
GRASA Event Locator Documentation

Release 1.0.0

Team Platypus

Jun 05, 2020

1	About this project	3
1.1	High level project description	3
1.2	Objectives	3
1.3	Project team	4
2	Features and Systems	5
2.1	Families	5
2.2	Program Providers	6
2.3	GRASA and Monroe County Staff	7
2.4	All users	8
3	Open source licenses	9
4	Base System	11
4.1	Change password	11
4.2	Update email	13
5	Event Curation System	17
5.1	How to invite a Provider to sign up	17
5.2	How to review Provider account registrations	19
5.3	How to review and publish Events	21
5.4	How to create a new Event	22
5.5	How to edit an existing Event	23
5.6	How to register a Provider account	24
6	Search System	27
6.1	How to search for Events	27
6.2	How to share an Event	30
7	Deployment architecture	31
8	Admin quick start guide	33
8.1	Dependencies	33
8.2	Set up database	33
8.3	Install your configuration	34
8.4	Container host set-up	34
8.5	Start Event Locator	34

8.6	Stop Event Locator	34
8.7	Restart Event Locator	35
9	Third-party APIs	37
9.1	Geocoding service: MapQuest API	37
9.2	Email/SMTP service	38
10	Upgrade guide	39
10.1	Why you might want to upgrade	39
10.2	Database Concerns	39
10.3	How To Actually Upgrade	40
11	How to: Add new dependencies / libraries	41
11.1	Set up a Pipenv shell	41
11.2	Installing dependencies for testing	41
11.3	Installing dependencies for the team	42
11.4	FAQ	42
12	How to: Conduct user testing	43
12.1	Background	43
12.2	Questionnaire	43
12.3	Tasks	44
12.4	Template	44
13	How to: Exec into a container	45
13.1	Background	45
13.2	Pre-requirements	45
13.3	Commands	45
14	How to: Rebuild search indexes	47
14.1	Background	47
14.2	Pre-requirements	47
14.3	Commands	47
15	Create new dev environment	49
15.1	Requirements	49
15.2	Setup	49
15.3	Run project with docker-compose	49
15.4	Open in web browser	50
16	Refresh existing dev environment	51
16.1	Create a fresh environment	51
16.2	Initial app configuration	51
17	Troubleshooting	53
17.1	Q: On Fedora, Pipenv fails with MySQL config error	53
17.2	Database changes during development	53

This is the main page for the GRASA Event Locator documentation. Other pages are available for you to navigate and explore below. For more information, see the [repo on GitHub](#).

CHAPTER 1

About this project

The GRASA Event Locator is an event locator system to connect Monroe County families to out-of-school programs in the Greater Rochester area. This application was created for the [Greater Rochester After-School Alliance](#) and Monroe County, New York by [Team Platypus](#) at the [Rochester Institute of Technology](#).

1.1 High level project description

This takes the form of a web page containing a map that can be searched for after-school events. Users are presented with further information and registration options. This information was originally delivered via the [Explore Monroe website](#) as well as a physical book, the “[Adult Guide to Youth Services](#)”, both which GRASA and the Monroe County government wish to retire due to lack of maintenance. GRASA representatives presented Team Platypus with several examples, including [Newark Thrives](#) and the [Dallas After-School Network](#) as examples of final deliverables.

The main highlight of the GRASA Event Locator is the map. Similar to the [Newark Thrives Youth Program Locator](#), the end user uses a search box with various filters and an interactive map to locate programs. Events are added to the site by providers via an account system. An administrator from GRASA and the Monroe County government approves these accounts, as well as creation and edits of an event.

For the purposes of this project, the Event Locator does not support programs and events beyond Monroe County.

1.2 Objectives

The system accomplishes the following objectives for these user groups:

1.2.1 Families

- Search for different programs and resources.
- Apply filters to better discover programs and resources that interest them.
- Find information to learn more about a specific program.

1.2.2 Program providers

- Add new programs with specific session info (dates/time/location) into system for approval by administrators.
- Update information for existing programs in system for approval by administrators.

1.2.3 GRASA and Monroe County staff

- Review and approve submitted programs.
- Confirm registration of new provider accounts.

1.3 Project team

The GRASA Events Locator was created by Team Platypus at the [Rochester Institute of Technology](#). This project was facilitated through the ISTE-500/501 Senior Development course offered by the [School of Information](#). The project team members are as follows:

- [DiDonato, Lauren](#)
- [Flory, Justin W.](#)
- [Larrimore, Nathaniel](#)
- [Leong, Harrison](#)
- [Levasseur, Eli](#)
- [Mon, Lei](#)

Features and Systems

This page explains features across all three systems of the Event Locator:

1. Base System
2. Event Curation System
3. Search System

Features are grouped together by which one of the three user groups they impact:

- Families
- Program Providers
- GRASA and Monroe County Staff

2.1 Families

2.1.1 Search for different events.

- **Status:** Implemented
- **Technology decision:** [Haystack](#), [Whoosh](#)
- **Relevant domain(s):** Database
- **Summary of approach:** Use Django Haystack to search the database of events using a keyword match search.
- **Completion criteria:**
 - Unauthenticated user runs successful search query to return results about after-school programs in an area.
 - Information is provided if it exists; the user is informed if it does not exist.

2.1.2 Apply filters to better discover events that interest a user.

- **Status:** Implemented
- **Technology decision:** Haystack, Django Forms, Jinja
- **Relevant domain(s):** Database
- **Summary of approach:** Create Haystack search filters that mirror metadata we save in our database.
- **Completion criteria:**
 - Unauthenticated user alters search query by choosing from a list of preset filters.
 - More detailed information is discovered in search when filters are displayed.

2.1.3 Find information to learn more about a specific event.

- **Status:** Implemented
- **Technology decision:** Django Forms, Jinja
- **Relevant domain(s):** Database, front-end
- **Summary of approach:** Create a unique, addressable page for every event by pulling data from the database and injecting it into front-end HTML with Jinja inclusions.
- **Completion criteria:**
 - Unauthenticated user is able to navigate to an event's unique page from search UI.
 - Any user can share a direct link to event page details with another user.

2.2 Program Providers

2.2.1 Add new events with specific metadata into system for approval by administrators.

- **Status:** Implemented
- **Technology decision:** Django Forms
- **Relevant domain(s):** Full stack
- **Summary of approach:** Create Forms to validate user input and create new events in the events database table.
- **Completion criteria:**
 - Provider is able to log into application.
 - Provider submits new event for review with required details:
 - * Title
 - * Description
 - * Website
 - * Address
 - * Suggested age groups
 - * Activity type

* Etc.

2.2.2 Update information for existing events in system for approval by administrators.

- **Status:** Implemented
- **Technology decision:** [User authentication modules](#) provided in Django, [Django Forms](#) for basic user input
- **Relevant domain(s):** Full stack
- **Summary of approach:** Allow an authenticated user to edit Forms for events they themselves have created.
- **Completion criteria:**
 - Provider is able to log into application.
 - Provider edits an existing event that they already contributed.
 - Event re-enters review queue for admin users.

2.3 GRASA and Monroe County Staff

2.3.1 Review and approve submitted events.

- **Status:** Implemented
- **Technology decision:** [User authentication modules](#) provided in Django, [Django Forms](#) for managing event data, email (via [Mailgun](#))
- **Relevant domain(s):** Full stack
- **Summary of approach:** A status field in the database will be edited for an event depending if it is pending review, approved, or rejected; only approved events appear on the public site.
- **Completion criteria:**
 - Admin is able to log into application.
 - Admin is able to edit a pending event and change its status (approved or rejected).
 - Provider is emailed when their event status is changed.

2.3.2 Confirm new provider accounts.

- **Status:** Implemented
- **Technology decision:** [User authentication modules](#) provided in Django, email (via [Mailgun](#))
- **Relevant domain(s):** Back-end, database
- **Summary of approach:** Allow anyone to self-register on the website using Django Forms to gather information about the user and saving the User object to the database when registration process completes.
- **Completion criteria:**
 - Admin is able to log into application.
 - Registered user is able to log in as a provider once admin confirms user's registration.

2.4 All users

2.4.1 Mobile-friendly user interface.

- **Status:** Implemented
- **Relevant domain(s):** Front-end
- **Summary of approach:** Ensure web application appears correctly on modern smartphones and mobile devices by manual QA testing.
- **Completion criteria:**
 - Core functionality works smoothly on mobile device as it does computing device.

CHAPTER 3

Open source licenses

This project was built with Free and Open Source Software. Thank you to the following projects for making this work possible:

Name	Version	License
Babel	2.7.0	BSD
Click	7.0	BSD
Django	2.2.6	BSD
Jinja2	2.10.3	BSD-3-Clause
MarkupSafe	1.1.1	BSD-3-Clause
PyYAML	5.1.2	MIT
Pygments	2.4.2	BSD License
Sphinx	2.2.1	BSD
Whoosh	2.7.4	Two-clause BSD license
alabaster	0.7.12	UNKNOWN
appdirs	1.4.3	MIT
argon2-cffi	19.2.0	MIT
asgiref	3.2.3	BSD
atomicwrites	1.3.0	MIT
attrs	19.3.0	MIT
bcrypt	3.1.7	Apache License, Version 2.0
black	19.10b0	MIT
certifi	2019.9.11	MPL-2.0
cffi	1.13.2	MIT
chardet	3.0.4	LGPL
commonmark	0.9.1	BSD-3-Clause
coverage	5.0b1	Apache 2.0
django-appconf	1.0.3	BSD
django-compressor	2.3	MIT
django-debug-toolbar	2.1	BSD
django-haystack	2.8.1	UNKNOWN

Continued on next page

Table 1 – continued from previous page

Name	Version	License
docutils	0.15.2	public domain, Python, 2-Clause BSD, GPL 3
entrypoints	0.3	UNKNOWN
filelock	3.0.12	Public Domain < http://unlicense.org >
flake8	3.7.9	MIT
idna	2.8	BSD-like
imagesize	1.1.0	MIT
importlib-metadata	0.23	Apache Software License
mccabe	0.6.1	Expat license
more-itertools	7.2.0	MIT
mysqlclient	1.4.4	GPL
packaging	19.2	BSD or Apache License, Version 2.0
pathspec	0.6.0	MPL 2.0
pluggy	0.13.0	MIT license
psycogp2-binary	2.8.4	LGPL with exceptions or ZPL
py	1.8.0	MIT license
pycodestyle	2.5.0	Expat license
pycparser	2.19	BSD
pyflakes	2.1.1	MIT
pyparsing	2.4.5	MIT License
pytest	5.2.4	MIT license
pytz	2019.3	MIT
rcssmin	1.0.6	Apache License
recommonmark	0.6.0	MIT
regex	2019.11.1	Python Software Foundation License
requests	2.22.0	Apache 2.0
rjsmin	1.1.0	Apache License, Version 2.0
six	1.13.0	MIT
snowballstemmer	2.0.0	BSD-3-Clause
sphinx-rtd-theme	0.4.3	MIT
sphinxcontrib-applehelp	1.0.1	UNKNOWN
sphinxcontrib-devhelp	1.0.1	UNKNOWN
sphinxcontrib-htmlhelp	1.0.2	UNKNOWN
sphinxcontrib-jsmath	1.0.1	BSD
sphinxcontrib-qthelp	1.0.2	UNKNOWN
sphinxcontrib-serializinghtml	1.1.3	UNKNOWN
sqlparse	0.3.0	BSD
toml	0.10.0	MIT
tox	3.14.1	MIT
typed-ast	1.4.0	Apache License 2.0
urllib3	1.25.7	MIT
virtualenv	16.7.7	MIT
wcwidth	0.1.7	MIT
zipp	0.6.0	UNKNOWN

Generated with [pip-licenses](#).

CHAPTER 4

Base System

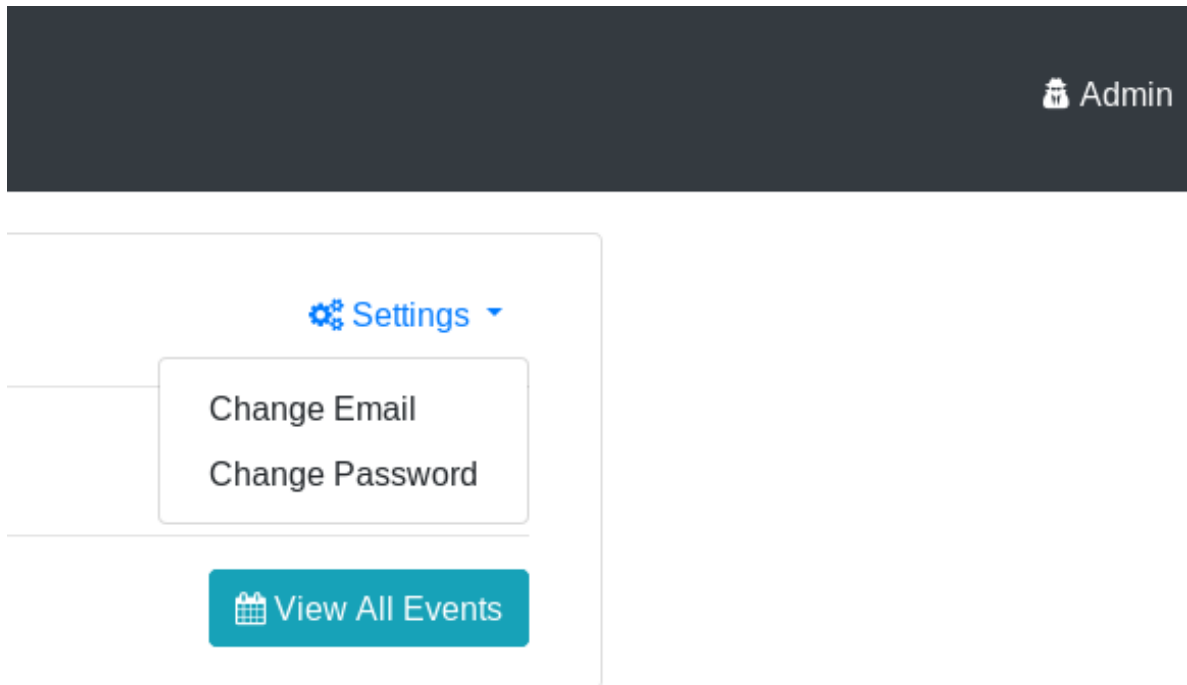
This chapter of **User Documentation** covers the **Base System** of the GRASA Event Locator. In this document, “Users” refers to **Administrator Users** of the application that are responsible for day-to-day review and curation of events in the Event Locator.

4.1 Change password

This page explains how to change the password of either an Admin or Provider account.

4.1.1 Admin User

1. Navigate to *Admin Portal* page by clicking on *Admin* button in top-right corner. *Admin Portal* opens.
2. Look for *Settings* drop-down menu in the center-right part of the screen.
3. Click *Change Password*:



Screenshot

of "Change Password" option in drop-down menu for Admins

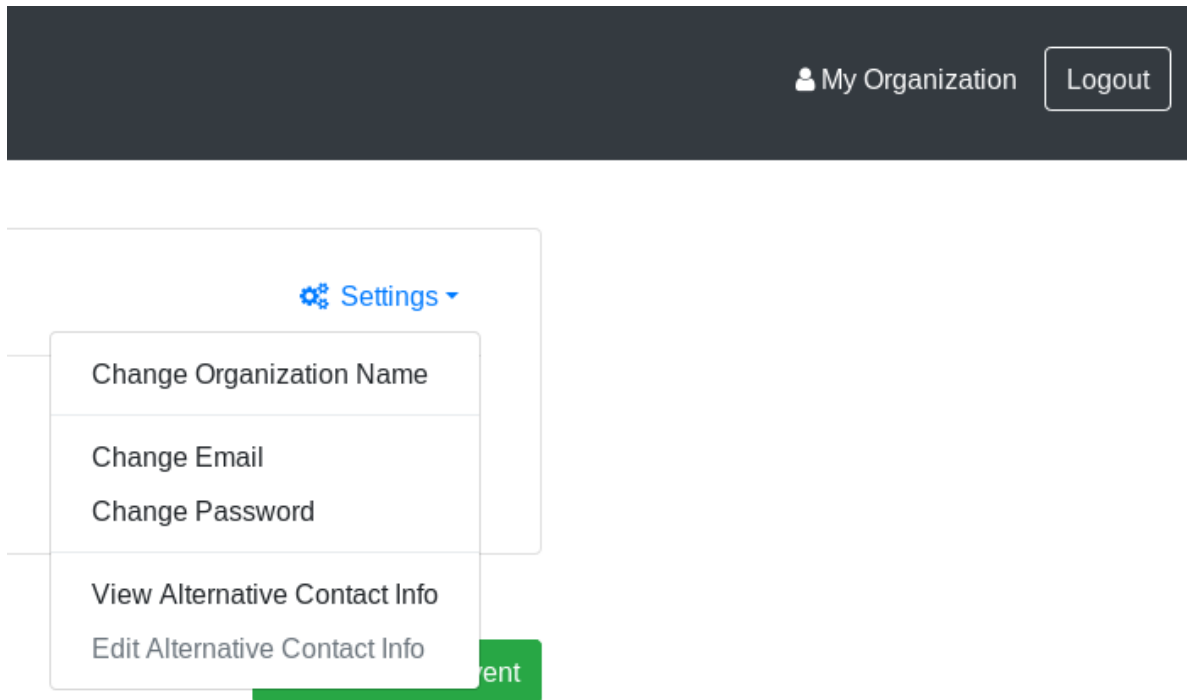
4. The *Change Password* box appears. Enter the current password, the new password, and confirm the new password.
5. Click *Save Changes*.

The password is now changed.

4.1.2 Provider User

The Provider password change experience is similar.

1. Navigate to *Provider Portal* page by clicking on organization name button in top-right corner. *Provider Portal* opens.
2. Look for *Settings* drop-down menu in the center-right part of the screen.
3. Click *Change Password*:



Screenshot

of "Change Password" option in drop-down menu for Providers

4. The *Change Password* box appears. Enter the current password, the new password, and confirm the new password.
5. Click *Save Changes*.

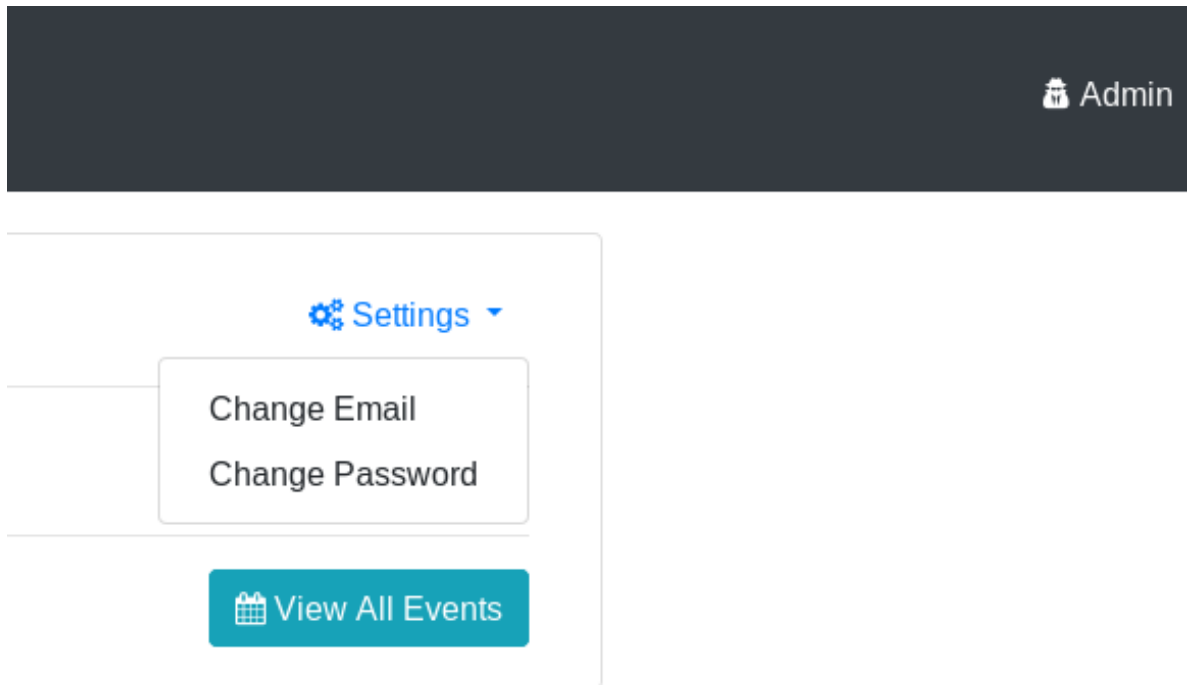
The password is now changed.

4.2 Update email

This page explains how to change the email of either an Admin User or a Provider User.

4.2.1 Admin User

1. Navigate to *Admin Portal* page by clicking on *Admin* button in top-right corner. *Admin Portal* opens.
2. Look for *Settings* drop-down menu in the center-right part of the screen.
3. Click *Change Email*:



Screenshot

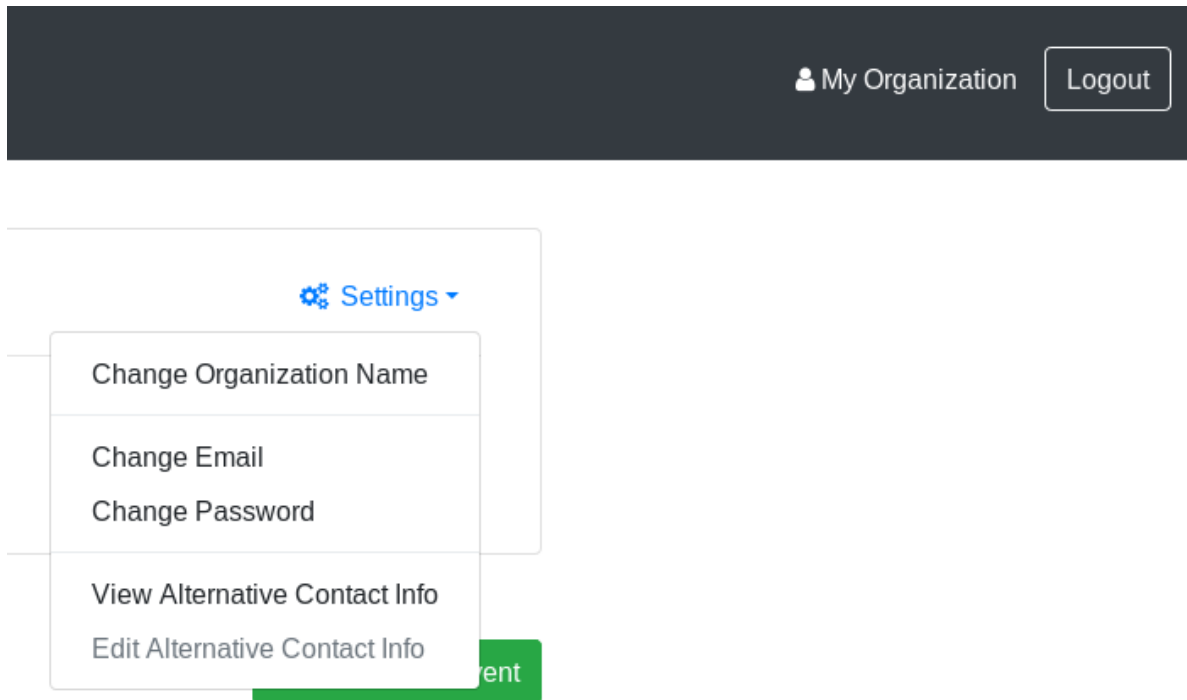
of "Change Email" option in drop-down menu for Admins

4. Enter the new email in the text box and click *Save*.
5. A notification email is sent to the old and new email addresses.

4.2.2 Provider User

The Provider password change experience is similar.

1. Navigate to *Provider Portal* page by clicking on organization name button in top-right corner. *Provider Portal* opens.
2. Look for *Settings* drop-down menu in the center-right part of the screen.
3. Click *Change Email*:



Screenshot

of "Change Email" option in drop-down menu for Providers

4. Enter the new email in the text box and click *Save*.
5. A notification email is sent to the old and new email addresses.

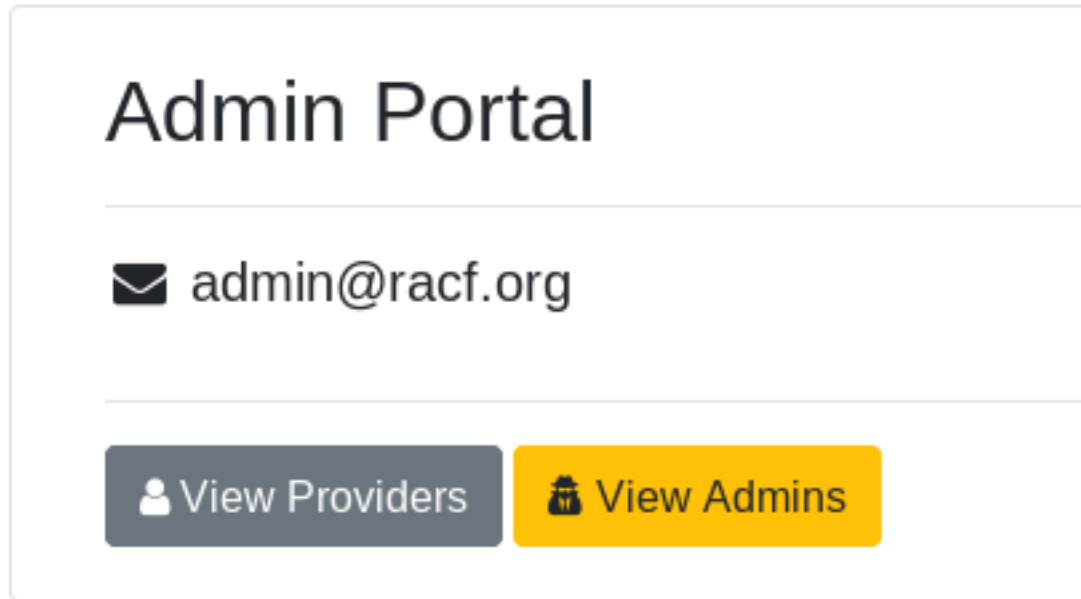
Event Curation System

This chapter of **User Documentation** covers the **Event Curation System** of the GRASA Event Locator. In this document, “Users” refers to **Administrator Users** of the application that are responsible for day-to-day review and curation of events in the Event Locator.

5.1 How to invite a Provider to sign up

This page explains how an Administrator can invite a Provider to sign up for the Event Locator via email.

5.1.1 1. Click *View Providers* button from Admin Portal:



Screenshot

of "View Providers" button from Admin Portal

5.1.2 2. Click *Invite Provider* button in upper-right corner:

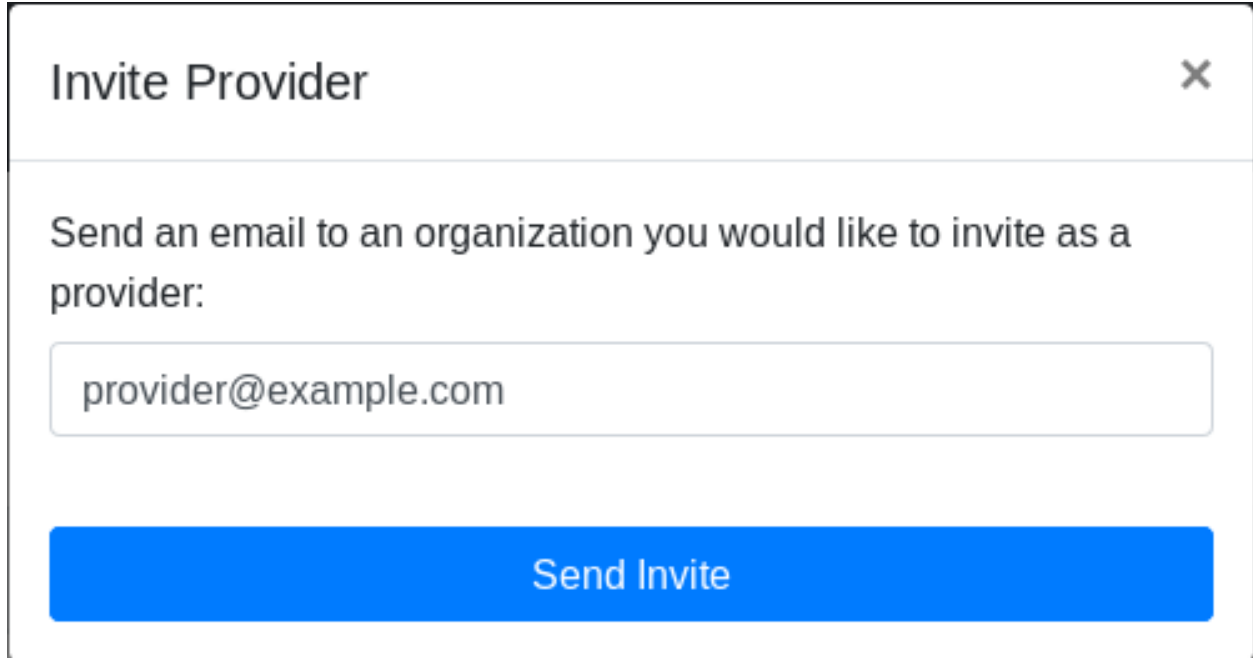
All Providers

← Back to Admin Portal					Invite Provider
Organization Name	Email	Alternative Contact	Number of Events	Delete	
RIT Women in Computing	wic@rit.edu	▲ Lana V. ✉ example@rit.edu ☎ 685-555-5555	4	Delete	

Screenshot

of "All Providers" page, including "Invite Provider" button

5.1.3 3. Enter email and click *Send Invite*:



Screenshot

of prompt for inviting someone via email

5.1.4 4. Wait for registration

Once *Send Invite* is clicked, an email with instructions to register is sent to the email address. It is up to the Provider to finish the registration. Their account is not created when they are invited; the invitation only prompts them register and start adding Events.

5.2 How to review Provider account registrations

This page explains how an Administrator reviews new Provider account registrations. New account registrations are either Approved or Denied.

5.2.1 View pending accounts

If there are pending accounts awaiting review, they appear on the **Admin Panel**:

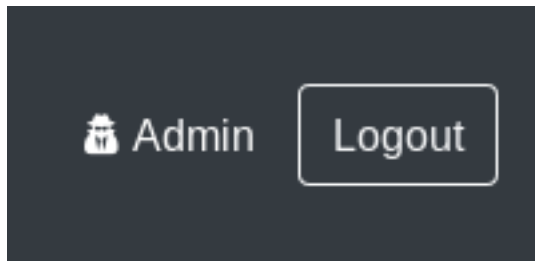
Pending Users

Organization Name	Alternative Contact	Status	Approve	Deny
Sunnyvale High School ✉ Login: jflory7+grasa3@gmail.com	👤 Principal Jakande ✉ principal@example.sunnyvalehigh.ny.us ☎ 575-899-2457	Pending	<button>Approve</button>	<button>Deny</button>

Screenshot

of a user pending review by an Administrator

The **Admin Panel** is accessed by clicking the organization name next to the *Logout* button:



Screenshot of buttons to access Admin Panel and logout

5.2.2 Approve an account

To approve an account, click *Approve* next to their entry in the *Pending Users* table. Once approved, the Provider and their organization contact are sent an approval notification via email.

5.2.3 Reject an account

To reject an account, click *Deny* next to their entry in the *Pending Users* table. You are prompted for a rejection reason:

A modal window titled 'Deny New Provider' with a close button (X) in the top right corner. The main content area has the heading 'Please explain your reason for denying My Organization:' followed by a text input field containing the text: '"My Organization" is not a real organization. Could you please reach out to us with more details at contact@racf.org? Thank you!'. Below the input field, it says 'This will be sent to provider@example.com.'. At the bottom right, there are two buttons: a grey 'Close' button and a red 'Deny Provider' button.

New Provider modal screenshot

Once rejected, an email is sent to the Provider with the provided rejection reason. They are instructed to follow up with an Administrator for info if needed.

5.3 How to review and publish Events

This page explains how an Administrator reviews Events to publish in the Search System. This includes new Events and edits to existing Events. Since the process is identical, these instructions work for both new Events and edits to existing Events.

5.3.1 View pending Events

If there are pending Events awaiting review, they appear on the **Admin Panel**:

New Events

Event Name	Organization	Status	View	Approve	Deny
After School in the Park	Monroe County	Pending	View	Approve	Deny
Student Art Month @ The MAG	University of Rochester MAG	Pending	View	Approve	Deny

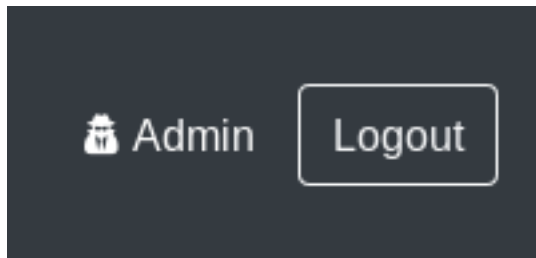
Updated Events

Event Name	Organization	Status	View	Approve	Deny
WiCHacks 2020	RIT Women in Computing	Pending	View	Approve	Deny

Screenshot

of Events pending review by an Administrator

The **Admin Panel** is accessed by clicking the organization name next to the *Logout* button:



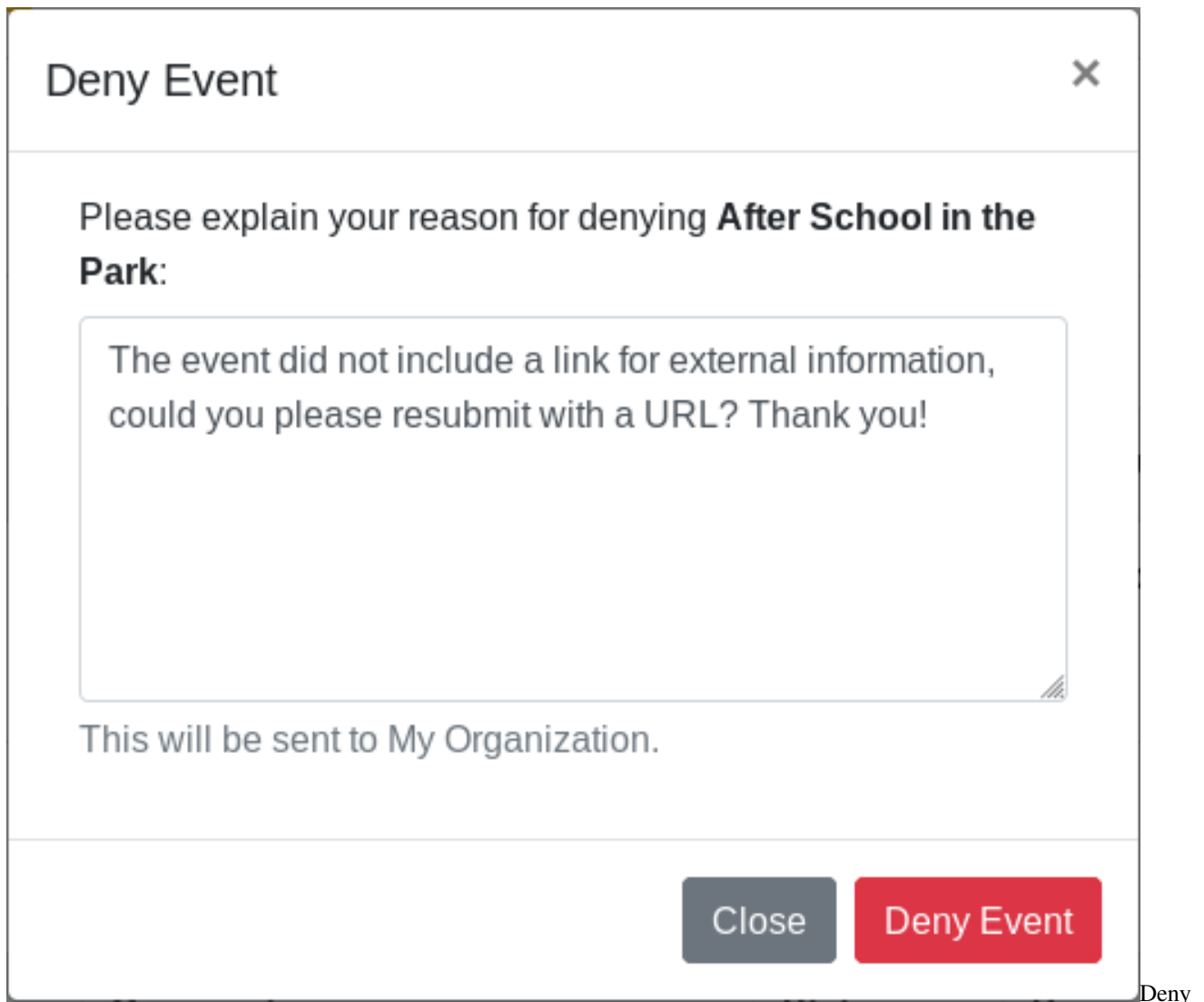
Screenshot of buttons to access Admin Panel and logout

5.3.2 Approve an Event

To approve an Event, click *Approve* next to their entry in the *Pending Users* table. Once approved, the Provider and their organization contact are sent an approval notification via email.

5.3.3 Reject an Event

To reject an Event, click *Deny* next to their entry in the *Pending Users* table. You are prompted for a rejection reason:



New Event modal screenshot

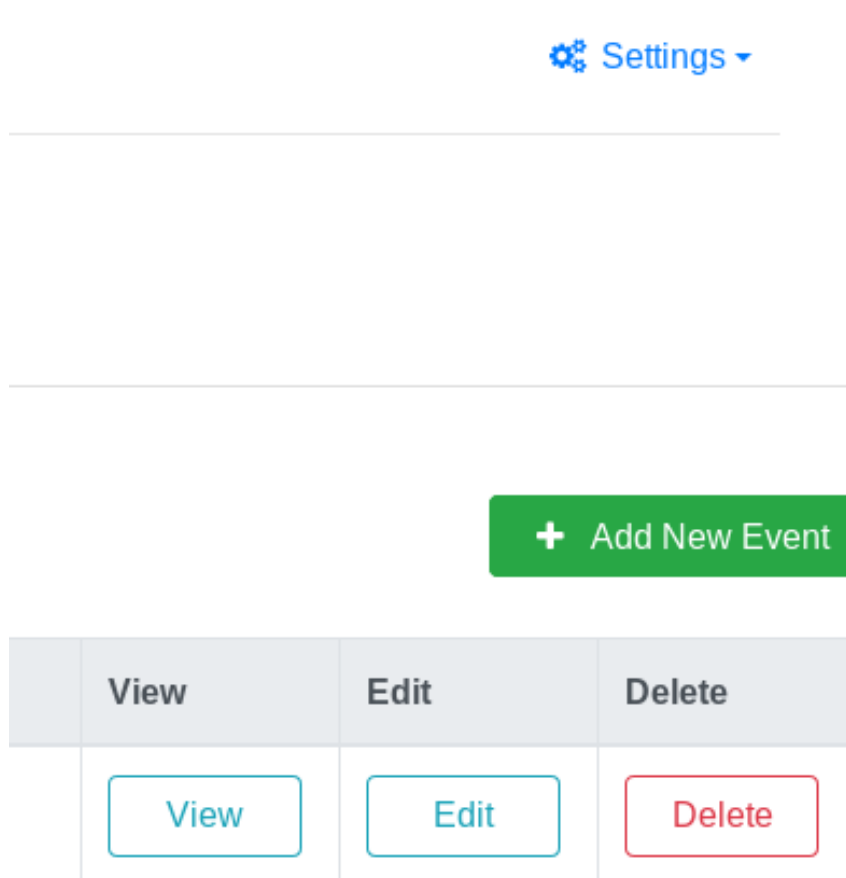
Once rejected, an email is sent to the Provider with the provided rejection reason. They are instructed to follow up with an Administrator for info if needed.

5.4 How to create a new Event

This page explains how to create new Events. Events are searched on a virtual “bulletin board” of events. This explains how to create a new Event and other details about writing evergreen content.

5.4.1 Add new Event

Use the *Add New Event* button from the **Provider Portal** to submit a new Event for review:



Screenshot

of "Add New Event" button in Provider Portal

Complete the form on the next page with details about your Event. Once done, click *Validate Event*. If you receive feedback about corrections, please double-check the data you entered is correct. Your new Event awaits approval by an Administrator user (*admins*: see [“How to review and publish new Events”](#))

5.4.2 How to write strategically

Since dates/times are not collected about events, the more metadata a Provider adds, the better chance an Event is found. The Event Locator is best for recurring events that repeat multiple times in a school year, semester, or term. If the Event happens only once, **provide a good description and website** for someone to learn more about your program, including when (month, day, year, and time). The website field is also great for external ticketing sites, e.g. [Eventbrite](#).

Worried about getting everything right? No worries, you will always have opportunities to edit your Event later.

5.5 How to edit an existing Event

This page explains how to edit existing Events.

5.5.1 Requirements

In order to edit Events, you must be logged in as a Provider user, looking at the **Provider Portal** page. This also assumes you have at least one Event that is already approved.

5.5.2 1. Edit from event list

On the **Provider Portal**, view your list of submitted Events. Choose the one you want to change and click *Edit*:

Events and Programs					+ Add New Event
Program Name	Status	View	Edit	Delete	
WiCHacks 2020	Approved	View	Edit	Delete	

Example

of editing Event (WiCHacks 2020) from Provider Portal

Like with creating new Events, change details of your Event as needed. When done, click *Validate Event* near the lower-right corner. Once submitted, your Event edit awaits review by an Administrator.

5.5.3 2. Wait for Administrator approval

While your edit awaits review, you will see two copies of your Event, both with different statuses:

Events and Programs					+ Add New Event
Program Name	Status	View	Edit	Delete	
WiCHacks 2020	Approved	View	Edit	Delete	
WiCHacks 2020	Edit Awaiting Approval	View	Edit	Delete	

Example

of an Event with an edit awaiting review by an Administrator: Even though two appear, only one Event exists

It is *not* recommended to edit an Event again while an edit is pending approval. While it is possible to edit an Event more than once, changes are not preserved across multiple edits. The full contents of the final edit approved (by an Administrator) is always the published version.

Once an edit is reviewed by an Administrator, the Provider receives an email notification of the status change. If approved, the Event edit is immediately displayed in the public site.

5.6 How to register a Provider account

This page explains how to create a new Provider account in the Event Locator.

5.6.1 1. Get an invitation

Typically, a Provider is invited to use the Event Locator by an Administrator. The invitation email includes a URL to the site registration page (*developers*: `localhost:8000/register.html`). A link can also be found from the *Login* page.

5.6.2 2. Complete registration form

The registration form helps Administrators understand who is registering and on behalf of what organization:

Register as a Provider

Provider accounts must be approved by administrators before events can be made.

Organization Name

Email Address

Password

Confirm Password

☐ Show Password

Alternative Contact Information

Name

Email

Phone

[What's this?](#)

Submit

Already have an account?
[Return to Login](#)

of registration form, taken on 2019 Nov. 17

Screenshot

- **Information about you:**
 - *Organization Name*: organization/entity/program that Provider is associated with
 - *Email Address*: Provider's email address, used for login
- **Information about your organization contact:**
 - *Name*: Name of someone an Administrator may contact in your organization
 - *Email*: Email address of that contact
 - *Phone*: Best phone number for that contact

Complete the requested information and click *Submit*. Your request to register an account was received.

5.6.3 3. Wait for Administrator approval

Because registration is public and an invitation is not required to register, all new accounts must be approved by Administrators. New accounts are reviewed by an Administrator before they are activated (*admins*: see “[How to review Provider account registrations](#)”) Once an Administrator reviews an account, a status change notification is sent to the Provider’s email address.

5.6.4 4. Log in

Once your account is activated, you can log in. Use the *Login* button in the top-right corner to log into the Event Locator. Once you enter your credentials, you are redirected to the **Provider Portal**.

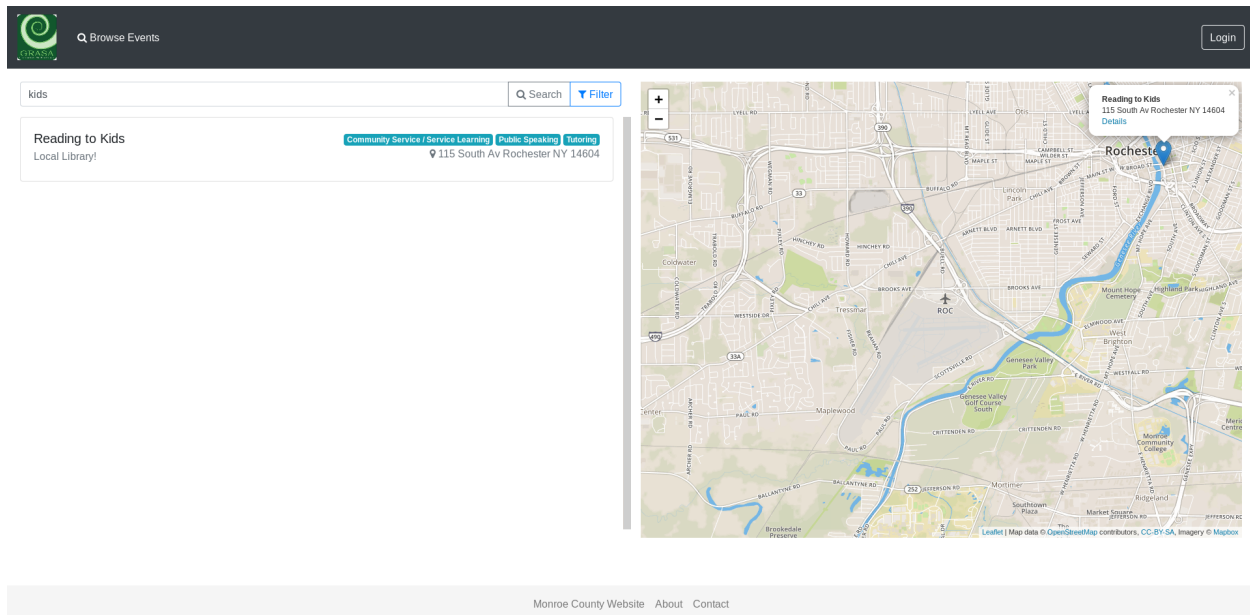
This chapter of **User Documentation** covers the **Search System** of the GRASA Event Locator. In this document, “Users” refers to **Administrator Users** of the application that are responsible for day-to-day review and curation of events in the Event Locator.

6.1 How to search for Events

This page explains how to search for Events in the Event Curation System. It also explains less obvious parts of the Search System.

6.1.1 Make a keyword search

The default home page is the Search System. Type a keyword into the search bar at the top of the page. Click *Search* or press enter to run your search query. Any matches will be shown in the search result list and on the map:



Screenshot

of an example search with keyword "kids"

6.1.2 Apply filters

A search can apply filters to narrow down a search on specific criteria. Types of search filters include the following:

- Activity
- Transportation provided
- Grades served
- Gender
- Fees
- Timing

Search Filters

Activity

+

Transportation

+

Grades Served

+

Gender

+

Fees

+

Timing

-

☐ Before-School

☐ After-School

☐ Evenings

☐ Weekends

☐ Summer

☐ Other-Time

Clear Filters

Apply Filters

Screenshot

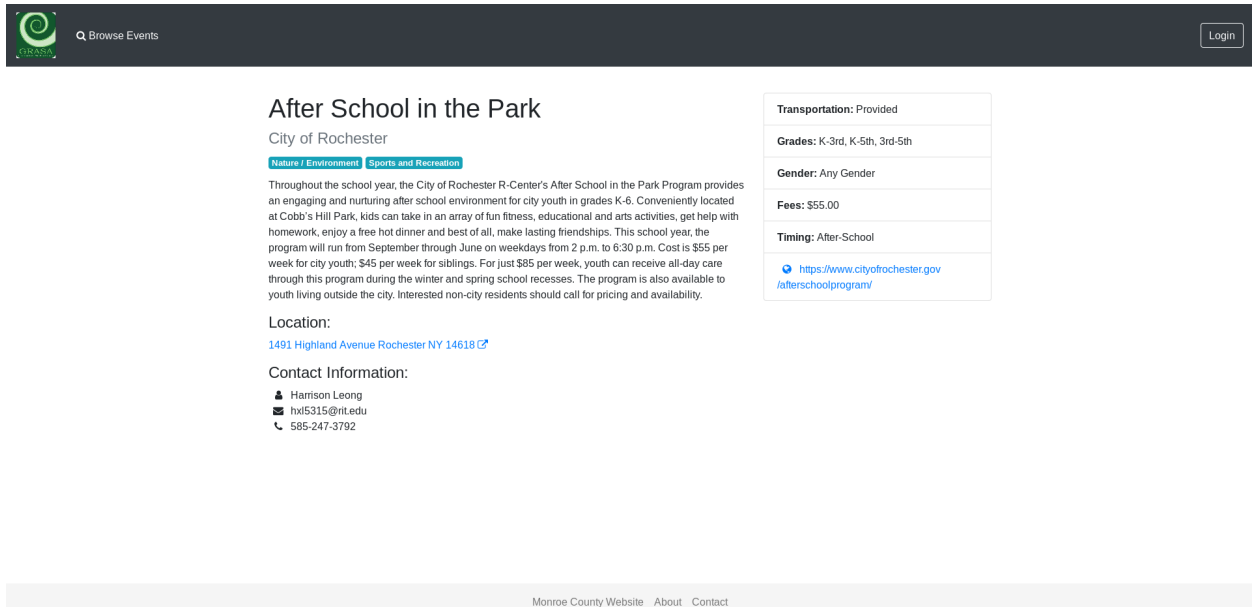
of available filters, "Timing" menu expanded

6.1.3 Other notes about Search System

- Search results are listed in alphabetical order by name of the event.

6.2 How to share an Event

This page explains how to share an Event with someone else, either as an email or on social media. The best way to share an event with others is to load the profile page of an Event. The profile page of an Event is opened by clicking on the Event entry in the search results. The URL is unique to a specific Event and can be shared with others. Someone else will be able to share the detailed view of a specific Event with someone else.



The screenshot shows the GRASA Event Locator interface. At the top is a dark navigation bar with the GRASA logo, a search bar containing 'Browse Events', and a 'Login' button. The main content area displays the event profile for 'After School in the Park' in the City of Rochester. The profile includes a description, location, contact information, and a table of event details.

After School in the Park
City of Rochester

Nature / Environment **Sports and Recreation**

Throughout the school year, the City of Rochester R-Center's After School in the Park Program provides an engaging and nurturing after school environment for city youth in grades K-6. Conveniently located at Cobb's Hill Park, kids can take in an array of fun fitness, educational and arts activities, get help with homework, enjoy a free hot dinner and best of all, make lasting friendships. This school year, the program will run from September through June on weekdays from 2 p.m. to 6:30 p.m. Cost is \$55 per week for city youth; \$45 per week for siblings. For just \$85 per week, youth can receive all-day care through this program during the winter and spring school recesses. The program is also available to youth living outside the city. Interested non-city residents should call for pricing and availability.

Location:
[1491 Highland Avenue Rochester NY 14618](#)

Contact Information:
 ▲ Harrison Leong
 ✉ hxl5315@rit.edu
 ☎ 585-247-3792

Transportation: Provided
Grades: K-3rd, K-5th, 3rd-5th
Gender: Any Gender
Fees: \$55.00
Timing: After-School
https://www.cityofrochester.gov/afterschoolprogram

Monroe County Website About Contact

Screenshot

of an example event profile page

CHAPTER 7

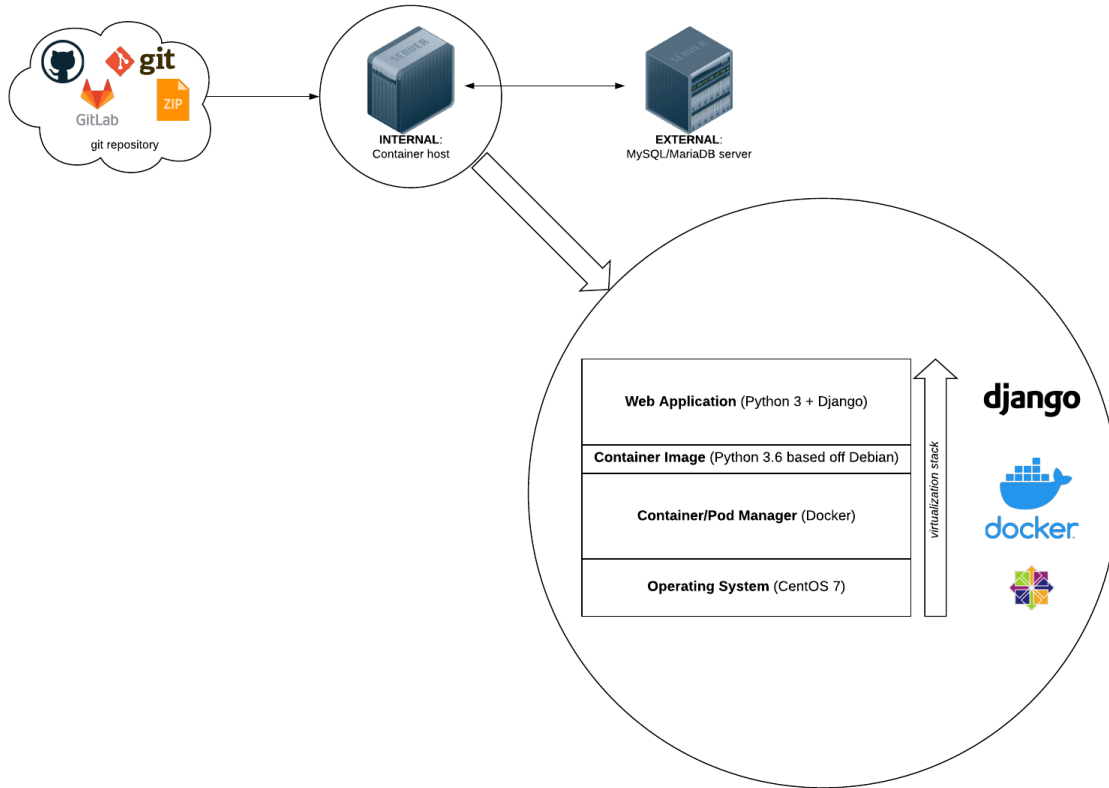
Deployment architecture

This document explains the anticipated deployment architecture of the GRASA Event Locator. At the most simple form, a production environment for the Event Locator will look like this:

GRASA Event Locator Deployment Architecture

Prepared by Justin W. Flory/Team Platypus

Last updated: 2019 Nov. 24



Logos of products and companies are property of their respective copyright holders. ZIP logo provided by icons8.com.

GRASA

Event Locator Deployment Architecture: git repo -> container host <-> database server

The bottom-right circle is a “closer look” at how the Container Host works:

- **Operating System:** CentOS 7 (7.6+ preferred)
- **Container Manager:** Docker (installed to base Operating System as a hypervisor of sorts)
- **Container Image:** Container Manager starts with Container Image pulled from [Docker Hub](#), a repository of Docker container images
 - In this case, the base Container Image is the official `python:3.6-stretch` image, which in turn is based off of a minimal Debian image.
- **Web Application:** GRASA Event Locator source code copied into Container Image and ran as Python 3/Django 2 web application

Admin quick start guide

This page explains how to install and set up the application for a production environment. These instructions focus on *new installations only*.

8.1 Dependencies

The following dependencies must be satisfied for the app to work:

1. CentOS 7.5+
2. [Python 3](#)
3. [Docker](#)

Once the dependencies are in place, place a copy of the project source code (preferably with `git`) in the directory of your choice.

8.2 Set up database

The Event Locator is designed to have a standalone database paired with it. The database is linked to the app through the config file.

To get started, install MySQL/MariaDB and set up a login for the application. On CentOS 7, the MariaDB package can be installed as follows:

```
sudo yum install -y mariadb-server mariadb
```

Create a user and database for the application. Make sure the user has write access to the database. Add these credentials and information (host, username, password, database name) to the config file (see `[config.yml.example]` (<https://github.com/jwflory/django-rit-grasa/blob/master/config.yml.example> “Upstream sample config file” for config file documentation).

8.3 Install your configuration

Copy the `config.yml.example` file to `config.yml` inside of the git repo. Edit the config file to your preferences (see `[config.yml.example]`(<https://github.com/jwflory/django-rit-grasa/blob/master/config.yml.example> “Upstream sample config file” for config file documentation).

NOTE: Any time you make changes to the config file, the container image *must* be rebuilt.

8.4 Container host set-up

We need to build the container with all the code and config file changes, and to do that use this command:

```
python3 up.py --setup
```

What this does is build the container from the `Dockerfile.production` file and migrates the database so that it will have all the necessary tables when the container is started.

8.5 Start Event Locator

Choose the port to run the app on (useful for HTTP vs. HTTPS):

```
up.py --start -p <port number>
```

This command starts the container, and then queries the database to load any already submitted events.

8.5.1 Run initialization script

When the application is installed for the first time, an initialization script must be executed. The initialization script sets up activity filters and creates a first admin user with the `admin_email` from the config file. The initial admin has a default password of `Password1` (please change after first login!). The script will *only* work if two conditions are both met:

1. No admin users already exist
2. Activity filters do not yet exist in database

Thus, it is important this step is followed first.

The initialization script is triggered by navigating to the following URL:

```
<site_name>/initial_setup
```

`_site_name` is also managed in the config file. For example, in development, this might be `localhost:8000/initial_setup`.

8.6 Stop Event Locator

If you need to stop the app for any reason, this can be done with this command:

```
up.py --stop
```

8.7 Restart Event Locator

If for any reason you need to restart the container without rebuilding, you can use this command:

```
up.py --restart -p <port number>
```

Third-party APIs

This page describes third-party API services used in the application, how they are used, and how to set them up.

9.1 Geocoding service: MapQuest API

An API token for this service is required for the map functionality of the Event Locator to work.

The MapQuest API translates an address provided by a Program Provider to a pair of latitude/longitude coordinates. These coordinates are used in the front-end to create markers on the map for events.

9.1.1 How it is used

The MapQuest API token is set in the configuration file (see `config.yml.example`). The secret token is automatically used in front-end HTML files. Administrators only need to maintain the API token in the production config file.

The API request is made when an address is added or changed on a new event or an edited event.

9.1.2 How to acquire API access

Register for a [MapQuest Developer API key](#).

9.1.3 Cost considerations

A free API key is limited to 15,000 requests a month. The API is only used when a new event is created or an existing event is edited (explained above). Therefore, it is highly unlikely for the API to graduate to a paid tier unless the application grows significantly. The next price tier is 30,000 requests a month for \$99/month; see [pricing and plans](#) for more information.

9.2 Email/SMTP service

The Event Locator also needs an SMTP server for dispatching email. This information is managed in the config file (see `config.yml.example` for an example). If an existing SMTP mail server already exists, it may be used in the application.

However, in the event that an on-premise SMTP server is not available, we suggest using any of the following services and their email APIs:

- [Mailgun](#) (about, pricing)
- [SendGrid](#) (about, pricing)
- [Mandrill](#) (about, pricing)

10.1 Why you might want to upgrade

1. if a new version were released or
2. You have new features you have developed that you would like to move into production.

Note: If you're adding your own development features, this guide assumes you have fully tested everything.

10.2 Database Concerns

It is *highly* recommended to make a backup of your database before making any upgrades. If the new version you are upgrading to has database schema changes, then you will need to upgrade your migrations file and remigrate your database. This will most likely wipe your data. The following scenarios are examples of where data may or may not be deleted:

- Deleting a model. If the user is doing this, there's guaranteed data loss for the obvious reason; you're deleting a table that had app data; that's the reason the table was there in the first place.
- Adding a model. There's no data loss from this; adding a table shouldn't affect the app's behavior, at least assuming no changes to the code.
- Removing columns from an existing table. Obvious data loss here, a whole column is being removed.
- Adding columns to an existing table. Adding some types of columns to a table will force you to also specify a default value (ie `contact_name = models.CharField(max_length=255, default="Name not Specified")`). Some will not (ie `models.TextField`). No data is lost, though if a column that requires a default is added to a table, existing entries will now have that default value in the column post-migration.

It is recommended to start a new database alongside your current database and update your config file with the new database information. Then you can copy your old data over. **This should be tested before moving new code to production using a copy of your data.** You have been warned.

10.3 How To Actually Upgrade

The upgrade process is rather simple. The steps for upgrading to a new version of the GRASA Event Locator are as follows:

1. Stop the current running application by running:

```
./up.py --stop
```

2. Move the new code into location. This can either be done by copying all the code into the directory using git, and downloading the new production code from a remote repository, or by downloading a zip file with all the code into a new directory and essentially starting over.
3. Rebuild the container to now include the new code using this command:

```
./up.py --setup
```

4. If the rebuild step is successful, then start the application with this command:

```
./up.py --start --port <port number>
```

If your data is going to remain intact and all your users are still there, then you should not need to go to `<site_name>/initial_setup` again. However if you're starting with a new, empty, database you will need to browse to `<site_name>/initial_setup` again.

How to: Add new dependencies / libraries

If we use a new library or framework, we need to **install it as a dependency** in the project. Dependencies are specific versions of third-party software we want to use in the project. Sometimes we want to test something out, without adding it to the project for everyone. Other times, we want to install something to the project for everyone. This guide explains both below:

11.1 Set up a Pipenv shell

First, for either option, start a new Pipenv shell for your local development. The Pipenv shell sets up a **Python virtual environment (virtualenv)** for you. This way, you can install Python packages via `pip` without special user privileges, among other reasons.

Using a command-line tool of your preference, change directories to the project git directory and run the following commands:

```
pip3 install --user pipenv
pipenv shell
```

Pipenv will set up a virtualenv for you. Once it is done, you can now install packages using `pip` into the virtualenv.

11.2 Installing dependencies for testing

If you are trying out new libraries or third-party software, but aren't sure if you want to keep it yet, use `pip install`.

Simply use `pip install <package name>` to install any package from the Python Package Index (PyPI). The package you install is available on your workstation, but not for everyone else. This lets you test something out without committing everyone to using it yet.

11.3 Installing dependencies for the team

If you are confident in using a library or third-party software, use `pipenv install`.

Once you know a package and are confident in keeping it, install it as a project dependency. This makes it available for everyone when they run `docker-compose build` again. Otherwise, when someone runs your changes, the Django app will crash because of a missing dependency.

There are two steps to installing new dependencies:

1. `pipenv install [--dev] <package name>`
2. `pipenv update --dev`

Run the first command with the `--dev` flag if installing a dependency only for development environments, not production. The second command updates the `Pipfile.lock` file with your changes and ensures the versions installed locally match what is specified in `Pipfile.lock`.

11.4 FAQ

11.4.1 When testing dependencies, should `pip` or `pip3` be run?

If in doubt, use them exactly as written above.

When you install `pipenv`, this guide assumes you are running Python 3. `pip3` is an explicit process call to your system's installed Python 3 interpreter. If you do not have Python 2 installed, `pip` probably does the same thing. Later, after opening a `pipenv shell`, the Python virtualenv rewrites the meaning of `pip` to whatever the specified `python_version` is in `Pipfile`. So long as you are in the shell, `pip` will *always* mean the same thing as `pip3`.

How to: Conduct user testing

This is a how-to article to explain how to run user testing sessions for the GRASA Event Locator. It intends to be a blueprint for these sessions but it is not exhaustive.

12.1 Background

User testing is a helpful way to collect feedback about how people use and interact with our application. To date, Lei Mon conducted user tests early on in the development cycle. This guide is formed from her notes and questions. For consistency's sake, **make sure user tests are as consistent as possible** across sessions.

12.2 Questionnaire

These are questions to answer during user testing. Feel free to add new ones through a pull request:

1. User demographics
 - Age, race/ethnicity, occupation, etc.
 - Ask interviewee how they identify
2. Estimated background with technology?
3. Overall familiarity with desktop or mobile devices?
4. What are users' thoughts while viewing search/filter on the homepage?
5. How easy or difficult was it to navigate across the pages?
6. Is there anything the user does not understand?
7. What are users' overall thoughts on the design and layout?
8. How long did it take the user to complete a task?
9. User's overall satisfaction with website?

10. What did the user like the most and the least?
11. Any problems finding the information wanted by the user?
12. Any difficulties using it on mobile devices?
13. Does the user prefer a mobile or desktop view?

12.3 Tasks

This is an idea list of tasks you can run a user through, depending what user they simulate.

12.3.1 Admins

- Change Site Logo
- Approve new user
- Reject new user
- Approve new event
- Reject new event

12.3.2 Providers

- Login
- Register
- Forgot Password
- Create Event
- Edit Event
- Change Organization Logo

12.3.3 Families

- Filter for an Event
- View Event

12.4 Template

CHAPTER 13

How to: Exec into a container

This how-to article explains how to open a shell inside of a Docker container for advanced debugging capabilities.

13.1 Background

Containers add a large of abstraction to working on the project. Or said differently, they hide many parts of your environment, compared to running it locally. Sometimes you need more advanced commands or functionality to debug a tricky problem. This guide teaches how to debug issues by opening a shell inside of a container.

13.2 Pre-requirements

Docker and **Docker Compose** should be installed. This guide is written for the **command-line interface** of Docker. This varies across operating systems, but refer to [Docker docs](#) for more help.

Start the local containers with `docker-compose up`. In a new window, note the names of the running containers (`docker ps`); you will need the name of the `web` container later.

13.3 Commands

```
$ docker exec -it <container name> <"echo 'Some command here'">
$ docker exec -it web_1 /usr/bin/bash
```

How to: Rebuild search indexes

This is a how-to article to rebuild the Whoosh/Haystack search indexes when new events are added to the database.

14.1 Background

New events are added to the database periodically. For the new events to appear in search, the index needs to be rebuilt. Once the index is rebuilt to include new events, they will appear in searches.

14.1.1 Development note

Automatic rebuilding of search indexes is planned. These steps are mostly intended for local development or to force a index rebuild.

14.2 Pre-requirements

If running the project as a container, use `docker exec` to open a shell inside the web app container. See the [How to: Exec into a container](#) doc for more info of how to do this.

If not running the project as a container, get to the environment where your Python 3 + Django installation exists.

14.3 Commands

If building index for first time:

```
python3 manage.py rebuild_index
```

If refreshing for new data, e.g. a new event:

```
python3 manage.py update_index
```

Create new dev environment

This page explains how to set up a new, local development environment to test the GRASA Event Locator system.

15.1 Requirements

- Docker
- docker-compose

15.2 Setup

A configuration file must be provided at start, either as a `config.yml` in the root directory of the project or a path specified as a `CONFIGPATH` environment variable. For local development, run the following command to get started with development:

```
cp config.yml.example config.yml
```

15.3 Run project with docker-compose

`docker-compose` is used for local development. It is convenient since it gives you a MySQL container to work alongside the application; you do not have to set one up. Use the following commands to build the containers and then start them:

```
docker-compose build
docker-compose up
```

These commands may require `sudo` depending on your operating system and installation option.

15.3.1 Run docker-compose in detached mode

Detached mode disables an output stream to your terminal. In other words, detached mode will not display logs in the terminal window while running. Use the following command to run in detached mode:

```
docker-compose up --detach
```

To shut down `docker-compose` in detached mode, use this command:

```
docker-compose down
```

15.4 Open in web browser

Once `docker-compose` is running, open a web browser. Visit localhost:8000 to view the site running locally.

CHAPTER 16

Refresh existing dev environment

Document owner: Harrison Leong (@leong96)

Are you working on this project, but afraid you messed something up? This page explains how to do a full reset and start with a fresh environment:

16.1 Create a fresh environment

1. Make sure the application is stopped (`docker-compose down`).
2. Delete the database Docker volume (`docker volume rm django-rit-grasa_djangograsa_db`).
3. Make a copy of `config.yml.example` and name it `config.yml`.
4. Rebuild the project containers and start the app (`docker-compose up --build`).
5. In a separate shell window, run `docker ps`, which shows all running containers. They should be *mariadb* and *django-rit-grasa*.
 - If for some reason, they are not both there, use `docker ps -a` to get the name of the list of all containers, including the non running ones, then `docker start [containerID]` to start them.
6. Run the command `docker exec -it [containerID for the django container] /bin/bash`.
7. Run `python3 manage.py migrate`
8. Run `python3 manage.py rebuild_index`
9. Visit localhost:8000 in a browser

16.2 Initial app configuration

These directions only apply to local development. See the *Admin Quickstart* for a production deployment.

1. Visit `localhost:8000/admin_user`
2. Visit `localhost:8000/create_database`
3. Site is now usable with the following admin account:

- **Username:** `grasatest@yahoo.com`
- **Password:** `Password1`

Note that the MySQL/MariaDB container is started automatically with a `grasa_event_locator` database, but this will not happen automatically in production.

This page is a collection of miscellaneous tips, tricks, and other tidbits of info to make it easier to do troubleshooting on the application.

17.1 Q: On Fedora, Pipenv fails with MySQL config error

On Fedora, `pipenv install` may fail with the following error:

```
OSError: mysql_config not found
```

Install the `mariadb-connector-c-devel` package. It includes the `mysql_config/mariadb_config` binary needed to install the `mysqlclient` library.

On Fedora:

```
sudo dnf install -y mariadb-connector-c-devel
```

17.2 Database changes during development

Occasionally, you get an error from the database changing (field not found, category matching query does not exist). You can check this by checking if `models.py` was changed recently.

Open a shell to the Django container by *exec'ing into the app container*. Run the following commands:

```
python manage.py makemigrations
python manage.py migrate
```

Try the task again. If it works, make sure the generated database migration script is committed to the git repository along with your other changes. If it does not work, try *refreshing your development environment*.