



Assignment 3: A Local Event Advertisement Portal

This assignment aims to help you practice databases, HTML, CSS, browser-side scripting with JavaScript, and server-side scripting with Python Flask¹. Your main task in this assignment is to develop a website for people who want to publish the local events on campus.

You are expected to use HTML to define the content of web pages, CSS to design the web pages, JavaScript for browser-side scripting for **web form validation purposes where appropriate**, and Python Flask for server-side scripting. You are also expected to use an **SQLite** database to store and manage data.

Overview

In this assignment, you are expected to develop a web portal to advertise local events on campus. These advertisements can be seen by any visitor to the website without the need to log in, but only registered users can post on the portal. There will also be admins who manage certain aspects of the website.

Functionalities

The website should have the following functionalities for people who want to use the platform:

- **Registration:**

The home page should have a link to access another page that includes a web form for people to register. They should be able to register with the following details: (1) username, (2) password, (3) full name, and (4) email address. All these details are mandatory.

The username should be unique for each person, and the password should include at least one upper case letter, one lower case letter, and one digit and its length should be at least ten. Additionally, if the email address starts with "org-", the user will be created as an admin.

If there is any problem, they will be directed to the registration page again and the problem will be shown to the user just under the form. Otherwise, they will be directed to a page which shows the confirmation message and includes a link to go back to the home page.

- **Login:**

The **home page** should have a web form to allow people to log in with their usernames and passwords. If there is any problem (such as invalid username, incorrect password, etc.), the problem will be shown to the user just under the form. Otherwise, the login form will be removed from the home page, and the following menu will be displayed for users:

[Home Page](#) | [Announced Events](#) | [My Profile](#) | [Logout](#)

or the following menu for admins:

[Home Page](#) | [Manage Societies](#) | [My Profile](#) | [Logout](#)

¹ <https://flask.palletsprojects.com/en/stable/>

- **Logout:**

After login, the home page should also have a link for people to log out from the website (see above). Once they log out, they will be directed to the home page again.

- **Advertised Events:**

After login, the home page should have a link for users to manage the events they announced. This page should include a web form where they can announce a new event with the following details: (1) name, (2) time & date (which the user can enter as any string to allow for flexibility), and (3) the societies involved with the event (Since there can be multiple, you can present all societies as check boxes), (4) description, and (5) entry fee. All the details are mandatory. You need to use radio buttons to allow the user to select if there is an entry fee (free or paid). If the event is paid, a text field should appear on the web form to enter the fee. The fee should be validated to be a number (which may or may not have decimal places).

A unique identifier will be automatically assigned to each event using AUTOINCREMENT in SQLite. If there is any problem, the problem will be shown just under the form. If an event has multiple societies involved, then all these societies will be shown separated by a comma (such as Society1,Society2).

This page will also list the previously announced events of the person who has logged in. The user should also be able to delete the announcement when s/he clicks on the corresponding Delete button.

Name	Time & Date	Societies	Description	Entry Fee	Delete
...	[Free/...]	<button>Delete</button>
...	[Free/...]	<button>Delete</button>

- **Manage Societies:**

After login, the home page should have a link for admins to manage societies. This page should include a web form where they can add new societies.

A unique identifier will be automatically assigned to each society using AUTOINCREMENT in SQLite. If there is any problem, the problem will be shown just under the form.

This page will also display a list of all existing societies, together with the number of events each society has participated in on the platform.

Society Name	The number of events
...	...

- **My Profile:**

This page displays the details of the profile of the currently logged-in user inside a web form and allows the user to change their details. All fields (except the username) will be displayed as text fields and an **Edit** button is displayed on the page. If the user wishes to modify their

details, they can perform the modifications then click the Edit button to update that information.

Username	...
Password
Name	John Smith
Email	sample@email.com
	Edit

- **Search:**

The website should also allow **all people** to search for announced events as explained below.

The **home page** should include a text box with a button which allows searching for announced events based on keywords. It will also include a combo box with all the societies to select the society to search for. There should be also an option to consider all societies.

The events which have any field that includes at least one of the keywords will be listed. Please note that it does not have to be a full match, so if the keyword is "game", and an event is named "Retro game jam", then it should be listed as well.

If a particular society is selected, the events are listed in a table, or "No events" will be displayed if no event is found. If all societies are selected, then the matching events should be categorised based on the societies as shown below, and "No events" should be displayed for a society which has no matching events. If a game belongs to multiple societies, then they should be listed under each of those societies. Note that the societies presented are just an example.

Robotics Society:

Title	Description	Genre	
...	See More
...	See More

Music Society:

Title	Description	Genre	
...	See More
...	See More

- **See Selected Events:**

When people click a link called "See More", they will be directed to a page which shows all the details of the selected events as shown below. This page should also include a link to go back to the home page.

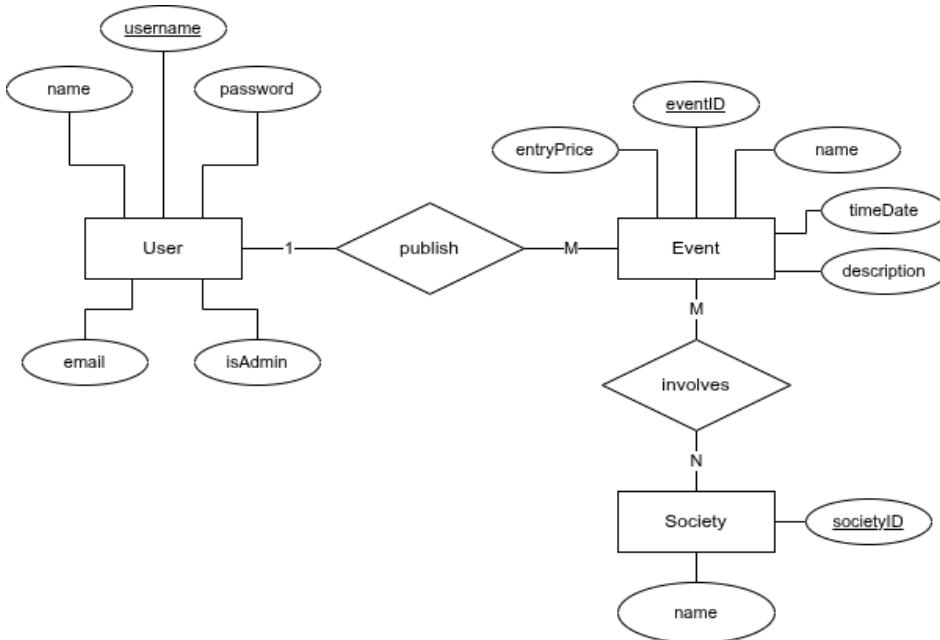
Name	...
Time & Date	...
Societies	...

Type	...
Description	...

Database

You need to create an **SQLite** database for this website based on the following entity-relationship diagram (ERD). You are expected to provide a separate Python script called **dbscript.py** to create the necessary tables.

Please note that even though eventID is a primary key for the Event entity, event name should be identified as UNIQUE in SQLite database to prevent duplicate records.



Web Deployment

In addition to the development of the website, you are requested to host your website on a publicly accessible domain. You may consult the tutorial document posted to ODTUCLASS to host your website on PythonAnywhere² and provide the website link in your ReadMe.txt.

Design Template

You are expected to use the following design template from [w3schools.com](https://www.w3schools.com). In this template, the CSS code is included in the HTML file; however, you are expected to create an external CSS file for the CSS code and include in your HTML documents. Even though this template is provided, you are expected to customise it for your website, by changing the font colours, background colours of some parts, etc.

```

<!DOCTYPE html>
<html lang="en">
<head>
<title>CSS Template</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<style>
* {

```

²<https://www.pythonanywhere.com>

```

        box-sizing: border-box;
        font-family: Arial, Helvetica, sans-serif;
    }
    body {
        margin: 0;
        font-family: Arial, Helvetica, sans-serif;
    }

    /* Style the top navigation bar */
    .topnav {
        overflow: hidden;
        background-color: #333;
    }

    /* Style the topnav links */
    .topnav a {
        float: left;
        display: block;
        color: #f2f2f2;
        text-align: center;
        padding: 14px 16px;
        text-decoration: none;
    }

    /* Change color on hover */
    .topnav a:hover {
        background-color: #ddd;
        color: black;
    }

    /* Style the content */
    .content {
        background-color: #ddd;
        padding: 10px;
        height: 200px; /* Should be removed. Only for demonstration */
    }

    /* Style the footer */
    .footer {
        background-color: #f1f1f1;
        padding: 10px;
    }
</style>
</head>
<body>

<div class="topnav">
    <a href="#">Link</a>
    <a href="#">Link</a>
    <a href="#">Link</a>
</div>

<div class="content">
    <h2>CSS Template</h2>
    <p>A topnav, content and a footer.</p>
</div>

<div class="footer">

```

```

<p>Footer</p>
</div>

</body>
</html>

```

Rules

- You need to write your program by using **Python 3.x**.
- You can **only** use all built-in functions and modules, apart from Flask.
- You need to create your CSS and JavaScript files as external files and include your HTML pages.
- You also need to create a file called **ReadMe.txt** which contains the following items. Please note that **if you do not submit ReadMe.txt, your submission will not be evaluated**.
 - Team members
 - Which version of Python 3.x you have used
 - Which operating system you have used
 - How you have worked as a team, especially how you have divided the tasks among the team members (who was responsible for what?), how you have communicated, how you have tested the program, etc.
 - The URL for the deployed website on PythonAnywhere.
- You need to put all your files (including your database file with the .db extension) into a folder that is named with your student id(s) and submit the compressed version of the folder in the **.zip** format.
- **Only one team member** should submit the assignment.
- **Code quality, modularity, efficiency, maintainability, and appropriate comments** will be part of the grading.

Grading Policy

The assignment will be graded as follows:

Grading Item	Mark (out of 100)
Database Creation Script	5
Login, Logout and Session Management (Such as, not allow people to access unauthorised pages)	10
Register	5
Announce event	5
Access events	5
Delete event	5
Manage societies	10
Manage profile	10
Search events and list them categorised by societies	20
See the details of the selected event	5
Navigation within the website	5
Form validation with JavaScript	5
Use of the CSS template and customisation of the template for the website	5
Deploy website	5