The DBS Admissions application was designed as a client–server system to handle student applications for postgraduate courses. The problem statement required applicants to submit personal details such as name, address, educational qualifications, the course they wish to enroll in, and their intended start year and month. Once the server receives this information, it must generate a unique registration number and store the record in a persistent database for future reference.

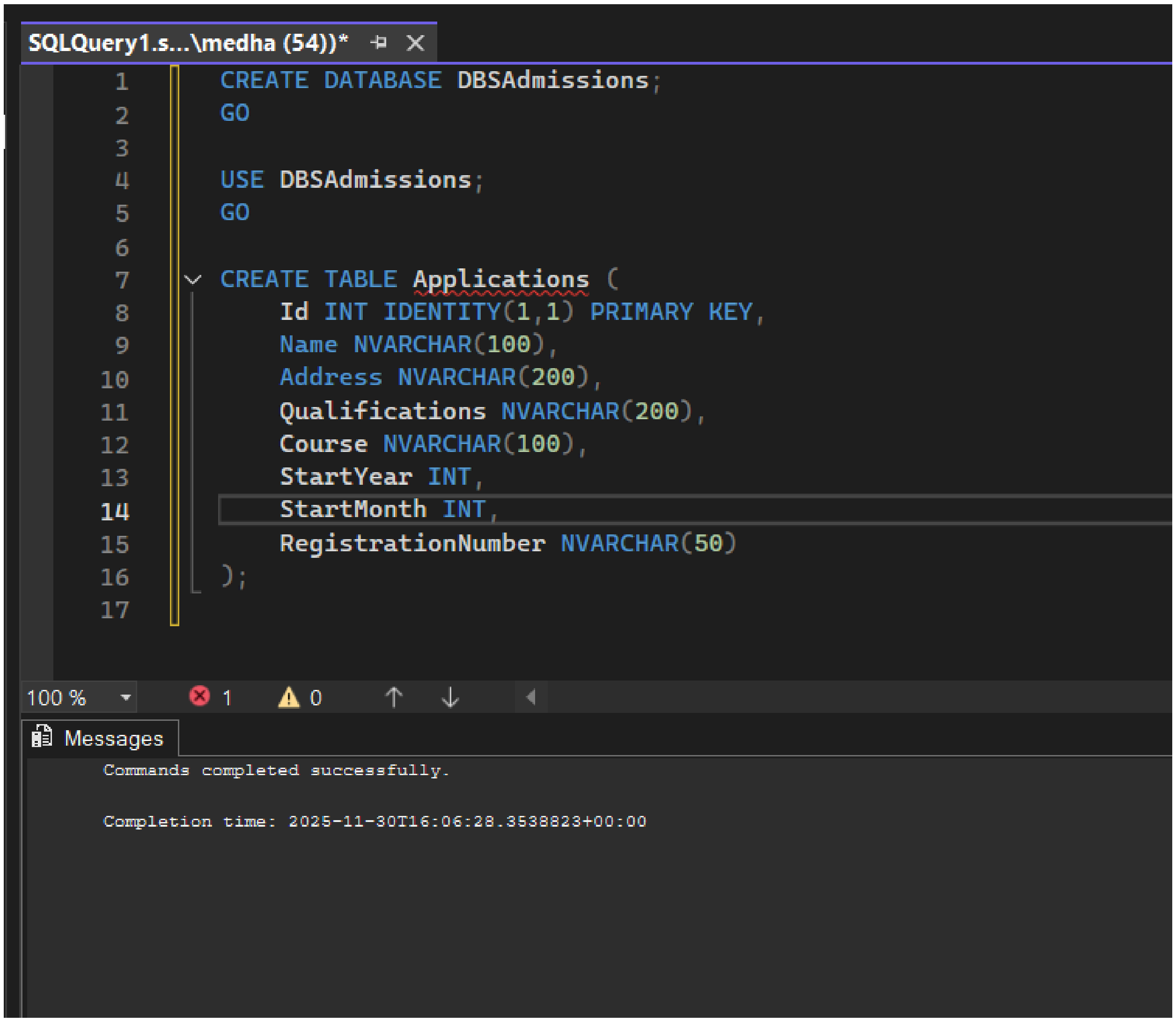
The solution uses Python and a connection-oriented protocol (TCP sockets). TCP was chosen because it guarantees reliable delivery of data between client and server, which is critical when handling admissions information. The system is divided into two main components: the client application and the server application.

The **client application** is console-based. It prompts the user to enter their details interactively. Input validation is included to ensure data integrity: the course must be selected from three valid options (MSc in Cyber Security, MSc Information Systems & Computing, MSc Data Analytics), the year must be a four-digit number (e.g., 2025), and the month must be between 1 and 12. If invalid input is provided, the client displays an error message and does not send the application to the server. Once valid data is collected, the client encodes the information as JSON and transmits it to the server using sendall(). After sending, the client waits for a response. If the server successfully inserts the record, the client displays a confirmation message along with the unique registration number generated by the server.

The **server application** listens for incoming TCP connections on a specified port. When a client connects, the server receives the JSON payload, decodes it, and parses the applicant's details. The server then connects to a disk-persistent relational database (SQL Server in this case) using the pyodbc library. A parameterized SQL INSERT statement is executed to store the applicant's information in the Applications table. Parameterization is important for security, as it prevents SQL injection attacks. After committing the transaction, the server generates a unique registration number based on the current timestamp (e.g., DBS-20251130175500). This number is returned to the client in a JSON response. The server also logs each step to the console: connection established, data received, parsed applicant details, SQL execution, and confirmation of successful insertion.

Security considerations include using parameterized queries, validating input both on the client and server, and restricting the server to listen only on localhost during development.

This implementation fulfills all requirements: a console-based client, a server that stores data in a persistent database, generation of a unique registration number, and communication over TCP. The system demonstrates reliable data transmission, proper validation, and secure database interaction.



SQLQuery1.s...\medha (54))*

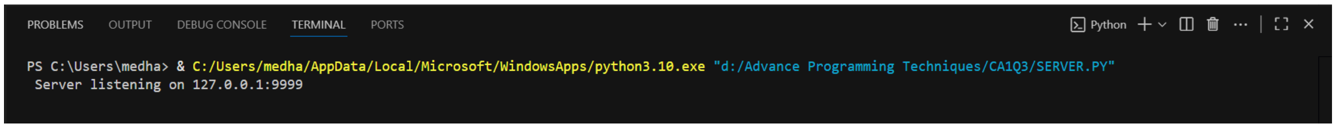
```sql
CREATE DATABASE DBSAdmissions;
GO

USE DBSAdmissions;
GO

CREATE TABLE Applications (
    Id INT IDENTITY(1,1) PRIMARY KEY,
    Name NVARCHAR(100),
    Address NVARCHAR(200),
    Qualifications NVARCHAR(200),
    Course NVARCHAR(100),
    StartYear INT,
    StartMonth INT,
    RegistrationNumber NVARCHAR(50)
);
```

100 % 1 0

Messages

```
Commands completed successfully.

Completion time: 2025-11-30T16:06:28.3538823+00:00
```

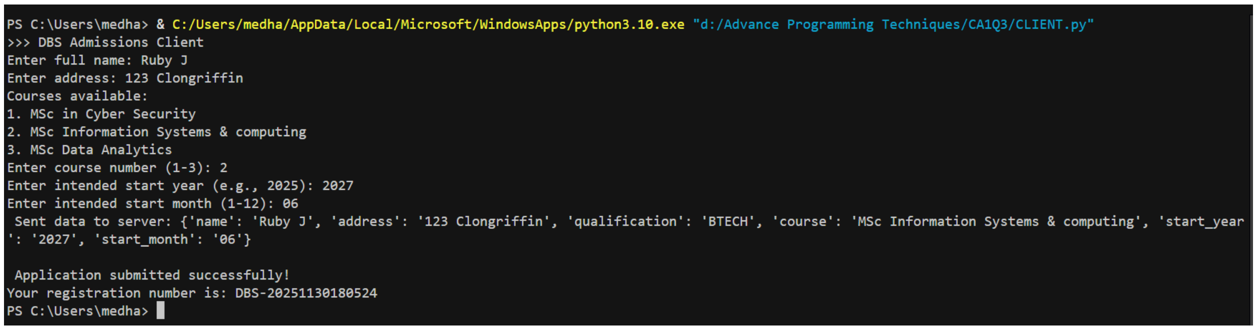
Server startup:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

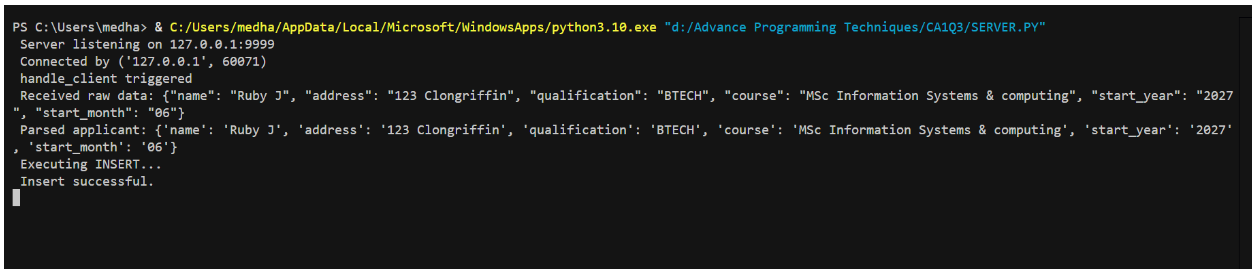
```
PS C:\Users\medha> & C:/Users/medha/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/Advance Programming Techniques/CA1Q3/SERVER.PY"
 Server listening on 127.0.0.1:9999
```

Client Console:



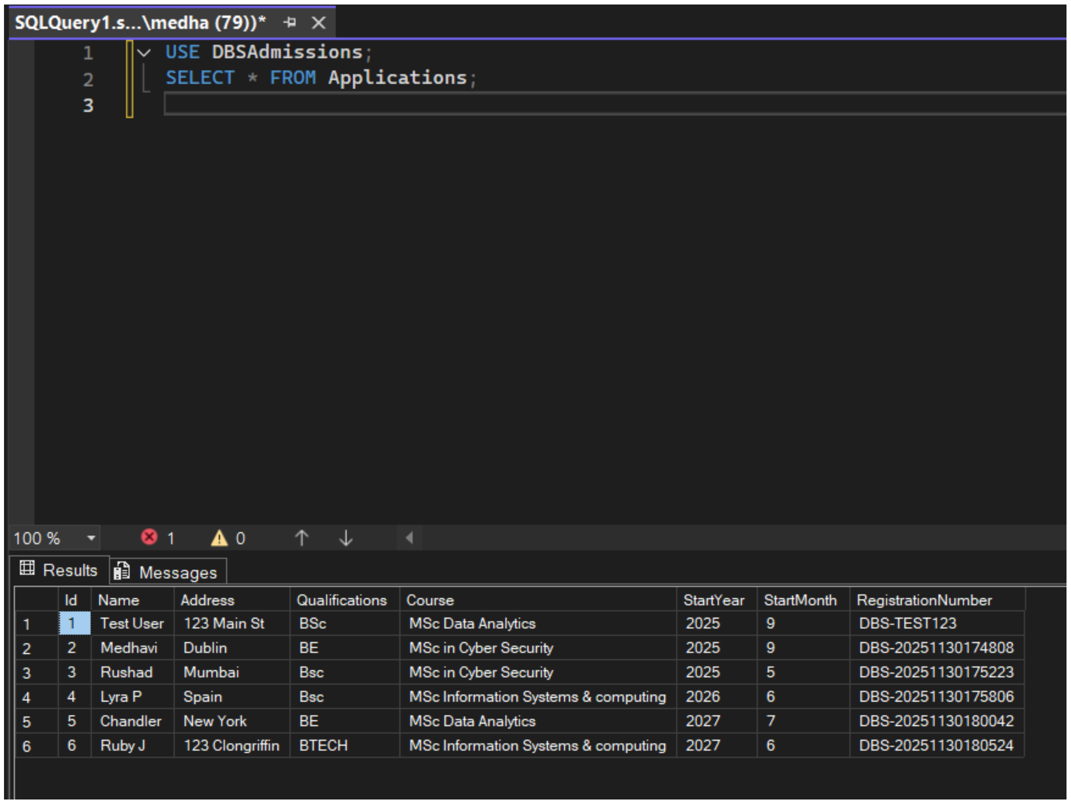
```
PS C:\Users\medha> & C:/Users/medha/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/Advance Programming Techniques/CA1Q3/CLIENT.py"
>>> DBS Admissions Client
Enter full name: Ruby J
Enter address: 123 Clongriffin
Courses available:
1. MSc in Cyber Security
2. MSc Information Systems & computing
3. MSc Data Analytics
Enter course number (1-3): 2
Enter intended start year (e.g., 2025): 2027
Enter intended start month (1-12): 06
 Sent data to server: {'name': 'Ruby J', 'address': '123 Clongriffin', 'qualification': 'BTECH', 'course': 'MSc Information Systems & computing', 'start_year
': '2027', 'start_month': '06'}

 Application submitted successfully!
Your registration number is: DBS-20251130180524
PS C:\Users\medha>
```

Server Console: Data inserted successfully



```
PS C:\Users\medha> & C:/Users/medha/AppData/Local/Microsoft/WindowsApps/python3.10.exe "d:/Advance Programming Techniques/CA1Q3/SERVER.PY"
 Server listening on 127.0.0.1:9999
 Connected by ('127.0.0.1', 60071)
 handle_client triggered
 Received raw data: {"name": "Ruby J", "address": "123 Clongriffin", "qualification": "BTECH", "course": "MSc Information Systems & computing", "start_year": "2027
", "start_month": "06"}
 Parsed applicant: {'name': 'Ruby J', 'address': '123 Clongriffin', 'qualification': 'BTECH', 'course': 'MSc Information Systems & computing', 'start_year': '2027'
, 'start_month': '06'}
 Executing INSERT...
 Insert successful.
```

Records Inserted Successfully in the Table:



SQLQuery1.s...\medha (79))*

```sql
USE DBSAdmissions;
SELECT * FROM Applications;
```

100 % ⊗ 1 ⚠ 0

Results | Messages

	Id	Name	Address	Qualifications	Course	StartYear	StartMonth	RegistrationNumber
1	1	Test User	123 Main St	BSc	MSc Data Analytics	2025	9	DBS-TEST123
2	2	Medhavi	Dublin	BE	MSc in Cyber Security	2025	9	DBS-20251130174808
3	3	Rushad	Mumbai	Bsc	MSc in Cyber Security	2025	5	DBS-20251130175223
4	4	Lyra P	Spain	Bsc	MSc Information Systems & computing	2026	6	DBS-20251130175806
5	5	Chandler	New York	BE	MSc Data Analytics	2027	7	DBS-20251130180042
6	6	Ruby J	123 Clongriffin	BTECH	MSc Information Systems & computing	2027	6	DBS-20251130180524