Individual Report – Booking System Login Integration

Balazs Barcza
National College of Ireland
Dublin, Ireland
x19190638@student.ncirl.ie

Our project was to create a meeting booking system. This application should be scalable. The project builds up to three main key modules. The login module and the calendar module and google calendar integration. I was responsible for creating a login system where the user can login-in and log out. We have to use Amazon Web Services to deploy our application. I have used the Python program language and Django framework and Bootstrap to create a web application. I have deployed to the cloud I have used Amazon Elastic Compute Cloud EC2. I have implemented the data storage in SQLite. In the settings, SQLite is utilized by default.

If you're new to databases or just want to try out Django, this is the simplest option. Because SQLite is incorporated into Python, you won't need to install anything else to support your database. This database type is perfect for the project because our application is using a different type of database. It makes the system more secure. If one of the modules is hacked still another model is safe. User authentication is a critical component of most applications, and it is defined as the process by which a device verifies the identity of someone connecting to a network resource. This refers to our web app in our situation. There are various technologies and web frameworks available today that make this task simple, but in our instance, I focused on the Django web framework to complete this operation.

I have used GitHub to develop the program and It was developed using Docker containers, as part of the CI/CD pipelines for the application. The plane was deploying the web application Elastic Beanstalk but unfortunately, I had an issue with the integration so I had to take another approach. So I have deployed a User authentication web application into the Amazon EC2. I will discuss my issue later in the report.

1. Project requirements

The following case study functioned as the basis for the project's requirements. Currently, the goal was to design a web application that
The user can register and can generate any event and make bookings to invite another user to the meeting. The user can add information about the meeting (location, requested documents, etc) and the user can send an invitation using a google calendar API. The minimum requirement for the web application is to include different modules and those modules connect. The following are the functional requirements for this application:

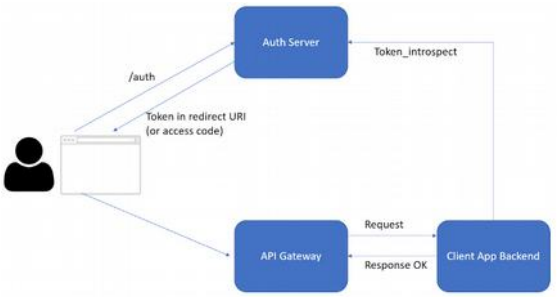
- *The client can create and modify and delete the event*
- *The client can send invitations to another user with Google Calendar*
- *New user can login and logout and make registration in the web application*

Non- functional requirements

- Logging into the website is less than 5 seconds
- SSH connection between the web app and the API
- Cache the user detail
- Database and file storage need to be managed automatically
- How the data will be secure in the cloud

2. Architecture and Design

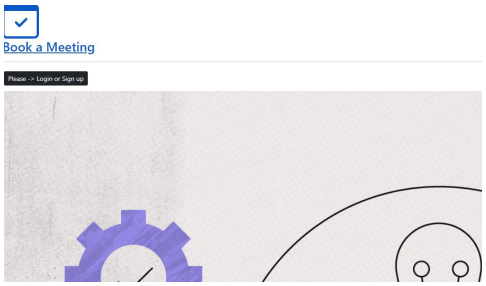
Django by default comes with the admin page, where the superuser can login and create other users and can add and remove other users and privileges. Our goal the user can register and login and logout from the web application and only the user Who is authentication can visit the calendar module. The superuser can delete and remove and can manage the registration.



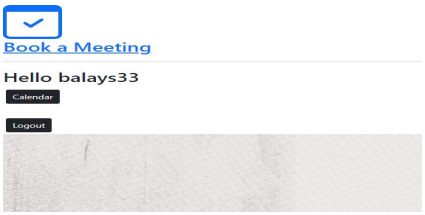
1, figure user login-logout back end: https://www.mathieupassenaud.fr/logout-oauth2/

The table is show how is working a login and logout back and system use token. In this project, we implement a build in authentication model from Django Framework.

I have developed the application with a welcome page where the user can visit the login and registration page.



2, figure welcome pageThe 2 figure we see the user is not login. If the user is logged in the cline gets a different welcome page. On this page, the user can jump to the calendar page.



3, figure the user logged in

In this image (3 figure), we can see the user logged in and we can see the user name (Hello balays33) and the link to the calendar, and the logout button. Only the authentication user can see the calendar button.

```
{% if request.user.is_authenticated %}
  <div>
    <p>Hello {{request.user}}</p>
    <div>
      <a href="{% url 'user_profile' user %}" class="btn btn-dark btn-block btn-sm m-1">Profile</a>
    </div>
    <div>
      <a href="{% url 'calendarview'  %}" class="btn btn-dark btn-block btn-sm m-1">Calendar</a>
    </div>
```

4, figure authentication code

{% if request.user.is_authenticated %} This code shows that only the user who is registered on the website can see the calendar button. The redirect URL needs to provide and user can visit: https://calendar-project-nci.me/

3. Module Implementation

In this section, we will check the code implementations and the libraries.

```
from django.shortcuts import render, redirect
from django.contrib import messages                    # django flash messages
from django.http import HttpResponse
from django.contrib.auth import authenticate, login, logout   #user autheticate
from django.contrib.auth.decorators import login_required     # user  wants to acces a page
#from .forms import CustomUserCreationForm                    # user registration form
#from django.contrib.auth.forms import CustomUserCreationForm
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
```

This is the important libraries hast to be implement to make the user authentication to work.

From django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required

The application was deployed to the EC2 this is the link where the program can be visited: http://52.215.255.86:8080/ I have used the port 8080 because I have running a docker container on the EC2.

Username **admin** and password **adminadmin**

The application was pushed to the docker hub and from there was pulled to the AWS EC2.

The image can pull from:

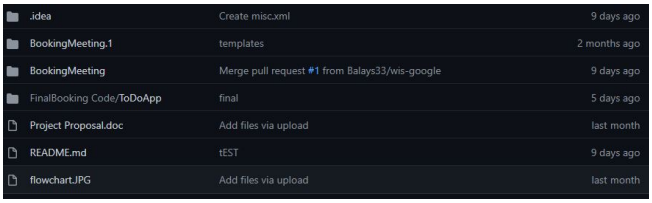
docker pull balays33/bookingm_by_balazs:bookingmeeting

The project can be find the github :

https://github.com/Balays33/Scalable-Cloud-Programming-PGDCLOUD_SEP-2022

4. Conclusion

In this project, I had an opportunity to understand and apply the theory of scalable programming, architecture design, and parallel programming.
Our team learned a few new technologies and how should work in a team. Unfortunately my side I had a few issues. I had created a Django web application and I wanted to deploy it on the elastic beanstalk but I could not manage it because the Django version was not compatible. I have created the application on the 2.3 version and I wanted to integrate the Django rest framework to create a working API connection with another module. The Django rest framework only works with 3 or high versions. unfortunately, the SQLite is not working in version 3 and I could not deploy the app EB. I have implemented the Serialization to create a profile page but after the libraries import the SQLite stopped working.
Because the time was too short I had to create a new app and I was not managing everything to work the proper way. For this reason, I have a different folder on the GitHub website.



BookingMeeting.1 is only a template
BookingMeeting working registration page unfortunately not code mess up with Django rest framework so not possible to run on the EC2 & Elastic Beanstalk possible run on the local machine and AWS Cloud9
FinalBooking Code/ToDoApp restarted the project to make a running application on the Cloud missing the registration page.

Even though AWS resources didn't work when needed, the application that was produced was the consequence of a tough word from the three of us. It was difficult for us to overcome the obstacles.