



Object Oriented Programming

Submitted by: Mesum & Mehnaz Ameen

Project Title: Typing Master in C++

Instructor: Nafeesa Ambreen

**Department of Computer Science
UNIVERSITY OF BALTISTAN, SKARDU**

Table of Contents

1.	Project Introduction	3
2.	Problem Statement / Understanding the Problem.....	3
3.	Design the Solution	3
4.	Pseudocode.....	4
5.	Develop Algorithm.....	4
6.	Flowchart Diagram.....	5
7.	Main Modules	6
8.	Screenshots	6
9.	OOP Concepts Used	13
10.	Extra Terms: Explanation, Purpose & Usage.....	14
11.	Conclusion	14

1. Project Introduction

Typing Master in C++ is a console-based program designed to test and improve typing speed and accuracy. It applies Object-Oriented Programming (OOP) principles and file handling to manage user registration, login, and leaderboard functionality. The project helps users track their typing performance in an offline environment using a simple and efficient design.

2. Problem Statement / Understanding the Problem

Many users want to measure their typing speed and accuracy without relying on online tools. Existing tools require an internet connection and lack personal tracking. This project aims to build an offline C++ typing application that allows users to:

- Register and log in
 - Take typing tests at different difficulty levels
 - Automatically calculate speed, accuracy, and time
 - Save results for future reference
-

3. Design the Solution

The solution is built using OOP concepts for modularity and reusability.

- **User Class:** Handles registration and login using file handling.
- **Abstract Test Class:** Defines structure and functions for tests.
- **Derived Classes (EasyTest, HardTest):** Implement difficulty-specific test logic.
- **File Handling:** Stores user and leaderboard data permanently.

Flow of Control :

User -> Signup/Login -> Choose Test Level -> Type Text ->
Calculate Speed & Accuracy -> Display & Save Results

4. Pseudocode

```
BEGIN
DISPLAY main menu
IF user selects SIGNUP THEN
ASK username & password
SAVE user in file
ELSE IF user selects LOGIN THEN
VERIFY credentials
IF valid THEN
SHOW typing test options
IF Easy Test THEN
DISPLAY easy text
ELSE IF Hard Test THEN
DISPLAY hard text
ENDIF
START timer
TAKE user input
STOP timer
CALCULATE speed, accuracy, time
SAVE in leaderboard
ENDIF
ENDIF
END
```

5. Develop Algorithm

1. Start program
2. Display menu (Signup, Login, View Leaderboard, Exit)
3. If Signup selected → Save user credentials
4. If Login selected → Validate user

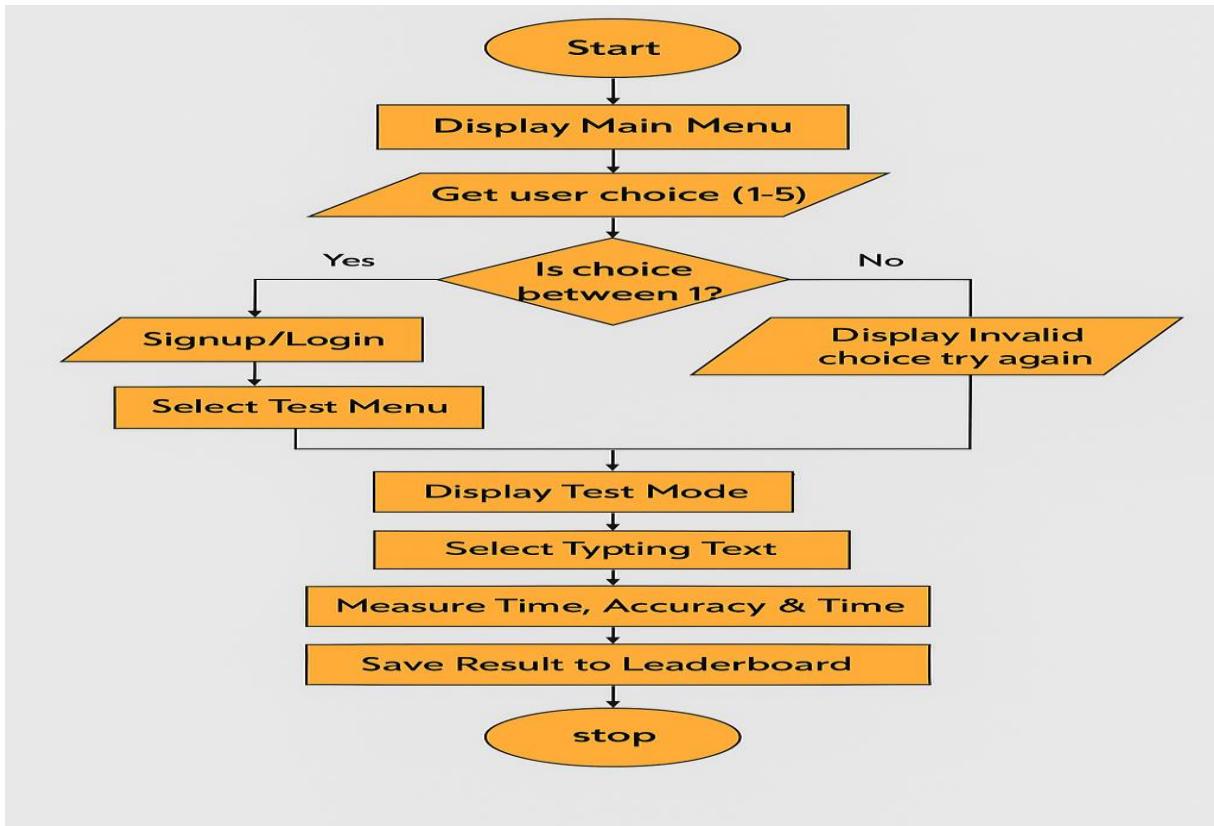
5. If successful login → Display test menu
6. Start typing test timer
7. Take input → Stop timer

8. Calculate:
 8. Speed = (characters / 5) / (time / 60)
 9. Accuracy = (correct characters / total characters) * 100

10. Display and save results

End program

6. Flowchart Diagram



7. Main Modules

Module	Description
User	Manages signup, login, and file storage of users
Test (Abstract)	Base class defining test logic
EasyTest	Implements easy-level test
HardTest	Implements hard-level test
File Handling	Reads/writes user and leaderboard data
Menu System	Handles navigation and input

8. Screenshots

Source Code

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include <ctime>
using namespace std;

class User {
private:
    string username;
    string password;

public:
    User(string u = "", string p = "") {
        username = u;
        password = p;
    }

    string getUsername() {
        return username;
    }
    string getPassword() {
        return password;
    }

    void setUsername(string u) {
        username = u;
    }
    void setPassword(string p) {
        password = p;
    }
}
```

```
void setPassword(string p) {
    password = p;
}

void saveUser() { // Save user in clean table
    fstream check("users.txt", ios::in);
    bool fileExists = check.good();
    check.close();

    ofstream fout("users.txt", ios::app);
    if (!fileExists) {
        fout << "=====\\n";
        fout << "      Registered Users\\n";
        fout << "=====\\n";
        fout << left << setw(15) << "Username" << setw(15) << "Password" << endl;
        fout << "-----\\n";
    }

    fout << left << setw(15) << username << setw(15) << password << endl;
    fout.close();
}

static bool userExists(string u) { // Check if user exists
    ifstream fin("users.txt");
    if (!fin) return false;
    string usr, pass, line;

    // Skip header Lines
    for (int i = 0; i < 5 && getline(fin, line); i++) {

        while (fin >> usr >> pass) {
            if (usr == u) {
                fin.close();
                return true;
            }
        }
        fin.close();
    }
    return false;
}

static bool login(string u, string p) { // Verify Login
    ifstream fin("users.txt");
    if (!fin) return false;
    string usr, pass, line;

    for (int i = 0; i < 5 && getline(fin, line); i++) //Skip header Lines

        while (fin >> usr >> pass) {
            if (usr == u && pass == p) {
                fin.close();
                return true;
            }
        }
}
```

```

        fin.close();
        return false;
    }
}

class Test { // ABSTRACT CLASS: TEST
protected:
    string text;
    float timeTaken;
    float speed;
    float accuracy;

public:
    virtual void startTest(string username) = 0; // pure virtual function

    float calculateSpeed(int words, float seconds) {
        return (words / (seconds / 60)); // WPM formula
    }

    float calculateAccuracy(string input) {
        int correct = 0;
        for (size_t i = 0; i < input.size() && i < text.size(); i++)
            if (input[i] == text[i]) correct++;
        return ((float)correct / text.size()) * 100;
    }

    void saveResult(string username) {
        fstream check("leaderboard.txt", ios::in);
        bool fileExists = check.good();
        check.close();

        ofstream out("leaderboard.txt", ios::app);
        if (!fileExists) {
            out << "=====\\n";
            out << "          Typing Master Leaderboard\\n";
            out << "=====\\n";
            out << left << setw(15) << "Username"
                << setw(15) << "Speed(WPM)"
                << setw(15) << "Accuracy(%)"
                << setw(15) << "Time(sec)" << endl;
            out << "-----\\n";
        }

        out << left << setw(15) << username
            << setw(15) << fixed << setprecision(2) << speed
            << setw(15) << fixed << setprecision(2) << accuracy
            << setw(15) << fixed << setprecision(2) << timeTaken << endl;
        out.close();
    }
}

class EasyTest : public Test { // CLASS: EASY TEST
public:
    EasyTest() {
        text = "The quick brown fox jumps over the lazy dog";
    }
    void startTest(string username) override {
        cout << "\\nTyping Test (EASY):\\n" << text << "\\n\\nStart typing:\\n";
        cin.ignore();
        string input;
        clock_t start = clock();
        getline(cin, input);
        clock_t end = clock();

        timeTaken = float(end - start) / CLOCKS_PER_SEC;
        int words = input.size() / 5;
        speed = calculateSpeed(words, timeTaken);
        accuracy = calculateAccuracy(input);

        cout << "\\n===== Result =====\\n";
        cout << "Time: " << timeTaken << " sec\\n";
        cout << "Speed: " << speed << " WPM\\n";
    }
}

```

```

        cout << "Accuracy: " << accuracy << "%\n";
        saveResult(username);
    }
}

class HardTest : public Test { // CLASS: HARD TEST
public:
    HardTest() {
        text = "Object oriented programming in C++ allows classes, inheritance and polymorphism to create reusable code.";
    }
    void startTest(string username) override {
        cout << "\nTyping Test (HARD):\n" << text << "\n\nStart typing:\n";
        cin.ignore();
        string input;
        clock_t start = clock();
        getline(cin, input);
        clock_t end = clock();

        timeTaken = float(end - start) / CLOCKS_PER_SEC;
        int words = input.size() / 5;
        speed = calculateSpeed(words, timeTaken);
        accuracy = calculateAccuracy(input);

        cout << "\n===== Result =====\n";
        cout << "Time: " << timeTaken << " sec\n";
        cout << "Speed: " << speed << " WPM\n";
        cout << "Accuracy: " << accuracy << "%\n";

        saveResult(username);
    }
}

void viewLeaderboard() { // VIEW LEADERBOARD
    ifstream fin("leaderboard.txt");

    if (!fin) {
        cout << "\nNo leaderboard data found yet.\n";
        return;
    }
    cout << "\n===== Leaderboard =====\n";
    string line;
    while (getline(fin, line))
        cout << line << endl;
    fin.close();
}

void viewUsers() { // VIEW USERS
    ifstream fin("users.txt");
    if (!fin) {
        cout << "\nNo users registered yet.\n";
        return;
    }
    cout << "\n===== Registered Users =====\n";
    string line;
    while (getline(fin, line))
        cout << line << endl;
    fin.close();
}

void signup() { // SIGNUP FUNCTION
    string u, p;
    cout << "\nEnter new username: ";
    cin >> u;
    if (User::userExists(u)) {
        cout << "Username already exists! Try another.\n";
        return;
    }
    cout << "Enter password: ";
    cin >> p;

    User newUser(u, p);
    newUser.saveUser();
    cout << "User registered successfully!\n";
}

```

```

}

bool login(string &loggedUser) { // LOGIN FUNCTION
    string u, p;
    cout << "\nEnter username: ";
    cin >> u;
    cout << "Enter password: ";
    cin >> p;

    if (User::login(u, p)) {
        loggedUser = u;
        cout << "\nLogin successful! Welcome, " << u << "!\n";
        return true;
    } else {
        cout << "Invalid credentials.\n";
        return false;
    }
}

int main() {
    string loggedUser;
    int choice;

    while (true) {
        cout << "\n=====\n";
        cout << "      Typing Master\n";
        cout << "      Developed by Mesum & Mehnaz Ameen\n";
        cout << "=====\\n";

        cout << "1. Signup\\n2. Login\\n3. View Leaderboard\\n4. View Registered Users\\n5. Exit\\nChoice: ";
        cin >> choice;

        switch (choice) { // switch statement
        case 1:
            signup();
            break;

        case 2:
            if (login(loggedUser)) {
                int opt;
                do { // do while Loop
                    cout << "\nLogged in as " << loggedUser << "\\n";
                    cout << "1. Easy Test\\n2. Hard Test\\n3. View Leaderboard\\n4. Logout\\nChoice: ";
                    cin >> opt;

                    if (opt == 1) {
                        EasyTest c;
                        c.startTest(loggedUser);
                    } else if (opt == 2) {
                        HardTest f;
                        f.startTest(loggedUser);
                    } else if (opt == 3) {
                        viewLeaderboard();
                    } else if (opt == 4) {
                        cout << "Logged out.\\n";
                        loggedUser = "";
                        break;
                    } else {
                        cout << "Invalid choice!\\n";
                    }
                } while (opt != 4);
            }
            break;

        case 3:
            viewLeaderboard();
            break;
        case 4:
            viewUsers();
            break;
        case 5:
            cout << "Goodbye!\\n";
            return 0;
        default:
            cout << "Invalid choice!\\n";
        }
    }
}

```

Output

```
=====
      Typing Master
    Developed by Mesum & Mehnaz Ameen
=====
1. Signup
2. Login
3. View Leaderboard
4. View Registered Users
5. Exit
Choice: 1
```

```
Enter new username: Kamran
Enter password: kamran123
User registered successfully!
```

```
Enter username: Kamran
Enter password: kamran123

Login successful! Welcome, Kamran!

Logged in as Kamran
1. Easy Test
2. Hard Test
3. View Leaderboard
4. Logout
Choice:
```

```
Choice: 1

Typing Test (EASY):
The quick brown fox jumps over the lazy dog

Start typing:
The quick brown fox jumps over the lazy dog

===== Result =====
Time: 12.06 sec
Speed: 39.801 WPM
Accuracy: 100%
```

```
Logged in as Kamran
1. Easy Test
2. Hard Test
3. View Leaderboard
4. Logout
Choice: 2

Typing Test (HARD):
Object oriented programming in C++ allows classes, inheritance and polymorphism to create reusable code.

Start typing:
Object oriented programming in C++ allows classes, inheritance and polymorphism to create reusable code.

===== Result =====
Time: 29.528 sec
Speed: 40.6394 WPM
Accuracy: 100%
```

```

Logged in as Kamran
1. Easy Test
2. Hard Test
3. View Leaderboard
4. Logout
Choice: 3

===== Leaderboard =====
=====
          Typing Master Leaderboard
=====
Username      Speed(WPM)    Accuracy(%)    Time(sec)
-----
Mesum        35.24        100.00        13.62
Mesum        59.02        100.00        8.13
Mehnaz       38.40        100.00        12.50
Kamran       39.80        100.00        12.06
Kamran       40.64        100.00        29.53

```

```

Logged in as Kamran
1. Easy Test
2. Hard Test
3. View Leaderboard
4. Logout
Choice: 4
Logged out.

=====
          Typing Master
      Developed by Mesum & Mehnaz Ameen
=====
1. Signup
2. Login
3. View Leaderboard
4. View Registered Users
5. Exit
Choice: 5
Goodbye!

-----
Process exited after 560.2 seconds with return value 0
Press any key to continue . . .

```

File Handling

 leaderboard	11/11/2025 1:15 AM	Text Document	1 KB
 Typing Master C++	11/11/2025 12:48 AM	C++ Source File	9 KB
 Typing Master C++	11/11/2025 1:07 AM	Application	1,895 KB
 users	11/11/2025 1:11 AM	Text Document	1 KB

Typing Master Leaderboard			
Username	Speed(WPM)	Accuracy(%)	Time(sec)
Mesum	35.24	100.00	13.62
Mesum	59.02	100.00	8.13
Mehnaz	38.40	100.00	12.50
Kamran	39.80	100.00	12.06
Kamran	40.64	100.00	29.53

leaderboard.txt

Registered Users	
Username	Password
Mesum	12345
Mehnaz	Ameen
Kamran	kamran123

User.txt

9. OOP Concepts Used

Concept	Usage
Class & Object	Used for User, Test, Easy Test, and Hard Test
Inheritance	Easy Test & Hard Test inherit from Test
Polymorphism	Virtual function start Test() overridden in derived classes
Encapsulation	Private data with getter/setter methods
Abstraction	Test class hides test details

10. Extra Terms: Explanation, Purpose & Usage

Term	Purpose	Usage
fstream	File handling	Manage data in users.txt & leaderboard.txt
clock_t	Time calculation	Measure typing duration
setw()	Formatting	Align columns in tables
setprecision()	Precision	Display 2-decimal results
getline()	Input	Capture full typed text
virtual	Polymorphism	Enable dynamic test behavior

11. Conclusion

The Typing Master in C++ project successfully demonstrates the power of OOP and file handling in developing practical console applications. It allows users to practice typing, analyze performance, and track progress. The modular, reusable, and efficient code design meets all academic and functional objectives for a C++ OOP project.