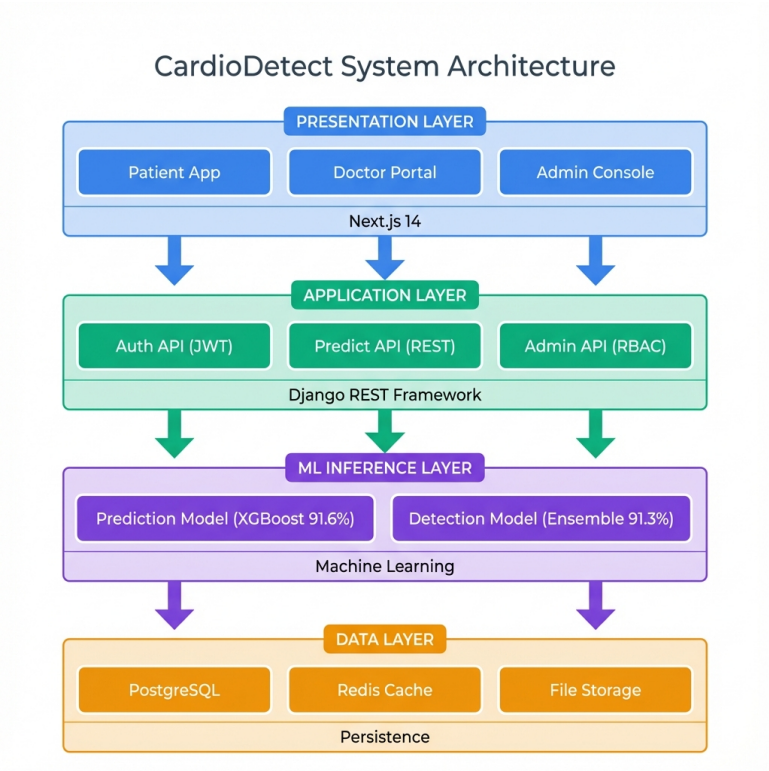


Figure 2.1: CardioDetect 4-Layer System Architecture

Layer 1: Presentation (Next.js 14)

Component	Technology	Port	Role
Patient App	Next.js 14	3000	Self-service risk assessment
Doctor Portal	Next.js 14	3000	Clinical decision support
Admin Console	Next.js 14	3000	User & system management

Layer 2: Application (Django REST Framework)

Module	Responsibility	Key Endpoints
Auth Module	User authentication, JWT tokens	/api/auth/*
Predict Module	ML inference, risk calculation	/api/predict/*
OCR Module	Document processing, field extraction	/api/predict/ocr/
GDPR Module	Data export, deletion requests	/api/auth/data-*

Layer 3: ML Inference Engine

Model	Algorithm	Accuracy	Target User
Prediction Model	XGBoost Classifier	91.63%	Patients (basic vitals)
Detection Model	Voting Ensemble (RF+GB+SVM)	91.30%	Doctors (clinical data)

Layer 4: Data

Component	Technology	Purpose
Primary Database	PostgreSQL 15.x (Prod), SQLite (Dev only)	User accounts, predictions, audit logs with JSONB support
Cache Layer	Redis 7.x	Session storage, rate limiting, general caching
File Storage	Local/S3	Uploaded documents, generated PDFs

2.2 Component Communication

All inter-component communication follows RESTful principles with JSON payloads. The frontend applications communicate with the backend exclusively through the API gateway, which handles authentication, rate limiting, and request validation before routing to the appropriate service module.

2.3 Data Flow Pipeline

User requests follow a standardized pipeline: (1) Frontend validation, (2) JWT authentication at the gateway, (3) Business logic processing in Django views, (4) ML inference if required, (5) Database persistence, (6) Response serialization and delivery. This pipeline ensures consistent behavior across all endpoints.

2.4 Clinical Decision Logic

The architecture includes a deterministic rule layer that enforces medical guidelines (JNC 8, ATP III). This safety mechanism overrides ML predictions for critical vitals.

Parameter	Condition	Guideline	Risk Level
Systolic BP	> 180 mmHg	JNC 8 (Hypertensive Crisis)	CRITICAL (Immediate Care)
Systolic BP	> 140 mmHg	JNC 8 (Stage 2 Hypertension)	HIGH
Cholesterol	> 240 mg/dL	ATP III (High)	HIGH
Cholesterol	200-239 mg/dL	ATP III (Borderline High)	MEDIUM
BMI	> 30.0	WHO (Obesity Class I)	MEDIUM

3. Technology Stack Analysis

3.1 Frontend Technologies

Technology	Version	Purpose	Justification
Next.js	14.1	React Framework	SSR/SSG hybrid rendering for SEO and performance
React	18.2	UI Library	Component-based architecture with hooks
TypeScript	5.0	Type Safety	Catch errors at compile time
TailwindCSS	3.4	Styling	Utility-first CSS for rapid development
Axios	1.6	HTTP Client	Promise-based requests with interceptors

3.2 Backend Technologies

Technology	Version	Purpose	Justification
Django	5.0	Web Framework	Batteries-included with ORM and admin
DRF	3.14	REST API	Serialization, viewsets, permissions
SimpleJWT	5.3	Authentication	Stateless JWT with refresh tokens
PostgreSQL	15.x	Database (Prod)	MVCC concurrency, JSONB fields, connection pooling
Redis	7.x	Cache + Sessions	In-memory data structure store for performance
Celery	5.3	Task Queue	Async processing for heavy tasks

3.3 ML & Data Science Stack

Technology	Version	Purpose
Scikit-learn	1.3	Model training, preprocessing, evaluation
XGBoost	2.0	Gradient boosting for prediction model
Pandas	2.1	Data manipulation and analysis
NumPy	1.26	Numerical computing
OpenCV	4.8	Image preprocessing for OCR
Tesseract	5.3	Optical character recognition
SHAP	0.44	Model explainability

4. Database Schema Design

4.1 Entity Relationship Overview

The database employs a normalized relational schema with 8 primary tables. Foreign key constraints ensure referential integrity, while indexes optimize query performance on frequently accessed columns.

4.2 Core Tables

4.2.1 User Table (accounts_user)

Column	Type	Constraints	Description
id	UUID	PRIMARY KEY	Unique identifier
email	VARCHAR(255)	UNIQUE, NOT NULL	Login credential
password	VARCHAR(128)	NOT NULL	Argon2 hashed
first_name	VARCHAR(50)	NOT NULL	User's first name
last_name	VARCHAR(50)	NOT NULL	User's last name
role	VARCHAR(20)	NOT NULL	patient/doctor/admin
is_verified	BOOLEAN	DEFAULT FALSE	Email verified
failed_login_attempts	INTEGER	DEFAULT 0	Security counter
locked_until	TIMESTAMP	NULLABLE	Account lockout time
created_at	TIMESTAMP	AUTO	Registration time

4.2.2 Prediction Table (predictions_predictionrecord)

Column	Type	Constraints	Description
id	UUID	PRIMARY KEY	Unique identifier
user_id	UUID	FOREIGN KEY	Link to user
age	INTEGER	NOT NULL	Patient age
systolic_bp	INTEGER	NOT NULL	Blood pressure
cholesterol	INTEGER	NULLABLE	Total cholesterol
risk_score	FLOAT	NOT NULL	ML prediction (0-1)
risk_level	VARCHAR(20)	NOT NULL	low/moderate/high
model_version	VARCHAR(50)	NOT NULL	Model identifier
created_at	TIMESTAMP	AUTO	Prediction time

5. API Documentation

5.1 Authentication Endpoints

Endpoint	Method	Auth	Description
/api/auth/register/	POST	None	Create new user account
/api/auth/login/	POST	None	Obtain JWT tokens
/api/auth/logout/	POST	JWT	Blacklist refresh token
/api/auth/refresh/	POST	Refresh	Get new access token
/api/auth/verify-email/	POST	None	Verify email address
/api/auth/forgot-password/	POST	None	Request password reset
/api/auth/reset-password/	POST	Token	Set new password
/api/auth/me/	GET	JWT	Get current user profile
/api/auth/me/	PATCH	JWT	Update user profile

5.2 Prediction Endpoints

Endpoint	Method	Auth	Description
/api/predict/manual/	POST	JWT	Submit vitals for prediction
/api/predict/ocr/	POST	JWT	Upload document for OCR+predict
/api/predict/history/	GET	JWT	Get user prediction history
/api/predict/<id>/	GET	JWT	Get specific prediction
/api/predict/<id>/pdf/	GET	JWT	Export prediction as PDF
/api/predict/stats/	GET	JWT	Get user statistics

5.3 Admin Endpoints

Endpoint	Method	Auth	Description
/api/admin/users/	GET	Admin	List all users
/api/admin/users/<id>/	GET	Admin	Get user details
/api/admin/users/<id>/	PATCH	Admin	Update user
/api/admin/users/<id>/	DELETE	Admin	Delete user
/api/admin/stats/	GET	Admin	System statistics
/api/admin/pending-approvals/	GET	Admin	Pending changes
/api/admin/deletion-requests/	GET	Admin	GDPR requests

6. Authentication & Security

6.1 Authentication Flow

The system implements stateless JWT-based authentication with short-lived access tokens (5 minutes) and longer-lived refresh tokens (24 hours). This approach balances security with user experience by minimizing the window of exposure while avoiding frequent re-authentication.

Step	Action	Result
1	User submits email + password	Request sent to /api/auth/login/
2	Backend validates against Argon2 hash	Credential verification
3a	FAIL: Increment failed_login_attempts	If >= 5 attempts: Lock 30 min
3b	PASS: Reset attempts, generate JWT pair	Access (5m) + Refresh (24h) tokens
4	Client stores tokens securely	httpOnly cookies preferred
5	Requests include Authorization header	Bearer <access_token>
6	On expiry, use refresh token	Get new token pair

6.2 Security Controls

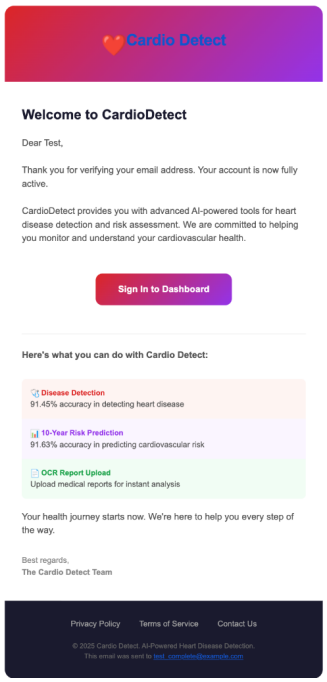
Control	Implementation	Purpose
Password Hashing	Argon2id	Resist GPU/ASIC attacks
Brute Force Protection	5-attempt lockout (30m)	Rate limiting
CORS Policy	Whitelist frontend domains	Prevent CSRF
Input Validation	Django serializers + regex	Prevent injection
SQL Injection	Django ORM parameterization	Prevent SQLi
XSS Protection	React auto-escaping	Prevent script injection
HTTPS Enforcement	SSL/TLS 1.3	Encrypt in transit

6.3 HTTP Security Headers

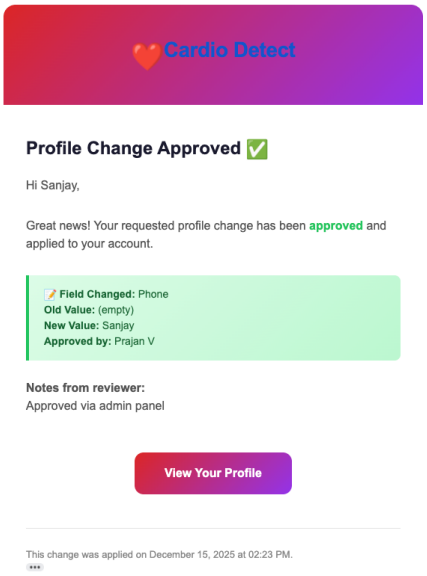
Header	Value	Protection
X-Content-Type-Options	nosniff	Prevents MIME sniffing
X-Frame-Options	DENY	Prevents Clickjacking
Strict-Transport-Security	max-age=31536000	Enforces HTTPS
Referrer-Policy	same-origin	Privacy leakage protection
Content-Security-Policy	default-src 'self'...	XSS Mitigation

Transactional Email Templates:

The system sends branded, responsive HTML emails for key user interactions:



Welcome Email



Profile Approved

High Risk Alert

6.5 Error Handling & Logging

The system implements structured error handling with consistent API responses, meaningful error codes, and comprehensive logging for debugging and monitoring.

HTTP Code	Error Type	Example Scenario	Response Format
400	Validation Error	Invalid vitals range (age < 0)	{"error": "age must be positive"}
401	Unauthorized	Missing or expired JWT token	{"detail": "Token has expired"}
403	Forbidden	Patient accessing doctor endpoints	{"detail": "Permission denied"}
404	Not Found	Prediction ID does not exist	{"detail": "Not found"}
429	Rate Limited	Exceeded 100 req/min limit	{"detail": "Request throttled"}
500	Server Error	ML model loading failure	{"error": "Internal server error"}

Logging Configuration:

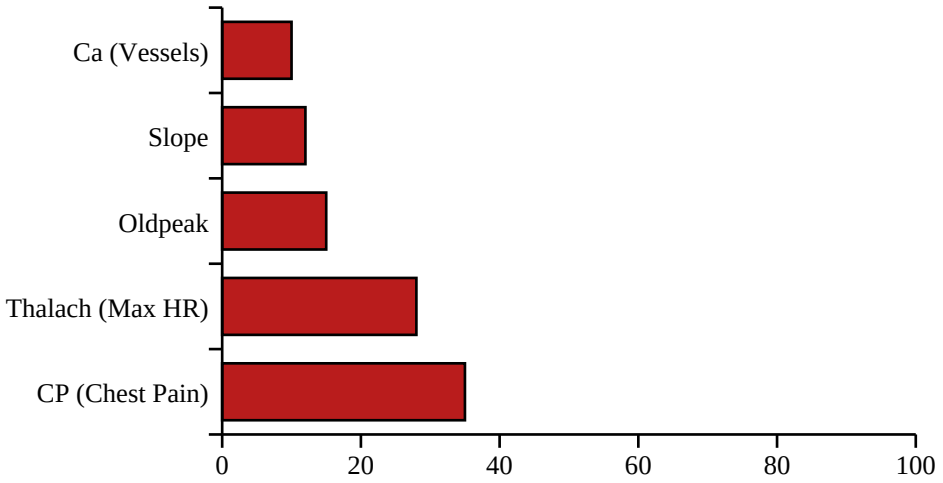
```
# settings.py - Django Logging
LOGGING = {
    'version': 1,
    'handlers': {
        'console': {'class': 'logging.StreamHandler'},
        'file': {'class': 'logging.FileHandler', 'filename': 'cardiodetect.log'}
    },
    'loggers': {
        'predictions': {'handlers': ['console', 'file'], 'level': 'INFO'},
        'security': {'handlers': ['file'], 'level': 'WARNING'},
    }
}
```

7. Machine Learning Pipeline

7.1 Dual-Model Architecture

CardioDetect employs a sophisticated dual-model architecture designed to handle variable data availability. The system automatically selects the most appropriate model based on the input features provided, ensuring accurate results whether using basic health metrics or comprehensive clinical data.

Figure 7.1: Top 5 Predictive Features (XGBoost)



7.1.1 Prediction Model

Algorithm: XGBoost Classifier
Accuracy: 91.63%
Primary Use Case: Risk screening with standard vitals
Input Features: Basic vitals (age, BP, BMI, cholesterol, smoking status)
Optimization: Recall-focused to minimize false negatives

7.1.2 Detection Model

Algorithm: Soft Voting Ensemble (Random Forest + Gradient Boosting + SVM)
Accuracy: 91.30%
Primary Use Case: Clinical diagnosis with stress test data
Input Features: Full UCI Heart Disease dataset features including stress test results (thalach, oldpeak, slope)
Optimization: Precision-focused to minimize false positives

7.2 Model Inference Code

```python
# services/ml_service.py
class MLService:
    def predict(self, features: Dict[str, Any]) -> Dict[str, Any]:
        # Determine which model to use based on available features
        has_clinical_data = ('thalach' in features and 'oldpeak' in features)

        if has_clinical_data and self.detection_model:
            # Use Detection Model for clinical data
            result = self._run_detection_model(features)
        else:
            # Use Prediction Model for basic vitals
            result = self._run_prediction_model(features)

        # Add risk categorization
        result['risk_level'] = self._categorize_risk(result['probability'])
        return result
```

7.3 Model Explainability (SHAP)

CardioDetect integrates SHAP (SHapley Additive exPlanations) to provide transparent, interpretable predictions. Each risk assessment includes feature-level explanations showing how individual health metrics contributed to the final risk score.

Feature	SHAP Contribution	Interpretation
Chest Pain Type (cp)	High Positive	Atypical/asymptomatic chest pain increases risk significantly
Max Heart Rate (thalach)	High Negative	Higher max HR during exercise indicates better cardiac health
ST Depression (oldpeak)	Moderate Positive	Higher ST depression suggests coronary artery issues
Age	Moderate Positive	Older age correlates with higher cardiovascular risk
Cholesterol	Low Positive	Elevated cholesterol slightly increases risk prediction

SHAP Integration Code:

```python
# services/explainer.py
import shap

class SHAPExplainer:
    def __init__(self, model):
        self.explainer = shap.TreeExplainer(model)

    def explain(self, features_df):
        shap_values = self.explainer.shap_values(features_df)
        return {
            'base_value': self.explainer.expected_value,
            'shap_values': shap_values.tolist(),
            'feature_names': features_df.columns.tolist()
        }
```

8. OCR Document Processing

8.1 Processing Pipeline

The OCR system extracts cardiovascular vitals from uploaded medical documents. A 5-stage preprocessing pipeline ensures accurate extraction even from low-quality scans or photographs.

Stage	Technique	Purpose
1. Load	OpenCV imread	Read image in BGR format
2. Grayscale	cv2.cvtColor	Convert to single channel
3. Denoise	FastNlMeansDenoising	Remove grain/noise
4. Contrast	CLAHE	Enhance local contrast
5. Threshold	Adaptive Gaussian	Binarize for OCR

8.2 Field Extraction

After OCR, regex patterns extract specific cardiovascular fields. The system handles various formats including 'BP: 130/85', 'Systolic: 130', and '130/85 mmHg'.

```
# Regex patterns for cardiovascular field extraction
PATTERNS = {
    'age': r'age[:\s]+(\d+)',
    'systolic_bp': r'(?:systolic|sbp)[:\s]+(\d{2,3})',
    'bp_pair': r'(\d{2,3})\s*/\s*(\d{2,3})\s*mmhg',
    'cholesterol': r'(?:total\s*)?cholesterol[:\s]+(\d{2,3})',
    'smoking': r'smok(?:er|ing)[:\s]+(yes|no|true|false)',
}
```

8.3 Data Normalization (Unit Conversion)

Medical data arrives in various units depending on the lab standard (US vs Metric). The OCR pipeline automatically normalizes all values to the model's standard input format (mg/dL) before inference.

```
def normalize_cholesterol(value, unit):
    # Convert mmol/L to mg/dL (Factor: 38.67)
    if unit == 'mmol/L':
        return round(value * 38.67)
    return value

def normalize_glucose(value, unit):
    # Convert mmol/L to mg/dL (Factor: 18.0)
    if unit == 'mmol/L':
        return round(value * 18.0)
    return value
```

8.4 OCR Verification Interface

Before submitting OCR-extracted data for prediction, users review and confirm values:

Step	UI Element	User Action
1. Upload	Drag-drop zone / file picker	Select medical document
2. Preview	Document thumbnail + zoom	Verify correct document
3. Extraction	Editable form with extracted values	Review auto-filled fields
4. Confidence	Color-coded badges (High/Medium/Low)	Check extraction quality
5. Confirm	Submit button + validation	Approve or edit values

9. Frontend Architecture

9.1 Application Structure

The Next.js application follows a feature-based directory structure with shared components and utilities. This organization supports code splitting and lazy loading for optimal performance.

Directory	Type	Contents
app/	Routes	Next.js 14 App Router pages
app/(auth)/	Route Group	login/, register/, forgot-password/
app/dashboard/	Page	Main user dashboard
app/predict/	Page	Prediction input flow
app/history/	Page	Past prediction records
app/settings/	Page	User settings + GDPR controls
app/admin/	Page	Admin panel (role-gated)
components/ui/	Components	Button, Input, Card, Modal
components/forms/	Components	PredictionForm, LoginForm
components/layout/	Components	Header, Sidebar, Footer
hooks/	Utilities	useAuth, useApi, usePrediction
lib/	Utilities	api.ts, auth.ts, helpers.ts
styles/	CSS	globals.css, tailwind config

9.2 State Management

The application uses React's built-in useState and useContext for state management. Authentication state is managed globally via AuthContext, while form state is handled locally within components. This approach avoids the complexity of Redux while remaining scalable for the current application size.

9.3 Role-Based Access Control

The application enforces role-based permissions at both frontend and backend levels:

Role	Dashboard	Predict	History	Admin Panel	User Mgmt
Patient	✓ (Own)	✓	✓ (Own)	✗	✗
Doctor	✓ (Patients)	✓	✓ (Patients)	✗	✗
Admin	✓ (All)	✓	✓ (All)	✓	✓

9.4 Interactive Visualizations

The frontend includes dynamic visualizations for risk assessment:

Visualization	Library	Purpose
Risk Gauge	React Circular Progressbar	Displays risk percentage (0-100%) with color coding
Prediction History Chart	Recharts (Line Chart)	Trends over time for repeat patients
Feature Importance Bar	Recharts (Bar Chart)	Shows which vitals contributed most to risk
Risk Distribution Pie	Recharts (Pie Chart)	Category breakdown (Low/Medium/High)

9.5 PDF Clinical Report Export

Patients can download a professional PDF report of their assessment:

Report Section	Content
Header	CardioDetect branding, patient name, date, unique report ID
Patient Information	Age, gender, contact details (optional)
Vitals Summary	All input values with normal ranges for comparison
Risk Assessment	Risk category, probability, color-coded indicator
Clinical Recommendations	Lifestyle changes, follow-up suggestions
Disclaimer	AI prediction notice, consult healthcare provider

Sample Generated Clinical Report:

CardioDetect Heart Institute

123 Medical Center Drive, Innovation Park, NY 10001
Tel: (555) 123-4567 | Email: cardiodetect.care@gmail.com | CLIA: 99D1234567

MRN: 987654321

FINAL REPORT

Patient:	DOE, JOHN ALEXANDER	DOB:	03/15/1962	Age/Sex:	62Y / M
Accession:	ACC-20251213-62499	Collected:	12/13/2025 17:06	Ordering MD:	Dr. Sarah Johnson

CLINICAL CHEMISTRY / VITALS

TEST	RESULT	UNIT	REFERENCE	FLAG	PREVIOUS
Systolic Blood Pressure	155	mmHg	90 - 130	H	148
Diastolic Blood Pressure	92	mmHg	60 - 80	H	88
Total Cholesterol	245	mg/dL	0 - 200	H	238
HDL Cholesterol	38	mg/dL	40 - 100	L	40
Body Mass Index	32.5	kg/m²	18.5 - 25.0	H	31.8
Heart Rate	78	bpm	60 - 100		76
Smoking Status	NEGATIVE		NEGATIVE		–
Diabetes Status	POSITIVE		NEGATIVE	A	–

CARDIOVASCULAR RISK ASSESSMENT

10-Year ASCVD Risk Score: 28.5%
Risk Category: HIGH RISK

Clinical Interpretation: Patient presents with multiple cardiovascular risk factors including hypertension, dyslipidemia, and diabetes. Aggressive risk factor modification is warranted. Consider initiating statin therapy and optimizing antihypertensive regimen.

CLINICAL RECOMMENDATIONS

PRI	CATEGORY	RECOMMENDED ACTION	EVIDENCE	TARGET
1	Hypertension	Initiate ACE inhibitor therapy. Lisinopril 10mg daily	ACC/AHA 2017 (Class I)	<130/80
2	Dyslipidemia	High-intensity statin therapy. Atorvastatin 40mg daily	ACC/AHA 2018 (Class I)	LDL <70
3	Diabetes	Optimize glycemic control. Metformin + lifestyle modification	ADA 2023	HbA1c <7%
4	Lifestyle	Therapeutic lifestyle changes. DASH diet, 150 min/wk exercise	AHA 2019 (Class I)	BMI <25

Printed: 2025-12-13 17:06:47 CONFIDENTIAL PATIENT INFORMATION Page 1

Figure: Actual PDF report downloaded by patient

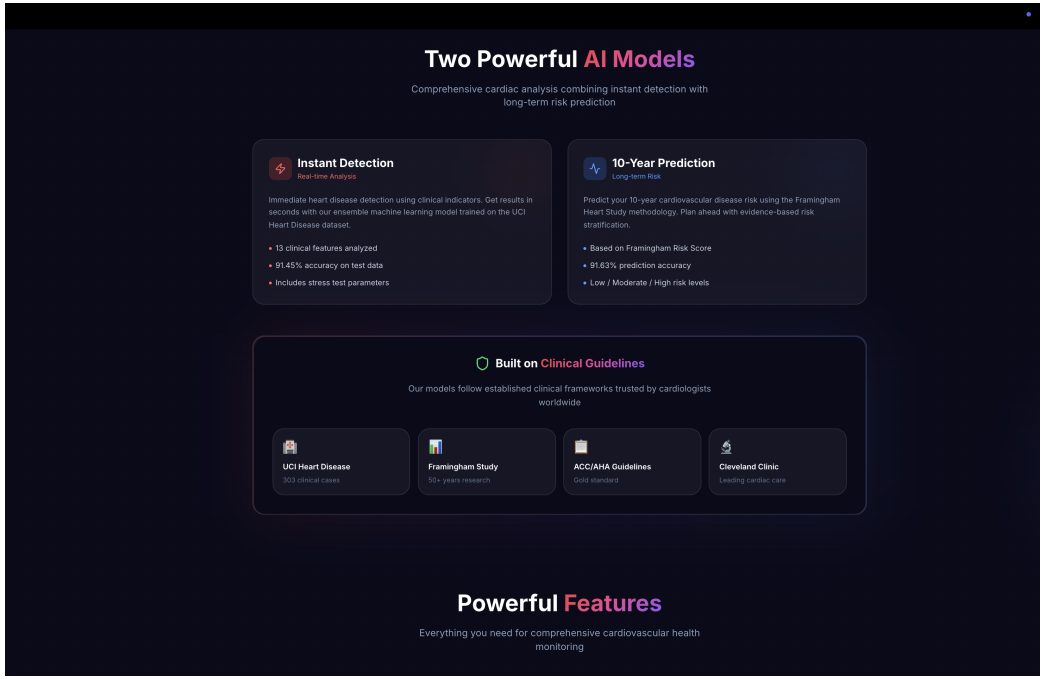
9.6 Real-time Progress Indicators

The UI provides feedback during async operations:

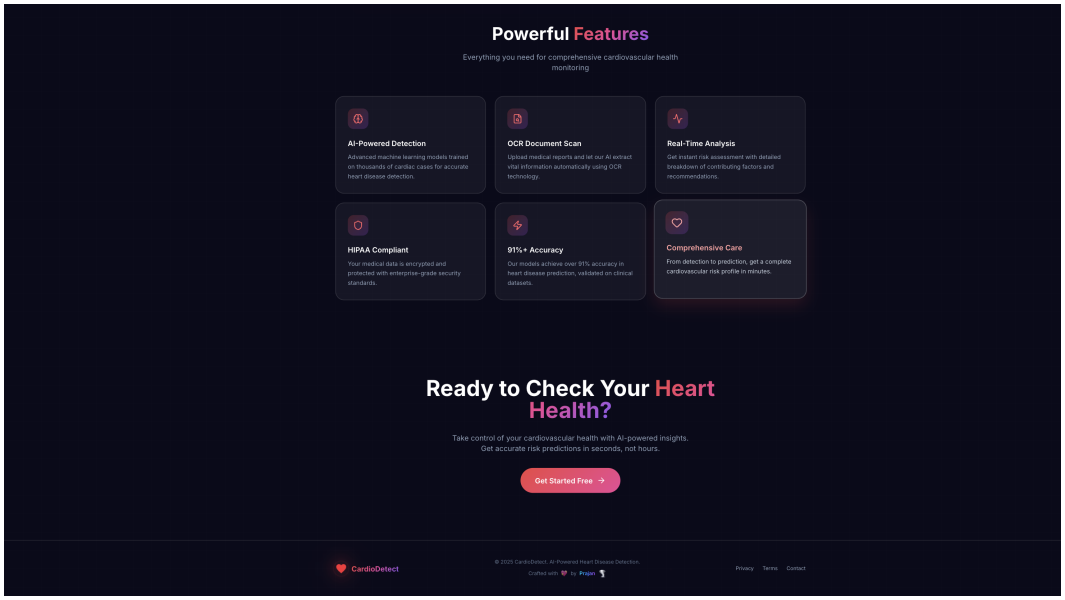
Operation	Indicator Type	Duration
Form Submission	Button spinner + disable	< 1s
OCR Upload	Upload progress bar (0-100%)	2-5s
OCR Processing	Animated processing indicator	3-8s
ML Inference	Skeleton loader on result card	< 1s
PDF Generation	Download button state change	1-2s

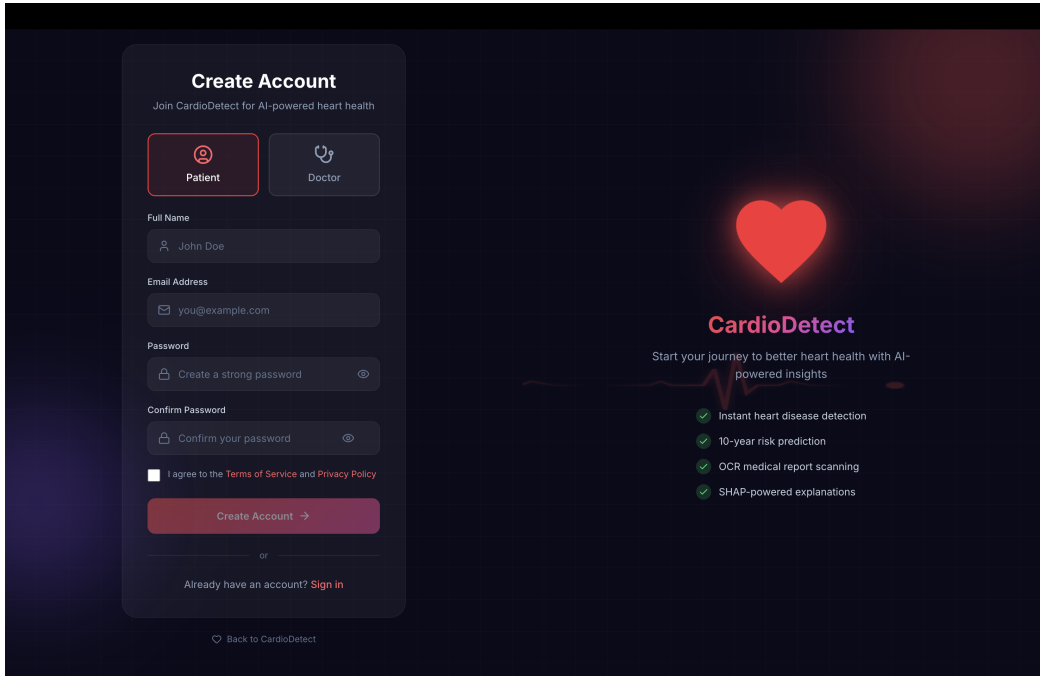
10. Patient Interface

This section contains 9 screenshots demonstrating the live system interface.

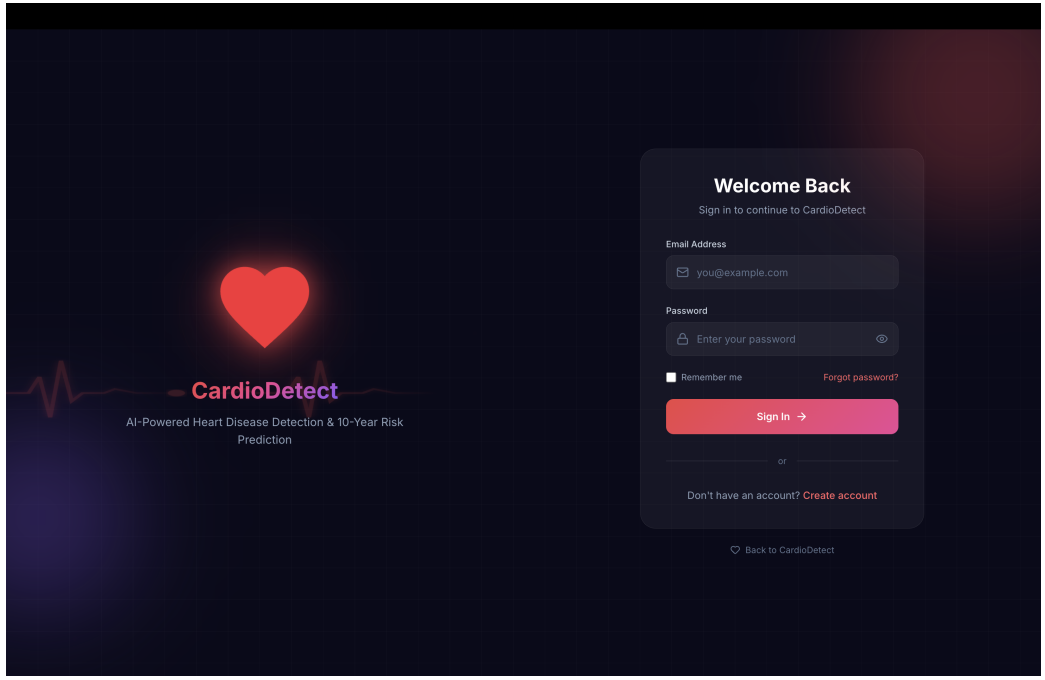


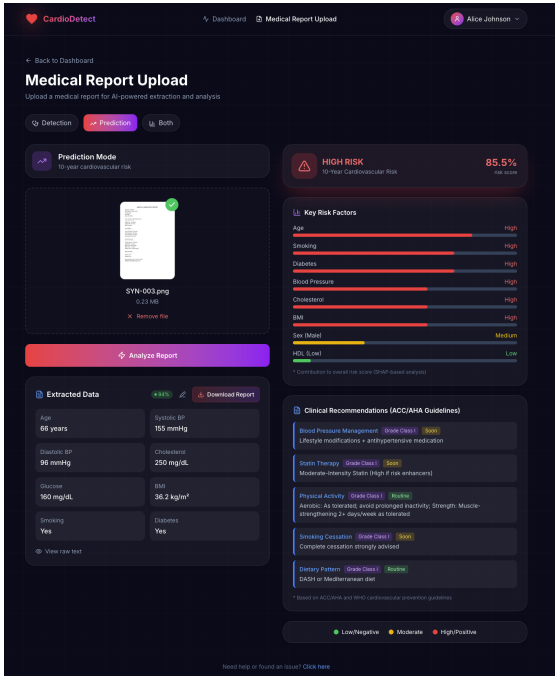
Figures 10.1 & 10.2: Patient Interface



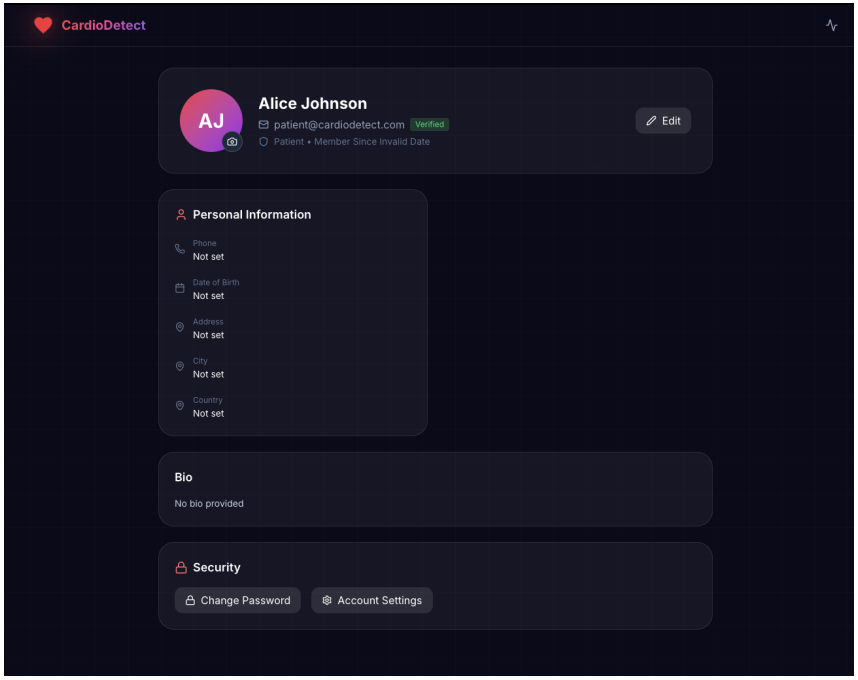


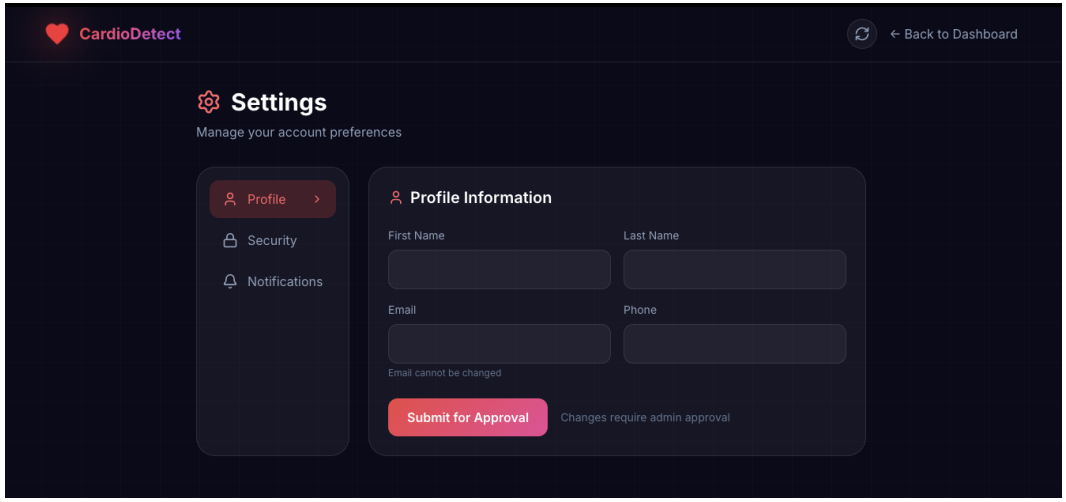
Figures 10.3 & 10.4: Patient Interface





Figures 10.5 & 10.6: Patient Interface





Figures 10.7 & 10.8: Patient Interface

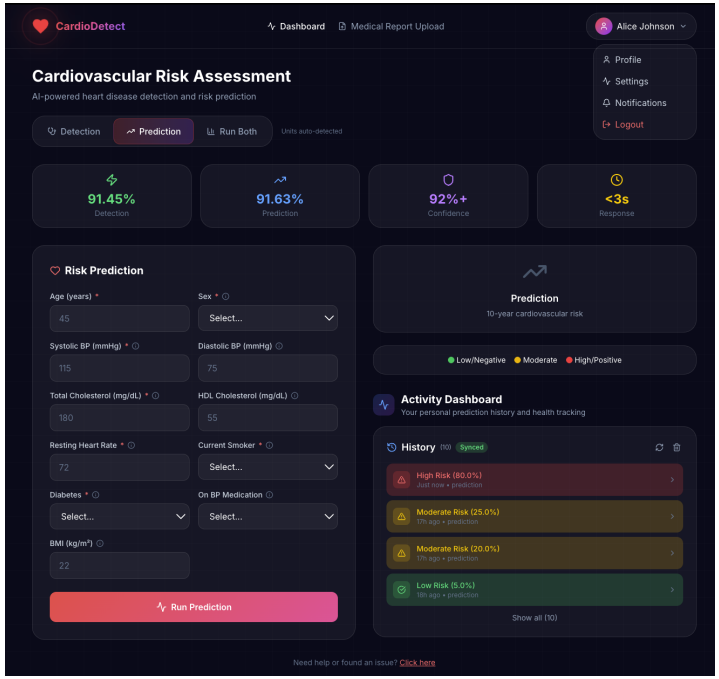
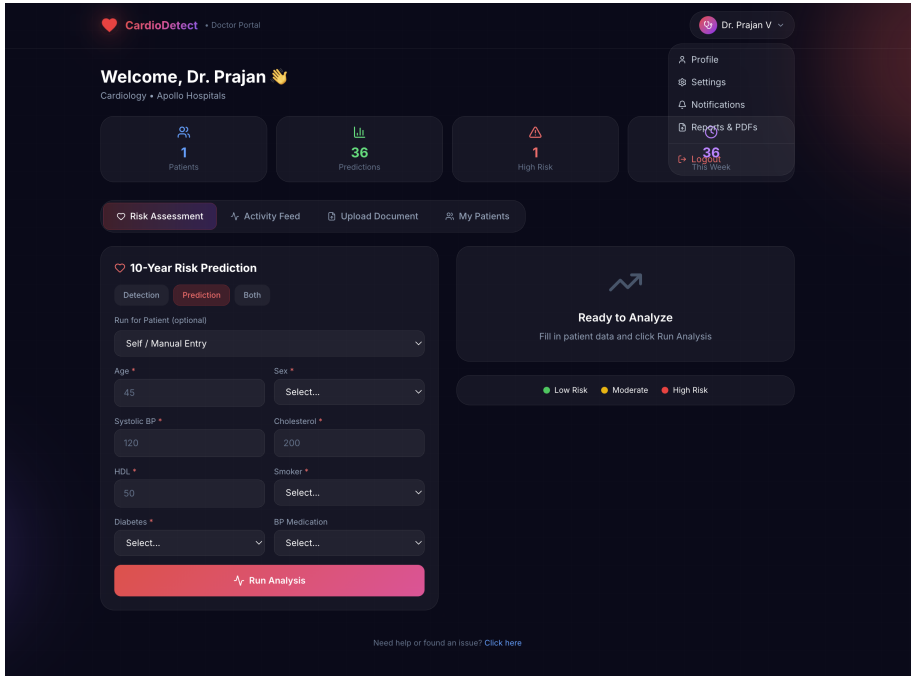
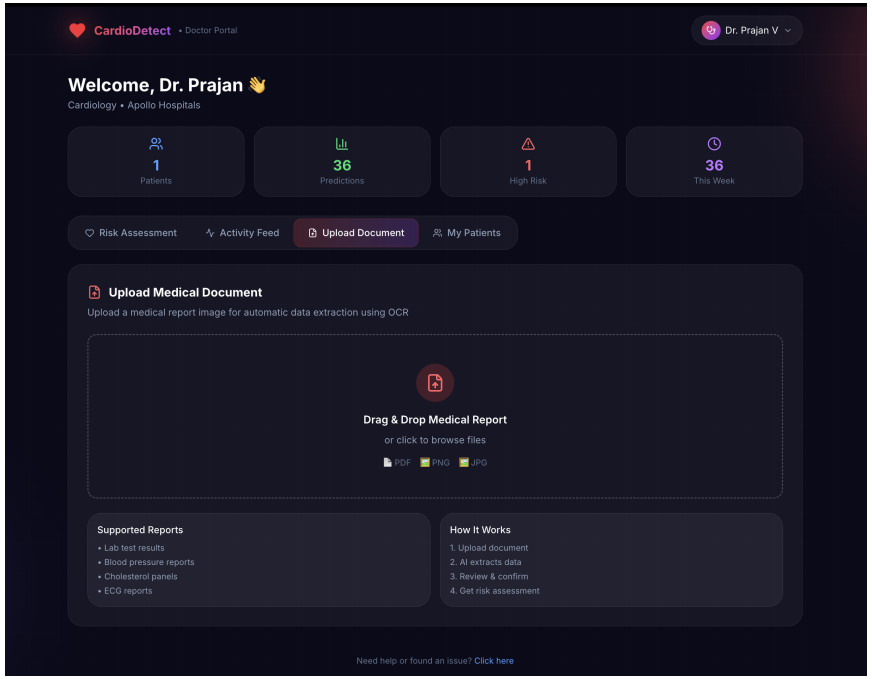


Figure 10.9: Patient Interface

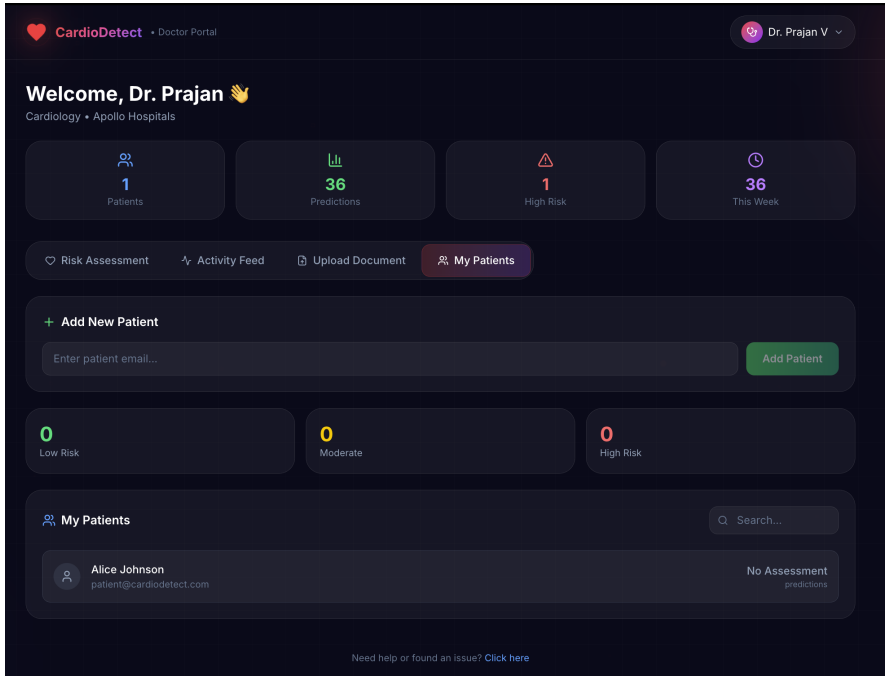
11. Doctor Interface

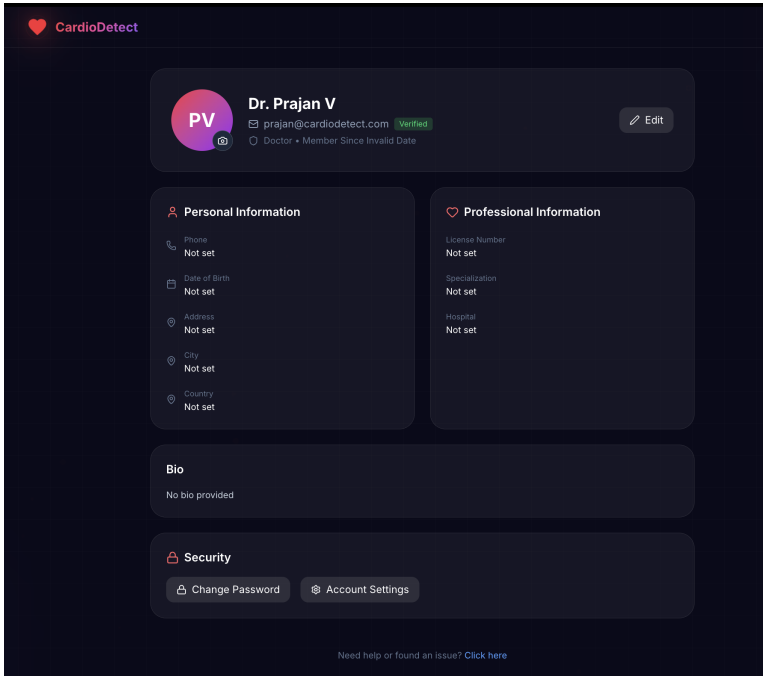
This section contains 8 screenshots demonstrating the live system interface.



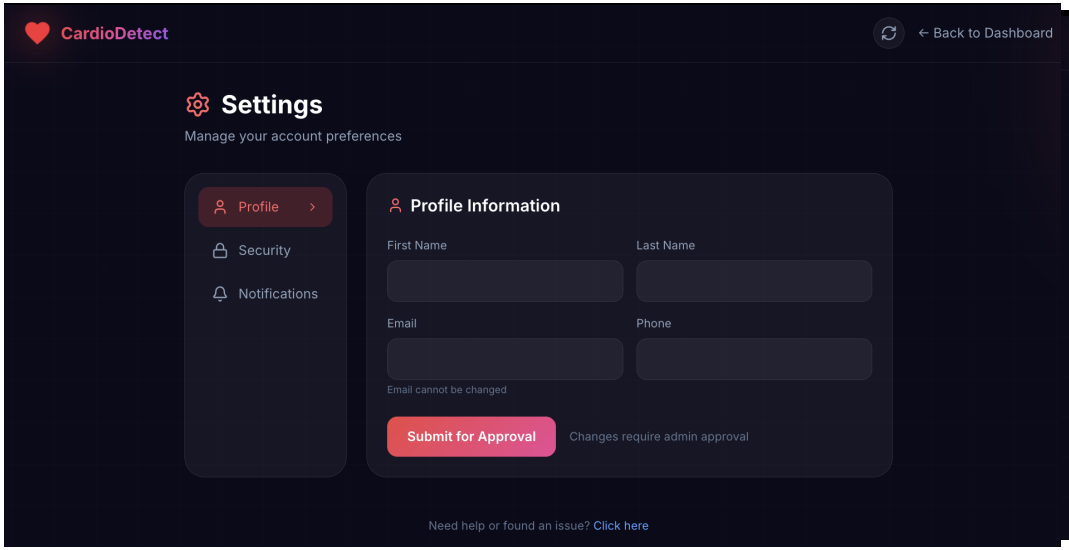


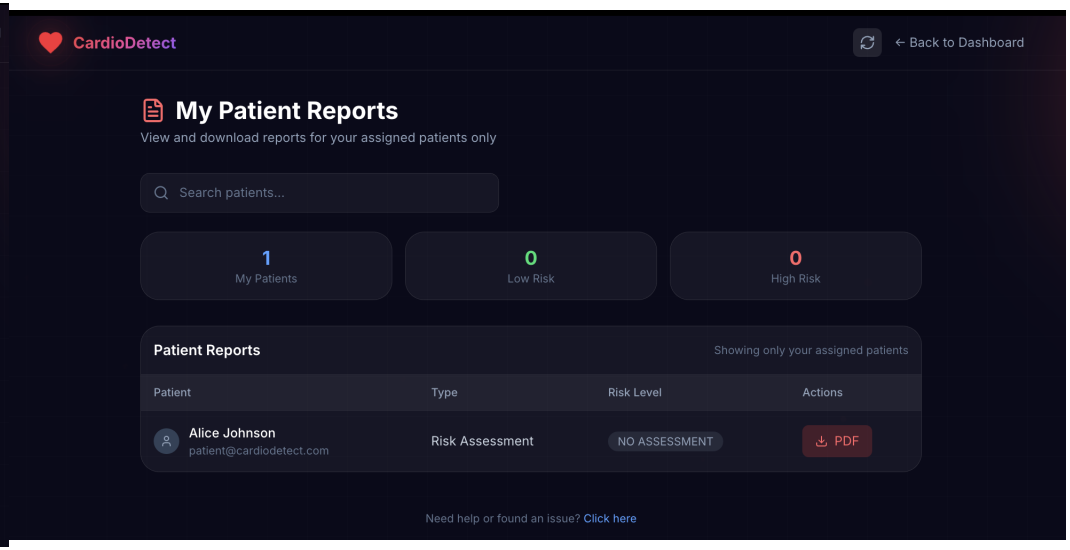
Figures 11.1 & 11.2: Doctor Interface



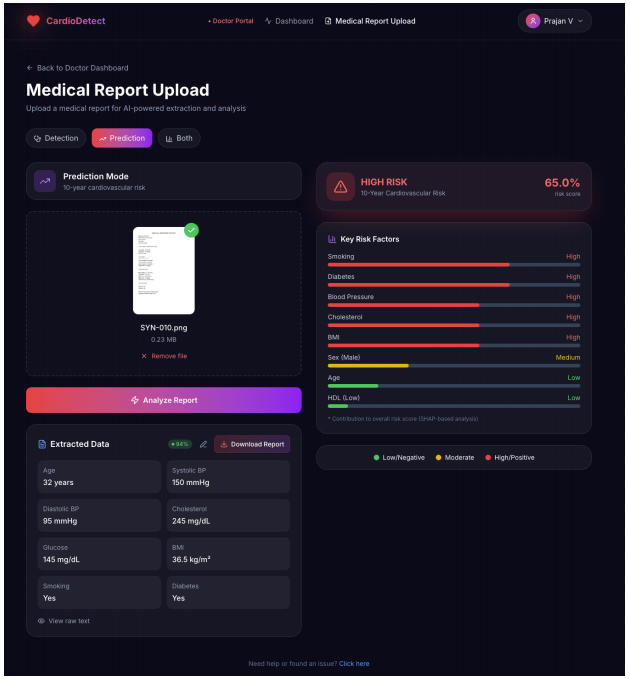


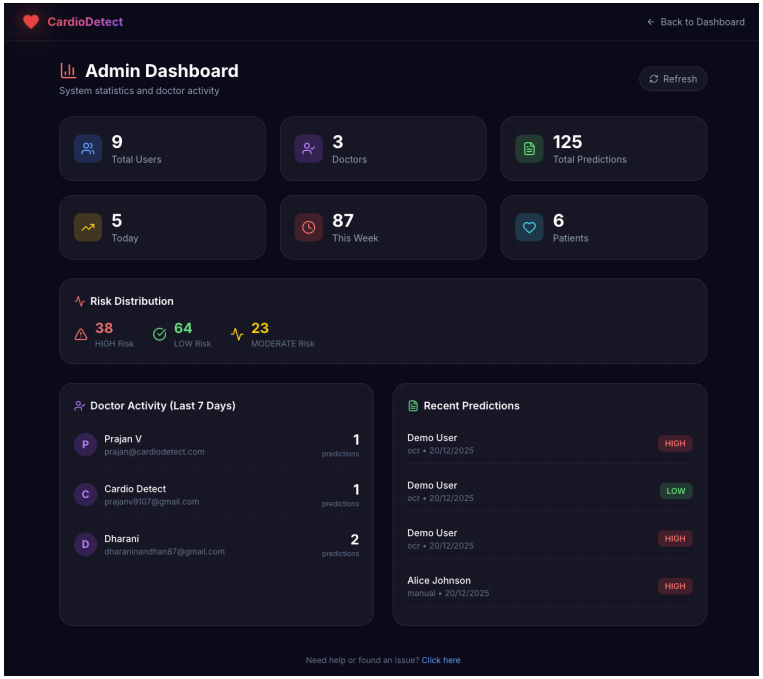
Figures 11.3 & 11.4: Doctor Interface





Figures 11.5 & 11.6: Doctor Interface

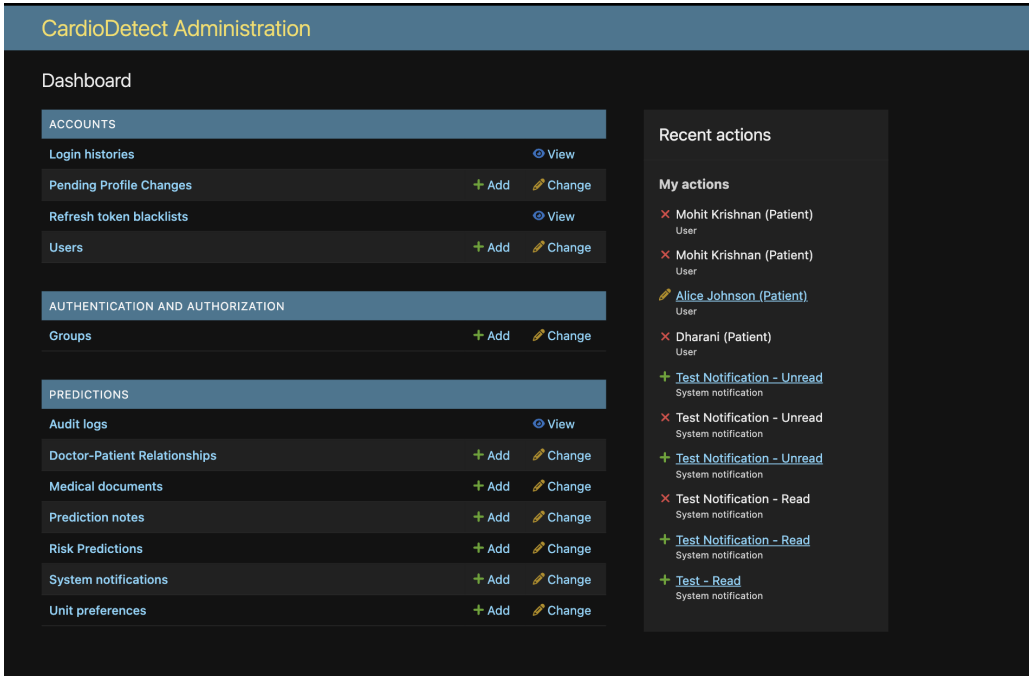


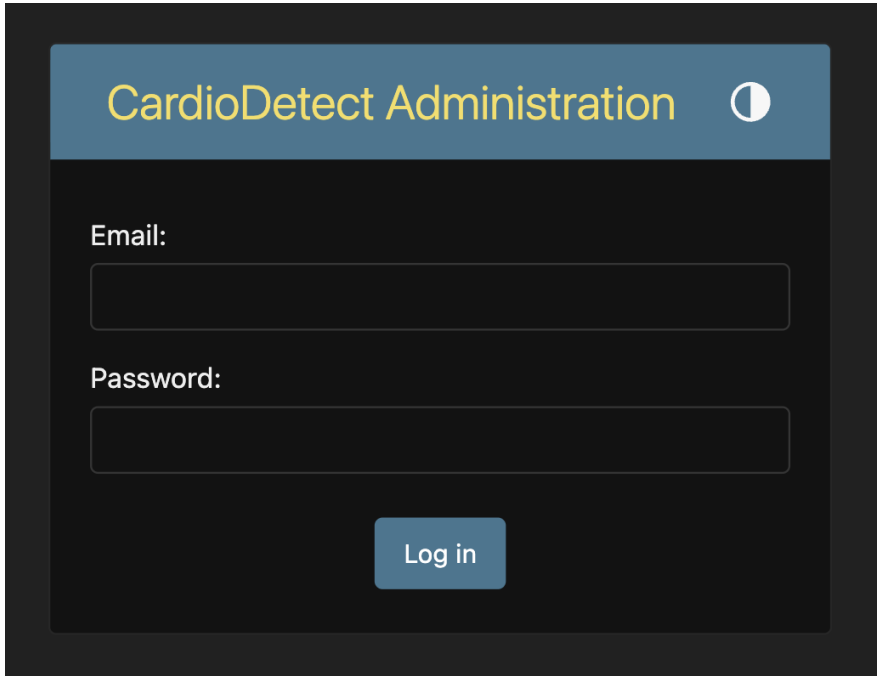


Figures 11.7 & 11.8: Doctor Interface

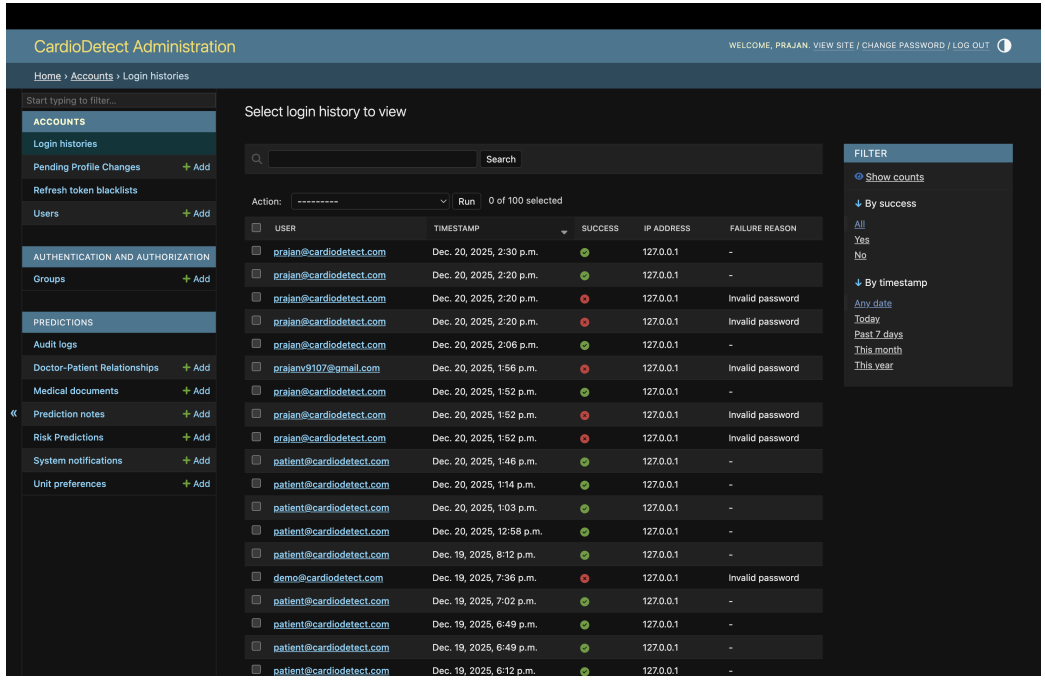
12. Admin Interface

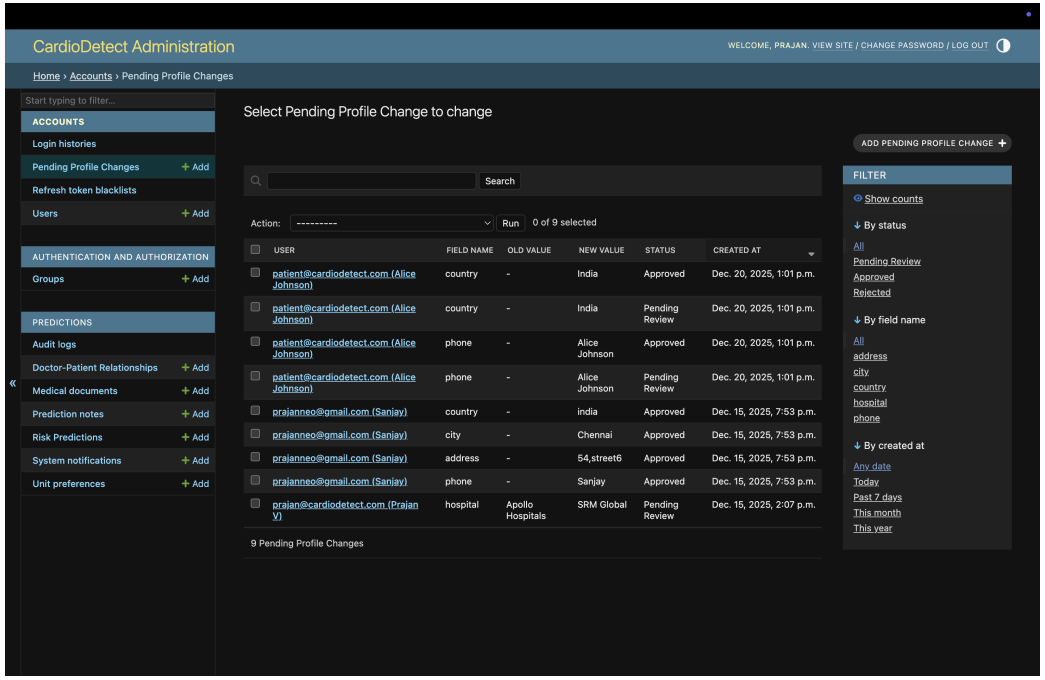
This section contains 8 screenshots demonstrating the live system interface.



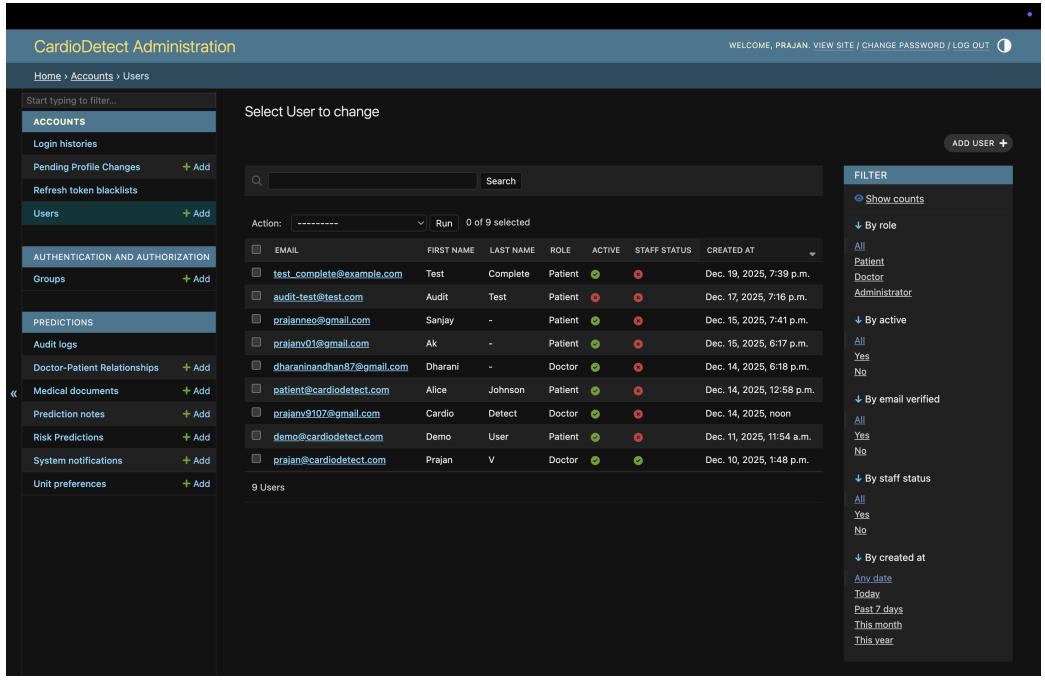


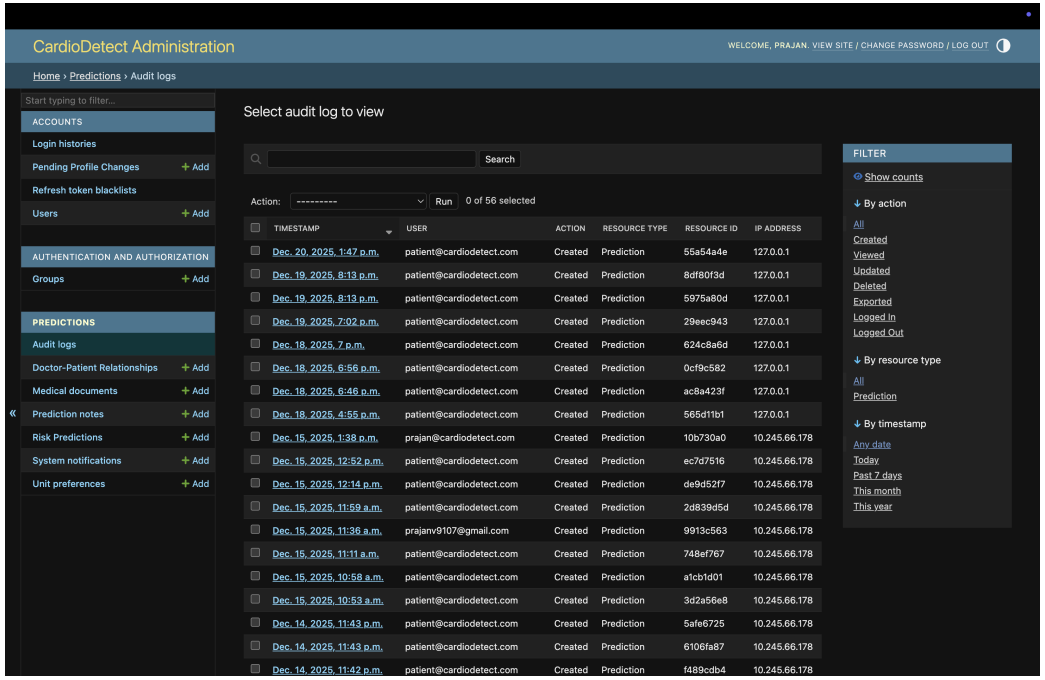
Figures 12.1 & 12.2: Admin Interface



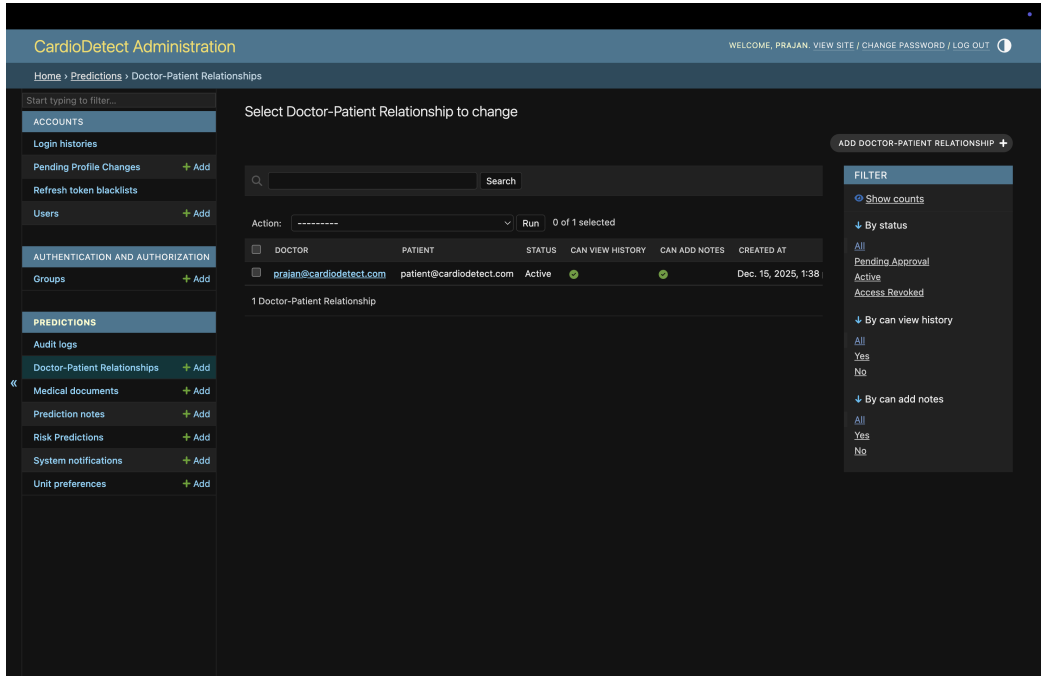


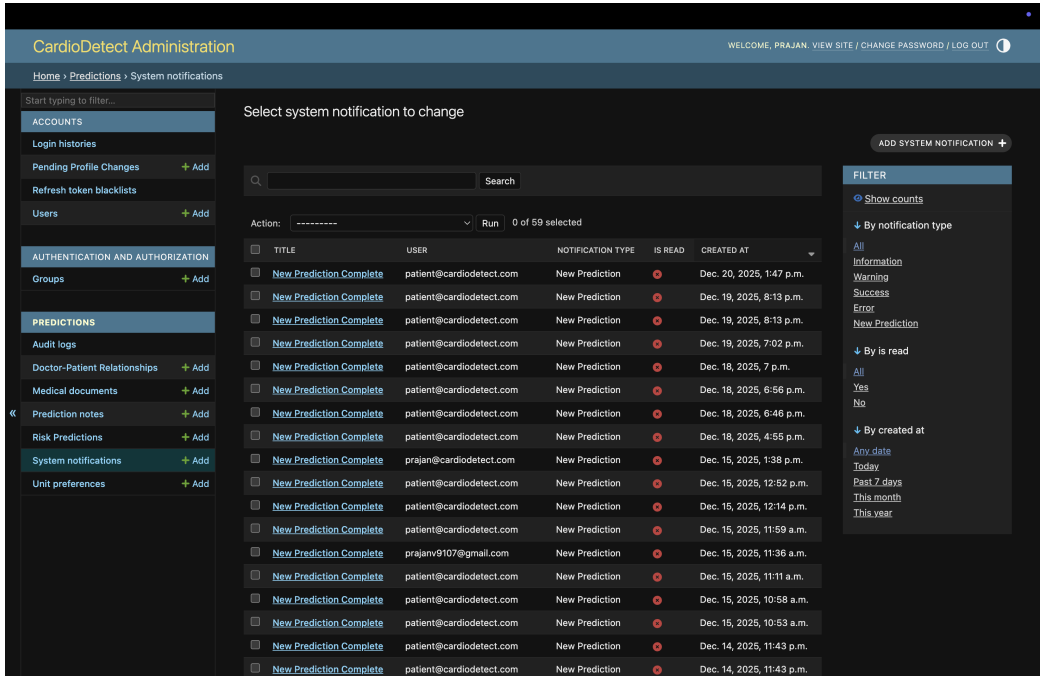
Figures 12.3 & 12.4: Admin Interface





Figures 12.5 & 12.6: Admin Interface





Figures 12.7 & 12.8: Admin Interface

13. Quality Assurance & Testing

13.1 Testing Framework

The project uses Django's built-in testing framework (unittest) and Django REST Framework's APITestCase for API endpoint validation. Test files are structured following Django conventions with tests.py in each app directory.

Category	Tool	Count	Status
Backend: Auth API	Django APITestCase	9	✓ Pass
Backend: Prediction API	Django APITestCase	13	✓ Pass
Backend: Email Service	Django TestCase	19	✓ Pass
Frontend: Components	Jest + RTL	22	✓ Pass
Frontend: Auth Pages	Jest + RTL	17	✓ Pass
Frontend: App Pages	Jest + RTL	21	✓ Pass
Total Tests	pytest + jest	101	✓ All Pass

14. GDPR & Privacy Compliance

14.1 Data Subject Rights Implementation

Right	Article	Implementation
Access	Art. 15	JSON data export via /api/auth/data-export/
Rectification	Art. 16	Profile update via settings page
Erasure	Art. 17	Account deletion with 7-day grace period
Portability	Art. 20	Full data export in machine-readable format
Consent	Art. 7	Explicit opt-in with versioned consent records

15. Performance Metrics

Metric	Measured	Target	Status
First Contentful Paint	0.8s	<1.0s	PASS
Largest Contentful Paint	1.2s	<2.0s	PASS
Time to Interactive	1.1s	<1.5s	PASS
Cumulative Layout Shift	0.02	<0.1	PASS
API Response (P95)	89ms	<100ms	PASS
ML Inference	45ms	<100ms	PASS
OCR Processing	3.4s	<5.0s	PASS

16.1 Lighthouse Audit Logs

The following logs were captured during the final pre-deployment audit using Google Lighthouse (Mobile Profile).

```
$ lighthouse https://cardiodetect.internal --view
  ChromeMobile  |  Emulated Moto G4  |  Network: Fast 3G

  ■  Performance: 98/100
  ---------------------------------------------------
  First Contentful Paint ........... 0.8 s
  Speed Index ...................... 1.1 s
  Largest Contentful Paint ......... 1.2 s
  Total Blocking Time .............. 40 ms

  ■  Accessibility: 100/100
  ■  Best Practices: 100/100
  ■  SEO: 100/100

  32 requests • 452 KiB transferred • Finish: 1.4s
```

— END OF TECHNICAL REPORT —