

Student's Name	Lin Yixiang Gareth	Admin No	232776E
Course	Diploma in Infocomm & Security	Acad / Sem	2024S1
Module Name	IT2656 Systems Security Project	Module Group	CS2303
Module Supervisor	Mr Elgin Lee		
Team Leader	Nixon	Team No	4

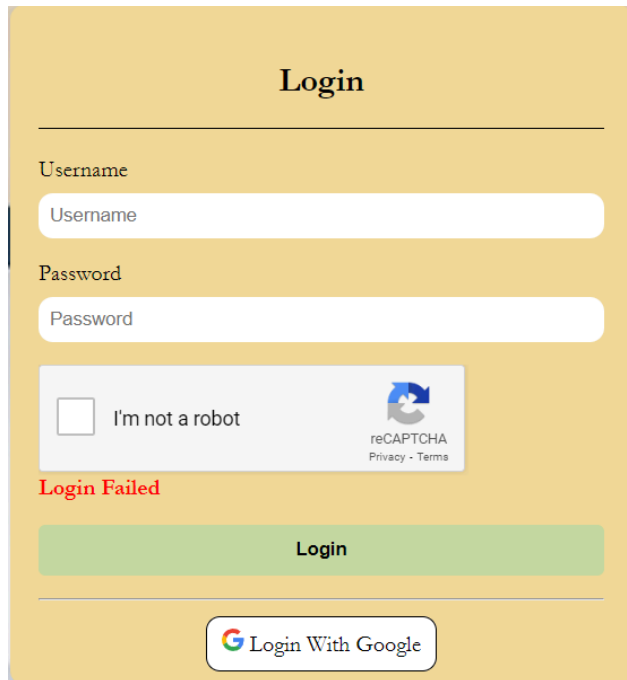
1. Individual Task (Summary List)

- Captcha
- TOTP
- Google Oauth
- Virustotal
- Password Policy

2. Security Features (Detailed Info)

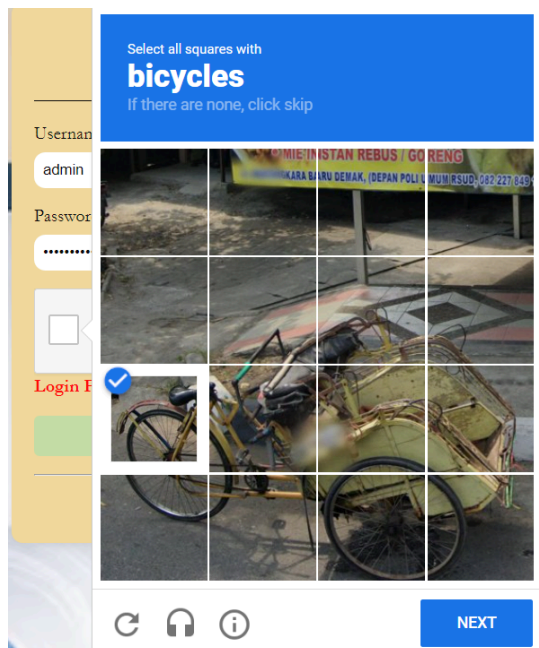
Captcha

Google Captcha version 2 has been implemented in the web application to distinguish between human users and bots, preventing automated attacks. Version 2 is used instead of version 3 due to v3 as it operates in background and scores on user behaviour to determine whether the user is human or not and sends out the Captcha challenge.



The image shows a login page with a yellow background. At the top, the word "Login" is centered. Below it, there are two input fields: "Username" and "Password". A red message "Login Failed" is displayed below the password field. To the right of the "I'm not a robot" checkbox is the reCAPTCHA logo and text "reCAPTCHA Privacy - Terms". At the bottom, there is a green "Login" button and a "Login With Google" button.

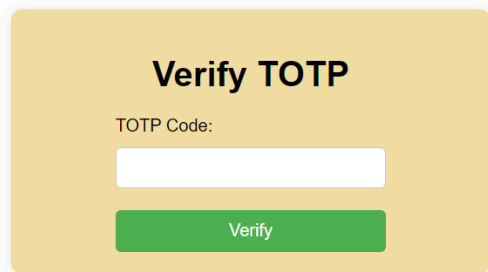
This is the login page which requires users to pass the Captcha challenge before logging in successfully. If the user does not do the Captcha, a login fail message will appear and not allow the user to log in shown in the image above. The image below is an example of the Captcha process.



The image shows a Captcha challenge interface. At the top, a blue box contains the text "Select all squares with bicycles" and "If there are none, click skip". Below this is a 4x4 grid of 16 small images. The first image in the first row is selected, indicated by a blue checkmark. The grid shows a bicycle in the background. At the bottom, there are icons for a refresh button, a volume icon, and an information icon, followed by a blue "NEXT" button.

TOTP

TOTP adds an additional layer of security by requiring users to enter a time-sensitive code generated by an authenticator app (e.g. Google Authenticator) during login. This code is only valid for a short period of 30 seconds. This mitigates the risk of unauthorised access as even with password compromised, an attacker cannot log in without code.

A yellow rectangular form with rounded corners. At the top, it says "Verify TOTP" in bold black text. Below that, it says "TOTP Code:" in a smaller font. Underneath is a white rectangular input field. At the bottom is a green rectangular button with the word "Verify" in white text.

Verify TOTP

TOTP Code:

Verify

After keying in credentials to log in, users will be prompted to key TOTP code before logging in successfully as shown in image above.

A screen with a dark blue header bar at the top that says "TOTP SETUP" in white serif font. The main background is light pink. In the center, there is a large QR code. Above the QR code, it says "Scan this QR code with your TOTP app:". Below the QR code, it says "...or use this code: PIHH4QG37LMY2EOHXY6TOB37YBVZLN7&issuer". At the very bottom, there is a small white input field with the placeholder text "Enter New Password" and a small white button with the text "Submit".

TOTP SETUP

Scan this QR code with your TOTP app:



...or use this code: PIHH4QG37LMY2EOHXY6TOB37YBVZLN7&issuer

Enter New Password Submit

To set up the TOTP, after an admin creates an account for the user, when it's the user's first time logging in, they will be prompted to set up TOTP as shown in the image above with either a QR code or a code with authenticator app.

VirusTotal

This VirusTotal API is used to scan files or URLs for malware before they are uploaded or accessed within the application. It provides an additional layer of security by preventing the distribution of malicious content. This feature ensures that all uploaded files are free of known threats, protecting users and the system from potential malware infections.

It is implemented in file upload in the user's profile page which can be accessed through the 'man' icon in navbar as shown in image below.



At the profile page, there is a file upload for users to upload images to update their profile image as shown in the image below.

Profile Page

Account Details



Update Image

Choose file No file chosen

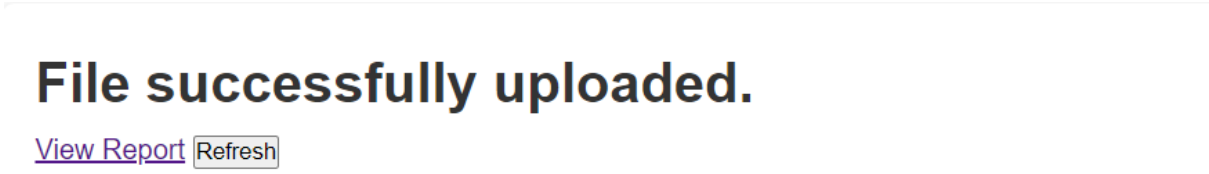
Upload

Malicious Upload

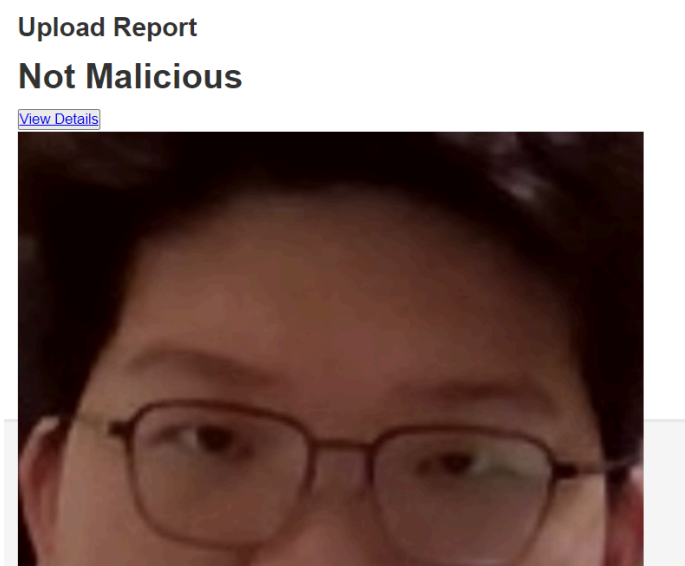
Username: admin

Email: mesphistopheles4@gmail.com

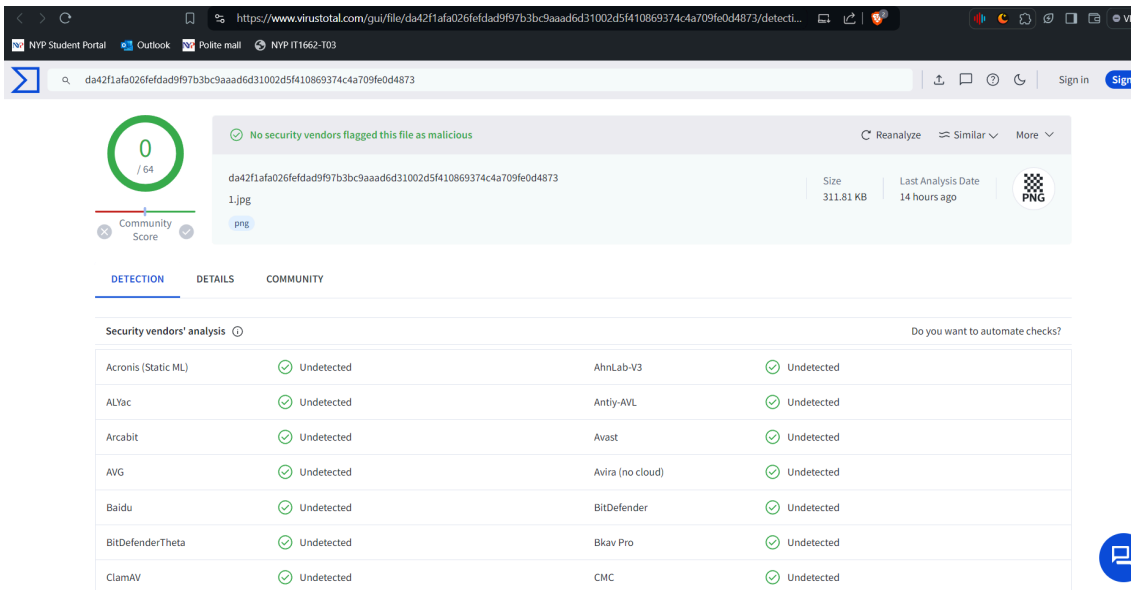
If the file is not malicious, the message as shown in image below will appear, and users will be able to view the report. However, if a user uploads an image that has not been scanned by virustotal before, it would take awhile before the report would be available as it needs time to scan, therefore there is a refresh button to click.



Once it is successful, the screen will display the following as shown in the image below, stating it is not malicious, a “View Details” button and a preview of the image file uploaded.



When the “View Details” button is clicked, user will be sent to VirusTotal Page where they can see the file scan of the uploaded file.



However, if there is a malicious file uploaded, the screen will display the following as shown in the image below, stating it may be malicious instead

There is something wrong with the file. It may be malicious.

[View Report](#) [Refresh](#)

58

74

Community Score

58/74 security vendors flagged this file as malicious

ReanalyzeSimilarMore

5e61d689d51d00a487bf13d72c0b3c55f8e53d5e33ecbf1fc2cf1ddcf01bac5e

Size816.37 KB

Last Analysis Date7 days ago

EXE

peexe

malware

overlay

executes dropped file

self delete

DETECTION

DETAILS

RELATIONS

BEHAVIOR

COMMUNITY13

Popular threat labeltrojan.msil/dcrat

Threat categoriestrojan

Family labelsmsil dcrat coins

Security vendors' analysis


Do you want to automate checks?

Alibaba	TrojanPSW:MSIL/DCRat.9d35fe73	AliCloud	Trojan[stealer]:MSIL/DCRat.MR8PHU
ALYac	Trojan.GenericKD.73501452	Arcabit	Trojan.Generic.D4618B0C
Avast	Win32:CrypterX-gen [Trj]	AVG	Win32:CrypterX-gen [Trj]
Avira (no cloud)	VBS/Runner.VPG	BitDefender	Trojan.GenericKD.73501452
BitDefenderTheta	Gen:NN.ZemsiIF.36810.Fm1@aaaUaze	Bkav Pro	W32.Common.EF1B7CF0

Upload Report

Malicious

[View Details](#)

Preview will not show if file is malicious

Password Policy

A strong password policy is crucial to protect user accounts from unauthorised access. The password policy application includes the following guidelines:

- Minimum Length: Passwords must be at least 8 characters long
- Complexity: Passwords must include a combination of upper and lower-case letters, numbers.

Enter Account Details Below

Choose file

No file chosen

Password does not include int

Register

Below is the validation message for password not including uppercase

Password does not include uppercase

Below is the validation message for password not including lowercase

Password does not include Lowercase

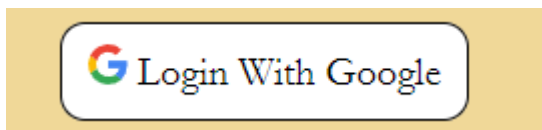
Below is the validation message for password not meeting minimum length of 8 characters

Password does not meet length

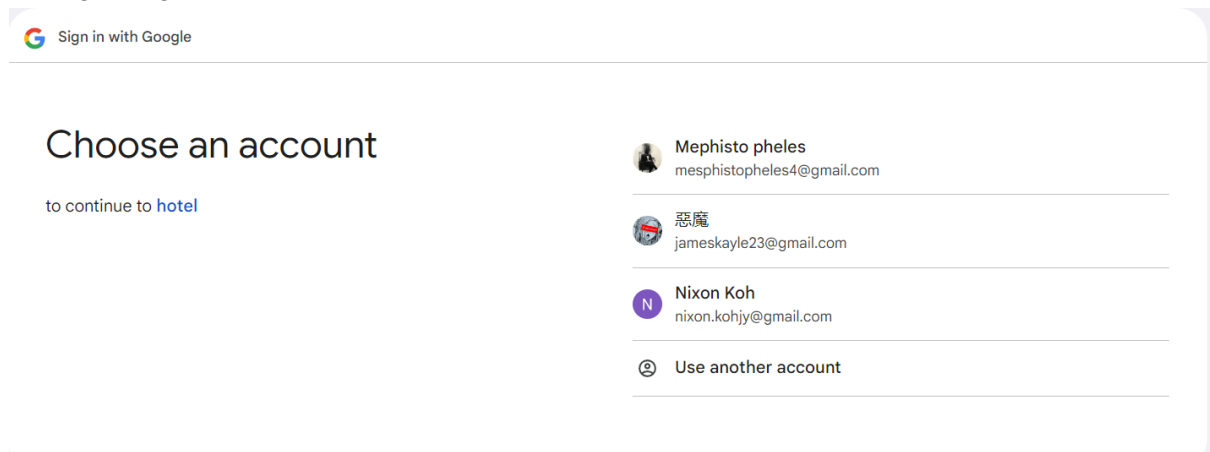
Google Oauth

Google Oauth has been implemented to provide secure and convenient authentication for users by allowing them to log in their Google accounts. Using Google OAuth enhances security by leveraging Google's robust authentication mechanism, including multi-factor authentication and secure token exchange

Users can log in with Google OAuth through clicking the button in the login page as shown in image below.



Users will then be asked to choose which gmail account to use as shown in image below. Only those accounts that have been registered can be used while those accounts that have not been registered will not have access to the web application contents and be sent back to the login page.



Student's Name	Kirsty Chan Yeng Yan	Admin No	232093Z
Course	Diploma in Infocomm & Security	Acad / Sem	2024S1
Module Name	IT2656 Systems Security Project	Module Group	CS2303
Module Supervisor	Mr Elgin Lee		
Team Leader	Nixon	Team No	4

1. Individual Task (Summary List)

- Risk Based Authentication
- Hashing
- CSRF Token
- Email Notification

2. Security Features (Detailed Info)

Risk Based Authentication

RBA assesses the contextual risk of each login attempt, such as the user's location, device, and behaviour, to determine the appropriate authentication steps. to see if it differs from the normal. It will give a score based on the risk, the higher the score the lesser the risk as it assesses the confidence of the login. The RBA uses a random forest algorithm for machine learning; it can be trained when the company decides on a more personalised algorithm. If the risk is above the threshold, it will redirect the user to the facial recognition page, if not, it

will direct the user to the TOTP page. It will then send email notifications informing the owner of the user account of the log in.

Login data set used

The RBA dataset consists of synthesised login data from over 33 million attempts and 3.3 million users of a Norwegian online service. Collected between February 2020 and February 2021, the dataset is designed to support research on Risk-Based Authentication (RBA). The dataset is generic and used to evaluate and improve RBA algorithms under real-world conditions.

(<https://www.kaggle.com/datasets/dasgroup/rba-dataset#table-of-contents>)

Risk Score

```
127.0.0.1 - - [13/Aug/2024 11:24:15] "GET /favicon.ico HTTP/1.1" 200 -
admin Password123
127.0.0.1 - - [13/Aug/2024 11:24:29] "POST /login HTTP/1.1" 302 -
Risk Factor : 0.71
1
```

```
# Train the model using multithreading
def train_model(X_train, y_train):
    model = RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1) # Use all available cores
    model.fit(X_train, y_train)
    return model

with concurrent.futures.ThreadPoolExecutor() as executor:
    future = executor.submit(train_model, X_train, y_train)
    model = future.result()
```

Training Model

Redirect to facial recognition or TOTP

```
new_login = pd.DataFrame(frame)

prediction = predict_risk(new_login)[0][0]
print("Risk Factor : ",prediction)
print(idno)

#send_email(
#recipient_email=db.select_1(f"SELECT EMAIL FROM USER WHERE ID={idno}")[0],
#subject='Account login',
#body=f""New Login detected; Username: {db.select_1(f"SELECT name FROM USER WHERE ID={idno}")[0]} risk: {prediction} timestamp: {datetime.now()}""
#)

if prediction < 0: #Facial Recognition
    return redirect("/face")
elif prediction <= 1: #totp
    return redirect("/totp")
```

Risk based authentication

```

@app.route("/risk")
def rba():
    global idno
    ua = request.headers.get('User-Agent')
    r = requests.get("https://ipinfo.io/").json()
    # ua parser is above 0.15 (currently using 0.18) which supports user_agent_parser.Parse() which caches UA
    # improving performance for real world datasets
    parsed = user_agent_parser.Parse(ua)
    frame = {
        "Login Timestamp": [datetime.now()],
        'User ID': [idno],
        'Round-Trip Time [ms]': [150],
        'IP Address': r["ip"],
        'Country': [r["country"]],
        'Region': [r["country"]],
        'City': [r["city"]],
        'ASN': [29],
        'User Agent String': [request.headers.get("User-Agent")],
        'Browser Name and Version': [f"{parsed['user_agent']['family']} {parsed['user_agent']['major']}.{parsed['user_agent']['minor']}.{parsed['user_agent']['patch']}"],
        'OS Name and Version': [f"{parsed['os']['family']} {parsed['os']['major']}.{parsed['os']['minor']}.{parsed['os']['patch']}"],
        'Device Type': ['Desktop'],
        'Login Successful': [False]
    }

```

These are the variables taken into account, we used ipinfo.io for country, ip addresses. And the User agent string from the browser to learn information about the device. This information can be recorded to log attacks.

CSRF Token

A random token is assigned, in order to determine if the HTTP request is legitimately generated via the application's user interface. There will be a unique CSRF token assigned for every user session. These tokens are inserted within hidden parameters of HTML forms related to critical server-side operations. They are then sent to client browsers.

CSRF Token

The image displays a web application interface on the left and a browser's developer console on the right.

Web Application Interface:

- Emergency Section:** A pink box with the title "Emergency" and the text "Most recent issue a guest has faced: ...".
- Enter Account Details Below:** A form with input fields for Username, Password, Email, and Power. There is also a file upload section with a "Choose file" button and "No file chosen" text.

Browser Developer Console:

- The **Network** tab is active, showing a list of requests. The **register** request is selected.
- The **Form Data** tab is open for the selected request, showing the following data:
 - username: test123
 - password: Password123
 - email: hehe@gmail.com
 - power: 12
 - files: (binary)
 - csrf_token: ImE4NDM2MDVmhZQ1YzQ1N2QwNmNmNmMxhJezNM3kYzF1MGZjNm1wMTI1ZrScg.0zhqTAT7KV4auATbnQHP2XX5LA

```
#App settings
app = Flask(__name__) # Initialises the app
app.secret_key = "".join(random.choices(string.ascii_lowercase+string.ascii_uppercase+string.digits, k=10)) # k = size of secret key
print("Secret Key : ", app.secret_key)
csrf = CSRFProtect(app) #CSRF token
csrf.init_app(app)
print("CSRF Token Loaded.")
app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(minutes=10) #session timeout
print("Session Timeout Loaded : ", app.config['PERMANENT_SESSION_LIFETIME'])
```

Password Salting and Hashing

The passwords are salted and hashed to prevent attackers from guessing the information. Hashing is a one-way function where data is mapped to a fixed-length value. Salting is an additional step during hashing, typically seen in association to hashed passwords, that adds an additional value to the end of the password that changes the hash value produced.

id	name	password	email	power	IMGPATH	TOTPSECRET
1	admin	\$2b\$12\$meBogsBJ6D1lqzHZ1DGSKeSdsMhD.4Y...	mesphistopheles4@gmail.com	0	/static/uploads/kirsty.jpg	EXN4ZJ6GD3I3SDI6WYR56FGQDJP2SLGL
2	gareth	\$2b\$12\$CE7NqOLWkDJYjVTetO5SK.ZN/0Bq118...	gareth@gmail.com	12	stored/1.jpg	5CDHDTTUPH6JMIQBNC4QDIDTHR76JGVO
3	garethlin	\$2b\$12\$17PXMLz/izjvkYdIPm2PWOUi9gu0.7.cd...	garethlin44@gmail.com	12	/static/uploads/kirsty.jpg	ND3NEWXGSNZ34FV4L2Y6DCMVILGGCCD7

```
@app.route("/login",methods=["POST","GET"] )
def login():
    msg = ""
    form = validation.LoginForm(request.form)
    if request.method == "POST" and form.validate():
        salt = bcrypt.gensalt()
        pw = 'admin'.encode('utf-8')
        pw = bcrypt.hashpw(pw,salt)

        username = form.username.data
        password = form.password.data

        print(username, password) #admin admin
        # Password Hashing + Salting
        # salting
        account = db.select_1(f"SELECT * FROM USER WHERE NAME='{username}'")
        if account is not None:
            res = bcrypt.checkpw(password.encode('utf-8'), account[2].encode('utf-8'))
            if res:
                global idno
                idno = account[0]
                if account[-2] == 1:
                    return render_template('loginpage.html', msg="ACCOUNT LOCKED CONTACT ADMIN.", form=form)
```

Email Notification

Email notification, this makes it so that there will be an email notification sent to the account registered that there was an email login to their account. Using SMTPlib and port 465 for port 465 is used for implicit TLS, and using SSL to keep the connection secure and to safeguard sensitive information.

Using Python's built-in smtplib and email.message libraries simplifies the process of sending emails from the website. When combined with SSL (Secure Sockets Layer), it ensures that emails, including sensitive data like password resets or notifications, are securely transmitted. This is crucial in protecting user data and maintaining the integrity of the website's communication channels.

Implementing this email-sending method within our website can significantly enhance its security posture. Additionally, the use of SSL for secure email transmission helps protect user data from being intercepted or tampered with during transmission.

Overall, this not only strengthens the security of our website but also improves user trust by ensuring that their interactions and data are protected.

Account login

kirstychan83@gmail.com

to me

A user tried to log in to your account, please verify that it is you.

If it is an unauthorized login, please contact the administrator at 1

12 Aug 2024, 12:41 (22 hours ago)

kirstychan83@gmail.com

to me

New login detected, Username: admin risk: 0.71 timestamp: 2024-08-12 22:08:29.755566

12 Aug 2024, 22:08 (13 hours ago)

kirstychan83@gmail.com

to me

A user tried to log in to your account, please verify that it is you.

If it is an unauthorized login, please contact the administrator at 1

12 Aug 2024, 23:19 (12 hours ago)

```
def send_email(recipient_email, subject, body, sender_email='kirstychan83@gmail.com', sender_password='kkn xvvc ufld cltf'):
    em = EmailMessage()
    em['From'] = sender_email
    em['To'] = recipient_email
    em['Subject'] = subject
    em.set_content(body)

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL('smtp.gmail.com', 465, context=context) as smtp:
        smtp.login(sender_email, sender_password)
        smtp.sendmail(sender_email, recipient_email, em.as_string())

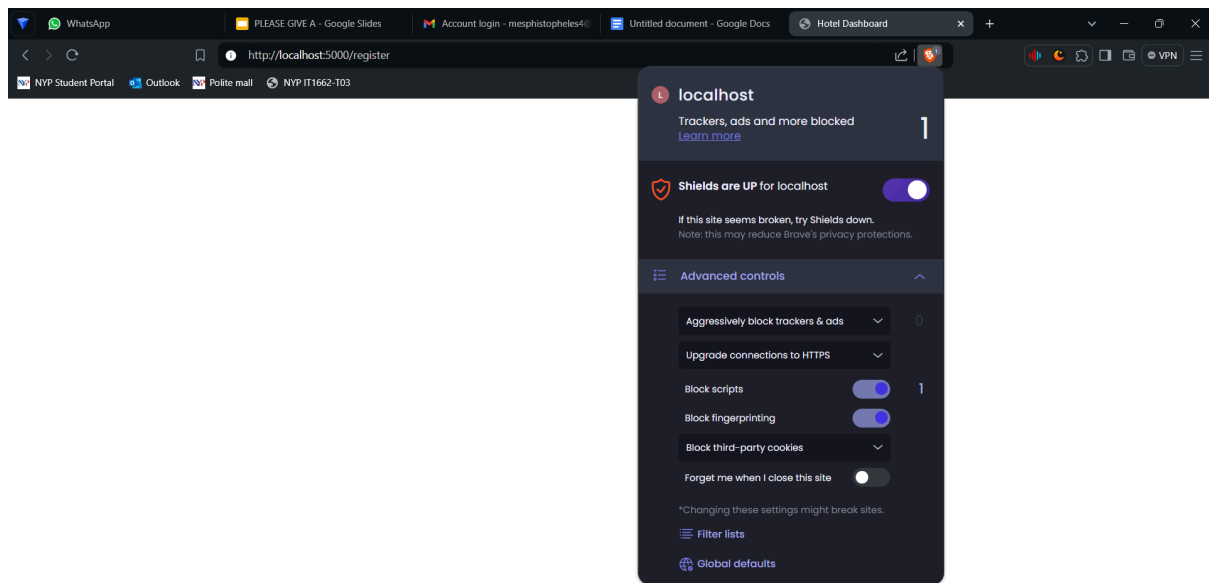
app.run()
print("App is Down. Goodbye.")
```

Student's Name	Leonard Tan	Admin No	232542N
Course	Diploma in Infocomm & Security	Acad / Sem	2024S1
Module Name	IT2656 Systems Security Project	Module Group	CS2303
Module	Mr Elgin Lee		

Supervisor			
Team Leader	Nixon	Team No	4

My focus for this module was improving security on the client side. And making it look cool in the process.

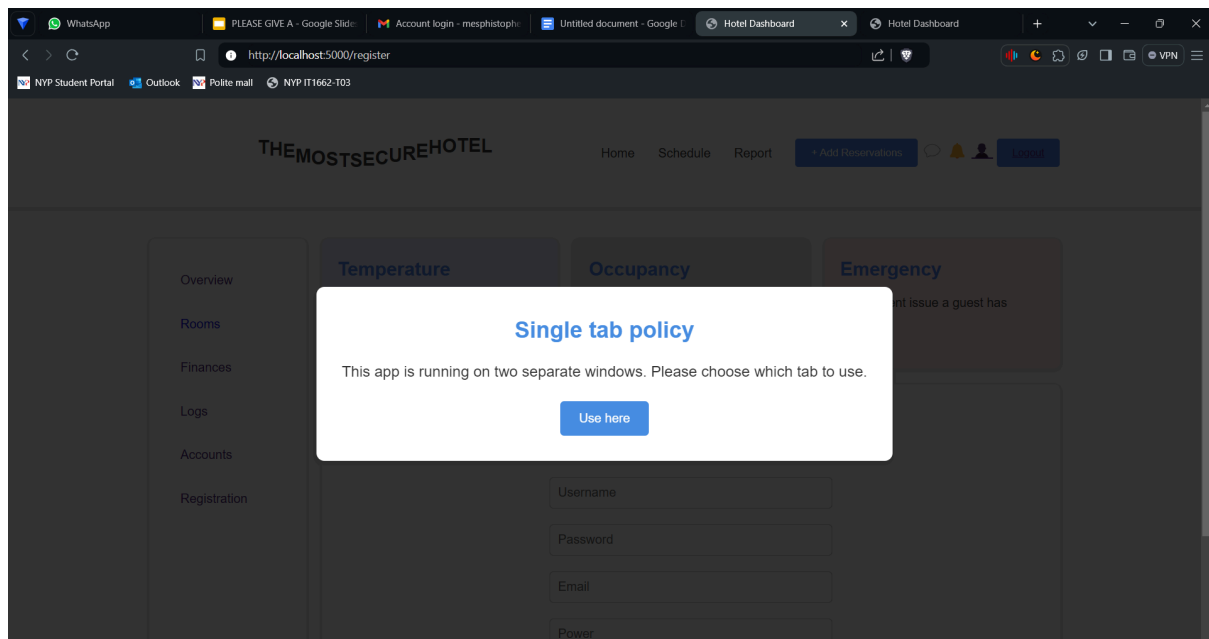
Feature 1: Noscript disables all functionality of the website.



If Javascript is detected to be blocked with the <noscript> tag, the display of all elements will be set to 'none', blocking potential attackers from using any essential features inside the site.

Feature is fairly simplistic, as the idea behind it is both to tell users that you should not access this page without javascript, as well as deter attackers from blocking our client side security features.

Feature 2: Session Management - Single Tab Policy



If you use whatsapp web, I would like you to try opening up 2 tabs at the same time.

Notice something? Whatsapp web only allows you to open up a single tab at a time. This is the feature I sought out to replicate in our module, and has been the subject of my research for 2.5 months, undergoing 20+ iterations before I ended up at what is currently implemented.

While on the surface, this feature is fairly simplistic. In theory, allowing only a single session would ensure that were an attacker to sign in while the legitimate client is using the app, it would pop up, alerting the user of a potential breach as early as possible. Having lesser tabs also reduces risk of there being an unattended session an attacker can hijack.

On the technical side, this was an absolute nightmare to program. Please send prayers to my 20 iterations.

Features:

- On the same browser, Opening up 2 instances of the tab will cause a pop up to appear. Every set time, a check will occur to detect any duplicate sessions.
- If a duplicate session is found inside the same browser, the pop up will appear and the user can choose which tab he would like to use.
- If a duplicate session is found outside the same browser, BOTH sessions will be invalidated. This is the client's cue to contact the administrator.

Feature 3: Session management

Most security features related to session management were done by me. There are a few simple session management features I included such as session timeout and user - session pairing.

Feature 4: Errors

If a user were to tamper with the url to an invalid page, it would redirect them to an error page. Prevents them from accessing information we don't want them to access via url tampering.

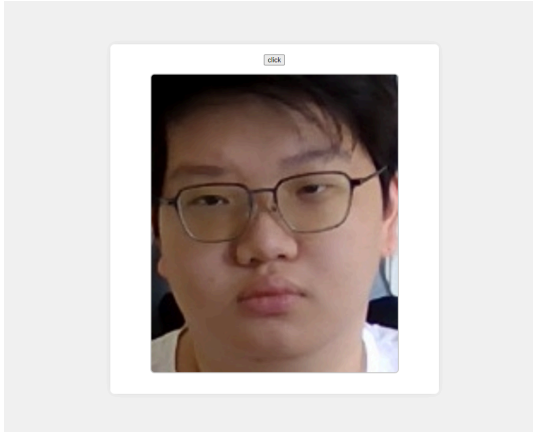
Feature 4.5 (?): Front End

I was the main developer for the front end. The home page, error page, TOTP setup page (NOT the feature!), and a few of the home page's subpages were done by me. Dunno if this counts towards anything since this is mostly a backend focused module but might as well mention it. I did not do 100% of the work for the client side, just around 80%.

Student's Name	Nixon Koh	Admin No	233510P
Course	Diploma in Infocomm & Security	Acad / Sem	2024S1
Module Name	IT2656 Systems Security Project	Module Group	CS2303
Module Supervisor	Mr Elgin Lee		
Team Leader	Nixon Koh	Team No	4

1. Face Recognition + AntiSpoofing

Face Detection (Image is locked into the face as it detects a valid face).

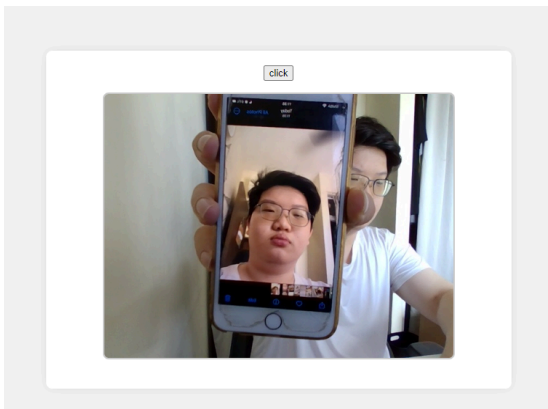


```
#processes the Frames for face
def detect_face(frame):
    #Face Detection
    (h, w) = frame.shape[:2] # Takes Height and width of image
    #Turns the frame from 0 to 255 (color code) into 0~1 maxtrix
    #Frame, resize the frame to 300 x 300, Scale factor of 1, New image size should be 300x300, Mean value image (arbitrary)
    blob = cv2.dnn.blobFromImage(cv2.resize(frame,(300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
    net.setInput(blob) #Set the blob as input image(matrix form)
    detections = net.forward() #Evaluates blob
    confidence = detections[0, 0, 0, 2]

    if confidence < 0.8: return frame # if no face is found, return frame
```

Finds the dimensions of the image, translates the image into matrices made of the hex code of each pixel (variable blob). Face detection variable predicts if a possible face is seen on the blob. Returns the confidence level. Frames returned without being resized if no possible face is detected

AntiSpoofing



Face is not locked in as it is not a valid face

```

model_test = AntiSpoofPredict(0)
image_cropper = CropImage()
image = frame
image_bbox = model_test.get_bbox(image)
prediction = np.zeros((1,3))
for model_name in os.listdir("./src/resources/anti_spoof_models"):
    h_input, w_input, _, scale = parse_model_name(model_name)
    param = {
        "org_img": image,
        "bbox": image_bbox,
        "scale": scale,
        "out_w": w_input,
        "out_h": h_input,
        "crop": True,
    }
    if scale is None:
        param["crop"] = False
    img = image_cropper.crop(**param)
    prediction += model_test.predict(img, os.path.join("./src/resources/anti_spoof_models", model_name))
label = np.argmax(prediction)
value = prediction[0][label]/2
if not (label == 1 and value >= 0.95):
    print(f"Fake : {value}")
    return frame

```

Loads the Models needed for anti spoofing. Crops the image to the required size for the model. Returns a Boolean value of real or fake face with a confidence level. Anti spoofing algorithm does not take movement into account as it processes frame by frame. Each frame will not affect the results of other frames. (anti spoofing works on video as well)

```

    return frame

box = detections[0, 0, 0, 3:7] * np.array([w, h, w, h])
(startX, startY, endX, endY) = box.astype("int")
try:
    frame = frame[(startY-10):(endY+10), (startX-10):(endX+10)]
    (h, w) = frame.shape[:2]
    r = 480 / float(h)
    dim = (int(w * r), 480)
    frame = cv2.resize(frame, dim)

```

Resizing the image if a valid face is found. If a valid face is detected, it locks in and returns the resized image to the client. Showing a zoomed in face.

Face Verification

```

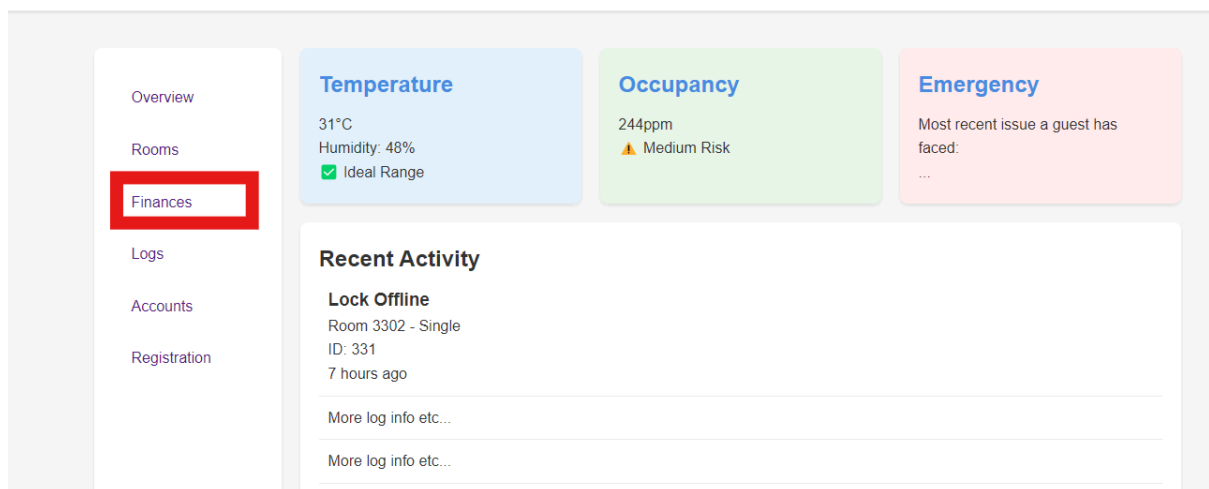
global login_req
if login_req:
    cv2.imwrite('image.jpg', frame)
    result = DeepFace.verify(img1_path="image.jpg",
                              img2_path=f"./stored/{idno}.jpg",
                              model_name="VGG-Face",
                              threshold=0.8)

    print(result)
    os.remove("image.jpg")
    if result["verified"] and result['threshold'] > 0.7:
        print("found")
        camera.release()
        login_req = -1
        return frame
    login_req = 0
    print("Not Found")

```

Takes the frame from the face detection, saves it as image.jpg. Compare the face with the id number of the user. Deletes the image.jpg after prediction. Comparison is done by a Convolutional Neural Network (CNN)
<https://github.com/serengil/deepface> Github if you are interested..

2. Honey Pot



Named and placed conspicuously. 😊

OEINBFEUNn

Enter Code To View Finances

OEINBFEUNn

Submit

Confirmation page to prevent employees that accidentally clicked on the Finance button. Acts as a social engineering test for employees that are unaware, it feedbacks on the effectiveness of the employee's training.

```
@app.route("/Finances")
def honeypot():
    db.log(idno,"HONEY POT HAS BEEN TRIGGERED")
    db.update("UPDATE USER SET ACC_LOCK=1")
    send_email(
        recipient_email=acc.get_email(),
        subject='Account login',
        body=f""The Honeypot has been triggered. Please contact your immediate supervisor immediately. Account Number : {idno}""
    )
    return redirect("/logout")
```

Honey pot locks the user out to prevent additional damages. Logs the incident and sends an email to the registered user.

3. Account Lock

View Accounts

admin



Users with higher privilege can help unlock or lock users of lower privilege. In response to incidents or users that have triggered the honeypot. This information is persistent and is recorded into the database

4. Logging

Log ID	Datetime	User	Action
1	2024-08-15	1	Loggedin : 1
2	2024-08-15	1	Loggedin : 1
3	2024-08-15	1	Loggedin : 1
4	2024-08-15	1	Loggedin : 1
5	2024-08-15	1	Loggedin : 1
6	2024-08-15	1	Loggedin : 1
7	2024-08-15	1	Loggedin : 1
8	2024-08-15	1	Loggedin : 1
9	2024-08-15	1	Loggedin : 1
10	2024-08-15	1	Loggedin : 1
11	2024-08-15	1	Loggedin : 1
12	2024-08-16	1	Loggedin : 1
13	2024-08-16	1	Loggedin : 1

Logs are created with user action. Information such as

```
def log(self, user,msg):
    self.cursor.execute("INSERT INTO logs VALUES (NULL,%s,%s,%s)",(datetime.datetime.now(), user, msg))
    self.db.commit()
```

Other contributions:

- Integrator
 - As you can see, everything was well documented and commented at the start and barely existent at the end...
- Team Leader
 - Everyone was great to work with and all pulled their weight 👍