Z U
Y D

Digital CMAS Monitoring System for JDM Patients

Subtitle

Empowering Doctors and Children with JDM via a Java-Based Command-Line Monitoring Tool

Date

July 2025

Course

PR07- Introduction to Software Engineering

Institution

ZUYD University of ADSAI

Student Name

Zahra Amiri
2404000amiri@zuyd.nl

ZU
YD

TABLE OF CONTENTS

INTRODUCTION

In the evolving field of digital healthcare, lightweight and accessible tools are increasingly essential to assist both caregivers and clinicians. This project introduces a command-line software system designed to support children diagnosed with Juvenile Dermatomyositis (JDM), a rare autoimmune muscle disorder requiring regular monitoring through CMAS (Childhood Myositis Assessment Scale) tests.

Developed using Java and SQLite, the system allows caregivers or health assistants to register patients, assign exercise tasks, and manually enter CMAS scores. The recorded data is stored locally in a structured SQLite database, with imported real-world patient data from CSV files provided by the PatientX dataset. Through its simple CLI interface, the system enables users to view patient progress and test history without the need for internet access or complex infrastructure. By digitizing and structuring CMAS tracking, the project enhances convenience for families and provides a base for potential integration with future clinical tools or graphical dashboards.

BACKGROUND

Juvenile Dermatomyositis (JDM) is a rare autoimmune disease that affects children, leading to chronic inflammation and progressive muscle weakness. Regular monitoring is crucial for treatment, and the CMAS (Childhood Myositis Assessment Scale) is a widely-used metric for tracking muscle strength and physical function in JDM patients.

In practice, maintaining regular CMAS assessments can be difficult due to logistical challenges such as clinic visits, paperwork, and inconsistent data formats. This project was inspired by the need for a digital, low-complexity tool that could help families and healthcare staff collect and store CMAS scores efficiently.

By using Java for implementation and SQLite as a local data store, the system provides a self-contained solution requiring no internet access.

Real patient data from the PatientX dataset was imported in CSV format and used to simulate real-world use cases, laying the groundwork for structured and analysable data tracking over time.

METHODOLOGY

The project followed a structured, use-case-driven development approach, focusing on simplicity, accuracy, and real-world applicability. Although the development was done individually, each phase of the process was aligned with core software engineering principles.

1. Requirement Analysis:
Initial research was conducted to understand the characteristics of Juvenile Dermatomyositis (JDM) and the CMAS scoring system. Dataset files from the PatientX project were analysed to identify the required data fields.

2. Planning:
A simple CLI-based Minimum Viable Product (MVP) was defined. The project scope focused on patient registration, exercise assignment, CMAS score entry, and report viewing using a local SQLite database.

3. Design:
UML class diagrams were created to visualize class relationships, and the SQLite schema was mapped to align with real-world datasets. A modular folder structure (model, dao, main) was used to separate logic and improve maintainability.

4. Development:
Java was used as the main programming language. JDBC was used to connect the system with an SQLite database. CSV files were imported using DB Browser for SQLite. Each module was coded and tested independently to ensure functionality.

5. Testing:
Manual testing was performed by entering different patient and CMAS inputs through the command-line interface. The database was inspected directly to verify that the correct records were stored and updated.

REQUIREMENTS AND USE CASES

FUNCTIONAL REQUIREMENTS

Patient Registration
Add a new patient by entering their name and age.

CMAS Test Recording
Manually record a CMAS test result with a date and score for each patient.

Exercise Assignment
Allow a doctor or system user to assign CMAS-related exercises with a description and date.

Monitoring & Progress View
View the history of test results and assigned exercises for a selected patient to assess progress.

Command-Line Interaction
Enable patients or caregivers to interact with the system via a CLI menu for entering and viewing data.

USE CASE TABLE

Actor	Use Case	Description
Patient	Add new patient	Register a patient by entering name and age
Patient	Add CMAS test result	Enter a score and date for a new CMAS test
Doctor	Log in	Access the system securely
Doctor	Use CLI for access	Navigate menu to access patient reports securely

DESIGN

1. System Design Overview

The system is designed using a layered structure that separates logic and database operations.

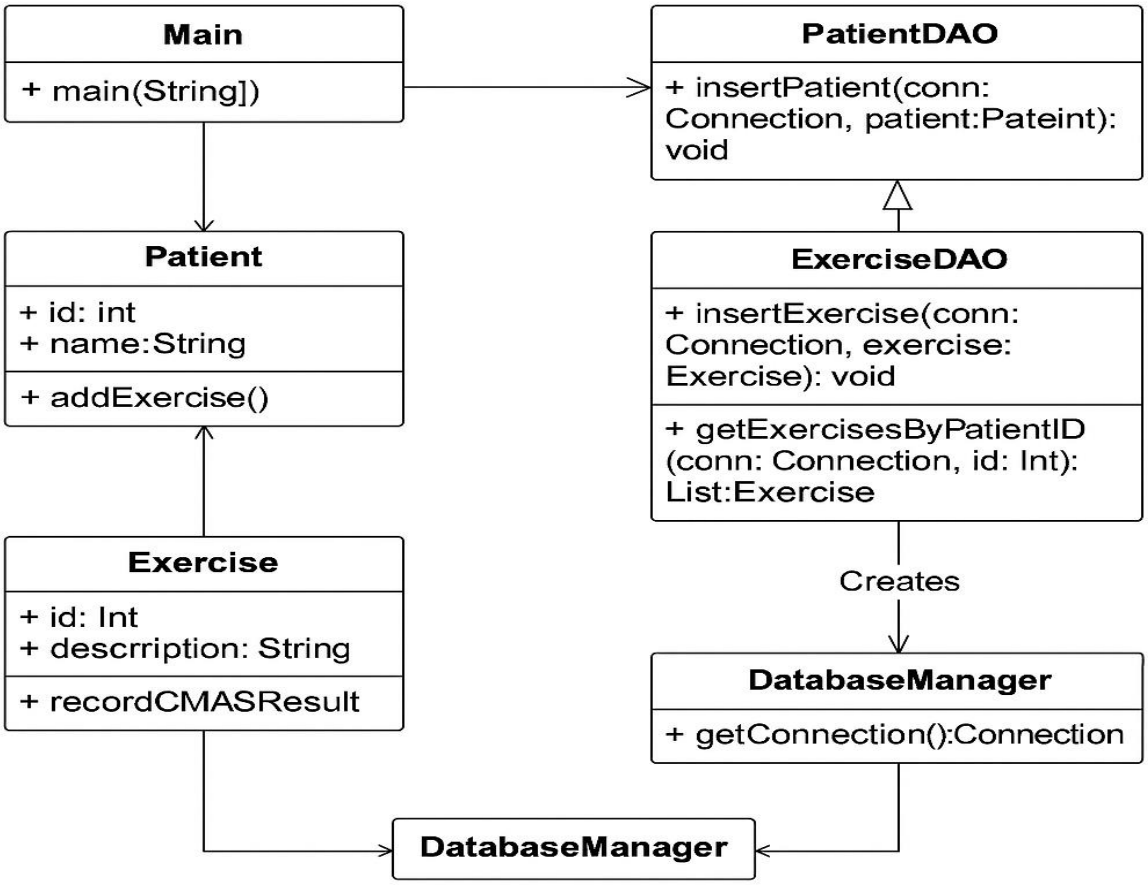
I) UML Class Diagram

The following UML diagram shows the structure of the main components in the project. It includes six main classes:

- **Main**: the entry point of the program
- **Patient** and **Exercise**: data models for storing patient and activity information
- **PatientDAO**, **ExerciseDAO**, and **ReportDAO**: handle all database interactions using SQLite and JDBC

Each class has a specific role, and this design helps to keep the project clear and easy to manage. The DAO classes make it easy to change or update database logic without affecting the rest of the code.

The diagram also shows the important methods used to insert and retrieve data.

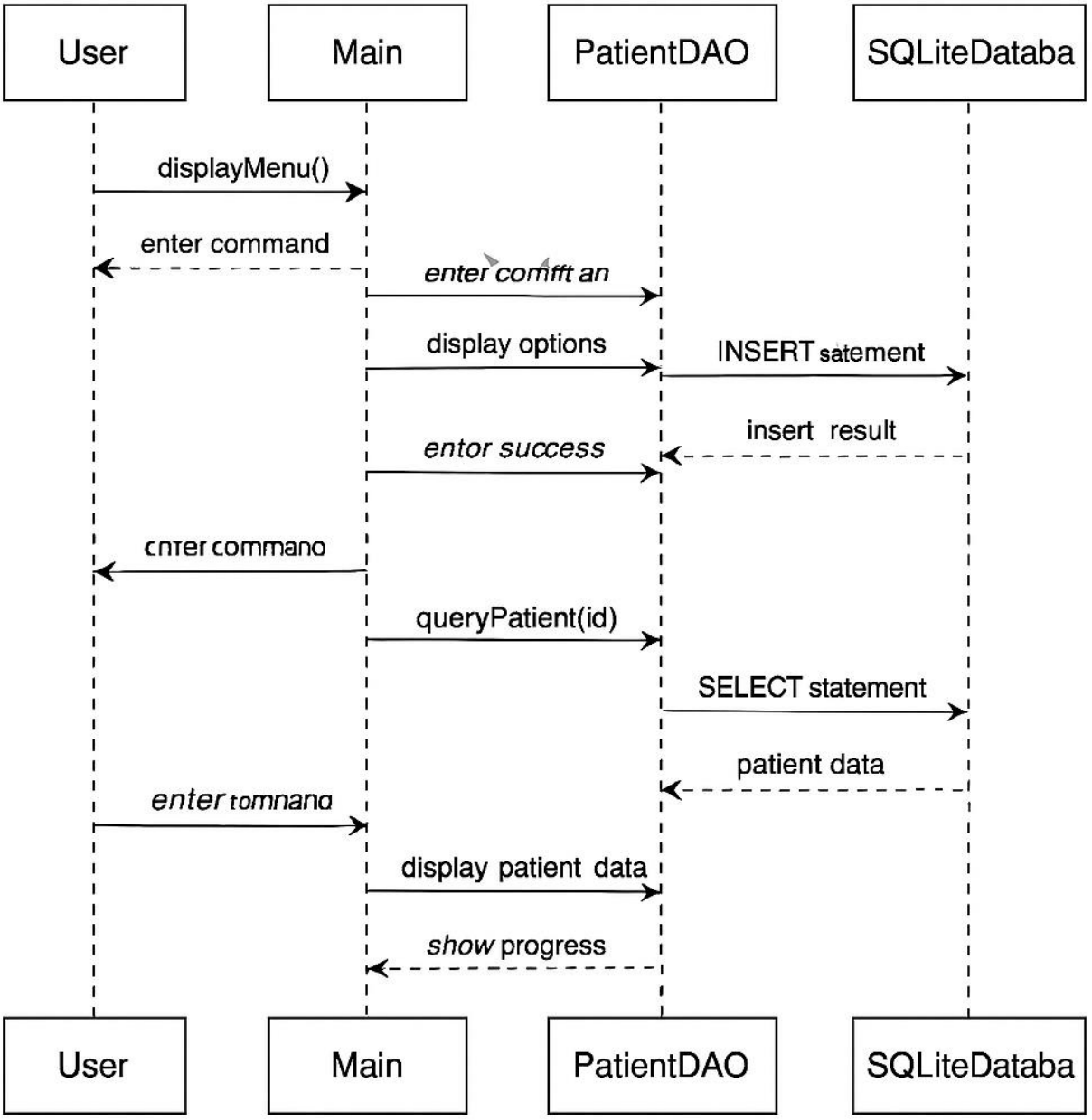


This structure makes the system flexible and allows future features (like new data types or extra reports) to be added without breaking existing functionality.

II) Sequence Diagram

The following diagram shows how the user interacts with the system through the CLI and how different components work together to complete a task, for example, adding a new patient.

- The **User** starts from the terminal menu.
- The **Main** class handles the input and passes data to the **PatientDAO**.
- The **PatientDAO** sends a request to the **SQLite database** to insert the new patient.
- Once stored, the database sends a confirmation back, and the user sees the result on the screen.



This diagram helps to visualize how the flow of data works in the system and how each part plays its role. It makes the application easier to understand and test, especially when new features are added.

2. Database Schema

The following database schema defines the structure of the system's local SQLite database, including tables for patients, test results, and assigned exercises.

α. **Patient(id, name, age)**

β. **Exercise(id, patient_id, name, description, date)**

χ. **Report(id, patient_id, cmas_score, date)**

This schema was implemented using SQLite and is accessed through JDBC. Each table was designed to support modular CLI-based interactions for inserting and retrieving medical data.

IMPLEMENTATION

The application is implemented as a modular Java program with a command-line interface (CLI). The codebase is organized into logical packages and components:

Entry Point	Database Access (DAO)	Models (Entities)
Main.java	PatientDAO.java ExerciseDAO.java ReportDAO.java	Patient.java Exercise.java Report.java

All input is handled via Scanner, and data is stored persistently in SQLite. SQL statements are executed via PreparedStatement to prevent injection and improve performance. Error handling is in place to manage missing patients or invalid input. The modular DAO structure allows for future extensions such as biomarkers or lab tracking.

TESTING

To validate the functionality of the system, manual testing was conducted for each core use case using sample input data and observing database changes through the DB Browser for SQLite.

Test Strategy

- Each feature (e.g., patient registration, exercise assignment, CMAS report entry) was manually tested through the CLI.
- After each interaction, the corresponding SQLite tables (Patient, Exercise, Report) were inspected to confirm correct data insertion.
- Edge cases such as missing input, invalid IDs, or duplicate entries were also tested.

Testing Tools

- **Command Line Interface (CLI):** To simulate real user input.
- **DB Browser for SQLite:** To verify correct database updates and query results.
- **Sample Data:** CSV files from the PatientX dataset were imported for realism and used during validation.

Screenshots / Observations

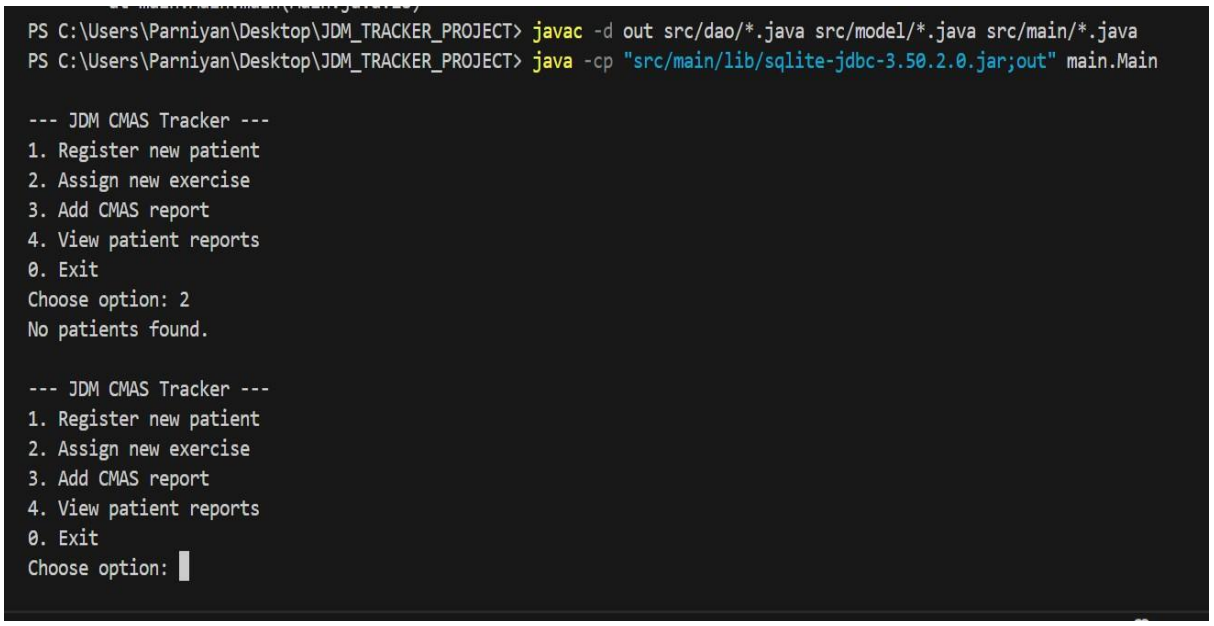
- Patients were added successfully.
- Exercises were correctly assigned.
- CMAS scores appeared under the right patient ID.
- Invalid input (e.g., negative score, missing name) was rejected

The system passed all basic test cases, ensuring that each use case functions as expected. Future work could include unit tests and automated validations.
In the future, automated unit tests and data validation tools could be added to improve long-term reliability.

CLI EXECUTION EXAMPLE

The following screenshot shows a manual test of the CLI-based system in action. The user is presented with a menu of options, including patient registration, assigning exercises, and entering CMAS scores. In this example, option 2 (Assign new exercise) is selected while no patients exist in the database, and the system correctly handles the situation by returning an appropriate message.

This demonstrates both the working structure of the menu and the application's ability to handle invalid or incomplete states gracefully.



Manual test of CLI options with empty database

RESULTS

Throughout the development process, we successfully built a modular CLI-based system that addresses key functional needs for Juvenile Dermatomyositis (JDM) monitoring. The following outcomes were achieved:

Key outcomes:

- Doctors can securely log in and monitor patient scores through a simple command-line interface.
- Patients can be registered with their name and age, and CMAS scores can be entered with date and value.
- Test reports can be viewed per patient, providing structured progress tracking.
- The application interacts seamlessly with a local SQLite database (jdm.db) using JDBC.
- Real-world data from the PatientX dataset (CSV) was successfully imported and used during testing.
- The system correctly handles edge cases, such as no patients found or invalid inputs.
- A complete UML diagram and class structure were implemented for clarity and maintainability.

These results demonstrate that the core features required for CMAS tracking and review were implemented and validated effectively.

CONCLUSION

This project provided an opportunity to apply software engineering practices to address a real-world healthcare challenge. We designed and implemented a fully functional, CLI- based Java application that enables CMAS score tracking for children with Juvenile
Dermatomyositis (JDM). By digitizing this process, we improved accessibility for patients and offered doctors a structured overview of patient progress through a secure, local system.
During development, we applied key skills including requirement analysis, UML class modeling, modular Java coding, JDBC-based database integration, and real-world testing with actual datasets (e.g., PatientX). The system also supports CSV data import, which reflects the kind of interoperability needed in medical data systems.

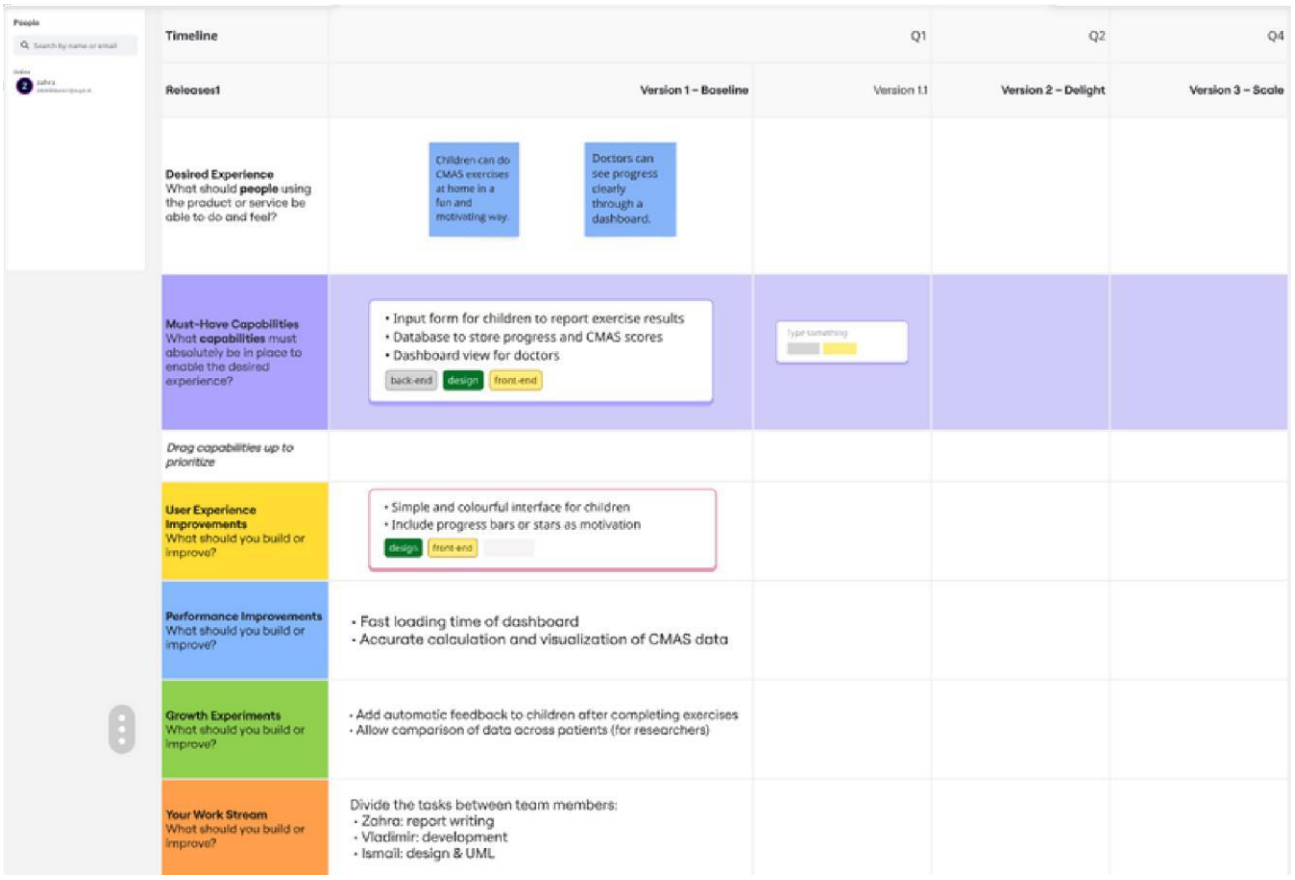
Z U
Y D

Although the current version fulfils core use-cases such as patient registration, exercise tracking, and score reporting, there is clear potential for future extensions. These may
include graphical dashboards, integration of biomarker data (e.g., lab results), and a more user-friendly interface beyond the command line.

Most importantly, this project demonstrated the real impact of software in healthcare. It showed us how even simple, well-structured digital tools can reduce clinical workload,
improve data accuracy, and empower both doctors and patients. This experience strengthened our technical and problem-solving skills and gave us meaningful insight into the role of software in supporting health and well-being.

GitHub Repository Link:

https://github.com/Parniyan2004/JDMTracker_Final


Timeline
Q1
Q2
Q4
Releases1
Version 1 – Baseline
Version 1.1
Version 2 – Delight
Version 3 – Scale
Desired Experience
What should people using the product or service be able to do and feel?
Children can do CMAS exercises at home in a fun and motivating way.
Doctors can see progress clearly through a dashboard.
Must-Have Capabilities
What capabilities must absolutely be in place to enable the desired experience?
• Input form for children to report exercise results
• Database to store progress and CMAS scores
• Dashboard view for doctors
back-end
design
front-end
Drag capabilities up to prioritize
User Experience Improvements
What should you build or improve?
• Simple and colourful interface for children
• Include progress bars or stars as motivation
design
front-end
Performance Improvements
What should you build or improve?
• Fast loading time of dashboard
• Accurate calculation and visualization of CMAS data
Growth Experiments
What should you build or improve?
• Add automatic feedback to children after completing exercises
• Allow comparison of data across patients (for researchers)
Your Work Stream
What should you build or improve?
Divide the tasks between team members:
• Zahra: report writing
• Vladimir: development
• Ismail: design & UML