

# C++ Programming Internship




DATE : 20 July – 20 August  
COURSE TITLE : Internship

STUDENT'S NAME Shehryar  
Ahmed Khan  
Student Id : CA/JUI/33416



## INTRODUCTION :

During this internship, I worked on four core C++ projects designed to strengthen my programming fundamentals and problem-solving skills. I developed a **CGPA Calculator** to compute academic performance using weighted grade logic. In the **Login and Registration System**, I implemented secure credential handling with file-based storage. The **Sudoku Solver** improved my understanding of recursive algorithms and constraint logic. Lastly, the **Banking System** allowed me to apply OOP concepts like classes and encapsulation for real-world financial operations. These tasks helped me enhance my coding structure, logical thinking, and real-life application skills in C++.



---

## OBJECTIVES

### Task 1: CGPA Calculator

To build a console-based tool that calculates semester GPA and overall CGPA by taking course grades and credit hours as input.

### Task 2: Login and Registration System


To implement a secure system for user registration and login using file handling for credential storage and validation.

### Task 3: Sudoku Solver

To create a Sudoku puzzle solver using the backtracking algorithm that adheres to standard Sudoku rules across rows, columns, and sub grids.

### Task 4: Banking System

To design a basic banking system using object-oriented programming principles, enabling account creation, fund transfers, and transaction history tracking.



## Task 1 :

Smart GPA — Semester & CGPA Calculator in C++

### OBJECTIVE :

To develop a C++ console-based application that helps students calculate their semester GPA and overall CGPA by inputting subject grades and credit hours.

### Output Presentation :

Displays a well-formatted table showing:

- Subject Name
- Letter Grade
- Credit Hours
- Grade Points

Clearly shows Semester GPA and CGPA (if opted)

## KEY FUNCTIONALITIES :

### Input Handling:

- Takes the total number of courses from the user.
- Accepts user-defined **subject names**, **letter grades**, and **credit hours** for each course.

### Grade Conversion Logic:

- Converts letter grades (e.g., A, B+, C-) into **grade points** using a standardized grading scale.
- Ensures data validity and prompts an error for invalid grades.

### GPA Calculation:

- Calculates **semester GPA** using the formula:  

$$\text{GPA} = \frac{\sum(\text{Grade Point} \times \text{Credit Hour})}{\sum(\text{Credit Hours})}$$

### CGPA Calculation (Optional) :

- If the user wants to calculate cumulative CGPA, the program accepts previous CGPA and total credit hours, then calculates:  

$$\text{New CGPA} = \frac{\text{Total Credits}(\text{Previous CGPA} \times \text{Previous Credits}) + \text{Current Grade Points}}{\text{Total Credits}}$$

### Technical Concepts Practiced:

- Use of functions.
- Vectors for dynamic data storage.
- String and character input management.

```

C:\Users\shery\OneDrive\Des  X  +  v
===== CGPA Calculator =====
Enter number of courses: 2

Course 1 name: OOP
Grade for OOP (A, B+, C-, etc.): A
Credit hours for OOP: 3

Course 2 name: Information sec
Grade for Information sec (A, B+, C-, etc.):
A-
Credit hours for Information sec: 3

===== Grade Summary =====
Subject      Grade      Credit Hours  Points
  OOP          A           3           4
Information sec  A-          3          3.7

?? Semester GPA: 3.85

Do you want to calculate overall CGPA? (Y/N): y
Enter previous CGPA: 3.5
Enter total previous credit hours: 5
Overall CGPA: 3.69

Calculation Complete. Keep up the hard work!

```

### Sample Output

## Task 2 :

### Secure Login and Registration System in C++

#### OBJECTIVE :

- To develop a basic user authentication system using C++.
- Enable users to register with a unique username and secure password.
- Allow login functionality with correct verification.
- Learn file handling and secure input techniques.
- Reinforce use of conditionals, functions, and standard C++ libraries.

#### Tools Used :

- **IDE:** Dev-C++ / VS Code with g++
- **Language:** C++
- **Concepts:** File I/O, HashMap (unordered\_map), Conditional Logic, Functions

## Breakdown of Code :

### 1. File Handling:

ifstream file(filename); ( Read user credentials from file )

ofstream file(filename, ios::app); (Append new users )

### 2. User Storage:







unordered\_map<string, string> users; (Efficient lookup for usernames)

### 3. Registration:

Inputs a new username and checks if it already exists.

Takes hidden password input using getch() and stores it.

Appends the credentials to users.txt.

 loginsystem		14/07/2025 1:03 pm	C++ Source File	3 KB
 loginsystem		14/07/2025 1:04 pm	Application	1,897 KB
 users		14/07/2025 1:04 pm	Text Document	1 KB

### 4. Login:

Requests username and password.

Compares with stored credentials.

Displays appropriate success or failure message.

// Program Checks for credentials if already exists it will show a error message.

```
===== Login & Registration Menu =====
1. Register
2. Login
3. Exit
Enter choice: 1

--- User Registration ---
Enter a new username: n1x5
Username already exists. Try another.
```

Already User Exists

```
--- User Login ---
Username: n1x5
Password: ***
Login successful. Welcome, n1x5!
```

Successful Login

## Learning Outcomes:

Gained hands-on experience with file-based credential management.  
Learned to secure password input from terminal using ASCII input.  
Improved understanding of how real-world login systems are built.  
Practiced logic branching and user feedback design.  
Strengthened file I/O and data mapping knowledge.

## Conclusion:

This task simulated the backend of an authentication system. It highlights practical skills applicable to login modules in real applications. The password masking adds a user-friendly security layer while demonstrating intermediate-level C++ capabilities.

---

## Task 3 :

### Title:

### Interactive Console-Based Sudoku Game using Backtracking

### Introduction:

This C++ program implements a user-interactive Sudoku solver designed as a learning aid and puzzle game. Users can guess numbers to solve a 9×9 Sudoku grid. The system verifies each guess in real time against a predefined valid solution. It demonstrates practical use of arrays, loops, conditionals, and logic validation, and forms a base for understanding backtracking and constraint solving techniques.

### Objectives:

- Reinforce understanding of 2D arrays and nested loops in C++
- Apply conditional logic to check grid constraints
- Provide real-time feedback to user inputs
- Allow interaction until complete solution is achieved
- Simulate a game-like environment for educational engagement
- Enhance critical thinking using Sudoku rules

## Core Functionalities:

Feature	Description
9×9 Grid Initialization	Starts with a partially filled Sudoku puzzle
User Guessing	Accepts user input for empty cells (row, column, number)
Answer Verification	Compares user input against solved puzzle
Error Handling	Displays error if the input is incorrect
Puzzle Completion Detection	Ends the program once all cells are correctly filled
Visual Representation	Clearly prints the Sudoku grid after every move with logical formatting

## Logic Used:

- Grid is represented using a 2D array: `int puzzle[9][9]`
- A hardcoded valid solution `[9][9]` is used for input validation
- Inputs for row, column, and number are taken
- Validity check ensures:
  - Only empty cells can be filled
  - Guesses are matched with solution array
  - After each valid guess, the grid is reprinted
- Game ends when no zero remains in the puzzle

## Code Structure Breakdown:

Function / Block	Purpose
<code>printGrid()</code>	Neatly formats and displays the current state of the Sudoku grid
<code>isPuzzleComplete()</code>	Checks whether all cells are filled (no zeros left)
<code>main()</code>	Orchestrates game loop, input, validation, and state updates

## Learning Outcomes:

Skill / Topic	Gained Knowledge
2D Array Manipulation	Grids and board-based data structures
Input Validation	Ensuring correct and expected user data
Control Flow	Use of while, if, and nested blocks
Game Design Basics	Turn-based input/output in CLI UI
Debugging	Real-time feedback for incorrect input

## Output :

```

C:\Users\shery\OneDrive\Des  ×  +  ▾  -  □  ×

===== Sudoku Interactive Game =====
Note: Rows and columns are 0-indexed (0 to 8).

Current Sudoku Grid:
5 3 . | . 7 . | . . .
6 . . | 1 9 5 | . . .
. 9 8 | . . . | . 6 .
-----+-----+-----
8 . . | . 6 . | . . 3
4 . . | 8 . 3 | . . 1
7 . . | . 2 . | . . 6
-----+-----+-----
. 6 . | . . . | 2 8 .
. . . | 4 1 9 | . . 5
. . . | . 8 . | . 7 9

```

// Firstly , It will shows unsolved puzzle .

```

Enter row (0-8): 0
Enter column (0-8): 2
Enter your guess (1-9): 4
? Correct!

```

It will ask user to enter row and column number user want to guess the number if its correct number will shown on place otherwise incorrect message will shown .



## Task 4 :

### Title :

Banking System

### Project Introduction :

The **n1x5 Banking System** is a basic C++ console application designed to simulate fundamental operations in a banking environment. The system offers user registration, secure login, balance inquiry, money transactions (deposit, withdraw, send money), and transaction history logging. It integrates concepts such as **file handling, password masking, object-oriented design, and secure authentication.**

### Objectives :

- Apply object-oriented programming to model a banking system.
- Use file I/O to persist user and transaction data.
- Implement data privacy via password masking using `_getch()`.
- Simulate basic banking functionalities like deposit, withdrawal, and funds transfer.
- Demonstrate backend logic for e-banking platforms like PayPal, SadaPay, etc.
- Generate transaction receipts and logs for audit purposes.

### Features Breakdown :

#### 1. User Registration

- Collects username, password (masked), account type (Saving/Current), and CVV PIN.
- Saves data securely in a `users.txt` file.
- Validates unique usernames.

#### 2. User Login

- Authenticates using username and password.
- Further verifies identity using a CVV PIN.

#### 3. Banking Operations (After Login)

- Deposit: Adds money to user's account and logs it.
- Withdraw: Subtracts funds if balance is sufficient.
- Check Balance: Displays current balance.
- Send Money:
  - Choose platform (PayPal, NayaPay, SadaPay, Paytm, Bank Transfer).
  - Enter account number/ID and amount.
  - Validates balance and records transaction.

#### 4. Transaction Logging

- Every operation is logged into `transactions.txt` with timestamp, user, and action.





- Acts as both audit trail and receipt for users.

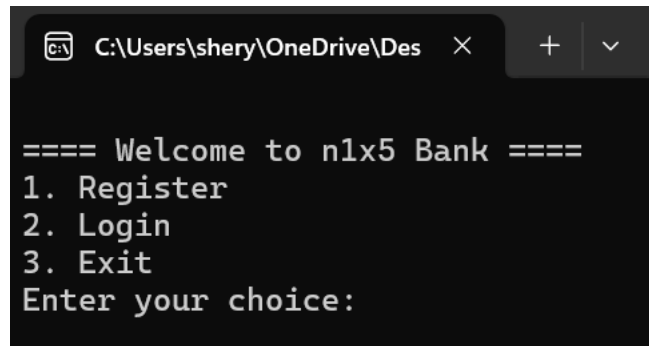
## Technical Implementation Breakdown :

Component	Description
struct Account	Stores user data: username, password, account type, CVV, balance.
loadAccounts()	Reads account data from file into a hash map.
saveAccounts()	Writes updated account data back to file.
getHiddenPassword()	Hides password input using <code>_getch()</code> from <code>&lt;conio.h&gt;</code> .
writeReceipt()	Appends logs/transactions to transactions.txt with timestamp.
registerUser()	Handles new user creation.
loginUser()	Authenticates users and provides banking menu.
main()	Entry point. Displays welcome screen and prompts for action.

## Files Created :

- users.txt – Stores user data.
- transactions.txt – Stores all transaction logs (receipts with timestamp).

 transactions		14/07/2025 3:53 pm	Text Document	1 KB
 users		14/07/2025 3:46 pm	Text Document	1 KB

A screenshot of a terminal window with a dark background. The title bar shows the file path 'C:\Users\shery\OneDrive\Des' and standard window controls. The terminal text reads: '==== Welcome to n1x5 Bank ====', followed by a numbered list: '1. Register', '2. Login', '3. Exit'. Below the list is the prompt 'Enter your choice:'.

```
C:\Users\shery\OneDrive\Des  ×  +  ∨  
  
==== Welcome to n1x5 Bank ====  
1. Register  
2. Login  
3. Exit  
Enter your choice:
```

Welcome Screen

```
-- Registration --  
Enter Username: Intern  
Enter Password: ***  
Select Account Type (Saving/Current): 2  
Set a 4-digit CVV PIN: 2888  
Registration successful.
```

Registration Process

```
Welcome, Intern!  
1. Deposit  
2. Withdraw  
3. Check Balance  
4. Send Money  
5. Exit  
Choice: |
```

Bank Menu

```
Welcome, Intern!  
1. Deposit  
2. Withdraw  
3. Check Balance  
4. Send Money  
5. Exit  
Choice: 1  
Enter deposit amount: 1000  
Deposited successfully.
```

Money Deposited

```

Welcome, Intern!
1. Deposit
2. Withdraw
3. Check Balance
4. Send Money
5. Exit
Choice: 2
Enter withdrawal amount: 200
Withdrawn successfully.

```

#### Money Withdraw

```

Welcome, Intern!
1. Deposit
2. Withdraw
3. Check Balance
4. Send Money
5. Exit
Choice: 3
Your balance is: $800

```

#### Balance Checked

```

Welcome, Intern!
1. Deposit
2. Withdraw
3. Check Balance
4. Send Money
5. Exit
Choice: 4
Select Method to Send Money:
1. PayPal
2. NayaPay
3. SadaPay
4. Paytm
5. Bank Transfer
Choice: 3
Enter recipient account number or ID: 9383
Enter amount to send: 400
Money sent successfully.

```

#### Money Transfer

```
File Edit View

Intern 123 2 2888 0
n1x5 123 1 2888 0
```

### Users History

transactions.txt			
File	Edit	View	
Mon Jul 14 15:47:03 2025	User: n1x5	Operation: Deposit	Amount: \$5000
Mon Jul 14 15:52:54 2025	User: n1x5	Operation: Check Balance	Amount: \$0
Mon Jul 14 15:53:05 2025	User: n1x5	Operation: Deposit	Amount: \$900000
Mon Jul 14 15:53:18 2025	User: n1x5	Operation: Send Money	Amount: \$1000   Details: PayPal to 908238293
Mon Jul 14 15:53:20 2025	User: n1x5	Operation: Check Balance	Amount: \$0
Mon Jul 14 16:05:24 2025	User: Intern	Operation: Registration	Amount: \$0
Mon Jul 14 16:07:16 2025	User: Intern	Operation: Deposit	Amount: \$1000
Mon Jul 14 16:08:34 2025	User: Intern	Operation: Withdraw	Amount: \$200
Mon Jul 14 16:09:18 2025	User: Intern	Operation: Check Balance	Amount: \$0
Mon Jul 14 16:10:47 2025	User: Intern	Operation: Send Money	Amount: \$400   Details: <u>SadaPay</u> to 9383

### User Activity History