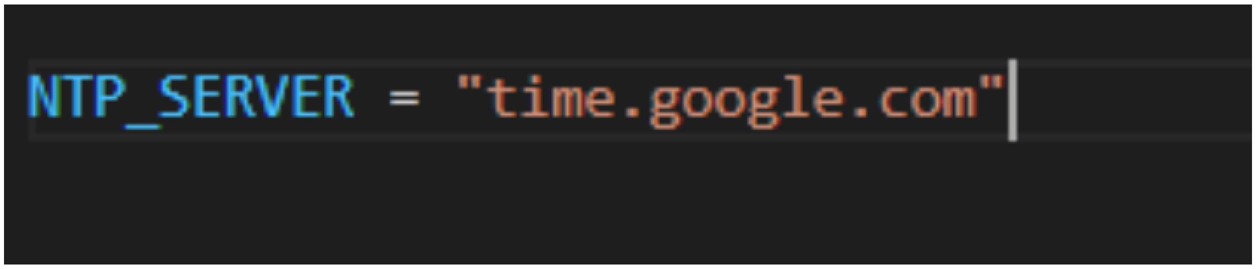
**Assignment-4**
Rahul Kumar(2020110)
Mohd Shakir(2020524)

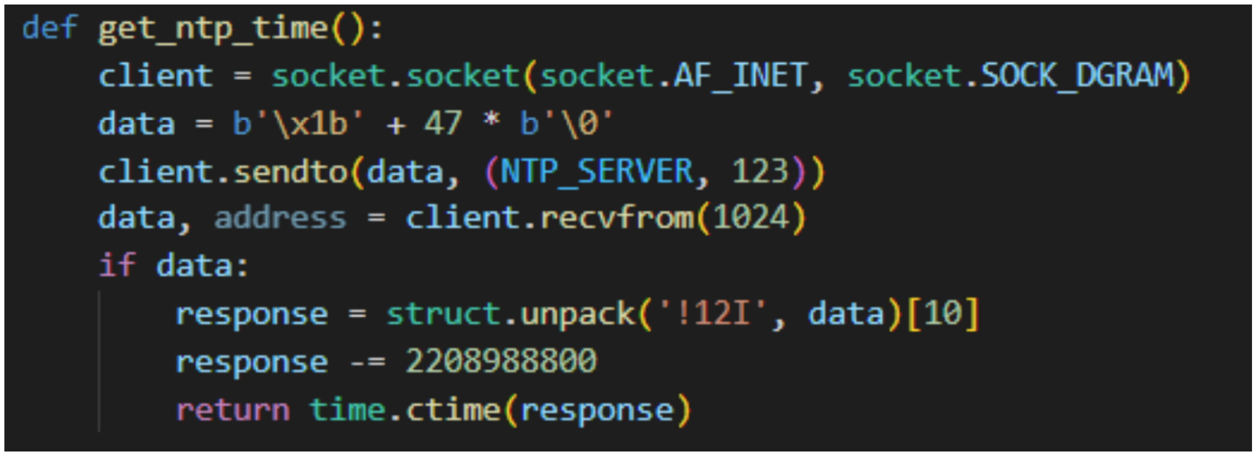
We have deployed a web server using Flask where clients can input their name, roll number, date of birth, and password. If the user is registered in our database, they will gain access to their grade card and degree certificate. Before accessing these documents, they undergo a verification process by the Director and Registrar, using public keys in the backend. Upon successful verification, users are granted permission to download both documents.

**1.How and where do you get the correct GMT date and time? Is the source reliable and the GMT date and time obtained in a secure manner? The term 'obtained' refers to security of Communication.**

**Ans.** We utilize NTP to retrieve the current date and time in seconds from a secure server, specifically "time.google.com", via a UDP socket. Subsequently, this data is converted into a human-readable format.



```
NTP_SERVER = "time.google.com"
```

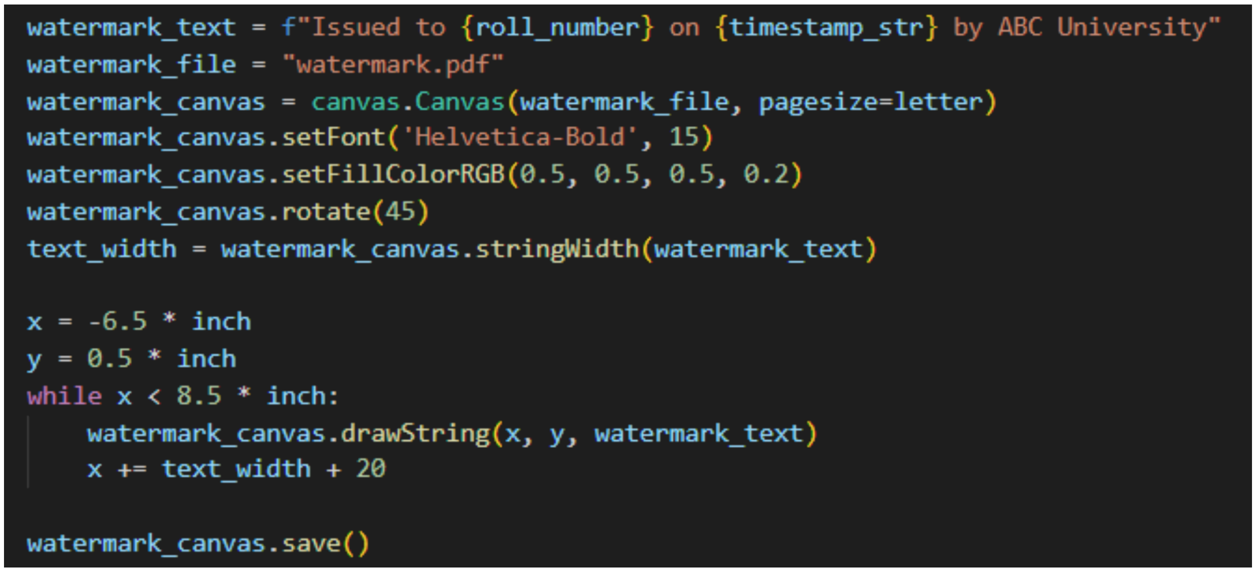


```
def get_ntp_time():
    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    data = b'\x1b' + 47 * b'\0'
    client.sendto(data, (NTP_SERVER, 123))
    data, address = client.recvfrom(1024)
    if data:
        response = struct.unpack('!12I', data)[10]
        response -= 2208988800
        return time.ctime(response)
```

**2.How do you ensure that only the graduate is able to download it (by providing information beyond the roll no, such as date of birth, home pin code, etc.?**

**Ans.** In the aforementioned query, we rely on a database that houses crucial information such as Date of Birth (D.O.B) and passwords for authentication purposes. This database is structured to securely store this sensitive data, ensuring its integrity and confidentiality. Specifically, the information is organized within a CSV (Comma-Separated Values) file format, facilitating easy access and manipulation while maintaining a standardized structure.

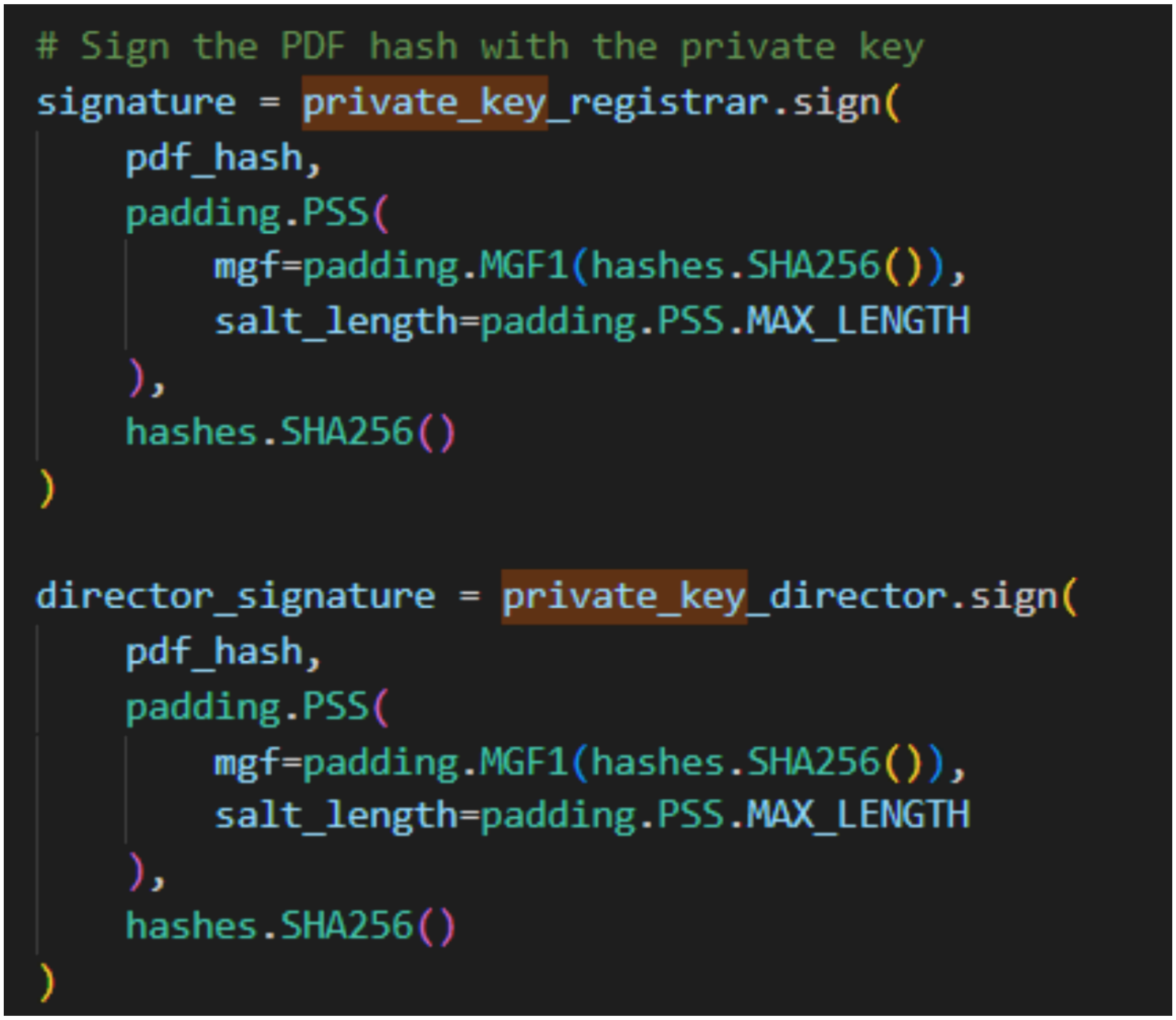
**3.Should the graduate decide to share the document with others, how can one trace the origin ofthe document (could watermarks be useful?)**
**Ans.** For this we used the watermarks you can check below :



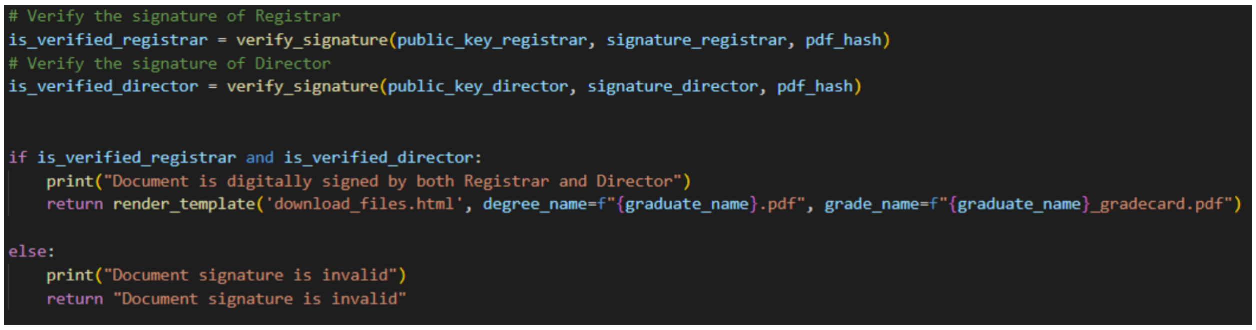
```
watermark_text = f"Issued to {roll_number} on {timestamp_str} by ABC University"
watermark_file = "watermark.pdf"
watermark_canvas = canvas.Canvas(watermark_file, pagesize=letter)
watermark_canvas.setFont('Helvetica-Bold', 15)
watermark_canvas.setFillColorRGB(0.5, 0.5, 0.5, 0.2)
watermark_canvas.rotate(45)
text_width = watermark_canvas.stringWidth(watermark_text)

x = -6.5 * inch
y = 0.5 * inch
while x < 8.5 * inch:
    watermark_canvas.drawString(x, y, watermark_text)
    x += text_width + 20

watermark_canvas.save()
```

**4.Do we need to have access to public-keys, and if so whose? And how?**
**Ans.** It's essential to have access to public keys for validating the digital signature of documents. These public keys can be acquired from a reliable key authority, like a certificate authority (CA), or via a public key infrastructure (PKI), which oversees digital certificates and keys. The web server has the capability to fetch the public keys from the university authorities and employ them to authenticate the digital signature of documents. In our system, we've stored these keys within a designated file for safekeeping.

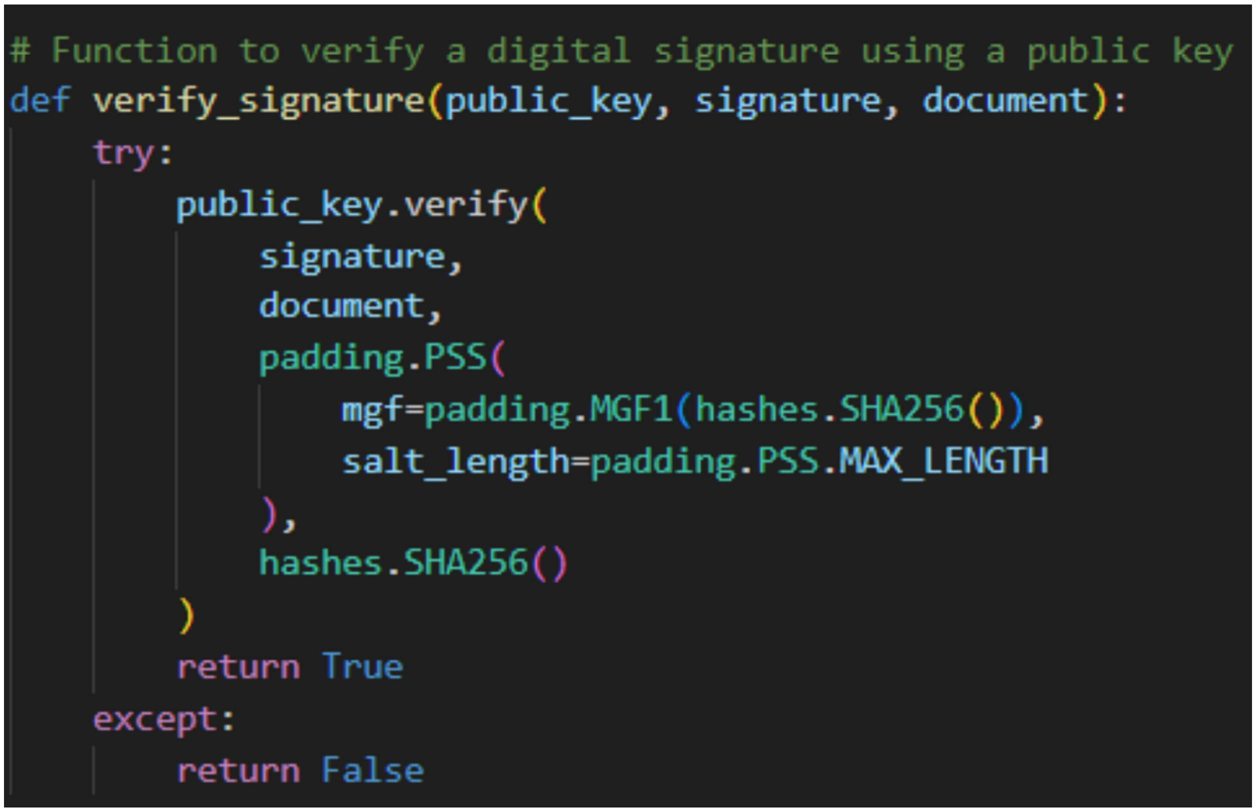
**5.20% Bonus points if you address this issue: How do you get the document to be digitally signed by two**
**persons sequentially (say the Registrar and then the Director?**
**Ans.** For the bonus segment, we engaged in a meticulous process wherein we generated not just one, but two pairs of cryptographic keys. Each document underwent a rigorous authentication procedure, being sequentially signed first by the director and subsequently by the registrar. Upon downloading these documents, we ensured their integrity and authenticity by meticulously verifying them using the corresponding public keys.



```python
# Sign the PDF hash with the private key
signature = private_key_registrar.sign(
    pdf_hash,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)

director_signature = private_key_director.sign(
    pdf_hash,
    padding.PSS(
        mgf=padding.MGF1(hashes.SHA256()),
        salt_length=padding.PSS.MAX_LENGTH
    ),
    hashes.SHA256()
)
```

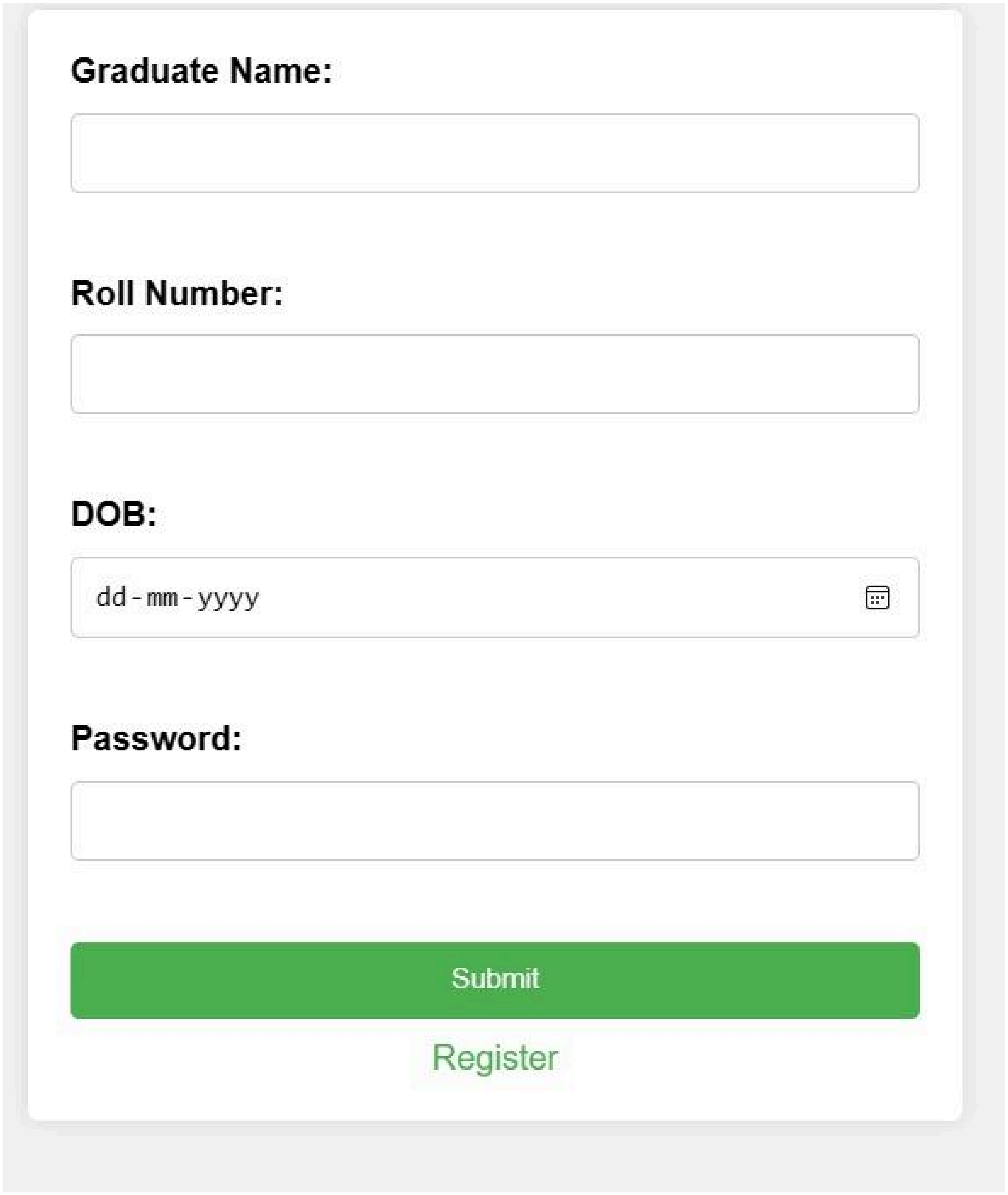


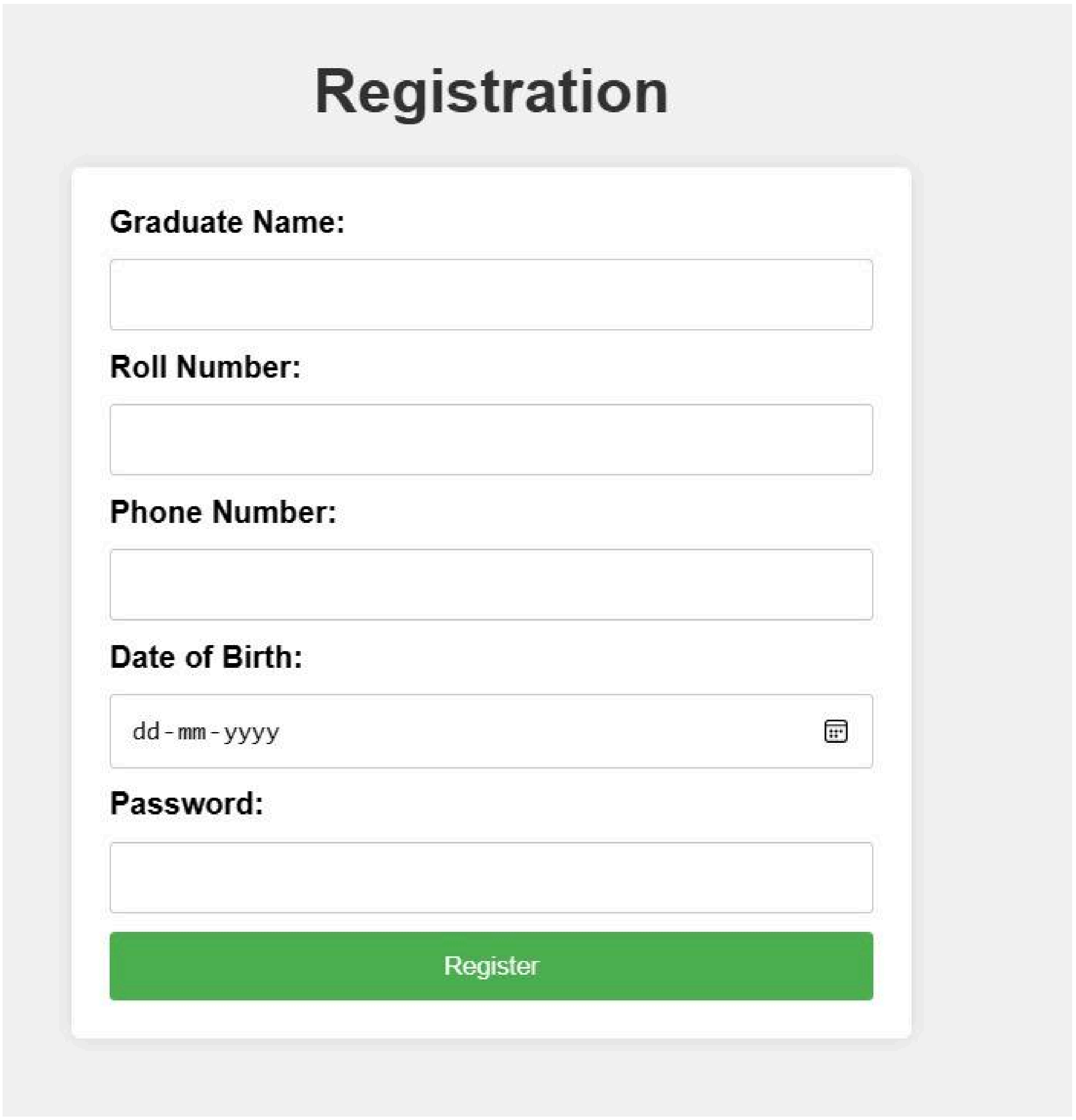
```python
# Verify the signature of Registrar
is_verified_registrar = verify_signature(public_key_registrar, signature_registrar, pdf_hash)
# Verify the signature of Director
is_verified_director = verify_signature(public_key_director, signature_director, pdf_hash)


if is_verified_registrar and is_verified_director:
    print("Document is digitally signed by both Registrar and Director")
    return render_template('download_files.html', degree_name=f"{graduate_name}.pdf", grade_name=f"{graduate_name}_gradecard.pdf")

else:
    print("Document signature is invalid")
    return "Document signature is invalid"
```



```python
# Function to verify a digital signature using a public key
def verify_signature(public_key, signature, document):
    try:
        public_key.verify(
            signature,
            document,
            padding.PSS(
                mgf=padding.MGF1(hashes.SHA256()),
                salt_length=padding.PSS.MAX_LENGTH
            ),
            hashes.SHA256()
        )
        return True
    except:
        return False
```

**Below is the Workflow implemented in the UI**


Graduate Name:
Roll Number:
DOB:
dd-mm-yyyy
Password:
Submit
Register



Registration

Graduate Name:

Roll Number:

Phone Number:

Date of Birth:

dd-mm-yyyy

Password:

Download Degree Certificate

Download Grade Certificate

Name: Abcd

Roll Number: 369

Timestamp: 2024-04-21 17:42:43.660 UTC

Name: Abcd

Roll Number: 369

Timestamp: 2024-04-21 17:42:43.694 UTC

Grade: X