**CCS User Guide**

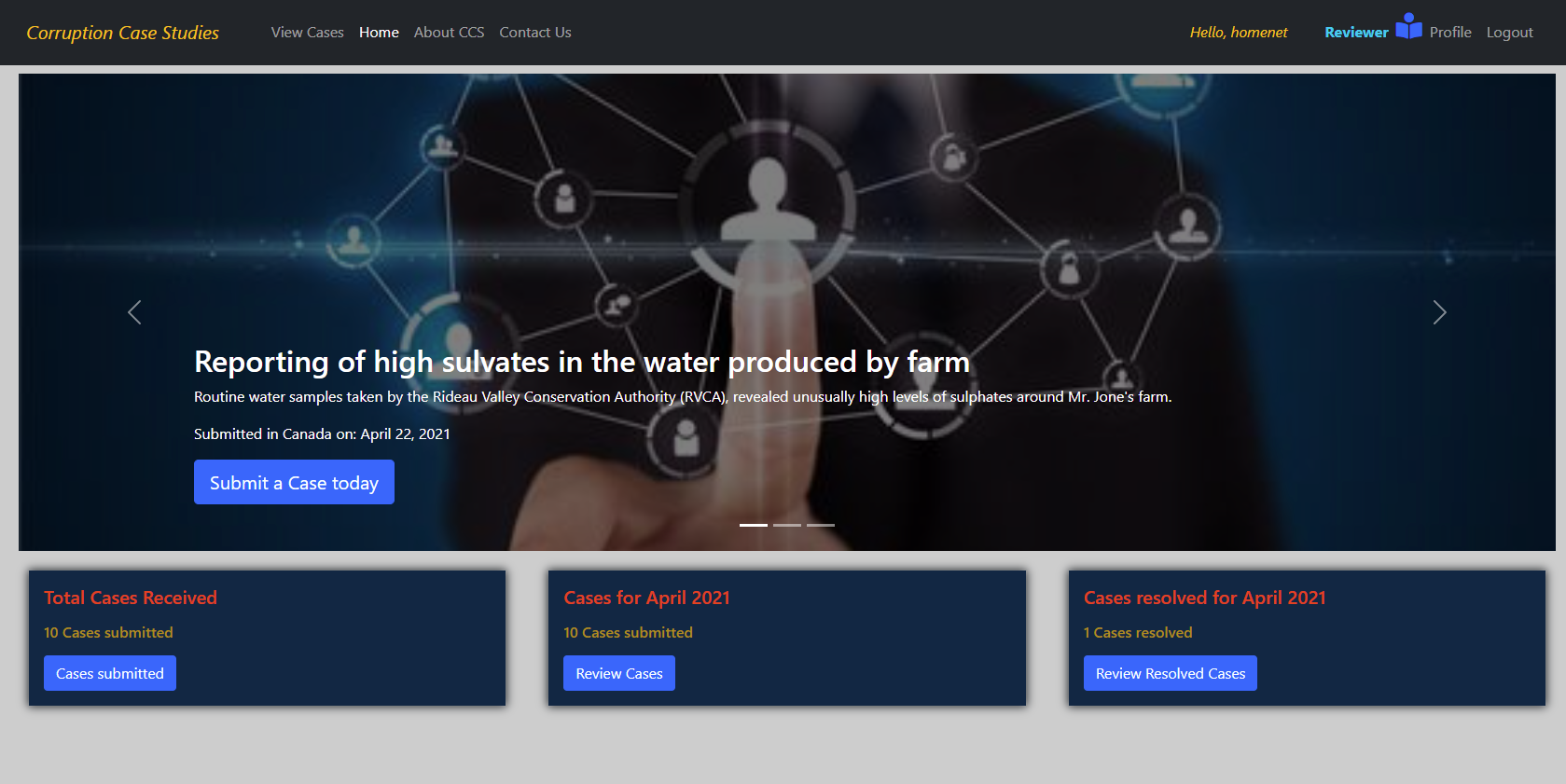
Prepared for CST8334

Created by Robert Lange and Alexander Riccio

The Corruption Case Studies (CCS) management information system application was developed in collaboration with the Canadian Center of Excellence for Anti-Corruption (CCEAC) and Algonquin College to be used to share in cooperation, real life situations from those who have faced or are facing corruption or potential corruption. The goal is to track, maintain and ultimate resolve such cases through this tracking mechanism and to employ the case study scenarios as a learning aid to participating agencies, ultimately uncovering potential patterns.

This User Guide was created to introduce the application’s key functional elements, configuration and ancillary tools. The application can run standalone or hosted with a Docker application container.

CCS was developed using Python within a Django framework and relies heavily on supporting libraries and the package management system (pip) written in Python for Django. The database backend relies on PostgreSQL either running within the container itself (provided), or external standalone database application. Within the Docker container, PostgreSQL database management can utilize the PgAdmin4 management tool to facilitate database administration activities.



Contents

[Installation within Docker 3](#_Toc70410431)

[Downloading the Image 3](#_Toc70410432)

[Running the Image 4](#_Toc70410433)

[Updating Your Site Domain 4](#_Toc70410434)

[Creating the Docker Image 6](#_Toc70410435)

[Entrypoint script 7](#_Toc70410436)

[Dockerfile Configuration Explained 8](#_Toc70410437)

[Using the application 10](#_Toc70410438)

[Home Page 10](#_Toc70410439)

[Account Management 11](#_Toc70410440)

[Account registration 11](#_Toc70410441)

[Roles 11](#_Toc70410442)

[Login and Reset Password 12](#_Toc70410443)

[CCS Admin 13](#_Toc70410444)

[Default Admin Account 14](#_Toc70410445)

[Reset Default Admin Password 14](#_Toc70410446)

[Creating New Reviewers and Admins 15](#_Toc70410447)

[Database Administration 21](#_Toc70410448)

[Postgres pgAdmin4: 21](#_Toc70410449)

[Scenario Workflow 25](#_Toc70410450)

[Submitting a Scenario 25](#_Toc70410451)

[Reviewing a Scenario 25](#_Toc70410452)

[Editing a scenario 26](#_Toc70410453)

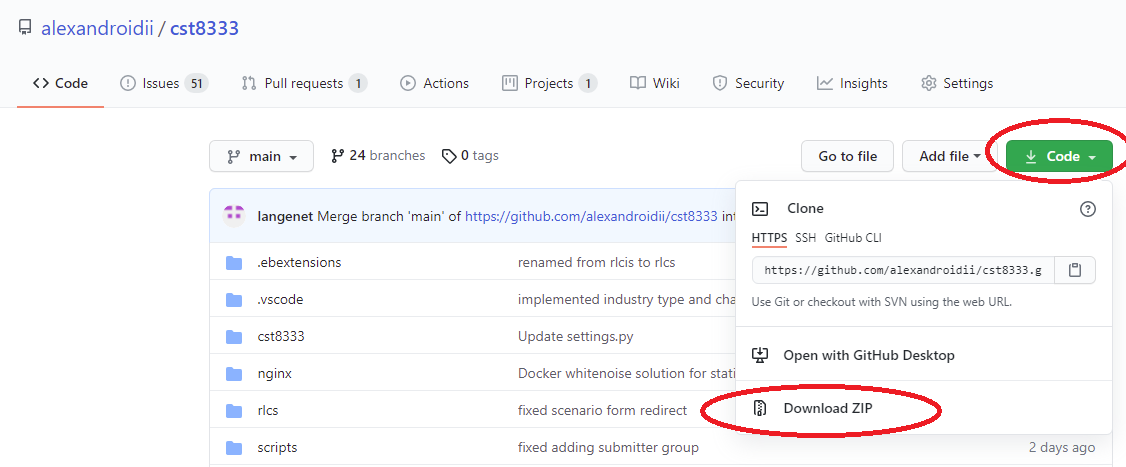
[Issues, Considerations and futures 27](#_Toc70410454)

# Installation within Docker

## Downloading the Image

1. To first run the application, you will need to install Docker at the following location:

<https://docs.docker.com/get-docker/>

1. You can either download the entire RLCS repository from Github by following these instructions, or skip to step 3:
   1. Go to <https://github.com/alexandroidii/cst8333>
   2. Click on the  button and then on to download the entire application.  
        
      
   3. Extract the contents of the folder on your local file system **(example c:\rlcs)**
2. If you did not download the entire application (not needed to run only to develop) go to the following file in the Github repository and copy the file to your local system:
   1. <https://github.com/alexandroidii/cst8333/blob/main/docker-compose.yml>
   2. On your local file system, create a file called “docker-compose.yml”. For example, in the folder **c:\rlcs\docker-compose.yml**
   3. Copy the contents from the docker-compose.yml file in the link in step 3.a into the file you created in step 3.b
   4. Go to the following <https://github.com/alexandroidii/cst8333/blob/main/.env.dev> link
   5. Copy the contents of the .env.dev file into a file on your local file system called “.env.dev”. For example, in the folder **c:\rlcs\.env.dev**
   6. Update the .env.dev file with your own domain information as well as email SMTP info.

## Running the Image

1. From a command line, run the following commands:
   1. docker images
      1. Note: If this command does not work, docker did not install correctly so you will need to return to step one to finish installing it
   2. cd c:\rlcs
      1. Note: If you gave your local folder a different name at step 2.c or 3.b then use that folder name instead
   3. docker login
      1. Note: Not sure if you need to login for this to work as it’s a public image.
   4. docker-compose up
      1. Note: you might be prompted by Docker to access local files. You can click yes or accept on that prompt.
2. You should be able to access the site now by going to <http://127.0.0.1:8000>

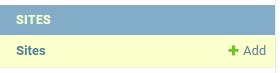
## Updating Your Site Domain

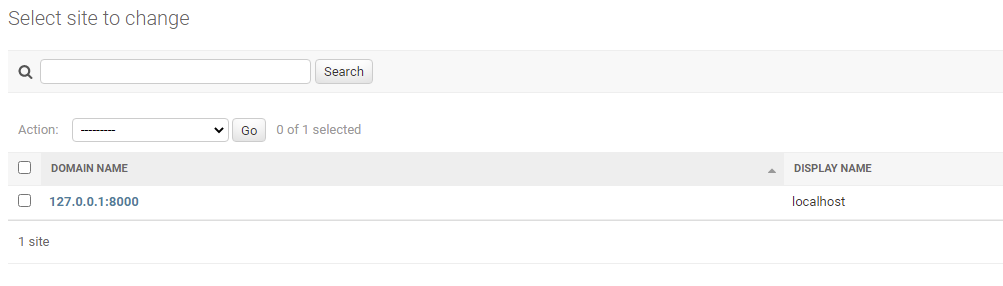
This is an important step of the application specific to the email functionality, as the hosted application makes use of the fully qualified domain name when sending email. This becomes important as users sign up, change passwords and review the Case Study (Reviewers only). Without a proper domain name attached to any outgoing emails which may include links back to the application, any users required to activate their accounts via the tokenized link back to the application will fail.

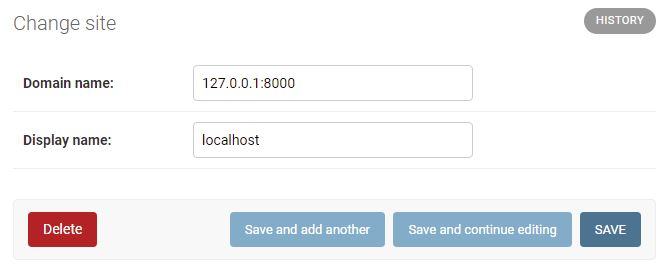
As with any application, it is highly recommended to change the Super-User credentials upon first logging into the application having Administrator (Admin).

You will need to log into the application with the default Admin credential ‘admin@admin.com’ and ‘12345678’ as a password, click on the “Sites” link and modify the site address to represent your own.

The following figure within the Admin pages depicts the Sites properties used. See the section on the application administration.







## Creating the Docker Image

If you need to rebuild the docker image, run the following commands from a windows command terminal:

#Push an image to Docker Hub:

# Build the container locally

docker-compose up --build

#List all containers

docker ps -a

#Login to docker

docker login

#commit image where b317f7acf873 is the container id you found when running "docker ps -a"

docker commit b317f7acf873 alexandroidii/cst8333-rlcs:1

#Confirm you see the new image:

docker images

#push the image up to docker hub

docker push alexandroidii/cst8333-rlcs:1

#pull the docker images ready to load

docker pull alexandroidii/cst8333-rlcs:1

#run the pulled images

docker run -it alexandroidii/cst8333-rlcs:1

Note: You will need to create your own Docker Hub account and use it to push up your image.

The current Docker image is located at:

<https://hub.docker.com/repository/docker/alexandroidii/cst8333-rlcs>

## Entrypoint script

Below are the contents of the Dockerfile and all relevant settings.

Entrypoint.sh has the commands used during the container initialization. When the application starts, it executes the commands within the Entrypoint.sh script to perform rudimentary tasks which include:

**Collectstatic**: This collects all relevant static files and images and places them in the static folders within the container image from the application’s root.

**Migrate**: As the application starts and particularly when started the first time, the database schema needs to be created with the application’s required database tables. Subsequent start-ups don’t recreate the schema, its existence is tested during initialization.

**Createsu**: In order to administer the system, a Django Superuser is required. This invoked Python script is executed to create the account.

**Dropdows**: Since the database schema is created with empty tables/entities, the data is required to populate is obtained via a Django Fixture.  A fixture is a collection of data that Django knows how to import into a database. The most straightforward way of creating a fixture if you've already got some data is to use the manage.py dumpdata command. ... Each time you run loaddata , the data will be read from the fixture and re-loaded into the database. The Fixture type used here is in yml format. Other options include XML if desired.

**Firstscenario**: A basic ‘dummy’ case is created which is then rendered in the landing page carousel. Without it, only the cards below the carousel are displayed. This results as the carousel are populated via a database query which instantiates the objects within the carousel – with no data, this isn’t rendered in the html view.

## Dockerfile Configuration Explained

Dockerfile**:**

3.8-alpine – version of light-weight Linux

ENV PYTHONUNBUFFERED=1 PYTHONUNBUFFERED to a non empty value ensures that the python output is sent straight to terminal (e.g. your container log) without being first buffered and that you can see the output of your application (e.g. django logs) in real time. (Optional)

ENV PATH="/scripts:${PATH}" Add scripts to the path of the running container

COPY ./requirements.txt . Copy requirements file include all the necessary library packages used in the running application container.

RUN apk update

RUN apk add postgresql-dev gcc python3-dev musl-dev Add postgresql dependencies

RUN apk add --no-cache --virtual .build-deps build-base linux-headers \

&& pip install Pillow Pillow dependency for image uploads

RUN pip3 install -r /requirements.txt Read in required libraries

RUN apk del .build-deps Remove temp folder

WORKDIR /rlcs\_app Set the working directory app name

COPY . . Copy the contents of the project into the container

COPY ./scripts /scripts Copy script useful for docker image

RUN chmod +x /scripts/\* Add executable flag to anything in the scripts folder

RUN mkdir -p /vol/web/media Creates new directories inside docker image (any images that get uploaded to the Django application) i.e. Scenario documents

RUN mkdir -p /vol/web/static Static file folder required for JavaScript, CSS etc

RUN adduser -D user Create a user to run the application within the container image (best practice in docker is to run application with a user who has less privilege than root user

RUN chown -R user:user /vol Set permission on the above volume for the user, user recursively throughout the directory

RUN chmod -R 755 /vol/web User has full access. Groups and other - read, execute

RUN chown -R user:user /rlcs\_app Change ownership of app folder to user

RUN chmod +x /rlcs\_app/\* Change to executable everything in app folder

USER user Switch to the new user we created

CMD ["entrypoint.sh"] Script used to start that application (run uWSGI and start the application) uWsgi is a tool used to run Python applications in production

**Contents of entrypoint.sh**

python manage.py collectstatic -noinput Collect all the static files for project and put them in the static root. Proxy (Nginx) can serve the static files more efficiently

python manage.py migrate Runs migration on container startup

python manage.py createsu Creates Superuser in the container

python manage.py loaddata dropdowns.yaml Populates all the dropdown fields in the cases app model

uwsgi --socket :8000 --master --enable-threads --module cst8333.wsgi

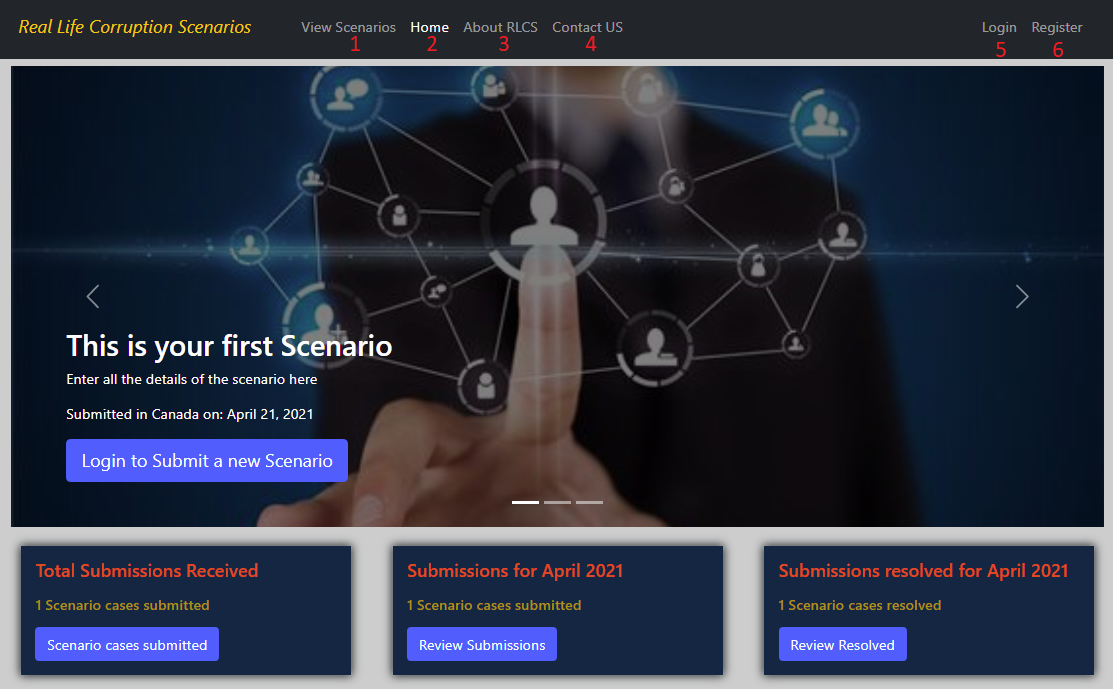
Command that runs the application using uWSGI (starts project and runs it in a uWSGI service)

Run it in the foreground serving the application rather in the background task

# Using the application

## Home Page

When you access the application for the first time, there are a few defaults that have been created to start you off.



You are greeted on the home page with some metrics and the latest 3 scenarios that have been submitted.

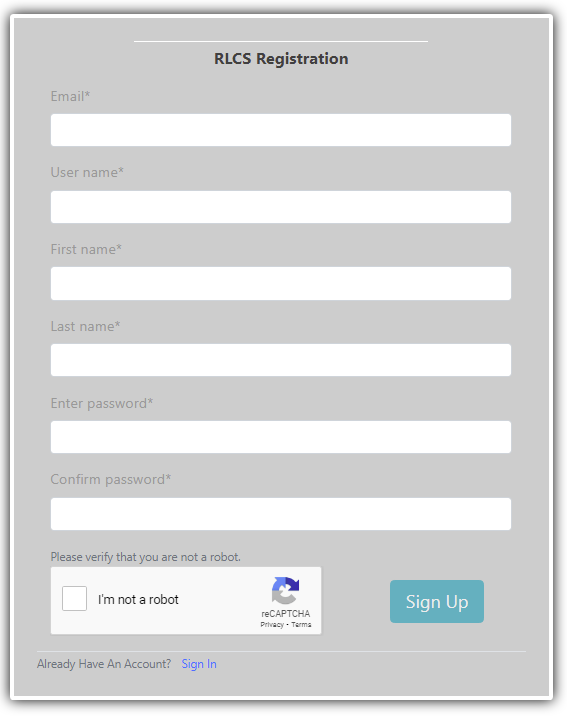
From here you can:

1. **View Scenarios:** A list of all the reviewed scenarios. If you are logged in, you will also see a list of scenarios you’ve submitted that have not been published publicly yet, and if you are a reviewer or admin, you will see all scenarios submitted, even if they are not published.
2. **Home:** return to the home page
3. **About RLCS:** a page that describes the purpose of RLCS
4. **Contact Us:** redirect to the CCEAC contact us page
5. **Login:** login to RLCS if you already have an account
6. **Register:** if you don’t have an account, you can register for one here.

# Account Management

## Account registration

Account registration requires the following information:



Once you’ve signed up, you will receive an email with a verification link. Click on that link to activate your account before you can login for the first time.

## Roles

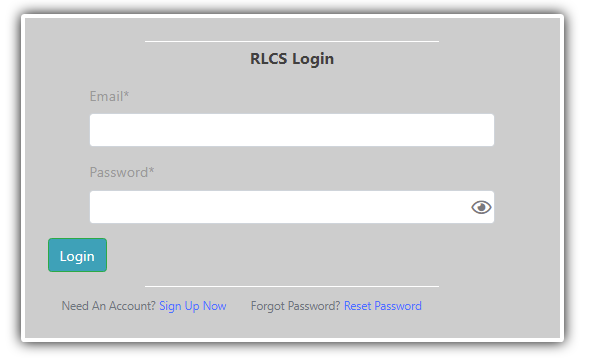
**Submitter:** By default, all accounts created this way will have the **Submitter** role, which gives them access to submit new scenarios and edit their own scenarios. When a new scenario is submitted, it is not public yet until a user with the Reviewer role publishes it.

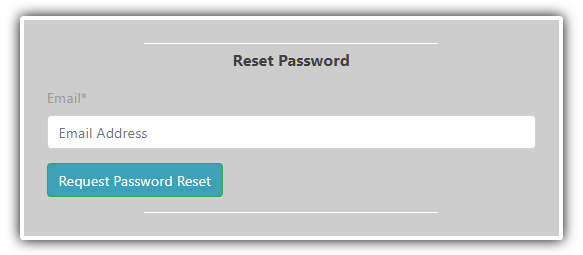
**Reviewer:** This role has the ability to review scenarios before **Publishing** them.

**Admin:** This is a Super User who can access the Admin Console and modify dropdown values, create new users, and assign roles.

## Login and Reset Password

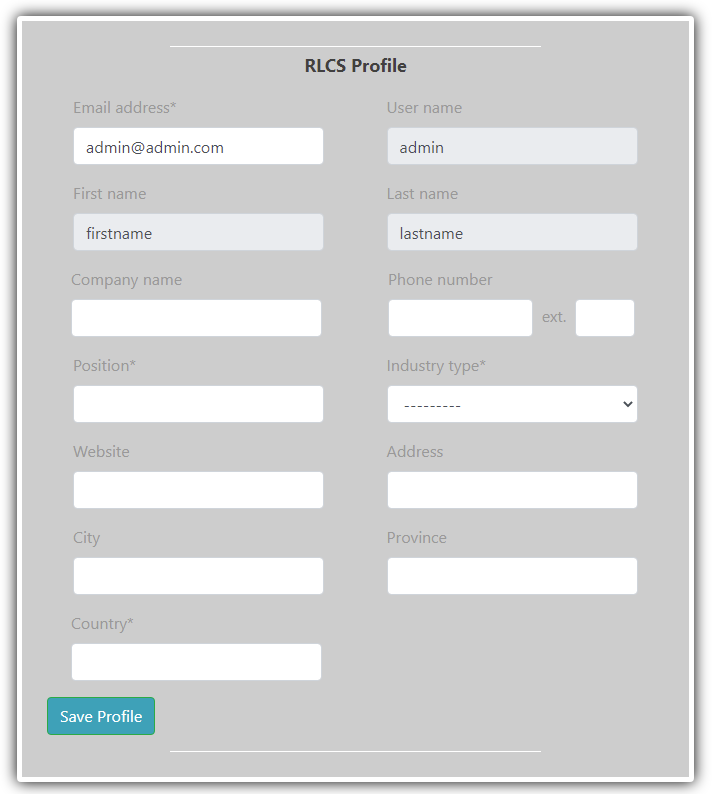
The login and reset password process is straight forward. Enter your credentials and click **Login**, or click on the **Reset Password** link.





An email will be sent with instructions on how to reset your password once you enter your email and click on **Request Password Reset**

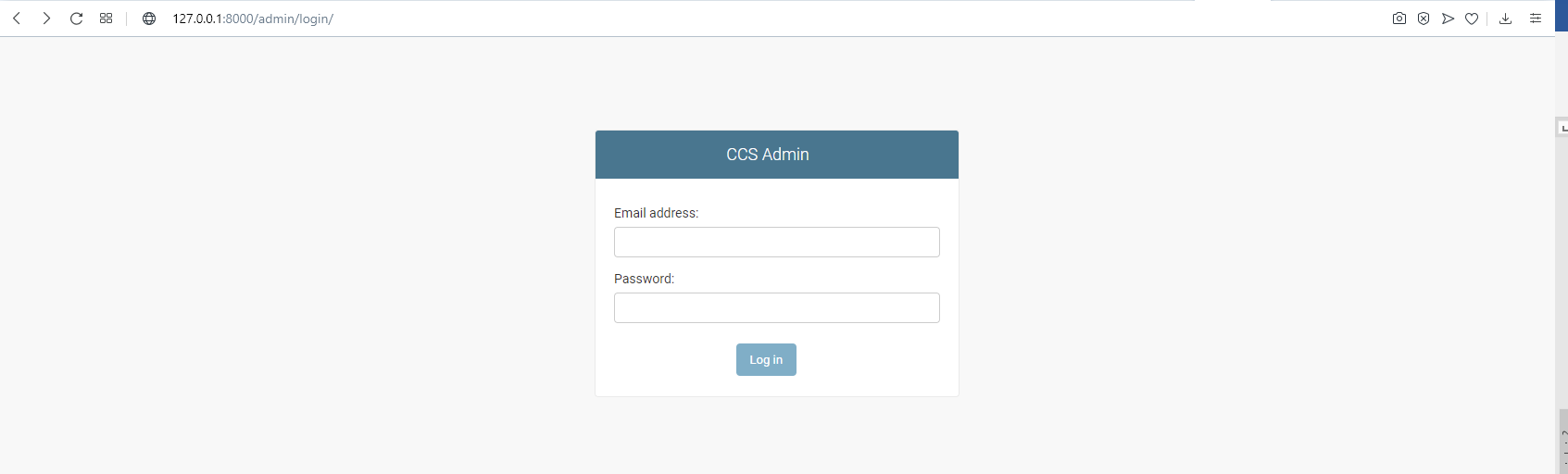
Once you login, you will be presented with a profile page to complete additional details.



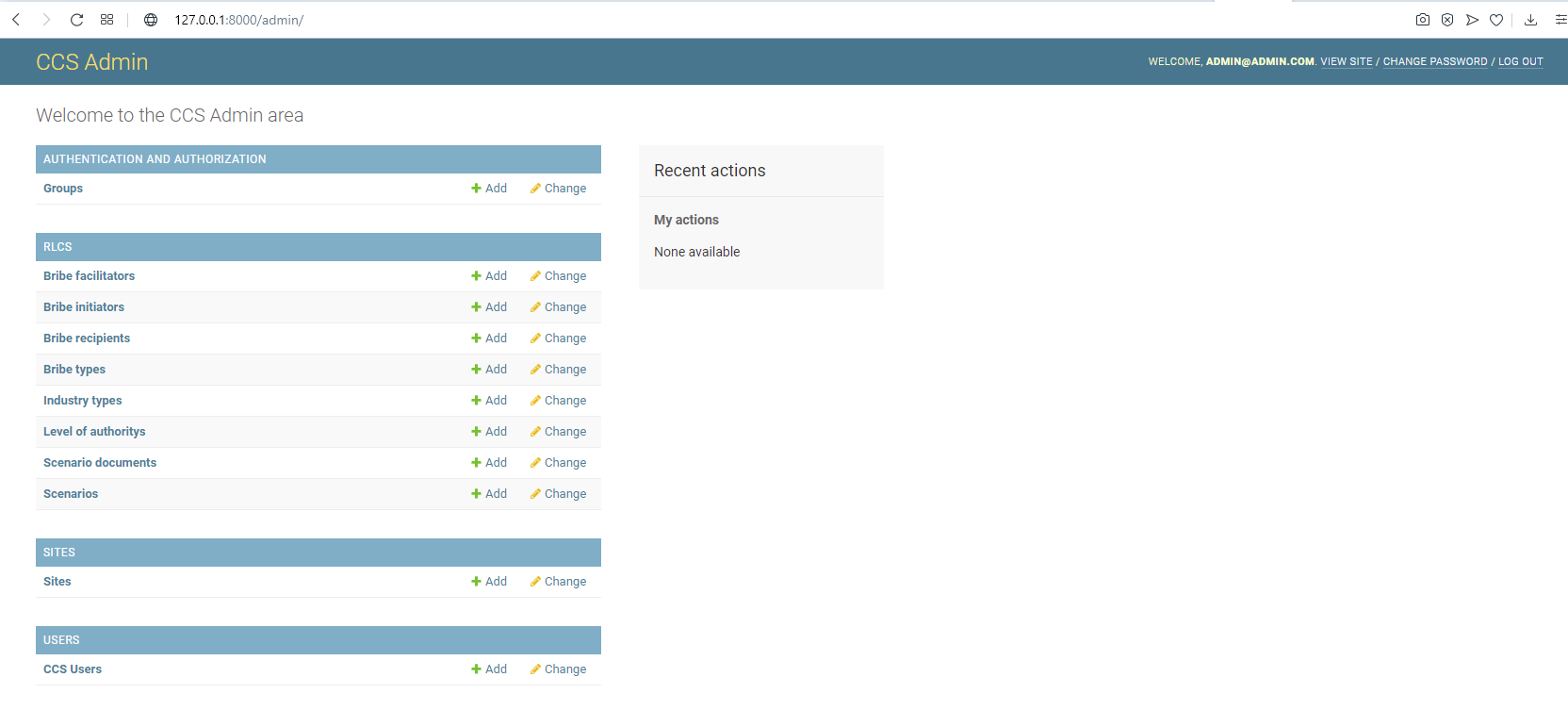
# CCS Admin

Access to the administration section of the application is through the Web-Admin interface which can be accessed via the following url: <http://127.0.0.1:8000/admin> or http://{domainname}:{port}/admin.

The following figure depicts the admin login screen.



Once the credentials have been verified, upon logging into the administration functions within the application, you will be presented with the following administration screen.



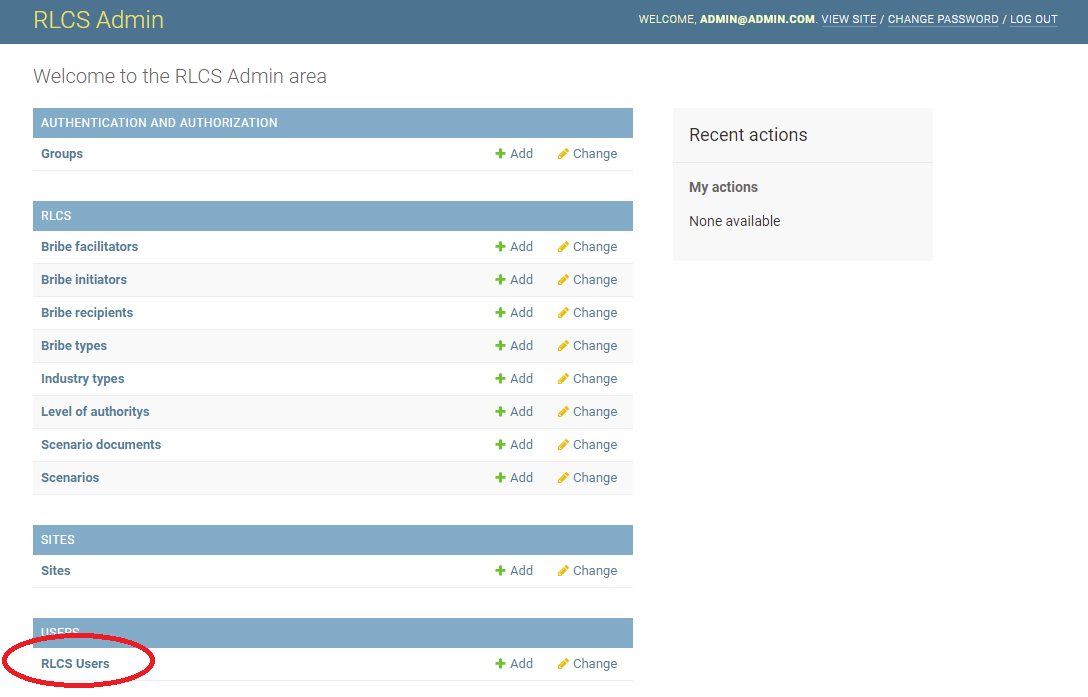
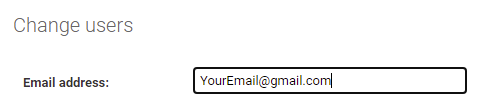
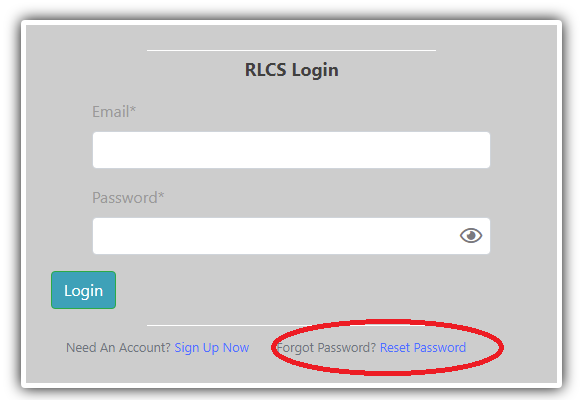
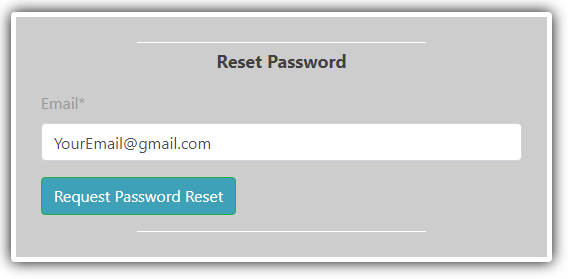
## Default Admin Account

By default, an admin account has been created to be able to login for the first time, access the admin console, and create new reviewers and administrators.

Default email: [admin@admin.com](mailto:admin@admin.com)  
Default password: 12345678

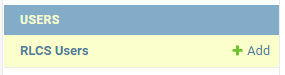
This is not secure and should be changed immediately upon running the application for the first time.

## Reset Default Admin Password

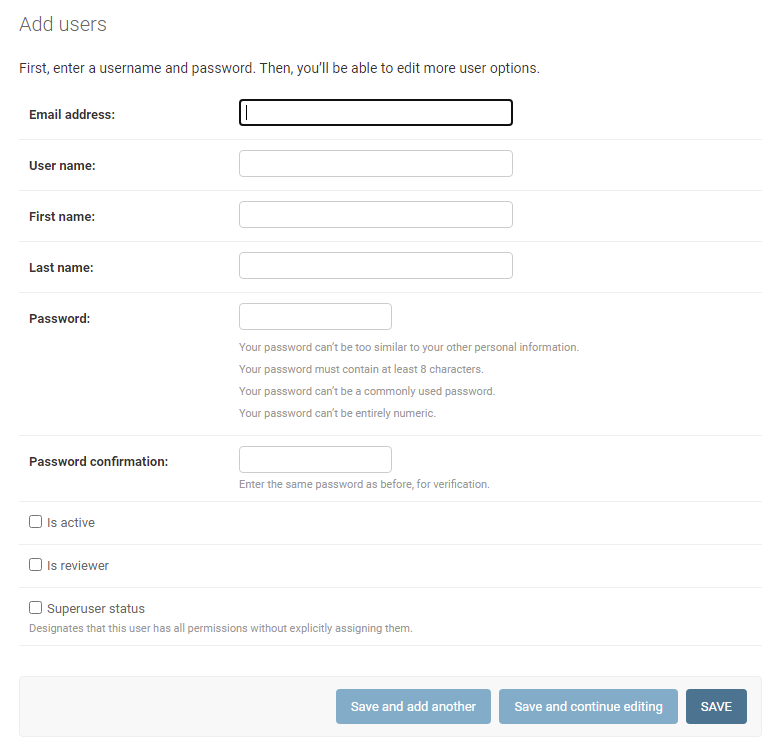
1. Login to RLCS with the default admin account from above
2. Click on the  icon to access the admin console.
   1. Note: Only those with superuser access can access the admin console.
3. On the **RLCS Admin page**, click on **RLCS Users**
4. Click on the [admin@admin.com](mailto:admin@admin.com) account
5. Change the email address to your own email address and then click  at the bottom of the page  
   
6. Click on **VIEW SITE** at the top of the page to return to the RLCS app.  
   
7. Click on  at the top right corner of the page
8. Click on  at the tope right corner of the page
9. Click on **Reset Password** at the bottom of the login form  
   
10. Enter your email address that you just changed in the Admin Console and click **Request Password Reset**  
    
11. Follow the instructions in the email to reset the password.

## Creating New Reviewers and Admins

To create a new Reviewer or Admin, you can either have someone register through the RLCS interface, or through the Admin console by clicking the  button in the **USERS** section.



Fill in the required fields and can assign the **Is reviewer** role or the **Superuser status** at the same time.



**IMPORTANT! Make sure you add the checkmark for Is active or else the user will not be able to login. In addition, if you want to create a new Superuser account, be sure to select Superuser status.**

**The Groups and drop-down lists for Bribe facilitators, Bribe initiators, Bribe recipients, Bribe type, Industry types and Level of authorities were loaded as part of the load data function within the entrypoint.sh script.**

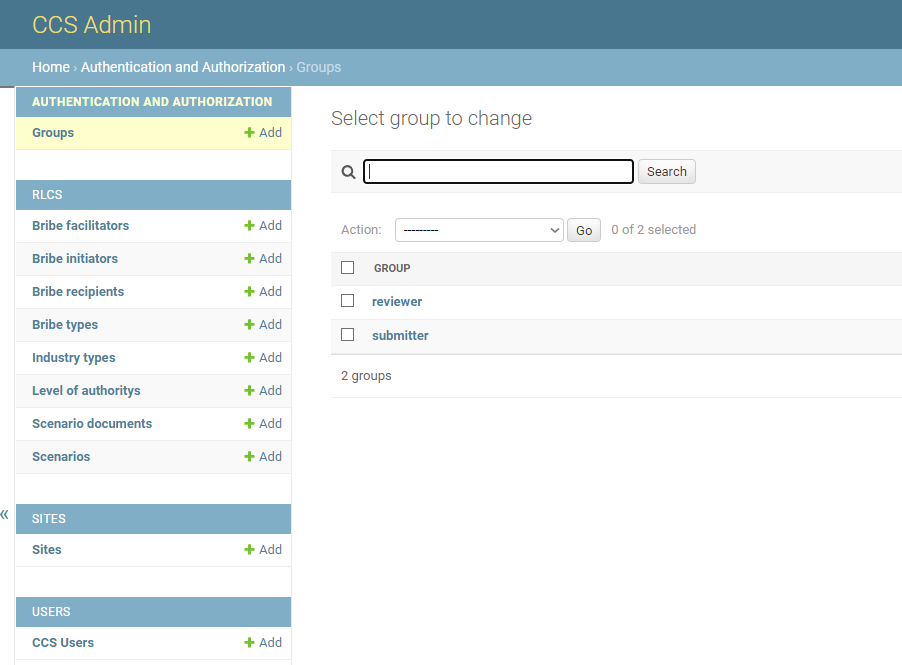
**Groups: Two groups are created and handled by the application.**

**Submitters: All newly registered users are submitters by default.**

**Reviewers: On those users checked in the user admin section are part of this group.**

**This is an important consideration since all users who are reviewers, will get notifications sent via email on submissions and reviewed/published cases. For each specific case, the email will contain a link to direct the user to the specific case submitted or reviewed. A login process is initiated prior to being directed to the specific case id for review.**

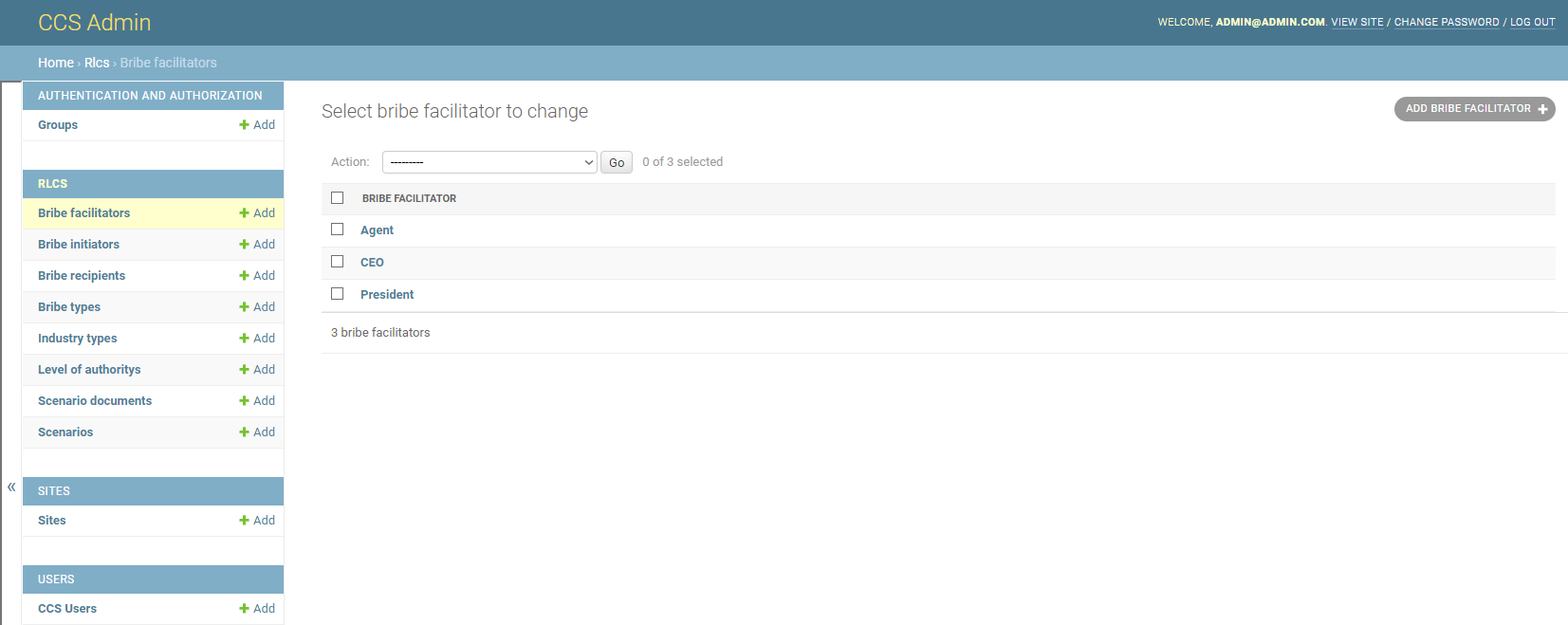
**The following figure shows the Groups authentication and authorization.**

****

**Bribe facilitators:**

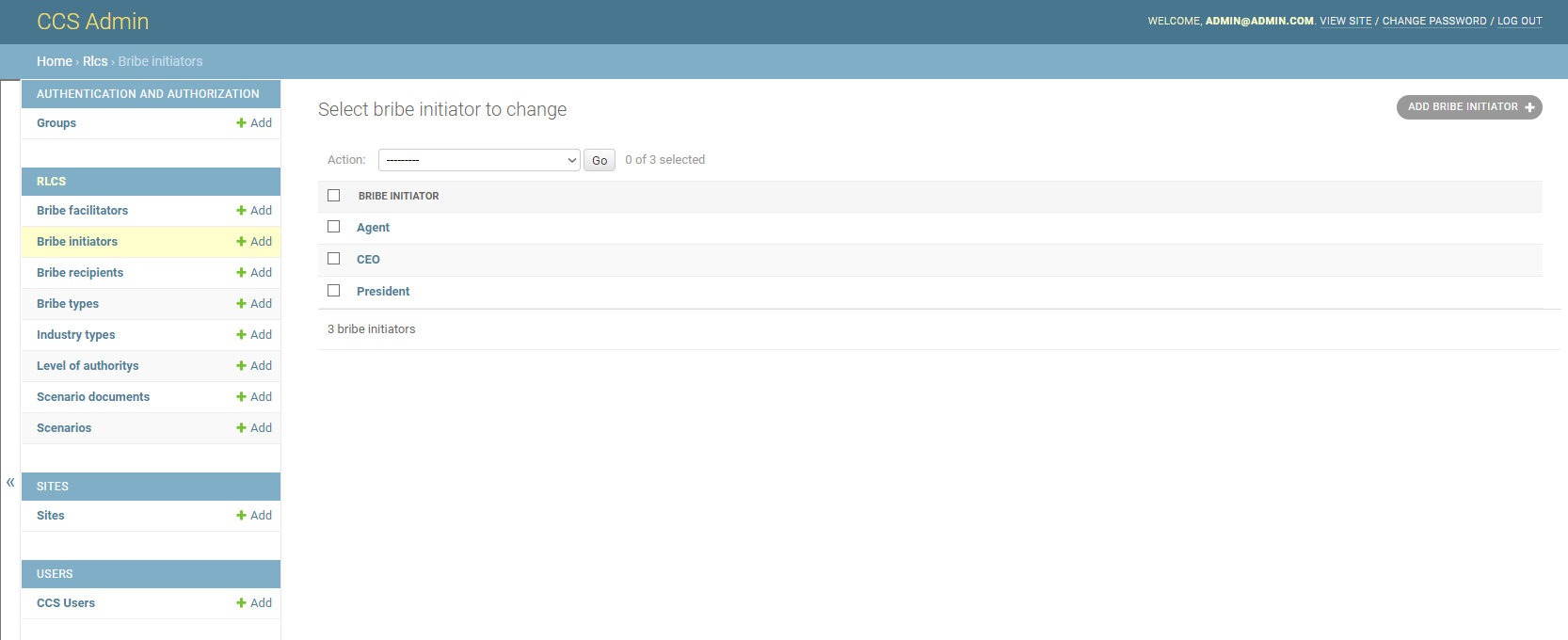
**This drop-down list contains the Bribe facilitators and new Bribe facilitators can be added to the list either to the right on the menu items on the left, or top right of the screen (ADD BRIBE FACILITATOR).**

**The subsequent figure shows this.**

****

**Bribe initiators:**

**The following figure shows the initiators. Much like the Bribe facilitators, this list can be added with new entries.**

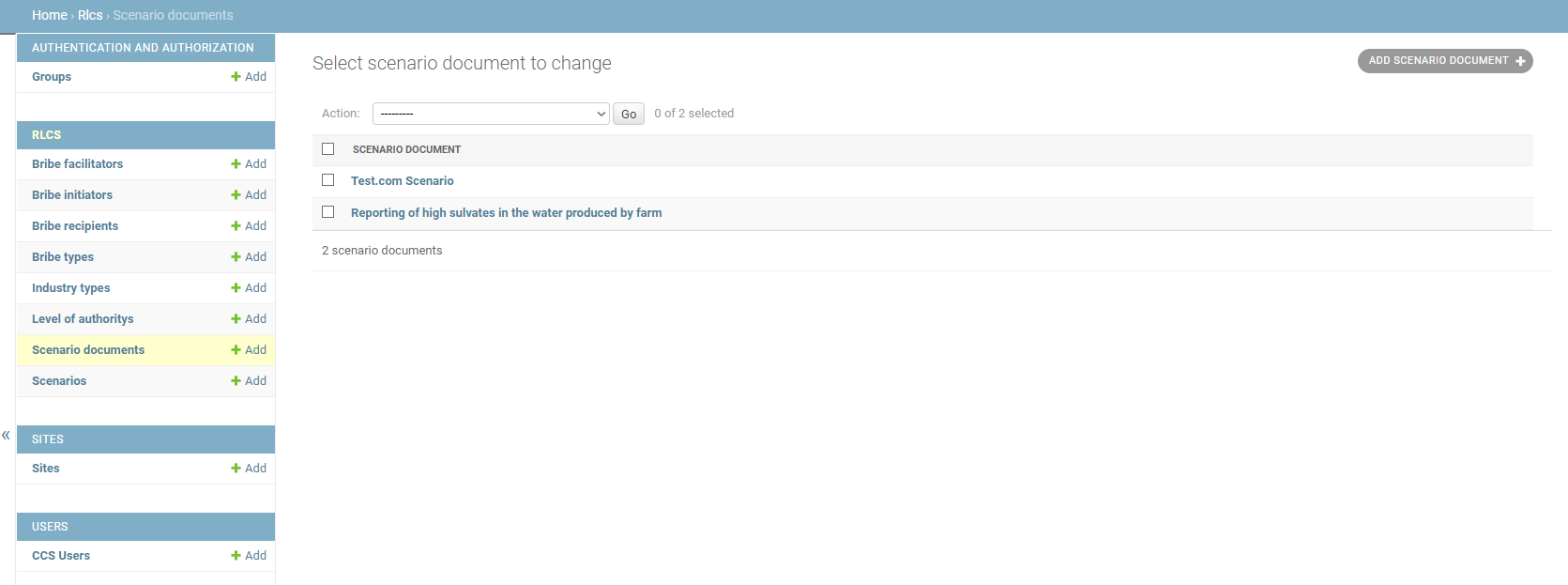
****

**The same process can be reviewed/amended for the remaining drop-down entries. Including, Bribe recipients, Bribe types, Industry types and level of authorities.**

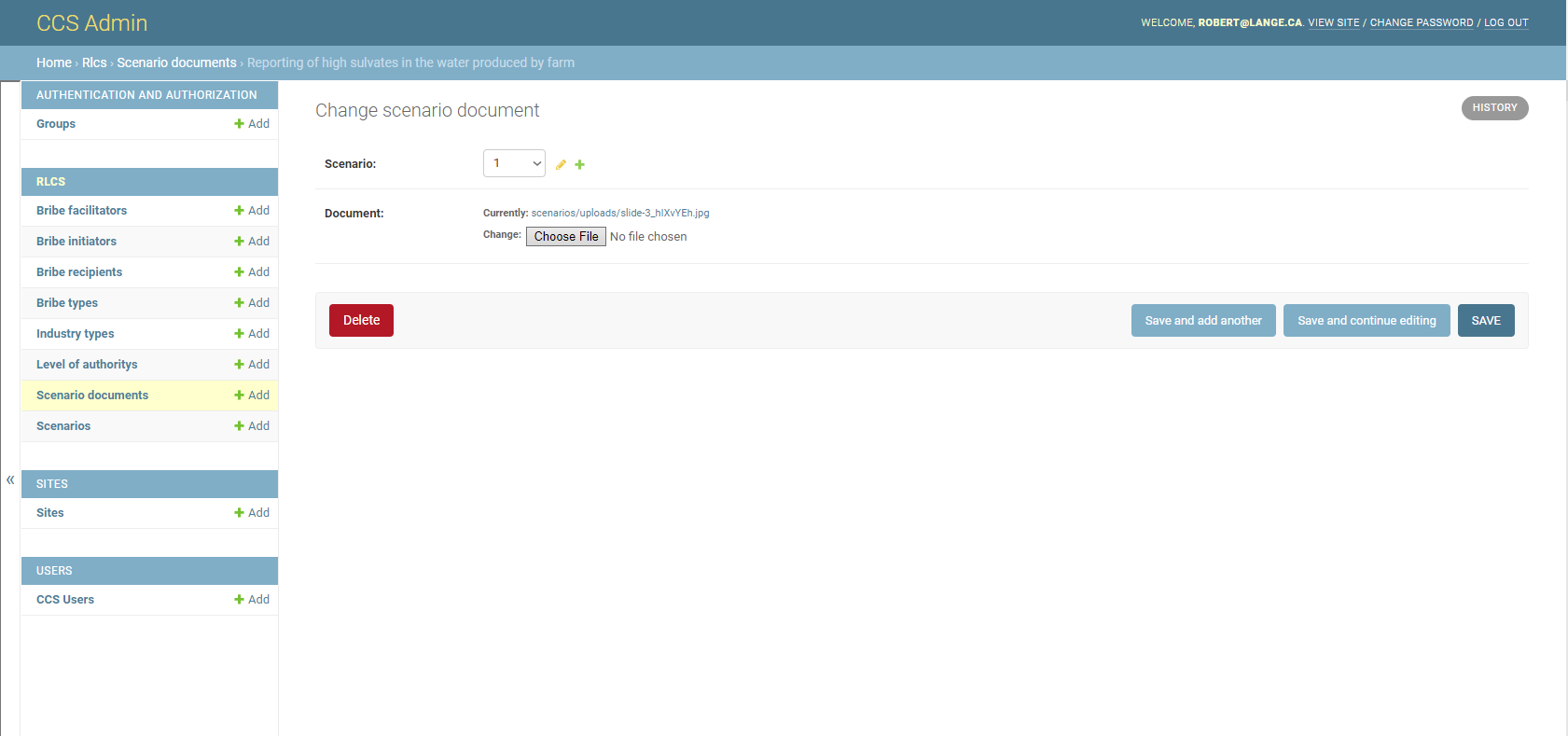
**Scenario documents:**

**All documents relating to cases are indexed within the database but are kept on the file system within the media/scenario/uploads folder.**

**The following figures show media related screens.**

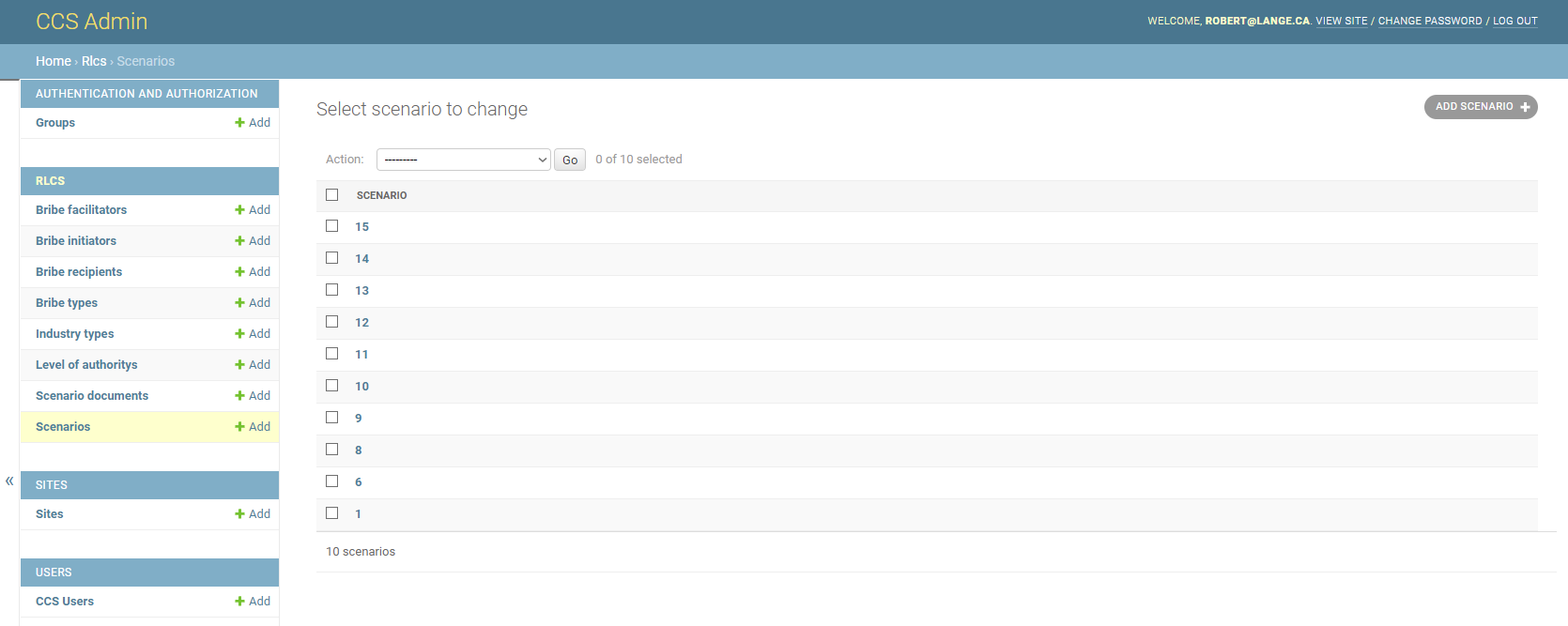
****

**Amendments to the document related to a specific case can be made as shown in the follow figure.**

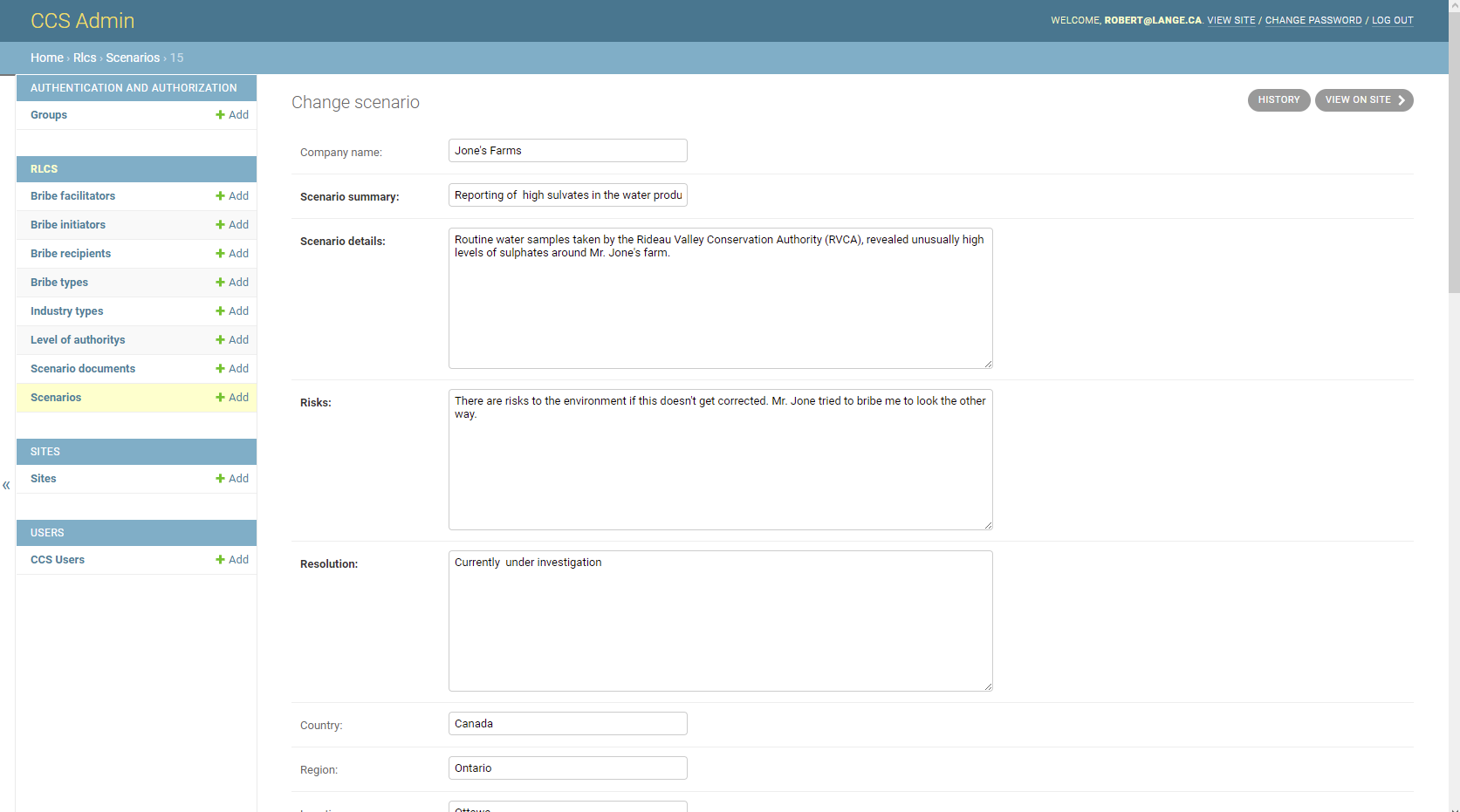
****

**The scenarios list, lists the scenario by way of index Id. If you know the index (by way of the user’s view cases menu), you can amend the case scenario. While this function is available, it is not the desired process for managing cases.**

**Like the other items notes, you can add/change a scenario case via this screen.**

****

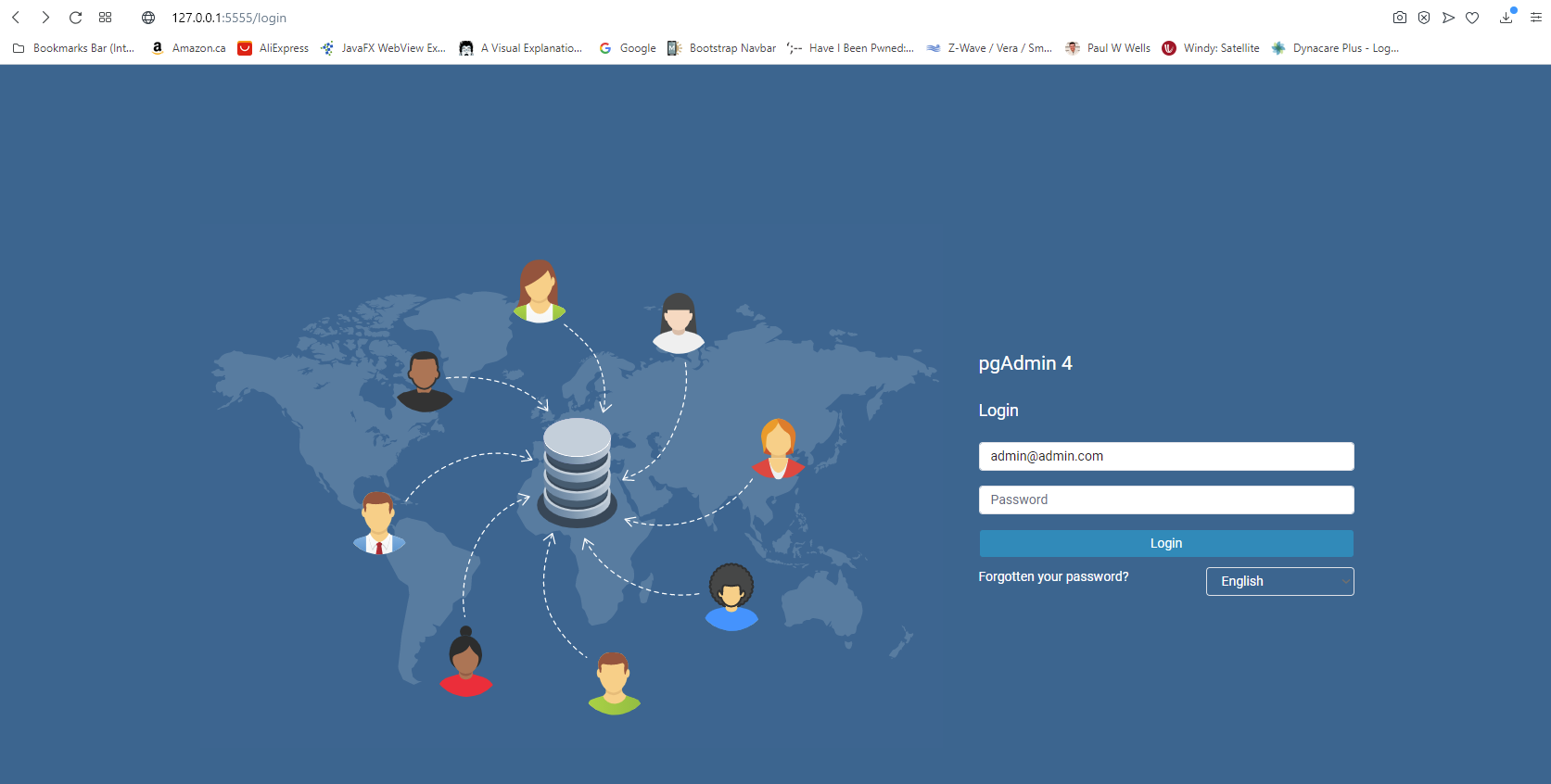
**Selecting any of the case Ids will bring you to the case. You can also view the case in the application view by selecting VIEW ON SITE – top right screen.**

****

# **Database Administration**

## Postgres pgAdmin4:

Management of postgresql database can be easily performed through the pgAdmin4 user interface which runs within the container. Access can be via the web at <http://domain:5555> or as a standalone development url <http://127.0.0.1:5555>.

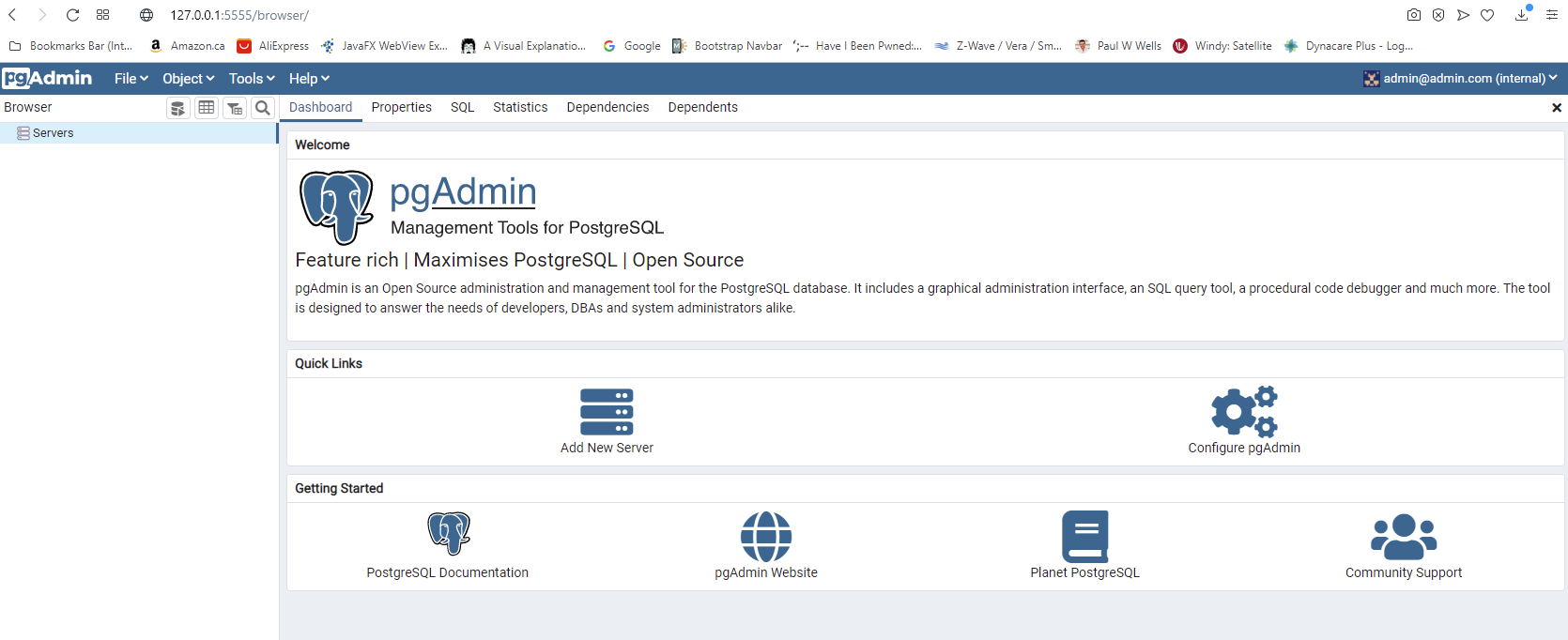


The credentials are defined in the .env.dev file and listed as:

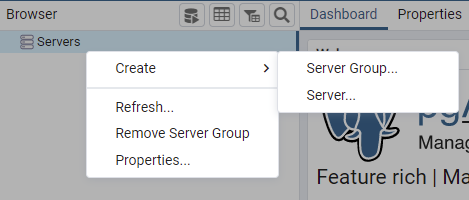
PGADMIN\_DEFAULT\_EMAIL=admin@admin.com

PGADMIN\_DEFAULT\_PASSWORD=admin

Once logged in the server needs to be added in order to manage the database.



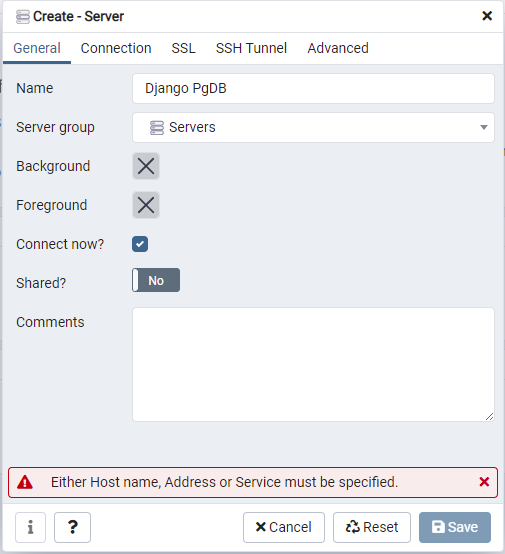
Right click on the Server icon to configure the server.

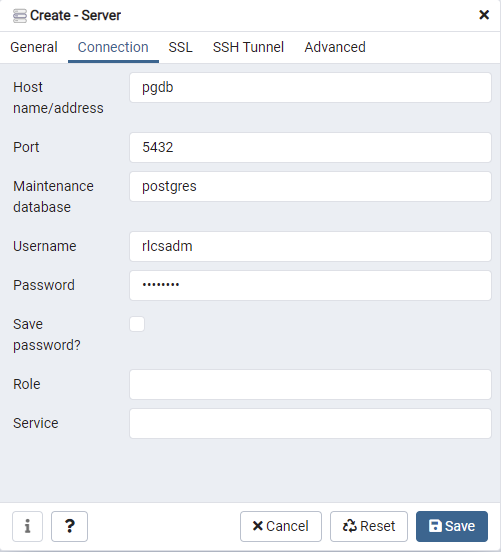


Give a name to the server database name and add the credentials to access it. These are specified in the .env.dev file as well.

POSTGRES\_USER=rlcsadm

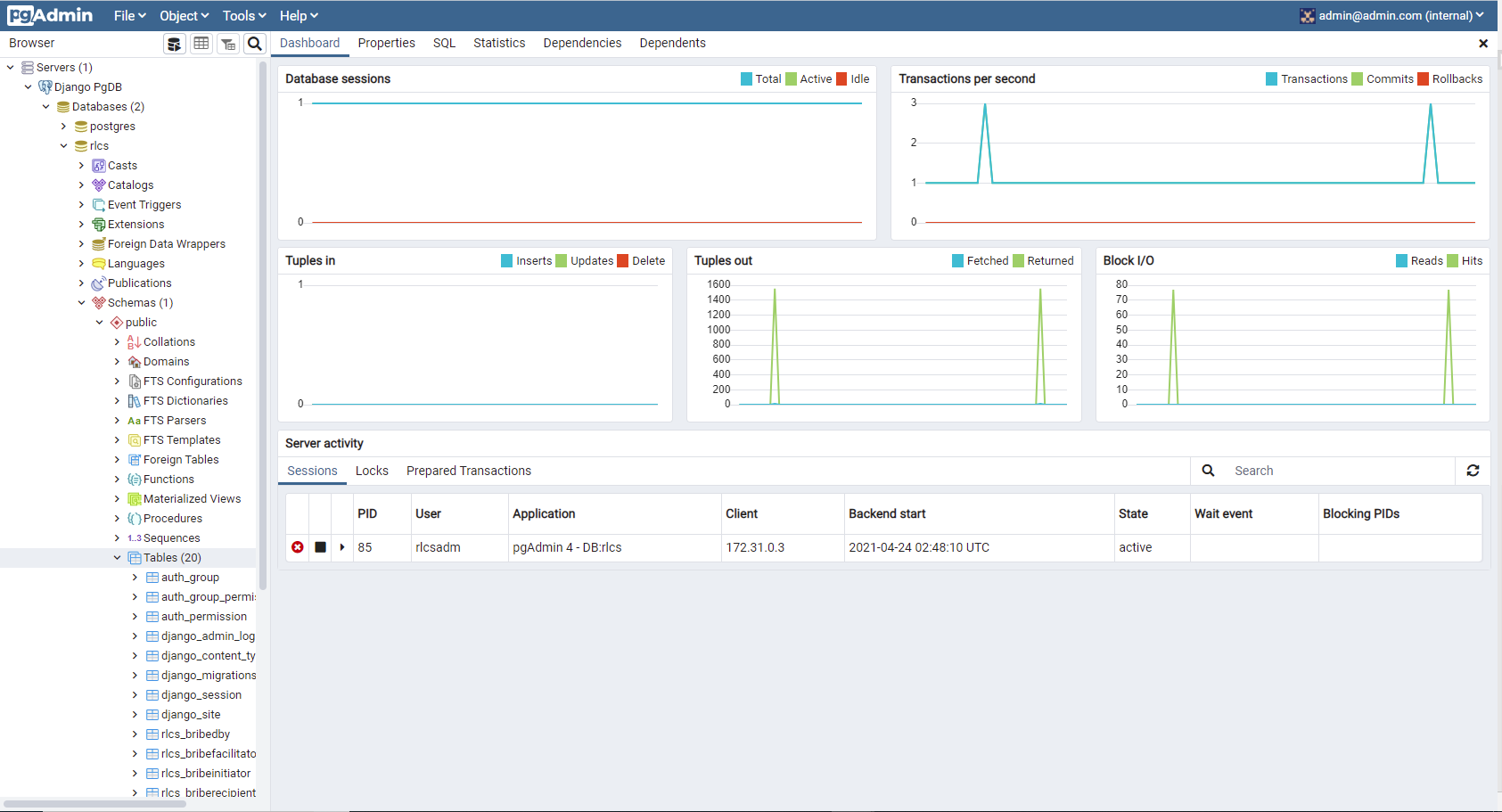
POSTGRES\_PASSWORD=12345678





Upon saving the configuration, you will have access to the database and tables, users etc. This information is persisted in the container and is only required once unless you rebuild the docker image.

The figure below shows the configured pgAdmin4 application screen.



# **Scenario Workflow**

## Submitting a Scenario

Once you’ve logged in with a **Submitter** account, you can go to the **View Scenarios** section and click on 

Fill in as much details on the form that you can provide as this will help others searching through the scenarios.

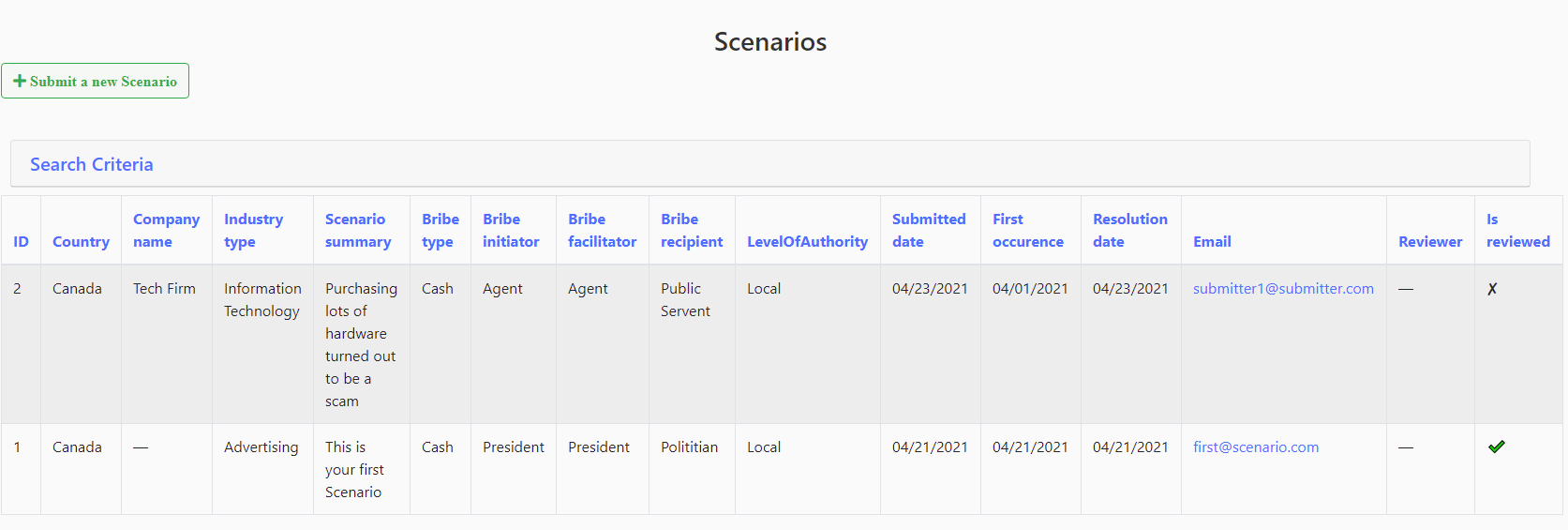
**Note:** You have the option to hide your Company name, Region, Location and Public Email from the public but it will be visible to the submitter who submitted the scenario, and all Reviewers and Admins of the RLCS system, by checking the following box:

****

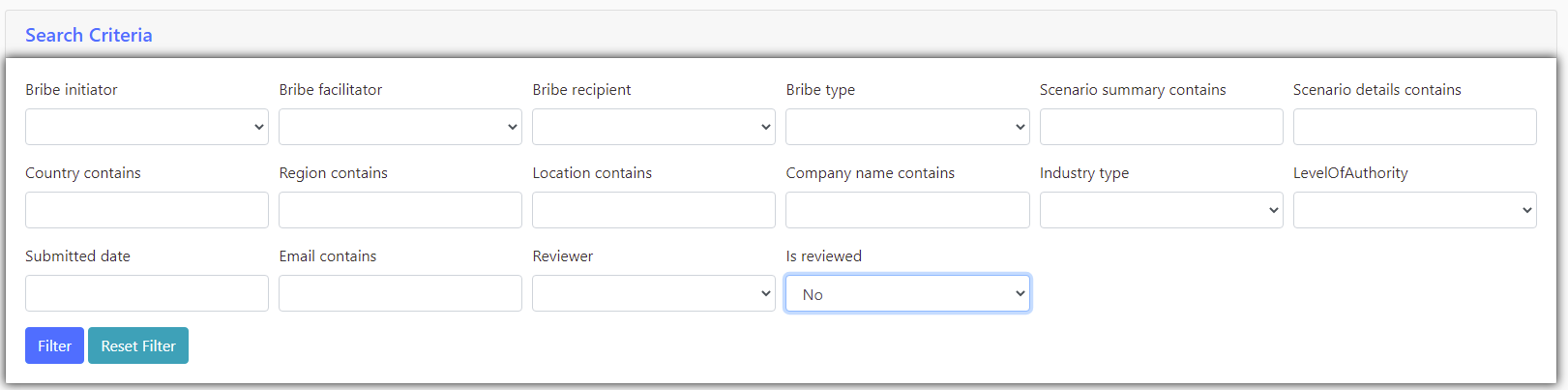
Once you are done, use the  button to send your scenario for Review.

## Reviewing a Scenario

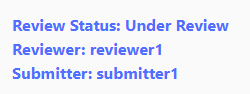
1. Login to the RLCS application with a user that has the **Reviewer** role and click on **View Scenarios**



1. You can see in the **Is Reviewed** column either an or . Look for scenarios with the  indication to start reviewing it.  
     
   You can also use the **Search Criteria** section to filter through the scenarios for those that are not reviewed yet.



You can see who is currently reviewing the scenario, who submitted it, and the current status of the scenario at the top right portion of the scenario page.



1. Click on the button to assign the scenario to yourself as well as make changes to the scenario.
2. Once the Reviewer is satisfied with the scenario, they can publish it publicly by clicking on the  button.

## Editing a scenario

Only the original submitter, Reviewers and Admins can modify a scenario. Once it has been modified, it will lose it’s Reviewed status and need to be reviewed again before it is displayed publicly.

# Issues, Considerations and futures

While every effort was given within the timeframe to create a refined application, there are a number of development issues which fell short of refinement or were not resolved in the final release. This section details them.

**Bootstrap Responsive actions for all screens:** Both Bootstrap4 and Bootstrap5 were used throughout the application which provides responsive screens in almost all cases. Responsive screens correctly render all content despite the size of the available window space. One item to note which had not been resolved was the password ‘eye’ icon used to show/hide the password field. This is implemented in both the login screen and reset password screen. Should a change in screen size or additional content get added to the Login and Password reset screen, the show/hide icon will fail to be located in the right place as it is statically placed. There is no change in functionality, but rather just presentation.

**SSL Secured Web access (Future):** This feature was discussed within the project, however, not implemented since in order to register a certificate for Secure Socket Layer, a domain name is required. A CSR Certificate Signing Request cannot be generated without this information.

**Amazon Web Services (Future):** An attempt was made to host the application within AWS, however a lack of time played heavily in this lack of this implementation. As the application has been containerized within Docker, the implementation should be easier to host.

**Running in production with NGINX Proxy (Future):** It is recommended that in the current form, the application not execute for production purpose as configured. For a production ready hosted application, the application should be utilized through a NGINX proxy which would then host static and image files. All requests would in fact be executed by the NGINX proxy which is designed for production. This is a key consideration as case documents may be added and reviewed for each submission. Time was a major constraint with this implementation.

**Clearing Filters in the View Cases:**

The Reset Filter button function does not clear all the filters. Clearing them manually resets however. That is by deleting the form fields, resets the view.

**Renaming the code base from Scenario to Cases:**

At the last minute and during final presentation, it was requested by CCEAC that we change the terminology from Scenarios as previously requests to Cases. While labels were changed to accommodate this request, no attempt was made to change any backed code or models – just too last minute.

**Rating system not implemented (Future):**

Some discussion occurred with respect to having a rating system per case much like that of Amazon – though the criteria was not clearly established nor how to display such a rating. Once more clearly defined, this would be a good ‘future’ feature to implement.

**Sub Id added to the Case when viewed within the View Case entry (Future)**

Viewing the submission Id while reviewing the case may provide the submitter or reviewer added information and reference. This was not considered during the development but may have added merit for the future.

**Submitter getting notification of submission via email: (Future)**

This was not considered earlier. However, this would be easy to implement within the signal.py of the rlcs application within the project. Currently, only users who are fall under the umbrella as reviewer gets the email notification. The email notification includes a link back to the submission Id and should the reviewer activate the link, they will get directed to the submission for review. A login redirection will first take place, but once completed, they will be directed within the view submission mechanism.