

---

# Check In System Using Face Recognition

---

**Krystian Opryszek**

B.Sc.(Hons) in Software Development

APRIL 10, 2022

**Final Year Project**

Advised by: Daniel Cregg

Department of Computer Science and Applied Physics  
Galway-Mayo Institute of Technology (GMIT)



Ollscoil  
Teicneolaíochta  
an Atlantaigh

Atlantic  
Technological  
University

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Context . . . . .	6
1.2	Application Description . . . . .	7
1.3	Objectives . . . . .	7
1.4	Scope . . . . .	8
1.5	Chapter Overview . . . . .	8
1.5.1	Introduction . . . . .	8
1.5.2	Methodology . . . . .	8
1.5.3	Major Issues . . . . .	8
1.5.4	Technology Review . . . . .	9
1.5.5	System Design . . . . .	9
1.5.6	System Evaluation . . . . .	9
1.5.7	Conclusion . . . . .	9
<b>2</b>	<b>Methodology</b>	<b>10</b>
2.1	Research Methodology . . . . .	10
2.1.1	Quantitative . . . . .	11
2.2	Software Methodology . . . . .	11
2.2.1	Rapid Application Development . . . . .	11
2.2.2	Requirements Planning (Phase 1) . . . . .	12
2.2.3	User Design (Phase 2) . . . . .	12
2.2.4	Construction (Phase 3) . . . . .	12
2.2.5	Cutover (Phase 4) . . . . .	12
2.3	Meetings . . . . .	13
2.4	Version Control System . . . . .	14
2.5	Testing . . . . .	14
<b>3</b>	<b>Technology Review</b>	<b>15</b>
3.1	Python . . . . .	15
3.2	OpenCv . . . . .	16
3.3	MongoDB . . . . .	18

3.3.1	GridFS . . . . .	18
3.3.2	Create Operations . . . . .	20
3.3.3	Read Operations . . . . .	20
3.3.4	Delete Operations . . . . .	21
3.4	Tkinter . . . . .	22
3.5	Face-recogniton . . . . .	23
<b>4</b>	<b>System Design</b>	<b>25</b>
4.1	Overview . . . . .	25
4.2	Presentation Tier . . . . .	26
4.3	Application Tier . . . . .	27
4.3.1	Main file . . . . .	27
4.3.2	Capture Image file . . . . .	28
4.3.3	Email Notification file . . . . .	29
4.3.4	Face file . . . . .	29
4.3.5	Train Image file . . . . .	29
4.4	Data Tier . . . . .	30
4.5	User Guide . . . . .	36
<b>5</b>	<b>System Evaluation</b>	<b>40</b>
5.1	Objectives . . . . .	40
5.2	Testing . . . . .	41
5.2.1	Application Functionality Testing . . . . .	41
5.2.2	Database Testing . . . . .	43
5.3	Limitations . . . . .	46
5.3.1	Failing to install dlib . . . . .	46
5.3.2	User input for saving images . . . . .	46
5.3.3	Saving student names to the database . . . . .	47
5.3.4	Send an email after completing a health check form . . . . .	47
5.3.5	Time frozen when program running . . . . .	48
5.3.6	Retrieving names from the database during checking-in . . . . .	48
<b>6</b>	<b>Conclusion</b>	<b>50</b>
6.1	Findings and Insights . . . . .	50
6.2	Future work . . . . .	51
6.3	Final thoughts . . . . .	52
<b>7</b>	<b>Appendices</b>	<b>53</b>
7.1	GitHub Repository . . . . .	53

# List of Figures

2.1	Phases of Rapid Application Development. . . . .	12
3.1	OpenCV detecting cars and people. . . . .	17
3.2	Create Operations. . . . .	20
3.3	Read Operations. . . . .	20
3.4	Delete Operations. . . . .	21
3.5	Main menu of the program developed using Tkinter. . . . .	22
4.1	Three-tier architecture diagram. . . . .	25
4.2	Three-tier architecture diagram. . . . .	36
4.3	Main menu of the application. . . . .	37
4.4	Main menu of the application. . . . .	37
4.5	Main menu of the application. . . . .	38
4.6	Three-tier architecture diagram. . . . .	39
5.1	Command-Line test using PyTest testing framework. . . . .	41
5.2	Error message when trying to install Dlib library. . . . .	46
5.3	Error message when trying to send an email. . . . .	47
5.4	Time stops after the program get's launched. . . . .	48

# About this project

**Abstract** In today's environment, we are all being tracked and monitored. When it comes to user authentication, people are constantly forgetting their passwords. Research shows that more than half of people are forgetting their passwords on regular basis. This is due to a number of factors such as the complexity of the chosen password. Recognizing and authenticating users by their faces is a smarter and faster way of checking in users compared to the traditional way of taking attendance such as paper timesheets or manual handwritten journals.

Face recognition is becoming a very popular way of authenticating users across multiple platforms and devices. Facial recognition is capable of identifying people by their faces. The main purpose of authenticating users by their face is that the user is not required to have an item and be capable of authenticating successfully.

**Authors** The author of this project is Krystian Opryszek, a final year student in the Galway-Mayo Institute of Technology completing an (Honours) degree in Software Development.

# Chapter 1

## Introduction

### 1.1 Context

For my final year project, I wanted to come up with an application that is at the right level for a 4th-year project. I decided to challenge myself and pick a programming language that I had very little experience with. I chose to do my project in Python because I wanted to gain more experience and improve my programming skills using Python. As I have a great interest in biometrics authentication, I decided to use facial recognition as a part of the authentication. With the current pandemic, students are filling out forms online, I decided It would be better to have a piece of software to do it instead. I decided to combine these together and come up with an application that will complete the form using the student's face. For an easier and quicker way of finding data, I decided to store all student's details on the database. Databases allow a large volume of data to be stored securely in one place.

Study shows that most people usually forget their password which leads them to reset it [2].

An online survey shows [3] that 23% of people sometimes forget their passwords, 51% of them declared they rarely forget and 22% declared they never forget passwords on an application they use frequently. On applications they use rarely, 35% of people forget their password, 39% sometimes forget and 4% never forget. Based on the above results, people have difficulty remembering their passwords. To eliminate this problem, I have designed this application so that students are not required to remember anything or have any item on them, all they need is their face to check-in.

Face recognition is being very popular and I wanted to show that it is possible to authenticate students by their faces. Face recognition is far from

perfect, it uses biological characteristics, these characteristics might change over time [4], causing scanning issues when scanning the face, if there is a slight change in the face the system will fail the authentication.

I wanted to come up with an application that has it all integrated into one. The idea of my project was to develop a modern and user-friendly application that will allow students to check-in using their faces in a smarter and faster way.

## 1.2 Application Description

This application is a face recognition program built using Python and OpenCV which allows capturing real-time computer vision. When starting the program for the first time, students are required to register. Once the student is registered, then he can proceed to check-in. During the check-in, the student is required to fill out a health check form that requires a mobile number, the college that the student is attending, and confirm that the student is symptom-free. The program then searches for the known faces and if the face in real-time matches the known faces which are stored on the database, then the student will successfully check-in.

## 1.3 Objectives

The aim of this project was to develop a smarter and faster way of checking in students. Here are the main objectives of this project.

- To allow students to register.
- To check in students by their face.
- To provide a smart and fast way of completing health check form.
- To provide a user-friendly user interface.
- To allow administrators to manage/view student details.
- Store student's details securely on a database.

## 1.4 Scope

Based on the objectives set at the beginning of the development, I knew the scope was large. Despite the scope being large, I knew I could achieve all the requirements on time. The tasks I had to carry out during the project included:

- Create a fully functional check-in system for recognizing students by their faces.
- Develop a smart and fast way of completing health check forms.
- Develop and design a modern user-friendly user interface that is easy to use and navigate through the application.
- Create a database that could securely store all student details.

## 1.5 Chapter Overview

I will discuss a brief overview of the chapters. Chapters that are included are Introduction, Methodology, Major Issues, Technology Review, System Design, System Evaluation and Conclusion.

### 1.5.1 Introduction

In this chapter, I outline the description of the chosen project, why I picked it, the objectives, and a brief overview of all chapters in this dissertation.

### 1.5.2 Methodology

This chapter covers the methodologies used in this project. This includes how I went about the development of the project, planning, meetings with the supervisor, weekly status updates, testing, and the research involved in this project.

### 1.5.3 Major Issues

In this chapter, I talk about the issues that I have run into while developing my project. It was a challenge to find a solution and resolve the issue. Some issues were easier to resolve and some required a lot of research and many hours to figure out. I state clearly all the issues and the solutions that I found helpful to resolve them.



### **1.5.4 Technology Review**

In this chapter, I have outlined all the technologies that have been used in development. Each technology is discussed in great detail with its advantages and disadvantages. The project uses the following technologies: Python as the main programming language that manages all the functionality, Tkinter as the GUI, OpenCV for capturing images, MongoDB for storing student details, and a face recognition library for recognizing faces.

### **1.5.5 System Design**

In this chapter, I explain the overall system architecture of the project. With the use of code snippets, I want to explain and show the way the project works.

### **1.5.6 System Evaluation**

This chapter evaluates the project against the objectives already set out in the Introduction.

### **1.5.7 Conclusion**

In the final chapter, I summarise the context and the objectives of the project. I go over the final goals and the learning outcome that I have gained throughout this project. Finally, I include some final thoughts about the project and state what could I do differently if I had to do this all over again.

# Chapter 2

## Methodology

In this chapter, I will describe the way I went about my project. I will look at the methodologies I have used throughout the project in terms of software development, research methodology, meetings with the supervisor, and the version control system used for hosting and managing the code.

### 2.1 Research Methodology

Shortly after our final assignment was released, I started to research some of the ideas for my assignment. For my final year project, I decided to challenge myself and pick a programming language in which I have little experience. I knew from the very beginning that I wanted to create my project using Python. I wanted to gain more experience and improve my programming skills.

At first, I started looking at several python applications to come up with a project idea. I found that Python is commonly used for developing websites, artificial intelligence (AI), and machine learning projects. I have previously created a movie website so I decided to go for something different. After reading more on machine learning, I came across deep learning which is a broader family of machine learning methods based on artificial neural networks.

Due to the pandemic, GMIT staff/students and visitors are required to complete a form prior to entry. This is a public health requirement to prevent the spread of COVID-19. While attending college, I noticed that not many students in my class have ever completed this form. I decided to come up with a project that will help to increase the number of students completing this form. After gathering all my thoughts, I decided to develop a check-in system using face recognition. I picked face recognition for the authentication process as it's becoming very popular and It has up to 99.38% accuracy.

### 2.1.1 Quantitative

Throughout the project, I applied Quantitative research methodology as I felt it would be best suited for my project. The quantitative research methodology focuses on numbers and statistics of data collected [13]. The purpose of quantitative research is to test the gathered data, look at the cause, and effect, and make predictions. I found this to be appropriate for my project from the early stages of the development process. Throughout the development, I performed a set of tests to collect data. I observed what happened and made predictions from the gathered data.

#### **Advantages:**

- Can be tested and checked
- Straightforward analysis
- Statistics can be derived from the results.
- Has precision and is definitive
- Can answer questions such as “how many?” and “how often?”

#### **Disadvantages:**

- Limited to numbers and figures.
- Can be misleading.
- Can be expensive.

## 2.2 Software Methodology

After looking at several software development methodologies I have discovered that the Rapid Application Development (RAD)[5] also called Rapid Application Building (RAB) approach would be best suited for my project.

### 2.2.1 Rapid Application Development

The RAD approach is a very popular strategy in software development that prioritizes the development of prototypes over designing and planning. I have chosen this approach because requirements can be changed at any time and quick development of prototypes. RAD has four phases:

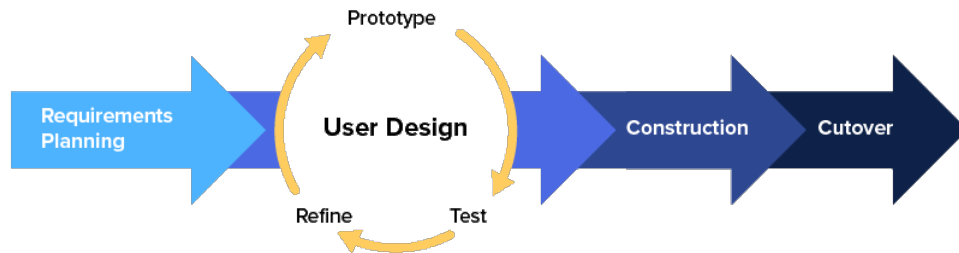


Figure 2.1: Phases of Rapid Application Development.

### 2.2.2 Requirements Planning (Phase 1)

This is the first phase of Rapid Application Development. In this phase, a series of meetings initiate the development process by understanding the objectives of the project. There is no need to sit down with end users and get detailed specifications, rather a broad requirement is all that is required [6].

### 2.2.3 User Design (Phase 2)

In the second phase of Rapid Application Development, the developers go straight into developing prototypes. Developers create a number of different prototypes with unique functions. Then they test it and see if it works, if it doesn't they create a new prototype until they are satisfied with the results. The goal is to present something as soon as possible [7].

### 2.2.4 Construction (Phase 3)

It is during the construction phase that prototypes developed in the previous phase are refined. This phase is very similar to the System development life cycle which focuses on development tasks. However, in RAD users can continue to make changes. It is time to develop and test the application so that it is ready for production.

### 2.2.5 Cutover (Phase 4)

This is the last phase in Rapid Application Development. This phase involves final changes, final testing before the launch, and user training before its available to use.

**Advantages:**

- Requirements can be changed at any time [6].
- The time between prototypes is short.
- Focuses on customers and their feedback.
- Reduces the development time.
- The finished product is ready very quickly.

**Disadvantages:**

- More complex to manage when compared to other models [6].
- Cannot work with large teams.
- Only suitable for projects which have a short development time.
- Requires user involvement throughout the life cycle.
- Highly depends on modeling skills.
- The cost of modeling is very high

## 2.3 Meetings

The weekly meetings have been initially organized at the start of the semester by my supervisor. In the meetings, we have discussed the goals and ideas for the project. Each week, I met with my supervisor to discuss the following:

- Any changes made to the application.
- Progress and feedback.
- Answer any questions from the supervisor.
- Demo the application.

Each week I have posted a weekly update status in which I outline what I was doing that week and what I was planning on doing in the upcoming week. Weekly update statuses helped me to organize and track my work each week.

## 2.4 Version Control System

I have used GitHub for hosting and managing my code. Any changes that I have made to my application have been regularly committed to my project repository. Any issues that I had discovered during the development phase were posted to the Issues section on GitHub. I made use of a Kanban-style board, which is available on GitHub to manage my project. With the use of columns To do, In Progress, and Done I was able to add any ideas I had regarding my project. I made regular changes and kept all columns up to date.

## 2.5 Testing

Throughout the development of the project, I decided to use testing as it's a process of verifying that an application does what it suppose to do. Testing is important as it prevents bugs and increases the quality of the software. In my project, I used Python's testing framework called **pytest** which helps to build better programs. Pytest is a command-line tool that automatically finds tests that have been written that start with the word "test" e.g. `test_one`.

# Chapter 3

## Technology Review

### 3.1 Python

The Python programming language was developed by Guido van Rossum and released in 1991. Python is a powerful general-purpose programming language [8], meaning it can be used in a range of applications. Python is widely used for tasks like web development, scripting, web scraping, data analysis, and automation. Python is interpreted, interactive, object-oriented programming language. In recent years, Python has become one of the best and most popular programming languages, because it has a simple syntax that gives more emphasis on natural language. Syntax in Python was designed to be readable and has some similarities to the English language. Fact that Python is an easy language to learn and use, Python codes can be written and executed much faster than other programming languages. Python supports operating systems such as Windows, Linux, and macOS.

**Code Example:** Python program to add two numbers.

```
1 # Python program to add two numbers
2
3 num1 = 15
4 num2 = 12
5
6 # Adding two nos
7 sum = num1 + num2
8
9 # printing values
10 print("Sum of {0} and {1} is {2}" .format(num1, num2, sum))
```

**Output:** Sum of 15 and 12 is 27

**Advantages:**

- **Clear and readable syntax** - Python was originally developed as a teaching language and Python's syntax is like the English language, making it easy to write, read and understand.
- **Libraries** - Python provides a number of packages and standard libraries. There are basically libraries for anything you can image, from web development to machine learning.
- **Free and Open-Source** - Python is an open-source, free programming language available to anyone.

**Disadvantages:**

- **Speed** - Compared to other programming languages such as Java or C++, Python is relatively slow because of the compilation and interpretation that occurs during the runtime.
- **Runtime errors** - The runtime errors are very common in Python which can be a disadvantage to the user. Python is dynamically typed, there can be changes in the data types at any time. For that reason, it requires more testing.
- **Mobile Development** - Python is not the best programming language for mobile applications as it uses a lot of memory and CPU time. It is not native to either iOS or Android because the deployment process can be slow and difficult. There are better options when it comes to developing mobile applications such as Java and Kotlin for Android and Swift for iOS.

## 3.2 OpenCv

OpenCV (Open Source Computer Vision Library) is a library mainly aimed at real-time computer vision [9]. OpenCV was launched in 1999 and was originally developed by Intel which then was later supported by Willow Garage. It is written in C/C++ and supports a variety of programming languages such as C++, Python, Java, etc. OpenCV runs on Windows, Android, Linux, and macOS. There are lots of applications that use OpenCV such as facial recognition, object detection, motion tracking, number plate recognition, and much more.



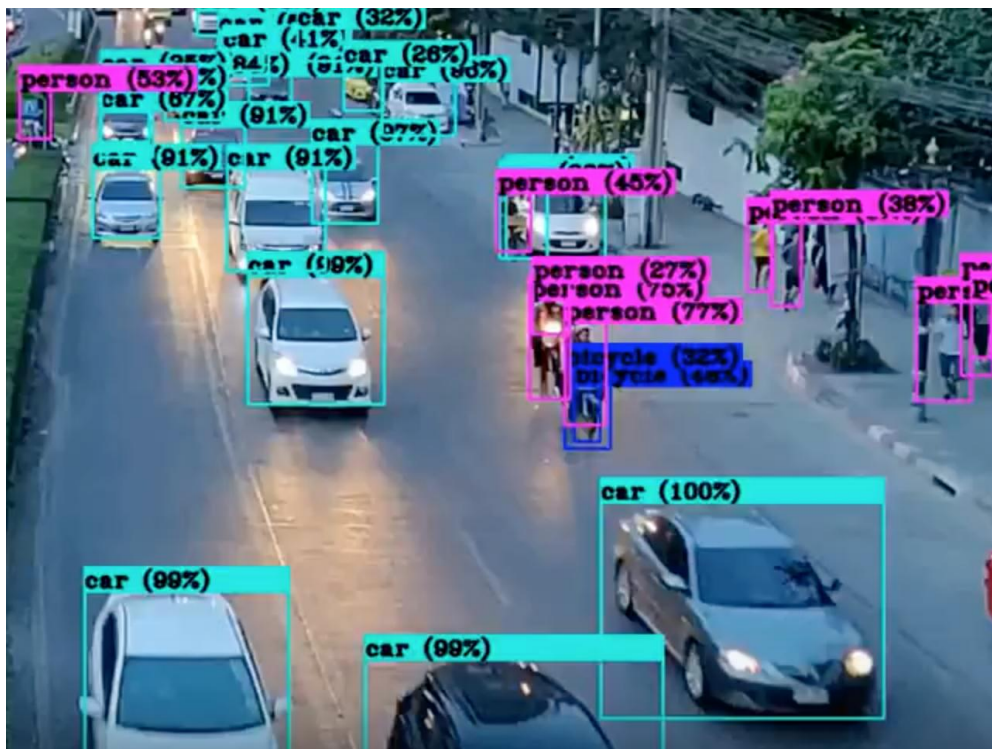


Figure 3.1: OpenCV detecting cars and people.

**Code Example:** Python program using OpenCV to read image.

```

1 import cv2
2
3 # To read the image.
4 img = cv2.imread("opencv.png", cv2.IMREAD_COLOR)
5
6 # Display the image on screen in a window.
7 cv2.imshow("OpenCV image", img)
8
9 # hold the screen until user close it.
10 cv2.waitKey(0)
11
12 # delete created window.
13 cv2.destroyAllWindows()

```

**Output:** Displays the image on screen in a window.

**Advantages:**

- **Free and open source** - OpenCV is an open-source machine learning software. OpenCV is a free-to-use library.

- **Various Algorithms** - OpenCV contains over 2,500 algorithms. Users can perform a variety of tasks like following face or eye movements.
- **Widely Supported** - Supports many devices such as Desktop, Android, and iOS. Runs on operating systems: Windows, Linux, macOS, NetBSD, FreeBSD, and OpenBSD.
- **Efficiency** - The library supports a wide range of matrix operations and is highly optimized. In comparison to Numpy, OpenCV is roughly 25 times faster. It's possible that there will be exceptions, especially since Numpy works with views instead of copies.
- **Low Memory Usage** - OpenCV uses low Ram usage, around 60-70 Mb.

**Disadvantages:**

- **Installation** - The installation of OpenCV can be complicated and this is due to a big number of dependencies and optimal things.

## 3.3 MongoDB

MongoDB is a document-oriented NoSQL database that is widely used for the storage of data [10]. It is one of the most popular database programs in the world. MongoDB was developed by MongoDB Inc which was released in 2009. Documents are written individually containing whatever data they require in JSON format, rather than storing data into tables, rows, and columns. MongoDB is secure, all the data which is stored is encrypted and can't be decrypted by other users or MongoDB. Only users who own the database can see and manage their data. MongoDB supports Windows, Linux, macOS, Solaris, and FreeBSD.

### 3.3.1 GridFS

GridFS is a method for storing and retrieving a file that exceeds the BSON document limit size of 16Mb. GridFS can store large files such as images, audio, and videos. GridFS stores separate parts of a document, instead of storing a single document. It uses two collections to store files. One collection stores the file chunks and the other stores file metadata.

- **Chunks** - Stores the binary chunks. It has a unique identifier for the chunk that is labeled as `_id`. The chunks are limited to a size of 255KB each.

**Example:** The chunks collection.

```
1 {  
2   "_id": ObjectId("624da424d311...."),  
3   "files_id": ObjectId("624da424d3...."),  
4   "n": 0  
5   "data": Binary('/9j/4AAQSkZJRg....', 0)  
6 }
```

- **Files** - Stores the file's metadata.

**Example:** The files collection.

```
1 {  
2   "_id": ObjectId("624430e3dfa1...."),  
3   "filename": "G00366895",  
4   "chunkSize": "261120",  
5   "length": "58728",  
6   "uploadDate": "2022-03-30T10:28:52.417+00:00"  
7 }
```

For storing student ID (image of the face) I used GridFS. GridFS stores large binary objects in MongoDB. The simplest way to use GridFS is to use a key/value interface `put()`.

```
1 a = fs.put(b"hello world")
```

Similar to storing the images, I've used GridFS to retrieve the images (student ID) from MongoDB. The simplest way to use GridFS is to use a key/value interface `get()`.

```
1 fs.get(a).read()
```

### 3.3.2 Create Operations

A create or insert operation adds new documents to a collection. If a collection doesn't exist, the insert operation will create the collection. This is used when performing the insertion of single or many documents into the database.

```
1 db.collection.insertOne()  
2 db.collection.insertMany()
```

```
db.users.insertOne(  ← collection  
  {  
    name: "sue",      ← field: value  
    age: 26,          ← field: value  
    status: "pending" ← field: value  } document  
  }  
)
```

Figure 3.2: Create Operations.

### 3.3.3 Read Operations

The read operations retrieve the document from a collection. I've used this operation to retrieve the student id (image of student's face). MongoDB provides the following to read documents from a collection:

```
1 db.collection.findOne()  
2 db.collection.insertMany()
```

```
db.users.find(                                ← collection  
  { age: { $gt: 18 } },                      ← query criteria  
  { name: 1, address: 1 }                    ← projection  
) .limit(5)                                  ← cursor modifier
```

Figure 3.3: Read Operations.

### 3.3.4 Delete Operations

Delete Operations remove documents from a collection. It removes single or many documents from a collection. MongoDB provides the following to delete documents of a collection:

```
1 db.collection.deleteOne()  
2 db.collection.deleteMany()  
  
db.users.deleteMany(  
    { status: "reject" }  
)
```




Figure 3.4: Delete Operations.

#### Advantages:

- **Flexible** - MongoDB is a good choice for modern applications as it offers a flexible schema for your data. MongoDB JSON-like data format allows you to have objects in one collection with different fields.
- **Sharding** - It's a process of storing data across multiple machines. With MongoDB, large data sets can be deployed using sharding.
- **High Speed** - MongoDB is a lot faster compared to the relational database. With the use of indexing, It's easy to access documents. MongoDB can insert and update multiple records at once with insertMany and updateMany.
- **Widely supported** - MongoDB is one of the most popular NoSQL databases and is widely used to store and manage unstructured data. It supports various programming languages such as Python, Java, C++, C#, and much more.

#### Disadvantages:

- **Joins not supported** - Doesn't support joins such as relational database, so you must perform multiple queries to retrieve and join data manually. This may affect the performance and slow down the execution time.

- **High memory usage** - Uses high memory for data storage. As joins aren't functional, there is a lot of redundant data. This increases the unnecessary usage of memory.
- **Limit Document Size** - The limit for one document is 16Mb. You can use GridFS to store large files that exceed 16Mb, this will store them in multiple chunks.

## 3.4 Tkinter

Python has many GUI (Graphical User Interface) frameworks but Tkinter is the only framework that's built into the Python standard library. Tkinter is a Python binding to the Tk GUI toolkit and was written by Steen Lumholt and Guido van Rossum, later revised by Fredrik Lundh. It provides a fast and easy way of creating GUI applications. Tkinter was designed to be used on desktop applications and supports operating systems such as Windows, Linux, and macOS.

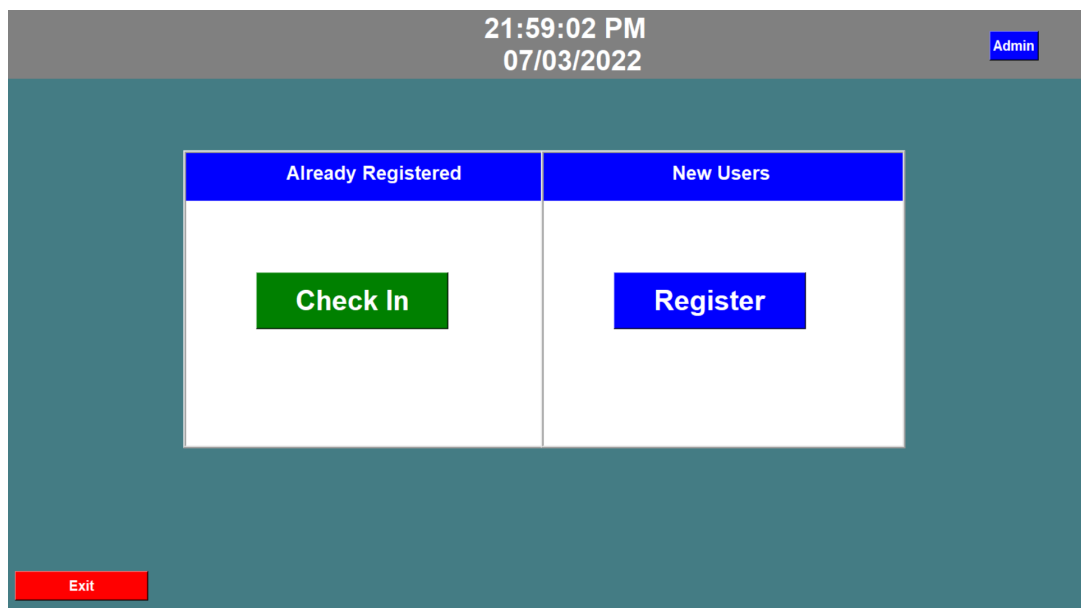


Figure 3.5: Main menu of the program developed using Tkinter.

**Code Example:** Basic Python Tkinter application with one widget.

```
1 from tkinter import *
2
3 # Create the root (base) window
4 root = Tk()
5 # Create a label with words
6 w = Label(root, text="Hello, world!")
7 # Put the label into the window
8 w.pack()
9
10 root.mainloop()
```

**Advantages:**

- **Easy to use** - Tkinter is easy and fast for implementing Graphical User Interface compared to other GUI toolkits. Tkinter is easy to understand and provides simple syntax.
- **Standard libraries** - Since Tkinter is Python's standard library, nothing extra is needed to be downloaded.
- **Flexible** - Tkinter is flexible and stable.

**Disadvantages:**

- **Widgets** - Compared to other GUI toolkits such as PyQt or wxPython, Tkinter does not include advanced widgets.
- **UI** - It doesn't have a reliable UI.

## 3.5 Face-recognition

Face recognition is built using dlib's state-of-the-art face recognition built with deep learning. Face recognition is a library in Python for recognizing and managing faces. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark [11].

**Advantages:**

- **High accuracy** - Face recognition library provides high accuracy of 99.38%.
- **Documentation** - The documentation for the face recognition library is very good as it shows you some of the examples. It helps you to understand the whole process, making it a lot easier to implement into your own code. Provides a full down on how to install it on various platforms.

**Disadvantages:**

- **Installation** - To install the face recognition library, you need to have dlib installed on your machine which is quite of a challenge.
- **Quality** - Image quality needs to be sharp and clear for the best accuracy results.



# Chapter 4

## System Design

### 4.1 Overview

The application was developed using a Three-tier architecture. The three-tier architecture is a client-server software architecture that consists of a presentation tier, an application tier, and a data tier. The presentation tier is the graphical user interface that allows communication with the other two tiers, the application tier handles the logic, and the data tier stores the information. hello.

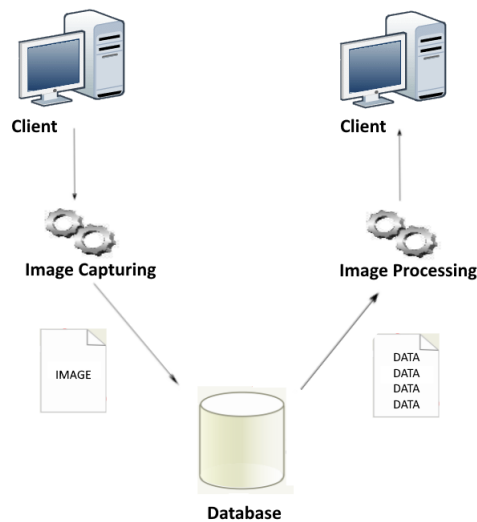


Figure 4.1: Three-tier architecture diagram.

## 4.2 Presentation Tier

The presentation tier is at the top level of the three-tier architecture. The presentation tier is the graphical user interface and communication layer of the application, where the end-user interacts with the application directly. This tier is built with Tkinter Python's standard GUI library. The end-user is presented with some menus which allow the user to navigate through the application. The main purpose of the presentation tier is to display the results to the client in a readable fashion which is possible with the use of GUI.

- **Main menu** - This is the main menu of the application that allows the end-user to navigate through the application. With the use of `grid_forget()`, it is possible to hide Tkinter windows when navigating through different menus.
- **Registration menu** - The registration menu allows the end-user to register to the application. User is asked to enter their student number (ID) which then opens a webcam that automatically takes an image.
- **Health check form menu** - Displays a smaller version of GMIT's health check form [1]. The user must fill out all fields in order to successfully check-in. Once all of the fields are filled, the application will save the form to the database and open a webcam which will then check in the user. If there are no students registered, an alert will be displayed on the user's screen.
- **Checked-in menu** - This menu appears once the user successfully checks in. This menu displays the student's ID, time, date, type of verification, and text saying the verification has been successful. This menu is only displayed for 5 seconds, then it takes the user back to the main menu.
- **Admin login menu** - This is a menu designed for admins only. This menu allows the admin to login in to access the admin's dashboard. Admin must enter the correct password to successfully log in. When the password is incorrect, an alert is displayed.
- **Admin dashboard menu** - Displays a menu with five frames that display, the total number of registered students, the total number of checked-in students, the total number of completed forms, total symptom-free students, and the total number of students with symptoms.

- **All students menu** - In this menu, two types of data are joined so they can be displayed together in the same menu. This menu displays the total number of registered students and their email addresses.
- **Student logs menu** - This menu displays all the checked-in students which shows their student ID, time, and date of check-in.
- **Health check menu** - Displays all the completed health check forms that show the student's phone number, the college they are attending, and the confirmation on whether they have symptoms or are symptom-free.

## 4.3 Application Tier

The application tier is also known as the business logic, logic tier, or middle tier, which is pulled out from the presentation tier. This tier is written using Python which controls the application's functionality.

### 4.3.1 Main file

- **Capture Image** - Checks if the entry field in the registration form is empty. If it is, it will display an error message on the screen. Otherwise, it will pass the entered student ID to the function for capturing an image. After the image is captured, the student ID and the email get stored in the database. Finally, the image gets deleted from the folder. GridFS requires a destination folder for saving and retrieving images from the database. To show that the image gets successfully saved in the database, the image gets deleted from the local machine.
- **Send Email** - When the student successfully completes the registration form, this function gets called and sends an email notification.
- **Threading** - To improve efficiency, the program is using threads to allow the execution of two functions at the same time. With this code optimization, there is no waiting time, the action is performed instantly, compared to 20 seconds of waiting time for an execution task to be completed.
- **Show Time** - This is a function that displays the current time which is used in the main menu and checked in the menu.

- **Facial Recognition** - This function waits to be called when the end-user performs a check-in. This is used when the health check form gets completed, once the button “Submit” gets clicked, it will then run this function. The main purpose of this function is to run the `Face_Match` which manages face recognition and allows the student to check-in.
- **Close Program** - Basic function for closing the program when it gets called. Used in the main menu when the button “Exit” gets clicked.
- **Password** - Used for login authentication to the admin’s dashboard. If the password entered is correct, it will then successfully display the admin dashboard, otherwise, it will display an alert with the message “wrong password”.
- **Delete by id** - This function is used to pass the entered student id to the database file which does the delete operation. Only administrators have the privileges to delete students from the database.
- **Delete all** - Function for deleting all data from the database. Only administrators have privileges to this action. An alert message is displayed on the screen which asks for confirmation on the deletion of all data from the database. This is a safety feature in case the button is clicked by accident.
- **View chart** - Opens a variety of MongoDB charts based on the stored data in the database. The charts are displayed in a web browser which is interactable, meaning that you can hover on the charts to view more statistics such as the percentage of students who have symptoms and much more.
- **On Submit** - Manages the entry fields, and stores all the entered values from the health check form in the database and as a .csv file. If any entry field is empty, an error message will be displayed on the screen. Finally, it checks if any of the students have symptoms, which means they have selected “No” on the health check form, then the program won’t allow them to check-in.

#### 4.3.2 Capture Image file

- **Capture** - This function is designed for capturing images. The end-user is presented with a real-time video stream that displays a rectangle which is used to indicate where the face should be placed during the image capturing. The user has 5 seconds before the image gets captured

and saved in the database. Finally, the webcam closes and the main menu is displayed. If the images get successfully saved, an information alert will be displayed on the screen.

### 4.3.3 Email Notification file

- **Email Notification** - Used for setting up the email notification system. Manages the following: ‘to’ who the email is being sent to, ‘subject’ which is the subject of the email, and finally, the ‘body’ which is the email message that gets sent. The message body uses HTML to create a modern look of the message body.
- **Send Notification** - This function is designed to send an email notification with a message confirming the registration. To reduce the time to complete the registration, the function is designed to only take in the student ID which then gets used in another file with the use of `get()`. The passed-in student ID then gets added to a string “@gmit.ie” which is used to send the email. This way, the program eliminates time for entering the email address.

### 4.3.4 Face file

- **Face Match** - This is a class that manages the entire face recognition. The images that are retrieved from the database, are then trained using the `face_recognition` library and get all the face encodings. After that, it gets all faces that it knows and finds the best match. With the use of the `face_recognition` library, it is possible to find the location of the faces which is used to find the best match. Once it finds the best match it will then display it to the end-user with the name of the best match.
- **Check-In** - This function is used to store student’s check-ins in the database and as a .csv file.

### 4.3.5 Train Image file

- **Get Encodings** - This is the training of the images which computes all the encodings. With the use of the `face_recognition` library, it finds the encodings and passes them to the Face Match class in the face file which then uses it to find the best match.

## 4.4 Data Tier

The data tier controls the database where the information is being stored. The data tier manages the read and writes access to the database. This tier is managed by MongoDB which is a document-oriented database that stores data in flexible and schema-less JSON-like documents.

**Example:** Data being stored on the database.

```

1  {
2      "_id": ObjectId("624430e3dfa14b67a4c690a7"),
3      "filename": "G00366895",
4      "chunkSize": "261120",
5      "length": "58728",
6      "uploadDate": "2022-03-30T10:28:52.417+00:00"
7  }
```

- **Store and retrieve** - This is used to store and retrieve student's images in the database. During the registration, the student ID that is entered gets stored on the database and then retrieved when checking in.

```

1  def store_retrieve(name):
2      # storing
3      db = mongo_conn()
4      file_location = "images/" + name.get() + ".jpg"
5      file_data = open(file_location, "rb")
6      data = file_data.read()
7      fs = gridfs.GridFS(db)
8      fs.put(data, filename = name.get())
9      messagebox.showinfo("Info", "Upload Completed!")
10     # retrieving
11     db = mongo_conn()
12     data = db.fs.files.find_one({'filename': name.get()})
13     my_id = data['_id']
14     fs = gridfs.GridFS(db)
15     outputdata = fs.get(my_id).read()
16     download_location = "download/" + name.get() + ".jpg"
17     output = open(download_location, "wb")
18     output.write(outputdata)
19     output.close()
20     print("download completed")
```

- **Store logs** - This is used to store the check-ins of all students. Stores student ID, time, and date of check-in.

```
1 def store_logs(name, timeString, dateString):
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["logs"]
5
6     post = {"Name": name, "Time": timeString,
7            "Date": dateString}
8     collection.insert_one(post)
```

- **Store form** - The values entered when completing the health check form are then stored as a basic text. Stores the mobile number, the college the student is attending, and the confirmation of whether the student is symptom-free or have any symptoms.

```
1 def store_form(mobile_var, college_attend, confirmation):
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["health_check_form"]
5
6     post = {"Mobile Number": mobile_var.get(), "College":
7            college_attend.get(), "Confirmation": confirmation.
8            get()}
9     collection.insert_one(post)
```

- **Store email** - During the registration, the value gets stored in the database. This stores student's email addresses.

```
1 def store_email(name):
2     cluster = MongoClient("connection string here")
3     db = cluster["Registration"]
4     collection = db["student_details"]
5
6     post = {"Email": name.get() + "@gmit.ie"}
7     collection.insert_one(post)
```

- **Delete student** - This is used to delete students by their id. It performs a `delete_one()` operation which deletes a single student from the database.

```
1 def delete_student(deleteID):
2     cluster = MongoClient("connection string here")
3     db = cluster["Registration"]
4     collection = db["fs.files"]
5
6     # Delete student by ID.
```

```

7     deleteStudentID = {"filename": deleteID.get()}
8     collection.delete_one(deleteStudentID)
9     print("[INFO] Student with ID: " + deleteID.get() + "
      has been deleted from the database.")

```

- **Delete email** - This is used to delete student email address from the database. It performs a `delete_one()` operation which means it only delete's one email address. This function is used with the previous function "Delete student" so there is no data left containing that specific id in the database.

```

1 def delete_email(deleteID):
2     cluster = MongoClient("connection string here")
3     db = cluster["Registration"]
4     collection = db["student_details"]
5
6     # Delete student email.
7     deleteEmail = {"Email": deleteID.get() + "@gmit.ie"}
8     collection.delete_one(deleteEmail)

```

- **Delete all** - This is used to delete all data in the database. Only administrators have the privileges to perform this action. It performs a `delete_all()` operation which deletes the following collections: `student_details`, `fs.files`, `fs.chunks`, `health_check_form` and `logs`.

```

1 def delete_all():
2     # Delete all registered students from the database.
3     cluster = MongoClient("connection string here")
4     db = cluster["Registration"]
5     collection = db["student_details"]
6     collection2 = db["fs.files"]
7     collection3 = db["fs.chunks"]
8
9     # Delete each collection.
10    collection.delete_many({})
11    collection2.delete_many({})
12    collection3.delete_many({})
13
14    # Delete students completed form and logs.
15    db2 = cluster["Students"]
16    collection4 = db2["health_check_form"]
17    collection5 = db2["logs"]
18
19    # Delete each collection.
20    collection4.delete_many({})
21    collection5.delete_many({})
22    print("[INFO] All data has been deleted from the
      database.")

```



- **All Students** - This is used to retrieve all student IDs and insert them into a Tkinter table to display them in the all student menu.

```

1 def all_students(n):
2     cluster = MongoClient("connection string here")
3     db = cluster["Registration"]
4     collection = db["fs.files"]
5
6     list = [['Students']]
7
8     list.clear()
9     list.append(["Students"])
10    cursor = collection.find({})
11    for text_fromDB in cursor:
12        filename = str(text_fromDB['filename'].encode('
utf-8')).decode("utf-8"))
13        list.append([filename])
14    return list

```

- **Student logs** - This is used to retrieve all student check-ins and insert the values in the Tkinter table to display them in the student logs menu.

```

1 def student_logs(n):
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["logs"]
5
6     list = [['Name', 'Time', 'Date']]
7
8     list.clear()
9     list.append(["Name", "Time", "Date"])
10    cursor = collection.find({})
11    for text_fromDB in cursor:
12        name = str(text_fromDB['Name'].encode('utf-8')).
decode("utf-8"))
13        time = str(text_fromDB['Time'].encode('utf-8')).
decode("utf-8"))
14        date = str(text_fromDB['Date'].encode('utf-8')).
decode("utf-8"))
15        list.append([name, time, date])
16    return list

```

- **Student details** - Used to retrieve student email address and insert the value in Tkinter table which is displayed in all student menu.

```

1 def student_details(n):
2     cluster = MongoClient("connection string here")
3     db = cluster["Registration"]

```

```

4     collection = db["student_details"]
5
6     list = [['Students Email']]
7
8     list.clear()
9     list.append(["Students Email"])
10    cursor = collection.find({})
11    for text_fromDB in cursor:
12        email = str(text_fromDB['Email'].encode('utf-8').
decode("utf-8"))
13        list.append([email])
14    return list

```

- **Health check** - The values of all completed forms are retrieved and the values are displayed in the Tkinter table which is then displayed in the health check menu.

```

1 def health_check(n):
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["health_check_form"]
5
6     list = [['Student Mobile Number', 'College Attending',
7 , 'Confirmation']]
8
9     list.clear()
10    list.append(["Student Mobile Number", "College
Attending", "Confirmation"])
11    cursor = collection.find({})
12    for text_fromDB in cursor:
13        mobile = str(text_fromDB['Mobile Number'].encode(
'utf-8').decode("utf-8"))
14        college = str(text_fromDB['College'].encode('utf
-8').decode("utf-8"))
15        conf = str(text_fromDB['Confirmation'].encode('
utf-8').decode("utf-8"))
16        list.append([mobile, college, conf])
17    return list

```

- **Total students** - This is used to retrieve the number of registered students in the database. The total number is then displayed in the admin's dashboard.

```

1 def total_health_check():
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["health_check_form"]
5

```

```
6     # number of documents in the collection
7     totalForms = collection.count_documents({})
8     return totalForms
```

- **Total check-in** - Used to retrieve the number of checked-in students. The total number is displayed in the admin's dashboard.

```
1 def total_checkin():
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["logs"]
5
6     # number of documents in the collection
7     totalCheckIns = collection.count_documents({})
8     return totalCheckIns
```

- **Total health check** - This retrieves the total number of completed health check forms and then is displayed in the admin's dashboard.

```
1 def total_health_check():
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["health_check_form"]
5
6     # number of documents in the collection
7     totalForms = collection.count_documents({})
8     return totalForms
```

- **Total with no symptoms** - This is used to retrieve the total number of symptom-free students who have selected the option "yes" when completing the health check form. This is then displayed in the admin's dashboard.

```
1 def total_with_no_symptoms():
2     cluster = MongoClient("connection string here")
3     db = cluster["Students"]
4     collection = db["health_check_form"]
5
6     totalNoSymptoms = collection.count_documents({'
Confirmation': "Yes"})
7     return totalNoSymptoms
```

- **Total with symptoms** - used to retrieve the number of students who have symptoms who selected the option "no" in the health check form. The total number is displayed in the admin's dashboard.

```
1 def total_with_symptoms():  
2     cluster = MongoClient("connection string here")  
3     db = cluster["Students"]  
4     collection = db["health_check_form"]  
5  
6     totalWithSymptoms = collection.count_documents({'  
Confirmation': "No"})  
7     return totalWithSymptoms
```

## 4.5 User Guide

To start the application run the following command:

```
1 python main.py
```

The application uses Tkinter as Graphical User Interface which allows the user to navigate through the application. The user will have the option to register or checking-in. New students, must register for the application first before checking in.

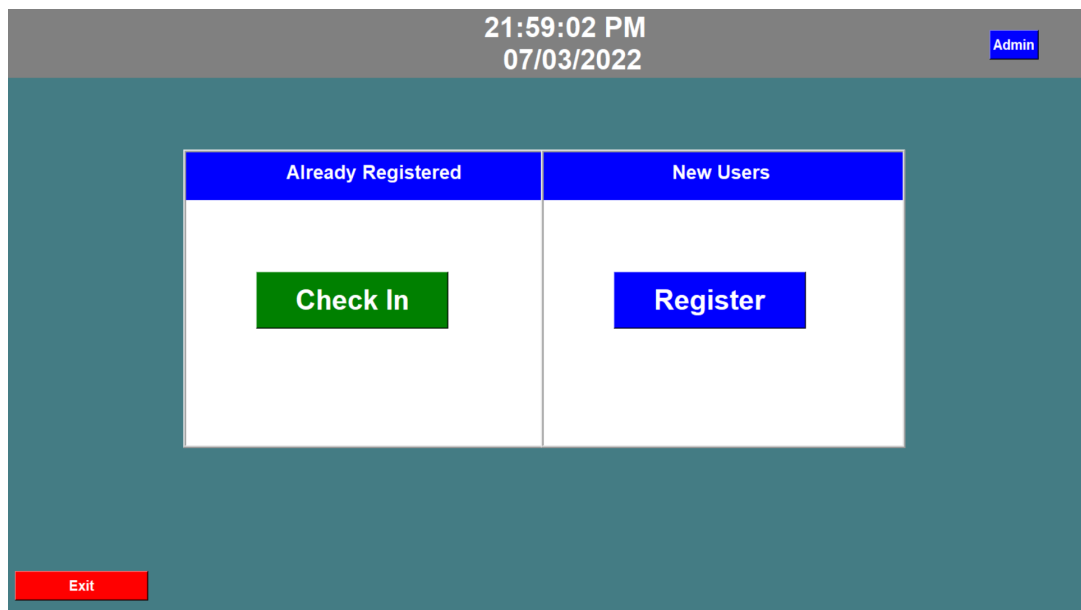


Figure 4.2: Three-tier architecture diagram.

The user will be provided with a registration menu that is very simple to complete, it only asks the user to enter his/hers student ID.

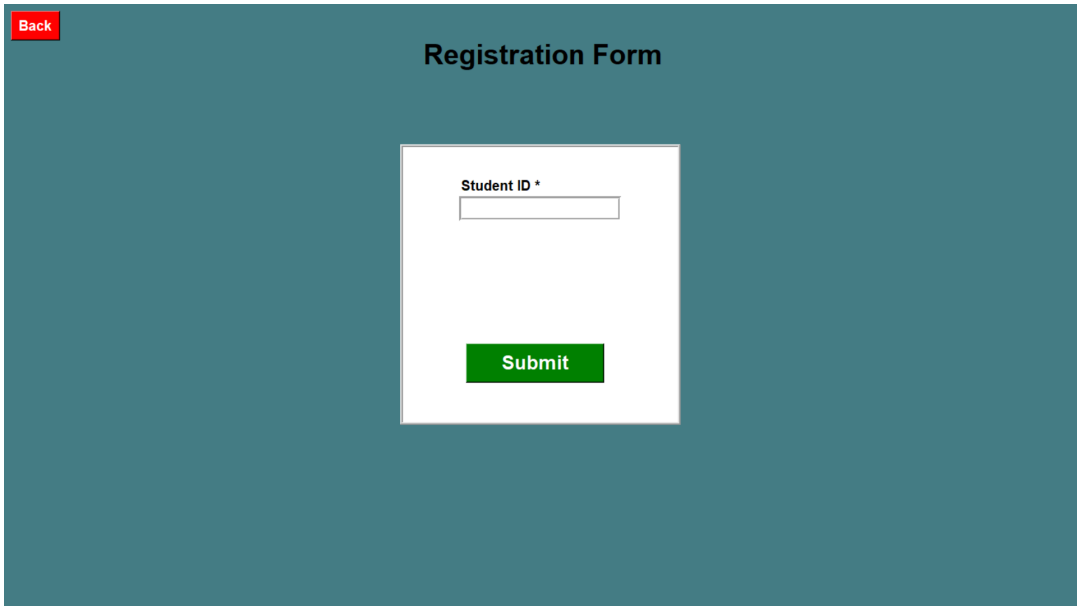
The image shows a web application interface with a teal background. In the top-left corner, there is a red button labeled "Back". Centered at the top is the title "Registration Form" in bold black text. Below the title is a white rectangular form box. Inside this box, the text "Student ID \*" is positioned above a single-line text input field. At the bottom of the white box is a green button with the word "Submit" in white text.

Figure 4.3: Main menu of the application.

Once a user enters his/hers student ID, a webcam will be launched. The user should place his head in the green box on the screen as shown below. The user has 5 seconds before the image gets taken.

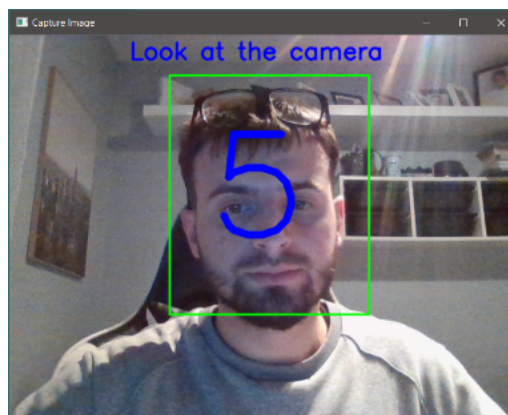


Figure 4.4: Main menu of the application.

If the registration is successful, an email will be sent to the student's email address. The email must be in this format in order to receive an email: **studentID@gmit.ie**.

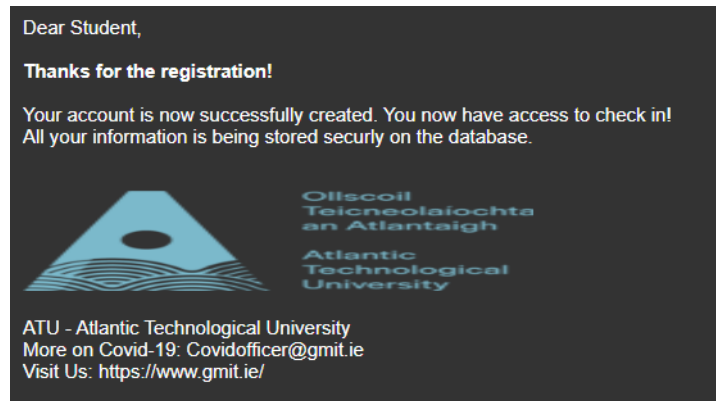
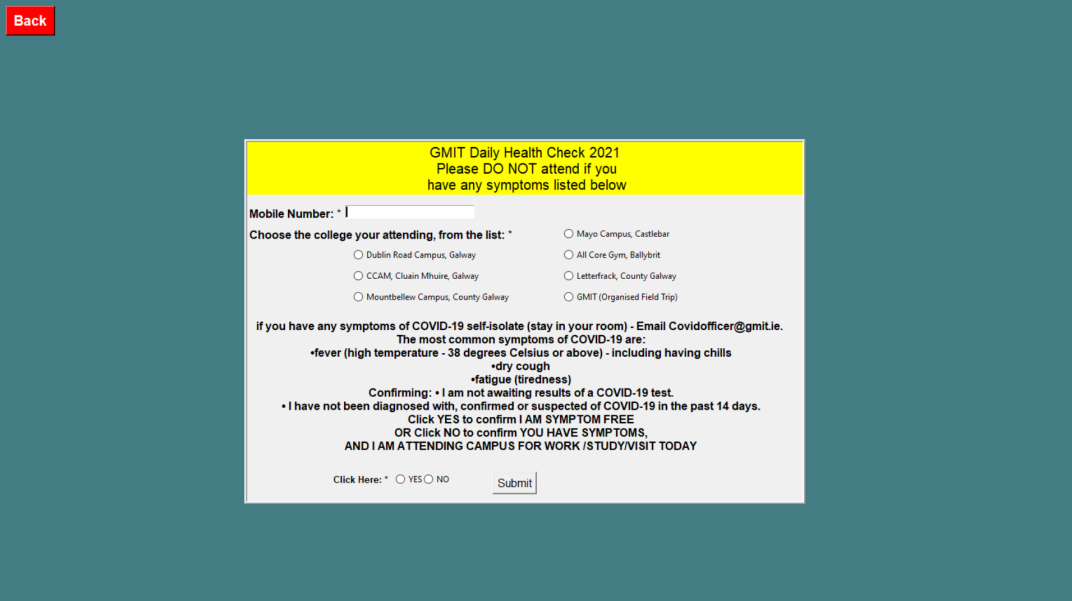


Figure 4.5: Main menu of the application.

Before the user gets checked in, the user is required to complete the health check form which is based on the GMIT health check form. The user will be provided with a menu that displays a form that requires the user to complete the following:

- Mobile number.
- College the user is attending.
- Confirmation if the user has any symptoms



The image shows a screenshot of a web form titled "GMIT Daily Health Check 2021". The form is set against a teal background. At the top left, there is a red "Back" button. The form itself has a yellow header with the title and a warning: "Please DO NOT attend if you have any symptoms listed below". Below the header, there is a "Mobile Number:" field with a text input. The next section is "Choose the college you attending, from the list: \*" with two columns of radio button options. The first column lists: Dublin Road Campus, Galway; CCAM, Cluain Mhuire, Galway; and Mountbellew Campus, County Galway. The second column lists: Mayo Campus, Castlebar; All Core Gym, Ballybrit; Letterfrack, County Galway; and GMIT (Organised Field Trip). Below the options, there is a paragraph of instructions: "if you have any symptoms of COVID-19 self-isolate (stay in your room) - Email Covidofficer@gmit.ie. The most common symptoms of COVID-19 are: •fever (high temperature - 38 degrees Celsius or above) - including having chills •dry cough •fatigue (tiredness)". This is followed by a "Confirming:" section with two bullet points: "• I am not awaiting results of a COVID-19 test." and "• I have not been diagnosed with, confirmed or suspected of COVID-19 in the past 14 days." Below this, there are instructions to "Click YES to confirm I AM SYMPTOM FREE" or "Click NO to confirm YOU HAVE SYMPTOMS, AND I AM ATTENDING CAMPUS FOR WORK /STUDY/VISIT TODAY". At the bottom, there is a "Click Here: \* YES NO" section with radio buttons and a "Submit" button.

Back

**GMIT Daily Health Check 2021**  
Please DO NOT attend if you have any symptoms listed below

Mobile Number: \* |

Choose the college you attending, from the list: \*

☐ Dublin Road Campus, Galway  
☐ CCAM, Cluain Mhuire, Galway  
☐ Mountbellew Campus, County Galway

☐ Mayo Campus, Castlebar  
☐ All Core Gym, Ballybrit  
☐ Letterfrack, County Galway  
☐ GMIT (Organised Field Trip)

if you have any symptoms of COVID-19 self-isolate (stay in your room) - Email Covidofficer@gmit.ie.  
The most common symptoms of COVID-19 are:  
•fever (high temperature - 38 degrees Celsius or above) - including having chills  
•dry cough  
•fatigue (tiredness)

Confirming: • I am not awaiting results of a COVID-19 test.  
• I have not been diagnosed with, confirmed or suspected of COVID-19 in the past 14 days.  
Click YES to confirm I AM SYMPTOM FREE  
OR Click NO to confirm YOU HAVE SYMPTOMS,  
AND I AM ATTENDING CAMPUS FOR WORK /STUDY/VISIT TODAY

Click Here: \* ☐ YES ☐ NO

Figure 4.6: Three-tier architecture diagram.

All fields must be filled and selected in order to complete the form. If the check-in is successful, a checked-in menu will be shown to the user.

# Chapter 5

## System Evaluation

### 5.1 Objectives

Based on the objectives from the Introduction, I believe I have successfully completed them all.

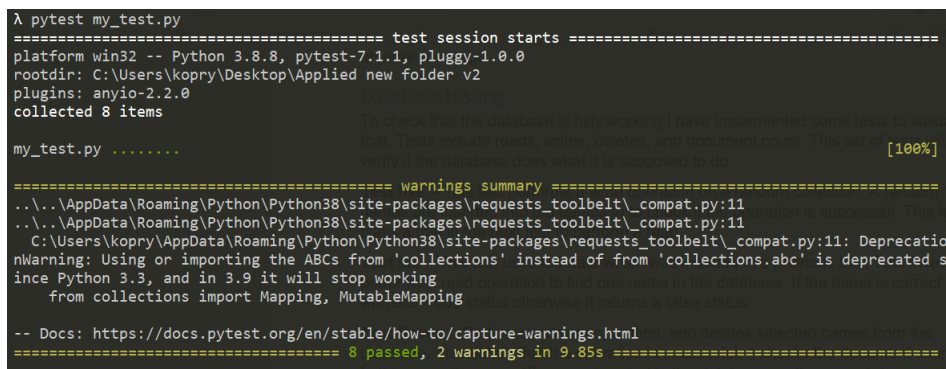
- **To allow students to register** – The application allows students to register. The registration process involves capturing of student's face.
- **To check in students by their faces** – This objective was one of the most important tasks in the entire project. This depended on whether the students can successfully check-in using their faces. The application is using `face_recognition` library which provides 99.38% of accuracy. The application provides very high precision when recognizing students. The application has been tested and the results have been good, but mistakes can happen.
- **To provide a smart and fast way of completing health check form** – Completes the form in a smart and fast way. To reduce the time for completing the form, the form has been shortened and the time for recognizing the student's face is very fast.
- **To provide a user-friendly user interface** – When it comes to the user interface, it is important to design a user-friendly user interface that is easy to use. This application uses Tkinter as the graphical user interface which is a modern GUI tool kit that is Python's standard library. When developing this project, I have focused on creating a modern user interface that will be easy to navigate through the application.



- **To allow administrators to manage/view student detail** – Administrators have the privileges to manage and view all student details.
- **Store student's details securely on a database** – It is important to store and manage all student's details on the database. The application uses MongoDB which stores all data securely.

## 5.2 Testing

Throughout the development of my project, I integrated a set of tests to evaluate and verify that the application is doing what is supposed to. I have used the PyTest testing framework to carry out multiple tests. I have broken down the tests into two parts, application functionality testing, and database testing. The application functionality testing is done to validate all the application's functionality and database testing to check the schema, tables, etc.



```

λ pytest my_test.py
===== test session starts =====
platform win32 -- Python 3.8.8, pytest-7.1.1, pluggy-1.0.0
rootdir: C:\Users\kopry\Desktop\Applied new folder v2
plugins: anyio-2.2.0
collected 8 items

my_test.py .....

===== warnings summary =====
..\AppData\Roaming\Python\Python38\site-packages\requests_toolbelt\compat.py:11
..\AppData\Roaming\Python\Python38\site-packages\requests_toolbelt\compat.py:11
C:\Users\kopry\AppData\Roaming\Python\Python38\site-packages\requests_toolbelt\compat.py:11: Deprecatio
nWarning: Using or importing the ABCs from 'collections' instead of from 'collections.abc' is deprecated s
ince Python 3.3, and in 3.9 it will stop working
  from collections import Mapping, MutableMapping

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 8 passed, 2 warnings in 9.85s =====

```

Figure 5.1: Command-Line test using PyTest testing framework.

### 5.2.1 Application Functionality Testing

I have carried out some tests to check if the application is doing what it suppose to. The set of tests includes Graphical User Interface and the main functions such as checking the webcam is correctly running.

- **Test One** – Checks whether or not the video stream is valid.

```

1 def test_one():
2     cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)
3     assert(cap.isOpened())

```

- **Test Two** – This test is carried out to validate if the image gets stored in the specified file. With the use of `imwrite()`, the image gets saved to the specified file.

```

1 def test_two():
2     cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
3
4     img_counter = 0
5
6     while True:
7         # Capture frame-by-frame
8         ret, frame = cap.read()
9         if not ret:
10            print("failed")
11            break
12        cv2.imshow("Registration", frame)
13
14        k = cv2.waitKey(1)
15        # click 'esc' to close the program
16        if k % 256 == 27:
17            print("Program closing..")
18            # closes the webcam
19            cap.release()
20            # destroys all the windows we created
21            cv2.destroyAllWindows()
22        # click 'space' to save the image
23        elif k % 256 == 32:
24            # saves users name as a image
25            img_name = "images/" + "testname" + ".jpg".
26            format(img_counter)
27            assert(cv2.imwrite(img_name, frame))
28            img_counter += 1
29            # closes the webcam
30            cap.release()
31            # destroys all the windows we created
32            cv2.destroyAllWindows()

```

- **Test Three** – Deletes the image from the “images” folder and then checks whether the folder is empty or not. If the folder is empty it will then display an error message otherwise it will do nothing.

```

1 def test_three():
2     # deletes the image of the folder
3     path = "images/" + "testname" + ".jpg"
4     os.remove(path)
5
6     if len(os.listdir('images')) == 0:
7         messagebox.showerror("Alert","There are no
8         students registred!\nRegister first!")

```

```

8         assert(True)
9     else:
10        assert(False)

```

- **Test Four** – This test performs validation on login authentication for the admin panel. It checks that the entered value for the pin matches the specified pin. If the pin is correct, it will display an alert with successful authentication otherwise it will display an error message.

```

1 def test_four():
2
3     def password():
4         if adminPass.get() == "1972":
5             messagebox.showinfo("Alert","ACCESS GRANTED!")
6         )
7         assert(True)
8         window.destroy()
9     else:
10        messagebox.showerror("Alert","WRONG PASSWORD!")
11    ")
12        window.destroy()
13        assert(False)
14
15    window = tk.Tk()
16    window.geometry('600x400+50+50')
17    #adminPass = '1972'
18    adminPass = tk.StringVar()
19
20    lbl_name = Label(window, text="Pin",fg="black",font=(
21    'Helvetica', 20, ' bold '))
22    lbl_name.place(x=50, y=40)
23
24    txt_name=Entry(window, textvariable=adminPass, font=(
25    'Helvetica', 15, ' bold '),bd=5,relief=GROOVE)
26    txt_name.place(x=50, y=80)
27    # submit form button
28    submit_button = Button(window, text = "Login",
29    command = password, bg = "green", font=('Helvetica',
30    12))
31    submit_button.place(x=50, y=150)
32    window.mainloop()

```

### 5.2.2 Database Testing

To check that the database is fully working I have implemented some tests to validate that. Tests include reads, writes, deletes, and document count. This set of tests will verify if the database does what it is supposed to do.

- **Test Five** – This test performs a write operation on a random collection. A random 12 names are inserted into the database to check if the operation is successful. This test plays a role in the next test to perform a read.

```
1 def test_five():
2     cluster = MongoClient("connection string here")
3     db = cluster["Random"]
4     collection = db["tests"]
5
6     mylist = [
7         { "name": "Ann", "Confirmation": "yes"},
8         { "name": "Paddy", "Confirmation": "yes"},
9         { "name": "Michael", "Confirmation": "no"},
10        { "name": "Peter", "Confirmation": "yes"},
11        { "name": "Bethany", "Confirmation": "no"},
12        { "name": "Sara", "Confirmation": "no"},
13        { "name": "Susan", "Confirmation": "no"},
14        { "name": "Hannah", "Confirmation": "yes"},
15        { "name": "Ben", "Confirmation": "no"},
16        { "name": "Milly", "Confirmation": "yes"},
17        { "name": "Ben", "Confirmation": "yes"},
18        { "name": "Ella", "Confirmation": "yes"}
19    ]
20
21    data = collection.insert_many(mylist)
22    assert(data)
```

- **Test Six** – This test uses the data which was inserted in a previous test, Test Five. It performs a read operation to find one name in the database. If the name is correct, it will return true status otherwise it returns a false status.

```
1 def test_six():
2     cluster = MongoClient("connection string here")
3     db = cluster["Random"]
4     collection = db["tests"]
5
6     get = {"name": "Ben"}
7     collection.find_one(get)
8
9     if get == get:
10        assert(True)
11    else:
12        assert(False)
```

- **Test Seven** – Performs a delete operation, and deletes selected names from the database. To validate that the operation is successful, it

checks the document count. In this case, there are 12 names inserted in the database, so the correct document count should be 11.

```
1 def test_seven():
2     cluster = MongoClient("connection string here")
3     db = cluster["Random"]
4     collection = db["tests"]
5
6     delete = {"name": "Bethany"}
7     collection.delete_one(delete)
8
9     # number of documents in the collection
10    checkDoc = collection.count_documents({})
11    if checkDoc == 11:
12        assert(True)
13    else:
14        assert(False)
```

- **Test Eight** – This test checks the document count with a specified field. It checks how many “yes” fields exist in the collection. If the document count is equal to the correct amount of “yes” fields it will return true status, otherwise, it will return false status.

```
1 def test_eighth():
2     cluster = MongoClient("connection string here")
3     db = cluster["Random"]
4     collection = db["tests"]
5
6     docCount_yes = collection.count_documents({'
Confirmation': "yes"})
7     if docCount_yes == 7:
8         assert(True)
9     else:
10        assert(False)
```

- **Test Nine** – This test checks the document count with a specified field. It checks how many “no” fields exist in the collection. If the document count is equal to the correct amount of “no” fields it will return true status, otherwise, it will return false status.

```
1 def test_nine():
2     cluster = MongoClient("connection string here")
3     db = cluster["Random"]
4     collection = db["tests"]
5
6     docCount_no = collection.count_documents({'
Confirmation': "no"})
```

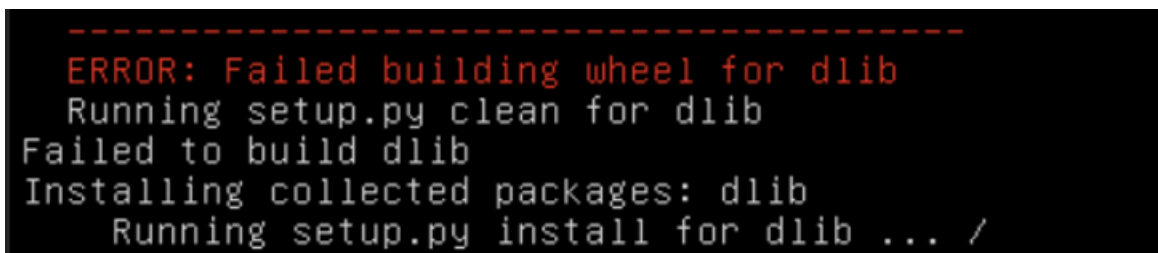
```
7     if docCount_no == 4:  
8         assert(True)  
9     else:  
10        assert(False)
```

## 5.3 Limitations

During the development of this project, I ran into some issues which were a challenge to resolve. I have used GitHub issues to record all the issues I have faced during the development. Overall the software doesn't contain any bugs.

### 5.3.1 Failing to install dlib

Just at the beginning of my project, I ran into an issue with installing Dlib library on my machine. In order to install `face_recognition` library, Dlib was required to be installed. To recognize and manipulate faces in Python the installation of `face_recognition` library was necessary. After doing some research online, I found a GitHub repository that contained few versions of Dlib. After trying each version, I have discovered that the version `dlib-19.19.0` has successfully installed on my machine.



```
-----  
ERROR: Failed building wheel for dlib  
Running setup.py clean for dlib  
Failed to build dlib  
Installing collected packages: dlib  
Running setup.py install for dlib ... /
```

Figure 5.2: Error message when trying to install Dlib library.

### 5.3.2 User input for saving images

Another issue that I ran into was when I tried to implement a function that will take user input from an input box during the registration process, and store it as an image in an images folder with the name of the user's input, which represents the student's name. I was capable of storing the variable using Tkinter `StringVar()` but since I'm new to Python and Tkinter, I wasn't sure how to get the value that I stored when passing in the variable.

After some research, I have found that I can get the value of the variable with the use of the `get()` method.

### 5.3.3 Saving student names to the database

My other issue was when I tried to save students' names as an image in the database. The program was only capable of storing hard-coded student names, which meant I could only store one student. After hours of looking online, I couldn't find any solution for storing multiple students in the database. Finally, after many tries, I have figured out where I was going wrong. This was due to the fact that rather than passing the name variable from the main file to the database file, I wasn't passing in any variable that allowed me to use the name variable. After passing in the name variable, I had access to use the value of the name variable which I was passing in during the registration process.

### 5.3.4 Send an email after completing a health check form

I had an issue when I wanted to send an email after completing the health check form. After implementing the email notification system, the program wasn't capable of sending an email. When I tried to send an email from the main file, I was getting an error message, as per figure 3.2. It was missing 3 arguments: 'to', 'subject' and 'body'. After searching online, I found a post on Stack Overflow which helped me to figure out the issue. Rather than calling the function for sending an email, I was calling the function that set's up the email notification system.

```
C:\Users\kopry\Applied-Project-and-Minor-Dissertation (main -> origin)
λ python main.py
Saving records...
Exception in Tkinter callback
Traceback (most recent call last):
  File "C:\Program Files\Python38\lib\tkinter\__init__.py", line 1892, in __call__
    return self.func(*args)
  File "main.py", line 293, in action
    sendEmail()
  File "main.py", line 33, in sendEmail
    emailNotification.email_notification()
TypeError: email_notification() missing 3 required positional arguments: 'to', 'subject', and 'body'
```

Figure 5.3: Error message when trying to send an email.

### 5.3.5 Time frozen when program running

Another issue I had was with the displaying of time in the main menu of the GUI. The time stopped as soon as the program was launched. It appears that something was causing the time to stop. Anytime I restarted the program, the time was updated but the time wasn't running. I've noticed that this issue appeared when I created a checked-in menu which was also displaying time. I did some testing to figure out what was wrong. Firstly I checked the checked-in menu as it also was displaying time. The time was working fine in the checked-in menu which left me confused. I couldn't find what was causing the time to stop until I went through the entire code and noticed that it was using the same function where the time was implemented. After removing the time from the checked-in menu, the time started to work fine in the main menu.

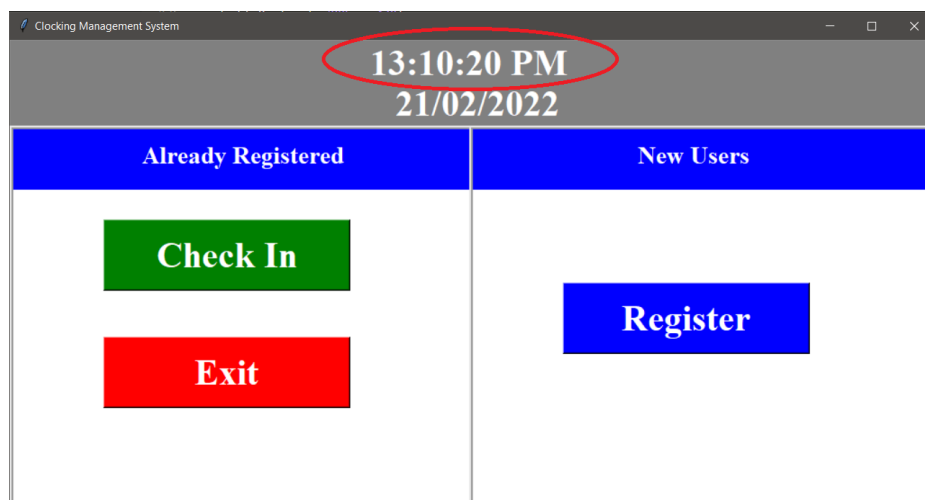


Figure 5.4: Time stops after the program get's launched.

### 5.3.6 Retrieving names from the database during checking-in

This issue has not been fully solved yet. There are a few issues that are stopping me to get this function completely working. Similar to storing the names in the database, the retrieving function uses hard-coded values. The first solution I came up with wasn't successful. When storing names in the database, I added functionality so that it also retrieves the image at the same time. To show that the name is actually being saved and retrieved, I removed



it from the images folder just before it was retrieved. Then when the student checked in, the image was retrieved and also removed from the download folder, leaving it empty. This was originally designed so the program is only uses the database to store the students' names (images).

This is when another issue was discovered. When an already registered student wanted to check-in, the program searches for an image in the download folder. Since the image was removed, the program is not capable of checking in the student.

The only way I make this work is by only removing the student name (image) in the images folder and keeping it in the download folder for future check-ins. This is not the best approach but it's capable of retrieving the names from the database

# Chapter 6

## Conclusion

Students are filling out health check forms online, some of them forget and more than a half don't even know it exists. Developing a piece of software that can do it instead in a smarter and faster way would potentially reduce the time for filling out the forms online. Having the kiosk which is an idea of standing stations at every door of ATU would increase the number of completed forms as students would be required to fill them out on the way in. This would have multiple benefits, ensuring that everyone arriving on the campus confirm their good health and preventing the spread of COVID-19.

The application uses face recognition as part of the authentication process which is becoming very popular in recent years. Face recognition is far from being perfect but with some improvements, it can succeed more. Face recognition uses biological characteristics, for example, eye color. These characteristics can change over time which could lower the accuracy of face recognition. One way of going around this problem is to include yearly updates of the image stored on the database.

### 6.1 Findings and Insights

Throughout the development of the project, there were a number of challenges when implementing new technologies. Just at the beginning of the development, there were problems installing dlib which was required for the `face_recognition` library. This issue has held me back in the development process of the application as without dlib, It was not possible to install the `face_recognition` library. Then I found that you are required to have specific python and dlib versions in order to install dlib. After many tries, **python version 3.8.8** and **dlib version 19.19.0** successfully installed dlib on my machine.

Another major issue was with retrieving the images from the database. When the student completes the registration for the first time, the image gets stored in the images folder so it can be stored in the database. Due to using `GridFS()`, a destination folder must be selected for the image to be saved and retrieved. Once the image is successfully saved on the database, it removes that image from the images folder. This is done to maintain the local machine less cluttered and to show that the image gets actually saved on the database. The same process was planned when retrieving the image from the database. Unfortunately, If the retrieved image gets removed from the downloads folder, the program will not be able to find any images. This is not the best approach but the application is fully functional and performs as planned.

During the implementation of the project, I gained great knowledge and experience working with some technologies which I have never used before. I have previously written Python code but I was not impressed with my programming skills and wanted to challenge myself and develop a python application with face recognition for the authentication process. Also, I have learned more about technologies that I have previously used like MongoDB. Throughout the project, I have learned that there is much more involved when developing any application such as the tests to check if the code and application do what they are supposed to. I believe that these findings and insights will benefit me in my future career.

## 6.2 Future work

- **Run the application on the cloud** – Currently, the application is not running on the cloud as the GUI Tkinter is a desktop GUI toolkit that does support deployments on the cloud. However, there is a way of making Tkinter run on the cloud but It involves some changes on the cloud. Having the application running on the cloud would have pre-installed dependencies which would not require the end-user to install each dependency one by one. Due to the fact that dlib is challenging to install, it would make end-users life easier.
- **Run application on android** – My initial goal was to have the application running on android using tools such as Termux or QPython. My application cannot run on android as I'm using Tkinter as my graphical user interface which won't work for creating mobile apps as it is meant

to be used on desktop only.

- **Fingerprint as a second recognition system** – The application is using face recognition as the authentication process for recognizing students during the checking-in process. Some students may not agree with capturing and storing their images due to their privacy, so a second recognition system would be an ideal solution for this. This would also enhance the security of the application and provide a choice of authentication system for the students.
- **Use Sqlite3** – As mentioned in the above chapter, there were some limitations with retrieving the images from the database. MongoDB provides upload of images up to 16 Mb, any larger documents would have to be stored using GridFS. With GridFS you need to specify the destination folder when storing and retrieving images. There were some issues during the checking-in process so I decided to leave the image in the download folder after being retrieved from the database. This wasn't the best approach for retrieving the images but the application performs as it is supposed to. For future development, I would consider using Sqlite3 which allows storing and retrieving images directly from the database.

## 6.3 Final thoughts

The development of this project had ups and downs, but I can happily say that all objectives have been completed and the application is fully functional. It has been a massive learning experience since day one and I truly enjoyed working on this project. Working with already familiar technologies like MongoDB I was able to create a connection between my application and the database. I have also improved my knowledge by using new technologies like Python, OpenCV, and `face_recognition` library. Working on my own I become very organized as it was depended on me how well I will reach in developing this project. I also gained great communications skills from the weekly meetings with my supervisor. Throughout the research, I have learned a lot about the technologies I used and many other interesting technologies. The entire project has been a great experience and I truly enjoyed every minute of the project. I believe I have met all the requirements and completed all the objectives to achieve the initial goal.

# Chapter 7

## Appendices

### 7.1 GitHub Repository

Throughout the development of my project I used GitHub to commit and make changes to my project.

**GitHub Link:** <https://github.com/krystianopryszek99/Applied-Project-and-Minor-Dissertation>

# Bibliography