Student Number: 20077401

Student Name: Rupesh Varma Hasthi

Course: MSc in Cyber security

Lecturer Name: Swathi Dongre

Module/Subject Title: Advanced Programming Techniques

Assignment Title: CA_ONE_(30%)

No of Words:1570

Repository Link: https://github.com/rupeshhasthi/advprogca.git

Table of Contents

1. Introduction

This report documents the design and implementation of four programming tasks completed using **C#** and **Python**. The assessment demonstrates competencies in object-oriented programming, client–server communication, database integration, data scraping, and structured data processing.

2. Question 1 – C# Contact Book Application

2.1 Overview

The goal was to build a **menu-driven contact book system** allowing users to add, update, view, and delete contact information.
The complete solution is implemented in **Program.cs in Que1 folder**.

The system is preloaded with 20 predefined contacts. Generative AI was used to create 20 random contacts with First Name, Last Name, Company, Mobile Number, Email and Birth date details.

2.2 Data Validation

- Mobile number is validated based on following factors:
  - Exactly **9 digits**
  - **Non-zero**, non-negative and numeric input
  - Rejects invalid formats using exception handling
- Email address is validated using built-in .NET parser.

2.3 Functionality

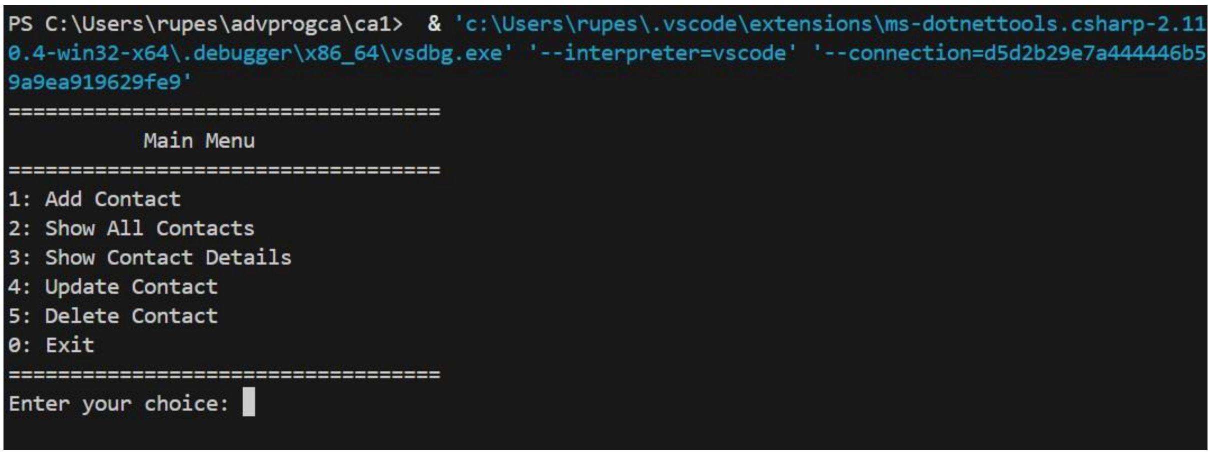
The system includes:

1. **Add Contact** – validates and stores new contact
2. **Show All Contacts** – lists contact names + mobile numbers
3. **Show Contact Details** – displays full structured record
4. **Update Contact** – modifies only selected fields
5. **Delete Contact** – removes contact from the list

The program handles invalid input using try-catch.

2.4 Execution with Screenshots

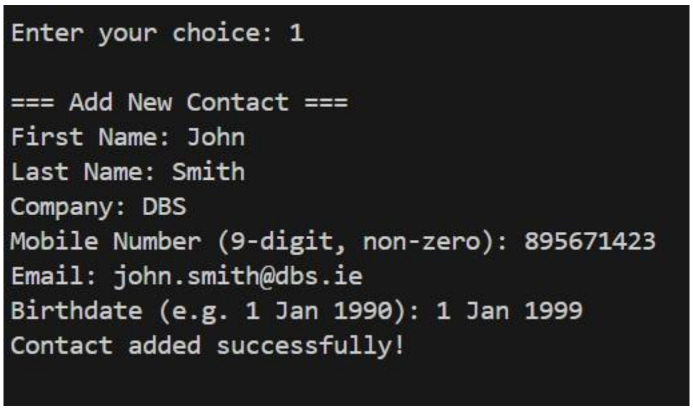
After executing the program, console displays Menu with all options.



```
PS C:\Users\rupes\advprogca\ca1>  & 'c:\Users\rupes\.vscode\extensions\ms-dotnettools.csharp-2.11
0.4-win32-x64\.debugger\x86_64\vsdbg.exe' '--interpreter=vscode' '--connection=d5d2b29e7a444446b5
9a9ea919629fe9'
===================================
           Main Menu
===================================
1: Add Contact
2: Show All Contacts
3: Show Contact Details
4: Update Contact
5: Delete Contact
0: Exit
===================================
Enter your choice: 
```

Figure 1 : Displays Menu when code executed

If we enter 1 which is to Add Contact, console asks us all required details of the contact we want to add.

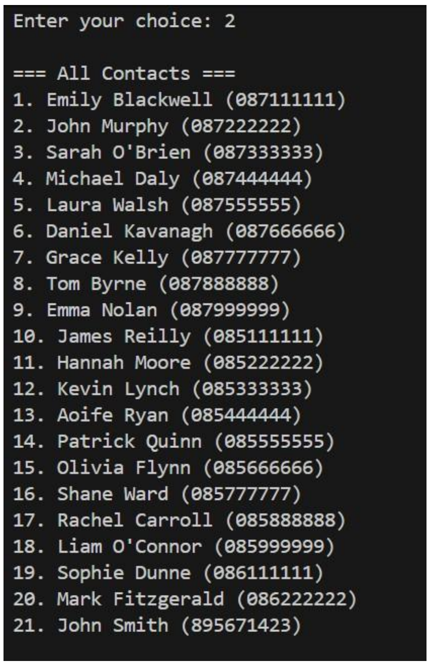


```
Enter your choice: 1

=== Add New Contact ===
First Name: John
Last Name: Smith
Company: DBS
Mobile Number (9-digit, non-zero): 895671423
Email: john.smith@dbs.ie
Birthdate (e.g. 1 Jan 1990): 1 Jan 1999
Contact added successfully!
```

Figure 2 : Add contact

After entering all requested details, program displays Contact added successfully! and asks for new choice. For Mobile Number and Email, validations are implemented which will be discussed below.

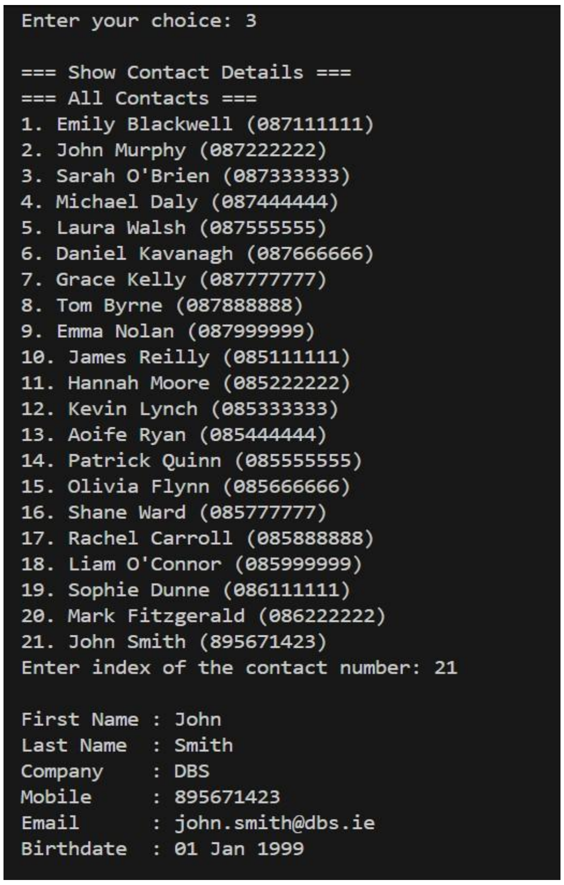
If we enter 2 which is to Show All Contacts, console displays all contacts and asks for new choice.



```
Enter your choice: 2

=== All Contacts ===
1. Emily Blackwell (087111111)
2. John Murphy (087222222)
3. Sarah O'Brien (087333333)
4. Michael Daly (087444444)
5. Laura Walsh (087555555)
6. Daniel Kavanagh (087666666)
7. Grace Kelly (087777777)
8. Tom Byrne (087888888)
9. Emma Nolan (087999999)
10. James Reilly (085111111)
11. Hannah Moore (085222222)
12. Kevin Lynch (085333333)
13. Aoife Ryan (085444444)
14. Patrick Quinn (085555555)
15. Olivia Flynn (085666666)
16. Shane Ward (085777777)
17. Rachel Carroll (085888888)
18. Liam O'Connor (085999999)
19. Sophie Dunne (086111111)
20. Mark Fitzgerald (086222222)
21. John Smith (895671423)
```

Figure 3 : Shows All Contacts

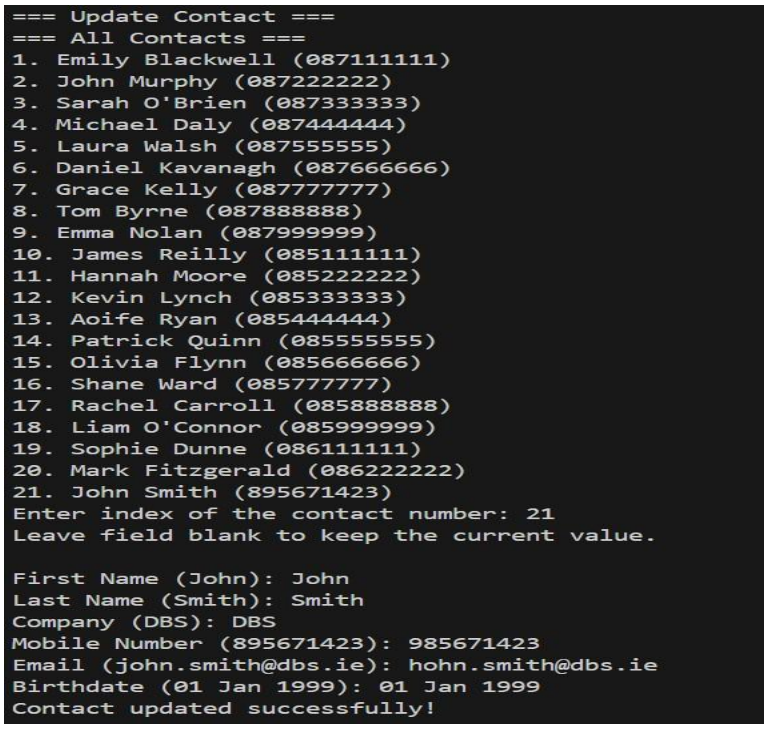
If we choose option 3, console prompts to select index of contact which we want to view. If correct index number is entered, contact details are displayed.



```
Enter your choice: 3

=== Show Contact Details ===
=== All Contacts ===
1. Emily Blackwell (087111111)
2. John Murphy (087222222)
3. Sarah O'Brien (087333333)
4. Michael Daly (087444444)
5. Laura Walsh (087555555)
6. Daniel Kavanagh (087666666)
7. Grace Kelly (087777777)
8. Tom Byrne (087888888)
9. Emma Nolan (087999999)
10. James Reilly (085111111)
11. Hannah Moore (085222222)
12. Kevin Lynch (085333333)
13. Aoife Ryan (085444444)
14. Patrick Quinn (085555555)
15. Olivia Flynn (085666666)
16. Shane Ward (085777777)
17. Rachel Carroll (085888888)
18. Liam O'Connor (085999999)
19. Sophie Dunne (086111111)
20. Mark Fitzgerald (086222222)
21. John Smith (895671423)
Enter index of the contact number: 21

First Name : John
Last Name  : Smith
Company    : DBS
Mobile     : 895671423
Email      : john.smith@dbs.ie
Birthdate  : 01 Jan 1999
```

Figure 4 : Show Contact details

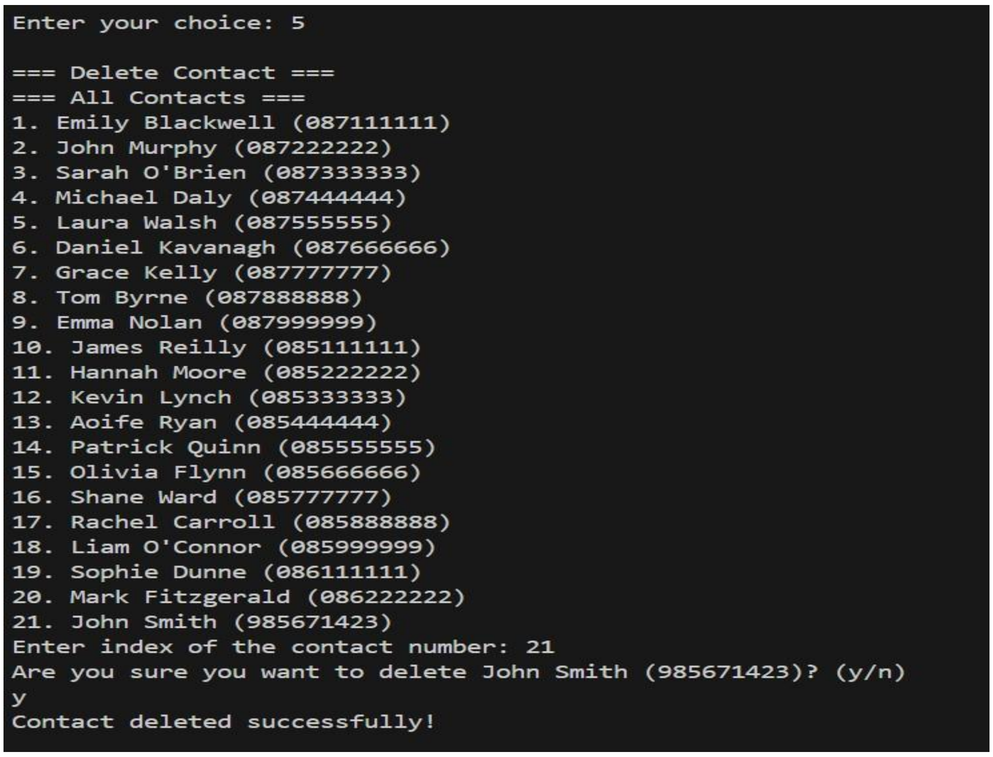
If we choose Option 4, console prompts to select index of contact which we want to update. If correct index number is entered, console asks to enter all contact details.



```
=== Update Contact ===
=== All Contacts ===
1. Emily Blackwell (087111111)
2. John Murphy (087222222)
3. Sarah O'Brien (087333333)
4. Michael Daly (087444444)
5. Laura Walsh (087555555)
6. Daniel Kavanagh (087666666)
7. Grace Kelly (087777777)
8. Tom Byrne (087888888)
9. Emma Nolan (087999999)
10. James Reilly (085111111)
11. Hannah Moore (085222222)
12. Kevin Lynch (085333333)
13. Aoife Ryan (085444444)
14. Patrick Quinn (085555555)
15. Olivia Flynn (085666666)
16. Shane Ward (085777777)
17. Rachel Carroll (085888888)
18. Liam O'Connor (085999999)
19. Sophie Dunne (086111111)
20. Mark Fitzgerald (086222222)
21. John Smith (895671423)
Enter index of the contact number: 21
Leave field blank to keep the current value.

First Name (John): John
Last Name (Smith): Smith
Company (DBS): DBS
Mobile Number (895671423): 985671423
Email (john.smith@dbs.ie): hohn.smith@dbs.ie
Birthdate (01 Jan 1999): 01 Jan 1999
Contact updated successfully!
```

Figure 5 : Update Contact

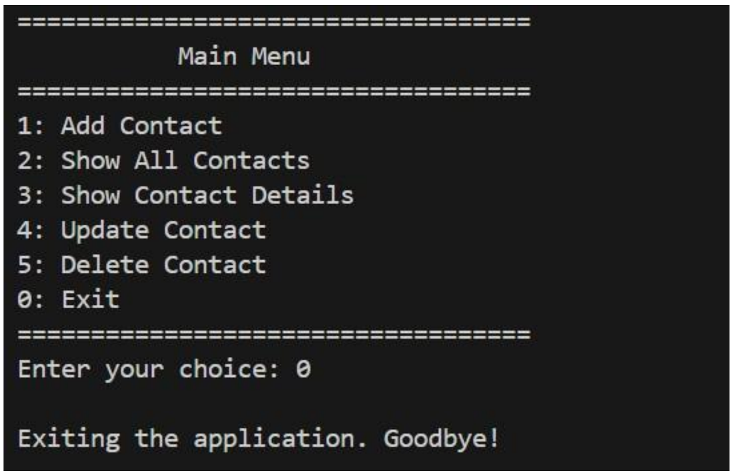
If we choose Option 5, console prompts to select index of contact which we want to delete. If correct index number is entered, console asks to confirm the deletion. Once confirmed, contact was deleted.



```
Enter your choice: 5

=== Delete Contact ===
=== All Contacts ===
1. Emily Blackwell (087111111)
2. John Murphy (087222222)
3. Sarah O'Brien (087333333)
4. Michael Daly (087444444)
5. Laura Walsh (087555555)
6. Daniel Kavanagh (087666666)
7. Grace Kelly (087777777)
8. Tom Byrne (087888888)
9. Emma Nolan (087999999)
10. James Reilly (085111111)
11. Hannah Moore (085222222)
12. Kevin Lynch (085333333)
13. Aoife Ryan (085444444)
14. Patrick Quinn (085555555)
15. Olivia Flynn (085666666)
16. Shane Ward (085777777)
17. Rachel Carroll (085888888)
18. Liam O'Connor (085999999)
19. Sophie Dunne (086111111)
20. Mark Fitzgerald (086222222)
21. John Smith (985671423)
Enter index of the contact number: 21
Are you sure you want to delete John Smith (985671423)? (y/n)
y
Contact deleted successfully!
```

Figure 6 : Delete Contact

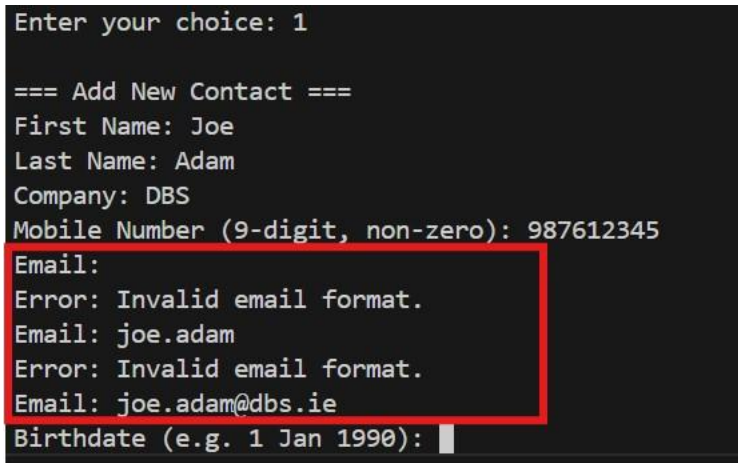
If we choose Option 0, program will exit.



```
===================================
           Main Menu
===================================
1: Add Contact
2: Show All Contacts
3: Show Contact Details
4: Update Contact
5: Delete Contact
0: Exit
===================================
Enter your choice: 0

Exiting the application. Goodbye!
```

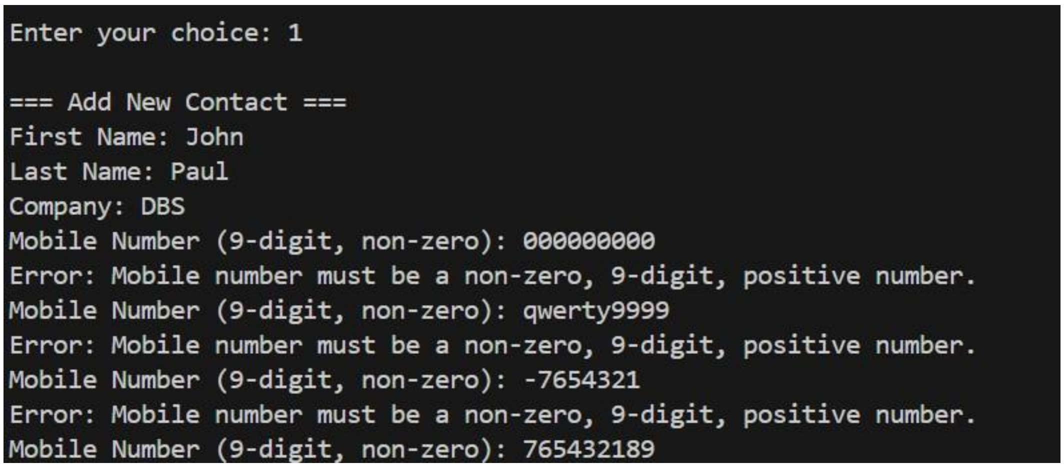
Figure 7 : Exit Program

Additionally Email validation and Mobile number validation are implemented.



```
Enter your choice: 1

=== Add New Contact ===
First Name: Joe
Last Name: Adam
Company: DBS
Mobile Number (9-digit, non-zero): 987612345
Email:
Error: Invalid email format.
Email: joe.adam
Error: Invalid email format.
Email: joe.adam@dbs.ie
Birthdate (e.g. 1 Jan 1990):
```

Figure 8 : Email validation



```
Enter your choice: 1

=== Add New Contact ===
First Name: John
Last Name: Paul
Company: DBS
Mobile Number (9-digit, non-zero): 000000000
Error: Mobile number must be a non-zero, 9-digit, positive number.
Mobile Number (9-digit, non-zero): qwerty9999
Error: Mobile number must be a non-zero, 9-digit, positive number.
Mobile Number (9-digit, non-zero): -7654321
Error: Mobile number must be a non-zero, 9-digit, positive number.
Mobile Number (9-digit, non-zero): 765432189
```

Figure 9 : Mobile number validation

3. Question 2 – File Extension Information System in C#

3.1 Overview

This program allows users to query information about at least **20 file extensions**, implemented in **Program.cs** in folder **Que2**.

The system supports commands:

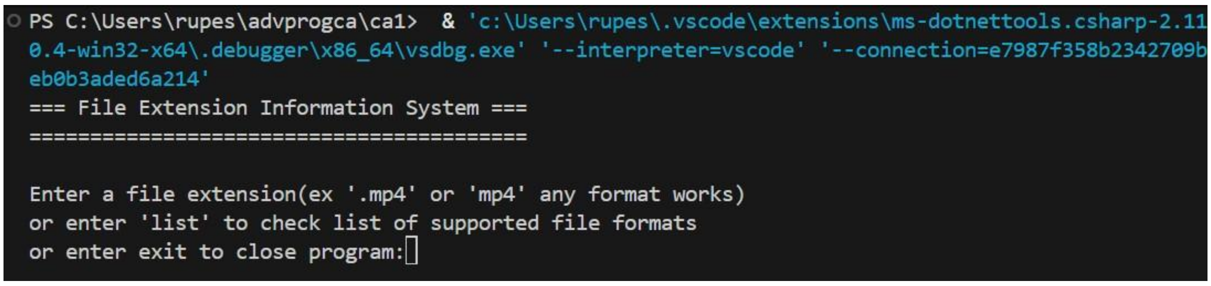
- Enter extension → show description
- list → show all supported extensions
- exit → quit program

Program implements a file-extension lookup system where users enter an extension, the program returns its meaning. It uses a **dictionary** as the main data structure, allowing key-value retrieval of file extensions and descriptions.

Generative AI was used to create 20 random file format names and information.

3.2 Execution with Screenshots

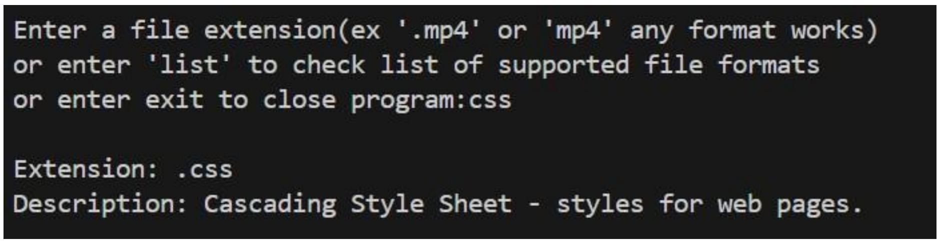
After executing the program, console displays to enter file format we want to know about or to enter list to know supported file formats or to enter exit to close program. User can enter file extension with or without dot. For example, we can enter format as ".mp4" or "mp4". Program checks for dot in front of file extension. If dot is present information will be provided. If not present, a dot will be appended and checked for file extension information.



```
PS C:\Users\rupes\advprogca\ca1>  & 'c:\Users\rupes\.vscode\extensions\ms-dotnettools.csharp-2.11
0.4-win32-x64\.debugger\x86_64\vsdbg.exe' '--interpreter=vscode' '--connection=e7987f358b2342709b
eb0b3aded6a214'
=== File Extension Information System ===
=========================================

Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:
```

Figure 10 : Asks to enter file extension

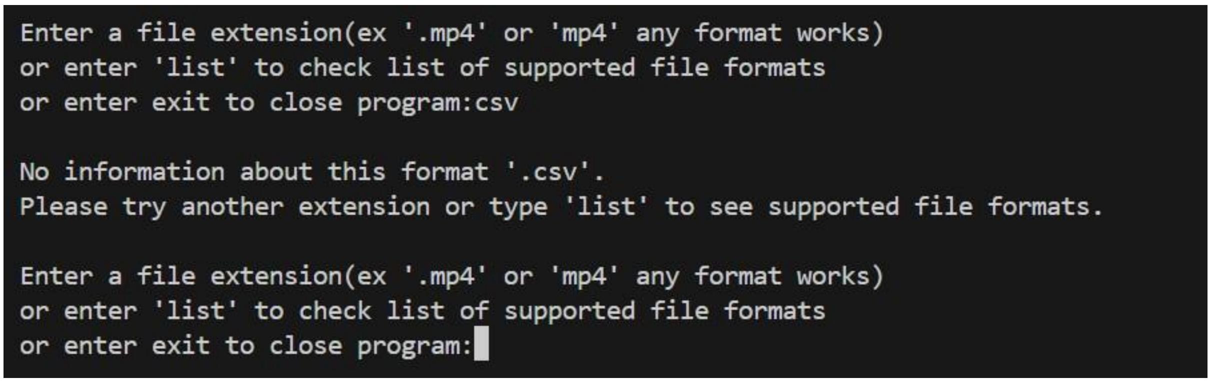
I have entered "css" as file extension and information about css file extension is provided.



```
Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:css

Extension: .css
Description: Cascading Style Sheet - styles for web pages.
```

Figure 11 : Information about extension

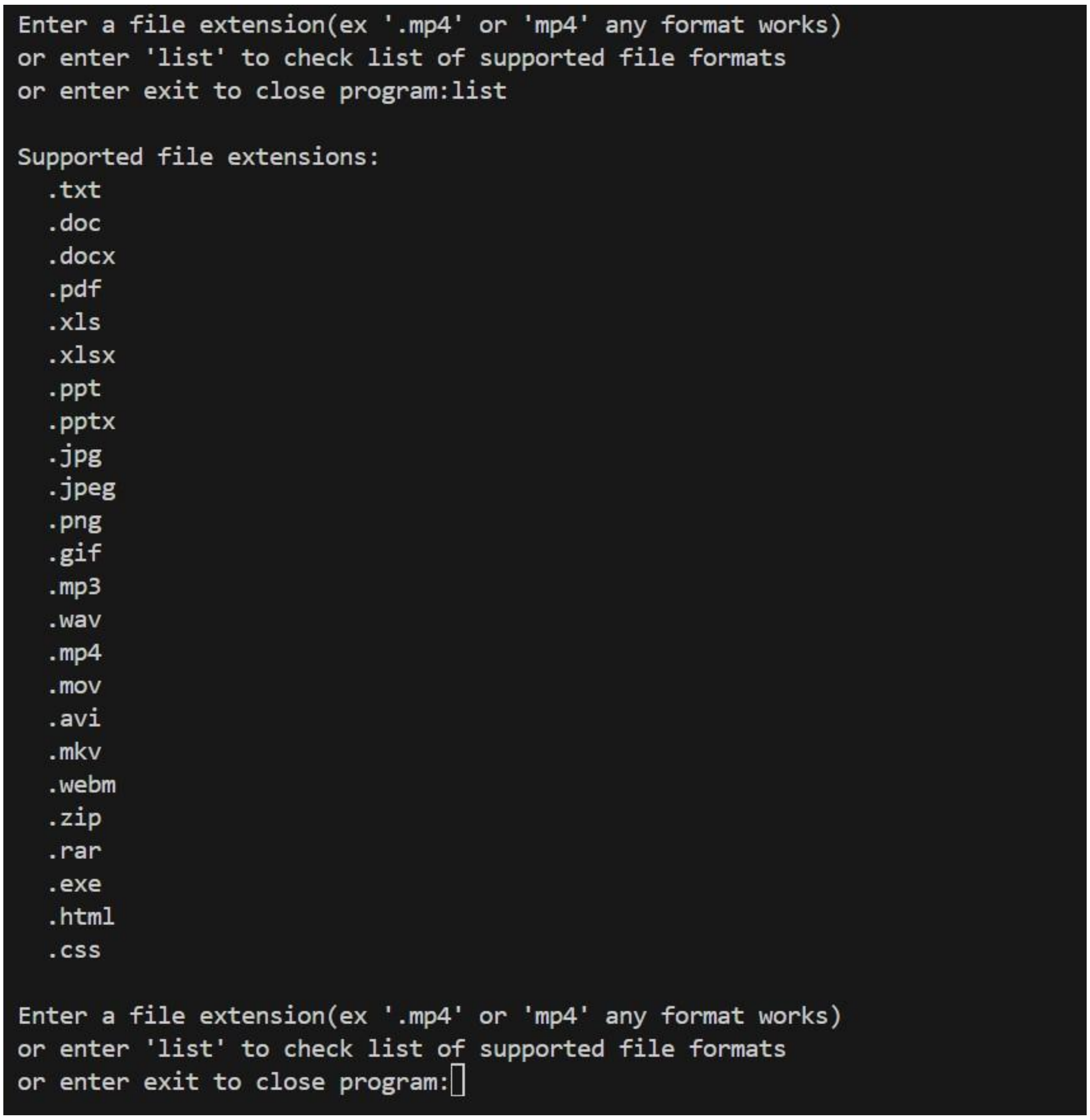
If we enter a format which is not in our program, console displays no information about entered file extension and asks to type list to know supported file formats.



```
Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:csv

No information about this format '.csv'.
Please try another extension or type 'list' to see supported file formats.

Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:
```

Figure 12 : No information about entered file format

If we enter list, all supported file extension details are displayed.



```
Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:list

Supported file extensions:
  .txt
  .doc
  .docx
  .pdf
  .xls
  .xlsx
  .ppt
  .pptx
  .jpg
  .jpeg
  .png
  .gif
  .mp3
  .wav
  .mp4
  .mov
  .avi
  .mkv
  .webm
  .zip
  .rar
  .exe
  .html
  .css

Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:
```

Figure 13 : Supported file extensions

If we enter exit, program will exit.

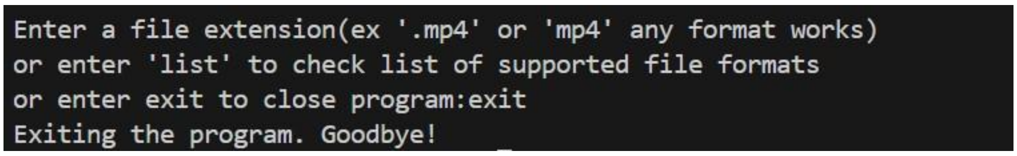

```
Enter a file extension(ex '.mp4' or 'mp4' any format works)
or enter 'list' to check list of supported file formats
or enter exit to close program:exit
Exiting the program. Goodbye!
```

Figure 14 : Exiting Program

4. Question 3 – DBS Admission System (Client–Server in Python)

4.1 Overview

This task required developing a **client and server**, storing applicant data in a database, and returning a unique registration number.

A **TCP-based client–server admission system** is implemented. The client asks for applicant information and sends to the server using **JSON format**. The server receives the JSON string, decodes it, validates the fields, and then stores the applicant's details in a **SQLite database (dbs_admissions.db)** using parameterized SQL queries. After inserting the record, the server generates a **unique registration number**, saves it, and sends it back to the client as a JSON response.

Files implemented:

- **Que3_client.py** – console-based input client
- **Que3_server.py** – TCP server with DB integration
- **dbs_admissions.db** – SQLite database

4.2 Server Functionality (Que3_server.py)

- Accepts TCP connection
- Receives JSON data from client
- Validates:
    - Name, address, qualifications
    - Course selection from **3 valid choices**
    - Year and month
- Inserts record into SQLite database using parameterized queries
- Generates registration number format: **DBS-YYYY-DDDDDD(DBS-Year-6 Digit number)**
- Sends confirmation back to the client

4.3 Client Functionality (Que3_client.py)

- Prompts user for all fields
- Ensures valid input types

- Transmits data to server using JSON
- Displays the registration number returned by the server

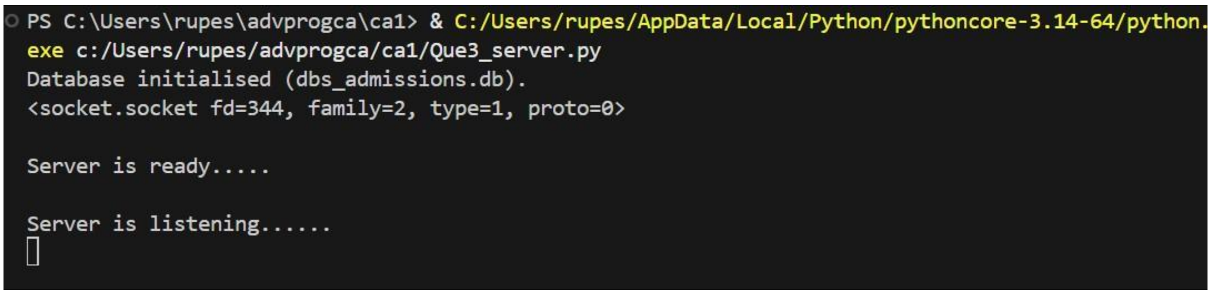
4.4 Database Design

Table: **applications**

Field	**Purpose**
id	Auto-increment primary key
name	Applicant name
address	Applicant address
qualifications	Education info
course	Selected MSc course
start_year	Intended start year
start_month	Intended start month

4.5 Execution with Screenshots

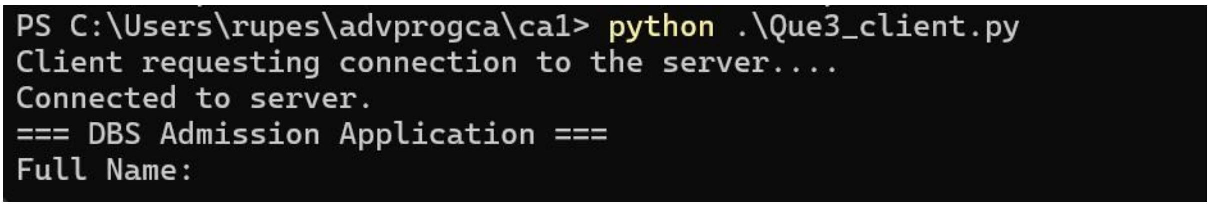
First **server (Que3_server.py)** was executed as it waits for incoming client connections. Server will be listening for connection for 100 seconds.



```
PS C:\Users\rupes\advprogca\ca1> & C:/Users/rupes/AppData/Local/Python/pythoncore-3.14-64/python.
exe c:/Users/rupes/advprogca/ca1/Que3_server.py
Database initialised (dbs_admissions.db).
<socket.socket fd=344, family=2, type=1, proto=0>

Server is ready.....

Server is listening......
```

Figure 15 : Server listening

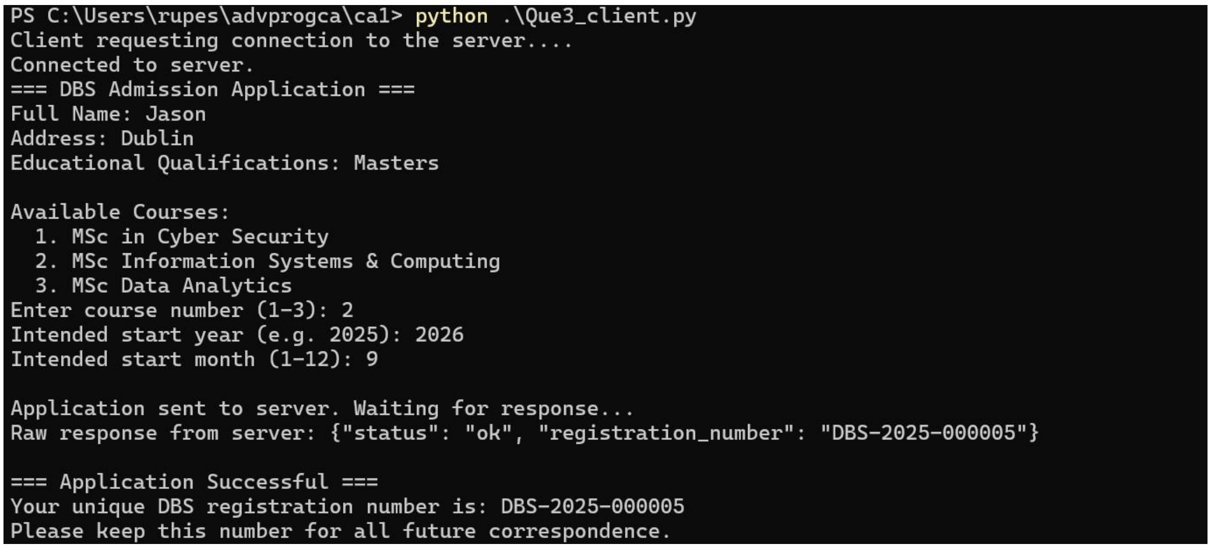
After the server starts, the **client (Que3_client.py)** is executed from a different terminal.



```
PS C:\Users\rupes\advprogca\ca1> python .\Que3_client.py
Client requesting connection to the server....
Connected to server.
=== DBS Admission Application ===
Full Name:
```

Figure 16 : Client Program connecting to server after execution

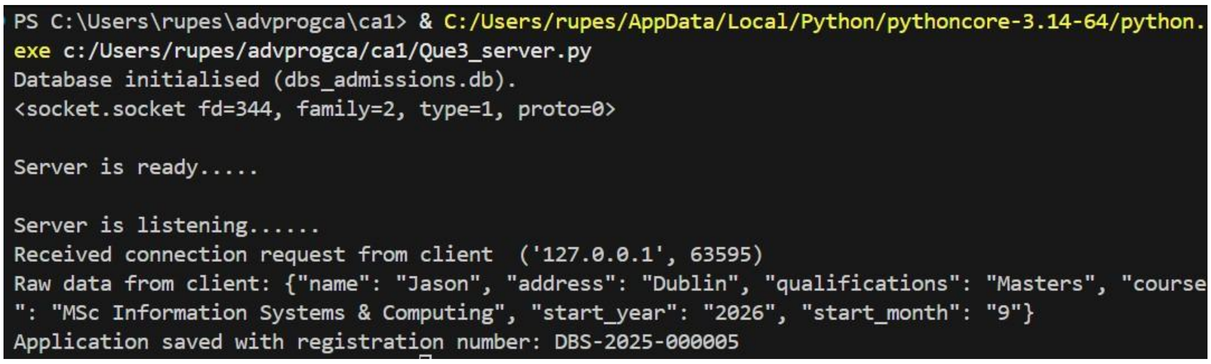
Client program requests applicant to enter details such as name, address, qualifications, course, and intended start date.



```
PS C:\Users\rupes\advprogca\ca1> python .\Que3_client.py
Client requesting connection to the server....
Connected to server.
=== DBS Admission Application ===
Full Name: Jason
Address: Dublin
Educational Qualifications: Masters

Available Courses:
  1. MSc in Cyber Security
  2. MSc Information Systems & Computing
  3. MSc Data Analytics
Enter course number (1-3): 2
Intended start year (e.g. 2025): 2026
Intended start month (1-12): 9

Application sent to server. Waiting for response...
Raw response from server: {"status": "ok", "registration_number": "DBS-2025-000005"}

=== Application Successful ===
Your unique DBS registration number is: DBS-2025-000005
Please keep this number for all future correspondence.
```

Figure 17 : Client asking for Applicant Details

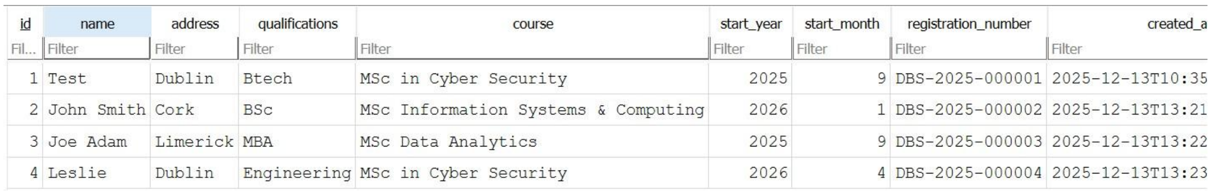
After successfully entering all details, unique registration number was generated and same registration number can be seen in the server program as well. Also we can see in Json data format.



```
PS C:\Users\rupes\advprogca\ca1> & C:/Users/rupes/AppData/Local/Python/pythoncore-3.14-64/python.
exe c:/Users/rupes/advprogca/ca1/Que3_server.py
Database initialised (dbs_admissions.db).
<socket.socket fd=344, family=2, type=1, proto=0>

Server is ready.....

Server is listening......
Received connection request from client  ('127.0.0.1', 63595)
Raw data from client: {"name": "Jason", "address": "Dublin", "qualifications": "Masters", "course
": "MSc Information Systems & Computing", "start_year": "2026", "start_month": "9"}
Application saved with registration number: DBS-2025-000005
```

Figure 18 : Server receiving all details from client

After registration number is generated, dbs_admissions.db is created for first entry and this file is updated after every new entry.



id	name	address	qualifications	course	start_year	start_month	registration_number	created_a
Fil...	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	Test	Dublin	Btech	MSc in Cyber Security	2025	9	DBS-2025-000001	2025-12-13T10:35
2	John Smith	Cork	BSc	MSc Information Systems & Computing	2026	1	DBS-2025-000002	2025-12-13T13:21
3	Joe Adam	Limerick	MBA	MSc Data Analytics	2025	9	DBS-2025-000003	2025-12-13T13:22
4	Leslie	Dublin	Engineering	MSc in Cyber Security	2026	4	DBS-2025-000004	2025-12-13T13:23

Figure 19 : dbs_admissions.db entries

5. Question 4 – Python Web Scraping & CSV Processing

5.1 Overview

This task extracts hotel room pricing data from **two separate HTML webpages**, consolidates them, stores the results in CSV, and displays them.
Implementation is in:

- **hotel1.html**
- **hotel2.html**
- **Que4_scraper.py**
- **hotel_prices.csv**

As no well-known hotel websites were allowing scraping, created two basic html pages hotel1.html and hotel2.html with minimum deatils

5.2 Source HTML Pages

Each page includes:

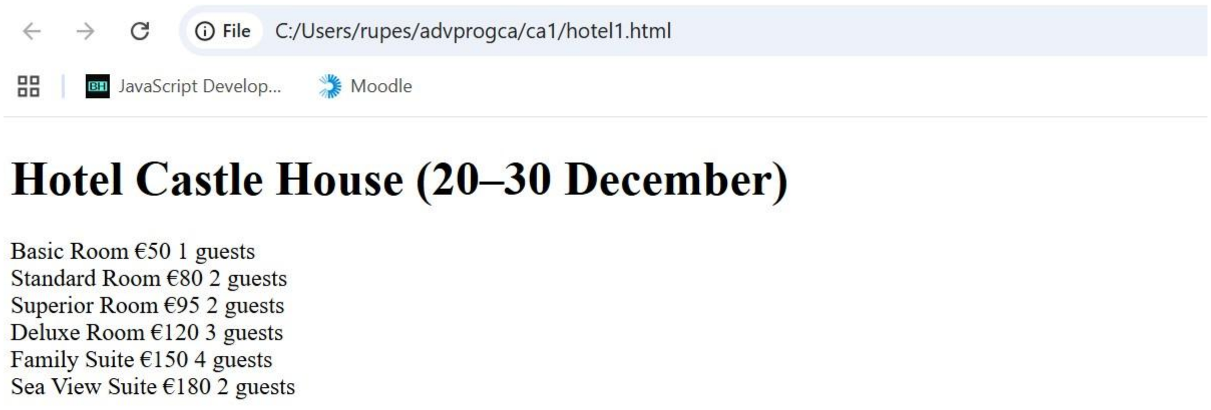
- Hotel name
- Minimum 5 room listings each (as atleast **10 rooms information should be gathered**)
- Fields for:
    - Room name
    - Price
    - Capacity

5.3 Scraper Logic (BeautifulSoup)

The script:

1. Opens both HTML files
2. Uses BeautifulSoup to parse .room-card elements
3. Extracts:
    - Room name
    - Price
    - Capacity
    - Hotel name
4. Adds them to a Python list of dictionaries

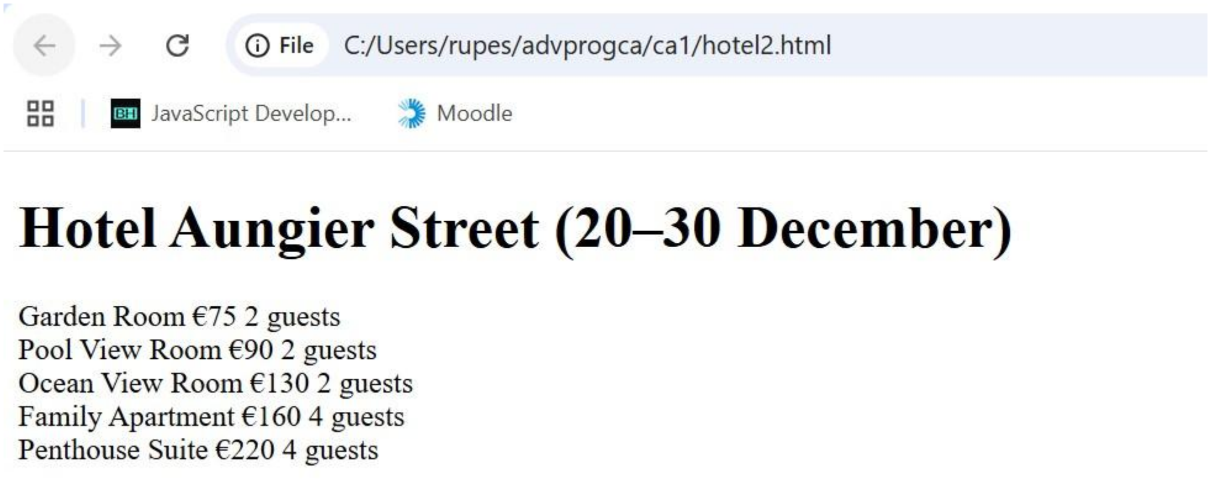
5.4 Execution with Screenshots



Hotel Castle House (20–30 December)

Basic Room €50 1 guests
Standard Room €80 2 guests
Superior Room €95 2 guests
Deluxe Room €120 3 guests
Family Suite €150 4 guests
Sea View Suite €180 2 guests

Figure 20: hotel1.html

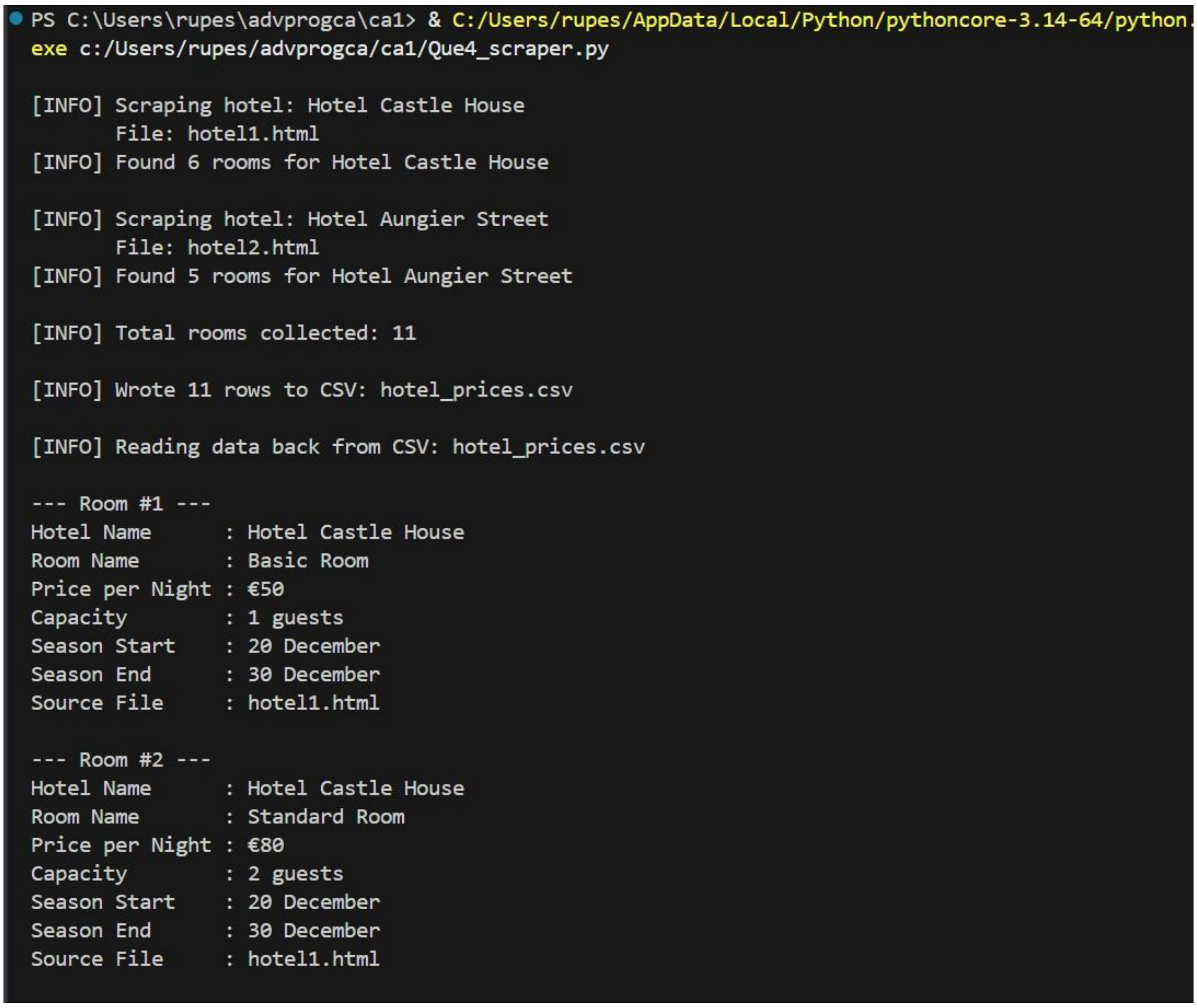


Hotel Aungier Street (20–30 December)

Garden Room €75 2 guests
Pool View Room €90 2 guests
Ocean View Room €130 2 guests
Family Apartment €160 4 guests
Penthouse Suite €220 4 guests

Figure 21: hotel2.html

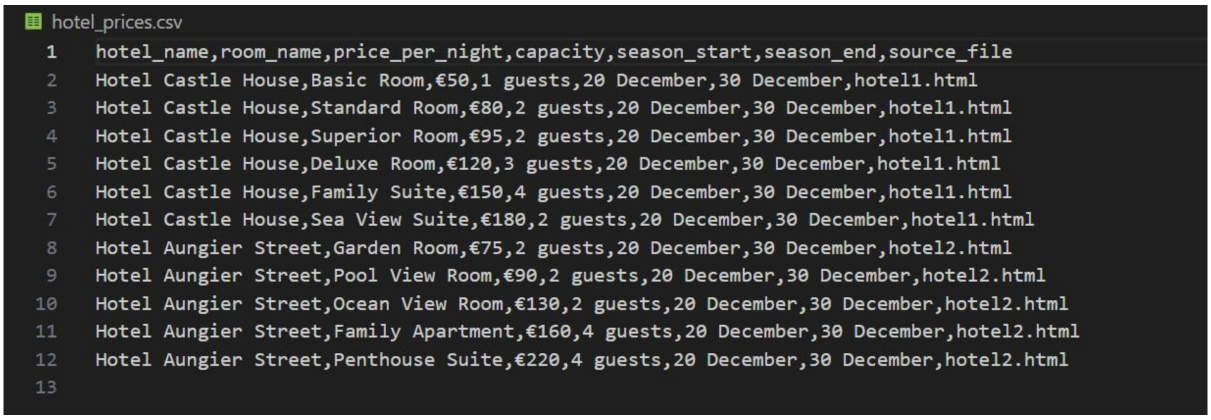
After executing program Que4_scraper.py, program displays information of all available rooms for the period December 20-30 from two hotels Castle House and Aungier Street which we created using hotel1.html and hotel2.html.



```
PS C:\Users\rupes\advprogca\ca1> & C:/Users/rupes/AppData/Local/Python/pythoncore-3.14-64/python.
exe c:/Users/rupes/advprogca/ca1/Que4_scraper.py

[INFO] Scraping hotel: Hotel Castle House
       File: hotel1.html
[INFO] Found 6 rooms for Hotel Castle House

[INFO] Scraping hotel: Hotel Aungier Street
       File: hotel2.html
[INFO] Found 5 rooms for Hotel Aungier Street

[INFO] Total rooms collected: 11

[INFO] Wrote 11 rows to CSV: hotel_prices.csv

[INFO] Reading data back from CSV: hotel_prices.csv

--- Room #1 ---
Hotel Name      : Hotel Castle House
Room Name       : Basic Room
Price per Night : €50
Capacity        : 1 guests
Season Start    : 20 December
Season End      : 30 December
Source File     : hotel1.html

--- Room #2 ---
Hotel Name      : Hotel Castle House
Room Name       : Standard Room
Price per Night : €80
Capacity        : 2 guests
Season Start    : 20 December
Season End      : 30 December
Source File     : hotel1.html
```

Figure 22 : Rooms Information

A new file hotel_prices.csv was also created and the contents of this csv file can be seen below.



hotel_prices.csv

```
hotel_name,room_name,price_per_night,capacity,season_start,season_end,source_file
Hotel Castle House,Basic Room,€50,1 guests,20 December,30 December,hotel1.html
Hotel Castle House,Standard Room,€80,2 guests,20 December,30 December,hotel1.html
Hotel Castle House,Superior Room,€95,2 guests,20 December,30 December,hotel1.html
Hotel Castle House,Deluxe Room,€120,3 guests,20 December,30 December,hotel1.html
Hotel Castle House,Family Suite,€150,4 guests,20 December,30 December,hotel1.html
Hotel Castle House,Sea View Suite,€180,2 guests,20 December,30 December,hotel1.html
Hotel Aungier Street,Garden Room,€75,2 guests,20 December,30 December,hotel2.html
Hotel Aungier Street,Pool View Room,€90,2 guests,20 December,30 December,hotel2.html
Hotel Aungier Street,Ocean View Room,€130,2 guests,20 December,30 December,hotel2.html
Hotel Aungier Street,Family Apartment,€160,4 guests,20 December,30 December,hotel2.html
Hotel Aungier Street,Penthouse Suite,€220,4 guests,20 December,30 December,hotel2.html

```

Figure 23 : hotel_prices.csv created

6. Conclusion

This assessment showcased practical skills across C# and Python programming domains. Generative AI(ChatGPT) was used to generate 20 random pre-loaded contacts for Question 1 and 20 random file extensions and information for Question 2.

The tasks demonstrated:

Part 1 :

- Understanding of **OOP principles (Constructor, Encapsulation for Que1 as Contact class has private setter MobileNumber)**
- Ability to design **menu-driven applications**
- Use of **data structures and exception handling**

Part 2 :

- Building a **TCP client–server system** with a real database
- Implementing **web scraping**, structured storage, and data presentation

7. References

BeautifulSoup (Python Web Scraping Library)

Richardson, L. (2023). *Beautiful Soup Documentation*. Crummy. https://www.crummy.com/software/BeautifulSoup/

JSON (JavaScript Object Notation Standard)

JSON. (2017). *The JavaScript Object Notation (JSON) Data Interchange Format*. IETF RFC 8259. https://www.rfc-editor.org/rfc/rfc8259

ChatGPT

OpenAI. (2023). *ChatGPT (Feb 2023 version)*. OpenAI. Available at: https://chat.openai.com/