**Faculty of Engineering, Applied Sciences, and Technology**

###### BSC 206 – Information Structures

###### Spring 2023 – Project

|  |  |
| --- | --- |
| Course Code: BSC 206 | Course Name: Information Structures |
| Date: March 27, 2023 | Time: 12:00 – 1:30 |
| Location: Hub 114 | Instructor(s): Dr. Said Elnaffar |
| Number of Students: 30 | Number of Pages: 4 |

|  |  |
| --- | --- |
| **Course Learning Outcomes** | **Quizzes /Assignments** |
| **CLO1.** Specify the appropriate data structures to solve specific data-representation and algorithmic problems. | **1** |
| **CLO2.** Design complex algorithms including trees, queues, maps and hash tables, stacks, and graphs. | **2** |
| **CLO3.** Implement queues, dictionary, sorting and text processing algorithms | **3** |
| **CLO4.** Analyze the complexity of an algorithm. | **4** |

**Team Members**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # | ID | Name | Report (5) | Code (5) | Individual Q&A (10) | Total (20) |
| 1 | 20210001705 | Yadidiah Kanaparthi | 0 | 0 | 0 | 0 |
| 2 | 20210001483 | Denver Dias | 0 | 0 | 0 | 0 |
| 3 | 20210001826 | Majed Alzenati | 0 | 0 | 0 | 0 |
| 4 | 20210001593 | Abdulrahman Mashnouq | 0 | 0 | 0 | 0 |
| 5 |  |  | 0 | 0 | 0 | 0 |

**BREAKDOWN OF GRADE**

|  |  |  |
| --- | --- | --- |
| **Q&A** | **Code** | **Report** |
| **CLO 1** | **CLO 2** | **CLO 3** |

# Project Description and Requirements

You are asked to build a real-life business app that maintains information about the menu of a restaurant. Business requirements are as follows:

* Build a database-oriented application for the menu of a restaurant. This menu consists of several meals. Each meal has a name, price, type (drink or sandwich) and the comprising list of ingredients.
* Front-End UI: using *PySimpleGui*
* Store and manage information about the restaurant menu.
* App functions must include, but not limited to:
  + Insert a meal in the menu.
  + List meals in the menu.
  + Delete a meal.
  + Update a meal (including the list of ingredients).
  + Search with filters (e.g., meals whose price is less than a value)
  + Save/Load to/from a file. (Bonus)
  + Quit
* Justify the data structures you use in this project.
* Your implementation should seamlessly hide the data structure underneath.
* Bonus: The data could be saved to and loaded from files (JSON, text files, etc.)
* Your app should practice high quality of coding principles using Object-Oriented programming.
* You are allowed to do it in groups of 5.

# Rubric

|  |  |  |  |
| --- | --- | --- | --- |
| Criterion | Needs Improvement | Good | Excellent |
| Q&A (Individual) | [0-3]  The student provided poor answers and insufficient evidence of contribution to the assignment. | [4-6]  The student answered some of the questions correctly and showed some contribution to the assignment. | [7-10]  The student answered all questions correctly and showed genuine contribution to the assignment. |
| Coding & Demo (Group) | [0-3]  The code does not work properly, and many requirements are missing. | [4-6]  The code works partially and implements some of the requirements. | [7-10]  The code works fully and as expected per the requirements with high quality and engineering. |
| Report (Group) | [0-1]  The report is poor, does not provide enough description of how the code works, and missing comments in code. | [2-3]  The report gives some description of the project, lists the code, and partially documents it. Some run scenarios are demonstrated by full screenshots. | [4-5]  The report gives good description of the project, lists the code, and documents it very well. All run scenarios are demonstrated by full screenshots. |

# Submission Instructions

1. Fill out the table on the front page with the names of the team members.
2. Attach your report to the end of this document (copy and paste).
3. **Only one submission per group.**

Graphical user interface, text

Description automatically generated

Graphical user interface, text

Description automatically generated

Sql code example:INSERT INTO meals (meal\_id, name, price, type, ingredients) VALUES (27, 'Portobello Mushroom Sandwich', 12.99, 'sandwich', 'Whole Wheat Bun, Grilled Portobello Mushroom, Tomato, Arugula, Balsamic Glaze');

import mysql.connector

import PySimpleGUI as sg

db=mysql.connector.connect(

    host="localhost",

    user="root",

    passwd="yadidiah",

    database="restrauntmenu"

)

myCursor = db.cursor()

sg.theme('DarkTeal3')

def insert\_meal(id,meal\_name,meal\_price,meal\_type,meal\_ingredients):

    query ="INSERT INTO meals(meal\_id, name, price, type, ingredients) VALUES({}, '{}', {}, '{}', '{}')".format(id,meal\_name,meal\_price,meal\_type,meal\_ingredients)

    myCursor.execute(query)

    db.commit()

    print(myCursor.rowcount, "Record inserted successfully into meals table")

def delete\_meal(id: int):

    query = "DELETE FROM meals WHERE meal\_id={}".format(id)

    myCursor.execute(query)

    print(myCursor.rowcount, "record(s) deleted from meals table")

    db.commit()

def update\_meal(id,change,ans):

    query = "UPDATE meals SET {} = {} WHERE meal\_id={}".format(change,ans,id)

    myCursor.execute(query)

    print(f"{change} has been changed for meal with id {id}")

    db.commit()

def filter\_search(filters):

    query = "SELECT \* FROM meals WHERE "

    filter\_queries = []

    for key, value in filters.items():

        if value is not None and value != "":

            if isinstance(value, str):

                if value[0] =="<":

                    val = int(value[1:])

                    filter\_queries.append(f"{key}<{val}")

                elif value[0] ==">":

                    val = int(value[1:])

                    filter\_queries.append(f"{key}>{val}")

                elif value[0] =="=":

                    val = int(value[1:])

                    filter\_queries.append(f"{key}={val}")

                else:

                    filter\_queries.append(f"{key} LIKE '%{value}%'")

            else:

                filter\_queries.append(f"{key}={value}")

    query += " AND ".join(filter\_queries)

    myCursor.execute(query)

    data = myCursor.fetchall()

    return data

myCursor.execute('SELECT \* FROM meals')

data = myCursor.fetchall()

meal\_options = [row[0] for row in myCursor.fetchall()]

options = ['id', 'name', 'price','type','ingredients']

layout = [

        [[sg.Table(key="table",values=data, headings=[i[0] for i in myCursor.description], max\_col\_width=75, auto\_size\_columns=True, justification='center')]],

        [sg.Text("Enter:")],

        [sg.Button("Insert Meal")],

        [sg.Text("Enter Meal ID (Number):",key='text1', visible=False), sg.Input(key='input1', visible=False),

        sg.Text('Enter Meal Name:',key='text2', visible=False), sg.InputText(key='input2', visible=False)],

        [sg.Text('Enter Meal Price (Number):',key='text3', visible=False), sg.InputText(key='input3', visible=False),

        sg.Text('Enter Meal Type',key='text4', visible=False), sg.InputText(key='input4', visible=False)],

        [sg.Text('Enter All Ingredients(add spaces before next value):',key='text5', visible=False), sg.InputText(key='input5', visible=False),

        sg.Button("Submit", key="submit",visible=False),

        sg.Button("Delete Meal")],

        [sg.Text("Enter the id to delete:", key ="delete\_id", visible=False),

        sg.Input(key='delete\_input', visible=False)],

        [sg.Button("Update Meal",key="update")],

        [sg.Text("Enter the id to Update:", key ="update\_id", visible=False),sg.Input(key='update\_input', visible=False)],

        [sg.Text("Enter what to change (add space if multiple):", key ="change", visible=False),sg.Input(key='change\_input', visible=False)],

        [sg.Text("Enter the changed values (add space if multiple):", key ="value", visible=False),sg.Input(key='value\_input', visible=False)],

        [sg.Button("Filter")],

        [sg.Listbox(values=options,key="filter\_box" ,size=(20, len(options)), select\_mode='multiple',visible=False)],

        [sg.Text("Enter filter:", key ="filter", visible=False),sg.Input(key='filter\_input', visible=False)],

        [sg.Button('Enter',key="enter")],

        [sg.Button("QUIT")]

        ]

window = sg.Window('Select your meals', layout)

# read the window for events

while True:

    event, values = window.read()

    if event == sg.WINDOW\_CLOSED:

        break

    if event == 'Insert Meal':

        # set the visibility of the four text and input fields to True

        window['text1'].update(visible=True)

        window['input1'].update(visible=True)

        window['text2'].update(visible=True)

        window['input2'].update(visible=True)

        window['text3'].update(visible=True)

        window['input3'].update(visible=True)

        window['text4'].update(visible=True)

        window['input4'].update(visible=True)

        window['text5'].update(visible=True)

        window['input5'].update(visible=True)

        window["submit"].update(visible=True)

    if event== 'submit':

            insert\_meal(values["input1"],values["input2"],values["input3"],values["input4"],values["input5"])

    elif event=='Delete Meal':

        window['delete\_id'].update(visible=True)

        window['delete\_input'].update(visible=True)

        if values["delete\_input"]:

            var = int(values["delete\_input"])

            delete\_meal(var)

        else:

            print("Please enter a valid meal ID.")

    elif event=="update":

        window['update\_id'].update(visible=True)

        window['update\_input'].update(visible=True)

        window['change'].update(visible=True)

        window['change\_input'].update(visible=True)

        window['value'].update(visible=True)

        window['value\_input'].update(visible=True)

        change = values["change\_input"].split()

        value = values["value\_input"].split()

        for i in range(len(change)):

            update\_meal(values["update\_input"],change[i],value[i])

    elif event=="Filter":

        window["filter\_box"].update(visible=True)

        window["filter"].update(visible=True)

        window["filter\_input"].update(visible=True)

        window["enter"].update(visible=True)

    if event== 'enter':

        list1 = values["filter\_input"].split()

        dict\_values={}

        for i in range(len(values["filter\_box"])):

            dict\_values[values["filter\_box"][i]]=list1[i]

        data = filter\_search(dict\_values)

        if data:

            window["table"].update(values=data,)

        else:

            print("No data found.")

    elif event == 'QUIT':

        sg.popup('GOODBYE! 😁 ')

        break

window.close()

db.close()

Text

Description automatically generated

Text

Description automatically generated

Used data structures = lists, tuples, dictionaries.

Complex data structures were not used as the data was already being stored in a sql database.