Attendance Model Process

Aagaman Sharma Pokharel, Ashim Nepal

*Department of Electronics and Computer Engineering, IOE Pulchowk*

[aagaman12658@gmail.com](mailto:aagaman12658@gmail.com)

[ashimnepal636@gmail.com](mailto:aagaman12658@gmail.com)

***Abstract*— Organize-IT is working for the ‘Regular Attendance’ application based on facial recognition that digitalize the attendance procedure for any organization. The product built is focused on proper management of attendees’ data in an educational or any form of organization and maintains regular user statistics. The Dashboard is created which keeps all the related information, the product is used for.**

***Keywords*— Facial Recognition, Digitalized Attendance System, Easy Management of Data, Regular User Statistics.**

1. INTRODUCTION

Organize-IT is a minimal viable product which records the daily attendance of students and working staffs.

The software is mainly designed to cater teachers, school, campus and organizations. The software is capable of verifying the target’s facial characteristics and record attendance based on that facial verification. It consists of a dashboard where user i.e. authorized body will be able to navigate through different classroom and store the attendance statistics of the students.

The user is also provided with the facility to create new classroom.

1. OVERVIEW

Facial recognition uses Face++ API’s transformation which takes images entry, compare it with reference image and perform attendance analyzing that the images matches or not on the system.

Initially for registration of user data, various details are asked from the user and authentic email’s entry are only made valid, & for sign-up the system redirects to a new page, accepts entry and later allows log-in from the user. The development of the product was all possible due to the uses of GUI, Firebase API by Google, Jansson Library, Curl API documentation, Gtk-4 libraries and short & quick response of queries from Chat GPT.

The libraries/API’s used are:

Curl Library Face++ API

GTK-4 Library Jansson Library

Firebase API Abstract API

.

Fig. 1 Example of a Homepage

The software uses GTK library for its GUI implementation. Similarly, CURL library for making URL requests. The API’s called by using CURL library making URL requests, in return, provide us with the json containing the proper information. The json provided by the API is parsed using Jansson library. Similarly, the software uses Firebase API for authentication purpose and for storing data using its real-time database feature.

Finally, the main function i.e., the facial recognition is carried out by using detect and compare feature of Face++ API.

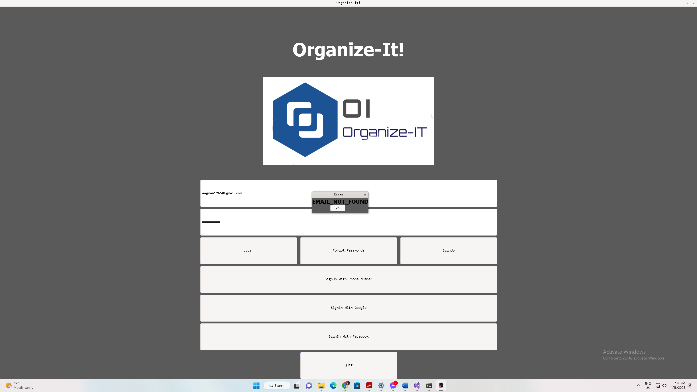


Fig. 2 Example of a Login page with respective error messages for invalid input.

1. FEATURES
2. **FACIAL RECOGNITION:** Product uses Facial recognition system that assists regular attendance based on Face++ API.
3. **DIGITALIZED ATTENDANCE SYSTEM**: Product digitalizes the attendance system for an organization creating user profiles for students & staff.
4. **EASY MANAGEMENT OF DATA:** The development of the product was focused for proper clustering and management of data of an organization.
5. **REGULAR USER STATISTICS:** The statistics of students and working staffs are recorded respectively in the Dashboard.
6. LIBRARIES
7. *Curl Library*

Curl is a command-line tool and library for transferring data with various protocols, including HTTP, FTP, and SMTP. It supports many options for authentication, proxy settings, and other features, and can be used for a wide range of tasks such as sending and receiving data from web servers, downloading files, and interacting with APIs. It is widely used in scripting and automation tasks and is available for many operating systems, including Windows, Linux, and macOS.

Here in this program, we have used curl to interact with different APIs.

1. GTK Library

GTK (GIMP Toolkit) is a cross-platform GUI (graphical user interface) toolkit that is primarily used to develop desktop applications on Linux, Unix, and Windows. It is written in C and provides a set of widgets, such as buttons, text fields, and menus, that can be used to build graphical user interfaces.

GTK is built on top of the GDK (GIMP Drawing Kit) library, which provides low-level functionality for handling graphics and user input. It also includes GLib, a utility library that provides a number of useful functions for working with data structures, strings, and other common programming tasks.

GTK is known for its clean and modern look and feel, and it is also highly customizable. It is used by many popular open-source desktop applications like GIMP, Inkscape, and Gnumeric. GTK is also well-documented and has a large, active community of developers contributing to it.

It also has some of the latest version called GTK3 and GTK4 which are more advanced and feature rich.

Here in this program, we have used GTK4 which is also the latest version of GTK

1. Jansson Library

Jansson is a C library for encoding, decoding and manipulating JSON (JavaScript Object Notation) data. It provides a simple and efficient API for handling JSON data in C, and is often used in applications that need to interact with JSON-based web services or other systems that use JSON as a data format.

The library provides functions for parsing JSON data from strings or files, creating JSON data from C data structures, and manipulating

JSON data in memory. It also provides support for working with JSON data in a type-safe way, with functions for converting between JSON data and C data types such as integers, strings, and arrays.

The jansson.h is the main header file of the Jansson library, it contains all the function and data structures definition which are needed to use the functionality of Jansson. It also includes other header files like jansson\_config.h, jansson\_private.h, jansson\_version.h.

It is lightweight and easy to use library which is widely used in many projects and is actively maintained.

Here we have used jansson library to parse json sent as a response by api.

1. APIs
2. *Firebase API*

Firebase also provides a REST (Representational State Transfer) API which can be used to access the functionality of Firebase from any language that can make HTTP requests and parse JSON data. The Firebase REST API allows developers to perform CRUD (Create, Read, Update, and Delete) operations on data stored in Firebase, as well as perform authentication and other administrative tasks.

The Firebase REST API is organized around the following main concepts:

Endpoints: Each Firebase service has a set of endpoints that can be used to access its functionality. For example, the Realtime Database has endpoints for creating and querying data, and the Authentication service has endpoints for creating and managing user accounts.

Request and response formats: The Firebase REST API uses standard HTTP methods such as GET, POST, PUT, and DELETE to interact with the service, and the data is exchanged in JSON format.

Authentication: The Firebase REST API requires authentication for all requests. Firebase uses JSON Web Tokens (JWT) to authenticate requests. The token is passed as a parameter in the request header, which is verified by the Firebase servers before processing the request.

Here Firebase API is used for authentication and saving data in real-time database provided by firebase API.

1. *Abstract API*

The Abstract email verification API takes an email address and identifies whether it is valid or not, and how risky we think it will be. This real-time or asynchronous capability will help you detect and suppress any invalid or disposable email addresses, which will help clean your email list, reduce your bounce rate, and thus improve your email delivery rate for legitimate users. Email verification is especially powerful when combined with tools like the IP geolocation API and / or the phone number validation API to provide simple checks for new users.

Here Abstract API is used to check whether email exists or not.

1. *Face++ API*

Face⁺⁺ AI Open Platform is a platform offering computer vision technologies that enable your applications to read and understand the world better. Face⁺⁺ allows you to easily add leading, deep learning-based image analysis recognition technologies into your applications, with simple and powerful APIs and SDKs.

Here API provided by face++ is used to verify whether the images uploaded match with each other or not.

1. PROCEDURE

The procedure involved in the development of this program are:

1. *Gathering the requirements.*

During the intial phase of the development of our program, we scoured over the internet to know the possible ways of developing a facial recognition based attendance software using c. During this process, we got to learn about different APIs which we later on incorporated in our program, linked multiple APIs together and create a functioning program out of it. The requesities we gathered during this phase include GTK library for GUI implementation, CURL library for making https requests, FIREBASE api for authentication and database, jansson library for parsing the json response, FACE++ api for facial recognition and abstract api for checking the validity of the emails.

2. *Designing Phase*

During this phase of the program we decided to design the UI of our program. In this phase we decided the number of windows we were going to use and the number of features we were going to incorporate in the initial version of our program. The features that were left for the future versions of our program were designed to display feature under development window whenever user tried to access them.

3. *Writing the source code*

In this phase of the development of our program, we wrote the source code of our program based on the previously created designs. Similarly we incorporated multiple libraries to enable multiple functionalities.

4. *Testing and Debugging*

While testing and debugging our program we used approaches like unit testing for testing program and different other approaches for debbuing some of which are listed below:

a. **Print statements**: One of the simplest and oldest debugging techniques is to insert print statements into the code to output the values of variables or the

flow of control. This can be useful for identifying the location of an error and understanding the state of the program at a particular point in time.

b. **Pair Programming**: Pair programming is a technique in which two programmers work together on the same code, one acting as the driver, writing the code, and the other acting as the navigator, reviewing the code and making suggestions. This approach is quite effective in finding bugs and improving the code quality.

c. **Debugging tools**: There are also many debugging tools available that can help identify and diagnose errors in a program. Some popular debugging tools include GDB, LLDB, and the debuggers built into IDEs such as Visual Studio and Xcode. These tools allow developers to step through the code line by line, set breakpoints, inspect variables, and view the call stack.

5. *Document the program*

During the documentation phase of the program, we included comments above the functions of the program source code and mentioned all the functions used in different files of the source code in our report

1. FILES
2. *.C Files*



1. *gtk\_gui\_main.c*

*The functions present in it are:*

1. *static void activate(GtkApplication\* app, gpointer user\_data);*

Function = creates login window(this function creates the basic appearance and functionlaities of appearance window)

1. *int css\_for\_window(GtkWidget\* window);*

Function = sets appearance properties for the text labels present in title bar and sets appearance properties for window

1. *int css\_for\_buttons(GtkWidget\* button);*

Function= sets appearance properties for the text labels present in buttons

1. *static void activate\_entry\_for\_password();*

Function = activates entry widget for taking password as input.

1. *static void activate\_entry\_for\_email();*

Function = activates entry widget for taking email as input.

1. *void change\_focus\_func(GtkWidget\* Widget1, GtkWidget\* Widget2);*

Function = changes focus from widget1 to widget2 when user presses enter

1. *char\* return\_email();*

Function = shares email of the user when called from any other files or functions

1. *char\* return\_password();*

Function = shares password of the user when called upon from any other files or functions and this variable gets destroyed during runtime so there is no risk of exposing password

1. *void invalid\_input();*

Function = displays invalid input prompt when user enters any invalid input.

1. *int css\_for\_label(GtkWidget\* label);*

Function = sets appearance properties for the text labels present inside window

1. *static void activate\_entry\_for\_name();*

Function = activates entry widget for taking name as input

1. *static void activate\_entry\_for\_password\_from\_signup();*

Function = activates entry widget for taking password as input. It seems similar to “activate\_entry\_for\_password()”, but this function is called from signup window in order to check whether the inputs in password widget and confirm password widget is same*.*

1. *static void activate\_entry\_for\_email\_from\_signup();*

Function = same as “activate\_entry\_for\_email()” but is called from signup window

1. *void clear\_entry();*

Function = clears entry in login window when user logs in to dashboard*.*

1. *void clear\_entry\_for\_signupWindow();*

Function = clears entries in signup window

1. *char\* return\_name();*

Function = shares name with any other files trying to access it.

1. *int return\_valueresult();*

Function = shares valueresult with any other file trying to access it. Here valueresult carries a value which determines whether the email is valid or not

1. *void under\_development();*

Function = displays under development feature for the features that are still under construction.

1. *int check\_email\_validity();*

Function = checks for the validity of email using “abstract” api.

1. *void signup\_normal\_window();*

Function = create signup window which accepts entries for name, email, password and confirm password.

1. *gtk\_dashboard\_window:*

*Function = creates new dashboard window*

*The functions present in it are:*

1. *void activate\_dashboard\_window(char\* email\_sent\_by\_main\_file);*
2. *void activate\_profile\_window();*
3. *static void activate\_attendance\_window();*
4. *void open\_attendance\_window();*
5. *void activate\_query\_window();*
6. *static void action\_clbk\_for\_txt\_file(GSimpleAction\* simple\_action, G\_GNUC\_UNUSED GVariant\* parameter, gpointer\* data);*
7. *void on\_response\_for\_txt\_file(GtkNativeDialog\* native, int response);*
8. *void open\_parent3();*
9. *void open\_parent2();*
10. *void open\_parent1();*
11. *void on\_clicking\_ok\_for\_create\_classroom\_query();*
12. *void save\_basic\_details();*
13. *void on\_clicking\_create\_classroom();*
14. *void open\_actual\_attendance\_window(GtkWidget\* button);*
15. *int change\_uri\_to\_localpath\_for\_txt\_file(gchar\* file\_uri);*
16. *void add\_metadata\_to\_main\_json(json\_t\* root);*
17. *static void activate\_entry\_for\_no\_of\_students();*
18. *int convert\_to\_int();*
19. *static void activate\_entry\_for\_classname();*
20. *loginwithgoogle.c*

*Function = provides log in facility where user can login using the google account already signed in in their browser.*

*The functions present in it are:*

1. *int login\_with\_google();*
2. *gboolean start\_login();*
3. *int extract\_and\_print\_code();*
4. *void print\_code();*
5. *int request\_for\_access\_token(char\* code);*
6. *int change\_id\_token\_with\_info(char\* idtoken);*
7. *static size\_t write\_data(void\* prt, size\_t size, size\_t nmemb, void\* stream);*
8. *void waiting\_window\_for\_login\_with\_google();*
9. *void change\_value\_for\_choice();*
10. *login\_normal.c*

Function = logs in user through normal process where user enters their email and password and gets authorized through that process.

The functions used in it are:

1. Int login\_normal();
2. *static size\_t write\_data(void\* prt, size\_t size, size\_t nmemb, void\* stream);*
3. *char\* return\_displayName();*
4. *SIGNUP\_Normal.c*

Function = Signs up new user.

The functions used in it are:

1. int signup\_normal();
2. *static size\_t write\_data(void\* prt, size\_t size, size\_t nmemb, void\* stream);*
3. *ErrorMessages.c*

Function =Hosts all the prompts for the error messages.

The functions present in it are:

1. void passwords\_do\_not\_match();
2. void signup\_successful();
3. void invalid\_email\_id();
4. void error\_message\_sent\_by\_api(char\* message);
5. void message\_sent\_to\_print(char\* message)
6. *attendance\_window\_public.c*

Function = starts window for taking attendance using facial recognition, updates the database with user entered student names.

1. int change\_uri\_to\_localpath\_for\_present\_photo(gchar\* file\_uri);
2. int change\_uri\_to\_localpath\_for\_reference\_photo(gchar\* file\_uri);
3. static size\_t write\_data(void\* ptr, size\_t size, size\_t nmemb, void\* stream);
4. void facetoken\_present\_image();
5. void facetoken\_ref\_image();
6. float attendance();
7. void remove\_present\_photo();
8. void remove\_referernce\_photo();
9. void remove\_both\_photos();
10. void on\_response\_for\_present\_photo(GtkNativeDialog\* native, int response);
11. Void on\_response\_for\_reference\_photo(GtkNativeDialog\* native, int response);
12. void on\_response\_for\_reference\_photo\_all\_at\_once(GtkNativeDialog\* native, int response);
13. int compare\_students(const void\* a, const void\* b);
14. static void action\_clbk\_for\_present\_photo(GSimpleAction\* simple\_action, G\_GNUC\_UNUSED GVariant\* parameter, gpointer\* data);
15. static void action\_clbk\_for\_reference\_photo(GSimpleAction\* simple\_action, G\_GNUC\_UNUSED GVariant\* parameter, gpointer\* data);
16. void action\_clbk\_for\_reference\_photo\_all\_at\_once(GSimpleAction\* simple\_action, G\_GNUC\_UNUSED GVariant\* parameter, gpointer\* data);
17. char\* sort\_names\_and\_send\_to\_db(char\* filepath\_for\_txt\_containing\_name,char\* email,char\* classname, int number\_of\_students);
18. void activate\_attendance\_window\_for\_taking\_attendance\_demo(char\* email);
19. void update\_db\_with\_updated\_data();
20. void increase\_class\_day\_by\_1();
21. void decrease\_class\_day\_by\_1();
22. int\* make\_all\_values\_minus\_one();
23. void activate\_attendance\_window\_for\_taking\_attendance(char\* email,char\*\* unique\_id, char\* classname, int number\_of\_classroom\_created\_by\_user);
24. *post\_\_basic\_file\_to\_database.c*
25. *int update\_resource\_file\_in\_database(char\* json\_data, char\* email, char\* unique\_identifier\_for\_each\_class);*
26. *char\* post\_resource\_file\_to\_database(char\* json\_data, char\* email);*
27. *char\* post\_resource\_file\_to\_database(char\* json\_data, char\* email);*
28. *char\* get\_classname\_only\_from\_resource\_json(char\* email, char \*unique\_id);*
29. *char\* get\_names\_present\_in\_resource\_json\_in\_database(char\* email, char\*\* unique\_id, char\* classname, int number\_of\_classroom\_created\_by\_user);*
30. *int get\_class\_days\_from\_each\_classroom(char\* email,char\* unique\_id\_for\_classroom);*
31. *int\* get\_present\_days\_for\_each\_classroom(char\* email, char\* unique\_id\_for\_classroom,int number\_of\_students);*
32. *int get\_number\_of\_studetns\_in\_classroom\_selected(char\*email,char\* unique\_id\_for\_classroom);*
33. *char\*\* get\_data\_present\_in\_main\_json\_in\_database(char\* email, int\* number\_of\_classes\_created\_by\_user, int\* class\_status);*
34. *void delete\_data\_previously\_present\_in\_main\_json\_in\_database(char\* email);*
35. *void get\_the\_part\_of\_email\_before\_service\_provider\_name(char\* email);*
36. *static size\_t write\_data(void\* ptr, size\_t size, size\_t nmemb, void\* stream);*
37. *static size\_t WriteMemoryCallback(void\* contents, size\_t size, size\_t nmemb, void\* userp);*
38. *Header Files*



1. CHALLENGES

Memory management was the major challenge that we faced. Exceptions were thrown here and there during the debugging process. It was difficult to find the statements which caused the program to throw exception. The main types of exceptions that we faced were:

1. Access violation reading location.
2. Access violation writing location.
3. Access violation executing location.

Most of these problems came into existence due to improper memory management. The problems were mitigated later after managing memory.

Similarly, the widespread use of GTK3 and latest release of GTK4 resulted into a form of headache for use. As most of the functions of GTK3 were deprecated in GTK4, we had to encounter a lot of problem when we tried to get our queries answered via some of the famous platforms like StackOverflow and ChatGPT.

Face++ api present for limited use. As face++ allowed us to use its API for limited number of times, we also faced concurrency limit exceeded error.

1. FUTURE PLANS
2. Additional Biometrics i.e., Fingerprint, Barcode, QR-Code.
3. Hardware Integration: Camera and sensor integration
4. Performance statistics of students in individual basis
5. Delete classroom feature
6. UI improvements
7. Saving image in our database.
8. CONCLUSION

Organize-IT is an umbrella of upcoming tech-products, not only limited to an attendance software. The product is further in development phase of additional features that includes the features which includes following features:

The project is not free of limitations. There might have been problems regarding lack of resources in some aspects. These few drawbacks have occurred merely due to time limitation and lack of secondary sources of information’s. Though we have tried our best to keep the report free from errors, if any errors found which was not deliberately made. If the project can help any person in any personal or commercial way, we will feel that the purpose of the project has been fulfilled.

ACKNOWLEDGMENT

The present report is the outcome of the C-programming Contest, ‘CAN YOU C?’ organized by IEEE, Pulchowk Unit. The objective of the project was to use the concept of C programming to develop a practical program that can come into commercial use and to familiarize the student with the core concept of programming techniques /knowledge s/he have earned in the campus.

The development of this project was something challenging and we invested a substantial amount of time, mind mapping for the proper functioning of the program. The project wouldn’t have a functional output without the core concepts of C and understanding of working API’s and libraries. The accomplishment of the project on time would have been arduous without implementation of external libraries for facial recognition, calling API’s and mainly implementing a user-friendly GUI and without the collective efforts of the team members.

REFERENCES

1. Daniel Steinberg, *Swedish open-source developer and curl maintainer*,

Sweden, 1996-2023.

1. akheron/jansson repository on Github [Online]. Available: [www.digip.org/jansson](http://www.digip.org/jansson)
2. Firebase Auth REST API, Creative Common Attribution 4.0 License,2023-01-17 UTC [Online]. Available: <https://firebase.google.com/docs/reference/rest/auth>
3. Firebase Database REST API, Creative Common Attribution 4.0 License, Apache 2.0 License, 2023-01-17 UTC [Online]. Available: <https://firebase.google.com/docs/reference/rest/database>
4. OpenAI, “ChatGPT,” [Online]. Available: <https://openai.com/models/gpt-2/>.
5. The GTK Team, “GTK-4.0,” [Online]. Available: <https://docs.gtk.org/gtk4/index.html>
6. Stack Overflow, [Online]. Available: <https://stackoverflow.com/>.
7. Abstract, The API Company, San Francisco & Paris, [Online]. Available: <https://www.abstractapi.com/>
8. Face++ AI, [Online]. Available: <https://www.faceplusplus.com/ai-face-detection/>.