Design and Implementation of a Web-based Multi-user Data Analysis Environment for a Vancouver Island Drug-checking Initiative

by

Deepak Kumar
B.Tech., Aligarh Muslim University, 2011

A project submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

©Deepak Kumar, 2019
University of Victoria

All rights reserved. This project may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Supervisory Committee

Design and Implementation of a Web-based Multi-user Data Analysis Environment for a Vancouver Island Drug-checking Initiative

by

Deepak Kumar
B.Tech., Aligarh Muslim University, 2011

Supervisory Committee

Dr. Daniel M. German, Co-Supervisor
Department of Computer Science

Dr. Dennis Hore, Co-Supervisor
Department of Chemistry

Abstract

This project provides the detailed design and implementation details of a web-based multi-user data analysis and visualization environment for a Vancouver Island drug-checking initiative, along with the design of an accompanying website. It also includes the data analysis performed to answer some initial research questions.

The drug-checking project collects analytical chemical data and survey data during the process for detailed analysis and data mining. The web-based multi-user data analysis environment enables people interested in analysis of drug-checking data to effectively collaborate using this database and also allows them to access all the required scientific, data analysis tools and libraries.

The website is intended to communicate the results and findings to the public for harm reduction. It displays the aggregate results from the data and features interactive data visualization to educate users about component mixture analysis. It also provides information about the drug-checking program to different stakeholders including users of drug-checking service, chemists, social workers, harm reduction workers, pharmacists, and those interested in further developing the instrumental methods. It provides information about mission and goals of the project, services offered, research aspects of the project, technologies used and some frequently asked questions.

Contents

List of Figures

Acknowledgements

I would like to thank **Dr. Dennis Hore** for his continuous support and mentoring throughout the project and my supervisor **Dr. Daniel M German** for his valuable feedback and suggestions.

I am thankful to get constant encouragement, support and guidance from all the team members of Vancouver Island Drug Checking Program.

Also, I would like to thank my friends and family who encouraged me during this project.

Chapter 1

Introduction

1.1 Overview

This project is based on a community drug checking program which is offering a free and confidential drug checking service at harm reduction agencies in Victoria, British Columbia. The drug-checking program has various goals and has brought together people from diverse backgrounds (including social work, chemistry, the pharmaceutical industry, and computer science) to achieve them. It provides service users information about the composition of illicit substances. This program is aimed to offer drug checking reports via the project website and other formats to inform service users and the general public of test results and aggregate data. It is also aimed to integrate the drug-checking services within harm reduction and health services to contribute to reduced illicit drug overdose events and fatalities.

1.2 Drug-checking workflow

The drug-checking service users can test their drugs and be made aware of the substances present in the drugs. They can discuss the results with project staff, access the information and supplies for safer drug use. When a user comes for drug checking, they need to sign the consent form to agree on the use of drug checking service. The program offers the optional survey to the users that ask about them, the substances they brought, and their thoughts on drug-checking service. While the survey is

being completed, the technician performs chemical analytical tests using a variety of different instruments. When the tests are completed, the chemical data and results from the instruments are collected and stored in a database. The Harm reduction work then conveys the results (such as main active ingredients, contaminants, unexpected compounds, inert cutting agents) to the user.

The drug-checking project is using several instruments for testing drugs and one of the main goals is to evaluate and compare the utility of several instruments and technologies in terms of cost, efficiency, and portability. The chemical data and results are stored in a database which can be used to answer chemistry research questions, to perform data analysis or to find interesting trends using data visualization. The collected data can be used to prepare drug checking report to inform service users and the general public of test results and aggregate data. Collected data and evaluations will help to develop a framework for further implementation, integration, and scale-up of drug checking, linking harm reduction and health equity to the provision of drug checking.

1.3 Contributions

The overall goal of the project is to enable data analysis and visualization over the drug-checking database to find important trends and answers to chemical research questions and communicate these findings to the public for harm reduction. To achieve these, a web-based multi-user data analysis environment has been set up which allows people interested in drug-checking data analysis quickly access the data, and they can perform data analysis. It is also being used for prototyping the data visualizations which can be communicated to the public through a website. To test the potential and practical applications of data analysis environment, I have implemented a few data analysis examples to answer some initial crucial questions. A website has been developed which is being used to communicate the aggregate results and trends to the public with the help of data visualizations prototyped in the data analysis environment. The website also provides other information about Drug-checking Program like goals, services offered, research aspects and technologies used.

I have specifically worked on

- A multi-user, web-based environment for data analysis, data visualization and exploratory computing specific to drug-checking project has been created. To set up the environment, I have
  - Installed and configured jupyterhub on the server to meet the project's need. Configuration of Jupyerhub includes specifying which spawner we want to use to spawn Jupyter notebook servers, specifying authentication services, specifying how to store user's data and specifying the list of users.
  - Created a docker image that will run on Jupyterhub and includes all the required tools and libraries.
  - Created a docker network.
  - Setup third-party authentication.
  - Configured reverse proxy to make it available on the web and installed SSL certificate for secured communication.
- To provide the examples of data analysis on drug checking database using the data analysis environment mentioned above, I performed data analysis using the preliminary data in the drug-checking database to answer some of the initial questions. These questions are as follows
  - What fraction of samples tested positive for fentanyl on the strip test? Of those positive hits, what fraction tested positive using any other technique used for drug checking, and what was it?
  - What substances were detected (as a fraction/percentage of the number of samples measured)?
  - What other substances were detected in samples having fentanyl,and what are their percentages with respect to samples having fentanyl?
- To communicate the findings, I have developed a website for the project. For this I have
  - Gathered information from multiple stakeholders for the layout and presentation.

- Developed an interactive visualization as an educational tool for users to understand drug-checking instrumental data.
- Implemented some placeholder graphics that display aggregate data from the database in the form of numbers and graphs.
- Implemented a content management system that enables project members to modify web content on their own.

Chapter 2

Web-based Multi-user Data Analysis Environment

One of the primary goals of the drug-checking project is to collect the survey data and the data produced by the instruments during drug testing process and answer research questions by analyzing that data. One of the challenges in achieving that goal is to effectively collaborate with the drug-checking project members and people interested in data analysis on the drug-checking database with high processing power and the availability of scientific and data analysis tools.

To address this challenge, I have set up a multi-user data research environment that is hosted on a UVic server. This environment contains all the required tools needed to access and analyze the drug checking database. An authorized person can access it and perform the analysis over drug-checking database. This environment also includes tools to create interactive data visualizations. Hence, It can be used as test bench for various visualizations and prototyping the visualization for public. This environment is implemented with Jupyterhub which internally uses a single-user Jupyter notebook server [2] and docker [4].

2.1 Architecture and Implementation

In this section, the architecture and design features of the Jupyterhub [2] will be described. Below are the major components of the Jupyterhub architecture.

- A user interface component that allows users to interact with the data easily
- The authentication component to ensure that only authenticated users can access the data and tools present in the analysis environment on the server.
- Spawner component that enables Jupyterhub to spawn a Jupyter notebook server for each user.
- Docker Data volumes to store each user's data
- The prebuild docker images which contains all the required scientific and data analysis tools required for the project.
- The notebook templates that help the users get started and preparing their analytic workflow.

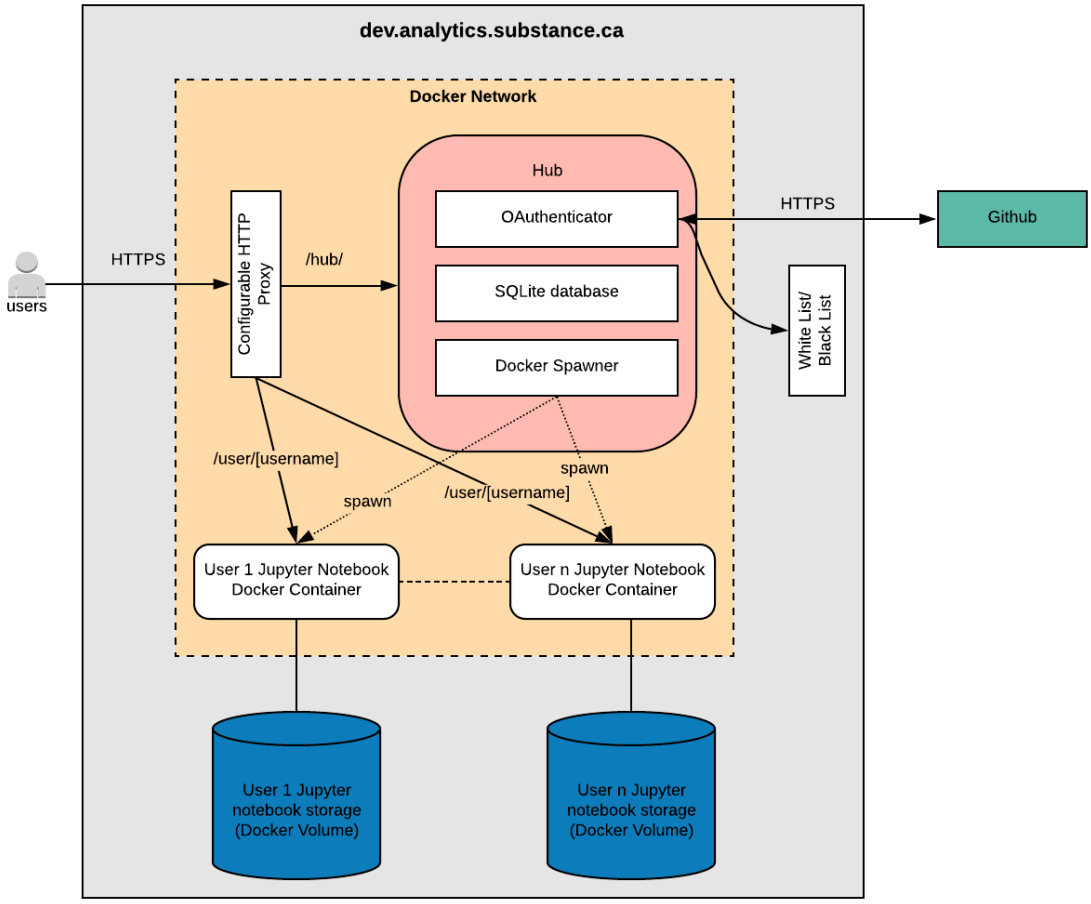

Figure 2.1: Multi-user data analysis environment architecture on hosting server

Authentication. The default authenticator used in Jupyterhub is PAM (Pluggable Authentication Module) which allows system users to authenticate with their username and passwords [10]. However, We have chosen OAuth third-party authentication method that allows users to authenticate using third-party accounts like Github, Google, Gitlab, OpenShift, etc. We are allowing users to authenticate via their Github account. To control the users dynamically, we maintain a whitelist and a blacklist of users that can be updated at the run time without any interruption to the service. The authenticator depends on the whitelist and blacklist to grant or deny access to the drug checking data analysis environment.

Spawner. Jupyterhub uses a spawner component to spawn a single notebook server for each user. There are different types of spawners (BatchSpawner, DockerSpawner, SimpleSpawner, WrapSpawner) having different properties. For our setup, we are using docker spawner which enables the Jupyterhub to spawn a single user notebook server within Docker containers. With the help of docker, each user will get his/her virtual environment that will be isolated from other users. With the help of a docker process for each user, it is easy for the administrator to manage work instances of users.

Docker Image. A docker image [4] has been created which contains all the required scientific , data visualization, and data analysis libraries. Python, R, and Julia have been installed inside the docker image. When this docker image runs in the docker container, all the tools become available for the users.

Docker Volumes. We are using docker volumes for persistent storage of the docker container data. Docker volumes store the data outside of the docker containers so even if container gets down for some reason, data will remain safe in the docker volume and will be available to the user when the container gets up and running. Each user has its docker volume associated with its docker container.

Docker Network. We have also created a user-defined docker network, and all the docker containers are running on this network. The advantage of this is that only containers on this network can access one another. The docker daemon runs an embedded DNS server to provide automatic service discovery for containers connected to user-defined networks. This allows us to access containers on the same network by name.

HTTPS Reverse Proxy. We are running Jupyerhub on the same machine as the website. The host machine is serving website content on HTTPS port 443, and the Nginx is being used as the public access point which means only Nginx can bind to the 443 port. To host the Jupyterhub on the port 443 as well, we have used a reverse proxy for Jupyerhub [2] and added it to the Nginx server which enables it to redirects the request to Jupyterhub.

2.2 Features and Capabilities

The Jupyter notebook offers a browser-based user interface that can support more than forty programming languages [6]. This project has included the support for three modern data analysis programming languages; Python, R, and Julia. The home page of Jupyterhub is a dashboard as shown in Figure 2.2. The purpose of this dashboard is to manage the files and notebooks. It displays all the file and notebooks present in the docker container along with the details like; size, last modified time and status of notebook(running or not). By clicking on the checkbox in-front of the file name, the user can choose to rename, duplicate, delete, view, shutdown or download the notebook. Inside the "Running" tab user can view the list of all the running terminal and notebooks. By clicking on "New" drop-down, the user can create a new directory, notebook, text file and terminal. The terminal provides the shell access to the single user notebook docker container which can be used by users to manage their notebooks, libraries and software. Also, any scientific or processing software that can be executed on Linux CLI can be used by each user in their docker containers with all the hardware and processing power available on the server. Apart from data analysis, it can also be used to create documents in HTML and markdown format which can be downloaded as an HTML, PDF, markdown, python, or LaTeXfile. It can also be used to create data visualization for various purpose including data analysis and prototyping visualizations intended for the public. I have included libraries such as `ipywidgets` and `ipython` that can be used to convert these visualizations into widgets as shown in Figure 2.3. In this example, I have wrapped around the matplotlib chart displaying the result of data analysis by the object of ipywidgets. Jupyter notebook allows user to generate HTML code to embed this visualization widgets into web

pages. To generate the HTML, user need to select “Embed Widgets” option under ”Widgets” menu of Jupyterhub. It will display a box containing the generated HTML and JavaScript as shown in Figure 2.4. This HTML code can be used to embed the visualization widgets in the website.

Figure 2.2: Jupyterhub dashboard to manage files and notebooks

Figure 2.3: Example of creating visualization widgets in Jupyterhub

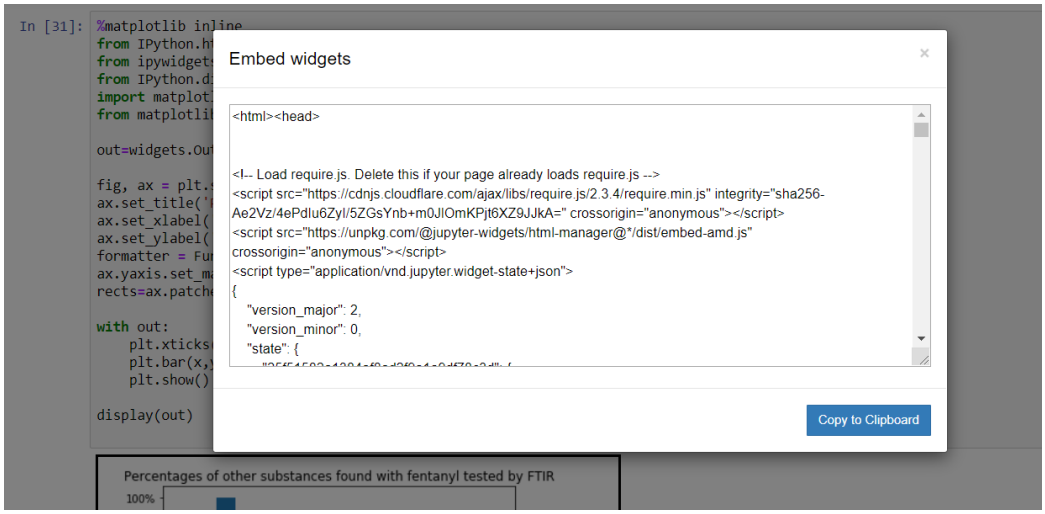

Figure 2.4: Example of generating HTML and JavaScript code to embed visualization widgets in webpage

Chapter 3

Data Analysis

In this section, I describe about some of the questions that are important for the drug checking project stakeholders, approaches to perform the data analysis, and some preliminary results from the analysis. As discussed in the Introduction, this project involves the collection of chemical data resulting from testing drugs using different analytical instruments. We are also collecting some optional survey data which is filled by service users when they use drug checking service. This data can be used to answers many questions useful for stakeholders.

3.1 Tools and Technologies

This section briefs about the tool, libraries and programming languages used for data analysis of the drug checking database

- **Python** [12] is one of the modern programming languages used for data analysis. It is easy to learn and supports many libraries for data analysis like pandas, numpy, etc.
- **Pandas** [11] is an open-source library which provides easy to use data structures and data analysis tools for Python programming language.
- **Matplotlib** [9] is a python 2D plotting library. Matplotlib can be used to create publication quality figures using Python scripts, the Jupyter notebook, web application servers, and four graphical user interface toolkits.

- **MySQL** MySQL is an open source relational database management system. It is being used to store and manage the drug-checking database.

- The **Jupyter notebook** [6] is an open-source application that allows to create and share documents that contain live code, equations, visualizations, and narrative text. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, and machine learning.

3.2 Approach

3.2.1 Data Collection

Data is collected at various drug checking sites in Victoria. The data collected at all the sites are stored in a central database server at UVic. With the help of Jupyterhub, the data at the central database server is accessed and data analysis is performed.

3.2.2 Identifying the Questions

To identify the questions important for this project, we had a discussion workshop which involved the technicians, chemists, social workers, and harm reduction workers. The discussion led us to many interesting questions some of which were identified to be answered in the long term and some in the short term. I performed data analysis to answer short term questions as follows.

1. **Question 1:** What fraction of samples tested positive for fentanyl on the strip test? Of those positive hits, what fraction tested positive using any other technique, and what technique was it?

2. **Question 2:** What substances were detected (as a fraction/percentage of the number of samples measured)?

3. **Question 3:** What other substances were detected in samples having fentanyl, and what are their percentages?

3.2.3 Understanding the drug checking database

It is essential to understand what data we are storing, how it is being stored in various tables and how different tables are related to each other. I worked together with the project database developer to understand the database tables and their structures. I also created a diagram to understand the relationship among different tables as shown in Figure 3.1.

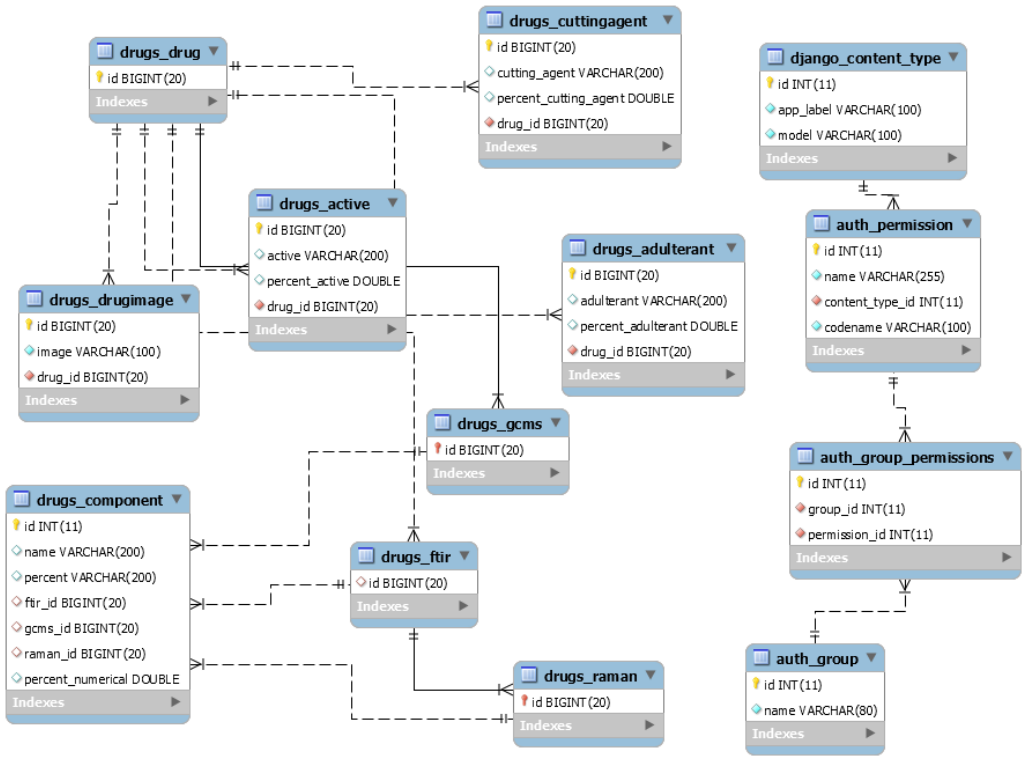

Figure 3.1: Diagram showing relations of tables in the drug-checking database

3.3 Questions

This section includes the three example questions outlined above, methods and results of data analysis on drug-checking database.

3.3.1 Question 1

Question. What fraction of samples tested positive for fentanyl on the strip test? Of those positive hits, what fraction tested positive using any other technique?

Method. Each drug sample is tested on two types of strip tests; FaStep and Rapid Response. The data is fetched using Complex nested SQL queries which can be breakdown into the following tasks

1. To get `drug_ids` of samples containing fentanyl from the `drugs_striptest` table.

2. To get the instruments' test ids (`ftir_id`, `gcms_id` and `raman_id`) from the instrument tables (`drugs_ftir`, `drugs_gcms` and `drugs_raman`) corresponding to the sample found in step 1.

3. To get the count of samples having fentanyl detected by instruments (FTIR, Raman, and GC-MS) from table `drugs_component`

4. Calculate the fractions and percentages as shown in Figure 3.2 using the outputs from step 1 and 3.

Below is the detailed description of steps performed for the analysis.

Fraction of samples tested positive forfentanyl on the strip test and of those positive hits,what fraction tested positive using any other technique

Below query is fetching the total number of samples tested using striptest.

In [23]:
```
fa_strip_total=pd.read_sql("select  count(drugs_drug.id) as Fa_count from drugs_drug inner join drugs_striptest on drugs_drug.id
fa_strip_total=fa_strip_total["Fa_count"][0]
```

drugs_drug table contains infomration about drug samples and drugs_striptest table contains the result of strip test corresponding to drug samples. In the below query, we are fetcching the count of fentanyl posive samples

In [31]:
```
df_fentanyl_positive=pd.read_sql("select  count(distinct drugs_drug.id) as fen_strip_fa from drugs_drug inner join "
                                 "drugs_striptest on drugs_drug.id = drugs_striptest.drug_id where drugs_striptest.result='{0}' "
                                 "and drugs_striptest.brand='{1}'".format("Positive","FaStep"), con=con)
fen_strip_fa=df_fentanyl_positive["fen_strip_fa"][0]
```

Using the result of above two queries i.e count of fentanyl positive samples and count of total samples , we are calculating the percentage of fentanyl positive samples tested using strip test

In [32]:
```
fen_strip_fa_percent=(fen_strip_fa/fa_strip_total)*100
```

In previous query, we found the drug ids of samples having fentanly tested by strip test. Now, We will find how many samples, out of the fentanyl positive samples detected by strip test, were detected fentanyl positive using FTIR. Below SQL query is used to find the same. Once we have count of fentanyl positive detected by FTIR, we can find out its percentage. Here, ftir_portion_fa contains the count of fentanyl positive samples test by FTIR for the fentanyl positive samples of FTIR which is stored in fen_strip_fa. So we have calculate the percentage using ftir_portion_fa and fen_strip_fa

In [34]:
```
ftir_portion_fa=pd.read_sql("select count(*) as ftir_portion_fa from drugs_component where ftir_id in "
                            "(select distinct id from drugs_ftir where drug_id in(select distinct drugs_drug.id from "
                            "drugs_drug inner join drugs_striptest on drugs_drug.id = drugs_striptest.drug_id where "
                            "drugs_striptest.result='Positive' and drugs_striptest.brand='FaStep')) and "
                            "drugs_component.name like '%fentanyl%'",con=con)
ftir_portion_fa=ftir_portion_fa["ftir_portion_fa"][0]
fen_ftir_fa_percent= (ftir_portion_fa/fen_strip_fa)*100
```

Similar query is used to find number of fentanyl positive, out of the fentanyl positive samples detected by strip test, detected using GCMS. fen_GCMS_fa_percent holds the percentage for GCMS.

In [37]:
```
GCMS_portion_fa=pd.read_sql("select count(*) as GCMS_portion_fa from drugs_component where gcms_id in "
                            "(select distinct id from drugs_gcms where drug_id in(select distinct drugs_drug.id from "
                            "drugs_drug inner join drugs_striptest on drugs_drug.id = drugs_striptest.drug_id where "
                            "drugs_striptest.result='Positive' and drugs_striptest.brand='FaStep')) and "
                            "drugs_component.name like '%Fen%'",con=con)
GCMS_portion_fa =GCMS_portion_fa["GCMS_portion_fa"][0]
fen_GCMS_fa_percent=(GCMS_portion_fa/fen_strip_fa)*100
```

Similar operation is perfromed for RAMAN.

In [39]:
```
raman_portion_fa=pd.read_sql("select count(*) as raman_portion_fa from drugs_component where raman_id in "
                             "(select distinct id from drugs_raman where drug_id in(select distinct drugs_drug.id from "
                             "drugs_drug inner join drugs_striptest on drugs_drug.id = drugs_striptest.drug_id where "
                             "drugs_striptest.result='Positive' and drugs_striptest.brand='FaStep')) and "
                             "drugs_component.name like '%fentanyl%'",con=con)
raman_portion_fa=raman_portion_fa["raman_portion_fa"][0]
fen_raman_fa_percent=(raman_portion_fa/fen_strip_fa)*100
```

Now we have all the required results as followed:

- fen_strip_fa_percent : Percentage of fentanyl positive samples tested using strip test
- fen_ftir_fa_percent : Percentage of fentanyl positive(out of fentanyl positive samples by strip test) sample tested by FTIR
- fen_GCMS_fa_percent: Percentage of fentanyl positive(out of fentanyl positive samples by strip test) sample tested by GCMS
- fen_raman_fa_percent: Percentage of fentanyl positive(out of fentanyl positive samples by strip test) sample tested by RAMAN

We are visualizing these results to have quick and better understanding using matplotlib

In [41]:
```
labels = ['Fentanyl Positive', 'Fentanyl Negative']
sizes = [fen_strip_fa_percent,100-fen_strip_fa_percent]

labels1 = ['Fentanyl Positive FTIR','Fentanyl Positive GCMS','Fentanyl Positive RAMAN',]
sizes1 = [fen_ftir_fa_percent,fen_GCMS_fa_percent,fen_raman_fa_percent]

fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=12.0
fig_size[1]=6.0
plt.rcParams["figure.figsize"]=fig_size

fig1, ax1 = plt.subplots(1,2)
ax1[0].pie(sizes, explode=(0.05,0), labels=labels, autopct='%1.2f%%',
        shadow=True, startangle=300)
ax1[0].axis('equal')
ax1[0].set_title("% of drug samples detected fentanyl by strip test")

formatter = FuncFormatter(lambda y, pos: "%d%%" % (y))
ax1[1].yaxis.set_major_formatter(formatter)

ax1[1].bar(labels1,sizes1,label="bar")
ax1[1].set_title("% of positive strip test detected fentanyl by other instruments")
plt.tight_layout()
plt.show()
```

Result. As shown in Figure 3.2, out of all drug samples tested by rapid response strip test 45% samples detected having fentanyl in it. Out of these fentanyl positive samples detected by rapid response strip test 33% detected fentanyl by FTIR, 40% detected fentanyl by GC-MS and 2% detected fentanyl by Raman.

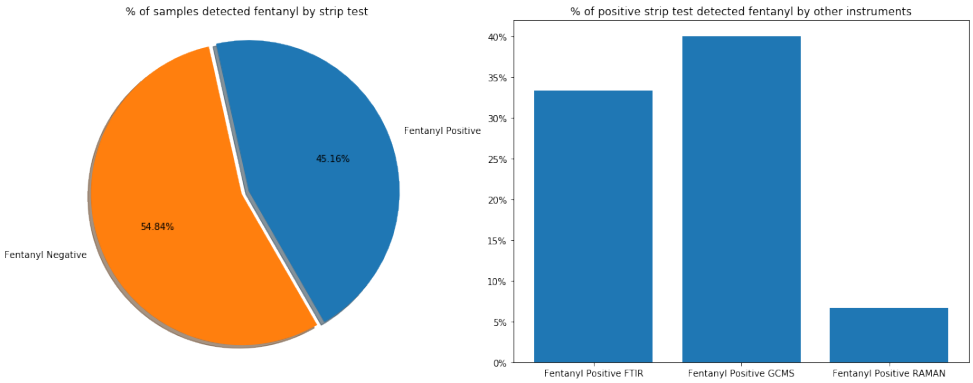

Figure 3.2: Percentage of fentanyl positive samples tested by strip test and percentage fentanyl positive by tested by FTIR, Raman, and GC-MS

3.3.2 Question 2

Question. What substances were detected as a fraction/percentage of the number of samples measured?

Method. There two tables (`drugs_component` and `drugs_active`) which contain the information of what substance detected for which sample. To do this analysis, the query returned the count of each detected substance. Then the percentage of substance calculated and visualized using bar chart as shown in Figure 3.3. Below is the detailed description of steps performed for the analysis.

What substances were detected as a frac-tion/percentage of the number of samples measured

drugs_active table contains the active component corresdoing to the drug samples. Below SQL query is fetching all count of all the active components. Here "y" contains the list of count of active components and "x" contains corresponding active components.

In [43]:
```
active_components= pd.read_sql('select count(*) as num,active from drugs_active group by active', con=con)
y=list(active_components['num'])
x=list(active_components['active'])
```

Now to find the percentage of each component we need to find the total number of samples whis is being fetched using below query

In [44]:
```
total_samples=pd.read_sql('select count(distinct drug_id) as total_samples from drugs_active', con=con)
total_samples=total_samples["total_samples"][0]
```

Percentage calculation

In [45]:
```
y_percent=[float("{:.2f}".format((x/total_samples)*100)) for x in y]
```

So y_percent holds the percentages of each component and x holds the corresponding component names. So we can visualize the result using a bar chart as following.

```
In [48]: fig_size = plt.rcParams["figure.figsize"]
         fig_size[0]=20.0
         fig_size[1]=6.0
         plt.rcParams["figure.figsize"]=fig_size
         fig, ax = plt.subplots()
         ax.set_title('Percentages of substances found in total number of samples')
         ax.set_xlabel('Substaces')
         ax.set_ylabel('Percentages')
         formatter = FuncFormatter(lambda y, pos: "%d%%" % (y))
         ax.yaxis.set_major_formatter(formatter)

         rects=ax.patches
         plt.xticks(rotation=90)
         plt.bar(x,y_percent, width=0.5)

         for rect, label in zip(rects, y_percent):
             height = rect.get_height()
             ax.text(rect.get_x() + rect.get_width() / 2, height , str(label)+"%",
                     ha='center', va='bottom')
```

Result. The histogram in Figure 3.3 shows the results of analysis. There are many substances detected as active ingredient of drug samples. Figure 3.3 shows the percentage of each substance detected from all tested drug samples. We can see that fentanyl has highest percentage among all the substances.

fractions.png

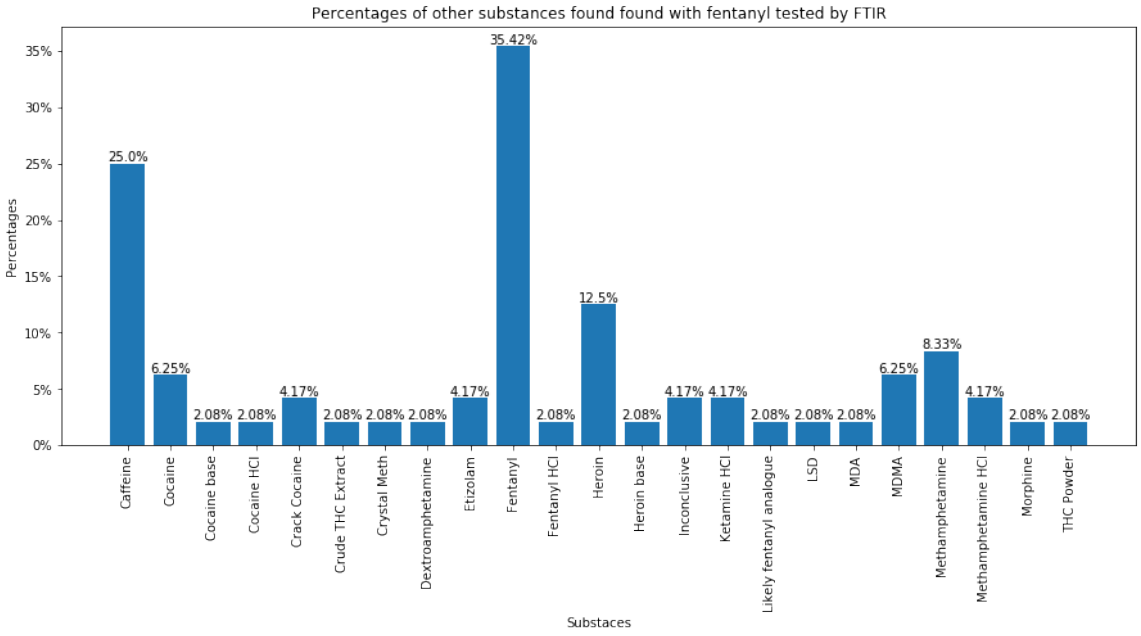

Figure 3.3: Substances detected as the percentage of total tested samples.

3.3.3 Question 3

Question. What other substances were detected in samples having fentanyl and what are their percentages?

Method. This analysis is done for the instruments, FTIR and GC-MS. Raman is not considered for this analysis because it has detected fentanyl in very less number of samples which is not sufficient for this analysis. Following are the steps performed.

1. Fetched and counted the `FTIR_ids` of the number of samples having fentanyl tested by FTIR.
2. Counted all the other substances present in the samples having fentanyl with the help of output from step 1
3. Calculated the percentages.
4. Visualized the result with the help of histogram as shown in Figure 3.4

The above steps are repeated to analyze with respect to GC-MS. The detailed description of the analysis is presented below.

What other substances were detected in samples having fentanyl and what are their percentages

Import all the required libraries as follows:

- matplotlib library to create data visualization charts
- pymysql library to make connection with the drug-checking database on server
- pandas library to for data processing

In [2]:
```
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
import pymysql
import pandas as pd
```

Create database connection. Below code returns a connection object which can be used to execute sql queries

In [3]:
```
con=pymysql.connect(host='mysql-test',db='substance_docker', user='jupyter_user', passwd='substance_135&')
```

Below, we are executing a SQL query using read_sql method of pandas. Read_sql takes two parameters; 1. SQL query and 2. Connection object. The SQL query is a nested query. First query is returning FTIR_ids having Fentanyl from drugs_component table. Second query is returning count of all the substances other than fentanyl correspoding to the FTIR_ids found in first query.

In [7]:
```
ftir_component=pd.read_sql("select count(substring(name,3,200)) as num,substring(name,3,200) as substance from "
                           "drugs_component where ftir_id <>'Nan' and ftir_id in (select ftir_id  from drugs_component where "
                           "ftir_id <>'Nan' and name like '%fentan%') and name not like '%fentan%' "
                           "group by substring(name,3,200) ", con=con)
y=list(ftir_component['num'])
x=list(ftir_component['substance'])
```

Next, We are fetching the count of samples having fentanyl tested by FTIR. and than calculating the percentage of substances found in the sampels having fentanyl using the results found in last query.

In [8]:
```
ftir_fen_total=pd.read_sql("select count(*) as ftir_fen_total  from drugs_component where ftir_id <>'Nan' "
                           "and name like '%fentan%'", con=con)
ftir_fen_total=ftir_fen_total["ftir_fen_total"][0]
y_percent=[float("{:.2f}".format((x/ftir_fen_total)*100)) for x in y]
```

"x" holds the name of substances and "y_percent" holds the corresponding percentage value. Using these values, we are creating a bar chart in matplotlib.

In [9]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=12.0
fig_size[1]=6.0
plt.rcParams["figure.figsize"]=fig_size
fig, ax = plt.subplots()
ax.set_title('Percentages of other substances found with fentanyl tested by FTIR')
ax.set_xlabel('Substaces')
ax.set_ylabel('Percentages')
formatter = FuncFormatter(lambda y, pos: "%d%%" % (y))
ax.yaxis.set_major_formatter(formatter)
rects=ax.patches
plt.xticks(rotation=90)
plt.bar(x,y_percent, width=0.5)

for rect, label in zip(rects, y_percent):

    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height , str(label)+"%",
            ha='center', va='bottom')
```

The above steps are being repeated for GCMS. The SQL query is a nested query. First query is returning GCMS_ids having Fentanyl from drugs_component table. Second query is returning count of all the substances other than fentanyl correspoding to the GCMS_ids found in first query.

In [17]:
```
GCMS_component=pd.read_sql("select count(substring(name,3,200)) as num,substring(name,3,200) as substance from "
                           "drugs_component where gcms_id in (select gcms_id  from drugs_component where gcms_id <>'Nan' "
                           "and name like '%tanyl%') and name not like '%tanyl%' group by substring(name,3,200)", con=con)
y=list(ftir_component['num'])
x=list(ftir_component['substance'])
```

Fetching the count of samples having fentanyl tested by GCMS and than calculating the percentage of substances found in the sampels having fentanyl using the results found in last query.

In [14]:
```
GCMS_fen_total=pd.read_sql("select count(*) as GCMS_fen_total  from drugs_component where gcms_id <>'Nan' and name like '%tanyl%
GCMS_fen_total=GCMS_fen_total["GCMS_fen_total"][0]
y_percent=[float("{:.2f}".format((x/GCMS_fen_total)*100)) for x in y]
```

"x" holds the name of substances and "y_percent" holds the corresponding percentage value. Using these values, we are creating a bar chart in matplotlib.

In [19]:
```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0]=8.0
fig_size[1]=6.0
plt.rcParams["figure.figsize"]=fig_size
fig, ax = plt.subplots()
ax.set_title('Percentages of other substances found with fentanyl tested by FTIR')
ax.set_xlabel('Substaces')
ax.set_ylabel('Percentages')
formatter = FuncFormatter(lambda y, pos: "%d%%" % (y))
ax.yaxis.set_major_formatter(formatter)
rects=ax.patches
plt.xticks(rotation=90)
plt.bar(x,y_percent,width=0.5)

for rect, label in zip(rects, y_percent):

    height = rect.get_height()
    ax.text(rect.get_x() + rect.get_width() / 2, height , str(label)+"%",
            ha='center', va='bottom')
```

Results. Results from the analysis concerning FTIR is shown in Figure 3.4. We can see that all the samples containing fentanyl also contains caffeine. The second most found substance is heroin, which is 17%. Inositol is in third place with 8%. Apart from these, FTIR also detected (Carbohydrate, Cocaine Base, Xylitol, Methamphetamine) with all having the same percentage, 3%.

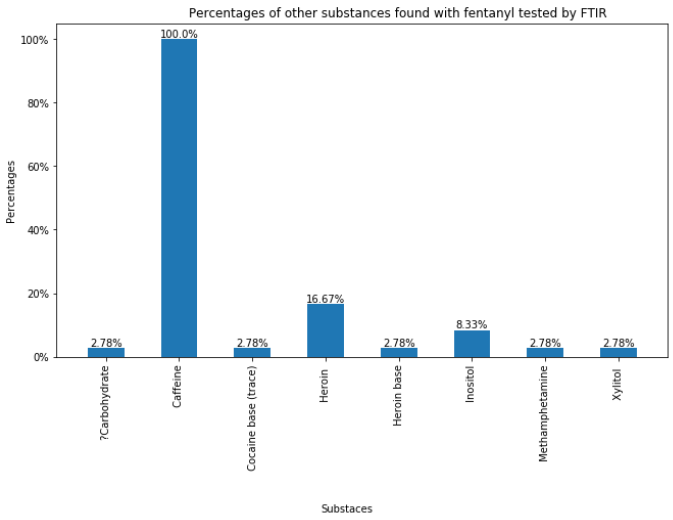

Figure 3.4: Percentage of the substances found in the samples having fentanyl tested by FTIR

Analysis corresponding to GC-MS also resulted in similar trends as shown in Figure 3.5. Caffeine has the highest percentage; 100%, Heroin is the second highest; 43% and Inositol is 14%. GC-MS did not detect any other substances apart from caffeine, heroin, and inositol.

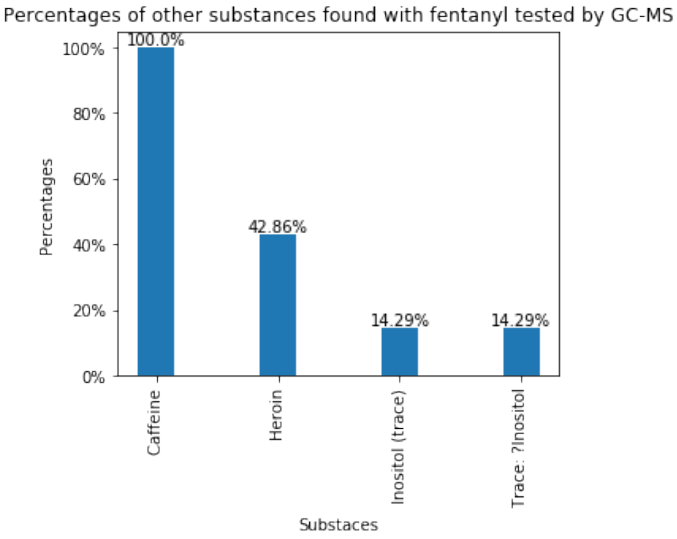

Figure 3.5: Percentage of the substances found in the samples having fentanyl tested by GC-MS

Chapter 4

Website

The drug checking project requires a public-facing website to provide various information to different groups like users of drug checking service, chemists, scientists, harm reduction and social workers. The website provides information like the mission of the project, what services are offered, research aspect of project, technologies, and instruments used in the project and partners. It also contains some data visualizations. One of them is a data visualization of substance mixture analysis which is an educational tool to educate users about how these black boxes work from chemical standpoint. Another data visualization section shows some trends in drugs usage and displays some aggregate data explained in more detail in later section of the report. This is a dynamic website where the content displayed on it is stored at a database. It also includes an admin panel which is being used for content management which means staff members can change the content without changing in the code.

4.1 Technologies and Tools

The tools and technologies I have used for the development of the website for the drug-checking project are

- **HTML, CSS, SCSS, JavaScript, JQuery, Bootstrap Framework.** These technologies are being used for front end development of the website.
- The **Django Framework** is a python based framework. It is being used to implement the backend of the website.

- **Chart.js**, a JavaScript library mainly used for data visualization
- **Chrome development tools** are used to perform debugging and testing

4.2 Design

Designing is one of the most critical parts to make the website appealing, understandable and usable by different groups of people [16]. A significant amount of time has been spent on understanding and working on the design. During the process, I have talked to the stakeholders involved in the project to know their expectations and to get their feedback. Hence, the design of the website went through various revisions before getting the final one. Below are the different parameters that have been considered while designing.

Simplicity. For usability and user experience reasons, it is essential to keep the design of website simple. Adding unnecessary design elements (for example, elements that serve no functional purpose) to the website will only make it difficult for visitors to accomplish their purpose. To keep the website simple, There have been taken specific measures like not to use too many colors, not many font styles, not too much unnecessary content or graphical elements.

Navigability. Having intuitive navigation is crucial to make sure that visitors can find what they are looking for. The website is structured using a single-page application layout [14] where the user can see all the content just by scrolling through the entire page . However, A menu bar is provided which always sticks at the top so that visitors can immediately navigate to any section on the website without scrolling.

Responsiveness. Responsive design [15] is an approach to the web design that makes the layout and content of the website respond to the variety of devices and screen sizes [8]. Nowadays when usage of smart devices (cell phones, tablets) are more than desktops/ laptops, it is required that the website should render correctly on these devices. The design of the drug checking website is also responsive. It automatically adjusts and changes the layout for different screen sizes. There are three layouts categories for the website; 1. Computers 2. Tablets and 3. Mobile phones, but the design may further adjust based on screen size within each of these categories. The responsive design also eliminates the need for separate app development for mobile

or tablet devices.

Figure 4.1: Responsive design of drug checking website

4.3 Architecture and Implementation

This section explains the architecture and implementation details of the drug-checking website. Implementation is based on the Django framework [5]. There are various reason to choose Django framework. The first reason is that it forces to organize and implement the project in a structured and modular way. It has the ability to scale quickly and flexibly from small web application to significant web application. It also provides an admin panel by default which we are using to manage the content of our website. We also want our website to rank high in the search result, and Django helps in that by generating readable URLs and links. The architecture of the website is shown in Figure 4.2.

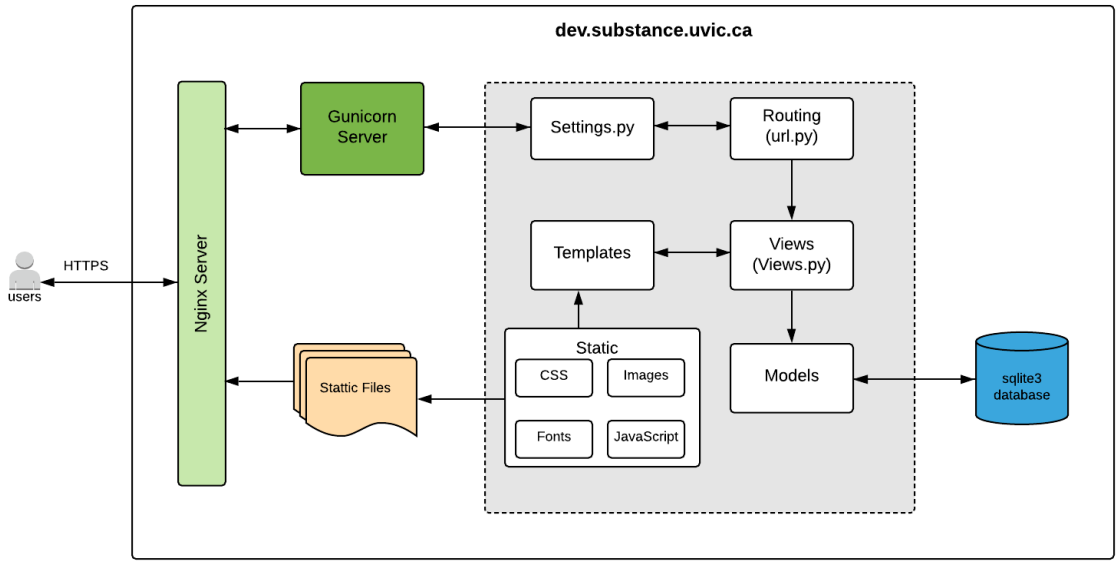

Figure 4.2: Drug checking website architecture on the hosting server

Below is the description of each element present in the website architecture

Setting.py contains all the setting related to the website. It contains database configurations, the location of static files, registered application and allowed hosts.

Routing. Django lets us create our URL. To create the URL, we created a python class called `url.py` and specified the URL along with the view associated with it. We have created a few URLs like "domain.ca/substance/" for the main page, "domain.ca/substance/faq" for FAQ page and "domain.ca/admin/" for admin panel.

Views. Views.py is a python module that contains all the view functions. View functions take a web request as input and return a web response. This response can be an HTML content of web page or redirect, or page not found error, or JSON response. When a URL is hit which is present in url.py, its corresponding view function is called from views.py.

Templates. Templates module contains all the HTML files. Whenever a view function wants to return the HTML content as web response, it calls the render method which takes three input parameters: the initial request, the path to HTML inside template module, and the dictionary of parameters.

Static. This website needs to serve some CSS, fonts, JavaScript and image files. In Django, these files are called static files, and they reside inside a directory called

`Static` . However, the static directory is used only during the development phase. In production, all the static file are transferred to another directory which has been discussed in the next section.

Static Files. When we deploy the site on the live server, We need to run the collectstatic command which collects all the static files of the website and place them in a separate directory which we call `STATIC_ROOT`. We also change the location of the static file in settings.py on the server.

Models. The Django website can access and manage the data using python objects. These objects are referred to as models. In models.py, We have defined all the database tables and structure of data to be stored in terms of classes and their properties and methods. Each class maps to a single database table.

Sqlite3 database. The content of the website is stored in an SQLite3 database. SQLite3 is a small, fast, high-reliability and self-contained database engine [13].

Nginx Server. Nginx is a web server. The reason for choosing the Nginx server is to take advantage of its high-performance connection handling mechanisms [7]. It faces the outside world and can serve static files(like images, CSS, media) directly. However, It can not directly interact with the Django application because Django accepts only python calls. For this purpose, Gunicorn server is used.

Gunicorn Server. Gunicorn (Green Unicorn) is a Python WSGI HTTP Server for UNIX [1]. It is being used to run the Django application. Since Django is a python framework, it can process only python request. Gunicorn takes the HTTP request and convert them into python calls and pass them to the application.

4.3.1 Dynamic content management

The content displayed on the website can be managed without the need for any change inside the code [3]. All the content are stored inside tables in the SQLite3 database which can be managed by an admin panel [5] provided by Django as shown in Figure 4.3. For example, If we want to add more questions under the FAQ section, then we need to create a new entry inside the FAQ table in the database, and the newly added question will be displayed on the website. In most cases, the change in the existing content of the website does not require a developer; it can be done by staff member having access to the admin portal. We should restrict the access of the

admin portal to only a few authorized people for security purpose.

Figure 4.3: Admin panel to manage website content

We can also create user accounts of the admin portal with different access level over different tables as shown in Figure 4.4. For this website, We have created user accounts with two access levels. One access level is a superuser which is allowed to add, delete and change the content of all the tables. Another access level is staff which can add, delete or change the content of the tables that contain less sensitive data.

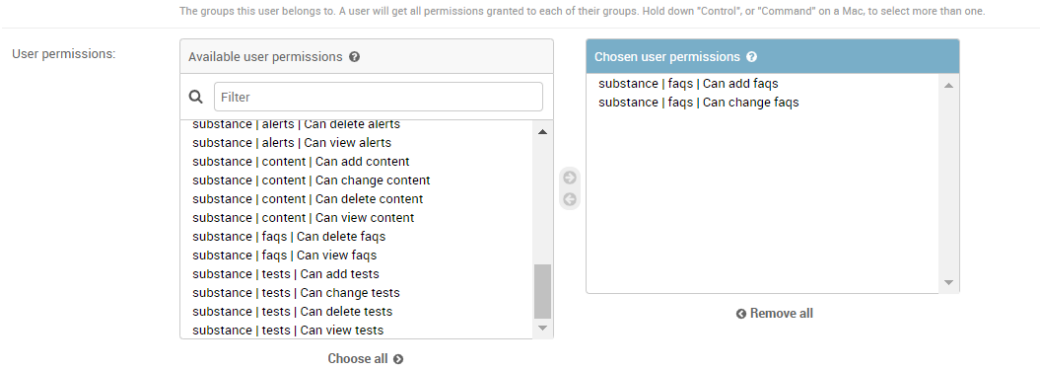

Figure 4.4: Admin panel to grant permissions on tables

4.3.2 Server Configuration and Website Deployment

This website is deployed and hosted on a UVic server provided for the drug-checking project. Below are the steps followed to deploy.

- Created a python virtual environment and installed all the project dependencies inside it.
- Installed and configured the Ngnix server.
- Installed and configured the Gunicorn application server. It is being used to run Django applications. It translates client requests in HTTP to Python calls that the application can process and return the response. The Gunicorn server creates a Unix socket and uses it to interact with the Nginx server.
- Transferred the project files to the UVic server and configured `ALLOWED_HOSTS` and `STATIC_ROOT` parameters inside settings.py.
- Installed and configured SSL certificate to run website over HTTPS protocol which allows only secure connections from a web server to a browser. We have taken the SSL certificate from "Let's Encrypt", a non-profit certificate authority run by the Internet Security Research Group (ISRG).

4.4 Drug-Checking Website Walk Through

This section contains a brief description of various sections present on the website.

Trends. This section reports some aggregate data related to drug usage. It also displays recent trends with the help of data visualization. The visualization is implemented using `chart.js`.

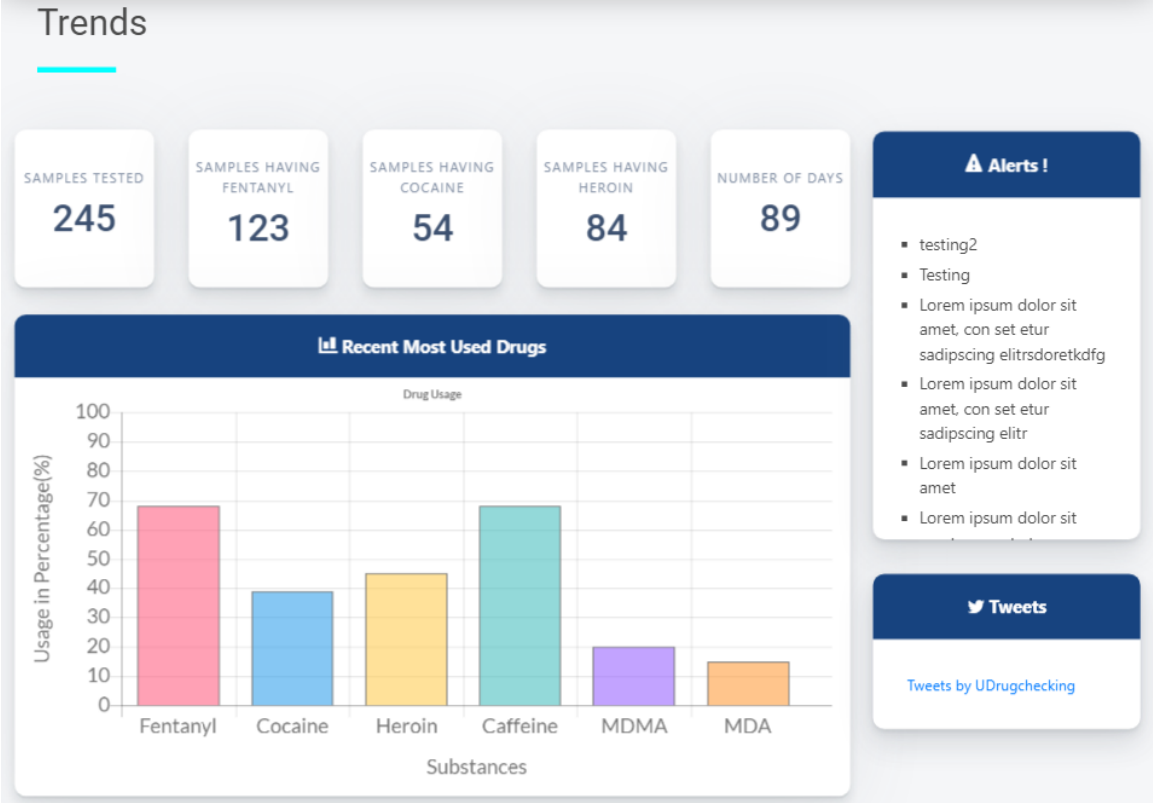

Figure 4.5: User Interface of Trends section on website

Another option is to use visualizations developed directly in the jupyterhub notebook using python and matplotlib to generate png files. Those scripts can be run on the server, and pngs included directly in the website. In this way, non-interactive but live content can be generated automatically.

Services. This section intends to tells the visitors about what services the drug checking project is offering as shown Figure 4.6. It includes the schedule of service at different sites and What user should expect/not expect out of the drug-checking service.

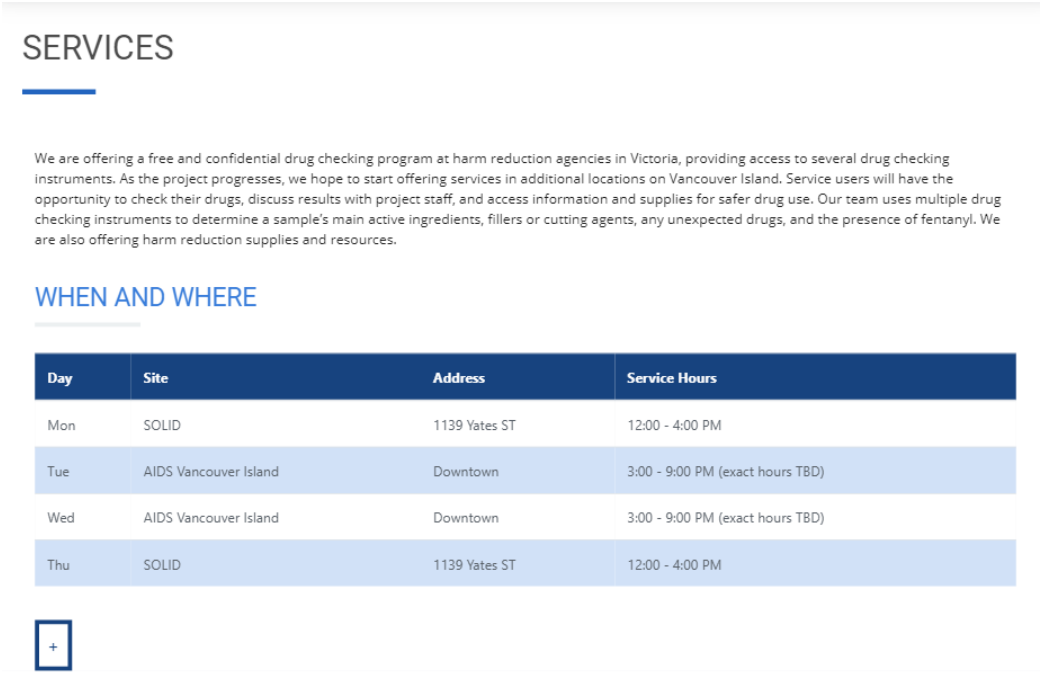

Figure 4.6: User Interface of Services section on website

Research. This section informs the visitors about the project goals and research aspect of the project. It also informs users about the surveys/interviews conducted at drug checking sites and motive behind this.

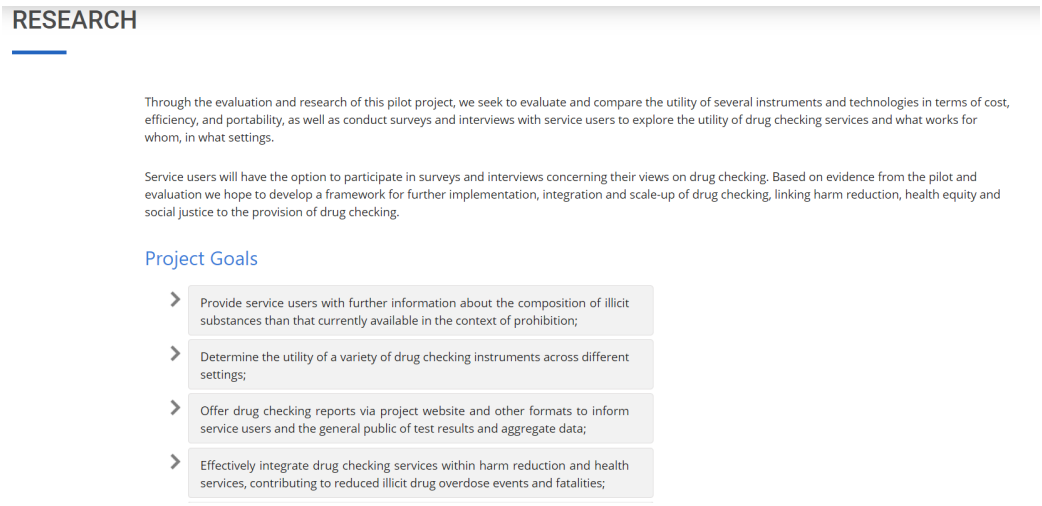

Figure 4.7: User Interface of Research section on website

Technologies. This section contains information about the instruments being used in the drug checking project. It features brief hardware descriptions of the

key instrumental methods: infrared absorption spectroscopy, Raman scattering spectroscopy, gas chromatography coupled with mass spectrometry (GC-MS), and antibody-based test strips. The level of detail provided is intended to be suitable for a general audience with scientific interest, but no background in this area.

It also contains a visualization that shows the raw spectral data for these different instruments, considering a mixture of cocaine, heroin, and fentanyl. This visualization allows the user to explore by selecting or deselecting the components of the mixture, and dynamically observe the changes in the instrument response. The gas chromatogram shows the peaks of substances as they are eluted; the user can click on any of the peaks to see the corresponding mass spectrum. The aim is to present this visualization as an educational tool for users so that they may have a better understanding of how these black boxes work, and also have an appreciation for the complexity of mixture analysis from a chemical standpoint.

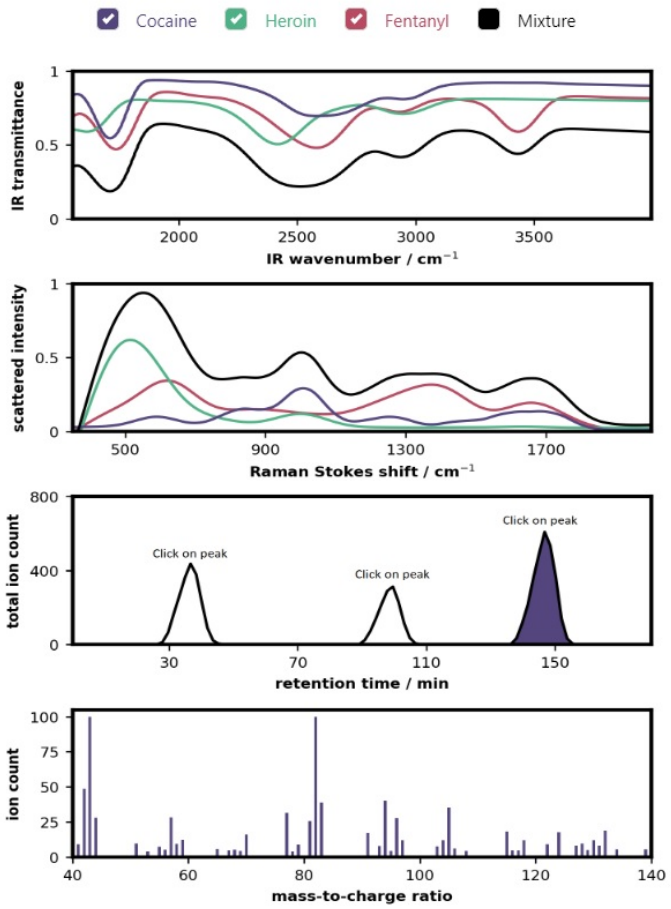

Figure 4.8: Visualization for substance mixture analysis

Who We Are. This section tells about the collaborators and partners of the drug checking project. It includes the funders, academic partners, social services, knowledge users and industry partners.

FAQ. This section contains important and frequently asked question related to the project. The answers are placed within the collapsible boxes and visitors can click on the question to view the answers. In this way, the user can view the maximum number of questions at a time without much scrolling.

Chapter 5

5.1 Conclusion

The Vancouver Island drug checking project has just finished its development phase and has already collected data on more than 500 drug samples. This project has allowed people to collaborate on the drug-checking database and enabled them to do data analysis and visualization for various purpose like, to find interesting trends, to answer research questions. The data analysis part of this project presented the examples of this. The website is communicating the findings to the public along with drug-checking program related other information. The description about each of these parts, followed by implementation details have been described in this report.

5.2 Future Work

The future work for this project may include the implementation of a user interface to add or remove users from the whitelist or the blacklist. Currently, the administrator has to change it in the configuration file.

As mentioned earlier in section 3.2.2, many questions need to be answered in the long term using data analysis. Some of these questions are as follows-

- How the trends detected in short term analysis changes over time
- Concentration variations of active ingredients
- Analysis of survey data
- Does the detection of specific substance changes the drug user's decision

References

[1] Gunicorn documentation. https://gunicorn.org/#docs. Retrieved 2019-02-20.

[2] Jupyterhub documentation. https://jupyterhub.readthedocs.io/en/stable/.

[3] M. Dobecki and W. Zabierowski. Web-based content management system. In *2010 International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, pages 177–179, Feb 2010.

[4] Docker. Docker documentation. https://docs.docker.com/.

[5] Django Software Foundation. Django documentation. https://docs.djangoproject.com/en/2.1/.

[6] Jupyter.org. Jupyter notebook documentation. https://jupyter.org/documentation.

[7] Douglas Kunda, Sipiwe Chihana, and Sinyinda Muwanei. Web server performance of apache and nginx: A systematic literature review. *2017 Computer Engineering and Intelligent Systems (IISTE)*, pages 43–52, 11 2017.

[8] Jeonghyun Lee, Imsu Lee, Iseul Kwon, Hyejin Yun, Jongwon Lee, Mansung Jung, and Hyenki Kim. Responsive web design according to the resolution. *2015 8th International Conference on u- and e-Service, Science and Technology (UNESST)*, pages 1–5, 2015.

[9] Matplotlib.org. Matplotlib 2.2.3 documentation. https://matplotlib.org/2.2.3/contents.html.

[10] Andrew G. Morgan. Pluggable authentication modules for linux. `https://www.linuxjournal.com/article/2120`, December 1997.

[11] Pydata.org. pandas: powerful python data analysis toolkit. `https://pandas.pydata.org/pandas-docs/stable/`.

[12] Python.org. Python3 documentation. `https://docs.python.org/3/`.

[13] sqlite.org. Sqlite documentation. `https://www.sqlite.org/docs.html`.

[14] J. Tesarik, L. Dolezal, and C. Kollmann. User interface design practices in simple single page web applications. In *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT)*, pages 223–228, Aug 2008.

[15] Bin Zhou Yujian Jiang Wei Jiang, Meng Zhang and Yingwei Zhang. Responsive web design mode and application. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*, pages 1303–1306, Sep. 2014.

[16] Y. Zhang, L. Wang, and W. Luo. Research on web design style based on digital technology. In *2017 9th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)*, pages 443–446, Jan 2017.