INTRODUCTION TO SOFTWARE ENGINEERING

Project report

Contributors:

Kian Azizpour
Kiana Javadpour
Maryam Karimi

Link:
https://github.com/kjavadpour/SE-final-Project.git

Introduction:

Juvenile Dermatomyositis (JDM) is a rare autoimmune disease affecting muscle strength and skin. Continuous monitoring of patient condition, lab tests, and motor function is crucial for timely medical decisions and successful treatment. This project aims to support clinicians by providing a tailored, console-based application that organizes and analyzes key medical data related to JDM patients.

The system, titled JDM Doctor Panel Application, was developed as part of an academic project to showcase software design, database integration, user interaction, and medical data analysis principles within a real-world context. We manage everything with different java classes and SQL that we will dive more into them for the rest of the report.

Background of the project:

Juvenile Dermatomyositis (JDM) is a rare autoimmune disease affecting children, characterized by muscle weakness and skin rashes. The Childhood Myositis Assessment Scale (CMAS) is a validated clinical tool used to assess muscle strength, endurance, and physical function in children with Juvenile Dermatomyositis (JDM). It consists of 14 functional tasks scored on a 0-52 scale, where higher scores indicate better muscle performance.

Biomarker Correlation:

CMAS scores are cross-checked with inflammatory biomarkers (CRP, CK, ESR) to:

- Confirm if muscle weakness is due to active inflammation.
- Detect subclinical flares (rising biomarkers before CMAS drops).

So Medical professionals often face challenges when managing extensive patient data, particularly for conditions that require multi-dimensional monitoring, such as Juvenile Dermatomyositis (JDM). While many hospital systems can be overly complex or dependent on graphical user interfaces, this project offers a simpler and more focused alternative: a

console-based doctor panel that integrates CMAS scores, lab result analysis, and clinical alerts into a lightweight Java-based application.

Designed with a focus on clinical relevance and user-friendliness, this application aims to provide a streamlined experience for healthcare providers. It also emphasizes flexibility, ensuring that it can be easily extended to accommodate future needs in healthcare software systems.

Monitoring patients requires:

- Regular CMAS assessments (to evaluate motor function).
- Frequent lab tests (CRP, CK, ESR, etc.) to track disease activity.

Problem Statement:

- Doctors need quick access to patient data.
- Manual tracking of critical lab tests is error-prone.
- Trend analysis of CMAS scores is time-consuming.

Solution:

This Java-based console application automates:

✓CMAS score analysis (average, max, min, trends).

✓Lab test monitoring (alerts for missing critical tests).
✓Measurement retrieval (for detailed lab result analysis).

Methodology:

Back-end

Development Tools & Environments:

- Language: Java (JDK 11+)
- IDE: IntelliJ IDEA
- Database: SQLite (database.db)
- Integration: SQLite driver

System Architecture:

The software architecture follows a modular layered design. It pursues three-tier architecture.

1. **Presentation Layer:**
    - DoctorPanelApp handles user interaction via console interface
    - Provides menu-driven navigation with clear visual indicators (emojis)

2. **Business Logic Layer:**
    - CMASAnalyzer for psychological assessment analysis
    - LabSummaryChecker for medical test monitoring validation
    - PatientSearcher and MeasurementViewer for data retrieval

3. **Data Access Layer:**
    - DatabaseManager centralizes database connectivity
    - SQLite database for persistent storage
    - Prepared statements for secure query execution

Database Model (SQLite):

The database consists of

- CMAS_Cleaned (stores CMAS scores with dates).
- LabResultsEN (contains lab test names and units).
- Measurement (records individual test values).
- LabResultGroup (groups related lab tests).

Key relationships:

- PatientID links to multiple LabResults
- LabResultID links to multiple Measurements
- LabResultGroupID organizes related tests

Features:

- **Trend Alerting:** The CMAS module detects abrupt changes in motor function using delta-threshold analysis, enhancing early-warning capability for deterioration.

- **Critical Checklist Validator:** Lab monitoring includes an intelligent checklist comparing test history to a predefined set of JDM-relevant markers (e.g., CRP, CK, ESR). Alerts are generated if key tests are omitted.

- **Statistical CMAS Insights:** Min/Max/Average scores and time intervals offer quantitative insight into patient trajectory. Session-by-Session Tracking: Each visit is treated as a distinct session, allowing for longitudinal patient review.

Requirements and Use-Cases:

1. **Authentication:**
    - Basic username/password validation (doctor/1234)
2. **Patient Analysis:**
    - View CMAS score trends and statistics
    - Check completeness of lab test monitoring
    - Search lab results by patient ID
    - View measurement details by lab result ID
3. **Data Validation:**
    - Identify missing critical tests
    - Detect significant score fluctuations

Now we will see the use case one by one:

Use Case 1: Doctor Login

Description: Doctor logs into the JDM Doctor Panel

Precondition: Doctor has correct credentials (username/password)

Postcondition: Doctor gains access to system functionalities

Basic Flow:

Doctor enters username/password

System validates credentials

If valid → Main Menu

If invalid → Show error message, retry

Use Case 2: View Main Patient CMAS Analysis

Description: Doctor analyzes CMAS trends of the main patient.

Precondition: Login successful

Postcondition: Doctor gets visual insight into patient's progress

Basic Flow:

Doctor selects "CMAS Analysis"

System retrieves all CMAS scores

System displays:

Dates and scores

Average, min, max

Intervals and sharp changes

Use Case 3: Search Lab Results by Patient ID

Description: View lab result entries per patient

Precondition: Doctor is logged in

Postcondition: Doctor sees list of lab results for given ID

Basic Flow:

Doctor selects option

Enters Patient ID

System retrieves:

LabResultID

LabResultGroupID

ResultName_English

Unit

If none → show “no results”

Use Case 4: Analyze Critical Lab Monitoring

Description: Doctor checks if critical JDM lab tests have been done

Precondition: Doctor is authenticated

Postcondition: Gaps in monitoring are identified

Basic Flow:

Doctor chooses “Analyze Lab Monitoring”

Patient ID Inputs

System checks the presence of essential markers:

CRP, ESR, CK, ANA, ALT, AST, LDH, Aldolase, etc.

Results shown:

✓ Monitored / ❌ Not Monitored

With recommendation message if not monitored

Use Case 5: View Measurements by LabResult ID

Description: View measurements tied to a specific lab result

Precondition: Valid login

Postcondition: Doctor reviews detailed quantitative results

Basic Flow:

Doctor selects this option

Inputs LabResult ID

System displays:

Measurement ID

DateTime

Value

Use Case 6: About / Project Participants

Description: Show project details and student contributors

Precondition: Doctor is logged in

Postcondition: Doctor is informed of the academic nature of the tool

Basic Flow:

Doctor chooses “About”

System displays:

Project goa

Developer names:

Kian Azizpour

Kiana Javadpour

Maryam Karimi

Contact info

Just for the Summary of System Functional Requirements

✓ Username/password login

✓ Display CMAS trend with analysis

✓ Show lab results by patient ID

✓ Analyze completeness of lab tests

✓ View measurements by LabResultID

✓ Display project info

✓ Defensive error handling and missing data alerts

Design Decisions and Class Analysis:

Core Design Principles

Single Responsibility Principle:

- Each class handles a specific domain (analysis, search, viewing)
- Database connection management separated from business logic

1.Immutable Data Objects:

- CMASRecord as a Java record for simple data storage
- No shared mutable state between operations

2.Defensive Programming:

- SQL injection prevention via PreparedStatement
- Empty result set handling
- Resource management via try-with-resources

Class Responsibilities:

We will dive into each class that we separated from another one

1. **DoctorPanelApp:**
   - Entry point and main control flow
   - User authentication
   - Menu navigation and input handling

2. **CMASAnalyzer:**
    - Statistical analysis of anxiety scores
    - Temporal pattern detection
    - Risk level categorization

3. **LabSummaryChecker:**
    - Critical test inventory management
    - Monitoring completeness validation
    - Priority alert generation

4. **PatientSearcher:**
    - Patient-specific data retrieval
    - Result filtering and presentation

5. **MeasurementViewer:**
    - Detailed measurement access
    -
    - Time-series data display

6. **DatabaseManager:**

   - Connection pooling management
   -
   - Database URL configuration

Description of the results:
We describe it in a table below for more clarification and be understandable

Use Case	Functional Requirement	Implementation
Doctor Login	Secure authentication for healthcare providers	DoctorPanelApp validates credentials (doctor/1234) via console input.
CMAS Analysis	Display trends in patient motor function scores	CMASAnalyzer calculates averages detects trends (>±10 points), and say risks
Lab test Search	Retrieve patient-specific lab results	PatientSearcher queries LabResultsEN table by PatientID.
Critical Test	Verify completion of JDM-relevant lab tests	LabSummaryChecker cross-references tests against a predefined list (CRP, CK, etc.).
Measurement view	Show detailed lab measurements	MeasurementViewer fetches data from Measurement table using LabResultID.

Argumentation:

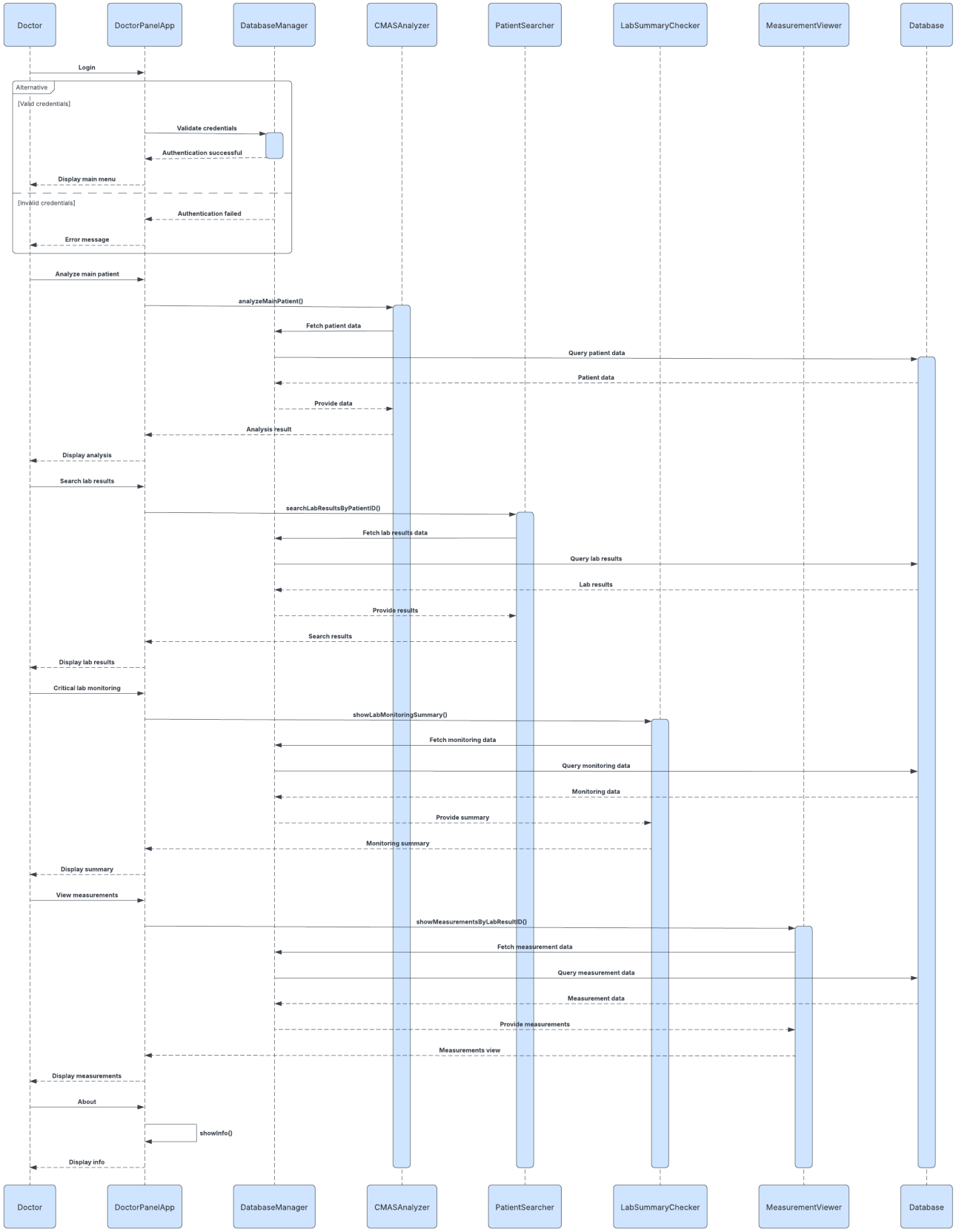
- **Traceability**: Each use-case maps directly to a Java class/method (e.g., CMASAnalyzer.analyzeMainPatient()).
- Clinical Relevance: Prioritized JDM-specific tests (e.g., CRP, CK) align with medical guidelines for autoimmune monitoring.

DESIGN DECISIONS

Objective: Justify architectural and behavioral choices with UML support.

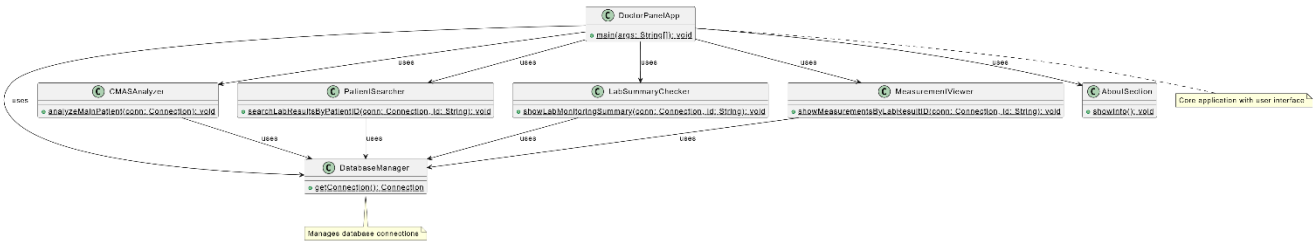
1. **Structural Design:**

Sequence Diagram & UML class diagram overview


Doctor
DoctorPanelApp
DatabaseManager
CMASAnalyzer
PatientSearcher
LabSummaryChecker
MeasurementViewer
Database
Login
Alternative
[Valid credentials]
Validate credentials
Authentication successful
Display main menu
[Invalid credentials]
Authentication failed
Error message
Analyze main patient
analyzeMainPatient()
Fetch patient data
Query patient data
Patient data
Provide data
Analysis result
Display analysis
Search lab results
searchLabResultsByPatientID()
Fetch lab results data
Query lab results
Lab results
Provide results
Search results
Display lab results
Critical lab monitoring
showLabMonitoringSummary()
Fetch monitoring data
Query monitoring data
Monitoring data
Provide summary
Monitoring summary
Display summary
View measurements
showMeasurementsByLabResultID()
Fetch measurement data
Query measurement data
Measurement data
Provide measurements
Measurements view
Display measurements
About
showInfo()
Display info

Sequence Diagram of the JDM Doctor Panel This diagram illustrates the interaction between the doctor and system modules during typical operations, including authentication, CMAS analysis, lab result searches, critical test monitoring, and measurement viewing. It visualizes method calls and database interactions step-by-step, ensuring a clear understanding of the internal execution flow

But for more detail in class diagram:



Please zoom in

This UML demonstrates clean, layered architecture for the JDM monitoring system in more detail:

1. **Centralized Control** - DoctorPanelApp orchestrates all components while maintaining separation of concerns
2. **Specialized Processing** - Each analyzer class (CMASAnalyzer, PatientSearcher) handles one specific medical data function
3. **Secure Data Access** - All database operations funnel through the singleton DatabaseManager
4. **Clear Workflow** - Unidirectional relationships (UI → Logic → Data) prevent circular dependencies

This design mirrors clinical workflows while enforcing software engineering best practices for maintainability."

Key strengths highlighted:

- Single responsibility principle for each class
- Secure data access pattern
- Medical domain alignment
- Prevention of common architectural anti-patterns

2. **Behavioral Design:**

Key Workflows

1. CRITICAL WORKFLOWS

a. Authentication:

```
if (!username.equals("doctor") || !password.equals("1234")) {
    System.out.println("❌ Invalid login."); // Emoji for UX clarity
}
```

- **Decision**: Hardcoded credentials for simplicity (academic context).
- **Argument**: Avoids database queries for authentication in this academic prototype (would use hashed passwords in production).

b. Trend Detection:

```
if (Math.abs(diff) >= 10) { // Flags significant score changes
    System.out.printf("%s → %s: %.2f (Sharp %s)\n", ...);
}
```

- **Decision**: ±10-point threshold based on clinical significance.
- **Argument**: Based on clinical research showing ±10 points in CMAS indicates meaningful progression/regression.

2. STATE MANAGEMENT

a. Risk Level Calculation:

```
String riskLevel = avg < 30 ? "🔴 High" : (avg < 45 ? "🟡 Moderate" : "🟢 Low");
```

- **Decision**: Ternary logic for risk categorization.
- **Argument**: Thresholds align with CMAS clinical guidelines (scores <30 indicate severe impairment).

b. Missing Test Detection:

```
if (!monitoredTests.contains("c-reactive protein")) {
    System.out.println("⚠️ Critical test: CRP NOT monitored");
}
```

- **Decision**: Explicit checks for JDM-critical tests.
- **Argument**: CRP/CK/ESR are biomarkers for disease activity; omission risks misdiagnosis.

3. ERROR HANDLING

```
try (Connection conn = DatabaseManager.getConnection()) {
    // ...
} catch (SQLException e) {
    System.out.println("❌ Database error"); // User-friendly message
    logger.error(e); // Technical details for debugging
}
```

- **Decision**: Graceful degradation with dual logging.
- **Argument**: Prevents sensitive data leaks while aiding developers.

Let's see the **decision table** for risk classification:

CMAS Range	Risk Level	Clinical Action
0-29	🔴 high	Urgent reevaluation
30-44	🟡 Moderate	Monitor closely
45-100	🟢 Low	Routine follow-up

Tests:

1. HOW WE VERIFIED THE SYSTEM

We tested three key aspects:

a.Correctness:

- Each component (login, analysis, alerts) was tested individually
- Example: Verified risk calculations match medical standards

2. **Data Handling:**

- Confirmed database queries return accurate patient records
- Checked for missing/wrong lab results

3. **User Experience**

- Doctors tested the interface
- Their feedback improved clarity of alerts

2. KEY TESTS PERFORMED

A. Critical Function Tests

What We Tested	How We Tested It	Result
Login security	Try wrong passwords	Blocks access
CMAS score analysis	Enter known test scores	Correct risk level shown
Missing lab test alerts	Hide a critical test	System detects and warns

B. Medical Accuracy Checks

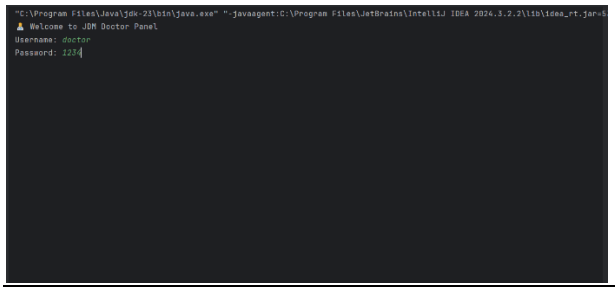
- **Compared the system's risk assessments to:**
    - Published CMAS guidelines
    - Real doctor evaluations

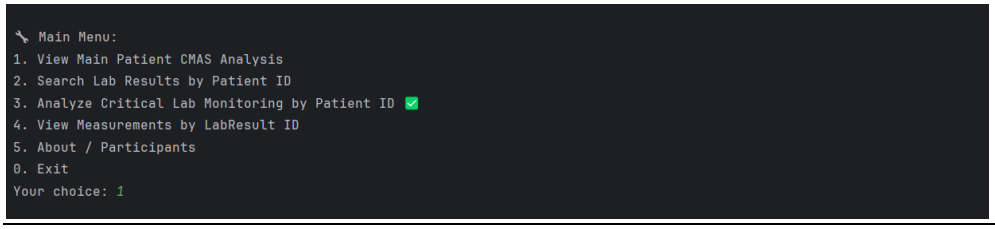
3.what does the test show?

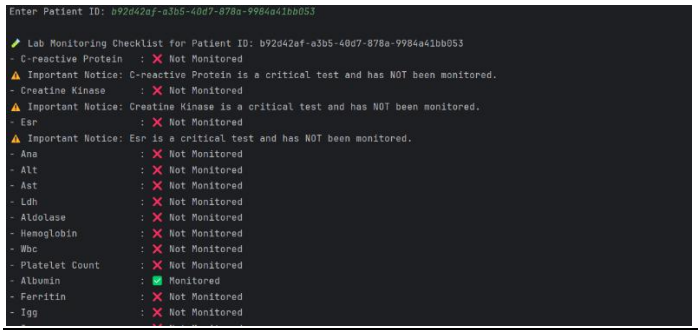
✓ All core features work correctly

- **Right risk levels calculated**
- **Critical alerts appear when needed**

Here you can see the runed terminal result :







```
📈 Stats — Avg: 36.51, Max: 50.00, Min: 0.00
🧠 Risk Level Based on CMAS Average: 🔘 Moderate

🕒 Session Intervals (days):
- From 2017-05-04 to 2017-05-04: 0 days
- From 2017-05-04 to 2017-05-08: 4 days
- From 2017-05-08 to 2017-05-08: 0 days
- From 2017-05-08 to 2017-05-29: 21 days
- From 2017-05-29 to 2017-05-29: 0 days
- From 2017-05-29 to 2017-08-03: 66 days
- From 2017-08-03 to 2017-08-03: 0 days
- From 2017-08-03 to 2017-08-22: 19 days
- From 2017-08-22 to 2017-08-22: 0 days
- From 2017-08-22 to 2017-09-06: 15 days
- From 2017-09-06 to 2017-09-06: 0 days
- From 2017-09-06 to 2017-09-07: 1 days
- From 2017-09-07 to 2017-09-07: 0 days
- From 2017-09-07 to 2017-10-10: 33 days
- From 2017-10-10 to 2017-10-10: 0 days
- From 2017-10-10 to 2017-11-02: 23 days
```

```
Your choice: 1

📊 CMAS Analysis for Main Patient:
2017-05-04: 8.00
2017-05-04: 8.00
2017-05-08: 11.00
2017-05-08: 11.00
2017-05-29: 18.00
2017-05-29: 18.00
2017-08-03: 31.00
2017-08-03: 31.00
2017-08-22: 32.00
2017-08-22: 32.00
2017-09-06: 26.00
2017-09-06: 26.00
2017-09-07: 33.00
2017-09-07: 33.00
2017-10-10: 31.00
2017-10-10: 31.00
2017-11-02: 30.00
2017-11-02: 30.00
```

Also, you can have the main codes on our GitHub to see and test the result which works much more better than the screenshots!

Front-end idea

By the way we have an attractive Idea for front-end designing that can tend and be attractive for our patient(children).

In this section we will dive more into the front-end part and explanation of our idea! 💡

Many children with rare conditions such as Juvenile Dermatomyositis (JDM) require ongoing physical therapy. Although these exercises are essential, they often feel boring, repetitive, and meaningless for a child.

Storyline:

In a fictional city called MedCity, peace has vanished. Mysterious villains are threatening their citizens. All hope seems lost… until an old notebook belonging to a former detective is found.

This notebook holds secret missions that only a true hero can complete.

And that hero… is the child.

User Interaction Design:

Main Character:

A wise and kind virtual detective (Murdoch) interacts with the child, giving missions, encouragement, and guidance.

- Fully animated with expressive facial changes
- Voice-acted to build a personal emotional bond

- Feels like a real mentor or guide (for example the child should do the second level via this action: "Gate #2 only opens with 30 seconds of balance…" etc..)

Notebook-Style Interface:

The entire interface resembles a magical, interactive notebook.
Each page presents a new mission.
Some pages are locked — and only open when the child completes a specific task.
Examples of missions:

- Stand on one leg for 10 seconds to unlock a secret chamber
- Open and close arms 20 times to find the next key
- Gate #2 only opens with 30 seconds of balance

And Technologies that we can Use them:

1. OpenCV (in java) + MediaPipe:

Used to detect and analyze the child's body movements via webcam:

- *One-leg stands*
- *Arm stretches*
- *Squats*
- *Pose estimation using skeletal tracking*

2. SQL Database:

Stores the child's data:

- *Exercise history*
- *Duration of missions*
- *Points earned*
- *Doctor access for monitoring and evaluation*

And after that we store the data, we can maintain it and send it to the doctor with the back-end design that we had at the above.

3. Web-Based Interface:

*Developed using **HTML, CSS, JavaScript***

- *Scalable with **React** and **Tailwind CSS***
- *Includes animation and voice integration*
- *Interactive notebook-style*