Final Report on

Project Ovijog

Submitted to,

SPL II Evaluation Committee 2020-2021
Bachelor of Science in Software Engineering
Institute of Information Technology
Noakhali Science and Technology University

Submitted By,

Armanur Rashid
armanur2514@student.nstu.edu.bd

Arnab Dey
arnab2514@student.nstu.edu.bd

Nayeem Khan
nayeem2514@student.nstu.edu.bd

Supervised By,

Md. Rafid Mostafiz
rafid.iit@nstu.edu.bd

Submission Date: September 20, 2022

Final Report on *Project Ovijog*

By,

Armanur Rashid	Arnab Dey	Nayeem Khan
ASH1925013M	ASH1925024M	ASH1925027M
Year 3 Term 1	Year 3 Term 1	Year 3 Term 1
armanur2514@student.nstu.edu.bd	arnab2514@student.nstu.edu.bd	nayeem2514@student.nstu.edu.bd

Approved By,

Md. Rafid Mostafiz
Lecturer
Institute of Information Technology
Noakhali Science and Technology University
rafid.iit@nstu.edu.bd

Table of Contents

Table of Figure

Table of Figure

Project Description

Introduction

Noakhali Science and Technology is one of the most renowned universities. There are many teachers and students in our university. Unfortunately, our teachers and students face a variety of problems in campus. For example, students suffer a lot due to lack of transport. There is no canteen in our campus. Many students have problems with their accommodation, drinking water and other problems. Besides, the teachers also face many problems including food, housing problems, lack of adequate buses for transportation. Both the teacher and the student face different problems. But it is a matter of regret that there is no internal system in our university where anyone can talk about their problems without revealing their identity. They posted their problems on the Facebook group. But in Facebook group, it is not clear how many students are facing the same problem. The profile of the person who post, is also visible to everyone. Trending problem, solved problem, unsolved problems are not seen. We see teachers and students wanting a platform where they can complain about their problems. This is one of the main reasons for our motivation.

Target Users

There are 3 target users in our application. The target users for our system are students, teachers, varsity authority. All users' priorities are the same. Everyone will come forward with everyone's problem.

Students

One of the main users of our system is the student. Our university has the largest number of students. So, the main target user of our application is student.

Teachers

Another target user of our system is the teachers. There are many teachers in our university. They were not getting any platform to voice their various problems. So, they are also our target users.

Academic Officials

Members of university authorities can also use our application. What problems are students and teachers are facing, the authority is not aware of which problems are occurring the most. The needed an application through which they could identify the most difficult problems. Which they will know from our application.

Requirements

Functional

1. User Registration and Login.
2. Create a new post
3. Edit post
4. Delete post.
5. Showing top solved, unsolved problem
6. Search posted problems by tags
7. Comment on others post
8. User can up or down vote
9. Filtering problems by category
10. Report against fake posts
11. Mark as Solved
12. Notification
13. Tag suggestion
14. Admin Control

Non-Functional

1. Users' password will be protected
2. Search, Post, and Filter with less buffering and load faster.
3. Make the code maintainable.
4. Privacy Requirements
5. The system provides security strategies.

Models, Tools and Resources

Models

In our SPL-II project we used evolutionary model. Using this model, it made our job a lot easier. Since the code is tested at the end of each cycle, at the end of the project we able to create an error free project. Evolutionary model is a combination of iterative and incremental approach to software development. Evolutionary model is commonly used when the client wants to start using the core features instead of waiting for the full project. Evolutionary model is also used in object-oriented software development because the system can be easily portioned into units in terms of object. The Evolutionary development model divides the development cycle into smaller, incremental waterfall models in which users are able to get access to the product at the end of each cycle. This time also taking experience from the previous year project, we want to implement evolutionary model. One of the main reasons we want to use this model is, it reduces the error because the core modules get tested thoroughly. And a user gets a chance to experiment partially developed system.

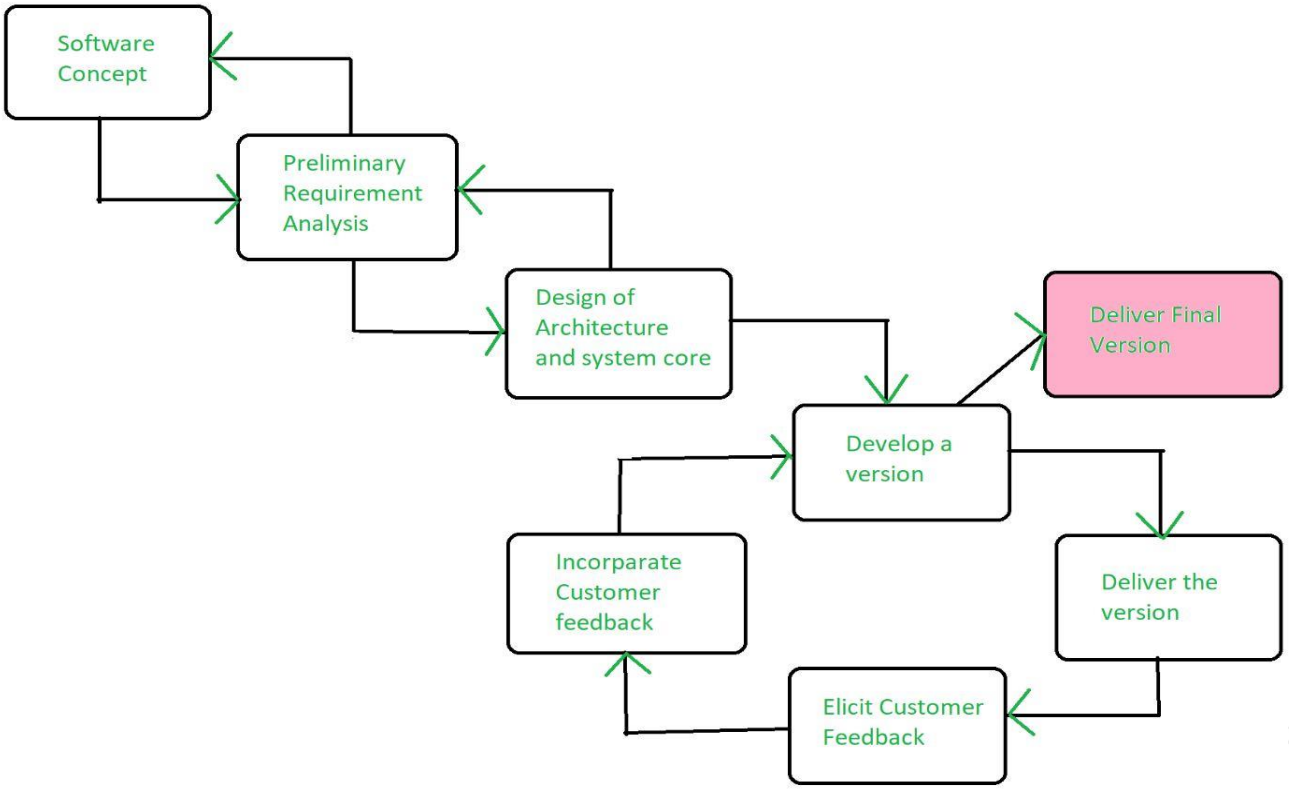

Figure 1: Evolutionary Model

Tools

Text editor	:	Visual Studio Code
Local Server	:	Xampp Control Panel
RDBMS	:	MySQL
Technologies	:	HTML, CSS, Bootstrap 5, PHP, JavaScript, SQL

Resources

Learning resources :
1. https://www.w3schools.com/
2. https://getbootstrap.com/
3. https://www.udemy.com/

Booking resources :
1. Software Engineering 9th Edition by Lan Sommerville
2. Requirements Engineering Fundamentals by Klaus Pohl
3. Database System Concepts 6th Edition by Abraham Silberschatz

Online resources :
1. https://github.com
2. https://stackoverflow.com/
3. https://getbootstrap.com/

Paper resources :
1. Semiz, G. & Berger, P. D. (2017). Determining the Factors that Drive Twitter Engagement-Rates. Archives of Business Research,5(2).

Project Information

Project Name	:	Project Ovijog
Team Name	:	Team Connector
Supervisor	:	Rafid Mostafiz Lecturer, Institute of Information Technology Noakhali Science and Technology University rafid.iit@nstu.edu.bd
Team Members	:	Armanur Rashid ASH1925013M Year 3 Term 1 armanur2514@student.nstu.edu.bd Arnab Dey ASH1925024M Year 3 Term 1 arnab2514@student.nstu.edu.bd Nayeem Khan ASH1925027M Year 3 Term 1 nayeem2514@student.nstu.edu.bd

User Guide

User Side

The user side architecture is shown in figure 2. Click a button it will go to the related page. What is on that page and what can be done from the page is shown in tree form. By seeing this, the user will get a complete guide on how to use the entire application. After the architecture everything is pictured with details.

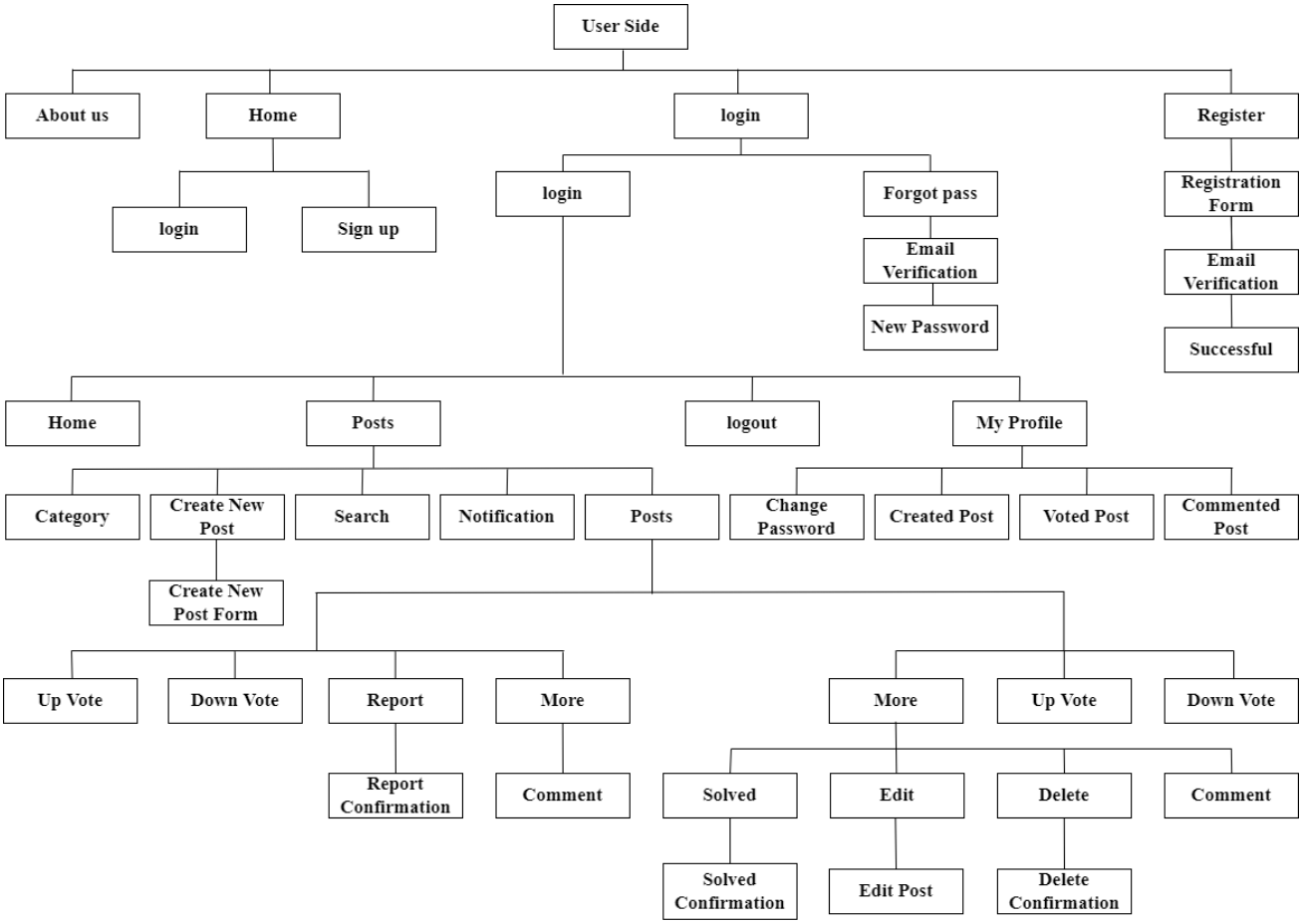

Figure 2: User Side Architecture

Home Page

This is our landing page. Some motivational speeches will show here. There are four links in the navbar. Clicking on the login button will take you to the login page. If you click on register, you will go to the new account opening page. Clicking on About will take you to the About page. There are also two buttons here - signup and login. Figure-3 shows the Home page of our application.

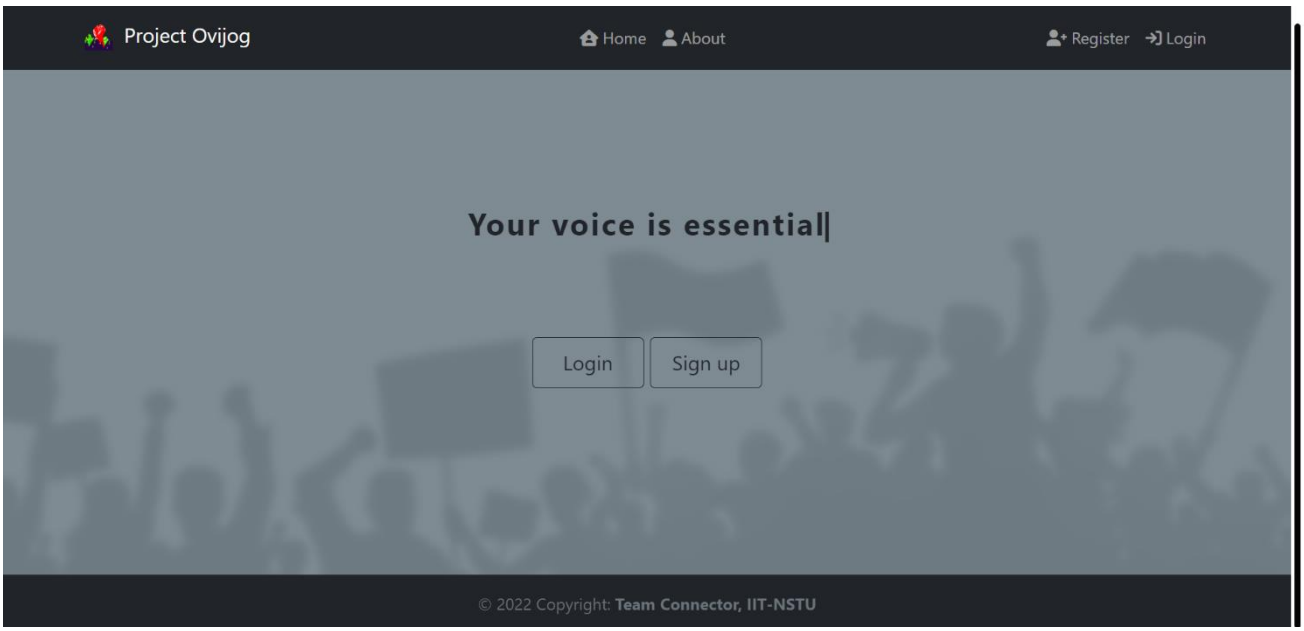

Figure 3: Home Page

Create Account

If you want to create a user account, you have to click Signup from Home Page or Register from Navbar. Buttons are marked in figure 4.

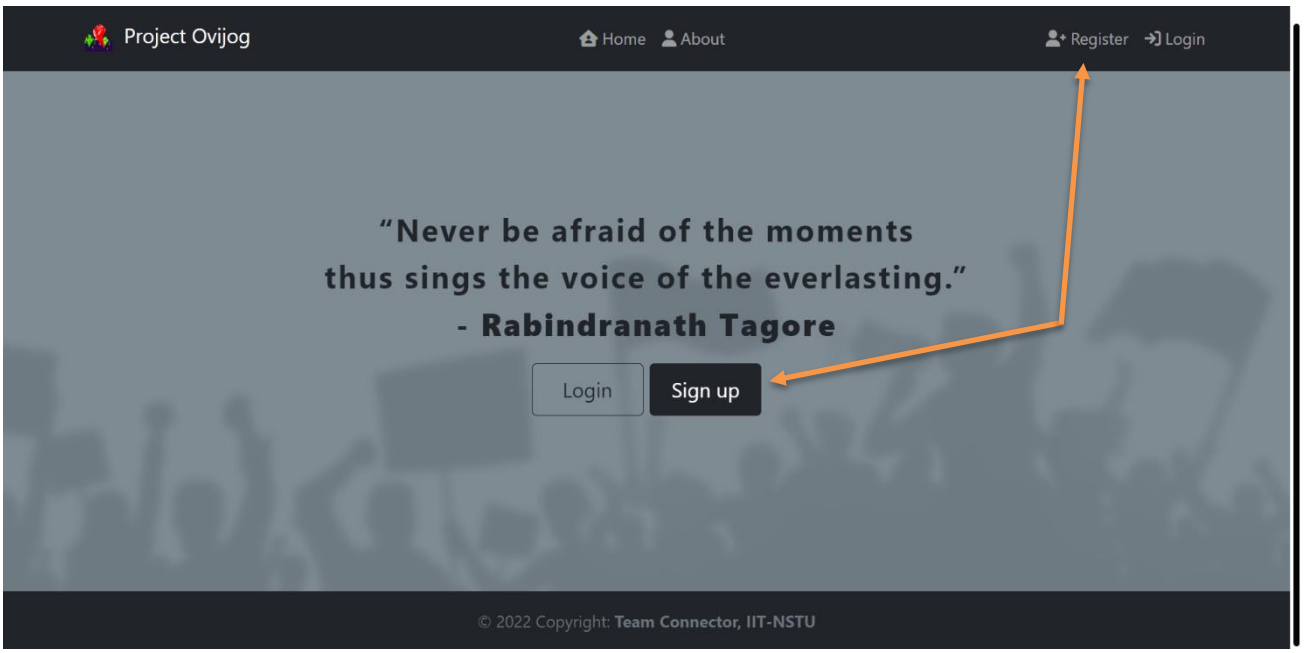

Figure 4: Create Account

Sign up Form

After clicking on register or signup from the home page or navbar, the new account opening page will appear. The user will enter the information required to open the account and click on the Register button.

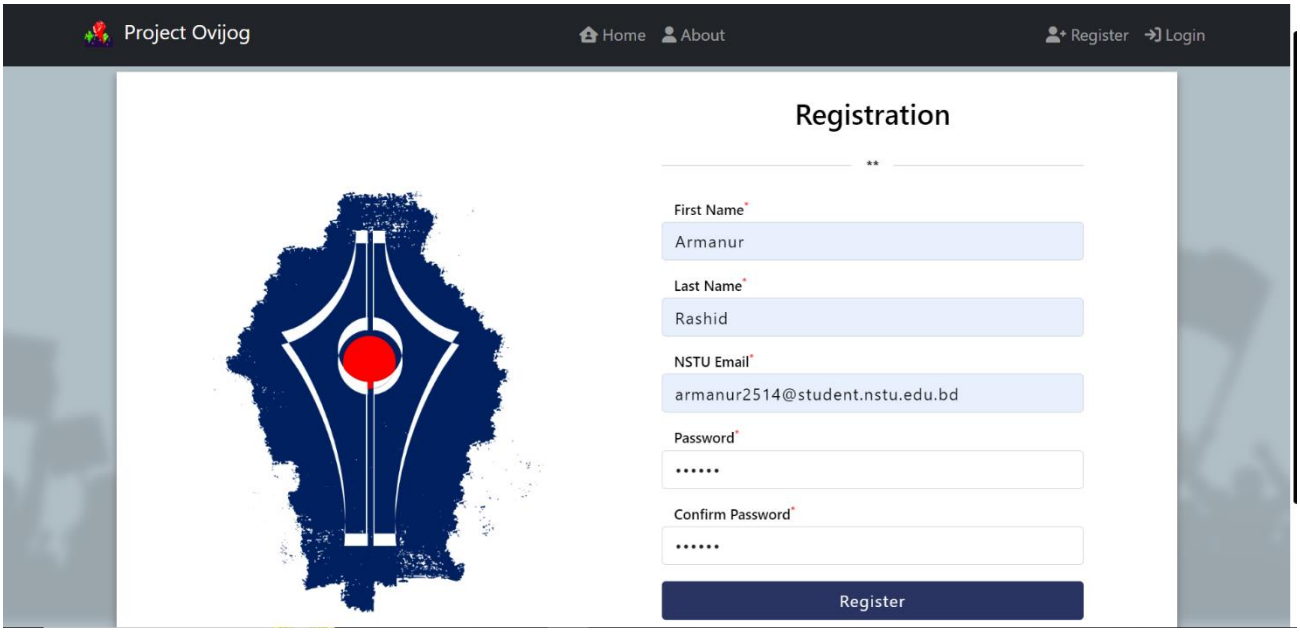

Figure 5: Signup Form

Error Message

If someone gives an email other than Edu Mail, or already has an account. Password must be less than 6 characters. Or if the password and confirm password do not match, an error message will be displayed

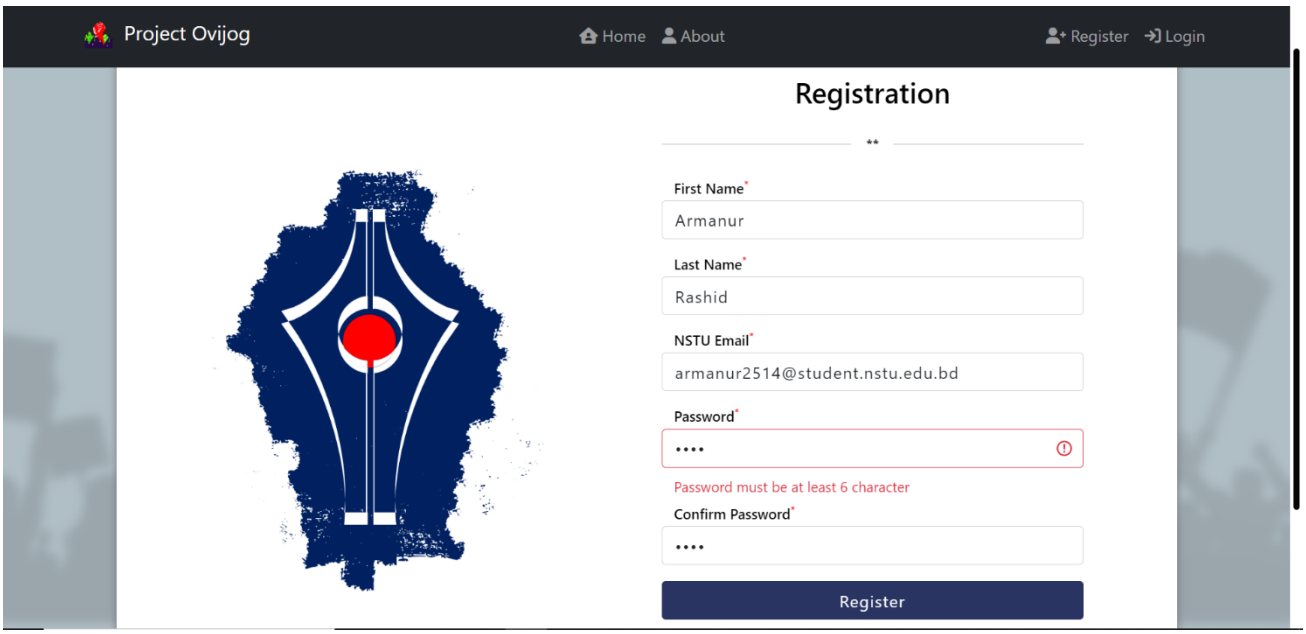

Figure 6: Signup Error Message

Successful Alert

A successful message will be displayed if the user has entered all the information correctly.

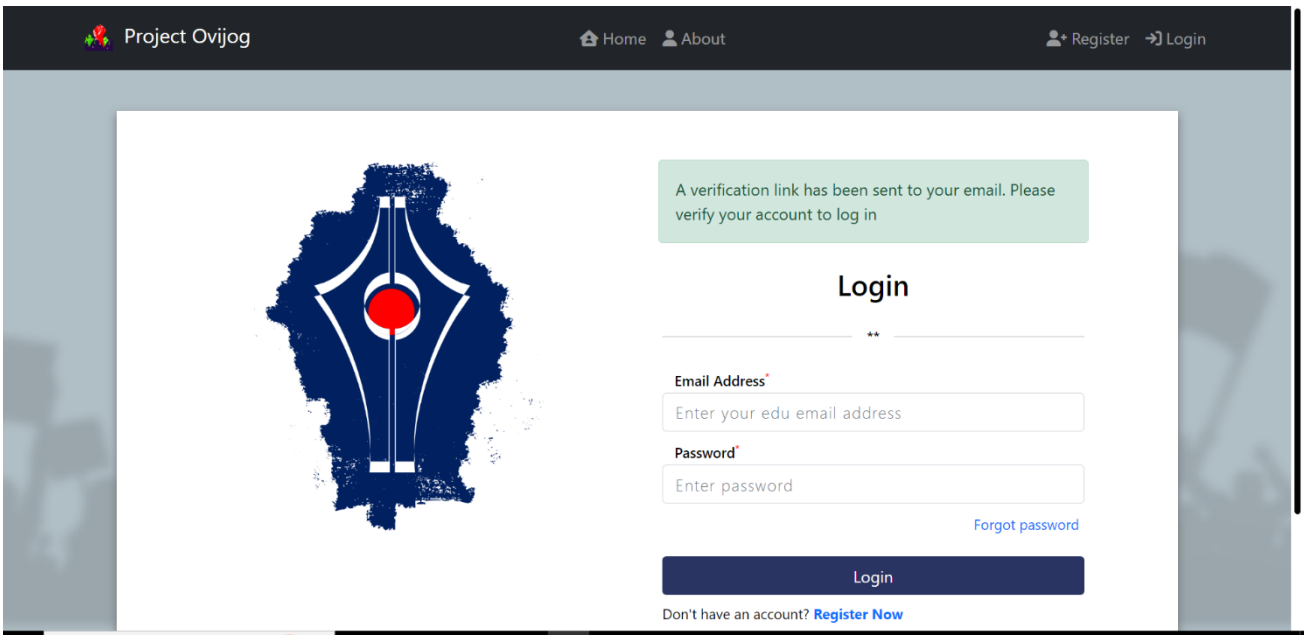

Figure 7: Signup Successful Message

Email Verification

If the user gives all the information correctly, a mail will be sent to the user's mail. There will be a link on which the user will verify his account. Figure-8 shows the mail which is sent to the users email.

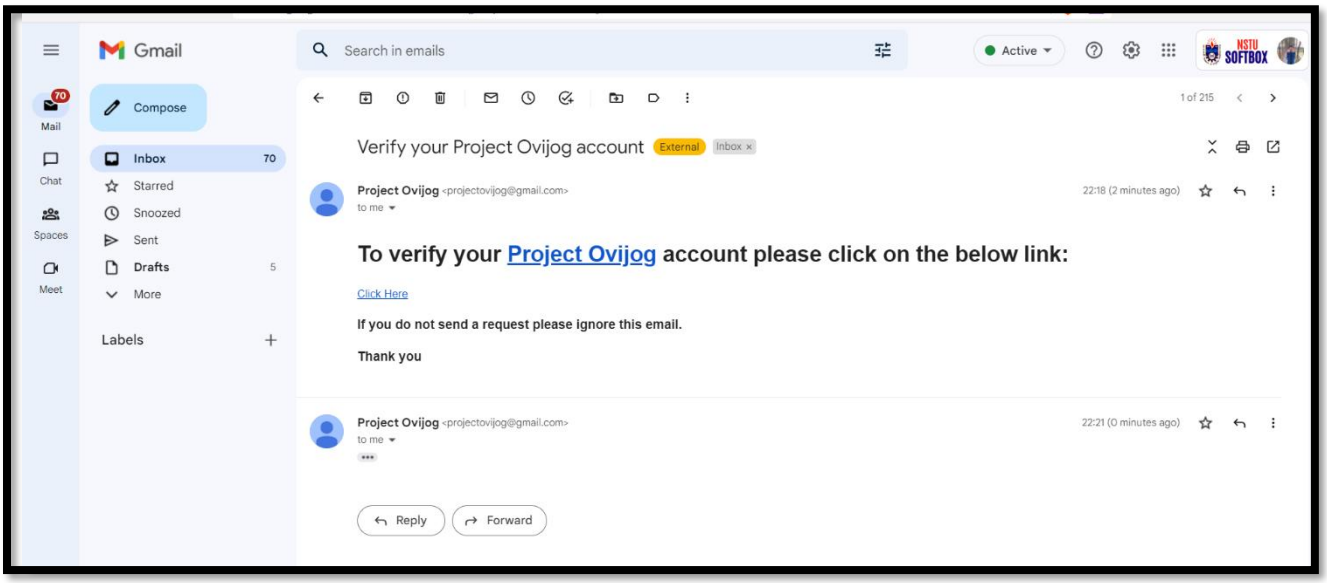

Figure 8: Email Verification for signup

Verification Message

A successful message will be displayed if the user has verified his account.

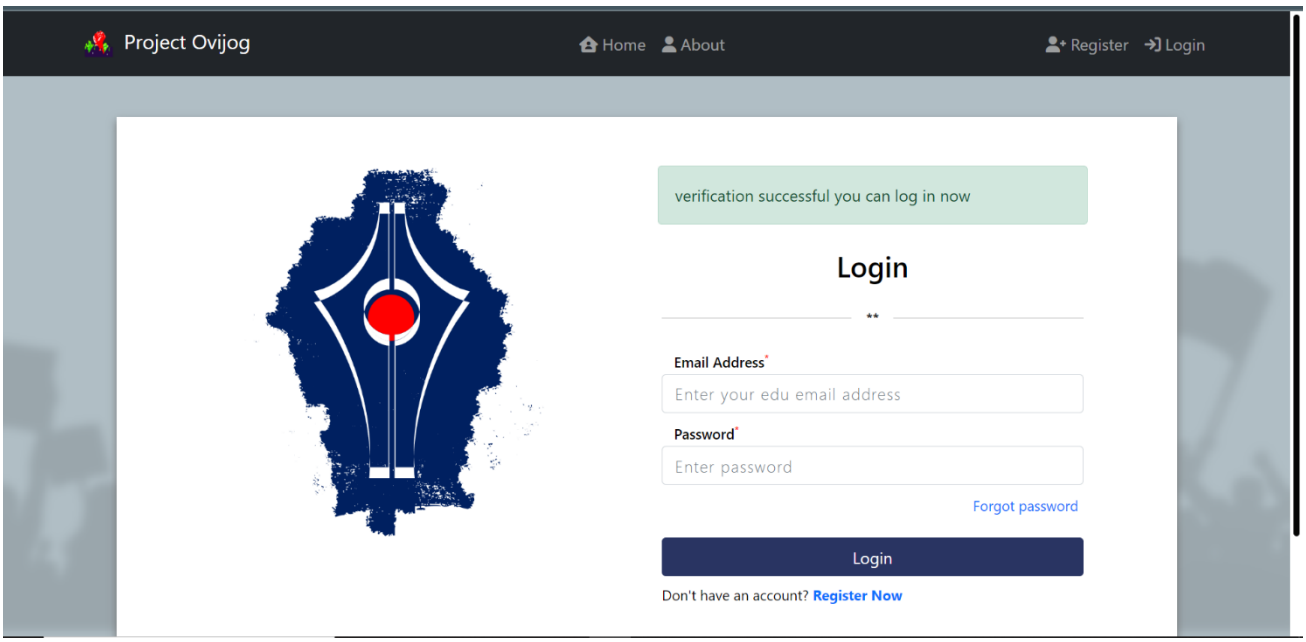

Figure 9: Verification Successful

Login

If you want to login to his account, you have to click Login from Home Page or Login from Navbar. Buttons are marked in figure 10.

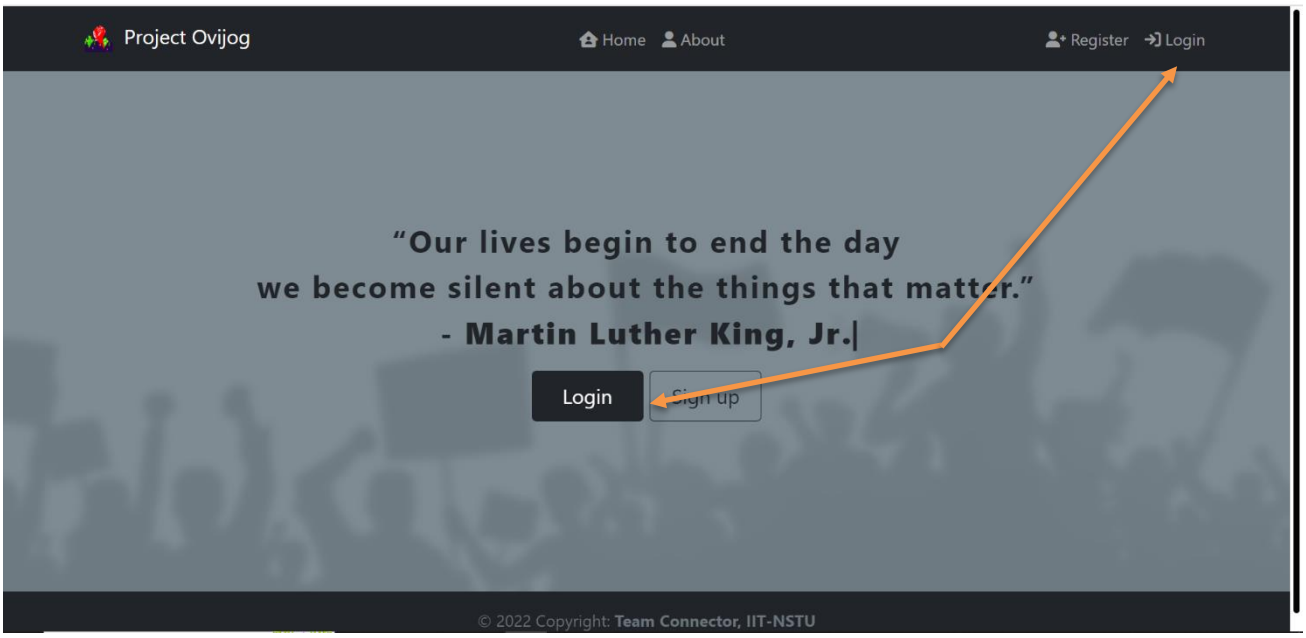

Figure 10: Login Page

Login form

After clicking on login from the home page or navbar, the login page will appear. The user will enter the information required to login his account and click on the Login button.

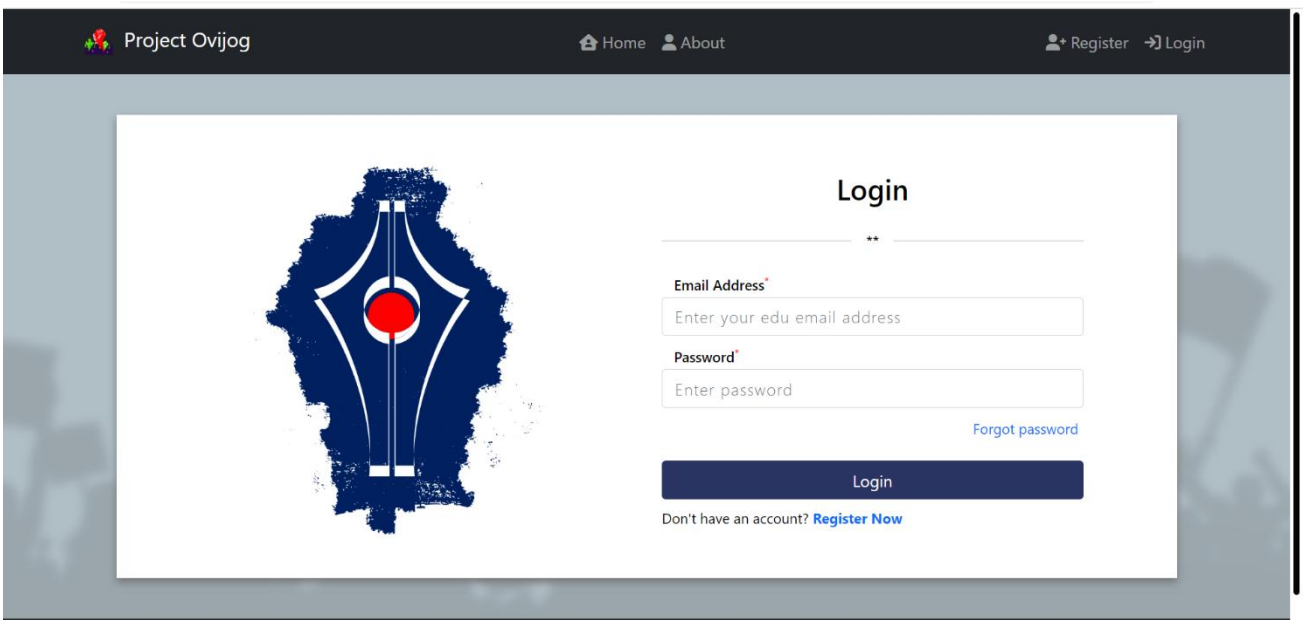

Figure 11: Login Form

Error Message

If user enter wrong email or password then error message will show like the figure 12.

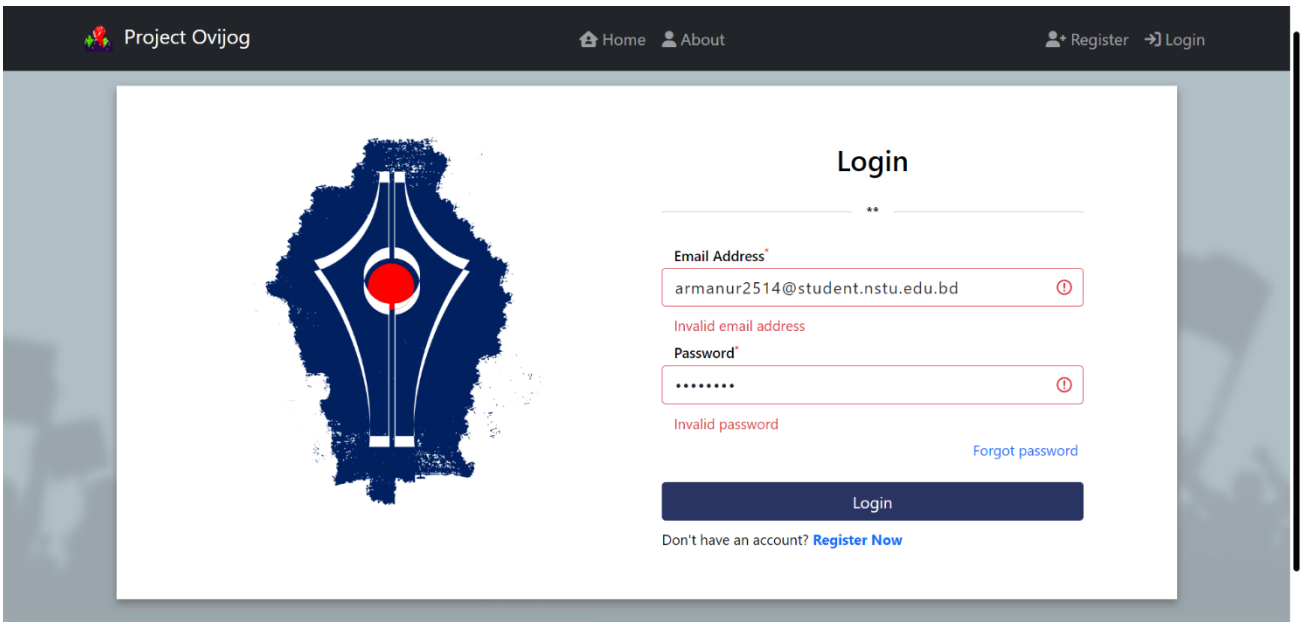

Figure 12: Login Error Message

Index

After logging in, users can see the index page where they can see the overall number of posts, the total number of solved posts, and the number of unsolved posts. The visitor can also view the top five solved and unsolved problems in this section.

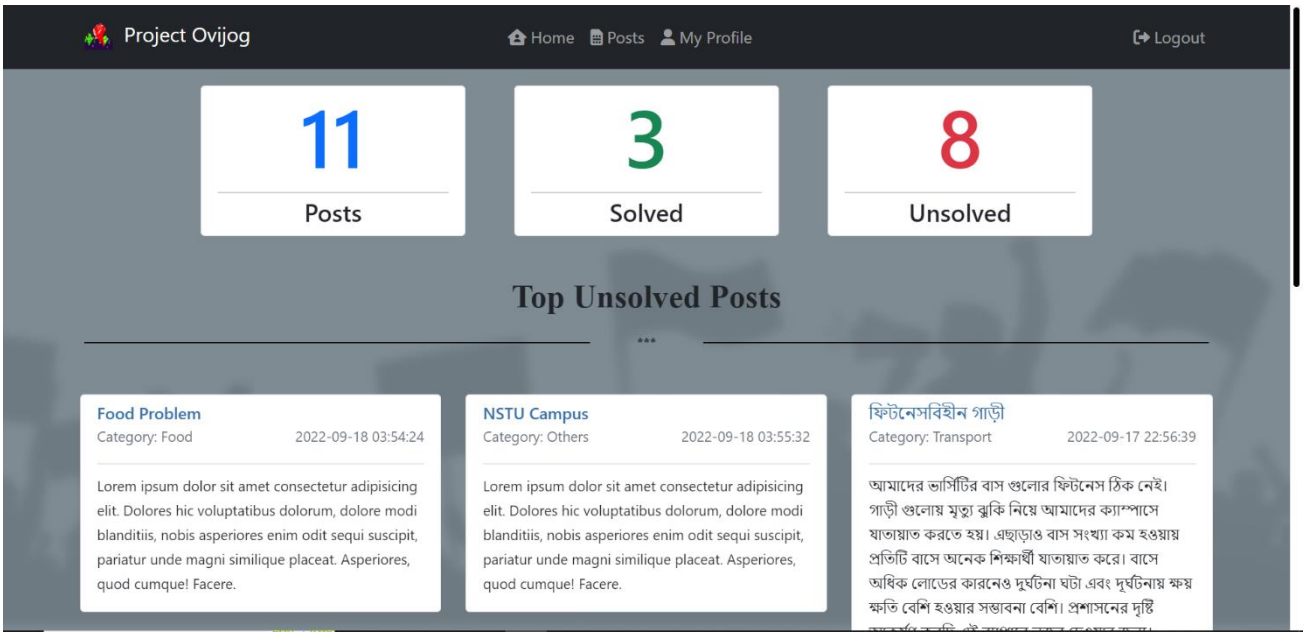

Figure 13: Index Page

Post

Users can view all posts on this page. The navbar will feature a search option here, allowing users to conduct keyword searches. The user can view the post relating to the given keywords given. Post page view is shown in the Figure-14.

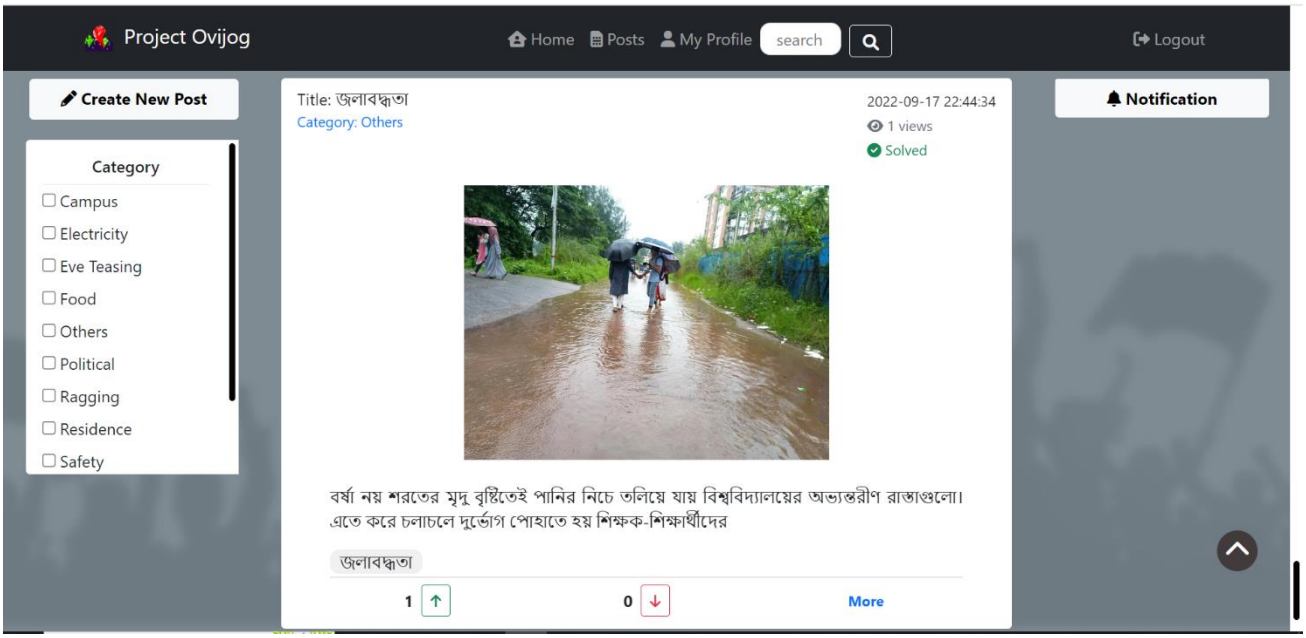

Figure 14: Post Page

View full post

If a user wants to edit, delete, comment or marks as solved in a particular post they have to go to the more option.

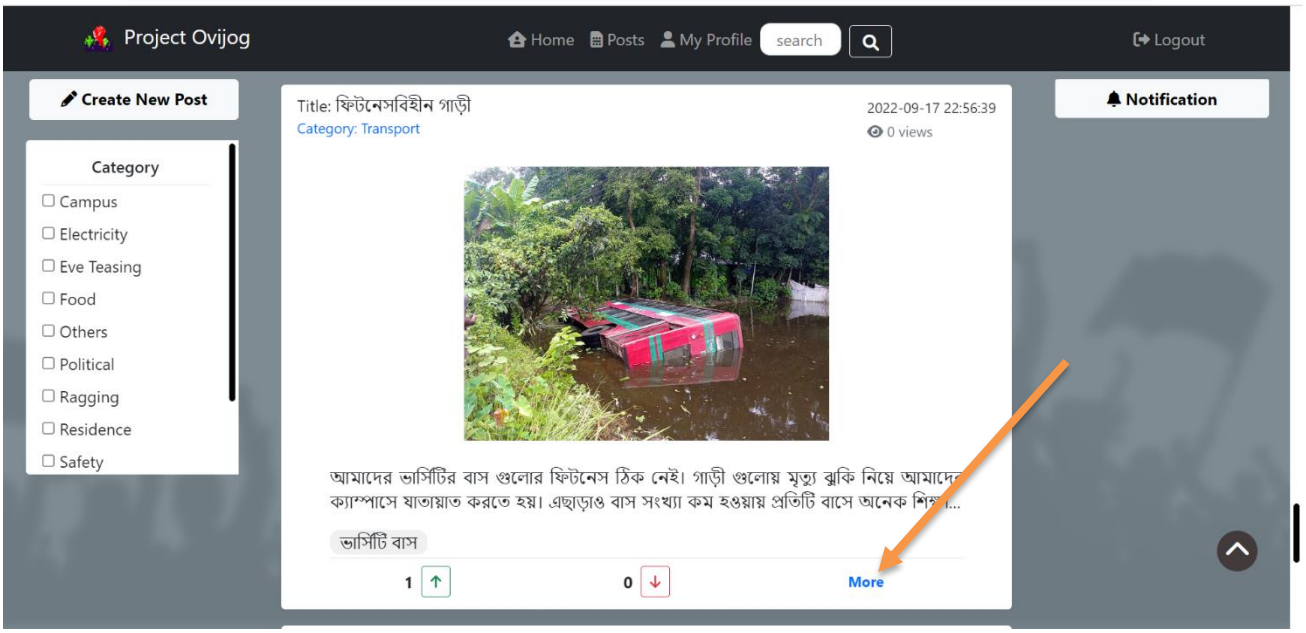

Figure 15: View Full Page

Up / Down vote

There are upvote and downvote buttons on this page. Users will be able to upvote and downvote specific posts. Which is considered to find the trending the post.

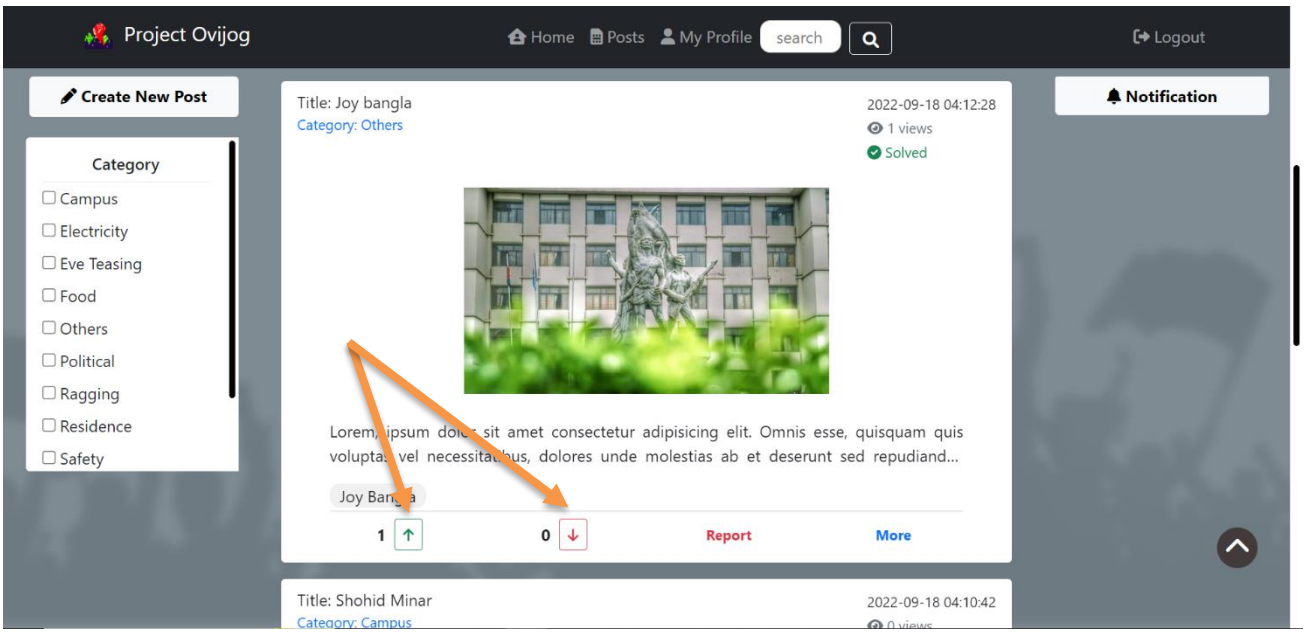

Figure 16: Voting

Create New Post

Users who select the "create new post" option are taken to a page where the required fields for the title, body, and category must be filled in. Tags and photos are not necessary. Create New Post page is shown in the figure-17. The user will have certain predefined categories in the category choice. The user can choose one of the categories that is more suitable for the post. Selecting a category is shown in the figure-18. When using the tag option, if the user types any content that has already been tagged, a recommendation will appear.

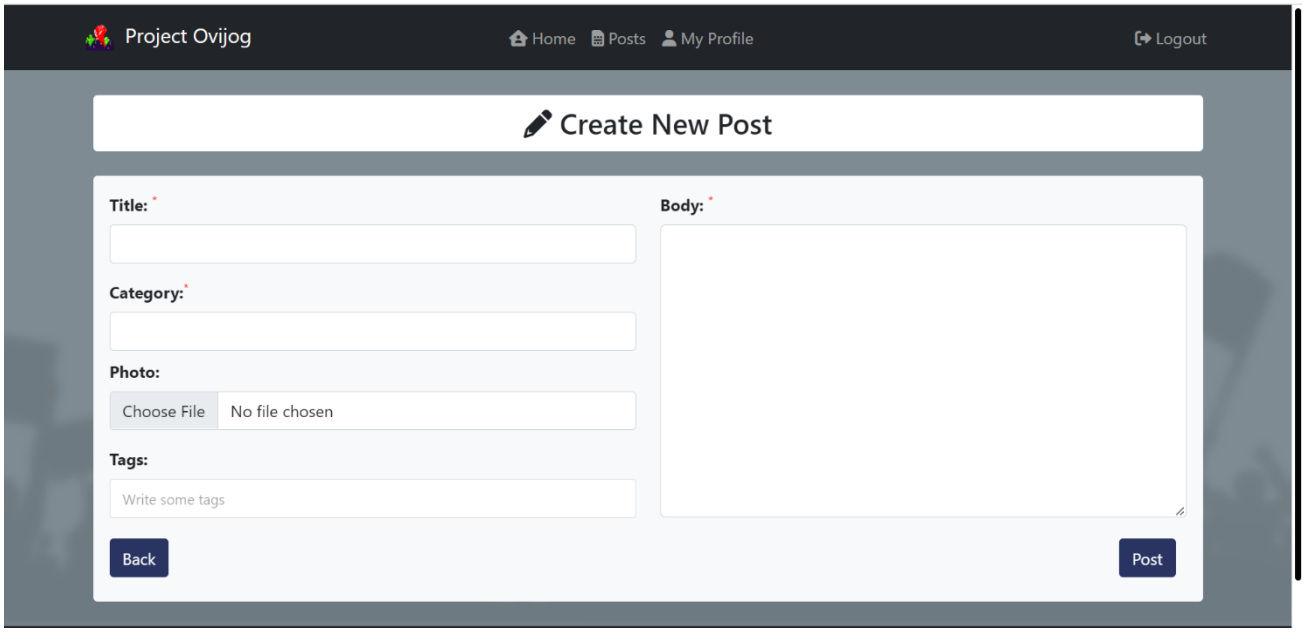

Figure 17: Create New Post

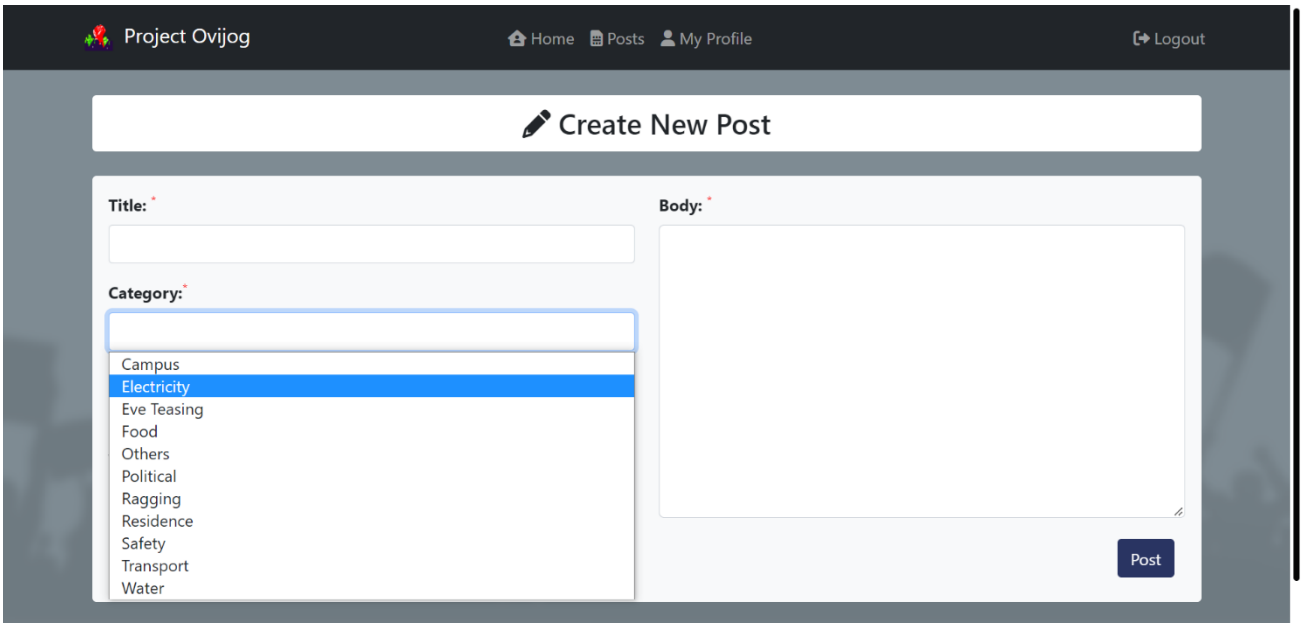

Figure 18: Choose Category

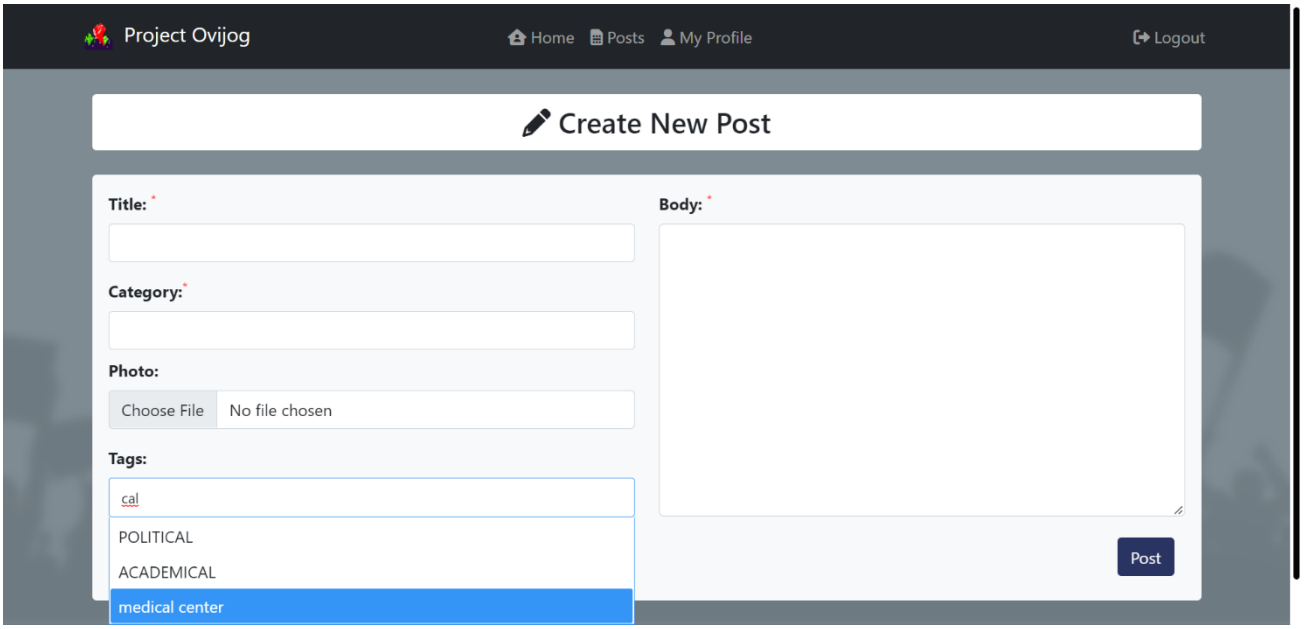

Figure 19: Tags

Edit Post

If a user wants to edit his posts, he will press edit button like (**Figure 20**) then he will find a page like (**Figure 21**). Here user can edit his own every post related content.

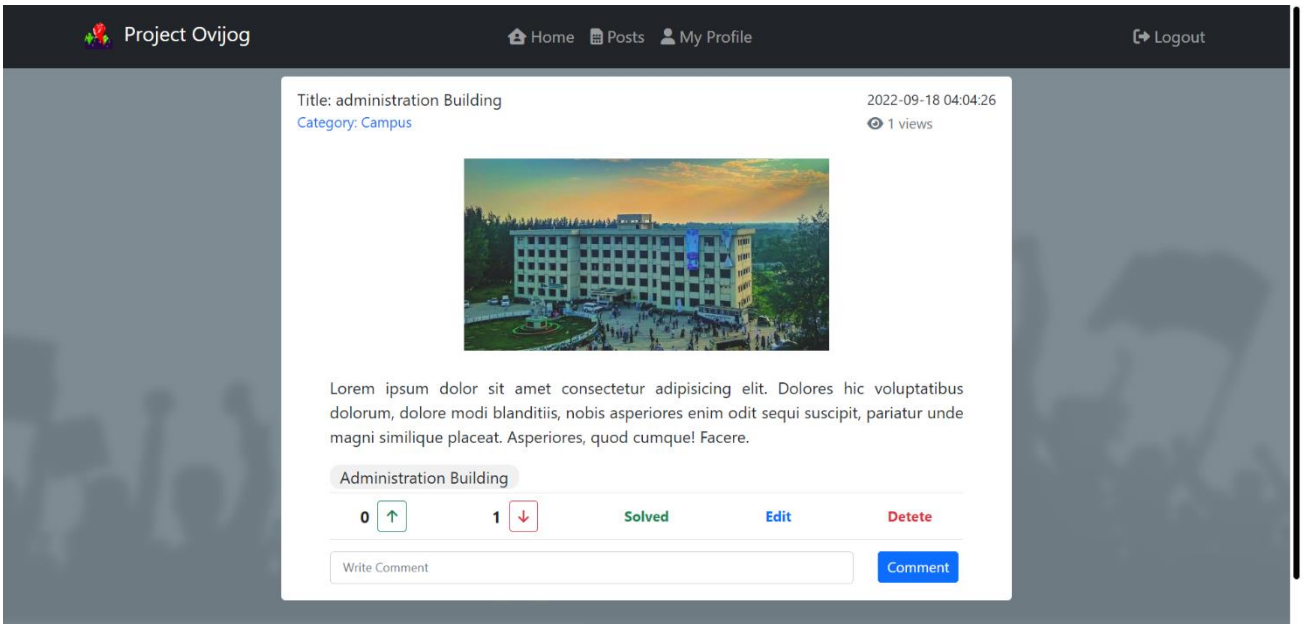

Figure 21: Edit Post (1)

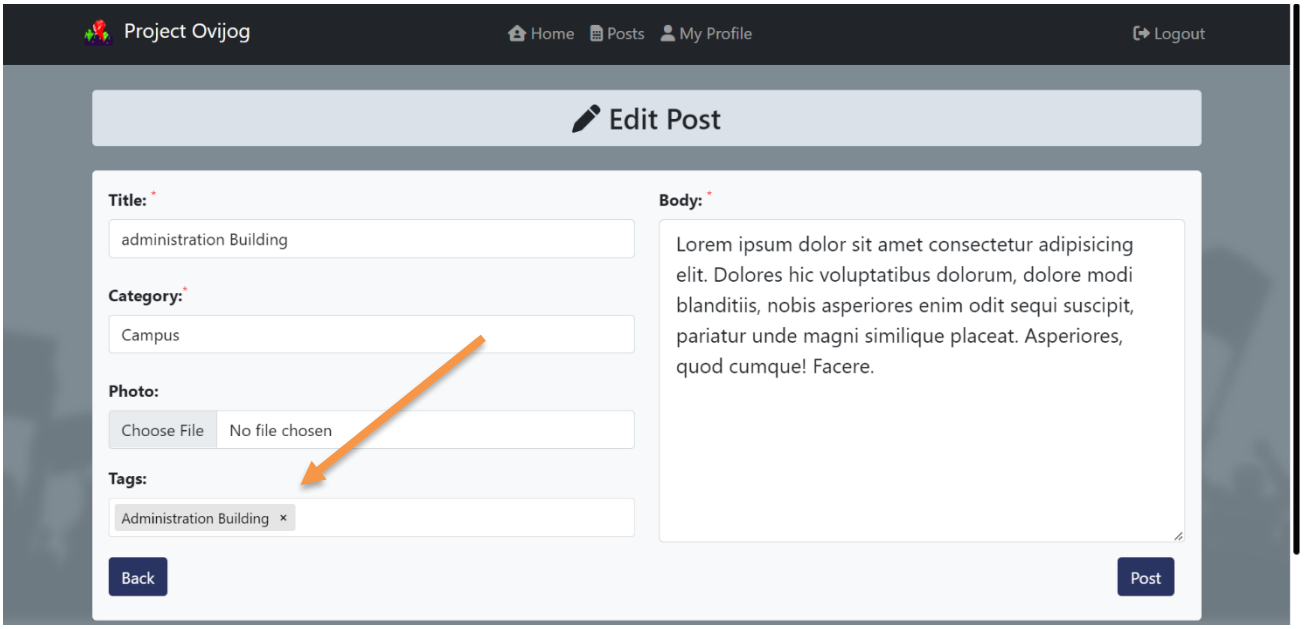

Figure 22: Edit Post (2)

Report

When a user clicks the "Report" button next to a post, a pop-up window will appear (Figure-22). The user will have some option and a text space to explain why he is disliking the post. After report has been completed a thank you pages will be shown like figure-23

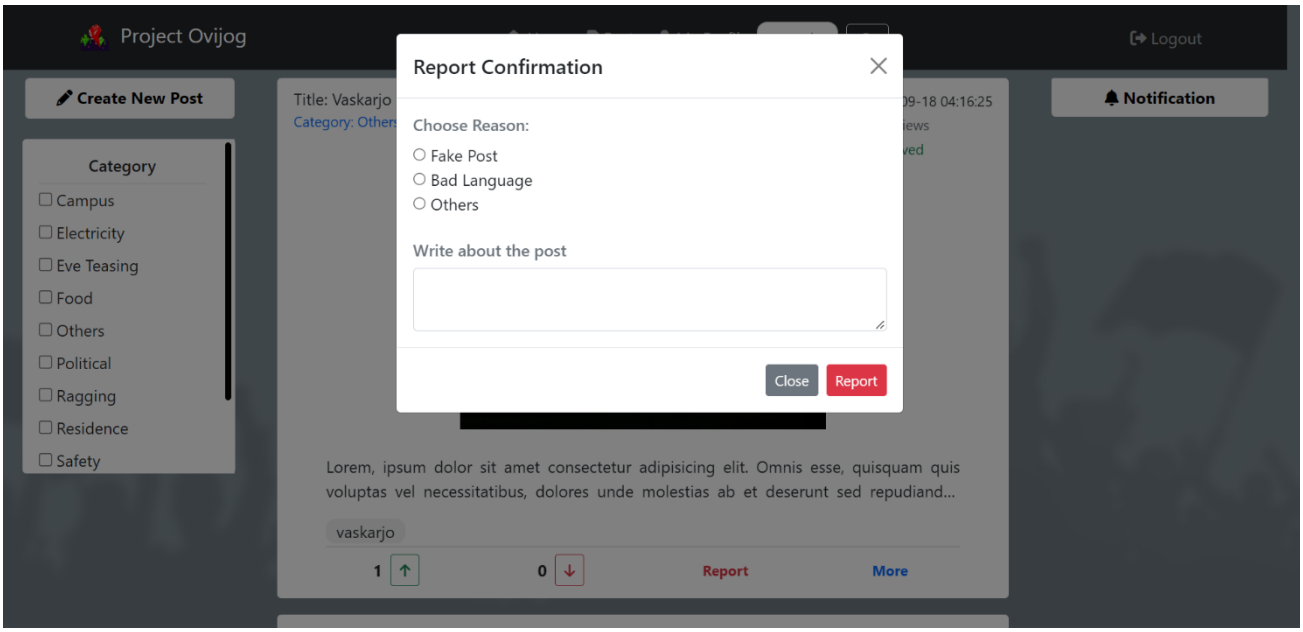

Figure 23: Report

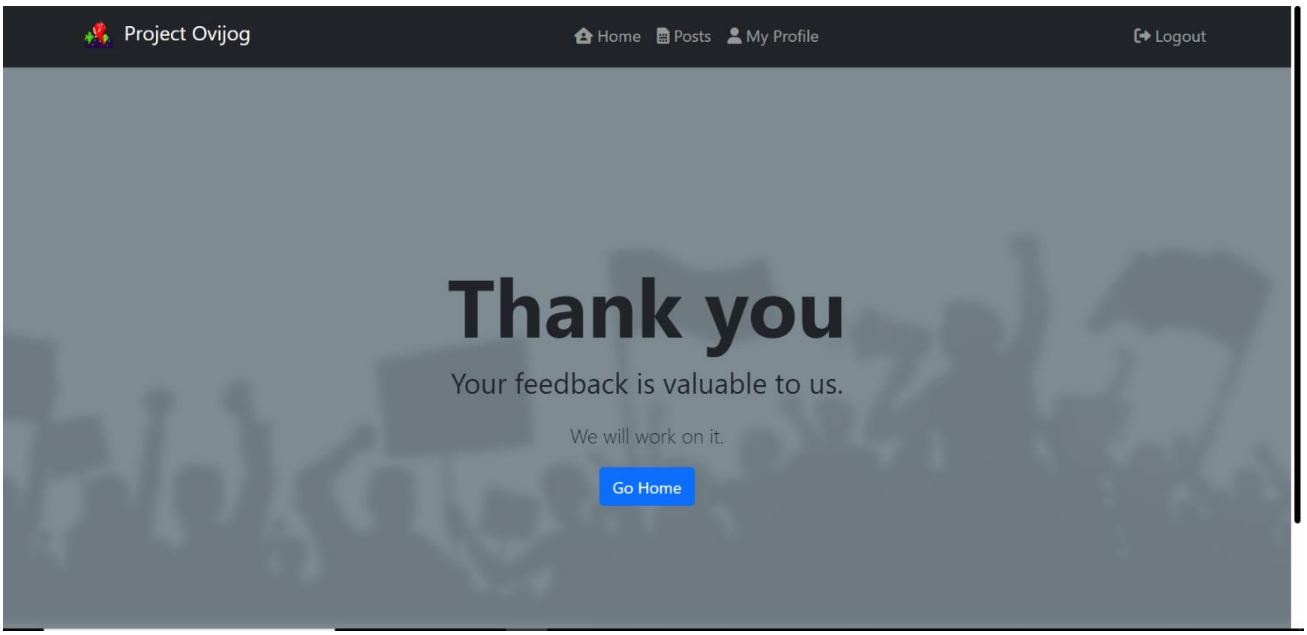

Figure 24: Report Successful

Solved

When a problem is resolved, the person who posted the issue may mark it as solved (Figure-24). The user must confirm the message after seeing a confirmation alert after pressing the solved button (Figure-25).

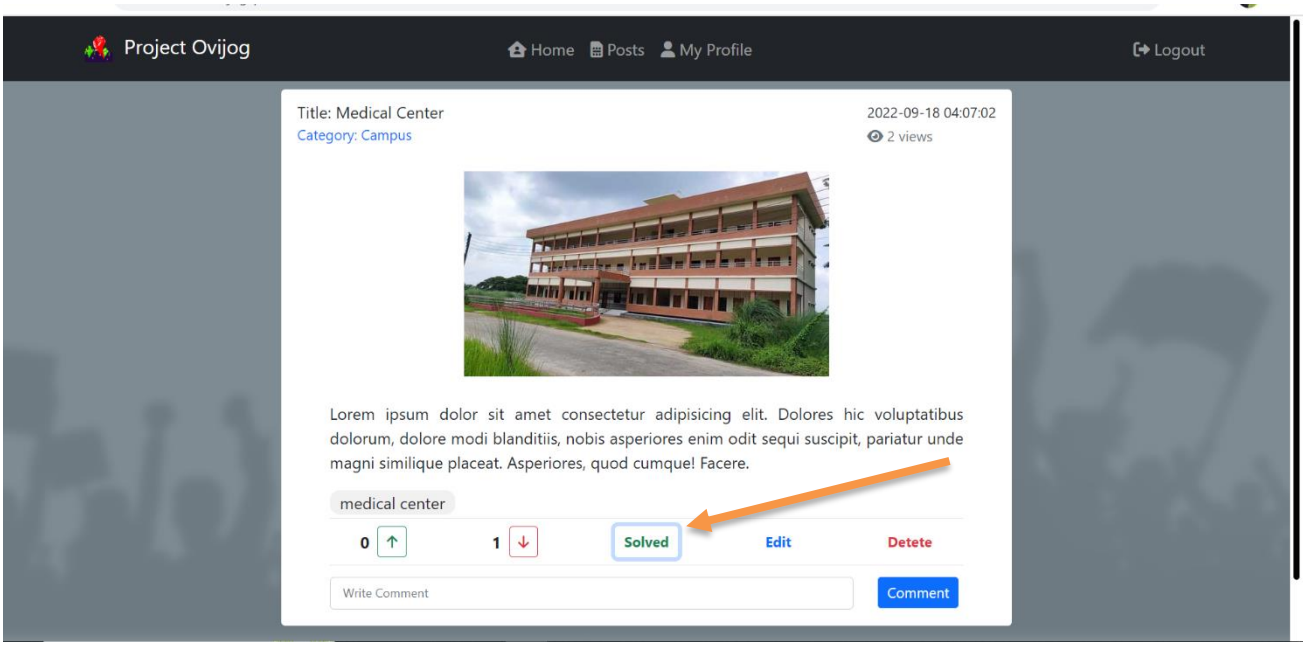

Figure 25: Solved

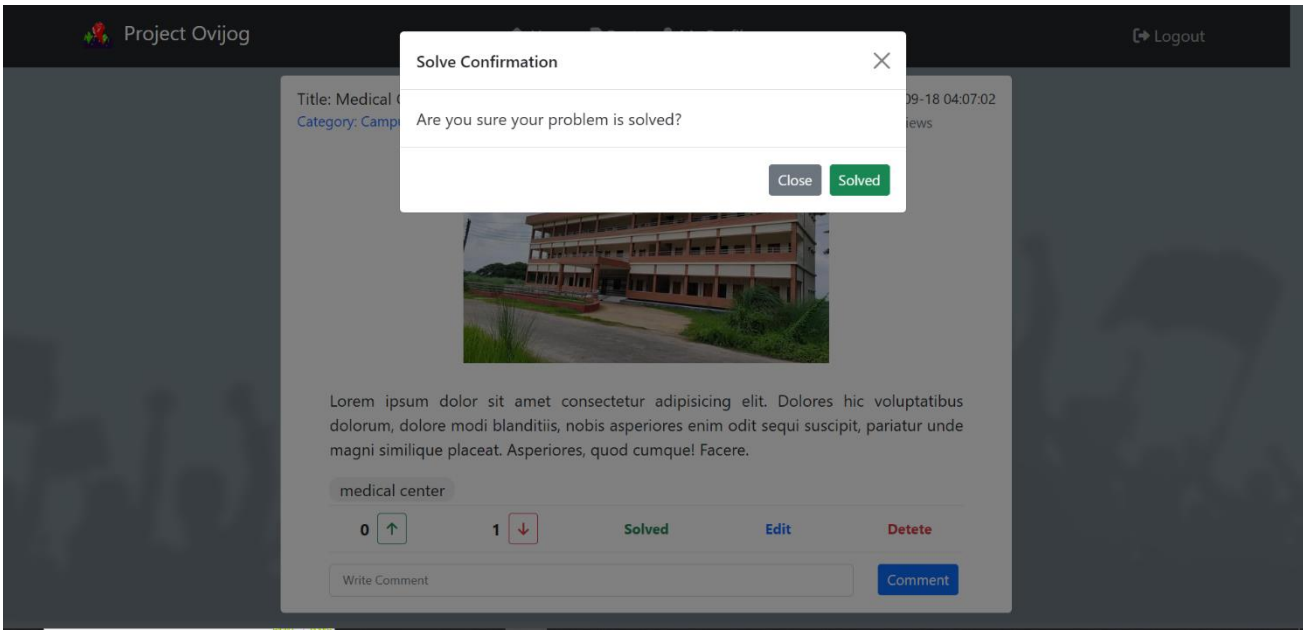

Figure 26: Solved Confirmation

Delete

The user who posted the problem has the ability to remove it at any time. An alert asking the user to confirm the deletion will appear once they click the delete button. Then user will confirm that massage.

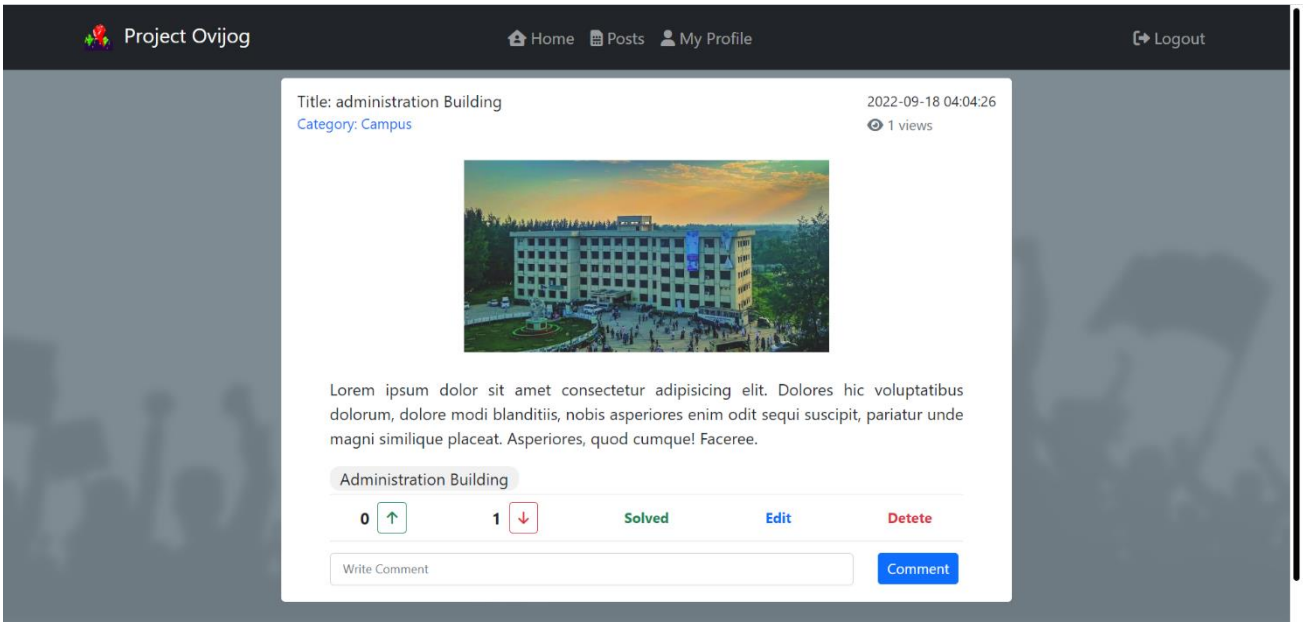

Figure 27: Delete

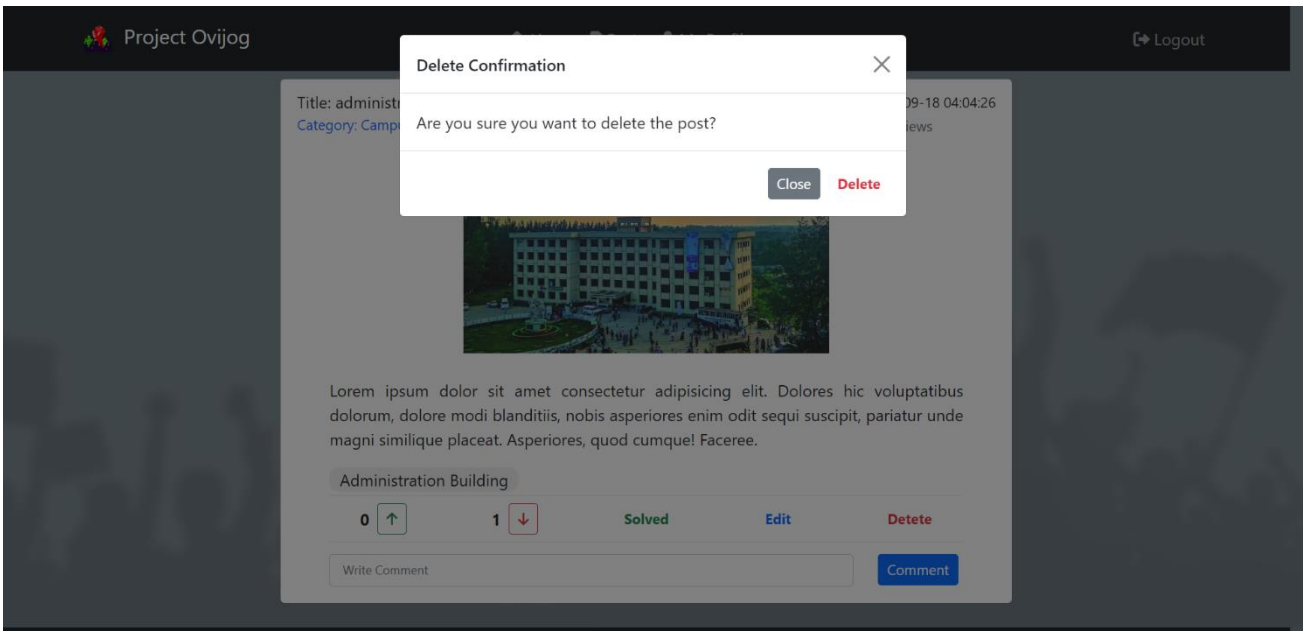

Figure 28: Delete Confirmation

Comment Section

User can comment on a post by clicking on more option. Comment will be displayed below the post section.

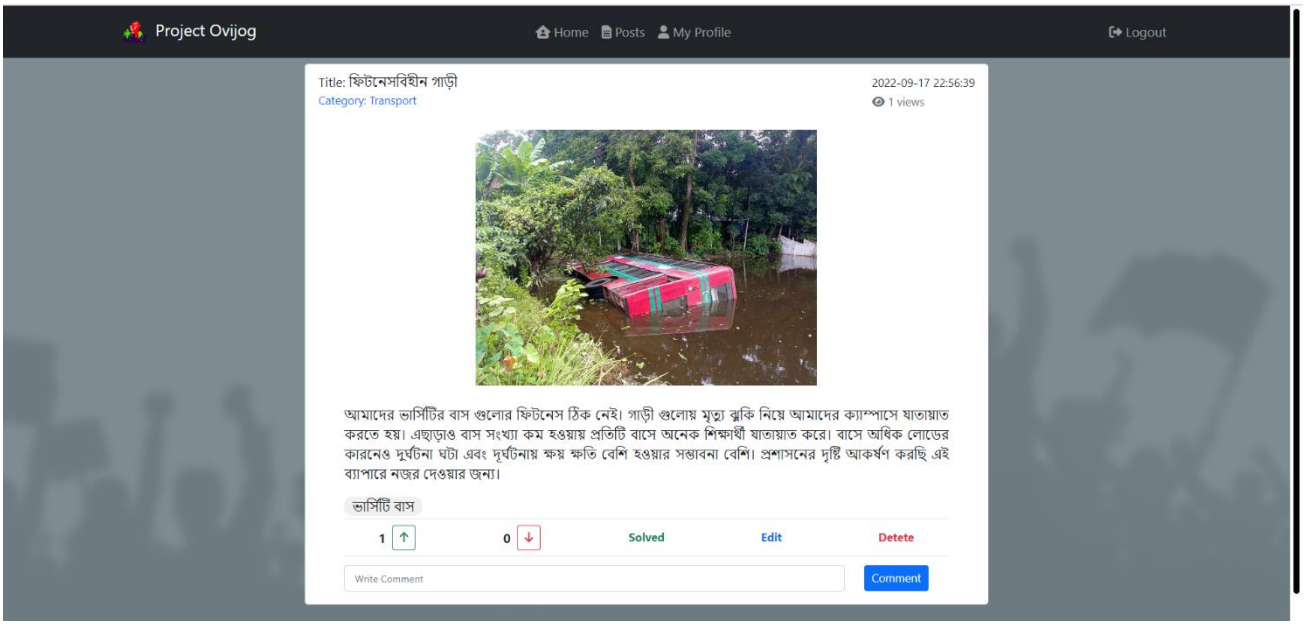

Figure 29: Comment Section

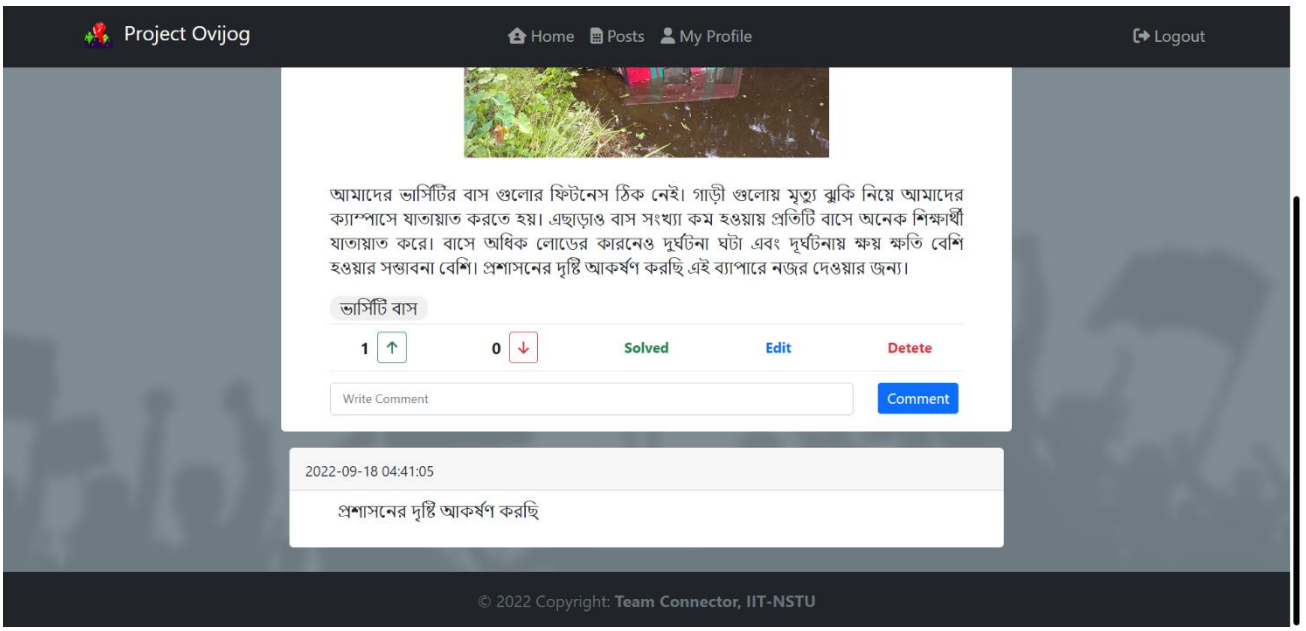

Figure 30: Comment

Notification

The user will see a notification in the notification section. When a user comments, likes or mark the post solved, the user who posted the item will be notified. The notification list will show like figure-31

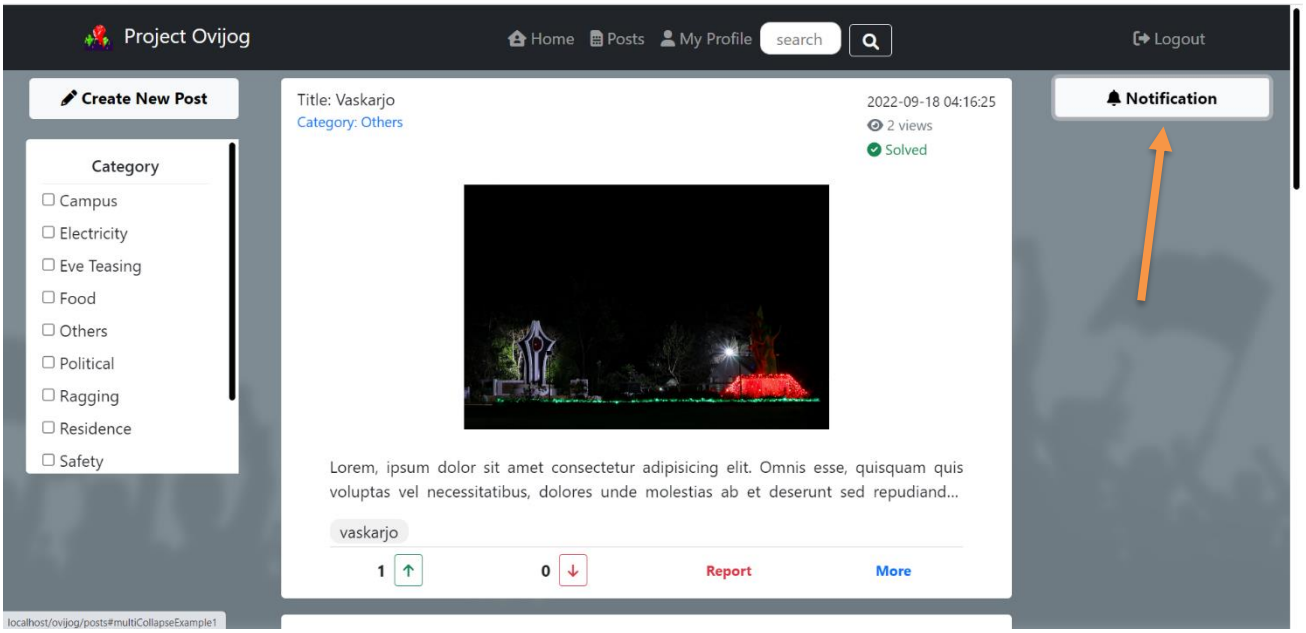

Figure 31: Notification

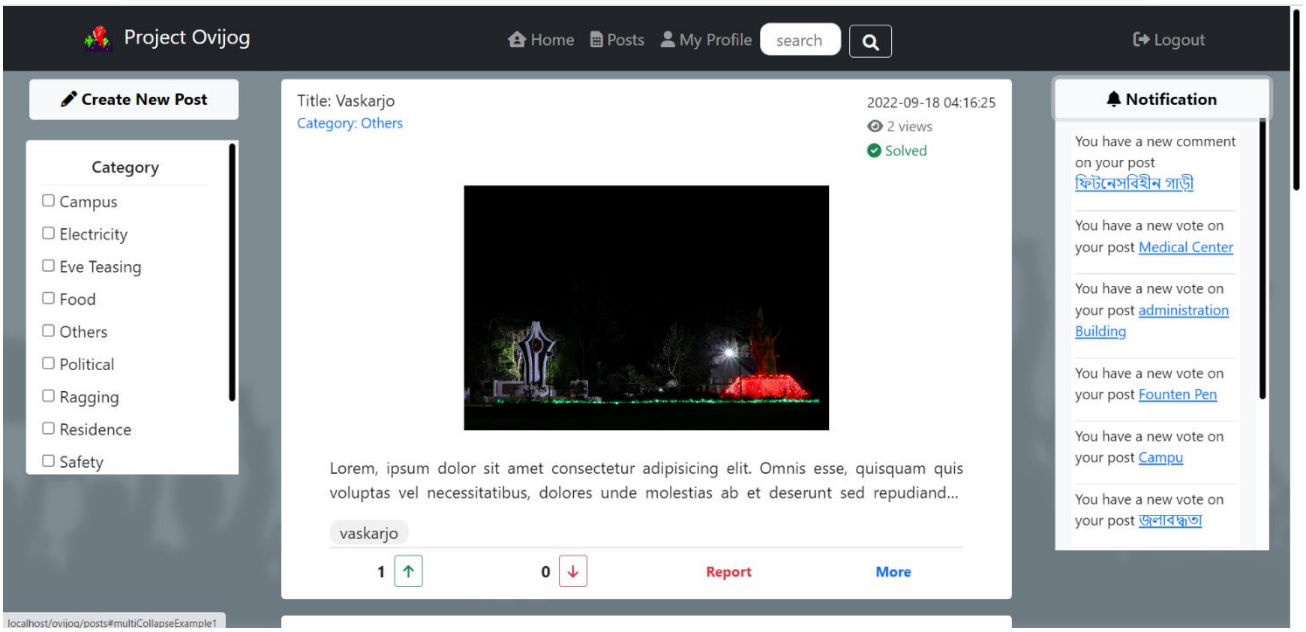

Figure 32: Notification List

Filtering

The user can filter the category to which they want to post (Figure-32). A single category or several categories may be chosen at once. When a user selects a single category, the results are shown based on that category (Figure-33), but if the user selects multiple categories, the results are shown based on several categories (Figure-34).

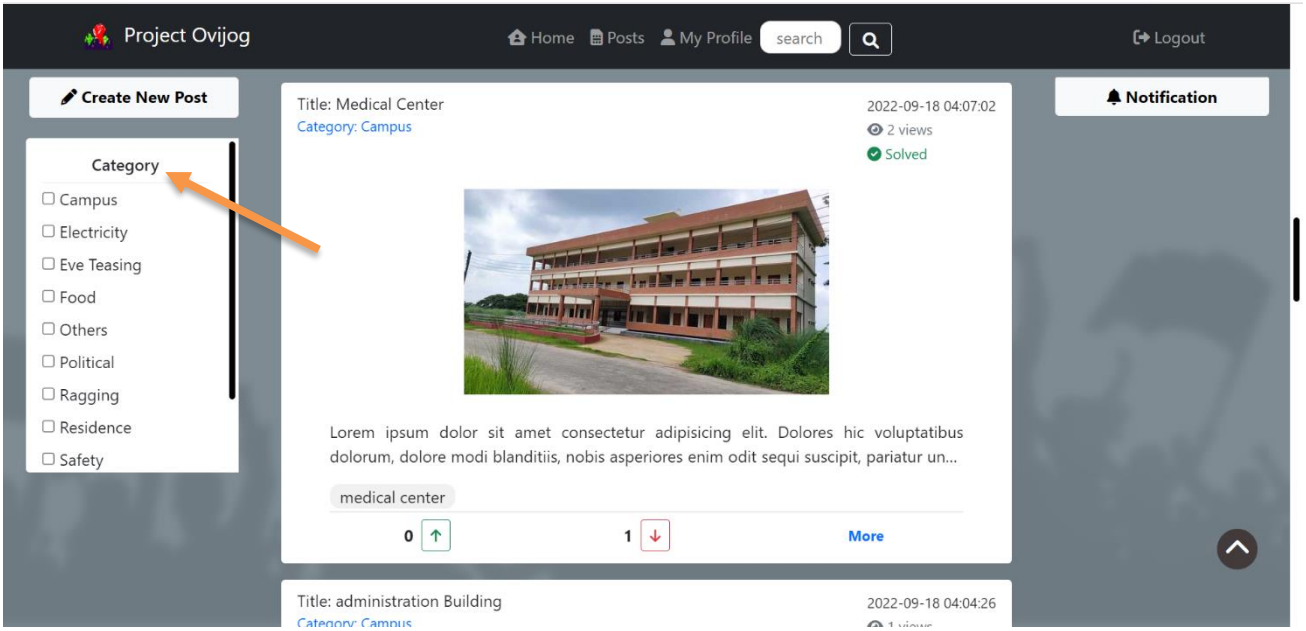

Figure 33: Filtering

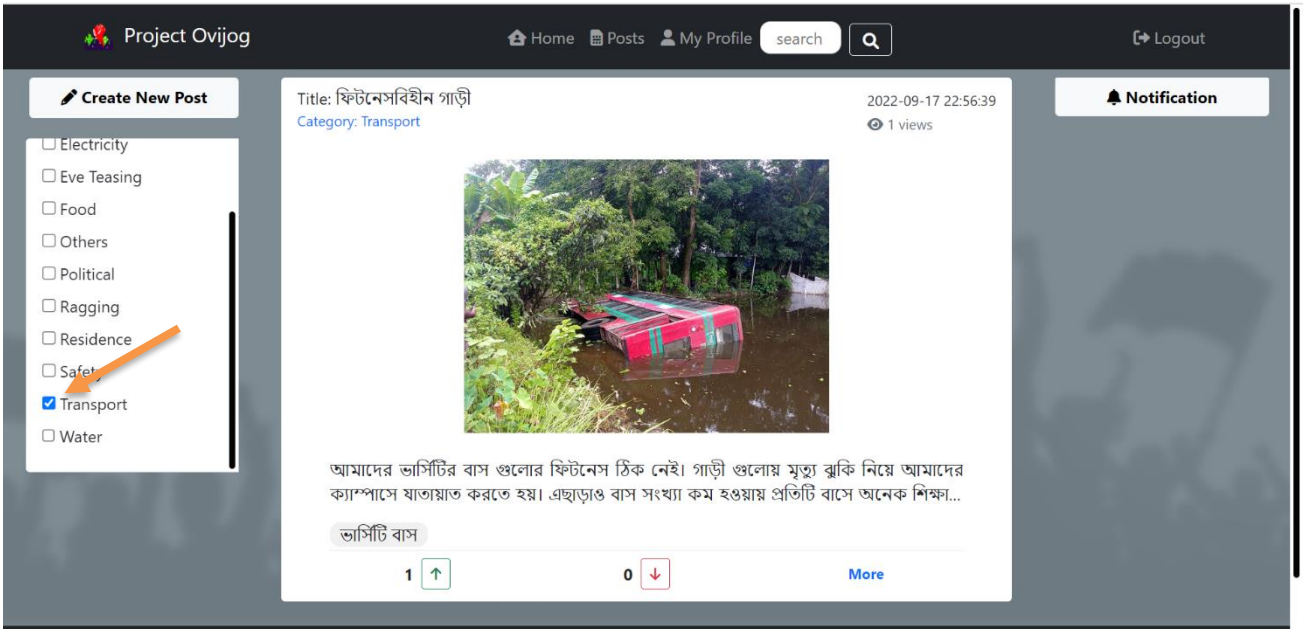

Figure 34: Single Category Filtering

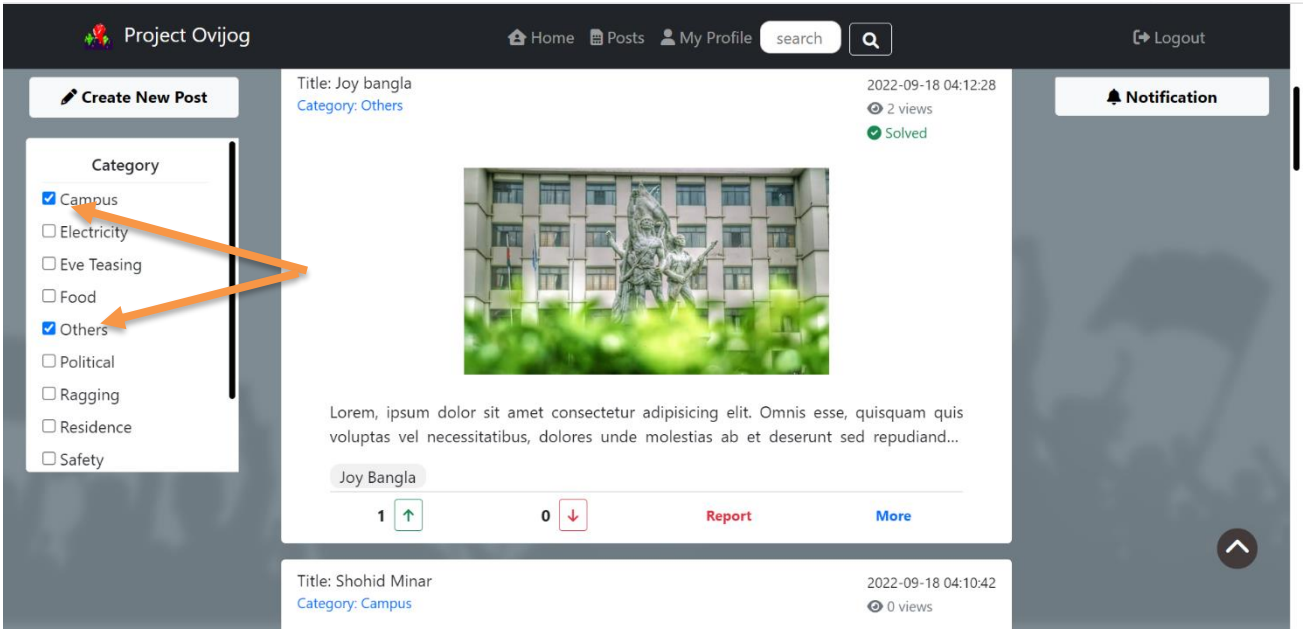

Figure 35: Multiple Category Filtering

Page Top

There is a button that will lead you to the top of the page.

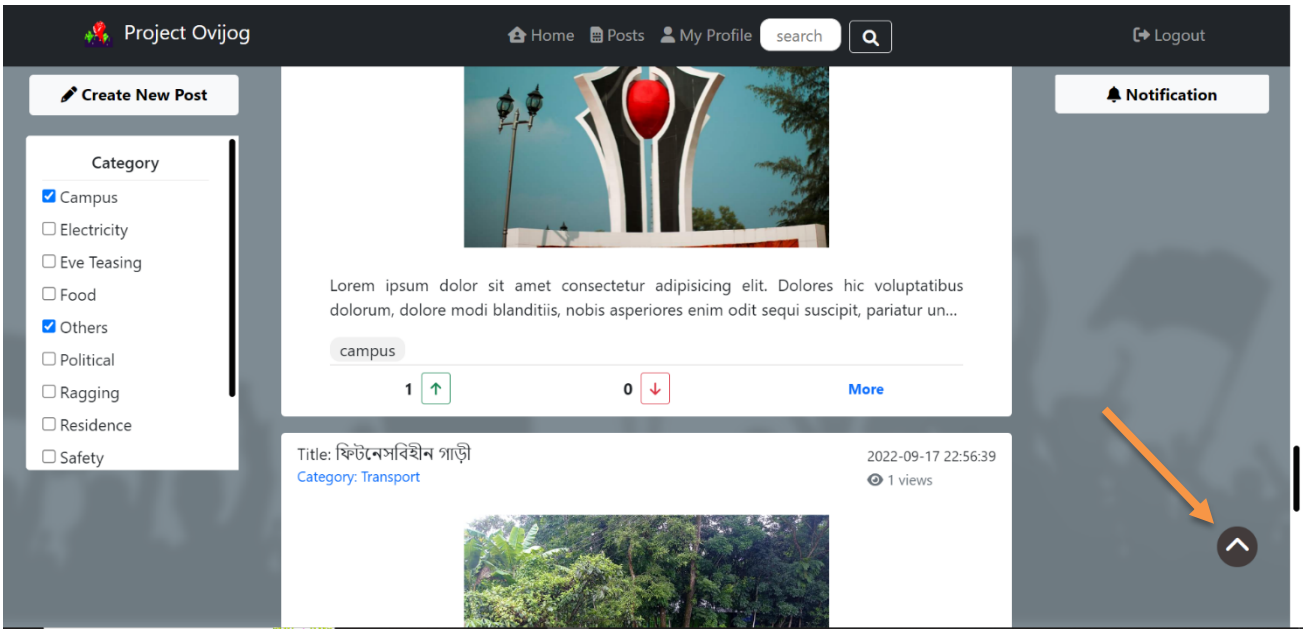

Figure 36: Go to page top

Profile

The user can view all of his information in the profile section. The user may also view his own posts here. By choosing the types, a user can view the posts on which he has voted and commented (Figure-37).

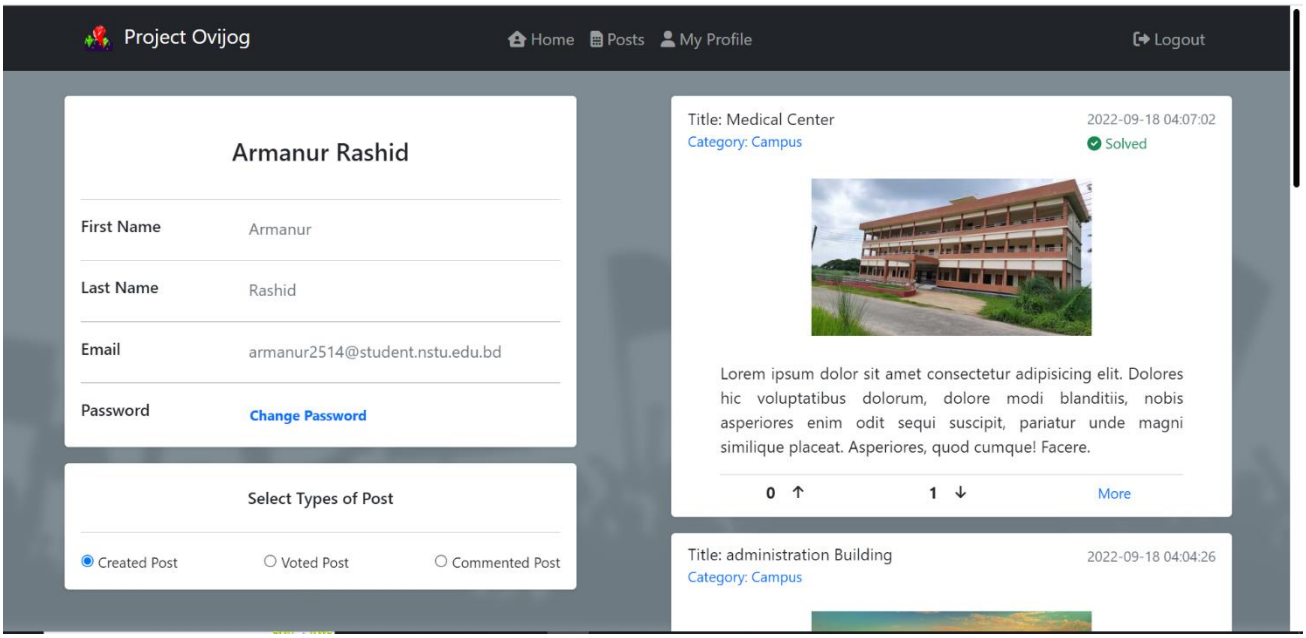

Figure 37: Profile

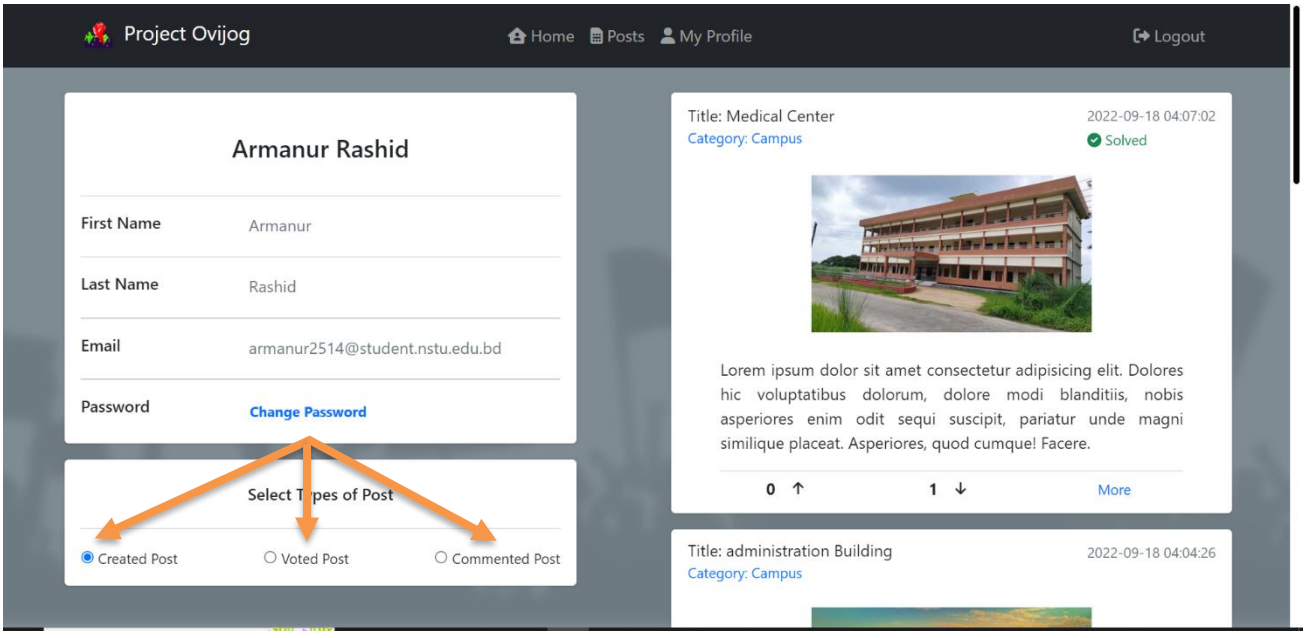

Figure 38: Activities

Change Password

Users can update their passwords by clicking the change password option, which triggers the appearance of a pop-up window. where the user must enter both his old password and his new password. Once you click the "Change Password" button, the password will be updated.

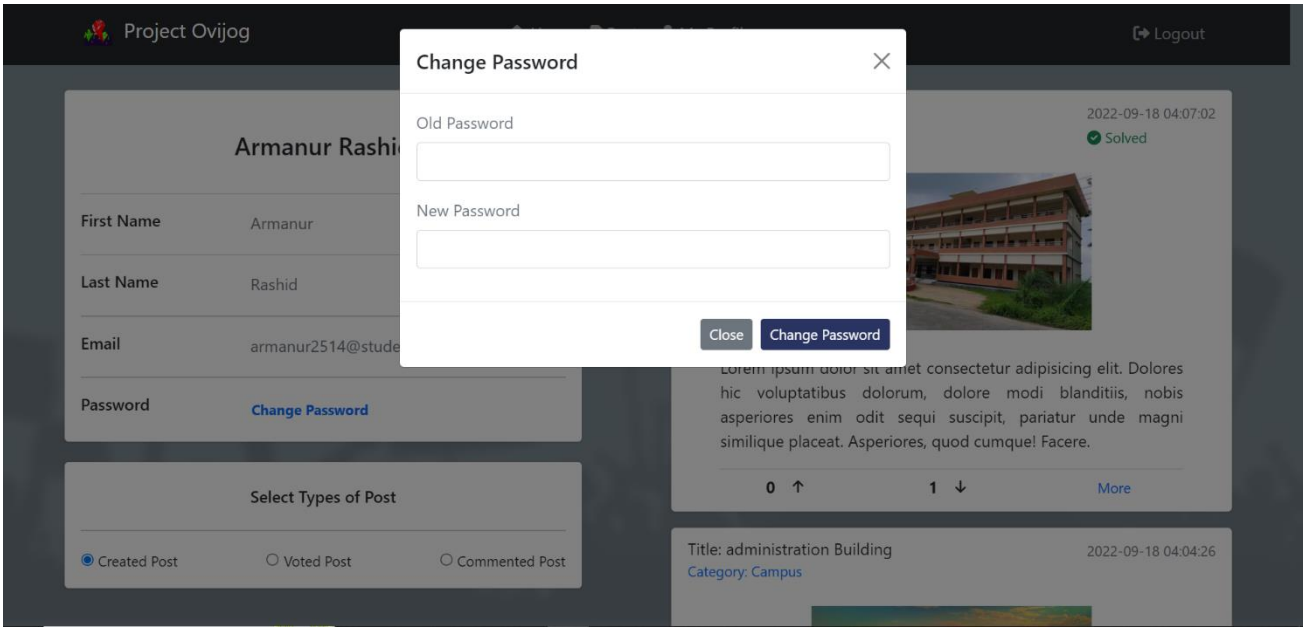

Figure 39: Change Password

Forgot Password

In the forgot password section, a pop-up box will appear, asking for the user's email. A verification link will be sent to the provided email (Figure-41).

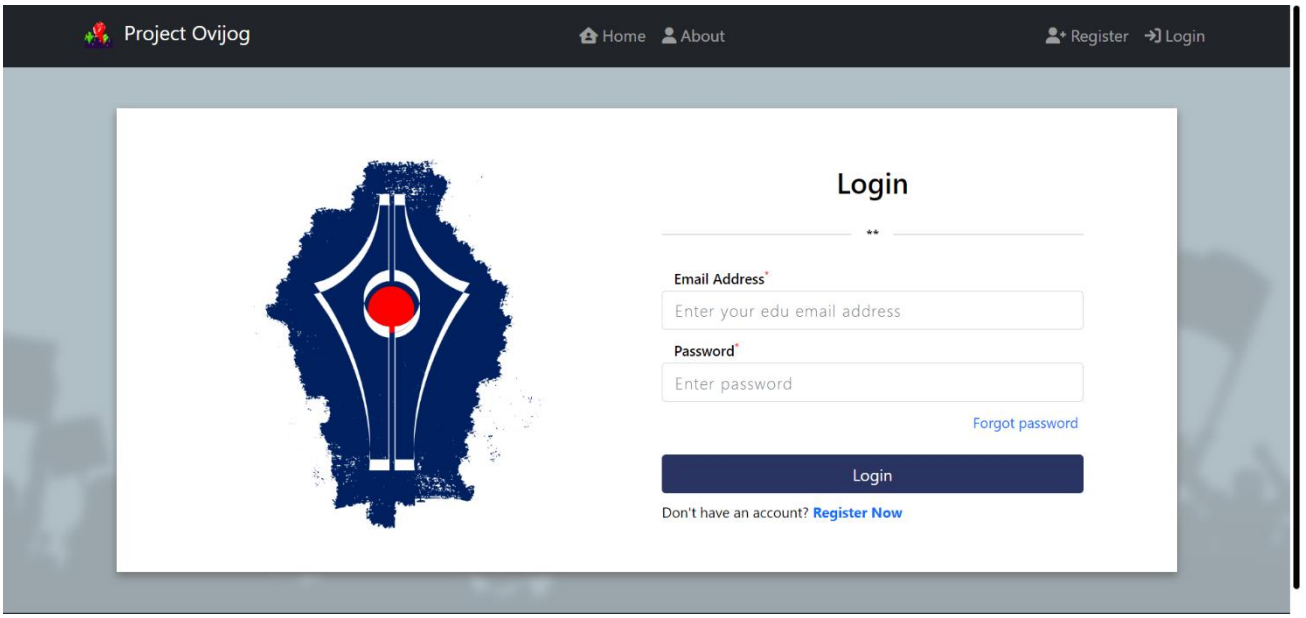

Figure 40: Forgot Password

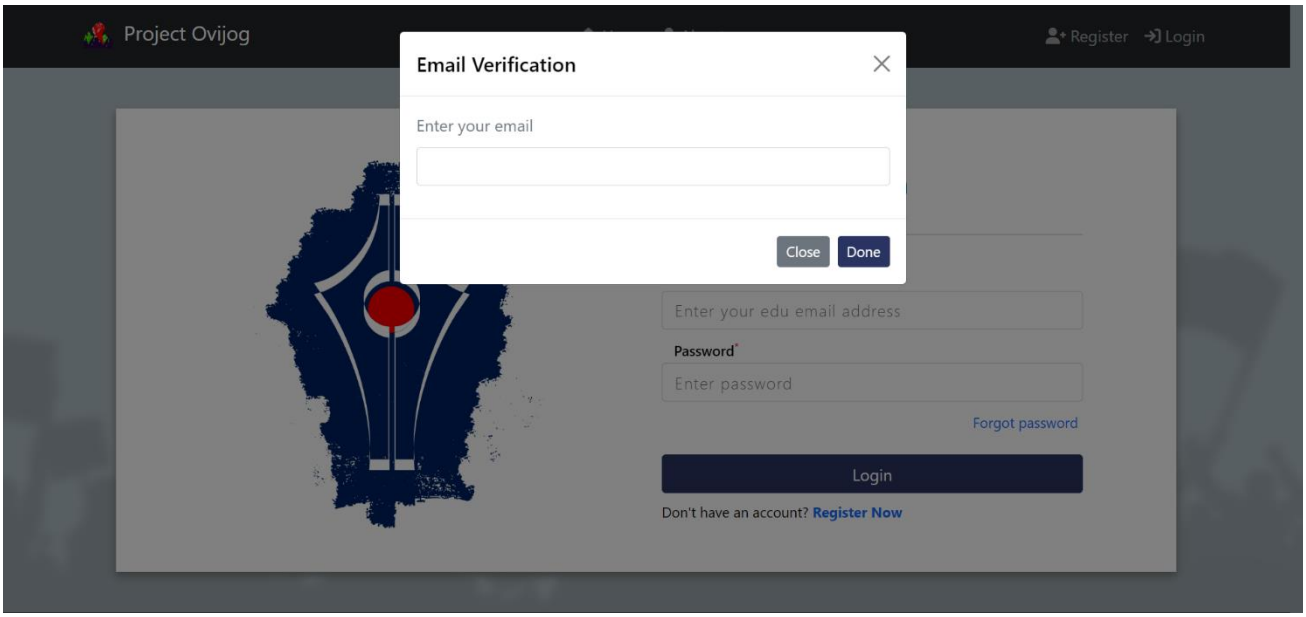

Figure 41: Email verification for forgot password

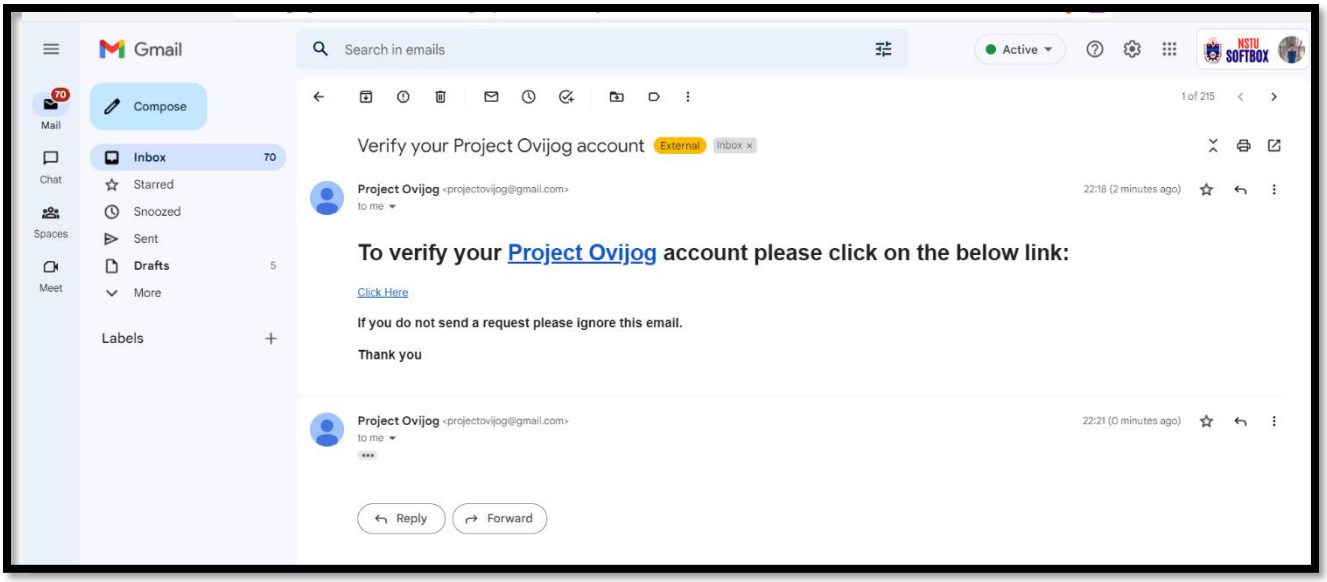

Figure 42: Email

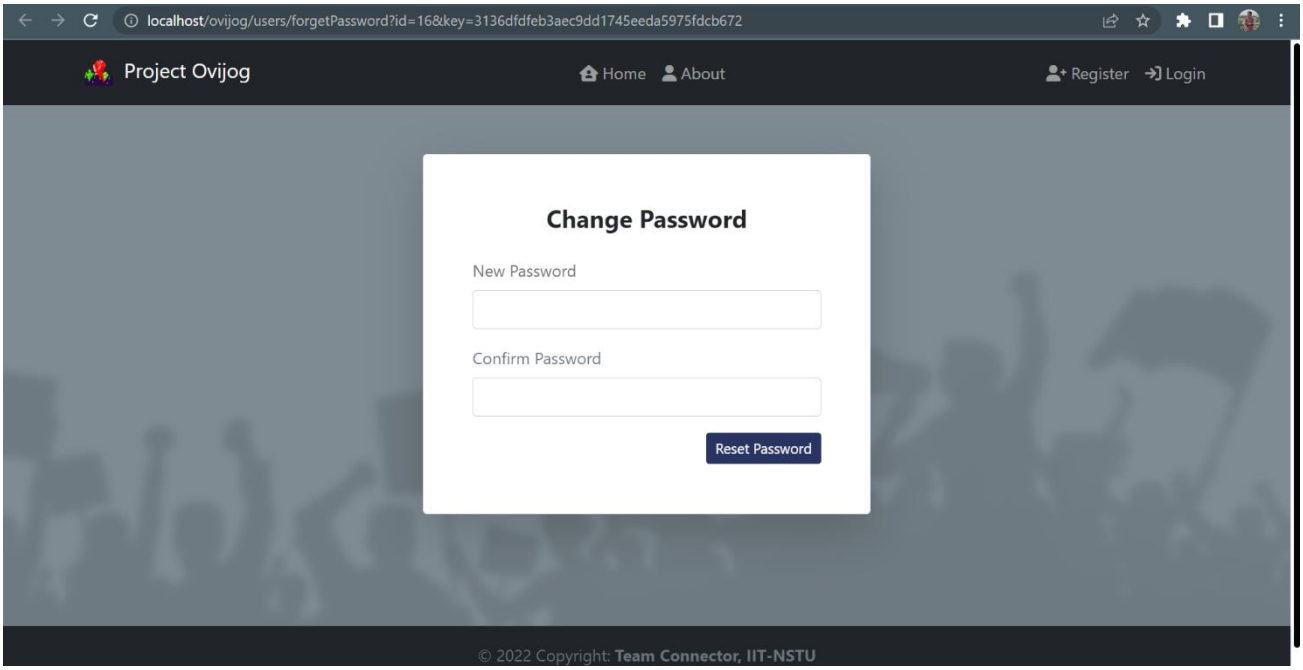

Figure 43: New Password

Logout

User can also log out from the system by clicking logout button.

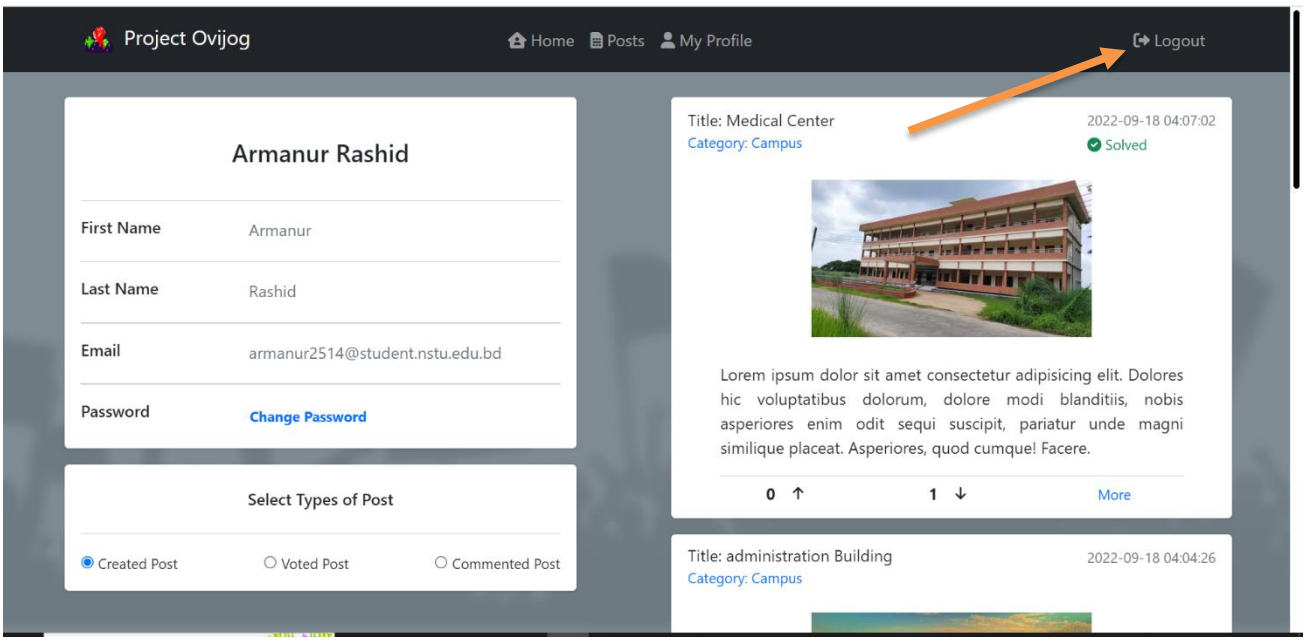

Figure 44: Logout

Admin Side

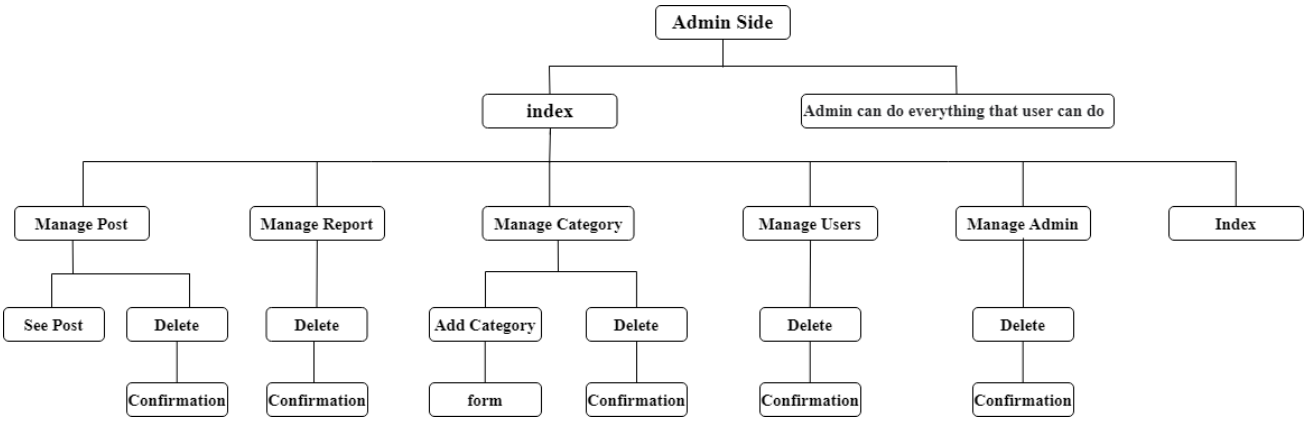

Figure 45: Admin Side Architecture

Admin index

The admin can see the number of users using the system, the number of posts, the number of posts that have been solved, the number of posts that have not yet been solved, and the number of reports (Figure-45)

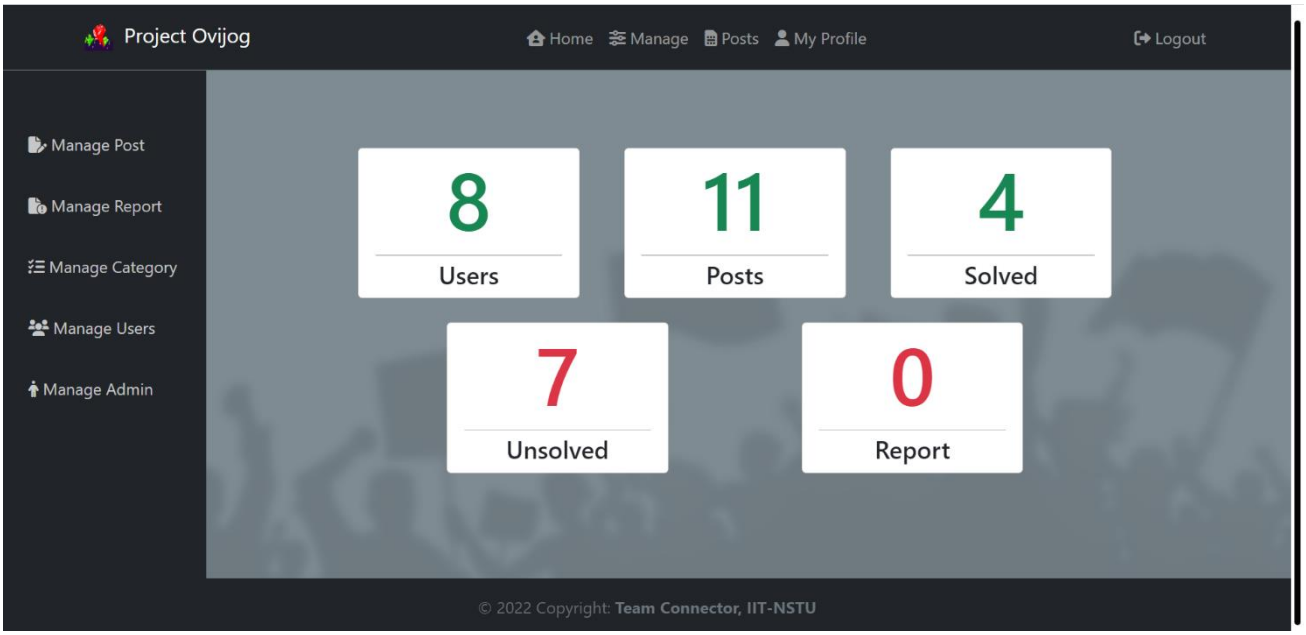

Figure 46: Admin Index

Manage Post

An admin can manage the posts in this section. Admin can see the post. Admin has the authority to delete any post.

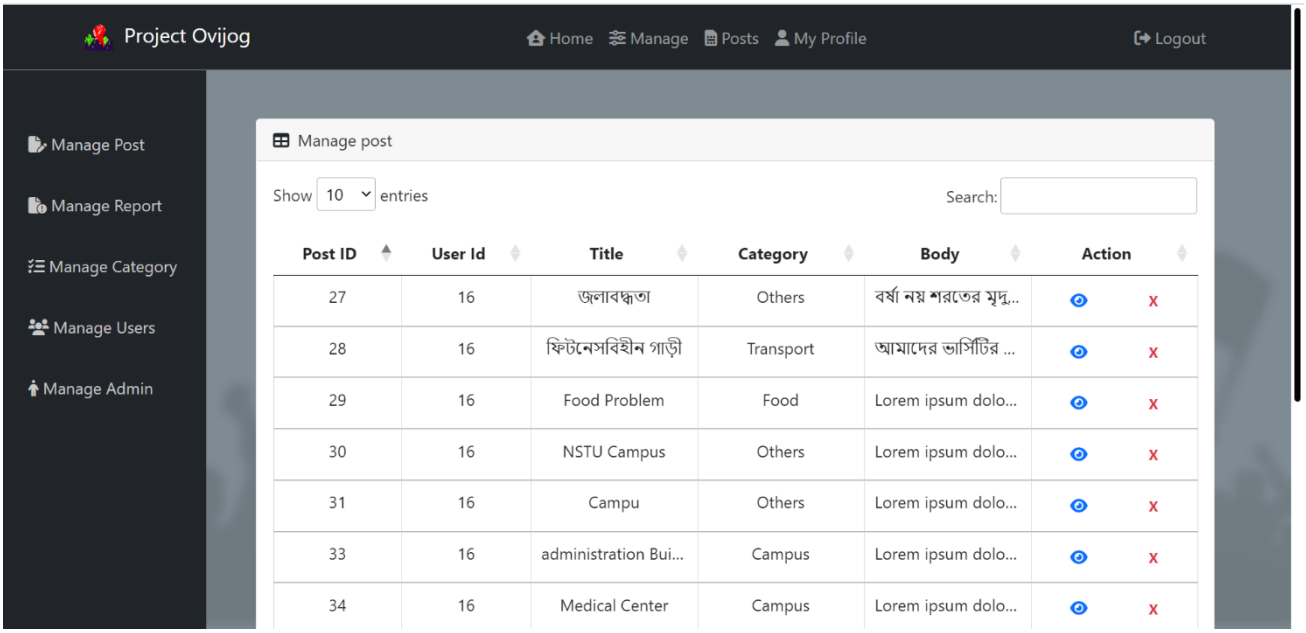

Figure 47: Manage Post

Manage Report

An admin can take actions against a reported posts in this section. Admin can see the post before taking any action. Admin has the authority to delete any post.

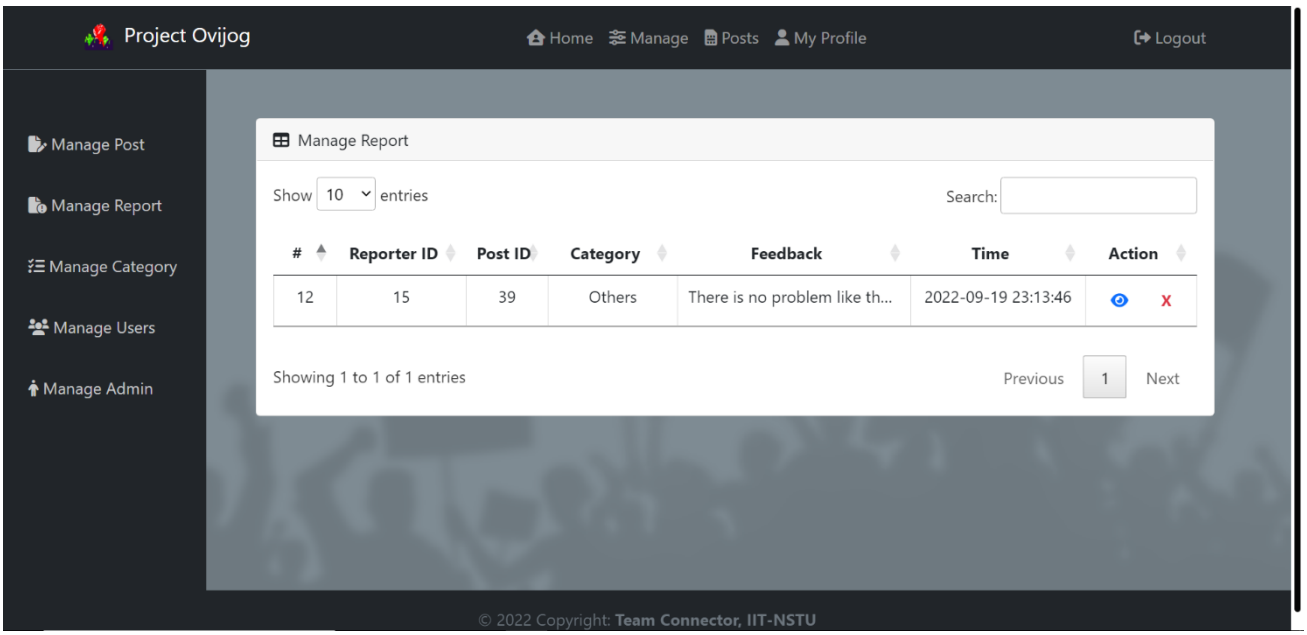

Figure 48: Manage Report

Manage Category

An admin can see which are the categories in this section. Admin can create a new category and remove a category (Figure-49)

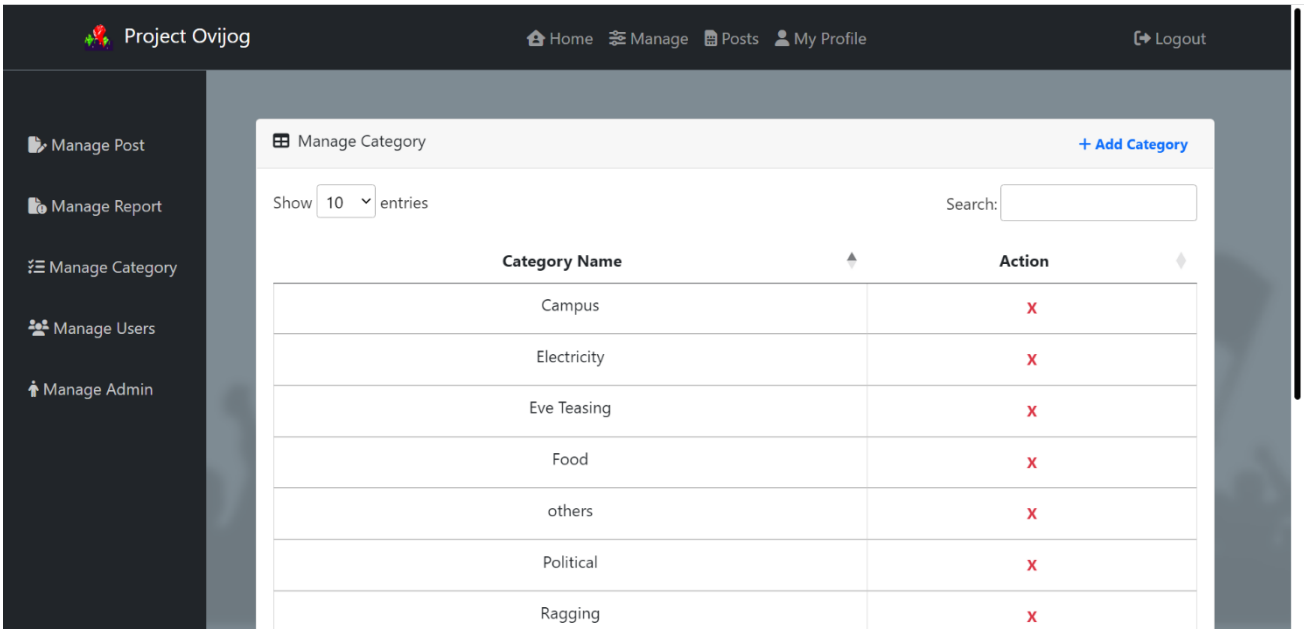

Figure 49: Manage Category

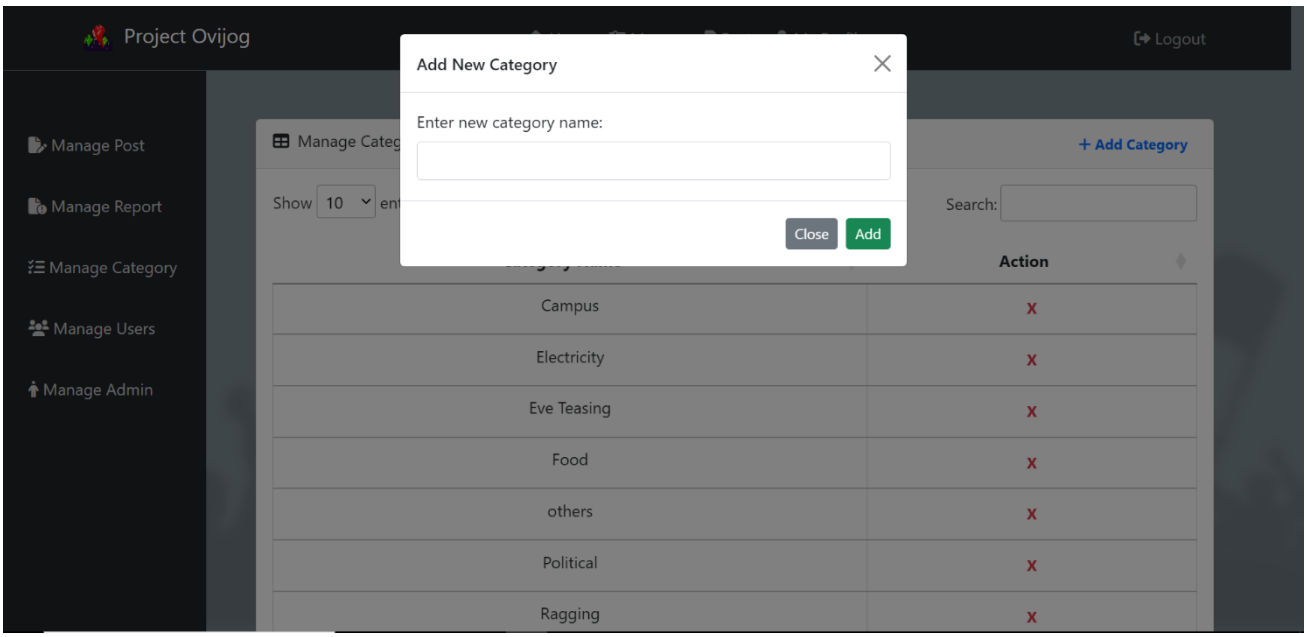

Figure 50: Add New Category

Manage Users

An admin can see who the users are in this section. Admin has the authority to remove any user.

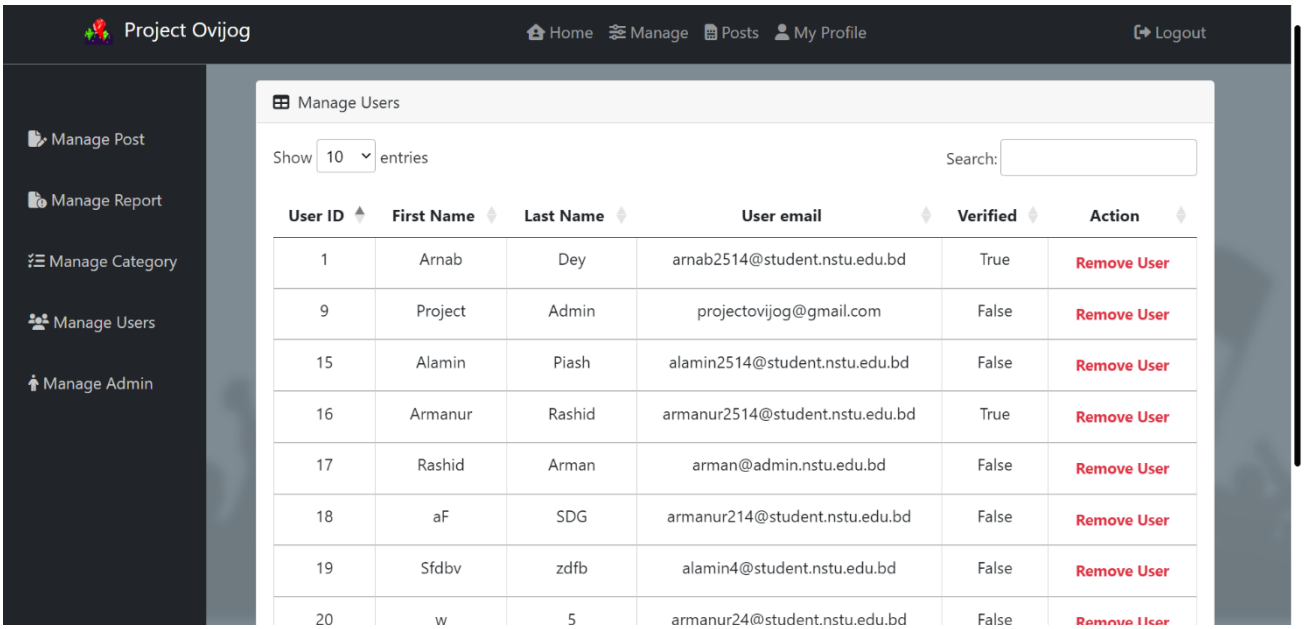

Figure 51: Manage Users

Manage Admin

An admin can see who the admins are in this section. By providing the user id of any legitimate user, one admin can add another admin and one admin can remove other admin (Figure-51,52).

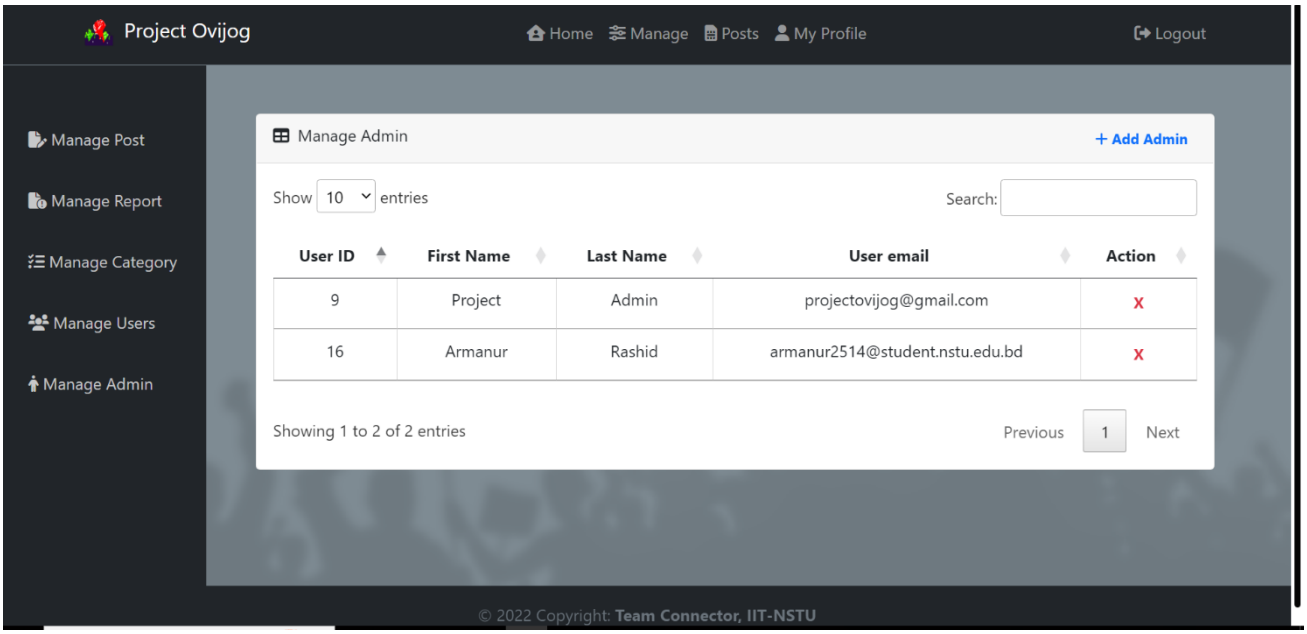

Figure 52: Manage Admin

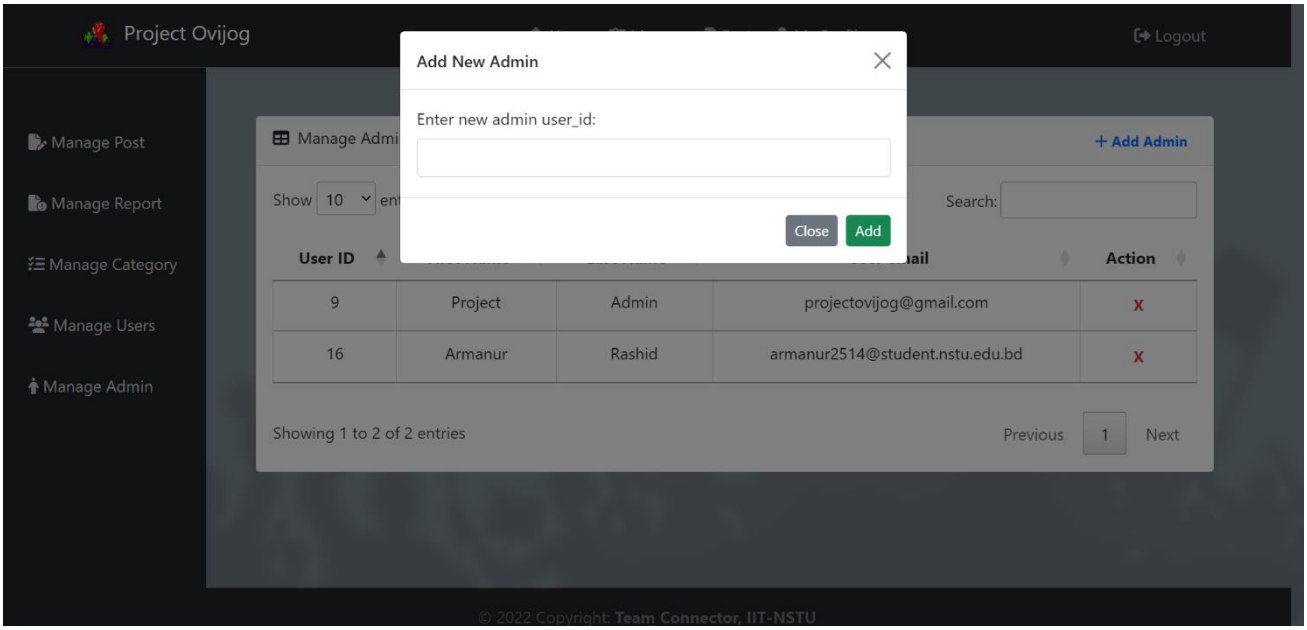

Figure 53: Add New Admin

Source Code Documentation

We follow the MVC (Model-View-Controller) architecture. The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application.

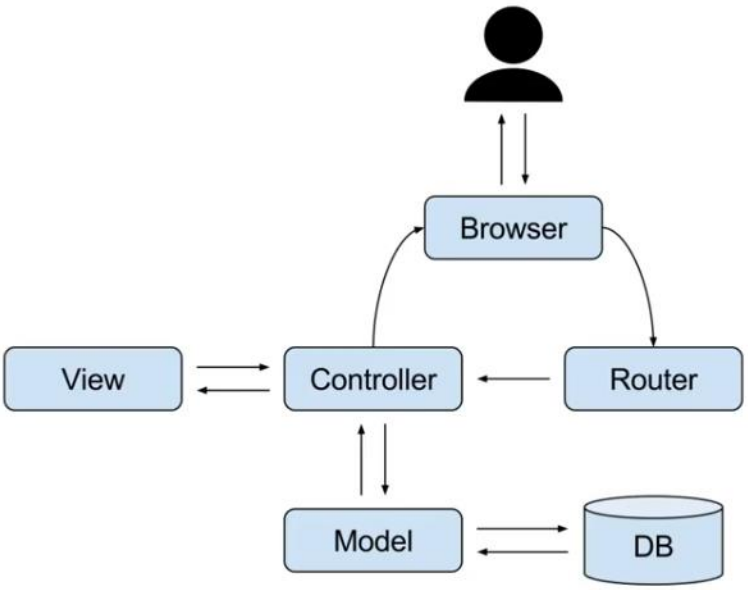

Figure 54: Model-View-Controller

Model: Model represents the shape of the data. Model objects store data retrieved from the database.

View: View in MVC is a user interface. View display model data to the user and also enables them to modify them.

Controller: The controller handles the user request. Typically, the user uses the view and raises an HTTP request, which will be handled by the controller. The controller processes the request and returns the appropriate view as a response.

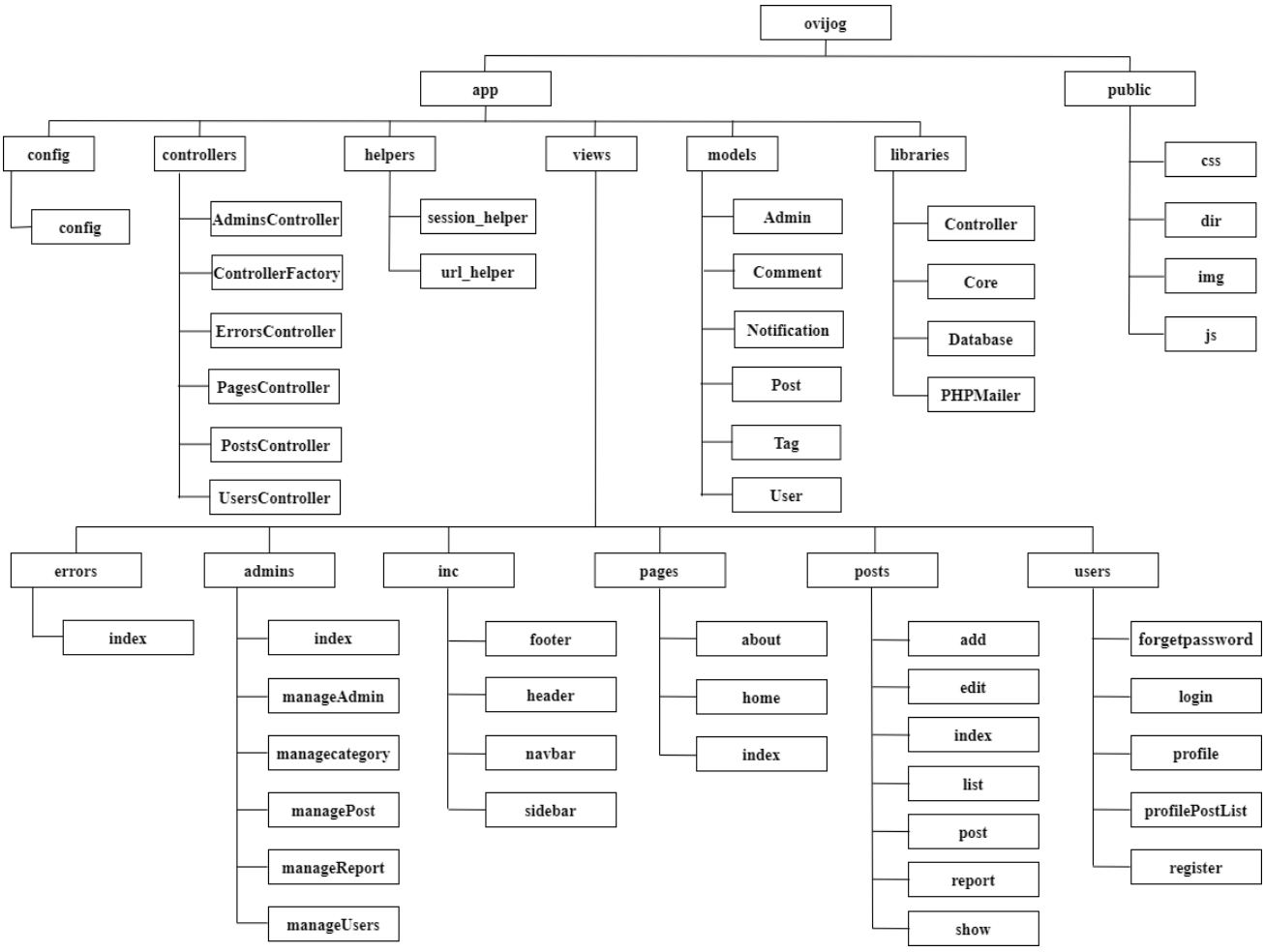

Figure 55: Code Structure

1 App

1.1 config

1.1.1 config.php

A configuration file, often known as a config file, specifies the system's parameters, options, settings, and preferences.

1.2 libraries

1.2.1 Core.php

App Core Class Handles URL & loads core controller URL format: /controller/method/params.

getUrl()

Description : This method split the url into parts
Return Value : array<string|int, string>|void - splitted url

1.2.2 Controller.php

Base controller defines two methods to load appropriate model and appropriate view as needed.

Model()

Description : Loads a model
Params : $model: model's name
Return Value : model instance

View()

Description : Loads a view
Params : $view: mixed, $data: mixed = []
Return Value : void

1.2.3 Database.php

Database class to efficiently and effectively write queries and perform.

bind()

Description : Bind the values in the statement
Params : $param: mixed, $value: mixed, $type: mixed = null
Return Value : void

execute()

Description : Execute Statement
Return Value : mixed – result

getInstance()

Description : Return Database instance

Return Value : instance of database.

lastInsertId()

Description : Returns last inserted id.
Return Value : false|string

query()

Description : Build a statement with given sql command
Params : $sql: mixed
Return Value : void

resultSet()

Description : Returns result set as objects.
Return Value : mixed - result set.

rowCount()

Description : Returns row count of the result.
Return Value : mixed – row count

single()

Description : Returns a single row of the result as object.
Return Value : mixed - single row.

1.3 Controllers

1.3.1 AdminsController.php

Admins controller that handles request's prefix with 'admins'.

AddCategory()

Description : This method handles requests '/admins/addCategory'.
Return Value : void

deletePost()

Description : This method handles requests '/admins/deletePost/params'.
Params : $id: mixed
Return Value : void

deleteReportedPost()

Description : This method handles requests '/admins/deletePortedPost/params'.
Params : $id: mixed
Return Value : void

deleteUser()

Description : This method handles requests '/admins/deleteUser/params'.
Params : $id: mixed
Return Value : void

index()

Description : This method handles requests '/admins/index' and '/admins'.
Return Value : void

makeAdmin()

Description : This method handles requests '/admins/makeAdmin'.
Return Value : void

manageAdmin()

Description : This method handles requests '/admins/manageAdmin'.
Return Value : void

manageCategory()

Description : This method handles requests '/admins/manageCategory'.
Return Value : void

managePost()

Description : This method handles requests '/admins/managePost'.
Return Value : void

manageReport()

Description : This method handles requests '/admins/manageReport'.
Return Value : void

manageUsers()

Description : This method handles requests '/admins/manageUsers'.
Return Value : void

model()

Description : Loads a model.
Params : $model: model's name
Return Value : mixed - model instance

removeAdminShip()

Description : This method handles requests '/admins/removeAdminShip/params'.
Params : $id: mixed
Return Value : void

removeCategory()

Description : This method handles requests '/admins/removeCategory/params'.
Params : $category: mixed
Return Value : void

view()

Description : Loads a view.
Params : $view: mixed, $data: mixed = []
Return Value : void

1.3.2 ErrorsController.php

Error controller that handles request's prefix with 'errors'.

index()

Description : This method handles requests '/errors/index' and '/errors'.
Return Value : void

model()

Description : Loads a model.
Params : $model: model's name
Return Value : mixed - model instance

view()

Description : Loads a view.
Params : $view: mixed, $data: mixed = []
Return Value : void

1.3.3 PagesController.php

Pages controller that handles request's prefix with 'pages'.

about()

Description : This method handle requests '/pages/about'..
Return Value : void

home()

Description : This method handles requests '/pages/home'.
Return Value : void

index()

Description : This method handles requests '/pages/index' and '/pages.
Return Value : void

model()

Description : Loads a model.
Params : $model: model's name
Return Value : mixed - model instance

view()

Description : Loads a view.
Params : $view: mixed, $data: mixed = []
Return Value : void

1.3.4 Posts Controller.php

Posts controller that handles request's prefix with 'posts'.

add()

Description : This method handles requests '/posts/addSolvedNotification/params0/params1'.

Params : $user_id: mixed, $post_id: mixed
Return Value : void

AddCommentNotification()

Description : Loads a view.
Params : $user_id: mixed, $post_id: mixed
Return Value : void

addSolvedNotification()

Description : This method handles requests '/posts/addVoteNotification/params'.
Params : $post_id: mixed
Return Value : void

addVoteNotification()

Description : This method handles requests '/posts/addVoteNotification/params'.
Params : $post_id: mixed
Return Value : void

comment()

Description : This method handles requests '/posts/comment/params'.
Params : $id: mixed
Return Value : void

delete()

Description : This method handles requests '/posts/delete/params'.
Params : $id: mixed
Return Value : void

edit()

Description : This method handles requests '/posts/edit/params'.
Params : $id: mixed
Return Value : void

getAlltags()

Description : This method handles requests '/posts/getAllTags'.
Return Value : void

index()

Description : This method handles requests '/posts/index' and '/index'.
Return Value : void

load()

Description : This method handles requests '/posts/load/params'.
Params : $page: mixed = 1
Return Value : void

markSolved()

Description : This method handles requests '/posts/markSolved'.
Return Value : void

model()

Description : Loads a model.
Return Value : mixed - model instance

report()

Description : This method handles requests '/posts/report/params'
Params : $id: mixed
Return Value : void

show()

Description : This method handles requests '/posts/show/params'.
Params : $id: mixed
Return Value : void

view()

Description : Loads a view.
Params : $view: mixed, $data: mixed = []
Return Value : void

vote()

Description : This method handles requests '/posts/vote/params0/params1'.
Params : $params0: mixed, $params1: mixed
Return Value : void

1.3.5 Users Controller.php

Users' controller that handles request's prefix with 'users'.

changePassword()

Description : This method handle srequests '/users/changePassword'.
Return Value : void

changePasswordForForget()

Description : This method handles requests '/users/changePasswordForForget'.
Return Value : void

createUserSession()

Description : This method handles requests '/users/createUserSession/params'.
Params : $user: mixed
Return Value : void

forgetPassword()

Description : This method handles requests '/users/forgetPassword'
Return Value : void

loadPosts()

Description : This method handles requests '/users/loadPosts'.
Params : $type: = 'created', $page: = 1
Return Value : void

login()

Description : This method handles requests '/users/login'.
Return Value : void

logout()

Description : This method handles requests '/users/logout'..
Return Value : void

model()

Description : Loads a model.
Return Value : void

profile()

Description : This method handles requests '/users/profile'.
Return Value : void

register()

Description : This method handles requests '/users/register'.
Return Value : void

sendChangePasswordRequest()

Description : This method handles requests '/users/sendChangePasswordRequest'.
Return Value : void

verify()

Description : This method handles requests '/users/verify'.
Return Value : void

view()

Description : Loads a view.
Params : $view: mixed, $data: mixed = []
Return Value : void

1.4 models

1.4.1 Admin.php

Admin Model works to retrieve, manipulate or delete data that is associated to the admin.

addCategory()

Description : Add a post category type to the database.
Params : $category: mixed
Return Value : bool

deleteUser()

Description : Delete a user from the database
Params : $id: mixed
Return Value : bool

getAllAdmins()

Description : This method fetches all the admins from database.
Return Value : mixed - all the admins.

getAllReports()

Description : This method fetches all the reports from database.
Return Value : mixed - all the reports from database.

getAllUsers()

Description : This method fetches all the users from database.
Return Value : mixed - all the users.

makeAdmin()

Description : Give admin ship to a specified user.
Params : $id: mixed
Return Value : bool

removeAdminShip()

Description : Remove admin ship of a specified user.
Params : $id: mixed
Return Value : bool

removeCategory()

Description : Remove a post category type from the database
Params : $category: mixed
Return Value : bool

1.4.2 Comment.php

Comment Model handle all data manipulation associated to comment.

addComment()

Description : Add a comment to the database.
Params : $id: mixed, $comment: mixed
Return Value : false|string

getComment()

Description : Fetch a comment from database specified by comment_id.
Params : $id: mixed
Return Value : mixed - fetched comment.

getPostComment()

Description : Fetch all comments from database for a post specified by post_id.
Params : $post_id: mixed

Return Value : mixed - all comments for a post.

1.4.3 Notification.php

Notification Model handle all data manipulation associated to notification.

addNotification()

Description : Add a notification to the database for a user associate to a post
Params : $user_id: mixed, $post_id: mixed, $text: mixed
Return Value : void

getAllNotification()

Description : This method fetches all the notification from database.
Return Value : mixed - all the notification.

1.4.4 Post.php

There is no function or method

1.4.5 Tag.php

Tag Model handle all data manipulation associated to Tags.

getAllTags()

Description : Fetch all distinct tags used for all existed posts.
Return Value : mixed - all distinct tags.

getTags()

Description : Fetch all the tags of a post.
Params : $post_id: mixed
Return Value : mixed - all tags of the specified post.

1.4.6 User.php

User Model works to retrieve, manipulate or delete data that is associated to the user's activity.

findUserByEmail()

Description : Find a verified user by email
Params : $email: mixed
Return Value : mixed

getCommentedPosts()

Description : Load all commented posts of current user by limit of 4 per page.
Params : $page: page number
Return Value : mixed – Votes

getCreatedPosts()

Description : Load all created posts of current user by limit of 4 per page.
Params : $page: page number
Return Value : mixed - votes

getposts()

Description : Fetch all post of current user.
Return Value : mixed - all post of current user.

getUserById()

Description : Fetch a user by id
Params : $id: mixed
Return Value : mixed - user details

getVotedPosts()

Description : Load all voted posts of current user by limit of 4 per page.
Params : $page: page number
Return Value : mixed - votes

login()

Description : Validate user email and password to login.
Params : $email: mixed, $password: mixed
Return Value : mixed

register()

Description : Register a user.
Params : $data: mixed
Return Value : bool

sendChangePasswordMail()

Description : Send change password mail to a user.
Params : $user_id: mixed, $mail: mixed
Return Value : void

sendVerifyMail()

Description : Send verify mail to specified email address
Params : $mail: mixed
Return Value : void

updatePassword()

Description : Update user password.
Params : $hashedPassword: mixed, $id: mixed = -1
Return Value : bool

verify()

Description : verify a user
Params : $user_id: mixed, $key: mixed
Return Value : bool

verifyPassword()

Description : Validate current user password.
Params : $password: mixed
Return Value : bool

1.5 helpers

1.5.1 session_helper.php

restoreSessionAvailable()

Description : Recover a session if it’s in cookie.
Return Value : void

isAcademicOfficial()

Description : Check for academic official users.
Return Value : bool

isLoggedIn()

Description : Check if a user session is already active.
Return Value : bool

security()

Description : This method redirects to login if no session is active.
Return Value : void

flash()

Description : Create a bootstrap flash alert with session.
Params : $name: mixed = '', $message: mixed = '', $class: mixed = 'alert alert-success'
Return Value : void

sendMail()

Description : Send a verify mail to specific email address with a verify link.
Params : $edu_mail: mixed, $link: mixed
Return Value : void

1.5.2 url_helper

redirect()

Description : This method redirected to a specific post.
Params : $page: page's location (e.g, 'users/login')
Return Value : void

1.6 views

Views are located in APPPATH/views. Views contain your HTML, which should never be found in your Controllers or any other class that is not specifically meant to create output. By separating your layout from your logic, you ensure that when you decide to change your layout you only have to change the views and won't have to care about the Controllers.

1.6.1 admins

It contains all the php files that will help admins to control the application.

index.php
It creates a view for admin index page.

manageAdmin.php
It creates a view for admin manageAdmin page.

manageCategory.php
It creates a view for admin manageCategory page.

managePost.php
It creates a view for admin managePost page.

manageReport.php
It creates a view for admin manageReport page.

manageUser.php
It creates a view for admin manageUsers page.

1.6.2 error

It contains the php files that will create an error view.

index.php
It creates an index view

inc
It contains all the php files which will be included by other files.

footer.php
It creates a footer view

header.php
It creates a header view

navbar.php
It creates a navbar view
sidebar.php
it creates a sidebar view for admin

pages
It creates a view of landing page, about page and main page

about.php
It creates a view about the application and developer

home.php
It creates a view of the information about posts, and also view the top solved and unsolved posts

index.php
It creates landing page of our application.

posts
It contains all the pages which related to posts.

add.php
It creates a view to create a new post.

edit.php
It creates a view to edit a post.

index.php
It creates a view to show all the posts, notifications.

list.php
It generates view for a list of posts.

post.php
It generates view for a single post.

report.php
It generates view for a report.

show.php
It makes a view for edit, delete or comment a post.

1.6.3 users

It contains all the files which are common to all users.

forgetpassword.php
It creates a view to the user of forgetpassword.

login.php
It creates a view to the user to login into the application.

profile.php
It makes a view to the user profile.

profilePostList.php
It generates a view of the users created, commented, voted posts.

register.php
It creates a view to the user to register an account.

1.7 bootstrap.php
It loads all the require files and libraries.

2 public
2.1 css
This folder contains all the CSS files which make the look and view of our application beautiful.

2.2 dir
This is the storage our application.

2.3 img
This folder contains all the img which are used in the whole project.

2.4 index.php
It is the only public php file which call an instance of core to handle a URL which is requested by a user.

Code Level

LOC : Line of code in the whole project. The Total number of code lines in whole project.
NCLOC : Non-comment line of code. The line which are not commented and this code accomplish our objective.
CLOC : Comment line of code. The line of code which are not working to calculate output.
ALOC : The average number of lines of code in each file.
DC : It means the number of comment line proportion to average code.

Table 1: Code Level

LOC	14024
NCLOC	12681
CLOC	1343
Average LOC	241
Density of Comments	10.5%

SRS & Development Mapping

Table 2: Mapping

No	Usecase	Class/file link	Method name
1	Registration	/app/controllers/UsersControllers/	register
		/app/models/User/	findUserByEmail register
		/app/views/users/register	
2	Activation Account	/app/controllers/UsersControllers/	verify
		/app/models/User/	verify sendVerifyMail
3	Create Post	/app/controllers/PostsControllers/	add
		/app/models/Post/	getCategories addPost
		/app/views/posts/add	
4	Edit Post	/app/controllers/PostsControllers/	edit
		/app/models/Post/	getCategories updatePost getPostById
		/app/models/Tag/	getTags
		/app/views/posts/edit	
5	Delete Post	/app/controllers/PostsControllers/	delete
		/app/models/Post/	getPostById deletePost
6	Comment	/app/controllers/PostsControllers/	comment
		/app/models/Comment/	addComment getComment
7	Notification	/app/controllers/PostsControllers/	addVoteNotification addCommentNotification addSolvedNotification
		/app/models/Post/	getPostById
		/app/models/Notification/	addNotification
8	Mark as Solved	/app/controllers/PostsControllers/	markSolved
		/app/models/Post/	markSolved
9	Report	/app/controllers/PostsControllers/	report

		/app/models/Post/	report
		/app/views/posts/report	
10	Voting	/app/controllers/PostsControllers/	vote
		/app/models/Post/	vote isVoted getVote
11	Search	/app/controllers/PostsControllers/	load
		/app/models/Post/	filterPostsWithLimit
		/app/views/posts/list	
12	Category Filtering	/app/controllers/PostsControllers/	load
		/app/models/Post/	filterPostsWithLimit
		/app/views/posts/list	
13	Forgot Password	/app/controllers/UsersControllers/	forgetPassword changePasswordForForget
		/app/models/User/	verify updatePassword
		/app/views/users/forgetpassword	
14	Change Password	/app/controllers/UsersControllers/	changePassword
		/app/models/User/	verifyPassword updatePassword
15	Edit Profile	/app/controllers/UsersControllers/	changePassword
		/app/models/User/	verifyPassword updatePassword
16	Trending Problem	/app/controllers/PagesControllers/	home
		/app/models/Post/	totalPostCount totalSolvedCount getTopUnsolved getTopSolved getPostById
		/app/views/pages/home	

Challenges and Future Work

Challenges

1. Making a web application for the first time was a big challenge for us.
2. Completing the project in time was challenges for us.
3. The calculation for the Top solved and unsolved problem was a challenge for us
4. Find out the same types of problem was a big challenge for us.

Future work

1. We will work on videos as well as picture.