THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

DECO3801: Design Computing Studio 3 – Final Report)

Website (HTTP): http://deco3801djangoweb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com

Website (HTTPS): https://deco3801djangoweb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com

Title: Chemicals of Emerging Concern Early Warning Social Network

Project: Web Interface for High Resolution Mass Spectrometry Data Analysis

Team: Team Innovation Hackers

Team Members: Annie Huang (43257490), Chun-Yin Wang (43578300), Minh Nguyen (43552065), Justin JiHoon Oh (43789991), Andre Wong (43584105), Fong Yng Dong (45196520), Jodi Hong Han Toh (4545921), Albert Thompson (43202445)

Contents

1.0 Summary

1.1 Product

Throughout the 2nd semester of 2018, Team Innovation Hackers has been working on developing a social network called Chemicals of Emerging Concern Environmental Warning Social Network (CECEWSN). The client Queensland Alliance for Environmental Health Services (QAEHS) had approached the UQ seeking a solution that would help advance the scientific community and assist environmental health chemists with progress of discovery through the construction of CECEWSN. As QAEHS has developed a machine learning and statistical learning algorithms that are used to process large sets of data generated from a High Resolution Spectrometer (HRMS). Through these algorithms one is able to identify known compounds and chemicals and discover new ones found in samples from an environment. However, the efficiency and reliability of these algorithms are dependent on the data that it uses to learn. However, the existing database that stores and maintains the current data is difficult to access, which has causing a detrimental effect of slowing the progress of discovery in the field of environmental science.

Innovation Hackers has proposed the solution of a website that offers its users the tools and assistance needed to increase the efficiency in the research, process, and analysis of the HRMS data. This website would aim to bridge the disparity in the way environmental scientists' process and store their data, and to also construct a social network to help increase the interaction and collaboration of environmental health chemists on a global scale. By offering a centralized approach to data process and the collaboration of data the site will be able to cut down the time spent on paperwork and research needed for its users.

1.2 Context of Use

As mentioned CECEWSN's website is focused on enabling environmental health chemists to analyze the chemicals and compounds contained inside their water samples in a user friendly and convenient format. With a user uploading data samples from their High Resolution Spectrometers either as an excel file or as a text file it is common for these files to be a few gigabytes in size. With the scale of these files the processing time to be completed without a way to store the acquisition methods or hardware utilized is very time consuming. With this website a user will be able to maximize their efficiency in the lab by reducing the input time need. This is achieved through the features and functionalities offered by the website which allows the users to reuse acquisition methods and data sample types.

Once a sample has been uploaded, CECESWN's integrated database stores a copy of the raw data for the verified user to access in the future. Additionally, the data will also be processed using QAEHS's machine and statistical learning algorithms. This processed data will be stored on the database where the user can choose to share or download the data for further personal use.

The website built has also focused on increasing interaction and collaboration of user's predominately through the implementation of a forum and through the combination of CECESWN database and the global search feature. Through CECEWSN's forum a user will be able to interact with other users, engaging in debates based upon this field of environmental study. A user can start, comment, reply, and bookmark posts they're interested in. From these bookmarks a user is able to be notified on any developments that have occurred. Whilst this is the main focus to increase collaboration of users, through the use of CECESWN's database and global search a user will able to search up valuable information about chemicals and compounds that may interest the user. Included in the returned sample will also be details about the user that uploaded that sample. Through the uploaders' details a user will be able to directly contact them and engage in conversation if they would like. Through these two modes of interaction the environmental health chemist society using CECEWSN will be able to increase the collaboration effectively.

In the final product, users will also be able to perform a global search for any compound and its relevant reports, view past data sets and custom visualizations of their uploaded data.

1.3 Client

Queensland Alliance for Environmental Health Sciences (QAEHS) is a research center that was jointly founded by Queensland Health and the University of Queensland. QAEHS has developed a machine learning and statistical learning algorithms that are used to learn models from large data sets that are generated from High Resolution Spectrometer (HRMS) data. These models are used to classify if unknown or known compounds and chemicals are observed in new and unseen samples from the environment. However, the efficiency and reliability of these models are dependent on the data that is used to learn the models. Due to this, QAEHS has sent a representive, Samual MacDonald to approach the University of Queensland seeking a solution that may offer its user the tools and assistance to efficiently research, process, and analyses HRMS data to maintain efficiency and reliability of the models.

1.4 Overview of Documentation

This document outlines all specifications and details required to install and operate the final product from both the system administrator and end-user perspective. Additionally, the document details the full project creation process, from ideation, development, testing, to deployment with a comprehensive code guide and database structure for future developer reference. This document will also extrapolate the weaknesses and strengths derived from user experience testing and functional testing to recommend avenues of future development of the final product.

2.0 FAQ

This section outlines questions relating to the setup and installation from an Administrators perspective as well as questions frequently asked by users. It also outlines common errors and their solutions, as well as other issues that were faced during the development of the system.

2.1 Administrator Questions

- **What programming language and frameworks are being used?**
    - Django, a Python framework, is being used for the backend development of the web application.
    - ReactJS is being used for the frontend development.

- **Any important software tools or libraries that are worth taking note of?**
    - JavaScript Web Token (JWT) is being used to implement token-based authentication on the backend API.
    - Boto3 is used to handle all communication between Django backend and Amazon Web Services.
    - Pipenv is the tool to manage the virtual environment for all the libraries of Python pip.

- **Which database is being used?**
    - There are two databases setup in this project for development and production separately. The databases are SQLite and Amazon RDS used during development and production respectively. Both are using SQL.

- **What software are needed for this project?**
    - Developers will require NPM to work with the React project
    - Python 3.6 with Pip to install Django framework
    - Pipenv to manage the Django packages.

- **What AWS services are being used?**

  - Amazon Elastic Beanstalk is used to control the Amazon EC2 and Amazon S3 environment. Amazon RDS has been set up separately, but has the option to also be controlled by Amazon Elastic Beanstalk.

- **What is the package manager for Amazon Linux?**
  - Amazon Linux uses the yum package manager commonly found in CentOS.

- **Questions regarding the installation of software and their dependencies**
  - Please refer to Section 3.1 Installing Dependencies for a thorough guide on installing the required software and their dependencies for this system

- **Questions regarding the deployment of the website to AWS Elastic Beanstalk**
  - Please refer to Section 3.3 Deploying to AWS Beanstalk (EBS) for a thorough guide on deploying the website to AWS Elastic Beanstalk

- **How is the forum setup?**
  - PhpBB provides a free, open-source forum template with advance forum features. Installation of PhpBB can be done via its installation web interface. Further details can be found on their official website (https://www.phpbb.com/support/docs/en/3.0/ug/quickstart/installation/)

- **If there are two database, does it mean the schema has to be managed separately?**
  - No, the Django framework uses object relational mapping (ORM) to modify and communicate with the database. It is designed to write once and use anywhere. Django has a great tutorial for this on their documentation (https://docs.djangoproject.com/en/2.1/intro/)

- **How do I SSH into AWS Elastic Beanstalk Environment?**
  - AWS Elastic Beanstalk is designed to eliminate the need to SSH into the instance. If you found yourself needing to do it, please reconsider your implementation and look at .ebextensions instead

(https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/ebextensions.html). However, if you really need to, you can SSH using EB CLI in the command line interface (https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb3-ssh.html).

2.2 Errors

- **When I run python manage.py runserver I am getting the error: “ImportError: Couldn't import Django. Are you sure it's installed and available on your PYTHONPATH environment variable? Did you forget to activate a virtual environment?**
    - Make sure you are running the command in the Pipenv shell, or have Pipenv run preceding the command.

- **I am getting the error: Error: Cannot find module <package_name> and the stack trace shows it occurred on node script/start.js**
    - Make sure you have run npm install, if you have already, ensure the package is defined in package.json and is present in the node_module/ directory.

- **My .bat file closes immediately and the development server failed to start.**
    - Make sure you are not running command prompt in administrator mode, and make sure you have done Pipenv install.

- **Error with the upload**
    - Check the IAM settings for the AWS commands in the EC2 SSH shell. To reconfigure your IAM, run - AWS configure.

- **After deploying to AWS Elastic Beanstalk, the website URL opened successfully but the page is blank.**
    - The Django server does not retrieve files from the EBS on the EC2 instance. All files and bundles are retrieved from S3, ensure that the command python manage.py collectstatic has been properly executed as defined in .ebextensions/ configuration directory.
- **After deploying to AWS Elastic Beanstalk, the website URL did not open successfully.**

    - This is most likely due to a configuration error either on Elastic Beanstalk or Django. The full error logs can be found in AWS Elastic Beanstalk console.

2.3 Other Issues

- **What are the costs associated with AWS?**
    - Final cost will be heavily dependent on the load and usage, and will vary depending on the tier of AWS services. Table 1 shows a rough estimation of typical operating cost, in USD, of AWS services.

AWS Services	Rates	Additional Costs	Free-Tier Duration/Restrictions
Amazon EC2 (t3.micro)	$0.0104/hour	-	1 year
Amazon EC2 (r4.xlarge)	$0.266 per Hour	-	-
Amazon S3	First 50TB/month: $0.023 per GB Next 450TB/month: $0.022 per GB Over 500TB/month: $0.021 per GB	GET, SELECT and all other Requests: $0.0004 per 1,000 requests PUT, COPY, POST or LIST Requests: $0.0005 per 1,000 Requests	For 1 year: 5GB storage 20,000 GET Requests 2,000 PUT Requests
Amazon Elastic Beanstalk (Free)	-	-	-
Amazon RDS (db.t2.micro)	$0.017/hour for 20gb	-	1 year

Table 1: Costs of AWS Service

- **How do I suspend the instance of AWS Elastic Beanstalk temporarily, I've tried stopping EC2, but AWS constantly creates new instances?**
    - Unfortunately, there's no easy way to temporarily suspend the instance. It is a feature of Elastic Beanstalk to ensure that there is always at least one running server. However, the same effect can be achieved by going on AWS Elastic Beanstalk console, in configuration, click modify capacity, changing the environment type to 'Load Balanced', then set the minimum and maximum instances to 0.

2.4 User Questions

- **What is CECEWSN?**
    - CECESWN stands for *Chemicals of Emerging Concern Early Warning Social* Network. CECEWSN is a web application that offers to help to bridge the disparity in the way environmental scientist process and store their data between institutions. It is designed to be user friendly and convenient approach for scientists to analyses and share the chemicals and compounds found in the sample that they have collected with other scientists, allowing for collaboration across all institutions.

- **Who is QAEHS?**
    - QAEHS stands for *Queensland Alliance for Environmental Health Sciences.* QAEHS is a research center that was jointly founded by Queensland Health and the University of Queensland

- **Where is the database hosted?**
    - The database is currently being hosted on Amazon RDS and Amazon S3.

- **What is a HRMS?**
    - HRMS stands for High Resolution Spectrometer.

- **How does a user sign up?**
    - A User can sign up by filling out the form on the Sign Up page. The Sign Up page can be accessed from the Navigation Bar.

- **What is the difference between a guest user and a registered user?**
  - The difference between the two types of users is the access to the features that they have. Guest users only have access to features that don't permit them to edit data in the database and make use of the algorithms.

	Guest	Registered Accounts
Global Search for Compound	Y	Y
Access to Help / FAQs	Y	Y
View the profile of other users	X	Y
Upload or Process Data	X	Y
View or Download Reports	X	Y
Add or Remove Acquisition Methods	X	Y
Access to Documentation	X	Y
Access to Forum	Y	Y

Table 2: User Access Permissions

- **How do I upgrade a user's access permission?**
  - Only an administrator of the website can change a user's access permissions.

- **How is a user's password stored?**
  - Passwords are stored on the server's database and hashed to prevent the user's password from being stolen or made known.

- **How do I access the database?**
  - Users are not able to access the database, they will only be able to interact with the database on the web application.

- **What file types will be accepted in the uploading and processing feature?**
    - The accepted file types are text files (.txt) and Excel files (.xlmx)

- **How long does it take to upload files?**
    - The duration to upload a single file depends on various factors such as your upload speed and the size of the file. On average, an upload can take a few hours to process.
- **Will a user be able to set the privacy of the data they upload?**
    - Users are able to modify their privacy setting from the profile page once they are logged in.

- **Can a user save specific acquisition method settings whilst uploading?**
    - A user can save their commonly used acquisition method settings via the Acquisition tab in the Navigation Bar.

- **Where is a user's data uploaded to?**
    - The user's data is uploaded to Amazon S3 where it will be processed by QAEHS algorithm.

- **What information will a user need to include whilst uploading their data?**
    - A user will need to include the samples information, the acquisition hardware information and the acquisition method.

- **What is the web address for CECESWN forum?**
    - Ec2-18-224-228-152.us-east-2.compute.amazonaws.com

- **What data is accessible from CECESWN and who can access them?**
    - Registered users can access processed data analysis reports. Guests and registered users can only access the global search function to view a heatmap of where a specific compound or chemical was found.

3.0 Installation and Setup Guide

The development environment is managed through development tools, which are used through the command line interface. Throughout the setup and installation guide, terminal, will be used as the alias for both command prompt (Windows) and terminal (Linux/MacOS). The following sections outlines the instructions to perform a number of tasks which are required to set up and install the software.

3.1 Installing Dependencies

3.1.1 Installing package management systems

Table 3 outlines the software and package managers that are used for installing dependencies for the front end and back end systems during the initial installation process.

Package	Purpose	Source
NPM	Managing dependencies for JavaScript	https://www.npmjs.com/get-npm
Python 3.6 and Pipenv	Managing dependencies for python	https://www.python.org/downloads/

Table 3: Package Management System for managing system dependencies

3.1.2 Install Django and Pipenv packages:

1. Open up your terminal and navigate to the root directory of the project.
    a. *(Windows)* Press the start menu, and type 'cmd' or 'PowerShell' in the search bar
    b. *(Linux)* Open up the terminal, method depends on your Linux distribution
    c. *(MacOS)* Press ALT+ENTER to bring up search, and type in 'terminal'
2. Install the Django Framework by running:
    a. pip install Django
3. Install Pipenv by running:
    a. pip install pipenv OR brew install pipenv

3.1.3 Install project dependencies (Method 1)

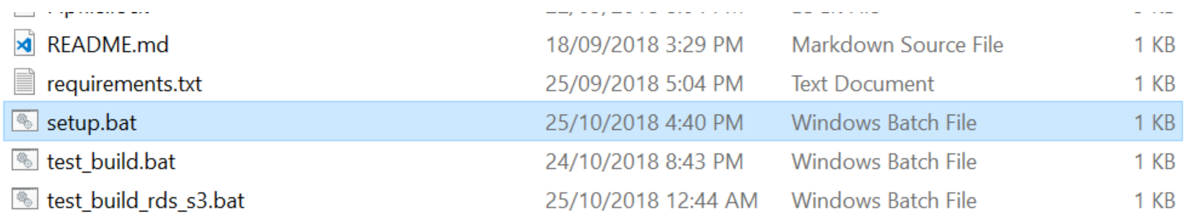
1. Open up your terminal and navigate to the root directory of the project.
2. Install all the Django dependencies by running:
    a. pipenv install

3. Install all the React dependencies by running:
    a. npm install

3.1.4 Install project dependencies (Method 2)

This method employs custom bat and scripts to automate the process of installing both Django and React dependencies.

- (Windows): Double click `setup.bat` in the `django-react-web` directory.



README.md	18/09/2018 3:29 PM	Markdown Source File	1 KB
requirements.txt	25/09/2018 5:04 PM	Text Document	1 KB
setup.bat	25/10/2018 4:40 PM	Windows Batch File	1 KB
test_build.bat	24/10/2018 8:43 PM	Windows Batch File	1 KB
test_build_rds_s3.bat	25/10/2018 12:44 AM	Windows Batch File	1 KB

4. (Linux/MacOS)
    a. Open up your terminal and run the following commands
        i. chmod +x setup.sh
        ii. ./setup.py
5. Please follow *Section 7.0 Source Code Structure* for more details on the source code

3.2. Running development servers

Django and React are on different development servers, in order to receive live feedback for changes on the frontend on port 8000, it is necessary to set DJANGO_DEV environment variable. More details can be found in *Appendix A: Environment Variables in DJANGO*. Otherwise it is necessary to build a production bundle every time to see new changes to the frontend.

The following sections outline how to run the Django and React servers in isolation and in parallel. Before following and of the instructions below, e*nsure you* have installed all the react dependencies and Django dependencies (refer to step 1.2).

3.2.1 Running the Django development server

Note: Refer to *Appendix B: Virtual* Environments for the motivation behind virtual environments.

1. Navigate to the `django-react-web` directory
2. Open up a virtual shell of your python with all the Django dependencies by running in your root directory (django-react-web/):
    a. pipenv shell

3. Set the environment variables by typing the following in your terminal:
    a. *(Windows (CMD))* set DJANGO_DEV=true
    b. *(Windows (Powershell))* $env:DJANGO_DEV='true'
    c. *(Linux/MacOS)* export DJANGO_DEV=true

***Note:** Refer to Appendix A: Environment Variables in DJANGO for more details on environment variables.*

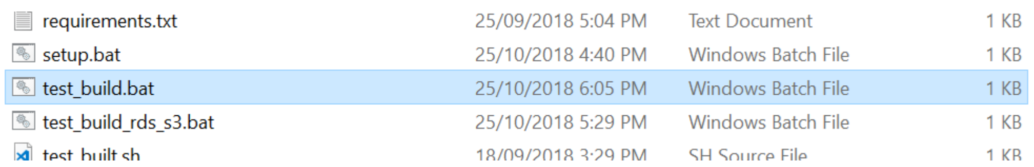
4. Run the Django development server on port 8000 by running:
    a. python manage.py runserver
5. View the website by typing the following in the browser:
    a. localhost:8000
6. *(Optionally)* If you want to work outside of the virtual shell, you can run:
    a. pipenv run python manage.py runserver

3.2.2 Running React development server

1. To launch the React server on port 3000, type in the terminal:
    a. npm start
2. To view the frontend of the website individually, type the url in the browser:
    a. localhost:3000

3.2.3 Running React and Django development server with custom scripts

1. Running the servers
    a. *Navigate to `django-react-web`*
    a. *(Windows):*
        i. *Navigate to `django-react-web`*
        ii. *Double click `test_build.bat` to launch the servers through custom .bat scrips*



requirements.txt	25/09/2018 5:04 PM	Text Document	1 KB
setup.bat	25/10/2018 4:40 PM	Windows Batch File	1 KB
test_build.bat	25/10/2018 6:05 PM	Windows Batch File	1 KB
test_build_rds_s3.bat	25/10/2018 5:29 PM	Windows Batch File	1 KB
test_built.sh	18/09/2018 3:29 PM	SH Source File	1 KB

    b. *(Linux/MacOS)*:
        i. Run `npm run start:live` to launch the servers
2. View the website by typing the following in the browser:
    a. localhost:8000

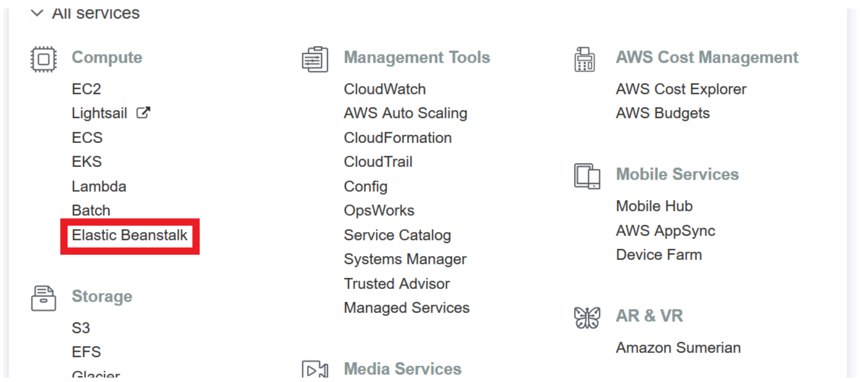
3.3. Deploying to AWS Elastic Beanstalk (EBS)

There are two ways to deploy an Elastic Beanstalk project:

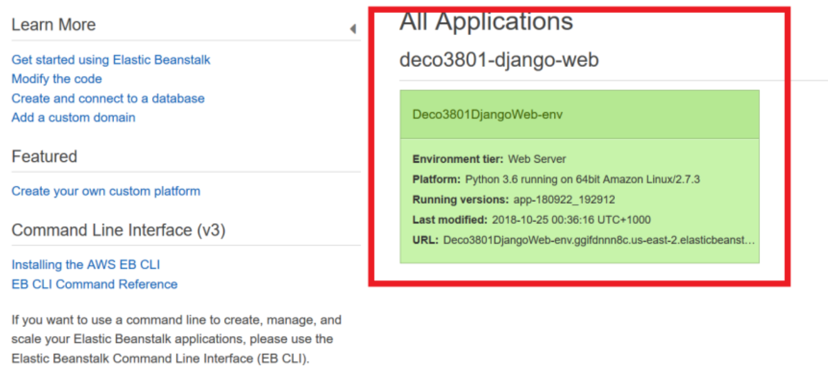
1. Using the web console
2. Using Elastic Beanstalk Command Line Interface (EB CLI)

3.3.1. Deploying EBS project through AWS web console:

1. Log into AWS console:
    a. console.aws.amazon.com/
2. Click on 'Elastic Beanstalk' under 'Compute' section



3. On the Elastic Beanstalk page, choose the 'deco3801-django-web' application:


Learn More
Get started using Elastic Beanstalk
Modify the code
Create and connect to a database
Add a custom domain
Featured
Create your own custom platform
Command Line Interface (v3)
Installing the AWS EB CLI
EB CLI Command Reference

If you want to use a command line to create, manage, and scale your Elastic Beanstalk applications, please use the Elastic Beanstalk Command Line Interface (EB CLI).

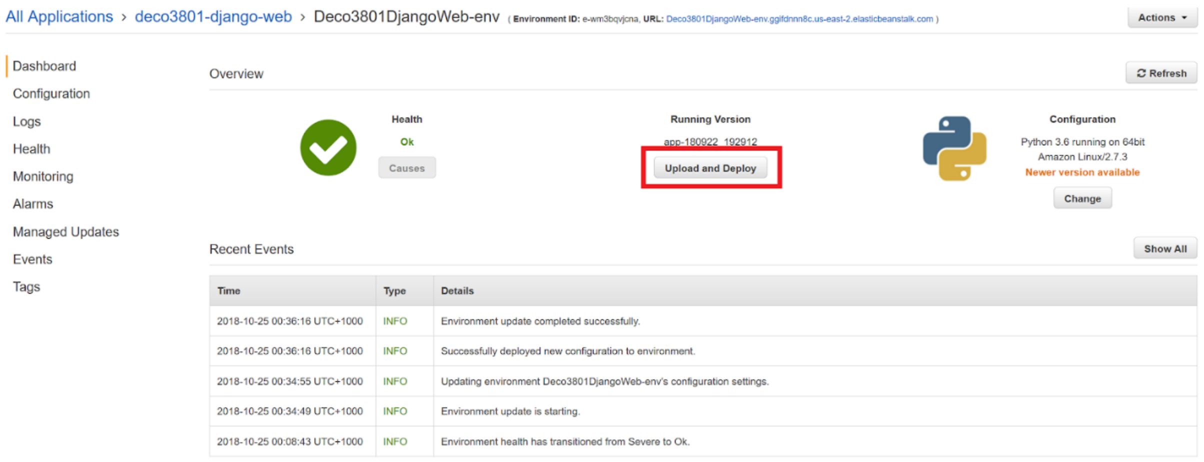
All Applications

deco3801-django-web

Deco3801DjangoWeb-env

Environment tier: Web Server
Platform: Python 3.6 running on 64bit Amazon Linux/2.7.3
Running versions: app-180922_192912
Last modified: 2018-10-25 00:36:16 UTC+1000
URL: Deco3801DjangoWeb-env.ggifdnnn8c.us-east-2.elasticbeanst...

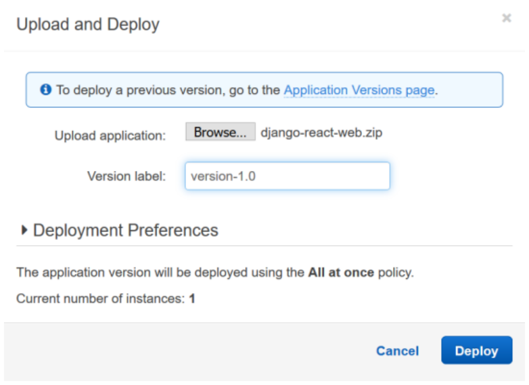
4. After the web console will has loaded the dashboard, click on the 'Upload and Deploy' button to deploy the application.



All Applications › deco3801-django-web › Deco3801DjangoWeb-env (**Environment ID:** e-wm3bqvjcna, **URL:** Deco3801DjangoWeb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com)

Time	Type	Details
2018-10-25 00:36:16 UTC+1000	INFO	Environment update completed successfully.
2018-10-25 00:36:16 UTC+1000	INFO	Successfully deployed new configuration to environment.
2018-10-25 00:34:55 UTC+1000	INFO	Updating environment Deco3801DjangoWeb-env's configuration settings.
2018-10-25 00:34:49 UTC+1000	INFO	Environment update is starting.
2018-10-25 00:08:43 UTC+1000	INFO	Environment health has transitioned from Severe to Ok.

5. Select the project files to upload, and define a version label to your liking:
    a. Compress your project folder into a .zip file. Different operating systems will have different methods.



To deploy a previous version, go to the Application Versions page.

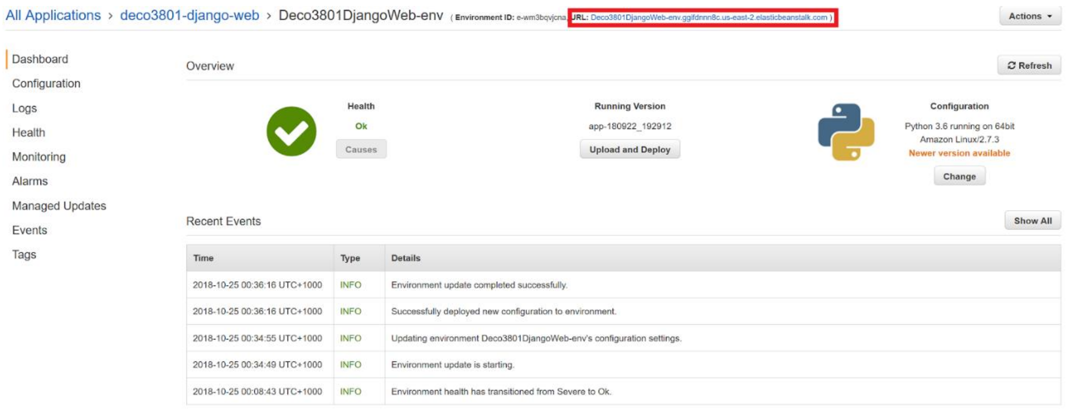
Upload application: Browse... django-react-web.zip

Version label: version-1.0

The application version will be deployed using the **All at once** policy.

Current number of instances: **1**

6. Click 'Deploy' to upload the website on to AWS. Click the link highlighted in red at the top of the page to view the website.



All Applications › deco3801-django-web › Deco3801DjangoWeb-env (**Environment ID:** e-wm3bqvjcna, **URL:** Deco3801DjangoWeb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com)

Time	Type	Details
2018-10-25 00:36:16 UTC+1000	INFO	Environment update completed successfully.
2018-10-25 00:36:16 UTC+1000	INFO	Successfully deployed new configuration to environment.
2018-10-25 00:34:55 UTC+1000	INFO	Updating environment Deco3801DjangoWeb-env's configuration settings.
2018-10-25 00:34:49 UTC+1000	INFO	Environment update is starting.
2018-10-25 00:08:43 UTC+1000	INFO	Environment health has transitioned from Severe to Ok.

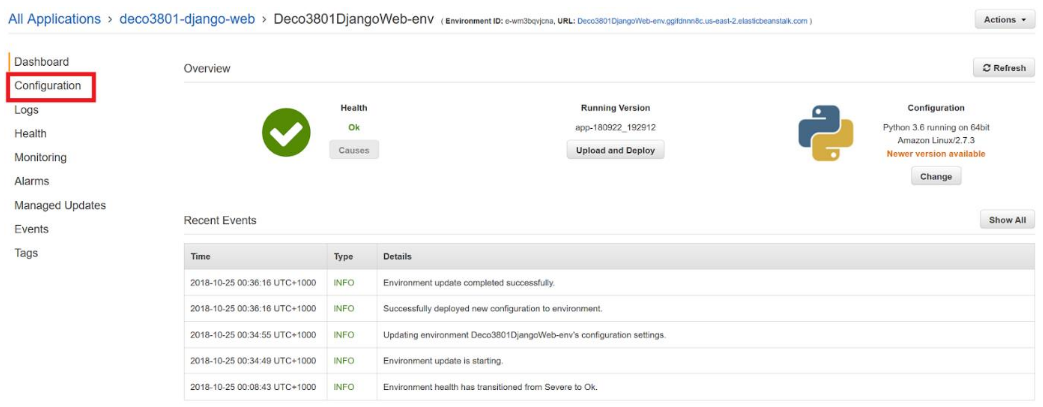
3.3.2. Deploying EBS project through Elastic Beanstalk Command Line Interface (EB CLI)

Deploying an EBS project through the EB CLI requires a number of steps which include the installation of the EB CLI, the configuration of the EB CLI and the deployment. The AWS website provides formal documentation and assistance for these steps which should be followed.

1. Install EB CLI
    a. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-install.html
2. Configure the EB CLI
    a. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-configuration.html
3. Deploy project, simply type in:
    a. Open up your terminal
    b. Run `eb deploy` to deploy the project
    c. https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/eb-cli3-getting-started.html#ebcli3-basics-deploy

3.3.3 Configuring Environment Variables in AWS Elastic Beanstalk Guide

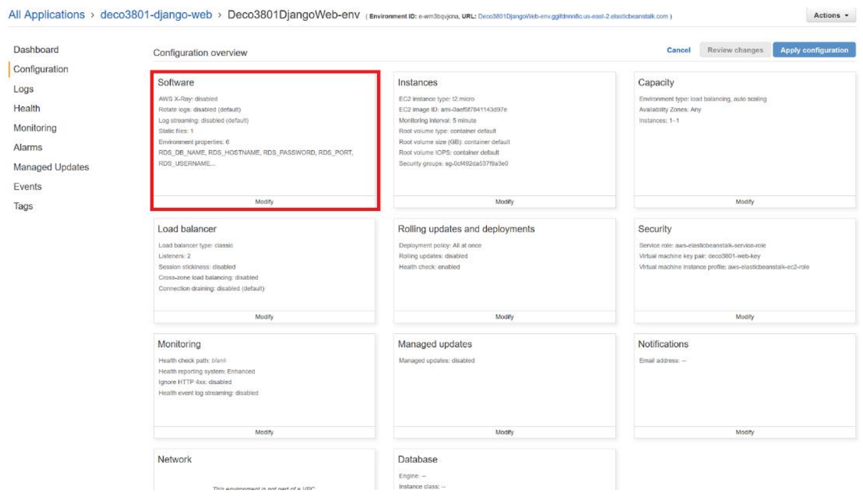
1. Follow steps 1 to 3 of *Deploying to AWS Elastic Beanstalk Guide* to reach the dashboard page of your Elastic Beanstalk application project.
2. Click 'Configuration' on the menu on the left side



All Applications › deco3801-django-web › Deco3801DjangoWeb-env (**Environment ID:** e-wm3bqvjcna, **URL:** Deco3801DjangoWeb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com)

Time	Type	Details
2018-10-25 00:36:16 UTC+1000	INFO	Environment update completed successfully.
2018-10-25 00:36:16 UTC+1000	INFO	Successfully deployed new configuration to environment.
2018-10-25 00:34:55 UTC+1000	INFO	Updating environment Deco3801DjangoWeb-env's configuration settings.
2018-10-25 00:34:49 UTC+1000	INFO	Environment update is starting.
2018-10-25 00:08:43 UTC+1000	INFO	Environment health has transitioned from Severe to Ok.

3. On the 'Configuration' page look for the 'Software' heading, and click 'Modify'


All Applications › deco3801-django-web › Deco3801DjangoWeb-env (**Environment ID:** e-wm3bqvjcna, **URL:** Deco3801DjangoWeb-env.ggifdnnn8c.us-east-2.elasticbeanstalk.com)

Software
AWS X-Ray: disabled
Rotate logs: disabled (default)
Log streaming: disabled (default)
Static files: 1
Environment properties: 6
RDS_DB_NAME, RDS_HOSTNAME, RDS_PASSWORD, RDS_PORT, RDS_USERNAME...

Instances
EC2 instance type: t2.micro
EC2 image ID: ami-0aef5f7841143d97e
Monitoring interval: 5 minute
Root volume type: container default
Root volume size (GB): container default
Root volume IOPS: container default
Security groups: sg-0cf492da537f9a3e0

Capacity
Environment type: load balancing, auto scaling
Availability Zones: Any
Instances: 1–1

Load balancer
Load balancer type: classic
Listeners: 2
Session stickiness: disabled
Cross-zone load balancing: disabled
Connection draining: disabled (default)

Rolling updates and deployments
Deployment policy: All at once
Rolling updates: disabled
Health check: enabled

Security
Service role: aws-elasticbeanstalk-service-role
Virtual machine key pair: deco3801-web-key
Virtual machine instance profile: aws-elasticbeanstalk-ec2-role

Monitoring
Health check path: blank
Health reporting system: Enhanced
Ignore HTTP 4xx: disabled
Health event log streaming: disabled

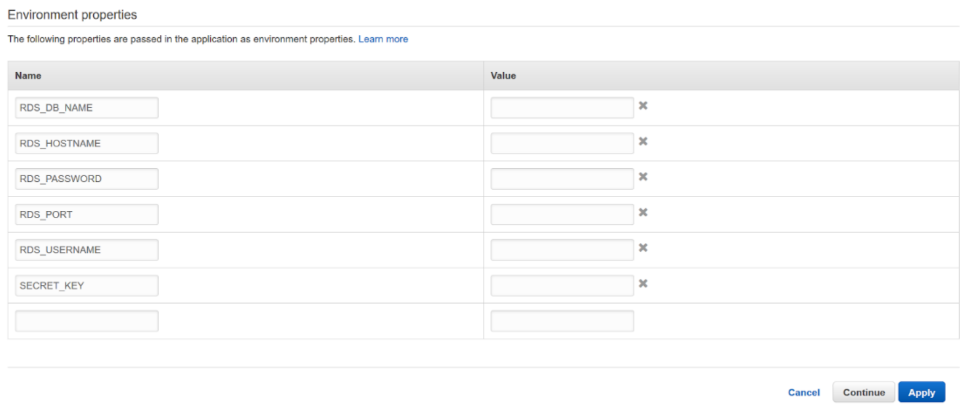
Managed updates
Managed updates: disabled

Notifications
Email address: –

Network
This environment is not part of a VPC

Database
Engine: –
Instance class: –

4. Scroll to the bottom of the page and look for ‘Environment properties’.
    a. Fill the environment variables accordingly to steps 4.1 of *Development Setup Guide*


Environment properties

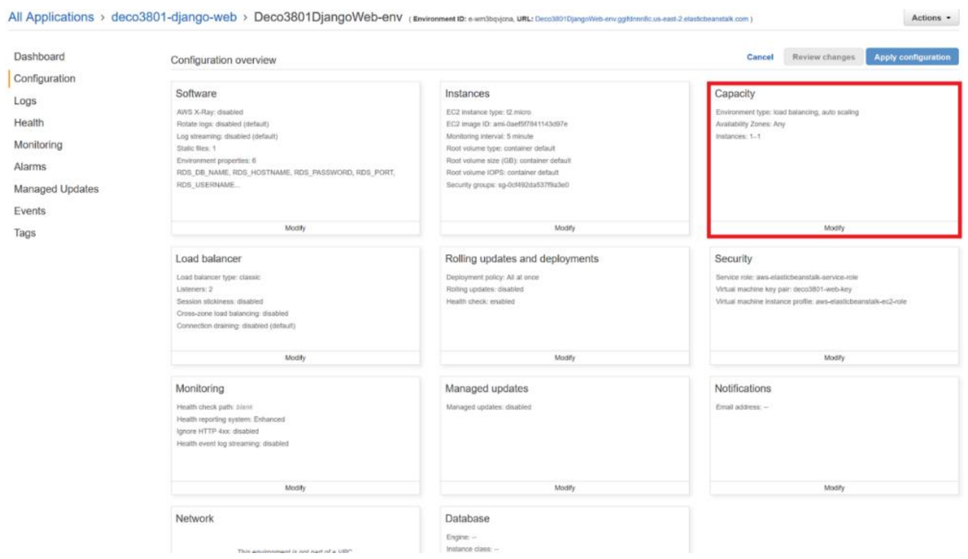
The following properties are passed in the application as environment properties. Learn more

Name	Value
RDS_DB_NAME	
RDS_HOSTNAME	
RDS_PASSWORD	
RDS_PORT	
RDS_USERNAME	
SECRET_KEY	

5. Click apply. Elastic Beanstalk will restart the server with the updated values

3.3.4 Suspending Elastic Beanstalk Server Temporarily

1. To get on the ‘Configuration’ page of Elastic Beanstalk, follow steps 1 to 2 from *Configuring Environment Variables in AWS Elastic Beanstalk Guide*.
2. Click on ‘Modify’ in ‘Capacity’ section.



(Environment ID: e-wm3bqvjcna, URL: Deco3801DjangoWeb-env.ggfdnnnfic.us-east-2.elasticbeanstalk.com)

Configuration overview

Software
AWS X-Ray: disabled
Rotate logs: disabled (default)
Log streaming: disabled (default)
Static files: 1
Environment properties: 6
RDS_DB_NAME, RDS_HOSTNAME, RDS_PASSWORD, RDS_PORT, RDS_USERNAME...

Instances
EC2 instance type: t2.micro
EC2 image ID: ami-0aef5f7841143d97e
Monitoring interval: 5 minute
Root volume type: container default
Root volume size (GB): container default
Root volume IOPS: container default
Security groups: sg-0cf492da537f9a3e0

Capacity
Environment type: load balancing, auto scaling
Availability Zones: Any
Instances: 1–1

Load balancer
Load balancer type: classic
Listeners: 2
Session stickiness: disabled
Cross-zone load balancing: disabled
Connection draining: disabled (default)

Rolling updates and deployments
Deployment policy: All at once
Rolling updates: disabled
Health check: enabled

Security
Service role: aws-elasticbeanstalk-service-role
Virtual machine key pair: deco3801-web-key
Virtual machine instance profile: aws-elasticbeanstalk-ec2-role

Monitoring
Health check path: blank
Health reporting system: Enhanced
Ignore HTTP 4xx: disabled
Health event log streaming: disabled

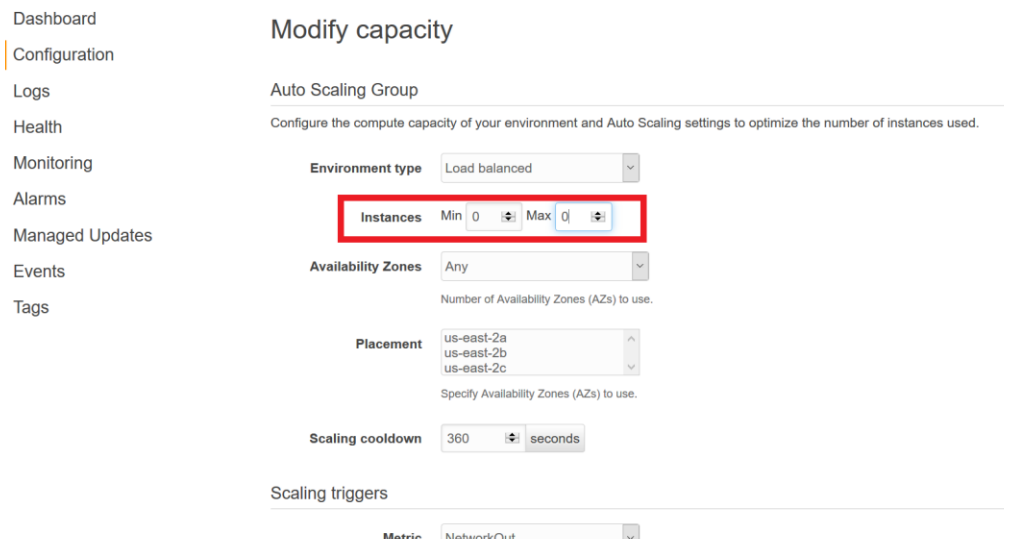
Managed updates
Managed updates: disabled

Notifications
Email address: --

Network
This environment is not part of a VPC

Database
Engine: --
Instance class: --

3. Under ‘Auto Scaling Group’ set the ‘Min’ and ‘Max’ ‘Instances’ to 0



Modify capacity

Auto Scaling Group

Configure the compute capacity of your environment and Auto Scaling settings to optimize the number of instances used.

Environment type: Load balanced

Instances: Min 0 Max 0

Availability Zones: Any

Number of Availability Zones (AZs) to use.

Placement: us-east-2a, us-east-2b, us-east-2c

Specify Availability Zones (AZs) to use.

Scaling cooldown: 360 seconds

Scaling triggers

Metric: NetworkOut

4. Hit the ‘Apply’ button on the bottom of the page.
5. To start the server again, change ‘min’ and ‘max’ back to 1.

3.4. Modifying the Database Schema Guide

There are two database being used in the project, SQLite and AWS RDS for development and production respectively.

3.4.1 Modifying the Database

For development SQLite database, follow Django’s official documentation (https://docs.djangoproject.com/en/2.1/intro/tutorial02/#introducing-the-django-admin).

For Amazon RDS, you can control the database with MySQL Workbench (https://aws.amazon.com/premiumsupport/knowledge-center/connect-rds-mysql-workbench/)

3.4.2 Modifying Database Schema Guide

1. SQL schema is created using Django's inbuilt object relational mapping.
   a. Official guide on Django Models: https://docs.djangoproject.com/en/2.1/intro/tutorial02/
   b. Official documentation on Django Models: https://docs.djangoproject.com/en/2.1/topics/db/models/
   The Model files can be found in *Section 7.1: Import Files* on source code structure

Note: When making changes to the Model, the following steps need to be followed to track changes to the database and update the relevant schemas, defined by your environment variables shown in *Appendix A: Environment Variables in DJANGO*

- Navigate to the root directory (django-react-web) of the project with terminal and run the following without quotes:
  1. "pipenv shell"
  2. "python manage.py makemigrations"
  3. "python manage.py migrate"

4.0 Test Plan and Results

4.1 Functional Testing Plan

Most of CECEWSN's functionality is provided by the server through a RESTful API which returns data in a JSON format at given URL endpoints. On the server-side, unit testing was performed on the API endpoints to ensure that they serve the correct data to the client-side. The API endpoints were tested as black boxes, with a given input (HTTP request with data if required), and expected output (HTTP status code and sometimes data).

On the web-client, testing was conducted on the websites functionality, user interface, and the communication between the web server and web client. These tests can be categorized into three groups - unit testing, integration testing, system testing, and acceptance testing.

Unit testing was employed to test code in individual files, classes, and methods across the front end and back end systems. The design of the software was tested through integration testing. This was achieved by testing the interfacing between several modules. System testing was used to test the requirements of the system and acceptance testing was used to test that the client needs and business needs were met and that the system was complete through end-users and the client representatives.

Unit testing on the server-side was performed using Django's built-in testing library based on Python's 'unittest' library. The unit testing and integration testing for the web client was conducted with two JavaScript libraries, enzyme and jest. These libraries were chosen because they are designed specifically for testing purposes on react applications.

4.2 Test Results

4.2.1 Unit Testing

4.2.1.1 Web Client Unit Test

The libraries enzyme and jest were used to test the behavior of individual classes and functions and the presentation of individual react components. The behavioral tests for individual units can be found under the `__tests__` folder which is located in the same directory as the files being tested. The presentational tests for individual react components can be found under in files suffixed with `.test.js`.

Description	Inputs	Expected Output	Actual Output	P/F	Comments
Test the UI of the <AppHeader /> React Component	none	Refer to Appendix C: Expected Output	Refer to Appendix C: Actual Output	P	

Table 4: Unit Tests for presentation of a single react component - <AppHeader />

Description	Inputs	Expected Output	Actual Output	P/F	Comments
Test logging in with invalid credentials	{ email: '', password: 'p4ssw0rd' }	Error message and redirect to the `Login` page	As expected	P	

Table 5: Unit Tests for behavior of a single react component

4.2.1.2 Server-side API Unit Test

Specific details on test inputs and outputs can be found in the testing code in /django-react-web/api/v1/tests.py

URL Endpoint	Description	Inputs	Expected Output	Actual Output	P/F	Comments
/api/v1/register/	Tests registration of a new user.	POST request with user details.	Status 200 OK. A new user is created and returned.	As expected.	P	If a username is not given it is randomly generated.
	Tests registration with an existing email address.	POST request with existing user details.	Status 422 Unprocessable Entity. The request is rejected and a descriptive error message is returned.	As expected.	P	
	Tests registration with incomplete or blank data.	Multiple POST requests with various incomplete or blank user details.	Status 422 Unprocessable Entity. The request is rejected and a descriptive error message is returned.	As expected.	P	
	Tests registration with an invalid email address.	POST request with invalid user details.	Status 422 Unprocessable Entity. The request is rejected	A new user with the invalid email	F	Server side checking of email addresses has not

			and a descriptive error message is returned.	address is created and returned.		been implemented.
/api/v1/users/	Tests getting user list without authorization.	GET request without credentials.	Status 401 Unauthorized.	As expected.	P	
	Tests getting user list.	GET request with credentials.	Status 200 OK. Returns a list of all users.	As expected.	P	A list of only one user is tested.
/api/v1/userid/	Tests getting the username of the logged in user without authorization.	GET request without credentials.	Status 401 Unauthorized.	As expected.	P	
	Tests getting the username of the logged in user.	GET request with credentials.	Status 200 OK. Returns the username of the logged in user.	As expected.	P	
/api/v1/userprofile/	Tests getting details of the logged in user	GET request without credentials.	Status 401 Unauthorized.	As expected.	P	

	without authorization.					
	Tests getting details of the logged in user.	GET request with credentials.	Status 200 OK. Returns details of the logged in user.	As expected.	P	
/api/v1/userprofile/update/	Tests updating user profile without authorization.	POST request without credentials.	Status 401 Unauthorized.	As expected.	P	
	Tests updating user profile.	POST request with credentials and update data.	Status 200 OK. Updates and returns the logged in user's data.	As expected.	P	
	Tests partially updating user profile.	POST request with credentials and update data.	Status 200 OK. Partially updates and returns user's data.	Status 422 Unprocessable Entity. Invalid parameters.	F	The server fails to update if a single update field is missing (email in this case).
/api/v1/password-recovery/request/	Tests requesting password recovery of an existing user.	GET request with existing	Status 200 OK. Sends a password recovery email.	As expected.	P	Sending the password recovery email is not tested

		user’s email address.				as it requires an SMTP server.
	Tests requesting password recovery of a non-existing user.	GET request with a non-existing user’s email address.	Status 422 Unprocessable Entity. The request is rejected and a descriptive error message is returned.	As expected.	P	
/api/v1/upload/	Tests uploading a file without authorization.	PUT request without credentials.	Status 401 Unauthorized.	As expected.	P	
	Tests uploading a file.	PUT request with credentials.	Status 201 Created. The uploaded file is stored with information of the upload time and user.	As expected.	P	The file type and contents is not validated.
	Tests uploading without giving a file.	PUT request with credentials and without a file.	Status 422 Unprocessable Entity. The request is rejected and a descriptive error message is returned.	As expected.	P	

Table 6: Unit Tests for a Server-side API

4.2.2 Integration Testing

Integration testing was not implemented in the project due to the system's design. The server's API endpoints work independently of each other so integration testing between endpoints would give the same results as unit testing. Testing the interactions between the client and server was attempted but fell short due to the limitations of testing libraries. The client relies on the server's API to retrieve and make changes to data, but the testing libraries run the client independently from any web servers, preventing integration testing.

4.2.3 System Testing

System testing was informally performed by team members continuously during development. The testing involved running CECEWSN locally and navigating its web interface to check specific features. Since bugs were immediately worked on once found by system testing, these test results were not recorded.

4.2.4 Acceptance Testing

4.2.4.1 User Acceptance Tests

Acceptance Requirement	Critical	P/F	Comments
The system must process one sample and view the results	Yes	F	Users can upload a file to be saved on the server. Sample processing can only be run on the production server due to its intense processing power requirements. This made it difficult to test alongside the uploading process and was not completed at the project end.
The system must be able to search for an existing compound	Yes	F	Users are able to search for the client's predefined chemical compounds. However, it does not search for newly discovered chemical compounds that are identified through their algorithms. Additionally, it does not provide a visualization of the occurrence and location of the compound.
Guests must not be able to view any user restricted features	Yes	P	Authentication is checked with JSON web tokens before showing restricted data.
Users must be able to use the process feature without external assistance	Yes	F	Users can upload files to be saved on the server, but, they cannot complete the process.
Users can send an enquiry and receive a response within 5 working days.	No	F	An enquiry is received and a response is added to the FAQ's section, but, the user does not receive a formal and direct response in relation to their enquiry.
Users can add custom and reusable HRMS instruments	No	P	Users can add custom and reusable HRMS instruments. However, these options are not displayed during the process feature.

Users can add custom and reusable chromatography systems	No	P	Users can add custom and reusable chromatography systems. However, these options are not displayed during the process feature.
Users can add custom and reusable columns	No	F	Adding custom and reusable columns has not been implemented.
View other registered users	No	P	Users can only view a list of all users but not individual user profiles other than their own.
Edit personal profile	Yes	P	User's email and name can be updated.

Table 7: User Acceptance Testing Result

5.0 User Evaluation Report

5.1 Executive Summary

This report provides a description and an analysis of the four User Evaluation methods utilized during the user testing phase of the web application: CECEWSN. Methods of evaluation includes heuristic evaluation, user scenarios, cognitive walkthroughs, as well as semi-structured experience interviews (SSEI) combined with contextual interviews (CI). Results of data evaluated show that our primary focus on user usability issues (e.g., ease of navigation, solid implementation of functionalities, simple interface) remained to be a vital strategy in designing the final product. The evaluation report concludes that the final web application has successful implementations of the findings from the four evaluation methods which tested for a satisfactory user experience. The few limitations of each method are discussed in each section. We propose that similar evaluation methods to be utilized when implementing future extensions. It is also important to note that although the team had creative freedom over the general interface design and aesthetics, all four user evaluation methods centered on providing a simpler user experience for scientists and other affiliated users who require access to the cloud-based algorithm using our web application. Therefore, we also recommend that in future user evaluations, each method utilized should be relevant and sufficiently centered around the correct target audience.

5.2 Semi-Structured Experience Interview & Contextual Interviews + Observations

5.2.1 Methods & Procedures

There were two main participants in the user tests, both of which participated as the clients and the main users of the CECEWSN web application. Four other participants have also participated in the tests, all of whom were university students with basic knowledge in computer science, UI, and other related fields. Semi-Structured Experience Interviews (SSEI) and Contextual interviews (CI) were conducted using a high-fidelity wireframe/prototype created on AdobeXD. Of the four university students, two participated in contextual interviews over the internet via digitally distributed wireframes. The remaining participants had participated in both SSEI and CIs in a standard library/casual environment using their preferred laptop or desktop devices.

The participants were given no clear scenarios of tasks when presented with the high-fidelity wireframes. Starting from the landing page of the web application, participants were given the freedom to navigate the application as they sought appropriate. The start of the initial observation began when the participants clicked on a navigation and ended when they closed out of the prototype.

Following the initial observation phase, a series of interview questions related to usability and ease of navigation were conducted. Participants were asked to briefly comment on ease of navigation as well as suggest similar applications that adopt similar navigation principles as our application. Sample questions can be found in *Appendix D: Semi-Structured Interview*.

5.2.2 Observations

During the observation phase, participants were observed for pauses, hesitations, and signs of struggle or confusion while navigating through the application. The time it took for each participant to start the application and terminate it was also recorded. Of the six participants, all six users took less than three minutes to start the application, navigate the application, and terminate it without any signs of struggle or confusion. All participants were also able to easily and successfully navigate to specific functionality pages. It is important to note that the users' cursor was mostly around the navigation bar – which allows quick access to all of the functionalities.

5.2.3 Findings & Recommendations

Usability reports and interview results showed the initial design provided smooth navigation and has adopted familiar layouts and principles obtained from the Heuristic Evaluation (also conducted in the design phase). The design was also consistent across different environments which include desktop/laptop setups and varying operating systems. In addition, the participants were not assigned specific tasks or exposed to the testing environment beforehand, which shows no other variables or carry-over effects may have influenced the results. Interview results also suggest that most users did not find any problems with the current implementation nor did they suggest alternative implementations.

It is recommended that in future development/additions (e.g., extensions), developers should utilize the simple, cleanly organized navigation bar and a dashboard. Results suggest that the navigation bar/dashboard combination greatly reduced navigation time and allowed quick access to the user's objectives.

5.2.4 Limitations

Semi-Structured Experience Interview and Contextual Interviews had limited input on testing functionalities of specific implementations (e.g., upload & process data/accessing forum) due to the limitations of the delivery. As the tests were centered on a high-fidelity wireframe, none of the actual functions were implemented. However, other evaluation methods (cognitive walkthrough) that were conducted should be sufficient in testing functionalities.

5.3 Scenarios & User Stories

5.3.1 Methods & Procedures

Stories and context behind why a specific user or user group would visit our website, what they would be utilizing, and how they are going to accomplish their tasks were created during the initial design phase. Trello boards were used to organize the scenarios during the initial design phase and implementation phase. Given the task sheet provided by the clients, each member of the Team Innovation Hackers created context-based scenarios and potential issues that might arise while the test users are navigating the web application. For each scenario created, members were assigned to design a potential solution to the scenario, which were implemented in the final build phase.

5.3.2 Observations

User stories and scenarios on the team Trello Board allowed us to find potential solutions and methods on accomplishing tasks as the application was building. Main pathways to accomplishing a specific goal were made clear before the method or function was implemented. It also allowed the group to discuss alternative pathways to achieving the same goal based on the stage of development.

5.3.3 Findings & Recommendations

Utilizing Trello and other similar applications to manage user stories to replicate user stories and potential scenarios proved to be an effective method of user evaluation. Stories and scenarios created a clear objective before development phase occurred, which allowed us to discuss the main and alternative methods to accomplish it. It is recommended that user stories and scenarios be utilized while implementing the main functionalities as they provide valuable insight in how the goals can be achieved.

It is also recommended that user stories be created around the context of target users. In this context, the target users (mainly scientists) will utilize our web application to access the cloud-based algorithm that will assist in analyzing and processing raw data. It is important to note that, while the team was granted creative freedom in the context of interface design, the core functionalities of the application needs to be strictly centered around the core, targeted users.

5.3.4 Limitations

The current implementation of the functionalities was specifically designed for scientists and other related users. While the clients have made clear the target users are specifically scientists, in some cases, other users may struggle to utilize our implemented functionalities. Our user stories and scenarios were effective in defining the possibilities of most, but not all the goals.

5.4 Heuristic Evaluation

5.4.1 Methods & Procedures

Utilizing the sample website that the clients have provided us with, some members of Team Innovation Hackers created a list of design principles and identified which of the principles the final product should replicate and avoid. Other similar social media web-application such as Facebook was also used for comparison. Principles adopted from similar applications were implemented in the initial designing/prototyping of the wireframes which was used for observations and interviews.

5.4.2 Observations

Given the functionalities that needed to be implemented as well as the vastly different user groups, it felt appropriate to discard most of the UI implementations of other social media web applications such as Facebook and Twitter. Based on the idea that our web application will be utilized mostly by scientists with specific purposes, it was inappropriate to clutter the feed/landing page with unnecessary information that has been adopted by Facebook.

5.4.3 Findings & Recommendations

Adopting the interface of other popular websites such as Facebook proved to be an effective method in reducing navigation time/usage time as the results show users are already familiar with it. Heuristic Evaluation also provided alternative methods in designing/prototyping wireframes for further observations and contextual interviews.

Review *Appendix F: Heuristic Evaluation* for detailed principles from sample websites provided by the clients

It is also important to note that the selection of principles was limited to what the clients had initially requested. While the team had freedom in the overall interface design, usability and the overall experience had to be appropriate for the specific users of the website. In this context, scientists and other affiliated users who require access to the cloud-based algorithm were the focused target of the design and there the evaluation methods were strictly centered around providing an intuitive and quick user experience. It is recommended that in future implementations of extensions, that user testers find appropriate heuristic samples that are relevant to the focused users of this application.

5.4.4 Limitations

Although comparing and adopting principles of other popular applications may provide familiarity and ease of navigation, they might not be directly relevant to our specific web application.

5.5 Cognitive Walkthrough

5.5.1 Methods & Procedures

One of the evaluators picked out two participants from the initial SSEIs and Cis to participate in the cognitive walkthrough evaluation. After completing the first observations, a set of tasks were presented to the participants. As the current implementation of the web application utilizes context-specific functions specifically designed for scientists with a set goal, tasks were assigned to the participants to accomplish specific goals. For example, participants were assigned to upload a sample raw data to the application to be analyzed and displayed on the final page. The Cognitive Walkthrough was conducted using an updated, functional Minimal Viable Product (MVP) as it was designed to evaluate findability, accessibility, and usability of the functionalities built into the application.

5.5.2 Observation

All of the functionalities within the application were easily findable, accessible, and usable. Simple tasks such as logging in, signing up, and accessing the dashboard replicated similar results as the first SSEI and CI evaluation methods. Specific functionalities as accessing the scientific forum and uploading raw data to be processed were also easily found, accessed, and utilized by the participants.

5.5.3 Findings & Recommendations

Cognitive walkthrough yielded similar results as the SSEIs and CIs. Users had similar usage time even with the assigned tasks which suggests that the user interface was intuitive and core functionalities were easily findable, accessible, and usable.

It is recommended that in future user evaluation using contextual walkthroughs, a new set of participants should be acquired. This will allow a more effective testing of system learnability without the influence of carryover effects.

5.5.4 Limitations

Given the participants were already exposed to the general layout and navigation of the web application through SSEI and CI evaluation methods, the results from the cognitive walkthrough may be slightly unreliable. Participants may have developed experience and/or familiarity with the web application, allowing easier access to specific functionalities.

6.0 Reflection

6.1 Team Management and Process

Throughout the entirety of this semester, the team has utilized scrum methodology in conjunction with a trello board to keep track of each team member's progress throughout the project. Annie, the scrum master, and the team attended fortnightly scrum meetings to establish constant communication and the next sprint cycle. Additionally, contact between team members was maintained though predominantly Facebook messenger, while contact with the clients was made predominantly with frequent emails. There was some minimal formal communication regarding development bugs or clarification of expectations recorded on slack.

In these meetings the team, headed by the scrum master, would participate in a brief standup updating the members on the contribution each individual had made in the previous fortnight, as well as any testing issues arising from their work. Following this standup, the team would then discuss as a whole the user stories to be focused on in the oncoming sprint cycle. Once these stories had been agreed upon, a list of tasks would be derived, with each member taking charge of a minimum of one trello card, and the sprint cycle would start.

On reflection, the team utilized scrum methodology somewhat effectively, although not to its full potential. While scrum methodology recognises no distinguishable roles other than product developer, the team was widely varied in their skill sets, experience, and personal preference. This saw the uneven distribution of work between team members during sprints that were design or development heavy, particularly at the start and end of the semester since front-end design and user experience were heavily favored in the starting user stories, and development and testing in the final user stories of the development cycle. Documentation was left as a free for all, to be written by those who had time to spare, or out of necessity due to certain team members' expertise in that particular area.

Although the team personally recognized this imbalance as a necessary evil, in retrospect it may have been more advantageous to use pairs of team members with different skill sets to ensure even distribution and therefore effective and timely delivery of deliverables in every single sprint cycle. Not only would this reduce these small groupings to essentially very capable product developers with all skills required, but each team member would be able to aid and even educate their peer on their own area of expertise. This would also erase the need for documentation to be written by the exact people who contributed to that precise area of development, as the task would also be able to be distributed across two people.

In regards to the Trello board, the team did not fully utilize the range of functionalities provided by the board. While the team adhered to the Kanban flow by taking responsibility for a card/task of their choice, they were unaware that they needed to specify testing criteria for the follow up quality assurance by their peers. This, in conjunction with the team's already present preference for only taking cards in their own areas of expertise, resulted in card flow blockages where the same few people would angastest the same people's work every sprint cycle. Moreover, as there was no standardized testing criteria, different members had distinct standards for whether the produced work was acceptable. This in turn resulted in many discussions and revisions of peers' work during weekly meetings and by extension the loss of speed and efficiency.

Additionally, as the sprint cycles progressed team members became less and less reactive on the Trello board. While tasks were still being defined as part of the sprint, the cards themselves were not being moved across to the "In Progress" or "Testing" columns, frequently remaining there until the next sprint meeting. This led to some minor confusion among the team about which tasks were actually still pending completion. This dysfunction may be attributed to several reasons. The main one was that team members felt that the presence of this board was somewhat redundant, as the same team members were repeatedly taking the same type of task (whether this was front end development, back end design, or user experience and testing) and therefore no one needed to express their intentions or progress to each other. Another reason was that as discussions revolving around work quality pervaded the scrum meetings more frequently, the time the team wrote the Trello cards in decreased, and therefore the effort in defining outlined tasks also decreased.

Despite the Trello board format becoming less regimented as the project progressed, team communication was consistently good and efficient both within the team and with external clients. While the team communicated mostly unofficially through a Facebook Messenger group chat, the chat was purely functional and detailed mostly issues contained within that sprint cycle. Each message within the team chat was first labelled with the issue - (e.g. Attendance) - then a closer detailed explanation of the topic. Directions to the Git repository or a link to the relevant Google document would then be inserted at the end for easy access and reference. Issues relating to certain people would be addressed directly to that person through the use of the tagging function, ruling out concern for those not involved. With these methods, the team chat was clean, minimal, and highly efficient in transmitting vital information. The team communication with the client progressively improved as the project continued. Through following feedback after the first communication mark, the team sent out a weekly summary email updating the client on progress on Monday or Tuesday each week. Agendas and Meeting minutes were also sent promptly within 24 hours of each meeting date.

Finally, dispute resolution and project ideation within the team was highly efficient. Each member contributed different perspectives and skill sets, and each member was afforded a high amount of respect by their peers while sharing their perspectives. Conflicts did not arise except in the instance of what constituted good quality work, and these were quickly dispersed due to the efficient handling of the situation by each team member. In the case that the offending team member could not remedy or improve their work, they would compensate the team member who did fix it by offering to do an extra amount of work in the next deliverable. This solution provided incentive for every team member to help their peers.

6.2 Design Assessment

Throughout the design process, our team would fortnightly meeting with our client. In these meetings the team would go in with an agenda set and would ask our clients a variety of questions we had accumulated since the last meeting. A majority of the questions posed to our clients where to do with certain aspects of the product in development. However the team tried to use these meetings more to gain feedback on the current design of the website, in the hope of gaining critical feedback to see where we could improve upon.

During the first meeting our client's presented a PowerPoint with an idea of what they would like included into the site. From this PowerPoint the team was able to gain understanding of what exactly the client was wanting the website to be focusing upon in terms of functions and features. The client had mention during this initial meeting that we would have total control on the design of the website but would like us to focus on these certain features but more importantly present those features in a user friendly and easy to use format. Whilst building a user friendly website should able be focused upon while designing a website, our client did stress this point. As the functions the client were wanting to implement are very complex and without an easy to follow process this site would not be able to achieve the goal of increase the efficiency of doing research with the website.

With the minimal constraints given to us by the client in terms of the design of the website we were able to develop the site to what we deemed as appropriate. This free rein given to us whilst good in some respects it was also detrimental when it came to received critical feedback from our client. It was challenging for us to receive the feedback we needed to know for the site, in terms of what we could possibly be improve upon. Through the fortnightly meetings we would show our clients the progression of the site and would consistently receive feedback with client stating the site is looking good. It could be possible they were pleased with our designs and actually had no critical feedback they could give us, whilst this could be possible it is highly unlikely to be true. However with this statement of us receiving little to no critical feedback on the design from our client it should be noted that they did have 3 other teams. With our client working with 4 separate teams including our own, it is far more likely that our client had made a conscious decision choosing to not give us critical feedback based upon them wanting to see where each team would end up in the final stages of development without them directly affecting the design.

The minimal critical feedback we received from our client and autonomy to work on the design ourselves forced the team to look elsewhere for answers on how we could improve on the design of the website. Through the user testing we had done in different stages of development did however confirm that our website had be built with bases of a solid design. A solid design in terms of users returning to us with minimal issues and mentioning and displaying to us how easy they able to move around the site. With this ease of use for users our team was very pleased with outcome as this had demonstrated to us that we had captured one important goals of the site and that UX Profile we had developed in early stages of the development was appropriate.

After the first initial meeting the UX Profile was one of the main focuses the team had to ensure was done correctly before we could begin production of CECESWN. As mentioned earlier our design did take inspiration from some of the suggestions our client had made to us whilst they presented to us during the first meeting with them. One of the points of inspiration we choose to consider and ultimately input was the easy to recognize icons for the navbar and homepage. With this main focus in early stages our team initial collaborate and equally inputted into decisions for the design of the website, the final production of our UX Profile did however come down to two of members who have a background UX development. By having these two working on the final stages of what the design should be definitely did help us to ensure that design was being built correctly.

Whilst the UX profile developed in early stages return initial positive feedback allowing us to progress, the feedback we received of the implementation of this profile into becoming the design of the website is of far greater importance. As the user feedback we received from ultimately confirmed to us that the design we had implemented was applicable in capturing the goal our client was seeking from the design of the website, of it being easy to use and being user friendly. It should be mentioned that the vast majority of the design implemented was by Minh Nguyen. Minh has done an excellent job in ensuring the front-end of our website was developed closely to our profile and allowing us to receive all the positive feedback we had gained from User Testing.

6.3 Final Product

As most of our user testing took place during the initial design and the building phase, critiques of our final product were limited to direct feedbacks from our clients. The clients being the core users of the application, their feedbacks were directly applied to our future development plans. As our initial design user evaluation results suggested, focusing on a simple user interface with emphasis on ease of navigation proved to be our most valuable design decision. To reiterate, allocating more resources on implementing usable functionalities was more appropriate than aesthetics and/or other extensions that may have violated our core initial design principles. However, the critiques also addressed the shortcomings in functional coverage. While our design principles remained consistent throughout the build phase, some of the functionalities were not fully covered. In addition to the positive feedback from the clients, we recommended that future implementations to the web application should follow the team's initial design principles. Detailed summary of the initial design process and focus can be found in the user evaluation methods section of the report. A detailed summary of the team's shortcomings can also be found in functional coverage.

6.4 Future Development Plans

The testing and evaluation results shows that the product is incomplete. The future development plans for this project mainly revolves around completing the remaining tasks that were specified on the project brief as well as improving the product to pass the systems and acceptance testing.

The functionality in the client and server has been designed and developed as required by the project brief. However, the communication between these two systems is incomplete and so the future development plans primarily focuses on interfacing these two systems. This includes making modifications to the software so the web application has a better user control flow. Additionally, the required tasks were mainly functional based and so the design work did not receive much attention. Moving forward, the design can be improved to look more aesthetic by maintaining a theme and consistent colors.

Future plans for this product also address concerns over its robustness and security. The development team employed the three tier client server model, however, because the communication between these systems were incomplete, these quality metrics need to be tested again.

CECEWSN employed a self-signed SSL as we are committed to the security of our users. This was used to encrypt data transmission across the HTTPS protocol. However, because the SSL is self-signed, it is not tied to a certificate authority. As a result, it is not stored in a trusted certificate authority’s database and cannot be authenticated. This leads to an unsafe warning alert on the user interface, which is not ideal to the user experience. Moving forward, we will transition into using a SSL that has been signed by a trusted authority.

7.0 Source Code Structure

Code structure has stayed largely unchanged from the MVP codebase. There are two directories in the root of the Git repository: *django-react-web/* and *forum-web/*. Each of these two folders contain code to be run on a web server. *django-react-web* serves the CECEWSN site where users will be able to upload their data to be processed. forum-web serves a forum site where users can discuss the results of the data

The CECEWSN site was developed using Django, a Python web framework, on the server-side and React, a JavaScript library, on the client-side. Django manages the database, authenticates users, provides data through an API, and serves the React application. React provides a responsive, component-based user interface, which is styled through Semantic-ui. The component's view is based on states and these states are managed by a state management library, Mobx.

The files in *django-react-web/* are either configuration files (*Pipfile*, *package.json*, *manage.py*) or automation scripts (*.sh* and *.bat* files). The directories in *django-react-web/* are described in *Table 1* the important directories have been bolded.

Directory	Contains files for	Description
users/	Django	Django app containing data models of Users and Affiliations
api/		Django app defining API endpoints
home/		Django app that serves the React application
backend/		Django configuration files
frontend/	React	Component files for the user interface
config/		Webpack configuration files
scripts/		Automation scripts for testing and building the React app

Table 8: Description of the Directories

7.1 Important files

users/models.py contains definitions of User and Affiliation 'models', Django's abstraction of a database table. The current schema has been displayed as an ER-diagram in Figure 1.

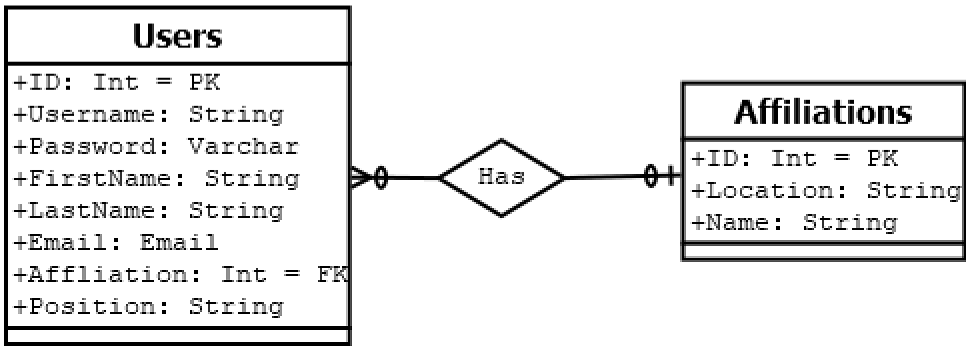


Figure 1: Current ER Diagram

api/v1/views.py defines the API endpoints that the React application can access to query the database. The server's API has been versioned with directories so that updates won't have to break client applications. Most of the API endpoints has been access restricted to signed in users using JSON Web Tokens.

api/v1/tests.py contains all the unit tests for the server-side API.

The file structure of the web client is described in *Figure 2*. The JavaScript modules are bundled by Webpack, a JavaScript bundler, which outputs a single JavaScript file that is minified and works with legacy browsers*. frontend/src/* contains the source files for building the client-side React application. The folders in *frontend/src/modules/* represent individual features that make up the website and consist of files that describe the user interface for that given feature. The folders in *frontend/src/api/* consist of classes that provide access to the backend API. *frontend/public* consists of scripts and style sheets that are used by the custom made react components.

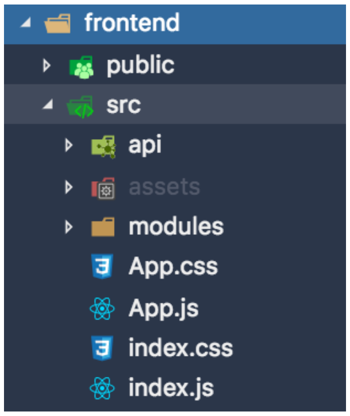


Figure 2: File Structure of Web Client

8.0 Functional Coverage

The project brief describes a primary task and an extension task, each of which can be decomposed into a collection of functionalities. This section outlines the functionalities of these two tasks as well as their completion status.

Table 9 outlines the functional requirements that fall under the primary tasks. These functionalities revolved around the basic web application and the integration of the AWS Architecture.

Status	Feature	Functionality
X	Upload and process data	Upload environmental data to process with the algorithms and display the results.
Y	View or download reports	View all historical results of processed data.
Y	Search for a compound	Search the website for a known or newly discovered chemical compound and view its location on a 2D globe.
Y	User	View the profile of the logged in user.
Y	Users	View the profile of all registered users.
Y	Acquisition methods	Allows users to add/remove HRMS instruments and chromatography systems for reusability.
Y	Forum	Platform for discussion and collaboration.
X	CEC alerts	Subscribe to receive CEC alerts for a chemical compounds that are discovered in the algorithm.
X	Documentation	Detailed page of materials relating to environmental science.
Y	Help/FAQ	Detailed page of common issues and solutions, as well as a platform to seek assistance.

Table 9: Functional Requirements of the primary task

Table 10 outlines the functionalities that fall under the extension tasks. The extension tasks is comprised of five optional tasks, each of which improve the product in a unique and different way. The priority of the project was to complete the primary tasks effectively instead diverging focus to complete the extension task as well. As such, there was not enough time to move onto the functionalities in Table 9.

Type	Description
Relational Database	Building and designing a AWS RDS back-end database for AWS
Visualization	Implementing visualization tools
Parallelization	Implementing a naive distribution of data across EC2 instances for parallelization
Translate code	Translating MATLAB scripts to python OO.

Table 10: Functional Requirements of the extension task

9.0 System Architecture Diagram

The project follows a 3-tier web application for both CECEWSN home landing page and the forum. The figure below shows the system architecture diagram for each tier of the web application with their respected AWS services.

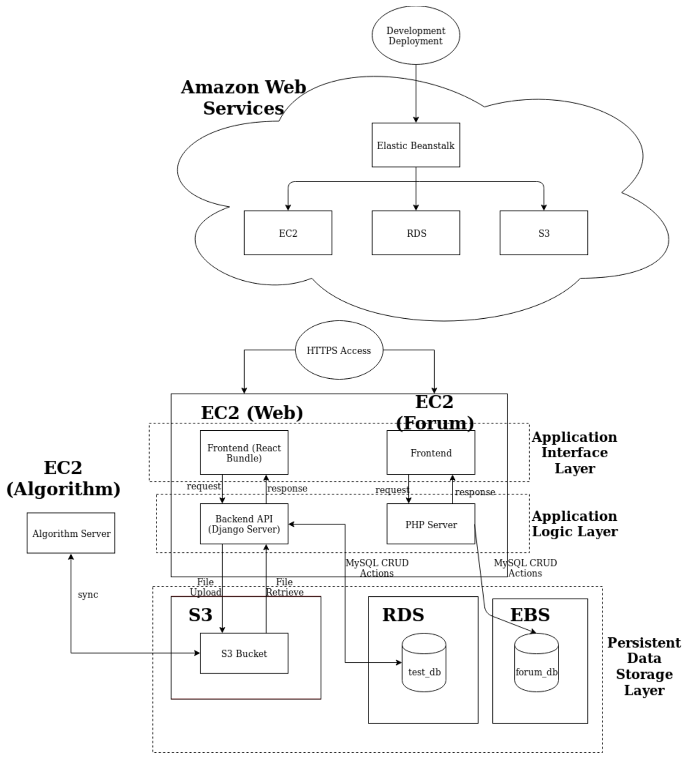


Figure 3: System Architecture

The CECEWSN landing page and the forum will be hosted on two different EC2 servers. The primary motivation for splitting these applications is to allow CECEWSN landing page to run a python backend that is capable of directly implementing the client's statistical algorithms, while allowing the usage of third-party PHP library for the forum. Another main advantage of splitting these applications is the increase in modularity for websites featuring different purposes and functionalities.

The Django backend only serves as a basic HTML template to render a React bundle on the client side. The Django server does not execute any dynamic server rendering. Instead, the Django backend provides a RESTful API server that uses token-based authentication for the frontend to request data, increasing scalability and modularity for any future applications.

However, there are a few user stories that have yet to be implemented primarily due to limited time and resources. One of them is the forum database that is currently being hosted locally on the EC2 instance along with the forum server. To allow all users accessing from different places, we will eventually migrating it over to Relational Database System for final production in the future. The other user story that is yet to be implemented is the Amazon S3 which will be used to store files uploaded from Django server to support EC2 more smoothly.

10.0 Appendix

Appendix A: Environment Variables in DJANGO

Several features of the application depend on environment variables. Without these environment variables, the project is configured to use local tools such as SQLite Database and local storage without using any AWS services:

Environment Variable	Purpose	Default
DJANGO_DEV	Determines whether Django will render the frontend from a bundled JavaScript frontend (none) or render from the React server on port 4000 (any other value)	none
SECRET_KEY	Secret string that Django uses to generate any cryptographic signing	'Secret'
AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY	AWS IAM credentials is used in order to access AWS services	none
AWS_STORAGE_BUCKET_NAME	The values of the aforementioned variables will not be included in this documentation for security purposes. They are values that can be found in the AWS console.	none
RDS_DB_NAME		none
RDS_USERNAME		none
RDS_PASSWORD		none
RDS_HOSTNAME		none
RDS_PORT		none

Appendix B: Virtual Environments

When you do pipenv install, all your project dependencies will be stored in in a ‘virtual environment’. Additional details on Pipenv and their motivations can be found on their official document:

- Official Documentation: https://pipenv.readthedocs.io/en/latest/
- Motivation behind virtual environments: https://realpython.com/python-virtual-environments-a-primer/

Therefore, without invoking the virtual environment, python will not be able to find all the installed packages on the machine.

Appendix C: Expect and Actual Output

Expected Output

```
exports[`Set: Rendering react components Header component
1`] = `
<div
  className="text-center p-5 App-header"
>
  <h2
    className="ui header text-light"
  >
    Chemicals of Emerging Concern Early Warning Social
Network
  </h2>
</div>
`;

    <div
      className="column"
    >
      <div
        className="ui label"
        onClick={[Function]}
      >
        <i
          aria-hidden="true"
          className="user massive icon"
        />
        My Acquisition Methods
      </div>
    </div>
  </div>
</div>
`;
```

Actual Output

```
<div
  className="text-center p-5 App-header"
>
  <h2
    className="ui header text-light"
  >
    Chemicals of Emerging Concern Early Warning Social
Network
  </h2>
</div>
`;

    <div
      className="column"
    >
      <div
        className="ui label"
        onClick={[Function]}
      >
        <i
          aria-hidden="true"
          className="user massive icon"
        />
        My Acquisition Methods
      </div>
    </div>
  </div>
</div>
`;
```

Appendix D: Sample Interview/Observation Sheet

Sample User

Informed consent form

User interface testing for DECO3801

This user testing exercise is for educational purposes only, and is being conducted as a course requirement for DECO3801, a course about human-computer interaction.

You will be asked to interact with a paper prototype, computer program or system, and/or to answer questions about your interaction. We are testing the design; we are not testing you in any way. The test will require no more than an hour of your time, and potentially less.

Consent is voluntary – you do not have to participate if you don't want to. If you do participate, you may withdraw your consent at any point, and all your data up to that point will be destroyed and not used.

All data collected is confidential and will be kept in a secure location, and your data will be indexed by a participant ID rather than by name.

All your data, including any recordings, will be erased/destroyed once class grades are released.

There is no reimbursement or payment for participation.

I have read the information above and give my consent to participate.

Participant Name: Sample User

Participant Email: sampleEmail@hotmail.com

Signature: S.U. Date: 10 / 09 / 2018

Researcher Name: Justin JiHoon Oh Date: 10 / 09 / 2018

Researcher Signature: J.O.

Researchers:

Justin JiHoon Oh, Annie Huang

Because this is an in-class educational exercise, performed by course students with UQ students, family or friends only, formal ethics approval has not been sought.

Observation Log

Activity	Notes
Open/Start Wireframe	Showed no particular signs of struggle/confusion
Navigate, click through all possible functions (dashboard, profile, upload/process data, etc.)	0 pauses, showed no signs of struggle. No questions. No pauses between clicks.
Closed App	No struggle, little disinterest.

Time Taken: 1 minute 35 seconds

Questionnaire

Interview Questions

1. **Was it easy to open the application?** No
2. **Was it easy to find a new route or access a different page?** Yes
3. **Were there any functions that you wanted but were not available while you were doing ____(insert the step(s) that the user is observed to have had difficulty at)?** No
4. **Were there any functions that you could not find easily? Why do you think it was difficult to find?** No

SSEI

1. **At all times, did you know what the system was doing? If No, what could be implemented to make this obvious?** Yes
2. **Did you have to think carefully about the process you were doing?** No
3. **Did you have trouble finding specific tools or navigating to certain pages?** No

When you were seeking to complete a task, did you find it difficult to find the next step? No

Appendix E: User Scenarios & Stories

Adopted from UQ HCI DECO2500 2016

1. Topic, situation of concern, and problem statement

Topic/ Project: Web Application CECEWSN

Situation of concern:

There needs to be an application that utilizes the algorithm created to process and analyses scientific data uploaded by scientists. The same users should also be able to store their data, data acquisition methods, files, access a forum for scientific discussions, and view their colleagues' user profiles

Problem statement:

Design and building a web application that combines the principles of Social Media platforms while allowing users to handle, upload, and download raw data.

2. Statement of initial requirements for the system to be developed

1. User should be able to navigate to the functionality of the application within 30 seconds without assistance
2. The application should use consistent icons and/or navigation (dashboard)
3. Interface should be easy to understand intuitive
4. The interface must be clean and clearly segmented so users can identify the area in the web application they want to navigate to.

3. Detailed description of personas
    a. **Steve, A Test User**
        i. ***Pain/Goal points:*** As a regular user of the application, user would like to be able to upload and process data by progressing through the necessary steps without any hindrance.
        ii. ***Goals:*** Rely on the cloud-based algorithm to easily take in my data, analyses it, and display the necessary information on my screen now and later.
        iii. ***Expectations:*** An application that can help Steve upload/process his raw data, have the file stored in a server which he can view, download, or save for later uses.

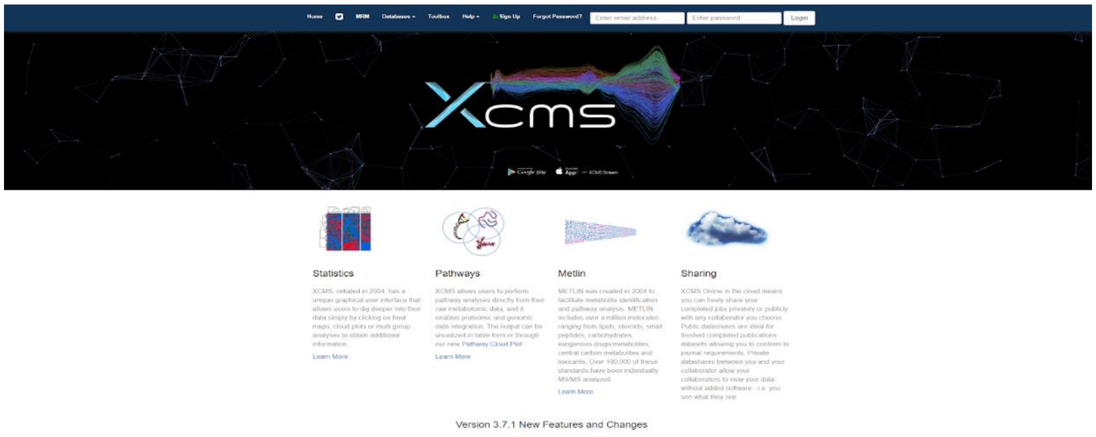
Appendix F: Heuristic Evaluation Designs

https://xcmsonline.scripps.edu/landing_page.php?pgcontent=mainPage

https://www.normandata.eu/?q=node/19

Design Principles

1. Mental Model (User's perspective)
    a. Consistency (ease of navigation)
    b. Obvious/Easy access to functionalities
    c. Organization
2. Simplicity
    a. No clutter
    b. Organized Feed
    c. Clear & distinct access points to functionalities
3. Functionality over aesthetics
    a. Consistent colors
    b. Organized methods



Statistics

XCMS, initiated in 2004, has a unique graphical user interface that allows users to dig deeper into their data simply by clicking on heat maps, cloud plots or multi group analyses to obtain additional information.

Learn More

Pathways

XCMS allows users to perform pathway analyses directly from their raw metabolomic data, and it enables proteomic and genomic data integration. The output can be visualized in table form or through our new Pathway Cloud Plot.

Learn More

Metlin

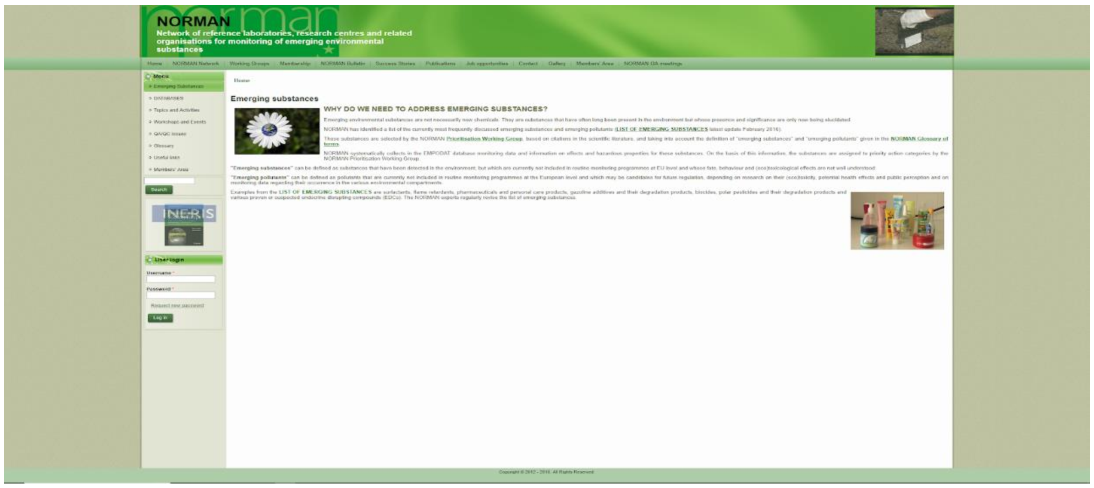
METLIN was created in 2004 to facilitate metabolite identification and pathway analysis. METLIN includes over a million molecules ranging from lipids, steroids, small peptides, carbohydrates, exogenous drugs/metabolites, central carbon metabolites and toxicants. Over 100,000 of these standards have been individually MS/MS analyzed.

Learn More

Sharing

XCMS Online in the cloud means you can freely share your completed jobs privately or publicly with any collaborator you choose. Public datashares are ideal for finished completed publications datasets allowing you to conform to journal requirements. Private datashares between you and your collaborator allow your collaborators to view your data without added software - i.e. you see what they see.

Version 3.7.1 New Features and Changes



Home

Emerging substances

WHY DO WE NEED TO ADDRESS EMERGING SUBSTANCES?

Emerging environmental substances are not necessarily new chemicals. They are substances that have often long been present in the environment but whose presence and significance are only now being elucidated.

NORMAN has identified a list of the currently most frequently discussed emerging substances and emerging pollutants (LIST OF EMERGING SUBSTANCES latest update February 2016).

These substances are selected by the NORMAN Prioritisation Working Group, based on citations in the scientific literature, and taking into account the definition of "emerging substances" and "emerging pollutants" given in the NORMAN Glossary of terms.

NORMAN systematically collects in the EMPODAT database monitoring data and information on effects and hazardous properties for these substances. On the basis of this information, the substances are assigned to priority action categories by the NORMAN Prioritisation Working Group.

"Emerging substances" can be defined as substances that have been detected in the environment, but which are currently not included in routine monitoring programmes at EU level and whose fate, behaviour and (eco)toxicological effects are not well understood.

"Emerging pollutants" can be defined as pollutants that are currently not included in routine monitoring programmes at the European level and which may be candidates for future regulation, depending on research on their (eco)toxicity, potential health effects and public perception and on monitoring data regarding their occurrence in the various environmental compartments.

Examples from the LIST OF EMERGING SUBSTANCES are surfactants, flame retardants, pharmaceuticals and personal care products, gasoline additives and their degradation products, biocides, polar pesticides and their degradation products and various proven or suspected endocrine disrupting compounds (EDCs). The NORMAN experts regularly revise the list of emerging substances.

Appendix G: Cognitive Walkthrough Tasks

Principles

- Methods/Access to accomplish a task must be Findable, Easily accessible, and Usable

Simple Task Example

Step	Task
1	Open browser/prototype/application/system
2	Navigate to the landing page
3	Click login/sign-up button
4	Enter appropriate details in the designated fields

5	Complete sign-up/login

Specific Task Example 1

Step	Task
1	Locate the file you want to upload/process
2	Navigate the website
3	Navigate to appropriate page to upload/file
4	Take the guided steps to uploading file
5	Choose acquisition methods/take required steps post upload
6	View/Access file
7	Close Application

Specific Task Example 2

Step	Task
1	Navigate to Home Page
2	Navigate to User Profile
3	Check user details & do not close application

s