**Feedback Management System**

****

**UTKRISHTA BHATTARAI**

**22’A’**

**Submitted to**

**Module Leader:**

­­**Sudeep Lal Bajimaya**



# **Abstract**

The whole project has been entitled with a name called Feedback Management system. All the requirements, use cases with scenario description, system architecture, the static and dynamic models and implementation details along with testing report and user manual is included in this report.

It is a web-based application and the main motto of this application is to remove current paper-based way of giving feedbacks and also motivate staffs for giving feedbacks as feedback are very important to every organization. This application also has a polling system where admin can post a poll and user can vote on that particular poll. This will help for taking decisions from users of an organization.

Talking about the technicalities of the project of the system it is made up from PHP, JavaScript, CSS along with the help of code igniter framework. MySQL database was used.

# **Acknowledgment**

I am very thankful to each and every teacher of Softwarica college of IT and e-Commerce. This project was successful with the help of teachers and effort of myself.

I am thankful to my module leader, Sudeep Lal Bajimaya for diligently supervising my project and providing me the guidelines for software analysis, modelling, implementation and documentation methods. He was not only limited to college hours, he was ready to help through social sites, email.

I am grateful to lecturer Kiran Rana for providing me the references on pertinent technical resources and motivating me to learn new thing. He was always ready to help me anywhere anytime.

I am very thankful to lecturer Bikash Ghimire, he helped me whenever I got in any programming related issues. He helped me a lot in Unit testing and in creating poll.

Likewise, I should especially mention lecturer Shyam Sundar Khatiwada, Nishant Shrestha, Niman Maharjan, Pratik Bhusal. Without them, I would not have the foundations in information technologies to build this project.

Contents

[**Abstract** 3](#_Toc15292174)

[**Acknowledgment** 4](#_Toc15292175)

[**List of keywords** 10](#_Toc15292176)

[**Chapter 1- Introduction** 11](#_Toc15292177)

[**1.1. Project Introduction** 11](#_Toc15292178)

[**1.2. Background and Justification** 11](#_Toc15292179)

[**1.3. Overview of the project** 11](#_Toc15292180)

[**1.4. Aims and Objectives** 12](#_Toc15292181)

[**Chapter 2: Analysis** 13](#_Toc15292182)

[**2.1 Introduction to analysis** 13](#_Toc15292183)

[**2.2. Analysis Methodology** 14](#_Toc15292184)

[**2.3. Feasibility study** 18](#_Toc15292185)

[**2.4. Software Requirement Specification** 18](#_Toc15292186)

[**2.4.1 Functional Requirement** 19](#_Toc15292187)

[**2.4.2. Non-functional requirements** 22](#_Toc15292188)

[**2.4.3. Moscow prioritization** 23](#_Toc15292189)

[**2.4.4. Hardware Software Specification** 24](#_Toc15292190)

[**2.5. Use case Diagram** 26](#_Toc15292191)

[**2.6. Initial Class Diagram** 29](#_Toc15292192)

[**Chapter 3: Design** 31](#_Toc15292193)

[**3.1. Introduction to Design** 31](#_Toc15292194)

[**3.2. Structural Diagram** 32](#_Toc15292195)

[**3.2.1 Final Class Diagram** 32](#_Toc15292196)

[**3.2.2. Data Flow Diagram** 34](#_Toc15292197)

[**3.3. Behavioural Diagram** 37](#_Toc15292198)

[**3.3.1. Activity Diagram** 37](#_Toc15292199)

[**3.3.2. Sequence Diagram** 40](#_Toc15292200)

[**3.4. Database Modelling** 45](#_Toc15292201)

[**3.4.1. Data Dictionary** 45](#_Toc15292202)

[**3.4.2. ER Diagram** 48](#_Toc15292203)

[**3.5. Architectural Model** 48](#_Toc15292204)

[**3.6. UI Modelling** 50](#_Toc15292205)

[**Chapter 4: Implementation** 57](#_Toc15292206)

[**4.1. Introduction** 57](#_Toc15292207)

[**4.2. Language Used** 57](#_Toc15292208)

[**4.3. Tools Used** 57](#_Toc15292209)

[**Chapter 5: Testing** 67](#_Toc15292210)

[**5.1. Black Box Testing** 67](#_Toc15292211)

[**5.2. Unit testing** 78](#_Toc15292212)

[**Chapter 6: Others project issues** 81](#_Toc15292213)

[**Chapter 7: Conclusion** 87](#_Toc15292214)

[**Chapter 8: Reference and Bibliography** 88](#_Toc15292215)

[**Chapter 9: Appendix** 90](#_Toc15292216)

[Figure 1: Rich Picture 15](#_Toc15292096)

[Figure 2: Problem based root definition 16](#_Toc15292097)

[Figure 3:Conceptual Model 17](#_Toc15292098)

[Figure 4:Use case diagram 26](#_Toc15292099)

[Figure 5: use case diagram 27](#_Toc15292100)

[Figure 6: use case diagram 28](#_Toc15292101)

[Figure 7: Initial Class diagram 30](#_Toc15292102)

[Figure 8: Final Class Diagram 33](#_Toc15292103)

[Figure 9: DFD for admin 35](#_Toc15292104)

[Figure 10: DFD for user 36](#_Toc15292105)

[Figure 11: Activity Diagram 39](#_Toc15292106)

[Figure 12: Sequence diagram for login 42](#_Toc15292107)

[Figure 13: Sequence diagram for feedback 43](#_Toc15292108)

[Figure 14: Sequence diagram for registration 43](#_Toc15292109)

[Figure 15: Sequence diagram for voting on poll 44](#_Toc15292110)

[Figure 16: ER diagram 48](#_Toc15292111)

[Figure 17: Three tier architecture 49](#_Toc15292112)

[Figure 18: prototype for login 50](#_Toc15292113)

[Figure 19: prototype for registration 50](#_Toc15292114)

[Figure 20: prototype for dashboard user 51](#_Toc15292115)

[Figure 21: prototype for feedback 51](#_Toc15292116)

[Figure 22: prototype for staff profile 52](#_Toc15292117)

[Figure 23: prototype for individual staff profile 52](#_Toc15292118)

[Figure 24: prototype for community forum 53](#_Toc15292119)

[Figure 25:: prototype for Poll 53](#_Toc15292120)

[Figure 26: prototype for view notices 54](#_Toc15292121)

[Figure 27: prototype for add department 54](#_Toc15292122)

[Figure 28: prototype for create poll 55](#_Toc15292123)

[Figure 29: prototype for add notices 56](#_Toc15292124)

[Figure 30: prototype for feedback view 56](#_Toc15292125)

[Figure 31: prototype for adding staffs 56](#_Toc15292126)

[Figure 32: login 58](#_Toc15292127)

[Figure 33: view for login 58](#_Toc15292128)

[Figure 34 view for login 59](#_Toc15292129)

[Figure 35: controller for login 59](#_Toc15292130)

[Figure 36:controller for login 60](#_Toc15292131)

[Figure 37:model for login 60](#_Toc15292132)

[Figure 38:model for login 61](#_Toc15292133)

[Figure 39: registration 61](#_Toc15292134)

[Figure 40: view for registration 62](#_Toc15292135)

[Figure 41: view for registration 62](#_Toc15292136)

[Figure 42: view for registration 63](#_Toc15292137)

[Figure 43: view for registration 63](#_Toc15292138)

[Figure 44: controller for registration 64](#_Toc15292139)

[Figure 45: controller for registration 64](#_Toc15292140)

[Figure 46: controller for registration 65](#_Toc15292141)

[Figure 47: controller for registration 65](#_Toc15292142)

[Figure 48: model for registration 66](#_Toc15292143)

[Figure 49: Black box testing for community forum 68](#_Toc15292144)

[Figure 50: Black box testing for account deletion 69](#_Toc15292145)

[Figure 51: Black box testing for login 70](#_Toc15292146)

[Figure 52: Black box testing for feedback 71](#_Toc15292147)

[Figure 53: Black box testing for showing details 72](#_Toc15292148)

[Figure 54: Black box testing for logout 73](#_Toc15292149)

[Figure 55: Black box testing for saving posts 74](#_Toc15292150)

[Figure 56: Black box testing for voting on poll 75](#_Toc15292151)

[Figure 57: Black box testing for registration 76](#_Toc15292152)

[Figure 58: Black box testing for updating details 77](#_Toc15292153)

[Figure 59: unit testing 1 78](#_Toc15292154)

[Figure 60: unit testing 2 78](#_Toc15292155)

[Figure 61: unit testing 3 79](#_Toc15292156)

[Figure 62: Output of unit testing 80](#_Toc15292157)

[Figure 63: Configuration management: 84](#_Toc15292158)

[Figure 64: User manual 85](#_Toc15292159)

[Figure 65: User manual 86](#_Toc15292160)

[Figure 66:User manual 87](#_Toc15292161)

[Figure 67: Poll UI 90](#_Toc15292162)

[Figure 68: Model for poll 92](#_Toc15292163)

[Figure 69: Controller for Poll 93](#_Toc15292164)

[Figure 70: View for Poll 94](#_Toc15292165)

[Figure 71: UI for community forum 94](#_Toc15292166)

[Figure 72: Model for community forum 95](#_Toc15292167)

[Figure 73: Controller for community forum 96](#_Toc15292168)

[Figure 74: View for community forum 97](#_Toc15292169)

[Figure 75: UI for post 98](#_Toc15292170)

[Figure 76: Model for post 98](#_Toc15292171)

[Figure 77: Controller for posts 99](#_Toc15292172)

[Figure 78: View for posts 99](#_Toc15292173)

[Table 1: Functional Requirement 20](#_Toc15292077)

[Table 2: Non Functional Requirement 23](#_Toc15292078)

[Table 3: MOSCOW Prioritization 23](#_Toc15292079)

[Table 4: MOSCOW prioritization for Admin 24](#_Toc15292080)

[Table 5: Non-functional Requirement 24](#_Toc15292081)

[Table 6: Potential classes and methods 29](#_Toc15292082)

[Table 7: Data dictionary for user 45](#_Toc15292083)

[Table 8: Data dictionary for department 45](#_Toc15292084)

[Table 9: Data dictionary for staff 45](#_Toc15292085)

[Table 10:Data dictionary for Likes 46](#_Toc15292086)

[Table 11: Data dictionary for Poll create 46](#_Toc15292087)

[Table 12: Data dictionary for vote on poll 46](#_Toc15292088)

[Table 13: Data dictionary for feedback 46](#_Toc15292089)

[Table 14: Data dictionary for question 47](#_Toc15292090)

[Table 15: Data dictionary for answer 47](#_Toc15292091)

[Table 16: Black box testing 67](#_Toc15292092)

[Table 17: Likelihood table 81](#_Toc15292093)

[Table 18: consequences table 82](#_Toc15292094)

[Table 19: Risks table 82](#_Toc15292095)

# **List of keywords**

Efficient, Dissatisfaction, chunk, queries, authority, methodologies, deployment, framework, estimation, demonstrate, criticism, feasibility, calamities, milestone, visualisation, consequences, website, budget.

# **Chapter 1- Introduction**

## **1.1. Project Introduction**

This project is about feedback management system. As stated in proposal, this web application will help in collecting feedbacks, response, criticism and decision from the users. This system has been introduced to remove paper work and make feedback collecting process easier and much efficient. Waterfall methodology has been used in the process of developing this system. The whole project is to be completed within two months.

## **1.2. Background and Justification**

Feedback is essential to the working and survival of all the regulatory mechanisms found throughout living and non-living nature, and in man-made systems such as education system and economy **(Business Dictionary,2019)**. Any organisation wants its work to be done in an efficient way, for any organisation to grow more it needs feedback from its staffs and customers, also to know about the behaviour of staffs collecting the feedback from different people is necessary. For e.g. in any college there are various staffs, teachers and students. Those feedback helps a lot to grow character of an individual and also know satisfaction and dis-satisfaction of students and parents. These feedbacks can help a lot to analyse staff’s   
behaviour and monitor them if necessary.

The complete web application has been designed and it will soon remove paper-based way of collecting feedback from the users. Feedback management system will not only facilitate in feedback collection, also, it will help the staffs of an organisation to collaborate in community forum and when some decision is to be made admin can simply create a poll where users can vote.

## **1.3. Overview of the project**

Complete web application for collecting feedbacks along with other added features has been made. Users can vote on poll, give feedbacks, chat in community forum, like staffs and view their details and from other side admin can view those which will help in decision making.

**Features of the project:**

* Displaying list of all staffs working in an organisation according to their department/category.
* Separate profile of each staffs and brief introduction about them.
* Allow the users to rate on any staff's/people profile.
* Provide separate feedback for any staffs/people registered into the system.
* Allow users to register into the system and login into the system using the credentials.
* Allowing users to chat with other users in a community forum.
* Allowing users to update delete their account if necessary.
* Allowing the system admin/Controller to register any staffs in the system.
* Allowing Admin to view feedbacks received and reviewing them.

## **1.4. Aims and Objectives**

The major aim of the project is to:

* Build a web application that helps in facilitating feedback collection process easier and much efficient. Also, to remove traditional paper-based feedback collection process.

Main objectives of the project are listed below.

**Project Objectives**

* To build the website being within the budget and under given time boundary.
* To build an accessible, user-friendly and reliable website.
* To get rid of loss of feedbacks.
* To ensure security of users using the website.
* To strengthen the relationship between various staffs of an organisation.
* To get rid of old paper-based feedback process and implement online system for collecting feedbacks.
* Making admin/controller to easily view feedback and review them accordingly.
* To ensure user satisfaction and make the feedback process efficient.

**Personal Objectives**

* To acquire knowledge on various programming language.
* To get knowledge about Software development methodologies and ways to implement them.
* To know how a complete website is designed and executed.
* To develop skills and problem-solving strength.

# **Chapter 2: Analysis**

## **2.1 Introduction to analysis**

Any detailed examination of anything complex in order to understand nature or to determine its essential features is called analysis **(Merriam-webster.com, 2019)**. In other words, it is the process of breaking down complex thing into small parts that can be understand easily. For any project it is important to understand what is happening and how the things will work. This all is done through a process called analysis. It is an important phase in Software development life cycle. In this phase the requirements are captured accordingly, in this phase a customer/client expresses what he wants the system to do in order to fulfil his requirements. Technical person or the person from developer team analyse each requirement provided by the users/clients and make sure that the requirements can be fulfilled in the software in the system without affecting its system functionality and without causing any problems.

Analysis is done for following reasons:

**Studying the current system**

Any project is initiated because there might be problem with the existing system. In order to make the current system efficient and make it free from errors and bugs new system needs to be designed. Analysis will help in collecting the facts from existing users also it will help to collect information about current system boundary, details of the system and the people affected by the system.

**Defining new system objectives**

Analysis will help to prioritize user requirements. It will give a clear idea of what the system should be and will make the developer team (Technical person) clear about the needs of user/client from the system. Analysis will help in understanding each and every aspects of current system and indicate how the things can be made efficient with the deployment of new system.

## **2.2. Analysis Methodology**

The literal meaning of methodology is theoretical, systematic study of the methods and ways applied to any fields of study. Analysis refers to studying and methodology refers to the systematic approach, so combining these, analysis methodology means the methods of studying the system by help of various resources that will help in making the study efficient and less time-consuming. There are mainly 9 analysis methodology that can be used in studying the system. Among them I will use Soft System Methodology. Soft approach involves six main steps.

* Analyse current system and produce a rich picture. A rich picture is somehow like a Data flow diagram but the specific process is not followed while drawing a rich picture.
* Second step is to define a root definition of significant parts of the system. Root definition can be issue based and primary task based. Problems faced by the current system is shown in this diagram.
* Third step is to conceptual diagram. It is a diagram in which an ideal system is shown, it models on how the data should flow and it gives an idea on how the system should be and function.
* Fourth step is to compare the conceptual system with the actual problem.
* Fifth step is to select feasible options that assist for easy development of the system, both developer and client team discuss on this topic.
* Sixth and the final step is to implement the new system.

**Advantages**

* It helps to structure a complex organisational and political situation.
* It focuses on user involvement rather than technical specification.
* It allows developer to roll back and repeat the phases if necessary.

**Disadvantages**

* It has no definite technique modelling tool, so it is hard to know about the correctness of diagrams.
* Since no specific rules and process are followed it is impossible to track the progress.
* This can sometime take too long time to reach an agreement, so there is chance that the software will not get delivered.

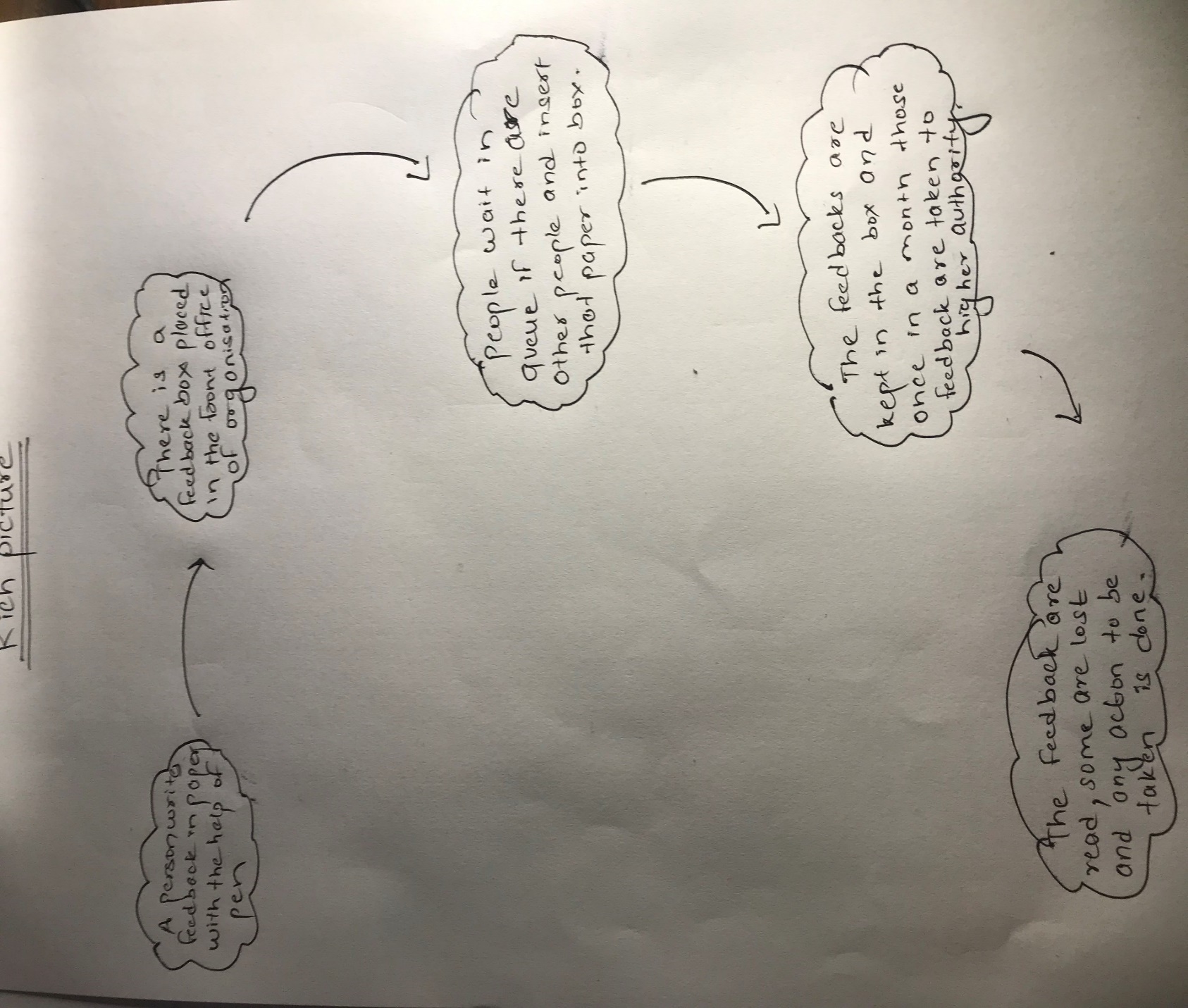


Figure : Rich Picture

This is task-based root definition. In this diagram current system is described.

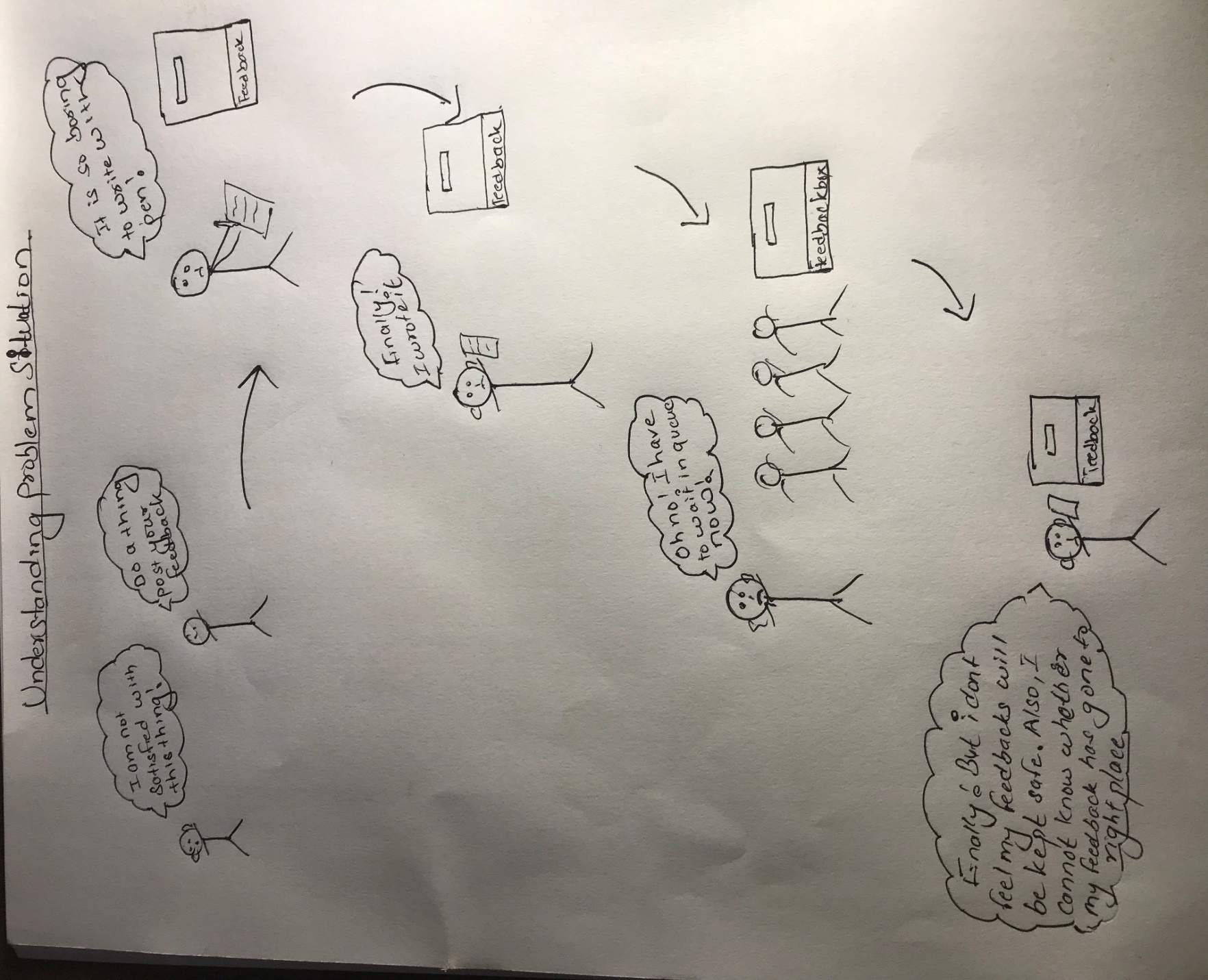


Figure : Problem based root definition

This is problem-based root definition. Problem faced by organisation with the use of current system is shown in this diagram.



Figure :Conceptual Model

This diagram models on how the new system should be. How data should flow and what process should be performed. It is now compared in real world and if the system seems feasible. It is implemented.

## **2.3. Feasibility study**

It is a study which involves an estimation of the level of proficiency mandatory for a project that can provide qualitative, qualitive assessments of various other resources, identification of mandatory points, general timetable and general cost estimation. **(My Accounting Course, 2019)**. This type of study helps for determining the possibility of an idea. And also, it helps in confirming that the project is legally, technically, socially, economically feasible. It also gives an idea on whether a project is worth the investment. Mainly there are five types of feasibility that should be done:

1. Technical feasibility: It gives a clear idea of capability of the developer/technical team of any organisation.
2. Economic feasibility: It gives a clear idea on the economic status of the project. It also gives an idea about the economic benefit that the project will give after the project is completed. So, in this phase a decision on whether a project should be initiated or not.
3. Legal feasibility: This type of study gives an information on whether a project is legally valid. This type of study gives an idea on legal boundary of the system.
4. Operational feasibility: It studies on how the project is going to solve the current problem faced by an organisation.
5. Scheduling feasibility. It is the most important study that needs to be done before initiating a project. It will give a clear idea on whether the project will get completed in given time frame and complete the given requirements.

## **2.4. Software Requirement Specification**

Software requirement specification is an explanation of system that is to be developed. SRS helps to lay out non-functional and functional requirement and this involves a set of use cases that explains user interaction that the software must have for the user satisfaction. It helps in maintaining an agreement between client and developer team on how the software will be and how a software should function. It helps in providing a realistic estimation on product costs, risks, schedules etc. It can help to prevent software project failure by identifying risk in early phase and the ways to mitigate them. Software Requirement specification helps to minimize the efforts and time that developers require a specific goal and also help in minimizing development cost. Software requirement specification serves other purposes. They are listed below:

* It provides feedback which guarantee to the client that the developer team understand the problems and issues that needs to be solved and the behaviour of the system that are necessary to address those problems.
* It helps to make agreement between user and the developer, and also it helps user to determine whether the stated requirements are fulfilled.
* It helps in determining the requirements of system which further helps in rough estimation of time and cost.

### **2.4.1 Functional Requirement**

It is a document which provides list of the operations and task or activities that the system must be able to perform. Functional requirements should include description of data to be entered into the system, operations performed by each interface, system reports or other outputs.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N. | Title | Description | Rational | Dependencies |
| FR-001 | Login into system | Admin as well as normal users can login with the credentials provided | Authentication of username and password | FR-002 |
| FR-002 | Registration | Users can register themselves in the system by providing necessary credentials | For being registered into the system. |  |
| FR-003 | View Staffs | Users can view staffs according to different category | To view staffs and also to assist in deleting, updating | FR-001 |
| FR-004 | Update Details | Users can update their credentials after being logged in to the system. | To update details if necessary | FR-001 |
| FR-005 | Delete Account | Users can delete their account and take break from the system if they want | To delete account permanently if needed. | FR-001 |
| FR-006 | View Individual Staffs account | User can check individual staffs accounts that are added in the system and get some information | To view account and edit if necessary | FR-001 |
| FR-007 | Rate staffs | User can give stars to individual staffs by going on their profile | To give rating so that other users can view highest rated and least rated staffs | FR-001 |
| FR-008 | Give Feedback | Users can log into the system, select the staff and give feedback | To give feedback which is the main thing to do on this system | FR-001 |
| FR-009 | View other important notices | Users once logged in the system can view notices posted by the admin. | To view notices that are posted by admin | FR-001 |
| FR-010 | Post Query | Users can post query in community forum | To post query if user faces any problems | FR-001 |
| FR-011 | Reply To posted Queries | Users can reply to others query too. | To reply to questions if some users have specific knowledge on that topic. | FR-001 |

Table : Functional Requirement

**Functional requirements for admin**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N. | Title | Description | Rational | Dependencies |
| FR-012 | Login into system | Admin can login to the system by entering username and password | Authentication of Username and password | - |
| FR-013 | Update User details | Admin can update user details if required | To update user details if user provide any error details. | FR-012 |
| FR-014 | Delete User | Admin can also remove user from the system if he/she violates the rules. | To permanently delete user account if any user violates the rules. | FR-012 |
| FR-015 | Add Staffs | Admin can add staffs according to various category | To add staffs so that user can view in their UI. | FR-012 |
| FR-016 | Remove Staffs | Admin can remove staffs if necessary | To remove details from system if required. | FR-012 |
| FR-017 | View Feedback | Admin can view feedbacks posted by various users. | To view feedbacks and take actions. | FR-012 |
| FR-018 | Add notices | Admin can add notice so that the user can view them | To post some important notices. | FR-012 |
| FR-019 | Create Poll | Admin can create poll | To get knowledge on user likes and dislikes. | FR-012 |

### **2.4.2. Non-functional requirements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S.N. | Title | Description | Rational | Dependencies |
| NF-001 | Security | Data of users as well as the system must be kept secure from theft | Conserving security | NF-003 |
| NF-002 | Availability | System should not be down. It should function 24\*7 | Conserving availability | NF-003 |
| NF-003 | Reliability | System must be reliable. User input and output must be handled properly | Data loss and incorrect data must be avoided | NF-001, NF-002, NF-005 |
| NF-004 | Testability | System must be tested thoroughly | Maintain Testability and correctness of software |  |
| NF-005 | Scalability | It should fit in any conditions. It should function even if users of system grow rapidly and in large number. | For expansion of business | NF-003 |
| NF-006 | Maintainability | System components must be maintained easily. | Solving small problems and fixing bugs. | NF-001 |
| NF-007 | Serviceability | Help service should be provided by developer team if necessary. | Helping users to solve the problems that they are facing. |  |
| NF-008 | Performance | Software should be optimized and less resources should be used. | Helps to boost performance and response time. | NF-001 |
| NF-009 | Recoverability | Data should be recovered in case of accidental deletion and damage. | Ensuring availability | NF-003 |
| NF-010 | Interoperability | Data are allowed for unrestricted sharing between different system. | To share information and resources. |  |

Table : Non Functional Requirement

### **2.4.3. Moscow prioritization**

In a DSDM project time is fixed. It is very necessary to understand the importance of work that needs to be done in specific order to meet deadlines. Moscow is a tool or technique that helps to understand and manage priorities in which following letters stand for:

* **M**ust Have
* **S**hould Have
* **C**ould Have
* **W**on’t Have this time

Prioritization of requirement of Feedback management system is given below:

For user

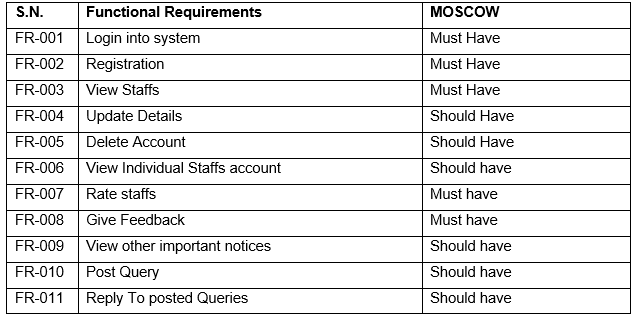


Table : MOSCOW Prioritization

MOSCOW prioritization for Admin

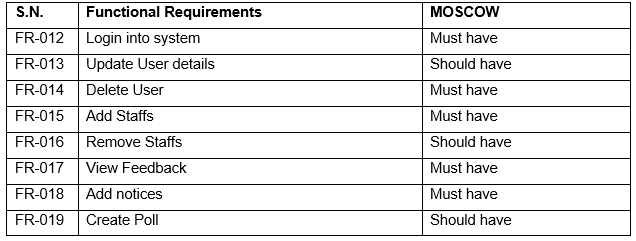


Table : MOSCOW prioritization for Admin

**Non-Functional Requirements**

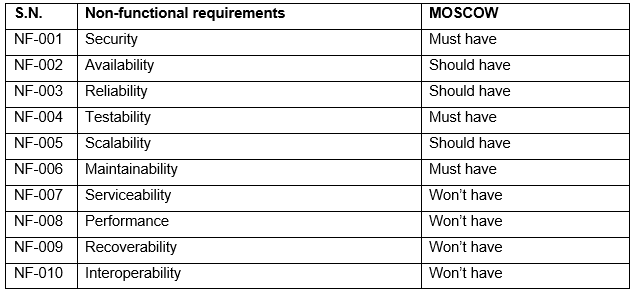


Table : Non-functional Requirement

### **2.4.4. Hardware Software Specification**

For development of any application/software there is use of Hardware and Software. Similarly, for this project use of Hardware and software is done. Following software and hardware are used in order to build the system

**Software**

IDE: Php storm

Xampp, Gitbash are used for programming purpose and GitHub is used for uploading the files in online repository so that we can have their backups.

**Hardware**

Ram used: 6 GB

Processor: Intel Core i7-2640M @2.80Ghz

Graphics: NVIDIA GeForce 520M

The system is deployed in server since the app is web based. And for accessing the system Browser like Google chrome and Mozilla Firefox with newest version must be used to use all functionalities properly.

## **2.5. Use case Diagram**

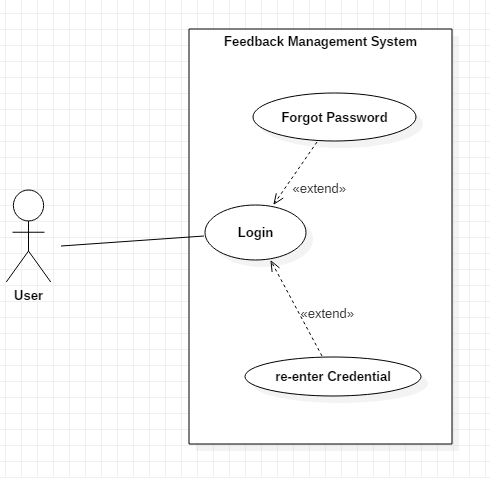
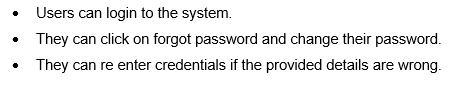


Figure :Use case diagram



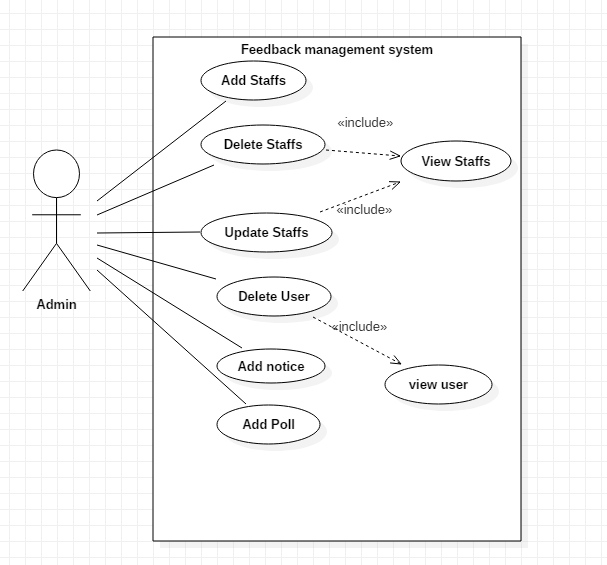
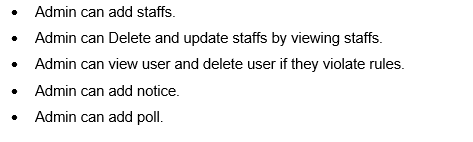


Figure : use case diagram



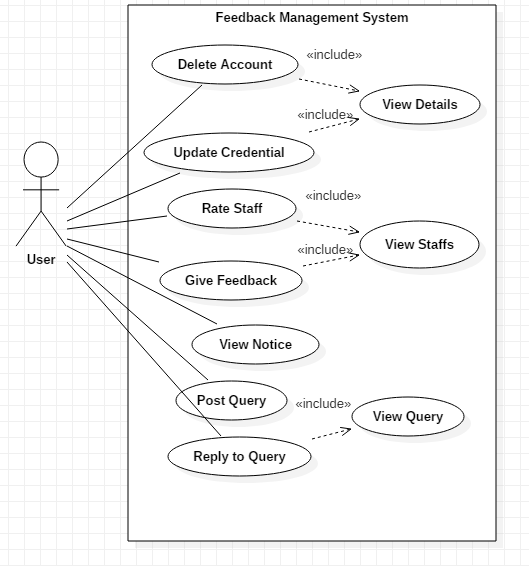


Figure : use case diagram



## **2.6. Initial Class Diagram**

Softwarica college is using paper-based feedback collecting system in order to collect feedbacks. But recently they are facing some problem with the current paper-based system because the existing data aren’t used properly. In order to solve the existing problem an online feedback management system is to be made. The new application should be web based and it should Allow Admin to add Staffs/Users that are in organisation. Users should be able to register on the system, login to the system. Following Functionalities should be provided

* Users should be able to view delete and update their given credentials.
* Admin should be able to add staffs, delete staffs.
* Admin should be able to add notices, delete notices and view notices.
* Users should be able to view staff’s profile and provide feedback.
* Users should be able to cast their votes in poll and reply and post queries.
* User should be able to give reviews and rate staffs.

From above scenario we can classify potential class and methods as

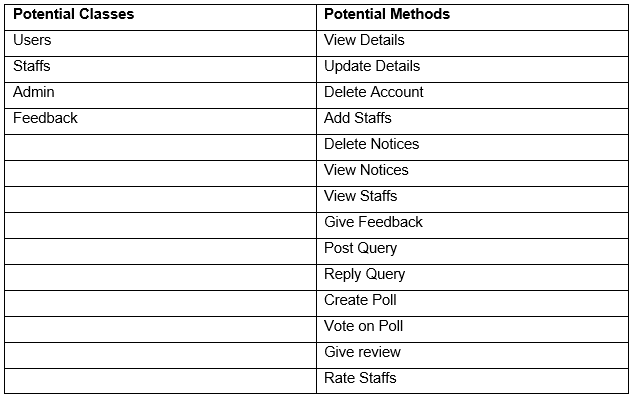


Table : Potential classes and methods

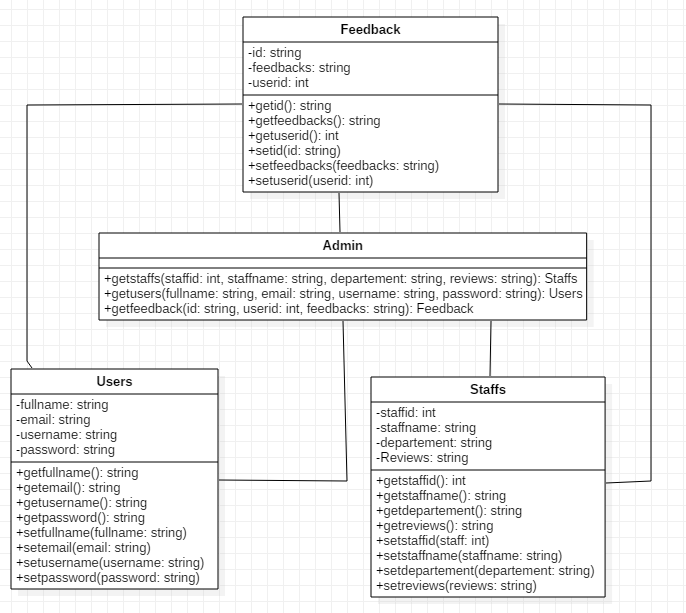


Figure : Initial Class diagram

­­

# **Chapter 3: Design**

## **3.1. Introduction to Design**

Design is the process in which user requirements are transformed into some useful forms which further helps a programmer in Coding or Implementation phase. Design is done in software development because it helps in preventing redundancy increases reusability and it is the best approach to mitigate the risk that we have no idea about.

## **3.2. Structural Diagram**

### **3.2.1 Final Class Diagram**

Class diagram in UML (Unified Modelling Language) is a static diagram that describes the system structure by demonstrating the system classes, attributes, operations and relationship that exists among the objects.

**Justification for the approach taken**

For following reasons, I have made use of class diagram:

1. It shows static structure and classifiers in a system.
2. It is useful for developer as well as team member too because this diagram shows different class and object and relationship that exists between them.
3. It also describes the functionalities done by the system.

**Notations Used**

**Class**



**Attribute**



**Generalization**



**Association**



**Multiplicity**



**Aggregation**



**Final Class Diagram**

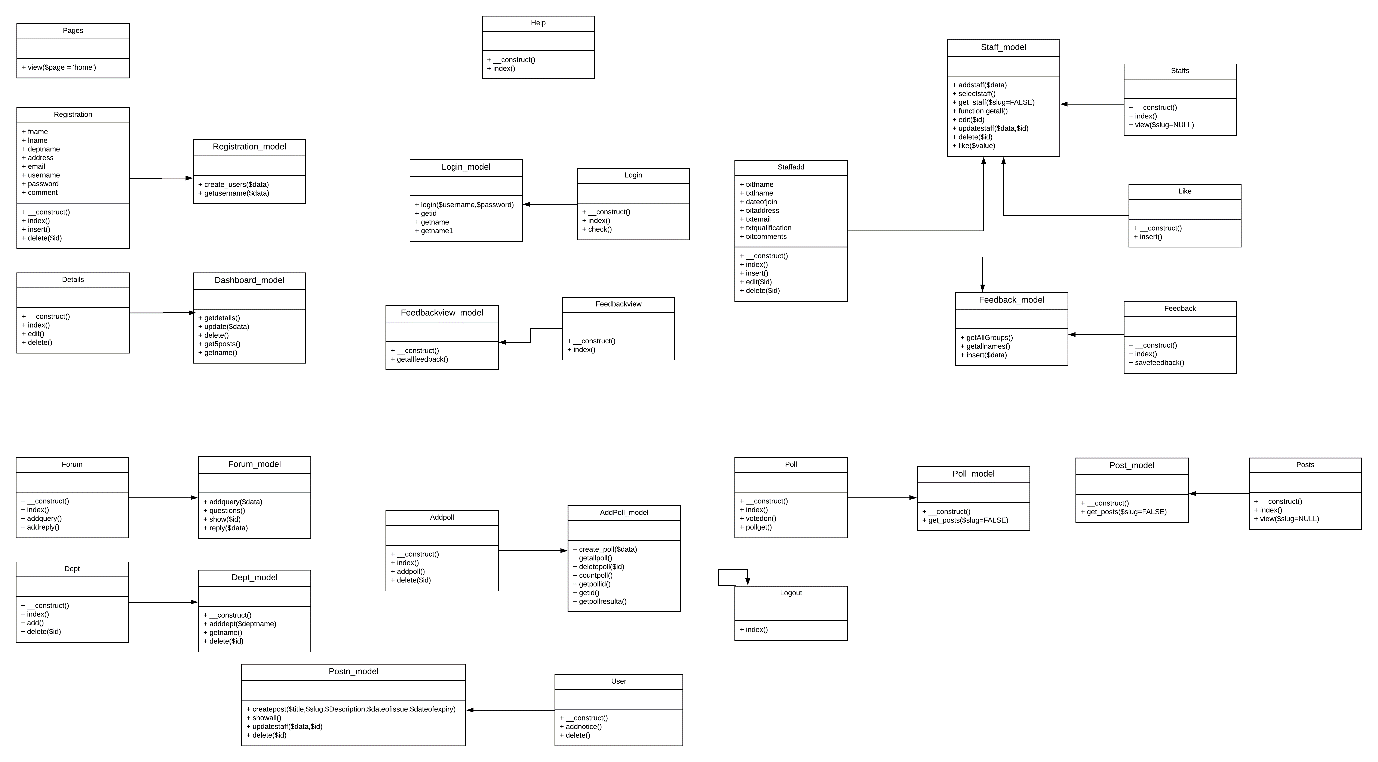
****

Figure : Final Class Diagram

**Description of Diagram**

Since the diagram is for CI framework, only models and controllers are to be shown. So, in this diagram models and controllers of both admin and users are shown.

### **3.2.2. Data Flow Diagram**

It is a diagram that models the flow of information for any information system or process. This modelling techniques helps to show the process that are involved in a system to make transfer of data from the input to the database (File Storage) and report production.

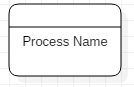
**Justification for the approach taken**

DFD helps in representing process or functions which manipulate, captures, stores and allocate data between the systems. For following reasons, I have used DFD diagram:

1. It helps in describing the logical flow of system.
2. This diagram uses simple notation so it easily understandable.
3. It also helps in creation of automated and manual system requirements.

**Notations Used**

**Process**



**Database**



**Data Flow**



**External Entity**



**Data Flow Diagram**

**For Admin**

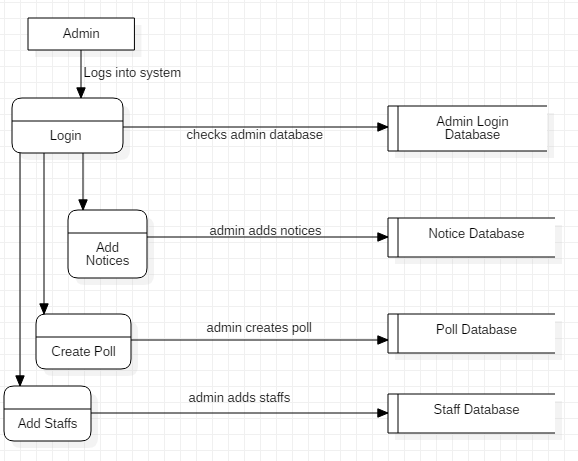


Figure : DFD for admin

This diagram is for admin and this diagram shows the process that can be done by admin. Admin can log into the system. After the admin is logged in to the system, admin can add notices, create poll, add staffs, add department and edit and delete them if necessary. The inputs are saved into respective database.

**For User**

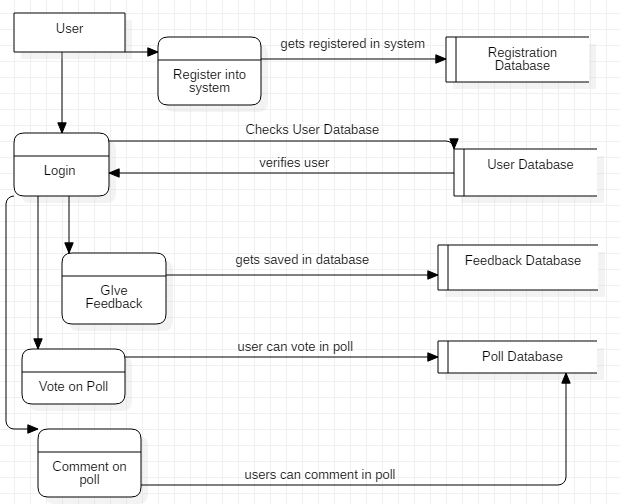


Figure : DFD for user

Users in other hand can register into the system. Only after the user register into the system, the user will have credentials to login to the system. Users should first login into the system. The credentials are checked from the database and only after the given credential matches with one of the databases, the user is logged in and he can use other functionalities. User can give feedbacks, vote on poll and chat in community forum. Similarly, the input is saved into respective databases.

## **3.3. Behavioural Diagram**

### **3.3.1. Activity Diagram**

It is an important diagram that models the dynamic aspects of the system. It is a kind of flowchart that helps to represent the flow from one activity to another activity. The flow can be sequential, concurrent or branched.

**Justification for the approach taken**

For Following reasons, I have made use of activity diagram:

1. It helps in describing the activity flow of system.
2. It describes the sequence among different activity.
3. It defines branched, concurrent, sequential flow of system.

**Notations Used**

**Start Point/ Initial**



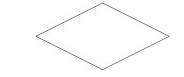
**Activity**



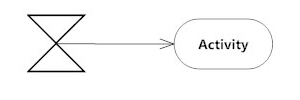
**Action Flow**



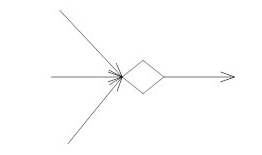
**Decision**



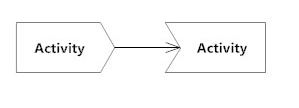
**Time Event**



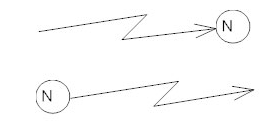
**Merge**



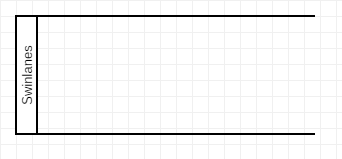
**Message sent and received**



**Interrupting Edge**



**Swim lanes**



**Activity Diagram**

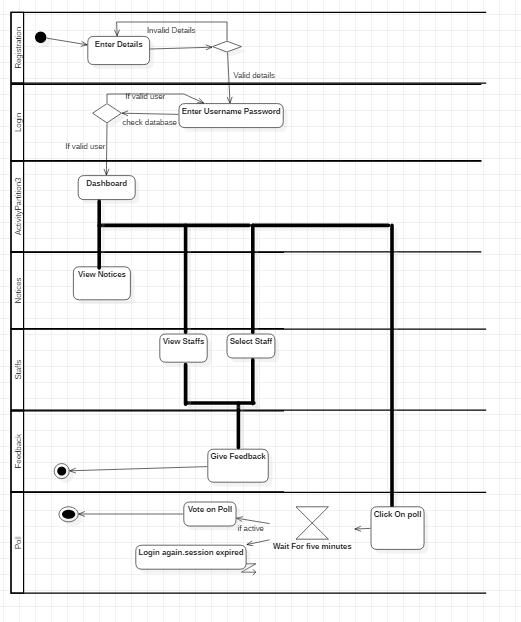


Figure : Activity Diagram

Activity diagram is used as behavioural diagram and I have used this diagram as to describe the activity flow of the system. All activity that can be done by user are modelled in this diagram. Also, time sequence, initial, end phase and interrupts are shown in this diagram. Decisions node and what decisions are to be made upon different conditions are shown in this diagram. The fork node is shown to show the task that are parallelly accessible. For example, from dashboard users have multiple option, he can view notices, view staffs and do other stuffs.

### **3.3.2. Sequence Diagram**

Sequence diagram are interaction diagrams that represents how different operations are carried out. They help to capture interaction among and between objects in situation of a collaboration.

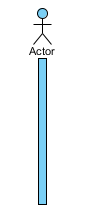
**Justification of approach taken**

For following reasons, I have made use of sequence diagram:

1. It models high-level interaction between objects that are active in a system.
2. It shows the interaction between objects within relationship that realizes a process.
3. It helps in understanding and the detailed functionality of an existing or upcoming scenario and also helps to plan them.

**Notations Used**

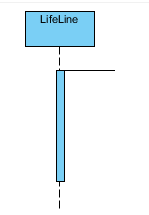
**Actor**



**Lifeline**



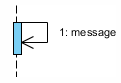
**Activation**



**Call Message**



**Self-Message**



**Sequence Diagram**

**For Login**

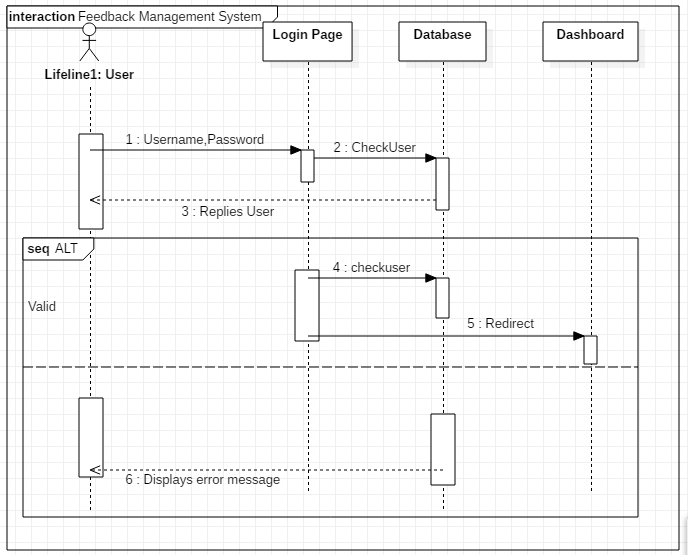


Figure : Sequence diagram for login

The login of user is checked after user pass credential in the form. The credentials are taken as input and they are checked in database. If the details are correct, he/she is redirected to dashboard, otherwise error message is shown.

**For Feedback**

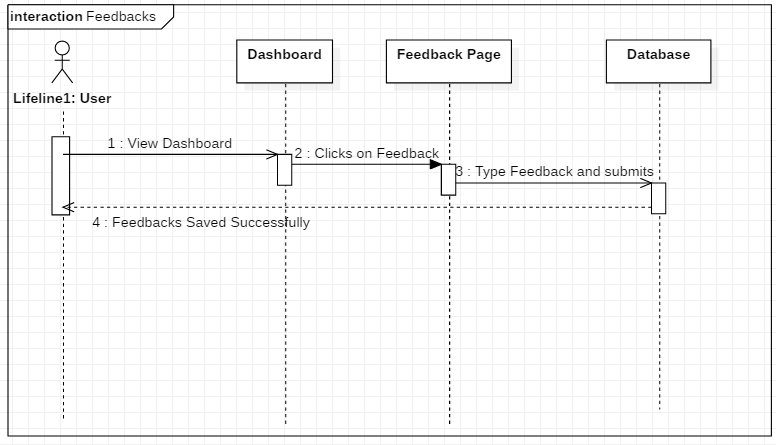


Figure : Sequence diagram for feedback

Users can easily access feedback page and they can select staff and write whatever they want to. After submission the feedbacks are successfully saved to database.

**For Registration**

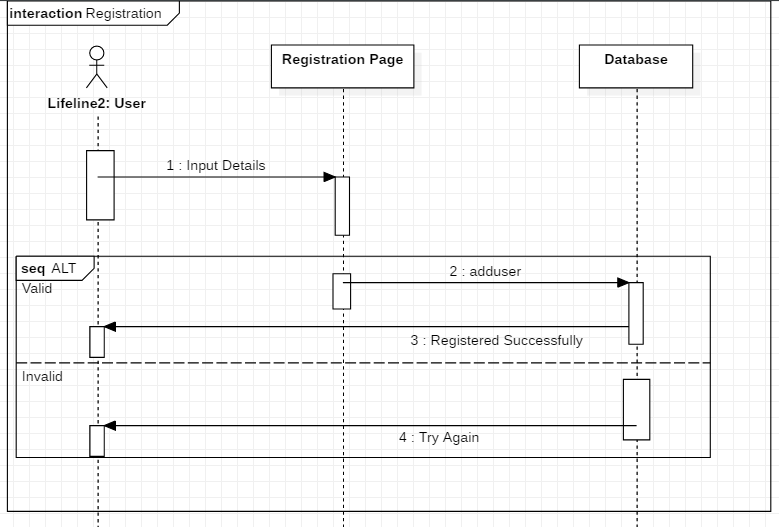


Figure : Sequence diagram for registration

During registration, if incorrect details are passed an error message is shown and the users should fill up the form again. Valid credentials are saved into database which user can use for login

**For Voting on Poll**

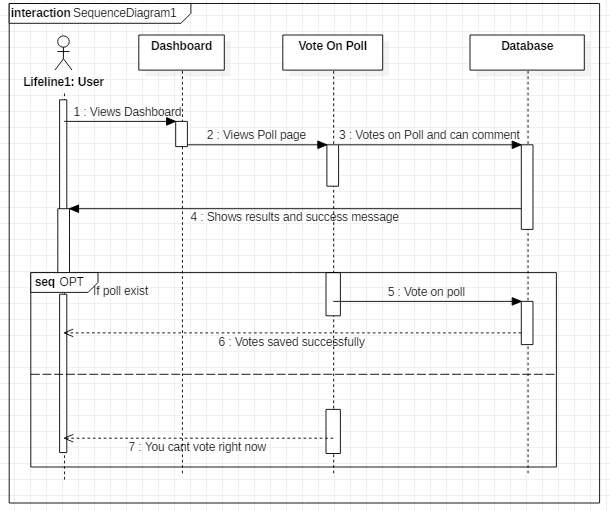


Figure : Sequence diagram for voting on poll

When user navigates to vote on poll, the database is checked. If any poll is active and is unvoted by the user being logged in, he/she can vote otherwise the user won’t be able to vote in same poll twice.

## **3.4. Database Modelling**

### **3.4.1. Data Dictionary**

Data dictionary is a type of table which describe the contents, structure, format and the relationship that exists among the elements. They can be used for manipulating the database and controlling access. The users of database don’t really have access to data dictionary it is all managed by Database Administrator. Data dictionary for my database is given below:

**User**

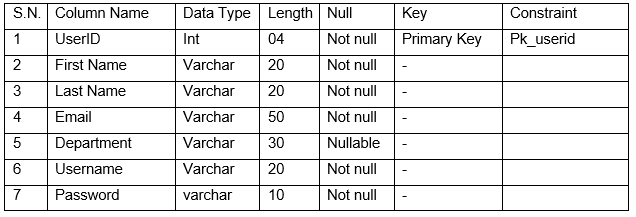


Table : Data dictionary for user

**Department**

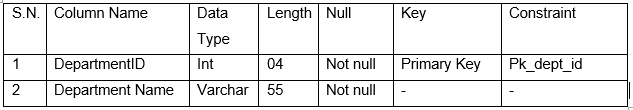


Table : Data dictionary for department

**Staff**

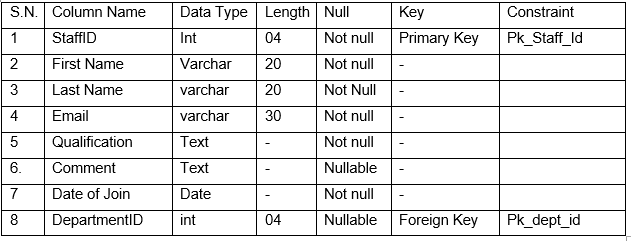


Table : Data dictionary for staff

**Likes**

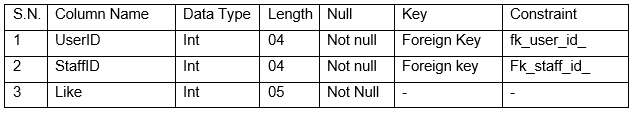


Table :Data dictionary for Likes

**Poll Create**

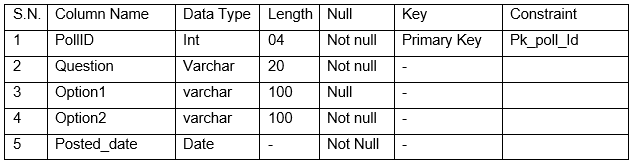


Table : Data dictionary for Poll create

**Vote on Poll**

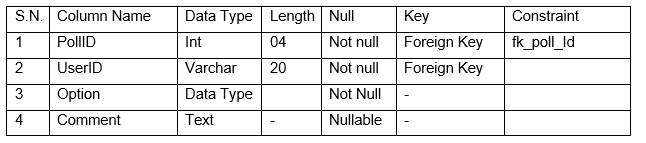


Table : Data dictionary for vote on poll

**Feedback**

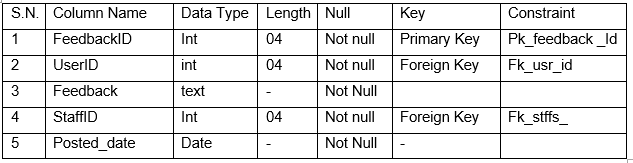


Table : Data dictionary for feedback

**Question**

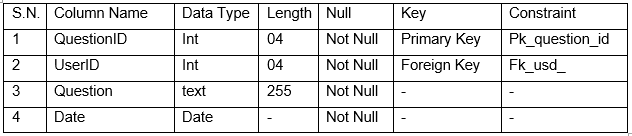


Table : Data dictionary for question

**Answer**

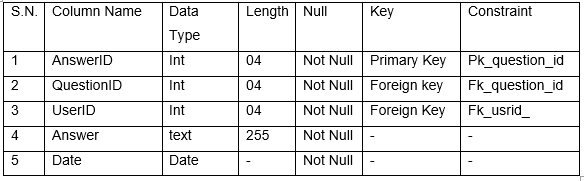


Table : Data dictionary for answer

### **3.4.2. ER Diagram**

It is a structural diagram used in database design which make uses of various symbols and connectors that shows the major entities within the system scope and the inter-relationship between objects. ER diagram are straightforward and shows relationships between entities in clear way and ER diagram can also be easily converted to other data model.

ER diagram for my database is given below:

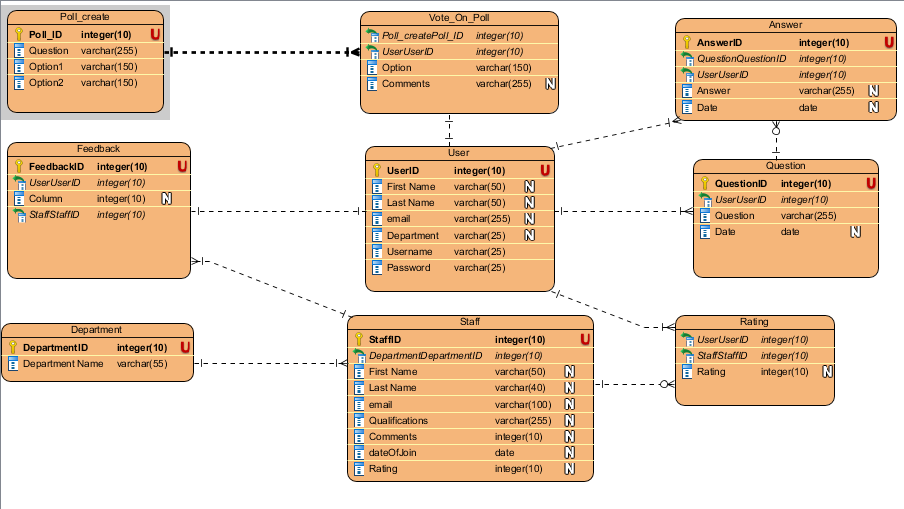


Figure : ER diagram

Above listed ER diagram is the Final ER diagram. Total of eight tables can be seen.

## **3.5. Architectural Model**

I have made use of three tier architecture. It is a type of software architecture which consist three tiers known as Presentation tier, Application Tier and Data Tier. Presentation tier is basically called template as it consists of user interface. They are accessed through web browser or web-based application which displays information and contents that are useful for a user. Application tier contains the functions business logic which drives to application core capabilities. It is written in c#, Python, php, C++. Lastly Data tier consists of the data storage system/database and data access layer. MySQL, Oracle, PostgreSQL are some examples of this system.

Benefits of using three tier architecture are:

1. It gives ability to make changes on working of one tier without making changes on another tier.
2. It allows reliability and independence of underlying services and servers.
3. It supports easy maintenance of coding and managing code base and business logic separately so that change in any tier won’t affect other.

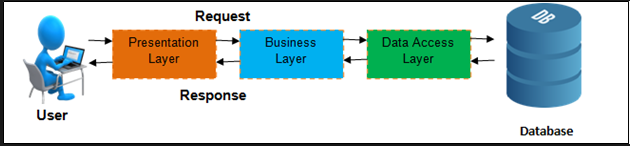


Figure : Three tier architecture

## **3.6. UI Modelling**

This is a type of modelling technique which shows how a User Interface will look like and how it would function. This is very important thing to do because if helps to get feedbacks from the users on the interface. So, if the client/User is not satisfied changes can be made upon user feedbacks.

For UI modelling I have used digital prototyping as it is incremental which means that we can do it again and again and another reason for using digital prototype is to minimize the use of traditional paper-based prototype.

**Login**

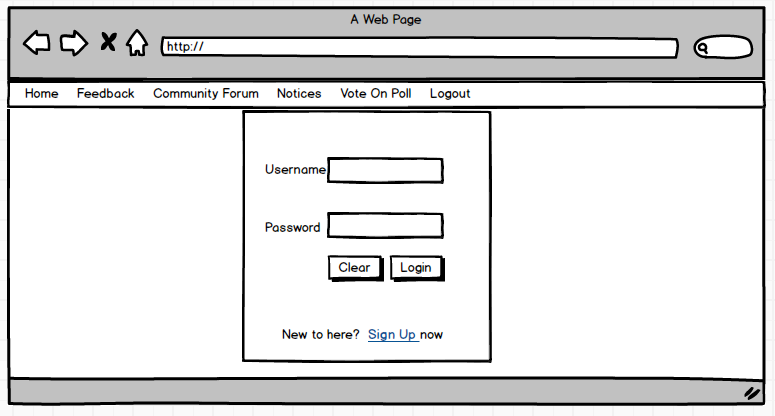


Figure : prototype for login

**Registration**

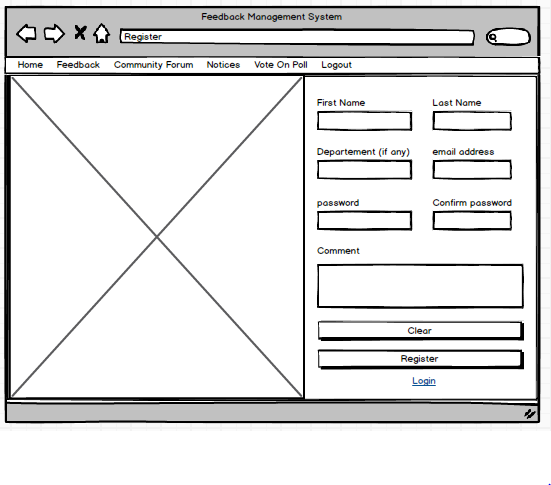


Figure : prototype for registration

**Dashboard-User**

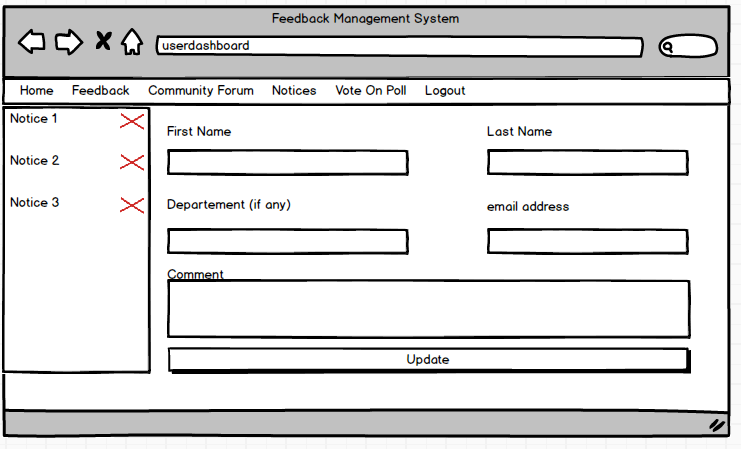


Figure : prototype for dashboard user

**Feedback**

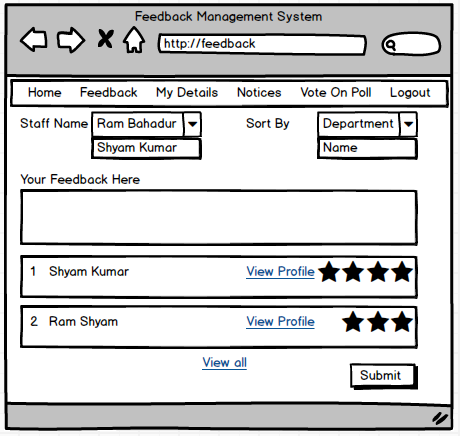


Figure : prototype for feedback

**Staff Profile**

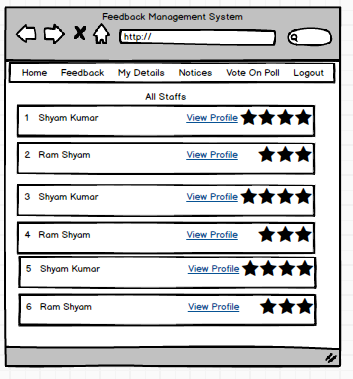


Figure : prototype for staff profile

**Individual Staff Profile**

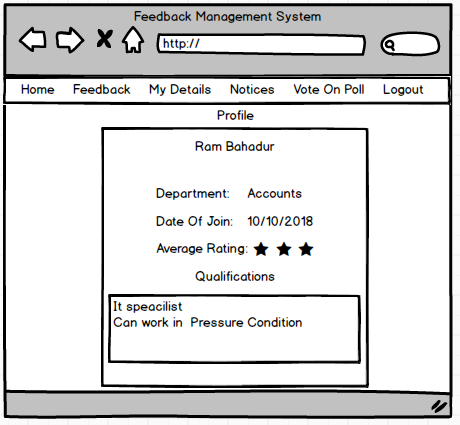


Figure : prototype for individual staff profile

**Community Forum**

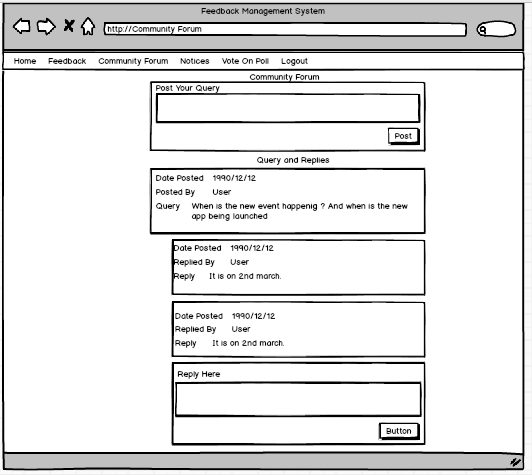


Figure : prototype for community forum

**Vote on Poll**

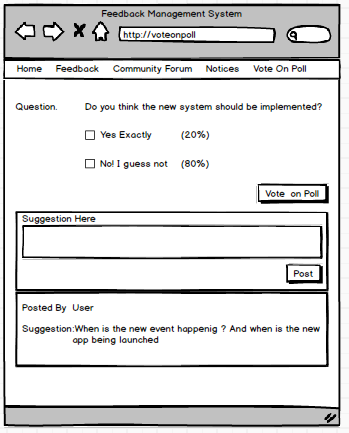


Figure :: prototype for Poll

**View Notices**

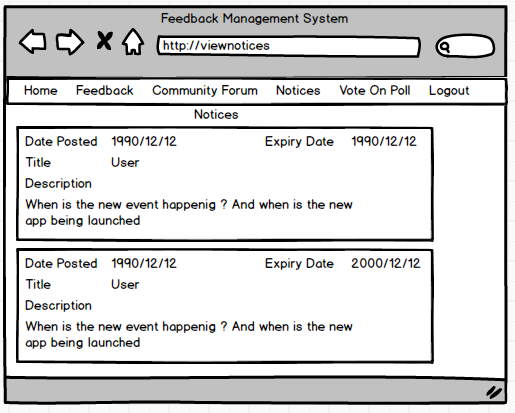


Figure : prototype for view notices

**Admin-Add Department**

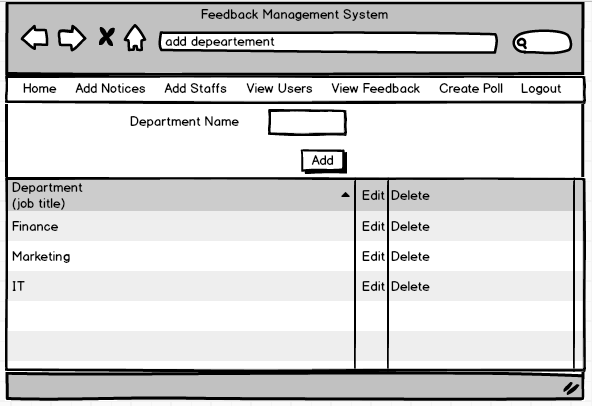


Figure : prototype for add department

**Create Poll**

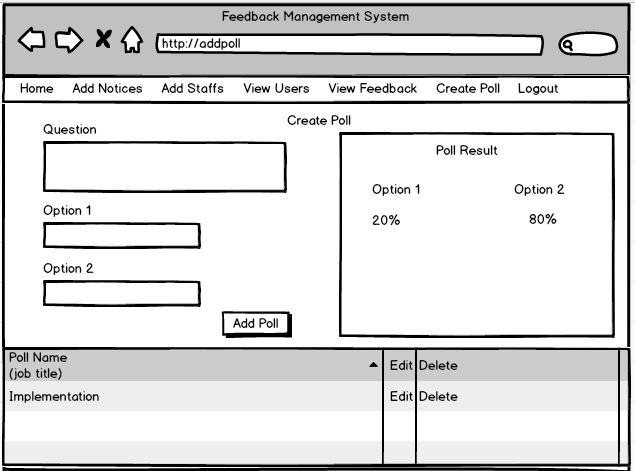


Figure : prototype for create poll

**Add Notices**

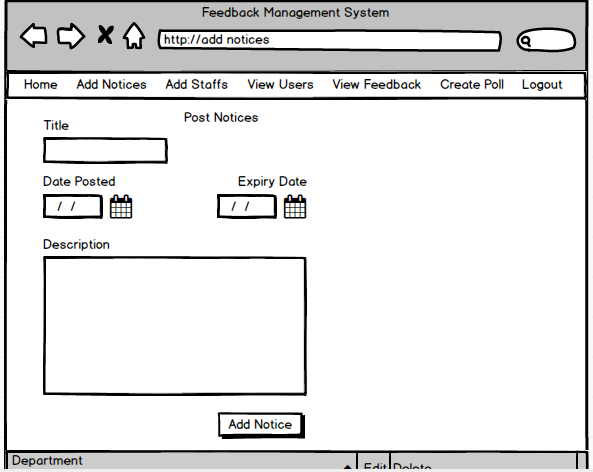


Figure : prototype for add notices

**View Feedbacks**

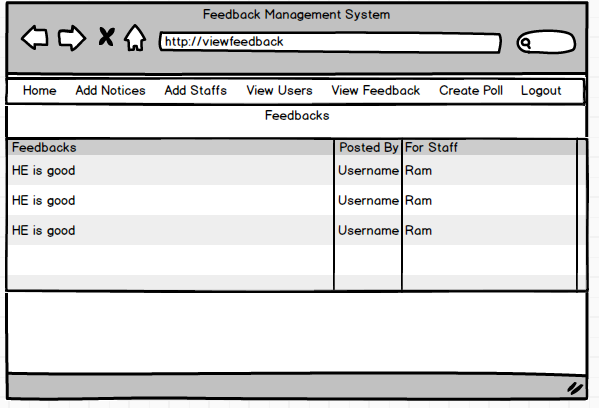


Figure : prototype for feedback view

**Add Staffs**

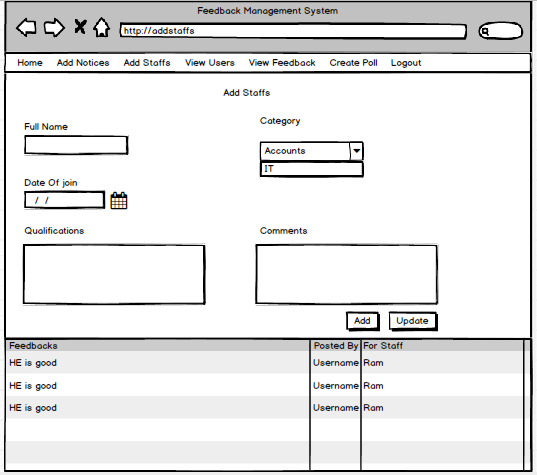


Figure : prototype for adding staffs

# **Chapter 4: Implementation**

## **4.1. Introduction**

Coding/ Implementation is the fourth part of software development. In this phase source code is written. Total of 20 days was allocated for this phase. This is the phase where actual work happens. It is the execution phase where all made plans are executed.

## **4.2. Language Used**

**Programming language**

* JavaScript
* PHP

**Mark-up Language**

* HTML

**Style sheet language**

* CSS

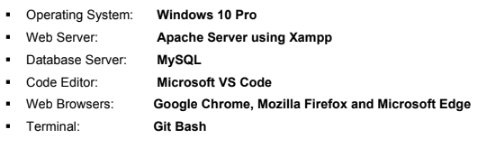
**Coding Structure**

* Model View Controller

**Framework Used**

* Code Igniter

## **4.3. Tools Used**



Some of the screenshots of coding are attached below and other remaining can be seen on chapter 9 appendix part:

**UI- Login**

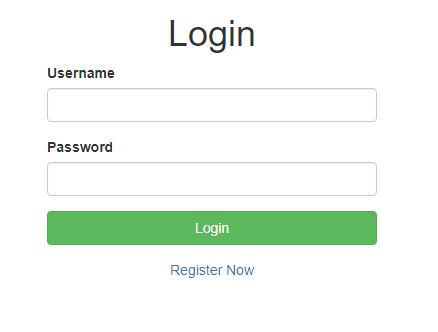


Figure : login

**View**



Figure : view for login

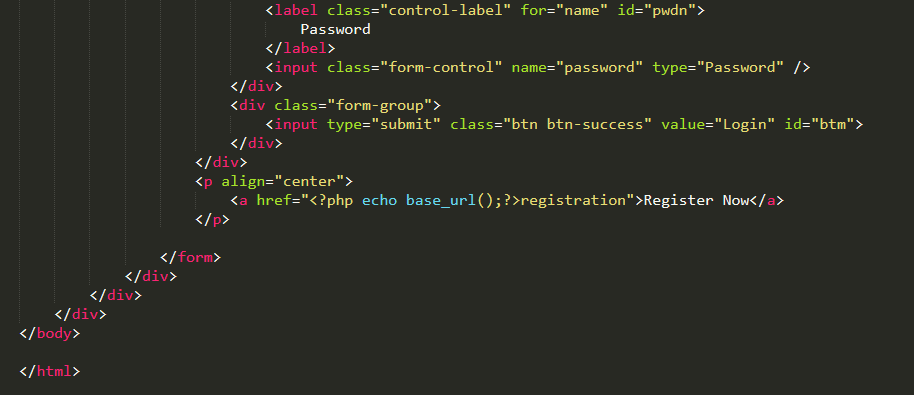


Figure view for login

**Controller**



Figure : controller for login



Figure :controller for login

**Model**



Figure :model for login

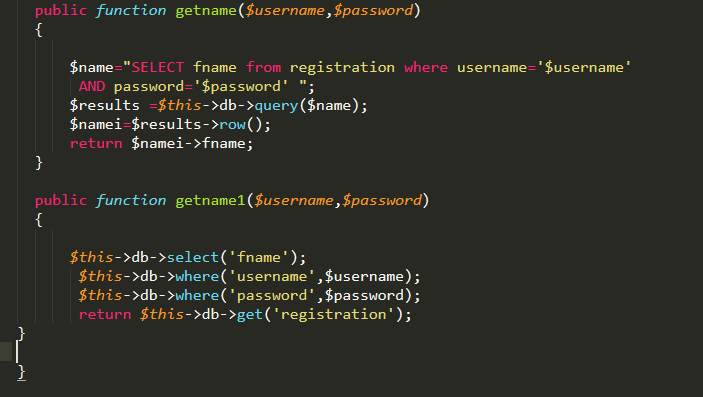


Figure :model for login

**UI – Registration**

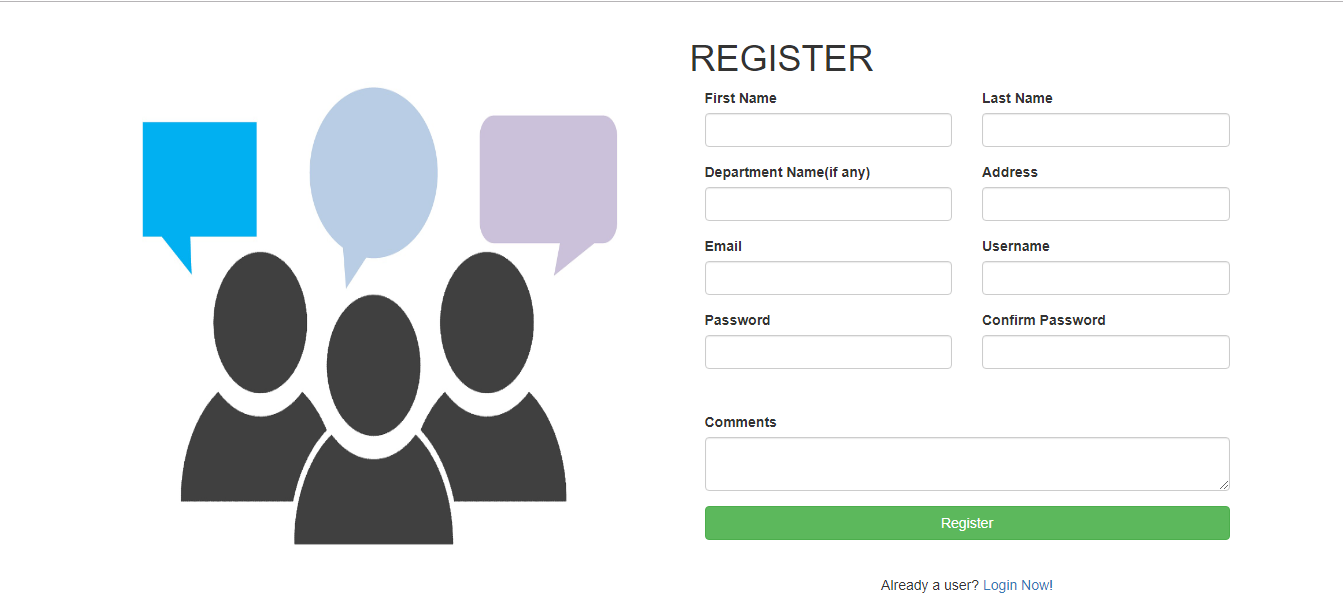


Figure : registration

**View**



Figure : view for registration



Figure : view for registration



Figure : view for registration

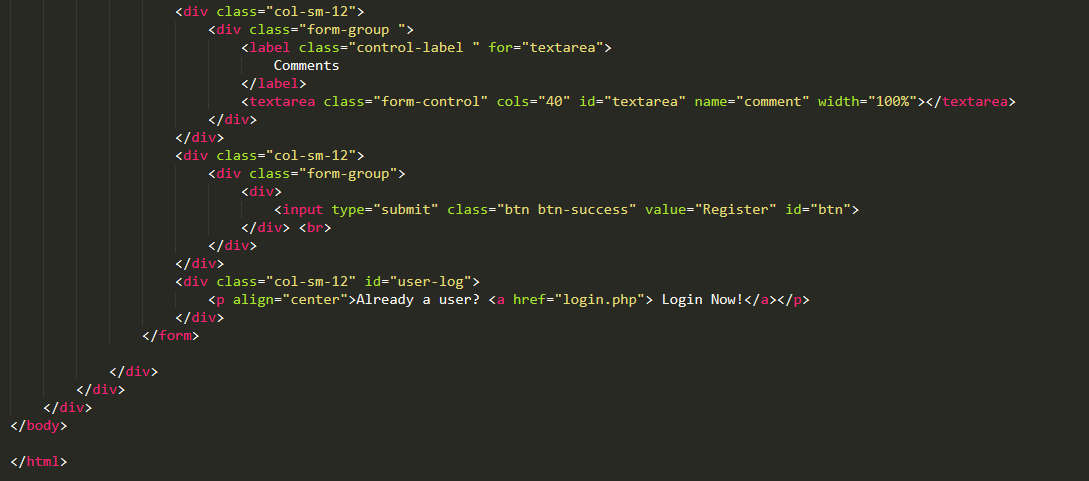


Figure : view for registration

**Controller**

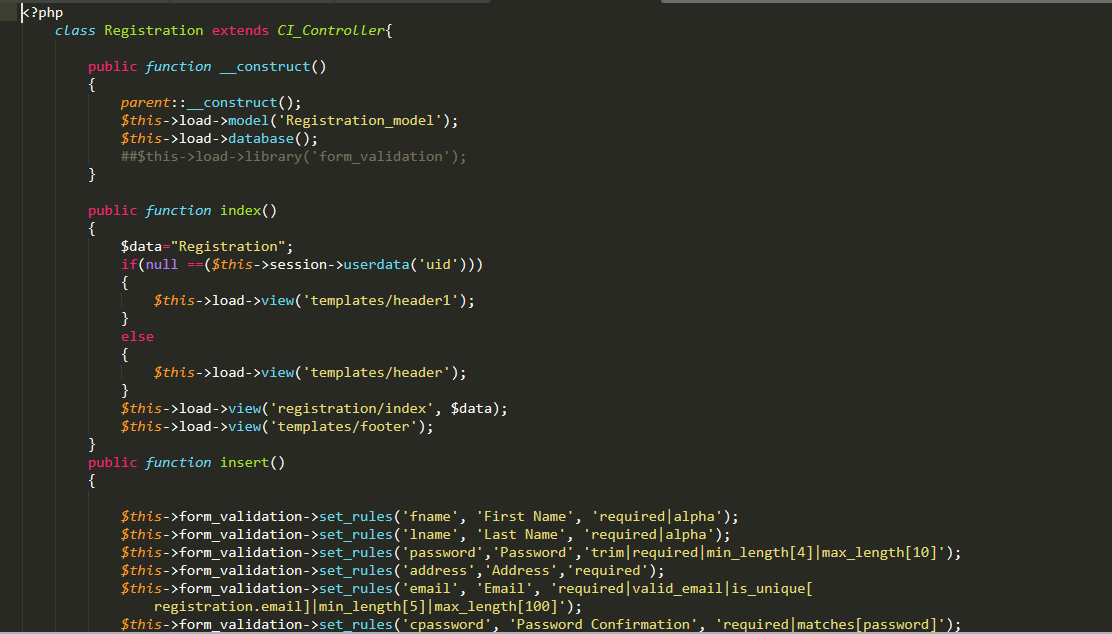


Figure : controller for registration

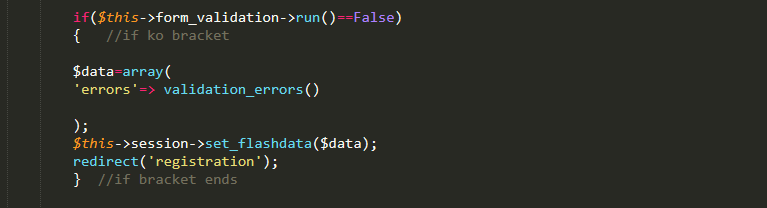


Figure : controller for registration

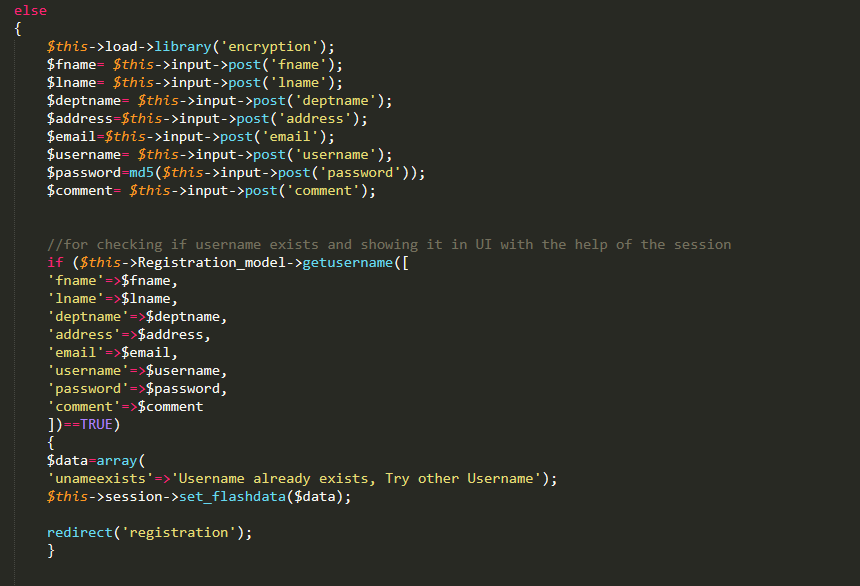


Figure : controller for registration

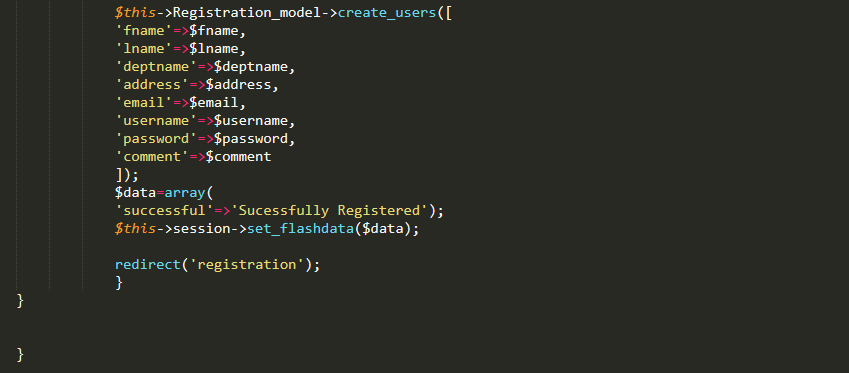


Figure : controller for registration

**Model**



Figure : model for registration

# **Chapter 5: Testing**

Testing is a process in which a program is executed in order to find the errors and bugs. It is the process of validation and verification of software or an application. It is performed as to ensure that bug free program and helps in output of quality product. There are various types of testing that helps in finding bugs in program, for this project I have used Black-box testing and unit testing (White box testing). Black box testing doesn’t deal with coding structure and it only focuses on input and output. Test cases should be made for Black box testing. Unit testing or white box testing deals with coding and each function are tested, to ensure its correctness.

## **5.1. Black Box Testing**

For black box test:

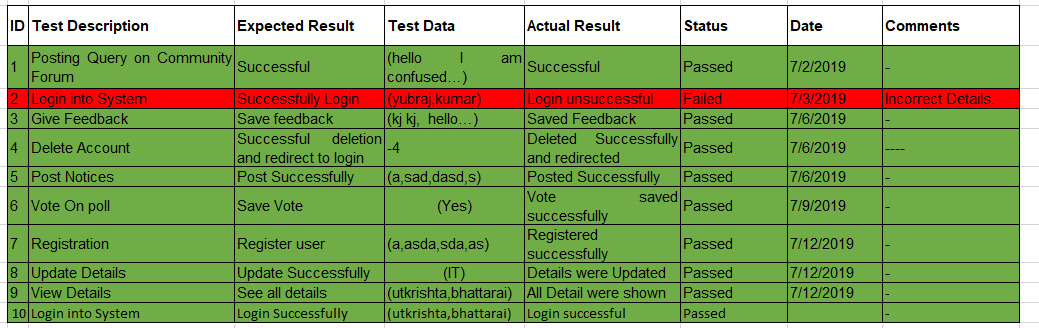


Table : Black box testing

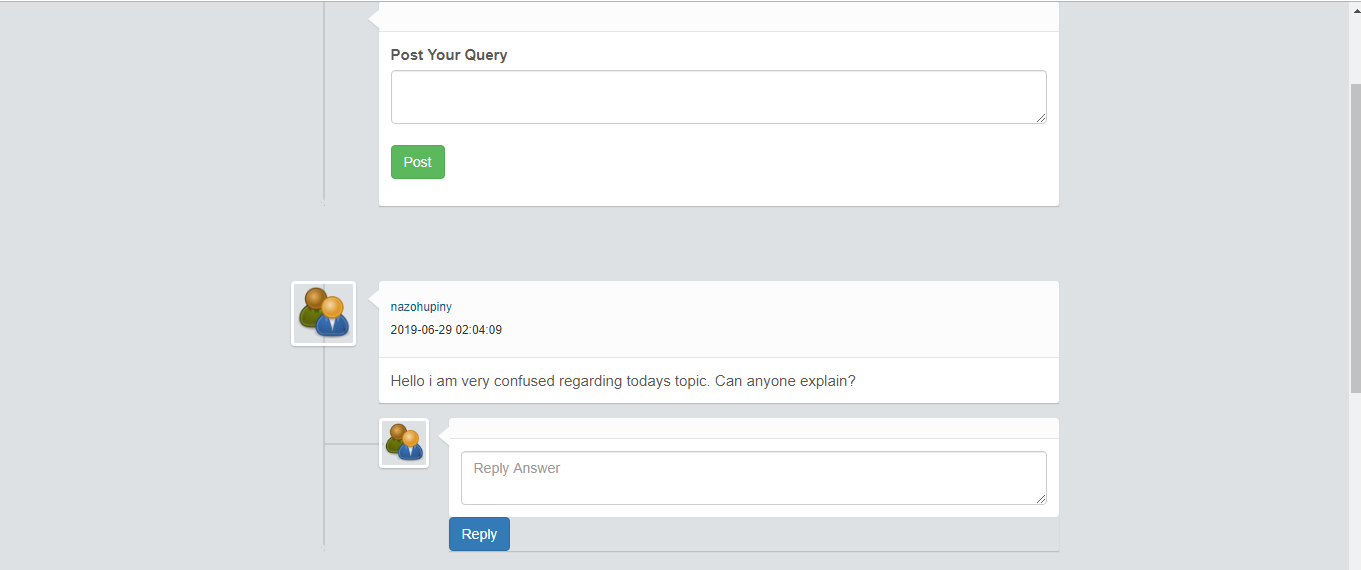
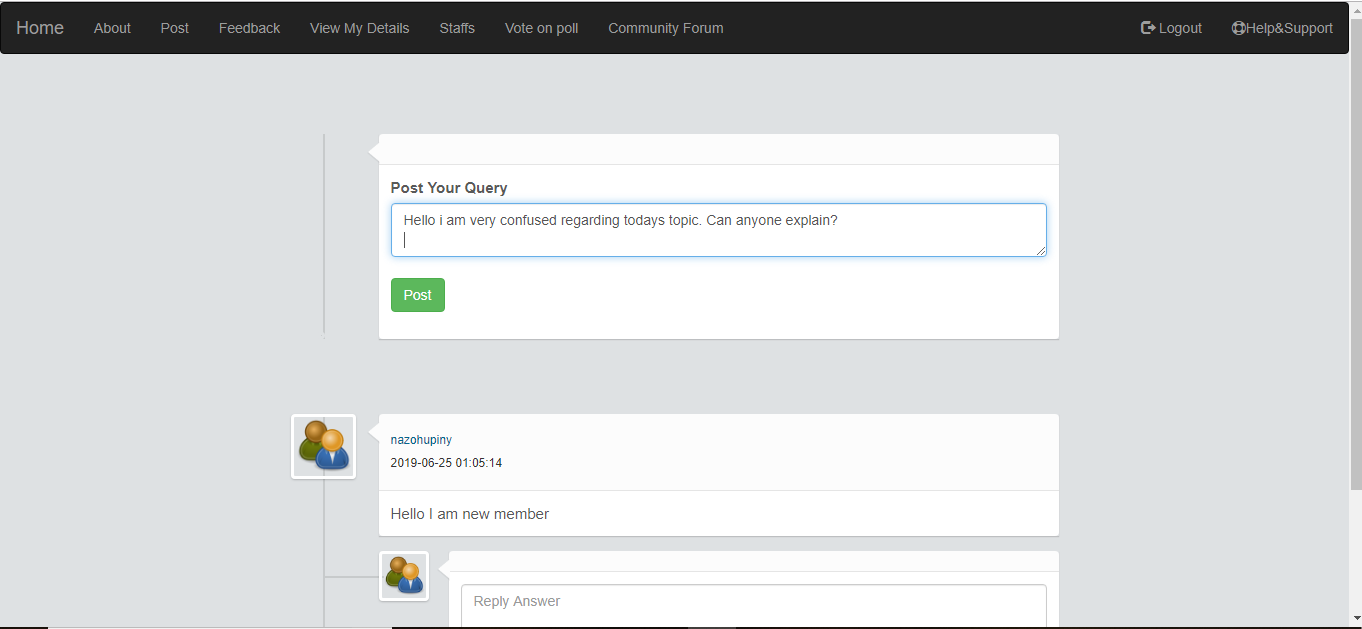


Figure : Black box testing for community forum

**Successfully posted in community forum.**

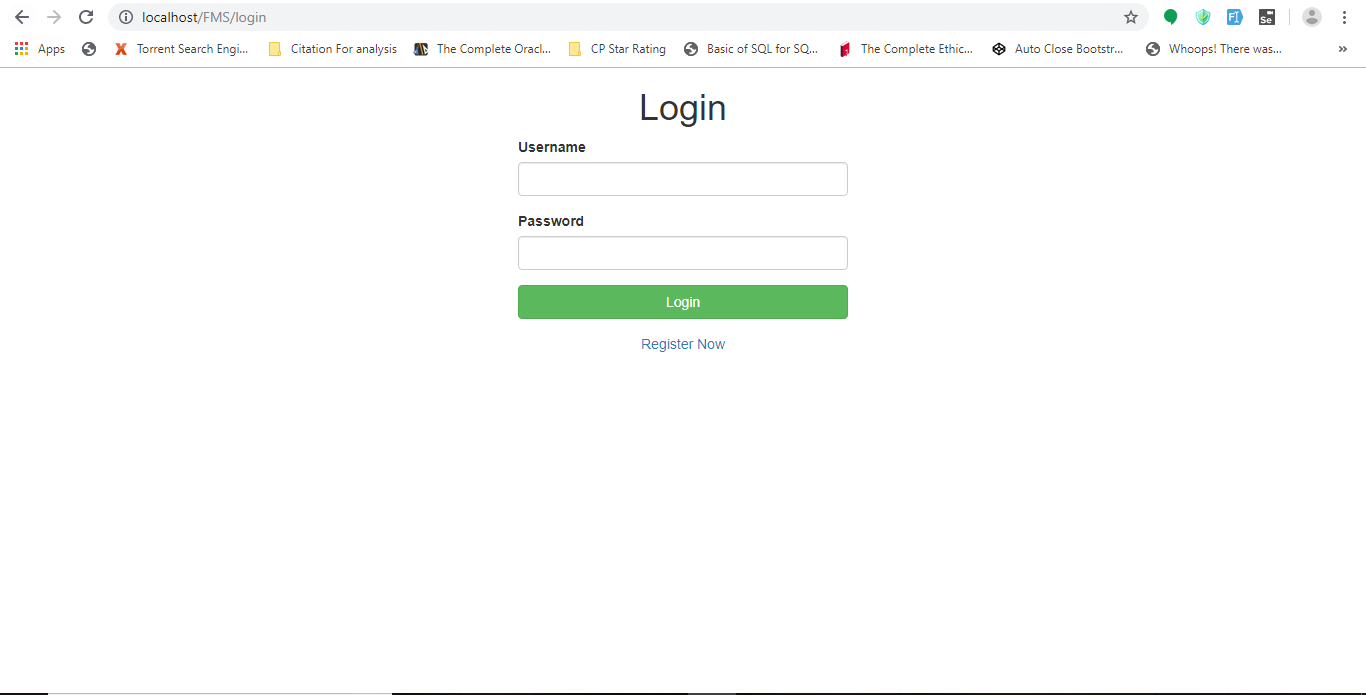
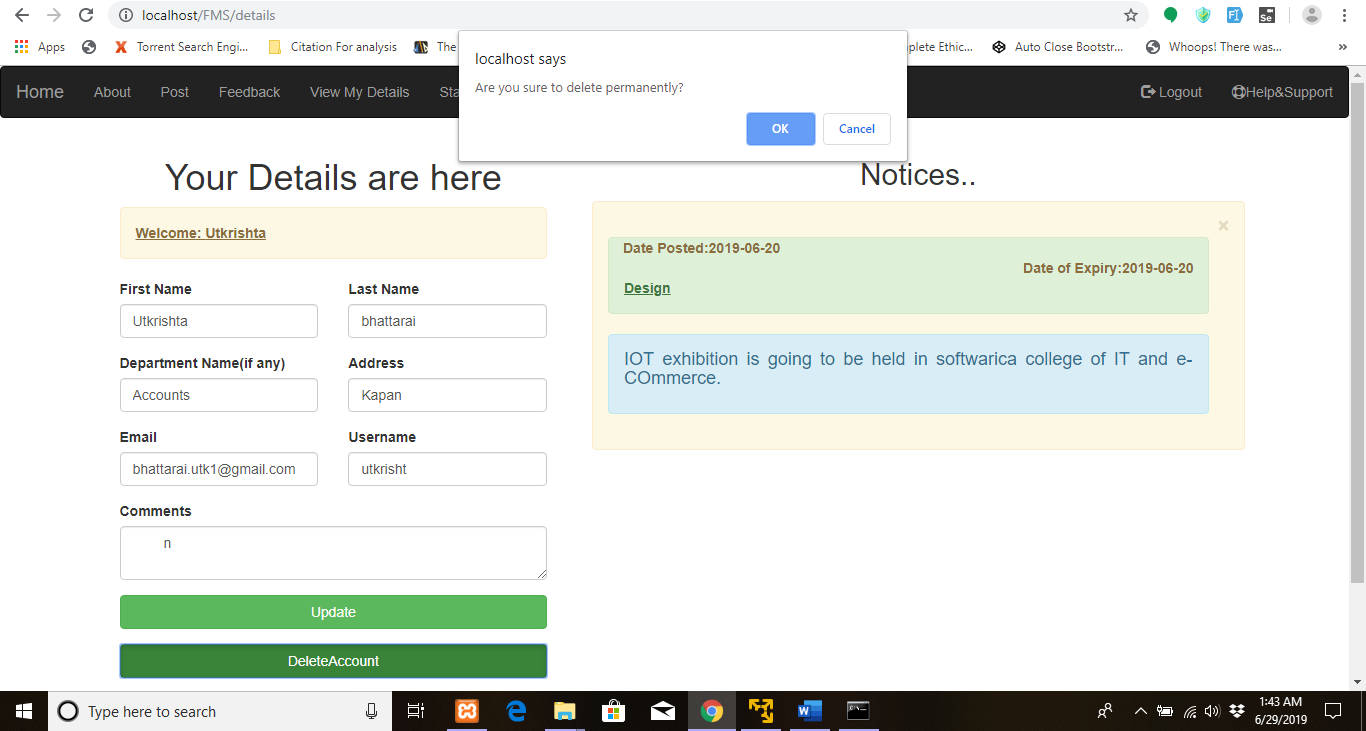


Figure : Black box testing for account deletion

**Account was permanently deleted.**

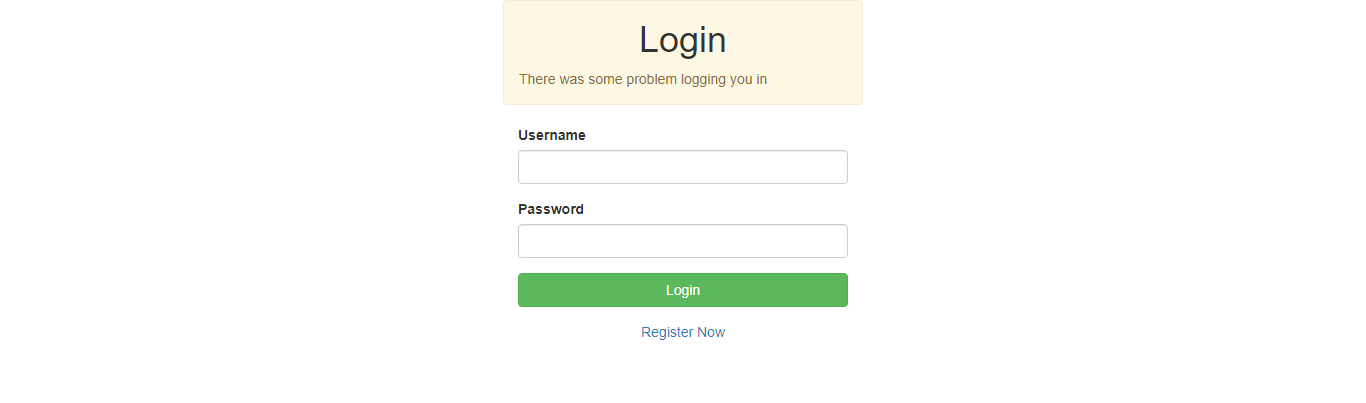
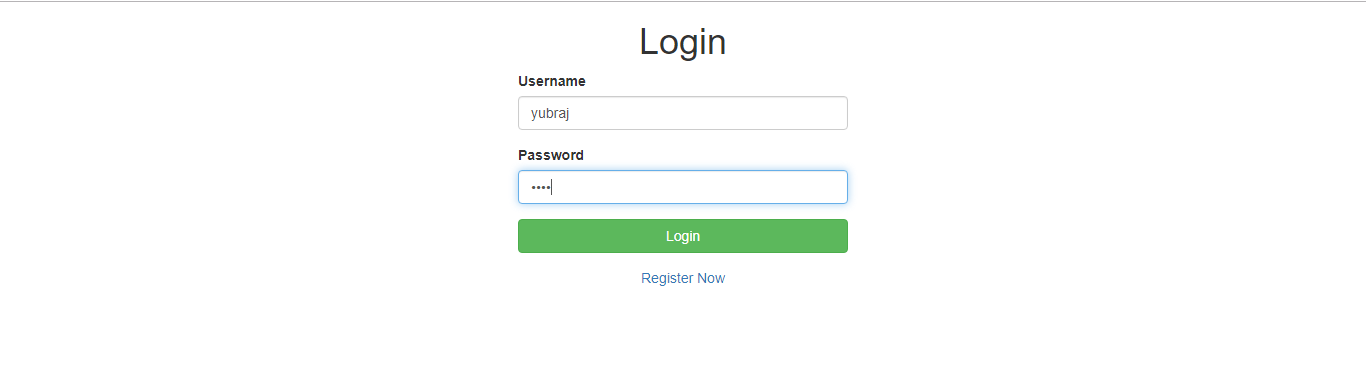


Figure : Black box testing for login

**Login was unsuccessful.**

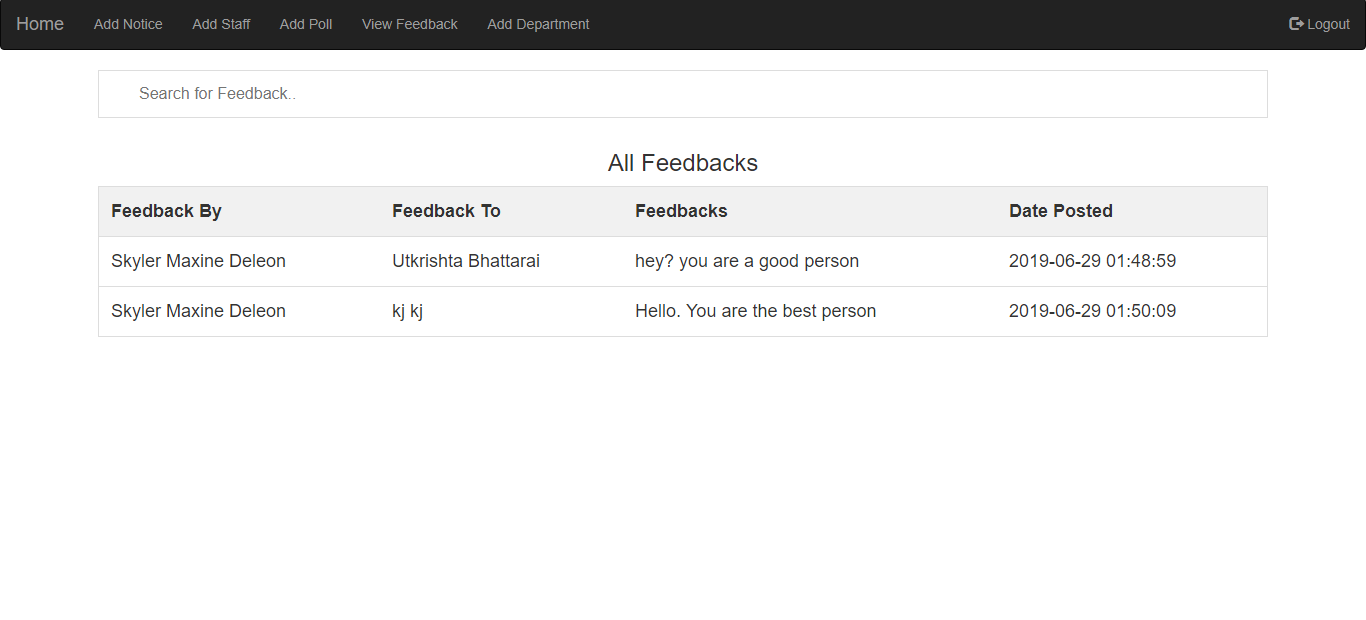
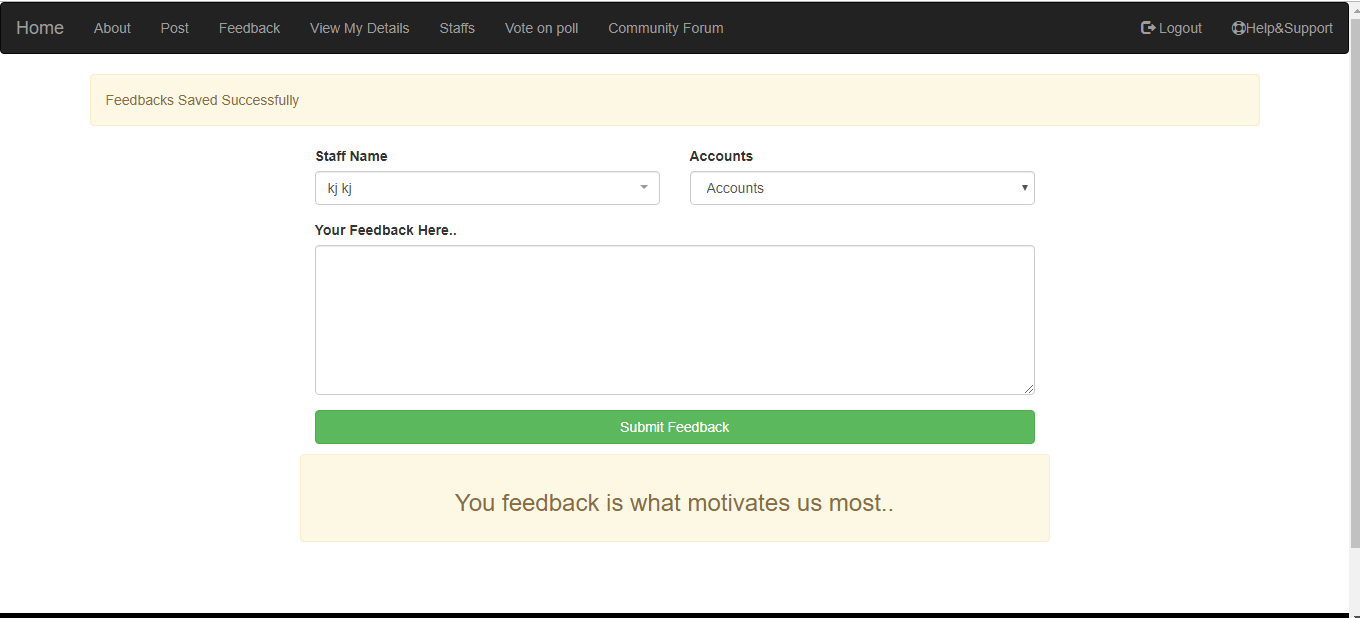
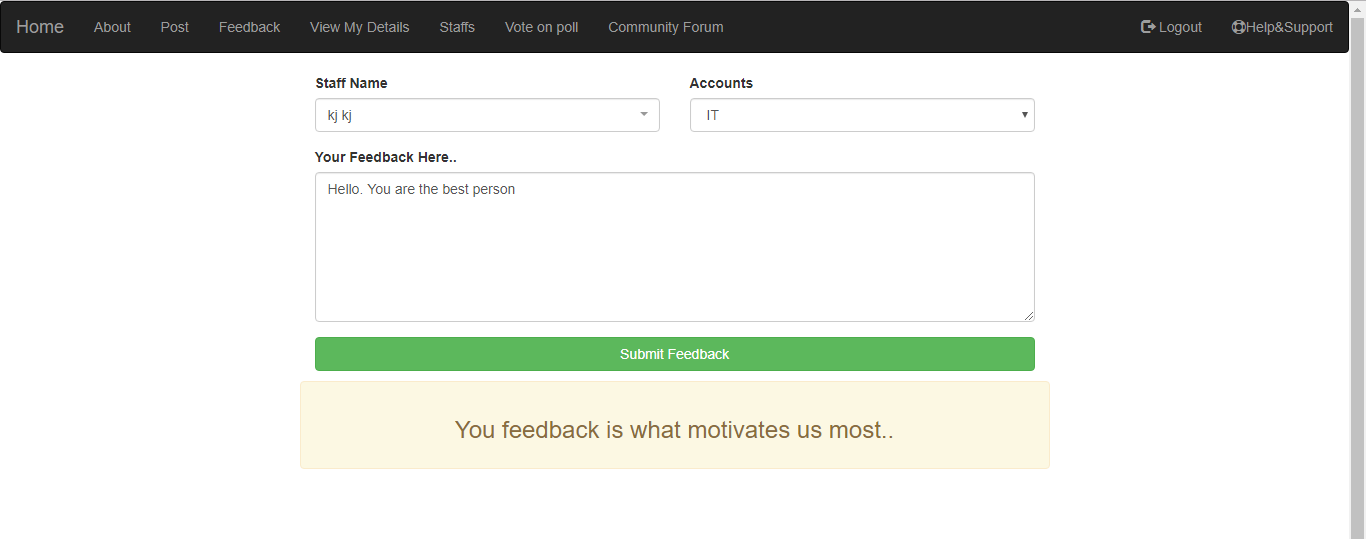


Figure : Black box testing for feedback

**Feedback was saved successfully.**

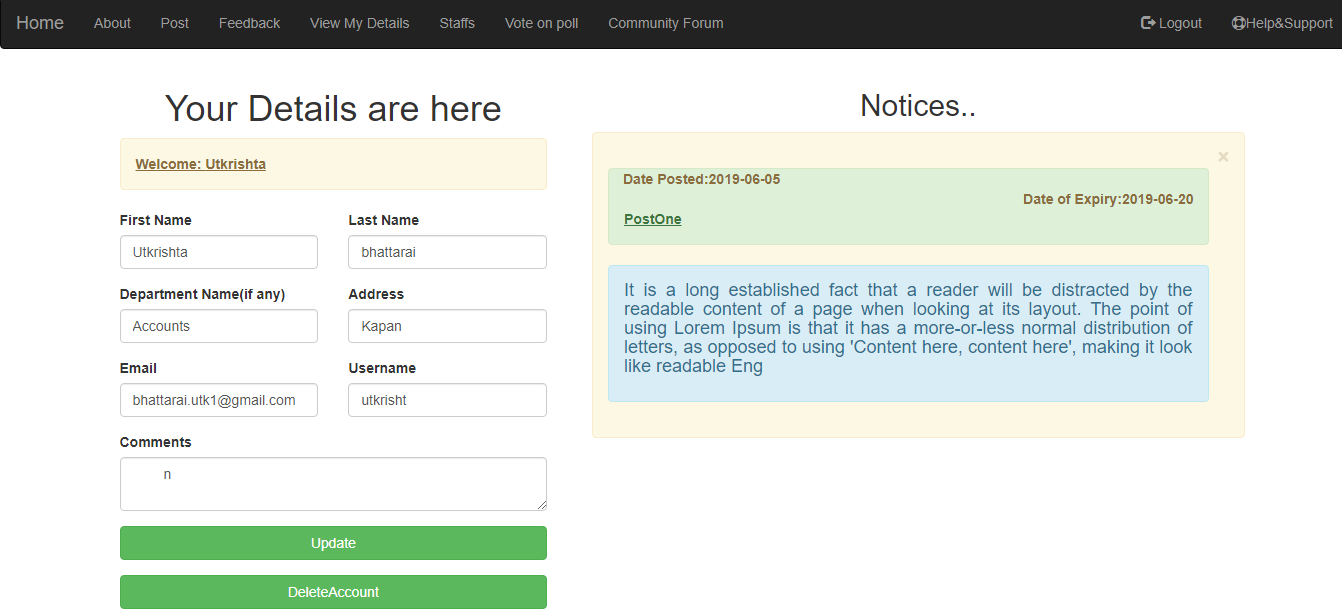
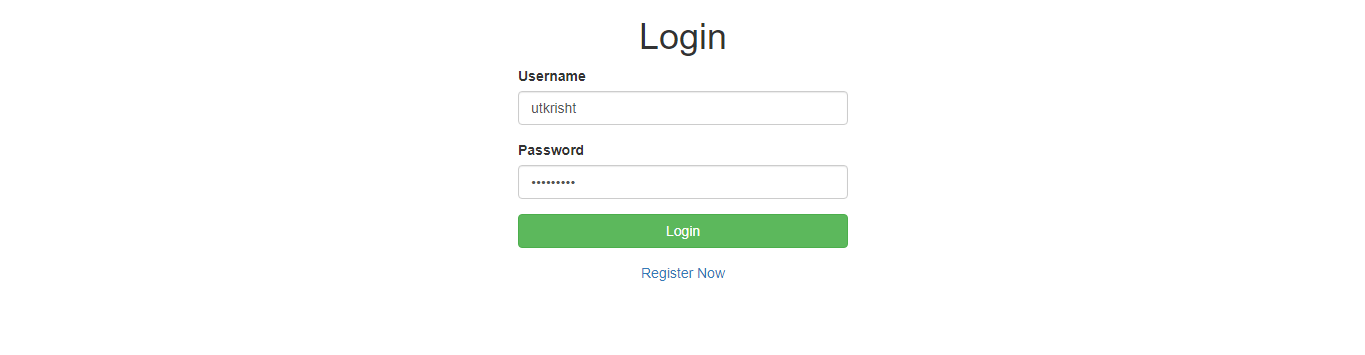


Figure : Black box testing for showing details

**Details Were Shown.**

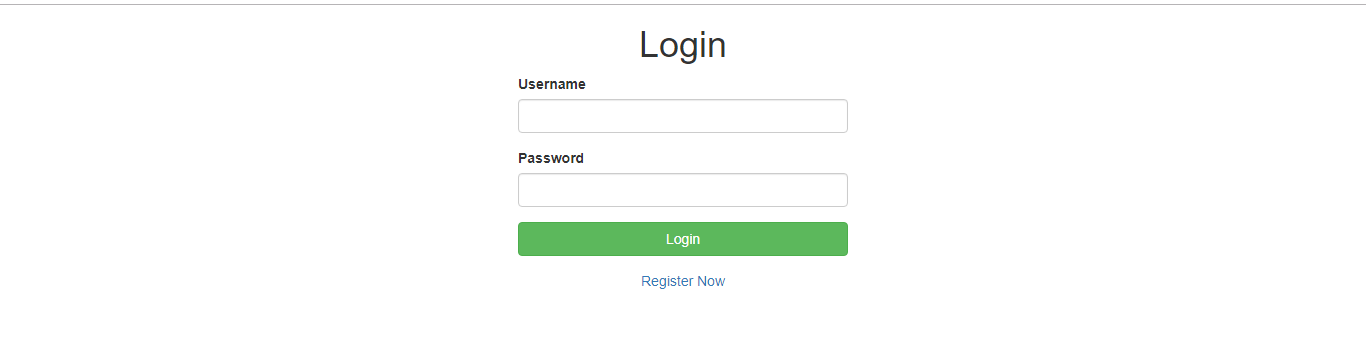
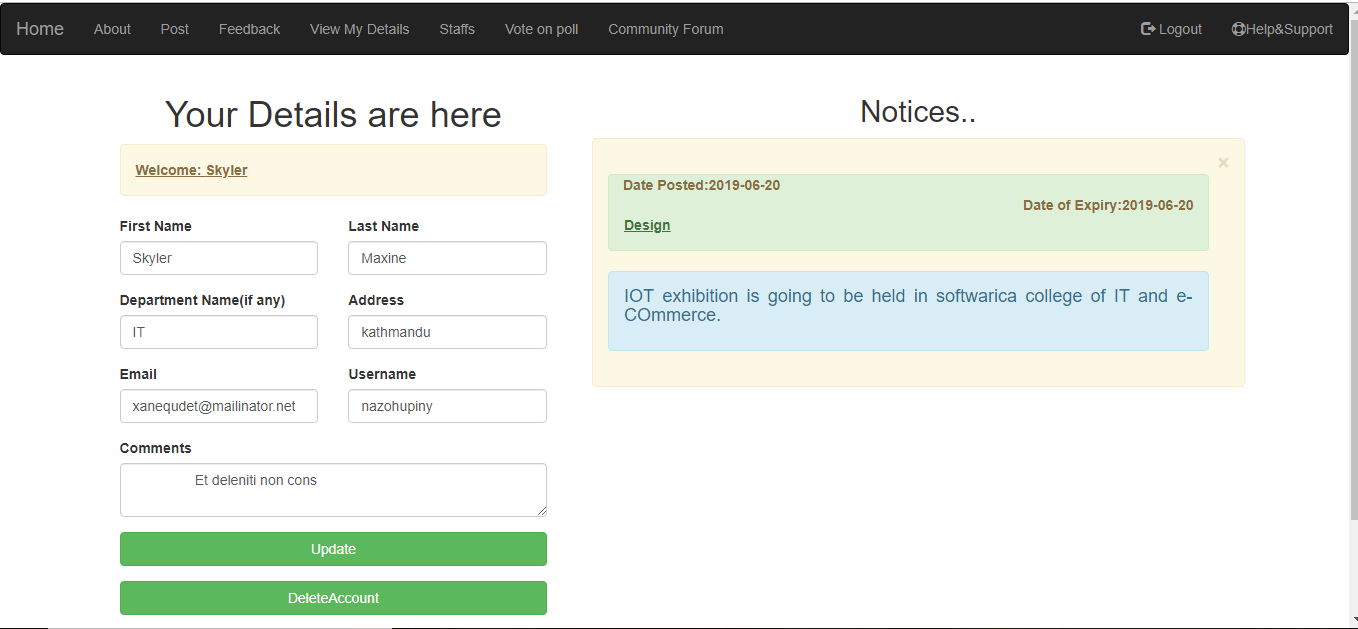


Figure : Black box testing for logout

**Successfully Logged out.**

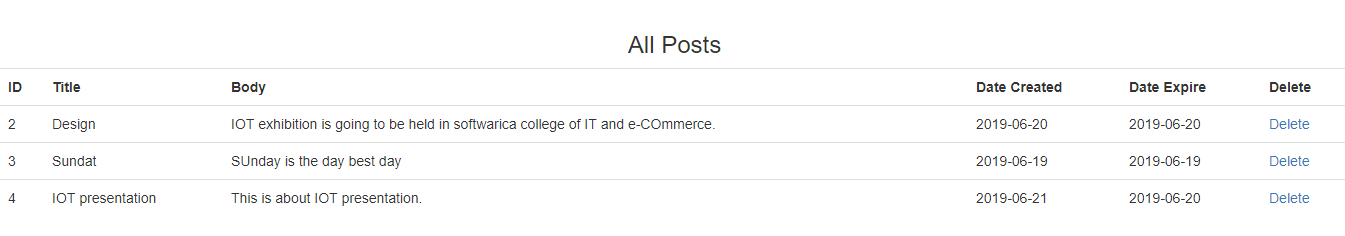
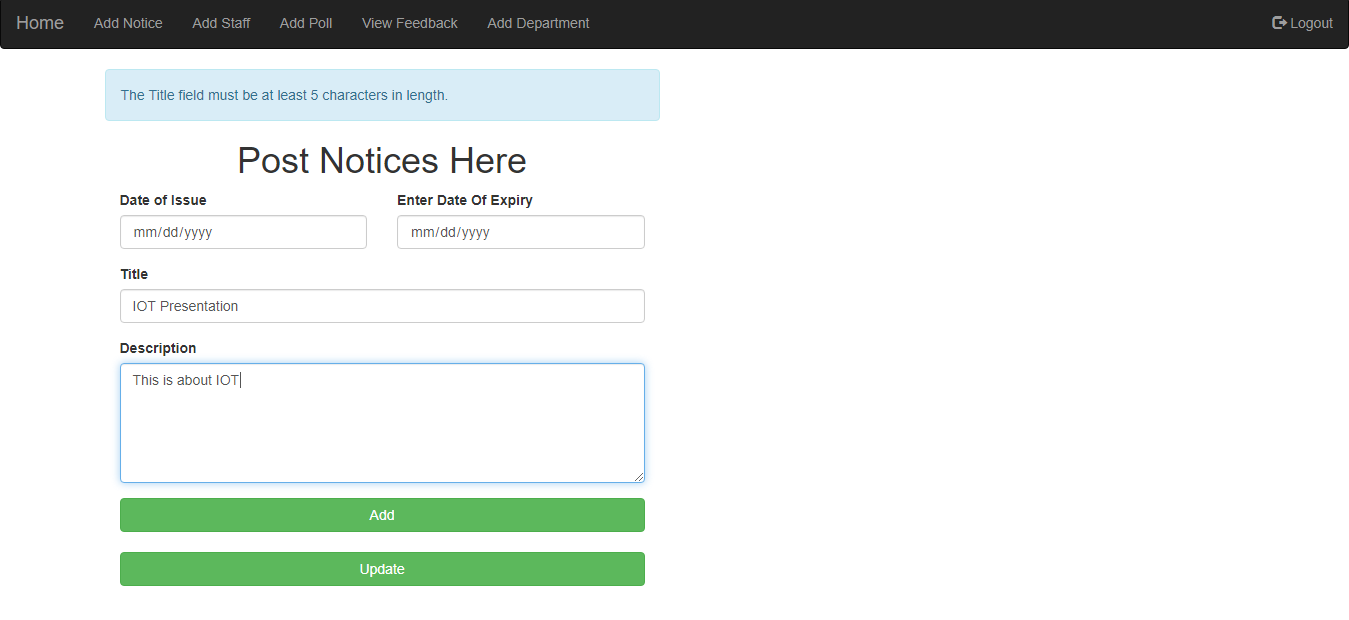


Figure : Black box testing for saving posts

**Post was saved successfully.**

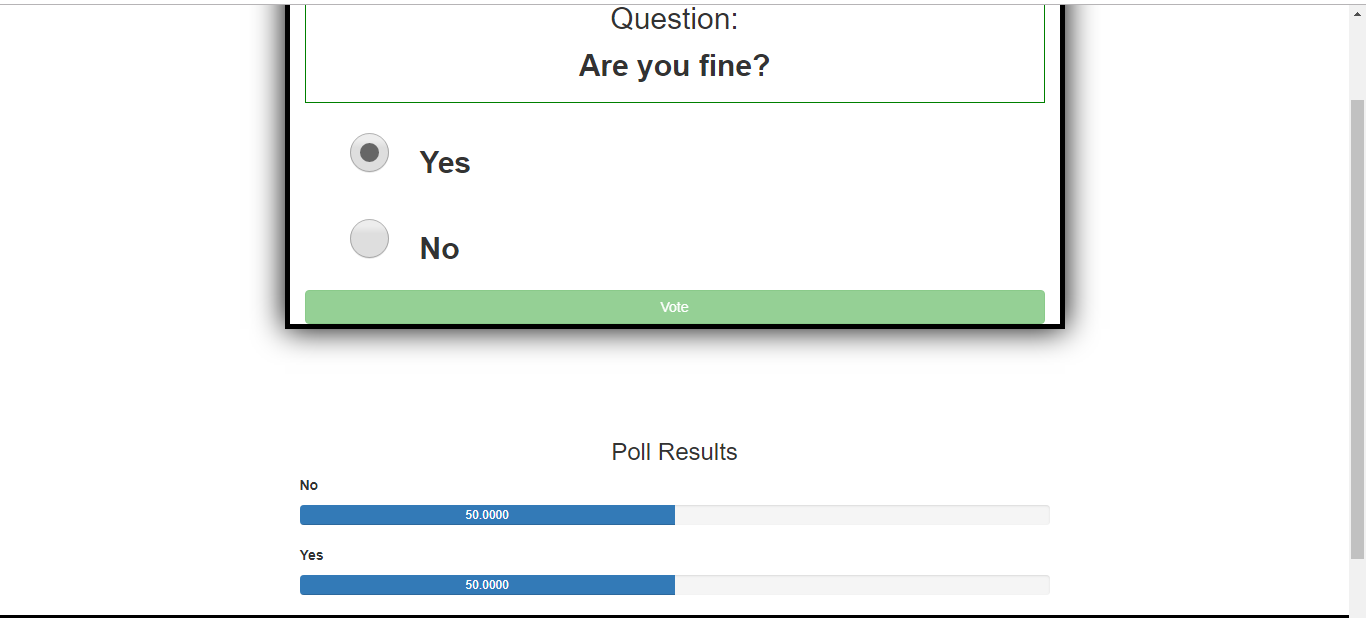
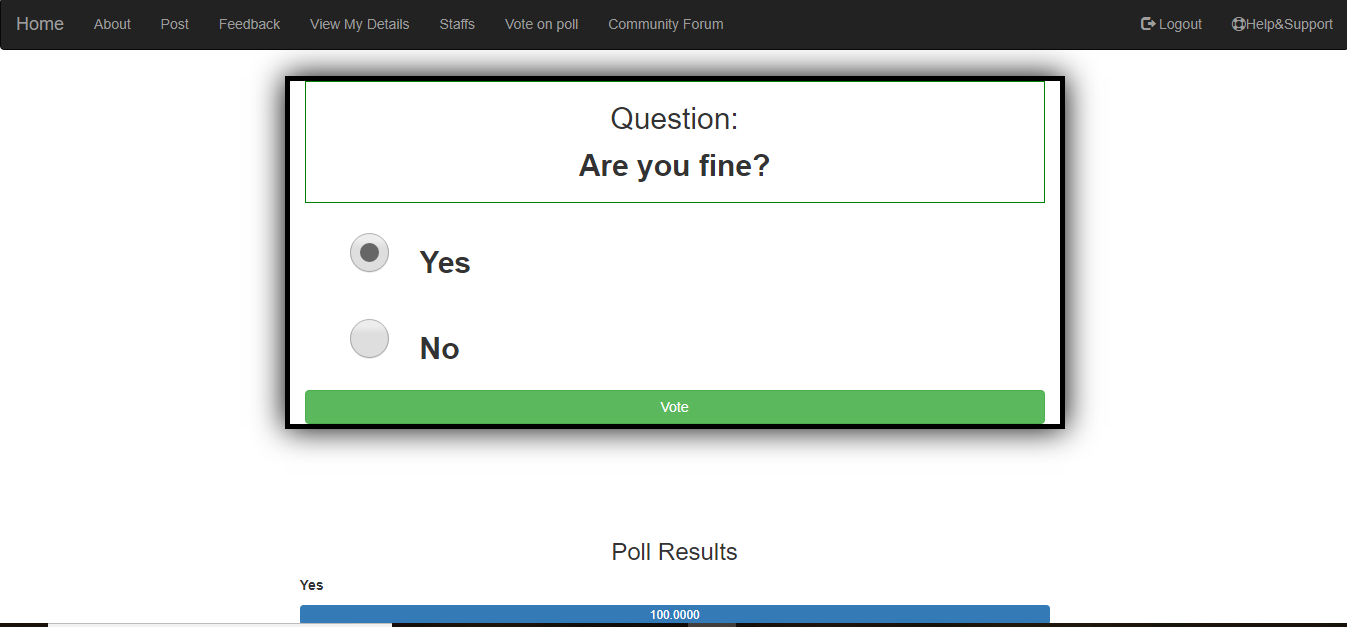


Figure : Black box testing for voting on poll

**Voted registered on poll**

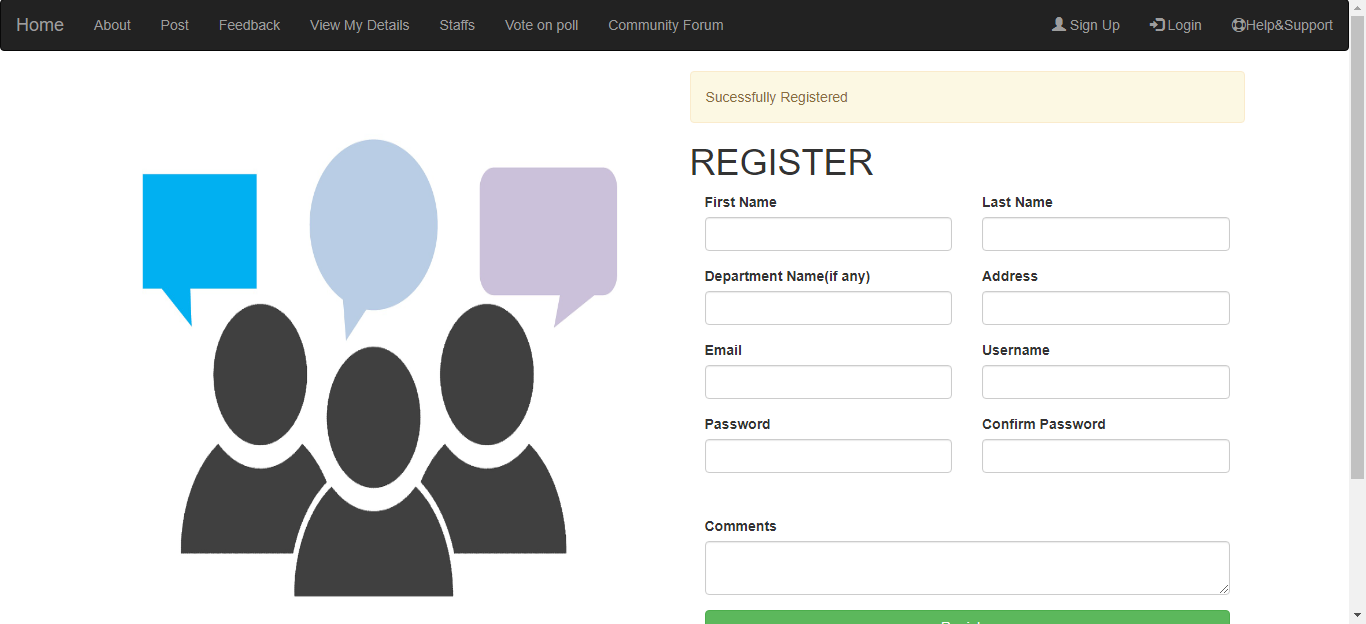
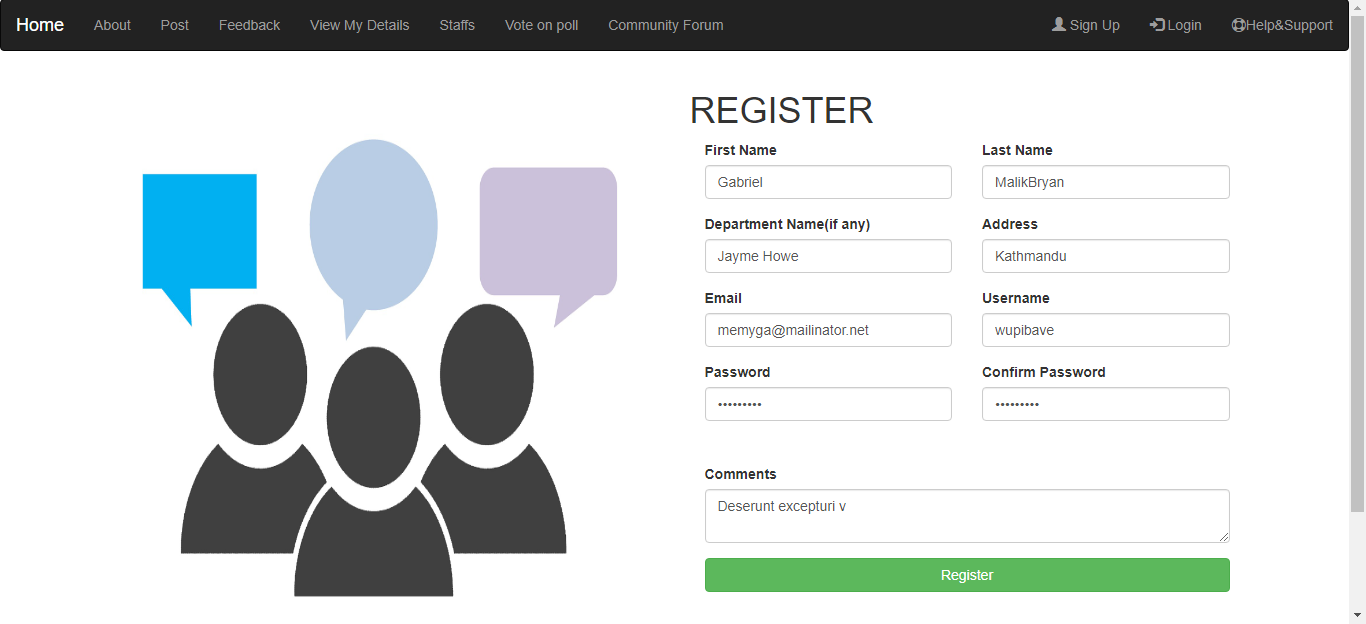


Figure : Black box testing for registration

**User Registered Successfully.**

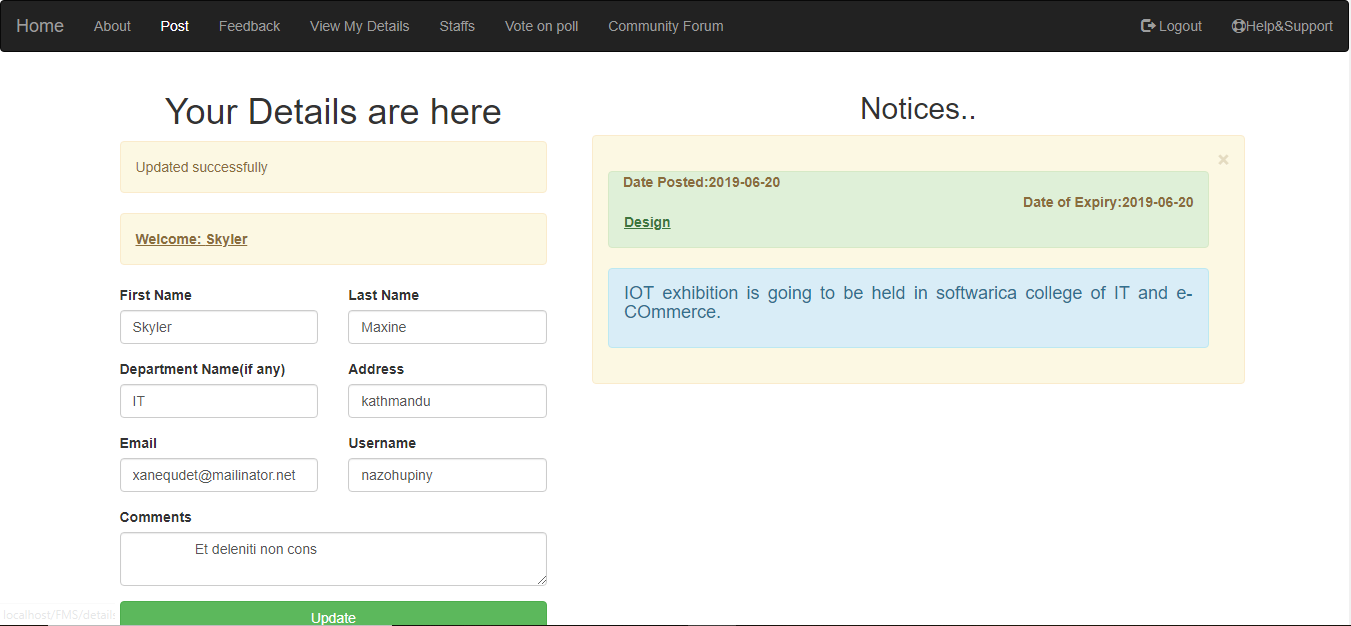
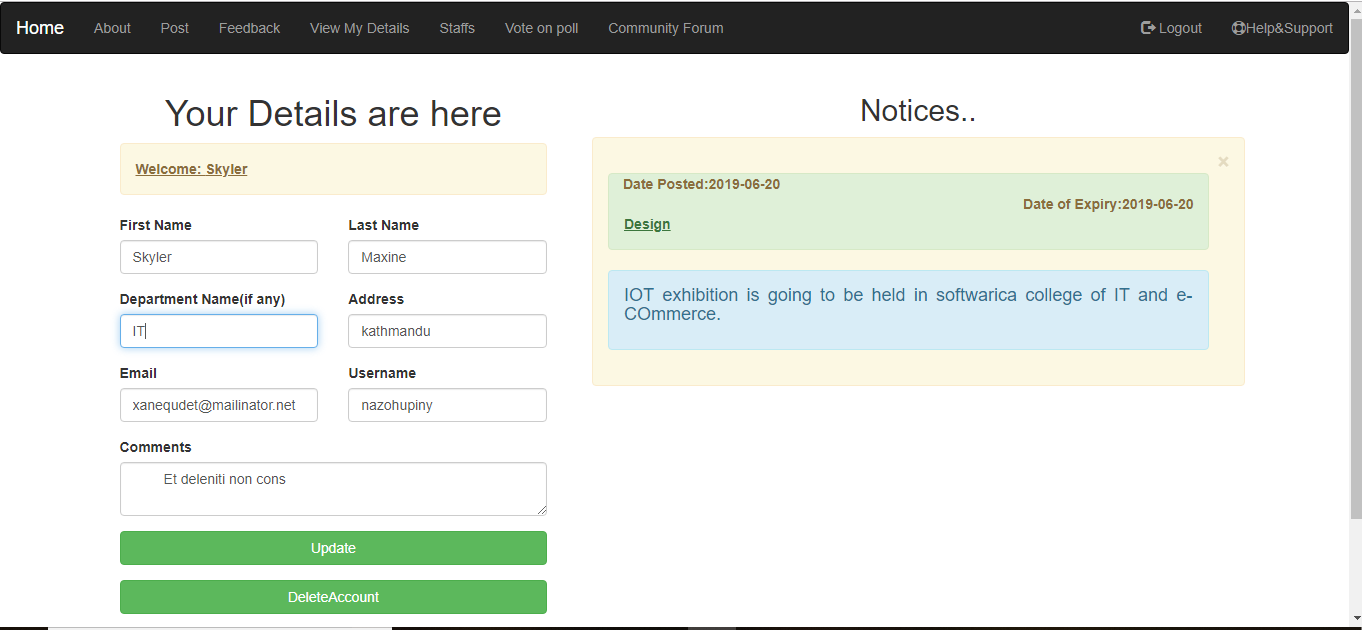


Figure : Black box testing for updating details

**Details updated successfully.**

## **5.2. Unit testing**

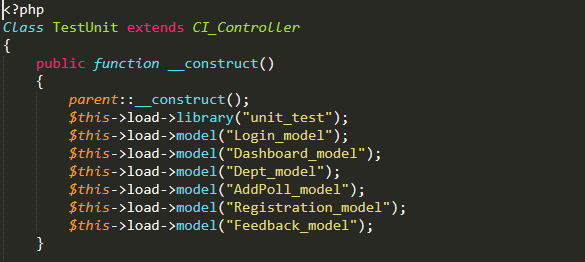


Figure : unit testing 1



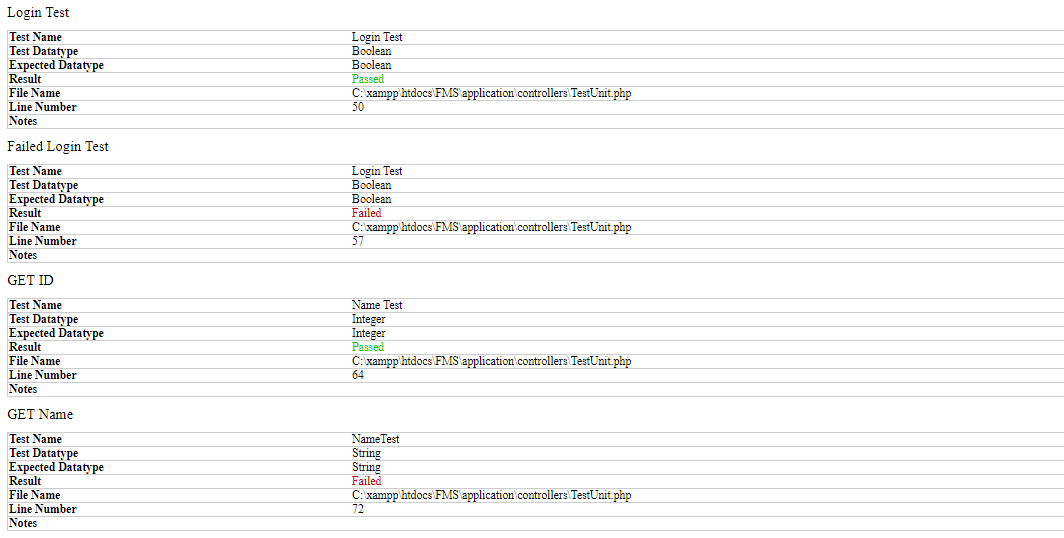
Figure : unit testing 2





Figure : unit testing 3

**Output of above test script**



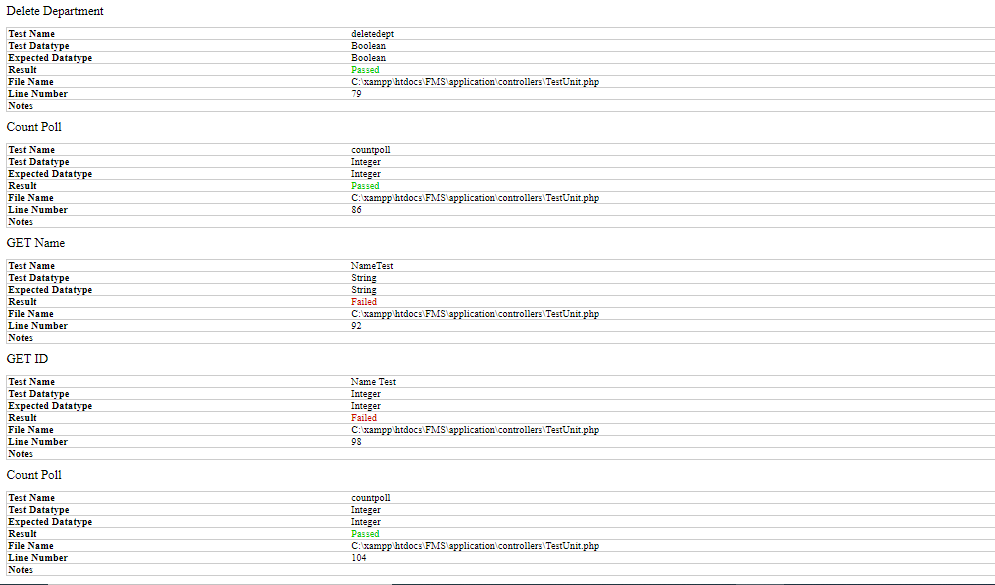


Figure : Output of unit testing

Testing was done.

# **Chapter 6: Others project issues**

**Limitation of the project:**

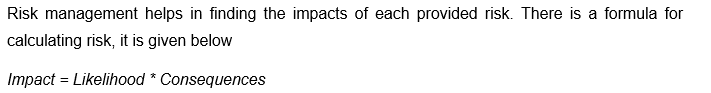
Feedback management system won’t be able to perform/do following things:

* It will not be able to let users chat anonymously in community forum and also while giving feedback.
* The system will not be able to limit incorrect user login.
* The users won’t be able to comment while voting on the poll, but this feature will be provided on newer updates.

**Future Work:** Almost all of the requirements gathered in initial phase is completed but some functionalities of the program are still pending in order to make the Feedback management system more useful. For the future work, I have planned following things:

* Replacing the current code with updated and more optimized code.
* Make use of more fonts, colors, images and videos to make website more beautiful and attractive.
* Replace the like button with dynamic star rating.
* Let users stay anonymous when they chat in community forum and when the give feedbacks.
* Make the system usable for multiple organisation.
* Add new security features like two factor authentications, limiting user login.

**Risk Management:** Every project contains risk. Risk management is the process of assessing, recognising possible risk that might arise in the future and solving it with the help of tools and technologies.



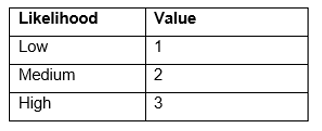


Table : Likelihood table

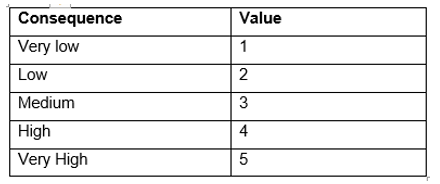


Table : consequences table

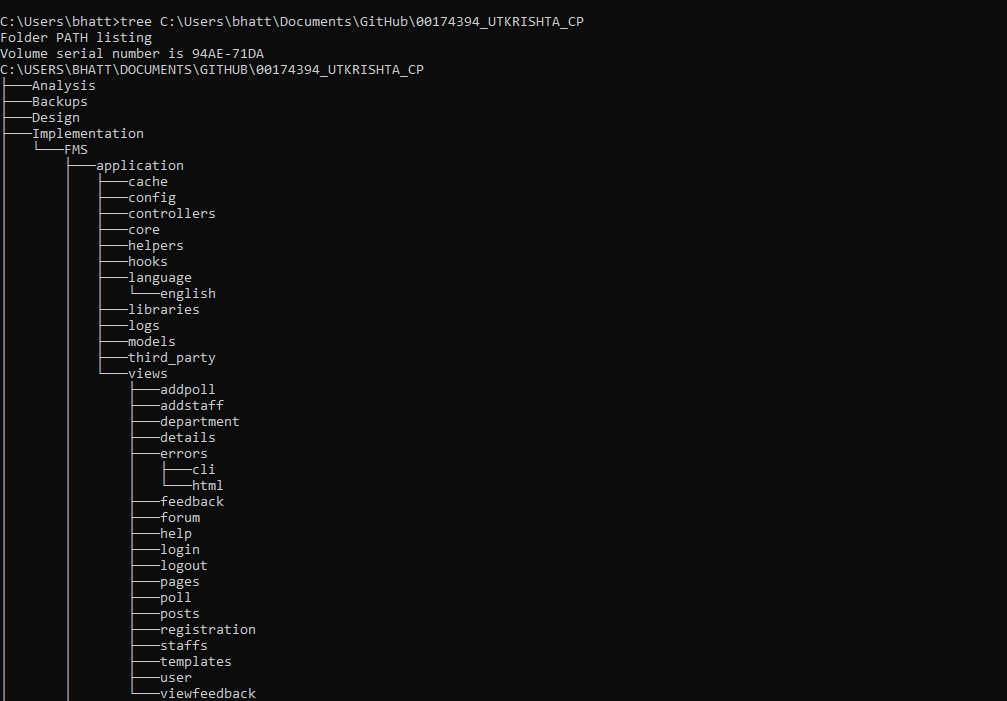
As stated, and listed in Proposal. Following risk were listed.

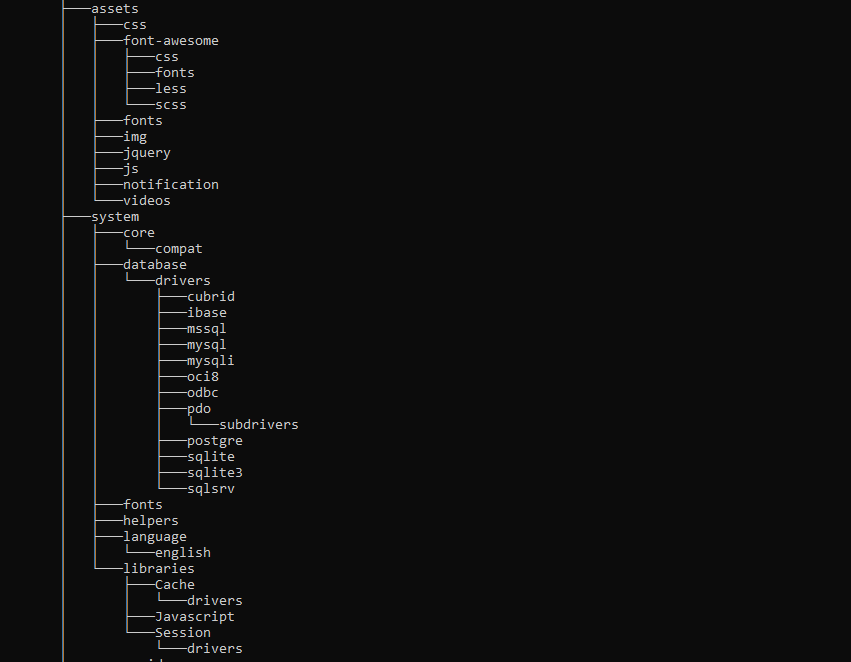


Table : Risks table

From above risk, risk of changing requirements was faced during the development of Feedback management system. Not all requirements were gathered properly so there were some issues with changing requirements. The change in requirement caused the project to delay a bit.

**Configuration management:** Configuration management is the process of tracing task and monitoring changes in development of business solutions. It is software engineering process for founding and preserving consistency of software/products performance. Firstly, it was used by military organisations to manage any changes made throughout the development of software. The changed files are safely backup in a separate folder so that it can be used in future, if necessary. For the development of Feedback management system, to keep backup of files and to save changed logs, I have used GitHub as a version controller and also to ensure its availability, I have also used online cloud storage for keeping the backup of files.





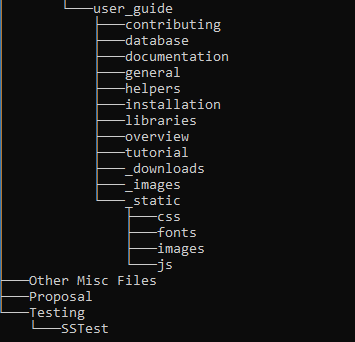


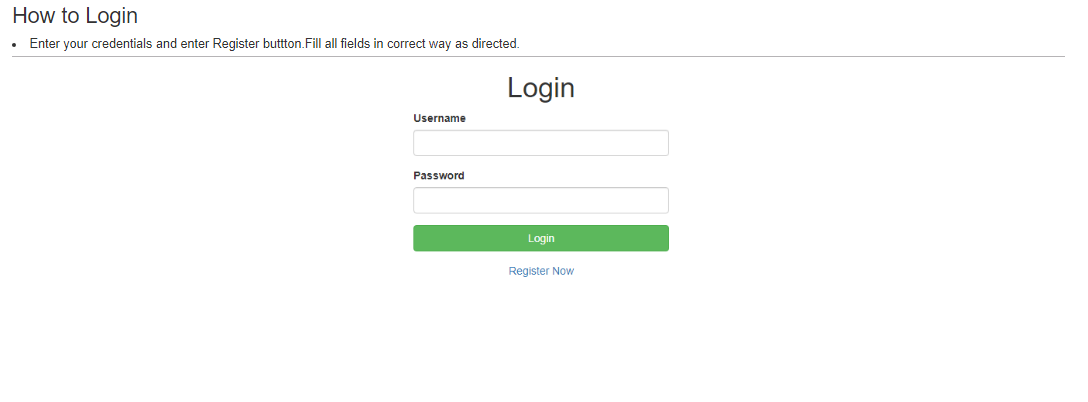
Figure : Configuration management:

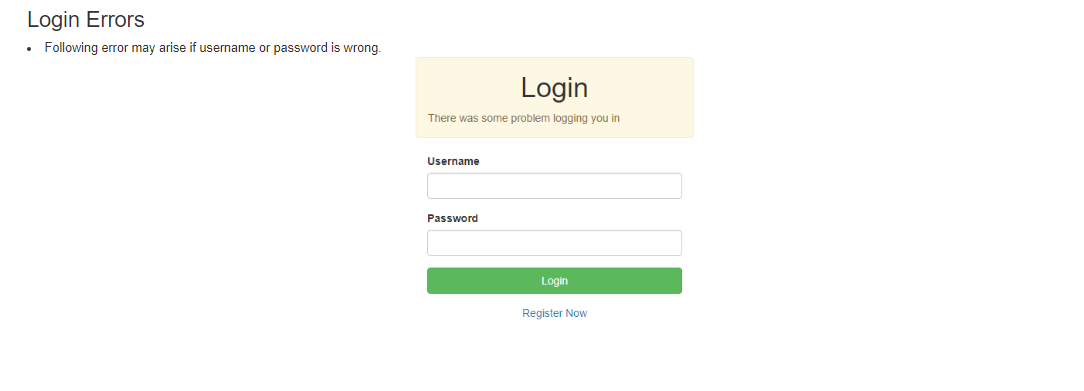
**Link to my GitHub id is :** <https://github.com/UtkrishtaBhattarai/00174394_UTKRISHTA_CP>

**User Manual:** For users to understand the system easily and for easy navigation, a user manual is made. In User Manual different pages and things to do in those pages are shown. I have named it as Help/Support and placed it in navigation bar so that user can view it easily.



Figure : User manual





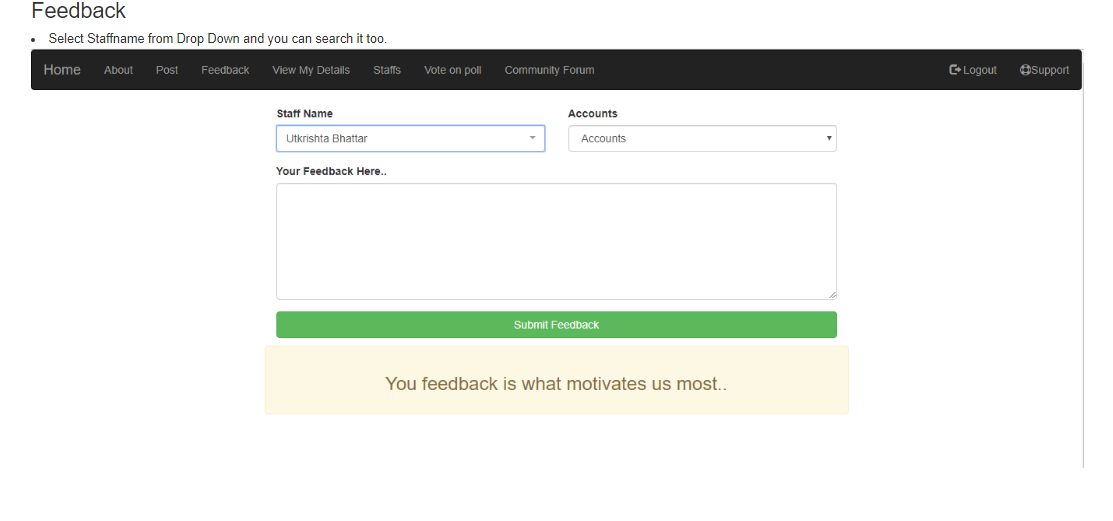
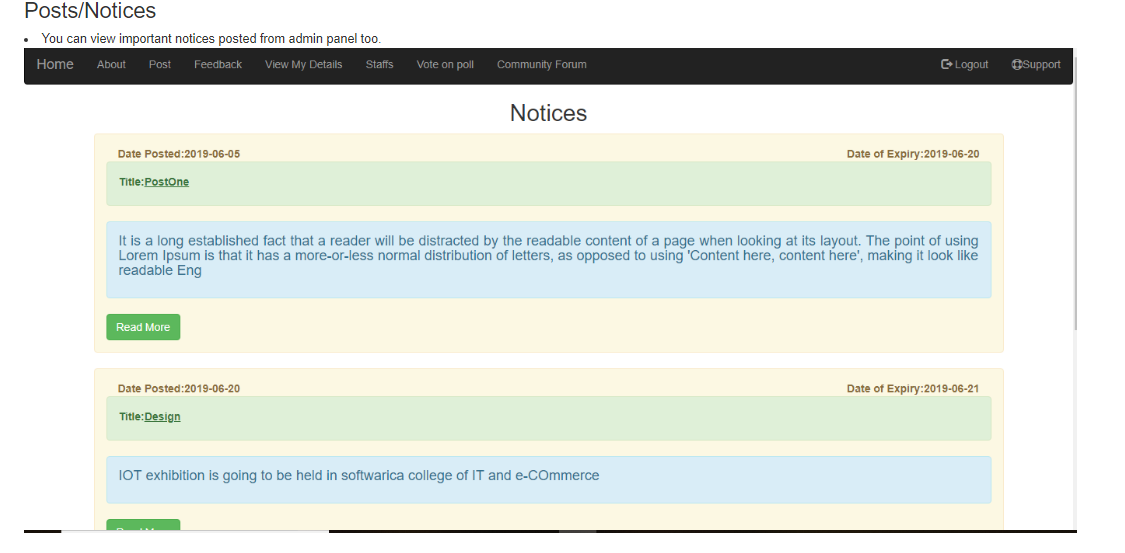
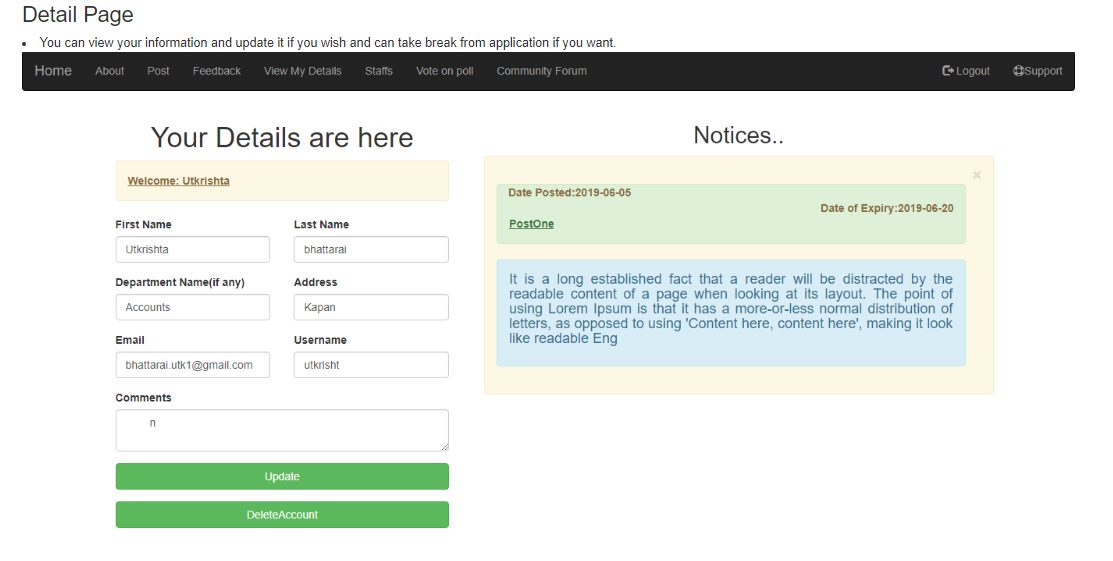


Figure : User manual





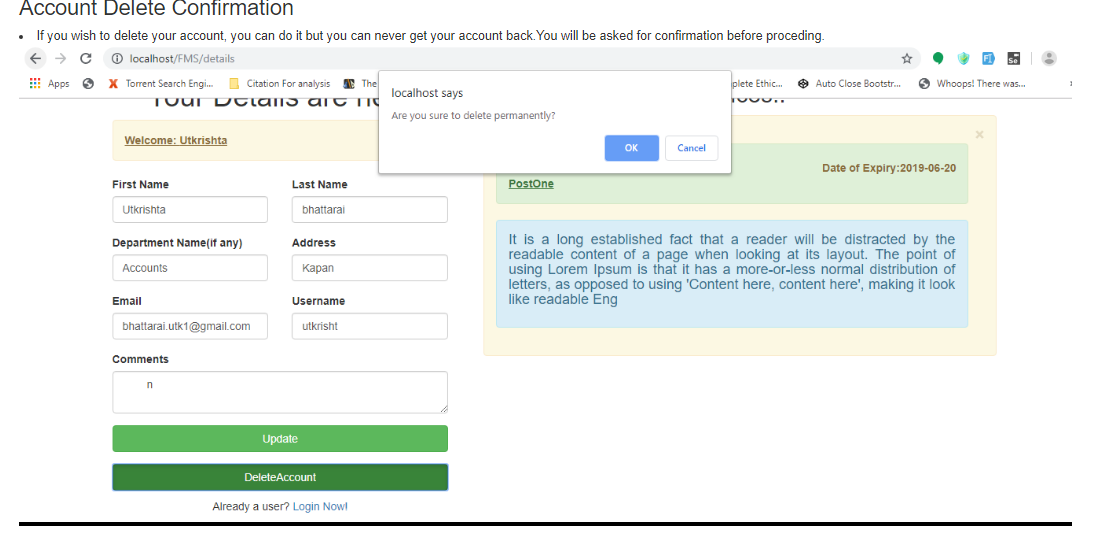


Figure :User manual

# **Chapter 7: Conclusion**

Finally, a complete project (Feedback management system) has been developed within given time and budget, fulfilling almost every requirement. Waterfall methodology was followed during development of this project, so all phases were seriously taken and only after completing first phase, next phase was initiated. Due to structured rule each phase was completed in time. Analysis was done in order to collect requirements from the end users and the product owner. The requirements were clearly taken so there was no issue on changing requirement, which didn’t cause any delay in project. Design phase was initiated in time, all necessary diagrams, models required for implementation phase were made. The information collected from analysis phase assisted in design phase. Necessary diagrams and modelling were done and the project entered into implementation phase. Implementation phase were properly handled by the team of developers and no issues were faced during implementation phase. The Web application was complete and it was time for testing. Two major testing i.e. Blackbox and Whitebox testing were done. All phases were documented properly as we might require it in coming future.

# **Chapter 8: Reference and Bibliography**

1. Marston, T. (2019). What is the 3-Tier Architecture? [online] Tonymarston.net. Available at: http://www.tonymarston.net/php-mysql/3-tier-architecture.html [Accessed 10 Jul. 2019].
2. YouTube. (2019). UML Tutorial - Use Case, Activity, Class and Sequence Diagrams - Essential Software Modelling. [online] Available at: https://www.youtube.com/watch?v=RMuMz5hQMf4 [Accessed 21 Jul. 2019].
3. Acronymfinder.com. (2019). NLA | Define NLA at Acronym Finder. [online] Available at: https://www.acronymfinder.com/NLA.html [Accessed 01 Jul. 2019].
4. Chartier, J. and Held, R. (2004). Testing. Seattle: Marquand Books.
5. Mifsud, J. (2019). Paper Prototyping as a Usability Testing Technique - Usability Geek. [online] Usability Geek. Available at: https://usabilitygeek.com/paper-prototyping-as-a-usability-testing-technique/ [Accessed 09 Jul. 2019].
6. TheFreeDictionary.com. (2019). NLA. [online] Available at: https://acronyms.thefreedictionary.com/nla [Accessed 10 Jul. 2019].
7. www.dictionary.com. (2019). Definition of manual | Dictionary.com. [online] Available at: https://www.dictionary.com/browse/manual [Accessed 11 Jul. 2019].
8. Diagram, h. and Phalak, N. (2019). How do I add a final variable to class diagram? [online] Stack Overflow. Available at: https://stackoverflow.com/questions/16252399/how-do-i-add-a-final-variable-to-class-diagram [Accessed 4 Jul. 2019].
9. Arthur, j. (2019). What is a Use Case? - Definition from Techopedia. [online] Techopedia.com. Available at: https://www.techopedia.com/definition/25813/use-case [Accessed 10 Jul. 2019].
10. Cms.gov. (2019). [online] Available at: https://www.cms.gov/research-statistics-data-and-systems/cms-information-technology/xlc/downloads/selectingdevelopmentapproach.pdf [Accessed 3 July. 2019].
11. The Economic Times. (2019). *Definition of Gantt Chart | What is Gantt Chart? Gantt Chart Meaning - The Economic Times*. [online] Available at: https://economictimes.indiatimes.com/definition/gantt-chart [Accessed 2 Apr. 2019].
12. Gordon, A., 2014. *What is a Work Breakdown Structure?.* [Online]   
    Available at: http://www.brighthubpm.com/templates-forms/2645-what-is-a-work-breakdown-structure/#imgn\_1  
    [Accessed 2 June. 2019].
13. Ravihansa, D., 2015. *System Development Methodologies.* [Online]   
    Available at: https://www.slideshare.net/devonravihansa18/system-development-methodologies-45452837  
    [Accessed 4 June. 2019].
14. Sharma, L., 2017. *Configuration Management.* [Online]   
    Available at: http://toolsqa.com/software-testing/configuration-management/  
    [Accessed 5 May. 2019].
15. sheffield.ac.uk, 2012. *How to: Write a Problem Statement.* [Online]   
    Available at: https://www.sheffield.ac.uk/polopoly\_fs/1.440722!/file/HowtoWriteaProblemStatement.pdf  
    [Accessed 8 May. 2019].
16. smartsheet, 2016. *How to Write a S.M.A.R.T. Project Objective.* [Online]   
    Available at: https://www.smartsheet.com/how-write-smart-project-objective  
    [Accessed 29 May. 2019].

# **Chapter 9: Appendix**

**Poll**

**UI**

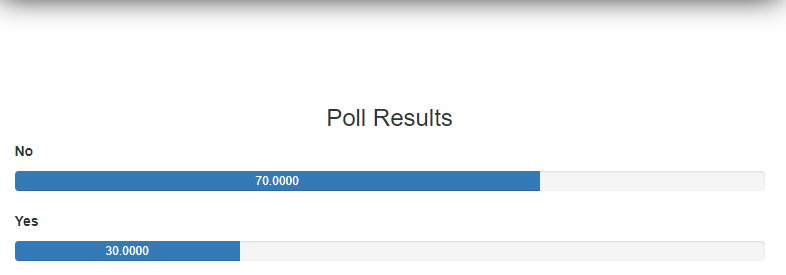


Figure : Poll UI

**Model**





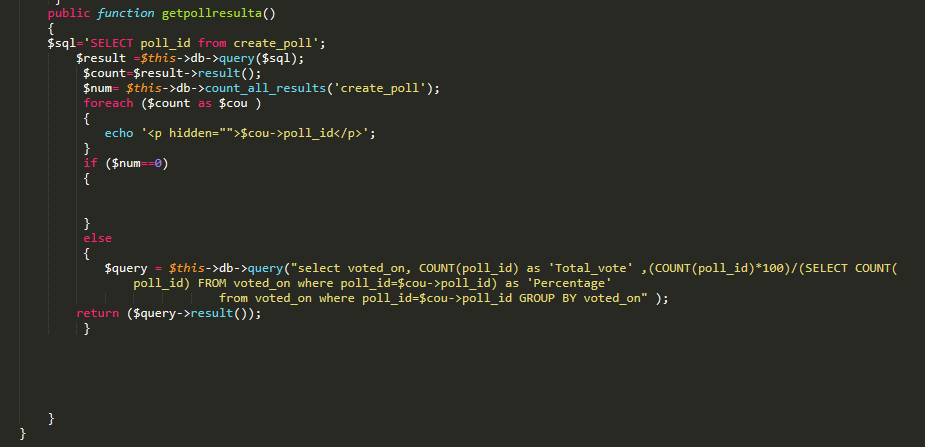


Figure : Model for poll

**Controller**





Figure : Controller for Poll

**View**

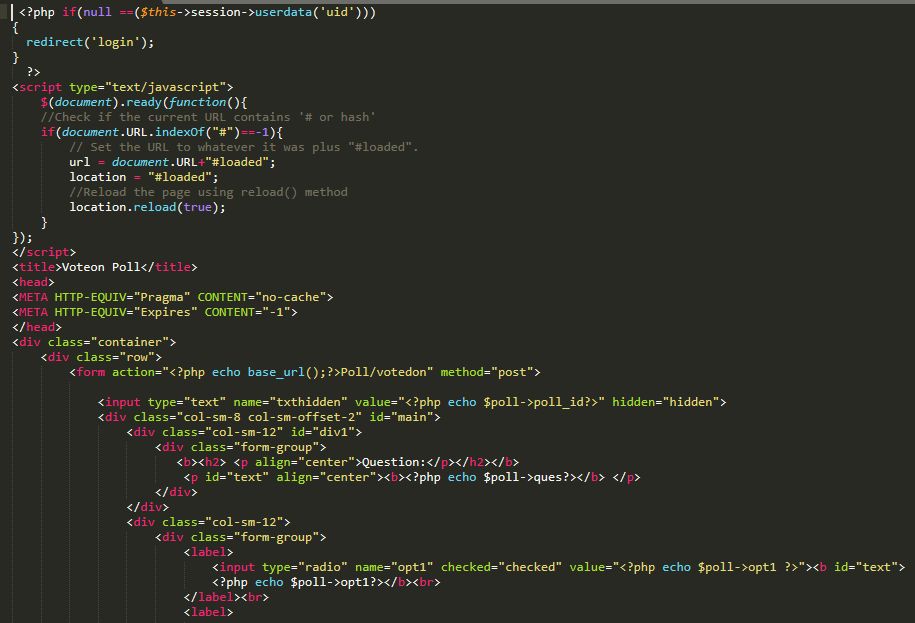




Figure : View for Poll

**Community Forum**

**UI**

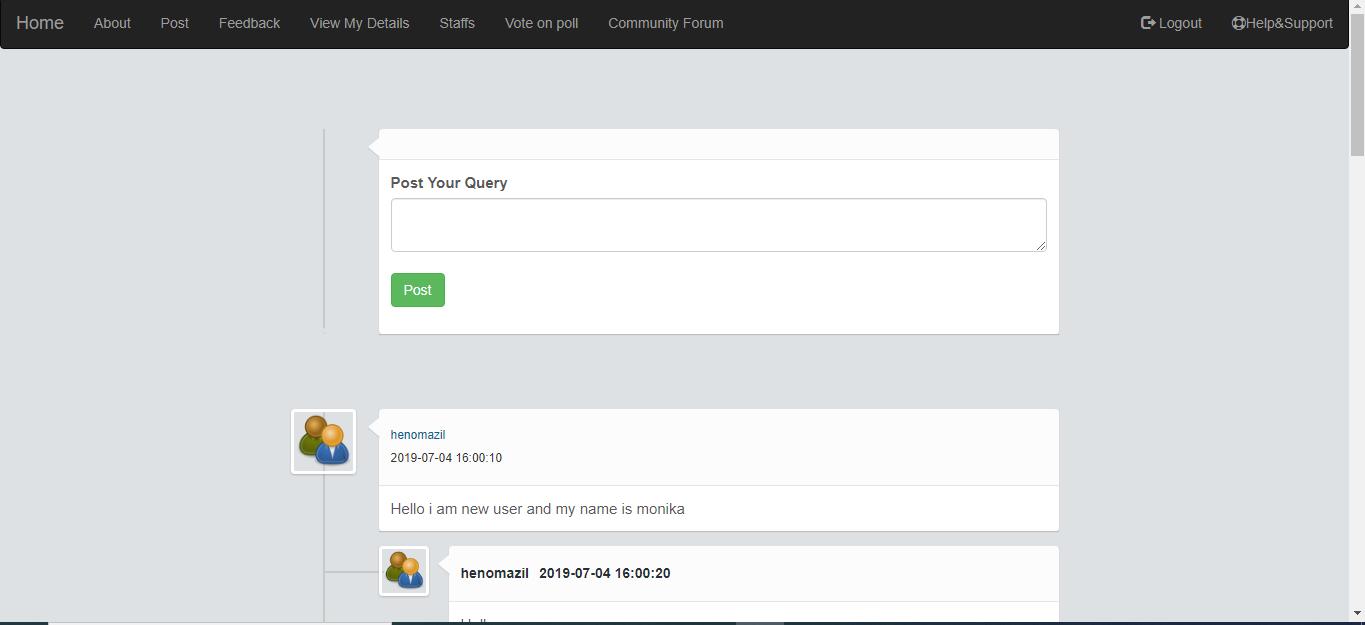


Figure : UI for community forum

**Model**

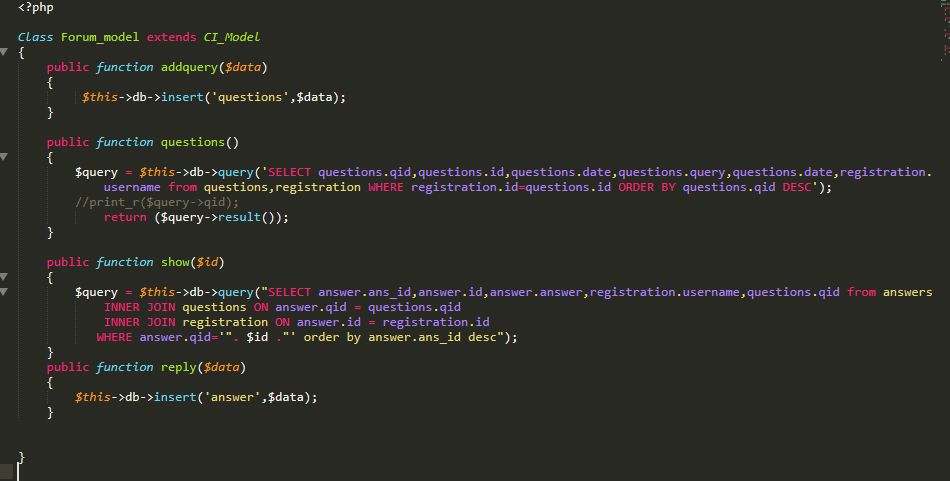


Figure : Model for community forum

**Controller**

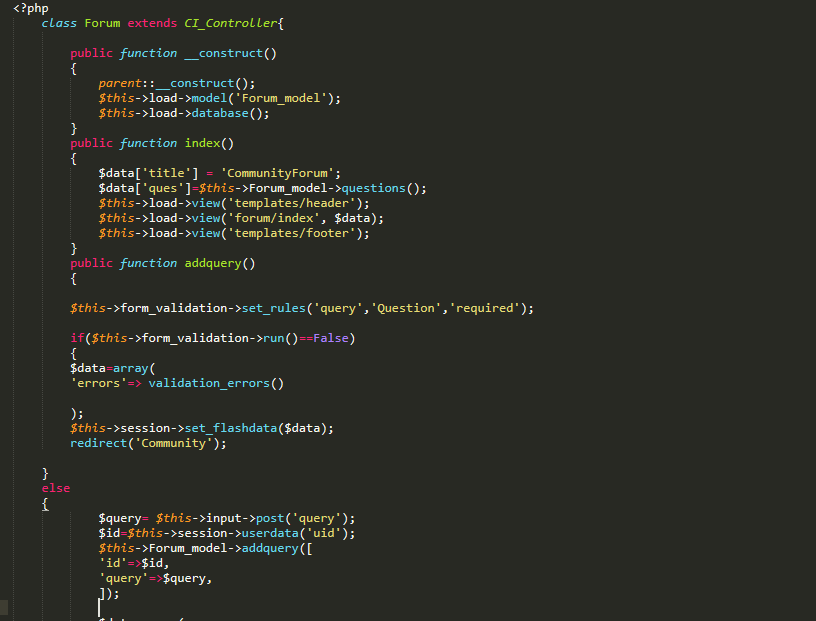
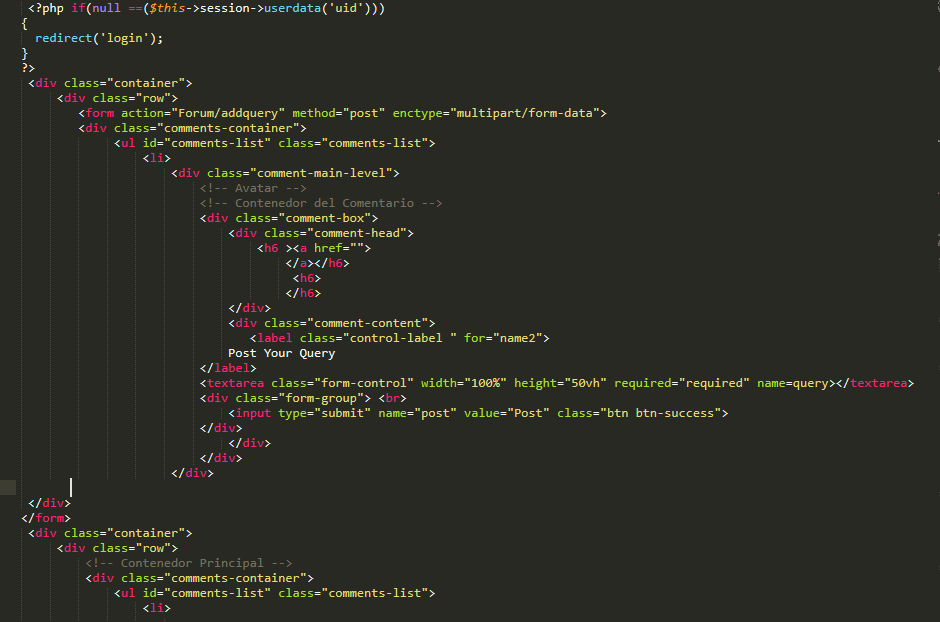
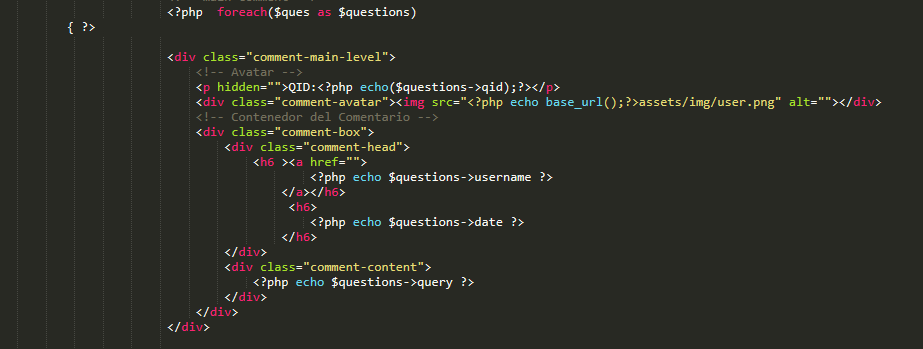




Figure : Controller for community forum

**View**





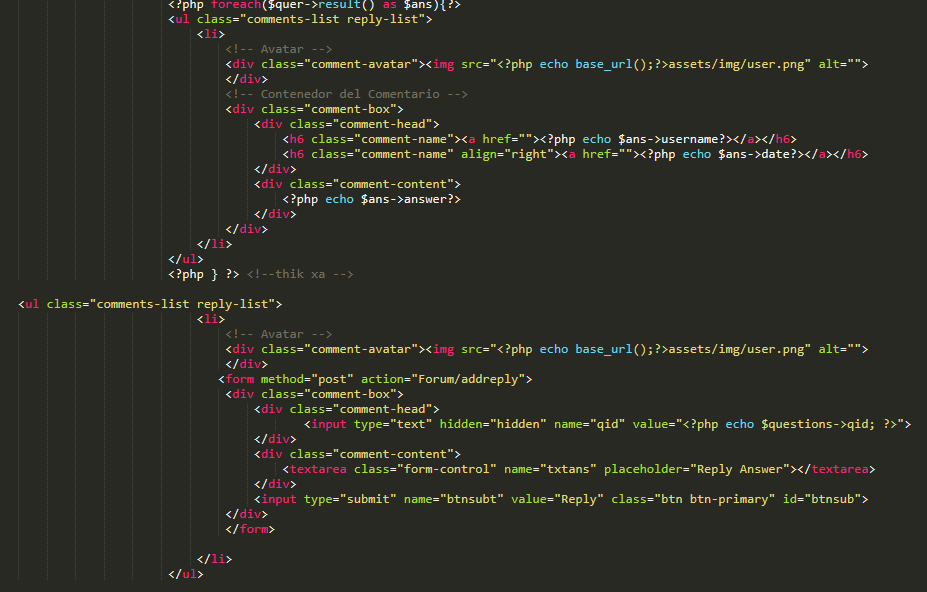




Figure : View for community forum

**Posts**

**UI**

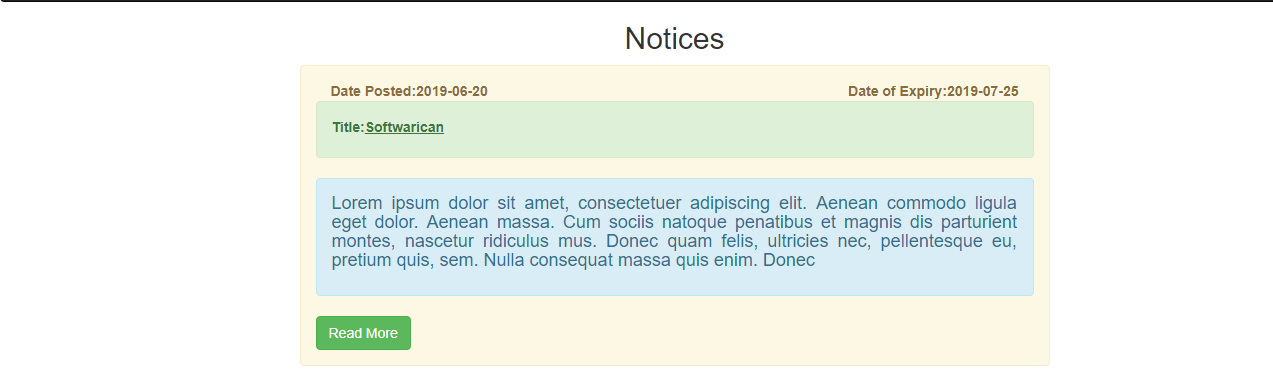


Figure : UI for post

**Model**



Figure : Model for post

**Controller**



Figure : Controller for posts

**View**



Figure : View for posts