Web-based Conference Scheduling Utility Program

Content

- Program Description
- Database design
- Program Implementation
- Program Usage

➢ Program Description

This web utility program enables the users to review information of conferences and their sessions. It also helps facilitate conference registration and authentication processes for participants. Authenticated users will have the ability to post, edit, and delete their own items. Users can also upload/download documents related to the sessions and mark/unmark their favorite sessions.

➢ Database design

A. Entity Relationship Diagram (ERD):

Entities:

1. Conference (id, name, start_date)
2. Session (id, name, date, start_time, duration, highlights)
3. TypeOfSession (id, name)
4. Speaker (id, name, phone, email, address, picture)
5. Location (id, name)
6. User (id, name, phone, email, address, picture)

Relationships:

1. Between Conference and Session (1:n):
   A conference can have no or some sessions, a session instance belongs to one and only one conference.

2. Between Speaker and Session (1:n):
   A speaker can present in no or some sessions, a session is led by one and only one speaker (If there might be some speakers for one session, we will treat them as a group of speakers and this also has the representative data, such as id, name, email, and phone number).

3. Between Session and TypeOfSession (n:1):
   A session belongs to a type of session.

4. Between Session and Location (n:1):
   A session takes place at one location.

5. Between User and Conference (n:n):

   5a. A user can organize none or some conferences; a conference is organized by only one user.

   5b. A user can register none or some conferences; a conference is registered by none or some users.

6. Between User and Session (n:n):
A user can mark none or some sessions to his or her wishlist; a session is marked by none or some users.

B. Tables and Classes:

1. **Conference** (id, name, date, owner_id)
owner_id is the id of the user who organizes the conference.
2. **Session** (id, name, date, start_time, duration, highlights, conference_id, type_of_session_id, speaker_id, location_id, room)
3. **TypeOfSession** (id, name)
4. **Speaker** (id, name, phone, email, address, picture)
5. **Location** (id, name)
6. **User** (id, name, phone, email, address, picture)
7. **Register** (user_id, conference_id) (Primary key: both attributes)
8. **WishList** (user_id, session_id) (Primary key: both attributes)

Comment about the attribute room (in **Session** table):
An address of a session will include all of those in a string: building, street, city, state, zipcode, country. The room number in a building should not be included in the address string for it can be used to create a new session The user would enter a room number only if that address exists. This is helpful also for further filtering or searching for the locations.

The final database design is implemented in **database_setup.py.** (There are totally 11 classes)

Here are 3 more classes:

1. Speaker and User are the "children" of **Person** since they have some similar attributes.
2. All classes having the name attribute are the "children" of class **NameSerialize**.
3. **Img** class: store images for the sessions.

➢ Program Implementation

1. Program Functionalities:

This program is implemented and submitted for project 3 of the Udacity's Full Stack Developer Nanodegree Program. Below is the list of its current functionalities.

- Registration and authentication process (via Google and Facebook).
- Display of conferences information, along with their latest sessions. http://localhost:5000/
- Provide the users capability to log in, add, update, or delete a conference which he or she organizes.
- Selecting a specific conference will show all the sessions scheduled for that conference. If the user is not the owner (the organizer) of a conference, he or she can only read the information of the sessions. Otherwise, he/she can see the options that allow him/her to add, edit, or delete each session, upload or remove the documents for it.

- User can download documents associated with a session posted by the organizers.
- User can select which session he or she is interested and mark it to his or her wishlist. They can unmark them at any time to remove them from their wishlist.
- The application provides a JSON and XML endpoint.

2. Program Package

- **database_setup.py**: contains all classes (tables) in database.
- **db_CRUD.py**: a module contains all the functions for data: Create, Retrieve, Update, and Delete data in database.
- **project.py**: a main program. This is a Flask Application.
- /static: contains all the necessary css and js files.
- /static/document: contains all the uploaded documents other than images. Images are stored in database.
- /template: contains html files.

3. Program Features:

- All functions for the url in the Flask Application are implemented with protections to prevent inadvertent accesses by the users. User authentication is always processed prior to access being granted.
- All the functions for url in the Flask Application process all particular values of the retrieved data (None or empty list).
- Foreign key integrity protection for all deletions of any instance of any class in the program. For example, the deletion of a conference or a session must follow the deletion of their entries in the *wishlist* of the users.
- Similarly, adding and modifying for any data always ensure the foreign key integrity constraints.
- In the application, there is a *getName()* function that collects the name of speakers, locations, types of sessions to the file *names.json*. This file is used for the *datalist input form* in html for the user to pick a name. When a user wants to add a new speaker, location, or type of session, the application always tries to help him or her not to create a new one if it already exists. This would save time for typing and also help keeping the integrity of the database (not redundancy nor duplication) for further filtering or searching.
  At this time, the name of the speaker must be unique, later we can change this with other implementation regarding his or her email.

4. Application Limitations:

- Lack of Register-to-Conference function for user.

- Lack of time related integrity constraints of data. For example, the relation between the start date of a conference and the date of its sessions, or the time of the sessions given by the same speaker (time conflict), etc.

- Lack of security check for the document upload function. The program just uses *imghdr* module to check the image file.

➢ Program Usage

1. First, you need to install **VirtualBox** *(https://www.virtualbox.org/wiki/Downloads)* and **Vagrant** *(href="https://www.vagrantup.com/downloads)* to your machine.

2. Then, you'll need to clone the repository of this project to your local machine.
   The folder tree can be found at the end of this document.

3. Register this application on Google and/or FB (**GuideForAppRegister.pdf**)

4. In the terminal window, go to the vagrant directory and type **vagrant up** to launch your virtual machine. This will take some time in your first run, because it needs to install some dependencies.

   vagrant/conference$vagrant up

5. Once it is up and running, type **vagrant ssh** to log into it. This will log your terminal in to the virtual machine, and you'll get a Linux shell prompt.

   vagrant/conference$vagrant ssh

6. On your virtual machine, go to **/vagrant/conference** directory and create the database.

   vagrant@vagrant-ubuntu-trusty-32:/$**cd /vagrant/conference**
   vagrant@vagrant-ubuntu-trusty-32:/vagrant/conference$**python database_setup.py**

7. Run the program:

   vagrant@vagrant-ubuntu-trusty-32:/vagrant/conference$**python project.py**

8. Access the program by visiting http://localhost:5000 locally on your browser.

Folder Tree cloned from GitHub:

```
vagrant/
        Document.pdf (this file)
        GuideForAppRegister.pdf
        pg_config.sh
        Vagrantfile
        .vagrant
        conference/
                database_setup.py
                project.py
                db_CRUD.py
```

client_secrets.json
fb_client_secrets.json
static/
    bootstrap.css
    bootstrap.min.css
    bootstrap-table.css
    bootstrap-table.min.css
    bootstrap-theme.css
    bootstrap-theme.min.css
    bootstrap-theme.css.map
    new_simple-sidebar.css
    bootstrap.css.map
    fileinput.min.css
    jquery.js
    jquery.min.js
    bootstrap.js
    bootstrap.min.js
    bootstrap-table.js
    bootstrap-table.min.js
    bootstrap-table-zh-CN.js
    fileinput_locale_LANG.js
    fileinput.min.js
static/document/ (for storing uploaded documents)
static/images/ (for temporary storing uploaded images)
template/
    login.html
    header.html
    main.html
    infor.html
    conferences.html
    showsessions.html
    addconference.html
    modifyconference.html
    deleteconference.html
    addsession.html
    modifysession.html
    deletesession.html
    showwishlist.html
    showdocument.html
    conferenceregister.html