Project Allocation System

1. **Project Statement:** Project Allocation System (PAS) automates and simplifies the process of Allocating projects to students. Teachers can simply add details on prompting for input and perform a number of operation modules including;
   - **Adding Projects**
   - **Updating Projects**
   - **Searching Projects**
   - **Deleting Projects**
   - **Display All Projects**

   It uses Linked lists to perform these operations without having to think about wasting memory. Linked list is a linear data structure just like array, but it has a number of advantages over array i.e. expand the size of the list depending on the number of nodes a user wants to add or deleting a specific node from the memory etc. All the instructions are straight forward and user-friendly.

2. **Description of Modules**: Description of modules including workings and expected inputs:
   1. **Adding Projects:** Adding a projects module includes;
      - **Prepend a project:** Adds projects at the beginning of the list.
      - **Append a project:** Adds projects at the end of the list.
      - **Add after a given project:** Adds projects after the given project.
   2. **Updating Projects:** This module includes;
      - **Update Title**
      - **Update Group Member names**
   3. **Searching Projects:** Searches a specific project and displays the details.
   4. **Deleting Projects:** Deletes a specific project.
   5. **Display All Projects:** Displays all projects present in list.

3. **Validity Checks**: From start of execution to end of program, various checks have been added so that only expected inputs can be processed. It includes
   - **User login check:** Only authorized users can enter and if user forgets password, then various other options are provided in order to recover it.
   - **Duplicate Project ID check:** Two projects can never have a same Project ID, which helps in real-time records.

4. **Source Code**:

```
#include <iostream>
#include <string>
using namespace std;

class Node
{
public:
    Node()
    {
```

```
        next = nullptr;
    }
    string projectTitle;
    string studentOneName, studentTwoName, studentThreeName;
    int projectID;
    Node* next;
};

class LinkedList
{
    Node* head;
    //creates a new node and returns its address
    Node* createNode(int projID, string projTitle, string studOne, string studTwo,string
studThree)
    {
        Node* newNode = new Node;
        newNode->projectTitle = projTitle;
        newNode->studentOneName = studOne;
        newNode->studentTwoName = studTwo;
        newNode->studentThreeName = studThree;
        newNode->projectID = projID;

        return newNode;
    }
public:
    LinkedList()
    {
        head = nullptr;
    }

    //Checks if list is empty or not
    bool isEmpty()
    {
        if(head == nullptr)
            return true;
        else
            return false;
    }

    //Add a new node at beginning
    void prependNode(int projectID, string projTitle, string studOne, string studTwo,string
studThree)
    {
        if(isEmpty())
        {
            head = createNode(projectID, projTitle, studOne, studTwo, studThree);
        }
        else
        {
            //creates a new node and add it to the beginning
            Node* newNode = createNode(projectID, projTitle, studOne, studTwo, studThree);
```

```
            newNode->next = head;
            head = newNode;
        }
    }

    //Add a new node at end of the list
    void appendNode(int projectID, string projTitle, string studOne, string studTwo,string
studThree)
    {
        if(isEmpty())
        {
            head = createNode(projectID, projTitle, studOne, studTwo, studThree);
        }
        else
        {
            //creates a new node and add it to the end
            Node* newNode = createNode(projectID, projTitle, studOne, studTwo, studThree);
            Node* temp = head;

            while(temp->next != nullptr)
            {
                temp = temp->next;
            }
            temp->next = newNode;
        }
    }

    //Add a new node at after a given node
    void addNodeAfter(int findID, int projectID, string projTitle, string studOne, string
studTwo,string studThree)
    {
        if(isEmpty())
        {
            head = createNode(projectID, projTitle, studOne, studTwo, studThree);
        }
        else
        {
            //creates a new node and add it after a given node
            Node* newNode = createNode(projectID, projTitle, studOne, studTwo, studThree);
            Node* temp = head;

            while(temp->projectID != findID)
            {
                temp = temp->next;
            }

            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
```

```
//Takes Project Title as parameter and returns ture if present
Node* searchNode(int projID)
{
    //bool flag = false;

    if(isEmpty())
        return nullptr;
    else
    {
        Node* temp = head;

        while(temp != nullptr)
        {
            if(temp->projectID == projID)
            {
                break;
            }
            temp = temp->next;
        }
        return temp;
    }
}
//updates existing node
void updateExistingNode(int projID)
{
    if(isEmpty())
        cout << "List is Empty!" << endl;
    else if(searchNode(projID) == nullptr)
    cout << "Project with this ID, does not exist!" << endl;
    else
    {
        int userInput = 0;
        string update = " ";
        Node* tempNode = head;

        while (tempNode->next != nullptr)
        {
            if(tempNode->projectID == projID)
                break;
            tempNode = tempNode->next;
        }

        do
        {
            cout << "1- Update Project Title \n2- Update Name of Member 1 \n3- Update
Name of Member 2 \n4- Update Name of Member 3 \n5- Exit\nPlease make a choice: "; cin
>> userInput;
            if(userInput == 1)
            {
                cout << "Please enter Project Title: "; cin.ignore(); getline(cin, update);
```

```
            tempNode->projectTitle = update;
        }
        else if(userInput == 2)
        {
            cout << "Please enter New Name: "; cin.ignore(); getline(cin, update);
            tempNode->studentOneName = update;
        }
        else if(userInput == 3)
        {
            cout << "Please enter New Name: "; cin.ignore(); getline(cin, update);
            tempNode->studentTwoName = update;
        }
        else if(userInput == 4)
        {
            cout << "Please enter New Name: "; cin.ignore(); getline(cin, update);
            tempNode->studentThreeName = update;
        }
        else if(userInput == 5)
        {
            break;
        }
        else
        {
            cout << "Invalid Choice!" << endl;
        }
    }while((userInput > 0 && userInput <= 4) || userInput != 5);
  }
}
//Finds a node and deletes
void deleteNode(int projID)
{
  if(isEmpty())
    cout << "List is Empty!" << endl;
  else if(searchNode(projID) == nullptr)
    cout << "Project with this ID, does not exist!" << endl;
  else
  {
    Node* tempNode = head;

    while (tempNode->next != nullptr)
    {
      if(tempNode->projectID == projID)
        break;
      tempNode = tempNode->next;
    }

    if(tempNode == head)
    {
      Node* delNode = head;
      head = head->next;
```

```
            delete delNode;
        }
        else if(tempNode->next == nullptr)
        {
            Node* delNode = tempNode->next;

            Node* tempForSearch = head;
            while(tempForSearch->next->next != nullptr)
            {
                tempForSearch = tempForSearch->next;
            }
            tempForSearch->next = nullptr;

            delete delNode;
        }
        else
        {
            Node* delNode = nullptr;
            Node* tempNode = head;
            Node* previousNode = nullptr;
            Node* nextNode = nullptr;

            while (tempNode->next != nullptr)
            {
                if(tempNode->next->projectID == projID)
                    break;
                tempNode = tempNode->next;
            }

            previousNode = tempNode;
            nextNode = tempNode->next->next;
            delNode = tempNode->next;

            previousNode->next = nextNode;

            delete delNode;
        }
    }
}
bool duplicateProjectID(unsigned int tempID)
{
    bool flag = false;

    if(isEmpty())
        return flag;
    else
    {
        Node* tempNode = head;

        while(tempNode != nullptr)
        {
```

```
            if(tempNode->projectID == tempID)
            {
                flag = true;
                break;
            }
            tempNode = tempNode->next;
        }
        return flag;
    }
}

//Displays linked list
void traverse()
{
    for (Node* temp = head; temp != nullptr; temp = temp->next)
    {
        cout << "Project ID: " << temp->projectID << endl;
        cout << "Project Title: " << temp->projectTitle << endl;
        cout << "Group Member 1: " << temp->studentOneName << endl;
        cout << "Group Member 2: " << temp->studentTwoName << endl;
        cout << "Group Member 3: " << temp->studentThreeName << endl;
    }
    cout << endl << endl;
}

};

int main()
{
    LinkedList projectsList;
    char userInput = '\0';
    string username, password;

    cout << "1- Enter Login \n2- Signup \n3- Reset Password \nPlease make a choice: "; cin
>> userInput;

    if(userInput == '1')
    {
        cout << "Please enter username: "; cin >> username;
        cout << "Please enter password: "; cin >> password;
    }
    else if(userInput == '2')
    {
        char choice = '\0';
        cout << "Only 1 username and password left: user \nPress 1 to assign: "; cin >>
choice;

        if(choice == '1')
        {
            username = "user";
            password = "user";
```

```
            cout << "Successfully assigned!" << endl;
        }
    }
    else
    {
        again:
        cout << "Please enter your phone number +92***-*****67: "; cin >> password;

        if(password == "+92300-1234567")
        {
            cout << "Your username and password is: admin\nPlease try logging again!" <<
endl;
        }
        else
        {
            cout << "Incorrect!" << endl;
            goto again;
        }
    }

    if((username == "admin" && password == "admin") || (username == "user" && password
== "user"))
    {
        do
        {
            cout << "-------------------------------------------------------------------------------------" <<
endl;
            cout << "\t\t\t\t\t\t\tWelcome to Project Allocation System" << endl;
            cout << "-------------------------------------------------------------------------------------" <<
endl;
            cout << "1- Add a Project \n2- Delete a Project \n3- Search a Project \n4- Update
an Existing Project Details \n5- Display All Projects \nPlease make a choice: "; cin >>
userInput;
            cout << "-------------------------------------------------------------------------------------" <<
endl;

            switch (userInput)
            {
                case '1':
                {
                    unsigned int projID = 0, find = 0;
                    string ProjectTitle = " ", name_1 = " ", name_2 = " ", name_3 = " ";
                    do
                    {
                        cout << "-------------------------------------------------------------------------------
---" << endl;
                        cout << "1- Prepend a Project \n2- Append a Project at End \n3- Add a
Project After Given Project \n4- Exit \nPlease make a choice: "; cin >> userInput;
                        cout << "-------------------------------------------------------------------------------
---" << endl;
                        if(userInput == '1')
```

```
                {
                    cout << "--------------------------------------------------------------------------------
-------" << endl;
                    cout << "\t\t\t\t\t\t\t\t\tAllocating New Project" << endl;
                    cout << "--------------------------------------------------------------------------------
-------" << endl;
                    cout << "Please enter Project ID: "; cin >> projID;

                    if(!projectsList.isEmpty())
                    {
                        if(projectsList.duplicateProjectID(projID))
                        {
                            cout << "Sorry this ID is already assigned!" << endl;
                        }
                        else
                        {
                            cout << "Please enter Project Title: "; cin.ignore(); getline(cin,
ProjectTitle);
                            cout << "Please enter Name of Member 1: "; cin.ignore(); getline(cin,
name_1);
                            cout << "Please enter Name of Member 2: "; cin.ignore(); getline(cin,
name_2);
                            cout << "Please enter Name of Member 3: "; cin.ignore(); getline(cin,
name_3);

                            projectsList.prependNode(projID, ProjectTitle, name_1, name_2,
name_3);
                        }
                    }
                    else
                    {
                        cout << "Please enter Project Title: "; cin.ignore(); getline(cin,
ProjectTitle);
                        cout << "Please enter Name of Member 1: "; cin.ignore(); getline(cin,
name_1);
                        cout << "Please enter Name of Member 2: "; cin.ignore(); getline(cin,
name_2);
                        cout << "Please enter Name of Member 3: "; cin.ignore(); getline(cin,
name_3);

                        projectsList.prependNode(projID, ProjectTitle, name_1, name_2,
name_3);
                    }
                }
                else if (userInput == '2')
                {
                    cout << "--------------------------------------------------------------------------------
-------" << endl;
                    cout << "\t\t\t\t\t\t\t\t\tAllocating New Project" << endl;
                    cout << "--------------------------------------------------------------------------------
-------" << endl;
```

```
                    cout << "Please enter Project ID: "; cin >> projID;

                    if(!projectsList.isEmpty())
                    {
                        if(projectsList.duplicateProjectID(projID))
                        {
                            cout << "Sorry this ID is already assigned!" << endl;
                        }
                        else
                        {
                            cout << "Please enter Project Title: "; cin.ignore(); getline(cin,
ProjectTitle);
                            cout << "Please enter Name of Member 1: "; cin.ignore(); getline(cin,
name_1);
                            cout << "Please enter Name of Member 2: "; cin.ignore(); getline(cin,
name_2);
                            cout << "Please enter Name of Member 3: "; cin.ignore(); getline(cin,
name_3);

                            projectsList.appendNode(projID, ProjectTitle, name_1, name_2,
name_3);
                        }
                    }
                    else
                    {
                        cout << "Please enter Project Title: "; cin.ignore(); getline(cin,
ProjectTitle);
                        cout << "Please enter Name of Member 1: "; cin.ignore(); getline(cin,
name_1);
                        cout << "Please enter Name of Member 2: "; cin.ignore(); getline(cin,
name_2);
                        cout << "Please enter Name of Member 3: "; cin.ignore(); getline(cin,
name_3);

                        projectsList.appendNode(projID, ProjectTitle, name_1, name_2,
name_3);
                    }
                }
                else if (userInput == '3')
                {
                    cout << "-------------------------------------------------------------------------------
-------" << endl;
                    cout << "\t\t\t\t\t\t\t\t\t\tAllocating New Project" << endl;
                    cout << "-------------------------------------------------------------------------------
-------" << endl;
                    cout << "Please enter Project ID to be found: "; cin >> find;

                    if(!projectsList.isEmpty())
                    {
                        if(projectsList.searchNode(find))
                        {
```

```
                        cout << "Please enter Project ID for New Project: "; cin >> projID;

                        if(projectsList.duplicateProjectID(projID))
                        {
                            cout << "Sorry this ID is already assigned!" << endl;
                        }
                        else
                        {
                            cout << "Please enter Project Title for New Project: "; cin.ignore();
getline(cin, ProjectTitle);
                            cout << "Please enter Name of Member 1 for New Project: ";
cin.ignore(); getline(cin, name_1);
                            cout << "Please enter Name of Member 2 for New Project: ";
cin.ignore(); getline(cin, name_2);
                            cout << "Please enter Name of Member 3 for New Project: ";
cin.ignore(); getline(cin, name_3);

                            projectsList.addNodeAfter(find, projID, ProjectTitle, name_1,
name_2, name_3);
                        }
                    }
                    else
                    {
                        cout << "Sorry node not found!" << endl;
                    }
                }
                else
                {
                    cout << "Please enter Project Title for New Project: "; cin.ignore();
getline(cin, ProjectTitle);
                    cout << "Please enter Name of Member 1 for New Project: ";
cin.ignore(); getline(cin, name_1);
                    cout << "Please enter Name of Member 2 for New Project: ";
cin.ignore(); getline(cin, name_2);
                    cout << "Please enter Name of Member 3 for New Project: ";
cin.ignore(); getline(cin, name_3);

                    projectsList.addNodeAfter(find, projID, ProjectTitle, name_1, name_2,
name_3);
                }
            }
            else if(userInput > '4')
            {
                cout << "Invalid Choice!" << endl;
            }
        }while(userInput != '4');
    }
        break;
    case '2':
    {
```

```
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                cout << "\t\t\t\t\t\t\t\t\t\tProject Deletion" << endl;
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                unsigned int find = 0;
                cout << "Please enter Project ID to be deleted: "; cin >> find;
                projectsList.deleteNode(find);
            }
                break;
            case '3':
            {
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                cout << "\t\t\t\t\t\t\t\t\tSearching a Project" << endl;
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                unsigned int find = 0;
                cout << "Please enter Project ID to be searched: "; cin >> find;

                Node* tempNode = projectsList.searchNode(find);

                    if(tempNode == nullptr)
                        cout << "Project Not Found!" << endl;
                    else
                    {
                        cout << "----------------------------" << endl;
                        cout << "\t\tProject Found!" << endl;
                        cout << "----------------------------" << endl;
                        cout << "Project ID: " << tempNode->projectID << endl;
                        cout << "Project Title: " << tempNode->projectTitle << endl;
                        cout << "Group Member 1: " << tempNode->studentOneName << endl;
                        cout << "Group Member 2: " << tempNode->studentTwoName << endl;
                        cout << "Group Member 3: " << tempNode->studentThreeName << endl;
                    }
            }
                break;
            case '4':
            {
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                cout << "\t\t\t\t\t\t\t\t\tProject Updation" << endl;
                cout << "-------------------------------------------------------------------------------------------
" << endl;
                unsigned int find = 0;
                cout << "Please enter Project ID to update: "; cin >> find;
                projectsList.updateExistingNode(find);
            }
                break;
            case '5':
            {
```

```
                cout << "-----------------------------------------------------------------------------------------
" << endl;
                cout << "\t\t\t\t\t\t\t\t\t\t\tAllocated Projects" << endl;
                cout << "-----------------------------------------------------------------------------------------
" << endl;
                if(projectsList.isEmpty())
                    cout << "List is empty, consider adding projects first!" << endl;
                else
                    projectsList.traverse();
            }
                break;

            case '0':
            {
                cout << "Exiting..." << endl;
            }
                break;

            default:
                cout << "Invalid Choice!" << endl;
                break;
        }

    }while(userInput != '0');
  }
  else
    cout << "Invalid username or password" << endl;

  return 0;
}
```