Capstone Project 3: Dashboard Development

Student Name: Samuel olumide adebimpe

Submission Date: May 25, 2025

Heart Risk Monitor Dashboard Report

Step 1: Dashboard Requirements

Dashboard Story and Types

The Heart Risk Monitor dashboard tells the story of patient heart health risk management in a healthcare setting. It aims to help healthcare professionals monitor, analyze, and take action on heart attack risk factors across patient populations. The project consists of two complementary dashboard types:

1. Risk Monitor Dashboard (Analytical): Focused on detailed analysis of individual patient risk factors, allowing healthcare providers to identify high-risk patients, track risk distributions, and analyze trends over time.

2. Population Health Dashboard (Managerial/Tactical): Designed for healthcare administrators and department managers to understand population-level trends, resource allocation needs, and demographic patterns in heart risk.

Target Users

- Primary Users: Healthcare providers (doctors, nurses, care coordinators)

- Secondary Users: Department managers, hospital administrators

- Demographics: Medical professionals with basic to moderate technical skills, aged 30-60

Key Metrics & Features

- Patient risk distribution (High, Medium, Low)

- Vital signs trends over time

- Lab results analysis against normal ranges

- Demographic analysis of risk factors

- Department-specific risk distributions

- Key indicators by gender, age, and income brackets

Required Visualizations

- Risk distribution pie charts

- Vital signs timeline line charts

- Lab results radar charts

- Patient metrics comparison bar charts

- Health risk by income level stacked bar chart

- Diet score by age group line chart

- Exercise vs Sleep distribution scatter plot

- Smoking & Stress by gender grouped column chart

Step 2: Database Integration

Database Connection

The dashboard connects to an SQLite database containing patient health data from the previous capstone project. The connection is implemented using Python's sqlite3 library within a Flask application structure:

```python

def get\_db\_connection():

"""Get SQLite database connection"""

db\_path = 'C:/Users/samuel/Desktop/new-project-main/capstone2\_project.db'

conn = sqlite3.connect(db\_path)

conn.row\_factory = sqlite3.Row

return conn

```

SQL Queries

1. Patient Data Retrieval Query

```sql

WITH OriginalPatients AS (

SELECT patient\_ID

FROM Patient

WHERE patient\_ID <= 9651

)

SELECT

p.patient\_ID,

p.age,

p.gender,

p.income,

v.SystolicBP,

v.DiastolicBP,

v.HeartRate,

v.BMI,

l.Cholesterol,

l.BloodSugar,

r.HeartAttackRiskText as risk\_level,

r.StressLevel

FROM

Patient p

JOIN

OriginalPatients op ON p.patient\_ID = op.patient\_ID

JOIN

VitalSigns v ON p.patient\_ID = v.PatientID

JOIN

LabResults l ON p.patient\_ID = l.PatientID

JOIN

RiskAssessment r ON p.patient\_ID = r.PatientID

```

2. Risk Distribution Query

```sql

SELECT

HeartAttackRiskText,

COUNT(\*) as count

FROM

RiskAssessment

WHERE

PatientID <= 9651

GROUP BY

HeartAttackRiskText

```

3. High Risk Patients Query

```sql

WITH OriginalPatients AS (

SELECT patient\_ID

FROM Patient

WHERE patient\_ID <= 9651

)

SELECT

p.patient\_ID,

p.age,

p.gender,

v.SystolicBP,

v.DiastolicBP,

l.Cholesterol,

l.BloodSugar,

r.HeartAttackRiskText as risk\_level

FROM

Patient p

JOIN

OriginalPatients op ON p.patient\_ID = op.patient\_ID

JOIN

VitalSigns v ON p.patient\_ID = v.PatientID

JOIN

LabResults l ON p.patient\_ID = l.PatientID

JOIN

RiskAssessment r ON p.patient\_ID = r.PatientID

WHERE

r.HeartAttackRiskText = 'High'

```

4. Health Metrics by Demographics Query

```sql

WITH OriginalPatients AS (

SELECT patient\_ID

FROM Patient

WHERE patient\_ID <= 9651

)

SELECT

p.gender,

p.age,

p.income,

l.SmokingStatus,

r.StressLevel,

r.HeartAttackRiskText as risk\_level,

d.DietScore,

l.ExerciseHours,

l.SleepHours

FROM

Patient p

JOIN

OriginalPatients op ON p.patient\_ID = op.patient\_ID

JOIN

Lifestyle l ON p.patient\_ID = l.PatientID

JOIN

RiskAssessment r ON p.patient\_ID = r.PatientID

JOIN

DietaryHabits d ON p.patient\_ID = d.PatientID

```

Step 4: Dashboard Development

Front-End Implementation

The front-end is built using:

- HTML5 for structure

- CSS3 (with Bootstrap 5) for styling and responsive design

- JavaScript with modern ES6+ features

- Chart.js for visualization

- FontAwesome for icons

Back-End Implementation

The back-end is implemented using:

- Python 3.9 with Flask web framework

- SQLite for database storage

- Flask Blueprint architecture for modular routing:

- Analytical dashboard routes

- Population Health dashboard routes

- Patient records routes

- API endpoints for data retrieval

Dashboard Screenshots

Step 5: Data Visualization

Quantitative Visualization Techniques

1. Line Charts: Used for visualizing trends in vital signs over time, showing how metrics like blood pressure, heart rate, and cholesterol levels change. Also used to display the relationship between diet scores and age groups.

2. Bar Charts: Implemented to compare patient metrics against population averages, allowing for easy comparison of quantitative values across categories.

3. Radar Charts: Used for lab results analysis, displaying multiple variables simultaneously and highlighting areas where patient results deviate from normal ranges.

4. Scatter Plots: Implemented to show the relationship between exercise hours and sleep hours, with points colored by risk level to reveal correlations.

5. Gauges/Donut Charts: Used in the patient count displays to show the relative proportions of patients at each risk level with a radial visualization.

Qualitative Visualization Techniques

1. Pie Charts: Used for risk distribution, showing the proportion of patients in each risk category (high, medium, low) with distinct color coding.

2. Stacked Bar Charts: Implemented for health risk by income level, showing how qualitative risk levels are distributed across different income groups.

3. Grouped Column Charts: Used for smoking & stress by gender, showing qualitative distributions across multiple categories simultaneously.

4. Color Coding: Applied consistently throughout the dashboards to indicate risk levels (red for high, yellow for medium, green for low), enhancing the intuitive understanding of data.

5. Icons and Visual Indicators: Used to represent patient status, department categories, and action buttons, providing visual cues that complement the data.

Interactive Elements

- Date Range Filters: Allow users to view data for specific time periods (7 days, 30 days, 90 days, 6 months, 1 year, or custom range)

- Risk Level Filters: Enable filtering of patient data by risk level (high, medium, low)

- Patient Search: Allows quick lookup of specific patients by ID

- Department Filters: Filter population health data by clinical department

- Demographic Filters: Filter by age range and gender

- Interactive Charts: Hovering over chart elements displays detailed data tooltips

Step 6: User Experience (UX) and Testing

UX Considerations and Improvements

1. Consistent Color Scheme: Implemented a consistent color palette throughout both dashboards that maintains accessibility standards and provides visual cues for risk levels.

2. Modern Typography: Replaced older fonts with modern, clean sans-serif fonts (Inter and Poppins) that improve readability across devices.

3. Responsive Design: Ensured that all dashboard elements resize appropriately for different screen sizes, maintaining usability on desktops, tablets, and mobile devices.

4. Intuitive Navigation: Implemented a persistent sidebar with clear icons and labels for easy navigation between dashboard sections.

5. Streamlined Filter Interface: Consolidated filters into a collapsible sidebar that provides filtering options without cluttering the main dashboard view.

6. Visual Hierarchy: Applied proper spacing, sizing, and font weights to establish a clear visual hierarchy that guides users through the dashboard content.

7. Interactive Feedback: Added hover effects and active states to interactive elements to provide visual feedback when users interact with the dashboard.

8. Error Handling: Implemented comprehensive error handling with user-friendly messages when data cannot be loaded or when search results return no matches.

9.Performance Optimization: Optimized database queries and implemented data caching to ensure rapid dashboard loading and response times.

10. Accessibility Improvements: Ensured proper contrast ratios, semantic HTML structure, and keyboard navigation to make the dashboards accessible to users with disabilities.

Testing Results

The dashboard was tested across different browsers (Chrome, Firefox, Safari) and devices to ensure consistent functionality. Performance testing showed that the dashboard loads within 3 seconds on standard connections and remains responsive during interactions.

User feedback highlighted the intuitive nature of the navigation and the clarity of the visualizations. Minor adjustments were made to address feedback regarding filter placement and chart legends.

Conclusion

The Heart Risk Monitor and Population Health dashboards successfully meet the requirements of the capstone project, providing healthcare professionals with powerful tools for monitoring and analyzing patient heart health risks. The implementation leverages modern web technologies, efficient database integration, and thoughtful UX design to create an intuitive, informative, and actionable dashboard experience.

The analytical dashboard gives detailed patient-level insights, while the managerial dashboard provides the high-level population view needed for strategic healthcare management. Together, they form a comprehensive heart risk monitoring system that can help improve patient outcomes through data-driven decision making.