**Login Portal with Face Authentication**

A PROJECT REPORT

*Submitted by*

**B Sai Abhishek BL.EN.U4AIE21015**

**B Ruchith Balaji BL.EN.U4AIE21017**

**Chillakuru Hari BL.EN.U4AIE21038**

***for the course***

***19CSE202- DATA BASE MANAGEMENT SYSTEMS***

***Guided and evaluated by***

***Divya Ma’am***

***Dept. of CSE,***

A close-up of a logo

Description automatically generated

**AMRITA SCHOOL OF ENGINEERING, BANGALORE**

**AMRITA VISHWA VIDHYAPEETHAM**

**BANGALORE-560 035**

**December 2023**

**ABSTRACT**

***Facial recognition has proven to be very useful and versatile, from Facebook photo tagging and Snapchat filters to modelling fluid dynamics and designing for augmented reality. However, facial recognition has only been used for user login services in conjunction with expensive and restrictive hardware technologies, such as in smart phone devices like the iPhone x. This project aims to apply machine learning techniques to reliably distinguish user accounts with only common cameras to make facial recognition logins more accessible to website and software developers. To show the feasibility of this idea, we created a web API that recognizes a user’s face to log them in to their account, and we will create a simple website to test the reliability of our system. In this paper, we discuss our database-centric architecture model, use cases and activity diagrams, technologies we used for the website, API, and machine learning algorithms. We also provide the screenshots of our system, the user manual, and our future.***

**INTRODUCTION**

**1.1 Background**

Artificial intelligence and Machine Learning have gained increasing popularity in recent years due to their ability to handle tasks that would otherwise take too much computational power, and due to their versatility, the wide range of problems they have been shown to solve. One of the most well known tasks that machine learning has made possible is face recognition.

Face recognition technology has been used for a variety of applications including automatic tagging in Facebook photos, Snapchat lenses that overlay dog ears on someones head, and security and surveillance, with the more recent capacity to track individuals moving throughout a closed space as they cross in front of security cameras .

Facial recognition systems rely on unique facial features as an additional layer of security to identify and distinguish people whether theyre new faces or old ones in a database. We set out to apply this technique to the field of internet security, along with Captchas, Im not a robot checkboxes, security questions, two factor authentication, and many others. Facial recognition has the potential to be a much simpler approach to security than remembering additional security information or connecting other accounts and devices

**1.2 Problem**

However, theres an obvious issue with using facial recognition to login to your account. Anyone with you picture can log into it too. Furthermore, if anyone hacks the site, they may gain access the the database of face data and be able to relate user accounts to actual faces, then do some reverse processing to label those faces with real names rather than whatever alias may have been their username. And is facial recognition even accurate enough to avoid false positives and log someone into the wrong account? Can someone just sit in front of a camera long enough to get sneak past a sites security?

Apple has provided a solution to solve this photo trick by relying on dual cameras and an array of projected infrared dots to detect depth in its new facial recognition system. However, such a solution is limited to devices with expensive hardware upgrades and cant be applied to lower cost applications. Higher costs often limit other improvements, complicate manufacturing, and raise the price for consumers.

**1.3 Solution**

To address the issues of impersonation, we propose a system that would use a video stream, rather than a still image, to check that the correct person is logging in. if our facial recognition detects a video frame without the correct matching face, the login step will fail. However, such a system could be spoofed by simply holding a video up to the camera, so we will also ask the user to perform some random gesture to ensure its a real person in front of the camera.

Our solution is pure software-based, requiring no additional hardware expenses, and can be applied to a wide range of applications including building security, unlocking cellphones, and website logins. The reliability and ease of use of our system will be reliant on the accuracy of the facial recognition and the set of gestures available to use.

**LITERATURE SURVEY**

**Paper-1**

***Basarkar, S., Bedmutha, S., Hire, S. and Lad, P., STUDENT ATTENDANCE SYSTEM USING FACE RECOGNITION.***

* In this paper the proposed student attendance system is based on face recognition algorithms. In this system each student image will be captured and then face regions then of all students is extracted and the pre-processed for further processing.
* After recognition of faces then it is passed for post-processing where the attendance of every student is generated into excel sheet.
* A robust face detection algorithm has been implemented.

**Paper-2**

**Pradyumna, J., Khan, T. and Kumar, K., Smart Attendance System using Face Recognition.**

* The goal of this paper is to outline the specifications for the "Face Track: Facial Recognition Attendance System" development.
* This system uses facial recognition technology to automate the tracking of attendance in a company.
* Uses Deep learning Techniques.

**Paper-3**

**Bhanushali, R., Agarwal, C., Dongare, T. and Sharma, S., Student Management System.**

* The system starts with the college entering an excel sheet containing the registered email id of the student that shall go for form submission.
* Only the students in this table who have a registered email id can fill out the form.
* The waterfall model is a sequential design approach used in software development processes.

**REQUIREMENTS**

**2.1 Functional Requirements**

* Account creation and login system.
* Certification with camera .
* Distinguish between real person and images .
* Recognize faces and gestures.

The functional requirements of our project describe features our system must have to be successful. Users need to be able to create accounts and store their facial data for the site to identify them later. The site needs to be able to identify users with facial recognition, which requires some form of video camera. The site must be able to stop impersonation, distinguishing between real people and fake copies. And finally, the site needs to be able to recognize faces and gestures to log users into the correct accounts.

**2.2 Non-Functional Requirements**

* Critical: Secure API
* Recommended: Fast authentication
* Suggested: Continuous improvement

The non-functional requirements describe features that improve the sites performance or would otherwise benefit the system. A secure API for storing and using face data to identify users and learn new faces would be ideal to avoid the risk of losing identifiable information in the event of a site hack. Since we imagine nobody wants to sit in front of a camera for minutes on end trying to access an account, the site should be able to identify users quickly, and provide a quick login experience. Since Software programs often require continuous debugging efforts as new features are added and new issues are discovered, the source code should be clear and easy to follow, relying on software tools to simplify the implementation.

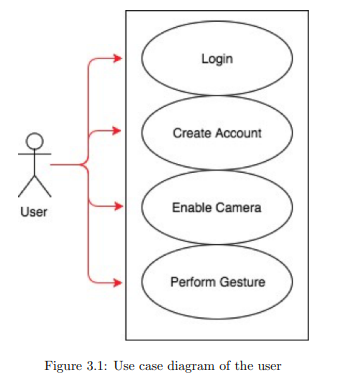
**2.3 Design Constraints**

* Uses computer camera
* Uses video
* Web based platform
* Camera frame speed

The constraints are limiting factors of our system, which in our case is mostly technology. Since we are building a login system for websites, the APIs and other software tools need to work well with websites. We also expect that everyone will be logging in with a computer (we aren’t supporting phones for this project), so our facial recognition needs to work in poor lighting conditions and without relying on expensive camera features.

**USE CASES**

The use case represents the list of actions and event steps which define the interactions among users, websites, and the API.



**3.1 User**

**3.1.1 User Case 1: Login**

* Name: Login
* Goal: Provide user access to the system
* Actor: User •
* Pre-conditions:

– The user has an active connection to website

– The user has previously signed up

– The user knows his or her username

* Steps:

– The user types in the username

– The user clicks ”verify”

– The system recognizes the user’s face and verifies that he or she is a real person

* Post-conditions:

– The user’s username has been verified in the system

* Exception: The user enters invalid username

**3.1.2 User Case 2: Create account:**

* Name: Create account
* Goal: Provide user access to the system
* Actor: User
* Pre-conditions:
  + The user has an active connection to website.
* Steps:

– The user types in the username

– The system takes some photos of the user

– The system stores in the pictures and username in the database.

* Post-conditions: – The user’s account is created.
* Exception: N/A

**3.1.3 User Case 3: Enable camera:**

* Name: Enable camera
* Goal: Provide the camera in user’s computer access to record user’s face
* Actor: User
* Pre-conditions:

User’s username has been verified.

* Steps:

The system pops out a request to enable camera.

The user clicks” yes”

* Post-conditions:

– The camera is enabled

* Exception: The computer does not have a camera

**3.1.4 User Case 4: Perform gesture:**

* Name: Perform gesture
* Goal: Verify whether the user is a real person or a photo
* Actor: User
* Pre-conditions:

– The user enters valid username

– The user’s camera is enabled

– The user’s face has been detected by the system

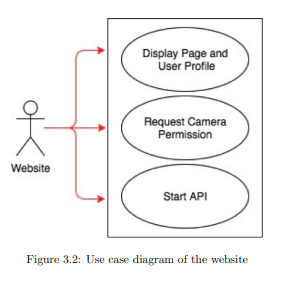
* Steps:

– The system displays a certain gesture for the user to perform

– The user performs the gesture

– The system analyses the user’s gesture

* Post-conditions: – User successfully login
* Exception: The user performs wrong gesture.

****

**3.2 Website**

**3.2.1 Use Case 1: Display page**

* Name: Display page
* Goal: Provide user access to the page in browser
* Actor: Website
* Pre-conditions:

– The user has an active connection to website – The browser is compatible to the user’s computer

* Steps:

– Read HTML, CSS, and JavaScript files

– Execute python files to make operations with server

* Post-conditions:

– The page is successfully displayed

* Exception: 404 Not Found

**3.2.2 Use Case 2: Start API**

* Name: Start API
* Goal: Retrieve information using API technologies
* Actor: Website
* Pre-conditions:

– The page is successfully display

– User is interacting with the system

* Steps:

– The system send a request to other websites using API to request information

– The system receives responses from other websites

* Post-conditions:

– The result is successfully return by the API

* Exception: The API does not work

**3.2.3 Use Case 3: Display user profile**

* Name: Display user profile
* Goal: Provide user permission to see his or her information
* Actor: Website
* Pre-conditions:

– The user is successfully logged in

– The user’s browser is compatible with his computer

* Steps:

– The system user’s information from the database

– The system lists the information on the web page

* Post-conditions: – User’s information is successfully displayed.
* Exception: Database connection error

**A diagram of a person with a person figure

Description automatically generated**

**3.3 API**

**3.3.1 Verify Face:**

* Name: Verify face
* Goal: Recognize user’s face in the camera
* Actor: API
* Pre-conditions: – The user’s camera is enabled
* Steps:

– Separate the camera video into multiple frames

– Use pre-trained machine learning model to check if there is a face in the frame

– Draw bounding box around the face detected

* Post-conditions:

– User’s face is successfully detected Exception: User’s face is not found

**3.3.2 Verify Gesture**

* Name: Verify gesture
* Goal: Test if the user is real
* Actor: API
* Pre-conditions: – User’s face is detected.
* Steps:

– The system displays a specific gesture on the screen

– The user perform the gesture accordingly – The system verify is the gesture is correct

* Post-conditions:
* – The gesture is verified and user is logged in
* Exception: User performs wrong gesture.

**3.3.3 Identify user to site:**

* Name: Identify user to site
* Goal: Map user’s face with those in the database • Actor: API
* Pre-conditions:

– The user’s face is detected

* Steps:

– Search the database

– Pick the user in the database with the most similar picture as the login user

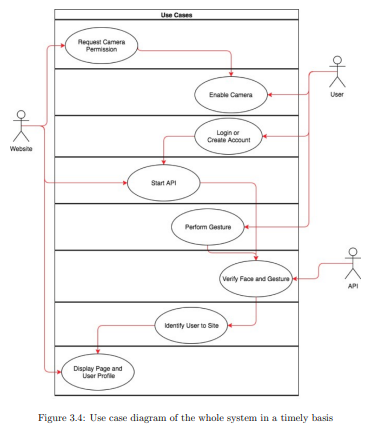
– return the user’s information

* Post-conditions:

– User is found in the database

* Exception: Database connection error.

**3.4 Swim Lane Diagram**

****

**A diagram of a server

Description automatically generatedARCHITECTURE**

Since all data will be stored in our database, we decide to build an advanced data-centric architecture for our design to make the data more accessible and manageable for users. In addition, considering security and management of APIs, we choose to have a three-tier architecture because it helps protect data security and improve manageability of different APIs. In our design, users post requests to the web server and the server will call different APIs depending on the requests from users. Multiple users can login to the system at same time from different places.

**SYSTEM IMPLEMENTATION**

**5.1 Main Page**

The main page (See figure56.1) shows two components of the system, login and registration. When the user first enters the page, he or she can either choose to login or create and account.

A screenshot of a login form

Description automatically generated

**5.2 Enter Username & Password**

The username entering page (See figure 5.2) allows the user to enter his or her username in both login and registration process. During the registration, if the username conflicts with those in the database, the system will have a warning” Username already exists”.

The password entering page allows user to enter the password. During the login process, if the password is incorrect, the system will show the warning” Invalid username or password”.

A login screen with green and white text

Description automatically generated

**5.3 Facial Detection and Recognition**

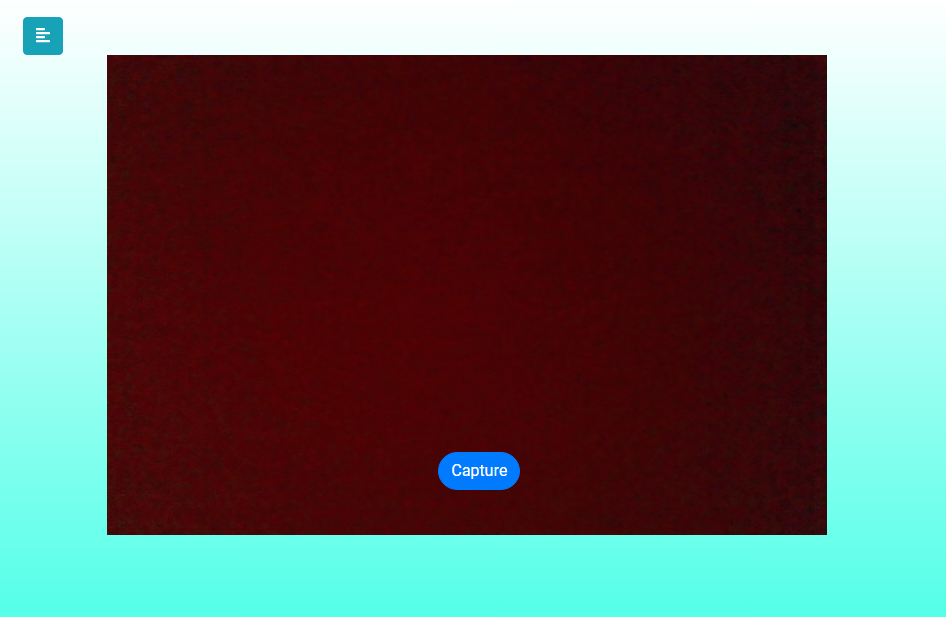
The facial detection and recognition technique retrieves machine learning algorithm, comparing user’s face data from the web cam with the photo in the database. If the photo is matched, the system will show ”authentication succeeded”; otherwise, the system will show ”authentication failed” and the user will be blocked from further steps.

**5.4 Personal Page**

After the user has been both authenticated and verified as a real person, the system will direct the user to his or her personal page.If the user presses ”Logout”, it will go back to the main page.

**A blue and white gradient

Description automatically generated**

****

**TECHNOLOGY USED**

**6.1 Programming Languages**

* **HTML5**

HTML5 is used to create documents on the web page. It defines the structure and layout of a Web document by using a variety of tags and attributes.

* **CSS3**

CSS3 is used to describe the presentation of Web pages. It also makes the web page responsive to different devices.

* **JavaScript**

JavaScript is used as a client side scripting language. Its code is written into an HTML page. When a user requests an HTML page, the script is sent to the browser.

* **Python**

Python is used for back end operations such as training machine learning models and retrieve information using the Facial Recognition API.

* **SQL**

SQL is used to communicate with a database. As the standard language for relational database management systems, SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database.

**6.2 Applications**

* **SQLlite**

SQLlite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is used to store both account information and pictures of the users.

* **Flask**

Flask is a python Web frameworks, which supports the development of the web API

**PROCEDURE**

* **Login and Registration**

We tested regular login and registration function. We first tested to register with a conflict username, and the system showed a warning. We then login with username and incorrect password, and the system also provided a warning.

* **Facial detection and recognition**

We tested our facial detection and recognition part using different faces. The system was able to detect all faces appeared on the screen and successfully authenticated the correct user.

* **Gesture recognition**

We tested gesture recognition by performing requested gestures assigned by the system. The system successfully recognized the gestures.

* **Image impersonation**

The last test we did is to test image impersonation. Face recognition test can be passed by using photos if the camera is not good enough to detect image depth. Our design solves this problem because it requires the person to perform gestures. We tested that although a different person holding the picture can pass the face recognition part, the gesture testing cannot be passed.

**FUTURE PLAN**

* **Recognize more gestures**

We plan to add more gestures into our system such as waving hands and opening mouth. The increasing number of random gestures will make the user harder to predict the move, and thus improve the security of the system.

* **Test with more people and a larger image set**

We plan to test with more people and get a more precise result of how accurate our system is on recognizing users’ faces and gestures. We will also improve the accuracy of the system by training and testing with a larger image set.

* **Store and encrypt face data in a separate database**

The photos of the users in the database have a potential security issue. To solve that, we plan to store and encrypt the photos of the user in a separate database so that even if the database is being hacked, the photos will not leak.

* **Replace users photo with face encoding**

To bring the security of our system to the next level, we could only store the characteristics of the user’s face in the database instead of the photo. Even if the face encoding data is leaked, the hacker cannot recover user’s face.

* **Remove the need for password in our implementation**

When our system reaches really high accuracy, we could remove process of entering the password during login. User’s account can be verified with facial recognition as the single security check.

**CONCLUSION**

Since the face recognition algorithm we use is more than 99% accurate, the site only needs to compare the user’s face to one stored image, rather than against a million different faces like Facebook, and our implementation checks the user’s identity over several frames, we believe that face recognition is a secure, reliable, and simple method of authentication. in the future, there may not even be need for passwords or even captcha checks to stop bots.

If we separated the database of face information from the site’s data and trained our own neural network, our implementation would be much more secure. just like signing in with your Google account doesn’t give your password away to the whatever site, logging in with your face shouldn’t give any site access to the associated data either.

Lastly, if we improved the set of gestures our site can detect, we could speed up the process substantially compared to only detecting blinks. making the authentication process quick and seamless requires making the gestures simple and unobtrusive, which we have found to be the largest hurdle we would need to solve for our design to be competitive.

**REFERENCES**

**[1]** S.R.Bharamagoudar, Geeta R.B & S.G.Totad, “Web service api for student information and course management systems”. International Journal of Advanced Research in Computer and Communication Engineering Vol. June 2013.

**[2]** Almahdi Alshareef, Ahmed Alkilany, "Toward a Student Information System for Sebha University, Libya", Fifth international conference on Innovative Computing Technology (INTECH 2015)-p 34-39

**[3]** Prabhu T Kannan, Srividya K Bansal, "Unimate: A Student Information System", International Conference on Advances in Computing, Communications and Informatics (ICACCI 2013)-p-1251-1256