Import of Events into Open Event Orga Server

GSoC 2017 – *Afroz Ahamad*; *BITS Pilani Hyderabad Campus, India*
Possible Mentor - *Niranjan Rajendran*

Abstract

This project aims at providing support for automatic import of events from various event listing sites into Open Event. This import of events is useful in two ways:

- Import events which are not already listed in Open Event.
- Allow a user to import events – created on other sites – directly into Open Event.

While working on this project, I aim to implement the following features:

- Show results from eventbrite in search
- Implement the query-server as a micro-service

This project has been discussed in the chat channels[1] leading to some key points:

- The user should not know that the events are being imported. (There should not be any graphical prompt suggesting import.)
- The query-server is to be used for initial listing of events.
- Add scraped JSON directly into database without making files/zip.

Table of Contents

Motivation

For quite some time, I have wanted to work on a project that is actually used by people for practical tasks. I have always wanted to be part of a community with similar interests where I could learn from the discussions. All this I have found in FOSSASIA, and so I would like to work on this project for GSoC.

As for the task itself, adding support for import of events into Open Event will actually help many people and seems like a good combination of challenging and practical, which is why I'd like to work on it. I hope to contribute a bit of my part to Open Event in making it 'the' open source event listing website.

I'd love to stay involved with FOSSASIA beyond GSoC. As I've said, I've started to really like the community, and hope to contribute to it and to the project for the foreseeable future.

Use Cases

- **Joe** has planned a local meetup of folks who watch 'Impractical Jokers' before the upcoming season. He has already created an event on eventbrite. To ensure that it reaches more people, he wants to create an event on Open Event too. He has to fill in every detail again which would be a waste of time and effort. This project helps Joe by letting him search for his own event on Open Event and importing it automatically for him.

- **Murr** has recently learned Docker and wants to see if there is a local meetup where he can meet other developers. He searches for 'Docker' on Open Event, but some events are not listed yet in the database. This project helps Murray by importing local events on 'Docker' automatically.

- **Q** has three cats who were born in the same week. He has created three events on meetup.com for their birthdays. He also wants to create events on Open Event, but having three similar events increase chances of errors causing conflicting details about the same event at two different places. This project imports all events separately for Q helping his love for his cats.

- **Sal** wants to join some group for hiking. He searches for it on Open Event and he finds one such event. Now that event was never created, but since someone has searched it already, it exists in Open Event's database.

Introduction of Components Involved

Open Event Orga Server

The Open Event Orga Server[2] enables organizers to manage events from concerts to conferences and meet-ups. It offers features for events with several tracks and venues. Event managers can create invitation forms for speakers and build schedules in a drag and drop interface. The event information is stored in a database. The system provides API endpoints to fetch the data, and to modify and update it. Organizers can import and export event data in a standard compressed file format that includes the event data in JSON and binary media files like images and audio.

Event Collect

This service will be developed as a collection of site-specific scrapers for event listing websites and APIs to use these scrapers. Currently the event-collect repository has a scraper for eventbrite and one for ticketleap and uses loklak API for collecting JSON data. These scrapers needs to be improved.

More scrapers need to be added to event-collect, like ones for

- Meetup
- Eventful
- Yelp

Event Collect's code is on GitHub.

Query Server

The Query Server was developed mostly by me. I wrote the initial implementation of the query server as a command line tool to resolve two mini projects (1) and (2) at labs.fossasia.org. Thanks to the community, it was soon published as a full fledged command line tool. Later, keeping in mind its requirement for other FOSSASIA services, and to be able to implement the query-server as a micro-service, I worked on taking it to a deployable state. Current situation : it is deployed on Heroku. I have also added docker-deployment compatibility to the query server to ease deployment of an instance of this server on a docker infrastructure.

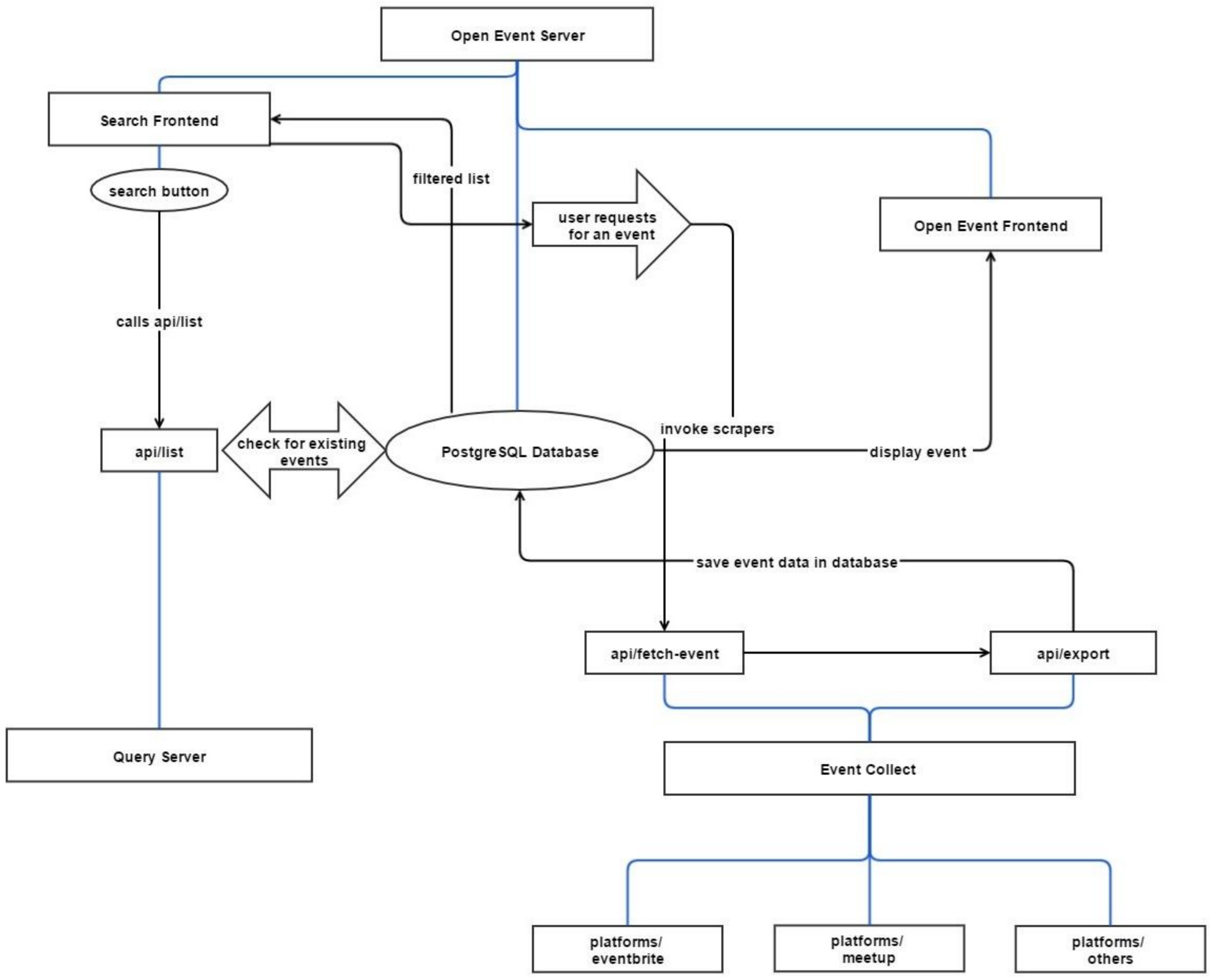
The main aim of this project is to provide web infrastructure (APIs and deployments) to FOSSASIA services which need to process query strings for any task. The server currently provides with XML feed for any query string.

The query-server code is on GitHub.

Live deployment of query-server is on Heroku.

Implementation

The following is a basic workflow for the system I have laid out for imports. It follows the basic design philosophy that we keep all modules separate and use services as a whole and specifically for what they are built, generally. (**The diagram follows on the next page.**)


Open Event Server
Search Frontend
search button
filtered list
user requests for an event
Open Event Frontend
calls api/list
api/list
check for existing events
PostgreSQL Database
invoke scrapers
display event
save event data in database
api/fetch-event
api/export
Query Server
Event Collect
platforms/ eventbrite
platforms/ meetup
platforms/ others

The Workflow Explained

Open Event Orga Server

Here the **'search'** functionality makes a call to **api/list** API provided by query-server which returns with events' 'Title' and 'Event Link' from the parsed XML feed. This list is displayed as Open Event's search results.

Now the results having been displayed, the user can click on any of the events.

When the user clicks on any event, the event is searched for in Open Event's database.

- The event page loads if the event is found.
- If the event does not already exist in the database, clicking on any event will
    - ➢ Insert this event's title and link in the database and get the **event_id**
    - ➢ Make a call to **api/fetch-event** in event-collect which then invokes a site-specific scraper to fetch data about the event the user has chosen
    - ➢ When the data is scraped, it is imported into Open Event database using the previously generated **event_id.**

(***There are mock-ups of these on later pages. Please use them to follow along with this description***).

The page will be loaded using **jquery ajax** as and when the scraping is done**.**When the imports are done, the search page refreshes with the new results.

The Open Event Orga Server exposes a well documented REST API that can be used by external services to access the data[3].

Query Server

The query-server is a mini-tool that is used to process a query string. The server itself is written in NodeJS while the scraper is in python which is run by spawning a child-process. The query string calls the search engine result scraper **rss-generator.py** that is based on the scraper at searss. This scraper takes search engine, *presently Google, Bing, DuckDuckGo and Yahoo* as additional input and searches on that search engine. The output from the scraper, which is XML, is written into MongoDB database with the query string as index. There exists a frontend where this XML feed is rendered after being prettified with the help of PrismJS.

The query-server will be used for initial listing of events from different search engines. This will be accessed through the following API.

- ➢ **api/list** : to provide with an initial list of events (titles and links) to be displayed on Open Event search results. When an event is searched on Open Event, the query is passed on to query-server where a search is made by calling **rss-generator.py** with appending some details for better event hunting. The discussions on chat channels[1] lead to using *GeoLoc and IP* for improving on local events. More details have to be thought as to how can we improve event listing.

The XML feed from the scraper is parsed for events inside query server to generate a list containing Event Titles and Links.

Each event in this list is then searched for in the database to check if it exists already. I have planned to use *fusejs.io* or *elastic search* to achieve fuzzy searching for events in Open Event database.

One example of what I wish to achieve by implementing this type of search in the database follows.

The user may search for
-*Google Cloud Event Delhi*
-*Google Event, Delhi*
-*Google Cloud, Delhi*
-*google cloud delhi*
-*Google Cloud Onboard Delhi*
-*Google Delhi Cloud event*

All these searches should match with "*Google Cloud Onboard Event, Delhi*" with good accuracy.

After removing duplicates and events which already exist in the database from this list have been deleted, each event is rendered on search frontend of Open Event as a separate event. The user can click on any of these event, which will make a call to event collect.

Open Event Frontend

The frontend required to render the above mentioned pages will be developed based on the mock-ups on later pages. Not in their entirety though, but the parts which will be required.

The following frontend is required for sure -

- **Displaying 'loading page' till the scrapers are running.**
  I have planned to use a template with a placeholder similar to the one used in Facebook feed while the posts load (***Please refer to mock-up image 3***). This will be a simple CSS snippet which will be displayed while the data is being scraped. Once the event data is imported, the existing graphical interface of Open Event will be used.

- **Loading search results found from query-server**
- **Open Event Search Bar**
- **Event**
  For these I do not see the need to change the Open Event's interface. I will integrate all these in the front-end as required. Changes in the UI, if necessary, will be discussed with the mentor.

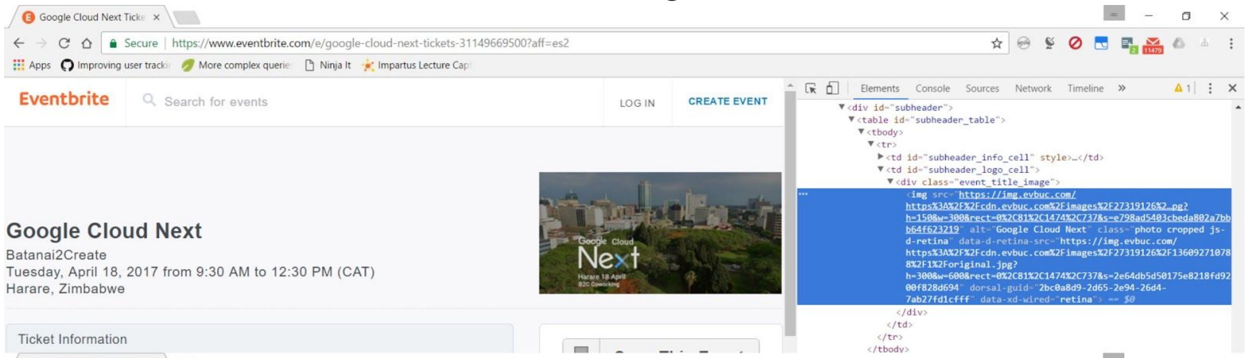
Event Collect

The event collect is developed as a separate module which has two parts:

- **Site specific scrapers**

In its present state, event collect has a scraper for eventbrite which, given a query, scrapes eventbrite search results and downloads JSON files of each event using Loklak's API. This does not collect media files, for which I have opened an [issue]() and am working on it. I have planned to add a similar scraper for meetup.com too.

The issue in this scraper with scraping the images is that in the event page on eventbrite, they are quite unstructured. This is evident from the following screenshot.



Google Cloud Next Ticke ×

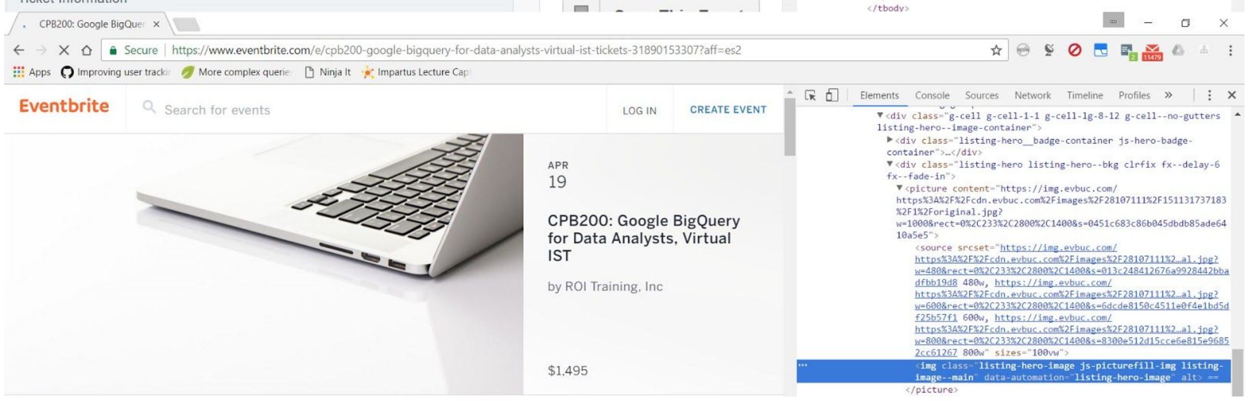
Secure | https://www.eventbrite.com/e/google-cloud-next-tickets-31149669500?aff=es2

Eventbrite | Search for events | LOG IN | CREATE EVENT

Google Cloud Next
Batanai2Create
Tuesday, April 18, 2017 from 9:30 AM to 12:30 PM (CAT)
Harare, Zimbabwe

Ticket Information

```
▼<div id="subheader">
  ▼<table id="subheader_table">
    ▼<tbody>
      ▼<tr>
        ▶<td id="subheader_info_cell" style>…</td>
        ▼<td id="subheader_logo_cell">
          ▼<div class="event_title_image">
              <img src="https://img.evbuc.com/
              https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F27319126%2…pg?
              h=150&w=300&rect=0%2C81%2C1474%2C737&s=e798ad5403cbeda802a7bb
              b64f623219" alt="Google Cloud Next" class="photo cropped js-
              d-retina" data-d-retina-src="https://img.evbuc.com/
              https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F27319126%2F13609271078
              8%2F1%2Foriginal.jpg?
              h=300&w=600&rect=0%2C81%2C1474%2C737&s=2e64db5d50175e8218fd92
              00f828d694" dorsal-guid="2bc0a8d9-2d65-2e94-26d4-
              7ab27fd1cfff" data-xd-wired="retina"> == $0
          </div>
        </td>
      </tr>
    </tbody>
```



CPB200: Google BigQuer ×

Secure | https://www.eventbrite.com/e/cpb200-google-bigquery-for-data-analysts-virtual-ist-tickets-31890153307?aff=es2

Eventbrite | Search for events | LOG IN | CREATE EVENT

APR
19

CPB200: Google BigQuery for Data Analysts, Virtual IST

by ROI Training, Inc

$1,495

```
▼<div class="g-cell g-cell-1-1 g-cell-lg-8-12 g-cell--no-gutters
listing-hero--image-container">
  ▶<div class="listing-hero__badge-container js-hero-badge-
  container">…</div>
  ▼<div class="listing-hero listing-hero--bkg clrfix fx--delay-6
  fx--fade-in">
    ▼<picture content="https://img.evbuc.com/
    https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F28107111%2F151131737183
    %2F1%2Foriginal.jpg?
    w=1000&rect=0%2C233%2C2800%2C1400&s=0451c683c86b045dbdb85ade64
    10a5e5">
        <source srcset="https://img.evbuc.com/
        https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F28107111%2…al.jpg?
        w=480&rect=0%2C233%2C2800%2C1400&s=013c248412676a9928442bba
        dfbb19d8 480w, https://img.evbuc.com/
        https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F28107111%2…al.jpg?
        w=600&rect=0%2C233%2C2800%2C1400&s=6dcde8150c4511e0f4e1bd5d
        f25b57f1 600w, https://img.evbuc.com/
        https%3A%2F%2Fcdn.evbuc.com%2Fimages%2F28107111%2…al.jpg?
        w=800&rect=0%2C233%2C2800%2C1400&s=8300e512d15cce6e815e9685
        2cc61267 800w" sizes="100vw">
        <img class="listing-hero-image js-picturefill-img listing-
        image--main" data-automation="listing-hero-image" alt> ==
      </picture>
```

This problem can be simplified by scraping the images from the event listing page itself, where they lie in a HTML div that is similar for every image.

I also plan to write the scrapers in Python 3.5 . This will help in making the code more readable and IDE-friendly, thanks to the improved tools for type annotations. This will require making the existing codes compliant to Python 3.5, which is not a big task in itself. I hope to complete it in the application period itself.

The scrapers can be developed in any form or any number of scrapers/scraping tools can be added as long as they are in alignment with the Open Event Import API's data format. Writing tests for these against the concurrent API formats will take care of this. This part will be covered by using a **json-validator** to check against a pre-generated schema.

At first, I plan to write scrapers for meetup.com. If I am ahead of my schedule by the end of first milestone or by the end of the coding period, I will add more scrapers as possible. The potential websites are the following, though these will be discussed with the Open Event community then.

➢ [Eventful]()
➢ [Yelp]()

- **REST APIs**

  The scrapers are exposed through a set of APIs, which will include, but not limited to,

  ➢ **api/fetch-event :** to scrape any event given the link and compose the data in a predefined JSON format which will be generated based on Open Event Import API.
  When this function is called on an event link, scrapers are invoked which collect event data such as *event, meta, forms etc.* This data will be validated against the generated JSON schema. The scraped JSON and directory structure for media files:

```
event
meta
forms
microlocations
sessions
session_types
speakers
sponsors
tracks
```

```
    - images\
        - sponsors\
        - speakers\
        - logo.ext
    - videos\
    - audios\
    - slides\
```

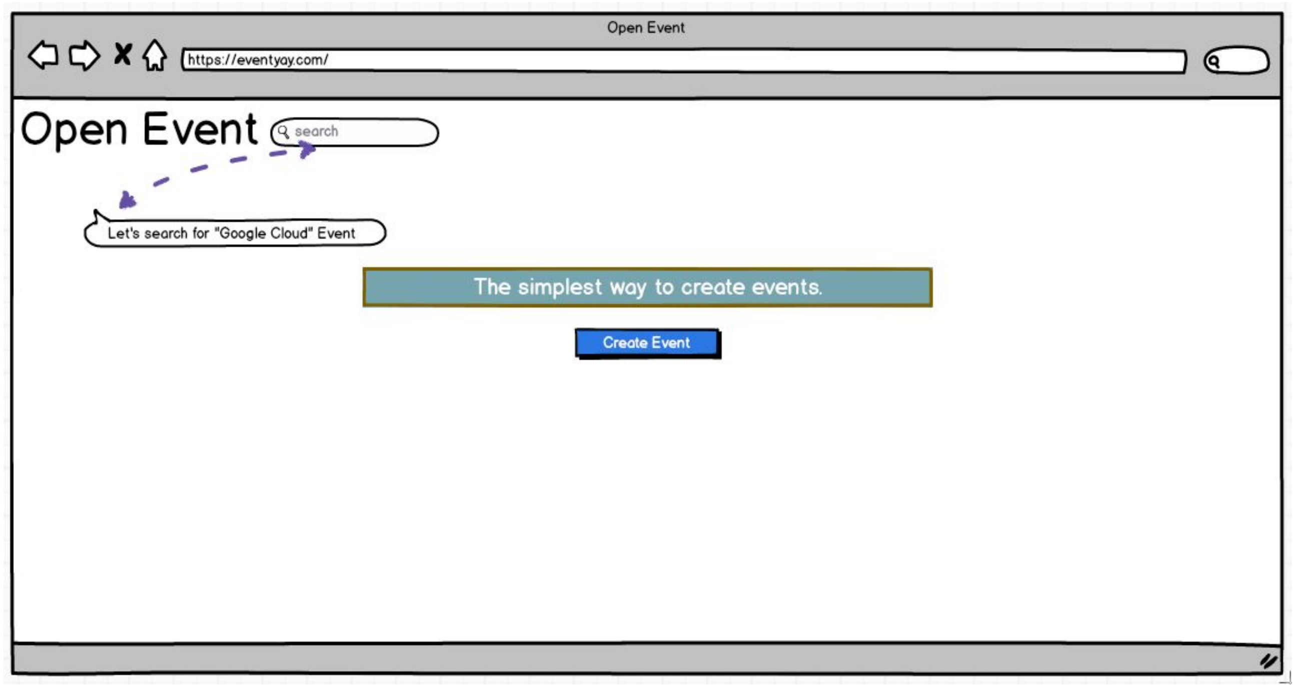
  ➢ **api/export** : to export all the JSON data containing event information into Open Event Server. As and when the scraping is complete, the data will be added into Open Event's database as a new event. This might be written inside **api/fetch-event** mentioned above as well, in which case a separate API would not be required. I will decide this while actually implementing these, based on how things would work then.

Tests

The query-server has automated build integration with Travis CI. I plan to add similar tests to event collect and improve tests on query server. I am new to testing and currently reading materials on unit testing and testing frameworks. I hope to develop a good understanding of these before the end of community bonding period so that I can write tests for my code during the coding period. As such, I am quarter-way through this Udacity course on Software Testing.

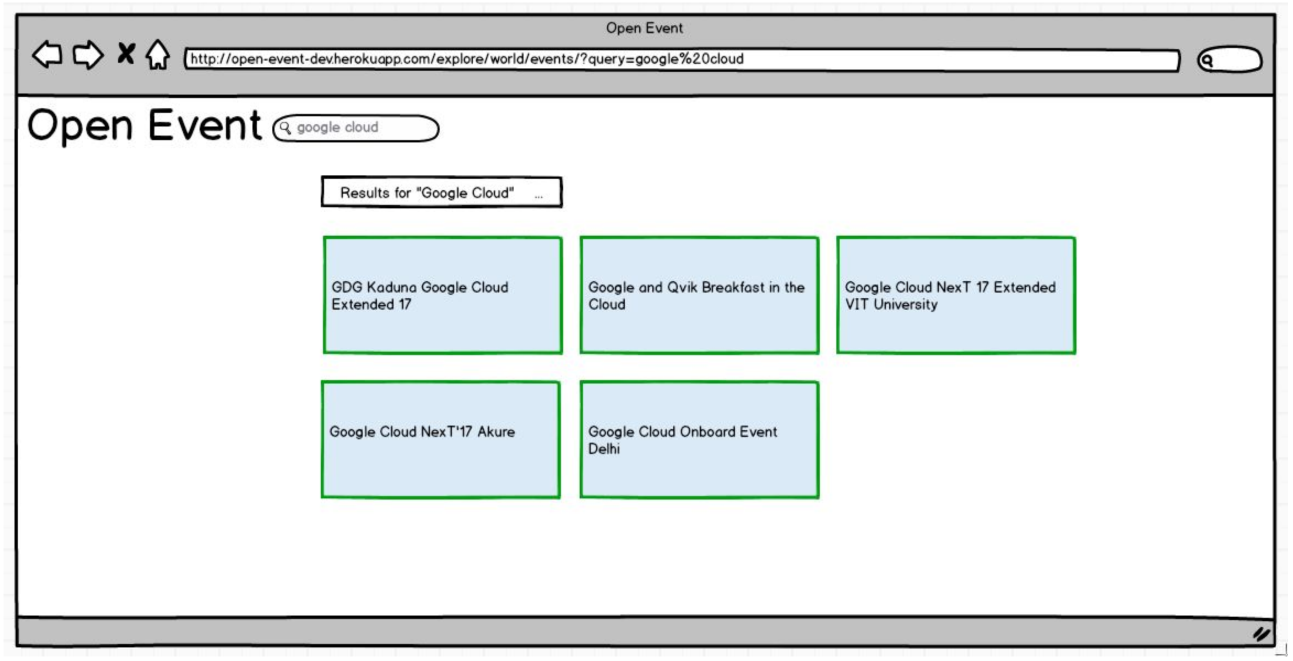
Mock-ups

This is the search page for event-yay. We try searching for “Google Cloud” event.


Open Event
https://eventyay.com/
Open Event
search
Let's search for "Google Cloud" Event
The simplest way to create events.
Create Event

(1)

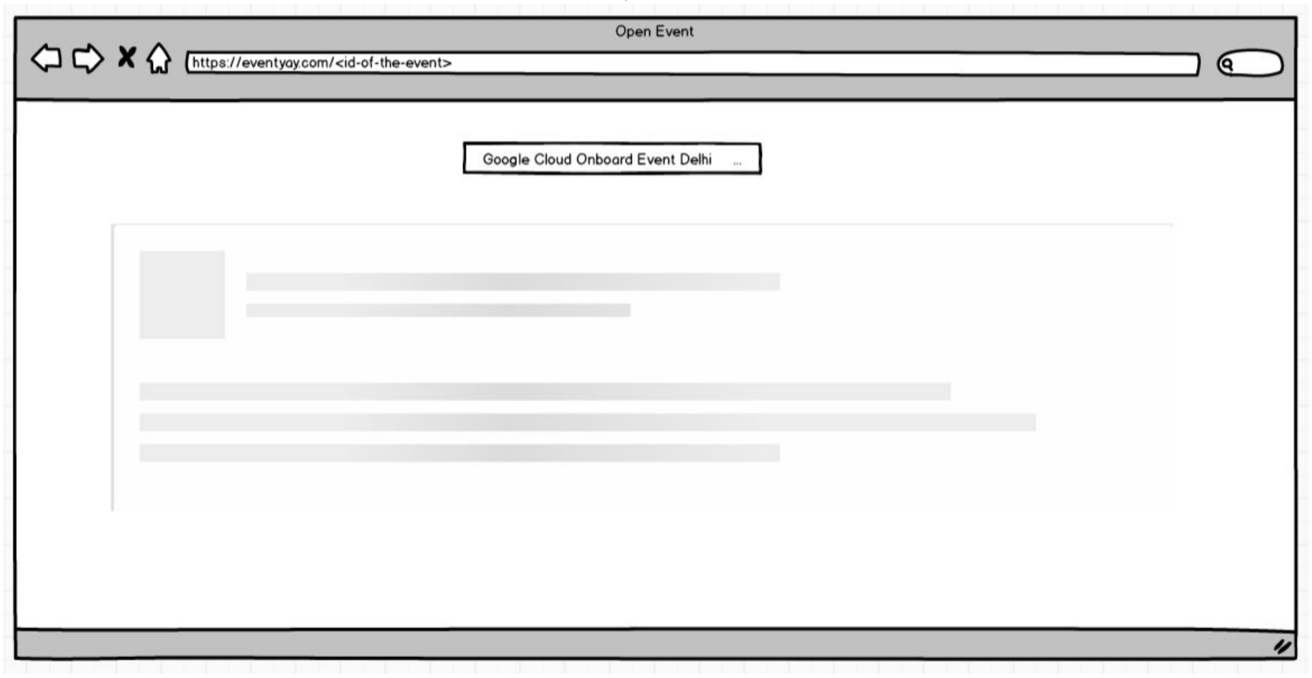
The Search results now enlist all events founds on other sites as well as those already present in the database.


Open Event
http://open-event-devherokuapp.com/explore/world/events/?query=google%20cloud
Open Event
google cloud
Results for "Google Cloud"
GDG Kaduna Google Cloud Extended 17
Google and Qvik Breakfast in the Cloud
Google Cloud NexT 17 Extended VIT University
Google Cloud NexT'17 Akure
Google Cloud Onboard Event Delhi

(2)

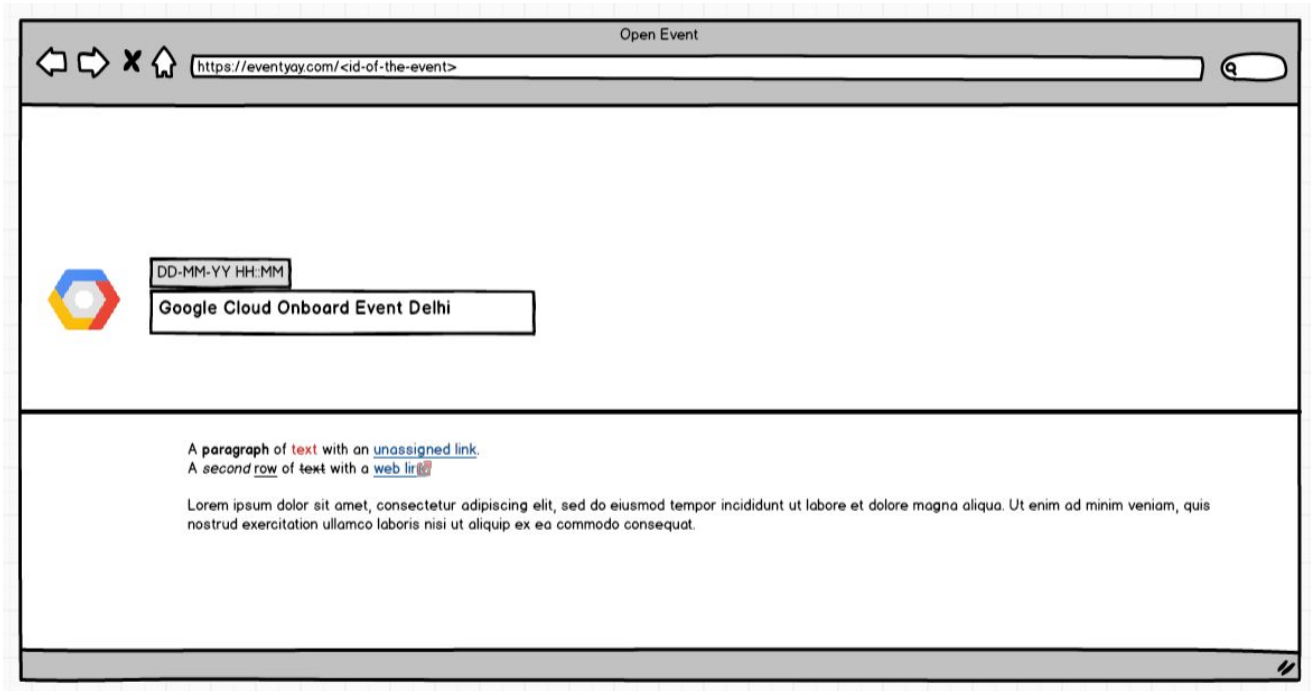
(The results listed in this mock-up are actual events derived from event collect.)

The user clicks on "Google Cloud Onboard Event, Delhi", which we assume is not already present in the database. The page starts loading.


Open Event
https://eventyay.com/<id-of-the-event>
Google Cloud Onboard Event Delhi ...

(3)

Once the events are imported, the page displays the event.


Open Event
https://eventyay.com/<id-of-the-event>
DD-MM-YY HH:MM
Google Cloud Onboard Event Delhi

A **paragraph** of text with an unassigned link.
A *second* row of ~~text~~ with a web link

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

(4)

(These are mockups of http://eventyay.com/**)**

Timeline

Application Period

Present – May 3	● During this time, I would continue working on fixing bugs in Open Event pertaining to Import/Export and hence in-depth study of the code of Open Event specifically the integrations with import/export will be done. There are a few issues still open. ● Fix standing issues with query-server and event-collect. ● Brush up on my Flask skills.

Community Bonding Period

May 4 – May 30	● I will mostly be dealing with JSON format but this period will also be used to read about different formats for open-event import/export. ● Finalize on the UI design for Import Frontend. ● Get in touch with my peers who will also be working on Open Event through GSoC. ● Set up a blog where I will update my progress during the entire Summer of Code period. ● Learn Software Testing and writing unit tests.

Coding Period

May 30 – June 9 *[10 days]*	● Enhance the scraper for eventbrite, adding scraping for images and media files. ● Develop scraper for meetup.com based on the scraper for eventbrite.
June 10 – June 16 *[7 days]*	● Automate generation of a valid JSON schema from the existing import API documentation ● Develop a JSON validator to validate scraped data format.
June 17– June 26 *[10 days]*	● Develop the **api/list** API to get the event title and links

	● I will work on integrating displaying all the events from **api/list** into Open Event frontend. ● Allow for fuzzy searching for events in database
June 26 – June 30	***Phase 1 submissions begin.***
June 27 – July 16 *[21 days]*	● Write the REST APIs for event-collect ● I will develop the import system of events for Open Event using the REST APIs of event collect and Import APIs.
July 17 – July 23 *[7 days]*	● Develop the proper frontend required for displaying an event loading, including a placeholder ● Add displaying event loading in the developed frontend
July 24 – July 28	***Phase 2 submissions begin.***
July 24 – August 13 *[21 days]*	● Integrate importing data from scrapers into Open Event ● Integrate displaying the imported data as an event in Open Event Frontend ● Integrate displaying the imported event in Open Event Frontend
August 14 – August 22 *[9 days]*	● Documenting event-collect APIs, add to improve existing documentation ● Handling edge cases, add tests and fix any outstanding bugs ● Buffer time for completing incomplete parts of the application (if any)
August 22 – August 29	***Wrapping up and Final Project Submission.***

- ➢ *For this timeline, I intend to mark each one of the bulleted points as a deliverable and as a milestone to be used for weekly assignments of Pull Requests during the coding period.*
- ➢ *I plan to allocate at least 40 hours per week on this project and share progress updates with the community.*

Optional Goals

If time permits, I would try to implement the following features also:

- As mentioned in the implementation part, I will add greater number of scrapers as time permits

- Import/Export: Implement handling of import of duplicates
- Implement import support for other formats

If not, these would be my stretch goals for the period beyond GSoC.

Other Commitments

I will be on vacation after my college term ends on May 12 till May 23. I will be available on the chat channels and mailing lists during this period, but not for any actual coding related tasks. This would not affect much as this is during the community bonding period and I have planned initial coding related tasks during the application period. I might set up my blog and blog a little about open source in general and GSoC and FOSSASIA in particular during this period.

In the coding period, my availability might be sporadic during the period starting from July 27 up to August 1 as I will be moving for college. However, I have planned my timeline around this, so it will not affect the project. I will be able to put in the extra hours to compensate for this as I will be available at all other times.

I do not have any other commitments during the coding period.

Future Roadmap

Open Event aims to be developed into a one stop resource for events and a big data hub holding information on all kinds of events across the web. This requires using Machine Learning and Big Data analytics on user data to automatically import events which he/she might be interested in. Adding these technologies is by far the stretch goal for this project. As such, this is not part of the minimum deliverable list of this project. I would love to be involved with Open Event beyond GSoC and once the import system gets into action, I am interested in building Machine Learning capabilities inside the server application.

About Me

I am a second-year Computer Science undergraduate student at Birla Institute of Technology and Science Pilani, Hyderabad Campus, India. I am comfortable with JavaScript and Python and I have been actively involved in web development for almost two years now. Additionally, I have some experience with Flask framework, which is used in Open Event Server. I am also quite comfortable in database handling with MongoDB and PostgreSQL (used in Open Event). I am familiar with other programming languages like JAVA, C++ and PHP and web technologies like Docker and Vagrant.

I am an open source enthusiast and an active member of the coding club and open-source society of my college. I have participated in various hackathons and coding competitions. This includes a MediaWiki Hackathon[4], where I sent some patches to mediawiki-core and a Mozilla Firefox OS Hackathon[5].

I am adept at using git, svn, grep and numerous other Linux development tools for working on large codebases such as that of Open Event itself as well as other small scale personal projects.

Personal Details

Education

- University: Birla Institute of Technology and Science, India
- Degree: B.E. (Hons) Computer Science
- Current year of study: Second Year

Location and Language

- Time zone: UTC+05:30
- Location: Chennai, India
- Language: English, Hindi

Online Presence

- FOSSASIA Slack and Gitter: enigmaeth
- Email address: enigmaeth@gmail.com
- GitHub profile: https://github.com/enigmaeth

Contributions to FOSSASIA

I have been working with FOSSASIA for over half a year now and have contributed to different FOSSASIA projects adding over a 1000 lines of code.

- I have completed two mini projects listed on labs.fossasia.org.
  - ➢ Develop a simple Query Server that stores a query string on a server
  - ➢ Implement a DoubleTerm Check as an Add-on for Query Server

- Specific Contributions
  - ➢ FOSSASIA
  - ➢ Query Server
  - ➢ Open Event Orga Server
  - ➢ Event Collect
  - ➢ searss

References

[1] https://gitter.im/fossasia/query-server
https://gitter.im/fossasia/open-event-orga-server
[2] https://github.com/fossasia/open-event-orga-server
[3] http://open-event-dev.herokuapp.com/api/v1/#/
[4] https://www.mediawiki.org/wiki/Wikimedia_Hackathon_BPHC
[5] https://atmoscsa.wordpress.com/