FINAL PROJECT  
TECHNICAL DESIGN DOCUMENT

**Submitted by:**

Jijo John Joseph (200521777)

Table of Contents

1. Executive Summary 3

2. Introduction 3

3. Technology Stack 3

4. Design Diagram 4

5. Project Workflow 4

6. Deliverables 22

7. Conclusion 22

8. References and Citation 22

# 1. Executive Summary

This project aims to perform sentiment analysis on Yelp reviews using MongoDB, RapidMiner, and Flask. By analysing customer sentiment, we can gain insights into customer satisfaction and help make out the best ice cream and desert shops in Toronto.

# 2. Introduction

Yelp is a renowned platform where customers express their experiences with various businesses. These reviews are a goldmine of data that, when analysed correctly, can provide valuable insights into customer satisfaction. This project proposes a system to scrape reviews of Ice Cream shops in Toronto from Yelp, analyse the sentiment of these reviews, and visualize the results in a user-friendly manner. This project is intended for newcomers in Toronto and tourists paying a visit to the city which would help them get an easier and faster means of knowing the best in touch. This would also help the business owners in making their service and offerings better to get more revenue.

# 3. Technology Stack

* Hardware requirement

MacBook Air or Pro / Manufacturers which support Windows operating system.

* Software Requirement

1. Python scripts: For scrapping reviews from yelp.
2. MongoDB: For storing and retrieving data.
3. RapidMiner: For data processing and sentiment analysis.
4. Flask: To display visualization on a Web application.

# 4. Design Diagram

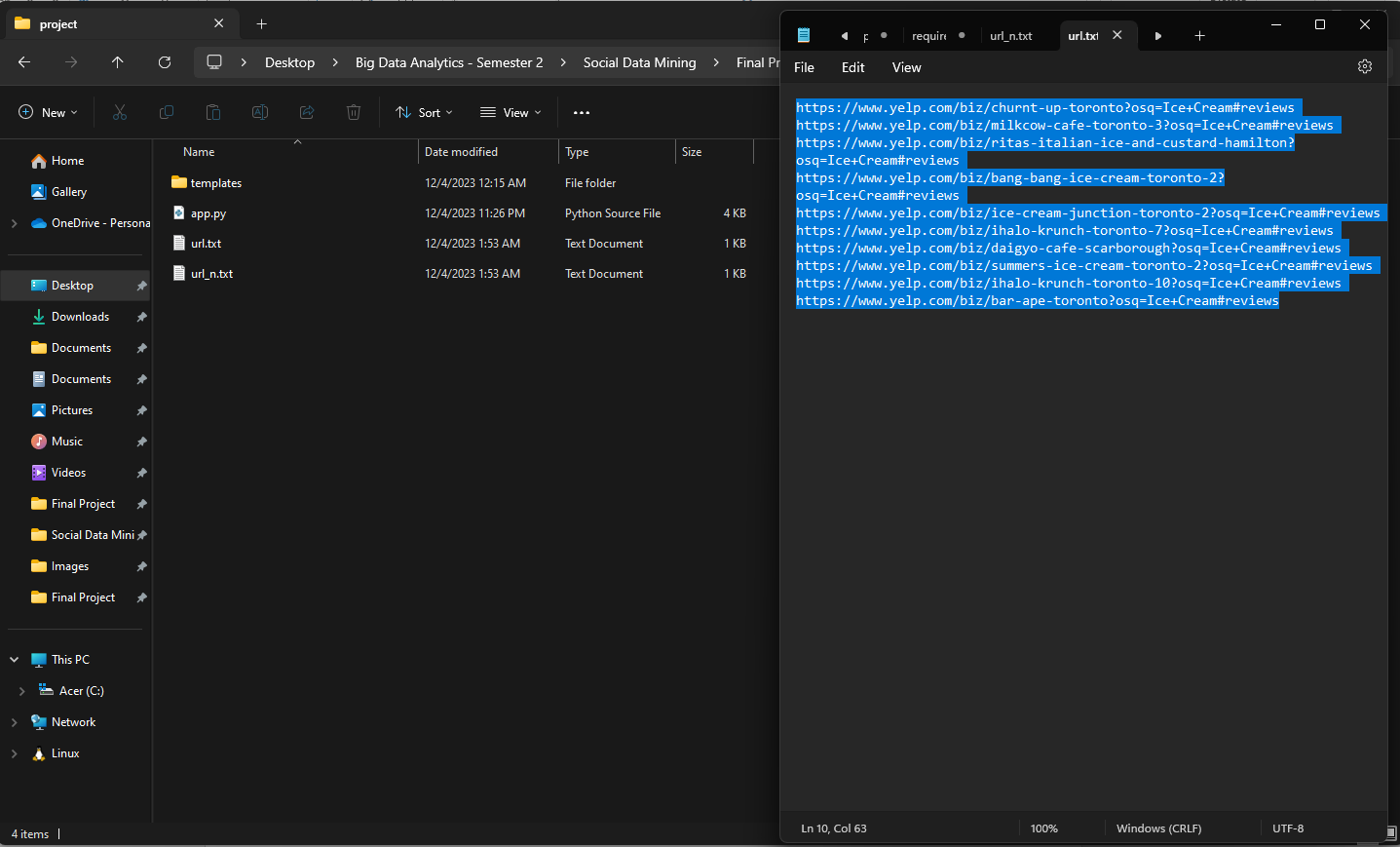
A screenshot of a computer game

Description automatically generated

# 5. Project Workflow

**Data Collection**

For data collection we may need to scrape the data from Yelp.com which would retrieve the reviews of various Ice Cream shops that are residing in Toronto. Initially we will retrieve the list of sample ice cream shops in Toronto on a file which is name url.txt.



**Scraping Python Script and storing DB:**

from flask import Flask, render\_template, jsonify

import pymongo

import requests

from bs4 import BeautifulSoup

import time

import urllib.parse

import uuid

import threading

#app = Flask(\_\_name\_\_)

# MongoDB database configuration

username = 'dbuser'

password = 'Pa$$word@61803'

escaped\_username = urllib.parse.quote\_plus(username)

escaped\_password = urllib.parse.quote\_plus(password)

database\_url = 'dbcluster1.2v8vfcl.mongodb.net'

db\_name = 'Database1'

# Creating MongoDB connection

uri = f"mongodb+srv://{escaped\_username}:{escaped\_password}@{database\_url}/{db\_name}"

# Connect to MongoDB using the URI

client = pymongo.MongoClient(uri)

db = client[db\_name]

collection = db['Collection4']

# Set user agent headers to mimic a web browser

headers = {

    'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/112.0'}

# Function to scrape reviews from a given URL

def get\_reviews(url):

    headers = {

        'User-Agent': 'Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:109.0) Gecko/20100101 Firefox/112.0'}

    soup = BeautifulSoup(requests.get(url, headers=headers).content, 'html.parser')

    biz\_id\_element = soup.find('meta', attrs={"name": "yelp-biz-id"})

    if biz\_id\_element is None:

        return []

    biz\_id = biz\_id\_element['content']

    review\_url = "https://www.yelp.ie/biz/{biz\_id}/review\_feed?rl=en&q=&sort\_by=relevance\_desc&start={start}"

    reviews = []

    for start in range(0, 100, 5):

        r\_url = review\_url.format(biz\_id=biz\_id, start=start)

        data = requests.get(r\_url, headers=headers).json()

        for r in data['reviews']:

            review\_text = BeautifulSoup(r['comment']['text'], 'html.parser').text

            reviews.append(review\_text)

    return reviews

# Function to process URLs, scrape reviews, and store them in the database

def process\_urls():

    with open('url.txt', 'r') as url\_file:

        urls = url\_file.read().splitlines()

    for url in urls:

        business\_name = BeautifulSoup(requests.get(url, headers=headers).content, 'html.parser').find('h1').text.strip()

        reviews = get\_reviews(url)

        review\_list = []

        for review in reviews:

            unique\_id = str(uuid.uuid4())

            review\_dict = {'ID': unique\_id, 'Business Name': business\_name, 'Review': review}

            review\_list.append(review\_dict)

        # Insert the reviews into the collection

        collection.insert\_many(review\_list)

**Data Storage**

The scraped data will be stored in MongoDB, a NoSQL database. This will allow us to efficiently store and retrieve the data. As we are working on the project , the database is free and can be used.

A screenshot of a computer

Description automatically generated

Collection 3 where the scrapped data is store with Business Name and Review . It will also contain the ID as well.

A screenshot of a computer

Description automatically generated

Collection 4 where the processed data after sentiment analysis is stored with the sentiment score and sentiment is stored.

A screenshot of a computer

Description automatically generated

**Data Processing**

We will use RapidMiner to read the data from MongoDB using the Read MongoDB operator. The data will then be converted from JSON format to a format suitable for analysis.

In order to process the data we will be using the Mongo DB read operator to read the scrapped data from the Mongo Database .

A screenshot of a computer

Description automatically generated

After reading the data, the data is in its json format which is processed using the JSON to Data operator that will convert the data and make it presentable for sentiment analysis.

A computer screen shot of a computer

Description automatically generated

Data cleaning is carried out through a python script which will remove all the unwanted characters in the review column using the Execute Python operator.

A diagram of a computer program

Description automatically generated with medium confidence

Data Cleaning Script:

**import** pandas **as** pd  
**import** re  
  
**def** clean\_reviews(data):  
 *# Define a function to clean a single review*  
 **def** clean\_review(review):  
 *# Remove non-alphanumeric characters*  
 review = re.sub(r'\W+', ' ', review)  
 **return** review  
  
 *# Apply the function to the review column*  
 data['Review'] = data['Review'].apply(clean\_review)  
  
 **return** data  
  
**def** rm\_main(data):  
 *# Clean the reviews*  
 data = clean\_reviews(data)  
   
 **return** data

**Sentiment Analysis**

We will use the Sentiment Analysis operator in RapidMiner to analyse the sentiment of the reviews. To achieve this we will use the Extract Sentiment operator to get the sentiment of each of the reviews corresponding to the Business Names or Ice cream shops.

A diagram of data cleaning

Description automatically generated

A Generate Attribute operator is used to give the Sentiment Scores for each of the values that were obtained in the Extract Sentiment step.

We use the below expression to give the sentiment values.

if(Score > 0, "Positive", if(Score < 0, "Negative", if(Score == 0, "Neutral", "")))

A diagram of data cleaning

Description automatically generated

**Result Storage**

The sentiment values will be generated as attributes and stored in another collection in the MongoDB database.

We use a python script as the operator for Mongo DB Write operator was not working as expected.

To store the sentiment data back to DB the below script was used.

import pymongo

import urllib.parse

def rm\_main(data):

    # MongoDB database configuration

    username = 'dbuser'

    password = 'Pa$$word@61803'

    escaped\_username = urllib.parse.quote\_plus(username)

    escaped\_password = urllib.parse.quote\_plus(password)

    database\_url = 'dbcluster1.2v8vfcl.mongodb.net'

    db\_name = 'Database1'

# Creating MongoDB connection

    uri = f"mongodb+srv://{escaped\_username}:{escaped\_password}@{database\_url}/{db\_name}"

# Connect to MongoDB using the URI

    client = pymongo.MongoClient(uri)

    db = client[db\_name]

    collection = db['Collection4']

    # Insert the data into the collection

    collection.insert\_many(data.to\_dict('records'))

    return data

A diagram of a diagram

Description automatically generated

In parallel, to have a secondary storage we use a Write Excel operator to store the Data on to an Excel file.

A diagram of a diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

Result after executing the flow:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Data Visualization**

We will use Flask, a Python web framework, to create a web application for visualizing the sentiment analysis results.

There are basically two files which we have to focus while creating the flask application that includes the python script and the html file.

App.py:  
from flask import Flask, render\_template, request

import pymongo

import urllib.parse

import matplotlib.pyplot as plt

import pandas as pd

app = Flask(\_\_name\_\_)

# MongoDB database configuration

username = 'dbuser'

password = 'Pa$$word@61803'

escaped\_username = urllib.parse.quote\_plus(username)

escaped\_password = urllib.parse.quote\_plus(password)

database\_url = 'dbcluster1.2v8vfcl.mongodb.net'

db\_name = 'Database1'

# Creating MongoDB connection

uri = f"mongodb+srv://{escaped\_username}:{escaped\_password}@{database\_url}/{db\_name}"

client = pymongo.MongoClient(uri)

db = client[db\_name]

collection = db['Collection4']

# Fetch data from MongoDB

data = list(collection.find({}))

# Convert data to DataFrame

df = pd.DataFrame(data)

@app.route('/')

def home():

    # Top 10 ice cream shops

    top\_10\_shops = df['Business Name'].value\_counts().nlargest(10)

    plt.figure(figsize=(10,5))

    plt.barh(top\_10\_shops.index, top\_10\_shops.values, color='skyblue')

    plt.xlabel('Number of Reviews')

    plt.title('Top 10 Ice Cream Shops')

    plt.gca().invert\_yaxis()

    plt.savefig('static/top\_10\_shops.png')

    # Sentiment distribution

    sentiment\_counts = df['Sentiment'].value\_counts()

    plt.figure(figsize=(5,5))

    plt.pie(sentiment\_counts.values, labels=sentiment\_counts.index, autopct='%1.1f%%', startangle=140)

    plt.axis('equal')

    plt.title('Sentiment Distribution')

    plt.savefig('static/sentiment\_distribution.png')

    return render\_template('index.html')

@app.route('/page1', methods=['GET', 'POST'])

def page1():

    # Top 10 ice cream shops

    top\_10\_shops = df['Business Name'].value\_counts().nlargest(10)

    if request.method == 'POST':

        shop = request.form.get('shop')

        # Sentiment distribution for the selected shop

        shop\_data = df[df['Business Name'] == shop]

        sentiment\_counts = shop\_data['Sentiment'].value\_counts()

        plt.figure(figsize=(5,5))

        plt.pie(sentiment\_counts.values, labels=sentiment\_counts.index, autopct='%1.1f%%', startangle=140)

        plt.axis('equal')

        plt.title(f'Sentiment Distribution for {shop}')

        plt.savefig('static/shop\_sentiment.png')

        return render\_template('page1.html', shops=top\_10\_shops.index.tolist(), img='shop\_sentiment.png')

    return render\_template('page1.html', shops=top\_10\_shops.index.tolist())

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <!-- Bootstrap CSS -->

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

    <title>SweetSpot Reviews</title>

    <style>

      body {

        background-image: url('/static/images/Logo.png');

        background-size: cover;

        background-repeat: no-repeat;

      }

      .btn-custom {

            background-color: #dc3545;

            color: black;

            border: none;

        }

        .btn-custom:hover {

            background-color: #c82333;

        }

            h1 {

                font-family: 'Amatic SC', cursive;

                color: #000000;

                text-align: Center;

                letter-spacing: 1px;

        font-weight: bold;

        font-size: 60px;

            }

      h3 {

                font-family: Garamond, serif;

                color: #000000;

                text-align: Center;

                letter-spacing: 1px;

        font-size: 18px;

            }

      h2 {

                font-family: 'Amatic SC', cursive;

                color: #000000;

                text-align: Center;

                letter-spacing: 1px;

        font-size: 20px;

            }

            p {

        padding: 5px 25px;

                font-family: 'Amatic SC', cursive;

                font-size: 20px;

                text-align: Center;

                color: #000000;

                font-weight: bold;

                letter-spacing: 3px;

                line-height: 20px;

        font-size: 30px;

        animation: flash 2s infinite alternate;

            }

      @keyframes flash {

        from {

          opacity: 1;

        }

        to {

          opacity: 0.1;

        }

      }

      .logo-img {

        max-width: max-content;

        display: block;

        margin: 0 auto;

      }

      .navbar {

        background-color: #000;

      }

      .navbar-brand, .navbar-nav .nav-link {

        color: #fff;

      }

      .chart-container {

        position: relative;

        height: 1;

        padding-bottom: 56.25%;

      }

      .chart-img1 {

        position: absolute;

        top: 0;

        left: 0;

        width: 100%;

        opacity: 0.7;

      }

      .chart-img2 {

        position: absolute;

        top: 0;

        left: 0;

        width: 80%;

        opacity: 0.7;

      }

      .chart-left {

        margin-left: 50px;

      }

    </style>

</head>

<body>

    <nav class="navbar navbar-expand-lg">

        <a class="navbar-brand" href="/">Home</a>

        <div class="collapse navbar-collapse">

            <ul class="navbar-nav ml-auto">

                <li class="nav-item">

                    <a class="nav-link" href="/page1">Ice Cream Shops</a>

                </li>

            </ul>

        </div>

    </nav>

    <h1 class="text-center">Ice Cream Shop Sentiment Analysis</h1>

    <div class="row">

        <div class="col-md-6 chart-left">

            <h2>Top 10 Ice Cream Shops</h2>

            <div class="chart-container">

                <img src="/static/top\_10\_shops.png" alt="Top 10 Ice Cream Shops" class="chart-img1">

            </div>

        </div>

        <div class="col-md-6 chart-left">

            <h2>Sentiment Distribution</h2>

            <div class="chart-container">

                <img src="/static/sentiment\_distribution.png" alt="Sentiment Distribution" class="chart-img2">

            </div>

        </div>

    </div>

</body>

</html>

Page 1:  
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Ice Cream Shop Sentiment Analysis - Page 1</title>

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

    <style>

        body {

            padding: 20px;

        }

        .chart-img {

            max-width: 100%;

            height: auto;

        }

        .navbar {

            background-color: #000;

        }

        .navbar-brand, .navbar-nav .nav-link {

            color: #fff;

        }

    </style>

</head>

<body>

    <nav class="navbar navbar-expand-lg">

        <a class="navbar-brand" href="/">Home</a>

        <div class="collapse navbar-collapse">

            <ul class="navbar-nav ml-auto">

                <li class="nav-item">

                    <a class="nav-link" href="/page1">Ice Cream Shops</a>

                </li>

            </ul>

        </div>

    </nav>

    <h1 class="text-center">Ice Cream Shop Sentiment Analysis</h1>

    <form method="POST">

        <div class="form-group">

            <label for="shop">Select an Ice Cream Shop:</label>

            <select class="form-control" id="shop" name="shop">

                {% for shop in shops %}

                <option>{{ shop }}</option>

                {% endfor %}

            </select>

        </div>

        <button type="submit" class="btn btn-primary">Submit</button>

    </form>

    {% if img %}

    <div class="mt-5">

        <h2>Sentiment Distribution</h2>

        <img src="/static/{{ img }}" alt="Sentiment Distribution" class="chart-img">

    </div>

    {% endif %}

</body>

</html>

Visualization:

A screen shot of a computer

Description automatically generated

A white screen with black text

Description automatically generated

# 6. Deliverables

1. A web scraping python script for collecting Yelp reviews.
2. A MongoDB database for storing the scraped data and the sentiment analysis results.
3. A RapidMiner process for reading the data from MongoDB, converting it, and performing sentiment analysis.
4. A Flask application for visualizing the sentiment analysis results.

# 7. Conclusion

This project gives us valuable insights into customer sentiment and help improve the service of ice cream shots. By investing in this project, we can increase customer satisfaction and ultimately increase revenue. This tool would be helpful for tourists and new comes in Toronto to visit ice cream shops in Toronto.

# 8. References and Citation

[Scrape Yelp Data using Python [Build Your Yelp Scraper] (scrapingdog.com)](https://www.scrapingdog.com/blog/how-to-scrape-yelp-reviews/)

[Beautiful Soup: Build a Web Scraper With Python – Real Python](https://realpython.com/beautiful-soup-web-scraper-python/)

[Flask - (Creating first simple application) - GeeksforGeeks](https://www.geeksforgeeks.org/flask-creating-first-simple-application/)

[How to build a web application using Flask and deploy it to the cloud (freecodecamp.org)](https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/)

[Build A Python Database With MongoDB | MongoDB](https://www.mongodb.com/languages/python)

[Connection Strings — MongoDB Manual](https://www.mongodb.com/docs/manual/reference/connection-string/)

[Can I use the connection string generated in the MongoDB Compass in Pymongo? - Stack Overflow](https://stackoverflow.com/questions/63568938/can-i-use-the-connection-string-generated-in-the-mongodb-compass-in-pymongo)

[Using MongoDB - RapidMiner Documentation](https://docs.rapidminer.com/9.4/studio/connect/database/mongodb.html)

[How to Integrate MongoDB Into RapidMiner - Zebra BI](https://zebrabi.com/guide/how-to-integrate-mongodb-into-rapidminer/)

[Putting data from MongoDB into a chart in python (using pandas dataframe) - Stack Overflow](https://stackoverflow.com/questions/70683894/putting-data-from-mongodb-into-a-chart-in-python-using-pandas-dataframe)

[Plot a pie chart in Python using Matplotlib - GeeksforGeeks](https://www.geeksforgeeks.org/plot-a-pie-chart-in-python-using-matplotlib/)

[Add HTML and CSS in Flask Web Application - ThinkInfi](https://thinkinfi.com/flask-adding-html-and-css/)

[Flask/Python - handling dropdown to open different HTML pages - Stack Overflow](https://stackoverflow.com/questions/35264455/flask-python-handling-dropdown-to-open-different-html-pages)

[HTML : How to create dropdown menu from python list using Flask and HTML - YouTube](https://www.youtube.com/watch?v=qtde16rN27U)

[Aggregation Stages — MongoDB Manual](https://www.mongodb.com/docs/manual/reference/operator/aggregation-pipeline/)

[Query and Projection Operators — MongoDB Manual](https://www.mongodb.com/docs/manual/reference/operator/query/)

[What is the re.sub() function in Python? (educative.io)](https://www.educative.io/answers/what-is-the-resub-function-in-python)

[Python | Regular Expressions | re.sub() | Codecademy](https://www.codecademy.com/resources/docs/python/regex/sub)