SECJ1023 PROGRAMMING TECHNIQUE II

SEMESTER 2, 2023/2024

GROUP PROJECT DELIVERABLE 4

PROJECT FINALE

MOVIE RECOMMENDATION SYSTEM

Lecturer : Dr Lizawati Binti Mi Yusuf

Section : 4

Group : 5

Presented by : Adriana Zulaikha Binti Zulkarman (A23CS0035)

Leo Min Xue (A23CS0237)

Melody Lui Ruo Ning (A23CS0244)

TOPICS

1. Project Description
2. Project Design
3. OO Concepts Employed
4. System Demonstration

1.0 PROJECT DESCRIPTION

The Movie Recommendation System is an advanced information filtering solution designed to enhance user experience by providing personalized movie recommendations. This system which is based on the ideas of a tailored user experience by making use of sophisticated algorithms to examine user interactions, behavior and preferences. At the same time, this system is able to keep track of variables like movie ratings, viewing history and preferred genres.

OBJECTIVES

-Improve user satisfaction by offering personalized movie suggestions.

-Broaden users' viewing experiences by suggesting diverse content.

-Maintain strong focus on user feedback to continuously enhance the system.

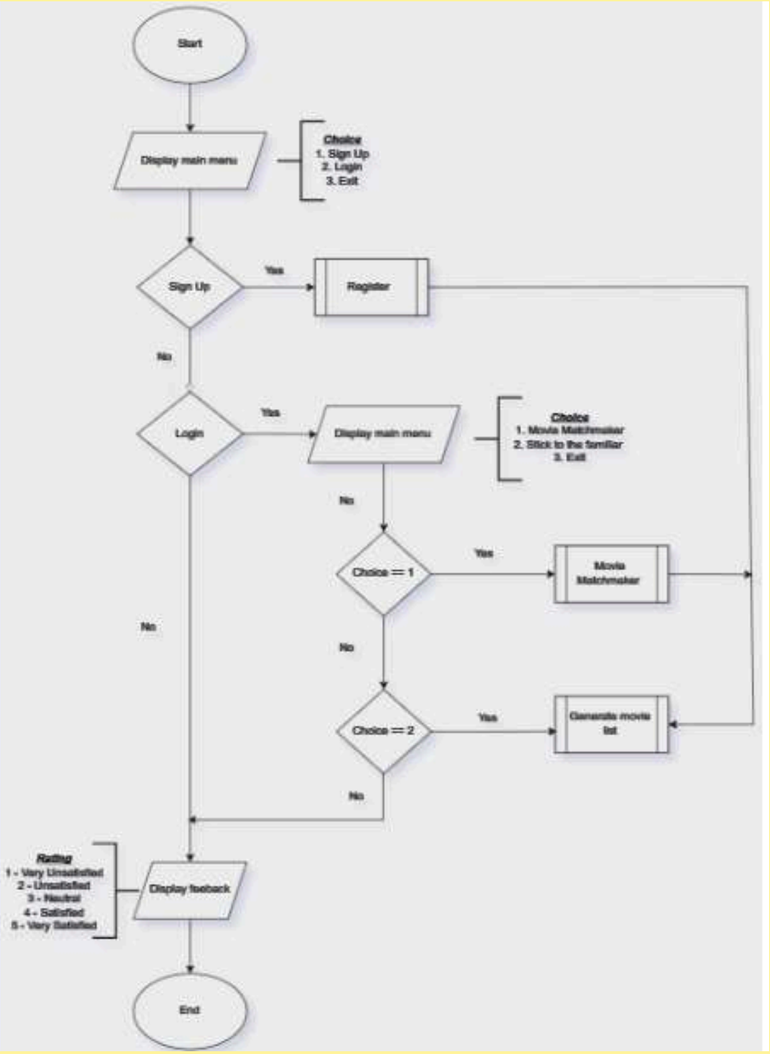
2.0 PROJECT DESIGN

1. Flow Chart

2. UML Diagram

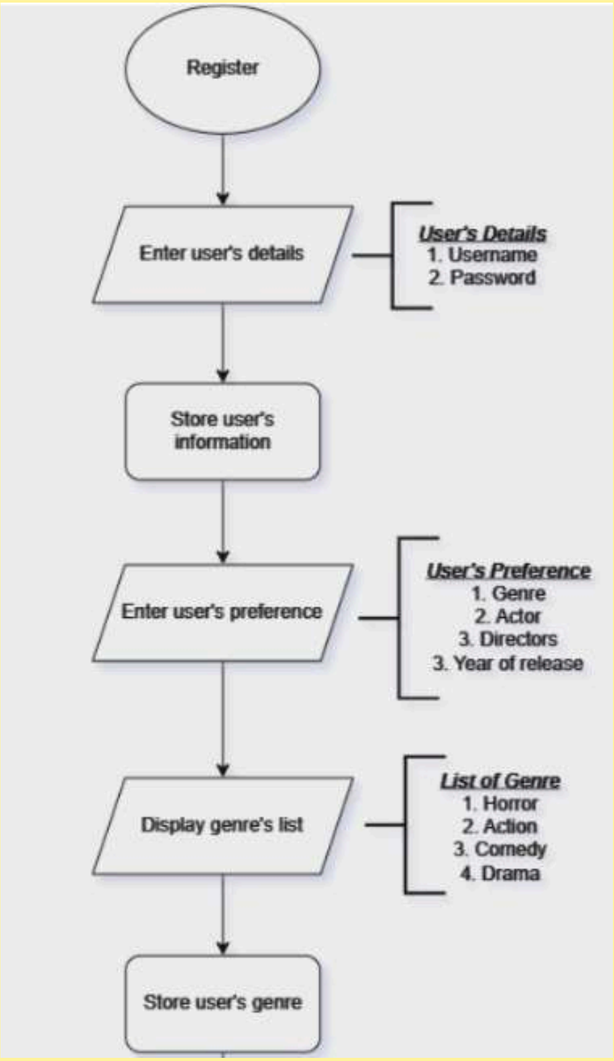
2.0 PROJECT DESIGN

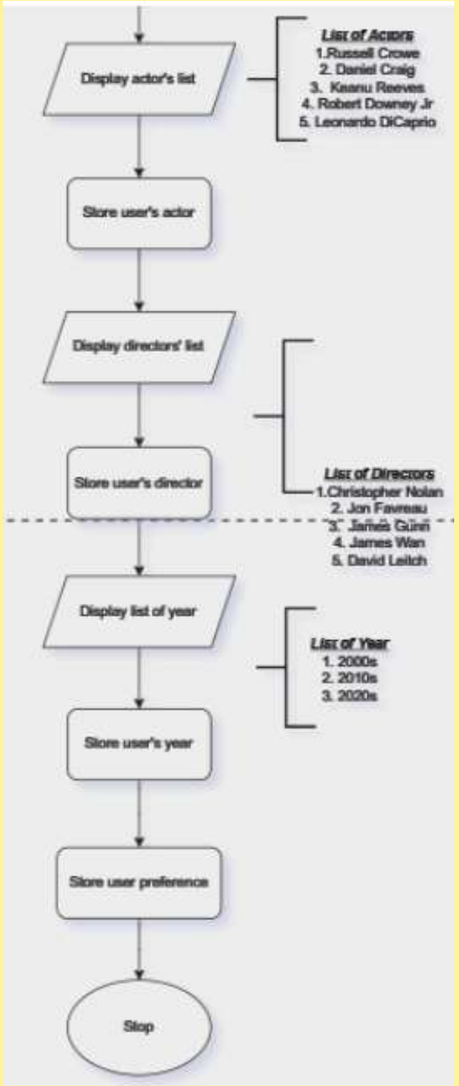
• FLOW CHART (MAIN PROCESS)



2.0 PROJECT DESIGN

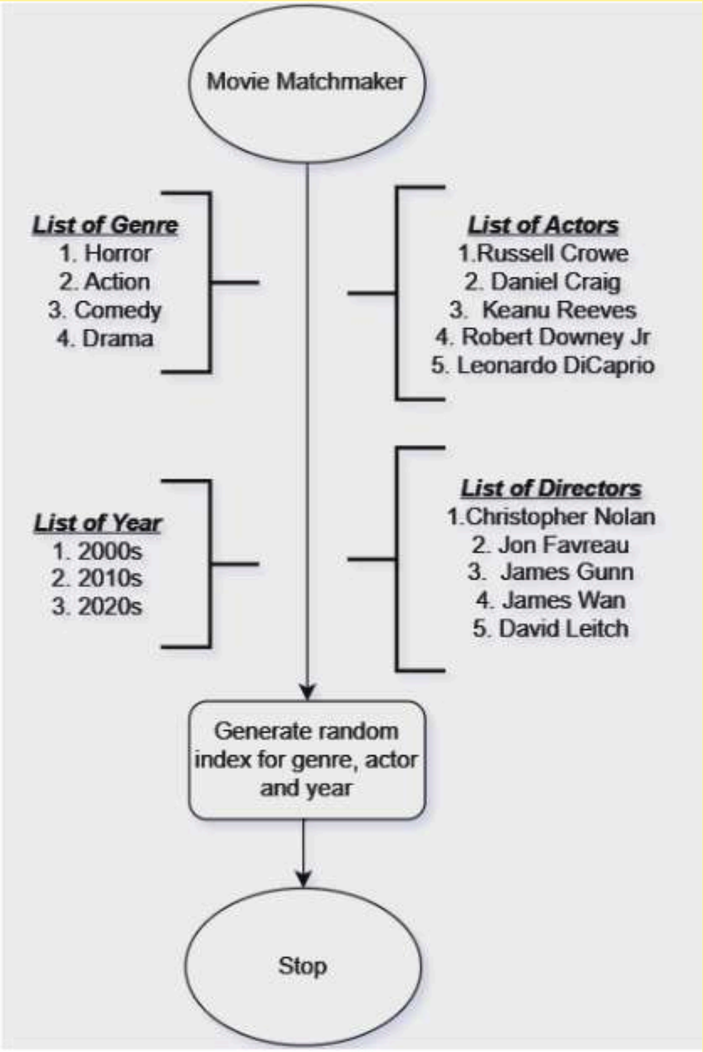
• FLOW CHART (REGISTRATION PROCESS)





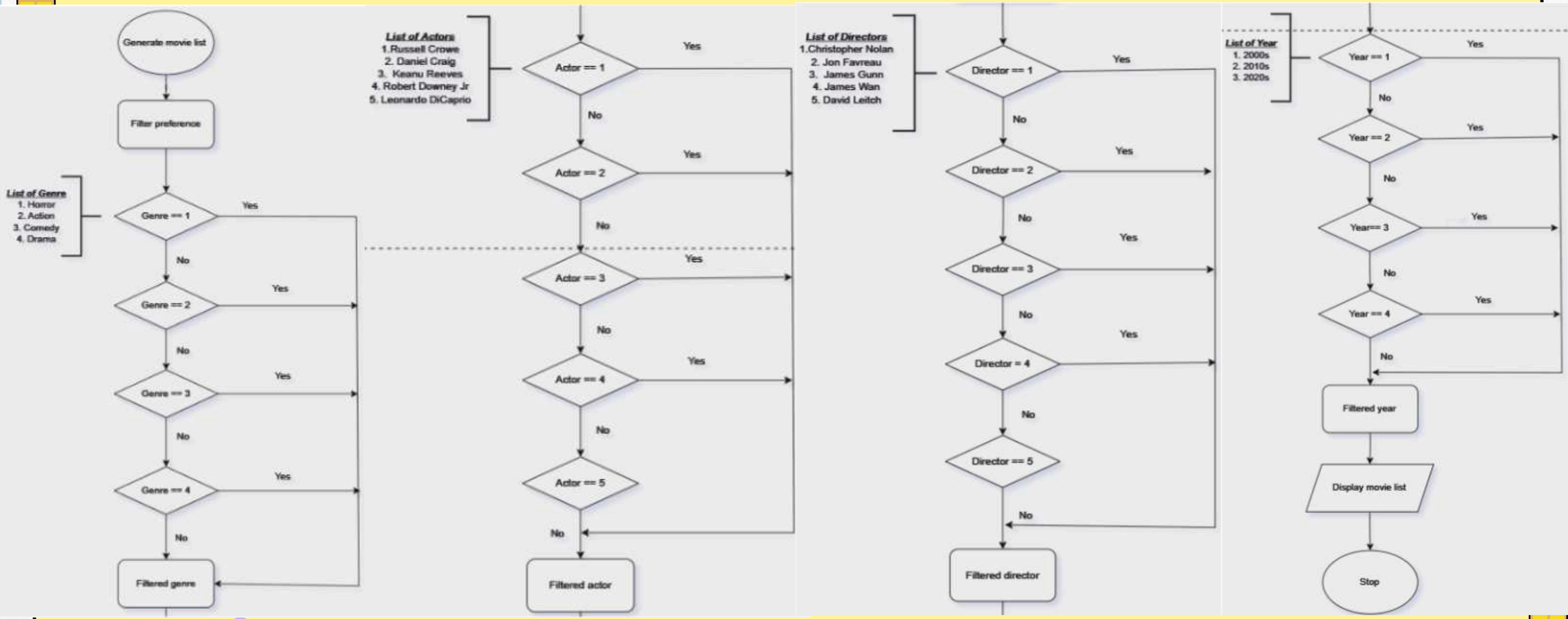
2.0 PROJECT DESIGN

• FLOW CHART (MOVIE MATCHMAKER)



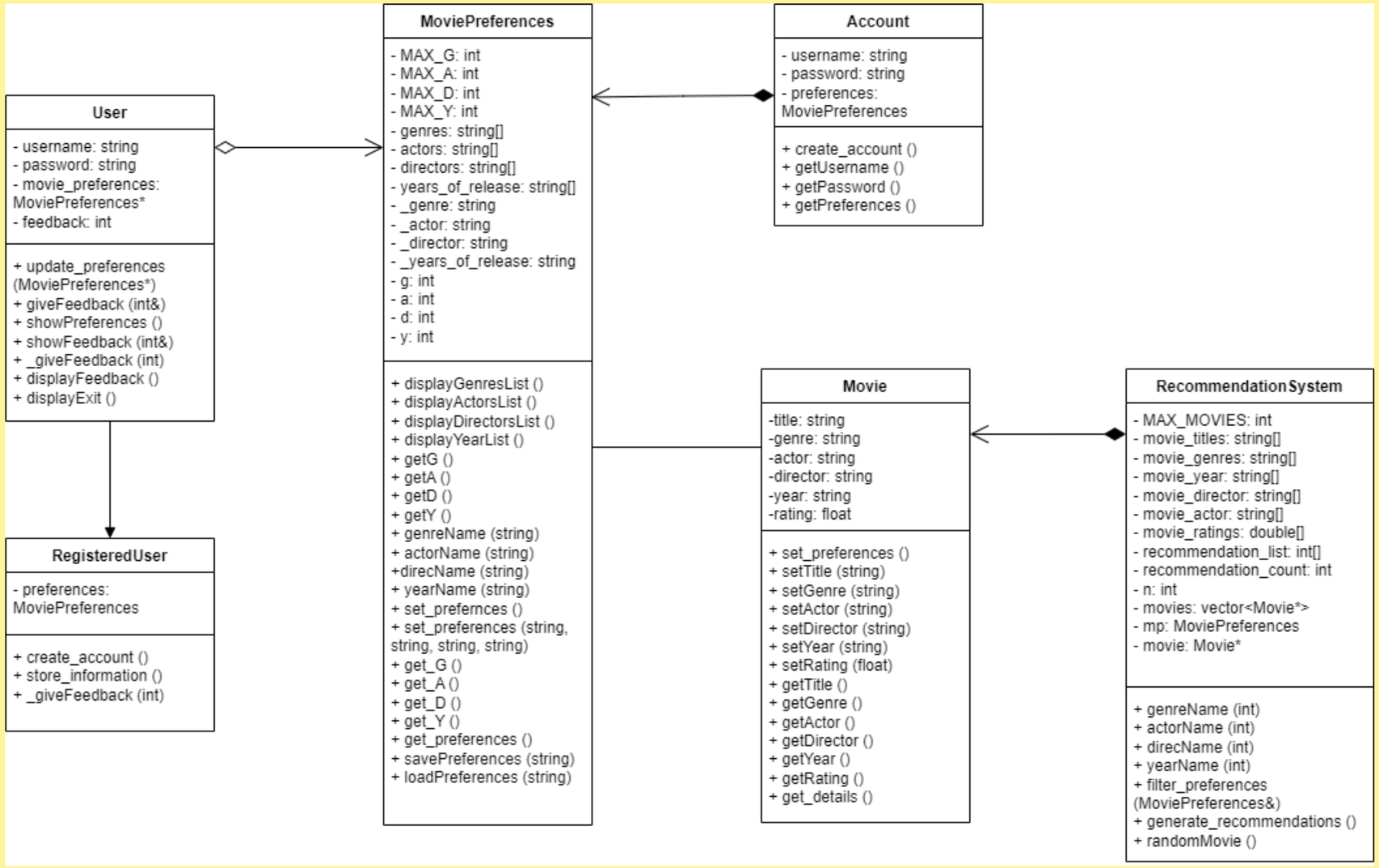
2.0 PROJECT DESIGN

• FLOW CHART (GENERATE MOVIE LIST)



2.0 PROJECT DESIGN

• UML DIAGRAM



3.0 OO CONCEPTS EMPLOYED

• ENCAPSULATION

```
// MoviePreferences class
class MoviePreferences {
private:
    static const int MAX_G = 4, MAX_A = 5, MAX_D = 5, MAX_Y = 3;
    string genres[MAX_G] = {"Horror", "Action", "Comedy", "Drama"};
    string actors[MAX_A] = {"Russell Crowe", "Daniel Craig", "Keanu Reeves", "Robert Downey Jr", "Leonardo DiCaprio"};
    string directors[MAX_D] = {"Christopher Nolan", "Jon Favreau", "James Gunn","James Wan", "David Leitch"};
    string years_of_release[MAX_Y] = {"2000s", "2010s", "2020s"};
    string _genre, _actor, _director, _years_of_release;
    int g, a, d, y;

public:
    MoviePreferences(string genre = "", string actor = "", string director = "", string year = "")
        : _genre(genre), _actor(actor), _director(director), _years_of_release(year) {}
    ~MoviePreferences() {}

    // Display genre
    void displayGenresList() {
        do {
            cout << " Genres's List :" << endl;
            for (int i = 0; i < MAX_G; ++i)
                cout << "  " << i+1 << " - " << genres[i] << endl;
            cout << " Choose one of the Genres => ";
            cin >> g;

            if (g <= 0 || g > MAX_G) {
                cout << endl;
                cout << " *** INVALID. Please choose between 1 - " << MAX_G << " ***" << endl << endl << endl;
            }
        } while (g <= 0 || g > MAX_G);
        cout << endl;
    }

    // Display actor
    void displayActorsList() {
        do {
            cout << " Actor's List :" << endl;
            for (int i = 0; i < MAX_A; ++i)
                cout << "  " << i+1 << " - " << actors[i] << endl;
            cout << " Choose one of the Actors => ";
            cin >> a;
```

```
// Display actor
void displayActorsList() {
    do {
        cout << " Actor's List :" << endl;
        for (int i = 0; i < MAX_A; ++i)
            cout << "  " << i+1 << " - " << actors[i] << endl;
        cout << " Choose one of the Actors => ";
        cin >> a;

        if (a <= 0 || a > MAX_A) {
            cout << endl;
            cout << " *** INVALID. Please choose between 1 - " << MAX_A << " ***" << endl << endl << endl;
        }
    } while (a <= 0 || a > MAX_A);
    cout << endl;
}

// Display director
void displayDirectorsList() {
    do {
        cout << " Director's List :" << endl;
        for (int i = 0; i < MAX_D; ++i)
            cout << "  " << i+1 << " - " << directors[i] << endl;
        cout << " Choose one of the Directors => ";
        cin >> d;

        if (d <= 0 || d > MAX_D) {
            cout << endl;
            cout << " *** INVALID. Please choose between 1 - " << MAX_D << " ***" << endl << endl << endl;
        }
    } while (d <= 0 || d > MAX_D);
    cout << endl;
}

// Display year
void displayYearList() {
    do {
        cout << " YEAR of RELEASE LIST :" << endl;
        for (int i = 0; i < MAX_Y; ++i)
            cout << "  " << i+1 << " - " << years_of_release[i] << endl;
```

```cpp
// Display year
void displayYearList() {
    do {
        cout << " YEAR of RELEASE LIST :" << endl;
        for (int i = 0; i < MAX_Y; ++i)
            cout << "  " << i+1 << " - " << years_of_release[i] << endl;
        cout << " Choose one of the Years => ";
        cin >> y;

        if (y <= 0 || y > MAX_Y) {
            cout << endl;
            cout << " *** INVALID. Please choose between 1 - " << MAX_Y << " ***" << endl << endl << endl;
        }
    } while (y <= 0 || y > MAX_Y);
    cout << endl << endl << endl;
}
```

```cpp
// Set user's preferences
void set_preferences() {
    if (g > 0 && g <= MAX_G) { _genre = genres[g - 1]; }
    if (a > 0 && a <= MAX_A) { _actor = actors[a - 1]; }
    if (d > 0 && d <= MAX_D) { _director = directors[d - 1]; }
    if (y > 0 && y <= MAX_Y) { _years_of_release = years_of_release[y - 1]; }
}

// Set user's preference
void set_preferences(string _g, string _a, string _d, string _y) {
    _genre = _g;
    _actor = _a;
    _director = _d;
    _years_of_release = _y;
}

// Getter function
string get_G () const { return _genre; }
string get_A () const { return _actor; }
string get_D () const { return _director; }
string get_Y () const { return _years_of_release; }

// Display preferences
void get_preferences() const {
    cout << " YOUR PREFERENCES:" << endl;
    cout << "-------------------" << endl;
    cout << "  Genre           => " << _genre << endl;
    cout << "  Actor           => " << _actor << endl;
    cout << "  Director        => " << _director << endl;
    cout << "  Year of Release => " << _years_of_release << endl << endl << endl;
}
```

```cpp
// Movie class
class Movie {
private:
    string title, genre, actor, director, year;
    float rating;

public:
    Movie(string t = "", string g = "", string a = "", string d = "", string y = "", float r = 0)
        : title(t), genre(g), actor(a), director(d), year(y), rating(r) {}

    // Pure virtual function for setting preferences
    virtual void set_preferences(){}

    // Setter functions
    void setTitle(string t) { title = t; }
    void setGenre(string g) { genre = g; }
    void setActor(string a) { actor = a; }
    void setDirector(string d) { director = d; }
    void setYear(string y) { year = y; }
    void setRating(float r) { rating = r; }

    // Getter functions
    string getTitle() const { return title; }
    string getGenre() const { return genre; }
    string getActor() const { return actor; }
    string getDirector() const { return director; }
    string getYear() const { return year; }
    float getRating() const { return rating; }
```

```
    // Display movie's details
    void get_details() const {
        cout << "  Title           : " << title << endl;
        cout << "  Genre           : " << genre << endl;
        cout << "  Actor           : " << actor << endl;
        cout << "  Director        : " << director << endl;
        cout << "  Year of Release : " << year << endl;
        cout << "  Rating          : " << rating << endl << endl;
    }
};
```

• AGGREGATiON

```cpp
// User class
// Aggregation
class User {
protected:
    string username, password;
    MoviePreferences* movie_preferences;
    int feedback;

public:
    User(const string& uname, const string& pwd) : username(uname), password(pwd) {}

    // Update movie preferences
    void update_preferences(MoviePreferences* preferences) {
        movie_preferences = preferences;
    }

    // Feedback
    void giveFeedback(int& feedback) {
        this -> feedback = feedback;
    }

    // Show movie preferences
    void showPreferences() const {
        movie_preferences->get_preferences();
    }
```

```cpp
// Reply to user feedback
void showFeedback(int& feedback) const {
    switch (feedback) {
        case 1 : cout << " (Y_Y) We're sorry to hear about your bad experience and appreciate you bringing this to our attention." << endl;
            break;
        case 2 : cout << " (UwU) We're sorry to hear about your negative experience, that's definitely not what we want for our customers." << endl;
            break;
        case 3 : cout << " (O.O) We're really grateful and appreciate you taking the time to share your rating with us." << endl;
            break;
        case 4 : cout << " (^.^) We're really happy to hear about your positive feedback!" << endl;
            break;
        case 5 : cout << " (^3^) We're glad that you enjoyed our service!" << endl;
            break;
        default : cout << " *** Invalid Rating ***" << endl;
            break;
    }
}

// Pure virtual function
virtual void _giveFeedback(int feedback) {
    this->feedback = feedback;
}
```

```
    }

    // Display feedback menu
    void displayFeedback () {
        int feedback;
        cout << " * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl;
        cout << " *  FEEDBACK                                          * " << endl;
        cout << " * ----------                                         * " << endl;
        cout << " *   1 - Very Unsatisfied   (Y_Y)                     * " << endl;
        cout << " *   2 - Unsatisfied        (UwU)                     * " << endl;
        cout << " *   3 - Neutral            (O.O)                     * " << endl;
        cout << " *   4 - Satisfied          (^.^)                     * " << endl;
        cout << " *   5 - Very Satisfied     (^3^)                     * " << endl;
        cout << " * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl;
        cout << " Please rate our system from 1 to 5: ";
    }

    // Display exit menu
    void displayExit() {
        cout << endl << endl << endl;
        cout << "\t\t\t* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl;
        cout << "\t\t            >>> Thank you for using our system , see you again next time! <<< " << endl;
        cout << "\t\t\t* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *" << endl << endl;
    }
};
```

• COMPOSITION

```
// Account class
// Composition
class Account {
private:
    string username, password;
    MoviePreferences preferences;

public:
    Account(const string& uname = "", const string& pwd = "", MoviePreferences pref = MoviePreferences())
        : username(uname), password(pwd), preferences(pref) {}

    // Getter function
    const string& getUsernme () const { return username; }
    const string& getPassword () const { return password; }

    // Create new account
    void create_account() {
        cout << "----------------------------------" << endl;
        cout << "  Enter username: ";
        getline(cin, username);
        cout << "  Enter password: ";
        getline(cin, password);
        cout << endl;
    }

    MoviePreferences getPreferences() const { return preferences; }
};
```

• COMPOSITION & AGGREGATION

```
// Composition & Aggregation
class RecommendationSystem {
private:
    static const int MAX_MOVIES = 120;
    string movie_titles[MAX_MOVIES];
    string movie_genres[MAX_MOVIES];
    string movie_year[MAX_MOVIES];
    string movie_director[MAX_MOVIES];
    string movie_actor[MAX_MOVIES];
    double movie_ratings[MAX_MOVIES];
    int recommendation_list[MAX_MOVIES];
    int recommendation_count;
    int n;

    vector<Movie*> movies;
    MoviePreferences mp;
    Movie* movie;

public:
    RecommendationSystem() : n(0), recommendation_count(0) {
        // Read the INPUT2 file
        ifstream inputFile("INPUT2.txt");
        if (!inputFile.is_open()) {
            cerr << " Error opening input file 'INPUT2.txt'" << endl;
            return;
        }
```

```
    for (int i = 0; i < MAX_MOVIES; i++) {
        string title, genre, year, director, actor;
        double rating;

        getline(inputFile, title, ',');
        getline(inputFile, genre, ',');
        getline(inputFile, year, ',');
        getline(inputFile, director, ',');
        getline(inputFile, actor, ',');
        inputFile >> rating;
        inputFile.ignore();

        movie_titles[i] = title;
        movie_genres[i] = genre;
        movie_year[i] = year;
        movie_director[i] = director;
        movie_actor[i] = actor;
        movie_ratings[i]  = rating;

        n++;
    }
    inputFile.close();

    // Initialize movies vector with pointers to Movie objects
    for (int i = 0; i < n; i++) {
        movies.push_back(new Movie(movie_titles[i], movie_genres[i], movie_actor[i], movie_director[i], movie_year[i], movie_ratings[i]));
    }
}
```

```
// Mapping
string genreName (int n) {
    map <int, string> GName ={{1, "Horror"}, {2, "Action"}, {3, "Comedy"}, {4, "Drama"}};
    return GName[n];
}
string actorName (int n) {
    map <int, string> AName ={{1, "Russell Crowe"}, {2, "Daniel Craig"}, {3, "Keanu Reeves"}, {4, "Robert Downey Jr"}, {5, "Leonardo DiCaprio"}};
    return AName[n];
}
string direcName (int n) {
    map <int, string> DName ={{1, "Christopher Nolan"}, {2, "Jon Favreau"}, {3, "James Gunn"}, {4, "James Wan"}, {5, "David Leitch"}};
    return DName[n];
}
string yearName (int n) {
    map <int, string> YName ={{1, "2000s"}, {2, "2010s"}, {3, "2020s"}};
    return YName[n];
}

// Filter preferences
void filter_preferences(MoviePreferences& mp) {
    recommendation_count = 0;
    for (int i = 0; i < n; ++i) {
        if (movie_genres[i] == genreName(mp.getG()) && movie_year[i] == yearName(mp.getY())) {
            recommendation_list[recommendation_count++] = i;
        } else if (movie_genres[i] == mp.get_G() && movie_year[i] == mp.get_Y() ) {
            recommendation_list[recommendation_count++] = i;
        }
    }
}
```

```
    // Generate movie recommendation list
    void generate_recommendations() const {
        cout << "-----------------------" << endl;
        cout << " Recommendations List:" << endl;
        cout << "-----------------------" << endl;
        for (int i = 0; i < recommendation_count; ++i) {
            int index = recommendation_list[i];
            movies[index]->get_details();
        }
    }

    // Movie Matchmaker : Generate random movie list
    void randomMovie () {
        cout << "-----------------------" << endl;
        cout << " Movie Matchmaker List:" << endl;
        cout << "-----------------------" << endl;
        int index;

        // To avoid duplicate
        bool choosen[MAX_MOVIES] = {false};

        // Randomly generate 10 movies only
        for (int i = 0; i < 10; i ++) {
            do {
                index = rand() % movies.size();
            } while (choosen[index]);
            if (choosen[index] = true)
                movies[index]->get_details();
        }
    }
};
```

• INHERITANCE

```
// RegisteredUser class
// Inheritance
class RegisteredUser : public User {
private:
    MoviePreferences preferences;

public:
    RegisteredUser(const string& uname, const string& pwd, const MoviePreferences& pref)
        : User(uname, pwd), preferences(pref) {}

    // Indicate new account is created
    void create_account() {
        cout << "Account created successfully for user: " << username << endl;
    }

    // Indicate user's information is stored
    void store_information() {
        cout << "Storing information for user: " << username << endl;
    }

    // Override the virtual function
    void _giveFeedback(int feedback) override {
    // Specific implementation for giveFeedback in RegisteredUser
        this->feedback = feedback;
        User::giveFeedback(feedback);
    }
};
```

• POLYMORPHISM

```
// Pure virtual function for setting preferences
virtual void set_preferences(){}
```

```
}

// Pure virtual function
virtual void _giveFeedback(int feedback) {
    this->feedback = feedback;
}
```

```
// Override the virtual function
void _giveFeedback(int feedback) override {
// Specific implementation for giveFeedback in RegisteredUser
    this->feedback = feedback;
    User::giveFeedback(feedback);
}
```

• ARRAY OF OBJECTS

```
// RecommendationSystem class
// Composition & Aggregation
class RecommendationSystem {
private:
    static const int MAX_MOVIES = 120;
    string movie_titles[MAX_MOVIES];
    string movie_genres[MAX_MOVIES];
    string movie_year[MAX_MOVIES];
    string movie_director[MAX_MOVIES];
    string movie_actor[MAX_MOVIES];
    double movie_ratings[MAX_MOVIES];
    int recommendation_list[MAX_MOVIES];
    int recommendation_count;
    int n;
```

```
// MoviePreferences class
class MoviePreferences {
private:
    static const int MAX_G = 4, MAX_A = 5, MAX_D = 5, MAX_Y = 3;
    string genres[MAX_G] = {"Horror", "Action", "Comedy", "Drama"};
    string actors[MAX_A] = {"Russell Crowe", "Daniel Craig", "Keanu Reeves", "Robert Downey Jr", "Leonardo DiCaprio"};
    string directors[MAX_D] = {"Christopher Nolan", "Jon Favreau", "James Gunn","James Wan", "David Leitch"};
    string years_of_release[MAX_Y] = {"2000s", "2010s", "2020s"};
    string _genre, _actor, _director, _years_of_release;
    int g, a, d, y;
```

• ADVANCED FEATURES

```
// Mapping
string genreName (int n) {
    map <int, string> GName ={{1, "Horror"}, {2, "Action"}, {3, "Comedy"}, {4, "Drama"}};
    return GName[n];
}
string actorName (int n) {
    map <int, string> AName ={{1, "Russell Crowe"}, {2, "Daniel Craig"}, {3, "Keanu Reeves"}, {4, "Robert Downey Jr"}, {5, "Leonardo DiCaprio"}};
    return AName[n];
}
string direcName (int n) {
    map <int, string> DName ={{1, "Christopher Nolan"}, {2, "Jon Favreau"}, {3, "James Gunn"}, {4, "James Wan"}, {5, "David Leitch"}};
    return DName[n];
}
string yearName (int n) {
    map <int, string> YName ={{1, "2000s"}, {2, "2010s"}, {3, "2020s"}};
    return YName[n];
}
```

```
vector<Movie*> movies;
MoviePreferences mp;
Movie* movie;
```

4.0 SYSTEM DEMONSTRATION

0.0

- Sign Up
- Login
- Exit

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                       *
*       (^_^)/   WELCOME TO MOVIE RECOMMENDATION SYSTEM   (^_^)/        *
*                                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *


 1 - Sign Up
 2 - Login
 3 - Exit
Choose 1, 2 or 3 : |
```

1 - SIGN UP FOR NEW USER
2 - LOGIN FOR OLD USER
3 - EXIT THE SYSTEM

4.0 SYSTEM DEMONSTRATION

1.0

- New user? --> Sign Up
- Old user? --> Login
- Exit

ENTER USERNAME & PASSWORD

1

```
 USER SIGN UP
-----------------------------------
  Enter username: Ethan
  Enter password: ethan99


 Sign Up Successful!
-----------------------------------
```

```
 USER SIGN UP
---------------------------------
  Enter username: Ethan
  Enter password: ethan99

 Username already exists. Please choose a different username.

 Sign Up Failed!
---------------------------------


 USER SIGN UP
---------------------------------
  Enter username:
```

DUPLICATE USERNAME & PASSWORD

2

```
 Genres's List :
  1 - Horror
  2 - Action
  3 - Comedy
  4 - Drama
 Choose one of the Genres => 3

 Actor's List :
  1 - Russell Crowe
  2 - Daniel Craig
  3 - Keanu Reeves
  4 - Robert Downey Jr
  5 - Leonardo DiCaprio
 Choose one of the Actors => 5

 Director's List :
  1 - Christopher Nolan
  2 - Jon Favreau
  3 - James Gunn
  4 - James Wan
  5 - David Leitch
 Choose one of the Directors => 2

 YEAR of RELEASE LIST :
  1 - 2000s
  2 - 2010s
  3 - 2020s
 Choose one of the Years => 3



 Preferences saved successfully!
 YOUR PREFERENCES:
-------------------
  Genre           => Comedy
  Actor           => Leonardo DiCaprio
  Director        => Jon Favreau
  Year of Release => 2020s
```

SELECT PREFERENCES

4.0 SYSTEM DEMONSTRATION

1.0

- New user? --> Sign Up
- Old user? --> Login
- Exit

GENERATE MOVIE LIST

3

```
----------------------
 Recommendations List:
----------------------
  Title           : Palm Springs
  Genre           : Comedy
  Actor           : Andy Samberg
  Director        : Max Barbakow
  Year of Release : 2020s
  Rating          : 7.4

  Title           : Borat Subsequent Moviefilm
  Genre           : Comedy
  Actor           : Sacha Baron Cohen
  Director        : Jason Woliner
  Year of Release : 2020s
  Rating          : 6.7

  Title           : The King of Staten Island
  Genre           : Comedy
  Actor           : Pete Davidson
  Director        : Judd Apatow
  Year of Release : 2020s
  Rating          : 7.1

  Title           : Eurovision Song Contest: The Story of Fire Saga
  Genre           : Comedy
  Actor           : Will Ferrell
  Director        : David Dobkin
  Year of Release : 2020s
  Rating          : 6.5

  Title           : Bad Trip
  Genre           : Comedy
  Actor           : Eric Andre
  Director        : Kitao Sakurai
  Year of Release : 2020s
  Rating          : 6.5

  Title           : Barb and Star Go to Vista Del Mar
  Genre           : Comedy
  Actor           : Kristen Wiig
```

FEEDBACK FROM USER

4

```
--------------------------------------------------------------------------------


 * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  FEEDBACK                                       *
 * ----------                                      *
 *   1 - Very Unsatisfied   (Y_Y)                  *
 *   2 - Unsatisfied        (UwU)                  *
 *   3 - Neutral            (O.O)                  *
 *   4 - Satisfied          (^.^)                  *
 *   5 - Very Satisfied     (^3^)                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * *
 Please rate our system from 1 to 5: 4


 (^.^) We're really happy to hear about your positive feedback!



                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                            >>> Thank you for using our system , see you again next time! <<<
                        * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Press any key to continue . . .
```

4.0 SYSTEM DEMONSTRATION

2.0

- New user? --> Sign Up
- Old user? --> Login
- Exit

1 - MOVIE MATCHMAKER

1

```
USER LOGIN
---------------------------------
 Enter username: Ethan
 Enter password: ethan99

Login Successful!
---------------------------------


Welcome to our Movie Recommendation System, Ethan! (^o^)/

* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  1 - Need Something Fresh? ---> Movie MatchMaker  *
*  2 - Stick with the familiar.                     *
*  3 - Exit                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
Choose 1, 2 or 3:
```

GENERATE MOVIE LIST

2

```
-----------------------
Movie Matchmaker List:
-----------------------
 Title           : The Social Network
 Genre           : Drama
 Actor           : Jesse Eisenberg
 Director        : David Fincher
 Year of Release : 2010s
 Rating          : 7.7

 Title           : Us
 Genre           : Horror
 Actor           : Lupita Nyong'o
 Director        : Jordan Peele
 Year of Release : 2010s
 Rating          : 6.9

 Title           : The Descent
 Genre           : Horror
 Actor           : Shauna Macdonald
 Director        : Neil Marshall
 Year of Release : 2000s
 Rating          : 7.2

 Title           : Get Out
 Genre           : Horror
 Actor           : Daniel Kaluuya
 Director        : Jordan Peele
 Year of Release : 2010s
 Rating          : 7.7
```

4.0 SYSTEM DEMONSTRATION

2.0

- New user? --> Sign Up
- Old user? --> Login
- Exit

2 - STICK WITH THE FAMILIAR

1

```
USER LOGIN
-----------------------------------
 Enter username: Ethan
 Enter password: ethan99

Login Successful!
-----------------------------------


Welcome to our Movie Recommendation System, Ethan! (^o^)/

* * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  1 - Need Something Fresh? ---> Movie MatchMaker  *
*  2 - Stick with the familiar.                     *
*  3 - Exit                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * * *
Choose 1, 2 or 3:
```

2

```
YOUR PREFERENCES:
-------------------
 Genre           => Comedy
 Actor           => Leonardo DiCaprio
 Director        => Jon Favreau
 Year of Release => 2020s


-----------------------------------------------------------------------


-----------------------
Recommendations List:
-----------------------
 Title           : Palm Springs
 Genre           : Comedy
 Actor           : Andy Samberg
 Director        : Max Barbakow
 Year of Release : 2020s
 Rating          : 7.4

 Title           : Borat Subsequent Moviefilm
 Genre           : Comedy
 Actor           : Sacha Baron Cohen
 Director        : Jason Woliner
 Year of Release : 2020s
 Rating          : 6.7

 Title           : The King of Staten Island
 Genre           : Comedy
 Actor           : Pete Davidson
 Director        : Judd Apatow
 Year of Release : 2020s
 Rating          : 7.1

 Title           : Eurovision Song Contest: The Story of Fire Saga
 Genre           : Comedy
 Actor           : Will Ferrell
 Director        : David Dobkin
 Year of Release : 2020s
```

DISPLAY USER'S STORED PREFERENCES AND GENERATE MOVIE LIST

4.0 SYSTEM DEMONSTRATION

2.0

- New user? --> Sign Up
- Old user? --> Login
- Exit

3 - EXIT

```
* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  1 - Need Something Fresh? ---> Movie MatchMaker  *
*  2 - Stick with the familiar.                     *
*  3 - Exit                                         *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
Choose 1, 2 or 3: 3



* * * * * * * * * * * * * * * * * * * * * * * * * * *
*  FEEDBACK                                         *
* ----------                                        *
*   1 - Very Unsatisfied   (Y_Y)                    *
*   2 - Unsatisfied        (UwU)                    *
*   3 - Neutral            (O.O)                    *
*   4 - Satisfied          (^.^)                    *
*   5 - Very Satisfied     (^3^)                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * *
Please rate our system from 1 to 5: 5


(^3^) We're glad that you enjoyed our service!



                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
                         >>> Thank you for using our system , see you again next time! <<<
                    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

Press any key to continue . . .
```

GET USER'S FEEDBACK BEFORE EXIT THE SYSTEM

THANK YOU