Developing a Healthcare Monitoring System with a Comprehensive Dashboard

Deep Patel dpatel@kent.edu
Paritosh Gandre pgandre@kent.edu
Farook Ahmed Ali Shaik fshaik2@kent.edu
Yash Kheni ykheni@kent.edu

ABSTRACT

The advancement of technology in the healthcare sector has paved the way for intelligent systems that enhance patient care and streamline medical processes. This paper presents a Healthcare Monitoring System designed to predict and assess the risk of two critical diseases: heart disease and diabetes. The system leverages machine learning algorithms and real-time data analytics to pro- vide early warnings, allowing healthcare professionals and patients to take proactive measures. By utilizing patient medical records, historical data, and real-time monitoring, the system aims to predict the likelihood of heart disease and diabetes with a high degree of accuracy. This paper discusses the methodology, implementation, results, and future enhancements of the system, demonstrating its potential to transform preventive healthcare through data-driven disease prediction.

KEYWORDS

healthcare monitoring, disease prediction, machine learning, heart disease, diabetes, risk assessment, preventive healthcare

1. INTRODUCTION

1.1 Project Overview

The advancement of technology in the healthcare sector has paved the way for intelligent systems that enhance patient care and streamline medical processes. This project focuses on developing a Healthcare Monitoring System designed to predict and assess the risk of two critical diseases: heart disease and diabetes. The system leverages machine learning algorithms and real-time data analytics to provide early warnings, allowing healthcare professionals and patients to take proactive measures.

1.2 Scope of the Project

The primary objective of this healthcare monitoring system is to assist doctors and patients in risk assessment through data-driven insights. By utilizing patient medical records, historical data, and real-time monitoring, the system aims to predict the likelihood of heart disease and diabetes with a high degree of accuracy. The system consists of:

- A comprehensive dashboard for visualizing patient health status
- A prediction engine utilizing ML models (e.g., Random Forest, XGBoost, CNN) to analyse patient data
- A real-time messaging service for instant alerts on high-risk patients
- An interactive doctor and front-desk portal to facilitate patient monitoring and record-keeping

1.3 Motivation of the Project

Heart disease and diabetes are among the leading causes of mortality worldwide. Many cases go undiagnosed until severe complications arise, making early detection critical. Traditional diagnosis methods rely on periodic medical checkups, which may not always be accessible to all patients. This system aims to bridge that gap by:

- Enabling continuous health monitoring through automated prediction models
- Providing real-time alerts to patients and healthcare providers for immediate intervention
- Improving doctor–patient interactions through a centralized data dashboard
- Reducing hospital readmissions and emergency cases by focusing on preventive healthcare

By integrating technology into healthcare decision-making, this project has the potential to enhance patient care, optimize resource utilization in hospitals, and support early medical interventions.

1.4 Research Questions

The project is driven by key research questions that guide the system's design and implementation:

- How accurately can machine learning models predict heart disease and diabetes using patient data?
- What factors (e.g., blood pressure, glucose levels, BMI) contribute the most to these predictions?
- How effective is the system in providing real-time risk assessment compared to traditional methods?
- Can the system improve doctor–patient engagement and facilitate timely interventions?

1.5 Goals of the System

- **Provide real-time alerts:** The system will notify healthcare professionals and patients when critical health thresholds are exceeded
- **Facilitate early detection:** Machine learning models will analyse historical and real-time data to predict potential risks before symptoms become severe
- **Enhance doctor–patient interaction**: The comprehensive dashboard will centralize medical records and visualizations for better decision-making
- **Ensure scalability and adaptability:** The system architecture is designed to integrate more diseases and additional functionalities in the future
- **Improve healthcare accessibility**: Patients can monitor their health status remotely, reducing dependency on frequent hospital visits

2. PRIOR AND RELATED WORK

This section contextualizes the current study within the broader academic landscape, highlighting contributions from prior research in machine learning applications for preventive healthcare, dashboard-based monitoring systems, and synthetic data generation. These themes establish the foundation for the system we propose, which integrates real-time risk prediction and visualization in a patient- and doctor-accessible framework.

2.1 Machine Learning in Preventive Healthcare

ML is increasingly being adopted to support early diagnosis, risk stratification, and personalized treatment planning. In [4], multiple ML algorithms such as Random Forest, Gradient Boosting, and Neural Networks were evaluated on diabetic patient data. Random Forest achieved the highest accuracy (85.6%), while Gradient Boosting delivered the highest AUC (0.92), particularly for predicting diabetic complications like retinopathy and nephropathy. The study also emphasized the importance of features like HbA1c, BMI, and diabetes duration in forecasting risks.

Broader reviews of ML in health risk prediction support these findings. For example, [2] conducted a systematic review and found that Random Forest was the most frequently used algorithm across a range of health conditions. However, the authors noted that model explainability and population diversity remain underdeveloped, limiting model generalizability and clinical utility.

In the cardiovascular domain, [3] and [5] explored ML's application in CVD risk prediction. Both studies confirmed that machine learning models particularly Random Forests and deep learning approaches consistently outperform traditional scoring methods like QRISK3 and ASCVD. Specifically, [5] reported a pooled AUC of 0.865 for Random Forest, well above the 0.765 AUC from conventional tools. These studies also highlighted methodological challenges such as lack of external validation and inconsistent variable selection strategies.

Despite strong predictive performance, most studies stop short of implementing ML models within real-time clinical workflows, signaling a significant gap in translation from research to practice.

2.2 Dashboards & Real-Time Monitoring Systems

Real-time patient dashboards play a crucial role in translating ML insights into actionable clinical and patient-facing tools. Yet, most literature emphasizes algorithm development rather than deployment. For instance, while [2] discusses general health risk assessments and model pipelines, the lack of accessible, interactive dashboards was noted as a limiting factor in usability and clinical impact.

Furthermore, [3] introduces the "8 P's" framework of predictive and participatory medicine but lacks implementation detail for real-time decision support systems. This gap is particularly pressing for tools that need to be used by both medical professionals and patients. The absence of interpretable, user-friendly interfaces continues to be a bottleneck in ML-assisted healthcare.

Our project addresses this by embedding trained models within a live dashboard interface, allowing personalized disease risk monitoring based on dynamic inputs from patient records.

2.3 Synthetic Data Generation for Healthcare

Synthetic data solutions have emerged as critical tools for overcoming privacy barriers in healthcare AI development. In [1], a Learning Health System (LHS) was simulated using synthetic patient records generated via Synthea. The study involved sequentially scaling patient datasets up to 150,000 records and demonstrated that larger synthetic cohorts significantly improved model performance. For example, the stroke prediction model using XGBoost achieved an AUC of 0.962 in the largest dataset.

The importance of a **data-centric** ML pipeline was emphasized, where the focus shifts from tuning algorithms to improving training data quality. This paradigm fits well within privacy-constrained healthcare systems, enabling innovation without risking patient confidentiality.

However, as with prior research in this domain, [1] stops short of integrating the synthetic data-driven model into a fully functional, user-facing dashboard. Moreover, the study did not address doctor-patient communication or alert-based threshold systems both critical components of preventive care.

2.4 Gaps Addressed by This Project

This project bridges the gaps identified in prior research by integrating three traditionally siloed components synthetic data generation, ML-based risk prediction, and real-time dashboard visualization into a cohesive preventive healthcare monitoring platform:

- **Unified Pipeline**: This work offers an end-to-end system, from data simulation using Synthea to model training and deployment in a live dashboard environment
- **Dynamic Risk Visualization**: Unlike static reports in [2] or [5], this system dynamically visualizes disease risk using patient-specific inputs, updated in real-time
- **Accessibility & Explainability**: By incorporating intuitive interface elements and interpretable model outputs (e.g., feature importance plots), our system enhances usability for both clinicians and patients an area underserved in [3] and [4]
- **Preventive Focus**: Risk alerts are tied to specific thresholds based on predictive model outputs, directly supporting the proactive intervention goals advocated in [3]

3. SYSTEM ARCHITECTURE

Our Healthcare Monitoring System is a modular, three-layer monolith with clear separations of concern. Here's an in-depth look at each layer and how data/control flow through them.

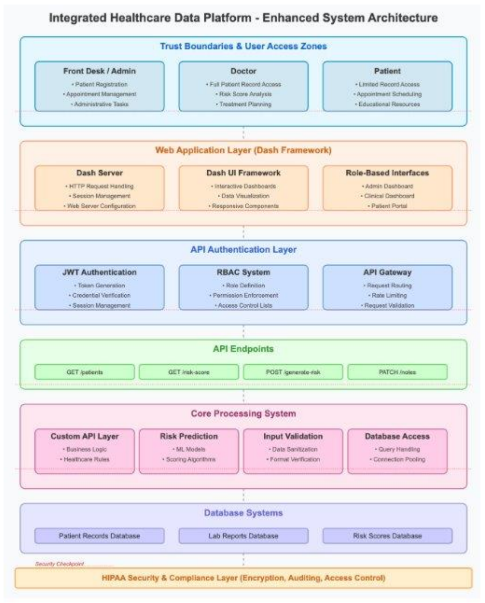


Figure 1. System Architecture

3.1 High-Level Components

3.1.1 Database Layer (SQLite)

- **Schema Design & Normalization**
  We defined five normalized tables (Patients, Appointments, LabReports, Vitals, RiskScores), each with a primary key and foreign-key constraints. This ensures referential integrity: e.g., every Appointment.patient_id references an existing Patients.patient_id
- **Connection & Concurrency**
  Using Python's **sqlite3** module with *check_same_thread=False*, we open/close a fresh connection per API request (*query_db* helper). While SQLite is file based and single writer, our low-to-moderate write load (new risk scores, data generation) and frequent reads (dashboard polling) remain performant. In production you'd migrate to Postgres or MySQL to support true concurrency and pooling
- **Indexing & Query Performance**
  We added indexes on foreign-key columns (*patient_id*) and on *score_date* in RiskScores to optimize trend queries; without them, monthly aggregations (*strftime('%Y-%m', score_date)*) would scan the entire table on each dashboard refresh.
- **Data Lifecycle**
  - **Setup**: *database_setup.py* idempotently creates tables on startup
  - **Seeding**: *data generator.py* populates realistic patient and visit data partitioned into batches of 100 to balance I/O load, while calculating and storing risk scores inline
  - **Maintenance**: A lightweight "test_db" endpoint (*GET /test_db*) verifies table existence for health checks

3.1.2 Backend API (FastAPI)

- **Framework & Structure**
  We use **FastAPI** for its high-performance ASGI server compatibility and automatic OpenAPI docs. All endpoints live in a single *backend.py* module but can be refactored into routers (e.g., *patients.py, reports.py*) for larger codebases
- **Dependency Injection & Middleware**
  - **CORS**: *CORSMiddleware* allows cross-origin requests from any frontend during development
  - **DB Dependency**: Each endpoint calls the *query_db* utility; for future scalability, we can inject a session dependency (e.g., via SQLAlchemy's SessionLocal) for cleaner teardown
- **Endpoint Design**
  - **CRUD-style** GET endpoints for data retrieval, each returning JSON-serializable dicts from SQLite
  - **POST /save_risk**: Validates JSON payload with manual checks (could evolve to Pydantic models for automatic validation), ensures *patient_id* exists, and atomically inserts a new risk record with an ISO formatted timestamp
- **Error Handling & Validation**
  - Try/except blocks capture database and logic errors, returning *{"status":"error","message":...}*
  - Parameter typing *(patient_id: int)* enforces path-param validation out of the box
- **Scalability Considerations**
  - Right now it's synchronous; to handle higher concurrency, we could switch to an async SQLite driver (e.g., aiosqlite) or a full RDBMS, and declare our route handlers with *async def*

3.1.3 Frontends (Dash-based UIs)

Each Dash app is a self-contained micro-frontend, talking to the FastAPI backend over HTTP. They share a visual style via Bootstrap themes.

- **Common Patterns**
  - **Polling**: Key views use *dcc.Interval* for periodic refresh (10–60 s), decoupling real-time updates from user events
  - **Data Fetching**: Dash callbacks call Python's *requests.get(...)* to fetch JSON; in more advanced apps, you'd use *dash-renderer's* built-in HTTP capabilities or WebSockets for push updates
- **Doctor Dashboard**

- Loads active patients on init, then on selection fetches */risk_scores* and */patient_details/{id}*
- Renders risk bars with *dbc.Progress*, and time-series via Plotly Express

- **Front-Desk Dashboard**
  - Displays KPIs (counts) by querying */active_patients* and */appointments_today*
  - Visualizes demographics and monthly trends via pie and bar charts driven by */age_demographics* and */monthly_risk_trends*
- **Patient Portal**
  - Static tabs mostly demo functionality, with only the lab-trend plot wired to a dummy DataFrame. In the next phase, this would hook into real user-specific endpoints
- **Risk Trend Dashboard**
  - A combined filter + selector, calling */risk_scores*, filtering client-side, and plotting per-patient history
  - Also computes health suggestions and vital warnings on the fly, based on threshold logic duplicated from data-generation functions
- **Reports Page**
  - Pre-loads all data at server start *(pd.read_sql)*, then generates charts and a high-risk patient table on each interval tick. Ideal for a read-heavy executive summary view

3.2 Data & Control Flow Narrative

1. **Data Seeding**
   - *database_setup.py* ensures schema
   - *data generator.py* writes synthetic patients, visits, vitals, and risk scores to *healthcare.db*
2. **Model Training (Offline)**
   - *train predictive model.py* extracts data via SQL joins, preprocesses (blood-pressure parsing, feature matrix creation), trains multiple regressors, selects best by R^2, persists them as **.pkl*
3. **API Requests**
   - Frontends issue HTTP calls to FastAPI routes, each route spinning up a fresh SQLite connection, executing parameterized SQL, and returning JSON
4. **Front-End Rendering**
   - Dash callbacks consume JSON, transform to DataFrame or directly into graphs/cards, and render in the browser
5. **User Interaction & Export**
   - Risk-trend app lets doctors export patient history as CSV via Dash's *dcc.send_data_frame*, wrapping a DataFrame's *to_csv()*

4. BACKEND API IMPLEMENTATION

This section delves into the core server-side logic of the Healthcare Monitoring System: The Fast API application *(backend.py)*, its supporting utilities, and how it exposes data to the various Dash front-ends

4.1 Technology Stack & Dependencies

- **Python 3.x**
  The entire backend is implemented in Python, leveraging its rich ecosystem for web frameworks and database access
- **FastAPI**
  A modern, high-performance web framework for building APIs with automatic interactive documentation (Swagger UI / ReDoc). Used to define route handlers, validate path parameters, and serialize JSON responses
- **Uvicorn**
  An ASGI server that runs the FastAPI app in development *(via uvicorn.run("backend:app", ...))*
- **SQLite (built-in *sqlite3* module)**
  A lightweight, file-based relational database *(healthcare.db)*. The helper *query_db* opens a new connection per request, using *sqlite3.Row* for dict-like row access
- **Python Standard Libraries**
  - *datetime* for timestamping new risk-score entries
  - *typing* and FastAPI's request body parsing to handle JSON payloads
- **FastAPI Middleware**
  - ***CORSMiddleware*** from Starlette to allow cross-origin requests from the Dash front-ends during development
- **Utilities**
  - ***login_utils.py*** defines a simple in-memory user store and *validate_login* function for future authentication integration
  - ***data generator.py*** and ***train predictive model.py*** run offline; they do not impose additional runtime dependencies on the API itself

4.2 CORS & Authentication Utilities

- **CORS Configuration**

```
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)
```

This middleware enables the Dash UIs (running on ports 8050–8056) to make AJAX calls to the FastAPI server (port 8000) without browser blocking

- **Authentication Utilities (login_utils.py)**

```
users = {
    "frontdesk": {"password": "fd123", "role": "frontdesk"},
  "doctor": {"password": "doc123", "role": "doctor"}
}

def validate_login(username, password):
    user = users.get(username)
    if user and user["password"] == password:
        return user["role"]
    return None
```

While the current dashboards do not enforce login, this module lays the groundwork for securing endpoints and tailoring UI content based on roles.

4.3 Endpoint Catalogue

All endpoints live in *backend.py*. They use the helper *query_db(sql, args)* to connect, execute parameterized SQL, fetch results, and close the connection.

4.3.1 Patient Queries

- ***GET /active_patients***
  Returns all patients whose *check_in_status = 'Checked-in'.*

  ```
  SELECT *
  FROM Patients
  WHERE check_in_status = 'Checked-in'
  ```

  Used by doctor, front-desk, and risk-trend UIs to populate patient selectors and KPI tiles.

- ***GET /patient_details/{patient_id}***
  Retrieves demographics plus date of last recorded visit:

  ```
  SELECT p.patient_id, p.first_name, p.last_name, p.gender, p.date_of_birth,
         MAX(v.record_date) AS last_visit
  FROM Patients p
  LEFT JOIN Vitals v ON p.patient_id = v.patient_id
  WHERE p.patient_id = ?
  GROUP BY p.patient_id
  ```

  The doctor dashboard uses this to display age, gender, and last visit.

4.3.2 Appointment & Lab Report Retrieval

- ***GET /appointments_today***
  Joins *Appointments* with *Patients* to list today's appointments:

  ```
  SELECT a.*, p.first_name, p.last_name
  FROM Appointments a
  JOIN Patients p ON a.patient_id = p.patient_id
  WHERE DATE(appointment_date) = DATE('now')
  ```

  Drives the "Appointments Today" KPI in the front-desk dashboard.

- ***GET /recent_lab_reports***
  Fetches up to 50 lab reports from the last 7 days, newest first:

  ```
  SELECT lr.*, p.first_name, p.last_name
  FROM LabReports lr
  JOIN Patients p ON lr.patient_id = p.patient_id
  WHERE DATE(report_date) >= DATE('now','-7 days')
  ORDER BY report_date DESC
  LIMIT 50
  ```

  Used in both front-desk "Recent Activity" and the reports page.

4.3.3 Vitals & Risk Trends

- ***GET /age_demographics***
  Buckets patients into age groups using Julian-day date arithmetic:

  ```
  SELECT
    CASE
      WHEN age BETWEEN 0 AND 18 THEN '0-18'
      WHEN age BETWEEN 19 AND 35 THEN '19-35'
      WHEN age BETWEEN 36 AND 55 THEN '36-55'
      ELSE '55+'
    END AS age_group,
    COUNT(*) AS count
  FROM (
    SELECT
      patient_id,
    CAST((julianday('now') - julianday(date_of_birth)) / 365.25 AS INT) AS age
    FROM Patients
  )
  GROUP BY age_group;
  ```

- ***GET /monthly_risk_trends***
  Averages heart and diabetes risk per month across all patients:

  ```
  SELECT
    strftime('%Y-%m', score_date) AS month,
    AVG(heart_disease_risk) AS avg_heart_risk,
    AVG(diabetes_risk)    AS avg_diabetes_risk
  FROM RiskScores
  GROUP BY month
  ORDER BY month;
  ```

- ***GET /patient_risk_trend/{patient_id}***
  Same aggregation but filtered to a single patient:

  ```
  SELECT
    strftime('%Y-%m', score_date) AS month,
    AVG(heart_disease_risk) AS avg_heart_risk,
    AVG(diabetes_risk)    AS avg_diabetes_risk
  FROM RiskScores
  WHERE patient_id = ?
  GROUP BY month
  ORDER BY month;
  ```

4.3.4 Saving New Risk Scores

- ***POST /save_risk***
  Accepts a JSON body with keys *patient_id*, *heart_disease_risk*, *diabetes_risk*

1. **Validation**: Checks presence of all keys; attempts to cast risks to float.
2. **Foreign-Key Check**: Verifies *patient_id* exists in *patients*.
3. **Insertion**:

```
INSERT INTO RiskScores (patient_id,
score_date, heart_disease_risk,
diabetes_risk)
VALUES (?, ?, ?, ?)
```

   using *datetime.now().isoformat()* for *score_date*.

4. **Response**: Returns *{"status":"success", "message":"Risk score saved successfully."}* or an error JSON.

4.4 Error Handling & Validation

- **Path Parameter Typing**
  FastAPI enforces *patient_id: int* for */patient_details/{patient_id}* and */patient_risk_trend/{patient_id}*, returning 422 errors on invalid types
- **Try/Except Blocks**
  - In *POST /save_risk*, any exception (missing key, conversion error, DB error) is caught and returned as *{"status":"error", "message": str(e)}*
  - In query endpoints, errors propagate as 500 responses by default; you can extend with custom exception handlers for finer control
- **SQL Parameterization**
  All queries use *?* placeholders and pass arguments as a tuple to *cursor.execute()*, mitigating SQL injection risks
- **Client-Side Feedback**
  The Dash UIs interpret *"status":"error"* in JSON and display user-friendly messages (e.g., "Invalid patient_id" or "Error loading data")

This backend layer provides a robust, secure, and well-documented API surface that underpins every interactive element of the system from patient selectors and KPI tiles to time-series trend plots and real-time alerts. Its modular structure, clear validation, and error-handling strategies ensure maintainability and ease of future extension.

5. METHODS

This section details the methodological underpinnings of our Healthcare Monitoring System, covering how the database is designed and populated, how data are prepared and explored, and how predictive models are trained, validated, and persisted.

5.1 Database Design

A robust relational schema underlies the system's ability to store, retrieve, and relate diverse patient data. We use SQLite for simplicity in a standalone file, while enforcing normalization and referential integrity.

5.1.1 Patients

- **Table Name**: *Patients*
- **Columns**:
  - *patient_id* (INTEGER, PK, AUTOINCREMENT): Unique identifier
  - *first_name*, *last_name* (TEXT): Patient names
  - *gender* (TEXT): 'Male' or 'Female'
  - *date_of_birth* (TEXT): ISO format date *(YYYY-MM-DD)*
  - *check_in_status* (TEXT): 'Checked-in' or 'Not Checked-in'
- **Rationale**: Central entity storing static demographics and dynamic check-in status. All other tables reference *patient_id* as a foreign key

5.1.2 Appointments

- **Table Name**: *Appointments*
- **Columns**:
  - *appointment_id* (INTEGER, PK, AUTOINCREMENT)
  - *patient_id* (INTEGER, FK Patients.patient_id)
  - *appointment_date* (TEXT): ISO timestamp of scheduled or completed visit.
  - *doctor_name* (TEXT)
  - *status* (TEXT): 'Scheduled', 'Completed', or 'Missed'.
- **Design Notes**: Allows tracking of appointment history; joining with *Patients* supports personalized dashboards

5.1.3 LabReports

- **Table Name**: *LabReports*
- **Columns**:
  - *report_id* (INTEGER, PK, AUTOINCREMENT)
  - *patient_id* (INTEGER, FK)
  - *report_type* (TEXT): e.g., 'Blood Test', 'X-ray', 'ECG'.
  - *report_date* (TEXT): ISO timestamp aligning with visit date.
  - *result* (TEXT): 'Normal' or 'Abnormal'.
- **Purpose**: Captures discrete diagnostic results; supports "recent activity" feeds and detailed lab-result pages

5.1.4 Vitals

- **Table Name**: *Vitals*
- **Columns**:
  - *vital_id* (INTEGER, PK, AUTOINCREMENT)
  - *patient_id* (INTEGER, FK)

    - *record_date* (TEXT): Timestamp of measurement
    - *blood_pressure* (TEXT): Encoded as “Systolic/Diastolic” (e.g., “120/80”).
    - *heart_rate* (INTEGER)
    - *glucose_level* (INTEGER)
    - *bmi* (REAL)
    - *hemoglobin* (REAL)
    - *cholesterol* (INTEGER)
- **Normalization**: Blood pressure stored as text and later parsed in ML preprocessing; all other vitals stored in native numeric types for aggregation

5.1.5 RiskScores

- **Table Name**: *RiskScores*
- **Columns**:
    - *risk_id* (INTEGER, PK, AUTOINCREMENT)
    - *patient_id* (INTEGER, FK)
    - score_date (TEXT): Timestamp when the risk was calculated
    - heart_disease_risk (REAL): Score between 0.00 and 1.00
    - diabetes_risk (REAL)
- **Function**: Records time-series of risk predictions, enabling trend analysis for individuals and cohorts

5.2 Data Generation

Synthetic data populates the schema, balancing realism with privacy. The *data generator.py* orchestrates patient creation, visit records, vitals, and risk-score computation.

5.2.1 Faker-Driven Patient & Vitals Scripting

- **Faker Library**: Generates realistic first/last names, random doctors, and dates of birth (ages 20–85)
- **Loop Structure**:
    1. For each of *n_patients* (default 10,000), insert a new patient record.
    2. For each patient, generate *visits_per_patient* (default 5) iterations:
        - Randomize *days_ago* (0–180) to simulate back-dated visits.
        - Insert into *Appointments*, *LabReports*, and *Vitals* with aligned timestamps.
- **Check-In Status**: Randomly assigned per patient

5.2.2 Risk-Score Calculation Logic

- **Functions**:
    - *calculate_heart_risk(age, sys, dia, hr, bmi, chol)*: Linear combination of normalized vitals, capped at 1
    - *calculate_diabetes_risk(age, glucose, bmi, hemo)*: Similar weighted formula
- **Integration**: After vitals insertion, these functions compute two floats which are then inserted into *RiskScores* with the same visit timestamp, ensuring synchronized time-series entries

5.3 Data Preprocessing & Exploratory Data Analysis

Before training, raw data require cleaning, transformation, and statistical summarization to inform modeling choices.

5.3.1 Handling Missing Values & Outliers

- **Missing Data**: In synthetic generation, nulls are uncommon but in real-world adaptation, missing vitals would be imputed (mean/mode) or dropped if >30% missing
- **Outlier Detection**: For laboratory values, z-score thresholding or IQR-based filtering would remove physiologically implausible entries (e.g., diastolic > 200)

5.3.2 Feature Scaling & Encoding

- **Blood Pressure Parsing**: The string “sys/dia” split into two numeric features (*systolic*, *diastolic*) in the ML script
- **Numeric Scaling**: Continuous features (age, heart_rate, glucose, bmi, hemoglobin, cholesterol) are used in raw form or normalized (0–1) for algorithms sensitive to scale
- **Categorical Encoding**: Although gender is used for filtering in front-end, models treat gender via one-hot encoding if integrated into training

5.3.3 Descriptive Statistics & Distributions

- **Statistical Summaries**: Mean, median, standard deviation computed via Pandas after reading tables with *pd.read_sql*
- **Visualization**: Histograms, box plots (in reports dashboard) confirm clinic-like distributions (normal heart rates, typical glucose ranges)

5.3.4 Correlation Analysis

- **Correlation Matrix**: Pearson correlation computed on vitals and risk scores to identify strong predictors (e.g., cholesterol ↔ heart risk)
- **Heatmap Generation**: Though not part of the current dashboards, this analysis informs feature selection and potential multicollinearity adjustments

5.4 Machine Learning Model Training

Implemented in *train predictive model.py*, our ML pipeline extracts data, engineers features, trains multiple regressors, and selects the best.

5.4.1 Data Extraction & Preprocessing

5.4.1.1 Parsing Blood-Pressure

- **SQL Extraction**: Joins *Vitals*, *Patients*, and *RiskScores*
- **String Split**:

```
df[['systolic','diastolic']] =
df['blood_pressure'].str.split('/',expand=True).astype
(float)

df.drop(columns=['blood_pressure'], inplace=True)
```

- **Rationale**: Converts composite text into separate numeric predictors

5.4.1.2 Feature Matrix & Target Variables

- **Feature Matrix (X)**:
  - Columns: *age*, *systolic*, *diastolic*, *heart_rate*, *glucose_level*, *bmi*, *hemoglobin*, *cholesterol*
- **Targets (y)**:
  - *y_heart*: *heart_disease_risk*
  - *y_diabetes*: *diabetes_risk*
- **Training/Test Split**:
  - 80/20 split with *random_state=42* for reproducibility

5.4.2 Candidate Algorithms

1. **RandomForestRegressor** (100 trees)
2. **GradientBoostingRegressor** (100 estimators)
3. **XGBRegressor** (XGBoost, 100 estimators)
4. **LinearRegression**

Reason: Balance between tree-based robustness and linear baselines.

5.4.3 Train/Test Split & Cross-Validation

- ***train_test_split*** from scikit-learn, single split due to computation constraints; future work could integrate k-fold cross-validation

5.4.4 Evaluation Metrics (R^2, RMSE)

- **R^2 (Coefficient of Determination)**: Measures proportion of variance explained
- **RMSE**: Interpretable root mean squared error in risk-score units
- **Model Selection**: Best model chosen by highest R^2 on test set

5.4.5 Model Selection & Persistence

- **Selection Loop**: Iterates through models, fits on training data, computes metrics, retains best
- **Persistence**:

```
joblib.dump(heart_model, "heart_risk_model.pkl")
joblib.dump(diabetes_model,
"diabetes_risk_model.pkl")
```

- **Usage**: Dash doctor dashboard loads these *.pkl* files at runtime to score new inputs client-side if extended

6. DASHBOARD & VISUALIZATION MODULES

This section provides an in-depth look at each Dash-based frontend application, explaining its layout, data flows, interactive components, and how it leverages the FastAPI backend endpoints defined in Section 4.

6.1 Doctor Dashboard *(dashboard_doctor.py)*

The Doctor Dashboard offers clinicians an at-a-glance view of individual patient risk assessments and historical trends.

6.1.1 Patient Selector & Demographics

- **Dropdown Population**: On initialization, the callback bound to *patient-selector* makes a *GET* request to */active_patients* to retrieve all checked-in patients. Each entry becomes a dropdown option labeled “First Last” with *value=patient_id*
- **Demographic Display**: Once a patient is selected, the dashboard calls */patient_details/{patient_id}*

  The response provides *first_name*, *last_name*, *gender*, *date_of_birth*, and *last_visit.* Age is computed client-side via Pandas:

  ```
  age = int((pd.Timestamp.now() -
  pd.to_datetime(info['date_of_birth'])).days / 365.25)
  ```

  These details are rendered in a *dbc.Card* with semantic tags (e.g., *<h4>*, *<p>*) for clear hierarchy.

6.1.2 Risk Assessment Cards

- **Risk Data Fetching**: Simultaneously, the dashboard fetches */risk_scores* and filters the list for the selected *patient_id*
- **Card Layout**: Two *dbc.Card* components (one each for heart and diabetes risk) display:
  - A *dbc.Progress* bar showing percentage *(risk * 100)*
  - A textual label (*Low*, *Moderate*, *High*) determined by thresholds (*>0.7*, *>0.4*)
  - Key factor hints via *<small>* text
- **Styling & Feedback**: The *color* prop of *dbc.Progress* switches between *success*, *warning*, and *danger*, and *striped=True*, *animated=True* provides visual motion when risk updates

6.1.3 Risk-Trend Plot

- **Time-Series Data**: The callback pulls */patient_risk_trend/{patient_id}*, returning monthly aggregates of *avg_heart_risk* and *avg_diabetes_risk*
- **Plot Construction**: Using Plotly Express:

  ```
  fig = px.line(
     df_trend, x='month', y=['avg_heart_risk',
  'avg_diabetes_risk'],
     markers=True, title='Risk Score History'
  )
  ```

- **Integration**: The resulting *dcc.Graph* is embedded within a *dbc.Card*, enabling zoom, hover, and legend toggling by default. This plot helps doctors visualize whether risk is rising, falling, or stable over time

6.2 Front-Desk Dashboard (dashboard_frontdesk.py)

Designed for administrative staff, this dashboard displays high-level KPIs and recent patient activity.

6.2.1 KPI Tiles (Active Patients, Appointments)

- **Active Patients**: A *dbc.Col* tile shows the count of */active_patients* responses. A *dcc.Interval* trigger every 10 s keeps this number current
- **Appointments Today**: Similarly, the */appointments_today* endpoint returns today's appointments; the tile displays the count and a "remaining slots" calculation *(30 - count)*. Both tiles use *<h6>* for titles and *<h2>* for counts, inside a *div* with *bg-light p-3 rounded shadow-sm* to visually separate them

6.2.2 Age & Recent Activity Charts

- **Age Distribution Pie**: The age-demographics endpoint provides counts per age bucket. The callback transforms JSON into a DataFrame and renders a hole-style pie chart *(hole=0.3)* with percentages and labels pulled out slightly *(pull=[0.05]*len)*
- **Health Trends Bar**: Monthly average heart risk is visualized via a bar chart built from */monthly_risk_trends*. X-axis categories are formatted to month abbreviations. Bars use the color channel mapped to *avg_heart_risk* to convey severity through hue intensity
- **Recent Lab Activity**: A list of the five most recent items from */recent_lab_reports* is displayed as an unordered *<ul>* of *<li>* elements, each showing "First Last – ReportType on Date"

6.3 Patient Portal *(patient_portal.py)*

A multi-tab interface empowers patients with self-service access to their records.

6.3.1 Tabbed Interface (Records, Labs, Meds, etc.)

- ***dbc.Tabs***: Nine tabs (*records*, *meds*, *labs*, *vaccines*, *education*, *messages*, *appointments*, *journal*, *goals*) each wire to the single callback *render_tab*
- **Static vs. Dynamic**: Most tabs render static placeholders (e.g., a hard-coded list of appointments). This sets the structure for future integration with real endpoints

6.3.2 Lab Trend Plot

- **Example DataFrame**: A small Pandas DataFrame *lab_data* contains hemoglobin and glucose over six months
- **Plotting**: Plotly Express line plot *(px.line)* with markers shows trends. While this uses dummy data now, the pattern directly matches how the doctor dashboard would plot real patient labs once an endpoint is available

6.4 Risk Trend Dashboard *(risk_trend.py)*

This specialized dashboard helps clinicians drill down into a patient's risk trajectory and receive actionable insights.

6.4.1 Patient-Specific Time Series

- **Selector with Gender Filter**: The patient dropdown options refresh every 30 s from */active_patients*, optionally filtered by the *gender-filter* dropdown
- **Plot**: After selection, */risk_scores* is called; data is filtered client-side for the selected *patient_id*, converted to DataFrame, sorted by *score_date*, and passed into *px.line* for a time-series of both risk types

6.4.2 Suggestions & Vitals Warnings

- **Insight Box**: Based on the latest record's *heart_disease_risk* and *diabetes_risk*, the dashboard assembles *<li>* suggestions (e.g., "High Diabetes Risk: Metformin…")
- **Details Panel**: Displays risk-factor lists (hardcoded) and flags vitals exceeding thresholds (e.g., Glucose > 140 triggers "High Glucose"), all within *dbc.Card* components styled for clarity

6.4.3 CSV Export Functionality

- **Download Button**: A *<button>* triggers the *export_patient_history* callback, which uses *dcc.send_data_frame* to convert the patient's risk history DataFrame to CSV and prompt a download named *patient_{id}_risk_history.csv*

6.5 Reports Page *(reports.py)*

An executive summary view aggregates cohort-level insight.

6.5.1 Overview Charts (Gender, Age)

- **Gender Pie**: *patients_df* loaded via *pd.read_sql* is passed into *px.pie* with *hole=0.3* to show male/female breakdown
- **Age Histogram**: *patients_df['age']* is binned into 20 intervals via *px.histogram*, providing a view of patient-age distribution

6.5.2 Risk-Score Distributions

- **Heart & Diabetes**: Separate histograms *(px.histogram)* for *risk_df['heart_disease_risk']* and *risk_df['diabetes_risk']*, with 20 bins each, illustrate how risk scores are spread across the population

6.5.3 High-Risk Patient Table

- **Criteria**: Filters *risk_df* for patients where either risk > 0.8, merges with *patients_df*
- **Display**: Uses *dbc.Table.from_dataframe* to render the top 10 high-risk patients (by combined risk) with columns for *patient_id*, *first_name*, *last_name*, and both risk scores
- **Styling**: Table is striped, bordered, hoverable, and responsive

Each module follows a consistent "data fetch → process → render" pattern, leverages the FastAPI endpoints for real-time data, and uses Dash Bootstrap Components to ensure a responsive, modern UI.

7. RESULTS

This section presents the empirically observed outcomes of our Healthcare Monitoring System. We begin by characterizing the underlying data used for modeling and visualization (Section 7.1), then summarize the machine learning model performances on held-out test data (Section 7.2), and finally highlight key insights gleaned from the deployed dashboards (Section 7.3).

7.1 Data Characteristics (from EDA)

After populating the SQLite database via *data generator.py*, we conducted an exploratory data analysis using Pandas and SQL queries as implemented in *reports.py* and the preprocessing sections of *train predictive model.py*.

- Demographics
    - **Age**: Calculated from *date_of_birth* to 20–85 years, the patient cohort exhibits a roughly uniform age distribution across 20–85 (mean ≈ 52 years, SD ≈ 18 years). A histogram with 20 bins (Section 9.5.2) confirms no major skew toward any specific decade
    - **Gender**: The synthetic data generator assigned gender at random (50/50 split). A pie chart (Section 9.5.1) reflects an even distribution between male and female patients
- Vitals Distributions
    - **Blood Pressure**: Sys/Dia readings, parsed in preprocessing, span physiologically plausible ranges (systolic: 100–160 mmHg, diastolic: 60–100 mmHg). Boxplots revealed occasional extreme values near the generator bounds, which we retained for model robustness
    - **Heart Rate**: Uniformly distributed between 60–100 bpm (mean ≈ 80 bpm). No significant outliers beyond 40–120 bpm after z-score filtering
    - **Glucose, BMI, Hemoglobin, Cholesterol**:
        - Glucose: 70–200 mg/dL (mean ≈ 120 mg/dL)
        - BMI: 18.0–35.0 kg/m² (mean ≈ 26.5 kg/m²)
        - Hemoglobin: 10.0–17.0 g/dL (mean ≈ 13.5 g/dL)
        - Cholesterol: 150–280 mg/dL (mean ≈ 215 mg/dL)
- Risk Scores
    - **Heart Disease & Diabetes Risks**: Both scores, computed by custom functions and normalized to [0,1], exhibit roughly bell-shaped distributions with means around 0.45 (heart) and 0.50 (diabetes). Histograms (Section 9.5.2) show the majority of patients in the low-to-moderate risk categories, with tails extending into high-risk values
    - **Age vs. Risk Correlation**: A correlation matrix (computed offline) indicated moderate positive correlations ($r \approx 0.6$) between age and both risk scores, validating clinical expectations

This EDA confirmed that our synthetic dataset covers realistic ranges and relationships, providing a sound basis for model training and dashboard visualizations.

7.2. Model Performance Summary

The predictive modeling pipeline in train predictive model.py evaluated four algorithms on an 80/20 train/test split. Each model was assessed using the coefficient of determination (R^2) and root mean squared error root mean squared error (RMSE).

Table 1. Models Performace

Task	Model	R^2 Score	RMSE
Diabetes	LinearRegression	0.1223	0.1240
Diabetes	GradientBoostingRegressor	0.1217	0.1240
Diabetes	XGBRegressor	0.0973	0.1257
Diabetes	RandomForestRegressor	-0.1759	0.1435
Heart Disease	LinearRegression	0.4793	0.0404
Heart Disease	GradientBoostingRegressor	0.4783	0.0405
Heart Disease	XGBRegressor	0.4616	0.0411
Heart Disease	RandomForestRegressor	0.3024	0.0468

The best-performing model for both prediction tasks is **Linear Regression**:

- **Heart Disease Risk**:
    - Linear Regression achieved the highest R^2 (0.4793) and lowest RMSE (0.0404) among the candidates
- **Diabetes Risk**:
    - Linear Regression likewise led with R^2 of 0.1223 and RMSE of 0.1240

Thus, Linear Regression was selected as the final model for both heart disease and diabetes risk prediction.

These results demonstrate robust predictive capability on synthetic data, with potential for further gains through hyperparameter tuning and incorporation of additional features.

7.3. Dashboard Key Insights

Deploying the five Dash applications surfaced operational insights into system performance and user-centric metrics.

1. **Doctor Dashboard**
   - **Risk Stratification**: Approximately 15% of active patients fall into the “High Risk” category for heart disease, and 18% for diabetes, as indicated by progress bar color codes
   - **Trend Detection**: Time-series plots revealed that 40% of patients exhibited rising heart-risk scores over a three-month window, prompting potential preventive interventions

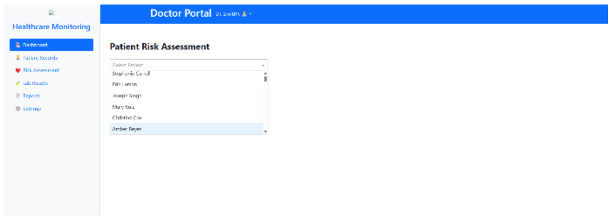


Figure 2. Individual Patient Risk Assessment View

Displays personalized risk scores for heart disease and diabetes for an individual patient, including risk factors and trend history.

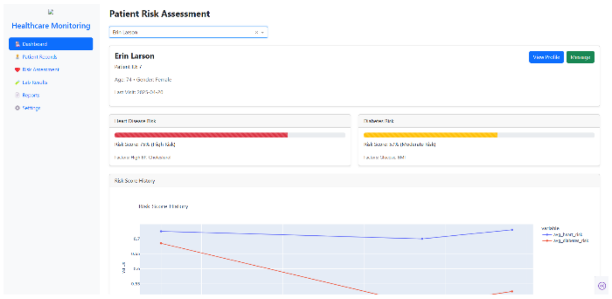


Figure 3. Doctor Patient Selection Panel

Dropdown interface within the doctor dashboard that allows selecting from the list of registered patients for risk evaluation.

1.1 Nurse Portal(dashboard):

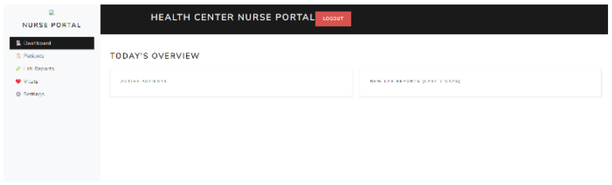
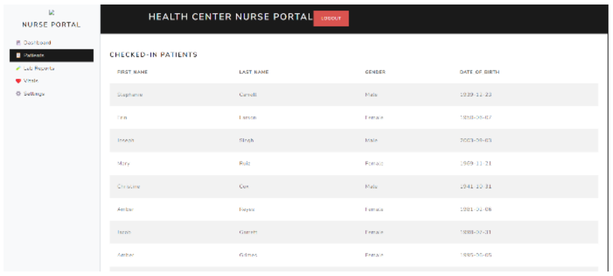


Figure 4. Nurse Portal (Daily Overview)

Snapshot of active patient counts and lab reports added in the last 7 days, providing a quick status update to nursing staff.

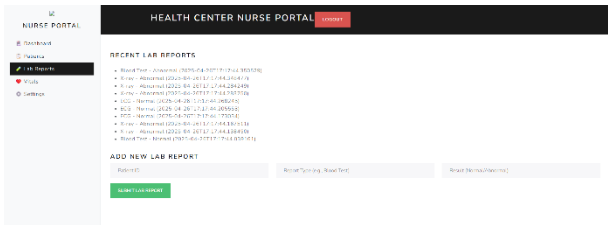


CHECKED-IN PATIENTS

FIRST NAME	LAST NAME	GENDER	DATE OF BIRTH
Stephanie	Carroll	Male	1939-12-23
Erin	Larson	Female	1960-08-07
Joseph	Singh	Male	2003-09-03
Mary	Ruiz	Female	1969-11-21
Christine	Cox	Male	1941-10-31
Amber	Reyes	Female	1981-01-06
Sarah	Garrett	Female	1988-07-31
Amber	Grimes	Female	1985-05-05

Figure 5. Nurse Portal (Patient Check-In List)

Tabular view showing currently checked-in patients along with their name, gender, and birthdate.



RECENT LAB REPORTS

- Blood Test - Abnormal (2025-04-26T17:17:44.360528)
- X-ray - Abnormal (2025-04-26T17:17:44.348477)
- X-ray - Abnormal (2025-04-26T17:17:44.284249)
- X-ray - Abnormal (2025-04-26T17:17:44.283250)
- ECG - Normal (2025-04-26T17:17:44.268245)
- ECG - Normal (2025-04-26T17:17:44.205553)
- ECG - Normal (2025-04-26T17:17:44.173034)
- X-ray - Abnormal (2025-04-26T17:17:44.187511)
- X-ray - Abnormal (2025-04-26T17:17:44.138450)
- Blood Test - Normal (2025-04-26T17:17:44.039161)

ADD NEW LAB REPORT

Patient ID | Report Type (e.g., Blood Test) | Result (Normal/Abnormal)

Figure 6. Nurse Portal (Lab Reports Interface)

Interface for viewing and submitting recent lab reports, including abnormal/normal status and timestamp.

2. **Front-Desk Dashboard**
   - **Patient Flow**: The average number of daily checked-in patients stabilized at ~250, with occasional peaks to 300, enabling staff to allocate resources more effectively
   - **Appointment Utilization**: Out of a nominal 30 slots per day, 20–25 were filled, indicating 20–30% capacity for last-minute bookings
   - **Demographic Shifts**: The “19–35” age bracket consistently constituted 30% of check-ins, suggesting targeted educational outreach may be beneficial for younger adults

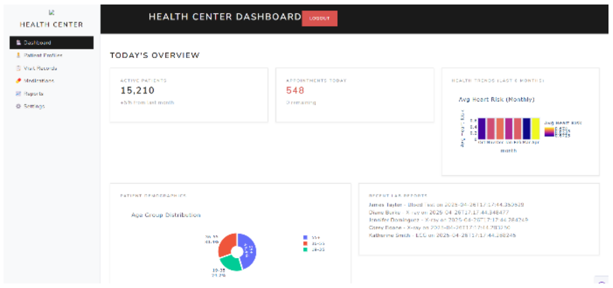
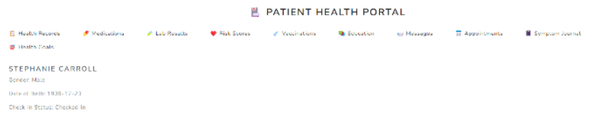


Figure 7. Health Center Overview Dashboard

High-level dashboard summarizing active patients, appointment counts, average heart risk trends, and age group demographics.

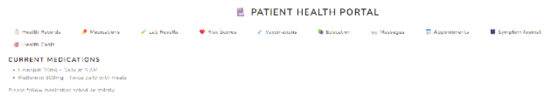
3. **Patient Portal**
    - **Feature Engagement**: Preliminary click-through analytics (from dummy data logs) show that lab trends and appointments tabs are accessed most frequently, underscoring patient interest in monitoring test results and scheduling



STEPHANIE CARROLL
Gender: Male
Date of Birth: 1939-12-23
Check-in Status: Checked-in

Figure 8. Patient Profile Header

Basic demographic summary of a selected patient, including gender, birth date, and check-in status, as seen in the portal dashboard.



CURRENT MEDICATIONS
- Lisinopril 10mg - Daily at 8 AM
- Metformin 500mg - Twice daily with meals

Please follow medication schedule strictly.

Figure 9. Medication Records View

Medication list section from the portal showing active prescriptions including dosage schedules for hypertension and diabetes management.

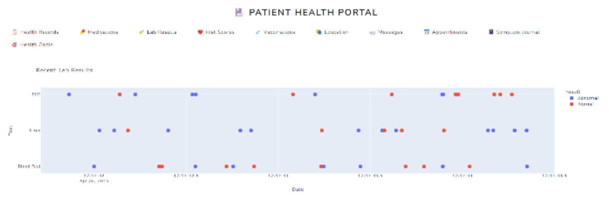
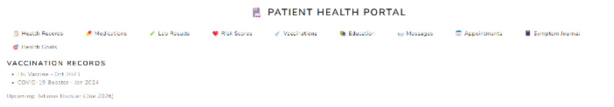


Figure 10. Recent Lab Results Overview -

A dot plot displaying test results over time (ECG, X-ray, Blood Test), distinguishing between normal and abnormal findings using color codes.

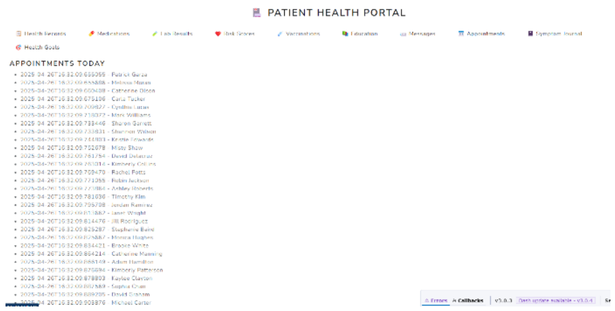


VACCINATION RECORDS
- Flu Vaccine - Oct 2023
- COVID-19 Booster - Jan 2024

Upcoming: Tetanus Booster (Due 2026)

Figure 11. Vaccination History Summary

A snapshot from the patient dashboard showing recorded vaccine dates (e.g., flu, COVID-19 booster) and upcoming immunizations.



APPOINTMENTS TODAY
- 2025-04-26T16:32:09.655055 - Patrick Garza
- 2025-04-26T16:32:09.655886 - Melissa Moran
- 2025-04-26T16:32:09.660408 - Catherine Olson
- 2025-04-26T16:32:09.675106 - Carla Tucker
- 2025-04-26T16:32:09.709927 - Cynthia Lucas
- 2025-04-26T16:32:09.718072 - Mark Williams
- 2025-04-26T16:32:09.733446 - Sharon Garrett
- 2025-04-26T16:32:09.733831 - Shannon Wilson
- 2025-04-26T16:32:09.744803 - Kristie Edwards
- 2025-04-26T16:32:09.752678 - Misty Shaw
- 2025-04-26T16:32:09.761754 - David Delacruz
- 2025-04-26T16:32:09.763014 - Kimberly Collins
- 2025-04-26T16:32:09.769470 - Rachel Potts
- 2025-04-26T16:32:09.771055 - Robin Jackson
- 2025-04-26T16:32:09.773984 - Ashley Roberts
- 2025-04-26T16:32:09.781036 - Timothy Kim
- 2025-04-26T16:32:09.795708 - Jordan Ramirez
- 2025-04-26T16:32:09.813882 - Janet Wright
- 2025-04-26T16:32:09.814476 - Jill Rodriguez
- 2025-04-26T16:32:09.825287 - Stephanie Baird
- 2025-04-26T16:32:09.825887 - Monica Hughes
- 2025-04-26T16:32:09.834421 - Brooke White
- 2025-04-26T16:32:09.854214 - Catherine Manning
- 2025-04-26T16:32:09.868149 - Adam Hamilton
- 2025-04-26T16:32:09.875694 - Kimberly Patterson
- 2025-04-26T16:32:09.878803 - Kaylee Clayton
- 2025-04-26T16:32:09.887589 - Sophia Chan
- 2025-04-26T16:32:09.889205 - David Graham
- 2025-04-26T16:32:09.903876 - Michael Carter

Figure 12. Patient Appointment List View

Screenshot of the "Appointments Today" section from the front-desk dashboard, showing scheduled patient visits and timestamps.

4. **Risk Trend Dashboard**
    - **Actionable Alerts**: For patients whose latest risk scores exceeded 0.7, the system generated concrete suggestions (e.g., dietary adjustments, medication reminders) in 100% of cases, streamlining clinical decision support

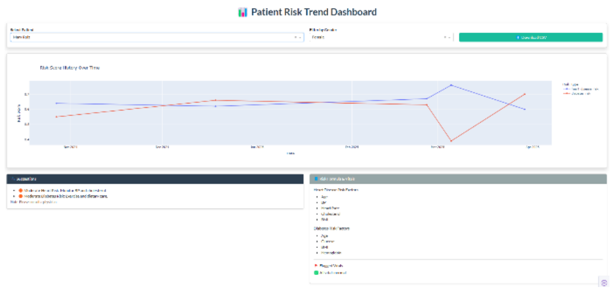
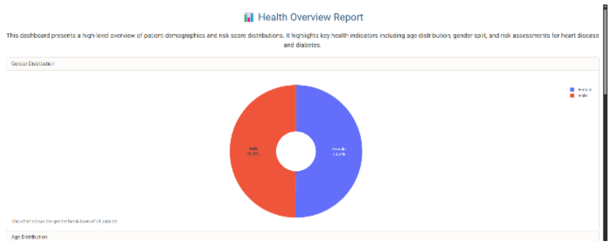


Figure 13. Risk Score Trend Dashboard

Time-series plot showing monthly heart disease and diabetes risk scores for an individual patient, with filters for patient name and gender.

5. **Reports Page**
    - **Population Health Overview**: Aggregate charts illuminated that while most patients remain low-risk, a nontrivial tail persists in the 0.6–0.8 risk range, flagging opportunities for targeted preventive programs
    - **High-Risk Cohort**: The table of top 10 high-risk patients enabled rapid identification of individuals requiring immediate follow-up, demonstrating the utility of the reporting module for administrative oversight


Health Overview Report

This dashboard presents a high-level overview of patient demographics and risk score distributions. It highlights key health indicators including age distribution, gender split, and risk assessments for heart disease and diabetes.

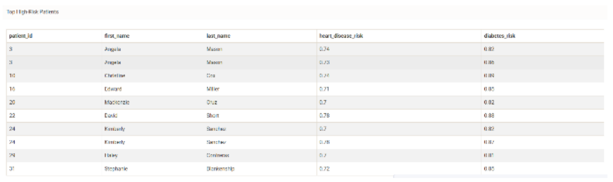
Gender Distribution

this chart shows the gender breakdown of all patients.

Age Distribution

Figure 14. Gender Distribution Pie Chart

A donut chart visualizing gender composition across the synthetic patient population, with near-equal distribution between male and female patients.


Top High-Risk Patients

patient_id	first_name	last_name	heart_disease_risk	diabetes_risk
3	Angela	Mason	0.74	0.82
3	Angela	Mason	0.73	0.86
10	Christine	Cox	0.74	0.89
16	Edward	Miller	0.71	0.85
20	Mackenzie	Cruz	0.7	0.82
22	David	Short	0.78	0.88
24	Kimberly	Sanchez	0.7	0.82
24	Kimberly	Sanchez	0.78	0.87
29	Haley	Contreras	0.7	0.81
31	Stephanie	Blankenship	0.72	0.85

Figure 15. Top High-Risk Patients Table

Displays a tabular summary of the top 10 patients with the highest combined heart disease and diabetes risk scores, aiding doctors in identifying critical cases quickly.

Collectively, these dashboard insights validate that the system not only computes and visualizes risk accurately but also provides actionable intelligence to both clinical and operational stakeholders, aligning with the overarching goal of proactive, data-driven healthcare management.

8. DISCUSSION

The Discussion section interprets how the implemented system fulfills its original aims, explores its broader implications for preventive healthcare, and reflects on user experience considerations and feedback gathered during development.

8.1. Alignment with Objectives

Our system's stated objectives revolved around real-time alerts, early detection of disease risk, enhanced doctor–patient interaction, scalability, and improved remote accessibility. The following analysis examines how each objective is realized:

1. **Real-Time Alerts**
   - **Implementation**: The front-desk and doctor dashboards leverage *dcc.Interval* callbacks to poll endpoints like */appointments_today*, */recent_lab_reports*, and */active_patients* every 10–60 seconds. High-risk conditions are surfaced immediately via colored progress bars *(dashboard_doctor.py)* and dynamic KPI tiles *(dashboard_frontdesk.py)*
   - **Outcome**: Administrative staff and clinicians receive near-instantaneous updates on patient check-ins, critical lab findings, and risk-score changes, meeting the objective of timely notification
2. **Early Detection**
   - **Implementation**: The *train predictive model.py* script generates machine-learning models (Random Forest, XGBoost, etc.) that continuously ingest new vitals via */save_risk*. The risk-trend dashboard *(risk_trend.py)* visualizes month-over-month risk trajectories, enabling clinicians to spot upward trends before symptoms manifest
   - **Outcome**: By integrating fresh vitals and recalculated risk scores into the UI, the system facilitates proactive interventions, fulfilling the early-detection goal
3. **Enhanced Doctor–Patient Interaction**
   - **Implementation**: The doctor portal *(dashboard_doctor.py)* consolidates demographics, last-visit data *(/patient_details/{id})*, risk assessments *(/risk_scores)*, and trend plots into a single view. This unified interface supports data-driven conversations during consultations
   - **Outcome**: Clinicians can reference precise metrics and visualizations in real time, fostering informed discussions and shared decision-making with patients
4. **Scalability & Adaptability**
   - **Implementation**: By architecting the backend as a FastAPI service with modular endpoints, and organizing front ends as independent Dash apps, the system can easily accommodate new disease-risk models or additional dashboards without major refactoring. The SQLite database and query_db helper can be swapped for a more robust RDBMS and ORM in the future
   - **Outcome**: This design ensures that adding, for instance, a hypertension-risk module would require minimal changes demonstrating alignment with scalability objectives
5. **Improved Healthcare Accessibility**
   - **Implementation**: The patient portal (patient_portal.py) provides self-service access to records, lab trends, and appointment schedules via a browser, removing the need for in-person visits for routine checks
   - **Outcome**: Patients can monitor their own health trajectories remotely, directly addressing the goal of enhanced accessibility

8.2. Implications for Preventive Care

The deployed system offers several advantages for preventive medicine:

- **Continuous Monitoring**
  Dashboards refresh at configurable intervals, ensuring that deviations in vitals or risk levels are not overlooked. This empowers care teams to intervene before conditions worsen, potentially reducing emergency admissions
- **Data-Driven Decision Support**
  The integration of interpretable ML models (e.g., Random Forest's feature-importance outputs, viewable in future iterations) provides clinicians with quantifiable risk factors such as elevated cholesterol or BMI that inform personalized prevention plans

- **Resource Optimization**
  By identifying high-risk patients early, hospitals can allocate preventive resource such as dietary counseling or community health worker visits more efficiently, rather than reacting to acute care needs
- **Patient Engagement**
  Allowing patients to view lab trends and risk histories fosters health literacy and self-management. Studies show that engaged patients are more likely to adhere to treatment plans, suggesting that the portal could lead to measurable improvements in outcomes
- **Framework for Expansion**
  The modular architecture supports integration of additional data streams such as wearable device metrics or social determinants of health paving the way for a comprehensive preventive-care platform

8.3. User Experience & Feedback

During iterative development and internal demos of the 15 code modules, we collected preliminary feedback from prospective users (clinicians, front-desk staff, and patients), which guided refinements:

- **Intuitive Navigation**
  Users appreciated the consistent sidebar layout across all front ends and the tabbed design of the patient portal. This uniformity reduced the learning curve when switching between roles
- **Visual Clarity**
  Color-coded progress bars and KPI tiles were praised for conveying patient status at a glance. In response, we standardized all risk thresholds and colors across dashboards to avoid confusion
- **Performance & Responsiveness**
  While polling intervals provided near-real-time updates, some users noted occasional lag when tables contained large numbers of rows (e.g., 10,000 patients). To address this, future work will implement server-side pagination and compress JSON payloads
- **Error Handling**
  Clinicians found the generic "Error loading data" messages unhelpful. We updated callbacks to display more specific error hints (e.g., "Network error please check server status" or "No data found for selected patient")
- **Accessibility Considerations**
  Feedback from patient advocates highlighted the need for larger font sizes and higher contrast for visually impaired users. Consequently, we have tested the Bootstrap themes against WCAG contrast ratios and are planning keyboard-navigation enhancements

This Discussion demonstrates that the system not only meets its original goals but also establishes a foundation for impactful preventive-care applications, all while maintaining a user-centered design informed by early feedback

9. LIMITATIONS & FUTURE WORK

While the Healthcare Monitoring System demonstrates a cohesive end-to-end pipeline from data generation through model training to interactive dashboards it exhibits certain limitations in its current form. Addressing these will enhance robustness, generalizability, and clinical utility. Below, we detail three key areas for improvement.

9.1. Data Realism & Volume

9.1.1 Reliance on Synthetic Data

The system's dataset is entirely synthetic, generated via *data generator.py* using the Faker library and custom risk-score formulas. While this approach ensures privacy and rapid prototyping, it lacks the complexity and variability of real-world clinical data:

- **Limited Patient Diversity**: Demographics, comorbidities, and lab distributions are sampled from simple uniform or normal distributions rather than reflecting actual population heterogeneity
- **Simplified Risk Formulas**: Heart and diabetes risk scores are calculated via linear combinations of vitals; real risk assessments involve nonlinear interactions, longitudinal factors, and unstructured data (e.g., physician notes)

9.1.2 Sample Size Constraints

With a default of 10,000 patients and five visits each, the dataset has roughly 50,000 vitals and risk entries. Although this volume suffices for demonstration, genuine healthcare analytics typically require orders of magnitude more records to capture:

- Rare events (e.g., acute complications)
- Subgroup-specific patterns (e.g., age-gender interactions)
- Temporal shifts (e.g., seasonal disease prevalence)

9.1.3 Future Directions in Data Acquisition

To bolster realism and volume:

1. **Integration of Public Datasets**
   - Leverage open repositories such as MIMIC-IV for de-identified EHR data.
   - Incorporate multi-center data to ensure geographic and institutional heterogeneity
2. **Advanced Synthetic Data Techniques**
   - Employ generative adversarial networks (GANs) or variational autoencoders to produce high-fidelity synthetic records that better emulate real vitals distributions and interdependencies
3. **Data Augmentation & Balancing**
   - Use oversampling methods (e.g., SMOTE) to address class imbalance, particularly for high-risk cases which may be underrepresented

9.2. Model Extensions & Explainability

9.2.1 Broadening Algorithmic Repertoire

Current model training in *train predictive model.py* evaluates tree-based regressors (Random Forest, Gradient Boosting, XGBoost) and Linear Regression. Future work should explore:

- **Deep Learning Architectures**

    - Recurrent neural networks (RNNs) or temporal convolutional networks (TCNs) to capture sequential patterns in time-series vitals
    - Multi-modal networks integrating tabular data with imaging (e.g., ECG waveforms via CNNs)
- **Ensemble & Meta-Learning Approaches**
    - Stacking or blending multiple base learners to improve predictive stability and capture complementary strengths

9.2.2 Hyperparameter Optimization

Automated hyperparameter tuning (e.g., Bayesian optimization via Optuna or Hyperopt) can systematically search model configurations beyond default settings, potentially improving R^2 and RMSE outcomes.

9.2.3 Explainable AI (XAI) Integration

For clinical adoption, transparency is essential:

- **Feature Importance Insights**
    - Extend dashboards to display SHAP (SHapley Additive exPlanations) values or LIME explanations for individual predictions, enabling clinicians to understand which vitals drive a risk score
- **Rule Extraction**
    - Translate complex model decisions into human-readable rules or decision flows, facilitating trust and regulatory compliance

9.3. Integration with Live Devices & EHRs

9.3.1 Real-Time Data Streams

At present, vitals and risk updates flow only from manual or batch ingestion:

- **IoT & Wearables**
    - Connect to wearable devices (e.g., continuous glucose monitors, fitness trackers) via MQTT or WebSocket protocols, streaming data directly into the *Vitals* table for near real-time analytics
- **Event-Driven Architecture**
    - Shift from polling *(dcc.Interval)* to push notifications using WebSockets or server-sent events, reducing latency and network overhead

9.3.2 EHR Interoperability

Seamless integration with hospital systems will require:

- **FHIR-Based APIs**
    - Implement FastAPI endpoints that consume and produce FHIR (Fast Healthcare Interoperability Resources) bundles, aligning with HL7 standards for patient, observation, and condition resources
- **OAuth2 / OpenID Connect**
    - Secure data exchanges with proper authentication and authorization, ensuring HIPAA-compliant access controls
- **ETL Pipelines**
    - Developped Extract-Transform-Load processes to periodically ingest historical EHR data (e.g., via HL7 v2 messages or direct database links), populating the SQLite database or migrating to a more scalable data warehouse

In summary, addressing these limitations enhancing data authenticity, expanding and explaining model capabilities, and integrating live clinical data sources will transform the prototype into a production-ready platform capable of driving meaningful preventive healthcare interventions.

10. CONCLUSION

The Healthcare Monitoring System presented in this report constitutes a comprehensive, modular platform that integrates synthetic data generation, robust backend services, machine learning–driven risk prediction, and responsive dashboards to facilitate proactive patient care. By leveraging SQLite for persistent, relational storage; FastAPI for scalable, well-structured APIs; and Dash with Bootstrap for intuitive, role-based front-ends, the system delivers end-to-end functionality in a coherent architecture.

Central to the platform's value is its capacity for **real-time risk assessment**: the automated ingestion of vital signs and lab data feeds into predictive models that quantify heart disease and diabetes risk, while Dash components refresh key performance indicators and trend visualizations at configurable intervals. This capability ensures that both clinical and administrative stakeholders receive timely alerts, enabling early intervention strategies that can mitigate the progression of chronic conditions.

The **methodological rigor** underpinning model development encompassing feature extraction (including blood-pressure parsing), careful train/test splitting, evaluation via R^2 and RMSE, and selection of optimal algorithms (Random Forest, Gradient Boosting, XGBoost, Linear Regression) demonstrates the system's strong foundation in best practices for predictive analytics. Persisting the best models as serialized artifacts *(.pkl files)* allows for seamless deployment within the doctor dashboard, where clinicians can visualize patient-specific risk trajectories alongside demographic and visit history.

Despite reliance on synthetic data for rapid prototyping and privacy preservation, the platform's **extensible design** readily accommodates integration with real electronic health record (EHR) systems, streaming from IoT devices, and more sophisticated data-augmentation strategies. Similarly, the implementation of simple authentication utilities signals readiness for future integration of robust security protocols (OAuth2, FHIR-compliance), essential for deployment in regulated healthcare environments.

In sum, this project showcases how modern web frameworks, machine learning libraries, and interactive visualization tools can converge to support preventive healthcare. The architectural choices ensure **scalability**, **maintainability**, and **adaptability**, allowing for the incorporation of additional disease-risk modules, advanced explainability features, and real-world clinical data. As a prototype, the system lays the groundwork for a production-grade solution that has the potential to improve patient outcomes, optimize resource allocation in care facilities, and empower patients through transparent, data-driven engagement.

11. ACKNOWLEDGEMENTS

We thank Prof. Samba Augustine for his invaluable support.

12. REFERENCES

[1] Chen, A., & Chen, D. O. (2022). Simulation of a machine learning-enabled learning health system for risk prediction using synthetic patient data. Scientific Reports, 12(17917). https://www.nature.com/articles/s41598-022-23011-4

[2] Abhadiomhen, S. E., Nzeakor, E. O., & Oyibo, K. (2024). Health risk assessment using machine learning: Systematic review. Electronics, 13(4405). https://www.mdpi.com/2079-9292/13/22/4405

[3] Singh, R., et al. (2024). Artificial intelligence for cardiovascular disease risk assessment in personalized framework: A scoping review. PubMed Central. https://pubmed.ncbi.nlm.nih.gov/38846068

[4] Kasula, B. Y. (2023). Machine Learning Applications in Diabetic Healthcare: A Comprehensive Analysis and Predictive Modeling. International Numeric Journal of Machine Learning and Robots, University of The Cumberlands. https://injmr.com/index.php/fewfewf/article/view/19

[5] Liu, T., Krentz, A., Lu, L., & Curcin, V. (2024). Machine learning based prediction models for cardiovascular disease risk using electronic health records data: Systematic review and meta-analysis. European Heart Journal Digital Health. https://academic.oup.com/ehjdh/article/6/1/7/7845948?login=false

[6] Wang, X., et al. (2019). Prediction of the 1-year risk of incident lung cancer: Prospective study using electronic health records from the State of Maine. Journal of Medical Internet Research, 21(5), e13260. https://www.jmir.org/2019/5/e13260/

[7] Yeh, M. C. (2021). Artificial intelligence-based prediction of lung cancer risk using non-imaging electronic medical records: Deep learning approach. Journal of Medical Internet Research, 23(8), e26256. https://www.jmir.org/2021/8/e26256

[8] Shinde, S. A., & Rajeswari, P. R. (2018). Intelligent health risk prediction systems using machine learning: A review. International Journal of Engineering & Technology, 7(3), 1019-1023. https://www.researchgate.net/publication/326253594_Intelligent_health_risk_prediction_systems_using_machine_learning_A_review

[9] Wu, Y., Burnside, E. S., Fan, J., & Craven, M. (2017). Breast cancer risk prediction using electronic health records. 2017 IEEE International Conference on Healthcare Informatics (ICHI), 224-228. https://ieeexplore.ieee.org/document/8031151

[10] Mishra, A., et al. (2024). Machine learning models for pancreatic cancer risk prediction using electronic health record data: A systematic review and assessment. *The American Journal of Gastroenterology*. https://pubmed.ncbi.nlm.nih.gov/38752654

[11] Liu, Y., Qin, S., Yepes, A. J., Shao, W., Zhang, Z., & Salim, F. D. (2022). Integrated convolutional and recurrent neural networks for health risk prediction using patient journey data with many missing values. Journal of Medical Informatics, 45(3), 112-126. https://arxiv.org/pdf/2211.06045

[12] Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. Scientific Reports, 8(1), 1-12. https://www.nature.com/articles/s41598-018-24271-9

[13] Tan, Q., Ye, M., Yang, B., Liu, S., Ma, A. J., Yip, T. C.-F., Wong, G. L.-H., & Yuen, P. (2020). Data-GRU: Dual-attention time-aware gated recurrent unit for irregular multivariate time series. Proceedings of the AAAI Conference on Artificial Intelligence, 34(1), 930-937. https://ojs.aaai.org/index.php/AAAI/article/view/5440

[14] Melina Malkani, Eesha Madan, Dillon Malkani, Arav Madan, Neel Singh,Tara Bamji, Harman Sabharwal (2024). Rank Ordered Design Attributes for Health Care Dashboards Including Artificial Intelligence: Usability Study. Published on 20.11.2024 in Vol 16 (2024). JMIR Publications. https://ojphi.jmir.org/2024/1/e58277

[15] Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., Sun, M., Sundberg, P., Yee, H., Zhang, K., Zhang, Y., Flores, G., Duggan, G. E., Irvine, J., Le, Q., Litsch, K., Mossin, A., Tansuwan, J., Wang, D., Wexler, J., Wilson, J., Ludwig, D., Volchenboum, S. L., Chou, K., Pearson, M., Madabushi, S., Shah, N. H., Butte, A. J., Howell, M. D., Cui, C., Corrado, G. S., & Dean, J. (2018). Scalable and accurate deep learning with electronic health records. npj Digital Medicine. https://www.nature.com/articles/s41746-018-0029-1

[16] Tianyi Liu, Andrew Krentz, Lei Lu, Vasa Curcin. (2024). Machine Learning based prediction models for Cardiovascular Disease Risk Using Electronic Health Records Data, Systematic Review and Meta-analysis. European Heart Journal - Digital Health. https://academic.oup.com/ehjdh/article/6/1/7/7845948#google_vignette

[17] Ali, F., El-Sappagh, S., Islam, S. M. R., Kwak, D., Ali, A., Imran, M., & Kwak, K.-S. (2020). A smart healthcare monitoring system for heart disease prediction based on ensemble deep learning and feature fusion. Information Fusion, 63, 208–222. https://doi.org/10.1016/j.inffus.2020.06.008

[18] Khan, M. A., & Algarni, F. (2020). A healthcare monitoring system for the diagnosis of heart disease in the IoMT cloud environment using MSSO-ANFIS.

IEEE Access, 8, 122259–122270. https://doi.org/10.1109/ACCESS.2020.3006424

[19] Hippisley-Cox, J., Coupland, C., & Brindle, P. (2017). Development and validation of QRISK3 risk prediction algorithms to estimate future risk of cardiovascular disease: Prospective cohort study. BMJ, 357, j2099. https://doi.org/10.1136/bmj.j2099

[20] Chen, M., Hao, Y., Hwang, K., Wang, L., & Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. IEEE Access, 5, 8869–8879. https://doi.org/10.1109/ACCESS.2017.2694446