

Title: Build an app for people to post and receive the latest crime near USC campus

Topic: The app may allow people to post the information (time, location, description...) about latest crime incidents they experience anonymously.

Motivation:

Nowadays, crime near USC campus happens more and more. When it happens, people usually call the police to solve, and then we will receive the alert emails after several hours. Sometimes it is a little late because people who do not know the situation will still go to the places. Sometimes DPS (department of public safety) think some crimes are not very severe, so we won't receive any alerts but we still need to pay attention to these incidents. Sometimes alert emails are submerged in tons of emails. In addition, it takes time to report the incidents. Therefore, an application including the latest information is very important.

Architecture and components of your app:

I used python Flask and mySQL to build the app.

First, when the application runs, there is a home page with the latest incidents added by users. Flask looks for template files inside the templates folder, so I created a html file called index.html. I defined the basic route / and its corresponding request handler and modified the main method to return the rendered template file:

```
@app.route("/")
def main():
    return render_template('index.html')
```

Then I created a signup.html, and also added the following CSS as signup.css to the static folder. I added another method called signup to render the signup page once a request comes to /showSignUp in app.py.

```
@app.route('/showSignUp')
def showSignUp():
    return render_template('signup.html')
```

Then I build route /showSignIn with signin.html, /userHome with userHome.html and error.html, /logout, and /showAddIncidents with addIncidents.html.

```

@app.route('/showSignIn')
def showSignIn():
    return render_template('signin.html')

@app.route('/userHome', methods = ['GET', 'POST'])
def userHome():
    if session.get('user'):
        return render_template('userHome.html')
    else:
        return render_template('error.html', error = 'Unauthorized Access')

@app.route('/logout')
def logout():
    session.pop('user', None)
    return redirect('/')

@app.route('/showAddIncidents')
def showAddIncidents():
    return render_template('addIncidents.html')

```

Its data flow:

As the back end, I used MySQL. I used stored procedures for our Python application to interact with the MySQL database. So, once the table incidents has been created, I created a stored procedure called sp_createUser to sign up a user. When creating a stored procedure to create a user in the incidents table, what need to first check if a user with the same username already exists. If it exists, I need to throw an error to the user, otherwise we'll create the user in the user table.

Next, I need a server-side method for the UI to interact with the MySQL database. I created a new method called signUp and also added a route /signup. I used AJAX to post our signup data to the signup method, so I need to specify the method in the route definition. Using request, I can read the posted values. Once the values are read, I simply checked if they are valid, and for the time being and just returned a simple message.

Then I created the MySQL connection. Once the connection is created, I require a cursor to query our stored procedure. I used conn connection to create a cursor. Before calling the create user stored procedure, I made my password salted using a helper provided by Werkzeug. I used the salting module to create the hashed password. If the procedure is executed successfully, then I'll commit the changes and return the success message.

```

@app.route('/signUp',methods=['POST','GET'])
def signUp():
    # read the posted values from the UI
    _name = request.form['inputName']
    _email = request.form['inputEmail']
    _password = request.form['inputPassword']

    # validate the received values
    if _name and _email and _password:
        with closing(mysql.connect()) as conn:
            with closing(conn.cursor()) as cursor:
                _hash_password = generate_password_hash(_password)

                cursor.callproc('sp_createUser',(_name,_email,_password))
                data = cursor.fetchall()

                if len(data) == 0:
                    conn.commit()
                    return json.dumps({'message':'User created successfully !'})
                else:
                    return json.dumps({'error':str(data[0])})
    else:
        return json.dumps({'html':'<span>Enter the required fields</span>'})

```

Once having the name, email address, and password, I can simply call the MySQL stored procedure to create the new user. I used Flask-MySQL to connect with MySQL. Along with that, I included the following MySQL configurations.

```

app = Flask(__name__)
mysql = MySQL()

# MySQL configurations
app.config['MYSQL_DATABASE_USER'] = 'root'
app.config['MYSQL_DATABASE_PASSWORD'] = 'stella0902'
app.config['MYSQL_DATABASE_DB'] = 'incidents'
app.config['MYSQL_DATABASE_HOST'] = 'localhost'
mysql.init_app(app)

```

Then I created sp_validateLogin to check if we log in successfully. If so, it turns to user home page.

```

@app.route('/validateLogin',methods=['POST'])
def validateLogin():
    try:
        _username = request.form['inputEmail']
        _password = request.form['inputPassword']

        con = mysql.connect()
        cursor = con.cursor()
        cursor.callproc('sp_validateLogin',(_username,))
        data = cursor.fetchall()
        if len(data) > 0:
            if data[0][3]==_password:
                session['user'] = data[0][0]
                return redirect('/userHome')
            else:
                return render_template('error.html',error = 'Wrong Email address or Password')
        else:
            return render_template('error.html',error='Wrong Email address or Password')

    except Exception as e:
        return render_template('error.html',error=str(e))
    finally:
        cursor.close()
        con.close()

```

I created sp_addIncidents to add the incidents and sp_GetIncidentsByUser to get results. If I can add the incident successful, it will turn to the results.

```

@app.route('/addIncidents', methods = ['POST'])
def addIncidents():
    if request.method == 'POST':
        if session.get('user'):
            _user = session.get('user')
            _title = request.form['title']
            _date = request.form['date']
            _location = request.form['location']
            _vehicle = request.form['vehicle']
            _incident = request.form['incident']
            _suspect = request.form['suspect']

            with closing(mysql.connect()) as conn:
                with closing(conn.cursor()) as cursor:
                    cursor.callproc('sp_addIncidents',(_user, _title, _date, _location,
                    data = cursor.fetchall()

                    if len(data) == 0:
                        conn.commit()
                        return redirect('/getIncidents')
                    else:
                        return render_template('error.html',error = 'An error occurred!')

        else:
            return render_template('error.html',error = 'Unauthorized Access')

```

```

@app.route('/getIncidents', methods = ['GET'])
def getIncidents():
    if request.method == 'GET':
        if session.get('user'):
            _user = session.get('user')

            con = mysql.connect()
            cursor = con.cursor()
            cursor.callproc('sp_GetIncidentsByUser', (_user,))
            incidents = cursor.fetchall()

            incidents_lst = []
            for i in incidents:
                incident_dict = {
                    'Use_id': i[0],
                    'Report Offense': i[1],
                    'Date & Time of Occurrence': i[2],
                    'Location': i[3],
                    'Vehicle Description': i[4],
                    'Incident Description': i[5],
                    'Suspect Description': i[6]}
                incidents_lst.append(incident_dict)

            return json.dumps(incidents_lst)
        else:
            return render_template('error.html', error = 'Unauthorized Access')

```

Screenshots on the main functions of the app:

I pointed your browser to <http://localhost:551/> and I had the below screen:

- [Home](#)
- [Sign In](#)
- [Sign Up](#)

Then I pointed to SingUp.

Sign Up

Please fill in this form to create an account.

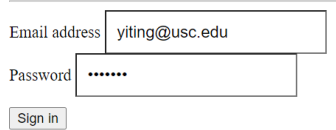
Full Name	<input type="text" value="Name"/>
Email address	<input type="text" value="Email address"/>
Password	<input type="password" value="Password"/>
<input type="button" value="Sign up"/>	

After I input the full name, email address and password, it turned to a page including a message of “User created successfully!”.

```
{"message": "User created successfully !"}
```

Then we go back to signIn.

Sign In



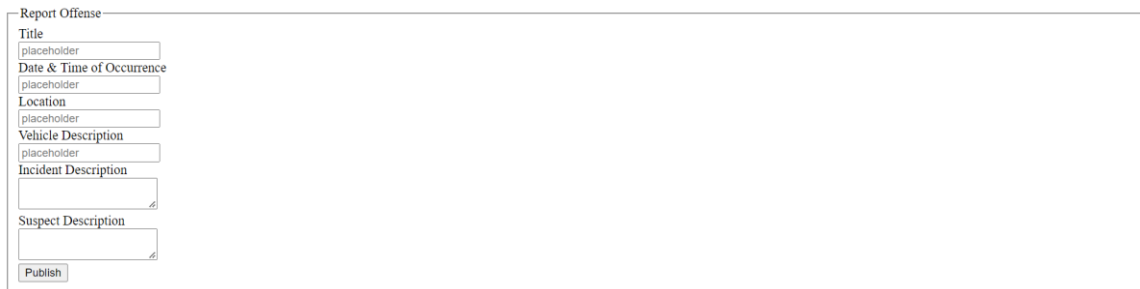
A sign-in form with two input fields and a button. The first field is labeled 'Email address' and contains the text 'yiting@usc.edu'. The second field is labeled 'Password' and contains seven dots. Below the fields is a button labeled 'Sign in'.

After I signed in successfully, it turned to the user home.

- [Logout](#)
- [Add Incidents](#)

Python Flask App

Then I clicked the button of add incidents. Then it came to that page.



A form titled 'Report Offense' with several input fields and a button. The fields are: 'Title' (placeholder), 'Date & Time of Occurrence' (placeholder), 'Location' (placeholder), 'Vehicle Description' (placeholder), 'Incident Description' (text area), and 'Suspect Description' (text area). At the bottom is a button labeled 'Publish'.

I input the information to report, click the button to publish. Then showed the results.

```
[{"Date & Time of Occurrence": "1", "Incident Description": "1", "Location": "1", "Report Offense": "1", "Suspect Description": "1", "Use_id": "yiting@usc.edu", "Vehicle Description": "1"}]
```

Reflection on learning experiences:

This time I learn how to build a web page from the beginning, which really meaningful.

Also, it is related to our class, like html, mySql.

Challenges faced:

The challenge for me is to create your own flask application, add functionality with templates and forms, and deploy my own flask application. I've never learned the knowledge before and I have to learn it by myself.

Demo: <https://drive.google.com/drive/u/0/folders/0AMdBV9TnYFePUk9PVA>

Team members: Yiting Wang

Responsibility: Temporarily all.