**SOFTWARE DESIGN**

**SPECIFICATION**

**ONLINE DISCUSSION PORTAL**

**Shronit Bhargava**

**Utkarsh Srivastava**

**Sourabh Kaushal**

**Yatham Praveen**

**Siddhartha Kille**

**1.0 Introduction:**

The Software Design Document is a document to provide the knowledge about the working of the software and also how the components will work in accordance with the user and the web interface. This software design document will be used as an aid to understand the working of the software. The processes involved are depicted using class diagram, sequence diagrams and state diagram.

* 1. **Purpose:**

The purpose of this Software Design Document is to provide detailed information regarding how the software will be used and the components involved in this software and how the interaction between the user and the web interface will take place. Also, this document will provide detailed information regarding the processes involved when a user inputs provides the input and the result that is generated.

* 1. **Scope:**

This Software Design Document is for providing information about the base level working of the software and shows the feasibility for large scale use of this software. This document will cover the aspects of this particular software and detailed description of the modules and the components and their interaction with the user and the web interface.

This document will provide the basic working of the software in a particular sequence.

* 1. **Definitions acronyms and abbreviations:**

Not Applicable

* 1. **References:**

This document has been created based on the modules provided by the IEEE Standard 1016.

* 1. **Overview:**

The document includes:

1. Introduction
2. Sequence diagram
3. Class diagram
4. State machine diagram
5. Detailed description of components
6. Design Decision and trade offs
7. Pseudo code of the software

**2. System architecture description**

**2.1 Overview of modules / components**

**Componenets overview :**

Sequence Diagram: Sequence diagram will depict the usability of the software sequentially which makes the customer completely understand the procedural behaviour of the software.

Class Diagram: Class diagram depicts the relation between different classes present in the software scenario. It depicts the methodology used by a particular class and how the variables of one class affects the methodology of another class. This will help us to calculate the level of cohesion and coupling in the software.

State Diagram: State diagram depicts the behaviour of the state when a particular input arrives.

When an input goes in a particular state, it traverses to another state based on the given parameters.

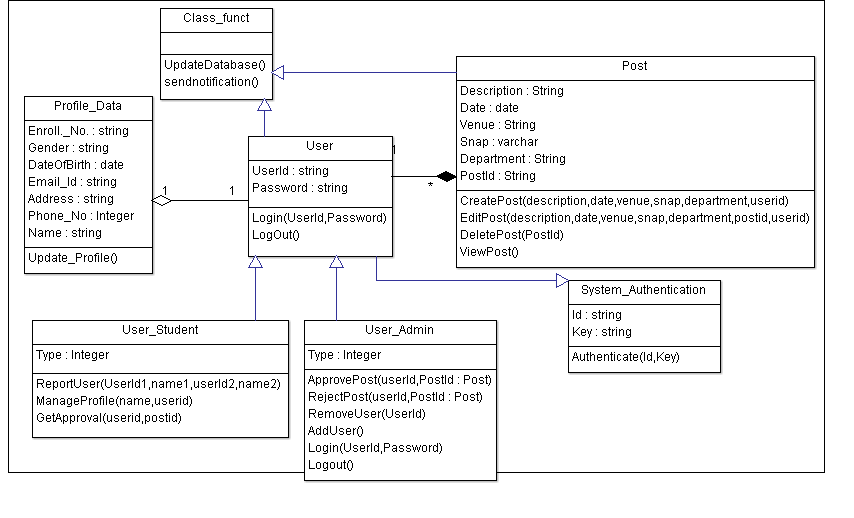
It gives a detailed information about the usability of the software from user’s perspective.

Pseudo code: Pseudo code gives the rough idea of working of the software in technical terms.

It gives the details of the different classes and how it is inherited or is a parent of another classes and how they are affected by the variables in all the classes.

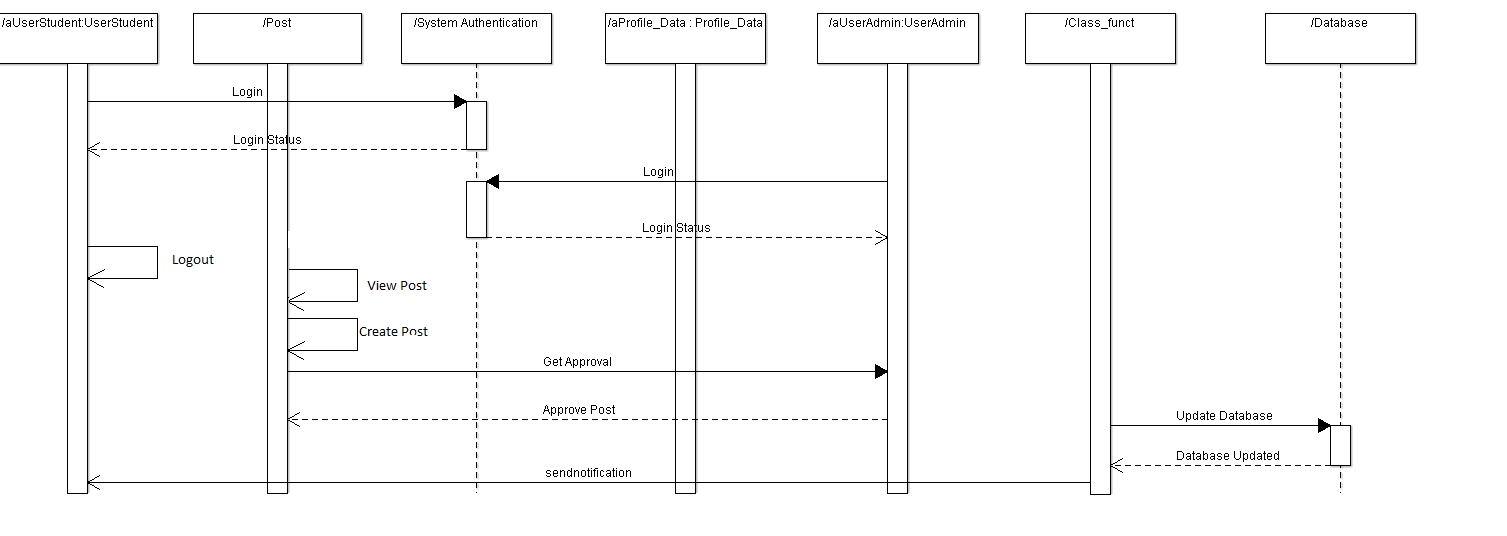
**2.2 Structure and relationships**

**Class Diagram : -**

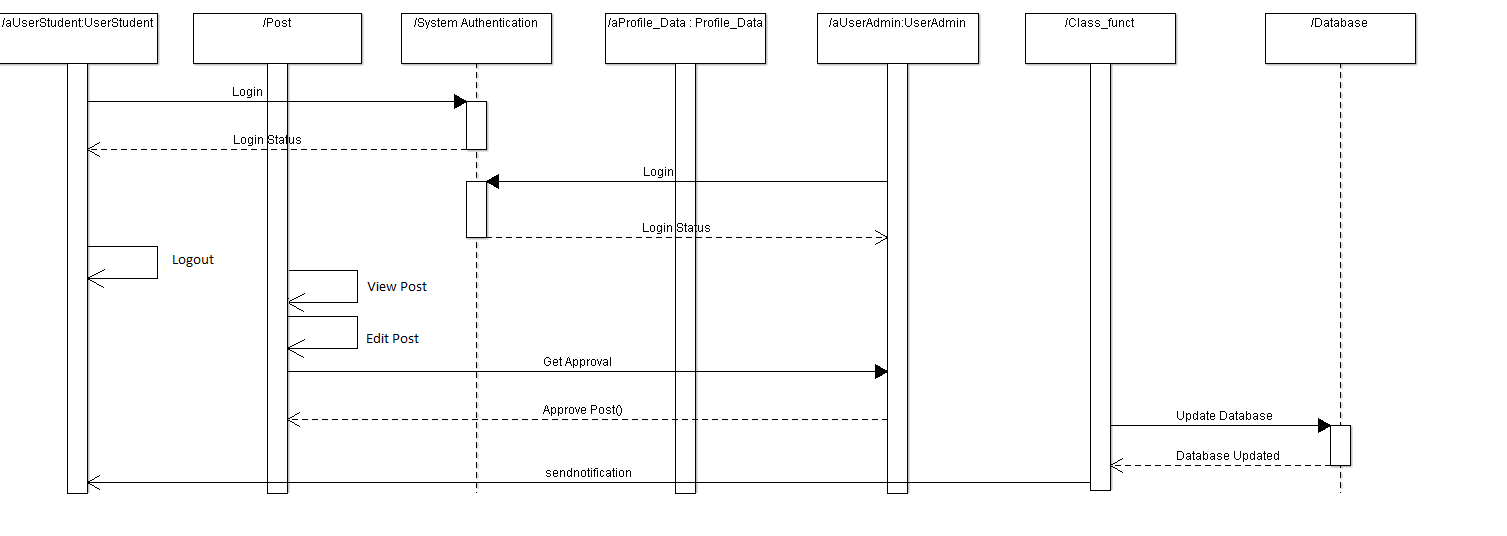
****

**Sequence Diagram :-**

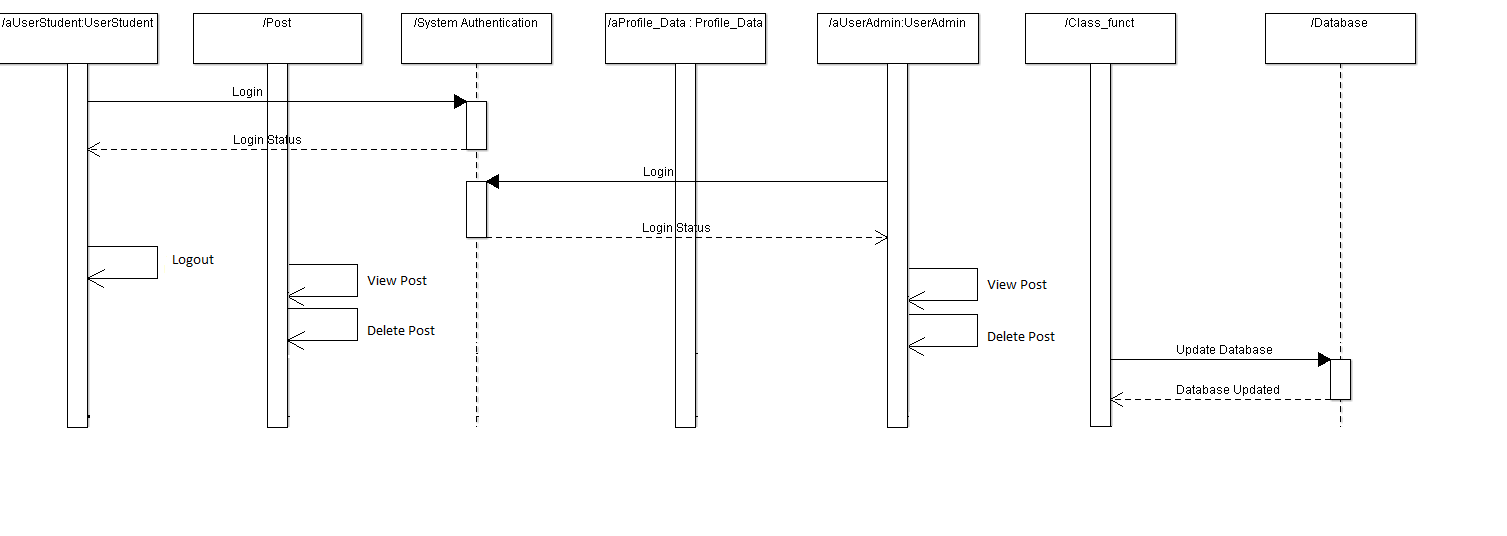
**For Post Creation Functionality**

****

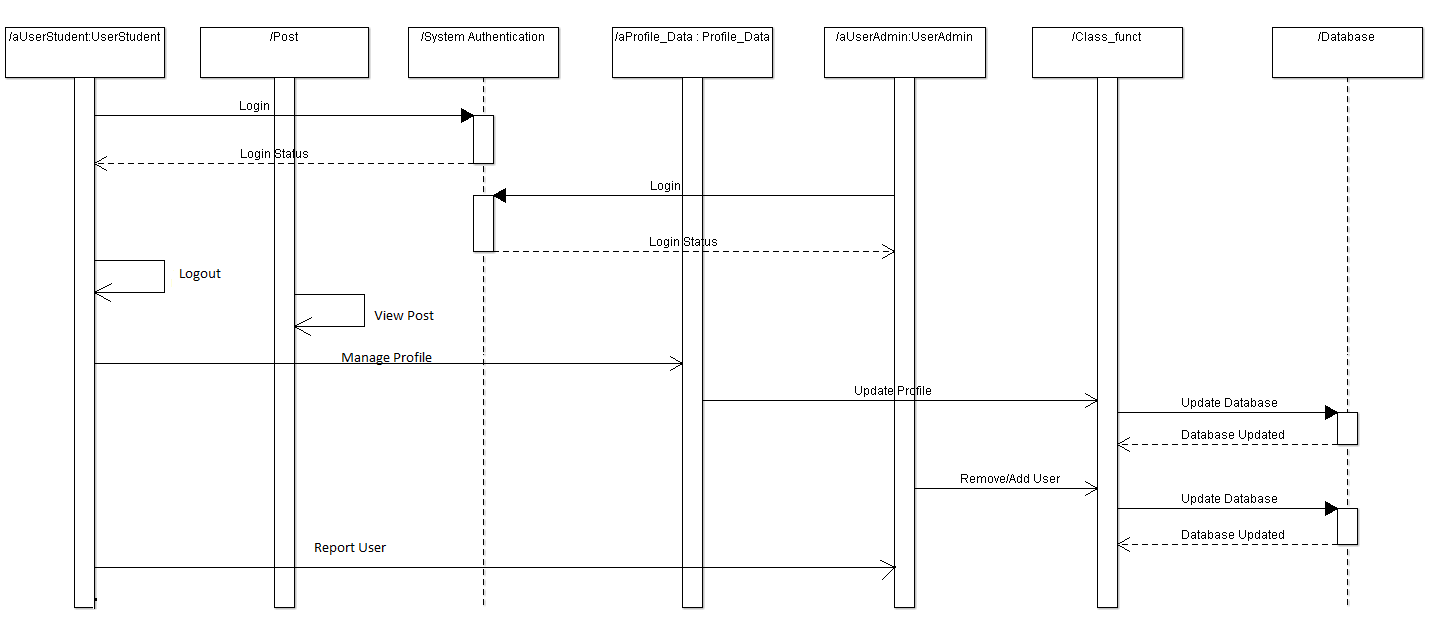
**For Post Edition functionality**

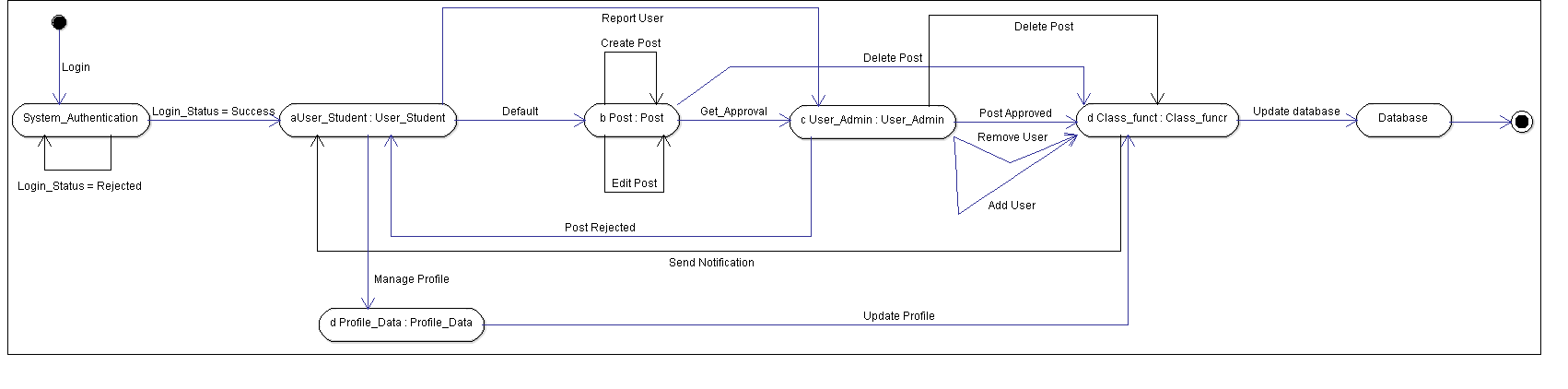
****

**For Post Deletion functionality**

****

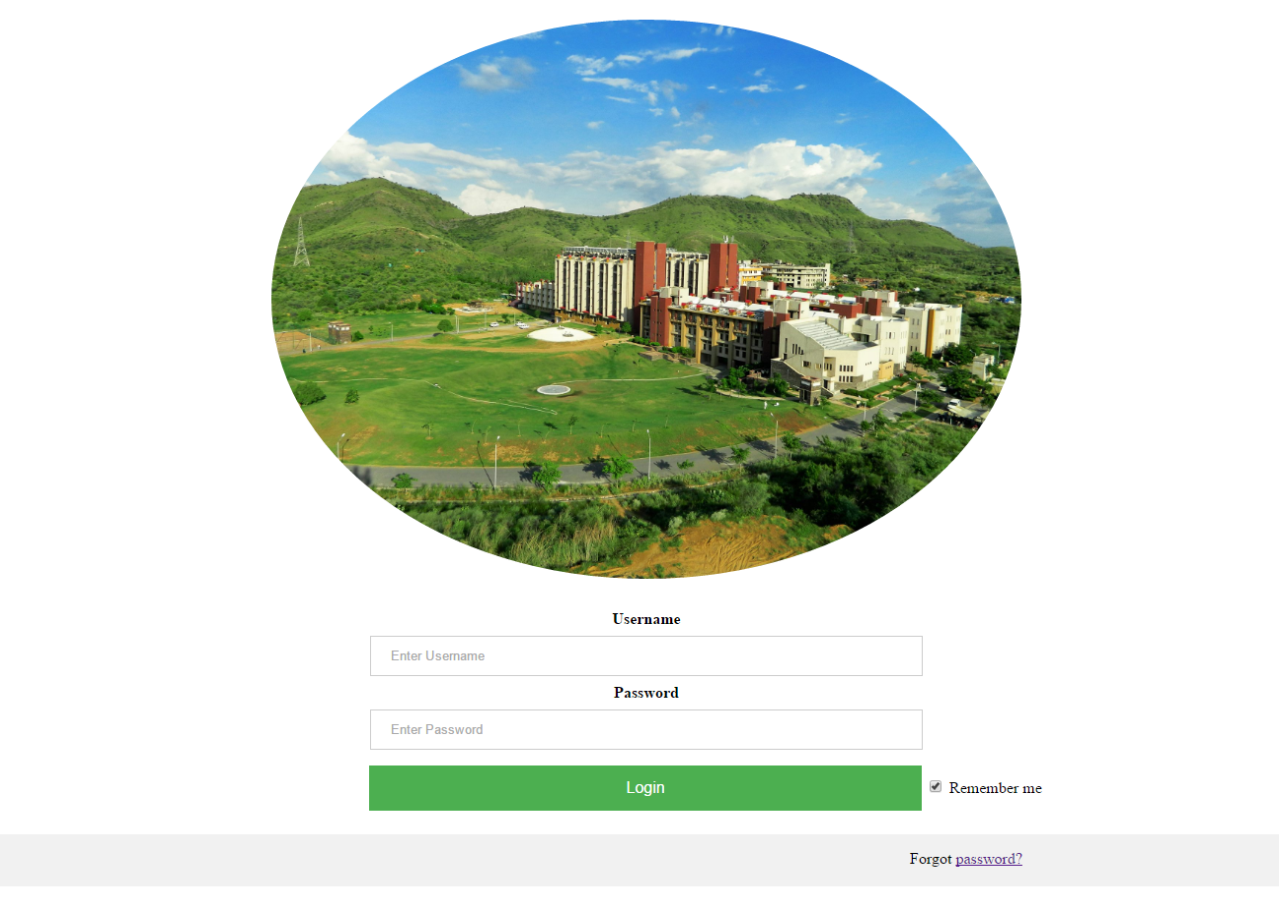
**For Manage Profile functionality**

****

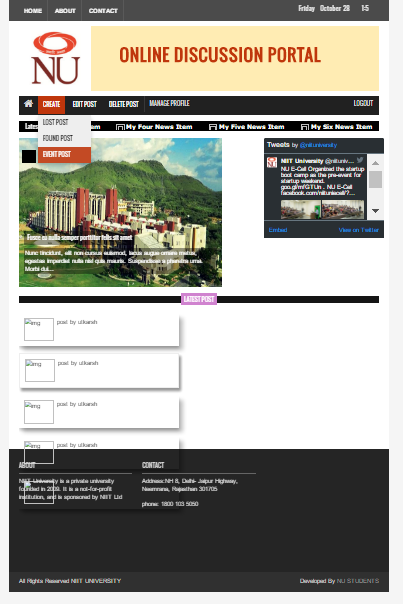
**State Diagram : -**

**2.3 User interface issues**

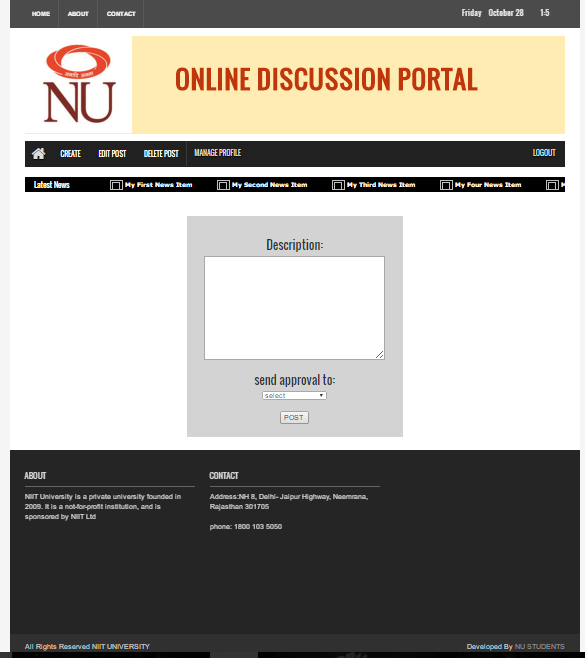
**Login Page:** This is the first page that appears on entering the website. It asks user his credentials for logging in into his/her account.

****

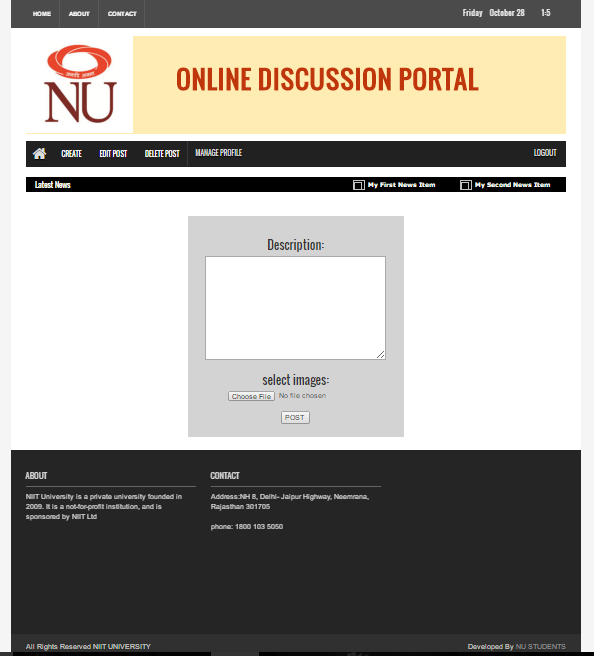
**Landing Page**: As soon as the user gets authentication, he/she is redirected to this page. He/she can see all other posts on the wall.

****

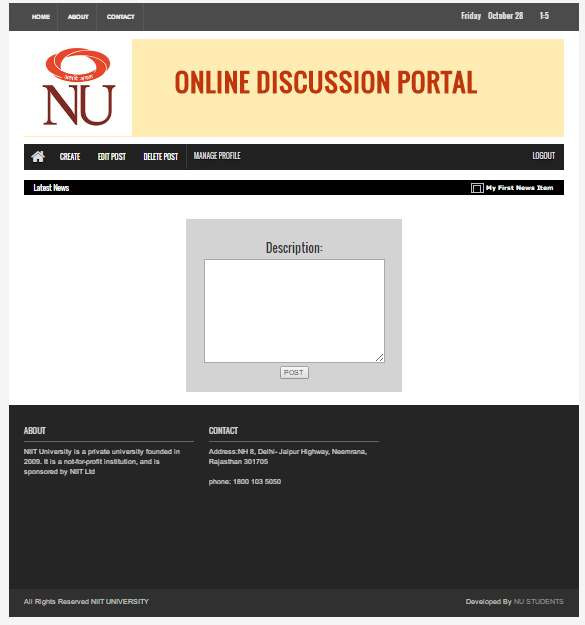
**Create Event-post Tab:** This is how the page looks when you want to create an event and switch to the event creating tab.

****

**Found-post Tab:** This is how the page looks when you want to create a post regarding a found item and switch to the found-post tab. Image is necessary to post.

****

**Lost-post Tab:** This is how the page looks when you want to create a post regarding a lost item and switch to the lost-post tab. Image is not necessary.

****

**INTERFACE ISSUES**

* We had maintain the formality of the design since its being used for a university purpose
* The design would not be fancy.
* We had to keep in mind all the states the website would achieve while creating.
* Also we tried adding some extra feed so that others can also view it.

**3. Detailed description of components**

**3.1 Component template description**

**Description of Class Diagram : -**

**CLASS – User**

This class contains all the Methods and variables related to login and authentication of User for the Web app.

**Variables**

User\_id: Type string, this is the user id entered by the user to login to his/her account.

String: Type Password, this is the string entered by user with username to login to his/her account.

**Methods**

Login (User\_id, Password): Return Type Void, arguments are string and password and this Method takes the user credentials and compare with database to login to user account.

Logout ( ): Return type Void, Logs out the user from the web session.

**CLASS – User\_Student**

This class is a sub class of the class User and contains only actions and methods related to student user.

**Variables**

User\_type: Type int, this specifies whether the user is a admin/student user. 0 stands for usual or student user.

**Methods**

Manage\_profile (name, User\_id): Return Type Void, this Method is called when a user wants to update his profile data.

report\_user ( user\_id1,name1,user\_id2,name): Return Type Void, this Method is called when a user wants to report on another user and makes a request to admin.

get\_approval (User\_id, Post\_id): Return Type Void, this Method is called when a user tries to create a post or edit it and makes a request to admin.

**CLASS – User\_Admin**

This class is a sub class of the class User and contains only actions and methods related to admin user.

**Variables**

User\_type: Type int, this specifies whether the user is a admin/student. 1 stands for admin user.

**Methods**

approve\_post (User\_id, post\_id): Return Type void, This Method is called when the admin wants to approve the requested post (create/edit).

reject\_post (User\_id, post\_id): Return Type void, This Method is called when the admin wants to Reject the requested post (create/edit).

remove\_user (User\_id): Return Type Void, This Method is called when admin wants to delete/terminate a user account.

add\_user (User\_id): Return Type Void, This Method is called when admin wants to create/add a new user account.

**CLASS – Post**

This class contains all attributes i.e. variables and methods related to posts and it is contained by user.

**Variables**

Description: Type String, This consists of the main content of a post created by a student/admin user.

Date: Type Date, This consists the time and date of the event created as post by the user.

Venue: Type string, this string contains the name of the venue an event is being organised in the post created by a user.

Image: Type var\_char, this variable will contain the image the user wants to add an image with the post.

Post\_dept: Type string, This string contain the department the post that needs to be approved before getting posted and only reaches that specific department.

Post\_id: Type String, Each post has its own post id and used when sent for approval and referral.

**Methods**

create\_post (description, date, venue, image, post\_dept, User\_id): Return Type Void, This Method is called when a user wants to create a post (lost post/event post/found post). It adds a post\_id to a newly created post.

edit\_post (description, date, venue, image, post\_dept, User\_id, post\_id): Return Type Void, This Method is called when a user wants to edit a post (lost post/event post/found post). This sends an approval request to the specified department.

delete\_post (Post\_id): Return Type String, this Method is called when a user tries to delete a post, and it returns a string confirming the deletion of the post.

view\_post (): Return Type void, This Method allows a user to see the posts created by other users on their wall.

**CLASS – Profile\_Data**

This class contains all variables related to the information/data related to the profile of the account holder and methods for updating them

**Variables**

Name: Type String, Contains the name of the account holder.

Enrollment\_no: Type String, Contains the unique enrolment number of the account holder.

Gender: Type String, contains what gender type the user belongs to.

DOB: Type date, stores the Date of birth of the account holder.

Phone\_number: Type long, stores the phone number of the account holder.

Email\_id: Type String, This variable contains the email-id of the account holder.

Address: Type String, This string contains the address of the account holder.

**Methods**

update\_profile ( ) : Return Type Void, This Method is called when user tries to update his profile data.

**CLASS – System\_authenticate**

This class specifically contain the variables and methods for authentication of user during the login.

**Variables**

Id: Type String, this variable consists the user Id which is entered by the user when he tries to login to his/her account.

key: Type String, This is entered by the user along with username to authenticate login into his/her account.

**Methods**

Authenticate (Id, key): Return type Void, This Method is called when user clicks the login button after he/she enters his/her Id with Password.

**Description of Sequence Diagrams : -**

* **To login/logout the account:**

1. Student user sends the login request to system authentication.
2. System authenticator checks the credentials against the database and sends the status to the user whether it is a valid credential or not.
3. Similarly, Admin user sends the login request to system authentication and system authenticator sends the status to admin.
4. User student and User admin initiates the logout action and the user gets logged out of the web site.

* **To create a post:**

1. User on our web portal creates a post which will consist of the description of the post, date and name of the student.
2. The post will be sent for approval to User admin who will check the validity of the post and will approve or reject the post accordingly.
3. After the approval the database containing the posts will be updated thereafter.
4. A notification to the User Student will be sent about the post after the database gets updated.

* **To edit a post:**

1. If a user on our web portal wants to edit a post, he will access his post on our web portal and will edit the post
2. The edited post will be sent for approval to User admin who will check again the validity of the post and will approve or reject the post accordingly.
3. After the approval the database containing the posts will be updated thereafter.
4. A notification to the User Student will be sent about the post after the database gets updated.

* **To delete a post:**

1. User student will execute the delete post action on the web interface and the post will be deleted instantly.
2. The database containing the posts will be updated thereafter.

* **To manage the profile:**

1. User will request to update the profile in the profile data.
2. The profile data will send the request to update the database.
3. Accordingly the database will be updated after valid confirmation.

* **To remove or add a user:**

1. If a user admin wants to add or remove the user, he will directly access the database and will update it accordingly.
2. The database containing the information about the user will be updated thereafter.

* **To report a user:**

1. When there is misuse of the web portal a user student can report another user’s activity to User admin.
2. The information will be further sent to authority in charge and proper action will be taken thereafter.

**Description of State Diagram : -**

**State diagram:**

* User will login into the portal by using given credentials.
* If these entries are matching with the data in the database then login is successful.
* If entries doesn’t match then he will be redirected again to login page.
* User on default can view post on the entry page.
* On the entry page user can create post, edit post, delete post and manage profile.
* User will send approval for create post and edit post to the admin.
* If admin approves for the create post and edit post, then database will be updated, post will be posted on the view post and notification will be sent to all the users.
* If post is rejected by the admin then user will get the message about the rejection.
* When user update profile then data will be updated in the database.
* Post can be deleted by the user without any approval of the admin and database will be updated.
* Admin can also delete post if he is having any issues and database will be updated.
* Admin can add user and remove user .This changes will be updated in the database.
* If any user find any abusive post user can report to the admin.
* The details of the user who posted that abusive post will be sent to admin from the database .

**4.0 Design decisions and tradeoffs**

A table format that gives your team a concise way to document, analyze, and communicate the alternatives you are considering for scope and features. The trade-off table helps the team capture the critical factors associated with each design alternative under consideration, and allows them to compare the impact of various combinations on the project's cost, schedule, resources, and risk.

Alignment on such tradeoffs is critical to keeping any remote team members on track with the goals of the project, and able to make sound design decisions etc. even though they are not "on site" with the rest of the team every day.

|  |  |  |
| --- | --- | --- |
| **FACTORS** | **ALTERNATIVE 1** | **ALTERNATIVE 2** |
| **Customer needs addressed** | * our software is familiar to the users * And it is flexible too. There is scope to extend features according to the users. * Best fit for domestic sites. | * Our mailing system is quite tedious and it takes lot of time to address customer needs .It also needs chain of approvals. * Reducing the time consumption and making the things ease to the users. |
| **Critical features** | Meet customer expectations precisely (Intuitive)Easy to useIntegratedMedia support  * High performance | * Integrated * Powerful administration * Securities * Traffic |
| **Technical Risks** | securities issues because our database is dynamic  There is possibility of congestion due to lot of functionalities. | Actually technically there is no technical risks but not flexible for use. |
| **Maintainability** | Very good  It is easily maintainable because control  Is in the hands of single administrator. | Good  In this anyone can post anything this will create congestion and will take time to post. |
| **Target Customer Acceptance** | Acceptable  We are aggregating the sequence of activities in a single portal with reducing the time consumption and accuracy. | Acceptable  Currently they are using this system for communication .this is formal way of communication and time consumption and suitable for informal way. |
| **Production lead time** | In this all the activities are in the single portal which will reduce the lead time | In this system lead time is maximum because most of sequence of activities with approval of respective department head. |
| **Project Schedule** | Our project design and front end part is completed by 26th October and portal will be functional by 21 November. | Thisis not applicablebecause this is already built by google. |
| **Project cost** | This is student based project and still not evaluated. | This is public based application and no need of any cost except internet cost to connect to email. |
| **Resources needed and project budget** | 5 students studying computer science  web server to host our portal  And dynamic database to automatically update the data. | Integrated email system with internet connection to all the users. |
| **Notes and**  **Recommendations** | Strong documentation and manual support.  Although product fits most of our specs well, the risk associated with working with them as a vendor (non-responsiveness etc.) is a deal-breaker for us | This is google product and strongly recommended by the users but this is not suitable for our environment in the university because of time consuming.  They are not the best fit for all requirements, but they have been responsive in pre-contract investigation work |

**6.0 Pseudocode for components**

class User

{

String User\_id;

Password String;

void login(User\_id,Password)

{

input:User\_id;

input:Password;

if(call authenticate(User\_id,password)==1)

call view\_post();

else

print "Incorrect credentials";

}

void logout()

{

break;//exit system

}

}

class User\_Student extends User

{

int User\_type;

void manage\_profile(Name,User\_id)

{

call update\_profile(Name,User\_id);

}

void report\_user(Name,User\_id)

{

//sends name and user id of another user to admin

}

void get\_approval(User\_id,Post\_id)

{

call approve\_post();

}

}

class User\_Admin extends User

{

int User\_type;

String approve\_post(User\_id,post\_id);

{

get input(validity of the post)

if(validity=="valid")

update\_database(obj\_post);//add post to database

else

print(call reject\_post());

}

String reject\_post((User\_id,post\_id);

{

return "Post rejected"

}

void remove\_user(User\_id)

{

call update\_database(User\_id)

}

void add\_user()

{

call update\_database();

}

}

class post contained by User

{

String Description;

date Date;

String Venue;

var\_char image;

String post\_dept;

String post\_id;

void create\_post(description,date,venue,image,post\_dept,User\_id);

{

input:description,date,venue,image,post\_dept,User\_id

approve\_post()

}

void edit\_post(description,date,venue,image,post\_dept,User\_id,post\_id)

{

input:description,date,venue,image,post\_dept,User\_id;

approve\_post();

}

String delete\_post(Post\_id)

{

update\_database(post\_id);//delete the post from database

}

void view\_post()

{

print contents of post object

}

}

class Profile\_Data

{

String Name;

String Enrollment\_no;

String Gender;

date DOB;

long phone\_number;

String email\_id;

String address;

void update\_profile(Name,User\_id);

{

call update\_database(Name,User\_id)

}

}

class System\_authenticate

{

String Id;

String key;

int authenticate(User\_id,Key)

{

if(User\_id==Id)&&(Key==Password)

return 1;

else

return 0;

}

}

class funct

{

void update\_database()//parametrized function

{

enter data into database

}

void send\_notification()

{

print "New post"

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*