DSA Project Report

Project Title	**CLI Based Facebook App**
Grp Member	**Muhamad Huzzifa & Hafiz Abdul Rehman**
Reg No	**2020-CS-606 & 2020-CS-624**
Grp No	**Grp-01**

UNIVERSITY OF ENGINEERING AND TECNOLOGY LAHORE NEW CAMPUS

Contents

Classes

1. Str Class (I / O for Sign Up and Log In)
2. Post Class (I / O for for Post Class)
3. PNode Class (Post Node (next and pre))
4. PostLinking Class (add, update and delete Post)
5. FNode Class (Friend Node (name, next and pre))
6. FriendLinking Class (add, un-follow and View friend)
7. Node Class (str, post and friend Object)
8. Sign-Up Class (Insert User, Log-In and Search function)
9. Comment Class (I / O for Comment)
10. CNode Class (Comment Object, next and pre)
11. CommentLinking Class (Add and View Comment)
12. Message Class (I / O for Message)
13. MNode Class (Message Object, next and pre)
14. MessageLinking Class (Send, View and Delete Message)

Personal Tab

Data Structure:

Doubly linked list

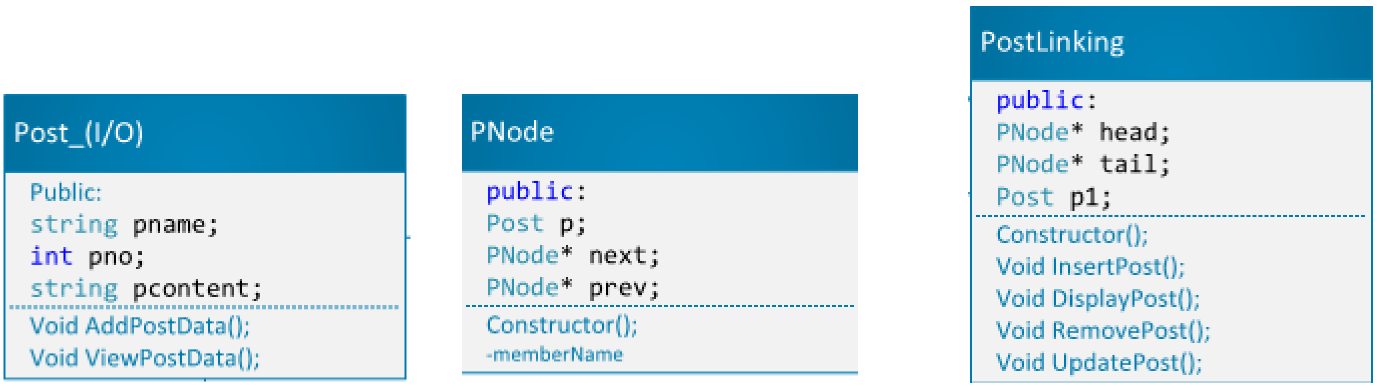
Functionality:

1) ADD POST
2) VIEW POST
3) REMOVE POST
4) UPDATE POST

Description:

In this Tab we are using Doubly linked list (Data Structure). In add Post function the Log-In User added some posts and in the View Tab user can view his post. In Remove post Function user can delete and update his post. This class is linking with Node and post (I / O) class.

Diagram:



```
Post_(I/O)
Public:
string pname;
int pno;
string pcontent;
Void AddPostData();
Void ViewPostData();
```

```
PNode
public:
Post p;
PNode* next;
PNode* prev;
Constructor();
-memberName
```

```
PostLinking
public:
PNode* head;
PNode* tail;
Post p1;
Constructor();
Void InsertPost();
Void DisplayPost();
Void RemovePost();
Void UpdatePost();
```

Home Tab

Data Structure:

Doubly linked list

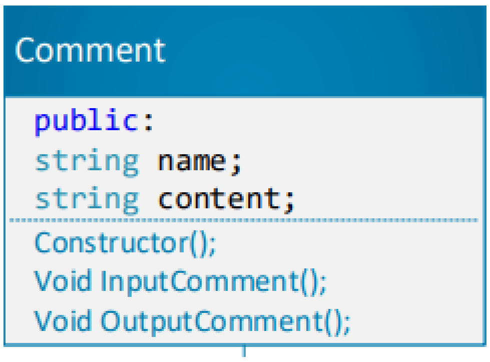
Functionality:

1) ADD POST
2) VIEW POST
3) REMOVE POST
4) UPDATE POST
5) ADD COMMENT
6) VIEW COMMENT

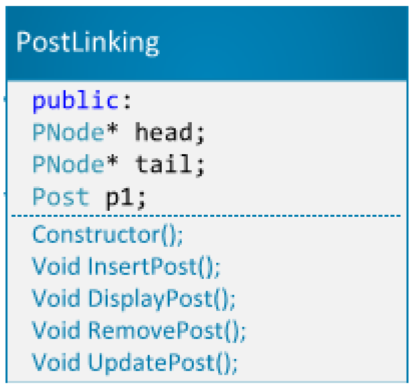
Description:

In this Tab we are using Doubly linked list (Data Structure). In add Post function the Log-In User added some posts and in the View Tab user can view his post and his friend post. In Remove post Function user can delete and update his post. In this function we add some comment on his post and view comments on his post. This class is linking with Node and post (I / O) class.

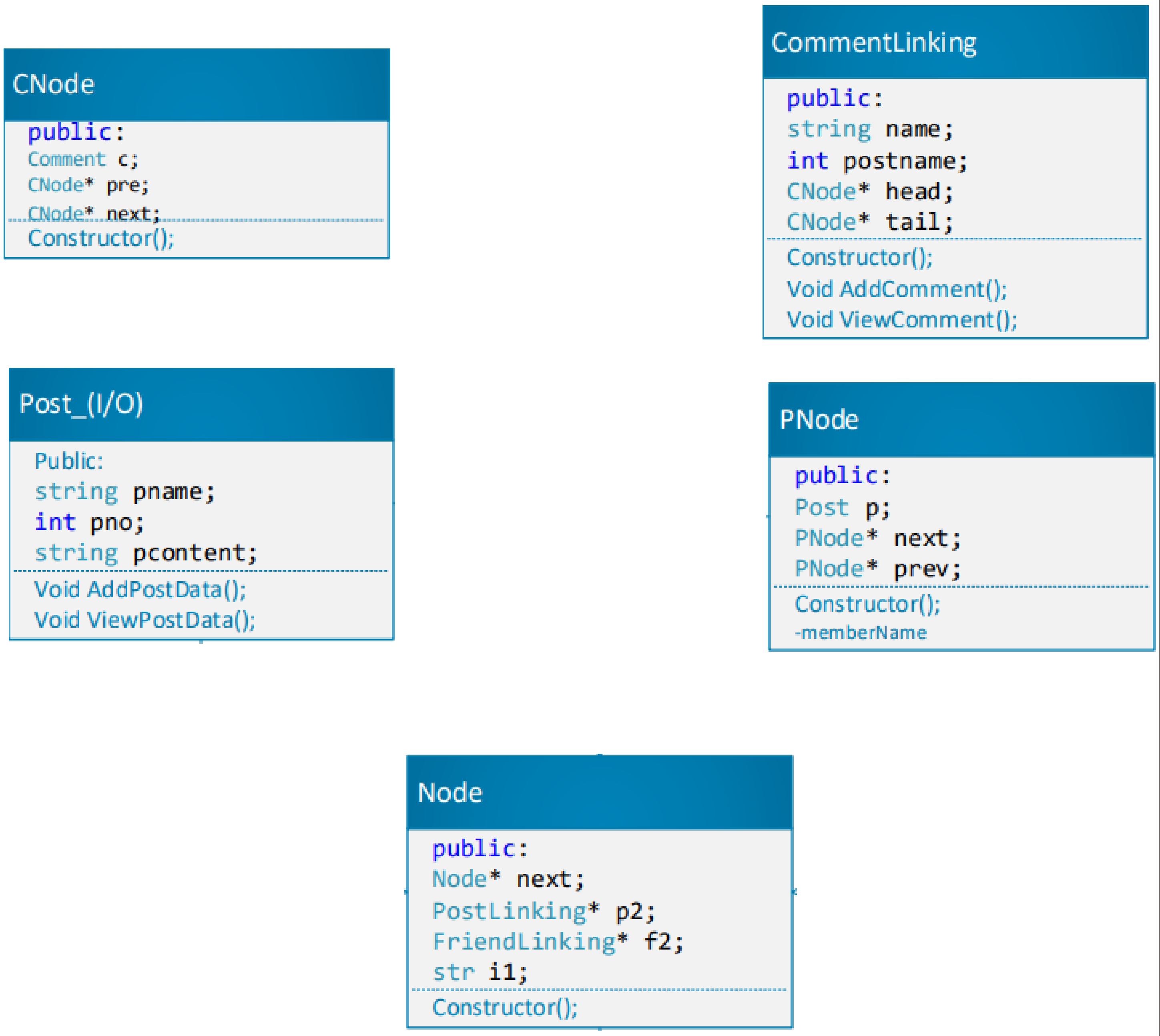
Diagram:



```
Comment
public:
string name;
string content;
Constructor();
Void InputComment();
Void OutputComment();
```



```
PostLinking
public:
PNode* head;
PNode* tail;
Post p1;
Constructor();
Void InsertPost();
Void DisplayPost();
Void RemovePost();
Void UpdatePost();
```



```
CNode
public:
Comment c;
CNode* pre;
CNode* next;
-----------------
Constructor();
```

```
CommentLinking
public:
string name;
int postname;
CNode* head;
CNode* tail;
-----------------
Constructor();
Void AddComment();
Void ViewComment();
```

```
Post_(I/O)
Public:
string pname;
int pno;
string pcontent;
-----------------
Void AddPostData();
Void ViewPostData();
```

```
PNode
public:
Post p;
PNode* next;
PNode* prev;
-----------------
Constructor();
-memberName
```

```
Node
public:
Node* next;
PostLinking* p2;
FriendLinking* f2;
str i1;
-----------------
Constructor();
```

Friends Tab

Data Structure:

Doubly linked list

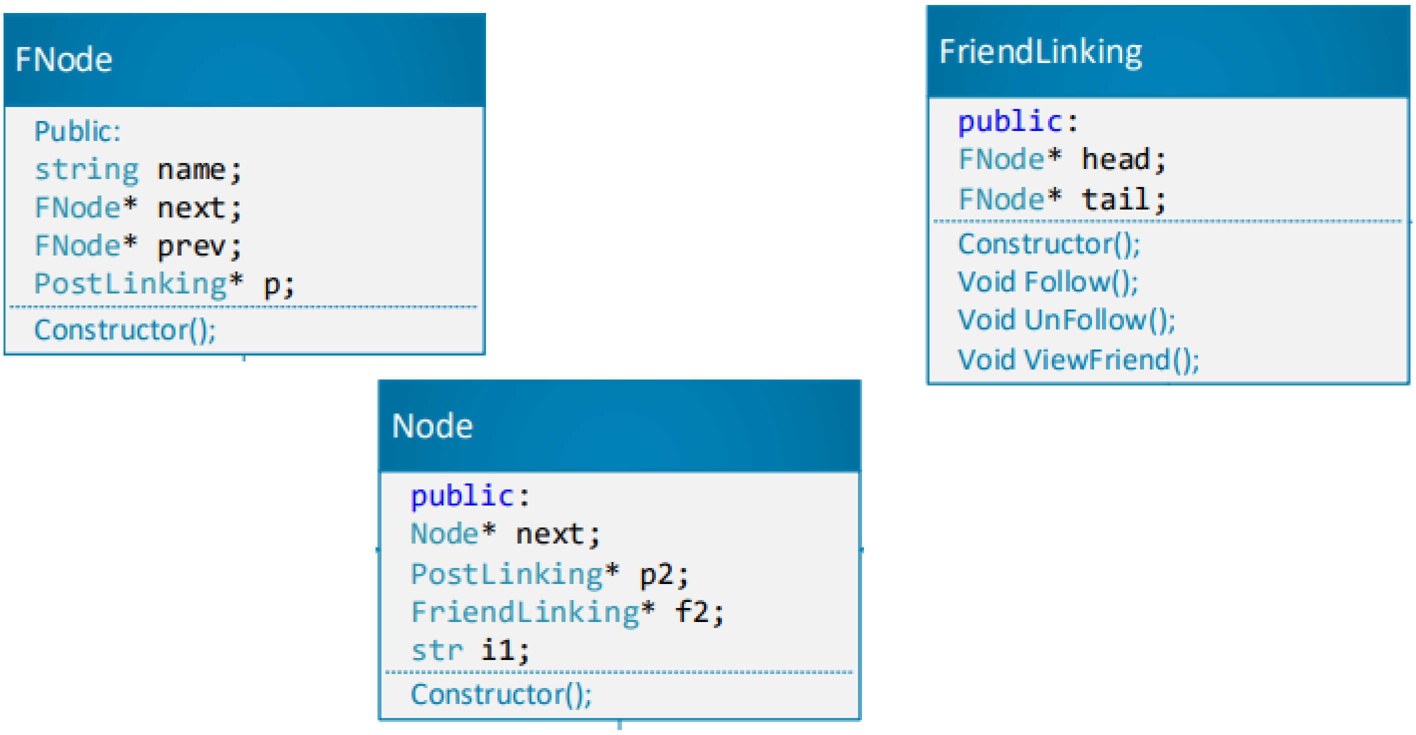
Functionality:

1) ADD FRIEND
2) VIEW FRIENDLIST

Description:

In this Tab we are using Doubly linked list (Data Structure). In add Friend function the Log-In User added some friend that are already sign-Up and in the View Friend List user can view his friend in the List.

Diagram:



```
FNode
Public:
string name;
FNode* next;
FNode* prev;
PostLinking* p;
Constructor();
```

```
FriendLinking
public:
FNode* head;
FNode* tail;
Constructor();
Void Follow();
Void UnFollow();
Void ViewFriend();
```

```
Node
public:
Node* next;
PostLinking* p2;
FriendLinking* f2;
str i1;
Constructor();
```

Message Tab

Data Structure:

Doubly linked list

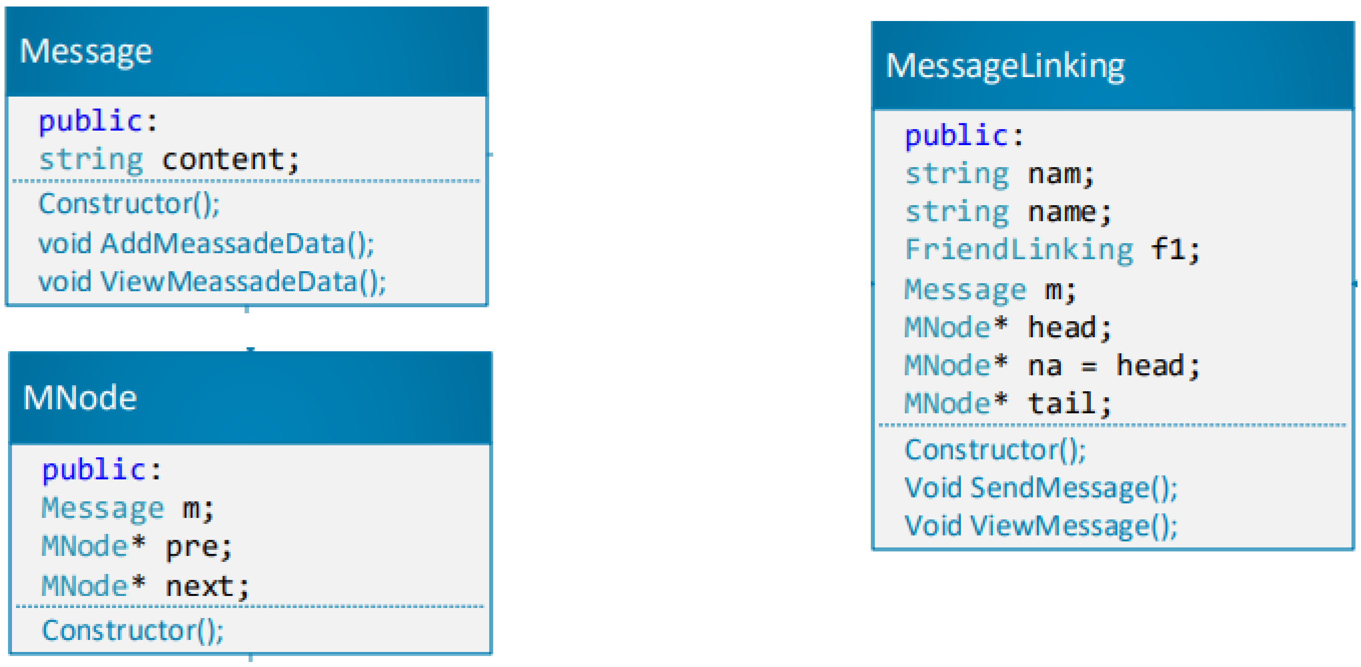
Functionality:

1) SELECT FRIEND AND SEND MESSAGE
2) VIEW MESSAGE
3) DELETE MESSAGE

Description:

In this Tab we are using Doubly linked list (Data Structure). In send Message function the Log-In User send message to added friends by selecting one of them. User can delete message and view Message through their Proper Function

Diagram:



```
Message
public:
string content;
Constructor();
void AddMeassadeData();
void ViewMeassadeData();
```

```
MNode
public:
Message m;
MNode* pre;
MNode* next;
Constructor();
```

```
MessageLinking
public:
string nam;
string name;
FriendLinking f1;
Message m;
MNode* head;
MNode* na = head;
MNode* tail;
Constructor();
Void SendMessage();
Void ViewMessage();
```