MARMARA UNIVERSITY

FACULTY OF ENGINEERING

CSE3055 Database Systems

Project Step - 4

Database Management System for GT Stone - Web Interface

150119766 Emre Sağıroğlu (GrRep)

150119816 Ömür Tarık Gökkurt

150119066 Ertan Karaoğlu

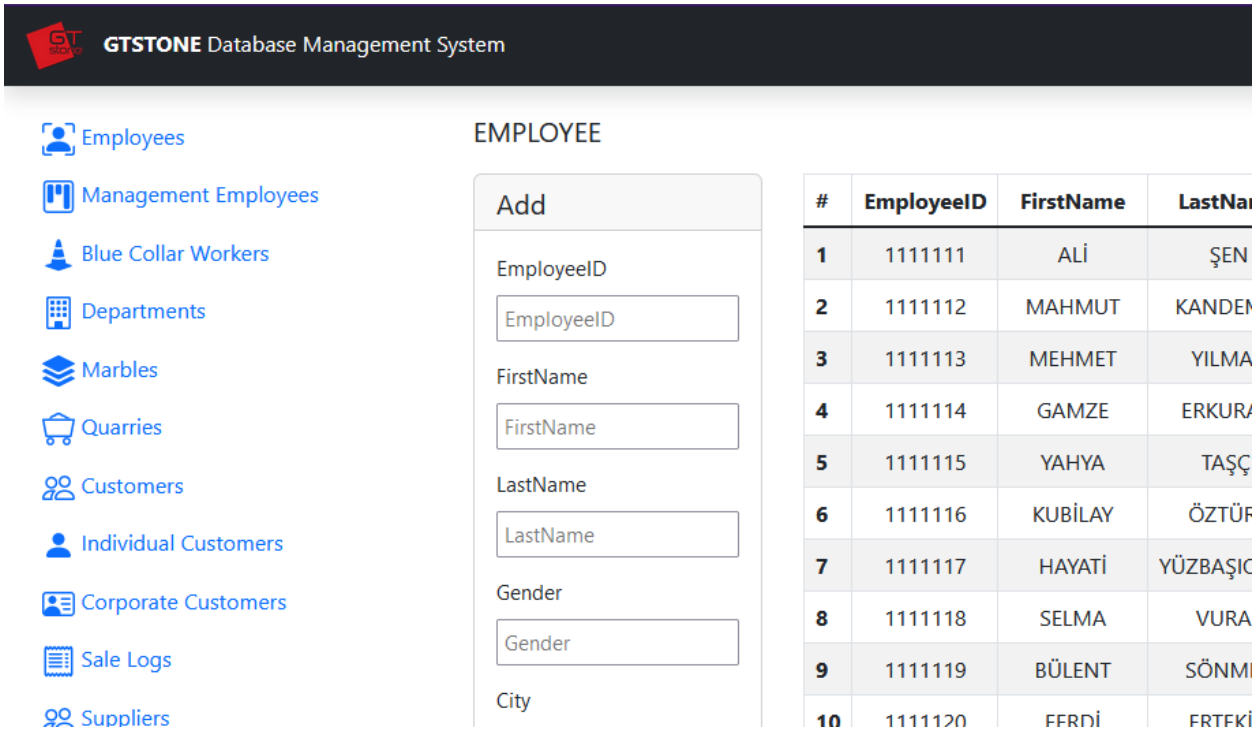
ABOUT THE PROJECT

This web interface is created using **Python** 3.10 with libraries **_thread**, **socket** and **pyodbc**. Threading and socket libraries come pre-installed with Python.

You need to install **pyodbc** library to run the script. Pyodbc library is used for database connection. It allows us to run queries given as a string.

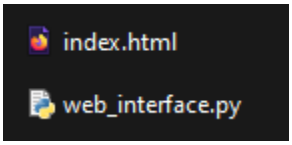
```
>> pip install pyodbc
```

There is an HTML file **index.html** which contains the main interface that wraps the content created from the script. **Bootstrap** is used to stylize HTML elements.



HOW TO RUN

There are two files in the project directory. After installing pyodbc, run script **web_interface.py**



After running the script, you will be prompted to enter the database server name:

```
Enter server name (You can find server name by running query 'SELECT @@SERVERNAME')
```

You can run the following query to find your server name, copy the result and paste to the prompt:

```
Select @@SERVERNAME
```

	(No column name)
1	DESKTOP-U4QR8K0\MSSQLSERVER01

```
Enter server name (You can find server name by running query 'SELECT @@SERVERNAME')
DESKTOP-U4QR8K0\MSSQLSERVER01
Socket binded to port 1234
Listening on port 1234...
```

Server is running, you can access the web interface by entering **localhost:1234** to your browser. You can change the port number by editing the following code in **web_interface.py**

```
host = '0.0.0.0'
port = 1234  # This
```

HOW IT WORKS

Python script uses socket programming to receive and send appropriate responses to the client who connects through the browser.

We create the server socket here and listen for incoming connections, after a connection is established, we run the **threaded** function which contains the main logic. Threaded approach is used to serve multiple users simultaneously.

```
def main():
    host = '0.0.0.0'
    port = 1234  # This

    server_socket= socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((host, port))
    print("Socket binded to port", port)

    while True:
        server_socket.listen(5)
        print('Listening on port %s...' % port)
        client_socket, client_addr = server_socket.accept()
        start_new_thread(threaded, (client_socket, client_addr))
```

HTML file is loaded and divided in two parts, any content will go in between these two parts. There is a “CUT_HERE” string inside the HTML file to indicate where to divide:

```
index = open("index.html", 'r')

parts = index.read().split("<!--CUT_HERE-->")
HTML_BEGIN, HTML_END = parts[0], parts[2]
```

Here we connect the database with **pyodbc**. **connection.cursor()** object is what we use to execute queries:

```
# Establishing connection to SQL server with pyodbc library:
connection = pyodbc.connect('Driver={SQL Server};Server=%s;Database=GTSTONE;Trusted_Connection=yes;' % server_name)

# Cursor instance is what we use to execute queries:
cursor = connection.cursor()
```

There are two other functions **create_table()** and **create_form()**. They wrap the information returned from select queries and create HTML elements that will be shown in the web interface.

```
def create_table(columns, info):
    # Creates HTML <table> from the SQL output:

    html = """<div style="float:left"><table class="table table-bordered
    table-striped table-hover"><thead><tr><th scope="col">#</th>"""

    for c in columns:
        html += """<th style="text-align:center;" scope="col">%s</th> """ %
```

In the threaded() function which is the main handler for the user, there are 5 different outcomes determined by if-else statements. The GET request created from the browser's address bar is checked with these if-else statements.

First outcome is landing page, when user enters localhost:1234/

```
if request == "/":  # If landing page:
```

Second outcome is if the user clicks on links in the sidebar with the table names. For example if the user clicks on Marbles, **http://localhost:1234/table=marble** will be seen in the address bar and the request will be **/table=marble.**

```
elif "table=" in request:
```

Third outcome is for Views of the database. Request will look like **/view=MonthlySales**

```
elif "view=" in request:
```

Fourth outcome is to insert a new entry to the table, which is created from the <form> HTML element in the web interface. Request will look like:
/Addmarble?MarbleID=9&MarbleName=SKY_BLUE_ROYAL&Color=BLUE&InStock=Y&AmountOfStock=300&UnitPrice=99

```
elif "Add" in request:
```

Fifth outcome is else statement which will be executed when an invalid URL is entered:

```
else: # Invalid URL, responding with "Not Found":
```

As an example, if user requests for a table:

First, a <h5> HTML element is created for the title of the page that will be returned at the end. Table name extracted from the request is assigned to the **table** variable. Then we execute select-all query with this table name using the **cursor** object. We extract column names from **cursor.description**.

```
title = """<h5 style="margin-bottom:20px; margin-top:20px;">%s</h5>""" % request.split("table=")[1].upper()
table = request.split("table=")[1]
cursor.execute('SELECT * FROM %s' % table)
columns = []
for i in cursor.description:
    columns.append(i[0])
```

Then we create <form> and <table> elements using the output of the select query. Then we add together all the HTML code, create the HTTP response and respond to the client:

```
form_element = create_form(table.upper, table, columns, "Add")

info = cursor.fetchall()
html = create_table(columns, info)
html = HTML_BEGIN + title + form_element + html + HTML_END
content_length = "Content-Length: " + str(len(html))
http_response = HTTP_VER + STATUS_OK + '\n' + content_length + "\n\n" + html
client.sendall(http_response.encode())
```

It works similarly for Add and View requests.