Jail-Database

Authors: James Xie, Kevin Guo
Emails: hx2227@nyu.edu, kg3354@nyu.edu
Github link: Jail-Database

1. ***Database Design:***

ER-Diagram:

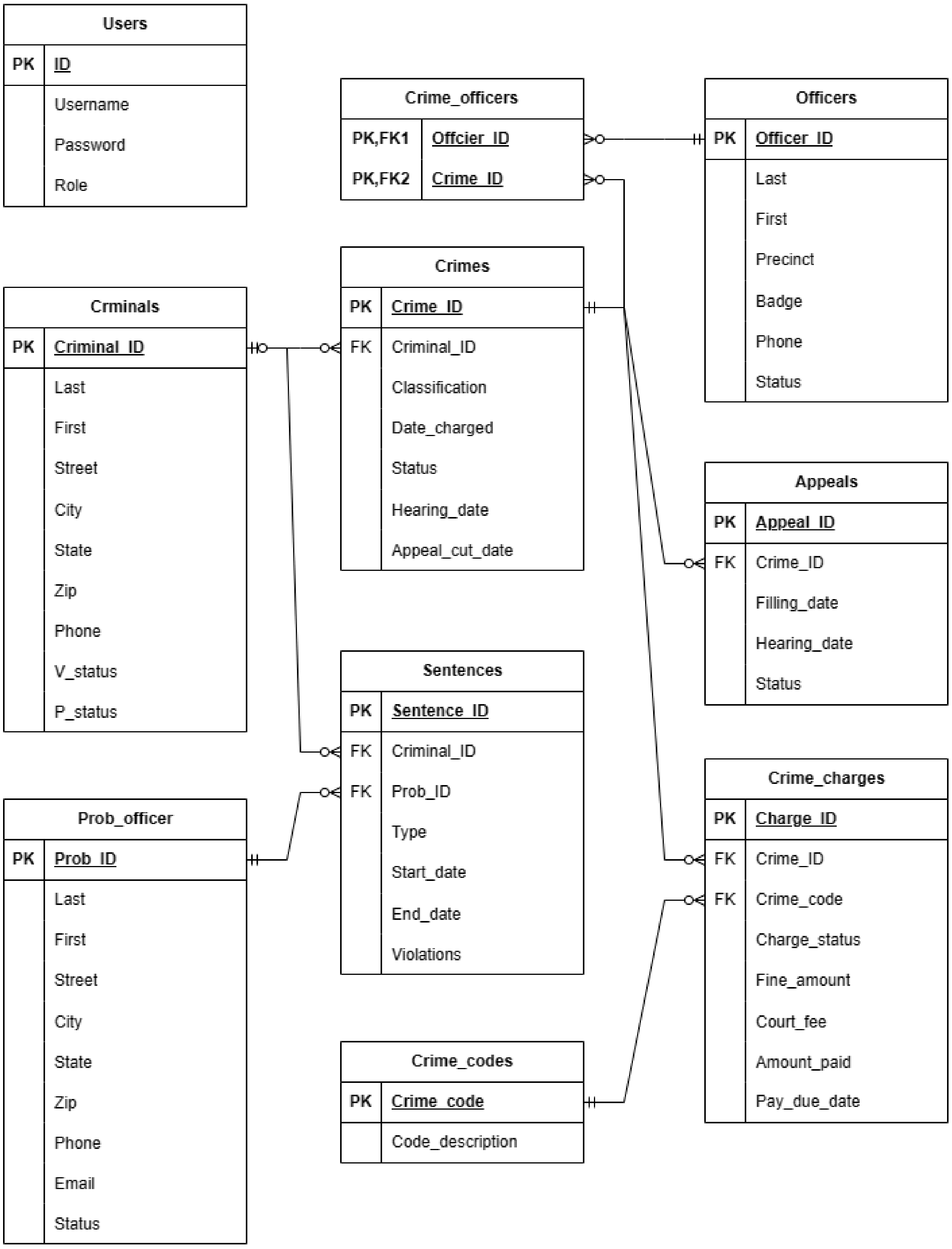
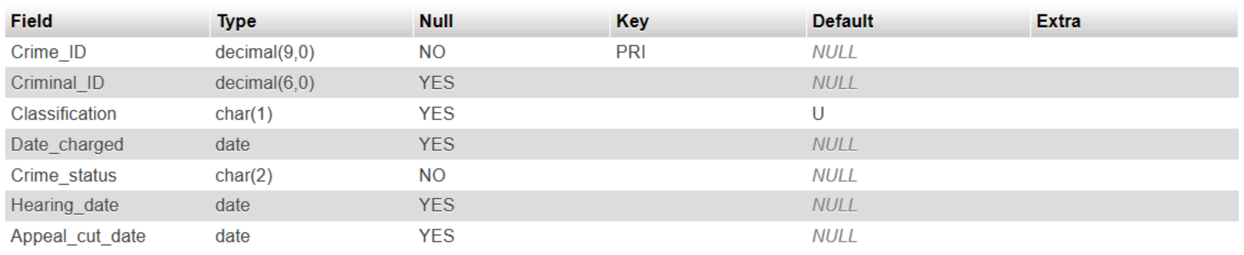


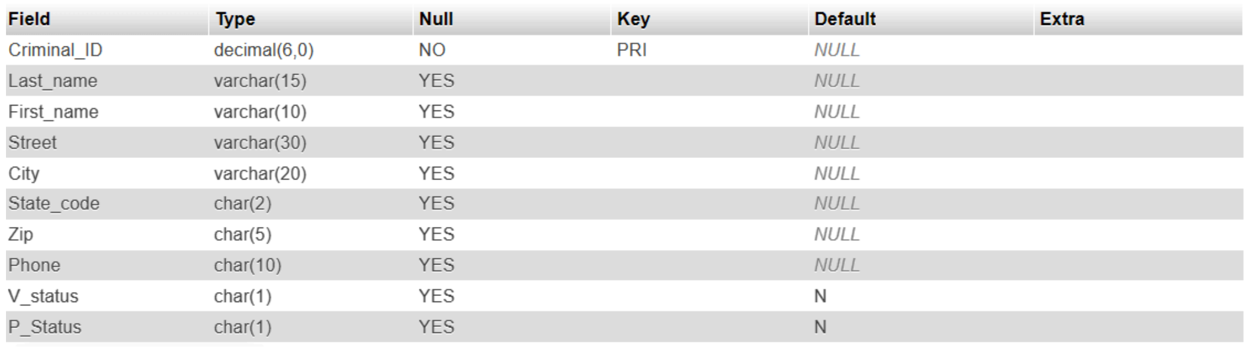
Table Schema Statements:

Crimes:



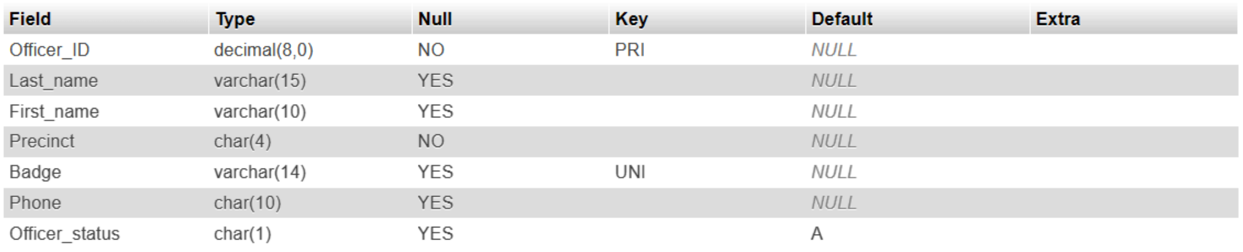
Field	Type	Null	Key	Default	Extra
Crime_ID	decimal(9,0)	NO	PRI	*NULL*	
Criminal_ID	decimal(6,0)	YES		*NULL*	
Classification	char(1)	YES		U	
Date_charged	date	YES		*NULL*	
Crime_status	char(2)	NO		*NULL*	
Hearing_date	date	YES		*NULL*	
Appeal_cut_date	date	YES		*NULL*	

Criminal:



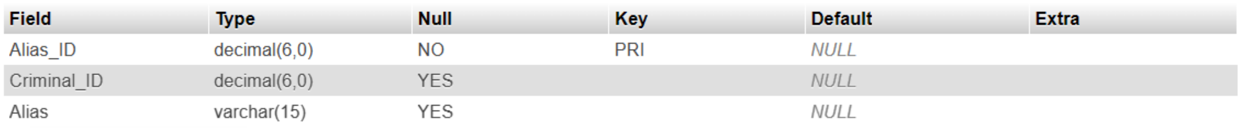
Field	Type	Null	Key	Default	Extra
Criminal_ID	decimal(6,0)	NO	PRI	*NULL*	
Last_name	varchar(15)	YES		*NULL*	
First_name	varchar(10)	YES		*NULL*	
Street	varchar(30)	YES		*NULL*	
City	varchar(20)	YES		*NULL*	
State_code	char(2)	YES		*NULL*	
Zip	char(5)	YES		*NULL*	
Phone	char(10)	YES		*NULL*	
V_status	char(1)	YES		N	
P_Status	char(1)	YES		N	

Officiers:



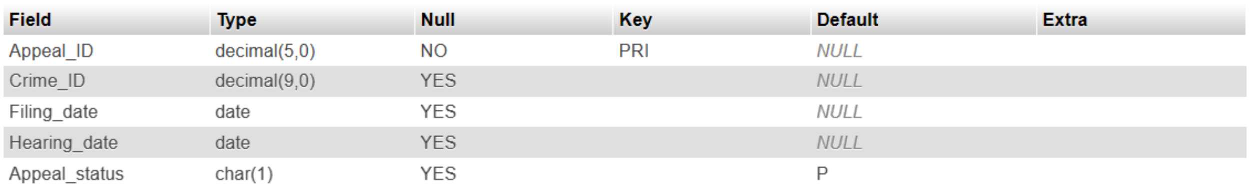
Field	Type	Null	Key	Default	Extra
Officer_ID	decimal(8,0)	NO	PRI	*NULL*	
Last_name	varchar(15)	YES		*NULL*	
First_name	varchar(10)	YES		*NULL*	
Precinct	char(4)	NO		*NULL*	
Badge	varchar(14)	YES	UNI	*NULL*	
Phone	char(10)	YES		*NULL*	
Officer_status	char(1)	YES		A	

Alias:



Field	Type	Null	Key	Default	Extra
Alias_ID	decimal(6,0)	NO	PRI	*NULL*	
Criminal_ID	decimal(6,0)	YES		*NULL*	
Alias	varchar(15)	YES		*NULL*	

Appeals:



Field	Type	Null	Key	Default	Extra
Appeal_ID	decimal(5,0)	NO	PRI	*NULL*	
Crime_ID	decimal(9,0)	YES		*NULL*	
Filing_date	date	YES		*NULL*	
Hearing_date	date	YES		*NULL*	
Appeal_status	char(1)	YES		P	

Crime_charges:

Field	Type	Null	Key	Default	Extra
Charge_ID	decimal(10,0)	NO	PRI	*NULL*	
Crime_ID	decimal(9,0)	YES		*NULL*	
Crime_code	decimal(3,0)	YES		*NULL*	
Charge_status	char(2)	YES		*NULL*	
Fine_amount	decimal(7,2)	YES		*NULL*	
Court_fee	decimal(7,2)	YES		*NULL*	
Amount_paid	decimal(7,2)	YES		*NULL*	
Pay_due_date	date	YES		*NULL*	

Crime_codes:

Field	Type	Null	Key	Default	Extra
Crime_code	decimal(3,0)	NO	PRI	*NULL*	
Code_description	varchar(30)	NO	UNI	*NULL*	

Crime_offciers:

Field	Type	Null	Key	Default	Extra
Crime_ID	decimal(9,0)	NO	PRI	*NULL*	
Officer_ID	decimal(8,0)	NO	PRI	*NULL*	

Prob_officers:

Field	Type	Null	Key	Default	Extra
Prob_ID	decimal(5,0)	NO	PRI	*NULL*	
Last_name	varchar(15)	YES		*NULL*	
First_name	varchar(10)	YES		*NULL*	
Street	varchar(30)	YES		*NULL*	
City	varchar(20)	YES		*NULL*	
State_code	char(2)	YES		*NULL*	
Zip	char(5)	YES		*NULL*	
Phone	char(10)	YES		*NULL*	
Email	varchar(30)	YES		*NULL*	
Prob_Status	char(1)	NO		*NULL*	

Sentences:

Field	Type	Null	Key	Default	Extra
Sentence_ID	decimal(6,0)	NO	PRI	*NULL*	
Criminal_ID	decimal(6,0)	YES		*NULL*	
Sentence_type	char(1)	YES		*NULL*	
Prob_ID	decimal(5,0)	YES		*NULL*	
StartDate	date	YES		*NULL*	
EndDate	date	YES		*NULL*	
Violations	decimal(3,0)	NO		*NULL*	

Users:

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	*NULL*	auto_increment
username	varchar(255)	NO	UNI	*NULL*	
password	varchar(255)	NO		*NULL*	
role	char(1)	YES		r	

2. ***Database Programming***

<u>Database:</u>
Database is hosted on phpmyadmin

<u>Application:</u>
The application is hosted on a web server.

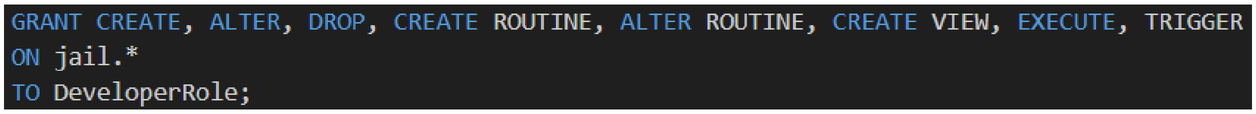
<u>Deployment Instruction:</u>
1) Create jail database in phpmyadmin
2) Import JAIL-SETUP.sql to set up the database
3) Run app.py to activate the application
4) Access the application on https://[server ip]:3000/
5) Register/Log in

<u>Advanced SQL Commands:</u>
Procedures are used to insert new rows into 'Crimes', 'Criminals', and 'Officers' tables.
The procedures also call functions that would automatically return the next PK (ID) to be assigned to the newly inserted row, since the users are responsible for inputting the ID of the new Crime/Criminal/Officer

3. ***Database Security***

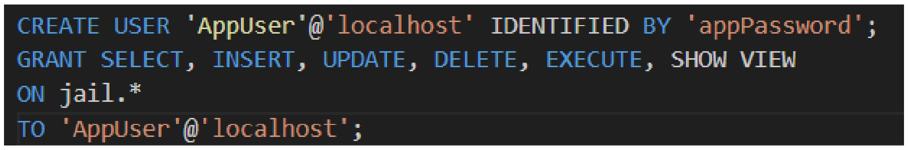
Database Security is implemented for developers and end users.

Created a DeveloperRole that only has privileges on CREATE, ALTER, DROP any tables, and CREATE or ALTER any routine and views, EXECUTE routines, and creating or dropping TRIGGER.



```
GRANT CREATE, ALTER, DROP, CREATE ROUTINE, ALTER ROUTINE, CREATE VIEW, EXECUTE, TRIGGER
ON jail.*
TO DeveloperRole;
```

The app user was granted with privileges such as SELECT, INSERT, UPDATE, DELETE, EXECUTE, and SHOW VIEW, all necessary for the application to function properly.



```
CREATE USER 'AppUser'@'localhost' IDENTIFIED BY 'appPassword';
GRANT SELECT, INSERT, UPDATE, DELETE, EXECUTE, SHOW VIEW
ON jail.*
TO 'AppUser'@'localhost';
```

4. ***Application Security***

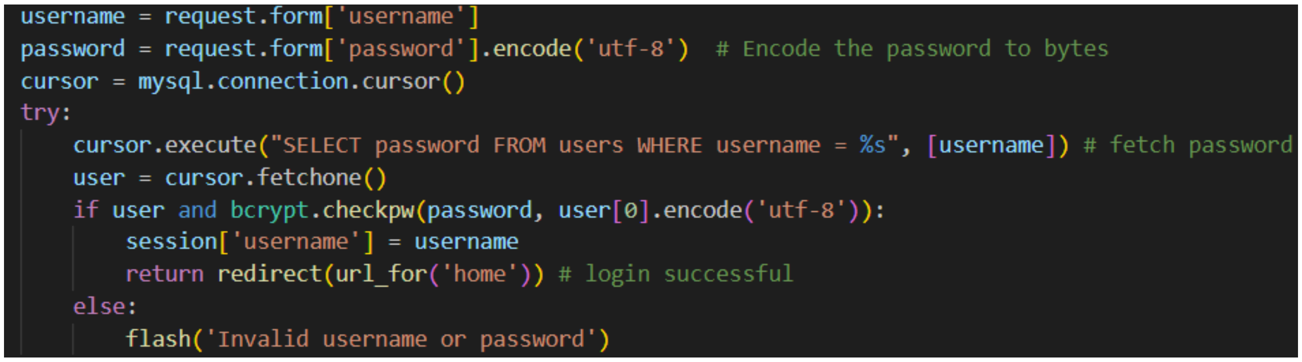
Implemented application security in user authentication, password hashing, control of execution flow, and difference in data display.
There are three types of users: Guests, Common users, and Administrators

Guests only have the access to the home page, and cannot proceed until login

Common users only have the read privileges, they can view and search for criminals/crimes/officers, but they cannot add, update, or delete any instances.
Administrators have full access to the application.

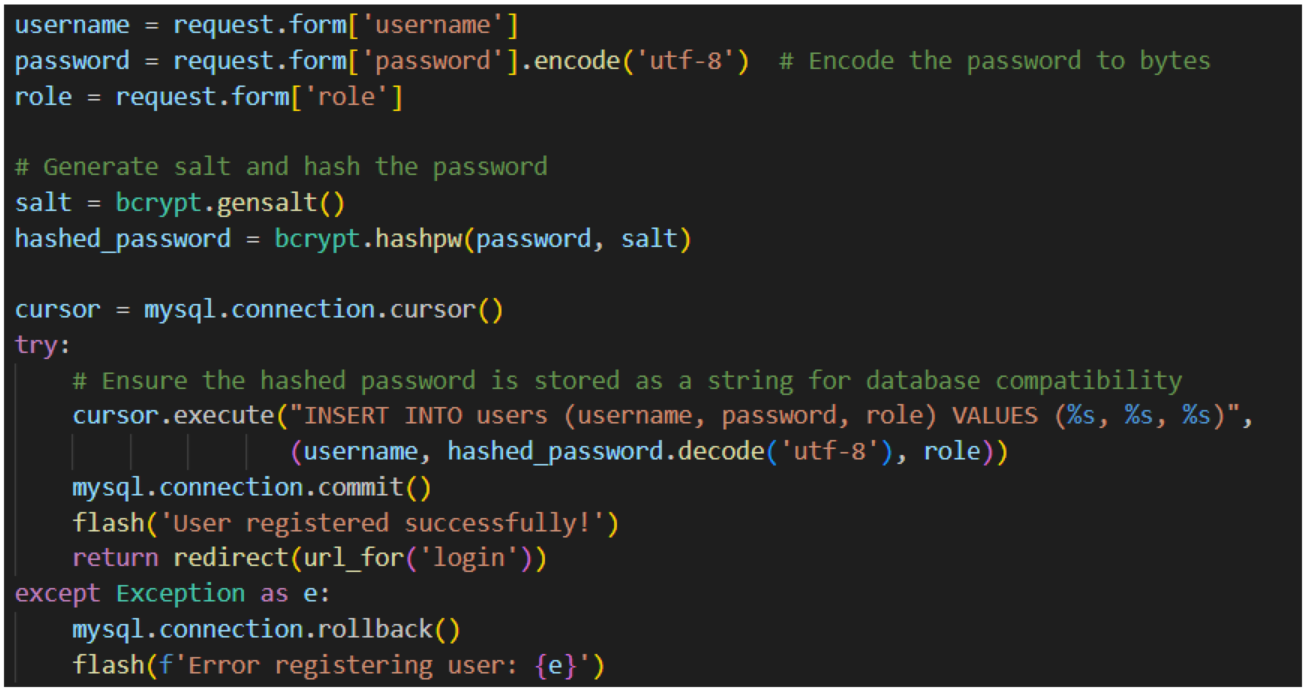
Users are assigned to role Guest by default until they login.
There is a table in the database to keep track of usernames and passwords (hashed).

All users must login before they can view any data:



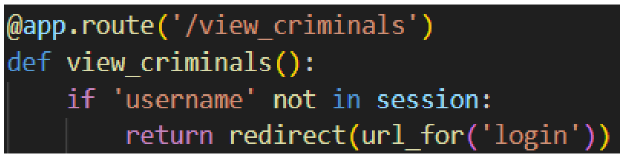
```
username = request.form['username']
password = request.form['password'].encode('utf-8')  # Encode the password to bytes
cursor = mysql.connection.cursor()
try:
    cursor.execute("SELECT password FROM users WHERE username = %s", [username]) # fetch password
    user = cursor.fetchone()
    if user and bcrypt.checkpw(password, user[0].encode('utf-8')):
        session['username'] = username
        return redirect(url_for('home')) # login successful
    else:
        flash('Invalid username or password')
```

Users without an account should first register:



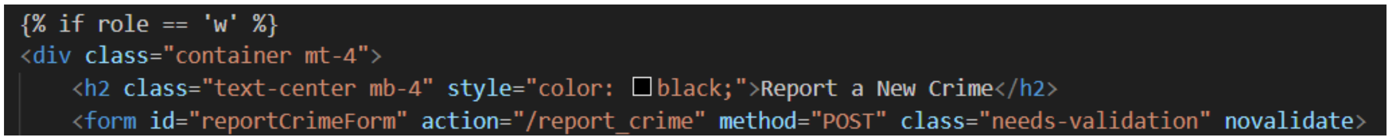
```
username = request.form['username']
password = request.form['password'].encode('utf-8')  # Encode the password to bytes
role = request.form['role']

# Generate salt and hash the password
salt = bcrypt.gensalt()
hashed_password = bcrypt.hashpw(password, salt)

cursor = mysql.connection.cursor()
try:
    # Ensure the hashed password is stored as a string for database compatibility
    cursor.execute("INSERT INTO users (username, password, role) VALUES (%s, %s, %s)",
                   (username, hashed_password.decode('utf-8'), role))
    mysql.connection.commit()
    flash('User registered successfully!')
    return redirect(url_for('login'))
except Exception as e:
    mysql.connection.rollback()
    flash(f'Error registering user: {e}')
```

All displays require user role verification, if verification fails, they will be redirected to login page:

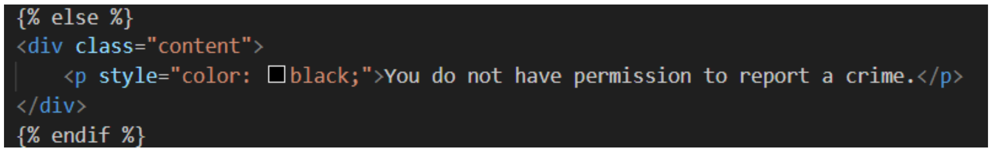


```
@app.route('/view_criminals')
def view_criminals():
    if 'username' not in session:
        return redirect(url_for('login'))
```

Certain displays and functionalities are only available to roles with write-permission (Administrators)



```
{% if role == 'w' %}
<div class="container mt-4">
    <h2 class="text-center mb-4" style="color: black;">Report a New Crime</h2>
    <form id="reportCrimeForm" action="/report_crime" method="POST" class="needs-validation" novalidate>
```



```
{% else %}
<div class="content">
    <p style="color: black;">You do not have permission to report a crime.</p>
</div>
{% endif %}
```