Title: Ration Management System

Submitted by:

Imtiaz(240026),Sharjeel(240016),

Shabbir(240028),Naila (240026)

Course Details:

OOP, 2nd Semester, Air university

Submitted to: M Bilal Khan

2. Declaration

hereby declare that this project titled **"Ration Management System"** is our original work. we affirm that it has not been submitted to any other institution or organization for academic or professional purposes. All sources of information have been appropriately acknowledged.

3. Certificate

This is to certify that the project titled **"Ration Management System"** has been completed to the satisfaction of the requirements set forth by Air University. It has been thoroughly reviewed and approved by M Bilal Khan as part of the academic coursework for OOP

4. Acknowledgments

We would like to express our heartfelt gratitude to **M Bilal Khan** for their invaluable guidance and support throughout this project. We also extend our thanks to our friends for their encouragement and assistance. Lastly, we thank everyone who contributed to the successful completion of this project.

5. Synopsis

Objective:
The aim of this project is to create a system that simplifies and improves the management of ration distribution. It includes features for managing recipients' information, checking their eligibility, organizing ration items, and generating ration tokens.

Key Features:

- Add and store recipient details like CNIC, name, address, and eligibility status.
- Manage ration items such as sugar, flour, and oil.
- Generate tokens for eligible recipients to claim their ration.

Tools Used:

- **Programming Language:** C++
- **Development Environment:** Visual Studio Code / GCC Compiler
- **Libraries:** Standard Template Library (STL)

6. Project Structure

The project is divided into three main modules:

1. Recipient Management Module:

- Store recipient information such as CNIC, name, and address.
- Check eligibility using predefined district codes.

2. Ration Management Module:

- Add ration items (e.g., sugar, flour, oil, milk) and their quantities.
- Update inventory as items are distributed.

3. Token Generation Module:

- Generate tokens for eligible recipients.
- Allocate ration items based on availability.

Key Functionalities:

- Validate recipients using CNIC.
- Dynamically allocate ration items based on inventory.
- Generate date-stamped tokens.

7. Requirements

1.2 Hardware Requirements:

- **Operating System:** Windows/Linux/MacOS
- **RAM:** 4 GB or more
- **Processor:** Intel i3 or equivalent

Software Requirements:

- C++ Compiler (GCC/MSVC)
- Integrated Development Environment (e.g., Visual Studio Code)

8. Code Highlights

The project's code includes detailed comments to explain:

- The purpose of each function.
- How recipient data is processed.
- Inventory management logic.
- Token generation mechanics.

This ensures that the code is easy to understand and maintain.

9. Conclusion

The **Ration Management System** effectively automates the process of distributing ration to eligible recipients. It ensures accurate inventory tracking and provides easy access to ration tokens. In the future, this system could be enhanced by:

- Adding a database for better data management.
- Creating a mobile or web-based interface for remote access.
- Developing advanced reporting tools to monitor distribution trends.

Software Requirements:

- C++ Compiler (GCC/MSVC)
- Integrated Development Environment (e.g., Visual Studio Code)

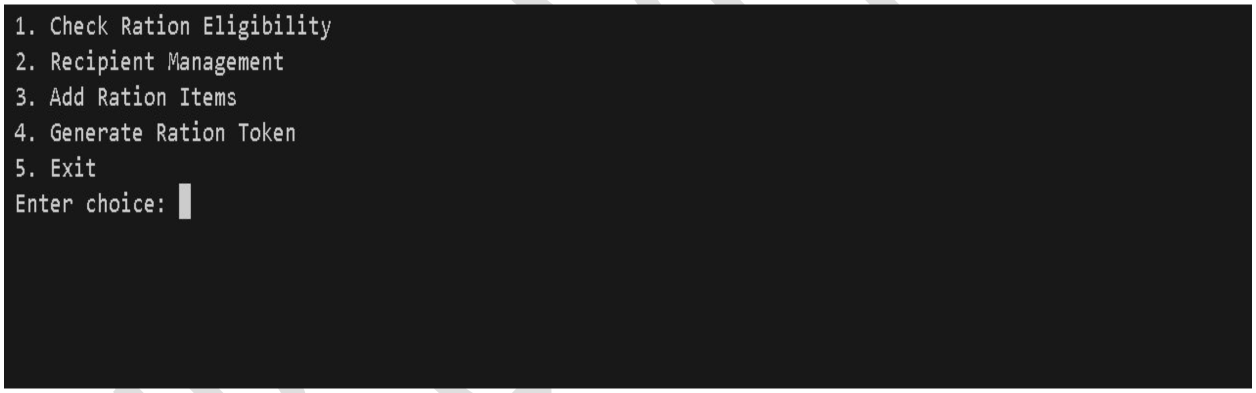
```cpp
#include <iostream>
#include <unordered_map>
#include <string>
#include <ctime>
#include <iomanip>
#include <sstream>

using namespace std;

// Class to store recipient information
class Recipient {
public:
    string cnic;               // CNIC of the recipient
    string name;               // Name of the recipient
    string address;            // Address of the recipient
    bool isEligible;           // Eligibility status
    string rationTokenDate;    // Date of ration token generation
    unordered_map<string, int> rationReceived; // Ration items received

    // Default constructor
    Recipient() {
        cnic = ""; // Initialize CNIC as an empty string
        name = ""; // Initialize name as an empty string
        address = ""; // Initialize address as an empty string
        isEligible = false; // Set eligibility to false by default
        rationTokenDate = ""; // Initialize ration token date as an empty string
    }

    // Parameterized constructor
    Recipient(string c, string n, string a, bool e) {
        cnic = c; // Assign CNIC
        name = n; // Assign name
        address = a; // Assign address
        isEligible = e; // Assign eligibility status
        rationTokenDate = ""; // Initialize ration token date as an empty string
    }
};

// Class to manage the ration system
class RationManagementSystem {
private:
    unordered_map<string, Recipient> recipients; // Store all recipients
    unordered_map<string, int> rationItems;      // Store all ration items and
their quantities

    // Function to check eligibility based on CNIC
    bool checkEligibility(string cnic) {
        unordered_map<string, string> eligibleDistricts = {
            // Map of eligible district codes and their names
            {"41401", "Badin"}, {"42301", "Jacobabad"}, {"41203", "Khairpur"},
            {"41201", "Sukkur"}, {"41304", "Thatta"}, {"53401", "Gwadar"},
            {"53102", "Lasbela"}, {"53204", "Jaffarabad"}, {"53301", "Killa Saifullah"},
            {"53105", "Mastung"}, {"71501", "Gilgit"}, {"71602", "Diamer"},
            {"71703", "Nagar"}, {"71504", "Ghizer"}, {"71605", "Ghanche"},
            {"17101", "Charsadda"}, {"17202", "Swat"}, {"17103", "Dera Ismail Khan"},
            {"17204", "Mardan"}, {"17105", "Nowshera"}, {"32101", "Dera Ghazi Khan"},
            {"32202", "Rajanpur"}, {"32303", "Muzaffargarh"}, {"32404", "Layyah"},
            {"32505", "Rahim Yar Khan"}
        };


        // Extract district code from CNIC
        string districtCode = cnic.substr(0, 5);
        // Check if the district code is in the map
        return eligibleDistricts.find(districtCode) != eligibleDistricts.end();
    }


    // Function to get the current date as a string
    string getCurrentDate() {
        time_t now = time(0); // Get current time
        tm *ltm = localtime(&now); // Convert to local time structure
        stringstream dateStream;
        // Format the date as YYYY-MM-DD
        dateStream << 1900 + ltm->tm_year << "-" << setw(2) << setfill('0') << 1 + ltm->tm_mon << "-" << setw(2) << setfill('0') << ltm->tm_mday;
        return dateStream.str();
    }
```

```
public:
    // Constructor to initialize ration items
    RationManagementSystem() {
        rationItems["sugar/tea"] = 0;
        rationItems["flour"] = 0;
        rationItems["oil"] = 0;
        rationItems["milk"] = 0;
    }

    // Function to add a new recipient
    void addRecipient() {
        string cnic, name, address;
        cout << "Enter CNIC: ";
        cin >> cnic; // Input CNIC
        cout << "Enter Name: ";
        cin.ignore(); // Ignore leftover newline character
        getline(cin, name); // Input name
        cout << "Enter Address: ";
        getline(cin, address); // Input address

        // Check eligibility and add recipient
        bool isEligible = checkEligibility(cnic);
        recipients[cnic] = Recipient(cnic, name, address, isEligible);
        cout << "Recipient " << name << " added successfully." << endl;
    }
```

```cpp
    // Function to view all recipients
    void viewRecipients() {
        for (auto &pair : recipients) {
            Recipient &recipient = pair.second; // Get recipient object
            cout << "CNIC: " << recipient.cnic << ", Name: " << recipient.name << ", Address: " << recipient.address << ", Eligibility: " << (recipient.isEligible ? "Eligible" : "Not Eligible") << ", Ration Token Date: " << recipient.rationTokenDate << endl;


            if (!recipient.rationReceived.empty()) {
                cout << "Ration Items Received:" << endl;
                for (auto &item : recipient.rationReceived) {
                    cout << "  " << item.first << ": " << item.second << endl;
                }
            }
        }
    }


    // Function to add ration items to inventory
    void addRationItem() {
        string item;
        int quantity;
        cout << "Enter Ration Item: ";
        cin >> item; // Input ration item
        cout << "Enter Quantity: ";
        cin >> quantity; // Input quantity
        // Check if the item exists in inventory
        if (rationItems.find(item) != rationItems.end()) {
            rationItems[item] += quantity; // Update quantity
            cout << quantity << " of " << item << " added successfully." << endl;
        } else {
            cout << "Invalid ration item." << endl;
        }
    }


    // Function to generate a ration token for a recipient
    void generateRationToken() {
        string cnic;
        cout << "Enter CNIC of the recipient: ";
        cin >> cnic; // Input CNIC


        // Check if the recipient exists
        if (recipients.find(cnic) != recipients.end()) {
            string date = getCurrentDate(); // Get current date
            recipients[cnic].rationTokenDate = date; // Set token date


            unordered_map<string, int> rationGiven;
            for (auto &item : rationItems) {
                if (item.second > 0) {
                    int quantity = 1; // Each recipient gets 1 unit of each
available item
                    rationGiven[item.first] = quantity;
                    item.second -= quantity; // Reduce inventory
                }
            }


            recipients[cnic].rationReceived = rationGiven; // Record ration
received


            cout << "Ration token generated for " << recipients[cnic].name << "
on " << date << "." << endl;
            cout << "Ration Items:" << endl;
            for (auto &item : rationGiven) {
                cout << "  " << item.first << ": " << item.second << endl;
            }
        } else {
            cout << "Recipient not found." << endl;
        }
    }


    // Function to check eligibility (public interface)
    bool isEligible(string cnic) {
        return checkEligibility(cnic);
    }
};


int main() {
    RationManagementSystem rms; // Create system object
    int choice;
    while (true) {
        // Display menu options
        cout << "\n1. Check Ration Eligibility" << endl;
        cout << "2. Recipient Management" << endl;
        cout << "3. Add Ration Items" << endl;
        cout << "4. Generate Ration Token" << endl;
        cout << "5. Exit" << endl;
        cout << "Enter choice: ";
        cin >> choice; // Input choice


        switch (choice) {
        case 1: {
            string cnic;
            cout << "Enter CNIC: ";
            cin >> cnic; // Input CNIC
            bool eligibility = rms.isEligible(cnic); // Check eligibility
            cout << "Eligibility: " << (eligibility ? "Eligible" : "Not
Eligible") << endl;
            break;
        }
        case 2:
            rms.addRecipient(); // Add recipient
            break;
        case 3:
            rms.addRationItem(); // Add ration item
            break;
        case 4:
```

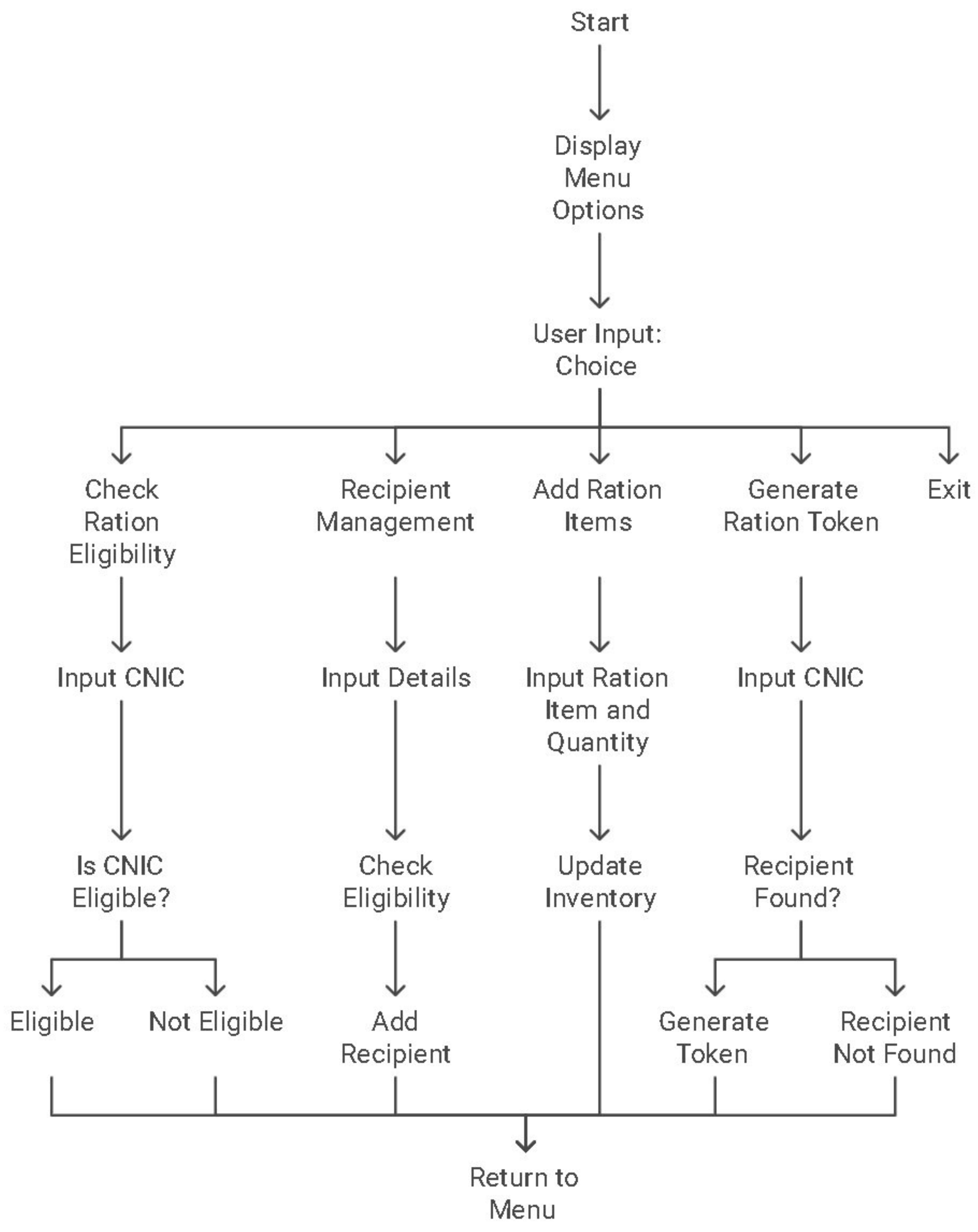
```cpp
            rms.generateRationToken(); // Generate token

            break;

        case 5:

            return 0; // Exit program

        default:

            cout << "Invalid choice. Please try again." << endl;

        }

    }

}
```

Screen Shorts



```
1. Check Ration Eligibility
2. Recipient Management
3. Add Ration Items
4. Generate Ration Token
5. Exit
Enter choice: 
```

Flow chart


Start
Display Menu Options
User Input: Choice
Check Ration Eligibility
Recipient Management
Add Ration Items
Generate Ration Token
Exit
Input CNIC
Input Details
Input Ration Item and Quantity
Input CNIC
Is CNIC Eligible?
Check Eligibility
Update Inventory
Recipient Found?
Eligible
Not Eligible
Add Recipient
Generate Token
Recipient Not Found
Return to Menu