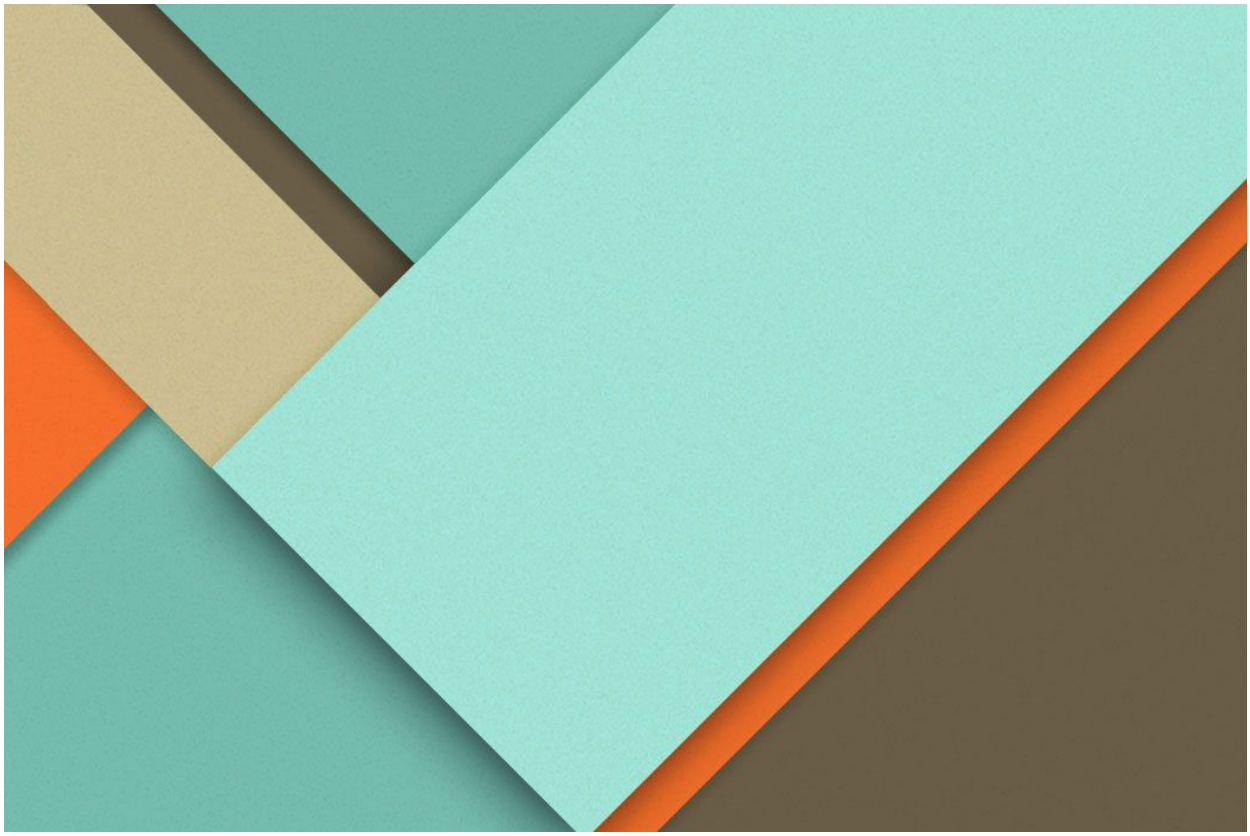

Event Planner System

29.10.2017

Catherine Rakama, Ana Cansino
Final Project, KTH

Overview

Management System for planning events

Goals

1. Workflow of event requests (event initiation and application)
2. Workflow of tasks distribution to services/production departments. (choose one sub-team from each department)
3. Staff recruitment management.
4. Financial requests management.

System Specifications

The system uses python programming language for implementation and Flask framework to help organize components to an MVC architecture.

For Backend, the system uses Object Relational Mapping(ORM) that presents a method of associating user-defined Python classes with database tables, and instances of those classes (objects) with rows in their corresponding tables.

How to Install

Virtual Environment:

You can get better materials online on how to set up python virtual environment(***Virtualenv***) on windows or Ubuntu system or see Appendix.

Dependencies:

Install required dependencies from ***requirements.txt*** file found in project directory. CD/navigate to project directory, activate virtual environment and issue the following command:

pip install -r requirements.txt to install project's dependencies

Database:

I. Create Database for the project:

Go to ***mysql shell*** by issuing the following command. **-p** assumes that you had set password for your mysql during installation. If not just use ***mysql -u root.*** Dd

mysql -u root -p

mysql> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin2017';

mysql> CREATE DATABASE eventsplanner_db;

mysql> GRANT ALL PRIVILEGES ON eventsplanner_db . * TO 'useradmin'@'localhost';

II. Add users to DB

Create users: On your command line, go to project folder, activate virtual environment then type ***flask shell*** to get to python flask shell. Then issue commands as follows to create users.

Or On a new terminal, go to python shell by issuing ***flask shell*** command: Issue the following commands to create different users who will access your system. **Admin** user created previously is the main admin and initiator of major actions.

>>> from app.models import RemoteUser

>>> from app import db

>>> sbtadmin = RemoteUser(email="sbtadmin@gmail.com",username="sbtadmin", password="sbtadmin2017", is_music_sbtadmin=True)

>>> db.session.add(sbtadmin)

>>> db.session.commit()

Using the above format, create other users by replacing ***sbtadmin*** in every part of the statement with the following names respectively.

- ***fmadmin***
- ***scsadmin***
- ***hradmin***
- ***pmadmin***
- ***amadmin***

III. Database Migrations:

At first, initialize your migration object by issuing: **`$ flask db init.`** The object will help you to run migrations using Flask-Migrate thus managing changes to your database.

Everytime you make changes to you models(model.py) you have to migrate and upgrade your database as follows:

```
$ flask db migrate
```

```
$ flask db upgrade
```

You can check your tables if they have been created by issuing the following commands

```
$ mysql -u root
```

```
mysql> use eventsplanner_db;
```

```
mysql> show tables;
```

IV. Database Configurations/SetUp:

Create an instance directory in the remoterefapp directory, and then create a config.py file. Add the following configuration settings:

- SECRET_KEY = 'p9Bv<3Eid9%$i01'
  SQLALCHEMY_DATABASE_URI = 'mysql://admin:admin2017@localhost/eventsplanner_db
- You can include other sensitive configuration settings such as emails passwords and address

Test the app locally

- Set the FLASK_CONFIG and FLASK_APP environment variables before running the app las follows:
- $ export FLASK_CONFIG=development
- $ export FLASK_APP=run.py
- $ flask run

Access in at local host: http://127.0.0.1:5000/

Milestones

I. Release 1.0.0

First release contains 60% of the core functions of the system

II. Release 1.1.0

To include the remaining functionalities and test codes for all view functions

Appendix

Setting Up Python Development Environment

Installation varies with different OS, it may not be configured correctly sometimes and you may need to debug and fix the problems using online tutorials..

- ➔ Update your OS by issuing sudo apt-get update or sudo apt-get -y upgrade which comes with python pre-installed.
- ➔ Use python 2.7.12 for this project
- ➔ Install pip to help you manage software packages for Python.
- ➔ Install other needed development tools sudo apt-get install build-essential libssl-dev libffi-dev sudo libapache2-mod-wsgi python-dev
- ➔ Enable mod_wsgi (*an Apache HTTP server mod that enables Apache to serve Flask applications.*) , run the following command: sudo a2enmod wsgi
- ➔ Set Up a Virtual Environment - to ensure this project's set of dependencies won't disrupt any of your other projects. Use the following commands for setup:

```
$ sudo apt install virtualenvwrapper

$ echo "source /usr/share/virtualenvwrapper/virtualenvwrapper.sh" >> ~/.bashrc

$ export WORKON_HOME=~/.virtualenvs

$ mkdir $WORKON_HOME

$ echo "export WORKON_HOME=$WORKON_HOME" >> ~/.bashrc
```

- ➔ Create a Virtual environment for your project: mkvirtualenv -p python2.7 remoteapp
- ➔ You will use deactivate or workon anywhere on your terminal to deactivate and activate your virtual environment. e.g workon remoteapp