Technical Report

David Auchterlonie (Student number: 1585335)
Ivan Von Staden (Student number: 1838664)
Jesse van der Merwe (Student number: 1829172)
Matthew Cresswell (Student number: 1834243)

May 2020

1 Design Overview

The program uses a modular design structure - similar to that of the MVC model. As seen in the Program Structure (figure 1 below), the program is divided into three parts; the front end, the back end and the setup. The front end handles user input validation and all Graphical User Interface (GUI) aspects of the program. The back end is where the majority of the data processing occurs and is the core location for program functionality. The setup is only run once at the start of the program in order to ensure that all required tables, folders, table-relationships, etc. are in place.

1.1 Back End

Within the back end, the functionality of the program is split between four main Application Program Interfaces; the DataAPI, LoggingAPI and BillingAPI. These programs handle the main functionality of the application and all the processes involving the transfer of data to and from the database.

1.2 Front End

In the front end, the functionality of the program is split between various User Interface scripts. These scripts handle all front end functionality. An example includes the Main_UI script which provides functionality to the main menu screen of the application. All these scripts handle all the processes involved in the transfer of data and information from the application to the user.

1.3 Setup

Folders for program's generated data (such as the generated PDF invoices, as well as the various text documents used for logging) can be found in the root of the program. Any required folders and files are automatically generated when the program is run. The required tables are automatically generated, with the required relationships, foreign keys, etc.

2 Class/library breakdown

2.1 DataAPI

This library is responsible for all data processing between the front end and back end of the program. It includes functionality such as "AddBox" , "MoveBox" and other basic database management commands. Furthermore it also holds auxiliary data processing such as finding samples that need to be returned or destroyed within the next week, suggesting places for samples to be put when entering the bio-bank.

2.1.1 Basic Functionality

- Adding: this uses an sqlite3 INSERT statement to manually add fridges, boxes, samples, sample tests, collections, and users to the database.
- Moving: this uses an sqlite3 UPDATE statement to edit the database to reflect the desired changes.
- Deleting: this uses an sqlite3 DELETE statement to remove information from the database.

2.1.2 Automatic Sample Addition

Automatic sample addition is done by reading in a comma separated value (CSV) file. Each line in this file refers to one record, and thus one sample entry. Each value in each entry is checked to see if all of the information is correct and/or does not break any required data validation. If an addition fails, a message will pop up telling the user on which line it failed and why it failed. If all lines (samples) pass this validation check, they are then all automatically added to the database system.

2.2 SetupAPI

This library is responsible for initializing directories and dependencies the program needs to run properly. This includes things like creating folder directories for logs, input data and invoices. These folders are created in the root of the program. It is also responsible for creating any missing tables in the database.

2.3 LoggingAPI

The LoggingAPI is located in the back end only and is never interacted with from the front end. The LoggingAPI is responsible for logging all data changes made in the database. This API uses functions that pass in what change was made, when it was made and by who. This information is then saved into three different types of logs; the general log, individual log and billing log.

- The general log is a text file in which all changes to the database are appended.
- There is an individual text file that acts as a log for each sample that only records changes made that directly affect that specific sample.
- The billing log is a general log the tracks all billings made by appending the relavent information to this billing text file.

All these logging text files are saved in the logs folder found in the root of the the program directory.

2.4 BillingAPI

This API deals with everything related to billing. This has two main functions:

1. Automatic billing: Using the built in Linux time-based job scheduler, Cron, the program automatically bills any customers that have samples in the biobank by generating the required PDF invoice documents. This occurs on the first of every month, at 01:00. This can be manually tested by using the "Invoice for Storage" function within the Invoice Mode. This button is coloured in red as it is purely for testing purposes.

2. Manual billing of new samples: If any new samples are manually added to the biobank, employees can manually tell the program to generate PDF invoices for these samples. This can only occur once per new sample.

The information used in the billing and PDF generation - for example the details of a customer - are pulled automatically from the database.

2.5 User Interface

2.5.1 Login

This is the first screen presented to a user at startup. It takes two inputs (username and password) and checks it against what is stored in the Login Table of the database. If there is a matching username and password in the Login Table, the user is granted access. The program also fetches what access level this user has:

- Supervisor - this is the highest access level of the program and allows the user to perform any and all functions, including accessing the edit mode (see below).
  - There is a default supervisor login role pre-programmed into the system to allow for testing with username: **admin** and password: **admin**
- Employee - this is the second highest access level of the program and allows the user to perform most functions, NOT including accessing the edit mode (see below).
- Customer - this is the lowest access level of the program, and only allows the user to view the View Mode. The user is assumed to be a customer, and is also limited to only seeing the samples in the database from their collections.
  - There is a default customer login role pre-programmed into the program to allow for testing with username **C1** and password: **1234**

It should be noted that as an employee/supervisor logs into the program, if there are any samples that are due (or overdue) to be returned/destroyed within the upcoming week, a notification will appear.

2.5.2 Main Menu

This screen is used to navigate through to the different areas of the system. Certain areas and functions will be locked, depending on the user's access level.

2.5.3 Edit Mode

This section is only accessible to users with the highest access level. This is where the functionality to add, remove, and move various components of the database is accessed. This is done through calling commands in the DataAPI. There are menus for fridges, boxes, samples, collections, users, as well as the automatic sample addition functionality. The system will display what was added, moved, or removed if the operation is successful or what the error is if the operation fails.

2.5.4 View Mode

A dynamic screen is generated depending on the information within the database - i.e. buttons are created for each fridge and, when a fridge is selected, buttons are created for each box within that fridge. When a box is selected, all the samples in that box are displayed in table form with buttons for viewing corresponding sample tests.

2.5.5 Invoice Mode

1. Invoice for New Samples: The user can use this function to generate PDF invoices for any new samples that have been added to the biobank. Newly added samples will only be billed once with this function.

2. Invoice for Storage: An automatic billing is scheduled to run once a month, generating PDF invoices for all clients for all their samples stored within the bio-bank during that month. There is currently a button (labelled ”Invoice for Storage”) to manually perform this task, in order to test and display the functionality. However, in future, this button will be removed as the process is automatically run in the background.

The system will generate the PDF invoices and save them in the Invoices folder (in the root directory of the program). A small message display will show which clients were billed, and for how much.

2.5.6 Search Mode

This mode allows the user to enter parameters to search through tables within the database. For example, a user can display a table of all the fridges in the bio-bank, or search for a fridge with a particular ID. This functionality is available for fridges, boxes, samples, sample tests, and collections.

2.5.7 Stock Take

This rudimentary stock take function takes in user input for the type of stock to count (i.e. either fridges, boxes or samples) as well as the number of the stock expected. It then performs a basic check and tells the user whether there is a stock mismatch or not.

3 Unit Tests

In order to run the 41 unit tests currently being used to test the program, run the following code in the console:

python3 UnitTest.py

When run, this should display the result of each test in the terminal.

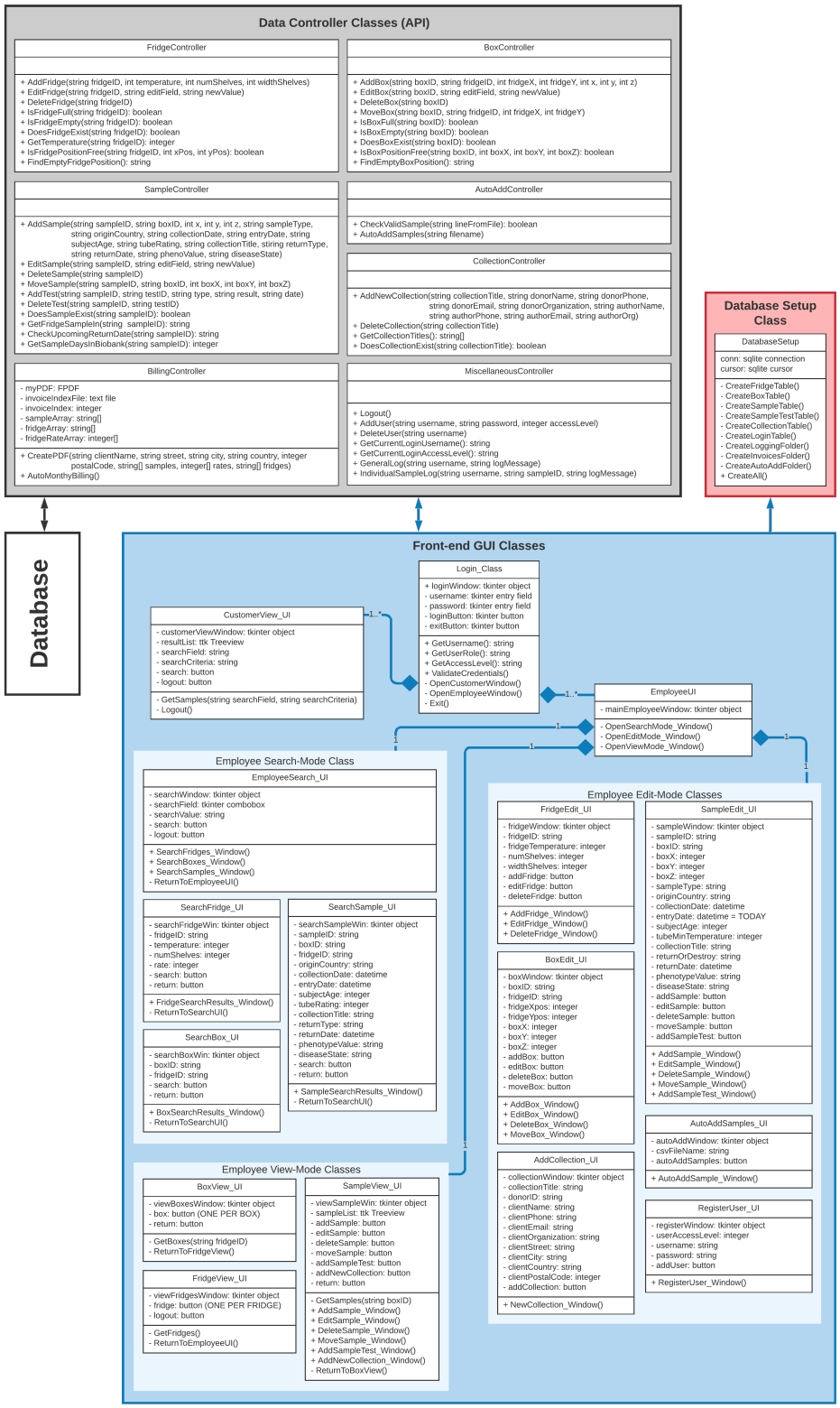

Figure 1: Program structure