ESOF-4250 DESIGN PROJECT REPORT

OVER THE SHOULDER ATTACK PREVENTION USING
BIOMETRIC AUTHENTICATION AND DYNAMIC PASSWORDS

EDITED BY

YASH GUPTA - 0869229
PETER SERTIC - 0694592
MARKO JAVORAC - 1107295

Lakehead University

Contents

1 Abstract

Digital security is of utmost importance in today's digital world. Security policies are put in place to increase data security and confidentiality, however, does nothing to protect against over the shoulder attacks. This paper proposes a new security policy, including an access control matrix, capability lists, and access control lists, as well as a multi method authentication system. The proposed system utilizes biometric facial recognition and dynamic passwords on top of traditional user name and password authentication. The system successfully identifies users based on their facial features while providing a secure dynamic password at the same time.

2 Introduction

As software systems evolve and become more critical in everyday life, the security policies that individuals and corporations adhere to must ensure that all digital assets are protected from unauthorized access. Every system needs a unique and custom tailored security policy that takes into consideration the design principles of Least Privilege and Need To Know, as security should only allow information to those who require it to perform their obligations. Security policies are vital to ensure the integrity, validity, and confidentiality of data. There are many threats that a system can be susceptible to, many of which are unknown. The security policies are made to mitigate these threats. Failure to do so can result in unauthorized breaches of data. Data breaches must be publicly disclosed when personal user data is stolen, which can cause many negative consequences, including legal action, negative public perception, and financial losses. Due to the harsh consequences of failing to protect user data, it is vital that a robust and secure security policy is put in place.

3 Problem Statement

Due to recent events regarding the unauthorized exposure of students' private data, Fake University has decided to redefine their internal software security policies and authentication methods. In this paper we propose a new security policy that covers all users along with a new biometric face authentication system which when combined, will provide a comprehensive and bleeding edge security posture for the university.

4 Security Policy

To ensure that students and staff can continue to use the schools system, a new security policy must be implemented. This policy should cover all users and data objects to ensure the highest level of security.

Security access is governed using Role Based Access Control(RBAC). RBAC consists of two groups, roles and objects. Network access to each object is restricted based on each user's role [1]. Only the necessary amount of access required to complete a user's task is granted, enforcing the Least Privilege principle and enhancing security.

First we will define all the users and data objects with the associated privileges. To implement this, we will develop an Access Control Matrix along with a Control List and Capability List.

4.1 Policy Object & Roles

A list and description of every Object and Role considered in the security policy is described below.

4.1.1 Roles

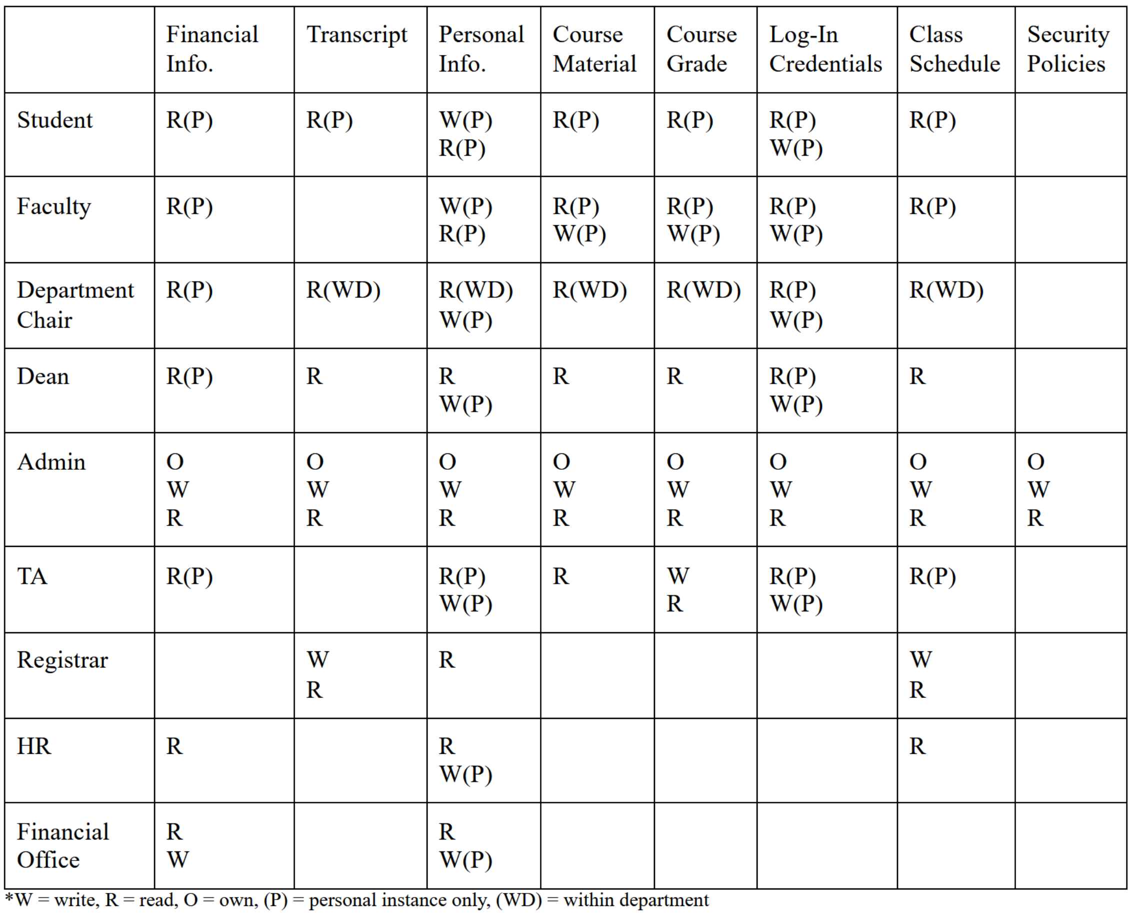
- Student: Students attending class at the university
- Faculty: Professors and Educators teaching at the university
- TA: Teaching assistants for a professor
- Department Chair: Head of a given department(Engineering, Mathematics, Business,etc.)
- Dean: Head of the University
- Humane Resources(HR): Provides services regarding employee interaction and workplace behavior.
- Registrar: Provides services regarding enrollment, academic , and financial records. Class Schedule.
- Financial Office: Provides services regarding payments, financial records, school based loans and bursaries.
- Admin: Security Policy Administrator.

4.1.2 Objects

- Log-In Credentials: Username, Password.
- Personal Info: Full Name, Address, Email, Phone, Student ID Number.
- Financial Info: School payments, bursaries, scholarships.
- Transcript: Final Grades to date. Classes completed.
- Course Grade: Marks in currently active courses.
- Course Material: Material for currently active courses.
- Class Schedule: Time, room number, class code, Teacher.
- Security Policies: Policies in place to ensure proper information confidentiality and integrity.

4.2 Access Control Matrix

An access control matrix is an abstract way to represent the security policies of a software system. In our implementation, it represents which role type can access which asset type and to which degree. In our case, the degrees of access can vary due to the inherent complexity of each asset.



	Financial Info.	Transcript	Personal Info.	Course Material	Course Grade	Log-In Credentials	Class Schedule	Security Policies

Student	R(P)	R(P)	W(P)
			R(P)
				R(P)	R(P)	R(P)
						W(P)
							R(P)	

Faculty	R(P)		W(P)
			R(P)
				R(P)
				W(P)
					R(P)
					W(P)
						R(P)
						W(P)
							R(P)	

Department Chair	R(P)	R(WD)	R(WD)
			W(P)
				R(WD)	R(WD)	R(P)
						W(P)
							R(WD)	

Dean	R(P)	R	R
			W(P)
				R	R	R(P)
						W(P)
							R	

Admin	O W R	O W R	O W R	O W R	O W R	O W R	O W R	O W R

TA	R(P)		R(P)
			W(P)
				R	W
					R
						R(P)
						W(P)
							R(P)	

Registrar		W R	R				W R	
HR	R		R W(P)				R	
Financial Office	R W		R W(P)					

*W = write, R = read, O = own, (P) = personal instance only, (WD) = within department

Figure 4.1: Access Control Matrix

4.3 Access Control List

An Access Control List is an operating system-level list of permission with respect to a data asset. Each element in the list specifies the role and specific permission given to that role.

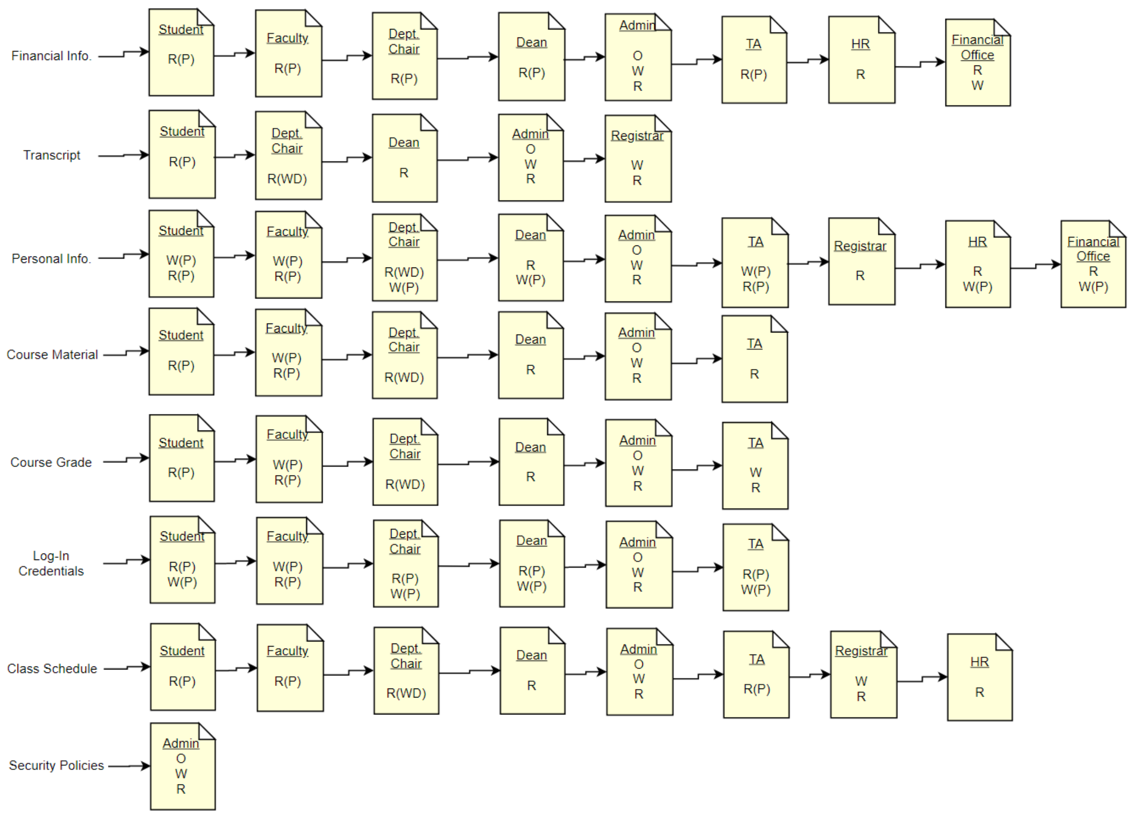


Figure 4.2: **Access Control List**

(*W = write, R = read, O = own, (P) = personal instance only, (WD) = within department)

4.4 Capability List

Capability lists are another convenient and accessible representation of a systems security policy. In the Access control list, we visualized permission through the lens of the data asset, while in the capability list, we structure the policies through the eyes of a specific role. This gives IT support staff an easy reference point for seeing a particular user's permissions.

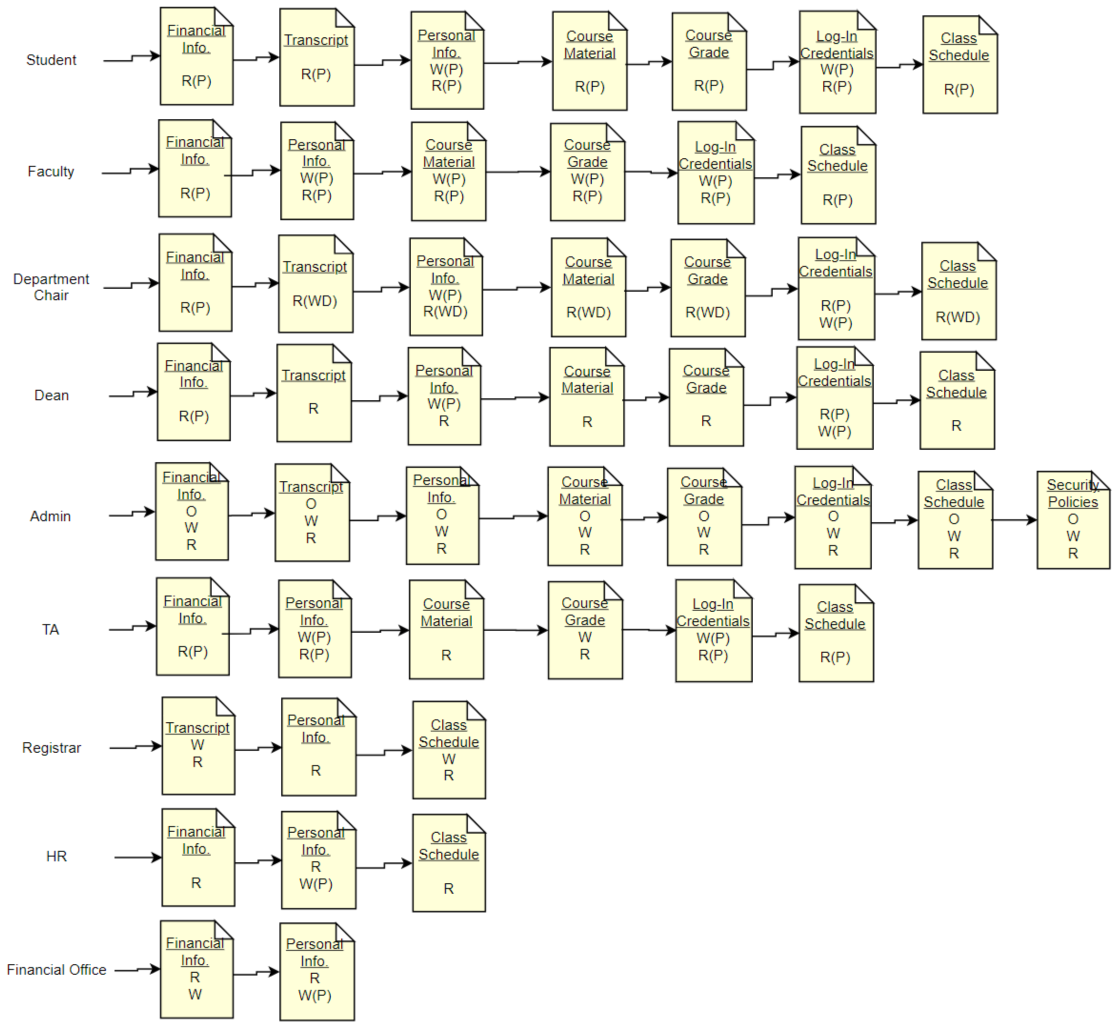


Figure 4.3: **Capability List**

(*W = write, R = read, O = own, (P) = personal instance only, (WD) = within department)

5 Authentication Mechanism

To ensure that users are able to authenticate in a safe and efficient manner, Fake University will be implementing multi-factor authentication system based on the latest industry standards. The authentication mechanism will use :

- Password-based authentication
- Dynamic Passwords
- Biometric authentication

The user will be able to log in securely using the password based authentication system along with either Dynamic Password (One Time Passwords) or using their Biometrics (Facial Recognition).

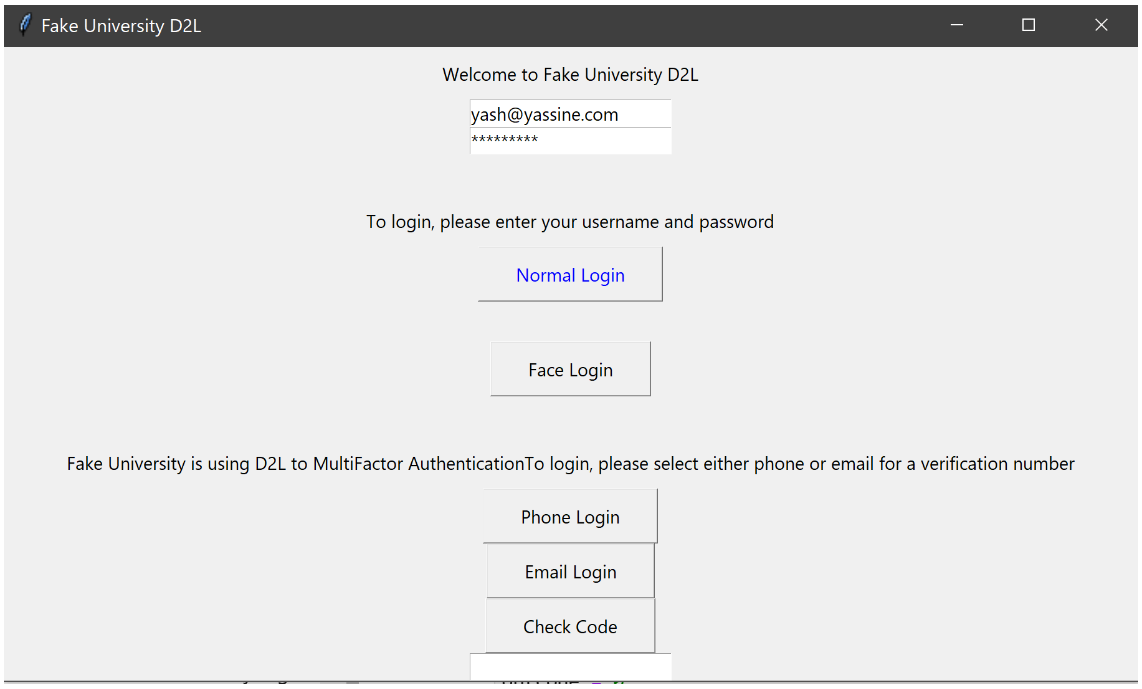

Fake University D2L
Welcome to Fake University D2L
yash@yassine.com
****
To login, please enter your username and password
Normal Login
Face Login
Fake University is using D2L to MultiFactor AuthenticationTo login, please select either phone or email for a verification number
Phone Login
Email Login
Check Code

Figure 5.1: **Fake University Authentication System**

5.1 Password Based Authentication

Password based Authentication system is the most common way of authentication where each user has a unique combination of user-name and password. This is enforced by treating the user-name as unique key i.e. no two users can have the same username. In our system, we are using Google's firebase hosting services. Firebase has real time database and authentication services. The authentication service can be used for authenticating users. Once the user has been added to the authentication system, no one can see the password since it is hashed (encrypted). Once the user enters their username and password, the credentials are sent to the firebase authentication service which then authenticates is there is a user by that username and if the credentials are correct.

5.2 Dynamic Passwords

After the user is authenticated by the firebase, further authentication can be performed by using dynamic passwords for example One-time-passwords. One-time-passwords can be used to to provide an extra layer of security and create a over-the-shoulder immune authentication system. To enable this, two unique methods were implemented.

5.2.1 Phone authentication

To provide a seamless and secure experience, we have implemented a Multifactor authentication system built on the Twilio Communications API[2]. Once the user logs in using their correct Fake University username and password, they will have the option to send a verification code to your phone. This code will only be valid for a set period of time(30s in our demonstration) and sent to the phone number preconfigured when the user is initially registering. Once the code is received and entered into the system, the user will have access to all the services provided. This approach is one of the final methods deployed to prevent over the shoulder attacks as the assailant will not be in possession of both the victims login and authentication device. As stated before, Twilo's text API was integrated with the application to provide the necessary technological infrastructure to communicate with such devices. For our demonstration, Twilio provided a free option for prototyping but to be used for production, Fake University will be required to pay additional fees. A wide variety of payment options must be presented to Fake university management prior to any public deployment. Although this may add significant costs over its lifetime, the cost of repeated unauthorized access to Fake Universities digital assets could be disastrous.

5.2.2 Email Authentication

Email Authentication provides an alternative to phone authentication for enforcing dynamic passwords. This authentication system allows users to use a dynamic password or a one-time-password sent to their secondary chosen email. In our system the user is required to submit a secondary email to the university database or in our case the firebase database as part of their personal information. Once the user logs in using the password based authentication and choose email authentication, the system will receive the chosen backup email using the users username as the key and will send the code. This code will only be valid for a set period of time(30s in our demonstration). Our implementation uses pythons smtp library along with email library. Our implementation uses Gmail's smtp services and requires a Gmail account for sending the authentication email.

5.3 Biometric Authentication

Biometric Authentication is one most useful and secure form of authentication due to the fact that a person's biometrics like finger prints or facial information are unique to that person only and cannot be replicated easily. In our system, we are using facial recognition to recognise a user based on their unique facial features.

Face recognition systems have grown from infant research projects to large-scale public deployment. From cellphones to public transit systems, face recognition is used to increase the speed and over quality of the service. While there are a variety of implementations, recent developments in deep learning technology have allowed for the rapid development of highly accurate systems. By building a model that can quickly identify whether the user is who they say they are, Fake University will be provide a safe and effective tool for authentication.

5.3.1 Dataset

The first step to developing a model for face recognition is constructing a dataset of the users face. When the user is first registered, they will be prompted to enter a face capturing utility that will rapidly develop the required dataset. Along with the images entered, the system will also generate modified versions of the set with added noise, rotations and other methods to vary the pictures. This will ensure the next phase has enough data to work in variable conditions.

In our implementation, since it is a prototype, the dataset was created separately for only one user. The data is created by using openCV's face detection feature using harr features to get the extract the face of the user for the model to understand what the user looks like.

A significant number of neural networks have a direct performance correlation with the size of their training dataset[3]. To ensure that the facial authentication system can be deployed rapidly with minimal time and user image input, a variety of data augmentation methods were used to expand the dataset. A total of 4 techniques were used including, horizontal and vertical translations, rotations, and brightness variations. Combining these methods provides a quick and easy way of expanding the dataset with corresponding model performance boosts. While this does not result in perfect performance, it does allow users to quickly submit a minimal amount of face images during registration and enhances the overall user experience.

5.3.2 Model Training

Convolutional Neural Networks are really useful for image classification and can be used for face recognition due to their ability to learn features in the spacial domain. In our implementation, we are training the user's face dataset created in the previous step to train the model to learn about the unique facial features of the user to build a classifier to differentiate the user from others.

5.3.3 Model Deployment

Once the model is trained and validated. The model is saved under the users user name and the file name is saved along with their personal info on the firebase database which can then be extracted if the user chose the biometric authentication method.

To authenticate the user, the interface will pop up a camera window to detect the users face and will send it to the saved model trained for the user. If the user is detected, then the user will be granted access.

6 Future Work

To provide more accurate face detection, future work will include new data augmentation methods such as stacking augmenters and colour manipulation. Different neural networks as well as face detection techniques can also be researched to improve the model accuracy. Further work on MFA will include enhancing the format of the verification code message and testing different time limits.

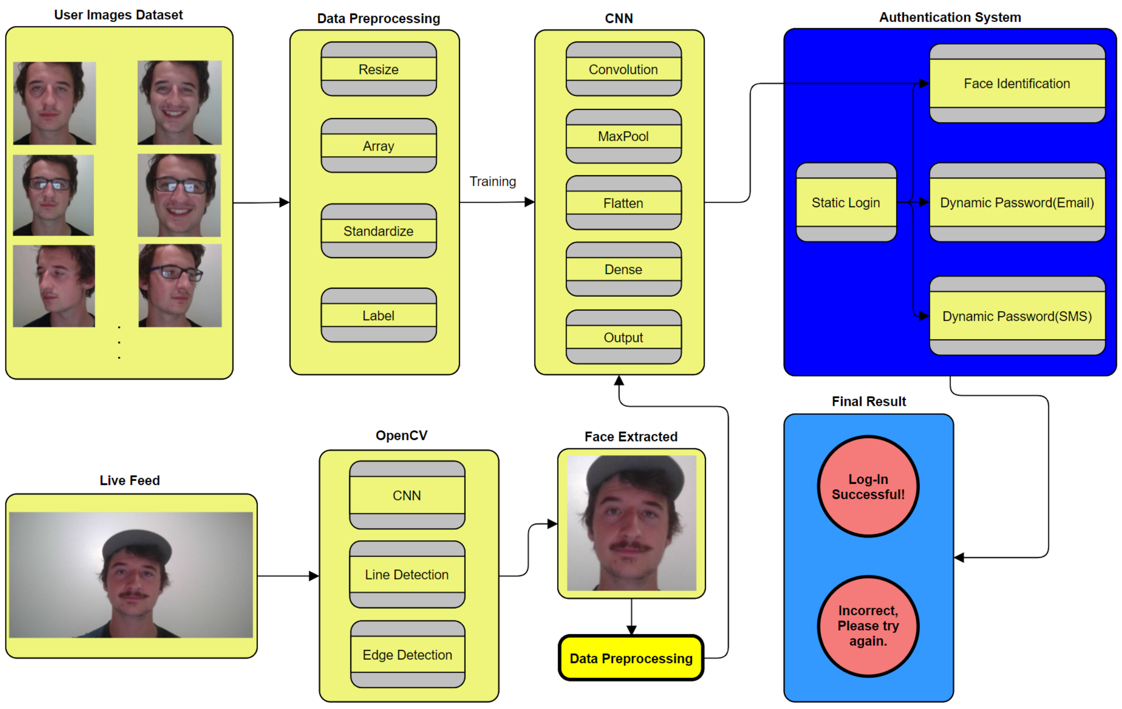


Figure 5.2: **Facial Recognition System Overview**

7 Conclusion

In this paper, we successfully described a new security policy that covers all roles and digital data assets across the entire Fake University. All roles and assets are thoroughly defined and then applied in an Access Control Matrix, Access control List, and Capability List. We also propose a new sign-in mechanism that protects against over-the-shoulder attacks. This mechanism implements 2 unique authentication methods, including Facial Biometrics and Dynamic Passwords on top of a static authentication system built on Firebase.

References

[1] Yeagley, Geoff. “IT Security Policies and Procedures: Why You Need Them.” Home - Compass IT Compliance, Compass IT Compliance LLC, 10 Sept. 2015, 10:54:42 AM, www.compassitc.com/blog/it-security-policies-and-procedures-why-you-need-them.

[2] https://www.twilio.com/company

[3] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

8 Appendix

8.1 System Implementation

```python
#---------------------------------------------------------
# 4250 - Software Safety and Security
# Author:
#  Marko Javorac
#  Yash Gupta
#  Peter Sertic
# Date: December 4th, 2020
# Description: Main GUI application for Fake University
#---------------------------------------------------------

#---------------------------------------------
# Python Imports
#---------------------------------------------
from tkinter import *
from PIL import ImageTk,Image
import time
import tkinter as tk
import numpy as np
import cv2
import pyrebase
from twilio.rest import Client
import time
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import math, random
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import tensorflow as tf
import keras
from keras.models import model_from_json

from tensorflow.keras.layers import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
import numpy as np
import os
from sklearn.pipeline import Pipeline
from keras.models import model_from_json
#---------------------------------------------
# Functions definitions used throughout code
#---------------------------------------------

# Home page setup
def homePage():

    #Set up homepage
    global home
    home = Tk()
    home.title('Fake University D2L')
    Title = Label(home, text ="Welcome to Fake University D2L", pady=20,padx=
        100).pack()

    #Username text input field
    global userName
    userName = Entry(home, width= 20)
    userName.pack()
    userName.insert(0, "yash@yassine.com")

    #Password text input field
    global password
    password = Entry(home, show="*", width= 20)
    password.pack()
    password.insert(0, "yashgupta")

    #Spare space
    space = Label(home, text ="", pady=10).pack()

    userGuide = Label(home,
        text =" To login, please enter your username and password ",
            pady=20,padx= 100).pack()

    #Create button widget
    global normalLogin
    normalLogin = Button(home, text = "Normal Login", padx=50, pady=10,
        command=normalLoginClick, fg='blue').pack()
    space2 = Label(home, text ="", pady=10).pack()

    #Need to fix this
    #leaveApp = Button(home, text="Quit", command=home.destroy).pack()

    #Run it
    home.mainloop()


#Frame constructor
def show_frame():
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)

    global img
    img = Image.fromarray(cv2image)

    global imgtk
    imgtk = ImageTk.PhotoImage(image=img)
    lmain.imgtk = imgtk

    lmain.configure(image=imgtk)
    lmain.after(10, show_frame)


#Set up CV frame
def cvFrame():

    #CV frame start
    faceWindow = tk.Toplevel()
    faceWindow.wm_title("Face Login")

    #Generate frame
    imageFrame = tk.Frame(faceWindow, width=300, height=250)
    imageFrame.grid(row=0, column=0, padx=10, pady=2)

    #Grab frames
    global lmain
    lmain = tk.Label(imageFrame)
    lmain.grid(row=0, column=0)

    #Get capture
    global cap
    cap = cv2.VideoCapture(0)

    #Show the CV frame
    show_frame()

    #run window
    window.mainloop()


# Normal Login Click function
def normalLoginClick():

    #Check if pass and user match
    userNameCheck = userName.get()
    passwordCheck = password.get()

    if(authentication(userNameCheck,passwordCheck)==True):


        #Set up second stage of Login
        faceLogin = Button(home, text = "Face Login", padx=50, pady=10,
            command=faceLoginClick).pack()
        space3 = Label(home, text ="", pady=10).pack()

        userGuide = Label(home,
            text ="Fake University is using D2L to MultiFactor Authentication" +
            "To login, please select either phone or email for a verification
                number",
            pady=20,padx= 100).pack()

        #For sending codes to either phone or email
        phoneLogin = Button(home, text = "Phone Login", padx=50, pady=10,
            command=phoneLoginClick).pack()
        emailLogin = Button(home, text = "Email Login", padx=50, pady=10,
            command=emailLoginClick).pack()

        #Checking verification code
        checkVeriCode = Button(home, text = "Check Code", padx=50, pady=10,
            command=checkCodeClick).pack()

        global veriCode
        veriCode = Entry(home, width= 20)
        veriCode.pack()

    else:
        wrong = Label(home, text="Wrong Combo", fg='red').pack()

# Setup function for users home page
def userHome():

        userHome = Tk()
        userHome.title("User Home")
        welcome = Label(userHome, text="Welcome Marko", width=100).pack()
        logout = Button(userHome, text="Logout",
            command=lambda:[userHome.destroy(),logoutFunc()]).pack()


# Logout functionality
def logoutFunc():

    homePage()


# Face Login Click function
def faceLoginClick():

    #cvFrame()

    detector_face()
    #Place holder for face authentication
    #if((userName.get() == 'Marko') & (password.get() == 'Apple')):

    #else:
    #    wrong = Label(root, text="Wrong Combo").pack()


    print('Face Login Testing')

def genotp():

    digits = "0123456789"
    OTP = ""
    for i in range(6) :
        OTP += digits[math.floor(random.random() * 10)]
    return OTP

def emailOtp(sms_gateway,otp):
    email = "yashbest61@gmail.com"
    pas = "Yashgupta"
    smtp = "smtp.gmail.com"
    port = 587
    server = smtplib.SMTP(smtp,port)
    server.starttls()
    server.login(email,pas)
    msg = MIMEMultipart()
    msg['From'] = email
    msg['To'] = sms_gateway
    msg['Subject'] = "OTP for Authentication"
    body = otp
    msg.attach(MIMEText(body, 'plain'))
    sms = msg.as_string()
    server.sendmail(email,sms_gateway,sms)
    server.quit()
#Email Login Click function
def emailLoginClick():

    global db
    global signin
    global otpcode
    global tstart

    username = signin['email'].split('@')[0];
    backup_email = db.child('users').child(username).get().val()['back_email']
    otp = genotp();
    otpcode = otp
    emailOtp(backup_email,otp)
    #Sedning email code
    tstart = time.perf_counter()

    print('Phones Login Testing')

#Phone Login Click function
def phoneLoginClick():

    global db
    global signin
    global otpcode
    global tstart

    username = signin['email'].split('@')[0];
    phone = db.child('users').child(username).get().val()['phone']
    otp = genotp()
    otpcode = otp
    print(otp)
    print(otpcode)
    client = Client("AC27b850ec7609edcd7b0827379f203670",
        "ca4ec4ee5097c2e82ec72d62d61a11cc")
    client.messages.create(to=phone,
                 from_="+17609913479",
                 body=otp)
    tstart = time.perf_counter()

    #Sending phone code

    print('Phones Login Testing')
```

```
#Phone Login Click function
def checkCodeClick():

    global tstop
    global tstart
    tstop = time.perf_counter()
    total_time = round(tstop-tstart)

    global otpcode
    #Pulling code
    code = veriCode.get()
    code = str(code)
    print("code is "+code)
    print("\notp is "+otpcode)
    errorval = 0;
    #Place holder for face authentication
    if(code != otpcode):
        errorval = -1
    elif(total_time>30):
        errorval = -2


    if(errorval ==0):
        home.destroy()
        userHome()
    elif(errorval==-1):
        wrong = Label(home, text="Wrong Code").pack()
    else:
        wrong = Label(home, text="Code Timed Out").pack()

    print('Phones Login Testing')

def authentication(email,password):
    global firebase
    global signin
    auth = firbase.auth()
    try:
        sign_in = auth.sign_in_with_email_and_password(email,password)
        signin = sign_in
        return True
    except:
        return False;
def check_face(img_path):
    data = []


    image = load_img(img_path, target_size=(200, 200))
    image = img_to_array(image)
    image = preprocess_input(image)
    data.append(image)
    model = tf.keras.models.load_model("frrmodel.h5")
    data = np.array(data, dtype="float32")
    result = model.predict(data)
    result = tf.keras.backend.argmax(result).numpy()

    return result
```

```
def detector_face():

    face_classifier = cv2.CascadeClassifier('harr.xml')
    cap = cv2.VideoCapture(0)
    count = 0;
    global home

    while True:
        k = 10;
        ret, frame = cap.read()
        if face_extractor(frame,face_classifier) is not None:
            count+=1
            face = cv2.resize(face_extractor(frame,face_classifier),(400,400))

            file_name_path = './checkImages/' + str(count) + '.jpg'
            cv2.imwrite(file_name_path,face)
            if(check_face(file_name_path)[0]==0):
                print('marko found')
                k = 12
            else:
                print('Not marko')
    #
        cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
            cv2.imshow('face cropper',face)
        else:
            pass
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if k==12:
            home.destroy()
            userHome()
            break
        if key == ord("q"):
            break

    cap.release()
    cv2.destroyAllWindows()
def face_extractor(img,face_classifier):

    faces = face_classifier.detectMultiScale(img,1.3,5)
    if faces is ():
        return None;
    for (x,y,w,h) in faces:
        x = x-10
        y = y-10
        cropped_face = img[y:y+h+50,x:x+w+50]
    return cropped_face
#--------------------------------------------
# Main Logic of application
#--------------------------------------------
firebaseConfig = {
    "apiKey": "AIzaSyAEuJHIG80LKP2I7cDq20MWbjP3aYWBLhY",
    "authDomain": "esof4250-84a4c.firebaseapp.com",
    "databaseURL": 'https://esof4250-84a4c-default-rtdb.firebaseio.com/',
    "projectId": "esof4250-84a4c",
    "storageBucket": "esof4250-84a4c.appspot.com",
    "messagingSenderId": "735055618415",
    "appId": "1:735055618415:web:e1f6f253adc9388f05c89a",
    "measurementId": "G-B92191L3CK"
      };
firbase = pyrebase.initialize_app(firebaseConfig)
db = firbase.database()
# print(db.child('users').child('yash').get().val()['phone'])
fire_auth = 0;
optcode = 0;
signin = 0;
tstart = 0;
tstop = 0;
homePage()
```

8.2 Data Creation and Model Training

```
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
import tensorflow as tf
import keras
from keras.models import model_from_json

from tensorflow.keras.layers import Input
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
import numpy as np
import os
from sklearn.pipeline import Pipeline
from keras.models import model_from_json
def face_extractor(img):
    faces = face_classifier.detectMultiScale(img,1.3,5)

    if faces is ():
        return None;
    for (x,y,w,h) in faces:
        x = x-10
        y = y-10
        cropped_face = img[y:y+h+50,x:x+w+50]
    return cropped_face
face_classifier = cv2.CascadeClassifier('harr.xml')
```

```python
cap = cv2.VideoCapture(0)
count = 0;

while True:
    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count+=1
        face = cv2.resize(face_extractor(frame),(400,400))

        file_name_path = './yimages/' + str(count) + '.jpg'
        cv2.imwrite(file_name_path,face)

#
    cv2.putText(face,str(count),(50,50),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,0),2)
        cv2.imshow('face cropper',face)
    else:
        pass
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
# Directory The image dataset is stored
DIRECTORY = r"C:\Users\yashb"
CATEGORIES = ["MarkoDataset", "without_mask"]
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:
    path = os.path.join(DIRECTORY, category)
    for img in os.listdir(path):
        img_path = os.path.join(path, img)
        image = load_img(img_path, target_size=(200, 200))
        image = img_to_array(image)
        image = preprocess_input(image)

        data.append(image)
        if (category == CATEGORIES[1]):
#             zak
            labels.append(1)
        else:
            labels.append(0)

data = np.array(data, dtype="float32")
labels = np.array(labels)
(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.30, random_state=42)

# creating a model
model = Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(200, 200,
    3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(2))
model.compile(optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])

history = model.fit(trainX, trainY, epochs=3,
                   validation_data=(testX, testY))
model.save("frrmodel.h5")


[INFO] loading images...
Train on 2544 samples, validate on 1091 samples
Epoch 1/3
2544/2544 [==============================] - 61s 24ms/sample - loss: 0.1548 -
    accuracy: 0.9332 - val_loss: 0.0131 - val_accuracy: 0.9936
Epoch 2/3
2544/2544 [==============================] - 58s 23ms/sample - loss: 0.0230 -
    accuracy: 0.9902 - val_loss: 0.0317 - val_accuracy: 0.9908
Epoch 3/3
2544/2544 [==============================] - 54s 21ms/sample - loss: 0.0406 -
    accuracy: 0.9921 - val_loss: 0.0561 - val_accuracy: 0.9817
```

8.3 Data Augmentation

```
from numpy import expand_dims
from keras.preprocessing.image import load_img
from keras.preprocessing.image import save_img
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import array_to_img
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from PIL import Image, ImageDraw

#------------------------------------
# Horizontal/Vertical Image Shifter
#------------------------------------

#Loading images
#img = load_img('marko.jpg')

#Converting to array
#data = img_to_array(img)

#Expanding
#samples = expand_dims(data, 0)

#Unique data augmentaters
vertdatagen = ImageDataGenerator(height_shift_range=0.5)
```

```
horizdatagen = ImageDataGenerator(width_shift_range=[-150,150])
rotdatagen = ImageDataGenerator(rotation_range=90)
bridatagen = ImageDataGenerator(brightness_range=[0.2,1.0])
zoomdatagen = ImageDataGenerator(zoom_range=[0.5,1.0])

# #Augmentation Iteraters
# horit = horizdatagen.flow(samples, batch_size=1)
# vertit = vertdatagen.flow(samples, batch_size=1)
# rotit = rotdatagen.flow(samples, batch_size=1)
# briit = bridatagen.flow(samples, batch_size=1)
# zoomit = zoomdatagen.flow(samples, batch_size=1)

#-----------------------
#Main Augmentation Loop
#-----------------------
for i in range(278):

    #Loading images
    img = load_img(str(i)+'.jpg')

    #Converting to array
    data = img_to_array(img)

    #Expanding
    samples = expand_dims(data, 0)

    #Unique data augmentaters
    vertdatagen = ImageDataGenerator(height_shift_range=0.5)
    horizdatagen = ImageDataGenerator(width_shift_range=[-150,150])
    rotdatagen = ImageDataGenerator(rotation_range=90)
    bridatagen = ImageDataGenerator(brightness_range=[0.2,1.0])
    zoomdatagen = ImageDataGenerator(zoom_range=[0.5,1.0])

    #Augmentation Iteraters
    horit = horizdatagen.flow(samples, batch_size=1)
    vertit = vertdatagen.flow(samples, batch_size=1)
    rotit = rotdatagen.flow(samples, batch_size=1)
    briit = bridatagen.flow(samples, batch_size=1)
    zoomit = zoomdatagen.flow(samples, batch_size=1)

    #-----------------------
    #Horizontal Augmentation
    #-----------------------

    # generate batch of images
    batch = horit.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    #Saveout
    save_img('haug_'+str(i)+'.jpg', image)

    #-----------------------
    #Vertical Augmentation
    #-----------------------

    # generate batch of images
    batch = vertit.next()
```

```
#convert to unsigned integers for viewing
image = batch[0].astype('uint8')
#Saveout
save_img('vaug_'+str(i)+'.jpg', image)

#-----------------------
#Rotate Augmentation
#-----------------------

# generate batch of images
batch = rotit.next()
#convert to unsigned integers for viewing
image = batch[0].astype('uint8')
#Saveout
save_img('raug_'+str(i)+'.jpg', image)

#-----------------------
#Bright Augmentation
#-----------------------

# generate batch of images
batch = briit.next()
#convert to unsigned integers for viewing
image = batch[0].astype('uint8')
#Saveout
save_img('baug_'+str(i)+'.jpg', image)

#-----------------------
#Zoom Augmentation
#-----------------------

# generate batch of images
batch = zoomit.next()
# convert to unsigned integers for viewing
image = batch[0].astype('uint8')
#Saveout
save_img('zaug_'+str(i)+'.jpg', image)

print('Augmented..')
```