

# **AMRITSAR GROUP OF COLLEGES**

Autonomous status conferred by UGC under UGC act-1956, (2f), NAAC-A Grade (Formerly Known as Amritsar College of Engineering & Technology | Amritsar Pharmacy College)

## **PROJECT REPORT**

On

## **“HOSPITAL MANAGEMENT SYSTEM”**

Submitted in the Partial fulfilment of the requirement for the Award of Degree of

## **Bachelor of Technology**

In

## **COMPUTER SCIENCE & ENGINEERING**

Batch (2022-2026)



### **Submitted To:**

Er. Shelly Bhalla  
(Assistant Professor)

### **Submitted by:**

Akshita Chadha (2233561)  
Aniket Singh (2233571)  
Archie Arora (2233579)  
Binod Prasad Joshi (2233587)

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Amritsar Group of Colleges, Amritsar**

## **ACKNOWLEDGEMENT**

We would like to express our gratitude and appreciation to several persons who helped us giving the possibility to complete this report and conducting the project work. We express our special thanks and gratitude to our prestigious teacher and mentor **Er. Shelly Bhalla, Assistant Professor, Department of CSE, Amritsar Group of Colleges, Amritsar**, for his guidance and consistent encouragement in completing this project.

We are extremely thankful to **Dr. Sandeep Kad, Head of Department** and all the faculty members of CSE Department at **Amritsar Group of Colleges, Amritsar** for their co- operation and their guidance and encouragement to complete this project work on time.

We also want to thank all our friends for their contribution in successful completion of this project.

The completion of this project has helped us in gaining more knowledge related to Object Oriented Programming.

## **DECLARATION**

We Akshita, Aniket, Archie and Binod here as a team declare that the project work entitled “HOSPITAL MANAGEMENT SYSTEM” is an authentic record of our own work carried out as per the requirements of Data Structure Lab for the award of degree of B. Tech (CSE), Amritsar Group of Colleges, Amritsar, under the guidance of Er. Shelly Bhalla (Assistant Professor).

Akshita Chadha  
(2233561)

Aniket Singh  
(2233571)

Archie Arora  
(2233587)

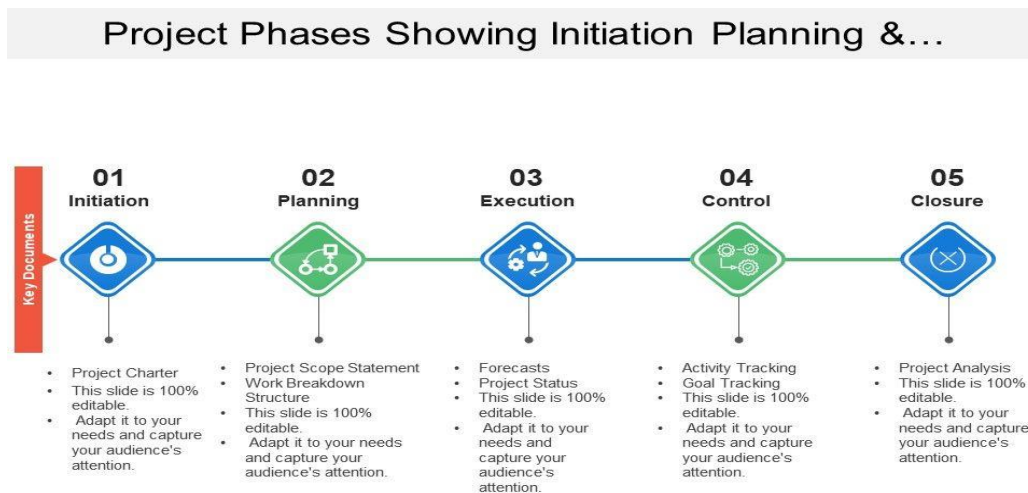
Binod Prasad Joshi  
(2233587)

## Table of Contents

<b>Introduction &amp; operation the data Structure .....</b>	<b>1</b>
<b>Objectives.....</b>	<b>4</b>
<b>Algorithms .....</b>	<b>5</b>
<b>Outputs .....</b>	<b>11</b>
<b>References.....</b>	<b>19</b>

## Introduction & operation the data Structure

A data structure is like a special blueprint that helps us to sort, process, find and save data in an organized manner. Whether it's a simple or complex structure, its main purpose is to make sure that the data can be easily accessed and used in the right way. In the world of computer programming, we carefully choose or create a data structure to store our data, so that we can work on it with different methods and techniques. Each data structure contains important details about the data itself, as well as the connections between the data and the operations that can be performed on it. Essentially, a data structure is a clever way of arranging and storing various types of data in the computer's memory. It's not just about storing the data, but also maintaining the logical relationships between individual data elements. Some might even say that a data structure is like a mathematical or logical model that helps us organize and make sense of our data.



### Operations Performed:-

#### 1. Login System:

- The program begins by presenting a login system with two roles: "Admin" and "User."
- Users are prompted to choose a role and then enter a username and password for authentication

## **2. Admin Role:**

If an administrator logs in successfully, they have access to the following operations:

### **Checking in Patients:**

- Admins can add patient details, including the patient's last name, first name, department, age, room number, and whether the patient is disabled.
- This information is saved in a file named "checkin.txt."

#### **i. Checking Out Patients:**

- Admins can remove a patient's check-in record by providing the room number.

#### **ii. Viewing Patients:**

- Admins can view a list of checked-in patients, displaying details like last name, first name, department, age, room number, and disability status.

#### **iii. Adding Employees:**

- Admins can add employee details, including last name, first name, position, and department.
- This information is saved in a file named "EmployeeData.txt."

#### **iv. Viewing Employees:**

- Admins can view a list of employees, displaying details such as last name, first name, position, and department.

#### **v. Viewing Registered Patients:**

- Admins can view a list of registered patients, including details like name, age, gender, and contact number.
- This information is stored in a file named "patient\_records.txt."

#### **vi. Viewing Appointments:**

- Admins can view a list of scheduled appointments, including patient names, facility types, facility names, dates, and times.
- This information is stored in a file named "Appointments.txt."

## **3. User Role:**

- When a regular user logs in, they can perform the following operations:

#### **i. Register as a New User:**

- Users can register with a username and password.

- This registration allows them to access further features.
- ii. Log In:**
  - Registered users can log in with their username and password to access additional functionalities.
- iii. Register Patients:**
  - Users can register patients by providing details such as name, age, gender, and contact number.
  - This patient information is saved in a file named "patient\_records.txt."
- iv. Schedule Appointments:**
  - Users can schedule appointments for patients, choosing available time slots for different facilities and facility types.
  - Appointment details are saved in a file named "Appointments.txt."
- v. View Available Appointment Slots:**
  - Users can view available appointment slots for different facilities. The code calculates the available slots based on scheduled appointments.
- 4. Data Storage:**
  - The program uses text files to store and manage data, including patient information, employee details, and appointment recor.

## Objectives

The code we have provided aims to create a simple yet effective hospital management system that caters to the needs of both administrators and regular users. With a focus on user authentication, data management, and task automation, our goal is to streamline the hospital operations and ensure a smooth experience for all parties involved.

Our system offers different user roles, allowing administrators to have access to a range of features such as patient check-in and check-out, employee and patient data management, appointment scheduling, and patient registration. On the other hand, regular users, once registered, can also perform tasks such as registering patients, scheduling appointments, and viewing available appointment slots. To keep things organized and easily accessible, all data is managed through text files. These files store records of patient information, employee data, and scheduled appointments. Our code also boasts a user-friendly menu-driven interface, providing a variety of functionalities that can be executed based on user roles and access permissions.

While we have implemented basic security and authentication measures, it's important to note that our code is primarily intended for educational purposes. It serves as a great foundation for understanding hospital management systems, but it may not have the depth and security features required for a comprehensive real-world application.

### **Proposed system will contain: -**

1. Implement more robust user authentication mechanisms, such as encryption and secure password storage.
2. Support multiple user roles (admin, doctor, nurse, receptionist, etc.) with distinct permissions.
3. Improve patient management by adding features for patient registration, updating patient information, and managing medical records.
4. Implement a patient portal for patients to access their own records and schedule appointments.
5. Create a sophisticated appointment scheduling system.



## Algorithms

### i. Login Class:

Class Login

Data:

- username (string)
- password (string)

Constructor Login (u, p):

- Initialize username with u
- Initialize password with p

Function getUsername():

- Return username

Function getPassword():

- Return password

### ii. UserData Class:

Class UserData

Data:

- fname (string)

Constructor UserData(file):

- Initialize fname with file

Function addUser(login):

- Input: Login object login

Output: bool (true if successful, false if error)

1. Open the file named fname in append mode.
2. If the file is open, write the username and password from the login object to the file.
3. Close the file and return true.
4. If the file cannot be opened, display an error message and return false.

Function verify (username, password):

Input: username (string), password (string)

Output: bool (true if verified, false if not)

1. Open the file named fname.
2. If the file is open, read username and password pairs from the file.
3. If a matching pair is found, close the file and return true.
4. If the file cannot be opened or no matching pair is found, display an error message and return false.

### **iii. Admin Class (Inherited from UserData):**

Class Admin (Inherited from UserData)

Constructor Admin(file):

Call the constructor of UserData with file

Function authenticate (username, password):

Input: username (string), password (string)

Output: bool (true if verified, false if not)

1. Call the verify method of the UserData class with the provided username and password.
2. Return the result of the verify method.

### **iv. Patient Struct:**

Struct Patient

Data:

- lastName (string)
- firstName (string)
- department (string)
- age (int)
- roomNo (int)
- isDisabled (bool)

Constructor Patient():

Initialize age and roomNo with 0

Initialize isDisabled with false

Constructor Patient(lName, fName, dept, a, room, disabled):

Initialize lastName with lName

Initialize firstName with fName

Initialize department with dept

Initialize age with a

Initialize roomNo with room

Initialize isDisabled with disabled

#### **v.    TreeNode Struct:**

Struct TreeNode

Data:

- data (Patient)
- left (TreeNode\*)
- right (TreeNode\*)

Constructor TreeNode(patient):

Initialize data with the provided Patient object

Initialize left and right as null

**vi. PatientTree Class:**

Class PatientTree

Data:

- root (TreeNode\*)

Constructor PatientTree():

Initialize root as null

Function checkInPatient(patient):

Input: Patient object patient

Output: None

1. Call insertPatient(root, patient) to insert the patient into the tree.

Function displayPatientsInOrder():

Input: None

Output: None

1. Call displayInOrder(root) to display the patients in the tree in alphabetical order.

Function insertPatient(node, patient):

Input: TreeNode node, Patient object patient

Output: TreeNode (updated)

1. If node is null, return a new TreeNode with the provided patient.
2. Prioritize disabled patients over non-disabled and older patients over younger.
3. Update the left or right child of the node accordingly.
4. Return the updated node.

Function displayInOrder(node):

Input: TreeNode node

Output: None

1. If the node is not null, recursively traverse the left subtree.
2. Display the patient information from the node.
3. Recursively traverse the right subtree.

## **vii. AdminFeat Class:**

Class AdminFeat

Data:

- patTree (PatientTree)

Function checkInPatient():

1. Create a Patient object to store patient details.
2. Prompt for and read patient details.
3. Use the PatientTree (patTree) to check in the patient.
4. Append the patient details to the "checkin.txt" file.

Function viewPatient():

1. Display a header for the patient list.
2. Open and read the "checkin.txt" file to retrieve patient records.
3. For each patient record, create a Patient object and add it to the PatientTree (patTree).
4. Call the PatientTree's displayPatientsInOrder method to display patients in alphabetical order.

Function addEmployee():

1. Prompt for and read employee details.
2. Append the employee details to the "EmployeeData.txt" file.

Function checkOutPatient():

1. Open "checkin.txt" for reading and "temp.txt" for writing.
2. Prompt for the Room No of the patient to check out.
3. While reading lines from "checkin.txt," check for the target Room No.
4. If found, mark the patient as checked out and do not copy their details to "temp.txt."
5. Close both files.
6. Replace "checkin.txt" with "temp.txt" to update the records.

Function viewEmployee():

1. Open and read the "EmployeeData.txt" file to retrieve employee records.
2. Display a header for the employee list.
3. For each employee record, display the Last Name, First Name, Position, and Department.

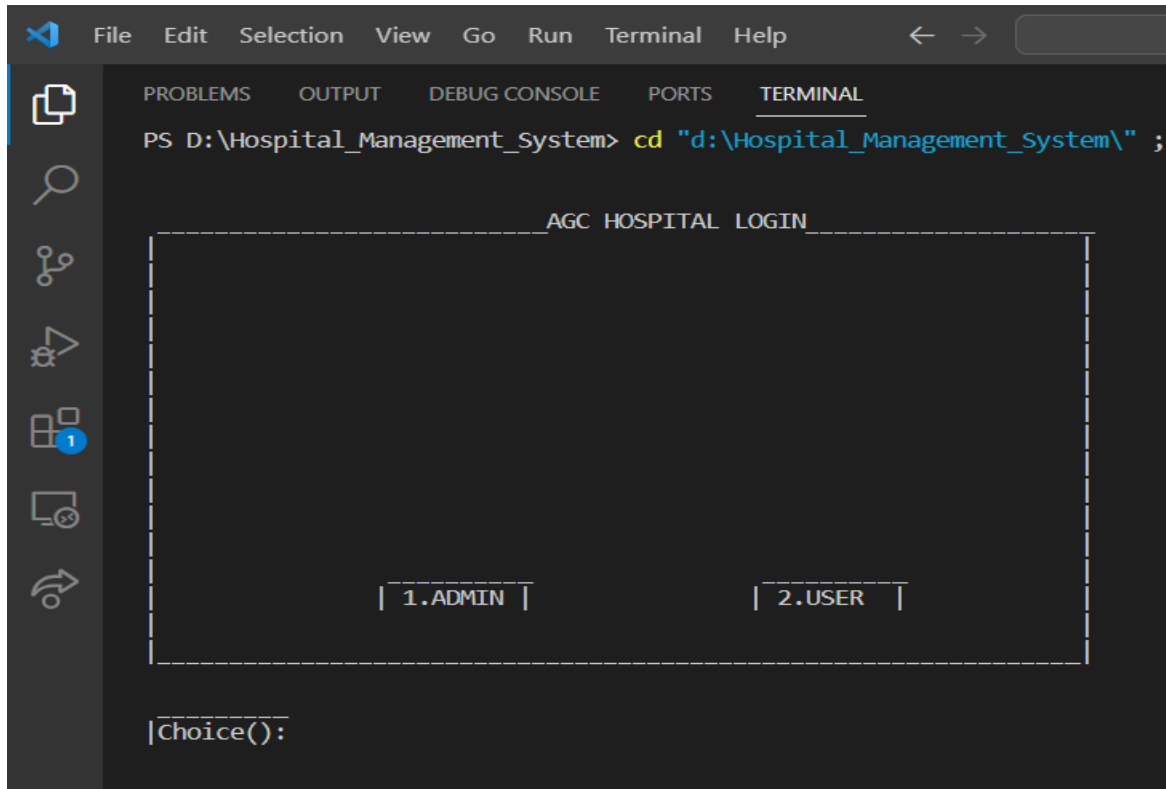
Function viewRegPat():

1. Open and read the "patient\_records.txt" file to retrieve registered patient records.
2. Display a header for the registered patient list.
3. For each patient record, display the Name, Age, Gender, and Contact Number.

Function viewAppointments():

1. Open and read the "Appointments.txt" file to retrieve appointment records.
2. Display a header for the scheduled appointments list.
3. For each appointment record, display the Patient Name, Facility Type, Facility Name, Date, and Time.

## Outputs



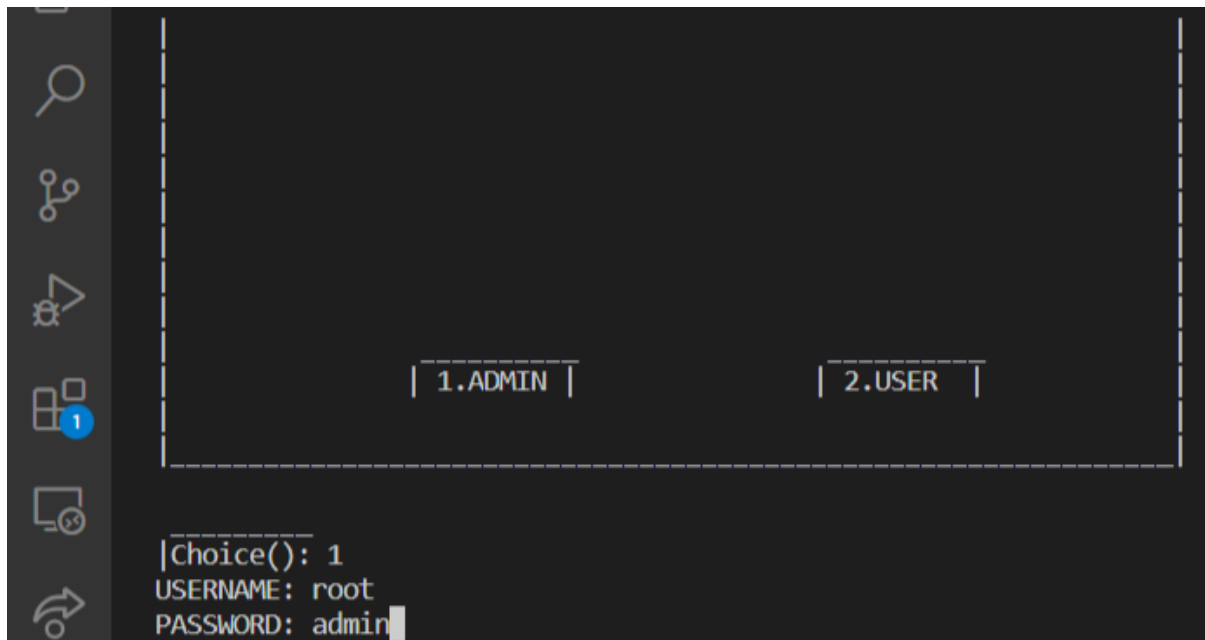
```
PS D:\Hospital_Management_System> cd "d:\Hospital_Management_System\" ;

AGC HOSPITAL LOGIN

| 1.ADMIN | | 2.USER |

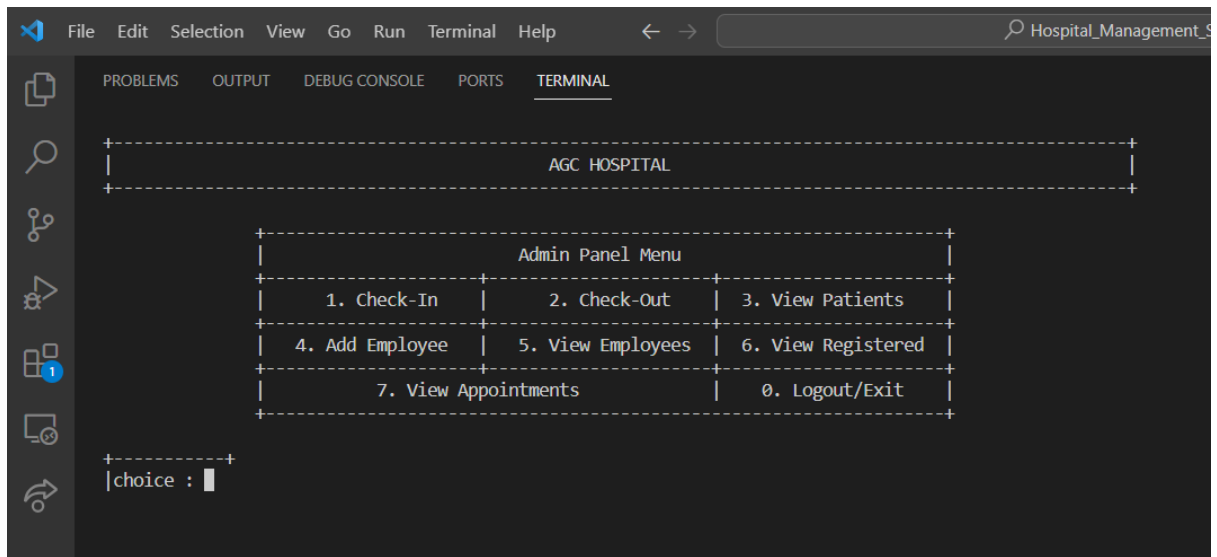
|Choice():
```

After pressing 1



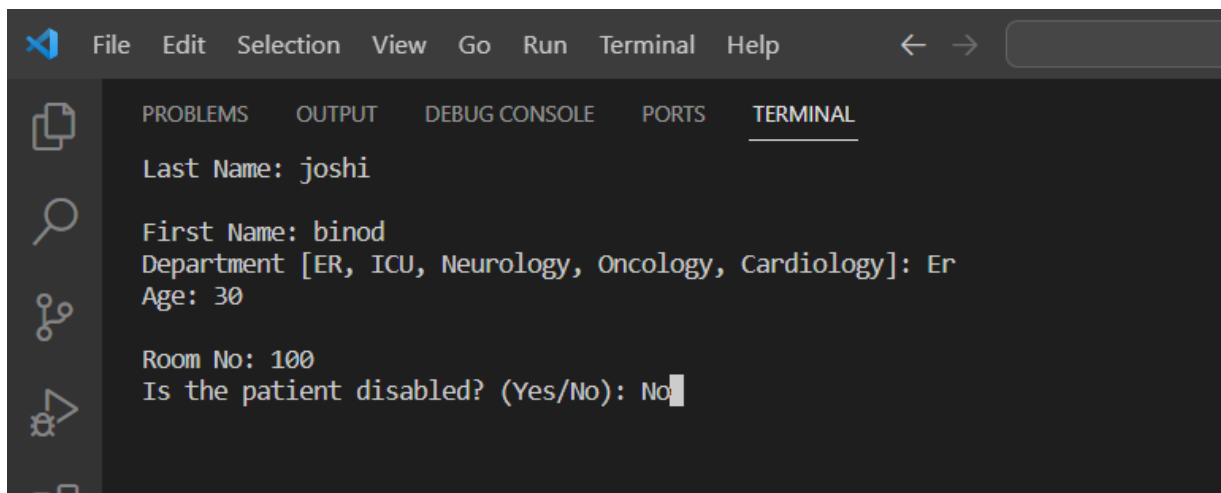
```
|Choice(): 1
USERNAME: root
PASSWORD: admin
```

After Pressing Enter



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+-----+
|                                     |
|                               AGC HOSPITAL                               |
|                                     |
|+-----+
|                                     |
|                               Admin Panel Menu                               |
|+-----+
| 1. Check-In | 2. Check-Out | 3. View Patients |
|+-----+
| 4. Add Employee | 5. View Employees | 6. View Registered |
|+-----+
| 7. View Appointments | 0. Logout/Exit |
|+-----+
|
|choice : |
```

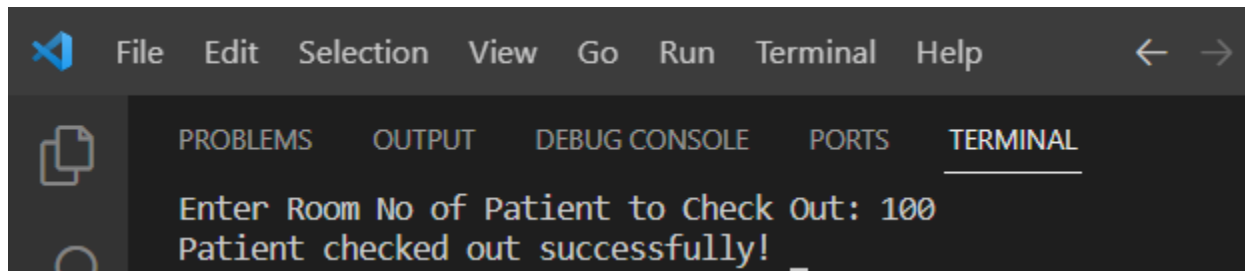
Press 1



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
Last Name: joshi
First Name: binod
Department [ER, ICU, Neurology, Oncology, Cardiology]: Er
Age: 30
Room No: 100
Is the patient disabled? (Yes/No): No
```

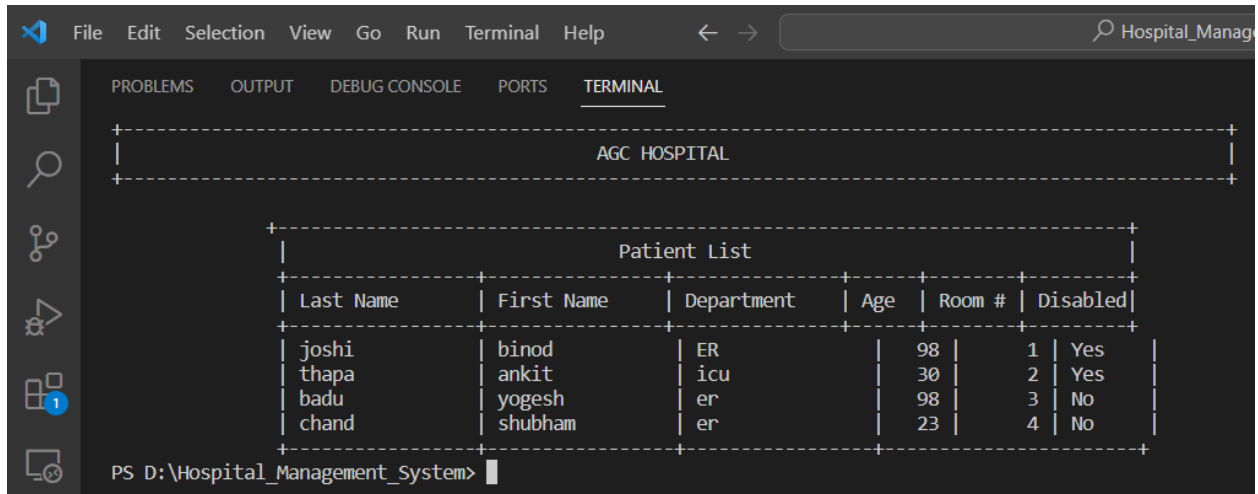


Press 2



A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel shows the 'TERMINAL' tab selected. The output text reads: 'Enter Room No of Patient to Check Out: 100' followed by 'Patient checked out successfully!' on the next line.

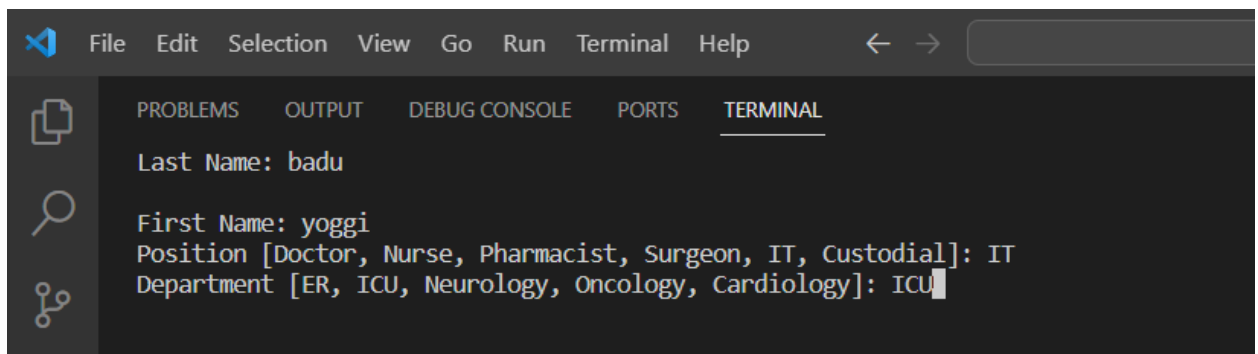
Press 3



A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel shows the 'TERMINAL' tab selected. The output text displays a table header 'AGC HOSPITAL' followed by a 'Patient List' table. The table has columns: Last Name, First Name, Department, Age, Room #, and Disabled. The data rows are: joshi, binod, ER, 98, 1, Yes; thapa, ankit, icu, 30, 2, Yes; badu, yogesh, er, 98, 3, No; and chand, shubham, er, 23, 4, No. The prompt 'PS D:\Hospital\_Management\_System>' is visible at the bottom.

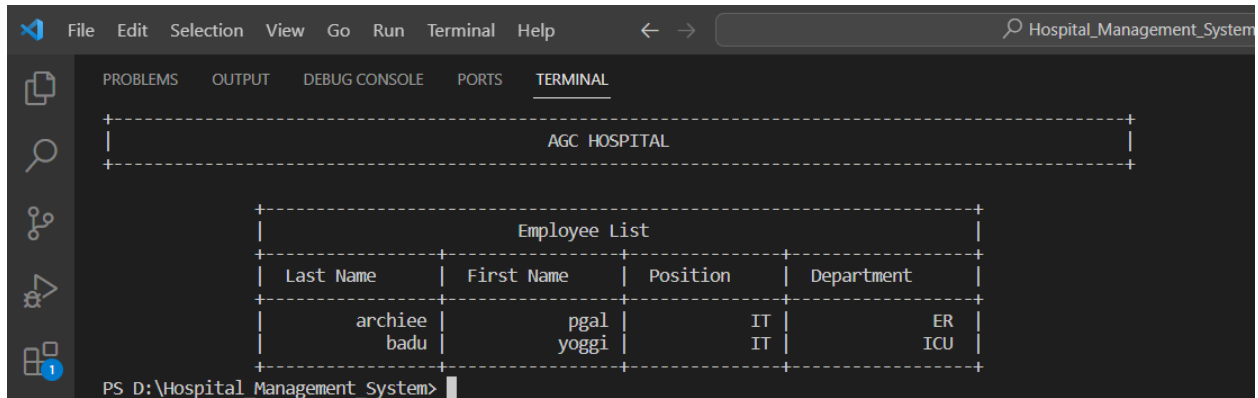
Last Name	First Name	Department	Age	Room #	Disabled
joshi	binod	ER	98	1	Yes
thapa	ankit	icu	30	2	Yes
badu	yogesh	er	98	3	No
chand	shubham	er	23	4	No

Press 4



A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel shows the 'TERMINAL' tab selected. The output text shows the user input for patient details: 'Last Name: badu', 'First Name: yoggi', 'Position [Doctor, Nurse, Pharmacist, Surgeon, IT, Custodial]: IT', and 'Department [ER, ICU, Neurology, Oncology, Cardiology]: ICU'.

Press 5

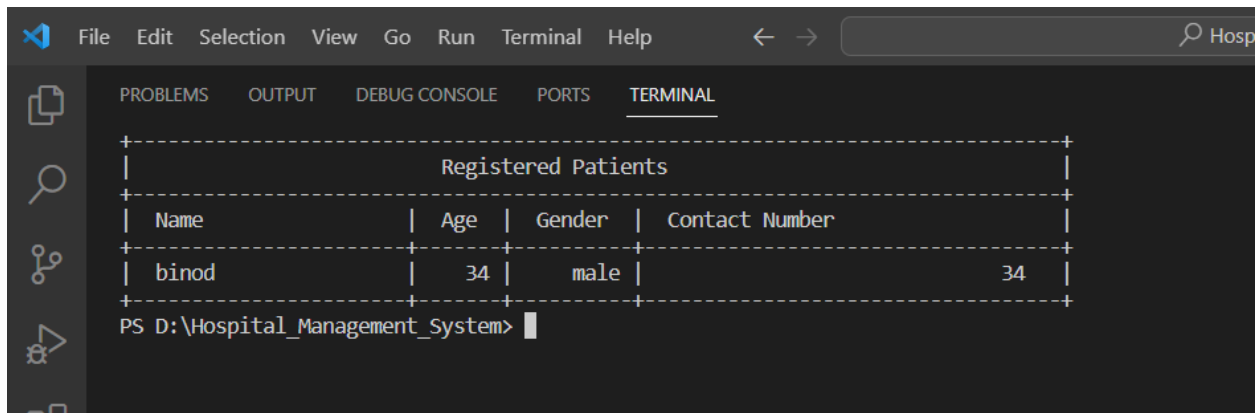


```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+-----+
|                                     |
|                               AGC HOSPITAL                               |
|                                     |
+-----+

+-----+
|                               Employee List                               |
+-----+
| Last Name | First Name | Position | Department |
+-----+
| archiee   | badu       | IT       | ER         |
| pgal      | yoggi      | IT       | ICU        |
+-----+

PS D:\Hospital_Management_System>
```

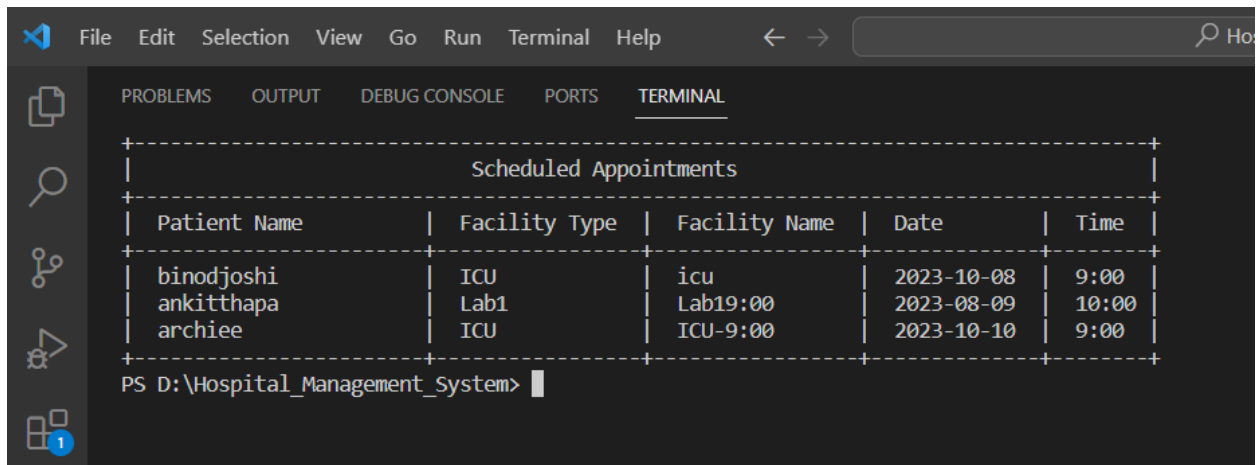
Press 6



```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+-----+
|                                     |
|                               Registered Patients                               |
|                                     |
+-----+
| Name       | Age | Gender | Contact Number |
+-----+
| binod      | 34  | male   | 34             |
+-----+

PS D:\Hospital_Management_System>
```

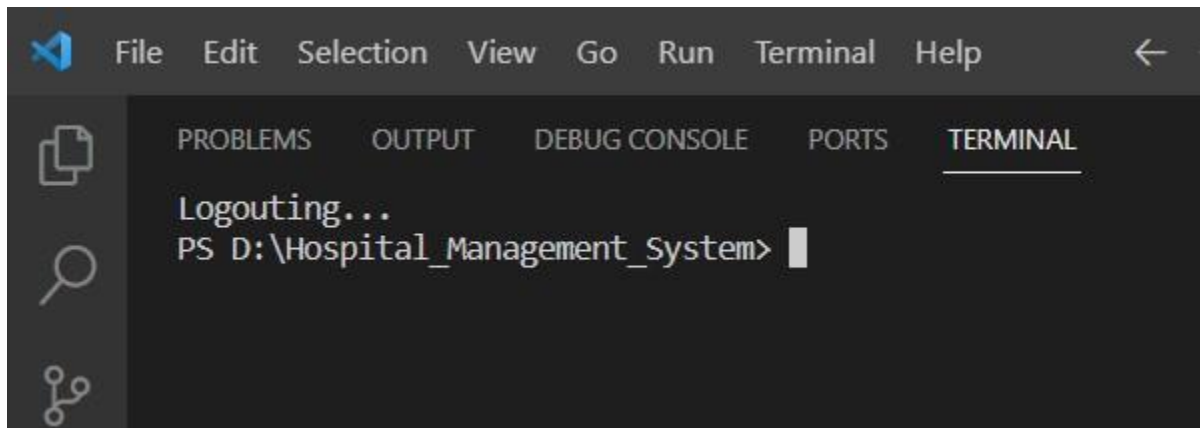
Press 7



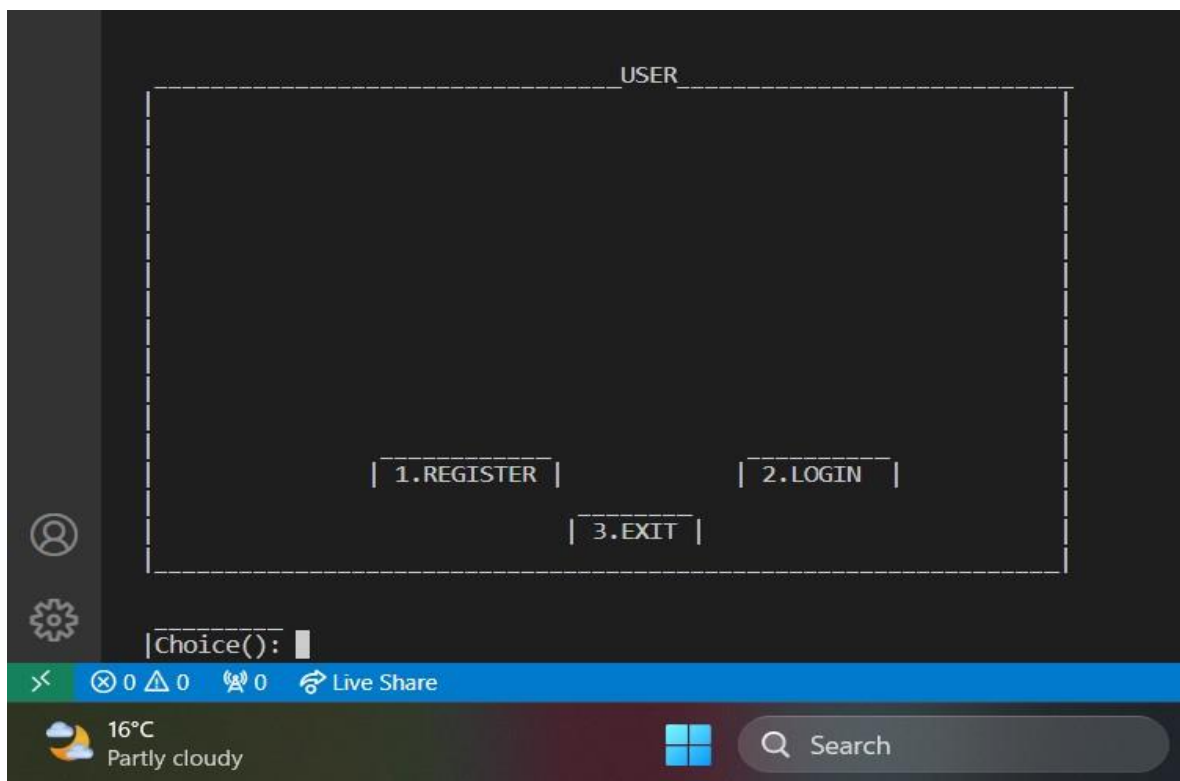
```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+-----+
|                                     |
|                               Scheduled Appointments                               |
|                                     |
+-----+
| Patient Name | Facility Type | Facility Name | Date       | Time |
+-----+
| binodjoshi   | ICU          | icu           | 2023-10-08 | 9:00 |
| ankitthapa   | Lab1         | Lab19:00      | 2023-08-09 | 10:00 |
| archiee      | ICU          | ICU-9:00      | 2023-10-10 | 9:00 |
+-----+

PS D:\Hospital_Management_System>
```

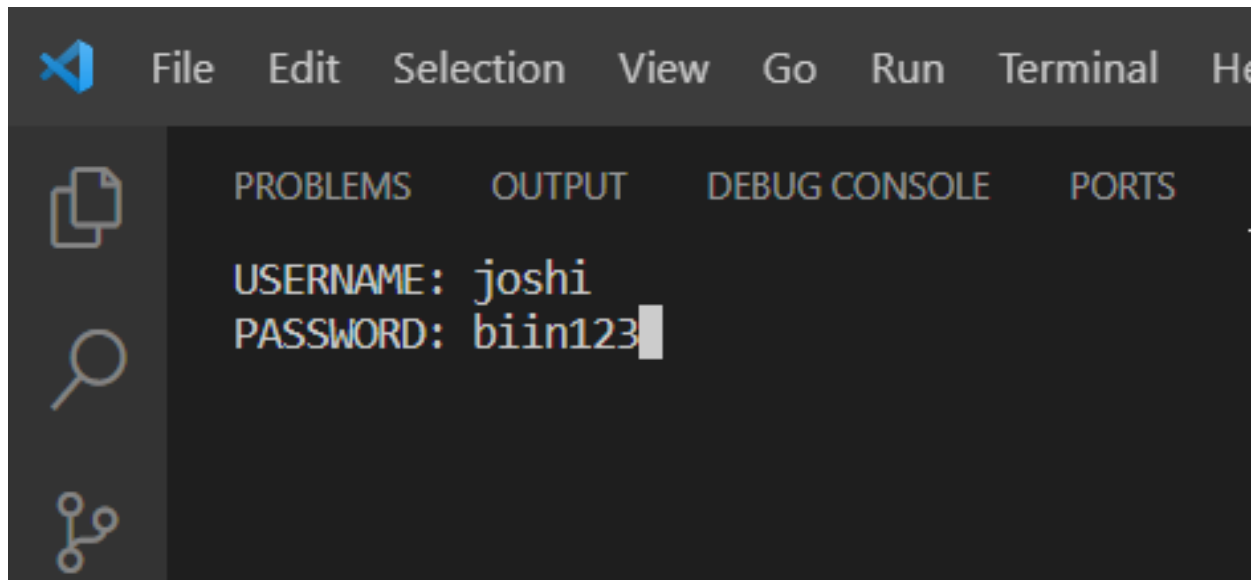
Press 0



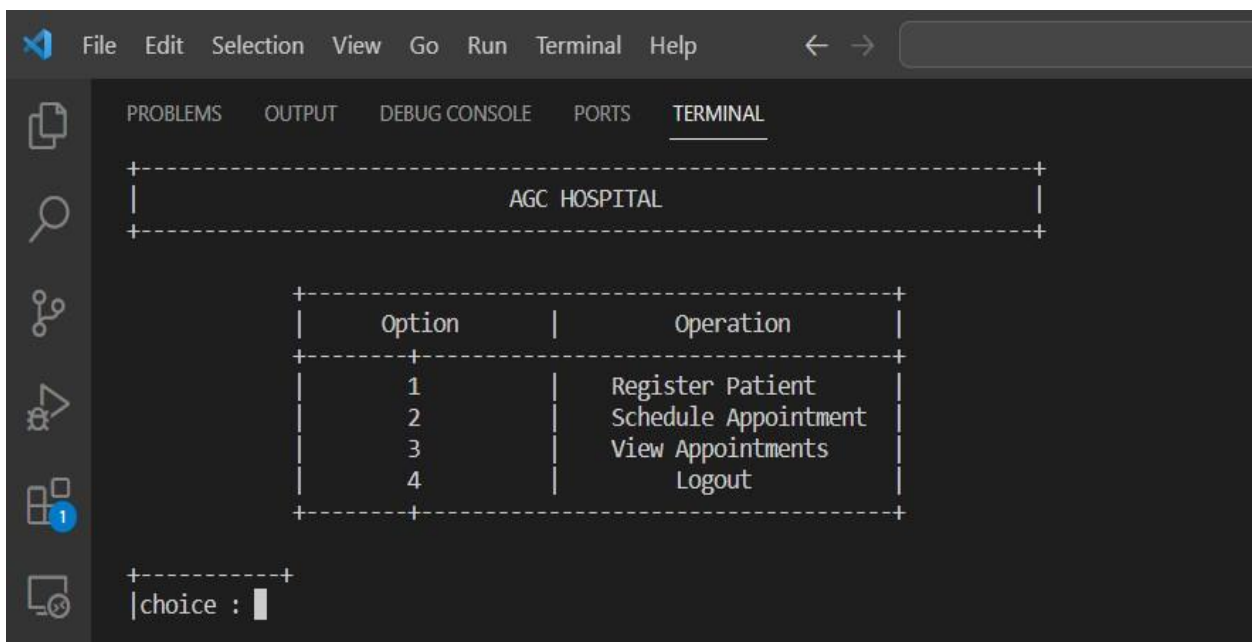
After Pressing 2 in Login Page



After Pressing 1



After pressing 2



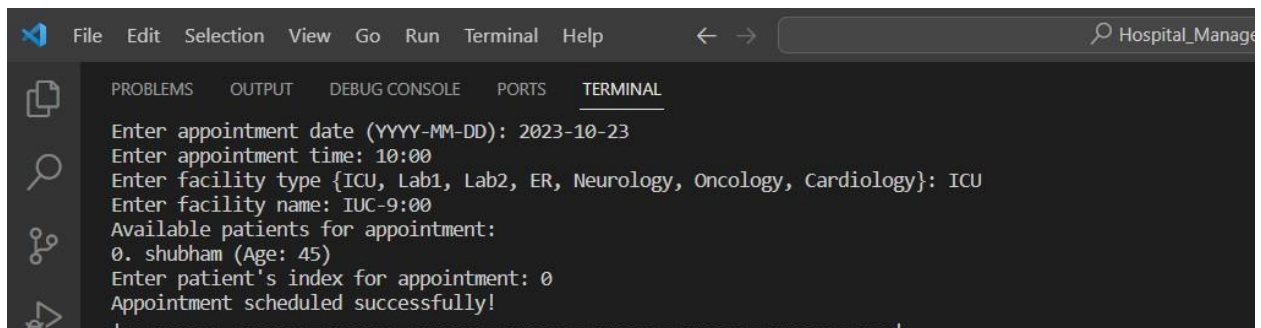
Press 1



A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel is active, showing the following text: "Enter patient name: shubham", "Enter patient age: 45", "Enter patient gender: male", "Enter contact number: 90374653", and "Patient registered successfully!". The terminal is decorated with dashed lines and a plus sign at the bottom.

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
Enter patient name: shubham
Enter patient age: 45
Enter patient gender: male
Enter contact number: 90374653
Patient registered successfully!
+-----+
```

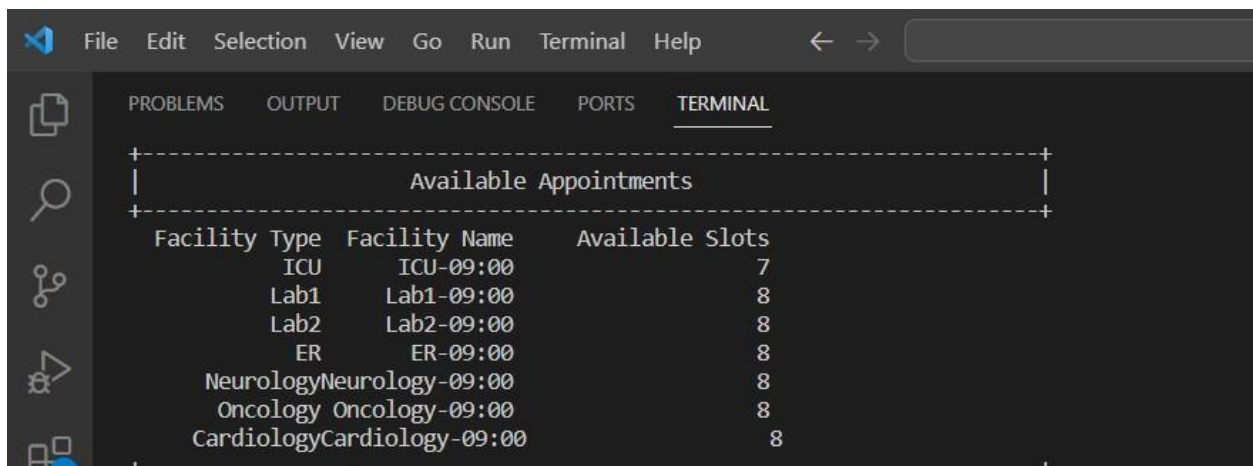
Press 2



A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel is active, showing the following text: "Enter appointment date (YYYY-MM-DD): 2023-10-23", "Enter appointment time: 10:00", "Enter facility type {ICU, Lab1, Lab2, ER, Neurology, Oncology, Cardiology}: ICU", "Enter facility name: IUC-9:00", "Available patients for appointment:", "0. shubham (Age: 45)", "Enter patient's index for appointment: 0", and "Appointment scheduled successfully!". The terminal is decorated with dashed lines and a plus sign at the bottom.

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
Enter appointment date (YYYY-MM-DD): 2023-10-23
Enter appointment time: 10:00
Enter facility type {ICU, Lab1, Lab2, ER, Neurology, Oncology, Cardiology}: ICU
Enter facility name: IUC-9:00
Available patients for appointment:
0. shubham (Age: 45)
Enter patient's index for appointment: 0
Appointment scheduled successfully!
+-----+
```

Press 3



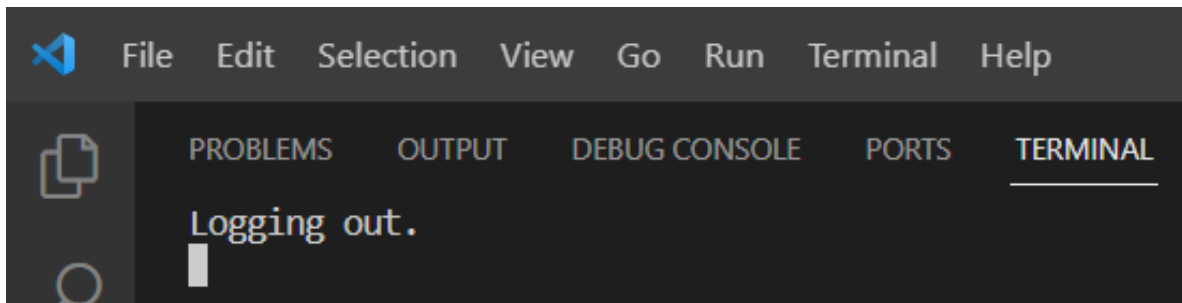
A screenshot of the Visual Studio Code terminal window. The menu bar at the top includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The terminal panel is active, showing a table of available appointments. The table has three columns: Facility Type, Facility Name, and Available Slots. The data is as follows:

Facility Type	Facility Name	Available Slots
ICU	ICU-09:00	7
Lab1	Lab1-09:00	8
Lab2	Lab2-09:00	8
ER	ER-09:00	8
Neurology	Neurology-09:00	8
Oncology	Oncology-09:00	8
Cardiology	Cardiology-09:00	8

The terminal is decorated with dashed lines and a plus sign at the bottom.

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL
+-----+
|                                     |
|               Available Appointments               |
|-----+-----+-----+-----+-----+-----+
| Facility Type | Facility Name | Available Slots |
|-----+-----+-----+-----+-----+-----+
| ICU           | ICU-09:00     | 7               |
| Lab1          | Lab1-09:00    | 8               |
| Lab2          | Lab2-09:00    | 8               |
| ER            | ER-09:00      | 8               |
| Neurology     | Neurology-09:00 | 8               |
| Oncology      | Oncology-09:00 | 8               |
| Cardiology    | Cardiology-09:00 | 8               |
+-----+-----+-----+-----+-----+-----+
+-----+
```

Press 4



## References

### Web URLs

[https://w3schools.com/cpp/cpp\\_files.asp](https://w3schools.com/cpp/cpp_files.asp)

<https://www.geeksforgeeks.org/how-to-work-with-file-handling-in-c/>

[https://www.slideteam.net/media/catalog/product/cache/960x720/p/r/project\\_phases\\_showing\\_initiation\\_planning\\_and\\_execution\\_with\\_project\\_scope\\_and\\_analysis\\_Slide01.jpg](https://www.slideteam.net/media/catalog/product/cache/960x720/p/r/project_phases_showing_initiation_planning_and_execution_with_project_scope_and_analysis_Slide01.jpg)

<https://www.geeksforgeeks.org/tree-data-structure/>