Code First Girls Degree Project Report – Grapevine

By Mutiyat Adamo, Divya Singh Chouhan, Neneh Patel and Mihaela Strilciuc

Introduction

The aim for our project was to create an event web application called Grapevine. We chose the name Grapevine because our app uses an API and other websites to circulate data on our webpage.

Grapevine is an event application where users can view events within the UK and view tickets for different types of events. Users will not be able to purchase tickets directly on our website, however, users who wish to book tickets will be directed to an external link to buy tickets.

The webpage will use data extracted from Ticketmaster's API and web scraping exhibition data from the Art Fund website showcasing museum and art gallery attractions. We have also implemented Google Maps to view the location of the events. This adds to our website's unique selling point as a hub for events of all types. Most websites solely focus on music and concerts, so we want to bring together a wider range of events to choose from.

Grapevine will be different to other events websites as we will include a profile section where users can add notes on their excitement for their preferred events as well as write some comments after the events. Users will also have access to a blog section where they can see notes added by other users. Users will need to have an account by signing up to Grapevine in order to have access to all pages on Grapevine. However, users without an account can view the homepage.

Background

Grapevine calls on Ticketmaster's API to obtain events data and allow users to retrieve and search for events within the UK. The API data extracted from Ticketmaster include: title, date, city, image, venue name and location of the events.

Our web page also showcases web scraping data extracted from the Art Fund and Eventbrite websites and has also been used to provide the same information for the exhibitions.
We have also implemented the use of Google Maps API which takes the latitude and longitude from the events to give location in maps. We have used <iframe> code along with our HTML pages to implement this functionality on each individual event page. This prevents the search or events pages from loading too slowly, and only gives extra location information if the user is interested.

In our initial planning stage we used a shared Google Slides document to help organize our ideas and work together on the initial planning. This included identifying which roles we should undertake as part of the project.

Specifications and design

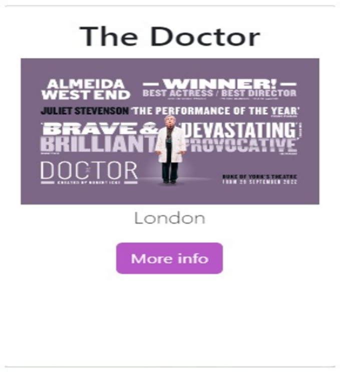
Requirements for user experience:

- User login details
- Search (Title/Venue/Location/Event)
- Event details
- Access to event location through Google Maps API on individual event page

- User note section on user profile
- Blog showing variety of user notes, accessed by all users

For our project we used the following tools and languages:

- **Frontend**: we used HTML, CSS and Bootstrap for our user interface to make our web page visually appealing and interactive for the user. We have a set color theme created which we have used throughout our website to match our grapevine name and theme. Unsplash for homepage background images.
- **Backend**: Python where we wrote our codes, Flask framework to develop our application by integrating our HTML files and our API codes.
- **API**: TicketMaster, Google Maps, Postman and BeautifulSoup used to obtain data for our webpage for the event name, title, city, venue, image and booking URL.
- **Web scraping**: Some of the information from Art Fund and Eventbrite was not readily accessible within an API function. Using this method, we retrieved the same information from the API but from the remaining two websites.



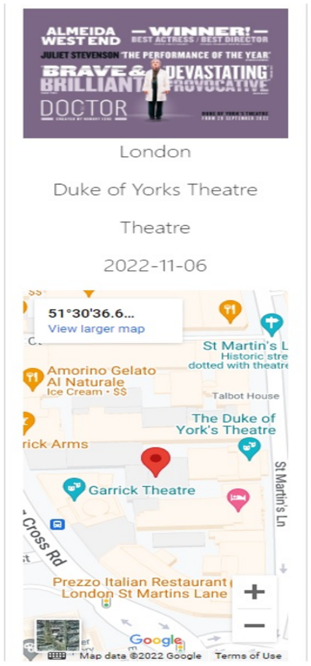
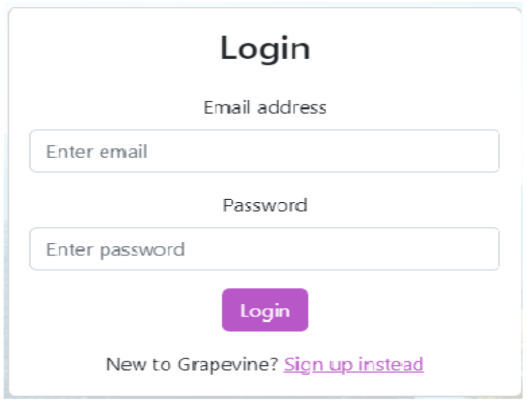


Figure: Search result with Google Maps API integration on individual event page.

- **Database**: SQLAlchemy and MySQL to create our database and manage our users' login credentials, safely storing our users' password using the hash function SHA-256 to safely encrypt and secure our users' password. Our database also stores any user note entries made on our webpage.



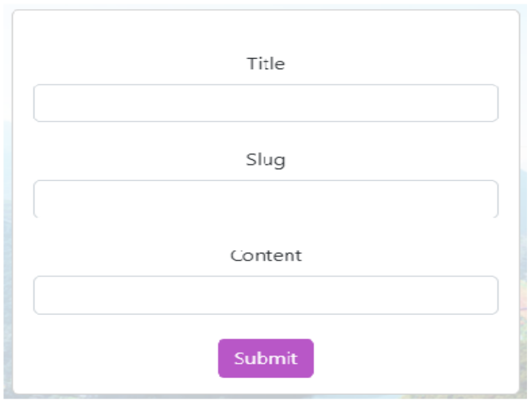


Figure : Login and add notes page

- **Trello board**: we created a project workspace where we had our project board. On this board, we had key requirements for our project from the user and owner point of view documented using the MSCW method (Must Have, Should Have, Could Have, Won't Have). We also designed our board using the agile methodology of First, Second and Third Sprint to organize requirements completion in order. A backlog for tasks awaiting completion.
  A link to our Trello board is here:
  https://trello.com/invite/cfgteamproject/bfc75513579fcb8046adee7ab95f7b88

  Below is a workflow design of the request and response flow for our application:

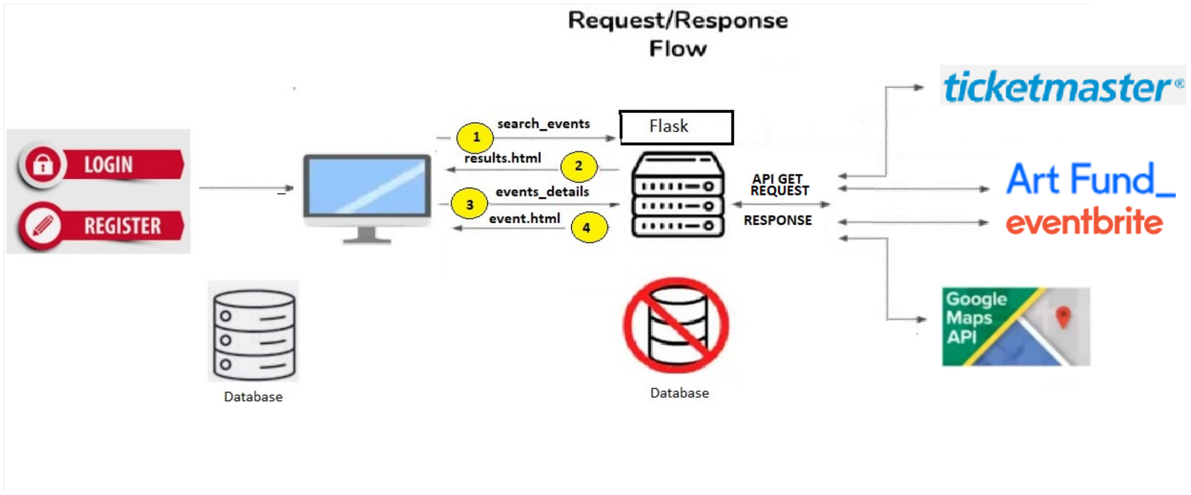
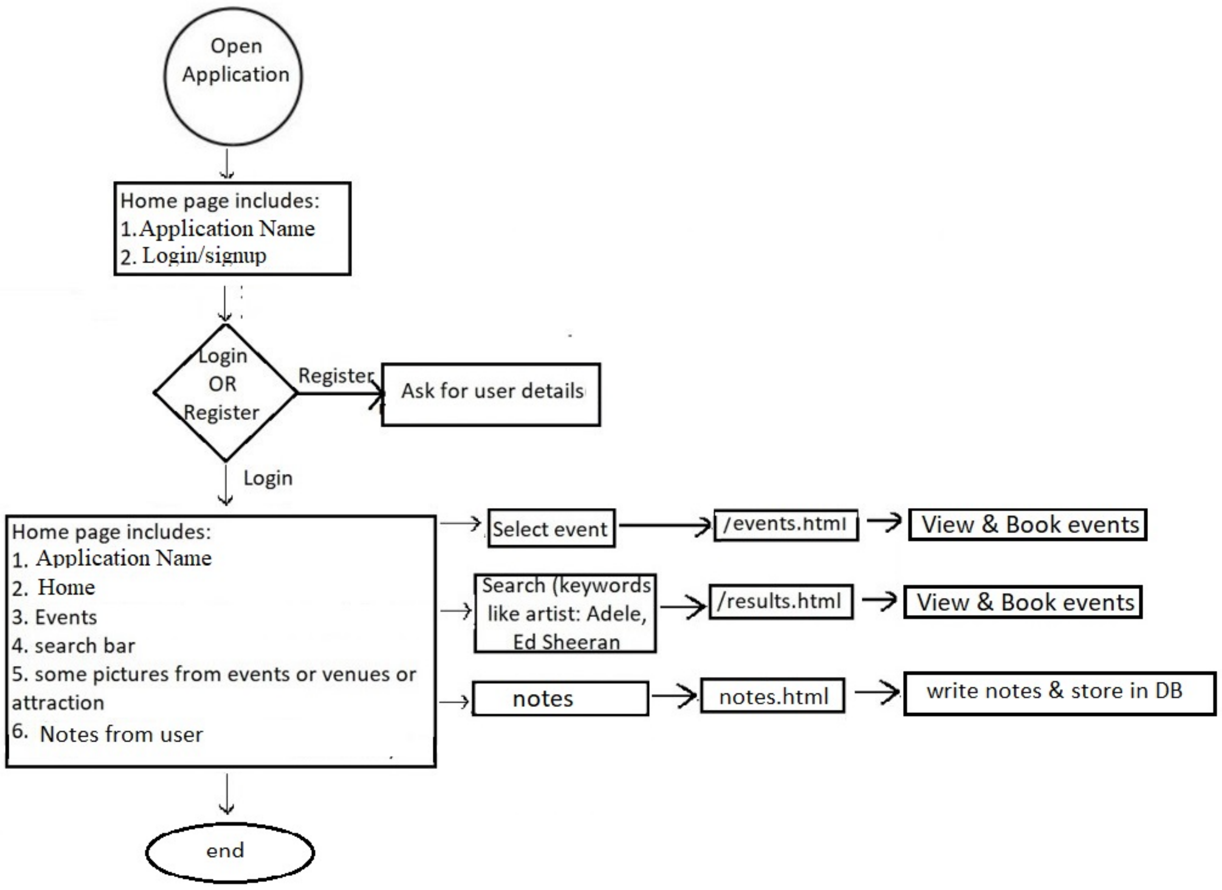


Figure: Workflow Design



This is a design architecture of our plans for our website prior to development:

Figure: Flowchart representation

Implementation and execution

We have defined the roles as:

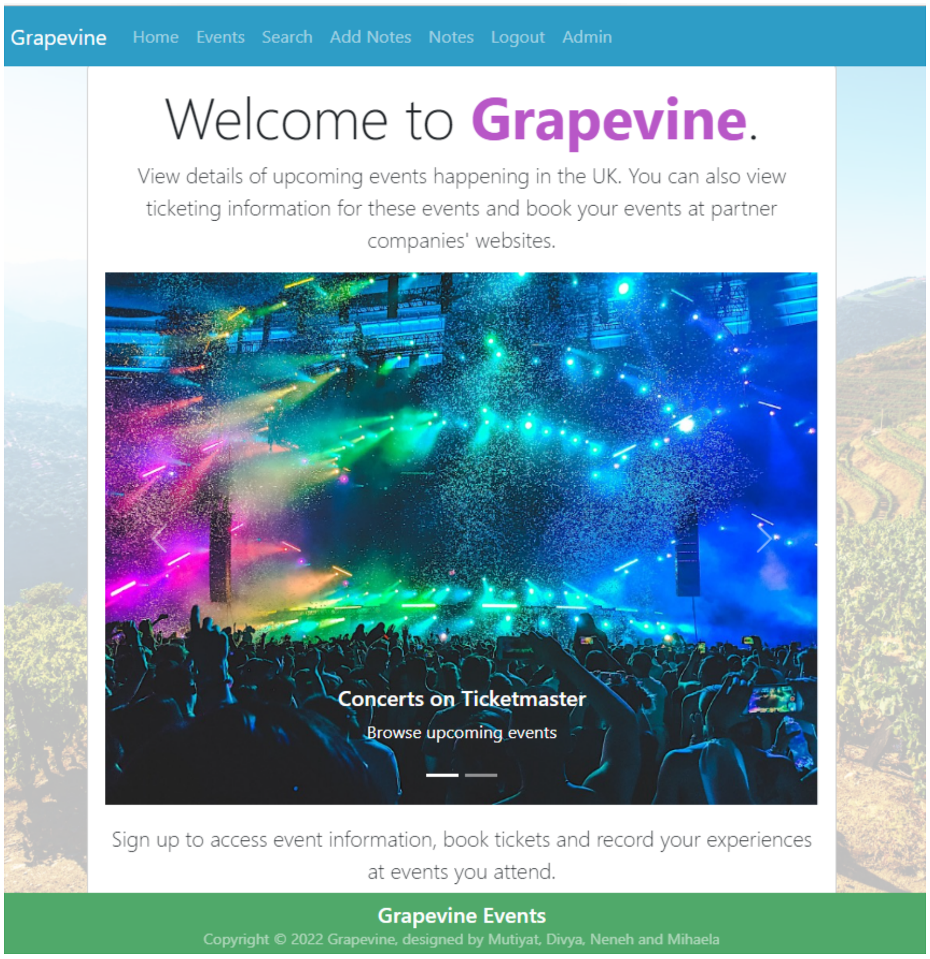
- Front End: Mutiyat

(Initially we decided to explore different APIs like Ticketmaster, Eventbrite, Museum and arts)

- API: (Ticketmaster) : Divya
  (Google Maps) : Divya
- Web scraping: (Art Fund) : Neneh
  (Eventbrite): Neneh
- Backend: Mihaela, Divya
- Database: Mihaela
- Testing: Mihaela, Mutiyat, Divya, Neneh
- Once all parts were integrated, we completed the final testing of our application, individual tests are done on each part and also a database test.

We used Pycharm as our IDE - Python to write our code for the web page. We used HTML, Jinja2, CSS and Bootstrap framework to develop our webpage's frontend.
We also used python libraries such as: Flask framework, SQLAlchemy, MySQL connector, Werkzeug, WTForms, requests, render_template, redirect, flask-login and flash.

We also created a GitHub organization and repository where we regularly pushed our files so that all team members had access to individual codes written. This allowed us to work collaboratively reviewing and integrating other members' code to our individual project work.



Welcome to Grapevine.

View details of upcoming events happening in the UK. You can also view ticketing information for these events and book your events at partner companies' websites.

Sign up to access event information, book tickets and record your experiences at events you attend.

Grapevine Events

Copyright © 2022 Grapevine, designed by Mutiyat, Divya, Neneh and Mihaela

Figure : Home Page

Challenges and Achievements

Challenges:

- Our main challenges during the project involved integrating data obtained from APIs and web scraping to the front-end.
- One of the major issues with ticketmaster APIs was repetition of the same data and we sorted out this by iterating and filtering through image urls in order to prevent duplicate events.

Achievements:

- Achievements include integrating all three websites into the search function and allowing the user to search across all of them.
- Integrating the Google Maps API was another significant achievement as users often want to see how easy it is to attend an event before going to the booking stage of the user journey.

Decision to change something:

- We decided to remove the option for the email capability and also the option for the owner to send a newsletter.

Agile Development Strategies

As a group, we have made a SWOT analysis which really helped us in distributing the different roles.

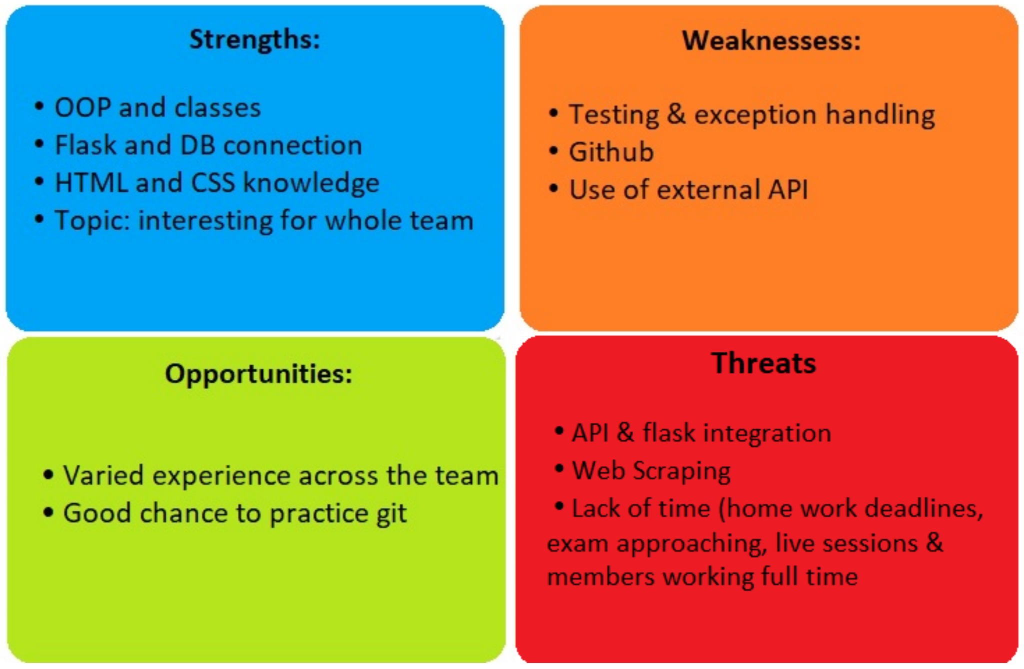

Strengths:
• OOP and classes
• Flask and DB connection
• HTML and CSS knowledge
• Topic: interesting for whole team
Weaknessess:
• Testing & exception handling
• Github
• Use of external API
Opportunities:
• Varied experience across the team
• Good chance to practice git
Threats
• API & flask integration
• Web Scraping
• Lack of time (home work deadlines, exam approaching, live sessions & members working full time

Figure: SWOT group analysis

Over the course of our project, we successfully focused on maintaining constant collaboration and communication.

We have planned color-coded labels on the Trello board (Must, should, Could, Won't,customer, owner) to organize tasks for a sprint.

Also, we have created a list: "Backlog" "first sprint," "second sprint" and "Third sprint.

In the first week we discussed project planning, creating an architecture diagram and a Trello board.

In the second week, we have tried to implement the skeleton structure of our project including fetching data from the APIs (Ticketmaster, Eventbrite, Art Fund and Google Maps) , creating a front end and also creating a database for login/signup.

In our final week we have integrated all the APIs together with the front end and database. We have also added a notes section in the database where users can add some notes (title, slug and content). This week we were also involved in code review, front-end development and testing.
We have used Github both as version control and to share the completed code for each of our tasks. Also, we have planned to use Github as a tool to help our group with refactoring and reviewing completed code sections.
This agile approach has allowed us to be more flexible, quicker to adapt changes during the project and overall become more innovative and modifiable. We held daily meetings between Mondays - Thursdays to try and develop the project and keep momentum going during the week. Additionally, our Sunday evening meetings served as both a retrospective of the previous week, and a sprint planning session for the following week's tasks.

Testing and Evaluation

TESTING AND EVALUATION

Testing strategy

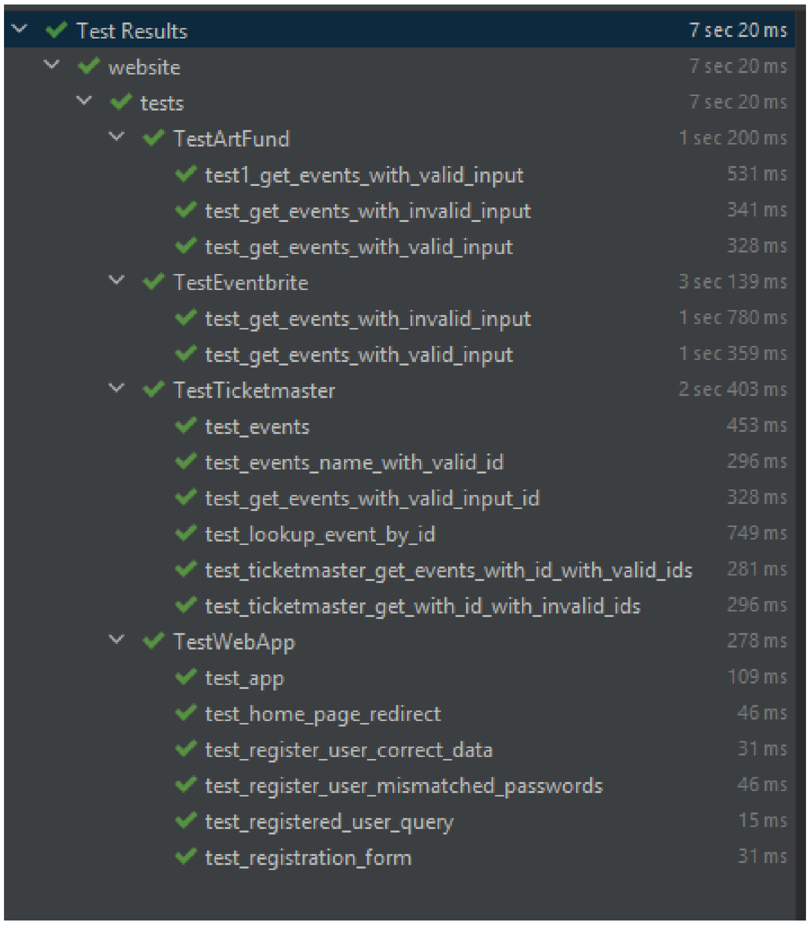
To test that our application worked, we decided to create a separate directory named "test" with a file to use the unittest module.
It was a challenge at first to test our Grapevine code, as it consists of four different APIs and a database. We were testing:

- If structure of our function worked as expected
- If the connection to the external service (different API) was running correctly
- The database is running as expected

All of our tests ran successfully.

Part of the aims of the tests are to ensure that the tests are accurate to the current date, however it is possible that when the tests are reviewed, the live data will be different to what we have tested, and they will not work. To retrieve new test data, you can achieve this by using the following method -
Run the selected function in event.py in the website>api folder which corresponds to the relevant test that you would like to run. Enter the selected data (location, id, slug, etc) into the test. For example, you may need to retrieve an event from the function by searching for an event query. Enter the relevant information into the test and press the green arrow to run it.

Functional and user testing



```
✓ Test Results                                              7 sec 20 ms
  ✓ website                                                 7 sec 20 ms
    ✓ tests                                                 7 sec 20 ms
      ✓ TestArtFund                                         1 sec 200 ms
          ✓ test1_get_events_with_valid_input               531 ms
          ✓ test_get_events_with_invalid_input              341 ms
          ✓ test_get_events_with_valid_input                328 ms
      ✓ TestEventbrite                                      3 sec 139 ms
          ✓ test_get_events_with_invalid_input              1 sec 780 ms
          ✓ test_get_events_with_valid_input                1 sec 359 ms
      ✓ TestTicketmaster                                    2 sec 403 ms
          ✓ test_events                                     453 ms
          ✓ test_events_name_with_valid_id                  296 ms
          ✓ test_get_events_with_valid_input_id             328 ms
          ✓ test_lookup_event_by_id                         749 ms
          ✓ test_ticketmaster_get_events_with_id_with_valid_ids  281 ms
          ✓ test_ticketmaster_get_with_id_with_invalid_ids  296 ms
      ✓ TestWebApp                                          278 ms
          ✓ test_app                                        109 ms
          ✓ test_home_page_redirect                         46 ms
          ✓ test_register_user_correct_data                 31 ms
          ✓ test_register_user_mismatched_passwords         46 ms
          ✓ test_registered_user_query                      15 ms
          ✓ test_registration_form                          31 ms
```

System limitations

Throughout the course of the project, we discovered some limitations to our project, for example:

- The information returned from the latitude and longitude from Art Fund web scraping was not as accurate as that of the Eventbrite or Ticketmaster. This meant that not all of the events were able to be integrated with the Google Maps API and show a location on their individual 'More Info' page.
- The Admin side of the website is not fully developed, this is something we would like to implement in the future. The admin will be able to remove users' notes, respond to feedback, perform website maintenance
- Another thing we would like to implement in the future is the option for the user to add their events to a collection and that collection to be saved in the database.

Conclusion

In this project, we have learnt to use Python and SQL to create a functioning website that allows users to login/signup, view and search for events in the United Kingdom. The website successfully returns data from an API and web scraped information. Users can also add notes which are stored in our secure database. We have also learnt to apply front-end development using HTML, CSS and bootstrap in order to create an attractive platform. In future, we hope to develop the website further to give suggestions to users based on their past search and create a more detailed user profile/communication with other website users.