

|  |
| --- |
| Project F.R.A.M.E  2019-2020 |
|  |
| April 4  Project F.R.A.M.E  Authored by: Samuel Tredgett, James Clark, Hugo A’Violet |

A picture containing stereo

Description automatically generated

|  |
| --- |
| Abstract F.R.A.M.E in a nutshell Project F.R.A.M.E (Facial Recognition Attendance Monitoring Engine) is a lightweight program intended for use by the lecturers at the University of Kent, with potential expansion for use in other universities.  F.R.A.M.E offers several core features including inbuilt scheduling, fully functional facial recognition software, back-end database storage and direct data relay to lecturers’ emails. |

1. Introduction

F.R.A.M.E is a general purpose attendance recording software with the intended goal of taking work out of the hands of lecturers whilst simultaneous streamlining and improving the accuracy of recording which students are in attendance of any given event. Our objective was to create a system that could be easily integrated with minimal interruptions to university workflow, allowing lecturers to maximise the short amount of time they have for face-to-face student interactions.

* 1. Background

Before we began working on project F.R.A.M.E, we had several skills to learn and several materials to learn how to use. This was a crucial stage in the process as all members of the group were required to learn a varied set of complex skills in a relatively short space of time, making the task difficult.

* 1. A Critique of Attendance Monitoring

The current attendance monitoring system in place at the University of Kent falls short in several areas. Namely, convenience, simplicity, reliability and accuracy. Having to hand pieces of paper out that require signatures every lecture, to often in excess of a hundred plus students, is not only an inconvenient for lecturers and students alike, it is also wasteful in times where being paperless is becoming more expected of institutions.   
  
It also by nature facilitates the ability for students to sign their friends into lectures when some are still at home sleeping off the night before. This causes inaccuracies in the attendance data, and should a lecturer or student lose a piece of paper then this problem is only exacerbated.

* 1. Python   
     1.3.1 Python API  
     As a group we decided on Python as our programming language for several reasons. Python can be a lightweight tool to design simple prototypes quickly. We used this to our advantage in early stages of design to see what possibilities and limitations we would run into during development. An example of this was deciding what facial recognition system was best to reach our goal.   
       
     1.3.2 Facial Recognition Libraries  
     As a group we decided on Python as our programming language for several reasons. Python can be a lightweight tool to design simple prototypes quickly. We used this to our advantage in early stages of design to see what possibilities and limitations we would run into during development. An example of this was deciding what facial recognition system was best to reach our goal.   
       
     We originally planned to create a machine learning algorithm that would learn overtime the faces of students to gain a better accuracy score. However, if we would have taken this method, we would have to have a large dataset of existing students to begin testing. A learning algorithm would also take a long time to gain precision. Therefore, instead of this we concluded that the face-recognition python library was more suitable when iterating over faces who attended a specific module.   
       
     The advantage this library had was it handled the face embedding and face matching in one sequence, speeding up the time it would have to recognize a specific face. We also wouldn’t have to update the entire face database if a new user was added. When programming the facial recognition function, we originally had the database set up where all the students were in one folder. This iteration method would be relatively slow and unnecessary when certain students weren’t attending the module. Therefore, this was remedied so when a class started the facial recognition database would point to the specific faces in that module.  
       
     MatPlotLib was the data analytics python library we used to provide visual representation to our gathered data. This library's focus was to create visually interesting and useful graphs for lecturers and schools to use. Difficulties faced with using this library was understanding how data had to be inputted into the system. An example we faced with this was making sure the X and Y axis were showing the correct data. We had to extrapolate our list of lists data generated by our database into two separate lists, then input these into our graph separately.
  2. MySQL Development with Python  
     MySQL was our database of choice for several reasons. Firstly, mySQL benefits from scalability, therefore with a system that’s designed to create new data every time a class has finished, it’s essential to use such a framework. Using mySQL with Python was difficult and had its fair share of bugs. Executing SQL statements within Python requires the “mysqlConnection” library. It also requires every time a SQL statement is called to create a connection to the server. Originally, all the SQL connections shared a common variable name, however once we had to run parallel connections, we needed to separate these out so our program wouldn’t be slow when interacting with our server. The former issue we had was our class timer being too slow, due to the amount of linear SQL queries. This was fixed with parallelism.
  3. Amazon Web Services  
     Amazon Web Services was the platform we chose to host our mySQL database on. The reasons for this was because it was important to always have uptime and for multiple people to be working on the system at once. Having many connections connected to the database on a local machine would be taxing and inefficient. AWS provides a free plan that is as competitive as other paid plans.
  4. User Interface (TKINTER)  
     Tkinter was another python library that we utilised. It’s important that the application is as user friendly and interactive as possible. Tkinter was chosen because of the vast amounts of existing documentation available, along with its lightweight usability and customization. During development, understanding Tkinter and how we can customize our GUI to provide a minimal and professional experience to our user’s was important. Originally, certain key components such as the ‘Select Room’ menu option appeared as a text box that Staff would have to manually type in. This type of interactivity wouldn’t work when the system has been designed to be a ‘touch-screen’ experience. Therefore, we had to develop the system to incorporate a touchscreen keyboard. This was done by creating a variety of buttons that when pressed would update or clear the room number. Understanding Tkinter and its relationship with Windows, also gave us the ability to create a Student Mode, which was essential to incorporate.

1. Requirements
   1. User Stories   
      ¾ basic user stories
   2. Non-functional Requirements  
      Creating a system that was lightweight and easy to use was one of our main goals for the deliverable software we created. We’ve set ourselves three main non-functional requirements, ease of use, speed of use, and reliability.  
       Ease of use –   
       Speed of use -   
       Reliability -
2. Development Stages
   1. Prototyping  
        
      4.1.1. summary  
      4.1.2 features  
      4.1.3 challenges  
      4.1.4 results
   2. Delivered Project  
        
        
      4.2.1. summary  
      4.2.2. features  
      4.2.3 challenges  
      4.2.4 results
   3. Potential Future options

4.3.1 summary

4.3.2. features  
4.3.3 challenges  
4.3.4 results

1. Waterfall Development

Explain the general structure of the project development stages.

1. Quality Assurance  
   Elaborate on quality control measures i.e. testing etc.  
   mention M.I.T face database for testing on other peoples faces (using online resources available for testing facial recognition)

Bibliography –

Python documentation

Library documentation

Facial recognition (relevant paper)

Software design

Attendance monitoring