The University of Melbourne

Master of Data Science

May, 2022

COMP90024: Cluster and Cloud Computing

Melbourne Liveability Index

Student Names:	*University IDs*
Abhishek Dagar	1338713
Nithin Mathew	1328669
Prateek Bansal	1311270
Ritwik Giri	1301272
Vibhuti Rajpal	1305409

ABSTRACT

In the fast-paced city of Melbourne, everyone wants to get ahead of each other. Competitions have become so fierce that people don't spend enough time reflecting upon themselves. Most people do not have enough time to make a calculative and informed decision before deciding on the place to reside or make an investment. There are various factors that affect the living condition of an area. The application aims to enable the user to make an intelligent housing or investment decision before selecting not only his dream house but also his dream community.

The project utilizes the data from AURIN (Australian Urban Research Infrastructure Network), Twitter and Better education to evaluate the livability index of different regions in and around the Greater Melbourne region. The calculated liveability score is guided by the information related to housing, schooling, healthcare, environment, and entertainment conditions of a particular region. In the end, AURIN and Twitter data were correlated to improve our understanding of the overall liveability of Greater Melbourne region.

Contents

Section 1

Overview

Melbourne is considered to be the most liveable city by the Economist Intelligence Unit[1]. However, the suburbs in and around the Greater Melbourne region are not included in the survey. Our application uses data from Twitter, AURIN and other external sources and aims to drill down to the postcode level to calculate the liveability of the individual postcodes and hence estimate the liveability of the entire Greater Melbourne region.

1.1 Project Lifecycle

The project was divided into three major categories:

1.1.1 Use case selection

The team scrutinized different ideas related to different factors that affect the liveability condition of the areas in and around Greater Melbourne. The below factors are considered due to their inevitable importance in the day-to-day lives of a normal individual.

1. **Housing**
   (a) The Rent Affordability Index (RAI) data is taken from the AURIN database
   (b) It takes into account the household incomes
   (c) Higher the index score higher the affordability
2. **Schooling**
   (a) Data from Better Education1 is used as a source of truth
   (b) Victoria government publishes the score based on the SA1 (Statistical Area) scores
   (c) Higher the SES score, better the school community
3. **Healthcare**
   (a) Data about the number of hospitals and total beds in the postcodes are extracted from AURIN and are used to analyse the liveability index.
4. **Environment**
   (a) Data about the number of polluting activities are taken from AURIN and used to analyse the liveability index.

5. **Entertainment**
    (a) Number of Cafe, bars and sports facilities
    (b) Data is collected from the AURIN database

1.1.2 Development Strategy

The project was divided into smaller chunks and different implementation stages were finalized.

1. **Data Analysis**
    (a) Data for different scenarios are analysed and different features are selected accordingly
    (b) AURIN data tables are identified for each scenario
2. **Data Fetch**
    (a) AURIN APIs for the use case scenarios are identified
    (b) Twitter APIs are used to fetch both real-time and past tweets
3. **Database Design**
    (a) CouchDB is used to store the data from Twitter and AURIN APIs
    (b) CouchDB is set up in 4 instances with 3 shards each
4. **AURIN Harvester**
    (a) Data from AURIN APIs is pushed to the CouchDB
    (b) Hospitals data is taken by downloading the JSON file, as AURIN does not provide API for the same
5. **Twitter Harvester**
    (a) Twitter search API is used to extract past tweets according to the category
    (b) Real-time tweet streaming is implemented
    (c) Real-time tweet streaming is implemented
    (d) Relevant tweets are harvested based on the keyword search

6. **UI Design**

   (a) React is used to design the front end

   (b) The leaflet is used to display the location of the postcodes

   (c) Material UI is used to display real-time tweet streams

1.1.3 Implementation Strategy

Each team member contributed by taking either one or more than one component from the development strategy.

1. **Instance creation**

   (a) The instance is created and configured using ANSIBLE in 4 nodes of the MRC project

   (b) The number of liveability categories is divided among the number of instances

2. **Database Creation**

   (a) To improve the query functionality, CouchDB is configured in each of the 4 nodes

   (b) Our replication strategy enabled us to create a fault-tolerant database

   (c) Views are created to allow UI to fetch documents from CouchDB

3. **Docker Build and Run**

   (a) Each component (Twitter Harvestor, AURIN Harvestor, flask APIs and UI) is dockerized and configured in the MRC instances

Section 2

Team Contributions

A private gitlab repository was created to track the progress and issues of the project. It helped in the version control.

Gitlab URL: https://gitlab.unimelb.edu.au/rgiri/assignment2-comp90024

2.1 Individual Contributions

1. **Abhishek Dagar**: My role was to design, develop and deploy the UI interface using React JS on a docker environment. I collaborated with Ritwik and Vibhuti to implement couch Db views and Flask APIs which were required to populate different UI components and to make them more insight-driven.

2. **Nithin Mathew**: My role in the project was to design the system architecture, infrastructure and automate deployment. I worked on the Ansible scripts to automate the creation and configuration of instances and deployments of all components of the application. This task is described in further detail in the Ansible deployment section of the report. I also supported the rest of the team with solving issues, designing solution approaches and use cases. I also worked on the bulk_twitter_harvester with Vibhuti using snscrape to retrieve tweets older than 7 days that could not be retrieved using the search tweets API. My responsibility also included dockerizing the application components built by my teammates.

3. **Prateek Bansal**:My role was to develop and refine the tweet harvester. Moreover, I worked closely on defining the data views in CouchDB and creating a flask API for the project. I, along with my other teammates, worked on the key analysis and researched the AURIN data to get a better understanding to build up a story. Also, I worked on drafting the structure for the Project report.

4. **Ritwik Giri**:My role was to design and develop the AURIN harvester and to identify the multiple-use case scenarios. I worked with Vibhuti to develop the Twitter harvester and added the tweet stream functionality to the Twitter harvester. I worked with Nithin on utilizing snscrape to retrieve historic tweets. I worked along with Nithin and Vibhuti and created use case scenarios and data inferences between AURIN and Twitter Data. I used my local machine to perform couchDB back up, which was used during the redeployment process. I also along with Nithin, managed the project lifecycle. I was also responsible for documenting the project.

5. **Vibhuti Rajpal**:My role in this assignment was to design develop and refine the twitter harvester. Ritwik helped me develop the twitter streaming API. I was also responsible to add twitter data into couchdb and create views and flask APIs to be consumed by react UI. I also collaborated with Abhishek to develop the react application.

Section 3

System Architecture

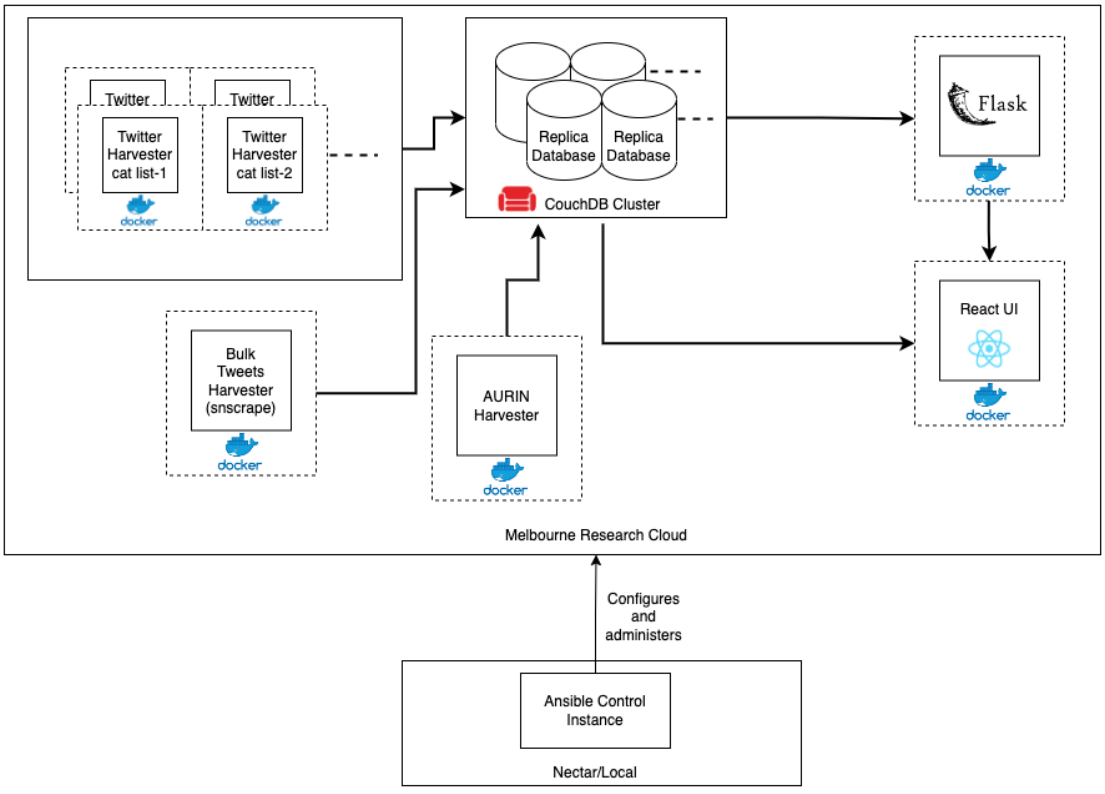

Figure 3.1: System Architecture Design

The system architecture consists of 5 key components. It utilizes the MRC cloud resources to deploy the "Melbourne Liveability Index" application.

- ANSIBLE
- CouchDB
- Twitter Harvester
- AURIN Harvester
- Flask
- React UI

Each of these components are discussed in detail in the further sections

Section 4

ANSIBLE Deployment

Ansible is a widely popular open-source software provisioning, configuration management, and deployment tool that enables **infrastructure as code** paradigm. It runs on most unix-like systems and can be used to configure/manage both unix-like and windows machines. It follows its own declarative syntax to describe configuration instructions. Ansible runs agent-less, i.e, it temporarily connects to the system via SSH or windows remote management to run its tasks.

Our goal with using Ansible was to setup 'n' instances, install dependencies and deploy our application with a single sequence of scripts (which can be combined into a single executable script). This sequence is illustrated in the below flow diagram 4.1

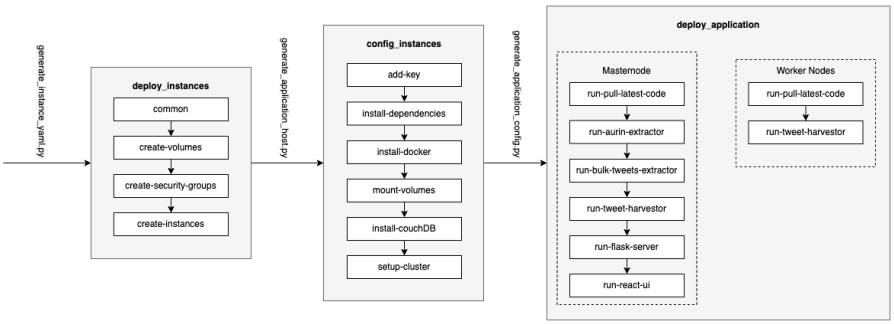

Figure 4.1: Ansible Execution Flow

- **Creating Instances on MRC**

  The first process in this sequence is creating and configuring instances on MRC as per requirement. This was accomplished using a small python helper script, generate_instance_yaml.py, to generate a yaml file containing details of instances, security groups, rules, storage volumes, etc dynamically based on the command line argument it took 'n', the number of instances to be configured. It creates a file, instances.yaml, as shown in figure 4.2

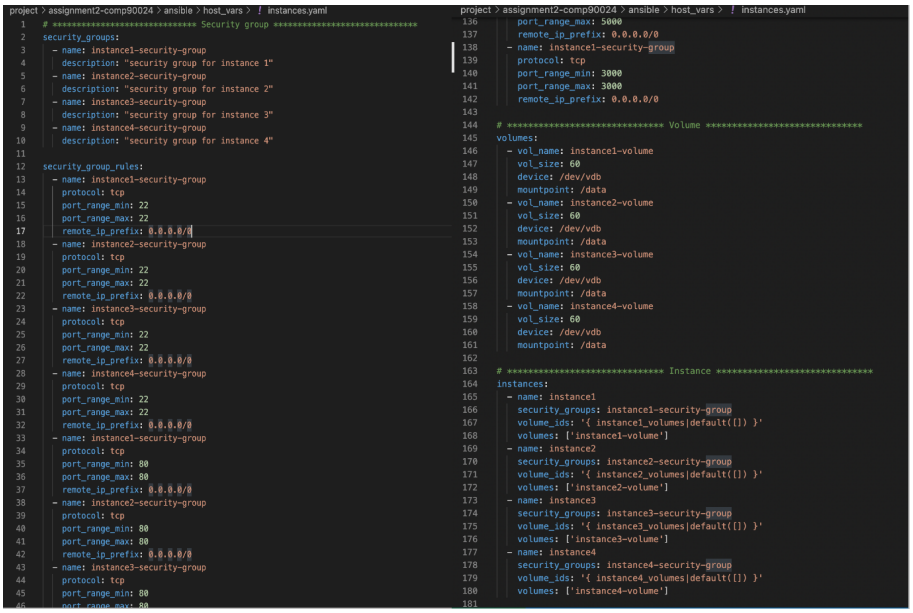

Figure 4.2: Instances.yaml

Once the instances.yaml is created, the deploy_instances.sh uses the Ansible playbook deploy_instances.yaml to run the tasks associated with the following roles.

1. **common:** This role is used to install dependencies such as python-pip, openstack sdk on the Ansible control node.
2. **create-volumes:** This role creates storage volumes on MRC as per the volumes section in instances.yaml.
3. **create-security-groups:** This crucial role creates the security group to be associated with each instance on MRC. Each security

group is associated with a list of ports to be exposed to enable instances to communicate with each other (couchdb, flask server, etc) and enable external access (React UI).

4. **create-instances:** This role creates instances on MRC based on the configurations in instances.yaml and maps the previously created volumes, security groups to it. Once the instances are created the role tests the connection to the instances and adds the instance details to the Ansible in memory inventory and also to a local file called wm_inventory_file.ini.

   Once the playbook is executed then another python script, generate_application_host.py, is called to create an application_hosts.ini file that defined host groups, instances, masternode, workers and database, to be used in later plays as shown in Figure 4.3.

project > assignment2-comp90024 > ansible > inventory > application_hosts.ini

```
1   [instances]
2   115.146.94.21
3   115.146.93.197
4   115.146.94.147
5   115.146.92.125
6
7   [database:children]
8   masternode
9   workers
10
11  [masternode]
12  115.146.94.21
13
14  [workers]
15  115.146.93.197
16  115.146.94.147
17  115.146.92.125
```

Figure 4.3: application_hosts.ini.yaml

- **Configuring Instances:** The next step in the deployment sequence is configuring the instances to deploy the applications and setup the database cluster. This is achieved using the config_instances.sh executable that uses the config_instances.yaml playbook comprising the following roles.

1. **add-key:** This role copies the ssh key to connect with the instances to the /.ssh folder of the control node instance and adds the key to the ssh-agent to enable ssh communication between the control node and the configured instances.
2. **install-docker:** This role uninstalls older versions of docker, sets up the right repository to download and installs docker on all the instances which is required to build and run the application.
3. **install-dependencies:** This role installs all the dependencies on the instances to enable the deployment of the application such as apt-transport-https, python3-pip, build-essential, ca-certificates, git, curl. It also tries to update pip to ensure pip is updated if its already installed.
4. **mount-volumes:** This role installs the dependency of xfsprogs and formats the volumes to XFS filesystem on the VM and mount them on the location specified in the mountpoint specified in the instances.yaml.
5. **install-couchDB:** This role is responsible to setup couchDB on the instances. It creates a couchdb user on all the instances, creates a directory to be mapped into the couchDB container, creates an etc directory for the couchDB local.ini file, copies the local.ini file to that directory and starts a container with couchDB image (couchdb:latest) with the required config as shown in Figure 4.4.

   Note that mounting the /data/couchdb/etc to /opt/couchdb/etc/local.d within the container ensures that couchdb picks up its local.ini config within the container from the location it was previously copied to.

   It also maps the ports within the docker container to the instances' ports, namely the couchdb port, the Erlang port and the cluster communication port which is used by nodes to talk to each other.
6. **setup-cluster:** Once the couchDB container is running on all instances, the next thing to be done is setup a cluster so that the nodes talk to each other and we can leverage the functionalities of couchDB such as sharding and replication. This is done using the setup-cluster role as follows:

```yaml
- name: Start a container with a couchdb
  become: yes
  docker_container:
    name: "{{ couchdb_container_name}}"
    image: "{{ couchdb_image }}"
    ports:
      - "{{couchdb_port}}:{{couchdb_port}}"
      - "{{erlang_port}}:{{erlang_port}}"
      - "{{cluster_com_port}}:{{cluster_com_port}}"
    volumes:
      - /data/couchdb/etc:/opt/couchdb/etc/local.d
    env:
      COUCHDB_PASSWORD: "{{ couchdb_pass }}"
      COUCHDB_USER: "{{ couchdb_user }}"
      NODENAME: "{{ ansible_host }}"
      COUCHDB_SECRET: "{{ couchdb_cookie }}"
```

Figure 4.4: Docker run command for couchDB container

(a) **Enabling Cluster:** This step enables the cluster by registering all the nodes that are to be added to the cluster with the setup coordination node (couchdb does not have the concept of master node, the setup coordination node is only used to setup the cluster but it does not have any special privileges). This is done as follows calling the "_cluster_setup" api with action "enable_cluster". See Figure 4.5.

(b) **Add nodes to cluster:** This step involves adding nodes to the cluster using the "_cluster_setup" api with action "add_node" for each node to be added. See Figure 4.6.

(c) **Finalize cluster setup:** The last step in the cluster creation is the finalize cluster setup task. This is done calling the same "_cluster_setup" api with the action "finish_cluster". See Figure 4.7.

(d) **Create DB tables:** Once the cluster is created and finalised the next step in sequence is the creation of tables to realize our

```
- name: setup couchdb cluster
  become: yes
  uri:
    url: http://{{ groups['masternode'][0] }}:{{couchdb_port}}/_cluster_setup
    status_code: 201
    method: POST
    user: "{{ couchdb_user }}"
    password:  "{{ couchdb_pass }}"
    force_basic_auth: yes
    return_content: yes
    body_format: json
    body: "{\"action\": \"enable_cluster\", \"bind_address\":\"0.0.0.0\",\
          \"username\": \"{{ couchdb_user }}\", \"password\":\"{{ couchdb_pass }}\", \"port\": \"{{couchdb_port}}\",\
          \"remote_node\": \"{{ item }}\", \"node_count\": \"{{ groups['database'] | length }}\",\
          \"remote_current_user\":\"{{ couchdb_user }}\", \"remote_current_password\":\"{{ couchdb_pass }}\"}"
    headers:
      Content-Type: "application/json"
  loop: "{{ groups['workers'] }}"
```

Figure 4.5: Enable cluster task

```
- name: add nodes to cluster
  become: yes
  uri:
    url: http://{{ groups['masternode'][0] }}:{{couchdb_port}}/_cluster_setup
    status_code: 201
    method: POST
    user: "{{ couchdb_user }}"
    password:  "{{ couchdb_pass }}"
    force_basic_auth: yes
    return_content: yes
    body_format: json
    body: "{\"action\": \"add_node\", \"host\":\"{{ item }}\",\
          \"port\": \"{{ couchdb_port }}\", \"username\": \"{{ couchdb_user }}\", \"password\":\"{{ couchdb_pass }}\"}"
    headers:
      Content-Type: "application/json"
  loop: "{{ groups['workers'] }}"
```

Figure 4.6: Add Node to cluster task

```
- name: finish setup
  become: yes
  uri:
    url: http://{{ groups['masternode'][0] }}:{{couchdb_port}}/_cluster_setup
    status_code: 200,201,409
    method: POST
    user: "{{ couchdb_user }}"
    password:  "{{ couchdb_pass }}"
    force_basic_auth: yes
    return_content: yes
    body_format: json
    body: "{\"action\": \"finish_cluster\"}"
    headers:
      Content-Type: "application/json"
```

Figure 4.7: Finalize cluster creation

use-cases. This is done using the “Create DB Tables” task using the ‘couch_tables’ list from host_vars/couchDB.yaml. See Figure 4.8.

```
- name: Create DB Tables
  become: yes
  uri:
    url: http://{{ groups['masternode'][0] }}:{{couchdb_port}}/{{item.name}}?q=4&n=3
    status_code: 200,201,409
    method: PUT
    user: "{{ couchdb_user }}"
    password:  "{{ couchdb_pass }}"
    force_basic_auth: yes
    return_content: yes
  loop: "{{couch_tables}}"
```

```
TASK [setup-cluster : Create DB Tables] ****************************************************************************************
ok: [115.146.94.21] => (item={u'design_name': u'getLatestRAI', u'name': u'housing_rent_aff', u'view': u'getLatestRAI.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEnvData', u'name': u'env_pollutant', u'view': u'getEnvData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getTopSchoolInfo', u'name': u'sch_rank', u'view': u'getTopSchoolInfo.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntCafeData', u'name': u'ent_cafe_seating', u'view': u'getEntCafeData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntBarData', u'name': u'ent_bar_seating', u'view': u'getEntBarData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getSportsData', u'name': u'ent_sports', u'view': u'getSportsData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntCarParkData', u'name': u'ent_car_park', u'view': u'getEntCarParkData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getHospitalData', u'name': u'health_no_beds', u'view': u'getHospitalData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getPostCode_BBox', u'name': u'postcode_bbox', u'view': u'getPostCode_BBox.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiments_count', u'name': u'archived_tweets', u'view': u'archived_tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiment_view', u'name': u'tweets', u'view': u'tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiment_view_bulk_tweets', u'name': u'bulk_tweets', u'view': u'bulk_tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'liveability', u'name': u'liveability', u'view': u'liveabilityView.json'})
ok: [115.146.94.21] => (item={u'name': u'realtime_tweets'})
```

Figure 4.8: DB creation task and output

(e) **Create DB Views:** Once the tables are created, we create views to ease the process of querying documents from the table by enabling filtering, sorting and faster query time using the creation of B-Trees. This is done using the "Create DB Views" task also using the 'couch_tables' list from host_vars/couchDB.yaml. See Figure 4.9.

- **Deploying Application:** Once the instances are created and configured along with the database, the next step involves building and running the application components on the instances a per the requirement. This is achieved by the deploy_application.sh executable that runs the Ansible playbook deploy_application.yaml.

  Before running this another python helper script, generate_application_config.py, is executed to create the .env file for each instance of the twitter harvester which takes the number of instances as a command line argument and splits categories among the harvester instances to enable them to search multiple categories parallelly. It generates a file that looks like Figure 4.10 for each instance.

  The playbook deploy_application.yaml includes the following roles that are executed to build and deploy the application components.

```
- name: Create DB Views
  become: yes
  uri:
    url: http://{{ groups['masternode'][0] }}:{{couchdb_port}}/{{item.name}}/_design/{{item.design_name}}
    status_code: 200,201,409
    method: PUT
    user: "{{ couchdb_user }}"
    password:  "{{ couchdb_pass }}"
    headers:
      Accept: "application/json"
      Content-Type: "application/json"
    body: "{{ lookup('file', 'files/views/'+item.view) }}"
    body_format: json
    force_basic_auth: yes
    return_content: yes
  register: result
  loop: "{{couch_tables}}"
  when: item.design_name is defined
```

```
TASK [setup-cluster : Create DB Views] *****************************************************************************************
ok: [115.146.94.21] => (item={u'design_name': u'getLatestRAI', u'name': u'housing_rent_aff', u'view': u'getLatestRAI.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEnvData', u'name': u'env_pollutant', u'view': u'getEnvData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getTopSchoolInfo', u'name': u'sch_rank', u'view': u'getTopSchoolInfo.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntCafeData', u'name': u'ent_cafe_seating', u'view': u'getEntCafeData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntBarData', u'name': u'ent_bar_seating', u'view': u'getEntBarData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getSportsData', u'name': u'ent_sports', u'view': u'getSportsData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getEntCarParkData', u'name': u'ent_car_park', u'view': u'getEntCarParkData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getHospitalData', u'name': u'health_no_beds', u'view': u'getHospitalData.json'})
ok: [115.146.94.21] => (item={u'design_name': u'getPostCode_BBox', u'name': u'postcode_bbox', u'view': u'getPostCode_BBox.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiments_count', u'name': u'archived_tweets', u'view': u'archived_tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiment_view', u'name': u'tweets', u'view': u'tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'sentiment_view_bulk_tweets', u'name': u'bulk_tweets', u'view': u'bulk_tweets.json'})
ok: [115.146.94.21] => (item={u'design_name': u'liveability', u'name': u'liveability', u'view': u'liveabilityView.json'})
skipping: [115.146.94.21] => (item={u'name': u'realtime_tweets'})
```

Figure 4.9: DB view creation task and output

```
Bearer_Token = [redacted]
consumer_key = "[redacted]"
consumer_secret = "[redacted]"
access_token = "[redacted]"
access_token_secret = "[redacted]"

geoCode = "-37.840935,144.946457,200km"
jsonfile = "twitterData.json"
csvfile = "searchWords.csv"

tweetsdbname="tweets"
realtime_tweetsdbname="realtime_tweets"
archived_dbname="archived_tweets"
bulk_dbname="bulk_tweets"

categories="Environment,Housing"

melb_bb_y_min = 144.3336
melb_bb_y_max = 145.8784
melb_bb_x_min = -38.5030
melb_bb_x_max = -37.1751
```

Figure 4.10: .env File for each harvester instance

1. **clone-git-repository:** The first task in this play is to clone the git repository on each of the instances using ssh. To enable this the an ssh private key is added to each instance and the public key is added to the ssh key section of the gitlab user preferences, then a directory /project is created in each instance and the repository is cloned to this directory.

2. **run-aurin-extractor:** This task is executed on the master node of the cluster. Aurin extractor is used to extract datasets from Aurin and perform relevant aggregations and push it to the respective couchDB tables. It first uses regex to replace couchdb url in the config json as seen in Figure 4.11. Once the url is replaced it builds a docker image and runs the container in background using the -d flag.

```
- name: Update CouchDB masternode url
  become: yes
  replace:
    dest: /home/ubuntu/project/assignment2-comp90024/AURIN_Harvestor/config/couchDB_config.json
    regexp: '[0-9]+(?:\.[0-9]+){3}:[0-9]+'
    replace: "{{groups['masternode'][0]}}:{{couchdb_port}}"
```

Figure 4.11: Regex to replace couchdb url

3. **run-bulk-tweets-extractor:** This task is also executed on the master node of the cluster. Bulk tweets extractor leverages snsscrape a python library that enables scraping social networking services to retrieve tweets older than 7 days which cannot be retrieved using the search tweets api. This task also leverages regex to replace couchDB URL in the .env file and builds the docker image and runs the container in background using the -d flag.
4. **run-tweet-harvestor:** This task is executed on both the masternode and the workers of the cluster. Tweet harvester leverages the search tweets API and realtime streaming API to retrieve tweets based on categories configured in the .env file. This task also starts with replacing the couchDB URL in the .envdb file and then copies the relevant instance¡n¿.env, where n is the instance number file created by the generate_application_config.py as the .env file to be used. It then builds the docker image and runs the container in background using the -d flag.
5. **run-flask-server:** This task is also executed on the masternode. Flask server is built to support complex queries from the react API to couchdb to reduce the computation in the UI. This task also leverages regex to replace couchDB URL in the .envdb_api file and builds the docker image and runs the container in background using the -d flag and also maps the 5000 port in the docker container to 5000 port in the instance to enable it to be accessed externally.

6. **run-react-ui:** This task is also executed on the masternode. This task deploys the react UI that is used to visualize the scenarios and provide the user with an interactive interface. This task also leverages regex to replace couchDB URL and the flask server URL in the .env file and builds the docker image and runs the container in background using the -d flag and also maps the 3000 port in the docker container to 3000 port in the instance to enable it to be accessed externally.

Figure 4.12 shows the output this playbook.

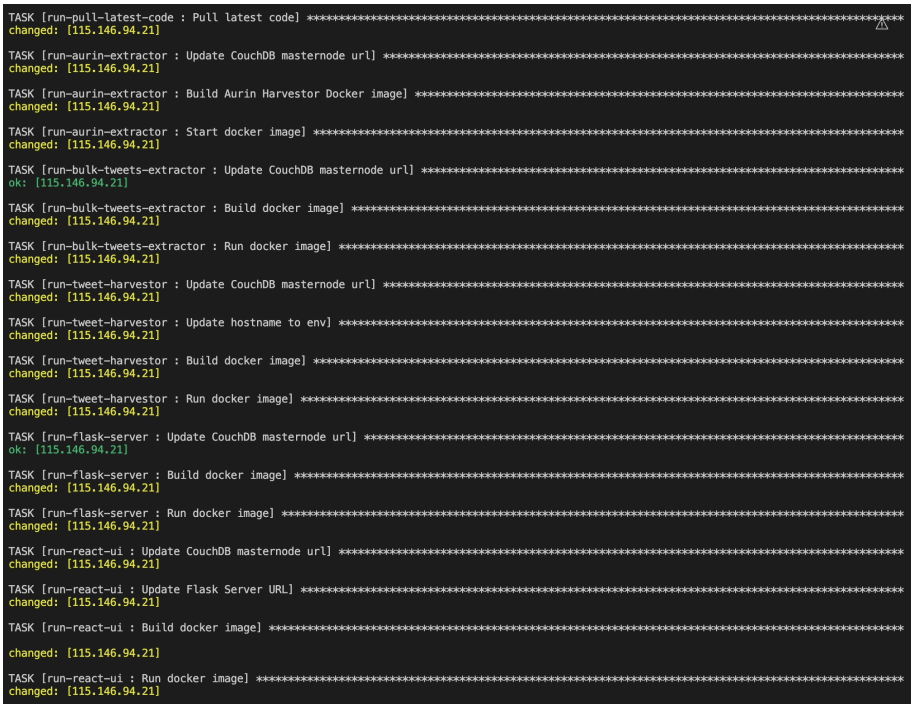

Figure 4.12: deploy_application.yaml playbook output

Section 5

Twitter Harvestor

Twitter is a social networking site. It is amongst the most popular social media platforms available today with millions of active users. The posts by users are known as "tweets", which are being used by our application.

Twitter has various APIs to help get data as per the need. As per our requirement, we wanted to get tweets from within the greater Melbourne region. The pulled tweets are based on specific keywords for each category.

5.1 Keyword Identification

To get started, we first identified keywords related to the factors chosen: housing, education, environment, health care, and entertainment. After some research, we realized that the tweets we receive may not be sufficient. Thus, to increase the number of tweets we receive, we also included the synonyms for the identified keywords.

5.2 Fetching Tweets using Twitter API

Twitter has numerous APIs that support keyword search. However, we couldn't use many of them. The key challenges for all these are mentioned below.

- **Search full archive API:** Only users with Academic Research access can use the API. Since we had a developer account, we couldn't get data using this.
- **Search 30-day:** It is a premium API and requires a payment to be made to use it.
- **Search recent tweets:** We were unable to filter tweets using the location and thus couldn't use this API.
- **Search tweets:** This endpoint served all our needs and we chose to use it. However, it only returned tweets from the last 7 days.

Limitation: It is worth mentioning that, Twitter has updated its terms and now allows to search tweets based on user location only as opposed to searching by tweet location earlier.

5.3 External Feed(s)

Using the search tweets API we were able to fetch about 70K tweets, but they weren't enough to help us make a decision. To resolve this issue, we decided to use other methods to fetch tweets.

- **Snscrape:** It is a scraper for social networking services. This library helped us pull tweets of our interest[2].
- **Data from the sample file:** We filtered tweets relevant to our application from the sample file provided to us as part of this assignment
- **Streaming API:** To keep updating the tweets database and get satisfactory results, we used the streaming API from Twitter.

5.4 Extracting information from the tweets

A tweet contains a lot of information; thus, we filtered the information required to make an informed decision. We analyzed the tweet text to understand which category the tweet belongs to. To get information about the sentiment of a tweet, we used, VADER (Valence Aware Dictionary and sEntiment Reasoner), a lexicon and rule-based sentiment analysis tool

5.5 Storing Data

We stored data into the CouchDB database with a flag, relevant which was set to true if the tweets belonged to the greater Melbourne area and contains one of the search keywords and false otherwise. This helped us to further enhance the accuracy of our application. We created views for the tweets stored which were then used by UI to display the data. Also we also implemented a cron job to update data received by streaming API into the tweets database at the end of the day. This helped us to keep our database updated.

5.6 Implementation Strategy

The Twitter harvester is executed on multiple instances. Each instance is configured to pull tweets for the category supplied via the environment file. **Also, we used the tweet id as the document id in order to avoid duplication of data.**

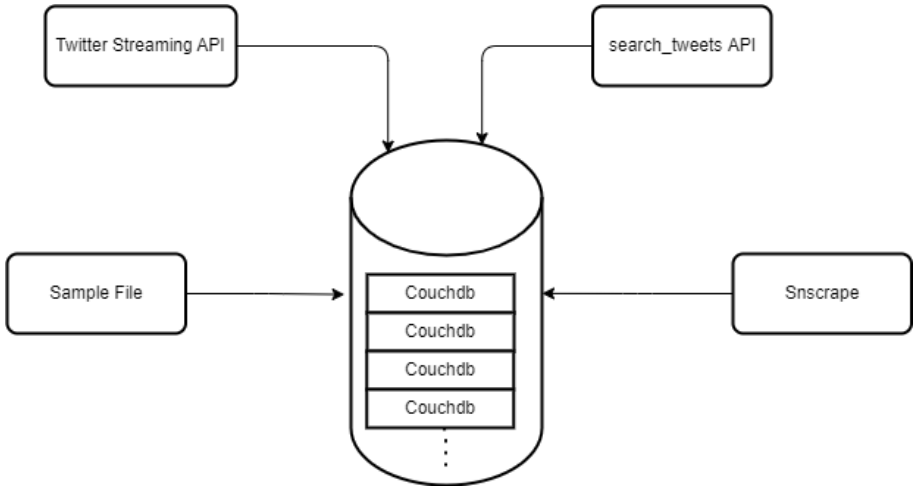

Figure 5.1: Twitter Harvester Flow chart

Section 6

AURIN Harvestor

Australian Urban Research Infrastructure Network is a one-stop online workbench with access to thousands of multidisciplinary datasets. It is developed to facilitate data-intensive research projects focused on Australian towns and cities.

6.1 Data Identification

The AURIN datasets are identified based on the use case selection. The following tables are used for analysis.

1. *sgs_rai_index_national_total_2021 (Housing)*
    (a) It gives information about the Rent affordability index of the area
    (b) National Shelter, Bendigo Bank, The Brotherhood of St Laurence, and SGS Economics and Planning releases the Rental Affordability Index (RAI) annually since 2019
    (c) RAI = (Median Income / Qualifying Income) x 100 where Median Income is the Household income and Qualifying Income is the household income required to pay rent where rent is equal to 30% of income.[3]
2. *UNSW_CFRC_NHPA_my_hospital_beds (Healthcare)*
    (a) It provides the number of beds to be used by the admitted patient [4]

(b) The data is divided into 5 categories:

i. **Category 1:** Beds less than 50
ii. **Category 2:** 50 to 100
iii. **Category 3:** 100 to 200
iv. **Category 4:** 200 to 500
v. **Category 5:** greater than 500

3. *national_pollutant_inventory_facilities_2018 (Environment)*

   (a) It gives information about the number of polluting factors and activities in the area

   (b) It includes point locations of facilities with known emissions of toxic substances. [5]

4. *com_clue_cafe_restaurant_bistro_seats_2017 (Entertainment)*

   (a) It gives information about the number of cafes and seats in an area

   (b) It is collected as part of the City of Melbourne's Census of Land Use and Employment (CLUE) [6]

5. *com_clue_bars_pubs_patron_capacity_2017 (Entertainment)*

   (a) It gives information about the number of pubs in an area

   (b) It is collected as part of the City of Melbourne's Census of Land Use and Employment (CLUE) [7]

6. *vic_sport_and_recreation_2015 (Entertainment)*

   (a) It gives information about the number of sports activities played in an area [8]

(Note: All the latest available tables are used for the analysis.)

6.2 Data Fetch

1. AURIN API supports openAPIs. The datasets can be accessed via the API calls.

2. Datasets, mentioned in the data identification section, are fetched as JSON and are then pushed to the CouchDB instance.

3. API for the hospital beds was not working, hence the data was downloaded in JSON and then pushed to the harvester.

6.3 External Data collection

1. Data for the SES score of the schools are taken from an external source, better education[9]. The data is passed to the AURIN harvestor as an excel file.

6.4 Greater Melbourne Liveability Index

1. Based on the fetched data from AURIN and other external resources, a score is generated for each postcode.

2. The score of a region takes into consideration the following parameters:

   (a) Rent affordability score

   (b) Estimated total number of hospital beds

   (c) Total Number of cafes, bars, and sports facilities

   (d) Total school enrolments and average SES score of all schools in the region

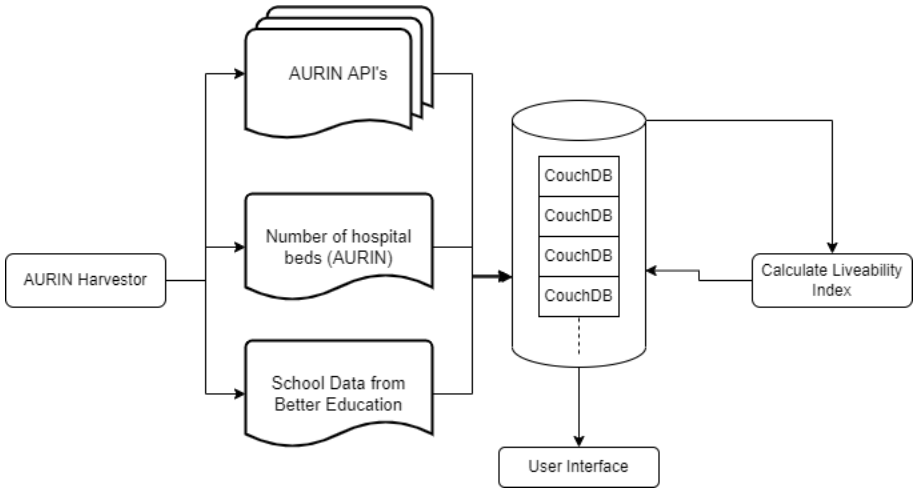

Figure 6.1: AURIN Harvester Flow chart

The liveability score represents the region's Liveability index. The higher the score, the more liveable the region is..

Section 7

UI Design

This application uses React JS to render UI. We used React as it is highly flexible and promotes code reusability by leveraging react components.

7.1 Visualization

We have created two views to visualize both the summary and the region wise livability information.

1. First view is the overall summary of different factors that we used to calculate the liveability of Melbourne by using graphical representation of the data. This was done to achieve better understanding of data through visualization components such as maps, tables, tiles, etc. Furthermore, we have rendered a "recent tweets" section showing the tweets posted in Melbourne for the corresponding liveability factors.

2. Second view further expands on the granularity of our data by relating it to corresponding insights that provide actionable details about the listed liveability factors. To achieve this, we have aimed to render the granular insights based on the postcode of the suburbs located in Melbourne. This postcode-wise graphical visualization allows the users to view the data in a more segregated, location-based manner, adding to more insight-driven analysis.

7.2 React Components

7.2.1 Map

Data is displayed on a map using leaflet map in React JS, which allows us to locate data points using latitude and longitude as well as displaying other information of that data point as a pop-up. The map uses light, low contrast style and vibrant icons for better visualization of the data points[10].

7.2.2 Table

The application is using the react-table library to display the summarized data of all postcodes. This includes their respective factor score as well as their rank which is calculated on the basis of these following five factors.

Since we have created an interactive table, users can sort the data in ascending or descending order just by clicking on any column name, which enables the user to gain better, and more actionable insights.

For example, if a user wants to find the postcode with the best schooling, then the user can simply click the "School" column and easily view the postcodes and detailed insights related to it.

7.3 Line Graph

We implement the line graph to show the quarter-wise rent affordability from the first quarter of 2018 till the second quarter of 2021. We have specifically used line graphs here as it offers better visualization and insights on the movement and momentum of the rent pattern from 2018 to 2021.

7.4 Bar Graph

The application has used the react-charts-2 library to implement the bar chart component. Using the bar chart, we have shown the number of bars, cafes, and sports facilities in our entertainment section. The reason behind making the choice of using a bar chart is that it provides a great comparison between different associated attributes.

7.5 Tiles

We have leveraged the tile component to display the data and insights that we have fetched from the tweets for all five liveability factors respectively, and the corresponding sentiments for the same. The insights rendered through the tiles include total tweets, positive, negative, and neutral sentiments; furthermore, we have color coded them for better understanding and analysis.

7.6 Live Tweets Streaming

We have used material-UI list to display the 20 latest tweets from the twitter live stream API. In order to better understand the sentiments of a tweet, we have assigned different colours to the background of the list object. Green symbolizes positive tweets, red symbolizes negative tweets and yellow helps identify tweets with neutral sentiments.

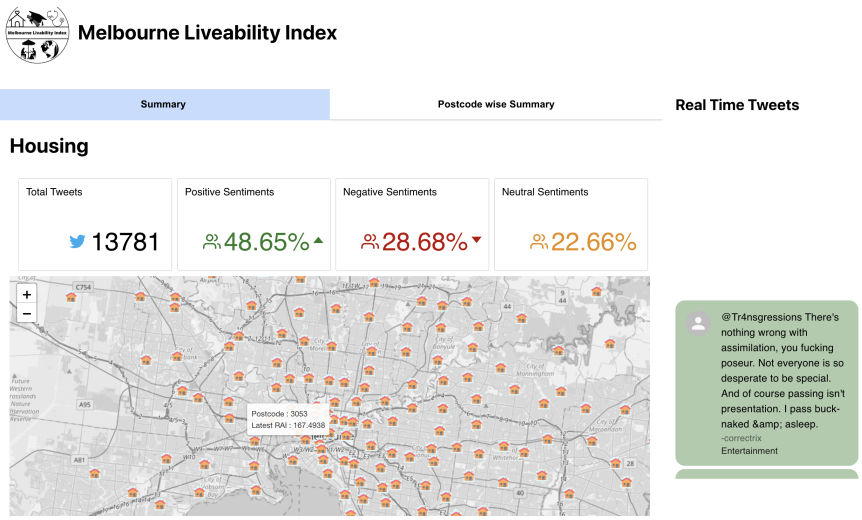

Figure 7.1: Map using leaflet

Figure 7.2: Line Chart depicting RAI information

Table Summary

Score	Postcode	2021 Q2 RAI	Total Enrollments	Avg SES	Total Polluting Activities	No. of cafe	No. of sports facilities	No. of hospital beds
4.06	3150	108.1731	10010	97.57	1	1	28	2250
3.93	3084	120.0534	3330	94	1	1	11	17000
3.92	3199	136.6397	740	91	1	1	46	8750
3.69	3000	148.3516	0	0	1	1586	5	125
3.61	3152	115.3846	4650	98.8	4	1	22	5750
3.57	3144	126.6417	650	97.5	1	1	16	12500
3.46	3121	115.3846	630	91.5	7	1	34	8000
3.37	3029	140.3326	5320	92.2	5	1	22	0
3.33	3128	136.6397	1210	95	5	1	11	10000
3.27	3168	138.4615	170	94	17	1	16	12500

<< Previous Next >> Page **1 of 67** | Go to page: 1 Show 10

Data collected from AURIN

Figure 7.3: Table with sorting functionality

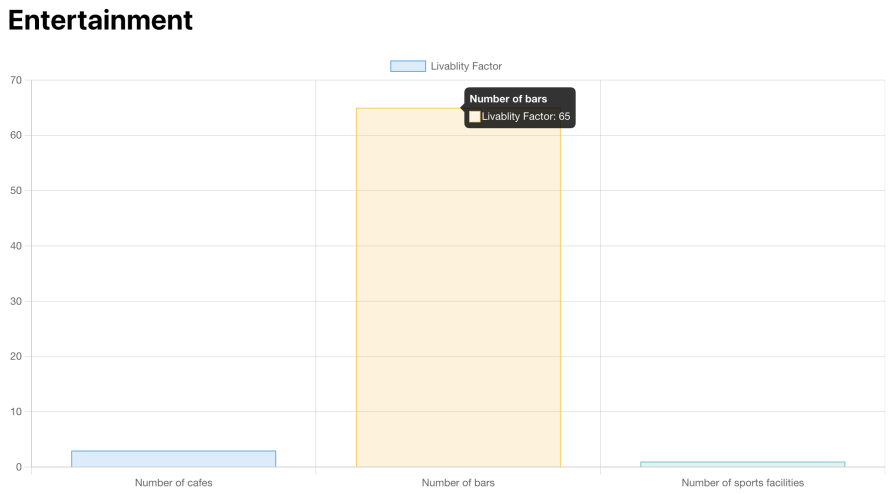

Figure 7.4: Bar graph depicting number of cafes, bars and sports facilities

Section 8

Data Inferences and Use Case Scenarios

The application takes data majorly from AURIN and Twitter platform. There are various inferences that can be made based on the sentiment data from Twitter and using factual data from AURIN.

Some of the inferences are discussed below:

8.1 Data Inferences

1. **Impact of COVID on Housing:** We can observe a sharp decrease in RAI after 2020, Q3. This is likely due the fact that the housing and rent prices soared up during pandemic. See Fig 8.1

2. **Impact of hospital location:** Hospitals are clustered around the CBD area. This is not preferred by people, especially who live in far away suburbs. Same can be inferred from the Twitter sentiments. See below figure for reference.See Fig 8.2

3. **Impact of school location:** Education is an important liveability factor. People prefer to stay in the areas closer to schools. With constant government efforts to improve the education industry, it is evident from the figure that the schools are scattered all over the Greater Melbourne region. The positive public sentiments can be observed from the fetched twitter data.See Fig 8.3

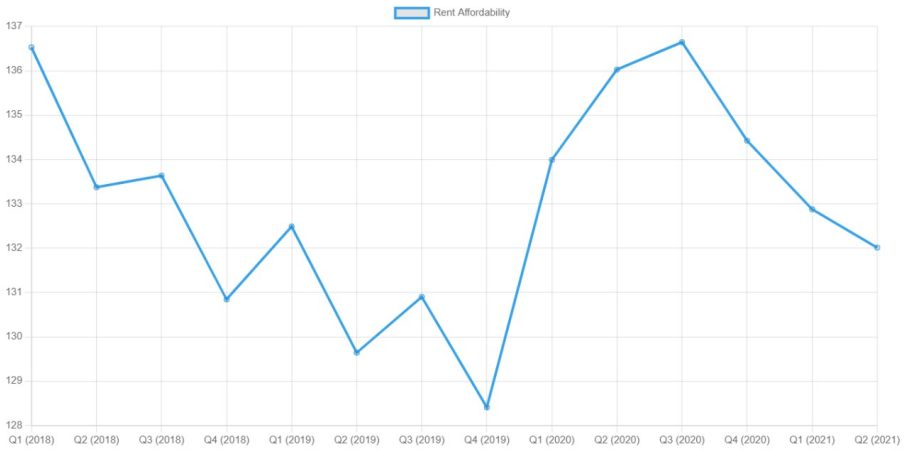

Figure 8.1: Greater Melbourne RAI Index

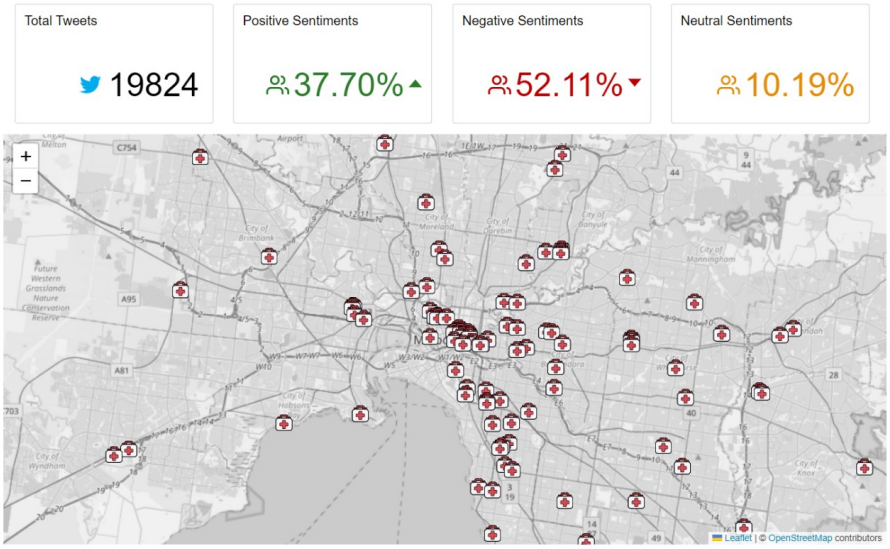

Figure 8.2: Greater Melbourne Hospital locations

4. **Impact of bars, cafes and sports facilities:** With the widespread entertainment options across the Greater Melbourne area people seem to be extremely happy. See Fig 8.4

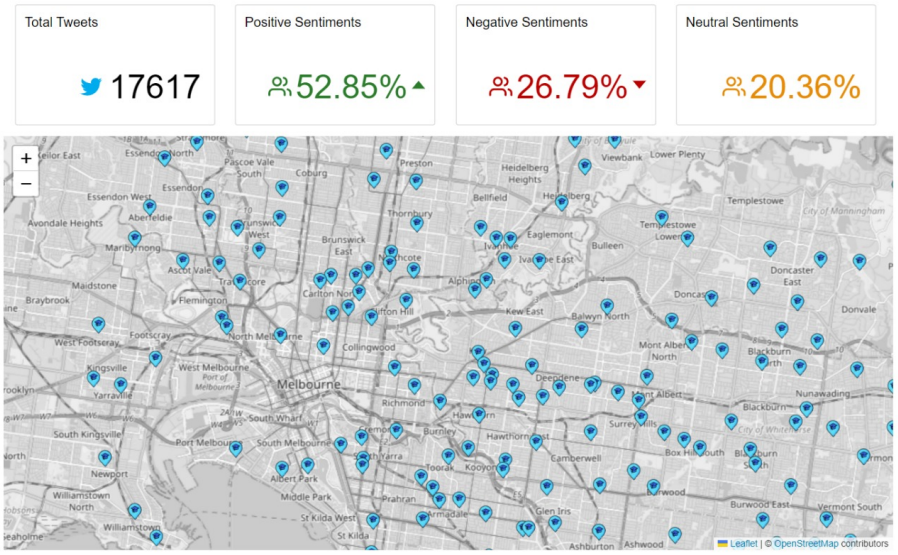

Figure 8.3: Greater Melbourne School locations

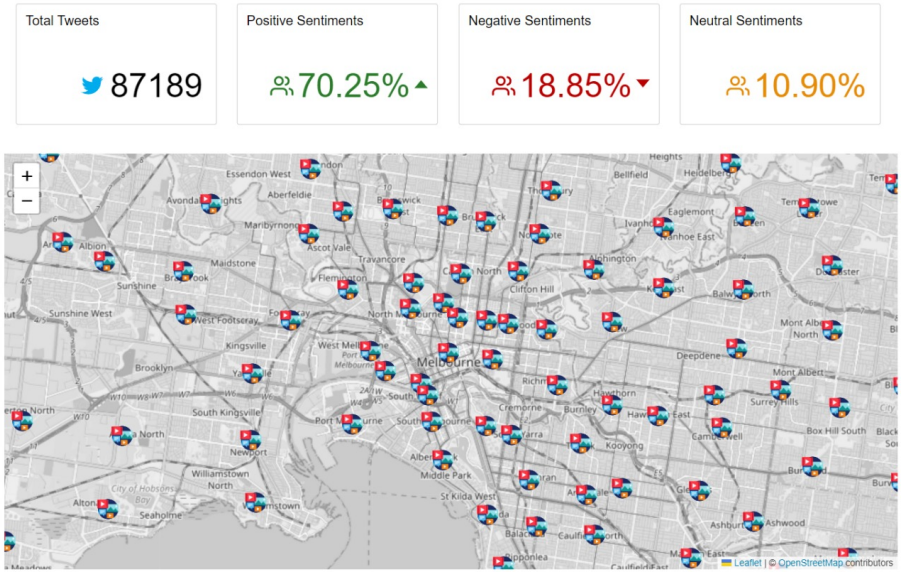

Figure 8.4: Greater Melbourne Entertainment locations

8.2 Use Case Scenarios

Melbourne Livability Index can be helpful in many real-life scenarios. It comments on the various factors, such as housing, healthcare, education etc. Below are some of the scenarios which can leverage the information from our application.

1. **Scenario 1 – An old couple looking to relocate :** Relocation can be a tedious task, especially for the older generation. It requires a lot of effort and one must make a wise decision.Our application helps identify various key factors to reduce the decision making hassles.
   It is easy to identify the postcode with nearby hospitals and amenities and which, also fits the budget.

2. **Scenario 2 – Government analysis:** The Melbourne Livability Index could be used by the government to evaluate the city's many aspects. The information gathered in terms of housing, entertainment options, healthcare facilities, schools, and the environment may provide insight into the city's current situation, and thus assist the government in determining what additional developments can be made to improve the city for residents and tourists.

3. **Scenario 3 – Student Rent:** Every year, Melbourne attracts thousands of international students. Rent plays an important role in helping them decide on their accommodation. Regions with high affordability index have a dense student population. Other factors like cafés, bars and sports facilities also play an important role.

4. **Scenario 4 – Young working couple looking to relocate:** A couple looking to move to Melbourne would like to examine the Melbourne Liveability Index in order to determine if the city provides all the necessary amenities. In terms of rent and location, housing is an incredibly essential consideration. If they have children, they may additionally take into account the location of nearby schools. They'd also like to live somewhere with basic leisure options like cafes, pubs, and sports centres, among other things.

Section 9

Key Challenges and Future Enhancements

Below are the component wise key challenges faced by the team, during the development lifecycle. Challenges along with the mitigation approach are discussed below.

9.1 Twitter Harvester

1. **Rate Limit for fetching tweet:** Twitter seach APIs with elevated access allows up to 900 requests over 15 minutes interval.
    (a) It was handled by setting the weight on rate limit parameter in Tweepy, which allowed the process to run in an uninterrupted manner.
2. **Developer account access:** There are several limitations for the elevated access twiiter developer profile. It is discussed in detail in the Twitter Harvestor section.
    (a) Snsscraper was used to scrape the old tweets for the past one year. More than 3.5 million tweets were pushed to couchDB
3. **Twitter Streaming APIs:** Twitter blocked the developer account multiple times while running the streaming APIs

4. **Tweets based on account location:** With the recent changes in the Twitter APIs, we were unable to search the tweets with geocode location. Account locations were used instead.

   (a) Use of account location created an extra step of filtering the relevant tweets in our application.

5. **Keyword functionality:** Although twitter APIs supports keyword search, but a lot of irrelevant tweets were fetched.

   (a) *Future Scope* The tweets are classified based on the intent classification model. This will improve the quality of the selected tweets.

6. **Redeployment Time:** After the redeployemnet of the application, tweets take some time to reflect in the user interface.

9.2 AURIN Harvester

1. **Unavailability of AURIN APIs:** Some datasets did not have AURIN openAPIs.

   (a) Datasets were manually downloaded and used in the Harvester.

2. **Recent Data not available:** Datasets were not up-to-date in the platform. All the latest available datasets were used to infer the results.

9.3 MRC

1. The User interface sometimes become unresponsive. However, it did not hamper the project development process.

9.4 CouchDB

1. Views can not be accessed from the table with incoming real time data.

   (a) An extra step is added by creating an staging table which pushes the real time tweets to the main tweet table at the regular intervals.

Section 10

User Guide

10.1 User Implementation Steps

This user guide provides end-user information on how they can deploy our system and can interact with the system to view and walk around with the web Application.
The end-user needs to follow the below-mentioned procedure in order to get the desired output:

1. Install the Python3 and ansible on the Linux-like machine where the code needs to be executed.

2. In order to get the source code and dependent files, the end-user needs to clone the git repository.
   Git Repository Link: https://gitlab.unimelb.edu.au/rgiri/assignment2-comp90024.git

3. Download the openrc.sh file from the MRC Dashboard. Place the same inside the ansible directory of the cloned project.

4. As the next step, create a key pair on the MRC by uploading a public ssh key generated using a local machine.

5. Copy the private key to ansible/config

6. Rename the private key to NectarGroupKey.pem

7. From the MRC Dashboard generate the OpenStack password. You may require to put the OpenStack password in many places. So, make sure it is easily accessible to you.

8. Open the terminal in the ansible directory and run the following commands to execute the deployment, while prompted enter your OpenStack password.

   (a) *./deploy_instance.sh*

   (b) *Python3 generate_application_host.py*

   (c) */config_instances.sh*

   (d) *python3 generate_application_config.py -n ¡number of instances to be created¿*

   (e) *./deploy_application.sh*

10.2 Useful Links

1. **ANSIBLE Deployment:** https://youtu.be/-HVDWYmRJSA
2. **UI Introduction:** https://youtu.be/BEzPUqfsRb8
3. **GitLab Link:** https://gitlab.unimelb.edu.au/rgiri/assignment2-comp90024.git

References

[1] Economist intelligent unit. URL https://global.vic.gov.au/victorias-capabilities/why-melbourne/one-of-the-worlds-most-liveable-cities.

[2] Sns scrape documentation. URL https://github.com/JustAnotherArchivist/snscrape.

[3] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access sgsep - rental affordability index - all dwellings for australia (polygon) q1 2011-q2 2021 on 12th may, 2022.

[4] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access victoria cfrc nhpa my hospital beds on 12th may, 2022.

[5] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access au_govt_dee national pollutant inventory facilities 2018 on 12th may, 2022.

[6] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access city of melbourne clue cafes, restaurants and bistros seats (points) 2017 on 12th may, 2022.

[7] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access city of melbourne clue bar, tavern and pub patrons capacity (point) 2017 on 12th may, 2022.

[8] This melbourne liveability index project used the ncris-enabled australian urban research infrastructure network (aurin) portal e-infrastructure to access victoria sport and recreation facility locations (2015-2016) on 12th may, 2022.

[9] Better education. URL `www.bettereducation.com.au`.

[10] Leaflet.js documentation. URL `https://leafletjs.com/`.