PROJECT REPORT

PROBLEM STATEMENT

This project is made with efficient use of Data Structures and File Handling. Also, to make it good looking and easier interface for users Graphics has been used wherever required.
What can users do ??
Well, there are lot of features for users to make their social links online. These features are as follows :

- View and send request to other users to connect
- Post their latesh updates, status, and other things
- Create online profile having information like DOB
- Watch all friends activity together under feature NEWSFEED
- See your friend`s profile

FUNTIONS USED

User-defined

void save_db();
void initialize();
void main_register();

```
void main_login();
void insert_register(struct member *,int);
void main_reset();
void dashboard();
void newsfeed(struct member *);
void member_stats(struct member *);
void add_status(struct member *);
void show_status(struct member *);
void send_req(struct member *);
void add_req(struct member *,int);
void show_req(struct member *);
void viewprofile(struct member *);
void friend_profile(struct member *);
void accept_req(struct member *,int);
void show_friends(struct member *);
void load_db(void);
void wheel(void);
int home_page(void);
void newbar(int,int,int,int);
void skbly7(int,int,char*);
void encrypt(char *);
struct member * search_by_id(int);
struct member * search_by_name(char []);
```

Pre-defined

```
outtextxy();
settextstyle();
setfontstyle();
flushall();
printf();
scanf();
gets();
strcpy();
bar();
malloc();
strcmp();
fopen();
fclose();
```

fread();
fwrite();
fprintf();
fscanf();
itoa();
getch();
clrscr();

IMPORTANT CORE FUNCTION`S DESCRIPTION

I feel that all functions used are equally important for the project.
But still some of the most important function along with it`s description are as follow :

void main_register()
This function is called when a user/client select the option to register a new account, this create a new node in which all the details are taken from screen. After it according to the first character of name it gets arranged in a multi linked list of 26 pointers.

void encrypt()
This function uses clamper encryption to make passwords of the user somewhat more secured.

void send_req()
This function provides the user the opportunity to search for friends by NAME or by unique ID provided on the network. This further calls more function to actually make it happen which is told in detailed further in Algorithm part. Thus, after searching efficiently the request is sent through one member to other.

void accept_req()
This function tells the user about all the pending friend request of other members who want to get connected to him. After which he/she is

promt to enter id of users whom they want to add. After which both users get connected.

void save_db()
This function is used to properly store all the linked list properly in a managed way into our files named member.txt, status.txt, req.txt and main.txt.

void load_db()
This function is used to fetch our linked list from files and relink them properly whenever it is called. It is currently called when our program starts.

ALGORITHM

void main_register()
Step by step detail of how the function work :

1. Basic graphics function to make it good look (Not to be counted in algo)
2. Asks for desired username, password, security question answer and other basic info.
3. Create a new node of struct member and assign node->req , node->status and
   node->friend to NULL. It also assigns node->ts , node->tf , and node->tr equal to zero.
4. Then it take the name`s first character i.e. node->name[0] and try to match it from A to Z and a to z.
5. If match is found it call other function insert_register which insert this node to our multi linked list.

How does insert_register work ?
It`s algo is as follows :

1. Check if our multi linked list starting with character taken during call is empty or not.
2. If it is empty i.e. point to NULL. It assign the address of new node to that.

3. If it is not NULL, i.e. have element. It assign node->next to the existing address and then it replace its address from top[character-65] with new node`s address.

void encrypt()
It takes a char pointer when called.
Algorithm :

1. Take one character of the string at a time, and replace its ASCII value with 30.
2. This continue till it find '\0' in the string and then stop.

void send_request()
The algorithm is as follows :

1. Ask user to enter choice of searching node by name or by id.
2. If the user enters a name it further calls a function search_by_name which returns the address of node with that name else if user choose ID, then it call search_by_id which return the address of node with that ID.
3. Then the user is displayed the info i.e. name and id, and ask for Y to send request.
4. After pressing yes, a new node of type id_struct is created with ID of the member who sent request, and is added to linked list of member to whom request is sent.
5. After this process is completed, a successful message is displayed to the user.

void accept_request()
The algorithm of this function is as follows :

1. Traverse in the link list stored in temp->req where temp is pointer to node of logined user. Thus display all pending friend request.
2. Ask the user to input id of which he/she want to accept friend request.
3. Then the node with id inputted by user is shifted from temp->req linked list to temp->friend linked list.
4. Then a new node of type id_struct is created with id of member who accepted friend request and add it into ->friend list of the user whose request has been accepted.
5. Finally a successful message is displayed to the user

void save_db()
The algorithm is as follows :

1. It opens 5 files named status.txt , main.txt, status.txt, req.txt and friend.txt in write mode.
2. It work from id=1001 i.e. our starting id to present id number and then do line 3 and 4.
3. Search for node with id provided by for loop i.e. 2nd line…
4. In writes all data of member node once at a time to main.txt, and then it transverse in all linked list in it i.e. of friend, req, and status and write them in their file respectively using fwrite.
5. After all writing work is done it closed all the pointer.

void load_db()
The algorithm is as follows :

1. It opens 5 files named count.txt , main.txt, status.txt, req.txt and friend.txt in read mode.
2. It take one node from main.txt and assign it id =1001.
3. It then took number of nodes from other files i.e. status.txt , req.txt and friend.txt equivalent to number given in the node taken from main.txt and assign them to the member node after making linked list.
4. It assign the node to multi linked list seeing the first letter of the name.
5. A new node is maken with id=id+1 and this way it work while the member file end.
6. Thus, we regenerate our old linked list.

LIST OF DATA STRUCTURES USED

The data structures used in the project are as follows :
1. Linked List
2. Stack
3. Multi Linked List