**Birzeit University - Faculty of Engineering & Technology**



**Electrical & Computer Engineering Department - ENCS313**

**Linux laboratory**

**Prepared by:**

**Roa makhtoob 1221636**

**Instructor: Aziz Qaroush**

**Teacher Assistant: Eng. Ahed Mafarjeh**

**Section: 2**

**Shaimaa dar taha 1222440**

**Instructor: Mohammad Jubran**

**Teacher Assistant:** **Eng. Ahed Mafarjeh**

**Section:1**

**Date: 22/8/2024**

**1. Introduction**

**Objective**:  
The purpose of this project is to develop a Medical Test Management System that efficiently stores, manages, and retrieves medical test data for individual patients. The system is designed to be a basic patient record management tool, focusing on handling medical test results using Python's object-oriented programming capabilities.

**Scope**:  
The system allows users to add new medical tests, update existing records, delete incorrect entries, and retrieve specific test results based on criteria such as patient ID, test type, or abnormal results. It provides options for filtering tests and generating summary reports.

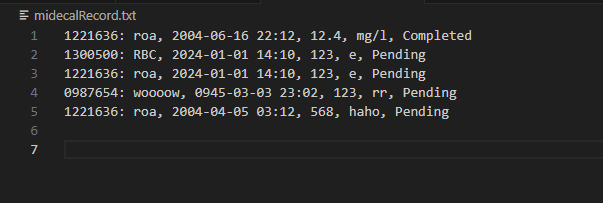
**2. System Design**

**2.1. Object-Oriented Design**  
The project is built using object-oriented principles. The following key classes were developed:

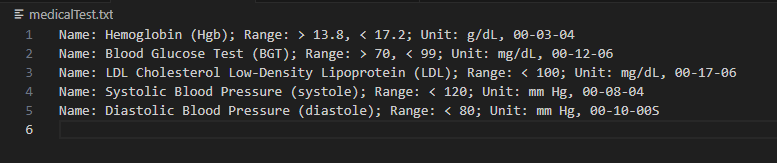
* **MedicalTestType**: Represents a medical test with attributes like name, normal range, unit, and turnaround time.
* **MedicalTestTypeManager**: Manages all medical tests, including adding, updating, and retrieving test data from a text file.

**2.2. File Structure**  
The system uses text files to store patient records and medical test details. These files follow a specific format that includes fields like patient ID, test name, result values, unit, and timestamps.

**MidecalRecord file**

****

**medicalTest file :**

****

**3. Implementation**

**3.1. Features**  
The following features were implemented:

* **Add New Medical Test**: Allows users to insert a new medical test type and save it in a file.
* **Add New Medical Test Record**: Allows users to store new test results with appropriate data validation.
* **Update Records**: Provides functionality to update patient test records and medical test details.
* **Filter Medical Tests**: Allows filtering of tests based on criteria like patient ID, test name, abnormal results, or specific date ranges.
* **Generate Reports**: The system can generate textual summary reports, providing statistics such as minimum, maximum, and average test values and turnaround times.
* **Export/Import Data**: Users can export and import medical records from/to a comma-separated file.

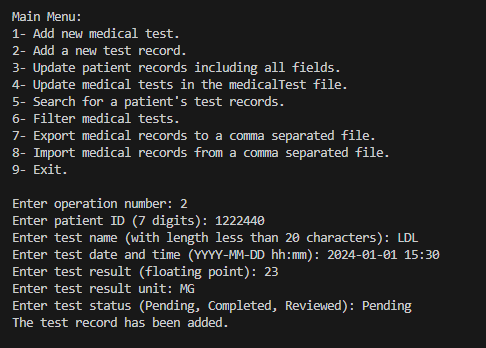
**3.2. Error Handling**  
Robust error handling mechanisms are in place to manage invalid file names, incorrect user inputs, and other potential issues like missing data. For example, the system validates input formats for patient IDs, dates, and ranges.

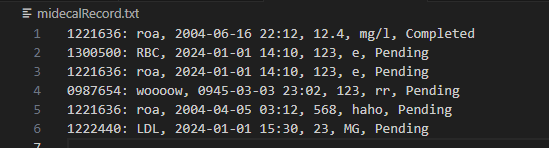
**3.3. Data Validation**  
Data validation ensures that only correct and meaningful data is entered into the system. This includes checking for valid numeric ranges, proper formatting of dates and times, and ensuring the logical consistency of test records.

**4. Testing**

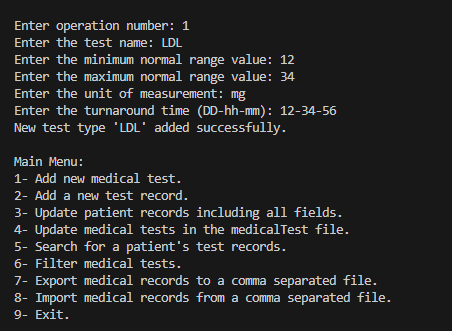
**4.1. Test Cases**  
The system was tested using several scenarios to ensure that it handles different types of data

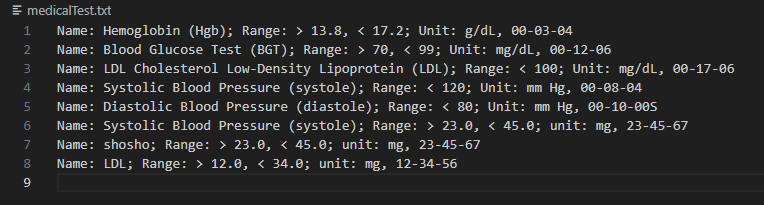
Case 1: Add new medical test: the system will allow user to insert new type of medical test and save it in the medicalTest file. The system will check the validity of the input data.





Case 2: Add a new medical test record: the system will allow the user to store a new medical test with the required data. The system will check the validity of the input data.





**4.2. Results**  
The system successfully passed all test cases, with correct storage, retrieval, and filtering of medical test records. Error handling was effective in managing invalid inputs.