Project Report

COURSE TITLE:

Object Oriented Programming in C++

PROJECT TITLE:

OOP-Based, Smart Currency Converter

PROJECT REPORT BY:

NAME	Reg No #
1. Muhammad Danail.	24ABSWE0021
2. Syed Zameer Bukhari.	24ABSWE0032

Project Supervised by:

Engr. Yasir Malik

Semester: 2nd

Date: 23/07/2025.

Dept. Of Software Engineering

UET Peshawar, Abbottabad Campus.

Instructor Signature

DECLARATION

We MUHAMMAD DANIAL (24ABSWE0021), SYED ZAMEER BUKHARI (24ABSWE0032), and hereby declare that we have produced the work presented in this report, during the scheduled period of study. We also declare that we have not taken any material from any source except referred the course instructor. If a violation of rules has occurred in this report, we shall be liable to punishable action.

Date: 20 June, 2025

MUHAMMAD DANIAL (21),
SYED ZAMEER BUKHARI (32).

1. Introduction to our project:

- In today's globalized economy, currency conversion plays a pivotal role in facilitating international trade, travel, and finance.
- Whether it's a business making cross-border payments or an individual sending money overseas, accurate and user-friendly currency converters are essential.
- This project aims to develop a Currency Converter system using 100% Object-Oriented Programming (OOP) in C++.
- The project will follow a multi-file structure to ensure clean, modular, and scalable code design.

2. Objectives:

The primary objectives of this Currency Converter project are:

- Implement 100% Object-Oriented Programming in C++ using all major OOP concepts.

- Allow user account creation with dynamic object allocation and constructor overloading.

- Convert major global currencies (USD, EUR, GBP, etc.) to PKR using inheritance and virtual functions.

- Enable the addition of new currencies at runtime using dynamic memory and class instantiation.

- Overload operators (such as + and =) for currency operations and logging.

- Ensure proper memory management by using new for creation and delete for cleanup.

- Build a user-friendly menu interface to handle all functionalities and use graphics to display a bar chart showing the most converted currencies.

3. Classes Structure Analysis:

Below is a high-level overview of the class structure that will be implemented in our project:

3.1. User Class :

Purpose: To create and manage user accounts.

Attributes:

- Constructor overloading (with/without email, phone).
- Dynamic Creation with (new) and display user info.

3.2. Currency (Abstract Base Class):

Purpose: Acts as a parent class for all specific currencies.

Attributes:

- Currency Name.
- rate To PKR.

3.3. Derived Currency Classes (Dollar, Euro, Pound, etc.):

Inherit From: Currency Class.

Purpose: Implement specific conversion logic.

Attributes:

- Override convert To PKR.
- Include overloaded operators for convenience.

3.4. Currency Manager Class :

Purpose: Manage list of available currencies.

Attributes:

- Add new currency using new.
- Search and convert selected currency.
- Delete all allocated memory.

3.5. Menu Class :

Purpose: Display menu, handle input/output.

Attributes:

- Navigate between features.
- Call functions based on user choice.

3.6. Graph Manager Class :

Purpose: Display a bar chart using graphics.

Attributes:

- Analyze logs.
- Draw graph showing most converted currencies.

4. Practical Significance of our Project :

The practical significance of this project lies in its real-world relevance, extensibility and technical depth as listed below:

4.1. Real-Time Currency Understanding:

Users can easily understand how much their money is worth in Pakistan Rupees using live-like exchange mechanisms.

4.2. Customizable Currency System:

By allowing new currencies to be dynamically added, it reflects the flexibility needed in financial systems.

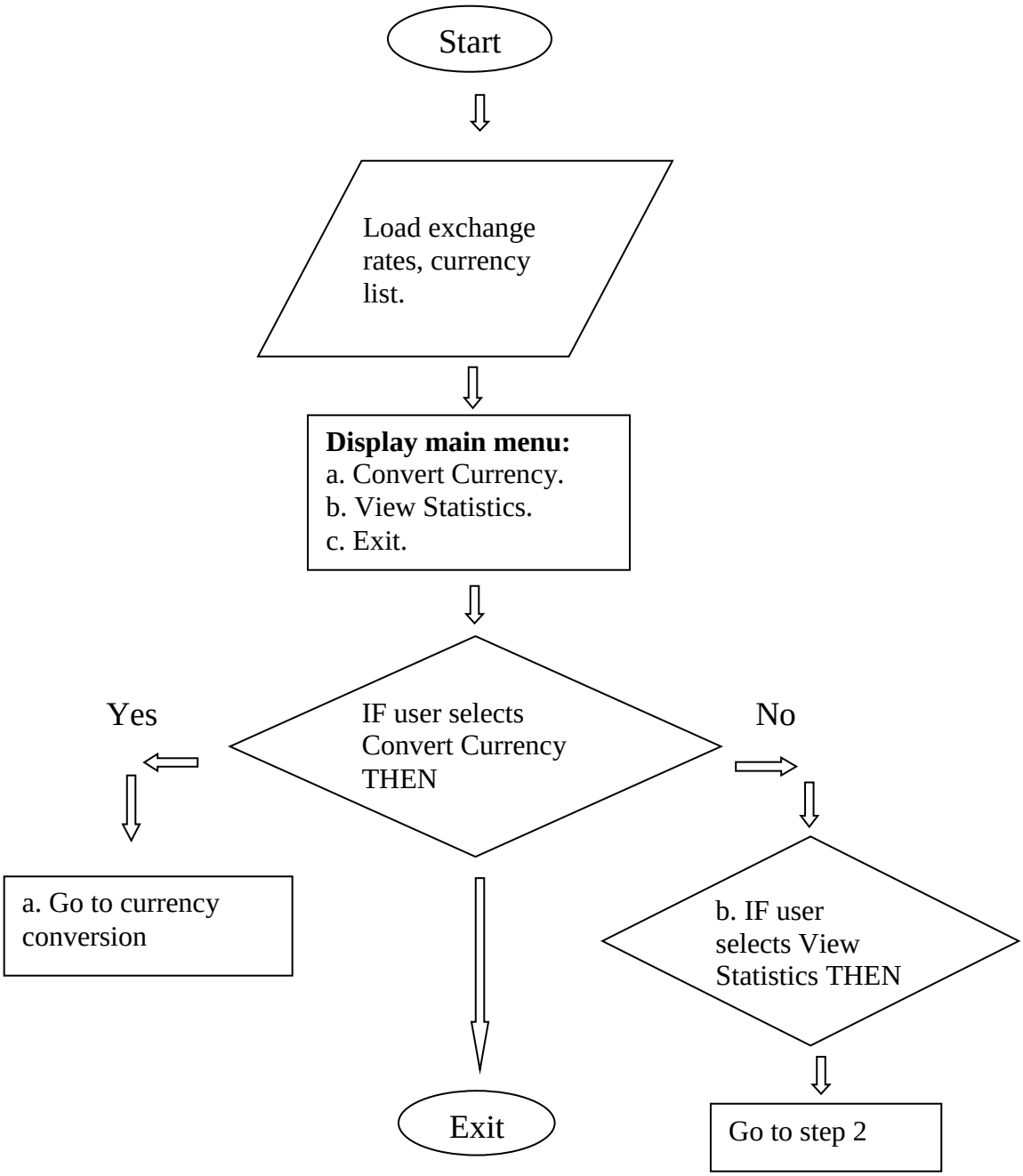
4.3. Reusability & Modularity:

The use of Object-Oriented Programming (OOP) principles allows clean modular code that is scalable and easy to maintain.

4.4. Memory-Efficient System:

Dynamic object creation and deletion ensures that only required resources used during runtime.

5. Flowchart for our Project:



6. Algorithm for our Project:

Step 1: Start the project.

Step 2: Load default exchange rates and currency list.

Step 3: Display main menu:

a. Convert Currency.

b. View Statistics.

c. Exit.

Step 4: Take user input for menu choice.

Step 5: IF user selects Convert Currency THEN

Step 5.1: Input source currency, target currency, and amount.

Step 5.2: Dynamically allocate source and target Currency objects using New.

Step 5.3: Use virtual function to convert the amount.

Step 5.4: Display converted result.

Step 5.5: Use file streams (ofstream in C++) within writeLog() to **append** each conversion to a log file (e.g., conversions.txt).

Step 5.6: Free memory using delete.

Step 5.7: Ask if user wants to convert again.

IF Yes → Go to Step 5

ELSE → Go to Step 3

Step 6: ELSE IF user selects View Statistics THEN

Step 6.1: Read data from conversion log file.

Step 6.2: Calculate most frequently converted currencies.

Step 6.3: Display result graphically (text/graphics).

Step 6.4: Ask if user wants to return to menu.

IF Yes → Go to Step 3

ELSE → Go to Step 6

Step 7: ELSE IF user selects Exit THEN

Step 7.1: Terminate the program.

Step 8: End.

7. Smart Currency Converter using 3 multi-file Structure:

7.1. Currency Exchange Rates File Code:

USD,US Dollar,278.45
EUR,Euro,300.20
GBP,British Pound,350.15
CAD,Canadian Dollar,205.80
AUD,Australian Dollar,190.65
SAR,Saudi Riyal,74.45
AED,UAE Dirham,76.05
JPY,Japanese Yen,1.95
CNY,Chinese Yuan,39.60
INR,Indian Rupee,3.40
PKR,Pakistani Rupee,1.00
NGN,Nigerian Naira,0.18
NZD,New Zealand Dollar,173.30
KWD,Kuwaiti Dinar,907.75
IRR,Iranian Rial,0.0066
RUB,Russian Ruble,3.10
MYR,Malaysian Ringgit,58.20
IDR,Indonesian Rupiah,0.017
TRY,Turkish Lira,8.15
EGP,Egyptian Pound,6.50
QAR,Qatari Riyal,76.80

7.2. Currencies Conversions Header File Code:

```
#ifndef CURRENCYCONVERSIONS_H
#define CURRENCYCONVERSIONS_H
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
using namespace std;

class Currency {
protected:
    string name;
    double rateToPKR;
public:
    Currency(string n, double rate) : name(n), rateToPKR(rate) {}
    // Operator Overloading
    double operator*(double amount) {
        return amount * rateToPKR;
    }

    double operator/(double pkrAmount) {
        return pkrAmount / rateToPKR;
    }

    double convertToPKR(double amount) {
        return amount * rateToPKR;
    }

    double convertFromPKR(double pkrAmount) {
        return pkrAmount / rateToPKR;
    }
    string getName() {
        return name;
    }
```

```
    double getRate() {
        return rateToPKR;
    }
};

class CurrencyManager {
    Currency** currencies;
    string* codes;
    int count;
public:
    CurrencyManager() {
        currencies = new Currency*[100];
        codes = new string[100];
        count = 0;
    }
    // Cleanup method (used instead of destructor)
    void cleanup() {
        for (int i = 0; i < count; i++) {
            delete currencies[i];
        }
        delete[] currencies;
        delete[] codes;
    }

    void loadFromFile(const string& filename) {
        count = 0;
        ifstream fin(filename.c_str());
        string line;
        while (getline(fin, line)) {
            size_t pos1 = line.find(',');
            size_t pos2 = line.rfind(',');
            if (pos1 != string::npos && pos2 != string::npos && pos1 != pos2) {
                string code = line.substr(0, pos1);
                string name = line.substr(pos1 + 1, pos2 - pos1 - 1);
                string rateStr = line.substr(pos2 + 1);
```

```
            stringstream ss(rateStr);
            double rate;
            if (ss >> rate)
{
                for (int i = 0; i < code.length(); i++) code[i] = toupper(code[i]);
                codes[count] = code;
                currencies[count] = new Currency(name, rate);
                count++;
            } else {
                cerr << "Invalid rate in file: " << line << endl;
            }
        }
    }
    fin.close();
  }
  Currency* getCurrency(string& code) {
    string upperCode = code;
    for (int i = 0; i < upperCode.length(); i++) upperCode[i] =
toupper(upperCode[i]);
    for (int i = 0; i < count; i++) {
      if (codes[i] == upperCode) return currencies[i];
    }
    return 0;
  }
  int getCount() { return count; }
  Currency* getCurrencyByIndex(int i) { return currencies[i]; }
  string getCodeByIndex(int i) { return codes[i]; }
};
#endif
```

7.3. Main.cpp File Code:

```
#include <iostream>
#include <fstream>
#include <string>
```

```
#include <iomanip>
#include <graphics.h>
#include "currencyconversions.h"
using namespace std;
class Person                    // Base Class
{protected:
   string name, email, phone;
public:
   Person(string n, string e, string p) : name(n), email(e), phone(p) {}
   virtual void logToFile() = 0;
};
// -------------------- User Class --------------------
class User : public Person {
   int userID;
public:
   User(string n, string e, string p) : Person(n, e, p), userID(1) {
      ifstream fin("user_logs.txt");
      string line;
      while (getline(fin, line)) {
         if (line.find("User ID:") != string::npos) {
            userID++; }
      }
      fin.close(); }
   void logToFile() override {
      ofstream fout("user_logs.txt", ios::app);
      fout << "\nUser ID: " << userID
           << ", Name: " << name
           << ", Email: " << email
           << ", Phone: " << phone << endl;
      fout.close();
   }
 public:
   int getID() const { return userID; }
   string getName() const { return name; }
   string getEmail() const { return email; }
   string getPhone() const { return phone; }
};
```

```
// Now define operator<< outside the class
ostream& operator<<(ostream& out, const User& u) {
    out << "User ID: " << u.getID()
        << ", Name: " << u.getName()
        << ", Email: " << u.getEmail()
        << ", Phone: " << u.getPhone();
    return out;
}// ---------- Menu Base Class ----------
class BaseMenu {
protected:
    string options[3] = {"Convert Currency", "View Board", "Exit"};
public:
    virtual void showMenu(int) = 0;
};
// ---------- Inherited Menu Class ----------
class Menu : public BaseMenu {
public:
    void showMenu(int highlight) override {
        system("cls");
        cout << "\nUse UP/DOWN arrow keys to move, ENTER to select.\n\n";
        for (int i = 0; i < 3; i++) {
            if (i == highlight) {
                cout << "\033[32m";  // ANSI code for green text
            }
            cout << (char)(i + 'a') << ". " << options[i] << endl;
            if (i == highlight) {
                cout << "\033[0m";  // Reset color
            }
        }
    }
};
// ---------- Logger Class ----------
class Logger {
public:
    void logConversion(int userID, string from, string to, double amount, double
result)
{
```

```
        ofstream log("user_logs.txt", ios::app);
        log << "Conversion (User " << userID << "): " << from << " -> PKR -> "
        << " | Amount: " << amount << " => " << result << endl;
        log.close();
        ofstream usage("graph_log.txt", ios::app);
        usage << to << endl;
        usage.close();
    }
};
// -------------------- GraphViewer Class --------------------
class GraphViewer {
public:
    void show() {
        ifstream fin("graph_log.txt");
        if (!fin.is_open()) return;
        string codes[100];
        int counts[100];
        int size = 0;
        string line;
        while (getline(fin, line)) {
            bool found = false;
            for (int i = 0; i < size; i++) {
                if (codes[i] == line) {
                    counts[i]++;
                    found = true;
                    break;
                }
            }
            if (!found) {
                codes[size] = line;
                counts[size] = 1;
                size++;
            }
        }
        fin.close();
        initwindow(1000, 600, "Currency Usage Graph");
        setbkcolor(WHITE);
        cleardevice();
```

```
        int x = 100, y = 500, width = 30, spacing = 20, maxHeight = 300;
        int maxCount = 1;
        for (int i = 0; i < size; i++) {
            if (counts[i] > maxCount) maxCount = counts[i];
        }

        for (int i = 0; i < size; i++) {
            int height = (counts[i] * maxHeight) / maxCount;
            setfillstyle(SOLID_FILL, (i + 2) % 15 + 1);
            bar(x, y - height, x + width, y);
            setcolor(BLACK);
            outtextxy(x, y + 10, (char*)codes[i].c_str());
            x += width + spacing;
        }
        delay(6000);
        closegraph();
    }
};
// -------------------- Welcome Screen --------------------
class WelcomeScreen {
public:
    void show() {
        initwindow(600, 300, "Welcome");
        setbkcolor(BLUE);
        cleardevice();
        setcolor(YELLOW);
        settextstyle(BOLD_FONT, HORIZ_DIR, 3);
        outtextxy(50, 100, (char*)"Welcome to Currency Converter!");
        delay(2500);
        closegraph();
    }
};
// -------------------- Currency Listing --------------------
void listCurrencies(CurrencyManager& manager) {
    cout << "\nAvailable Currencies:\n";
    for (int i = 0; i < manager.getCount(); i++) {
        cout << manager.getCodeByIndex(i) << " - " <<
manager.getCurrencyByIndex(i)->getName() << endl;
```

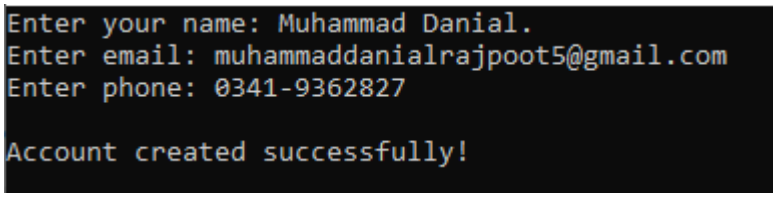
```
    }
}
// -------------------- Board Display --------------------
void showBoard(CurrencyManager& manager) {
    cout << left << setw(25) << "Currency Name" << setw(10) << "Code" <<
setw(15) << "Rate to PKR" << endl;


    cout << "-------------------------------------------------------------\n";
    for (int i = 0; i < manager.getCount(); i++) {
        cout << left << setw(25) << manager.getCurrencyByIndex(i)->getName()
             << setw(10) << manager.getCodeByIndex(i)
             << setw(15) << fixed << setprecision(2)
             << manager.getCurrencyByIndex(i)->convertToPKR(1.0) << endl;
    }
}
// -------------------- Main Function --------------------
int main() {
    WelcomeScreen welcome;
    welcome.show();
    string name, email, phone;
    cout << "Enter your name: "; getline(cin, name);
    cout << "Enter email: "; getline(cin, email);
    cout << "Enter phone: "; getline(cin, phone);

    Person* person = new User(name, email, phone);
    person->logToFile();
    cout << "\nAccount created successfully!\n";
    cout << "Use UP/DOWN arrow keys to move, ENTER to select.\n\n";
    CurrencyManager* manager = new CurrencyManager;
    Logger logger;
    GraphViewer graph;
    BaseMenu* menu = new Menu;

    int highlight = 0;
    while (true) {
        menu->showMenu(highlight);
        int key = _getch();
```

```
        if (key == 224) {
            key = _getch();
            if (key == 72) highlight = (highlight + 2) % 3; // UP
            if (key == 80) highlight = (highlight + 1) % 3; // DOWN
        } else if (key == 13) { // ENTER
            ofstream log("user_logs.txt", ios::app);
            log << "User selected option " << (char)(highlight + 'a') << endl;
            log.close();

            if (highlight == 0)
            {
                manager->loadFromFile("currencyexchange_rates.txt");
                listCurrencies(*manager);
                cout << "\nSelect currency you want to convert to PKR or any other
currency:\n";
                string fromCode, toCode;
                double amount;
                cout << "\nFrom: "; cin >> fromCode;
                cout << "To: "; cin >> toCode;
                cout << "Amount: "; cin >> amount;

                Currency* from = manager->getCurrency(fromCode);
                Currency* to = manager->getCurrency(toCode);

                if (from && to)
                {
                double inPKR = (*from) * amount;      // Overloaded * operator
               double result = (*to) / inPKR;
    // Overloaded / operator
                cout << fixed << setprecision(2)
                << amount << " " << fromCode
                << " = " << result << " " << toCode << "\n" << endl;
                logger.logConversion(((User*)person)->getID(), fromCode, toCode,
amount, result);
                }
              else
                {
```

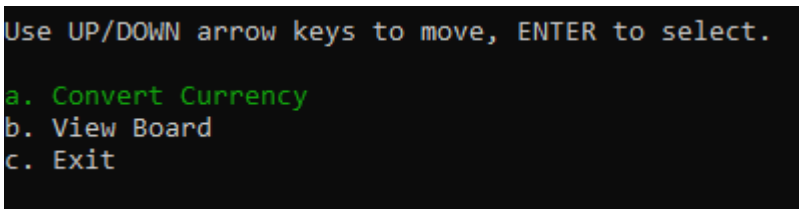
```
                cout << "Invalid currency code!\n";
            }
        } else if (highlight == 1) {
            manager->loadFromFile("currencyexchange_rates.txt");
            showBoard(*manager);
        } else if (highlight == 2) {
            cout << "\nThanks for using our converter!\n";
            graph.show();
            break;
        }
        system("pause");
    }
  }
  delete person;
  manager->cleanup();
  delete manager;
  delete menu;
  return 0;
}
```

8. Output for User-Manual of our Project:

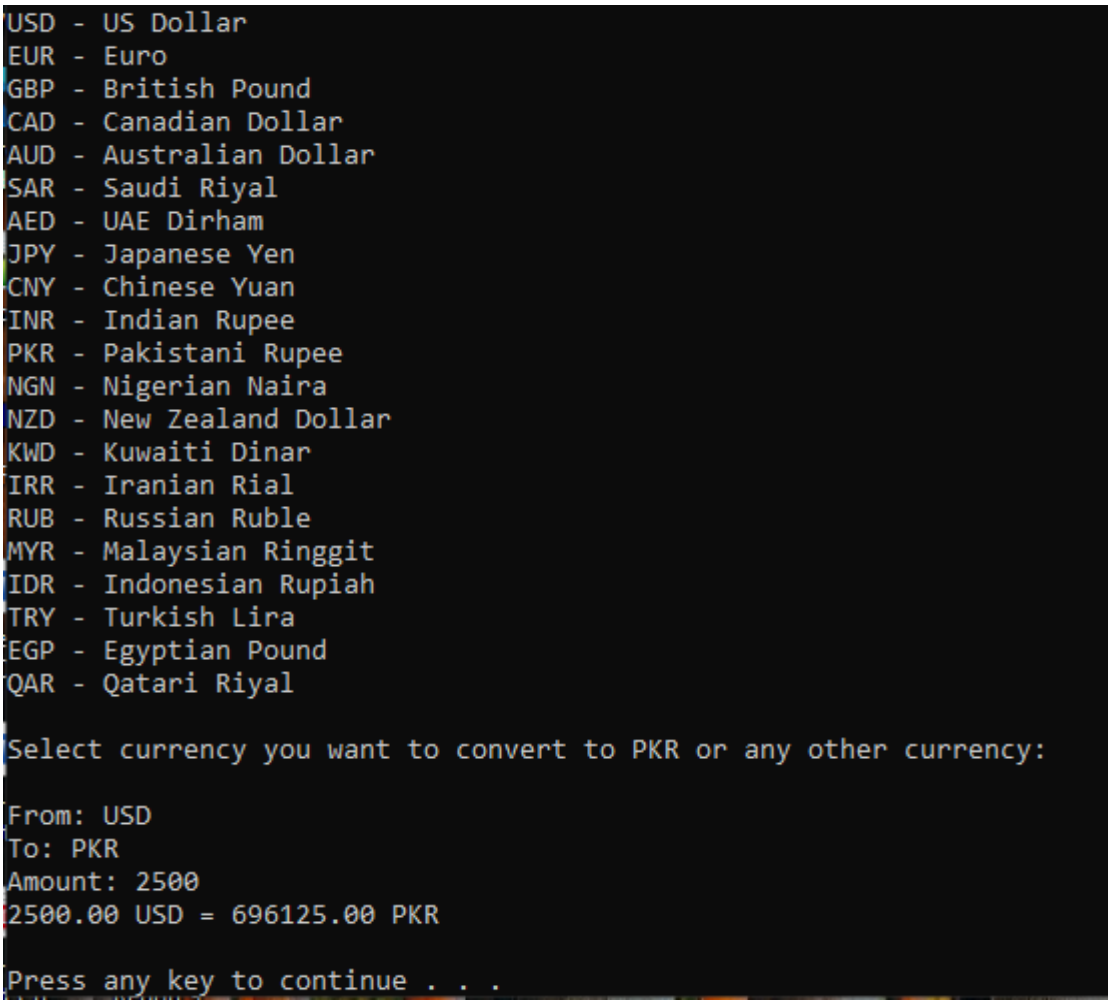
The following output demonstrates that the user record has been saved in the "user.log.txt" file.



9. Output for drop-down menu of our Project:



10. Output when User selects first option:



11. Output when User selects second option:

```
Use UP/DOWN arrow keys to move, ENTER to select.

a. Convert Currency
b. View Board
c. Exit
Currency Name              Code       Rate to PKR
-------------------------------------------------------------
US Dollar                  USD        278.45
Euro                       EUR        300.20
British Pound              GBP        350.15
Canadian Dollar            CAD        205.80
Australian Dollar          AUD        190.65
Saudi Riyal                SAR        74.45
UAE Dirham                 AED        76.05
Japanese Yen               JPY        1.95
Chinese Yuan               CNY        39.60
Indian Rupee               INR        3.40
Pakistani Rupee            PKR        1.00
Nigerian Naira             NGN        0.18
New Zealand Dollar         NZD        173.30
Kuwaiti Dinar              KWD        907.75
Iranian Rial               IRR        0.01
Russian Ruble              RUB        3.10
Malaysian Ringgit          MYR        58.20
Indonesian Rupiah          IDR        0.02
Turkish Lira               TRY        8.15
Egyptian Pound             EGP        6.50
Qatari Riyal               QAR        76.80
Press any key to continue . . .
```

12. Output when User selects last option:

13. Conclusion:

This project showcases a robust and interactive Smart Currency Converter built using OOP concepts in C++. It effectively handles real time conversions, user logging, and graphical insights through initializing graphics in C++ graphics and file-based data management. The design emphasizes clarity, usability, and modular coding practices idea.

14. References:

The followings are the references used by me:

- Coding Plateform: Codeblocks (“codeblocks-20.03mingw-setup.exe”).
- Borland Graphics Interface (BGI) via graphics.h and windows.h files. etc.

.