



Name: Karnishwar Ravikrishna (20086868)

Course Title: M.Sc. Cyber Security

Lecturer Name: Swati Dongre

Module/Subject Title: Advanced Programming Techniques

Assignment Title: CA_ONE_(30%)

No. of Words: 740

Table Of Contents:

Section	Page
Introduction	3
Part 1 – C# Programming	4
Question 1	4
Question 2	6
Part 2 – Python Programming	7
Question 3	7
Question 4	8
Conclusion	11
Git Repository Link	11

Introduction:

This report is demonstrated how I implemented the coding tasks given.

This report consists of two parts, in which the first part consists of two tasks that need to be completed with **C#** coding, and the second part consists of two tasks that needs to be completed with **Python** coding.

In PART I the first task given to me was to create a phone contact book that allows users to add, update, view, and delete contacts. Important concepts from object-oriented programming, including classes, objects, encapsulation, properties, and method overloading, were used in the development of this system. The second task was to create a C# program to identify different file extension types.

In PART II the first task given to me was to create Creating a client-server application via TCP is the first Python task. Before returning a unique registration number, the server saves the admission data that the client gathers from applicants in a database. The second task was to Web Scrap two Hotel sites and get the information like room rent and return all the details as a CSV files.

The report also includes Screenshots of the outputs of all the codes and also the link for GitHub repository used for this project.

Part 1 – C# Programming:

Question 1:

In this task, I was asked to create a Contact Book program where a user can add a contact, view all contacts, search for a contact, update details, and delete a contact. The program needed to store at least 20 contacts and validate that the mobile number entered is a non-zero, 9-digit number. I used Object-Oriented Programming by creating a Contact class to hold the details and a PhoneBook class to manage all contacts. I also used method overloading and properties to show encapsulation. A list was used to store multiple Contact objects.

```
Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
3
Name: Emily Blackwell, Number: 87111111
Name: Contact 2, Number: 80000002
Name: Contact 3, Number: 80000003
Name: Contact 4, Number: 80000004
Name: Contact 5, Number: 80000005
Name: Contact 6, Number: 80000006
Name: Contact 7, Number: 80000007
Name: Contact 8, Number: 80000008
Name: Contact 9, Number: 80000009
Name: Contact 10, Number: 80000010
Name: Contact 11, Number: 80000011
Name: Contact 12, Number: 80000012
Name: Contact 13, Number: 80000013
Name: Contact 14, Number: 80000014
Name: Contact 15, Number: 80000015
Name: Contact 16, Number: 80000016
Name: Contact 17, Number: 80000017
Name: Contact 18, Number: 80000018
Name: Contact 19, Number: 80000019
Name: Contact 20, Number: 80000020
Name: KV, Number: 123456789

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
```

Image 1 – ContactBook Result

```
○ DELL@MacBook-Air-5 20086868_Que_1 % /Users/DELL/.vscode/extensions/ms-dotnettools.csharp-2.100.11-darwin-arm64/.debugger/arm64/vsdbg --interpreter=vscode --connection=/var/folders/2y/5rv8w6js5kzbpxt5cygbfjm0000gn/T/CoreFxPipe_vsdbg-ui-63c5db1fe36d466daf44d27e1c725462
Welcome to 1Console PhoneBook app

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
1
Contact name:
KV
Contact number (9-digit, non-zero):
123456789
Contact added.

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
2
Contact number to search:
123456789
Name: KV, Number: 123456789
```

Image 2 – ContactBook Result

```
Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
4
Name search phrase:
KV
Name: KV, Number: 123456789

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
5
Enter current contact number to update:
123456789
New name (leave blank to keep same):
KA
New number (leave blank to keep same):
123456788
Contact updated.

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
6
Enter contact number to delete:
1234567898
Contact not found.
```

Image 3 – ContactBook Result

```

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
6
Enter contact number to delete:
123456788
Contact deleted.

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
1
Contact name:
SK
Contact number (9-digit, non-zero):
123456
Invalid number. Must be 9-digit, non-zero.

Select operation
1 Add contact
2 Display contact by number
3 View all contacts
4 Search for contacts for a given name
5 Update contact
6 Delete contact
Press 'x' to exit
1
Contact name:
SK
Contact number (9-digit, non-zero):
00000000
Invalid number. Must be 9-digit, non-zero.

```

Image 4 – ContactBook Result

Part 1 – C# Programming:

Question 2:

In this task I was asked to write a code that gives information about file extension types. So, if a user types **.mp3**, the code displays the full form of .mp3. It was also mentioned that the code should contain minimum 20 extension types and the exit command exits the operation. I used a Python Dictionary which is a key-value pair. Here, the key is the extension and Value is the full form of the Extension. Dictionary was used as it is simple to integrate and fast.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK

complete:13: command not found: compdef
DELL@MacBook-Air-5 20086868_Que_2 % /Users/DELL/.vscode/extensions/ms-dotnettools.csharp-2.100.11-darwin-arm64/.debugger/arm64/vsdbg --interpreter=vscode --connection=/var/folders/2y/5rv8w6js5kzbpxt5cygbfjm000gn/T/CoreFxPipe_vsdbg-ui-fe17ab900f2e4740b1107de092c07a96
== File Extension Information System ==

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .flac
.flac → Lossless Audio File

Enter a file extension (e.g. .mp4) or type 'exit' to quit: exit
DELL@MacBook-Air-5 20086868_Que_2 % /Users/DELL/.vscode/extensions/ms-dotnettools.csharp-2.100.11-darwin-arm64/.debugger/arm64/vsdbg --interpreter=vscode --connection=/var/folders/2y/5rv8w6js5kzbpxt5cygbfjm000gn/T/CoreFxPipe_vsdbg-ui-9aa8663de35b452f9d385a23a7d4706b
== File Extension Information System ==

Enter a file extension (e.g. .mp4) or type 'exit' to quit: exit
● DELL@MacBook-Air-5 20086868_Que_2 % clear
● DELL@MacBook-Air-5 20086868_Que_2 % /Users/DELL/.vscode/extensions/ms-dotnettools.csharp-2.100.11-darwin-arm64/.debugger/arm64/vsdbg --interpreter=vscode --connection=/var/folders/2y/5rv8w6js5kzbpxt5cygbfjm000gn/T/CoreFxPipe_vsdbg-ui-f954ae8ede5a472b80ff8b635fca03aa
== File Extension Information System ==

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .mp3
.mp3 → Audio File (MPEG-3)

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .mp4
.mp4 → Video File (MPEG-4)

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .flac
.flac → Lossless Audio File

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .xlsx
.xlsx → Microsoft Excel Spreadsheet

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .docx
.docx → Microsoft Word Document

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .pdf
.pdf → Portable Document Format

Enter a file extension (e.g. .mp4) or type 'exit' to quit: .html
.html → Webpage File (HTML)

Enter a file extension (e.g. .mp4) or type 'exit' to quit: exit
❖ DELL@MacBook-Air-5 20086868_Que_2 %

```

Que_2 - Debug Any CPU - Solution file opened: 20086868_Que_2.sln - Indexing completed - Java: Ready - Spaces: 4 - UTF-8 with BOM - LF - () C# - 88

Image 5 – File Extension Types Result

Part 2 – Python Programming:

Question 3:

In this task, I was asked to create a client–server application where the client collects admission details from a student and sends them to the server. The server receives the information, stores it, and sends back a unique registration number. I used TCP because TCP is reliable and ensures that all data sent from the client is delivered safely and in order. The client gets the applicant's name, address, qualifications, course choice, and intended start date, then sends this information to the server. The server stores the data in a file and generates a unique ID using Python UUID library. This unique ID is returned to the client as confirmation.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK + × ⋮ | X

complete:13: command not found: compdef
DELL@MacBook-Air-5:20086868_Que_3 % python Que_3_server.py
zsh: command not found: python
DELL@MacBook-Air-5:20086868_Que_3 % python3 Que_3_server.py
Server listening on 127.0.0.1:6000
Client connected: ('127.0.0.1', 59846)
Saved application for Karnishwar Reg No: DBS2025-0001
[]

complete:13: command not found: compdef
DELL@MacBook-Air-5:20086868_Que_3 % python3 Que_3_client.py
== DBS Admission Application Client ==
Name: Karnishwar
Address: Aungier Street
Educational qualifications: BE

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

Connecting to server 127.0.0.1:6000...

Application submitted successfully.
Your registration number is: DBS2025-0001
DELL@MacBook-Air-5:20086868_Que_3 %
```

The screenshot shows a terminal window with two sessions. The left session is a server (Que_3_server.py) and the right session is a client (Que_3_client.py). The client session shows the user input for registration details, including name, address, educational qualifications, and course selection. The application successfully submits the form, displaying the generated registration number DBS2025-0001.

Image 6 – Client-server Result [Left side = server, Right side = client]

Part 2 – Python Programming:

Question 4:

In this task, I was asked to scrape hotel room price information from two websites and present the data in a csv file in order. Since most sites did not allow scraping, I created two HTML webpages that acted as hotel sites. I used Python requests feature to fetch the webpage content and BeautifulSoup to extract the room names and prices using their HTML class names. After collecting the data, I stored it in a CSV file using Python csv module. At the end of the program, the CSV file was read and displayed in the terminal. HTML code for the hotel pages are also included in the project.

Kv's B&B – Rooms for 20–30 December

Standard Double Room

€85 per night

Sea View Double Room

€110 per night

Family Room (3 Guests)

€130 per night

Single Room

€70 per night

Deluxe Suite

€160 per night

Budget Twin Room

€80 per night

Image 7 – Hotel Website 1

Castle House B&B – Rooms for 20–30 December

City View Double Room

€95 per night

Economy Double

€75 per night

Family Room (4 Guests)

€140 per night

Studio Apartment

€155 per night

Single Room

€65 per night

Image 8 – Hotel Website 2

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK
complete:13: command not found: compdef
❶ DELL@MacBook-Air-5 20086868_Que_4 % pyh
zsh: command not found: pyh
❷ DELL@MacBook-Air-5 20086868_Que_4 % python3 -m http.server 9000
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/runpy.py", line 196, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/runpy.py", line 86, in _run_code
    exec(code, run_globals)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/http/server.py", line 1307, in <module>
    test()
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/http/server.py", line 1258, in test
    with ServerClass(addr, HandlerClass) as httpd:
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/socketserver.py", line 452, in __init__
    self.server_bind()
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/http/server.py", line 137, in server_bind
    socketserver.TCPServer.server_bind(self)
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/socketserver.py", line 466, in server_bind
    self.socket.bind(self.server_address)
OSError: [Errno 48] Address already in use
❸ DELL@MacBook-Air-5 20086868_Que_4 % python3 -m http.server 9000
Serving HTTP on :: port 9000 (http://[::]:9000) ...
::ffff:127.0.0.1 -- [07/Dec/2025 14:09:57] "GET /hotel1.html HTTP/1.1"
200 -
::ffff:127.0.0.1 -- [07/Dec/2025 14:09:57] "GET /hotel2.html HTTP/1.1"
200 -

```

The terminal window shows the execution of a Python script named Que_4.py. It attempts to start a local HTTP server on port 9000 but fails because the port is already in use. Instead, it serves two files, hotel1.html and hotel2.html, which contain hotel price data. The data is displayed in the terminal window.

Image 9 – Web Scraping Result (Hotel details displayed in the Terminal)

The code editor displays two files: Que_4.py and hotel_prices.csv. The Que_4.py script is a simple Python program that scrapes hotel price data from two files (hotel1.html and hotel2.html) and prints it to the terminal. The hotel_prices.csv file contains the scraped data in CSV format.

Index	Hotel, Room, Price
1	Kv's B&B, Standard Double Room, ~85 per night
2	Kv's B&B, Sea View Double Room, ~110 per night
3	Kv's B&B, Family Room (3 Guests), ~130 per night
4	Kv's B&B, Single Room, ~70 per night
5	Kv's B&B, Deluxe Suite, ~160 per night
6	Kv's B&B, Budget Twin Room, ~80 per night
7	Castle House B&B, City View Double Room, ~95 per night
8	Castle House B&B, Economy Double, ~75 per night
9	Castle House B&B, Family Room (4 Guests), ~140 per night
10	Castle House B&B, Studio Apartment, ~155 per night
11	Castle House B&B, Single Room, ~65 per night
12	Castle House B&B, Single Room, ~65 per night
13	

Image 10 – Web Scraping Result (Downloaded CSV File)

Conclusion:

Overall, the tasks in this assignment allowed me to apply different programming concepts. In Question 1, I created a Contact Book system where users can add, view, update, and delete contact information. I used Object-Oriented Programming concepts such as classes and objects and I applied encapsulation by storing contact details inside a Contact class with properties. In Question 2, I used a dictionary to efficiently store and retrieve file extension information. In Question 3, I implemented a TCP-based client–server system that fetched user data and generated a unique registration ID. In Question 4, I created two local webpages and used web scraping to extract hotel room prices and store them in a CSV file.

GitHub Repository Link:

<https://github.com/karnishwar1725/CA-1.git>