



Student Name: Arinze Akosa

Student Number: 20072632

Course Title: M.Sc. Cyber Security

Lecturer Name: Swati Dongre

Module/Subject Title: Advanced Programming Techniques

Assignment Title: CA_ONE_(30%)

No of Words: 645

<https://github.com/Sarizonn/AdvancedProgramming-ca1>

Introduction

This report shows the design and implementation of four programming tasks. Part I shows Object-Oriented Programming using C# for console applications, while Part II applies Python for network programming and web data extraction.

Part I: Programming in C#

Question 1: Contact Book Application

For this task, I created a menu-driven console application that functions as a digital phonebook. The system allows users to store contacts, view details, update information, and delete records when necessary.

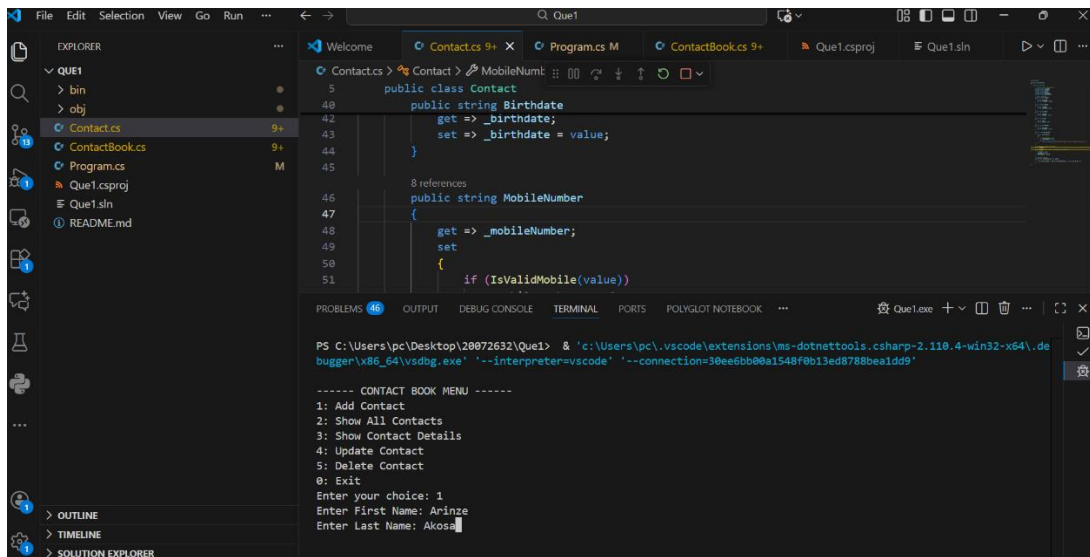
Implementation Details

- **OOP Principles:** I designed a class named `Contact` to represent a person. I used Encapsulation by keeping fields like `_firstName` and `_mobileNumber` private and exposing them through public properties. This allowed me to add logic to the `MobileNumber` property to ensure it only accepts valid 9-digit numbers.
- **Data Structure:** To store the contacts, I used a `List<Contact>` within the `ContactBook` class. Lists were the perfect choice here because they are dynamic—we can add or remove contacts without worrying about fixed sizes like arrays.
- **Functionality:**

I included a `SeedContacts()` method that automatically generates 20 sample contacts (e.g., "Person1", "Person2") when the program starts, meeting the minimum requirement.

The `AddContact()` method uses try-catch blocks to prevent the program from crashing if the user enters invalid data.

Screenshots:



```
----- CONTACT BOOK MENU -----
1: Add Contact
2: Show All Contacts
3: Show Contact Details
4: Update Contact
5: Delete Contact
0: Exit
Enter your choice: 1
Enter First Name: John
Enter Last Name: Oke
Enter Company: DBS
Enter Mobile Number (9 digits): 083456789
Enter Email: jn@gmail.com
Enter Birthdate: 11 feb 1999
Contact added successfully!

----- CONTACT BOOK MENU -----
1: Add Contact
2: Show All Contacts
3: Show Contact Details
4: Update Contact
5: Delete Contact
```

```
PROBLEMS 46 OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK ...

Enter Last Name: Akosa
Enter Company: DBS
Enter Mobile Number (9 digits): 892596354
Enter Email: Ako@gmail.com
Enter Birthdate: 11 jan 1999
Contact added successfully!

----- CONTACT BOOK MENU -----
1: Add Contact
2: Show All Contacts
3: Show Contact Details
4: Update Contact
5: Delete Contact
0: Exit
Enter your choice: █
```

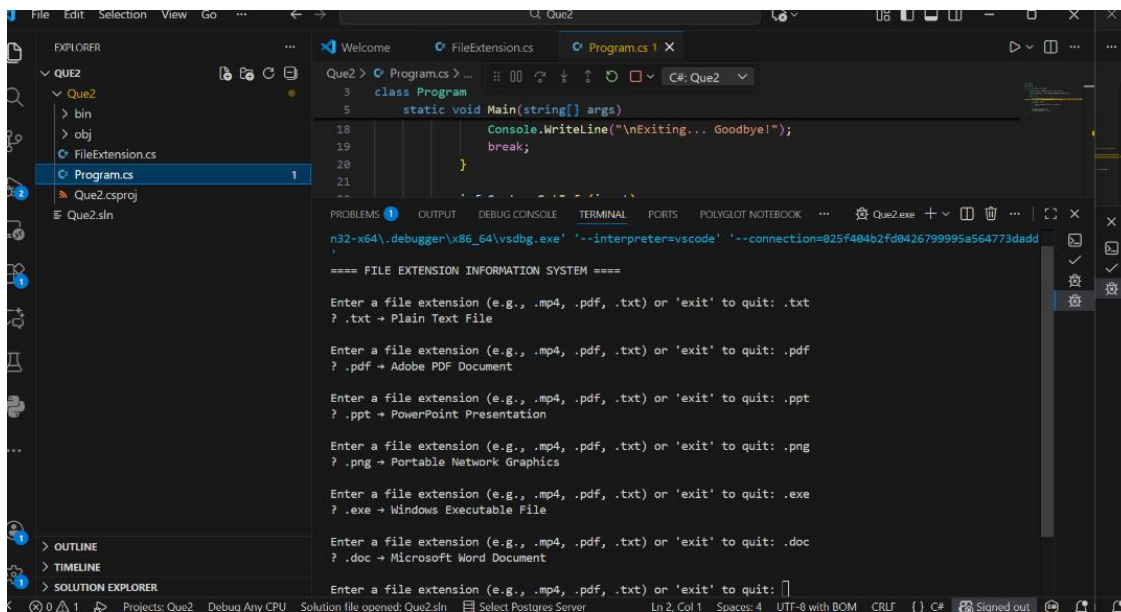
Question 2: File Extension Assistant

This program helps users identify file types based on their extensions. For example, if a user enters .ppt, the system tells them it is a "PowerPoint Presentation."

Implementation Details

- **Data Structure:** I used a Dictionary<string, string> to store the data. This is much faster than a list because it pairs a "Key" (the extension) directly with a "Value" (the description), allowing for instant lookups.
- **Logic:** The GetInfo method checks if the user's input exists in the dictionary.
 - If it exists (e.g., .pdf), it prints the description.
 - If the input is missing a dot (e.g., pdf) or is unknown, the program handles it gracefully by printing a friendly warning message instead of crashing.

Screenshots:



```
PS C:\Users\pc\Desktop\20072632\Que2> & 'c:\Users\pc\.vs
code\extensions\ms-dotnettools.csharp-2.110.4-win32-x64\
debugger\x86_64\vsdbg.exe' '--interpreter=vscode' '--conn
ection=025f404b2fd0426799995a564773dadd'
==== FILE EXTENSION INFORMATION SYSTEM ====

Enter a file extension (e.g., .mp4, .pdf, .txt) or 'exit'
to quit: .txt
? .txt → Plain Text File

Enter a file extension (e.g., .mp4, .pdf, .txt) or 'exit'
to quit: .pdf
? .pdf → Adobe PDF Document

Enter a file extension (e.g., .mp4, .pdf, .txt) or 'exit'
to quit: .ppt
? .ppt → PowerPoint Presentation

Enter a file extension (e.g., .mp4, .pdf, .txt) or 'exit'
to quit: .png
Enter a file extension (e.g., .mp4, .pdf, .txt) or 'exit'
to quit: .ppt
? .ppt → PowerPoint Presentation
```

Part II: Programming in Python

Question 3: Client-Server Admission System

This task shows a college admission system using a Client-Server architecture. The Client collects student data, and the Server saves it and issues a registration number.

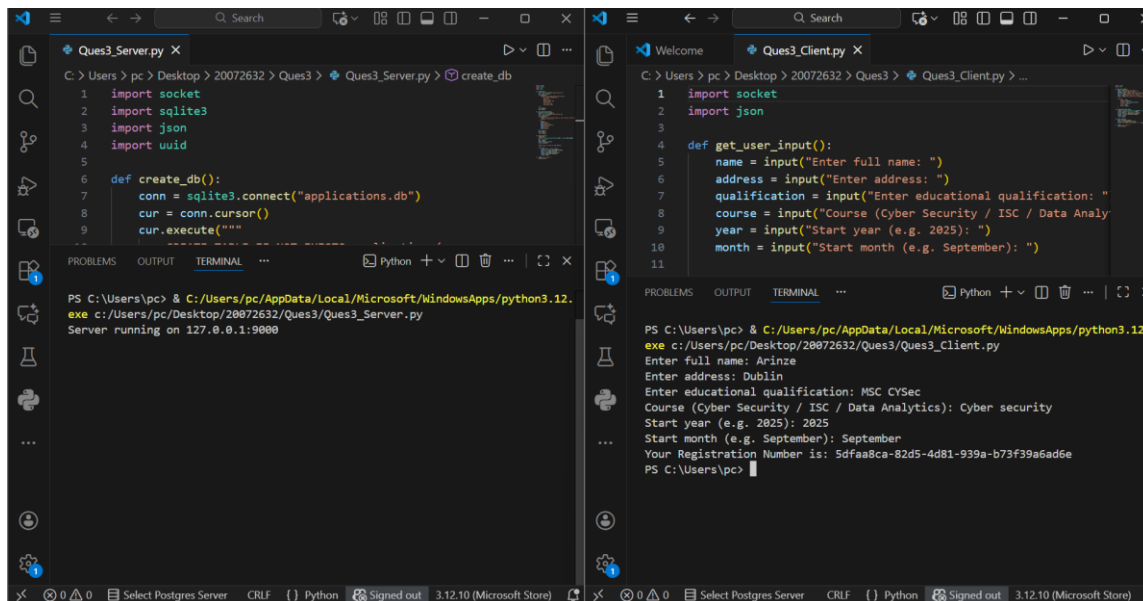
Implementation Details

- **Networking:** I used the Python socket library to implement the TCP protocol. I chose TCP because it is connection-oriented, ensuring that the student's personal data is delivered reliably without errors.
- **Database:** On the server side (Que3_Server.py), I used SQLite. The code creates a database file named applications.db. This ensures the data is persistent—meaning if I restart the server, the student records are still there.
- **Data Handling:**

The Client collects inputs (Name, Course, Address) and bundles them into a JSON format before sending.

The Server receives the JSON, assigns a unique ID using the uuid library, saves the record to the database, and sends the unique ID back to the Client as confirmation.

Screenshots:



The image shows two side-by-side screenshots of the Visual Studio Code editor. The left window displays the `Ques3_Server.py` file with the following code:

```
1 import socket
2 import sqlite3
3 import json
4 import uuid
5
6 def create_db():
7     conn = sqlite3.connect("applications.db")
8     cur = conn.cursor()
9     cur.execute("""
```

The terminal output for the server shows the command to run the server and its status:

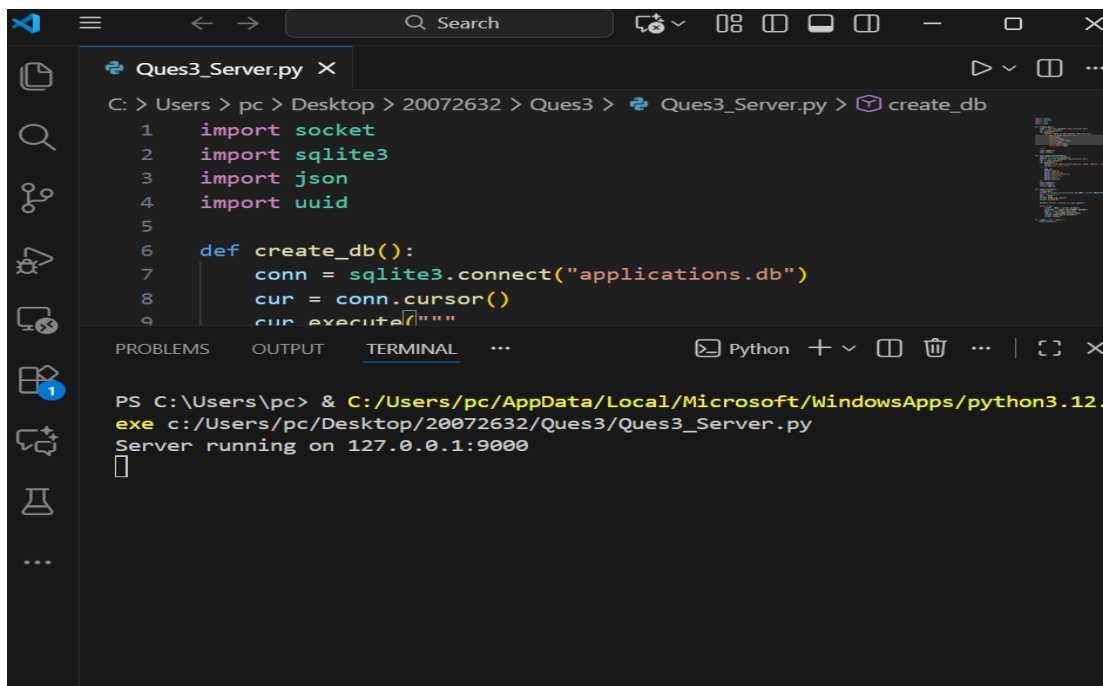
```
PS C:\Users\pc> & C:/Users/pc/AppData/Local/Microsoft/WindowsApps/python3.12.
exe c:/Users/pc/Desktop/20072632/Ques3/Ques3_Server.py
Server running on 127.0.0.1:9000
```

The right window displays the `Ques3_Client.py` file with the following code:

```
1 import socket
2 import json
3
4 def get_user_input():
5     name = input("Enter full name: ")
6     address = input("Enter address: ")
7     qualification = input("Enter educational qualification: ")
8     course = input("Course (Cyber Security / ISC / Data Analy")
9     year = input("Start year (e.g. 2025): ")
10    month = input("Start month (e.g. September): ")
11
```

The terminal output for the client shows the user input and the registration number generated by the server:

```
PS C:\Users\pc> & C:/Users/pc/AppData/Local/Microsoft/WindowsApps/python3.12.
exe c:/Users/pc/Desktop/20072632/Ques3/Ques3_Client.py
Enter full name: Arinze
Enter address: Dublin
Enter educational qualification: MSC CYSec
Course (Cyber Security / ISC / Data Analytics): Cyber security
Start year (e.g. 2025): 2025
Start month (e.g. September): September
Your Registration Number is: 5dfaa8ca-82d5-4d81-939a-b73f39a6ad6e
PS C:\Users\pc>
```

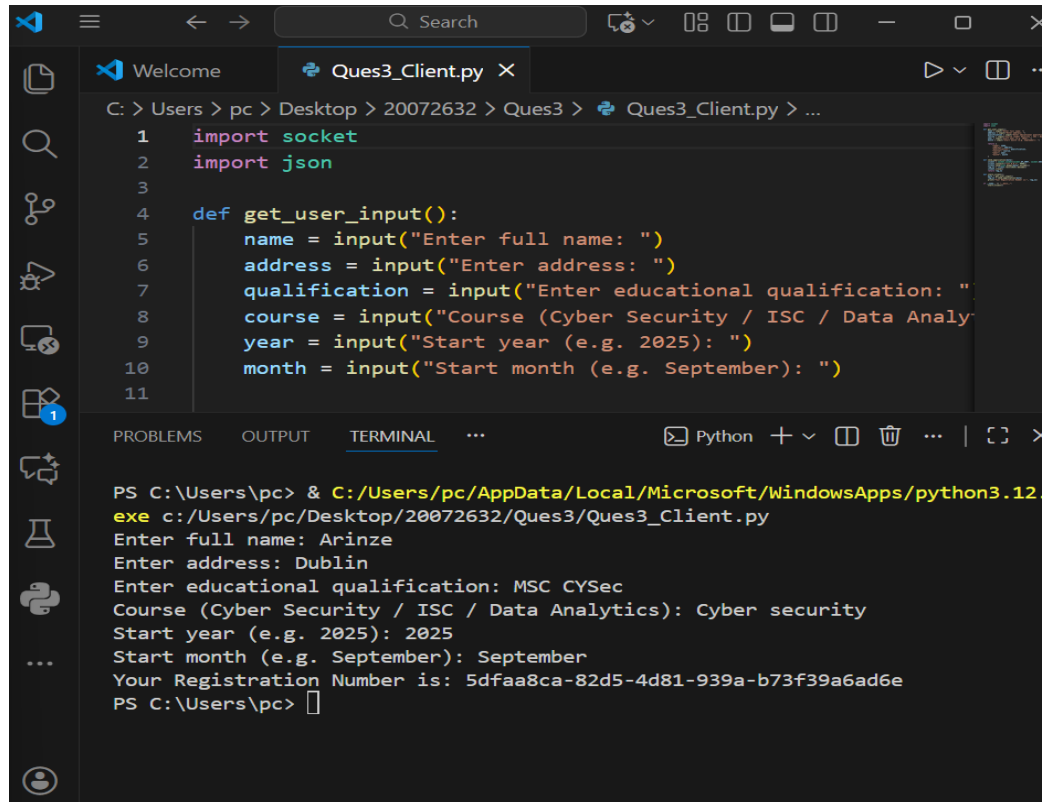


The image shows a screenshot of the Visual Studio Code editor with the `Ques3_Server.py` file open. The code is the same as in the previous screenshot:

```
1 import socket
2 import sqlite3
3 import json
4 import uuid
5
6 def create_db():
7     conn = sqlite3.connect("applications.db")
8     cur = conn.cursor()
9     cur.execute("""
```

The terminal output shows the server running on 127.0.0.1:9000:

```
PS C:\Users\pc> & C:/Users/pc/AppData/Local/Microsoft/WindowsApps/python3.12.
exe c:/Users/pc/Desktop/20072632/Ques3/Ques3_Server.py
Server running on 127.0.0.1:9000
```



The image shows a Visual Studio Code window with a file named `Ques3_Client.py` open. The code is a Python script that uses `socket` and `json` modules. It defines a function `get_user_input()` that prompts the user for their full name, address, educational qualification, course, start year, and start month. The script then generates a registration number based on these inputs. Below the code editor, the `TERMINAL` tab is active, showing the command prompt output of running the script. The user has entered the following information: full name 'Arinze', address 'Dublin', educational qualification 'MSC CYSec', course 'Cyber security', start year '2025', and start month 'September'. The terminal output shows the generated registration number: `5dfaa8ca-82d5-4d81-939a-b73f39a6ad6e`.

```
C: > Users > pc > Desktop > 20072632 > Ques3 > Ques3_Client.py > ...  
1 import socket  
2 import json  
3  
4 def get_user_input():  
5     name = input("Enter full name: ")  
6     address = input("Enter address: ")  
7     qualification = input("Enter educational qualification: ")  
8     course = input("Course (Cyber Security / ISC / Data Analy")  
9     year = input("Start year (e.g. 2025): ")  
10    month = input("Start month (e.g. September): ")  
11  
PS C:\Users\pc> & C:/Users/pc/AppData/Local/Microsoft/WindowsApps/python3.12.  
exe c:/Users/pc/Desktop/20072632/Ques3/Ques3_Client.py  
Enter full name: Arinze  
Enter address: Dublin  
Enter educational qualification: MSC CYSec  
Course (Cyber Security / ISC / Data Analytics): Cyber security  
Start year (e.g. 2025): 2025  
Start month (e.g. September): September  
Your Registration Number is: 5dfaa8ca-82d5-4d81-939a-b73f39a6ad6e  
PS C:\Users\pc>
```

Question 4: Hotel Price Scraper

The final task involved extracting room price data for a holiday period. I built a scraper that gathers data from two specific hotel listing sites: "DublinStays" and "Luxe Haven."

Implementation Details

- **Web Scraping:** I used the `requests` library to fetch the HTML and `BeautifulSoup` to search through it. The script looks for specific HTML tags (like `hotel-card` and `current-price`) to find the relevant information.
- **Data Cleaning:** Prices on websites often include text or symbols. I used Python's **Regular Expressions (re)** module to strip away currency symbols and extract just the numeric price for calculation.

- **Storage & Display:** The data is first written to a file named `hotel_prices.csv`. At the end of the script, the program reads this CSV file back and prints a neatly formatted table to the terminal so the user can compare prices easily.

Hotel	Location	Room Type	Price/Night
Clayton Hotel Burlington Road	Ballsbridge, Dublin	Standard King Room • 1 king	€150
The Chancery Hotel	Dublin City Centre, Dub	Standard King Room • 1 queen	€229
Beckett Locke	Dublin City Centre, Dub	One-Bedroom Apartment • 1 be	€200
The Grafton Hotel	Saint Stephen's Green,	Standard Double Room • 1 ful	€303
Harcourt Hotel	Saint Stephen's Green,	Double Room (Maximum Occupan	€177
Mespil Hotel	Ballsbridge, Dublin	Superior Twin Room • 2 twin	€195
Trinity Townhouse Hotel	Saint Stephen's Green,	Classic Double Room • 1 full	€356
The Dean Dublin	Saint Stephen's Green,	The Large • 1 king bed	€229
Clontarf Castle Hotel	Clontarf, Dublin	Standard Twin Room • 2 twin	€185
The Trinity City Hotel	Dublin City Centre, Dub	Double Room • 1 full bed	€225
Clayton Hotel Burlington Road	Ballsbridge, Dublin	Standard King Room • 1 king	€143
The Chancery Hotel	Dublin City Centre, Dub	Standard King Room • 1 queen	€234
Beckett Locke	Dublin City Centre, Dub	One-Bedroom Apartment • 1 be	€190
The Grafton Hotel	Saint Stephen's Green,	Standard Double Room • 1 ful	€309
Harcourt Hotel	Saint Stephen's Green,	Double Room (Maximum Occupan	€169
Mespil Hotel	Ballsbridge, Dublin	Superior Twin Room • 2 twin	€199
Trinity Townhouse Hotel	Saint Stephen's Green,	Classic Double Room • 1 full	€339
The Dean Dublin	Saint Stephen's Green,	The Large • 1 king bed	€234
Clontarf Castle Hotel	Clontarf, Dublin	Standard Twin Room • 2 twin	€189
The Trinity City Hotel	Dublin City Centre, Dub	Double Room • 1 full bed	€214

booking-hotels2.tiiny.site

Cyber Security Road... Cybersecurity 101... Welcome to Google... APT groups and thr... MITRE ATT&CK® Cybersecurity Gloss... Cyber Threats: Secu... All Bookmarks

Dublin Stays Home Hotels Why Us Help Sign In

Ireland > Dublin > **Hotels**

Find Your Perfect Stay in Dublin

Search 251 hotels and accommodations in Dublin, Ireland

DESTINATION	CHECK-IN	CHECK-OUT	GUESTS	
Dublin, Ireland	12/13/2025	12/14/2025	2 adults	Search

Why Book with DublinStays

Welcome to Luxe Haven

Experience luxury and comfort in the heart of paradise

Check-in

12/13/2025



Check-out

12/14/2025



Guests

1 Guest



[Search Rooms](#)