National College of Ireland

BSc(Hons) in Computing

Software development

2020/2021

Mark Gaffney

x17443036

x17443036@student.ncirl.ie

Dynamic Sample Processing Dashboard for Hospital Laboratories

Technical Report

Contents

Figures

Executive Summary

This report outlines the details of a project that aims to provide a system to improve the functionality of the CliniSys Laboratory Information System (LIS). The LIS currently does not automatically monitor the progress of samples as they are processed by the laboratory. With the current laboratory workload, detecting delays in reporting results on urgent samples is difficult and labour intensive. Targets for Turn Around Times (TATs) are often exceeded due to laboratory staff being too busy to run manual LIS searches and anticipate and correct delays in sample processing.

The aims and system requirements of the project address the user requirements and provide the features required to address the deficiencies in the LIS. These features are outlined and explained.

Visual examples are provided on key screens and displays. These screens are explained, and user case scenarios are given for key flows a user would follow. A high-level overview of the system architecture and implementation clearly explains the workings of the system. All important features are outlined and explained with visual representation.

Areas the project is limited by are explained and ways to overcome these limitations are outlined. The strengths and weaknesses of the program are explained and the effects that cause them are highlighted. The possible progression of the project in the future are outlined and what can affect the future of the project.

1.0 Introduction

1.1. Background

The idea behind the creation of this project was from experience gained while assisting with work in Tallaght Hospital's Laboratory. All major hospitals in Ireland have laboratories that analyse and process blood and other vital samples. The laboratories focus is being able to produce quick, accurate and precise results from samples which are to be reported back to doctors or nurses for diagnosis, treatment, or monitoring. Up to 70% of all medical diagnoses require laboratory results. For laboratories to be able to work effectively they use specialized systems which conduct tests on the samples that are received. There are many different machines, and each have a specific job to do with the samples. All machines are linked to the Laboratory Information Systems (LIS), which can process samples and produce files holding current information. Since the LIS was first introduced in hospitals in Ireland the improvements on the features within the system have not been able to keep up with current day technology. The outdated system features minimize laboratories possible effectiveness and overall speed when producing results.

Laboratory staff perform many routine tasks that ensure high quality results are produced in a timely manner. It is very easy for samples, or some tests, to be delayed during the multiple steps in processing all requests on each sample. Each year there is an increasing demand for laboratories to produce results. This increase in demand has led to the development of a replacement system. Individual Laboratory Information Systems will be replaced by a single, centralised, system for all medical laboratories in the country. Cerner are developing the MedLIS system, which is using Oracle for data management, but the project is many years behind schedule and does not provide the functionality proposed in this project.

A part of Laboratory staff's duty is to monitor the processing of samples and ensure the results are produced without delays. This task is currently done manually which takes up staff's time searching and running queries. If this process were to be automated, it would result in less wasted time with improved reports and speed in alerting when delays occur. The implementation of MedLIS will improve the overall laboratory system but it is missing these key features. MedLIS missing these features means that there is a need for the automation of these features which this program will be able to provide. Talks with Eoin Begley, who is Tallaght hospital Chief Medical Scientist in Biochemistry, confirmed that the program proposed is desired and welcome once approved by Hospital IT staff.

1.2. Aims

The purpose of this project is to automate the tracking of laboratory samples as they are processed. It will also provide analysis and graphical presentation of Turn Around Times (TATs) on previously processed samples for scientists and laboratory management.

- The program must provide a high quality and user-friendly dashboard with minimal clutter. It must be quick and easy to access different sections. To be accepted into Tallaght Hospital the program needs to be high standard and very reliable.
- The system needs to alert laboratory staff when tests for a sample have exceeded pre-defined timescales. The focus is on urgent requests, but the design ensures that all samples can be monitored, and appropriate alerts generated. Urgent requests are extremely time sensitive which means they require more attention. The automation of the monitoring of these urgent samples will ensure that results will be reported as fast as possible.
- Generating reports of TATs ensure the laboratory is providing a service to an anticipated level and in accordance with ISO 15189 accreditation standards. Statistics on delays and other details on sample processing is crucial and the provision of a graphical summary will ensure scientists can easily see how the laboratory is performing against its targets.
- The Dashboard should monitor a file holding samples every minute. Removing any processed samples, adding new samples, and maintaining the samples still in process.
- This project is centred around Tallaght Hospital's Biochemistry laboratory which is currently running CliniSys as its LIS. This is due to change with the upgrade to a centralized Laboratory Information System (LIS) named MedLIS. Ensuring that the project is adaptable to the new MedLIS system is very important to keep the program relevant in future years.
- The program must be a desktop run application which will remain running in the background continuously monitoring sample entries. Minimizing the system resources requirements will also be a factor.
- There must be a provision to allow changes to be applied to different sample test groups through a settings page. This is to adapt to change in test times or the processes samples undertake. Providing user configurable targets and parameters is essential for flexibility in the project.
- A user needs to be able to alter samples start time for a defined period of time if a processing error occurs. This is needed when a sample is delayed, and staff will need to start a certain process over again.
- A Real-time progression bar for estimating the expected for urgent sample TAT will allow laboratory scientists and requesting doctors anticipate when results can be expected.
- The graph created should be saved to the local machine for management to use when creating reports.
- The program must demonstrate the possibility of adding to and improving the functionality of underdeveloped IT systems and satisfy user requirements.

1.3. Technology

The main technologies that are used in this project are Java, SQLite and Java Libraries such as Swing and JFreeChart. Other programming languages are not as well suited for a desktop application, Python and other languages do not include a lot of supporting libraries for design. C++ was another alternative but is quite outdated and cannot provide as much support as Java. Electron is a tool specifically designed for desktop application creation and is written with JavaScript and HTML. Due to the number of resources needed to run an Electron application it was not optimal for this project.

The NetBeans IDE is where the programs GUI is designed, and the programs main functions for alerting and reporting are developed. NetBeans provides a lot of features for designing a high-quality GUI. Libraries imported through NetBeans are needed for some calculations and functions to operate correctly.

SQLite is used for a database management system, SQLite does not require a server for the database to be in parallel with. SQLite uses files to store and manage data, this is ideal for this project as laboratory systems need to be very secure and having internal servers can affect the systems integrity. The alerting and report charting will both use the data within the databases to run their functions. The login features will also use SQLite for storing user's login details. [14]

Rs2XML library is required for passing information from the database into the table which is displayed to the user on the Dashboard. The table needs to be constantly updated with new information on samples. Rs2XML's library allows this process to be executed reliably and completed quickly.[6]

Java Swing is a very important part of the program's dashboard GUI screens. The design of the screens and panels displayed to the user will all be created using the Swing library. The library has a plethora of resources for creating a well-designed and functional dashboard. The Swing components allow the user to interact and view information displayed by the system.[11]

JFreeChart is a library that is needed for graphing and charting data within Java. JFreeChart will extend the Swing functionalities within the project and will be used for the report section. JFreeChart provides many options for charting and plotting data which can be utilised to summaries data for the report section. It is important that the data presented is clear and visually pleasing.[8]

Opencsv is a crucial library for this project as it provides Java projects with a reliably way to read from a csv file. The sample monitoring and reporting features both provide the system with data in csv format. To guarantee that the data can be correctly processed and inserted into both databases Opencsv file needs to be used.[4]

A GitHub repository was created for version control of the project during the development process. GitHub ensures that there is always a backup version of the project available in case any problems occur with the local version.

1.4. Structure

The document is split into six sections.

- **Section 2** highlights the system's requirements. This section will explain in detail the processes within the program. The features users interact with are explained and how the system interacts with different sections. The data needed to run this program will be explained. The expectations of the user and the features that they wish to be implemented are outlined also. The design of the code base and important functions for running the alert and reporting features are detailed. GUI screen shots are provided which show and explain key pages. The testing completed on the application is outlined.
- **Section 3** concludes the program highlighting main strengths as well as the factors which limit expanding the application. Advantages of using the system and disadvantages with the system are explained.
- **Section 4** explains future development possibilities. Explaining how integration within other hospitals would be done. What would need to be changed to suit other laboratory systems is outlined.
- **Section 5** provides links to any material used during the development of this project.
- **Section 6** holds the projects proposal and monthly journals showing general progress.

2.0 System

2.1. Requirements

2.1.1. Functional Requirements

2.1.1.1. Use Case Diagram

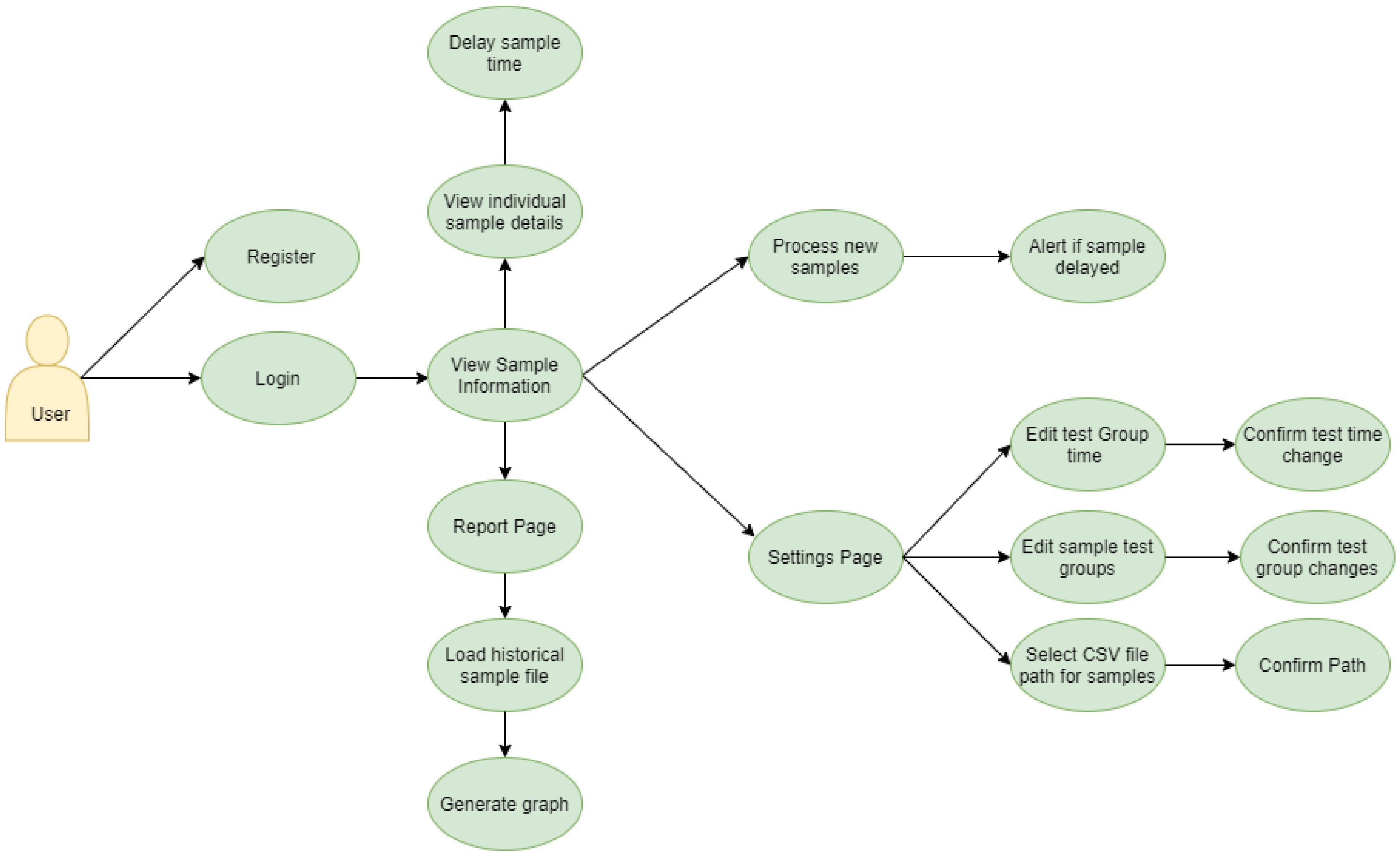


Figure 1 Overall system Use Case

2.1.1.2. Requirement 1 Registering of a user

Description & Priority

The use case for the registering of a user is needed for a new user to be inserted into the database.

Use Case

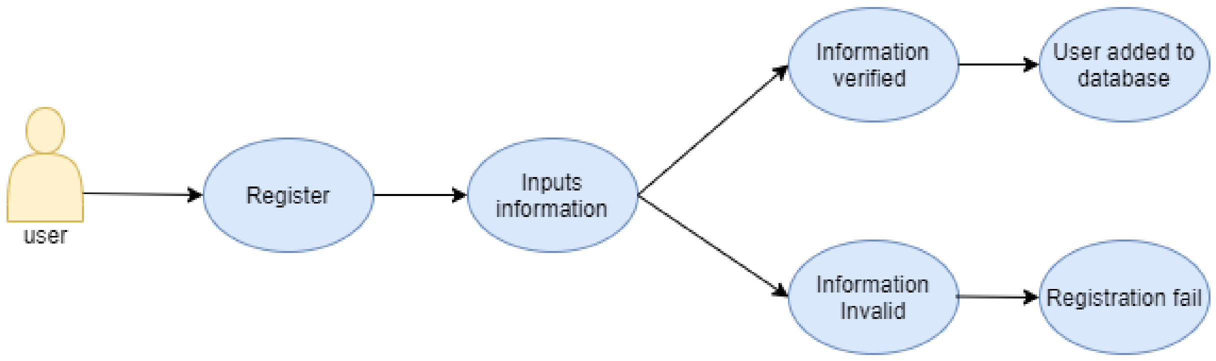


Figure 2 Register Use Case

<u>Scope</u>

The scope of this use case is to register a new user.

<u>Description</u>

This use case describes the process to be followed for a new user.

<u>Flow Description</u>

Precondition

The system requires a user to navigate to the register button.

Activation

This use case starts when a User clicks on the registration button.

Main flow

1. The user clicks on the registration button.
2. The user enters a Name and Password.
3. The system inserts Name and Password into database.
4. The user account is created.

Alternate flow

Null.

Exceptional flow

E1 : <User already exists>

1. The user clicks on the registration button.
2. The user enters a Name and Password.
3. The system inserts Name and Password into database.
4. User information already exists in database.
5. Cancel registration.

Termination

The user successfully registers.

Post condition

The system waits for user to login.

2.1.1.3. Requirement 2 Login for a user

Description & Priority

This use case is for the login of a user which is needed to access the main dashboard.

Use Case

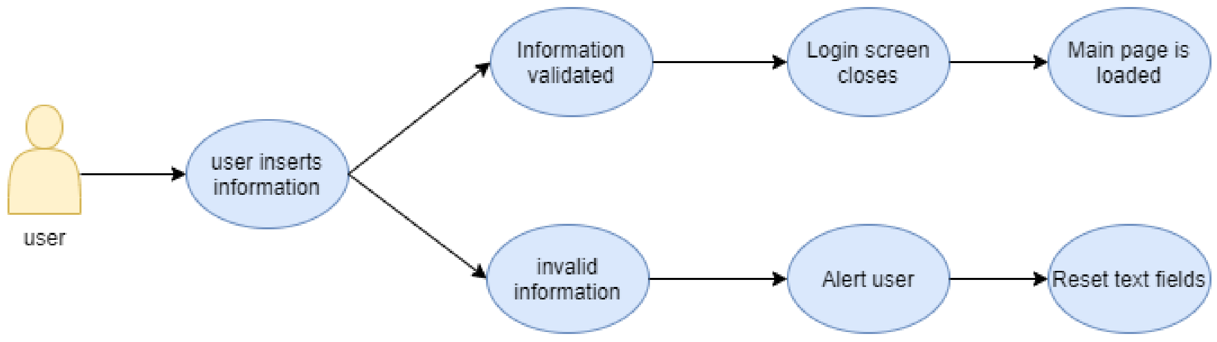


Figure 3 Login Use Case

Scope

The scope of this use case is to log into the system.

Description

This use case describes the process to be followed for a user to login.

Flow Description

Precondition

The system requires the user to input information into the text fields.

Activation

This use case starts when a User clicks on the registration button.

Main flow

1. The user enters in their details.
2. The user inputs were validated.
3. Log in screen closes.
4. Main page opens to user.

Alternate flow

Null.

Exceptional flow

1. The user enters in their details.
2. The user's inputs were incorrect.
3. Popup message.

4. User can enter details again.

Termination

The user is granted access to the system.

Post condition

The system waits for user to login.

2.1.1.4. Requirement 3 Show graph and statistics

Description & Priority

The use case for a user entering the report screen and enter a date range generate graph.

Use Case

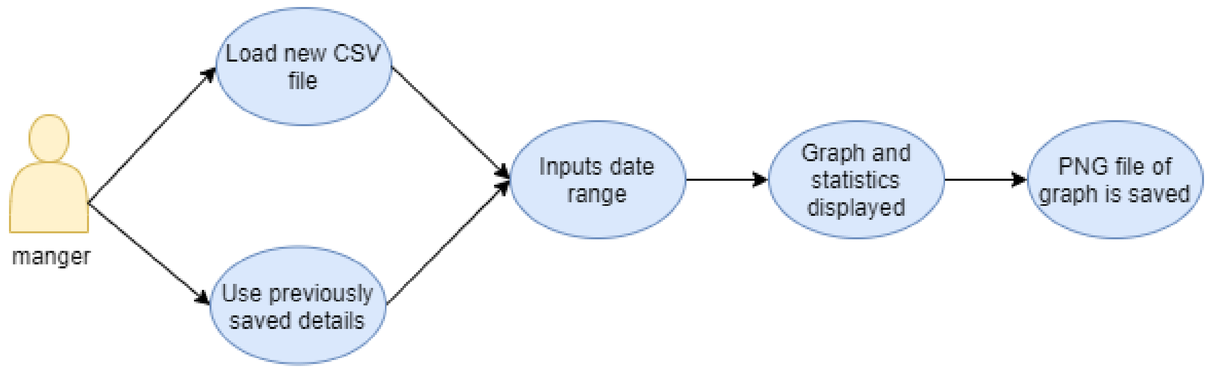

manger
Load new CSV file
Use previously saved details
Inputs date range
Graph and statistics displayed
PNG file of graph is saved

Figure 4 Graph and statistics Use Case

<u>Scope</u>

The scope of this use case is to give access to the report section.

<u>Description</u>

This use case describes the process to be after a user has successfully logged in and clicked onto the report button.

<u>Flow Description</u>

Precondition

The system requires a user to be logged in and to navigate to the report button.

Activation

This use case starts when a user clicks on the report button.

Main flow

1. A user clicks on the report button.
2. The user loads a historical sample csv file.
3. Input date range.
4. Graph and statistics are presented on the page.
5. Graph is saved as PNG.

Alternate flow

1. A user clicks on the report button.
2. The user uses the previously saved csv file.
3. A From date.

4. A To date.
5. Graph and stats are presented on the page.

Exceptional flow

1. A user clicks on the report button.
2. Selects Csv file to save.
3. Does not enter date range.
4. Alert message.

<u>Termination</u>

The user clicks to another page.

<u>Post condition</u>

The must be logged in and main page loaded.

2.1.1.5. Requirement 4 Alert when sample is delayed

Description & Priority

This use case is for when the system detects a delay in a sample.

Use Case

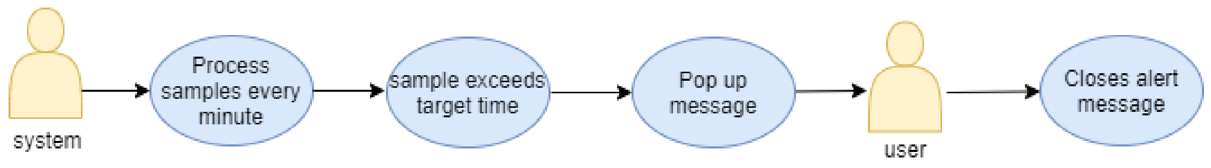


Figure 5 Alert when sample is delayed Use Case

Scope

The scope of this use case is to alert a user of a delay.

Description

This use case describes the process the system takes when a delay occurs.

Flow Description

Precondition

The user must have the application open and running on their device.

Activation

This use case starts when a user logs into the system.

Main flow

1. The system processes samples every minute.
2. A sample's time passes the expected time.
3. A popup alert message is displayed to the user.
4. The user closes the alert.

Alternate flow

Null.

Exceptional flow

1. The system checks all sample against the expected time.
2. No delay is found.

Termination

The user is notified and closes the alert screen.

Post condition

The system returns to normal state.

2.1.1.6. Requirement 5 Display sample details

Description & Priority

The use case for the user to view the sample information on the sample page.

Use Case

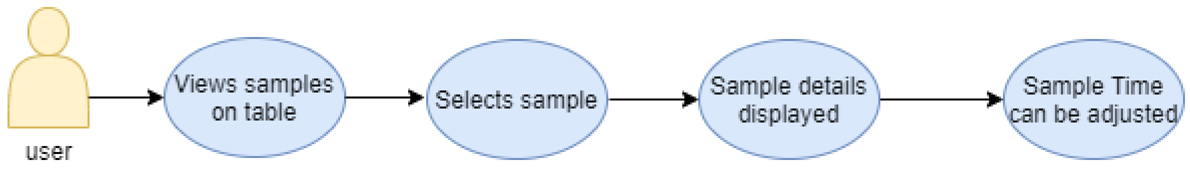


Figure 6 Display sample details Use Case

<u>Scope</u>

The scope of this use case is for a user to view samples information.

<u>Description</u>

This use case describes the process to be followed on the main page to view samples.

<u>Flow Description</u>

Precondition

The system requires a user to be logged into the system.

Activation

This use case starts when a user clicks on a sample.

Main flow

1. The user has loaded the dashboard.
2. The user clicks on a sample on the table.
3. The sample information is display.

Alternate flow

Null.

Exceptional flow

1. The user has loaded the main page.
2. There is no sample clicked.

<u>Termination</u>

The user sorted the samples by their chosen option.

<u>Post condition</u>

The system updates the sample information display.

2.1.1.7. Requirement 6 Change sample's start time

Description & Priority

The use case for the user to change the sample starting time in the table.

Use Case

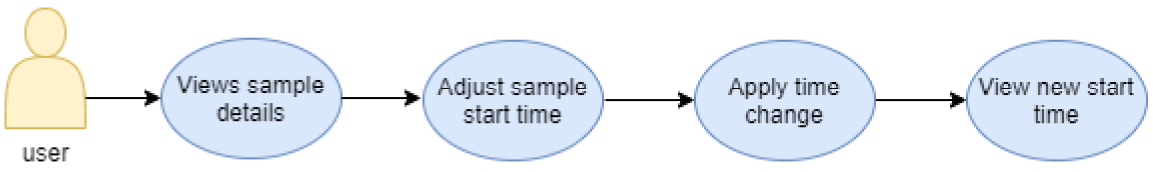


Figure 7 Change sample start time Use Case

Scope

The scope of this use case is for a user to change a sample's start time.

Description

This use case describes the process to be followed on the main dashboard page to edit sample's time.

Flow Description

Precondition

The system requires a user to be logged into the system.

Activation

This use case starts when a user clicks on a sample.

Main flow

1. The user clicks on a sample on the table.
2. The sample's information is displayed.
3. The user changes the start time.
4. The user applies the change.
5. The table gets updated.

Alternate flow

Null.

Exceptional flow

1. The user has loaded the main page.
2. There is no sample clicked.

Termination

The user sorted the samples by their chosen option.

Post condition

The system updates the sample information display.

2.1.1.8. Requirement 7 Edit test group delay time

Description & Priority

The use case for the settings page to edit a time for a sample test group.

Use Case

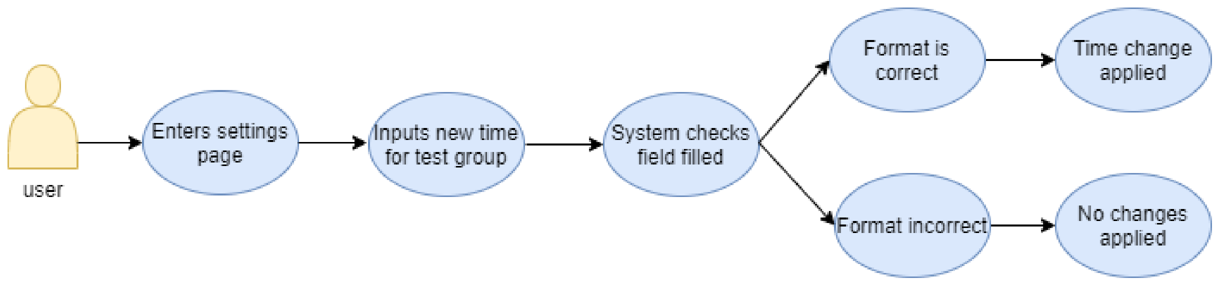


Figure 8 Edit test group delay time Use Case

Scope

The scope of this use case is to update the delay time of tests.

Description

This use case describes the process to be followed for a user to change a desired test group delay time.

Flow Description

Precondition

The system requires a user to click on the settings button from the dashboard.

Activation

This use case starts when a user clicks on the settings button.

Main flow

1. The user clicks on the settings button.
2. The user adjusts time input for desired group of tests.
3. The system checks format.
4. Format is correct.
5. Test time change is applied.

Alternate flow

Null.

Exceptional flow

1. The user clicks on the settings button.
2. The user adjusts time input for desired group of tests.
3. The system checks format.
4. Format incorrect.

5. No change applied.

<u>Termination</u>

The user must enter an input.

<u>Post condition</u>

The system updates the delay time for test groups.

2.1.1.9. Requirement 8 Add test to delay group.

Description & Priority

The use case for the user to add tests to a group in the settings page.

Use Case

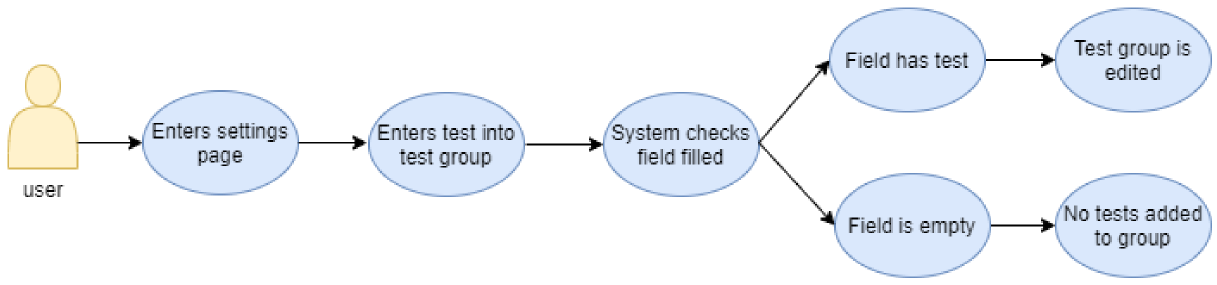


Figure 9 Add test to delay group Use Case

<u>Scope</u>

The scope of this use case is to customize the tests within the test groups for each sample to fall under.

<u>Description</u>

This use case describes the process to be followed when a user wishes to add, edit, or remove tests from a group.

<u>Flow Description</u>

Precondition

The system requires a user to be signed in and click on the settings page.

Activation

This use case starts when a user clicks on the settings page.

Main flow

1. User clicks on the settings page.
2. User clicks in desired group to edit.
3. User adds a test.
4. System checks if input box is empty.
5. Change is applied.

Alternate flow

Null.

Exceptional flow

1. User clicks on the settings page.
2. User clicks in desired group to edit.
3. User adds or removes tests.

4. User does not apply changes.

<u>Termination</u>

The user successfully changes the test group/groups.

<u>Post condition</u>

The system updates the group/groups of tests with users’ changes.

2.1.1.10. Requirement 9 Selecting csv file path for sample table.

Description & Priority

The use case for the user to add the path to a csv file for the sample table on settings page.

Use Case

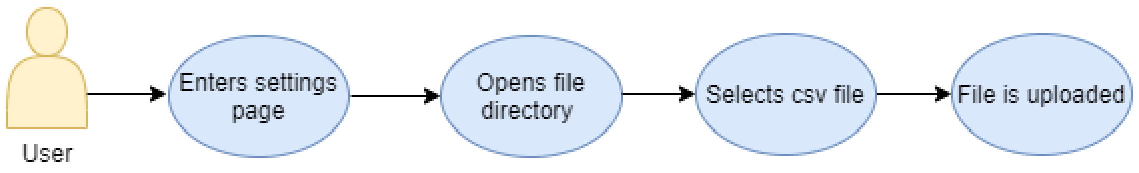


Figure 10 Selecting csv file for sample table Use Case

Scope

The scope of this use case is to select a new csv file with sample information to be monitored by the system.

Description

This use case describes the process to be followed when a user wishes to add a new or change the csv file location.

Flow Description

Precondition

The system requires a user to be signed in and click on the settings page.

Activation

This use case starts when a user clicks on the settings page.

Main flow

1. User clicks on the settings page.
2. User clicks on the set csv file path button.
3. Browses files.
4. User selects csv file.
5. The csv file path is updated.

Alternate flow

Null.

Exceptional flow

1. User clicks on the settings page.
2. User clicks on the set csv file path button.
3. Browses files.
4. User does not select a file.
5. The csv file path remains the same.

Termination

The user successfully changes the path for the csv file.

Post condition

The system reads information from the new csv file.

2.1.2. Non-Functional Requirements

Availability:

The project will need to be accessible every hour of every day of the year as hospital laboratories never halt sample processing. The availability of the project should be 100% of the time when the laboratory computer system is in use. The availability is very important as the monitoring of samples must run from when the project is launched till it is closed or the system is powered off.

Reliability:

The project should handle the general usage of features without complication. It must be able to handle a higher load of data than usually encountered. Delayed samples need to be alerted to staff consistently without fail or errors. The project needs to process and automate new sample data every minute and update the table for the user to view. It is very important that the project can be reliable, so the user is confident in the alerting features.

Usability:

Ensuring the project is easy to navigate and use is very important for providing a good user experience. New users should be easily able to tell what section they are located and what options are available to them. Each page of the system should be labelled clearly and use the same layout were possible to keep consistency. Text size and font should clearly inform the user of all relevant information and the navigation options.

Performance:

The project needs to process data and display any changes in a fast and efficient manner. Ensuring the system's speed and response time while being operated is very important for the user's experience. Live data needs to be updated every minute showing changes with the least amount of latency as possible. Functions which are performed on the data should not affect the response time of the user interface.

Recoverability:

If errors occur with data or any features the system needs to be able to handle the error and provide a solution to the problem. If there is an error with the data that is required a restart or a new path to be set, the user should be informed. It is crucial for the system to be able to correct any errors with data paths and then continue to operate.

Data Integrity:

The data used within the project is constantly updated and changed. The system needs to ensure the data is maintained in a manageable and secure way. When data is being monitored it is important that only relevant data is being selected and that the data is not being edited while functions are being run. This could lead to possible corruption of data. When the data is being processed from the files it is important that they are stored with the correct values for calculations.

2.1.3. Data Requirements

The data requirements for this program are a very important part of the program. The database in the program needs to have different tables for different sections of the program. The information from all current samples needs to be read off a file then transferred into a database. Adjustments to data will need to be completed during the transfer. The sample information then will be used for the main calculations and displays in the applications dashboard.

Another table will hold previous samples information for the statistics and reports which are generated. This database will be separated from the sample database and will be populated with historical data files. The files will need to be processed and adjusted before being inserted into the database. This reports table is needed for all reporting features and statistics as well as the graphing within the application.

User needs to have an account setup to access the system. This is to ensure people outside of the laboratory do not have access to tamper with sample data. A user can register a new account on the system by providing a name and password.

2.1.4. User Requirements

Following discussions with staff in the biochemistry laboratory of Tallaght hospital, the requirements for the project were determined. The primary requirement was to automate the labour-intensive monitoring of the progress of samples from receipt to final report. This will address a major deficiency in the functionality of the LIS. They also wish to set targets for expected times for tests and groups of tests contained within samples. The minimum alert required is for a pop-up dialogue box indicating the sample exceeding the target delay and the delay from sample receipt. They expressed a wish for a dashboard type screen that summarises the status of urgent samples at a glance. To avoid unnecessary alerts, they want the ability to extend the target delay time for specific samples. This will allow time for the sample process to be restarted, so the expected results time will be different to a sample that does not have to be restarted.

Laboratory management also wish to automate the statistical analysis of TAT data and have graphical reports generated that can be used to ensure they adhere to expected performance. This is a requirement for continued accreditation. They require analysis of TAT for each day and month. They need summary data on the total number of samples analysed the number and percentage exceeding the expected TAT. Trending analysis is also desired to highlight deterioration or improvements in service delivery.

2.1.5. Environmental Requirements

This project was designed for use on windows OS machines. To run the project on a system an install folder needs to be downloaded onto the system. This folder contains all the needed dependencies for the project to executed.

For the program to work as intended, the system it is installed on will need to have the correctly formatted csv files on it. Without these files the system will not be able to provide statistics or monitoring of samples. Example of csv files can be found in appendix 6.3.

2.1.6. Usability Requirements

The dashboard has a minimalistic layout to reduce an overload of information to the user. Each section on the dashboard has a consistent colour scheme and layout for a better user experience. All pages are named and outlined clearly with only relevant features displayed in each section.

Error handling clearly outlines any problem which has occurred. This will notify the user of the error details so it may be resolved. The alert system will be able to clearly inform the user of a delay and the details of the delayed item.

The application should load and display information quickly on most modern systems. Reducing loading and waiting time for a user helps to increase the users overall experience with the program. The programs design means it is not too intrusive for the user with no flashing screens or bright colours which could distract the user from any current task.

2.2. Design & Architecture

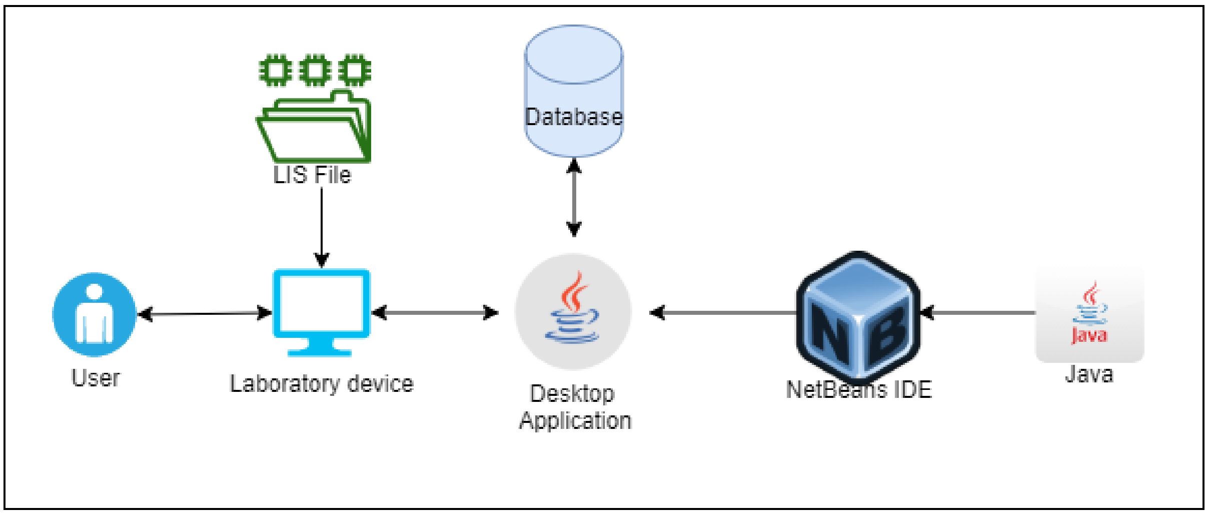


Figure 11 System Architecture

The user's system opens the desktop executable file. The file is linked to the code base and has the database connection. The user's system has the LIS file produced onto it. This file is fed from the system into the database. The database is linked with the application where the user can edit queries sent by the codebase. The SQLite database is constantly updated with files produced on the user's system.

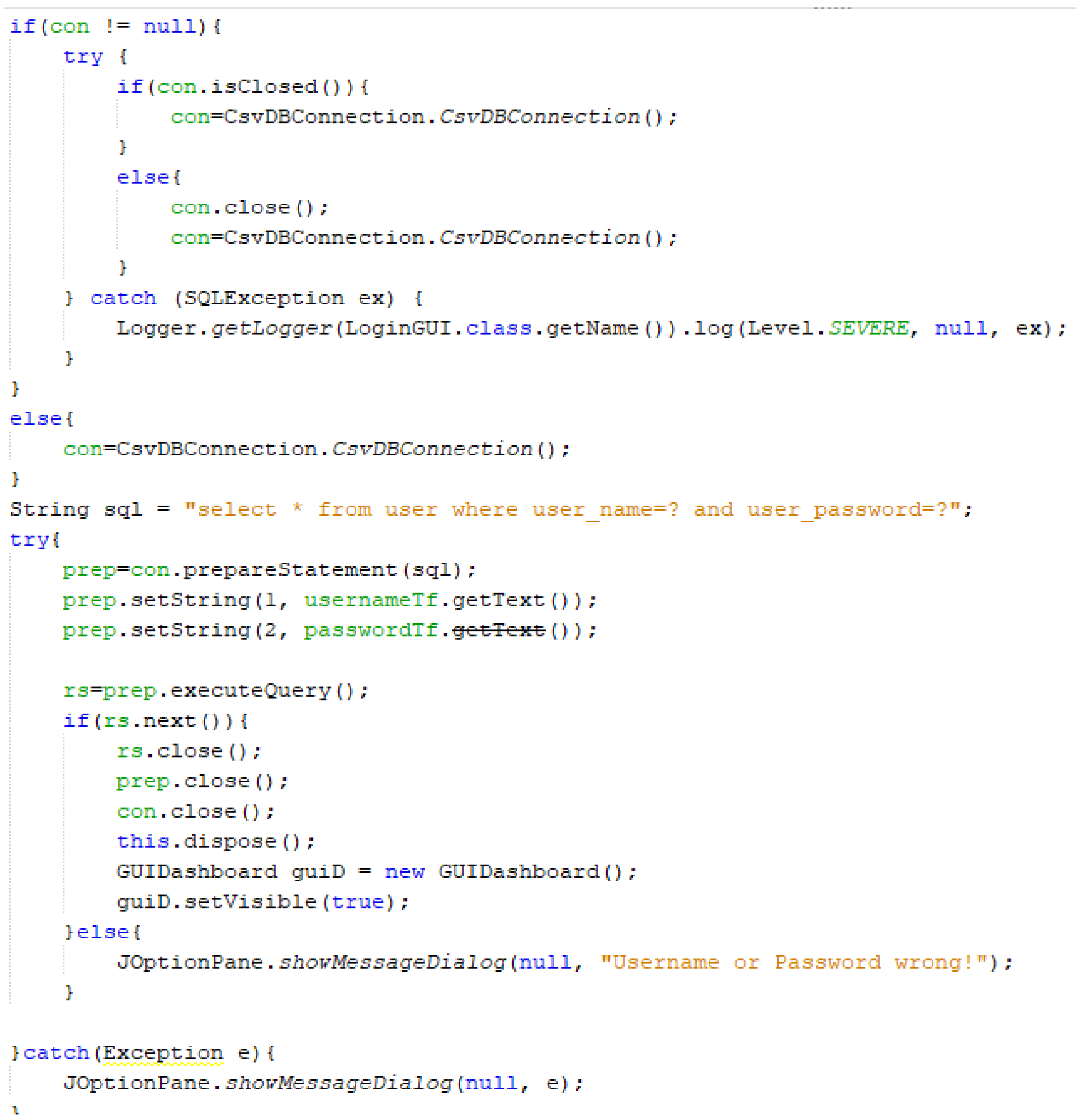
When a user inputs a command the program processes it through the java code then queries are run on the database if required. The user's dashboard is then updated with the request.

The Desktop executable loads the GUI from the java codebase. This GUI is connected with all aspects of the system. Adjustments can be made on different pages which link with the database for running queries.

2.3.Implementation

Login and Register:

The login GUI is connected to the database to verify the log in credentials. This connection is the first point the user will interact with the database. A user can create an account with the registration system. This is connected to the same database as the login so accounts can be created and logged in straight away. The user must enter a name and password. When the user uses the login or register features a database connection needs to be opened to run the query. After the connection has successfully connected a statement needs to be prepared and passed to the database. After the process is complete the statement can be closed and the connection to the database also can be closed. If the connection to the database is not closed errors can occur when trying to access the database in another feature. See figure 12 for the login function.



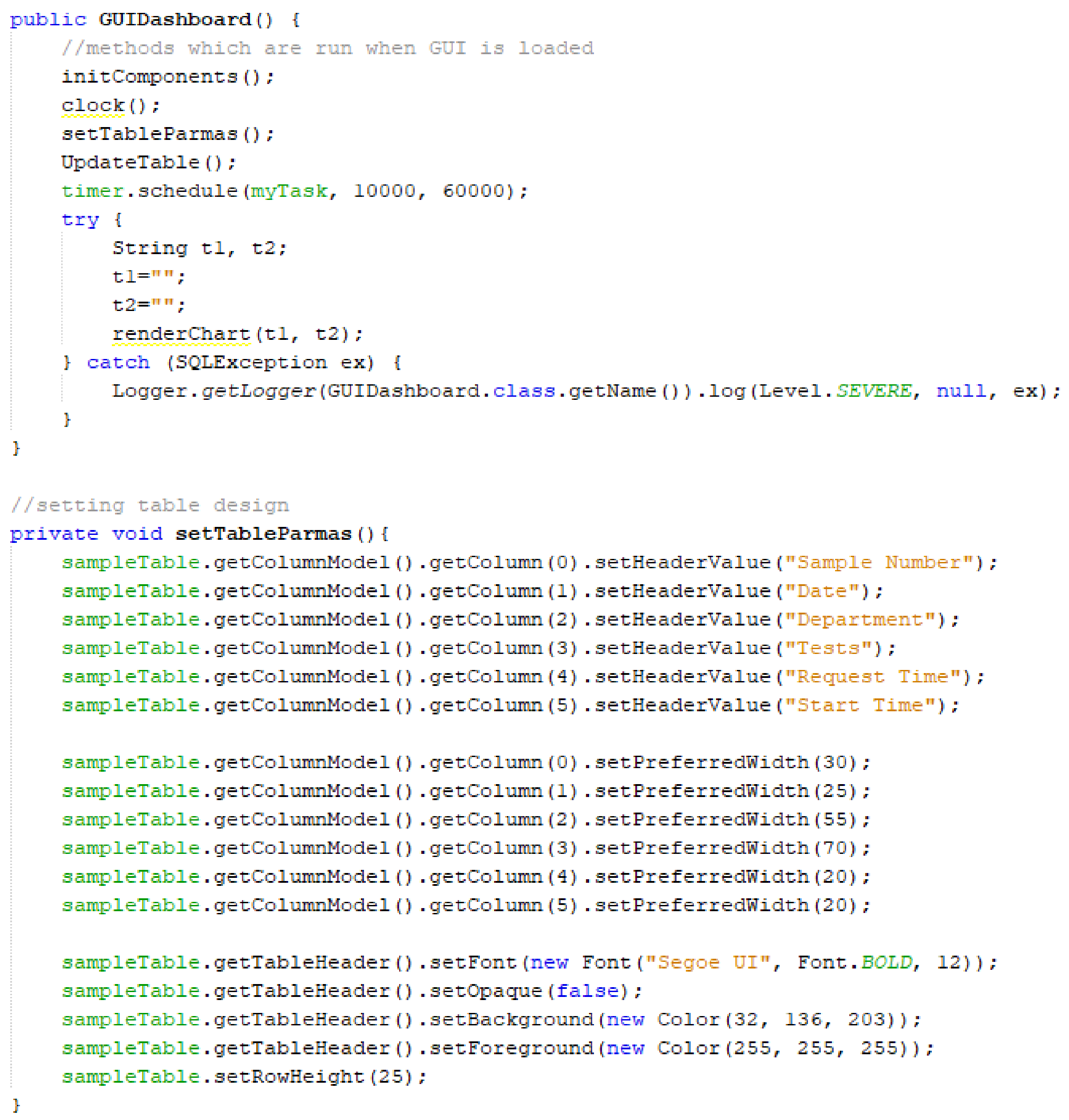
```
if(con != null){
    try {
        if(con.isClosed()){
            con=CsvDBConnection.CsvDBConnection();
        }
        else{
            con.close();
            con=CsvDBConnection.CsvDBConnection();
        }
    } catch (SQLException ex) {
        Logger.getLogger(LoginGUI.class.getName()).log(Level.SEVERE, null, ex);
    }
}
else{
    con=CsvDBConnection.CsvDBConnection();
}
String sql = "select * from user where user_name=? and user_password=?";
try{
    prep=con.prepareStatement(sql);
    prep.setString(1, usernameTf.getText());
    prep.setString(2, passwordTf.~~getText~~());

    rs=prep.executeQuery();
    if(rs.next()){
        rs.close();
        prep.close();
        con.close();
        this.dispose();
        GUIDashboard guiD = new GUIDashboard();
        guiD.setVisible(true);
    }else{
        JOptionPane.showMessageDialog(null, "Username or Password wrong!");
    }

}catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}
```

Figure 12 Login and Register code snippet

Table parameters:

The dashboard GUI is displayed after a user successfully logs in. This dashboard has multiple different panels to hide, and show select information to the user. Sample information is held in a different panel from the report and settings panels.

On the left of the sample screen a table is created with parameters set for the design and layout of the table. The table is populated with sample data from the database. The table is updated every minute with new samples. When the GUI first loads the functions for setting the table parameters and updating the table records are run. See in figure 13 from the class GUIDashboard.

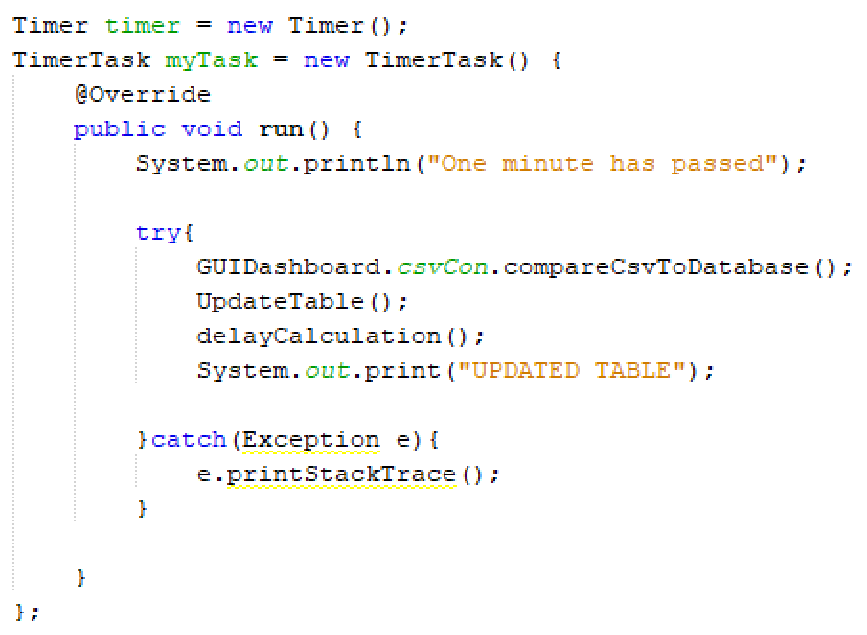


```
public GUIDashboard() {
    //methods which are run when GUI is loaded
    initComponents();
    clock();
    setTableParmas();
    UpdateTable();
    timer.schedule(myTask, 10000, 60000);
    try {
        String t1, t2;
        t1="";
        t2="";
        renderChart(t1, t2);
    } catch (SQLException ex) {
        Logger.getLogger(GUIDashboard.class.getName()).log(Level.SEVERE, null, ex);
    }
}

//setting table design
private void setTableParmas(){
    sampleTable.getColumnModel().getColumn(0).setHeaderValue("Sample Number");
    sampleTable.getColumnModel().getColumn(1).setHeaderValue("Date");
    sampleTable.getColumnModel().getColumn(2).setHeaderValue("Department");
    sampleTable.getColumnModel().getColumn(3).setHeaderValue("Tests");
    sampleTable.getColumnModel().getColumn(4).setHeaderValue("Request Time");
    sampleTable.getColumnModel().getColumn(5).setHeaderValue("Start Time");

    sampleTable.getColumnModel().getColumn(0).setPreferredWidth(30);
    sampleTable.getColumnModel().getColumn(1).setPreferredWidth(25);
    sampleTable.getColumnModel().getColumn(2).setPreferredWidth(55);
    sampleTable.getColumnModel().getColumn(3).setPreferredWidth(70);
    sampleTable.getColumnModel().getColumn(4).setPreferredWidth(20);
    sampleTable.getColumnModel().getColumn(5).setPreferredWidth(20);

    sampleTable.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 12));
    sampleTable.getTableHeader().setOpaque(false);
    sampleTable.getTableHeader().setBackground(new Color(32, 136, 203));
    sampleTable.getTableHeader().setForeground(new Color(255, 255, 255));
    sampleTable.setRowHeight(25);
}
```

Figure 13 Table parameters code snippet

Minute timer:

The timer function is located at the top of the Dashboard class and is run every minute. The function calls the compareCsvToDatabase function in the CsvDBConnection class to update the database. After the database is updated with the most recently updated samples in the csv file, the table is then updated. This loops every minute to ensure the table is constantly up to date with the latest updates in the csv file. This process allows the user to monitor the live samples. See figure 14 from the GUIDashboard class.

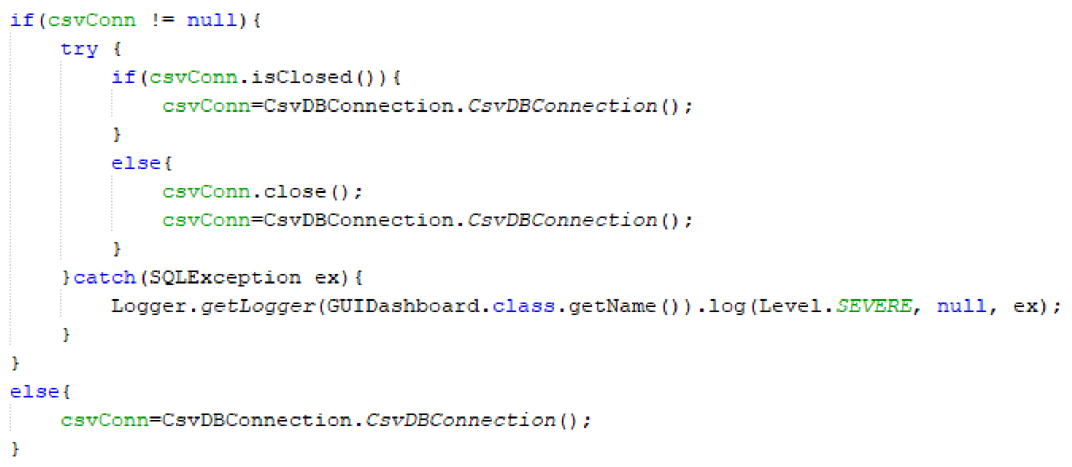


```java
Timer timer = new Timer();
TimerTask myTask = new TimerTask() {
    @Override
    public void run() {
        System.out.println("One minute has passed");

        try{
            GUIDashboard.csvCon.compareCsvToDatabase();
            UpdateTable();
            delayCalculation();
            System.out.print("UPDATED TABLE");

        }catch(Exception e){
            e.printStackTrace();
        }

    }
};
```

Figure 14 Minute timer code snippet

Database connection check:

When functions require a connection to one of the databases, a condition is first run to ensure there is not already a connection to the database that is trying to be accessed. Without doing this there can be a risk of locking the database. This happens if a pervious connection was not closed and the program is trying to create a second connection, this is not allowed and will lock the second connection from connecting to the database. This check is shown in figure 15.



```java
if(csvConn != null){
    try {
        if(csvConn.isClosed()){
            csvConn=CsvDBConnection.CsvDBConnection();
        }
        else{
            csvConn.close();
            csvConn=CsvDBConnection.CsvDBConnection();
        }
    }catch(SQLException ex){
        Logger.getLogger(GUIDashboard.class.getName()).log(Level.SEVERE, null, ex);
    }
}
else{
    csvConn=CsvDBConnection.CsvDBConnection();
}
```

Figure 15 Database connection check code snippet

Compare File with Database:

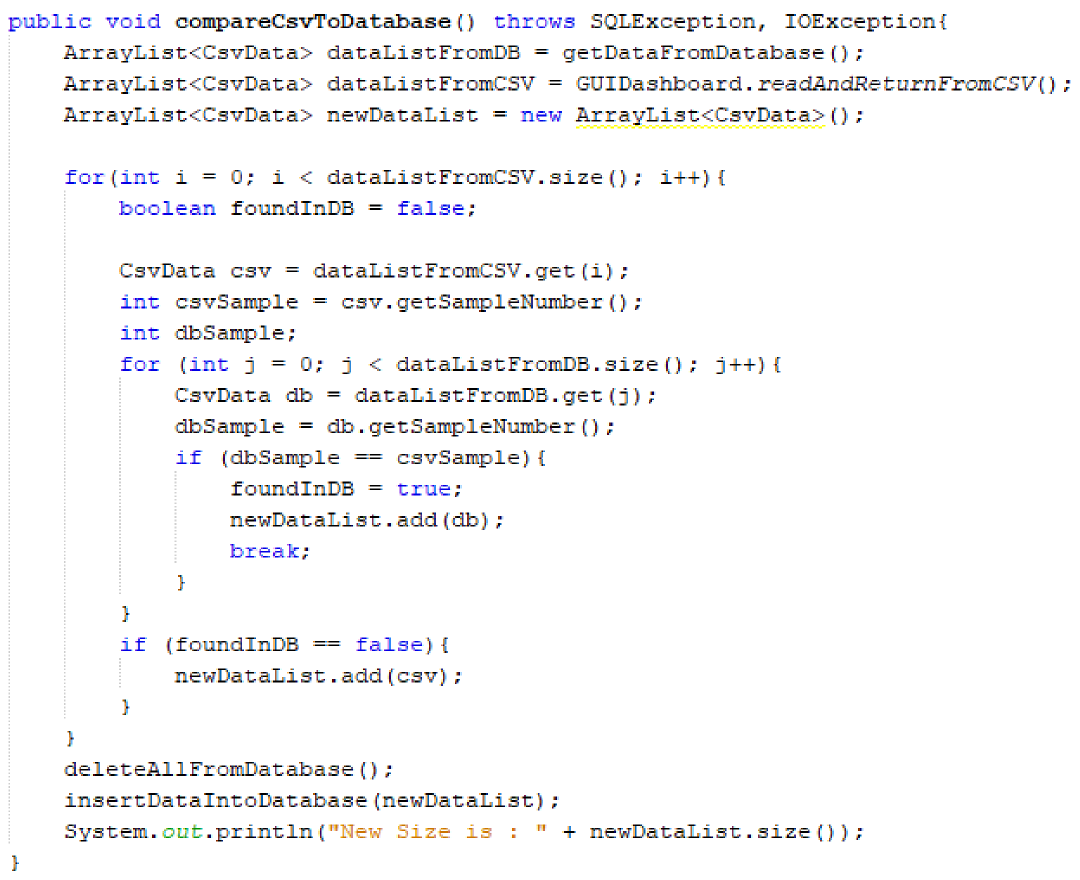
Figure 16 shows the compareCsvToDatebase function created in the CsvDBConnection class. This is a crucial component for keeping the database populated with only the current samples that are being processes. This function checks the updated csv file and compares the csv data with the database's data. This is important to ensure that the database does not continuously insert sample data.

Samples which are not present in the updated csv file but are present in the database need to be removed. This means that the sample has been processed and is no longer relevant for monitoring.

Samples which are present in both the database and the csv file need to be maintained in the database. The reason for this is to keep any adjustment the user could have applied on a sample's starting time. If the updated data from the csv file was to update the database all changes would be removed.

Any new samples which have been added into the laboratory will only be contained in the updated csv file. The new sample information needs to be added into the database with all the previous records.

This is done by using a number of ArrayLists to manage the database's data and the new data from the csv file. These records are compared and then added to a final ArrayList which is inserted into the database after the database's records are deleted.

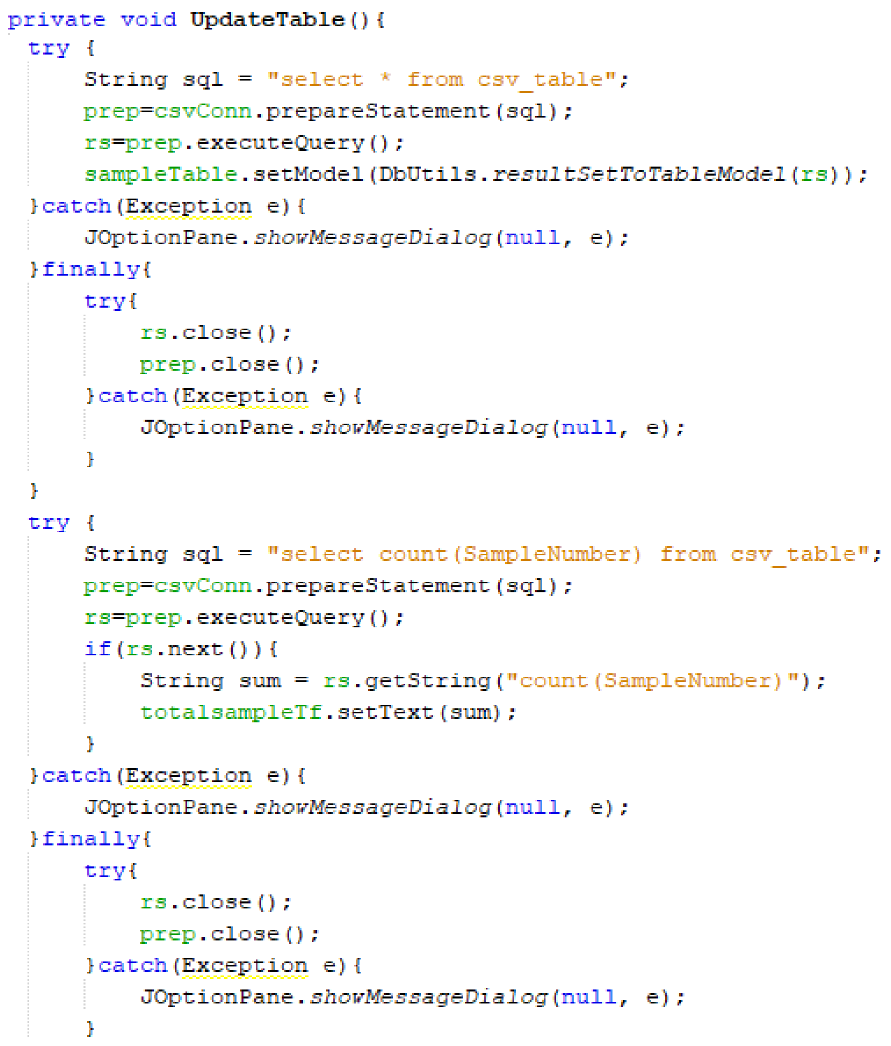


```
public void compareCsvToDatabase() throws SQLException, IOException{
    ArrayList<CsvData> dataListFromDB = getDataFromDatabase();
    ArrayList<CsvData> dataListFromCSV = GUIDashboard.readAndReturnFromCSV();
    ArrayList<CsvData> newDataList = new ArrayList<CsvData>();

    for(int i = 0; i < dataListFromCSV.size(); i++){
        boolean foundInDB = false;

        CsvData csv = dataListFromCSV.get(i);
        int csvSample = csv.getSampleNumber();
        int dbSample;
        for (int j = 0; j < dataListFromDB.size(); j++){
            CsvData db = dataListFromDB.get(j);
            dbSample = db.getSampleNumber();
            if (dbSample == csvSample){
                foundInDB = true;
                newDataList.add(db);
                break;
            }
        }
        if (foundInDB == false){
            newDataList.add(csv);
        }
    }
    deleteAllFromDatabase();
    insertDataIntoDatabase(newDataList);
    System.out.println("New Size is : " + newDataList.size());
}
```

Figure 16 Compare file with database function

Update Table:

The UpdateTable function is shown in figure 17 and is called in many different functions that affect the sample table. The function creates a database connection then prepared two statements. The first statement selects all the information from the database which is then inserted into the table on the GUI. The second statement is used to display the current count of samples being processed. After the queries have been completed the connection to the database is closed.

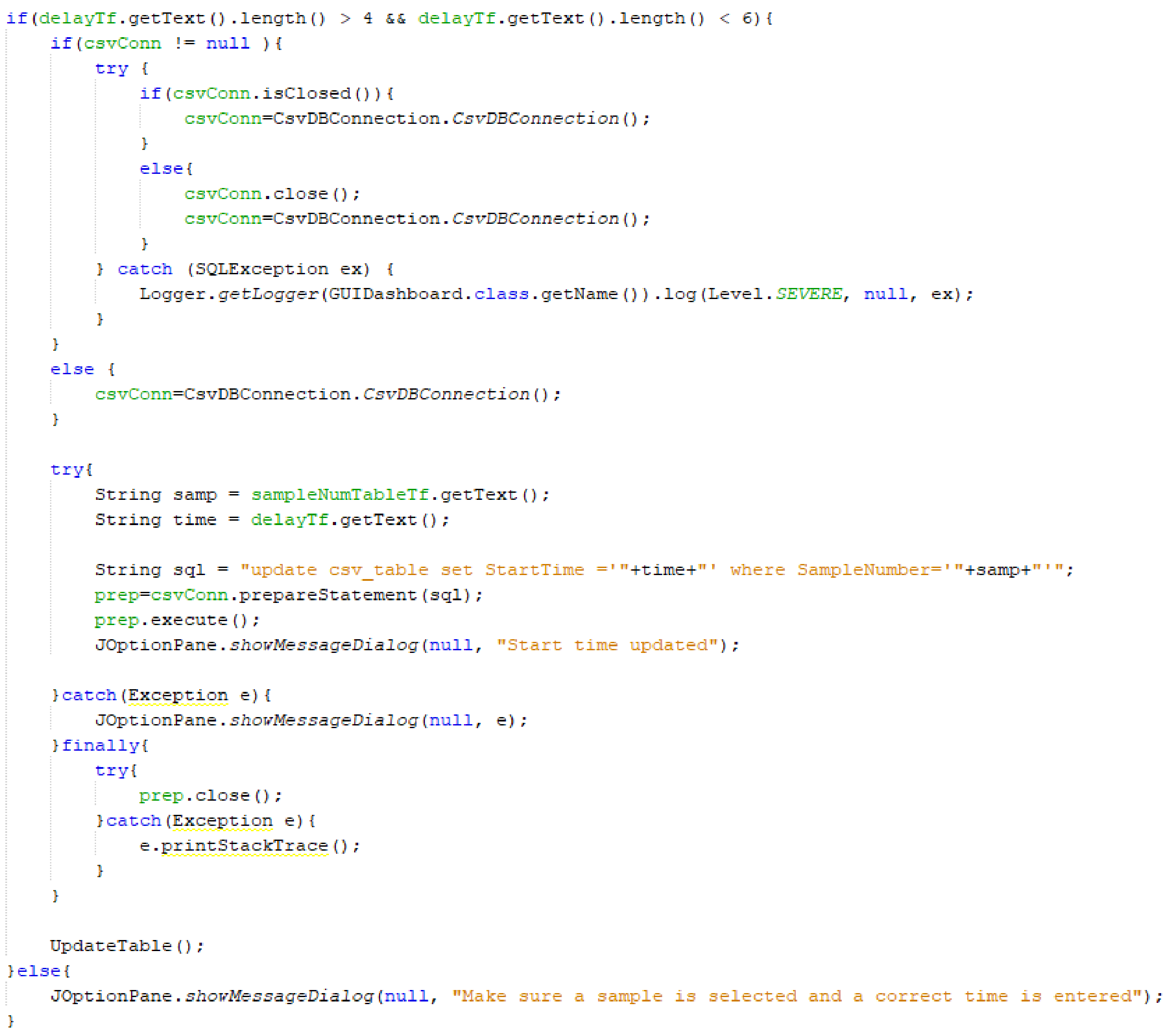


```
private void UpdateTable(){
  try {
      String sql = "select * from csv_table";
      prep=csvConn.prepareStatement(sql);
      rs=prep.executeQuery();
      sampleTable.setModel(DbUtils.resultSetToTableModel(rs));
  }catch(Exception e){
      JOptionPane.showMessageDialog(null, e);
  }finally{
      try{
          rs.close();
          prep.close();
      }catch(Exception e){
          JOptionPane.showMessageDialog(null, e);
      }
  }
  try {
      String sql = "select count(SampleNumber) from csv_table";
      prep=csvConn.prepareStatement(sql);
      rs=prep.executeQuery();
      if(rs.next()){
          String sum = rs.getString("count(SampleNumber)");
          totalsampleTf.setText(sum);
      }
  }catch(Exception e){
      JOptionPane.showMessageDialog(null, e);
  }finally{
      try{
          rs.close();
          prep.close();
      }catch(Exception e){
          JOptionPane.showMessageDialog(null, e);
      }
```

Figure 17 Update table function

Delay sample:

To allow the user to delay a sample's starting time the function in figure 18 is used. This takes the information that is presented when the user clicks on a sample in the table and allows the user to alter the start time of the selected sample. After the user enters a new time, they can click the update button to run this function for updating that sample in the database. After the sample is updated in the database the table on the Dashboard is updated to display the new starting time of the sample.

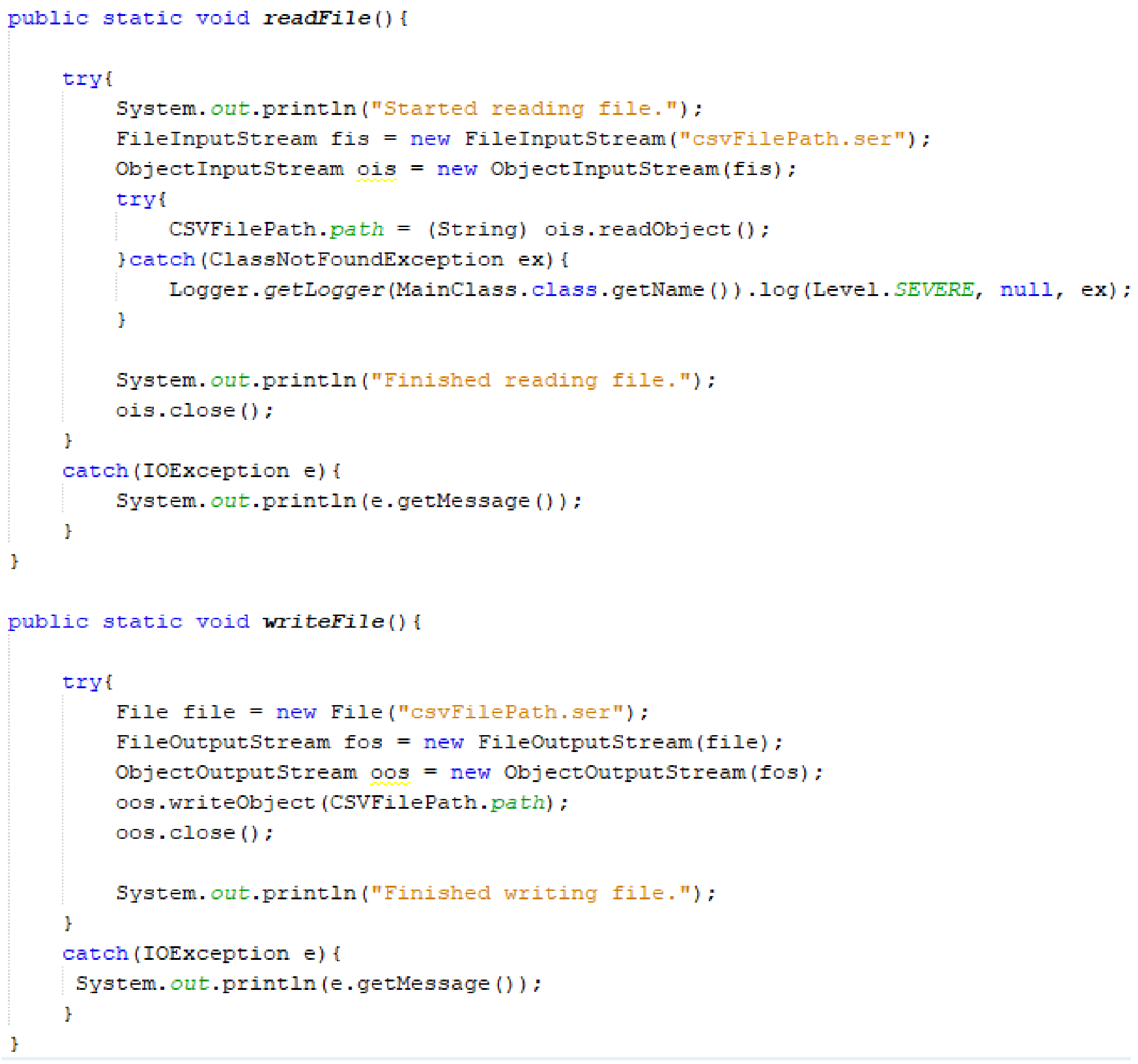


```
if(delayTf.getText().length() > 4 && delayTf.getText().length() < 6){
    if(csvConn != null ){
        try {
            if(csvConn.isClosed()){
                csvConn=CsvDBConnection.CsvDBConnection();
            }
            else{
                csvConn.close();
                csvConn=CsvDBConnection.CsvDBConnection();
            }
        } catch (SQLException ex) {
            Logger.getLogger(GUIDashboard.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    else {
        csvConn=CsvDBConnection.CsvDBConnection();
    }

    try{
        String samp = sampleNumTableTf.getText();
        String time = delayTf.getText();

        String sql = "update csv_table set StartTime ='"+time+"' where SampleNumber='"+samp+"'";
        prep=csvConn.prepareStatement(sql);
        prep.execute();
        JOptionPane.showMessageDialog(null, "Start time updated");

    }catch(Exception e){
        JOptionPane.showMessageDialog(null, e);
    }finally{
        try{
            prep.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    UpdateTable();
}else{
    JOptionPane.showMessageDialog(null, "Make sure a sample is selected and a correct time is entered");
}
```

Figure 18 Delay sample code snippet

Read and write file:

Read and write file functions in figure 19 are used for the sample table data. The user can save a new file path on their machine if they wish to monitor a different csv file. The read method is used for checking the csv file using the saved path for updates and updating the database if there are any changes to samples. The writeFile function saves the file path in a csvFilePath.ser file on the local machine. If the file path is changed by the user, the csvFilePath.ser file is updated with the new path. This function can be accessed in the settings page of the dashboard.

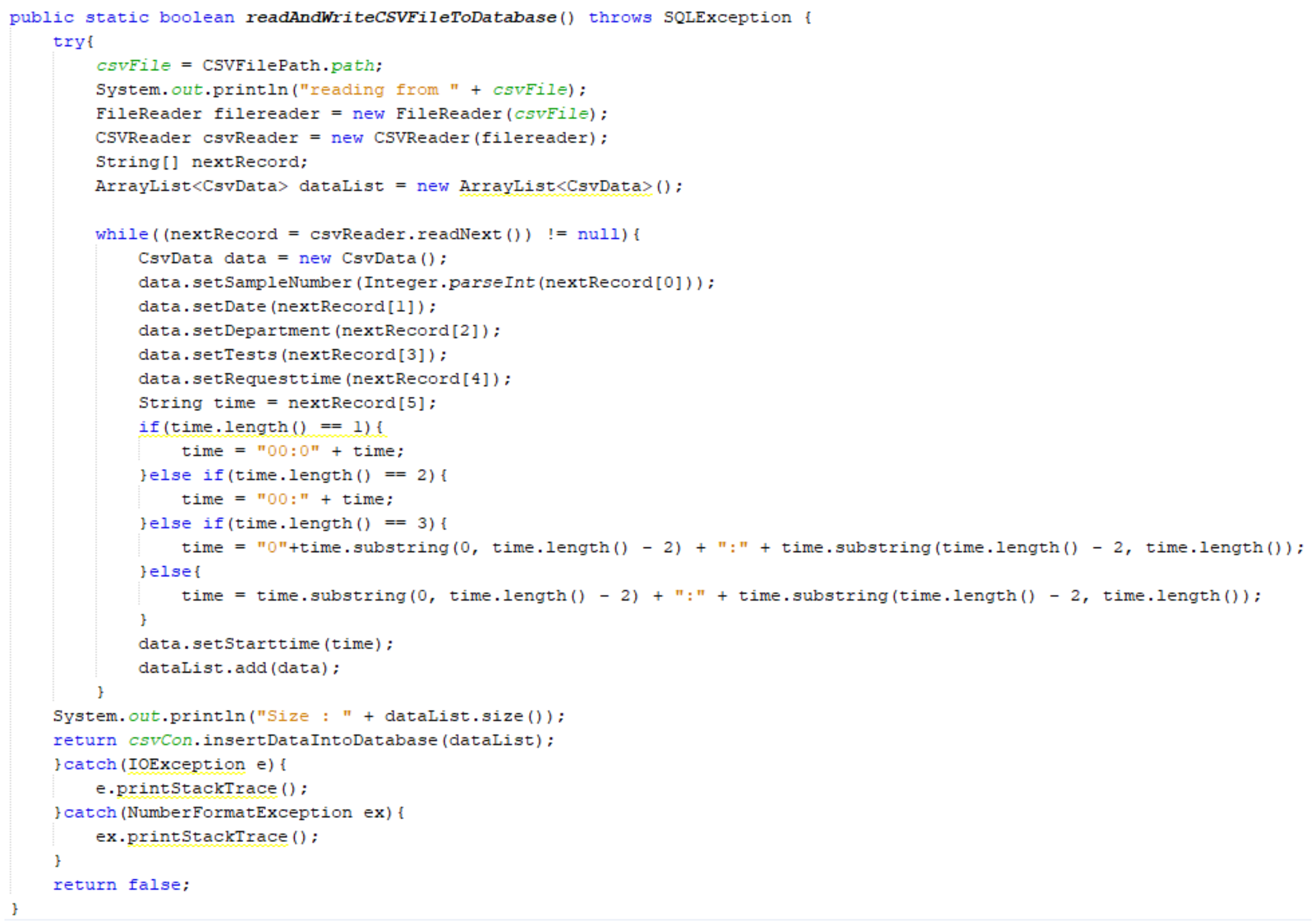


```
public static void readFile(){

    try{
        System.out.println("Started reading file.");
        FileInputStream fis = new FileInputStream("csvFilePath.ser");
        ObjectInputStream ois = new ObjectInputStream(fis);
        try{
            CSVFilePath.path = (String) ois.readObject();
        }catch(ClassNotFoundException ex){
            Logger.getLogger(MainClass.class.getName()).log(Level.SEVERE, null, ex);
        }

        System.out.println("Finished reading file.");
        ois.close();
    }
    catch(IOException e){
        System.out.println(e.getMessage());
    }
}

public static void writeFile(){

    try{
        File file = new File("csvFilePath.ser");
        FileOutputStream fos = new FileOutputStream(file);
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.writeObject(CSVFilePath.path);
        oos.close();

        System.out.println("Finished writing file.");
    }
    catch(IOException e){
     System.out.println(e.getMessage());
    }
}
```

Figure 19 Read and Write file functions

Load csv file from path set by user:

The function in figure 20 is run to load all the sample data from the csv file into the database. This function needs to process every record and assign the correct format to the starting time values. After all the records are processed, they are sent to the CsvDBConnection class to be compared to the database records. These records are displayed on the table on the sample screen.

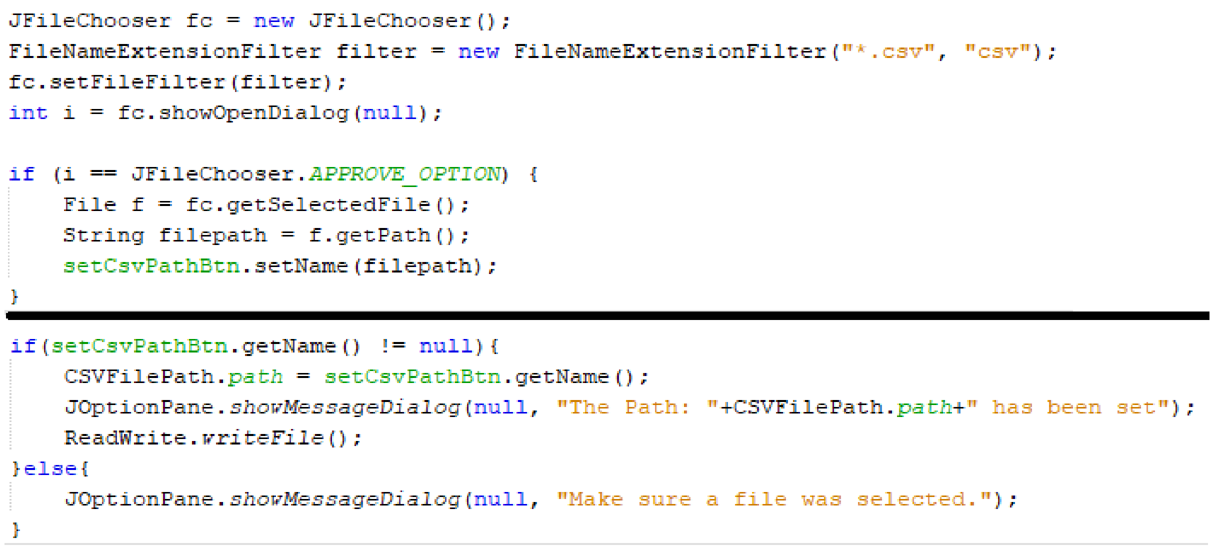


```java
public static boolean readAndWriteCSVFileToDatabase() throws SQLException {
    try{
        csvFile = CSVFilePath.path;
        System.out.println("reading from " + csvFile);
        FileReader filereader = new FileReader(csvFile);
        CSVReader csvReader = new CSVReader(filereader);
        String[] nextRecord;
        ArrayList<CsvData> dataList = new ArrayList<CsvData>();

        while((nextRecord = csvReader.readNext()) != null){
            CsvData data = new CsvData();
            data.setSampleNumber(Integer.parseInt(nextRecord[0]));
            data.setDate(nextRecord[1]);
            data.setDepartment(nextRecord[2]);
            data.setTests(nextRecord[3]);
            data.setRequesttime(nextRecord[4]);
            String time = nextRecord[5];
            if(time.length() == 1){
                time = "00:0" + time;
            }else if(time.length() == 2){
                time = "00:" + time;
            }else if(time.length() == 3){
                time = "0"+time.substring(0, time.length() - 2) + ":" + time.substring(time.length() - 2, time.length());
            }else{
                time = time.substring(0, time.length() - 2) + ":" + time.substring(time.length() - 2, time.length());
            }
            data.setStarttime(time);
            dataList.add(data);
        }
    System.out.println("Size : " + dataList.size());
    return csvCon.insertDataIntoDatabase(dataList);
    }catch(IOException e){
        e.printStackTrace();
    }catch(NumberFormatException ex){
        ex.printStackTrace();
    }
    return false;
}
```

Figure 20 Read and Write CSV file to database function

Find csv file and save path:

The snippets of code in figure 21 shows the process followed when a user wishes to save a csv file used for the sample table. The user is presented with a file explorer to which is used to locate the path to the file. After the file is selected the path is taken and passed to the Writefile function to save the path.



```
JFileChooser fc = new JFileChooser();
FileNameExtensionFilter filter = new FileNameExtensionFilter("*.csv", "csv");
fc.setFileFilter(filter);
int i = fc.showOpenDialog(null);

if (i == JFileChooser.APPROVE_OPTION) {
    File f = fc.getSelectedFile();
    String filepath = f.getPath();
    setCsvPathBtn.setName(filepath);
}
```

```
if(setCsvPathBtn.getName() != null){
    CSVFilePath.path = setCsvPathBtn.getName();
    JOptionPane.showMessageDialog(null, "The Path: "+CSVFilePath.path+" has been set");
    ReadWrite.writeFile();
}else{
    JOptionPane.showMessageDialog(null, "Make sure a file was selected.");
}
```

Figure 21 Csv file explorer and path save code snippet

Report data insert:

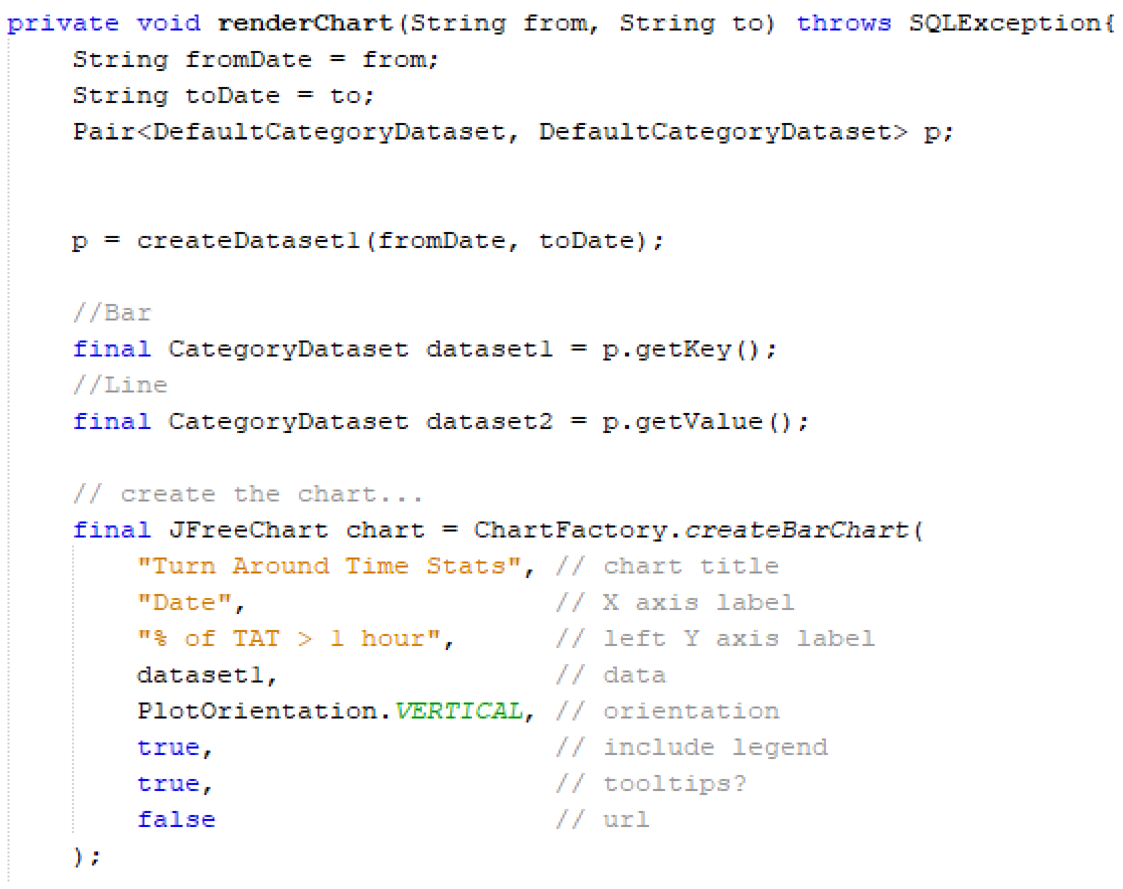
The data for the report screen first is selected with a file explorer that allows the user to select a csv file. This file is then passed into the readAndWriteCSVFileToReportDatabase function to process records, so they are in the correct format. After all the records have been processed an ArrayList is sent to the insertDataIntoDatabase function in the ReportDBConnection class. This function takes the ArrayList and inserts all records into the report table in the database. This database now can be queried on a date range for the graphing and statistics.

Render Graph:

The renderChart function in figure 22 takes in two parameters for selecting the date range set by the user. The date from and to are passed into a createDateset function used to calculate statistics from the date range for every date. This is then returned, and values are placed into a Pair, for the Line and Bar plots.

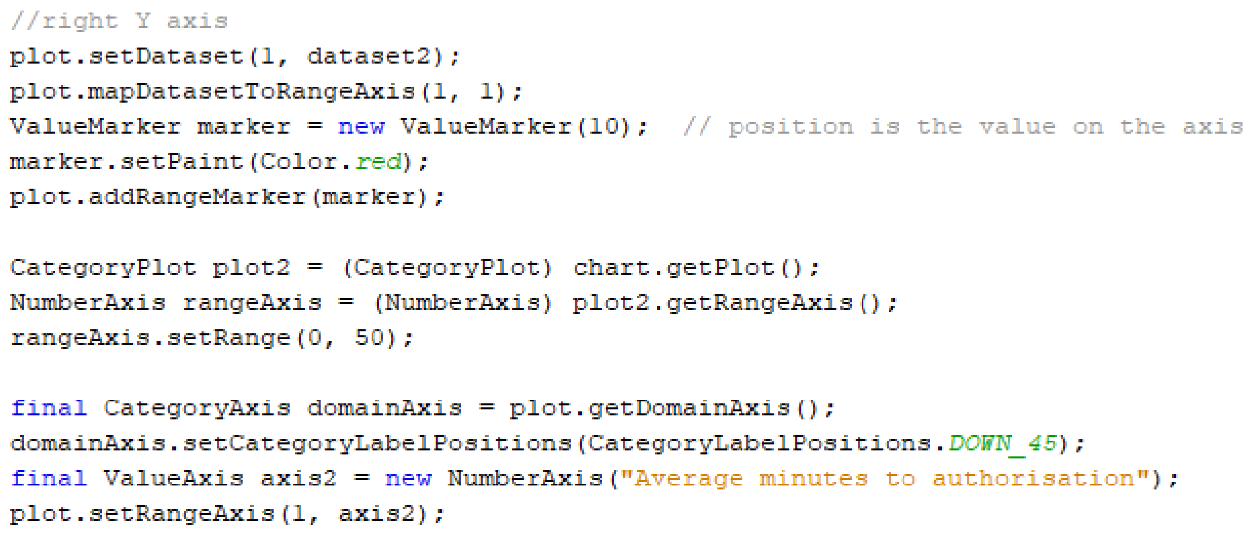
Two CategoryDatasets are created for the two plots and fed into the graph. The graph has the parameters set in the ChartFactory function when creating the chart variable. This is where the Bar plot is created for the “% over an hour” statistics. In figure 23 the second dataset is used to display the line plot with the “average minutes to authorization” values from the historical samples. The chart area is designed within this function as well, setting the values for the Y axis and having a horizonal line at the 10% value for the bar plot. This is to make it easier to see when a day has gone over the 10% delay recommendation.

The end of the renderChart function in figure 24 saves the graph to a PNG file on the local system. The chart is also set here so it cannot be zoomed in by the user. The chart is then placed into a panel set on the report screen.



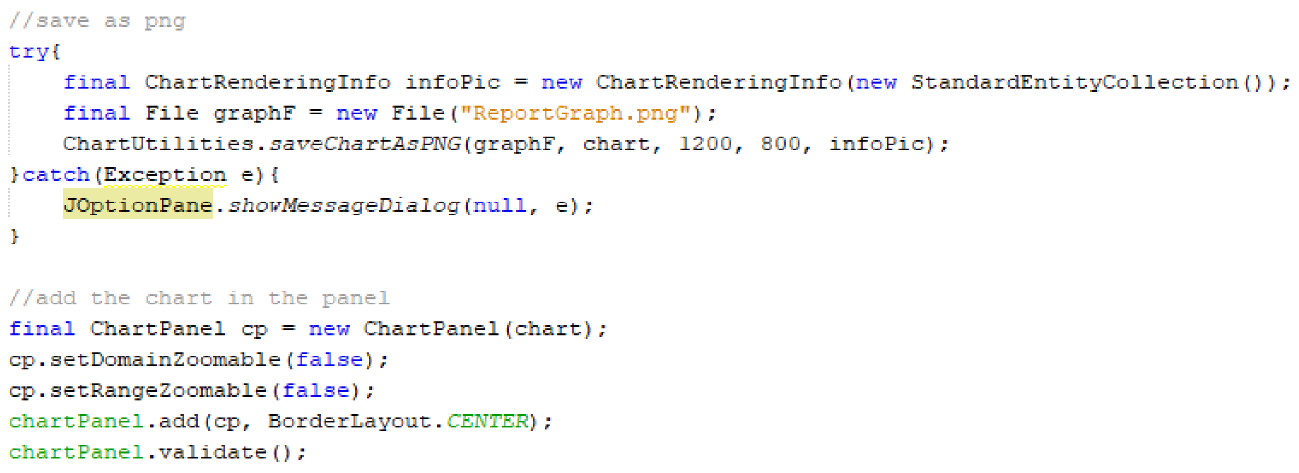
```java
private void renderChart(String from, String to) throws SQLException{
    String fromDate = from;
    String toDate = to;
    Pair<DefaultCategoryDataset, DefaultCategoryDataset> p;


    p = createDataset1(fromDate, toDate);

    //Bar
    final CategoryDataset dataset1 = p.getKey();
    //Line
    final CategoryDataset dataset2 = p.getValue();

    // create the chart...
    final JFreeChart chart = ChartFactory.createBarChart(
        "Turn Around Time Stats", // chart title
        "Date",                   // X axis label
        "% of TAT > 1 hour",      // left Y axis label
        dataset1,                 // data
        PlotOrientation.VERTICAL, // orientation
        true,                     // include legend
        true,                     // tooltips?
        false                     // url
    );
```

Figure 22 Render Chart function snippet



```java
//right Y axis
plot.setDataset(1, dataset2);
plot.mapDatasetToRangeAxis(1, 1);
ValueMarker marker = new ValueMarker(10);  // position is the value on the axis
marker.setPaint(Color.red);
plot.addRangeMarker(marker);

CategoryPlot plot2 = (CategoryPlot) chart.getPlot();
NumberAxis rangeAxis = (NumberAxis) plot2.getRangeAxis();
rangeAxis.setRange(0, 50);

final CategoryAxis domainAxis = plot.getDomainAxis();
domainAxis.setCategoryLabelPositions(CategoryLabelPositions.DOWN_45);
final ValueAxis axis2 = new NumberAxis("Average minutes to authorisation");
plot.setRangeAxis(1, axis2);
```

Figure 23 Render Chart function snippet



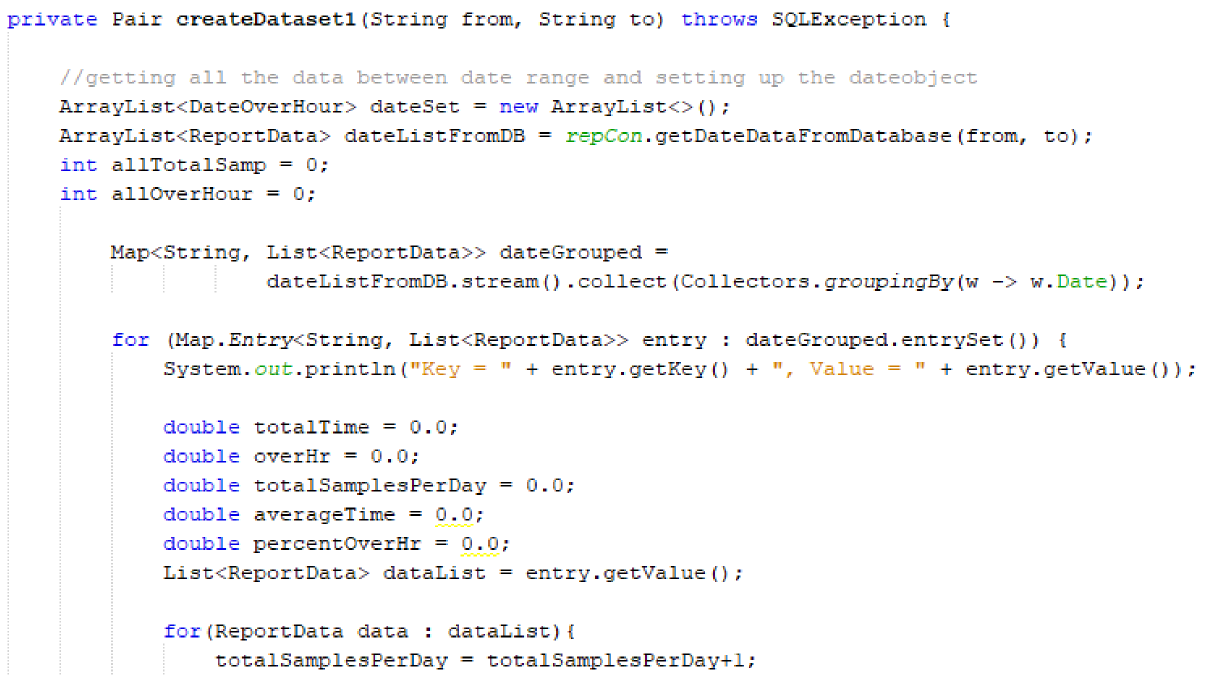
```java
//save as png
try{
    final ChartRenderingInfo infoPic = new ChartRenderingInfo(new StandardEntityCollection());
    final File graphF = new File("ReportGraph.png");
    ChartUtilities.saveChartAsPNG(graphF, chart, 1200, 800, infoPic);
}catch(Exception e){
    JOptionPane.showMessageDialog(null, e);
}

//add the chart in the panel
final ChartPanel cp = new ChartPanel(chart);
cp.setDomainZoomable(false);
cp.setRangeZoomable(false);
chartPanel.add(cp, BorderLayout.CENTER);
chartPanel.validate();
```

Figure 24 Render Chart function snippet

Making datasets:

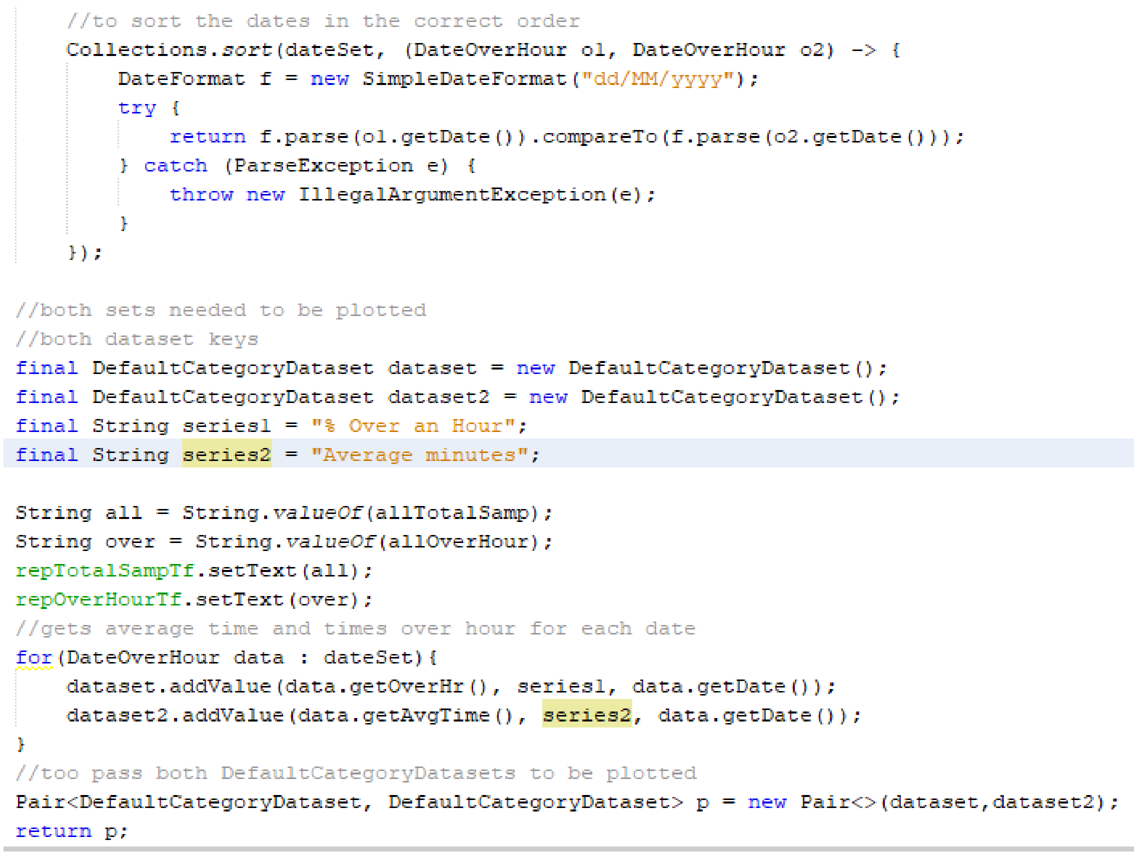
An ArrayList is created and populated with the data from the database with records in between the dates set on the report page. This ArrayList is then put into a Map with the Key of the Map being the Date value from each value in the ArrayList. This results in the Map containing all the data split by the different Date values. The Date key is then looped over so all the data on each individual day can be summarised. This is shown in figure 25.

The summarised data is organised in a ArrayList called dataSet which holds an object for holding the statistics needed to graph, this is then sorted by date. This is then passed into two DefaultCategoryDataset's which are used to display the statistics on the graph. A pair of the datasets is then returned to the renderChart method. This code can be found in figure 26.



```java
private Pair createDataset1(String from, String to) throws SQLException {

    //getting all the data between date range and setting up the dateobject
    ArrayList<DateOverHour> dateSet = new ArrayList<>();
    ArrayList<ReportData> dateListFromDB = repCon.getDateDataFromDatabase(from, to);
    int allTotalSamp = 0;
    int allOverHour = 0;

        Map<String, List<ReportData>> dateGrouped =
                    dateListFromDB.stream().collect(Collectors.groupingBy(w -> w.Date));

        for (Map.Entry<String, List<ReportData>> entry : dateGrouped.entrySet()) {
            System.out.println("Key = " + entry.getKey() + ", Value = " + entry.getValue());

            double totalTime = 0.0;
            double overHr = 0.0;
            double totalSamplesPerDay = 0.0;
            double averageTime = 0.0;
            double percentOverHr = 0.0;
            List<ReportData> dataList = entry.getValue();

            for(ReportData data : dataList){
                totalSamplesPerDay = totalSamplesPerDay+1;
```

Figure 25 Create Dataset function snippet



```java
    //to sort the dates in the correct order
    Collections.sort(dateSet, (DateOverHour o1, DateOverHour o2) -> {
        DateFormat f = new SimpleDateFormat("dd/MM/yyyy");
        try {
            return f.parse(o1.getDate()).compareTo(f.parse(o2.getDate()));
        } catch (ParseException e) {
            throw new IllegalArgumentException(e);
        }
    });

//both sets needed to be plotted
//both dataset keys
final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
final DefaultCategoryDataset dataset2 = new DefaultCategoryDataset();
final String series1 = "% Over an Hour";
final String series2 = "Average minutes";

String all = String.valueOf(allTotalSamp);
String over = String.valueOf(allOverHour);
repTotalSampTf.setText(all);
repOverHourTf.setText(over);
//gets average time and times over hour for each date
for(DateOverHour data : dateSet){
    dataset.addValue(data.getOverHr(), series1, data.getDate());
    dataset2.addValue(data.getAvgTime(), series2, data.getDate());
}
//too pass both DefaultCategoryDatasets to be plotted
Pair<DefaultCategoryDataset, DefaultCategoryDataset> p = new Pair<>(dataset,dataset2);
return p;
```

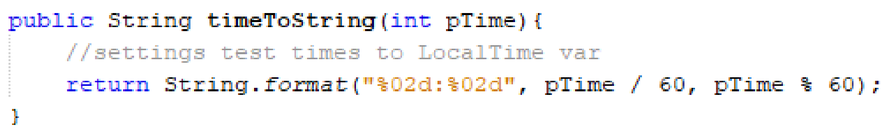
Figure 26 Create Dataset function snippet

Alerting feature:

The alert feature is run every minute when the samples are compared with the csv file. The delayCalculation function takes all the tests and times in the current samples in the database then calculates a delay time based on these values. This delay time is compared to the system's local time to check if a sample has exceeded its expected outputted time.

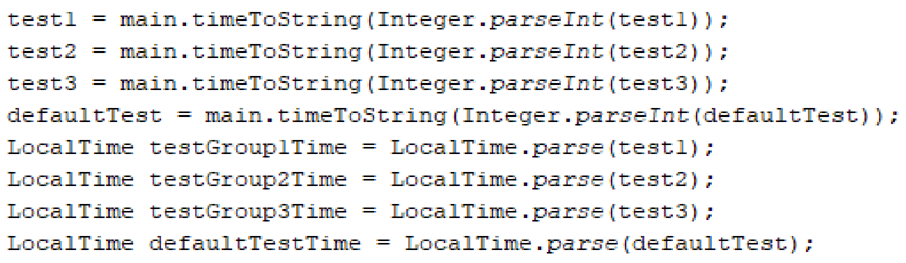
Each sample is checked to see if any of its test values have been entered in any of the test groups on the settings page. If a sample contains a test in a test group, the time assigned to the test group gets applied on that specific sample. If a sample has multiple tests which are found in different groups. The test group with the highest time value is set for the samples delay timer. The format of the test group time and sample start time need to be altered so they can be converted to a LocalTime value. This LocalTime is used for adding onto the sample's starting time. See figure 27 and 28 to see how the LocalTime is applied.

After the new time value has been added onto the sample's start time the time is compared with the system time. The date of the sample is used with the time to calculate if a sample has been delayed. Figure 29 shows the function used for calculating the time.



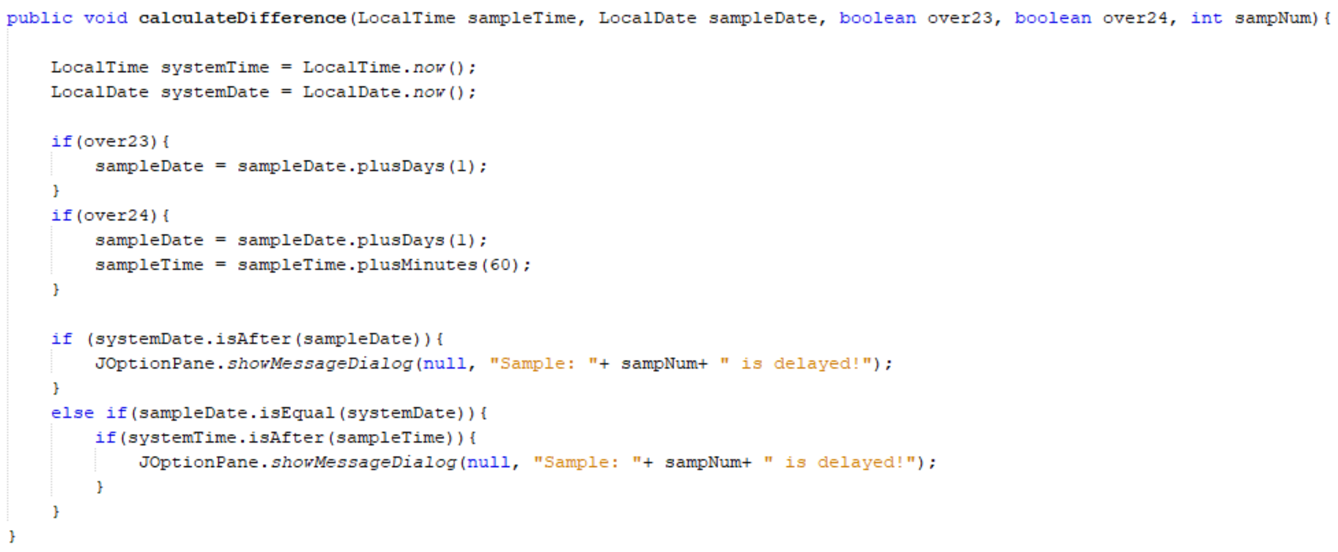
```
public String timeToString(int pTime){
    //settings test times to LocalTime var
    return String.format("%02d:%02d", pTime / 60, pTime % 60);
}
```

Figure 27 Time To String function



```
test1 = main.timeToString(Integer.parseInt(test1));
test2 = main.timeToString(Integer.parseInt(test2));
test3 = main.timeToString(Integer.parseInt(test3));
defaultTest = main.timeToString(Integer.parseInt(defaultTest));
LocalTime testGroup1Time = LocalTime.parse(test1);
LocalTime testGroup2Time = LocalTime.parse(test2);
LocalTime testGroup3Time = LocalTime.parse(test3);
LocalTime defaultTestTime = LocalTime.parse(defaultTest);
```

Figure 28 Tests group time to LocalTime code snippet



```
public void calculateDifference(LocalTime sampleTime, LocalDate sampleDate, boolean over23, boolean over24, int sampNum){

    LocalTime systemTime = LocalTime.now();
    LocalDate systemDate = LocalDate.now();

    if(over23){
        sampleDate = sampleDate.plusDays(1);
    }
    if(over24){
        sampleDate = sampleDate.plusDays(1);
        sampleTime = sampleTime.plusMinutes(60);
    }

    if (systemDate.isAfter(sampleDate)){
        JOptionPane.showMessageDialog(null, "Sample: "+ sampNum+ " is delayed!");
    }
    else if(sampleDate.isEqual(systemDate)){
        if(systemTime.isAfter(sampleTime)){
            JOptionPane.showMessageDialog(null, "Sample: "+ sampNum+ " is delayed!");
        }
    }
}
```

Figure 29 Calculate Difference function

2.4. Graphical User Interface (GUI)

Login:

The User log in screen gives the option for the user to input a username and password. This username and password are then checked in the database to verify after the user clicks the login button. The user will be given an error message if they entered in an invalid Username or Password. After the User successfully logs in the log in screen will close and open up the Dashboard. See figure 30.

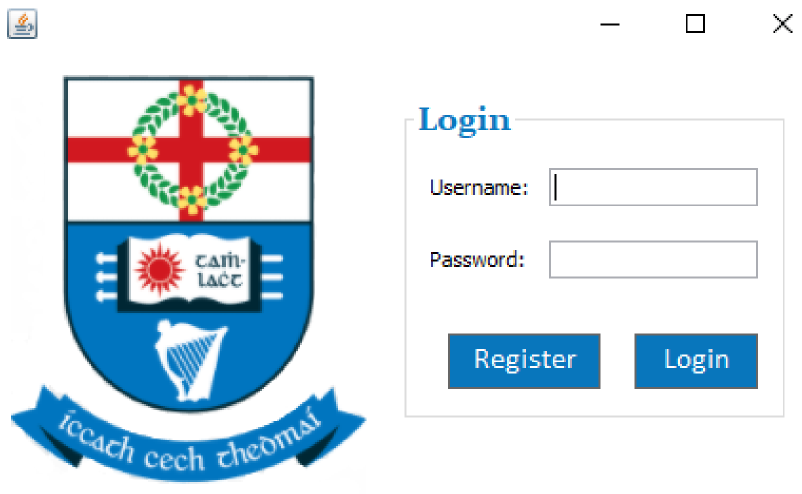


Figure 30 Login screen

Register:

The Registration screen shown in figure 31 has input fields the user needs to fill in to make an account. These fields are sent to the database and verified. An account will be created if all the fields are valid. The user can click the Register button after all fields are completed. This window will then close and go back to the Login page.

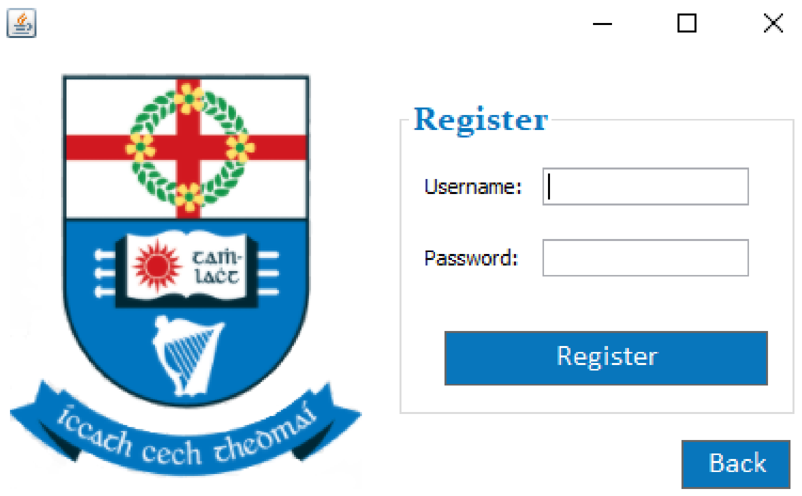


Figure 31 Register screen

Dashboard menu:

After a user successfully logs in the dashboard menu will open and the user is greeted with a welcome message. The top of the dashboard displays the current time and date. The centre of the menu has buttons to each section of the program. The dashboard can be minimized and moved around the screen by clicking and dragging top bar. The user can then decide which section they wish to view from this menu. See figure 32.

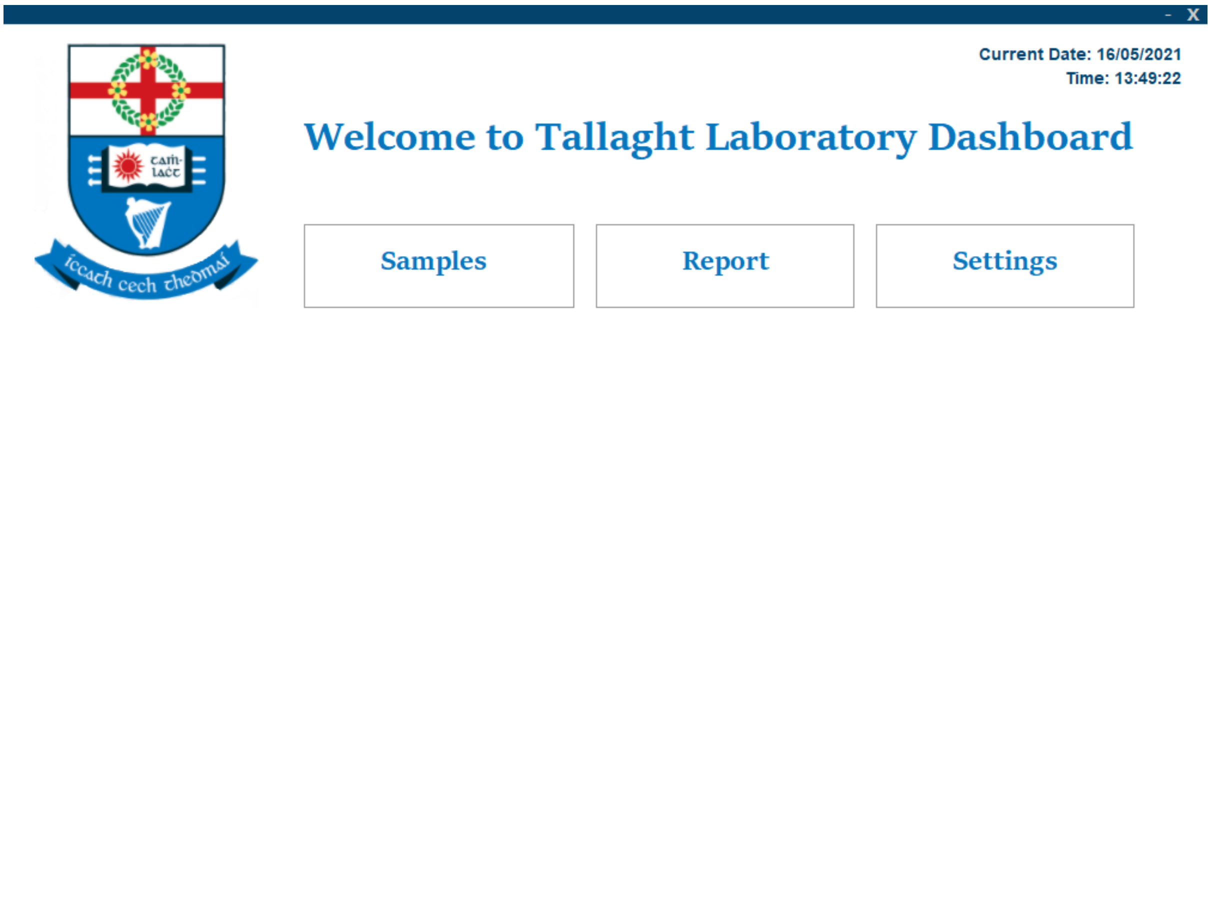
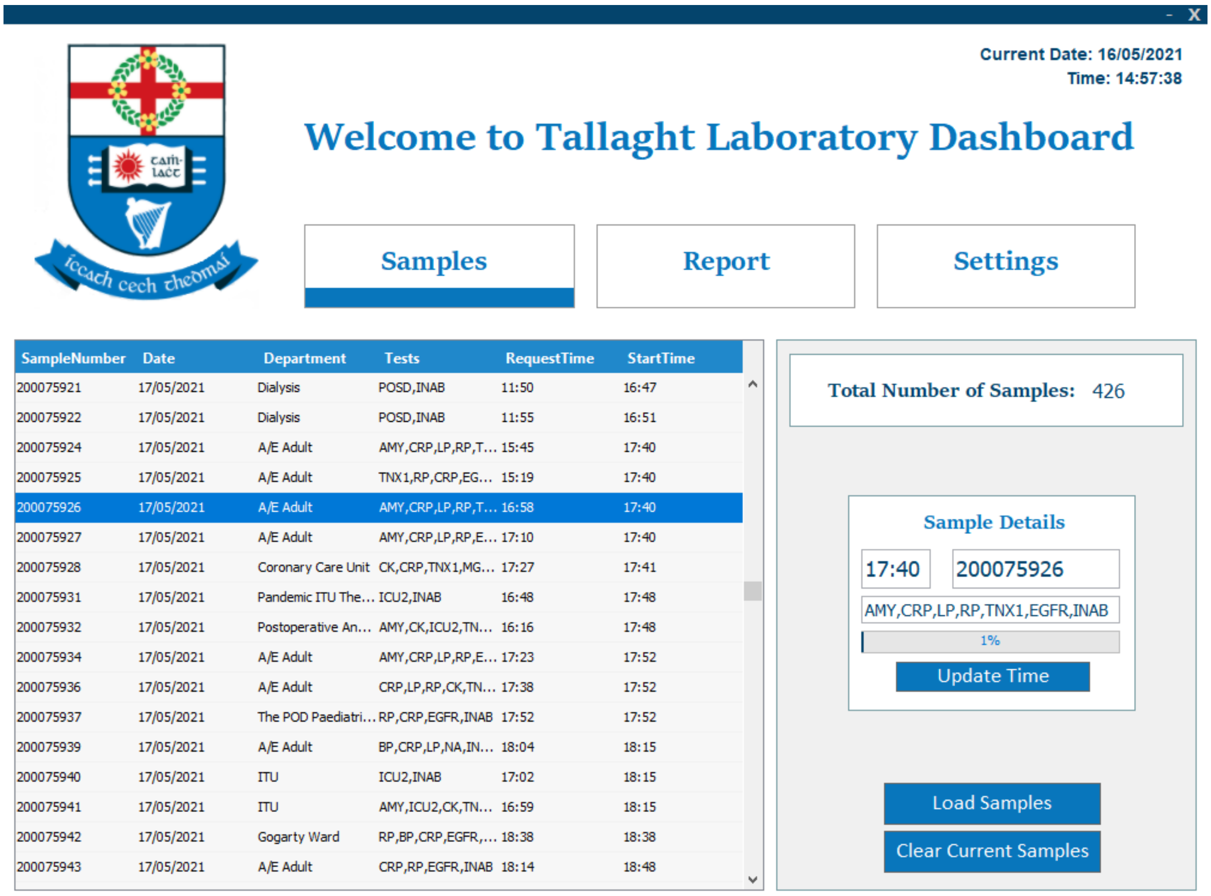


Figure 32 Dashboard

Sample screen:

The Sample page shown in figure 33 displays the current samples running in the Laboratory. Each sample has different fields which can be viewed by the user. The user has the option to select a sample and then change its currently starting time. They do this by selecting a sample and changing the time in the text box, then pressing the save button. The user can see how many samples are currently in the system. When the user clicks on a sample in the table the details are displayed on the left side of the screen. The progress bar displays the current percent of completion a sample is in its process.



Current Date: 16/05/2021
Time: 14:57:38

SampleNumber	Date	Department	Tests	RequestTime	StartTime
200075921	17/05/2021	Dialysis	POSD,INAB	11:50	16:47
200075922	17/05/2021	Dialysis	POSD,INAB	11:55	16:51
200075924	17/05/2021	A/E Adult	AMY,CRP,LP,RP,T...	15:45	17:40
200075925	17/05/2021	A/E Adult	TNX1,RP,CRP,EG...	15:19	17:40
200075926	17/05/2021	A/E Adult	AMY,CRP,LP,RP,T...	16:58	17:40
200075927	17/05/2021	A/E Adult	AMY,CRP,LP,RP,E...	17:10	17:40
200075928	17/05/2021	Coronary Care Unit	CK,CRP,TNX1,MG...	17:27	17:41
200075931	17/05/2021	Pandemic ITU The...	ICU2,INAB	16:48	17:48
200075932	17/05/2021	Postoperative An...	AMY,CK,ICU2,TN...	16:16	17:48
200075934	17/05/2021	A/E Adult	AMY,CRP,LP,RP,E...	17:23	17:52
200075936	17/05/2021	A/E Adult	CRP,LP,RP,CK,TN...	17:38	17:52
200075937	17/05/2021	The POD Paediatri...	RP,CRP,EGFR,INAB	17:52	17:52
200075939	17/05/2021	A/E Adult	BP,CRP,LP,NA,IN...	18:04	18:15
200075940	17/05/2021	ITU	ICU2,INAB	17:02	18:15
200075941	17/05/2021	ITU	AMY,ICU2,CK,TN...	16:59	18:15
200075942	17/05/2021	Gogarty Ward	RP,BP,CRP,EGFR,...	18:38	18:38
200075943	17/05/2021	A/E Adult	CRP,RP,EGFR,INAB	18:14	18:48

Total Number of Samples: 426

Sample Details

17:40 | 200075926

AMY,CRP,LP,RP,TNX1,EGFR,INAB

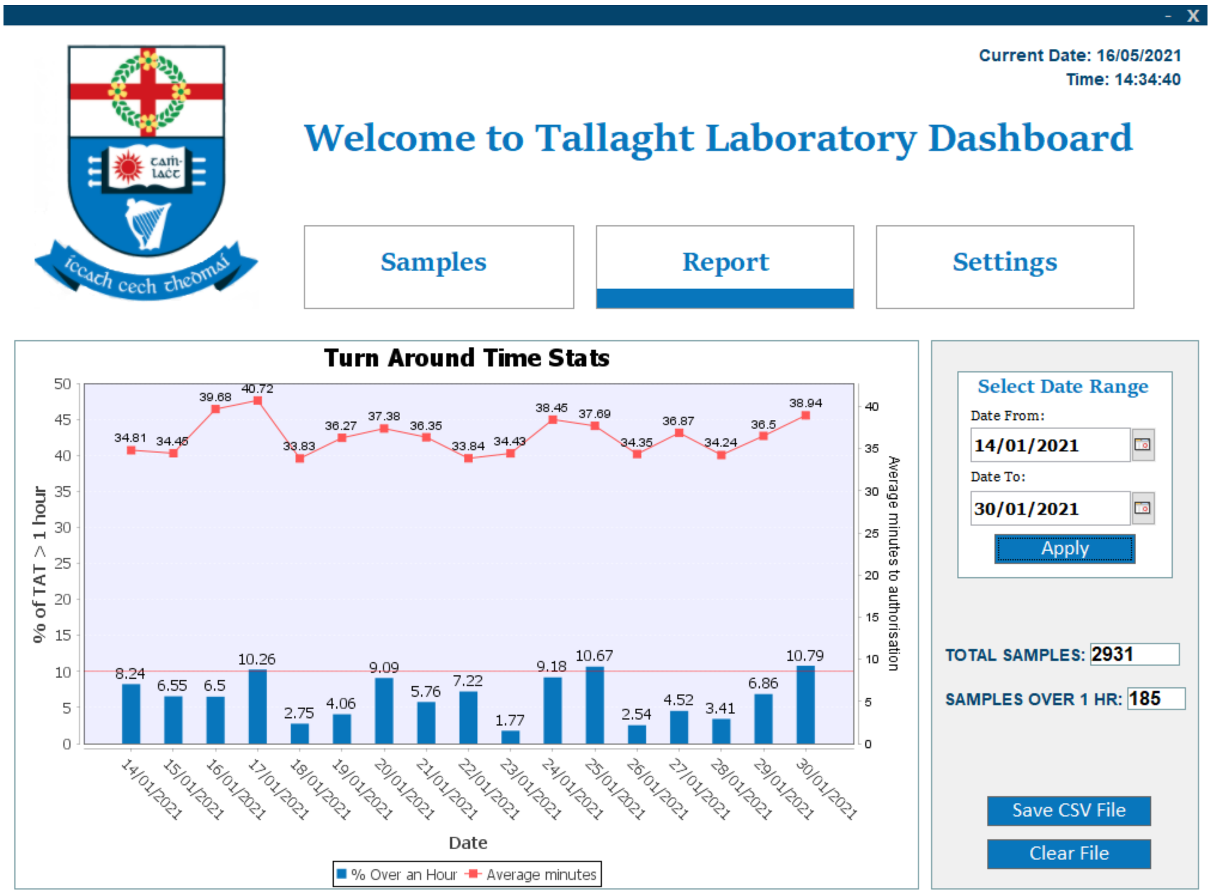
1%

Update Time

Figure 33 Samples screen

Report screen:

The Report page outlines key statistics from the past samples that have gone through the laboratories process. The page has buttons to filter by Daily, Weekly, or Monthly reports. After the user clicks a button, the fields will adjust with the desired statistics. The screen clearly outlines key figures and details which are important when analysing the TAT of processed samples. The report page is shown in figure 34.



Current Date: 16/05/2021
Time: 14:34:40

Welcome to Tallaght Laboratory Dashboard

Date	% Over an Hour	Average minutes
14/01/2021	8.24	34.81
15/01/2021	6.55	34.45
16/01/2021	6.5	39.68
17/01/2021	10.26	40.72
18/01/2021	2.75	33.83
19/01/2021	4.06	36.27
20/01/2021	9.09	37.38
21/01/2021	5.76	36.35
22/01/2021	7.22	33.84
23/01/2021	1.77	34.43
24/01/2021	9.18	38.45
25/01/2021	10.67	37.69
26/01/2021	2.54	34.35
27/01/2021	4.52	36.87
28/01/2021	3.41	34.24
29/01/2021	6.86	36.5
30/01/2021	10.79	38.94

Select Date Range
Date From: 14/01/2021
Date To: 30/01/2021
Apply

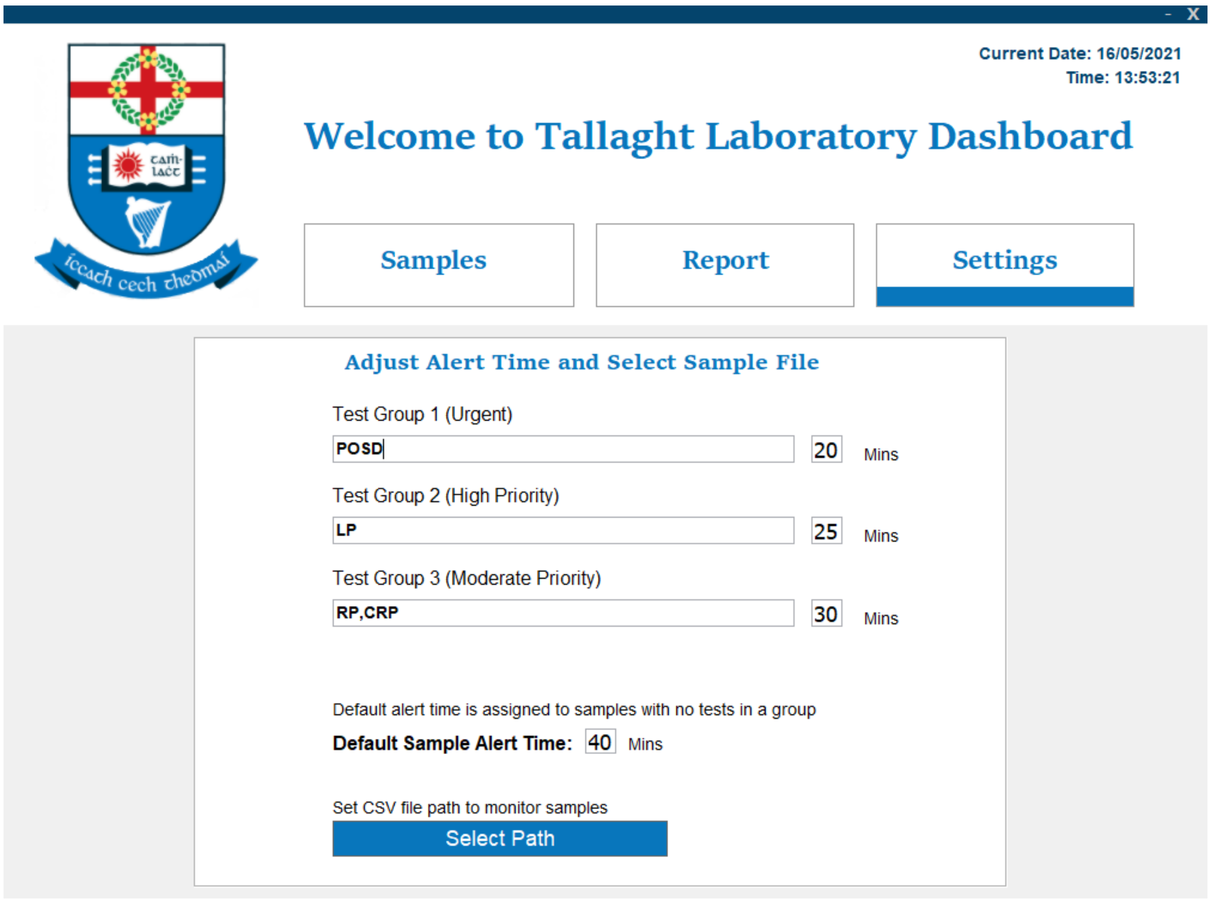
TOTAL SAMPLES: 2931
SAMPLES OVER 1 HR: 185

Save CSV File
Clear File

Figure 34 Report screen

Settings screen:

On the Settings page the user has different fields where they can edit shown in figure 35. Each field has a title highlighting the group of tests and the group of tests set delay time. If the user wishes to add or remove a test from a group, they can edit the field and click apply. The same process needs to be applied when a group of tests time is to be changed. The user can set the path to a csv file which is used for monitoring the current samples. The user can confirm changes by clicking apply before leaving the page.



Current Date: 16/05/2021
Time: 13:53:21

Welcome to Tallaght Laboratory Dashboard

Adjust Alert Time and Select Sample File

Test Group 1 (Urgent)
POSD | 20 Mins

Test Group 2 (High Priority)
LP | 25 Mins

Test Group 3 (Moderate Priority)
RP,CRP | 30 Mins

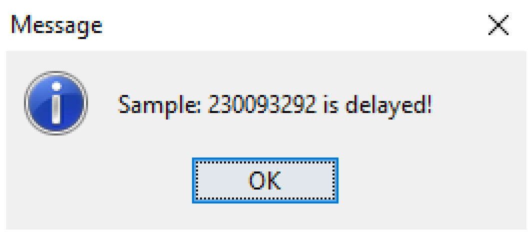
Default alert time is assigned to samples with no tests in a group
Default Sample Alert Time: 40 Mins

Set CSV file path to monitor samples

Figure 35 Settings screen

Alert message:

An alert will appear on the screen when a sample's set delay time has been exceeded. This will happen even if the user has minimised the dashboard and is using another program. This alert will let the user know what the sample number has been delayed. This will assist the user in finding the problem which has caused the delay. The user can exit off the alert screen by pressing the "OK" button shown in figure 36.

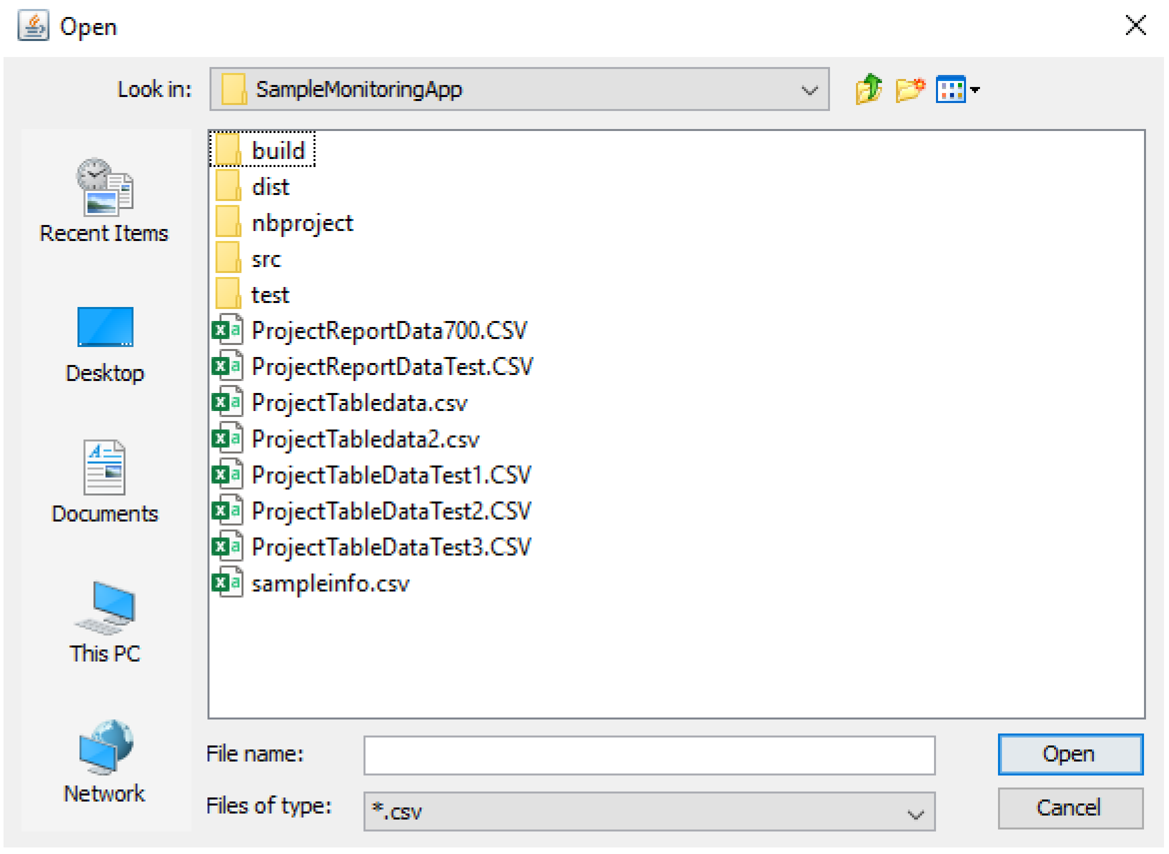


Sample: 230093292 is delayed!

Figure 36 Alert screen

File explorer screen:

A file explorer window appears when searching for a csv file in the settings page or on the report page. The explorer is set to filter for .csv files. When the directory with the desired csv file is found the user can either double click or select and open the file for it to be saved. The file explorer is shown in figure 37.



Look in: SampleMonitoringApp

build
dist
nbproject
src
test
ProjectReportData700.CSV
ProjectReportDataTest.CSV
ProjectTabledata.csv
ProjectTabledata2.csv
ProjectTableDataTest1.CSV
ProjectTableDataTest2.CSV
ProjectTableDataTest3.CSV
sampleinfo.csv

File name:

Files of type: *.csv

Figure 37 File explorer

Executable file:

The project can be run from an executable file installed on a machine. This file is located in an install folder. See figure 38 for the icon of the file.

Figure 38 Executable file icon

2.5.Testing

During the development process, all functions and code were dynamically and statically tested to detect any errors as early as possible. Manual testing was performed at all stages of development to quickly identify and solve errors.

Unit testing was conducted on all main classes in the project. Unit testing was preformed using JUnit testing. Unit testing ensures that individual components were working as intended. These tests were isolated from other components to be certain a function provides the desired outcome before integrating with other components. Functions for processing data used test files, after processing was completed, the results were compared with the expected output.

After all individual components of a single requirement were independently working as intended, tests were performed on all components of a requirement. The integration tests consisted of multiple functions working together. Most integration tests started with the inserting and processing of data into the database. Another function would select data from the database which would be returned to a different function used for calculating an expected value. The expected value and the actual value produced were compared. If the produced value consistently matched the expected value, it was successful.

System tests were completed by merging all functions which were previously tested together to form a system. The system testing was undertaken to ensure all requirements laid out for the project were completed and working as expected. All sections of the system were analysed to see if any errors occurred.

Acceptance testing was conducted after all errors found within the system tests were completed. The system as a whole was able to perform all 10 requirements that were set out in this report.

End User testing will be completed by Tallaght Hospital's IT staff. The staff will check if there are any possible harmful outcomes the program could have on the laboratories system. The overall quality of the program will be analysed and would need to be at a set standard to be accepted. Due to covid this could not be conducted during the development time of the project and will have to be done at a later date.

2.6. Evaluation

Each section of the project was evaluated individually to determine its level of completeness, effectiveness, and efficiency. The system was evaluated as a whole, to inspect when different sections of the project worked together.

User account:

User accounts were investigated to ensure details were correctly by registering a new account and logging in with the registered details. The details were adjusted within the database to ensure the login was working correctly.

Minute timer:

The minute timer was evaluated by starting the program and having the function print the system time in the console every minute. This was left running for two hours and the last printed message was checked with the current time. No problems were presented while inspecting the function.

Change sample start time:

A scenario for assessing the feature to update a sample's start time was performed. A sample was selected, and the time was changed, then the database record was checked to see if the change was applied. This process was successful for changing time, but a problem arose when a character which was not a number was inserted. The text field was updated to have parameters so that only a time value could be entered.

Alerting on sample delay:

The alerting and delay feature was found to been very successfully in notifying the user when a sample was delayed in the system. The dashboard was logged into and navigated to the samples page. The table was populated with test sample data. The start time of some samples in the table were changed to be 38 minutes before the system clock. The default alert time was set to 40 minutes.

After two minutes the samples which had been adjusted were alerted to the user as being delayed. An error was noticed when evaluating the alerting around midnight. Samples would be incorrectly calculated if a sample's time was before midnight and the system's time was after midnight. To fix this the date of the samples had to be used in conjunction with the starting time. This was compared with the system date to allow the program to know if the system's date has passed into the next day.

Table from database:

The updating of table data was evaluated to see if the csv file data was correctly being inserted into the database and displayed on the table. Test sample data was put into a csv file which was then selected as a path and inserted into the database. The sample data in the csv file was updated and resulted in an error with the database. The sample number field is the primary key so it would not accept any common samples found in the csv file and database. This was fixed by adding an update feature if the database contained the sample number that was being inserted from the csv file.

During this evaluation, an error was spotted when the table was updated which caused all the adjustments to sample start times to be removed. The table would also persist when it was not found in the csv data. To fix this another method was created for updating the database. This method would ensure all sample data which was not in the csv file was removed and that all sample start time adjustments were maintained.

Database:

Errors with the database existed during the inspection of updating the database records. The database would lock which resulted in stopping all other attempts to access or alter the data. This was a consequence of several database connections not being closed after connecting with the database. Close statements needed to be added at the end of all functions which interacted with the database.

Graph:

While examining the graphing function with test data, an error was presented if the mouse were dragged on the graph area. Zooming features had to be disabled just before the graph was displayed in the panel to stop this occurring.

The data displayed in the graph was between a start and end date. The screen displayed the values of each date in a random order. A sorting method was implemented to assemble the data in sequence by date.

All aspects of the project were evaluated, and from this evaluation it can be concluded that all requirements defined for the project were successfully implemented within the system.

3.0 Conclusions

Increase in laboratory workload and staff shortages makes this program a useful and desirable feature for Hospitals. Automation of manual services can lead to opportunities where programs like this one can help improve a work environment. The advantages of this program can lead to increased productivity in other sectors of Laboratory work as less focus would be required for manually monitoring sample times.

The project successfully provides a clear and efficient way to automate the monitoring of urgent samples and alert laboratory staff of delays in reporting tests on these samples. It also provides a means to summarise and graph TATs for laboratory management and accreditation purposes. Expected TAT can be configured by laboratory staff for all tests analysed. The program runs in the background and does not impact on the overall performance of the system it is running on. The functionality provided is not available on any LIS currently used in Ireland or on the new LIS which will replace the current systems. The version of the programme which was designed for the current “CliniSys” LIS can easily be adapted for the new “MedLIS” system planned to be introduced into all laboratories in Ireland.

The concept behind this program can be applied to the monitoring and alerting of any system where timed intervals are critical. In a hospital environment, this could be adapted to Diagnostic Imaging reporting or the monitoring of surgical waiting lists. In the past, Tallaght hospital had a serious issue with unreported X-ray results and a monitoring system like this would have identified the problem before it became a national issue.[13]

The program can only retrieve information that the Laboratory information system can provide. The CliniSys data structure is not consistent, and the same data is formatted differently in specific fields (e.g. HH:MM and HHMM for hours and minutes). Changes to the LIS queries producing the data for the system may require adjustments to the coding for the monitoring system.

The program is limited by the Laboratory information, this limits the information and amount the project is capable of doing. As a result, this means that expanding would be a challenge. If more information were able to be taken from a different system in the hospital the project could be expanded to assist in other parts of the hospital.

4.0 Further Development or Research

The integration within major hospitals around the country could be a possible path for this project to grow into. All hospitals lack what this system can offer in the automation of TAT monitoring. If Tallaght Hospital continues using the system, it could open paths for talks with other Hospitals. This would take time as Hospitals are very slow and hesitant to change but if there is a good standing in Tallaght maybe other hospitals would be open for this program.

With more time to put into the project a system could be implemented which would allow different hospitals to log in and use the program. Depending on the hospital, some additional features could be present while some could be disabled. This would also make the possibility for integration within any hospital system viable.

Gaining access to other systems within the hospitals would allow the project to be able to expand to allow doctors to view the current progress of blood samples. Doctors have no way of knowing what the current status of a blood sample is without physically going to the laboratory or phoning the laboratory staff. Doctors could be given access to the current samples and given a means of flagging a specific sample to laboratory staff.

If access to the system in control of the blood sample test results was allowed the project would be able to monitor and report on the analysis of the test results as well. This combined with Doctors having access would allow the system to provide a full overview of blood samples not just the monitoring of their times. Having access to the statistics of the blood test results would mean that the system could alert doctors or nurses if any sample has any abnormal or critical test results. These results need to be attended to as soon as possible. Monitoring of these results could speed up the response time of results being given to patients as well as the speed at which critical results are found.

Altering the project to suit these additional features would require more planning and time. Increasing the scale of the project would require a change to the database management system to be accessed on a server. Adapting the project to be a distributed system would give the project the highest potential for these future improvements.

5.0 References

1. DATE (JAVA PLATFORM SE 8), In-text: (Date (Java Platform SE 8), 2020), Your Bibliography: Docs.oracle.com. 2020. Date (Java Platform SE 8). [online] Available at: <https://docs.oracle.com/javase/8/docs/api/java/util/Date.html> [Accessed 05 December 2020].

2. DATE AND TIME FUNCTIONS, In-text: (Date And Time Functions, 2021), Your Bibliography: Sqlite.org. 2021. Date And Time Functions. [online] Available at: <https://sqlite.org/lang_datefunc.html> [Accessed 4 January 2021].

3. DATATYPES IN SQLITE VERSION 3, In-text: (Datatypes In SQLite Version 3, 2021), Your Bibliography: Sqlite.org. 2021. Datatypes In SQLite Version 3. [online] Available at: <https://www.sqlite.org/datatype3.html> [Accessed 4 January 2021].

4. OPENCSV, In-text: (opencsv –, 2021), Your Bibliography: Opencsv.sourceforge.net. 2021. opencsv –. [online] Available at: <http://opencsv.sourceforge.net/> [Accessed 10 January 2021].

5. STEFAN BECHTOLD, C. S., JUnit 5 User Guide, In-text: (Stefan Bechtold, 2021), Your Bibliography: Stefan Bechtold, C., 2021. JUnit 5 User Guide. [online] Junit.org. Available at: <https://junit.org/junit5/docs/current/user-guide/> [Accessed 5 March 2021].

6. RS2XML.JAR FREE DOWNLOAD - HACK SMILE, In-text: (rs2xml.jar Free Download - Hack Smile, 2021), Your Bibliography: Hack Smile. 2021. rs2xml.jar Free Download - Hack Smile. [online] Available at: <https://hacksmile.com/rs2xml-jar-free-download/> [Accessed 5 March 2021].

7. JFREECHART 1.5.0 API, In-text: (JFreeChart 1.5.0 API, 2021), Your Bibliography: Jfree.org. 2021. JFreeChart 1.5.0 API. [online] Available at: <https://www.jfree.org/jfreechart/api/javadoc/index.html> [Accessed 27 April 2021].

8. JFREECHART, In-text: (JFreeChart, 2021), Your Bibliography: Jfree.org. 2021. JFreeChart. [online] Available at: <https://www.jfree.org/jfreechart/> [Accessed 27 April 2021].

9. JCOMMON, In-text: (JCommon, 2021), Your Bibliography: Jfree.org. 2021. JCommon. [online] Available at: <https://www.jfree.org/jcommon/> [Accessed 27 April 2021].

10. JAVA PLATFORM SE 7. In-text: (Java Platform SE 7, 2021), Your Bibliography: Docs.oracle.com. 2021. Java Platform SE 7. [online] Available at: <https://docs.oracle.com/javase/7/docs/api/> [Accessed 1 May 2021].

11. JAVAX.SWING (JAVA PLATFORM SE 7), In-text: (javax.swing (Java Platform SE 7), 2021), Your Bibliography: Docs.oracle.com. 2021. javax.swing (Java Platform SE 7). [online] Available at: <https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html> [Accessed 5 May 2021].

12. JAVA SWING TUTORIAL – JAVATPOINT, In-text: (Java Swing Tutorial - javatpoint, 2021), Your Bibliography: www.javatpoint.com. 2021. Java Swing Tutorial - javatpoint. [online] Available at: <https://www.javatpoint.com/java-swing> [Accessed 5 May 2021].

13. X-RAY ERROR REPRESENTS FUNDAMENTAL FAILURE AT STATE'S FLAGSHIP HOSPITAL, In-text: (X-ray error represents fundamental failure at State's flagship hospital, 2021), Your Bibliography: The Irish Times. 2021. X-ray error represents fundamental failure at State's flagship hospital. [online] Available at: <https://www.irishtimes.com/news/x-ray-error-represents-fundamental-failure-at-state-s-flagship-hospital-1.635415> [Accessed 6 May 2021].

14. SQLITE HOME PAGE, In-text: (SQLite Home Page, 2021), Your Bibliography: Sqlite.org. 2021. SQLite Home Page. [online] Available at: <https://www.sqlite.org/index.html> [Accessed 10 May 2021].

6.0 Appendices

6.1. Project Plan

National College of Ireland

Project Proposal

Dynamic Sample Processing Dashboard for Hospital Laboratories

19/10/2020

BSc(Hons) in Computing

Software development

2017/2021

Mark Gaffney

x17443036

x17443036@student.ncirl.ie

Contents

6.1.1.0 Objectives

The aim of this project is to design a program to automate the monitoring of the Turn Around Time (TAT) for medical laboratory tests and to alert laboratory staff if delays are identified. This will ensure medical workers get results within expected timeframes and can act on the results. Timely reporting of laboratory results on patients in Emergency Departments speeds up decisions to admit, discharge or continue monitoring these patients.

To comply with accreditation, medical laboratories must monitor TAT's and demonstrate they achieve the performance agreed with service users. The proposed project will provide an automated reporting function for the statistics on the TAT's which can be used for accreditation purposes.

The program will consist of a Graphic User Interface (GUI) where the navigation between selections will be done. The GUI's layout will ensure users can clearly distinguish different areas of the program while displaying all relevant information.

The program should be integrated on the computers used in the laboratory systems within Tallaght hospital where it will be able to work as a background process. The program will need to be reviewed by IT staff at the hospital and be approved before integration. To gain approval the program will need to be high quality and reliable.

6.1.2.0 Background

Laboratory results are required in 70% of medical diagnoses. Laboratory workloads have been growing exponentially since the second half of the last century. This growth has been managed by improvements in automated instrumentation and, to a lesser extent, by increased staff numbers. The volume of data produced has necessitated the use of computerised systems, which replaced manual logs and reports in Ireland from the early 1980's. Laboratory Information Systems (LIS) were developed initially as local or regional projects. Some systems have been commercialised but, due to the limited number of medical laboratories in the developed world, they have not been enhanced sufficiently to provide all the functionality required to comply with the expectations of modern hospitals and accreditation bodies.

Medical laboratory staff perform many routine tasks to ensure reports are provided with the highest quality and in a timely manner, but they may be distracted by instrumentation problems, phone calls discussing results or surges in sample deliveries. Tallaght hospital Clinical Chemistry laboratory receives approximately 5,000 samples requiring around 20,000 tests each day. It is very easy for samples, or some tests, to be delayed during the multiple steps in processing all requests on each sample. The LIS can be used to monitor the status of requests and the staff can determine if there are any delays in producing results. Unfortunately, this requires the user to perform a search of the database and then examine the report generated to see if there are delays in reporting. This would not be serious problem if a member of staff could be dedicated to this task. If sufficient staff were available, one could spend the entire shift monitoring TAT and follow up of delayed reports. There would be variations in the frequency at which individuals would do the LIS searches and they could be delayed while investigating tests that exceed the designated TAT. An automated system, which would continuously monitor the time a test has been in the laboratory and alert the staff in the section where the test is performed, would ensure consistency and remove the need for a valuable medical scientist to run regular database searches.

The LIS's currently used in Ireland are legacy systems that were originally developed in the late 1980's and early 1990's. The main systems used by the majority of laboratories are "CliniSys" and "Apex". There is a plan to provide a single LIS for all laboratories in the Republic of Ireland. The system is being developed by Cerner and uses "Oracle" as its database. None of these systems, including the Cerner solution, provide an automated monitoring and alerting function for TAT's.

There is currently no dedicated laboratory TAT monitoring system in any of Irelands main hospitals. In Tallaght hospital the laboratory staff are alerted using an excel spread sheet running macro's that query reports generated by the LIS. This was developed in house and is not available in any other laboratory. This excel sheet alerts the staff when a blood sample exceeds the TAT by highlighting the sample in the sheet. Having the alerting done this way is very limited and causes problems because if any other excel sheet needs to be accessed it will stop this alerting.

6.1.3.0 Technical Approach

The system will be a desktop application that can be launched by the user. The application will then read through the files generated by LIS. The user may choose to navigate to a page to display current blood samples in the LIS as well as the details of the samples. After the user has launched the program it will continuously run checking each file produced by the LIS.

If the user wishes to view reports of statistics based on previous days, weeks, or months they may navigate from the samples page to the report section. Here is where they will be able to select an option to choose the reports of which they wish to view. These statistics will be calculated from data collected while the program is running. It will be able to show the number of times a certain test has failed over a time period.

When the application has been minimized it should alert with a popup box on the screen if a sample excessed its defined TAT. The popup box should outline information of the sample with the time placed within the system. The user will then be able to choose to delay the alert for a test and/or exit out of the popup box and the program will continue to check for other samples.

Users of the system will not require special permission or login with password protection. This is to minimize the chances of complication and will improve user experience. Security is already managed by IT and there is no sensitive information available on the program.

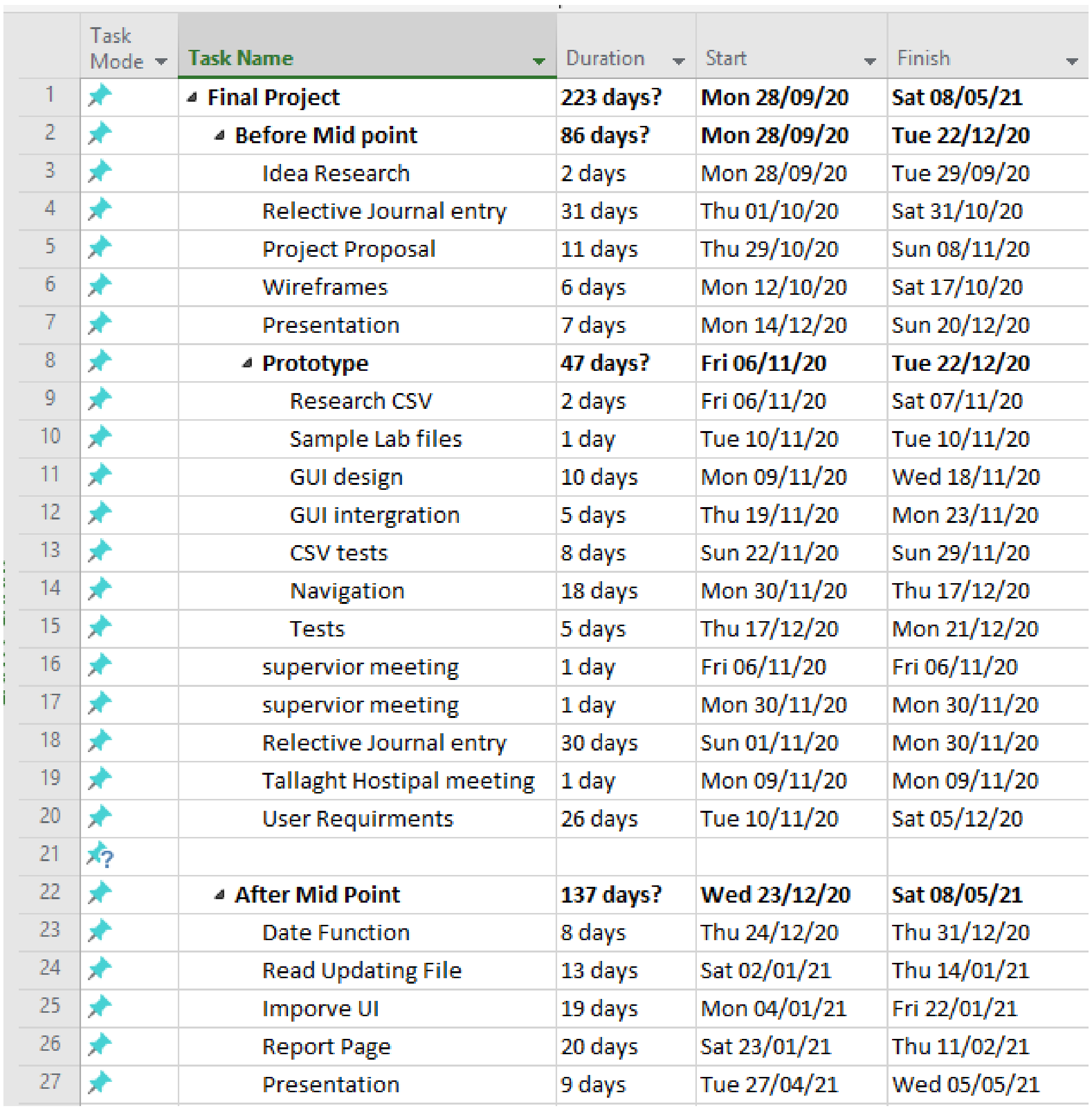
The system will need to be able to detect new files produced by the LIS. Files are updated every few minutes with updated information about samples and their details. The program will have to be able to read through the files and distinguish weather a sample is new or old and how long is has been in the system.

6.1.4.0 Special Resources Required

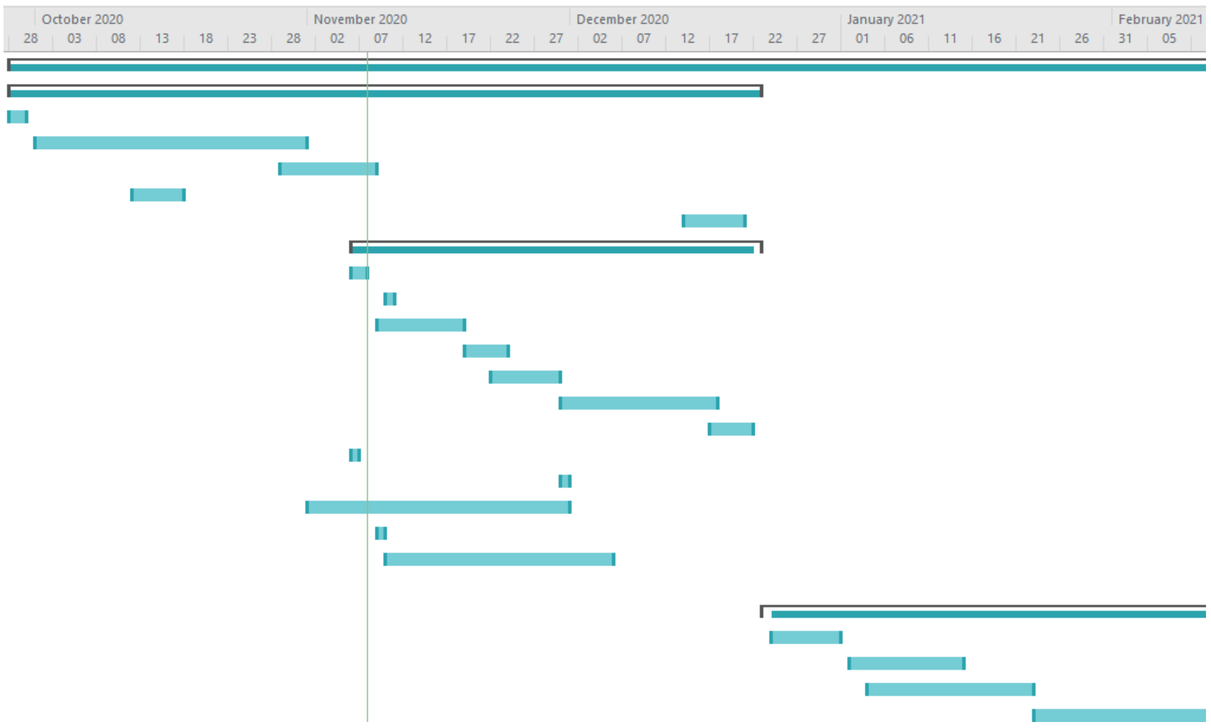
When working on my project I will need sample files from the LIS in Tallaght hospital. These files will not hold any patient identifying information to comply with the GDPR. For my application I will need sample files to run tests to confirm that my program can work with real world sample data.

No other new or specialized hardware will be needed for integration, this will ensure that the application can be run on a wide range of systems without requiring upgrades or additional installations. All resources are available at present which avoids additional fees.

6.1.5.0 Project Plan



	Task Mode	Task Name	Duration	Start	Finish
1	📌	**Final Project**	**223 days?**	**Mon 28/09/20**	**Sat 08/05/21**
2	📌	**Before Mid point**	**86 days?**	**Mon 28/09/20**	**Tue 22/12/20**
3	📌	Idea Research	2 days	Mon 28/09/20	Tue 29/09/20
4	📌	Relective Journal entry	31 days	Thu 01/10/20	Sat 31/10/20
5	📌	Project Proposal	11 days	Thu 29/10/20	Sun 08/11/20
6	📌	Wireframes	6 days	Mon 12/10/20	Sat 17/10/20
7	📌	Presentation	7 days	Mon 14/12/20	Sun 20/12/20
8	📌	**Prototype**	**47 days?**	**Fri 06/11/20**	**Tue 22/12/20**
9	📌	Research CSV	2 days	Fri 06/11/20	Sat 07/11/20
10	📌	Sample Lab files	1 day	Tue 10/11/20	Tue 10/11/20
11	📌	GUI design	10 days	Mon 09/11/20	Wed 18/11/20
12	📌	GUI intergration	5 days	Thu 19/11/20	Mon 23/11/20
13	📌	CSV tests	8 days	Sun 22/11/20	Sun 29/11/20
14	📌	Navigation	18 days	Mon 30/11/20	Thu 17/12/20
15	📌	Tests	5 days	Thu 17/12/20	Mon 21/12/20
16	📌	supervior meeting	1 day	Fri 06/11/20	Fri 06/11/20
17	📌	supervior meeting	1 day	Mon 30/11/20	Mon 30/11/20
18	📌	Relective Journal entry	30 days	Sun 01/11/20	Mon 30/11/20
19	📌	Tallaght Hostipal meeting	1 day	Mon 09/11/20	Mon 09/11/20
20	📌	User Requirments	26 days	Tue 10/11/20	Sat 05/12/20
21	📌?				
22	📌	**After Mid Point**	**137 days?**	**Wed 23/12/20**	**Sat 08/05/21**
23	📌	Date Function	8 days	Thu 24/12/20	Thu 31/12/20
24	📌	Read Updating File	13 days	Sat 02/01/21	Thu 14/01/21
25	📌	Imporve UI	19 days	Mon 04/01/21	Fri 22/01/21
26	📌	Report Page	20 days	Sat 23/01/21	Thu 11/02/21
27	📌	Presentation	9 days	Tue 27/04/21	Wed 05/05/21



Dates are estimates of when tasks can start and end.

6.1.6.0 Technical Details

The development of the program is currently planned to be developed in Java using NetBeans 8.2 for an Integrated Development Environment (IDE). NetBeans will provide useful tools for developing the GUI, it has a lot of build-in features that make it possible to design a very impressive interface. To make sure the program will be of high quality a lot of time will have to be spent to decide on page layouts and placement of Text Areas, Buttons and Drop-down menus.

Java is the current language I plan on developing the program in as it works well for making desktop programs. From college I have a lot of experience with Java which makes it easier to put more focus learning new aspects which will be needed for the program.

The program will need to read external files and process them into ArrayLists where the data can be processed. Many different java libraries will need to be imported into the main class to facilitate certain functions. “java.io.BufferedReader” and “java.io.FileReader” will be needed for reading CSV files as well as “java.util.Date” for reading times and dates from the data. Many more will be required later in development.

6.1.7.0 Evaluation

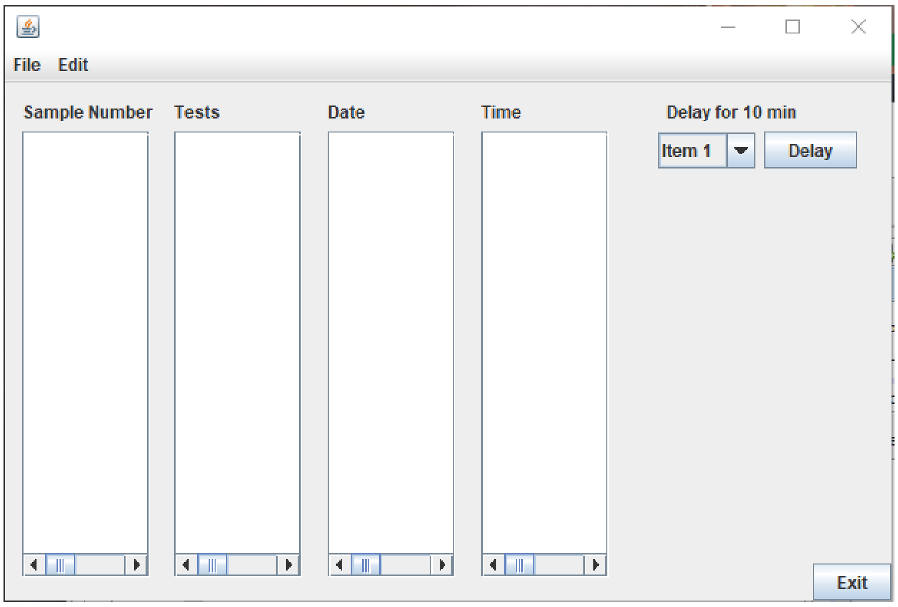
The program will be tested and run with the current excel macro system used in Tallaght. This is to confirm the program’s accuracy and reliability. The program will be installed on a computer in the laboratory where it will read the LIS generated CSV file. The program will then process this file and generate alerts and reports based on the sample information. Finally, the program will be validated by the users, by comparing the output with the current excel macro system.

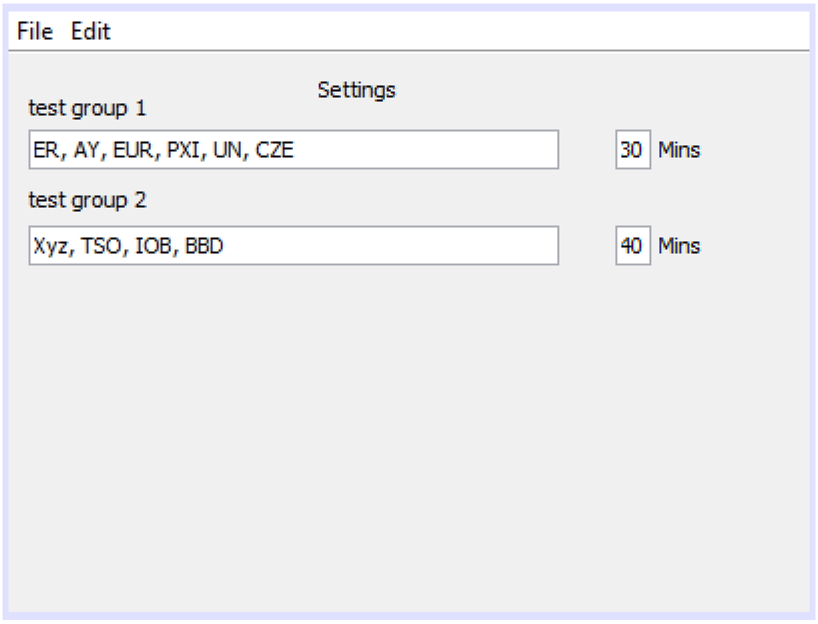
If the system is to be used as part of the laboratory processes, it will need to be validated to ensure it complies with accreditation standards ISO 15189.

October Journal Mark Gaffney

At the start of October I was planning out my project pitch video, I knew my idea quite well as I had spent some time over the pervious months thinking about it. The pitch video helped me to see that I knew what my idea was and how I wanted to do it but to explain it was a bit harder then expected. I will keep notes from now on so I can keep a clear path on what I want to be doing and what I have currently done.

The time spend not thinking of the pitch video I was focusing on GUI's for the hospital app. I think that the GUI will be really important because the app will be in a real business environment there is certain standards that need to be met. I designed some rough wireframes on how I thought the pages should be laid out. These where to test on how it would look and to see if they would be good design. The designs are just the start of what I had planned, they need a lot more work with visuals.





test group 1

ER, AY, EUR, PXI, UN, CZE 30 Mins

test group 2

Xyz, TSO, IOB, BBD 40 Mins

I only thought of designs for these two pages as the montioring page would be the main focus, then the settings page would be important as well. I was not fully commited with spending a lot of hours problem solving the app as I was not sure if it would be approved or not. I researched some topics for reading CSV files in java and how to store the values. I plan on using a bufferedreader to pass the CSV files into my program.

There was no contact with my supervisior as I wasn’t aware of who it would be until the end of the month. I was happy to find out that Cristina would be my supervisor. I have had her as a lecturer in the past and I know she is well skilled in java.

November Journal Mark Gaffney

The month of November was a bit slower compared to October as more focus was put towards other modules assessments.

In the middle of November I had a call with Eoin Begley in Tallaght hospital who is the Chief medical scientist in biochemistry in Tallaght. I talked with him about my project and if it could fit into the lab setup. He was fine with my program being added into the lab with approval by IT beforehand. I sent him my project proposal and course details so he can have more information about it.

In the last week of November, I emailed Paudy O’Gorman who is Chief Point of Care in the Mater to get some more information on the planned rollout for MedLIS. The MedLIS is the new system being brought into hospital laboratories. This will change how the systems work and will be integrating an oracle database. This new system will not support the reporting or alerting features I will be implementing in my project.

After some research Electron is an application that is used for making desktop application using JavaScript and web design features. This could be a good tool to use to develop my programs desktop app.

Current problems I have run into are finding a way to check when a file has been updated. This might be solved by including a database for the file and compare the difference but will need testing.

More research into how to run a program from a desktop application needs to be done as I can’t use Java with Electron and converting could be difficult. Electron also might need extra resources and not be very good for a lab environment.

A database structure needs to be planned out and connected soon for testing on storing the information in the CSV files.

December Journal Mark Gaffney

At the start of December preperation for the midpoint submission started. An outline of the objectives needed for the submission was made with what marks where assigned to what sections. The report was 40% of the submission so the main focus was to have my applcation design layout done then start the report as some section in the report needed screenshots and use cases for key features. After some thought on the page designs and layout the introduction of the report was started. All headings in sections 1 to 3 where given short paragraphs with ideas for what was needed in the section.

After each section was read through and an idea was made of what had to be put in the headings I started work on the Use cases as they are a huge part of the report in section 2. I could use the page layouts to think of use cases that are required for the application. I created the use cases from the first point of contact the user has with the application through to all the different pages and options. After the use cases were created I moved onto the other headings in the requirements section.

The GUI screenshots of the application pages was left til nearer the due date as if any changes where made the screenshots would have to be reuploaded. I focused on making a well designed dashboard which is clear to read and easy to navigate. Functionality of the applciation was not nessassary for most pages as that would mean the project would expected to be finsihed.

A meeting with Cristina on the Friday before the submission was arragened for questions about our projects and the submission. This meeting was not very productive as there was a lot of misunderstanding with the overall scope of the project. I couldn't take much away from the meeting as there wasn't enough time to implement changes that where suggested. This will have to be looked more into next year.

The demo of the submission was created and slides to example the idea behind the project. I think the presentation was a success I know I skipped over a key point but overall it seemed like an okay presnetation. The demo of the application got a bit messy but all the pages where talked about and the functionality which will be added in the furture was explained.

The plan for the start of next year will need to be clear with what has to be done for what dates as pressure from other modules is taking up a lot of time around the holidays. Focus will be placed on other modules until the 6th of january after that date reasearch into graphs with java can start.

January Journal Mark Gaffney

The time used in the start of january was focused on TABA assistments and the multimedia and mobile project. The first week was focused finishing up on the mobile app then time was focused on the Introduction to AI submission and the Strategic managemnet TABAs. After the 7th time was focuse on researching the next steps in the project.

College started back on the 25th, the time between then and the 7th of January was spend testing database types and changing the insertion of the CSV information. I found that SQlite could be a good option for a Database management system as it does not require a lot of dependancies and would be easier for implementing on the tallaght lab systems.

The database will need to be constistanly checked with the new information provided in the CSV file and if there is new data the database should update the whole table with the new information. The times for the samples will remain the same but if a sample is removed and isnt delayed it won't be monitored for delays anymore.

More testing on pulling from the database and editing the time with a button needs to be the next step. First making it easy to select a specific sample then the top will need to be edited on it or the delay timer will need to be customized for it.

I talked again with Eion Begley to show the design aspects and get inputs on layout so far. Will discuss more details when more features are added in. testing on the lab systems will be at a later stage.

February Journal Mark Gaffney

Feburuay was spent testing the SQLite database and editing the inputs from the CSV file. One module had an assesstment due on the 28th. Projects from other modules had to be researched during the month.

A project with a database was created for pulling information from the CSV file. The database has 10 columns, 6 of these are information which are usually provided by the CSV. Testing with a reported time of the samples completed was created and a time+date column which is a samples Date and Time joined into a single value.

The reported columns added was to test for the reporting feature of the system. More tests needs to be done where the time difference will be recorded for a column starting and finishing. This could be changed at a later date.

The information from the database is being printed on a JTable. This is very basic at the moment and does not allow the user to effect the database. The next step is to be able to delay a sample for the alerting system. This will need to be added onto a button for the user.

To sum up feburary, the csv file was successfully being passed into the database. The attributes where edited as needed. A new colum for time and date was created. Informaiton from the database is being displayed in a GUI with a JTable.

March Journal Mark Gaffney

The month of March was busy with reading week and other module assistments due towards the end. This ment not as much time could be dedicated to progressing in the project. The first week of the March was spend testing the sampling dashboard screen. The sampling screen processes the information but needs to be able to refresh the file constantly.

Time needed to be spend to researching methods needed to refresh the file feed to the database. Research was also done on how to implement the delay button with the samples on the dashboard. This feature needs to take the time of the sample and increase it on the page and in the database.

The report section of the dashboard needs to have its database dependancys setup and defined. The Java Swing library needed for displaying the report informaiton needs to be defined. The first 3 weeks of march were spent researching how these methods would be implemented into the project.

Errors with the Delay and report page are the main focus from the end of march. These are key parts of the project that when they get fixed other methods will be easier to implement.

The next major part of the project will be the settings page to allow delaying specific tests for any amount of time.

April Journal Mark Gaffney

The first two weeks of April were spent focuing on other module's projects. They were completed early into the month so there would be a lot of uninterupted time for the final project. For the first two weeks the research for graphing was conducted and the library JFreeChart was found to be the most sutible option for the reporting page.

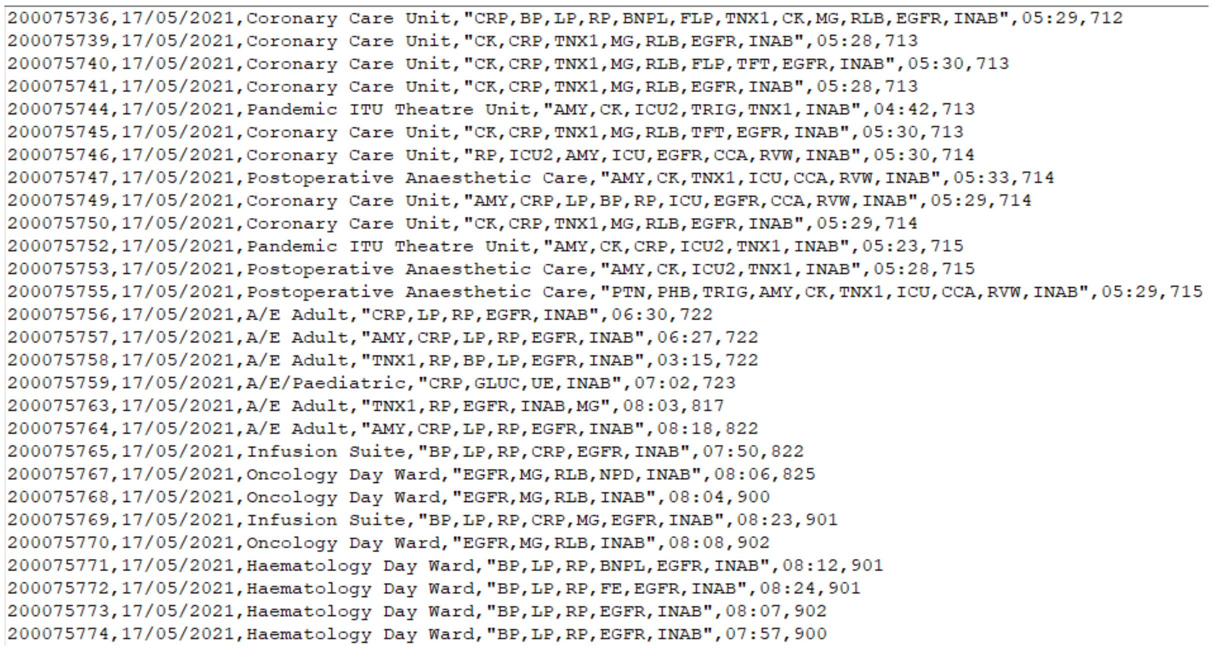
After the 14th more time was spend on the project. The design of the sample page was updated with additional text areas to display data information. The table was styled to be more user friendly and the option for a user to update a start time of a sample was added in succesfully. The database now can be updated with the users new input. A process bar was added to show how far along a sample that has been click on is.

The report page was changed to include a file finder which allows the user to insert historical data csv file which holds information on samples which has previously been processed. This file gets inserted into a separte database and is to be used for graphing. The design for the type of chart which would be used to display the data. A Bar and Line combined chart was the best option for displaying the important information. The chart has 2 Y values to make it clear and a horizonal line at 10% for the % of samples over an hour was inserted to show if any date exceeds the threshold.

The settings page has been changed to allow a path to be set by a user to a csv file which is used for updating the table on the sample page. A default alert time area has also been placed on the page to allow the user to change the default alert time.

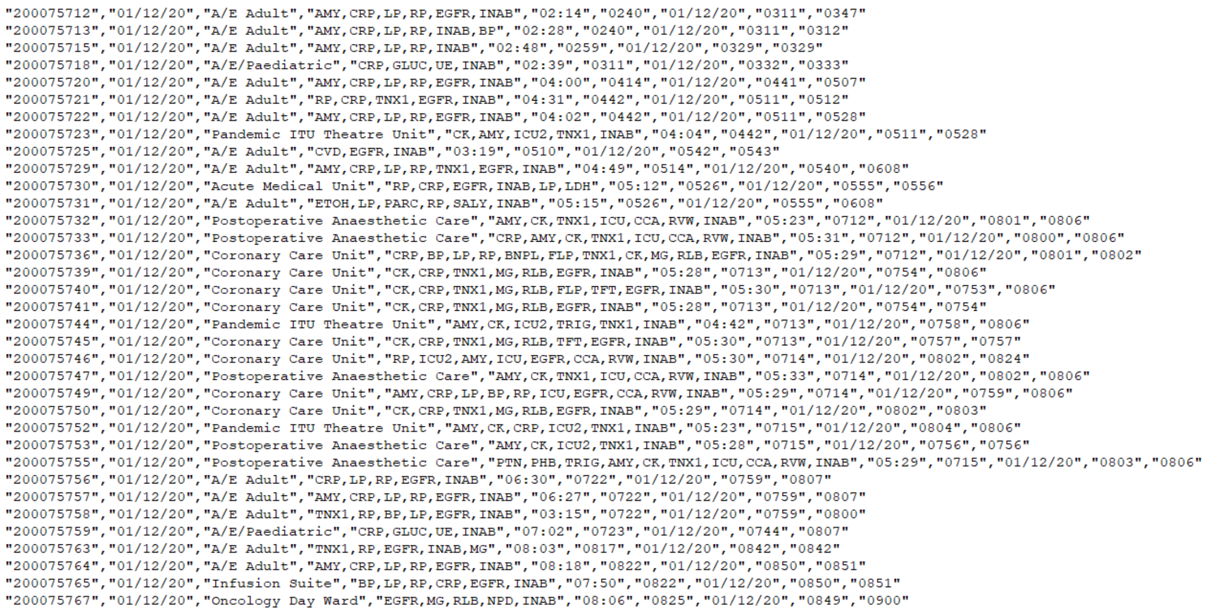
Plotting data from the database is the next main priority and then cleaning the code to be more presentable is another important part. A error needs to be fixed when loading information from the csv files into the databases were the time it takes is extreamly long.

6.3 Formatted files example



```
200075736,17/05/2021,Coronary Care Unit,"CRP,BP,LP,RP,BNPL,FLP,TNX1,CK,MG,RLB,EGFR,INAB",05:29,712
200075739,17/05/2021,Coronary Care Unit,"CK,CRP,TNX1,MG,RLB,EGFR,INAB",05:28,713
200075740,17/05/2021,Coronary Care Unit,"CK,CRP,TNX1,MG,RLB,FLP,TFT,EGFR,INAB",05:30,713
200075741,17/05/2021,Coronary Care Unit,"CK,CRP,TNX1,MG,RLB,EGFR,INAB",05:28,713
200075744,17/05/2021,Pandemic ITU Theatre Unit,"AMY,CK,ICU2,TRIG,TNX1,INAB",04:42,713
200075745,17/05/2021,Coronary Care Unit,"CK,CRP,TNX1,MG,RLB,TFT,EGFR,INAB",05:30,713
200075746,17/05/2021,Coronary Care Unit,"RP,ICU2,AMY,ICU,EGFR,CCA,RVW,INAB",05:30,714
200075747,17/05/2021,Postoperative Anaesthetic Care,"AMY,CK,TNX1,ICU,CCA,RVW,INAB",05:33,714
200075749,17/05/2021,Coronary Care Unit,"AMY,CRP,LP,BP,RP,ICU,EGFR,CCA,RVW,INAB",05:29,714
200075750,17/05/2021,Coronary Care Unit,"CK,CRP,TNX1,MG,RLB,EGFR,INAB",05:29,714
200075752,17/05/2021,Pandemic ITU Theatre Unit,"AMY,CK,CRP,ICU2,TNX1,INAB",05:23,715
200075753,17/05/2021,Postoperative Anaesthetic Care,"AMY,CK,ICU2,TNX1,INAB",05:28,715
200075755,17/05/2021,Postoperative Anaesthetic Care,"PTN,PHB,TRIG,AMY,CK,TNX1,ICU,CCA,RVW,INAB",05:29,715
200075756,17/05/2021,A/E Adult,"CRP,LP,RP,EGFR,INAB",06:30,722
200075757,17/05/2021,A/E Adult,"AMY,CRP,LP,RP,EGFR,INAB",06:27,722
200075758,17/05/2021,A/E Adult,"TNX1,RP,BP,LP,EGFR,INAB",03:15,722
200075759,17/05/2021,A/E/Paediatric,"CRP,GLUC,UE,INAB",07:02,723
200075763,17/05/2021,A/E Adult,"TNX1,RP,EGFR,INAB,MG",08:03,817
200075764,17/05/2021,A/E Adult,"AMY,CRP,LP,RP,EGFR,INAB",08:18,822
200075765,17/05/2021,Infusion Suite,"BP,LP,RP,CRP,EGFR,INAB",07:50,822
200075767,17/05/2021,Oncology Day Ward,"EGFR,MG,RLB,NPD,INAB",08:06,825
200075768,17/05/2021,Oncology Day Ward,"EGFR,MG,RLB,INAB",08:04,900
200075769,17/05/2021,Infusion Suite,"BP,LP,RP,CRP,MG,EGFR,INAB",08:23,901
200075770,17/05/2021,Oncology Day Ward,"EGFR,MG,RLB,INAB",08:08,902
200075771,17/05/2021,Haematology Day Ward,"BP,LP,RP,BNPL,EGFR,INAB",08:12,901
200075772,17/05/2021,Haematology Day Ward,"BP,LP,RP,FE,EGFR,INAB",08:24,901
200075773,17/05/2021,Haematology Day Ward,"BP,LP,RP,EGFR,INAB",08:07,902
200075774,17/05/2021,Haematology Day Ward,"BP,LP,RP,EGFR,INAB",07:57,900
```

Figure 39 Table csv data example



```
"200075712","01/12/20","A/E Adult","AMY,CRP,LP,RP,EGFR,INAB","02:14","0240","01/12/20","0311","0347"
"200075713","01/12/20","A/E Adult","AMY,CRP,LP,RP,INAB,BP","02:28","0240","01/12/20","0311","0312"
"200075715","01/12/20","A/E Adult","AMY,CRP,LP,RP,INAB","02:48","0259","01/12/20","0329","0329"
"200075718","01/12/20","A/E/Paediatric","CRP,GLUC,UE,INAB","02:39","0311","01/12/20","0332","0333"
"200075720","01/12/20","A/E Adult","AMY,CRP,LP,RP,EGFR,INAB","04:00","0414","01/12/20","0441","0507"
"200075721","01/12/20","A/E Adult","RP,CRP,TNX1,EGFR,INAB","04:31","0442","01/12/20","0511","0512"
"200075722","01/12/20","A/E Adult","AMY,CRP,LP,RP,EGFR,INAB","04:02","0442","01/12/20","0511","0528"
"200075723","01/12/20","Pandemic ITU Theatre Unit","CK,AMY,ICU2,TNX1,INAB","04:04","0442","01/12/20","0511","0528"
"200075725","01/12/20","A/E Adult","CVD,EGFR,INAB","03:19","0510","01/12/20","0542","0543"
"200075729","01/12/20","A/E Adult","AMY,CRP,LP,RP,TNX1,EGFR,INAB","04:49","0514","01/12/20","0540","0608"
"200075730","01/12/20","Acute Medical Unit","RP,CRP,EGFR,INAB,LP,LDH","05:12","0526","01/12/20","0555","0556"
"200075731","01/12/20","A/E Adult","ETOH,LP,PARC,RP,SALY,INAB","05:15","0526","01/12/20","0555","0608"
"200075732","01/12/20","Postoperative Anaesthetic Care","AMY,CK,TNX1,ICU,CCA,RVW,INAB","05:23","0712","01/12/20","0801","0806"
"200075733","01/12/20","Postoperative Anaesthetic Care","CRP,AMY,CK,TNX1,ICU,CCA,RVW,INAB","05:31","0712","01/12/20","0800","0806"
"200075736","01/12/20","Coronary Care Unit","CRP,BP,LP,RP,BNPL,FLP,TNX1,CK,MG,RLB,EGFR,INAB","05:29","0712","01/12/20","0801","0802"
"200075739","01/12/20","Coronary Care Unit","CK,CRP,TNX1,MG,RLB,EGFR,INAB","05:28","0713","01/12/20","0754","0806"
"200075740","01/12/20","Coronary Care Unit","CK,CRP,TNX1,MG,RLB,FLP,TFT,EGFR,INAB","05:30","0713","01/12/20","0753","0806"
"200075741","01/12/20","Coronary Care Unit","CK,CRP,TNX1,MG,RLB,EGFR,INAB","05:28","0713","01/12/20","0754","0754"
"200075744","01/12/20","Pandemic ITU Theatre Unit","AMY,CK,ICU2,TRIG,TNX1,INAB","04:42","0713","01/12/20","0758","0806"
"200075745","01/12/20","Coronary Care Unit","CK,CRP,TNX1,MG,RLB,TFT,EGFR,INAB","05:30","0713","01/12/20","0757","0757"
"200075746","01/12/20","Coronary Care Unit","RP,ICU2,AMY,ICU,EGFR,CCA,RVW,INAB","05:30","0714","01/12/20","0802","0824"
"200075747","01/12/20","Postoperative Anaesthetic Care","AMY,CK,TNX1,ICU,CCA,RVW,INAB","05:33","0714","01/12/20","0802","0806"
"200075749","01/12/20","Coronary Care Unit","AMY,CRP,LP,BP,RP,ICU,EGFR,CCA,RVW,INAB","05:29","0714","01/12/20","0759","0806"
"200075750","01/12/20","Coronary Care Unit","CK,CRP,TNX1,MG,RLB,EGFR,INAB","05:29","0714","01/12/20","0802","0803"
"200075752","01/12/20","Pandemic ITU Theatre Unit","AMY,CK,CRP,ICU2,TNX1,INAB","05:23","0715","01/12/20","0804","0806"
"200075753","01/12/20","Postoperative Anaesthetic Care","AMY,CK,ICU2,TNX1,INAB","05:28","0715","01/12/20","0756","0756"
"200075755","01/12/20","Postoperative Anaesthetic Care","PTN,PHB,TRIG,AMY,CK,TNX1,ICU,CCA,RVW,INAB","05:29","0715","01/12/20","0803","0806"
"200075756","01/12/20","A/E Adult","CRP,LP,RP,EGFR,INAB","06:30","0722","01/12/20","0759","0807"
"200075757","01/12/20","A/E Adult","AMY,CRP,LP,RP,EGFR,INAB","06:27","0722","01/12/20","0759","0807"
"200075758","01/12/20","A/E Adult","TNX1,RP,BP,LP,EGFR,INAB","03:15","0722","01/12/20","0759","0800"
"200075759","01/12/20","A/E/Paediatric","CRP,GLUC,UE,INAB","07:02","0723","01/12/20","0744","0807"
"200075763","01/12/20","A/E Adult","TNX1,RP,EGFR,INAB,MG","08:03","0817","01/12/20","0842","0842"
"200075764","01/12/20","A/E Adult","AMY,CRP,LP,RP,EGFR,INAB","08:18","0822","01/12/20","0850","0851"
"200075765","01/12/20","Infusion Suite","BP,LP,RP,CRP,EGFR,INAB","07:50","0822","01/12/20","0850","0851"
"200075767","01/12/20","Oncology Day Ward","EGFR,MG,RLB,NPD,INAB","08:06","0825","01/12/20","0849","0900"
```

Figure 40 Report csv data example