6th April 2025

Building a Distributed Web Application Using Cloud Computing

CT5169 – Fundamentals of Cloud Computing

CCS1 Postgraduate Diploma in Cloud Computing and Software Development

John Maguire – 24105284 – j.maguire15@universityofgalway.ie

Introduction

The task of this project was to build a distributed web application using cloud computing. The practicals held every Wednesday evening by *Dr Takfarinas Saber* and *Kouider Chadli* were instrumental to understanding how to complete the outlines tasks. Though I was not available to attend each evening, the recorded sessions proved invaluable to me for preparing to undertake this assignment.

Particularly I found myself returning to the content and Microsoft Teams sessions for "Lab 4: Networking (VM-VM)", "Lab 6: Docker", and "Lab 7: Using Flask Web Server", the latter of which I had open on my browser throughout the entire process.

These sessions laid the foundation of my understanding of sending information using port forwarding, the container which holds the database used to cache the built HTML content along with my introduction to MySQL, and using python packages to aid in the algorithmic logic of the application.

Web application overview

This web application includes two pages, one to search and one to display results. On searching for a term, the secondary virtual machine on my host laptop is connected via a python package and the term is selected from a table. If a result is found, that data is processed and quickly returned as the output to the results page.

If no data is found, then an algorithm connects via SSH to the AWS instance and runs the wiki.py script, the output is converted to HTML on the main virtual machine, its then processed for caching and returned as the output to the results page. The difference in speed depending on if the topic has been searched yet or not is apparent.

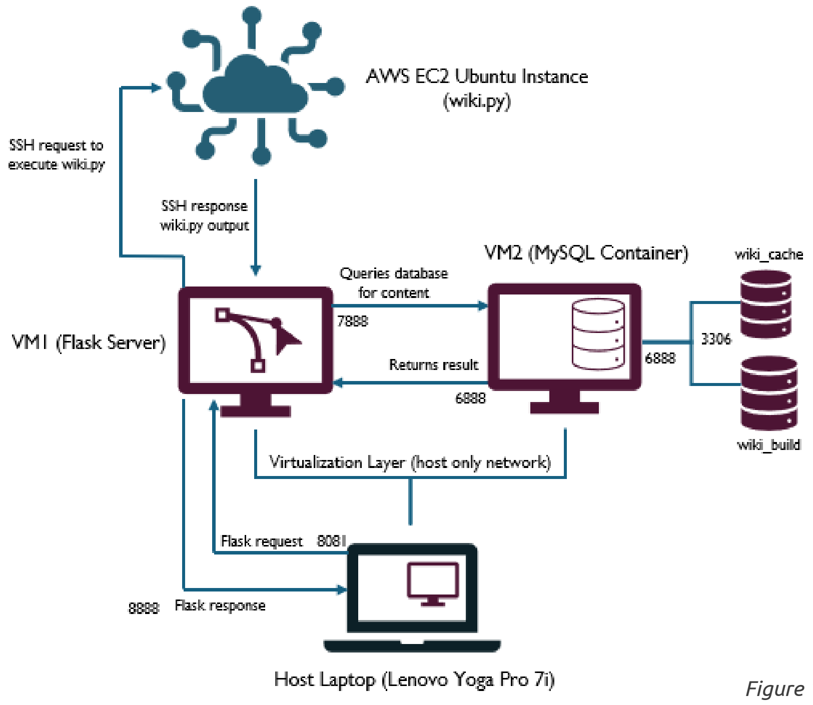

AWS EC2 Ubuntu Instance (wiki.py)
SSH request to execute wiki.py
SSH response wiki.py output
VM2 (MySQL Container)
wiki_cache
Queries database for content
7888
3306
6888
Returns result
6888
wiki_build
VM1 (Flask Server)
Virtualization Layer (host only network)
Flask request 8081
8888 Flask response
Host Laptop (Lenovo Yoga Pro 7i)

Figure 1 : Web application architecture

The user journey starts at http://localhost:8081/ on the search page. The user then types their search term into the text box and clicks '*Submit*'.

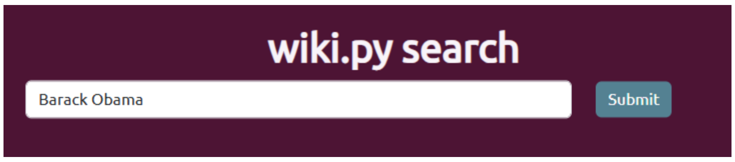
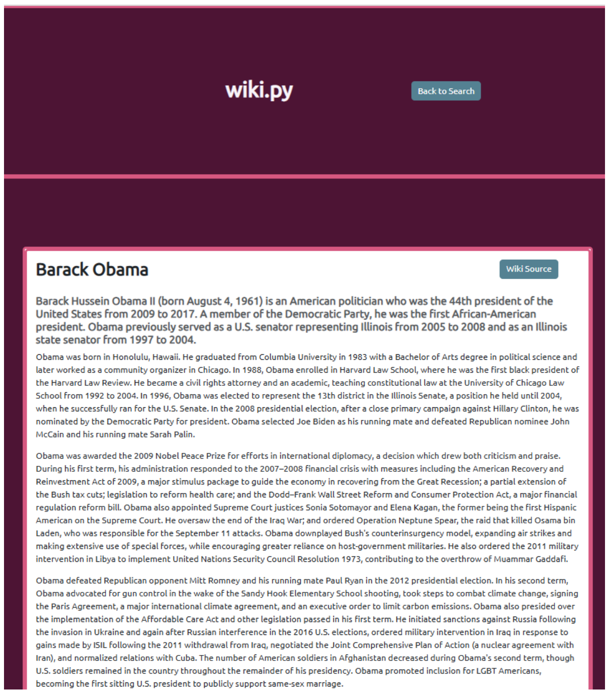


Figure 2 : index (Search Page)



Barack Obama

Barack Hussein Obama II (born August 4, 1961) is an American politician who was the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, he was the first African-American president. Obama previously served as a U.S. senator representing Illinois from 2005 to 2008 and as an Illinois state senator from 1997 to 2004.

Obama was born in Honolulu, Hawaii. He graduated from Columbia University in 1983 with a Bachelor of Arts degree in political science and later worked as a community organizer in Chicago. In 1988, Obama enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. He became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. In 1996, Obama was elected to represent the 13th district in the Illinois Senate, a position he held until 2004, when he successfully ran for the U.S. Senate. In the 2008 presidential election, after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama selected Joe Biden as his running mate and defeated Republican nominee John McCain and his running mate Sarah Palin.

Obama was awarded the 2009 Nobel Peace Prize for efforts in international diplomacy, a decision which drew both criticism and praise. During his first term, his administration responded to the 2007–2008 financial crisis with measures including the American Recovery and Reinvestment Act of 2009, a major stimulus package to guide the economy in recovering from the Great Recession; a partial extension of the Bush tax cuts; legislation to reform health care; and the Dodd–Frank Wall Street Reform and Consumer Protection Act, a major financial regulation reform bill. Obama also appointed Supreme Court justices Sonia Sotomayor and Elena Kagan, the former being the first Hispanic American on the Supreme Court. He oversaw the end of the Iraq War; and ordered Operation Neptune Spear, the raid that killed Osama bin Laden, who was responsible for the September 11 attacks. Obama downplayed Bush's counterinsurgency model, expanding air strikes and making extensive use of special forces, while encouraging greater reliance on host-government militaries. He also ordered the 2011 military intervention in Libya to implement United Nations Security Council Resolution 1973, contributing to the overthrow of Muammar Gaddafi.

Obama defeated Republican opponent Mitt Romney and his running mate Paul Ryan in the 2012 presidential election. In his second term, Obama advocated for gun control in the wake of the Sandy Hook Elementary School shooting, took steps to combat climate change, signing the Paris Agreement, a major international climate agreement, and an executive order to limit carbon emissions. Obama also presided over the implementation of the Affordable Care Act and other legislation passed in his first term. He initiated sanctions against Russia following the invasion in Ukraine and again after Russian interference in the 2016 U.S. elections, ordered military intervention in Iraq in response to gains made by ISIL following the 2011 withdrawal from Iraq, negotiated the Joint Comprehensive Plan of Action (a nuclear agreement with Iran), and normalized relations with Cuba. The number of American soldiers in Afghanistan decreased during Obama's second term, though U.S. soldiers remained in the country throughout the remainder of his presidency. Obama promoted inclusion for LGBT Americans, becoming the first sitting U.S. president to publicly support same-sex marriage.

Figure 3 : Results Page (/getResults)

On submit, the user is directed to http://localhost:8081/getResult. Where the wiki articles associated with the search are displayed within separate bootstrap cards. Subtitles are separated into nested cards within the article. The user can click to return to search page at the top of the viewport, and each wiki article has a button to take the user to the original wiki article.

If the user chose to return to the search page, and search for "Barack Obama" again, they would see a difference in load time of around 12 seconds [A].

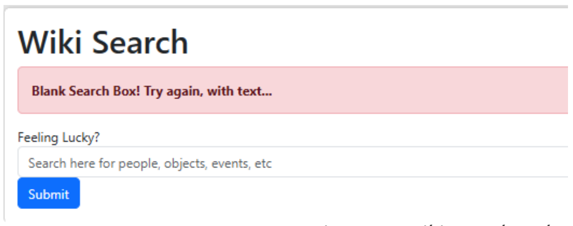
For debugging and refactoring purposes, a function is available via the address bar. ***'/clearCache'*** clears the table which stores the caches wiki articles.

Project tasks

Task 1: Building the web application

To start, I cloned the virtual machine from the final lab which into '*JohnMWebAppProject_FlaskServer*', which would have PyCharm installed and a sample flask project to work from. I then set up the following port forwarding rule [B].

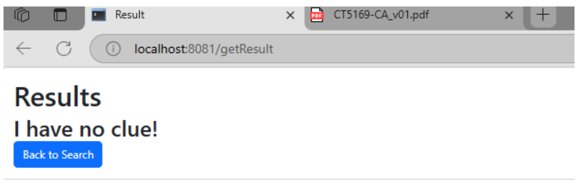
Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Flask	TCP		8081		8888



Blank Search Box! Try again, with text...

Figure 4 : Wiki Search task 1

The html file `'search.html'` is output to the index. The user is shown a simple search form within a bootstrap card, a simple script using JavaScript uses the bootstrap alarm element to let the user know they cannot search a blank term [C].



I have no clue!

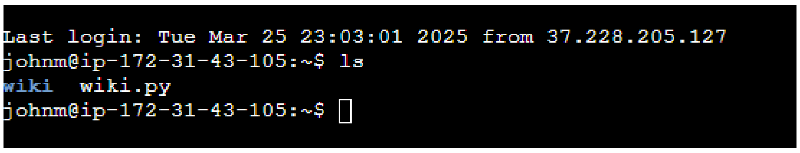
Figure 5 : Results Page task 1

Any term searched brings the user to the same page ***'/getResult'*** where the user is shows another card with no result. The user can click the "Back to Search" button to return to the search page [D].

Task 2: Getting Results from Wikipedia

I created the EC2 instance on AWS running Ubuntu which will be connected to by the flask application. The set up of this instance and the python virtual environment [E] was completed, then the code provided for 'wiki.py' was lifted and implemented unchanged to this instance.

In this figure, you can see the virtual environment directory under 'wiki'. The script which returns the Wikipedia articles is within wiki.py.



```
Last login: Tue Mar 25 23:03:01 2025 from 37.228.205.127
johnm@ip-172-31-43-105:~$ ls
wiki  wiki.py
johnm@ip-172-31-43-105:~$
```

Figure 6 : Instance Console

Using the '*paramiko*' package, the instance is connected to, and two commands are executed one after the other.

"source wiki/wiki_env/bin/activate && python3 wiki.py"

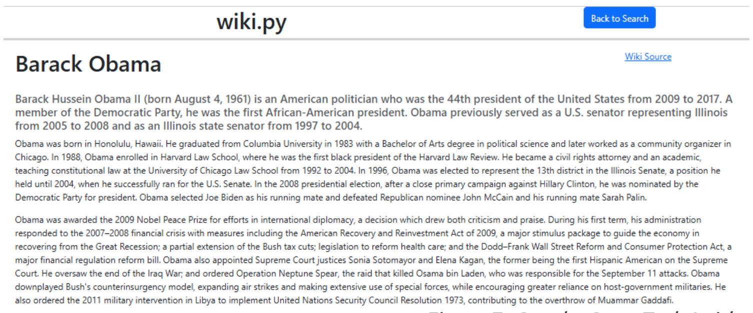
First, the virtual environment is activated, then immediately after, wiki.py is executed. The search term is added when calling '**.exec_command()**' under the paramiko connection.

```python
# Executing wiki.py
stdin, stdout, stderr = client.exec_command(cmd + " " + searchTerm)
```

The output is stored using '**.readlines()'**, the output is then traversed, and a multi-line string is used to write each line of html from the output [F].

```python
for items in output:
    print(items, end="")
    line_raw = items.split('\n')
    line = line_raw[0]
    if not line == '':
        # not including errors
        if not line.startswith("Oops"):
            # if the first line is the title/url/content line
            if line.startswith("title"):
                wiki_htmlView += '''
```

The html is built based on the start of the line. **"title"** flags the beginning of the article, where the line is split into segments so the Title, URL to original wiki page, and beginning content into its own segment. The title also flags that this section should be the start of a new card element.



Barack Obama

Barack Hussein Obama II (born August 4, 1961) is an American politician who was the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, he was the first African-American president. Obama previously served as a U.S. senator representing Illinois from 2005 to 2008 and as an Illinois state senator from 1997 to 2004.

Obama was born in Honolulu, Hawaii. He graduated from Columbia University in 1983 with a Bachelor of Arts degree in political science and later worked as a community organizer in Chicago. In 1988, Obama enrolled in Harvard Law School, where he was the first black president of the Harvard Law Review. He became a civil rights attorney and an academic, teaching constitutional law at the University of Chicago Law School from 1992 to 2004. In 1996, Obama was elected to represent the 13th district in the Illinois Senate, a position he held until 2004, when he successfully ran for the U.S. Senate. In the 2008 presidential election, after a close primary campaign against Hillary Clinton, he was nominated by the Democratic Party for president. Obama selected Joe Biden as his running mate and defeated Republican nominee John McCain and his running mate Sarah Palin.

Obama was awarded the 2009 Nobel Peace Prize for efforts in international diplomacy, a decision which drew both criticism and praise. During his first term, his administration responded to the 2007–2008 financial crisis with measures including the American Recovery and Reinvestment Act of 2009, a major stimulus package to guide the economy in recovering from the Great Recession; a partial extension of the Bush tax cuts; legislation to reform health care; and the Dodd–Frank Wall Street Reform and Consumer Protection Act, a major financial regulation reform bill. Obama also appointed Supreme Court justices Sonia Sotomayor and Elena Kagan, the former being the first Hispanic American on the Supreme Court. He oversaw the end of the Iraq War; and ordered Operation Neptune Spear, the raid that killed Osama bin Laden, who was responsible for the September 11 attacks. Obama downplayed Bush's counterinsurgency model, expanding air strikes and making extensive use of special forces, while encouraging greater reliance on host-government militaries. He also ordered the 2011 military intervention in Libya to implement United Nations Security Council Resolution 1973, contributing to the overthrow of Muammar Gaddafi.

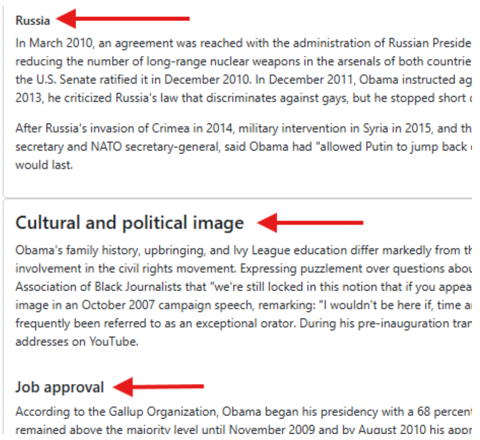
Figure 7 : Results Page Task 2 title

Headers appear within the output in the format of **"==heading=="**. An **"="** is added on both sides for a subheading. I used these delimiters to create the subtitles of the articles.

```python
# if next line is subtitle, e.g., "==sub==
elif line.startswith("=="):
    subTitle = line.split("=")
    if len(subTitle) == 5:
        wiki_htmlView += '</div></div><div class="card"><div class="card-body container"><h4 class="card-
title mb-2">' + \
                         subTitle[2] + '</h4>'
    elif len(subTitle) == 7:
        wiki_htmlView += '<br><h5 class="card-subtitle mb-2">' + subTitle[3] + '</h5>'
    elif len(subTitle) == 9:
        wiki_htmlView += '<br><h6 class="card-subtitle mb-2">' + subTitle[4] + '</h6>'
```

The headings format as in figure 8. The heading under "*Cultural and Political Image*" ends the previous card and begins the next.

Each new article also closes the previous card opened when the output starts with "title" as seen in figure 9.


Russia

In March 2010, an agreement was reached with the administration of Russian Preside
reducing the number of long-range nuclear weapons in the arsenals of both countrie
the U.S. Senate ratified it in December 2010. In December 2011, Obama instructed ag
2013, he criticized Russia's law that discriminates against gays, but he stopped short

After Russia's invasion of Crimea in 2014, military intervention in Syria in 2015, and th
secretary and NATO secretary-general, said Obama had "allowed Putin to jump back
would last.

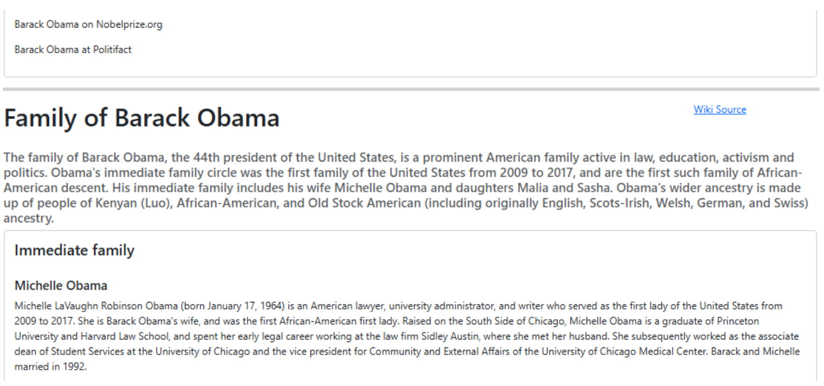
Cultural and political image

Obama's family history, upbringing, and Ivy League education differ markedly from th
involvement in the civil rights movement. Expressing puzzlement over questions abou
Association of Black Journalists that "we're still locked in this notion that if you appea
image in an October 2007 campaign speech, remarking: "I wouldn't be here if, time a
frequently been referred to as an exceptional orator. During his pre-inauguration tran
addresses on YouTube.

Job approval

According to the Gallup Organization, Obama began his presidency with a 68 percent
remained above the majority level until November 2009 and by August 2010 his appr

Figure 8 : Subtitle


Barack Obama on Nobelprize.org

Barack Obama at Politifact

Family of Barack Obama

Wiki Source

The family of Barack Obama, the 44th president of the United States, is a prominent American family active in law, education, activism and politics. Obama's immediate family circle was the first family of the United States from 2009 to 2017, and are the first such family of African-American descent. His immediate family includes his wife Michelle Obama and daughters Malia and Sasha. Obama's wider ancestry is made up of people of Kenyan (Luo), African-American, and Old Stock American (including originally English, Scots-Irish, Welsh, German, and Swiss) ancestry.

Immediate family

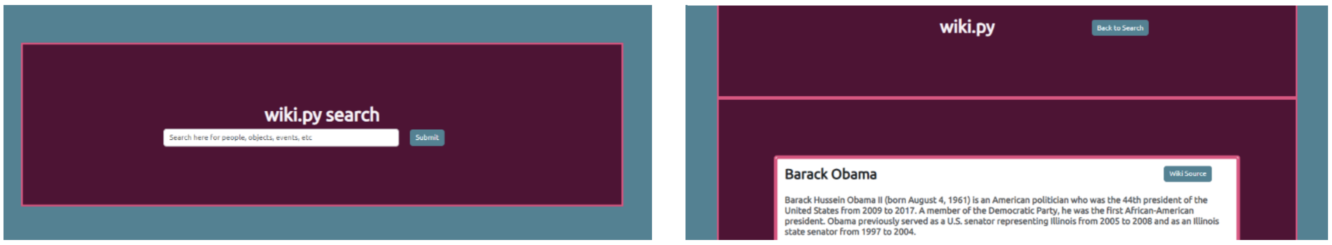
Michelle Obama

Michelle LaVaughn Robinson Obama (born January 17, 1964) is an American lawyer, university administrator, and writer who served as the first lady of the United States from 2009 to 2017. She is Barack Obama's wife, and was the first African-American first lady. Raised on the South Side of Chicago, Michelle Obama is a graduate of Princeton University and Harvard Law School, and spent her early legal career working at the law firm Sidley Austin, where she met her husband. She subsequently worked as the associate dean of Student Services at the University of Chicago and the vice president for Community and External Affairs of the University of Chicago Medical Center. Barack and Michelle married in 1992.

Figure 9 : New article

Task 3: Caching searches for future use

When completing this task, the application was refactored. I implemented CSS styling to the search page and to the results page.



Barack Hussein Obama II (born August 4, 1961) is an American politician who was the 44th president of the United States from 2009 to 2017. A member of the Democratic Party, he was the first African-American president. Obama previously served as a U.S. senator representing Illinois from 2005 to 2008 and as an Illinois state senator from 1997 to 2004.

Figure 10 : CSS styled pages

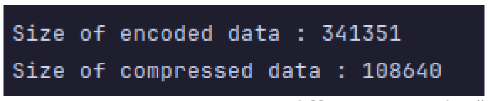
The /getResult algorithm was then broken into separate functions.

- **`get_wiki(searched):`** Connects to the EC2 instance and returns the Wikipedia content for the argument "`searched`".
- **`build_html(content, search):`** Builds the html based on the content argument which would be the Wikipedia articles retrieved from '`get_wiki()`'. The search term is also passed as an argument for the title of the html page. The html is also processed at the last step of the function and stored in the MySQL server.

Two new functions were then added. The first, '**`process_output(direction, data)`**'. The purpose of which is to encode then compress or decompress then decode the built html passed through '`data`'.

This is done to decrease the size of the html stored within the MySQL database table, which has a '`var char`' of max size 255 characters as the primary key identifier. Inserting and selecting the compressed data is done via the type '*`mediumblob`*' [G].

The ‘direction’ argument (either “out” or “in”) specifies which direction the data is being processed.



```
Size of encoded data : 341351
Size of compressed data : 108640
```

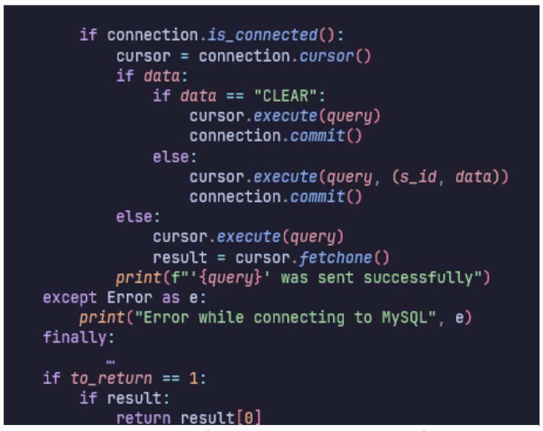
Figure 11 : Size difference using len()

The package ‘`zlib`’ is used to compress/decompress the data. Before compressing, the html is encoded line by line to a bytearray variable. Encoding is done via ‘utf-8’ so the html tags can be safely processed.

```
if direction == "in":
    # data must be decompressed, then decoded
    data_decompressed = zlib.decompress(data)
    data_decoded = data_decompressed.decode('utf-
8')
    return data_decoded
```

```
elif direction == "out":
    data_encoded = bytearray()
    for line in data:
        data_encoded.extend(line.encode('utf-8'))
    data_compressed = zlib.compress(data_encoded)
    return data_compressed
```

The second new function **‘`send_query(query,s_id,data,to_return)`’** [H] implemented the **‘`mysql.connector`’** package as was guided within the Lab 7 practical. This function connects to the SQL database and executes the given query. ‘`s_id`’ is used when inserting or selecting html data with the primary key, and ‘`data`’ is used to if data is being sent, otherwise it is left blank. If data is being returned, ‘`to_return`’ is set to 1, otherwise it is set to 0.



```
        if connection.is_connected():
            cursor = connection.cursor()
            if data:
                if data == "CLEAR":
                    cursor.execute(query)
                    connection.commit()
                else:
                    cursor.execute(query, (s_id, data))
                    connection.commit()
            else:
                cursor.execute(query)
                result = cursor.fetchone()
            print(f"'{query}' was sent successfully")
    except Error as e:
        print("Error while connecting to MySQL", e)
    finally:
            ...
    if to_return == 1:
        if result:
            return result[0]
```

Figure 12 : MySQL Function Structure

The flow of ‘/getResult’ now amounts to an if/else statement.

```
def getResult():
    search_term = request.form.get('wSearch')
    query = f"SELECT content_compress FROM {cache_table} WHERE search_term='" + search_term + "';"
    content = send_query(query, search_term, "", 1)

    if content:  # If content is not None (content was found)
        html_processed = process_output("in", content)
        return html_processed
    else:  # If content is not found, call wiki.py and store output
        content = get_wiki(search_term)
        return build_html(content, search_term)
```

Personal Challenges

The challenges I experienced formed during the implementation of task 3. I experienced difficulties when attempting to connect to the MySQL server.

At first, I mistakenly used port 7888 within the function to send queries when the port I was attempting to access was 6888 through port 7888. Then storing the data to a MySQL format caused difficulties. When first attempting to store the multi-line string, the format was not passing as an argument.

I then attempted to encode/decode with base64, which corrupted and changed the html output. Online guides and Stack overflow posts from developers with similar issues aided me when completing this task

Appendices

Appendix A : Build vs Cache time

Using the python package “***timeit***”, as suggested by this stack overflow post, time is tracked from the start of ‘/getResult’ as such:

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

def getResult():

    tic=timeit.default_timer()

Then the time is taken again at the end of the ‘build_html()’ function if the term is new or before returning the cache if the term is not new.

#within build_html

toc=timeit.default_timer()

print('Time to execute wiki build in seconds: '+str(toc - tic))

return html_build

#within /getResult

html_processed = process_output("in", content)

toc = timeit.default_timer()

print(f'Time to execute wiki build in seconds: '+str(toc-tic))

return html_processed

For searching “Barack Obama”, both processing times are as follows

Process | Execution time Seconds

wiki.py | 12.16530144

wiki_cache | 0.141495223

Appendix B : Port Forwarding

Port forwarding rules outlined here at Flask's official documentation and VirtualBox’s port forwarding guide here.

Appendix C : Bootstrap Elements

Bootstrap importing via CDN. | Documentation for bootstrap elements.

Alarm system for blank search lifted from my final assignment for *CT5167 Cloud Web Application Development*. When **‘onsubmit’** is set to return false, the page does not refresh and does not search:

```
<script>
        function checkSearch() {
            //Event Handler for search error
            let searchItem = document.getElementById("wSearch").value;
            //check to see if textbox is empty
            if(searchItem === "") {
                //if empty, page does not refresh onclick, and the bootstrap alert is triggered
                document.getElementById("searchAction").setAttribute("onsubmit","return false")
                document.getElementById("formatError").innerHTML = "<div class=\"alert alert-
danger\" role=\"alert\">\n" +
                    "<b>" + "Blank Search Box! Try again, with text..." + "\n" + "</b>" +
                    "</div>";
            } else {
                //if not empty, onsubmit is potentially removed (if it had already been triggered
                //then the action and method is added so we can get the result
                document.getElementById("searchAction").removeAttribute("onsubmit")
                document.getElementById("searchAction").setAttribute("action","/getResult")
                document.getElementById("searchAction").setAttribute("method","post")
            }
        }
    </script>
```

Appendix D : Hardcoded /getResult

HTML output for **'/getResult'** is hard coded and returns the same output each time.

```
@app.route('/getResult', methods=['POST'])
def getResult():
    resultOutput = '''
        <!DOCTYPE html>
        <html lang="en">
            <head>
                <meta charset="UTF-8">
                <title>Result</title>
                <!--Bootstrap Framework CDN-->
                <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet"
                    integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
                <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
                    integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
            </head>
            <body>
                <div class="card">
                    <div class="card-body">
                        <h1 class="card-title">Results</h1>
                        <h2 class="card-subtitle">I have no clue!</h2>
                        <a href="/" class="btn btn-primary">Back to Search</a>
                    </div>
                </div>
            </body>
        </html>'''
    return resultOutput
```

Appendix E : EC2 Setup Guides

Instance set up. | Venv set up.

Appendix F : Task 2 /getResult

Task 2 '/getResult' full implementation:

```
@app.route('/getResult', methods=['POST'])
def getResult():
    # Public IPv4 address on EC2 connect console, needs to be reviewed each time instance is stopped
and started
    instance_ip = "13.60.42.115"
    securityKeyFile = "/home/johnm/DistWebAppProj.pem"
    searchTerm = request.form.get('wSearch')
    cmd = "source wiki/wiki_env/bin/activate && python3 wiki.py"  # Activating venv before executing
wiki.py
```

```
    try:
        # Attempt to connect/ssh to the instance
        client = paramiko.SSHClient()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        key = paramiko.RSAKey.from_private_key_file(securityKeyFile)
        client.connect(hostname=instance_ip, username="johnm", pkey=key)

        # Executing wiki.py
        stdin, stdout, stderr = client.exec_command(cmd + " " + searchTerm)
        stdin.close()
        outerr = stderr.readlines()
        print("ERRORS: ", outerr)
        output = stdout.readlines()

        # html doc to return
        wiki_htmlView = '''
                    <!DOCTYPE html>
                    <html lang="en">
                    <head>
                        <meta charset="UTF-8">
                        <title>Results for ''' + searchTerm + '''</title>
                        <!--Bootstrap Framework CDN-->
                        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
                            integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
                        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
                            integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
                                crossorigin="anonymous"></script>
                    </head>
                    <body>
                        <div class="card">
                            <div class="container text-center">
                                <div class="row">
                                    <div class="col-8">
                                        <h1>wiki.py</h1>
                                    </div>
                                    <div class="col-4">
                                        <a href="/" class="btn btn-primary">Back to Search</a>
                                    </div>
                                </div>
                                <div>
                '''
        # print result to the console (
        print("output: ")
        for items in output:
            print(items, end="")
            line_raw = items.split('\n')
            line = line_raw[0]
            if not line == '':
                # not including errors
                if not line.startswith("Oops"):
                    # if the first line is the title/url/content line
                    if line.startswith("title"):
                        wiki_htmlView += '''</div></div></div></div></div><div class="card">
                                        <div class="card">
                                            <div class="card-body container">
                                                <div class="row">
                                                    <div class="col-10">
                                                        <h1 class="card-title">'''
                        # Split first line into title, url and content for title of card
                        title1 = line.split("title:", 1)
                        title2 = title1[1].split("URL:", 1)
                        title3 = title2[1].split("Content:", 1)
                        wiki_htmlView += title2[0] + '</h1>' + \
                                            '</div>' + \
                                            '<div class=col><a href="' + title3[0] + '">Wiki
Source</a></div></div>' + \
                                            '<br><h5 class="card-subtitle mb-2 text-muted">' + title3[
                                                1] + '</h5><div><div>'
                    # if next line is subtitle, e.g., "==sub==
                    elif line.startswith("=="):
                        subTitle = line.split("=")
                        if len(subTitle) == 5:
                            wiki_htmlView += '</div></div><div class="card"><div class="card-body
container"><h4 class="card-title mb-2">' + \
                                                subTitle[2] + '</h4>'
                        elif len(subTitle) == 7:
                            wiki_htmlView += '<br><h5 class="card-subtitle mb-2">' + subTitle[3] +
'</h5>'
                        elif len(subTitle) == 9:
                            wiki_htmlView += '<br><h6 class="card-subtitle mb-2">' + subTitle[4] +
'</h6>'
                    # if standard text
                    else:
                        wiki_htmlView += '<p>' + \
                                            line + \
                                            '</p>'
        # end html
        wiki_htmlView += ''' </div>
                        </div>
                    </div>
                </div>
            </body>
            </html>
        '''
        # Close client connection when finished
        client.close()
        return wiki_htmlView
    except Exception as e:
        print(e)
        return render_template('results.html', searchTerm=searchTerm,
                               results="Something went wrong, please try again...")
```

Appendix G : Encoding Process

Stack overflow posts : encoding, base64, blobs

Appendix H : send_query() implementation

```
# Function to submit query to db, data is data to be sent or blank if receiving. to_return
should be 0 if sending, 1 if retrieving

def send_query(query, s_id, data, to_return):

    result = None

    try:

        connection = mysql.connector.connect(host=wiki_mysql_ip,

                                             port='6888',

                                             database=db,

                                             user='root',

                                             password='mypassword')



        if connection.is_connected():
```

```
        cursor = connection.cursor()


        if data:
            if data == "CLEAR":
                cursor.execute(query)
                connection.commit()


            else:
                cursor.execute(query, (s_id, data))
                connection.commit()


        else:
            cursor.execute(query)
            result = cursor.fetchone()


        print(f"'{query}' was sent successfully")


except Error as e:
    print("Error while connecting to MySQL", e)


finally:
    if connection.is_connected():
        cursor.close()
        connection.close()


if to_return == 1:
    if result:
        return result[0]
```