

**TRIBHUVAN UNIVERSITY**

**FOLLOWUP**

**PROJECT REPORT**

**SUBMITTED TO**

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

BIRAT KSHITIZ COLLEGE

BIRATNAGAR, NEPAL

**IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR DEGREE IN COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**SUBMITTED BY**

CHANDAN KUMAR MEHTA (11448/073)

SANU KOIRALA (11455/073)

**April 2021**



**TRIBHUVAN UNIVERSITY**

**SUPERVISOR’S RECOMMENDATION**

I, hereby recommend that this project prepared under my supervision by **Chandan Kumar Mehta** and **Sanu Koirala** entitled “**FOLLOWUP**” in partial fulfillment of the requirements for the degree of BSc. In Computer Science and Information Technology be processed for the evaluation.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr.-Ing Deepranjan Dongol**

SUPERVISOR

BIRAT KSHITIZ COLLEGE

**STUDENT’S DECLARATION**

We, the undersigned solemnly declare that the report of the project work entitled “**FOLLOWUP**” submitted to Office of the Dean, Faculty of Science and Information Technology, Tribhuvan University, is based on our work carried out during the course of study under the supervision and guidance of **Dr.-Ing Deepranjan Dongol.** We assure that the statements made and conclusions drawn are an outcome of the project work. We further declare that, this project work is a record of original work and does not contain any part of any work submitted for the award of any other degree/diploma/certificate in this University or any other University.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Chandan Kumar Mehta** **Sanu Koirala**

Roll no: 11448/073 Roll no: 11455/073

Reg no.: 5-2-1074-0004-2016 Reg no.: 5-2-1074-0011-2016



**TRIBHUVAN UNIVERSITY**

**LETTER OF APPROVAL**

This is to certify that this project is prepared by Chandan Kumar Mehta and Sanu Koirala entitled “**FOLLOWUP**” in the partial fulfillment of the requirements for the degree of B.Sc. In Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| **-------------------------------------**  **Dr.-Ing Deepranjan Dongol**  Supervisor | **------------------------------------**  **Er. Deeyoranjan Dongol**  Principal |
| **--------------------------------**  **External Examiner**  **(**Tribhuvan University**)** | **-------------------------------------**  **Internal Examiner** |

# ACKNOWLEDGEMENT

We are using this opportunity to express our gratitude to the Department of Computer Science and Information Technology, Birat Kshitiz College for providing us the opportunity to explore our potentiality in the field of our interest through technology via this project.

We are thankful to our Supervisor **Dr.-Ing Deepranjan Dongol** for his support, coordination and valuable supervision for this project.

We are also thankful to **Mr. Saroj Koirala** (faculty, Birat Kshitiz College) for his support and valuable effort for this project.

We are also thankful to our college management team, faculty teachers and friends who helped us for accomplishing this project.

We would like to acknowledge and extend gratitude to everyone for his/her support and encouragement for this project.

# ABSTRACT

FollowUp is an online web application aimed to change the record management process. It consists of Admin and User. Admin can create user and check the activity that is performed by each user. User can add and view follow up records of the client. Each user has their separate follow up task where one is not able to view other's task whereas Admin is able to view all follow ups created by the users. Both Admin and User gets notified a day before the follow up is assigned to be attended. User can also maintain their to-do list and are notified as per the scheduled date. The system is built in HTML, CSS, Bootstrap and JavaScript as Front End tool and My SQL and Laravel as Back End tool.

FollowUp uses two algorithms: Linear Search Algorithm for searching specific job in the system and Priority Ranking algorithm for recommending jobs with highest number of follow up in the system.

**Key points:** Follow up, Notification, Reminder, Searching, Recommendation

Table of Contents

[ACKNOWLEDGEMENT v](#_Toc70777401)

[ABSTRACT vi](#_Toc70777402)

[LIST OF FIGURES ix](#_Toc70777403)

[LIST OF TABLES x](#_Toc70777404)

[LIST OF ABBREVIATIONS xi](#_Toc70777405)

[CHAPTER 1: INTRODUCTION 1](#_Toc70777406)

[1.1 Introduction 1](#_Toc70777407)

[1.2 Problem Statement 1](#_Toc70777408)

[1.3 Objective 1](#_Toc70777409)

[1.4 Scope and Limitation 2](#_Toc70777410)

[CHAPTER 2: SYSTEM ANALYSIS 3](#_Toc70777411)

[2.1 Literature Review 3](#_Toc70777412)

[2.2 Requirement Analysis 3](#_Toc70777413)

[2.2.1 Functional Requirements 3](#_Toc70777414)

[2.2.2 Non-functional Requirements 5](#_Toc70777415)

[2.3 Feasibility study 5](#_Toc70777416)

[2.3.1 Technical Feasibility 5](#_Toc70777417)

[2.3.2 Economical Feasibility 5](#_Toc70777418)

[2.3.3 Operational Feasibility 6](#_Toc70777419)

[2.4 Structuring System Requirements 6](#_Toc70777420)

[2.4.1 Data Modeling 6](#_Toc70777421)

[2.4.2 Process Modeling 8](#_Toc70777422)

[CHAPTER 3: SYSTEM DESIGN 11](#_Toc70777423)

[3.1 System Design 11](#_Toc70777424)

[3.1.1 Schema Diagram 11](#_Toc70777425)

[3.1.2 Input/output Interface 12](#_Toc70777426)

[3.2 Process Flow Diagram 14](#_Toc70777427)

[3.2.1 Activity Diagram 14](#_Toc70777428)

[CHAPTER 4: IMPLEMENTATION AND TESTING 16](#_Toc70777429)

[4.1 Implementation 16](#_Toc70777430)

[4.1.1 Methodology 16](#_Toc70777431)

[4.1.2 Working Mechanism 17](#_Toc70777432)

[4.1.3 Implementation Tools 17](#_Toc70777433)

[4.2 List of Controllers 19](#_Toc70777434)

[4.2.1 Job Controller 19](#_Toc70777435)

[4.2.2 FollowUp Controller 20](#_Toc70777436)

[4.2.3 Notification Controller 20](#_Toc70777437)

[4.2.4 Todo Controller 20](#_Toc70777438)

[4.3 Testing 20](#_Toc70777439)

[4.3.1 Unit Testing 20](#_Toc70777440)

[4.3.2 Integration Testing 22](#_Toc70777441)

[4.3.3 System testing: 22](#_Toc70777442)

[CHAPTER 5: CONCLUSION AND RECOMMENDATION 23](#_Toc70777443)

[5.1: Conclusion 23](#_Toc70777444)

[5.2: Recommendation 23](#_Toc70777445)

[REFERENCES 24](#_Toc70777446)

[BIBLIOGRAPHY 25](#_Toc70777447)

[APPENDIX I: CODE SNAPSHOT 26](#_Toc70777448)

[APPENDIX II: PROJECT SNAPSHOT 36](#_Toc70777449)

# LIST OF FIGURES

[Figure 1: Use Case Diagram of FollowUp 4](#_Toc70777450)

[Figure 2: ER Diagram of FollowUp 7](#_Toc70777451)

[Figure 3: Context Diagram of FollowUp 8](#_Toc70777452)

[Figure 4: Level-0 Data Flow Diagram of FollowUp 9](#_Toc70777453)

[Figure 5: Level-1 Data Flow Diagram of Followup management process 10](#_Toc70777454)

[Figure 6: Schema Diagram of FollowUp 11](#_Toc70777455)

[Figure 7: Input Interface Design 12](#_Toc70777456)

[Figure 8: User Output Interface 13](#_Toc70777457)

[Figure 9: Activity Diagram for Admin 14](#_Toc70777458)

[Figure 10: Activity Diagram for User 15](#_Toc70777459)

[Figure 11: Prototype Model of Software Development 16](#_Toc70777460)

[Figure 12: Snapshot of Login 36](#_Toc70777461)

[Figure 13: Snapshot of User Dashboard 36](#_Toc70777462)

[Figure 14: Snapshot of Editing User 37](#_Toc70777463)

[Figure 15: Snapshot of Adding Follow up 37](#_Toc70777464)

[Figure 16: Snapshot of Adding Follow up 38](#_Toc70777465)

[Figure 17: Snapshot of Adding Follow up 38](#_Toc70777466)

[Figure 18: Snapshot of Adding new Representative 39](#_Toc70777467)

[Figure 19: Snapshot Adding new Follow up 39](#_Toc70777468)

[Figure 20: Snapshot of User's To-do list 40](#_Toc70777469)

[Figure 21: Snapshot of adding to-do list 40](file:///C:\Users\Lenovo\Downloads\FollowUp%20(2).docx#_Toc70777470)

# LIST OF TABLES

[Table 1: TEST CASE FOR LOGIN SECTION 20](#_Toc70777471)

[Table 2: TEST CASE FOR USER LOGIN 21](#_Toc70777472)

[Table 3: TEST CASE FOR INTEGRATION TESTING 22](#_Toc70777473)

[Table 4: TEST CASE FOR SYSTEM TESTING 22](#_Toc70777474)

# LIST OF ABBREVIATIONS

**CSS Cascading Style Sheet**

**DFD Data Flow Diagram**

**ER Entity Relationship**

**HTML Hypertext Markup Language**

**IT Information Technology**

**MVC Model View Controller**

**MySQL My Structured Query Language**

**PHP Hypertext Preprocessor**

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

In today's world, Organizations have to co-ordinate and direct their products to the clients satisfying their every requirement to maintain a smooth and professional relationship. The approach so far has been to maintain a meeting on every now and then, to clarify who owes what duty in an organization. FollowUp focuses on that part.

FollowUpis a web application developed with the view of changing the paper based system of keeping records into computer based. It focuses on changing the approach of remembering whom to contact, when to contact and what is the deal behind that. It holds the record of each log made with the client and maintains a brief history of the log. It has a reminder for each follow-up needed to be made at a specific date with the client.

## 1.2 Problem Statement

The development of FollowUp is mainly influenced because of the lack of properly structured system for maintenance of records in organizations. Because of the absence of proper system, text editors are the only options for keeping records after the papers resulting in:

* Absence of notification and reminder for upcoming events.
* Hectic to manage a large pile of records.
* Vulnerable due to the lack of security.
* Time consuming for searching a file for specific record.
* High risk of mismanagement of records.
* Editing a specific record is nearly impossible.

FollowUp stands in need of a proper system for managing the records of follow ups within an organization.

## 1.3 Objective

Major objectives behind developing FollowUp are:

* To develop a proper structured system for keeping follow up records.
* To minimize time and effort for managing records.
* For easy maintenance of client's record.
* To minimize the mismanagement of record.
* To provide security on specific records.

## 1.4 Scope and Limitation

FollowUp has broad range of scope within an organization. Some of them are listed below:

* Any organization that requires maintaining follow up records can use it.
* To-do list provides a way to maintain personal tasks.
* New events are reminded beforehand.
* Search of record is effortless.
* Updating records of specific client is performed in no time.

There are some limitations of this system as:

* Only the Admin and User can use this application.
* Timely update of record completely depends upon the user.
* The end user must have basic technical knowledge to use this application.

# CHAPTER 2: SYSTEM ANALYSIS

## 2.1 Literature Review

The past work and the report on record management application [1] and how this report helps us to build the recruitment web application are discussed under this topic.

In the report, record management with Advanced Search Capabilities, the study shows that 75% of users stated that internet has been ‘very helpful’ in their use, and 18.4% see it’s ‘somewhat helpful’, that is a sum of 93.4%.

In the report, record management: Evidence from Agentology, it state that apart from being automated it is, at the same time, it is flexible enough to accommodate the varying nature of maintaining client's records. It connects client and organization all while using the seamless blend of human and technology. [2]

In the report, Using the web to look for work: Implications for record management, the authors focused upon three specific research questions: How does organization manage the client's record? How effective are these process? And how likely is appropriate using it? The data used to examine these questions come from various organizations submitting record-related queries to a major web search engine at three points in time over a five-year period [3]. Results of their study indicated that individuals seeking record management information generally submitted only one query with several terms and over 45 percent of record management queries contained a specific location reference. [4]

There are several follow up applications such as Agentology, Lead Follow-up and many more available in internet but in terms of market popularity in Nepal, they are not quite heard and known. We studied these web applications which are similar to our project and gather information about record management and its different aspect. We also got the information about how these website works to manage the client's record within an organization, how they gather the information from the client and how they represent these data.

## 2.2 Requirement Analysis

### 2.2.1 Functional Requirements

The system allows the following functionalities:

* Allows Admin and User to login.
* Allows Admin to manage users.
* Allows User to add and view follow ups.
* Allows User to edit follow ups.
* Allows Admin to delete follow ups.
* Allows User to maintain to-do list.
* Provides reminder of to-do list.
* Allows Admin to view report of follow ups.
* Provides reminder of follow ups.



Figure 1: Use Case Diagram of FollowUp

### 2.2.2 Non-functional Requirements

* **Security**

User of the system is only able to access using authorized username and password. Only the authorized user can change their profile content or password.

* **Usability**

This system is very easy to use and is easy to understand. It is user friendly.

* **Performance**

The developed system is able to provide correct and accurate functioning. Every feature presented by the system have quick response time.

* **Maintainability**

The system needs to be maintained. Updating database, monitoring and error handling should be carried out on a daily basis.

* **Reliability**

Each user is able to get the information of the client separately. Any issue that may occur may be fixed by the admin. Each registration for the follow up by user will be notified to admin.

## 2.3 Feasibility study

### 2.3.1 Technical Feasibility

The system is developed using Laravel, JavaScript and MySQL which make the system technically feasible. Laravel provides the technical accuracy and compatibility, JavaScript provides the technical guarantee of reliability whereas MySQL provides security of database of the system. The feasibility is analyzed by studying the complete functionality of the system.

* Compiles with any hardware and software ,including the latest technologies
* Runs on any browser such as: Chrome, Safari, Firefox, Brave
* Compatible to any version of used tools

### 2.3.2 Economical Feasibility

To develop this project, it does not require any additional hardware or software as the interface for this system is developed using the existing resources and technologies available. Also, from the estimation, it is clear that the benefits the organization is going to receive from the purposed system will overcome the initial cost of developing the system. In our project, MySQL, Laravel and Apache server are used which are available free of cost making the initial budget low. So, the project is economically feasible.

### 2.3.3 Operational Feasibility

The user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits. The well-planned design ensures the optimal utilization of the computer resources and helps in the improvement of performance status. The developed system is web based and user friendly making it operationally feasible.

## 2.4 Structuring System Requirements

### 2.4.1 Data Modeling

**ER Diagram**.

Figure 2 shows the ER diagram of the FollowUp. The ER diagram shows the relationship among the entities titled as Admin, User, follow up, organization and people. Each entity has their own unique attributes and relationship with each other.

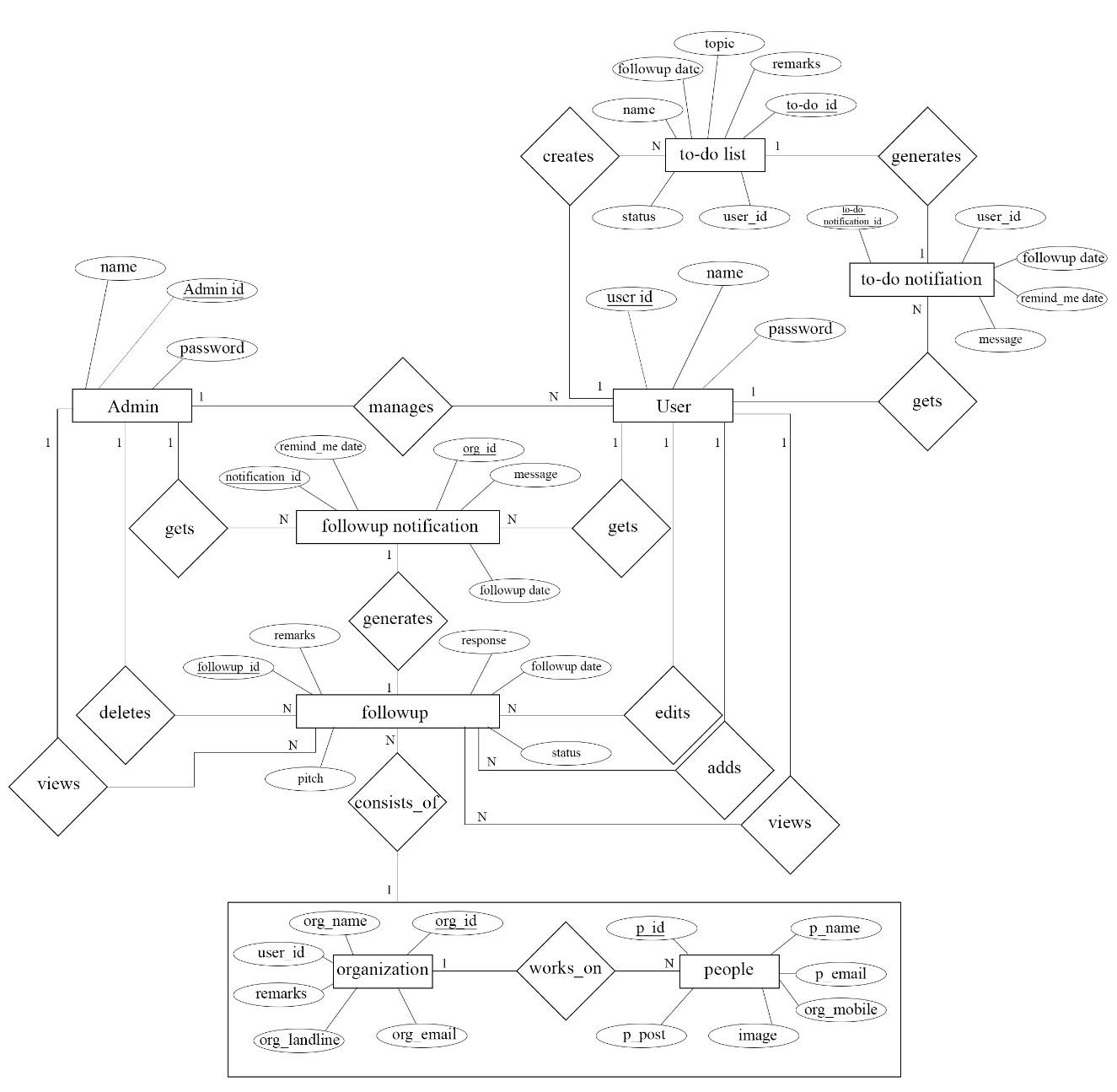


Figure 2: ER Diagram of FollowUp

### 2.4.2 Process Modeling

**Context Diagram**

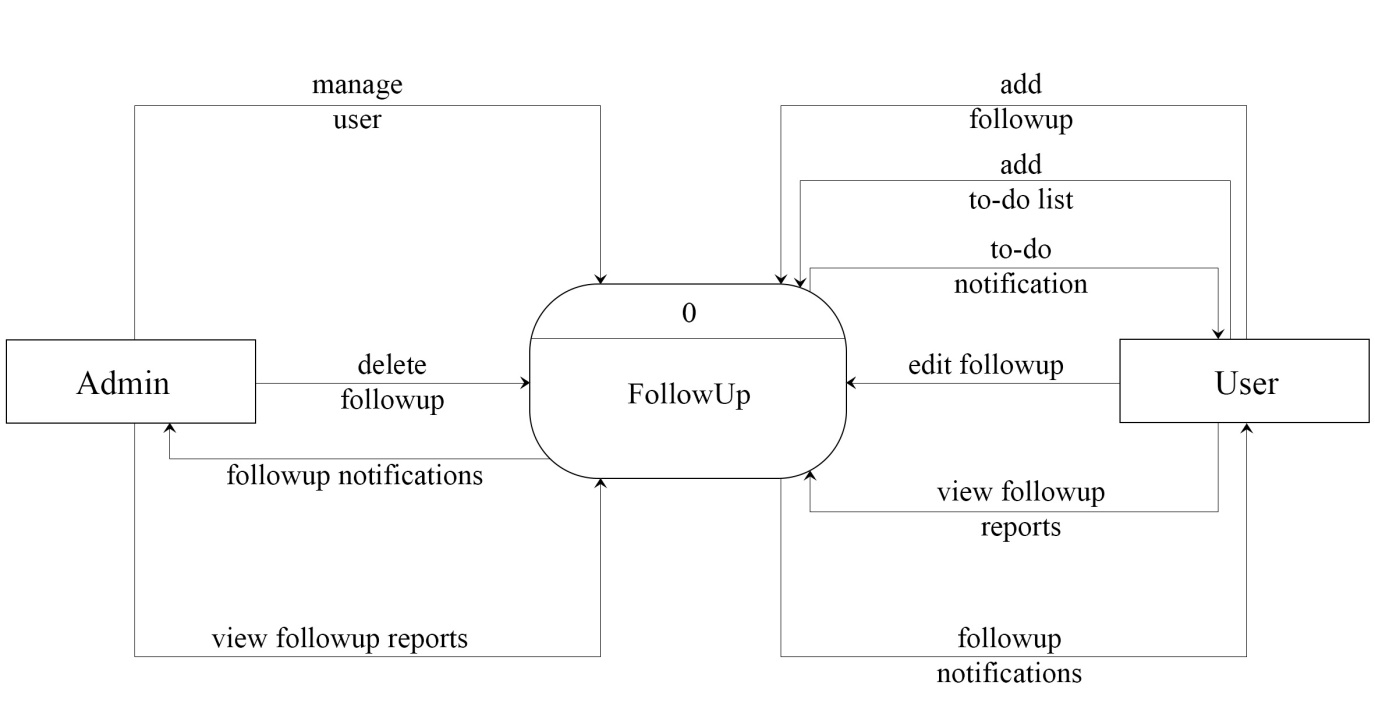


Figure 3: Context Diagram of FollowUp

Figure 3 shows the context diagram of the FollowUp system showcasing the flow of data between Admin and User. Here, Admin can manage the user and perform certain operation regarding follow up whereas User adds, edits and views follow up. The User also adds to-do list of their own.

**Data Flow Diagram level 0**

Figure 4 shows the level 0 Data Flow Diagram of FollowUp system. There are six major processes both the User and the Admin can perform in the system. These six process overall make up the system. The data table specifies the storage location of the data that are processed by each specific process.

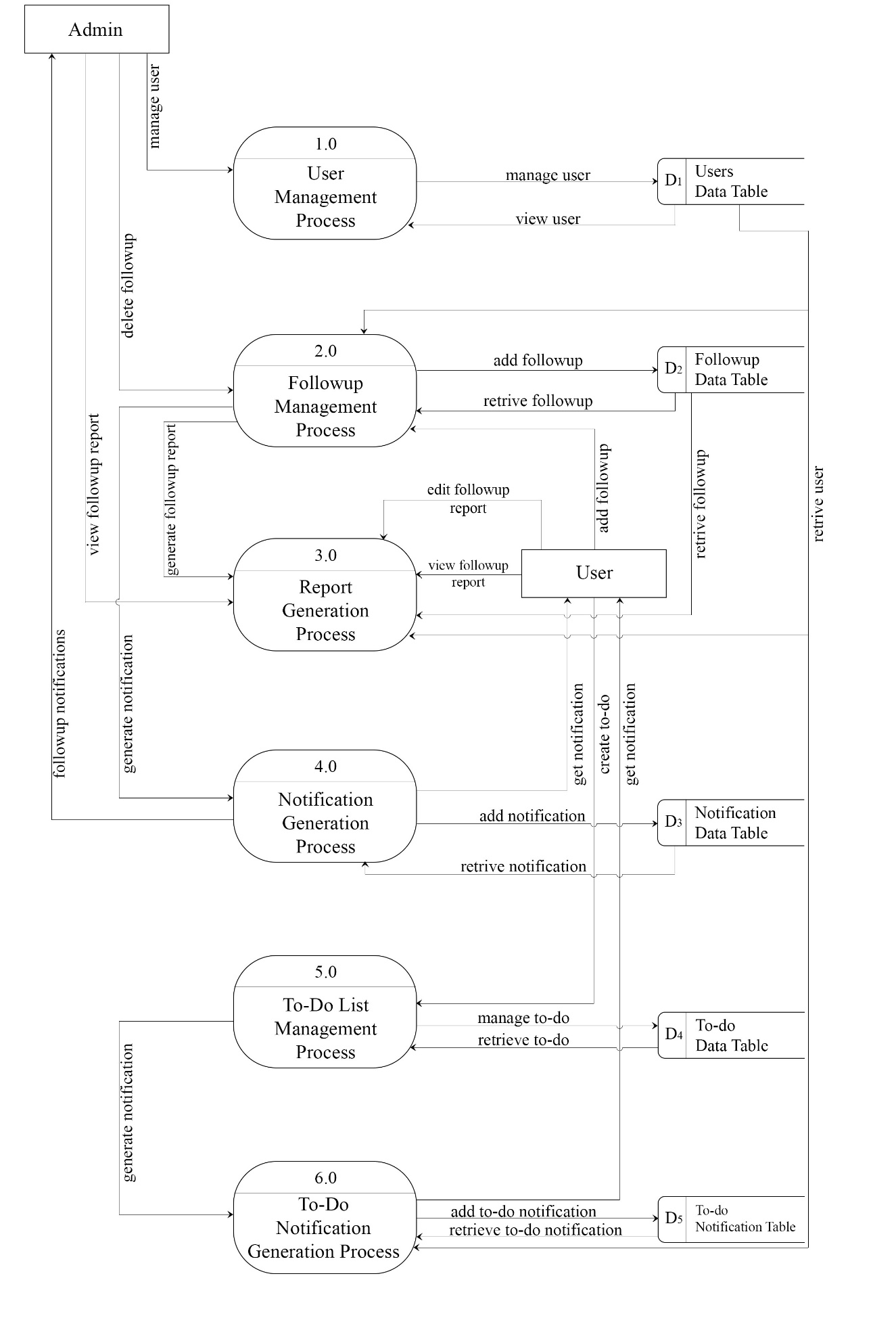


Figure 4: Level-0 Data Flow Diagram of FollowUp

**Data Flow Diagram level 1**

The diagram below shows the internal working of a specific process, Followup Management Process. Here, the Admin deletes the specified follow up created by the user. User adds organization details and representative details that consist of follow up

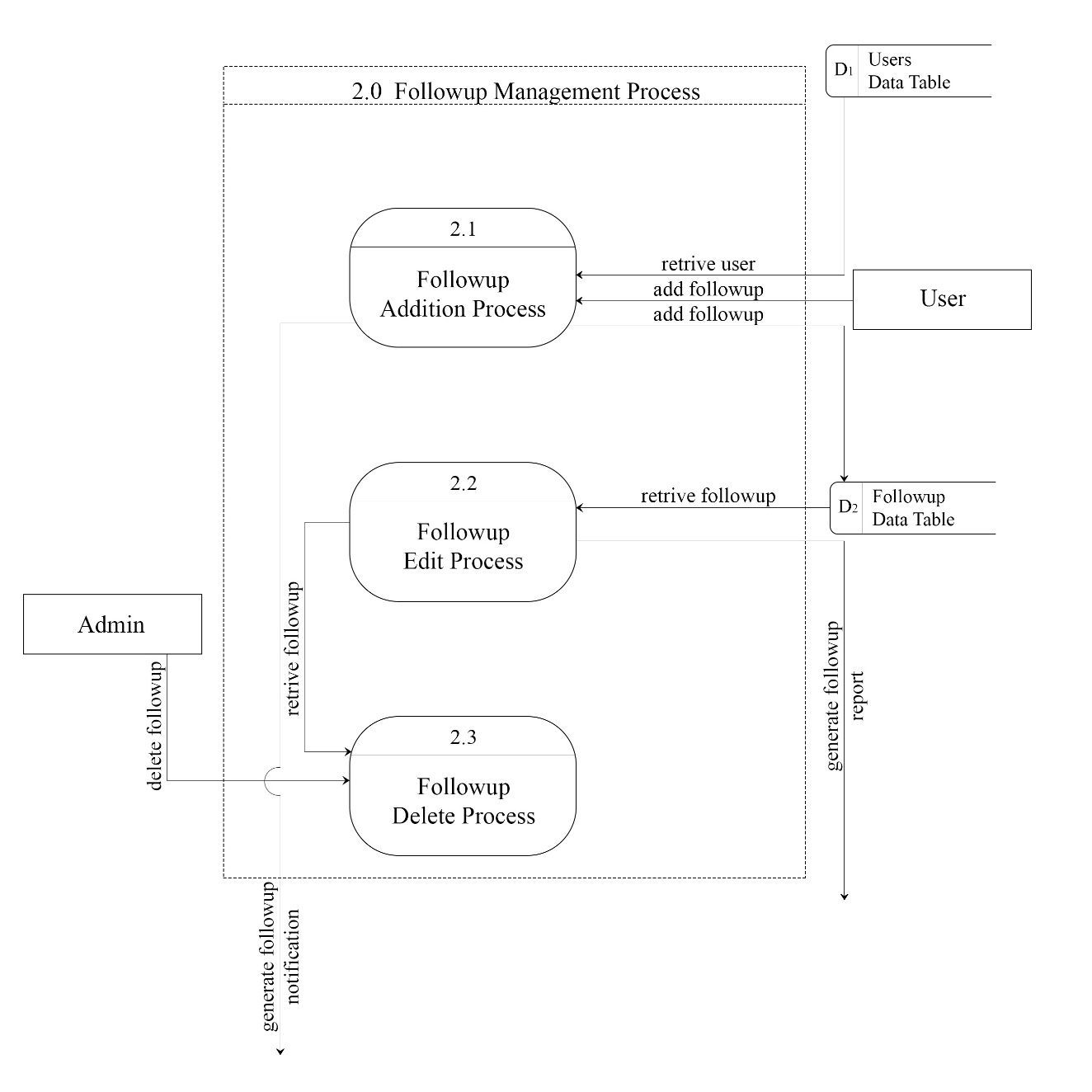
****

Figure 5: Level-1 Data Flow Diagram of Followup management process

# CHAPTER 3: SYSTEM DESIGN

## 3.1 System Design

In this system, there are two entities as: user and admin. The admin can create user and also can view the user activities. The user can add and view client's record. The admin can manage the record and the user. The user can add and manage their to-do list. Each follow ups created generates notification for both admin and user beforehand. User also gets reminder for their to-do tasks.

### 3.1.1 Schema Diagram

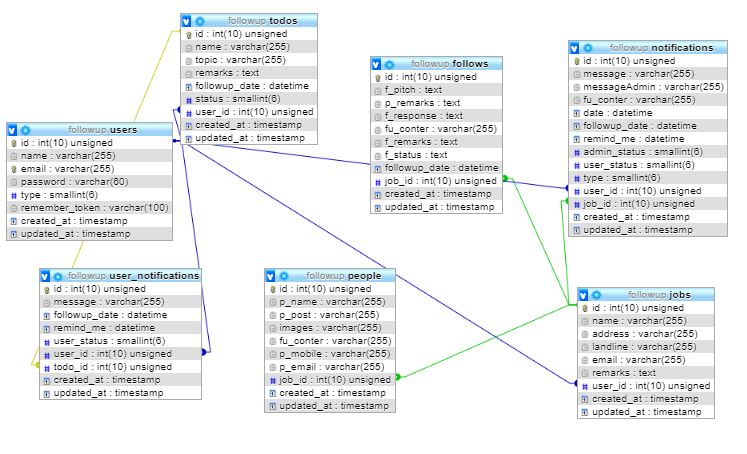
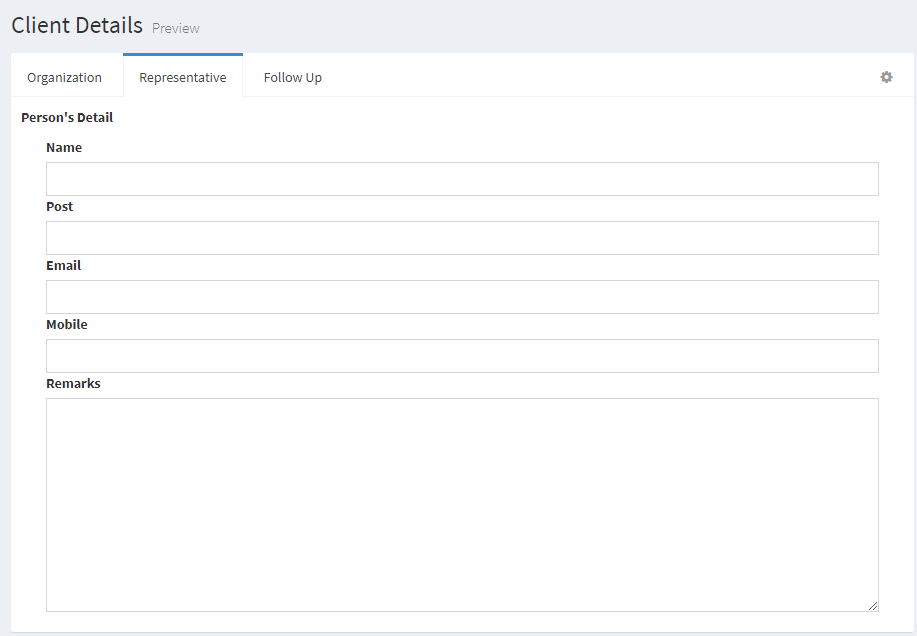
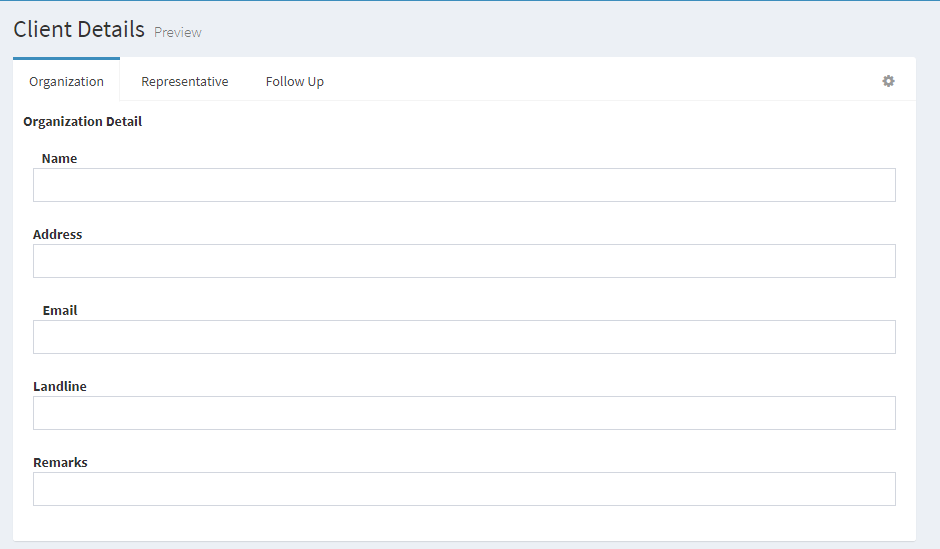


Figure 6: Schema Diagram of FollowUp

### 3.1.2 Input/output Interface



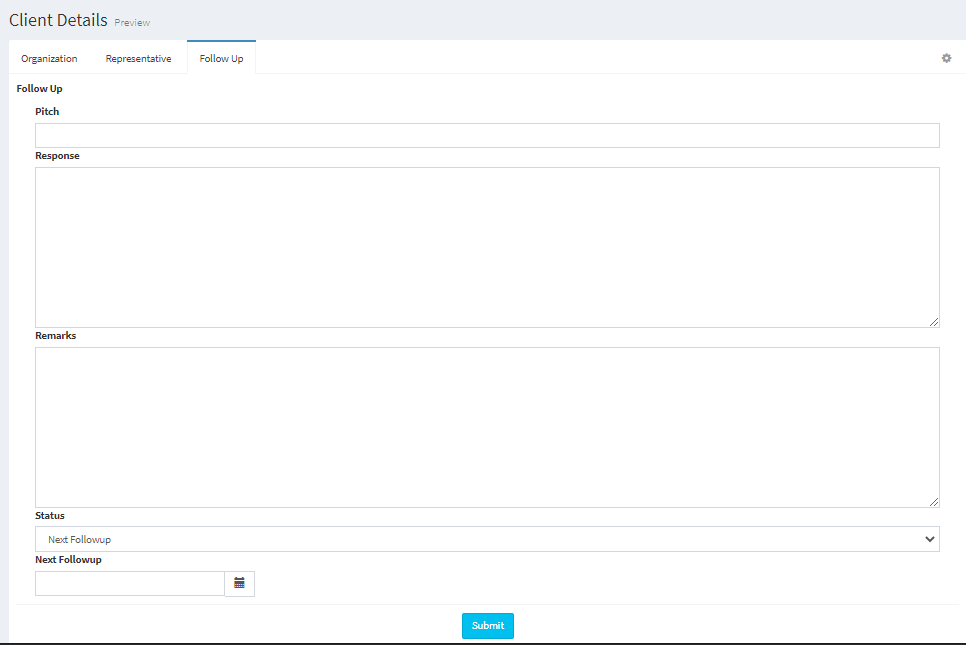


Figure 7: Input Interface Design

Figure 7 shows the input interface of the FollowUp system. User needs to enter the organization details and representative's detail associated with the organization and the follow up details in the system. The status field determines the status of the follow up i.e. whether the follow up should be followed by next date or follow up is successful or canceled.

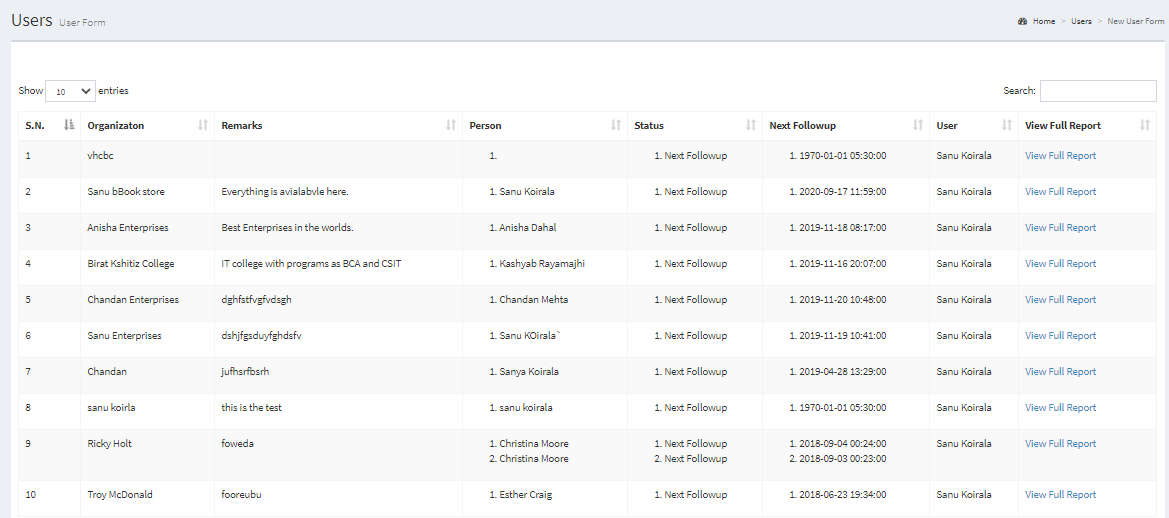


Figure 8: User Output Interface

Figure 8 shows the output interface of the FollowUp system. User can view follow up details of all organization from the system.

## 3.2 Process Flow Diagram

### 3.2.1 Activity Diagram

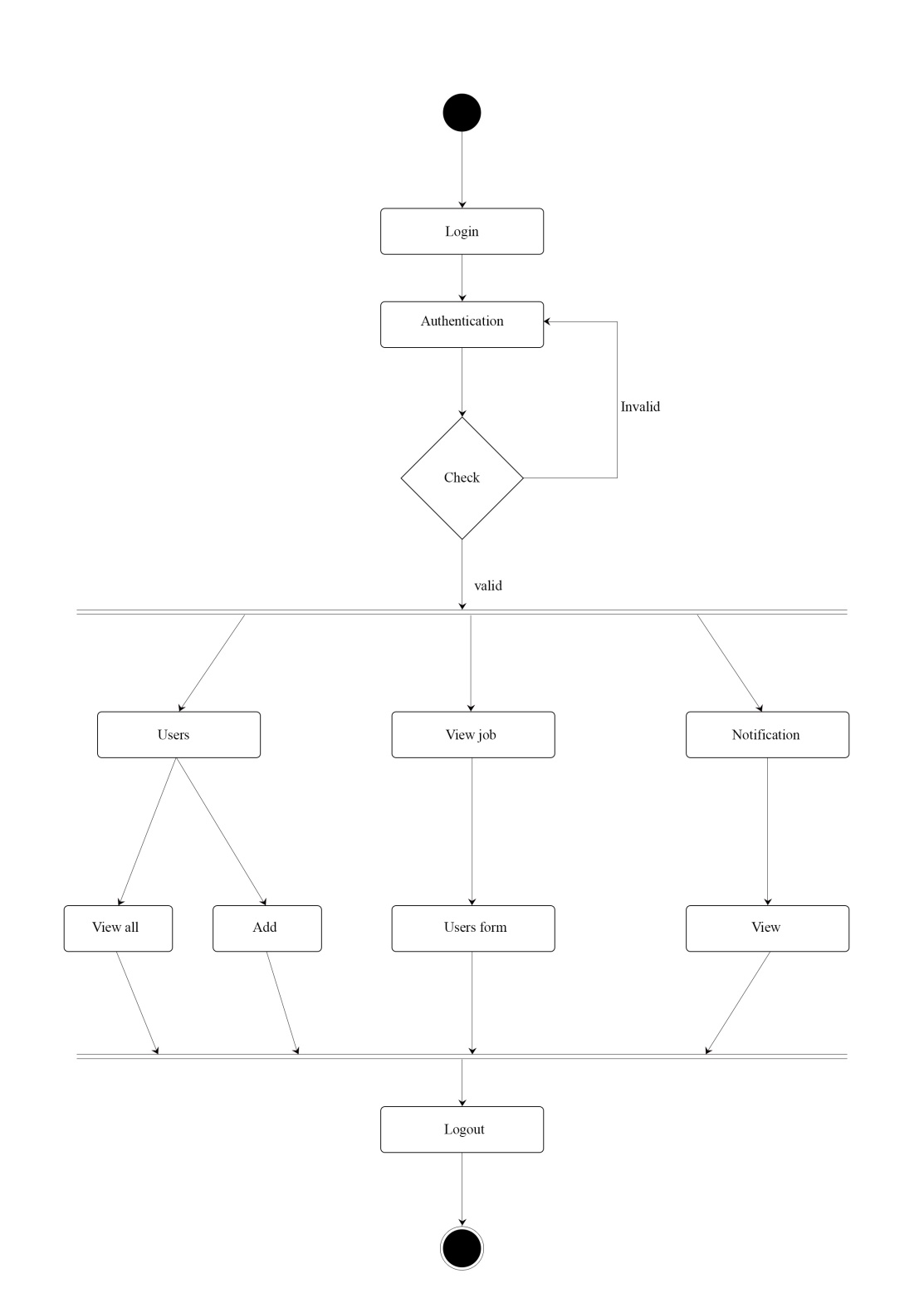


Figure 9: Activity Diagram for Admin

Figure 9 shows the Activity Diagram for Admin. After being authenticated inside the system, the Admin can perform two activities. One, to manage the user of the system and other to view the follow up report generated by the system. The Admin can also view the notification of the upcoming follow up.

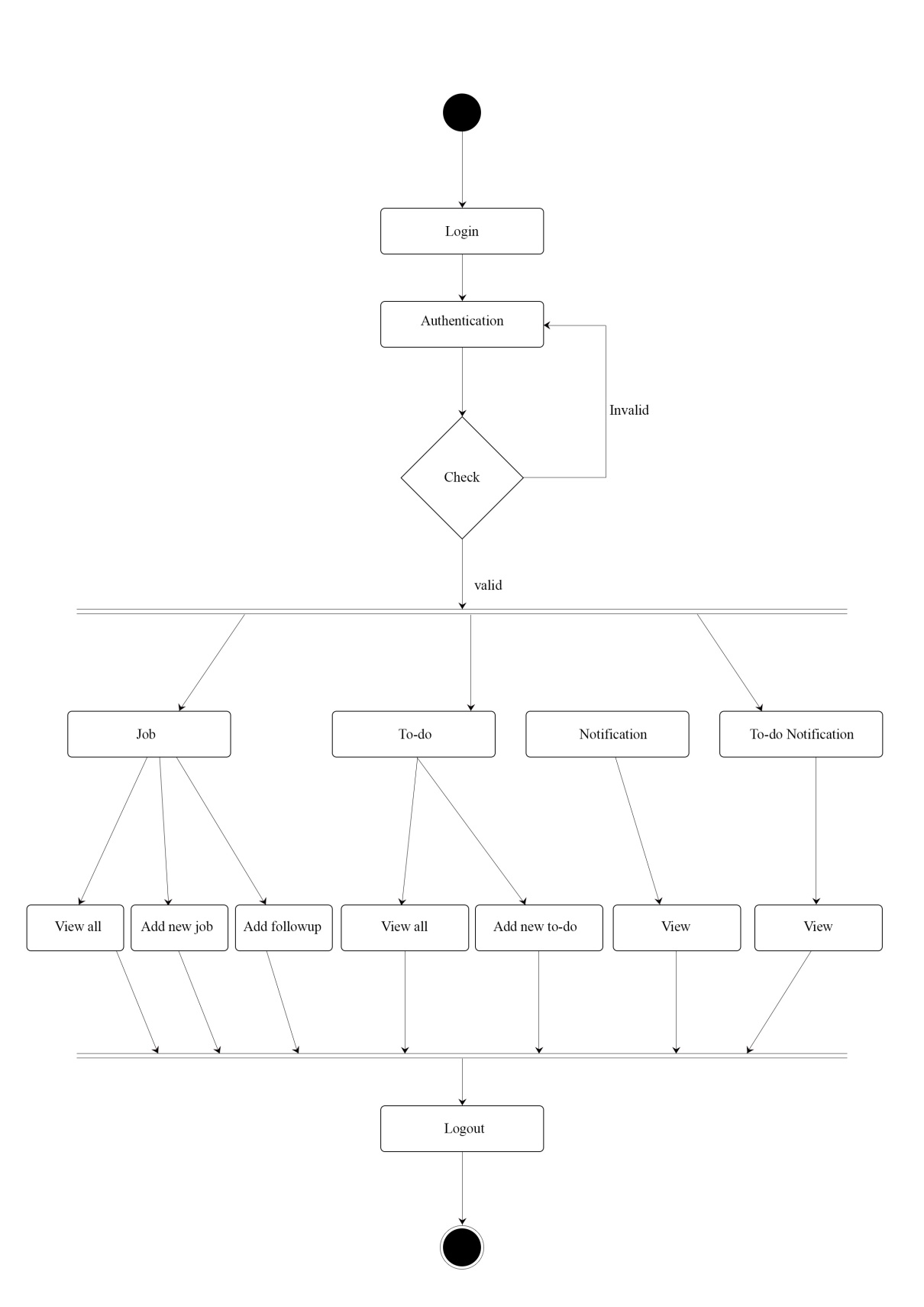


Figure 10: Activity Diagram for User

Figure 10 shows the activity diagram for User. After being authenticated, the User can perform activities like adding new job, adding follow up, adding to-do list, editing follow up, editing to-do list and so on. The user can also view the notification of upcoming follow up and the to-do list needed to be done.

# CHAPTER 4: IMPLEMENTATION AND TESTING

## 4.1 Implementation

To implement the system, various tools such as HTML, CSS, JavaScript and Bootstrap framework are used. The system is built in Laravel framework where MySQL is used for database and Apache server is used for server.

### 4.1.1 Methodology

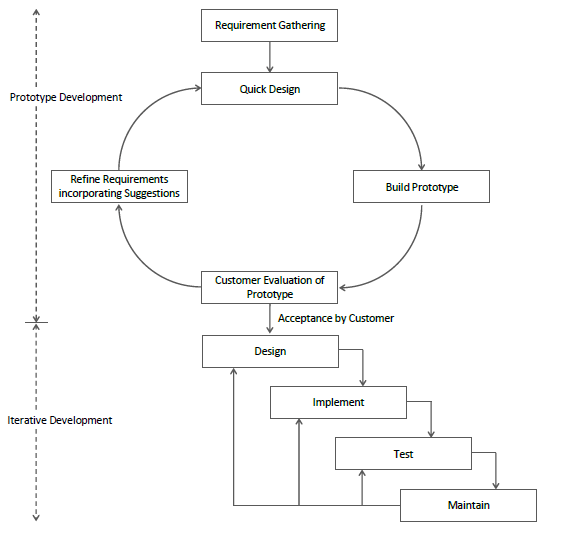


Figure 11: Prototype Model of Software Development

The FollowUp system is based on the prototype Model of Software Development. The above figure shows the overall flow of how the system was first approached and evolved through time to meet certain requirement. The lifecycle defines the phases the system has to go through to be more reliable and accurate as per the requirement. The prototyping model ensures the delivered product is as per the user's requirement. Each phase is detailed and carried out to deliver quality software product.

### 4.1.2 Working Mechanism

1. **Admin** can add and manage user. Admin can view follow up report generated by every user in the system and also delete them.
2. **User** can add and edit the client's follow up. User can also view the follow up of respective client. User can also maintain their to-do lists.

### 4.1.3 Implementation Tools

1. **Front End Tools**

**HTML**

In our project, it is used to interpret and compose text, images and other material into visual web pages. The whole user interface is designed using HTML.

**CSS**

In our project, it is designed to enable the separation of presentation and content, including layout, color and font. We used is to enhance the presentation of webpages written in HTML.

**JavaScript**

In our project, it is used for form validation and to hide and view form content.

**Bootstrap**

Our project contains different CSS and JS files which we used for styling. We used various predefined component like panel, form, button and navigation bar.

1. **Back End Tools**

**PHP**

In our project, it is used to retrieve and store the data in the database. It is also used to get and set the data in the HTML form. The Validation is done using PHP.

**MySQL**

The database of Follow Up is created using MySQL We used MYSQL for storing different data like client detail, organization detail, and follow up detail. We also used MYSQL for creating relationship between tables and also manipulating the data.

**Laravel Framework**

FollowUp is developed using Laravel framework following the Model-View-Controller(MVC) architectural pattern.

**Web Server**

FollowUp is displayed using Xampp sever to see how it will look while displaying using Web browser with the help of local host feature of the Apache server. It is implemented for running our Web application and used for deploying.

1. **Algorithm**
2. **Linear Search Algorithm**

FollowUp uses linear search algorithm to search a specific follow up record to further add new follow up of already existing client. The algorithm is implemented as:

LinearSearch (a,n,key)

Begin

for i= 0 to n-1 by 1 do

if a[i] = = key then

return 1;

end if

end for

return -1;

End

Here,

a= Organization Collective array

n= Number of data in the array

key = Organization\_id

The search will continue linearly checking every single index of the array until it finds the match, if the id matches with the id, user is searching then it will return the detail of the organization with a new form to add new follow up. If the searched id is not available in the system, it will display error message.

1. **Priority Ranking Algorithm**

FollowUp uses Priority Ranking algorithm to filter out the jobs that have the highest number of follow up to be at top. The algorithm recommends the organization with the highest number of follow up. The algorithm is implemented as:

While user add follow up

Fetch counter value from job table

Increment counter value by 1

Update job table

End

FollowUp uses this concept to implement the recommendation algorithm in the system. Here, with each addition of a followup of an organization, the value of counter increases. In view section, this counter value determines the priority of a job with the highest number of follow up. The algorithm then prioritizes the organization with highest number of counter in view section as:

While view all job

Fetch the data from job database

Order the data by counter in descending order

Return the view page

End

This is the implementation of Recommendation Algorithm named as Priority Ranking Algorithm in FollowUp system.

## 4.2 List of Controllers

### 4.2.1 Job Controller

This controller is used to control the view of the follow up. The store function is used to store the information about the organization, person and their follow up status. It also creates notification as per the status and scheduled date of the follow up. The show function is used to display the content added by the user. The edit function is used to edit the organization and people. The destroy function is used to delete the follow up record of specified organization.

### 4.2.2 FollowUp Controller

This controller is used to manage the follow up records. The search function is used to search for a specific follow up in the system. The store function is used to add follow up of already existing organization in the system. The edit function is used to edit the follow up report. The update function is used to update the content of the follow up records.

### 4.2.3 Notification Controller

This controller is used to generate notification of the follow up. This controller passes notification to both admin and the user.

### 4.2.4 Todo Controller

This controller manages all the functions regarding to-do. The store function is used to add the new to-do activity. The show function is used to view the to-do activity. The edit function is used to edit the to-do activity. The update function is used to update the to-do activity. All these functions are done by the user only.

## 4.3 Testing

### 4.3.1 Unit Testing

We have tested each module in an attempt to discover any error in its code. We tested the registration model to check if the user can register successfully or not and also we tested the login module for duplicate or empty username and password. Some of the unit test we performed is presented below:

1. **Testing the login section**

Series of valid and invalid username and password is inserted to check the authentication for the login section.

**Table 1: TEST CASE FOR LOGIN SECTION**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N** | **Test Case** | **Input Test Data** | **Expected Results** | **Actual Result** | **Status** |
| 1 | Test if user is able to login successfully | Email:  sanukoirala@gmail.com  Password:Destiny33 | User must be redirected into the user dashboard | User is redirected to the user dashboard | True |
| **S.N** | **Test Case** | **Input Test Data** | **Expected Results** | **Actual Result** | **Status** |
| 2. | Test if user is unable to login successfully | Email: sanukoirala@gmail.com  Password:ram33 | User must not be redirected to the user dashboard | User isn’t redirected to the user dashboard | False |
| 3. | Test with valid email, but empty password | Email:  sanukoirala@gmail.com  Password: | Please fill out the form message must be displayed | Please fill out the form message is displayed | True |
| 4. | Test with empty email, valid password | Email:  Password: abc123 | Please fill out the form message must be displayed | Please fill out the form message is displayed | True |

1. **Testing the managing section**

After a successful login, a user can manage the available section on it.

Table 2: TEST CASE FOR USER LOGIN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N.** | **Test Cases** | **Input Data** | **Expected Outcome** | **Actual Result** | **Status** |
| 1. | Test if user is able to add follow up successfully | Organization ,representative and follow up status | Follow up record must be added | Follow up record is added | True |
| 2. | Test if user is able to edit follow up successfully | Organization ,representative and follow up status | Follow up record must be edited | Follow up record is edited | True |
| 3. | Test if user is able to view follow up successfully | Submission of follow up | Follow up record must be viewed | Follow up record is viewed | True |

### 4.3.2 Integration Testing

We tested for the verification of email and password and redirected to the desired page. The test result is shown below:

Table 3: TEST CASE FOR INTEGRATION TESTING

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **S.N.** | **Test Cases** | **Role** | **Input Data** | **Expected Outcome** | **Actual Result** | **Status** |
| 1. | Enter the user dashboard after the login | User | Email:  sanukoirala@gmail.com  Password:abc123 | User must successfully login into the user dashboard | User is successfully login into the user dashboard | True |
| 2. | Enter the user dashboard after the login | Admin | Email:  test@gmail.com  Password: abc123 | Admin must successfully login into the user dashboard | Admin is successfully login into the user dashboard | True |

### 4.3.3 System testing:

We tested whether the admin and user can perform their roles as per designed or not. We also tested for the security by redirecting the admin url from the user dashboard and tested its performance.

Table 4: TEST CASE FOR SYSTEM TESTING

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N.** | **Test Cases** | **Input Data** | **Expected Outcome** | **Actual Result** | **Status** |
| 1. | Activity performed by user must be viewed by admin | Submission of activity | Activity performed by the user must be notified to the admin | Activity performed by the user is notified | True |
| 2. | Redirecting admin url from the user dashboard | url of admin dashboard | User must be redirected to the login page | User is redirected to the login page | True |

# CHAPTER 5: CONCLUSION AND RECOMMENDATION

## 5.1: Conclusion

FollowUp has the aim to provide services to organizations by managing records and events of clients. By this application, Admin can create users and check the activity that is performed by each user. User can add and view follow up records of the client. Each user has their separate follow up task where one is not able to view other's maintaining record isolation. User can maintain their to-do list.

## 5.2: Recommendation

By the effort of our college and our hard study, we were able to do some sort of work on web development through which we built this application. However, in this system named as FollowUp, there are few things to be recommended for it to be a perfect web application. We recommend doing the following work:

* Messaging between Admin and User can be added.
* Comment Section can be added for Admin.
* Discussion forum can be added.
* Test on the selected criteria can be added.
* Complete isolation of records can be added.

# REFERENCES

|  |  |
| --- | --- |
| [1] | R. Nation, "Improving your record management system," 4 April 2020. [Online]. Available: https://www.recordnations.com/articles/improving-record-management-system/. [Accessed 21 July 2020]. |
| [2] | Linkedin, "Linkedin," [Online]. Available: https://www.linkedin.com/company/agentology/about/. [Accessed 15 September 2020]. |
| [3] | K. Manikas, "Management, Record Management and Electronic Records," 2015. |
| [4] | M. S. Basnet, "College Management System," Mr. Sagar Basnet, Biratnagar, 2019. |

# BIBLIOGRAPHY

|  |  |
| --- | --- |
|  | javaTpoint, "Requirement Analysis," [Online]. Available: https://www.javatpoint.com/software-engineering-requirement-analysis. [Accessed 03 March 2020]. |
|  | S. Koirala, "Requirement Analysis," 03 March 2020. [Online]. |
|  | Laravel, "Middleware," [Online]. Available: https://laravel.com/docs/8.x/middleware#introduction. [Accessed 15 June 2020]. |
|  | L. l. LLC, "blog," [Online]. Available: https://www.leadliaison.com/blog/. [Accessed 06 March 2020]. |
|  | Tutorilaspoint, "SQL," [Online]. Available: https://www.tutorialspoint.com/sql/index.htm. [Accessed 05 April 2020]. |
|  | W3School, "PHP," [Online]. Available: https://www.w3schools.com/php/default.asp. [Accessed 21 April 2020]. |

# APPENDIX I: CODE SNAPSHOT

*// to insert followup*

*public* function store(Request **$request**)

    {

        $c**=**1;

    $*this***->**validate($request,[

      'name'**=>**'required',

      'address'**=>**'required',

      'landline'**=>**'required|min:9|numeric',

      'email'**=>**'required',

      'p\_mobile'**=>**'required|min:10|numeric',

      'p\_name'**=>**'required',

      'p\_email'**=>**'required',

      'p\_post'**=>**'required',

      'images' **=>** 'mimes:jpeg,jpg,png,gif|required|max:10000',

      'f\_pitch'**=>**'required'

     ]);

*// to show followup report*

*public* function show()

    {

        $datas**=**job**::**orderBy('created\_at','desc')**->**get();

**if**(Auth**::**user()**->***type***==**1){

**return** view('backend.job.show',compact(['datas']));

        }

*//Implmentation of reommendation algorithm*

        $datas**=**job**::**orderBy('counter','desc')**->**get();

**return** view('frontend.job.show',compact(['datas']));

    }

*// to edit followup*

*public* function edit(**$id**)

    {

        $person**=**person**::**where('job\_id',$id)**->**groupBy('p\_name')**->**select('id',\DB**::**raw("concat(p\_name,'-',p\_post)as name"))**->**distinct()**->**lists('name','id');

        $person**->**prepend('New',0);

        $merges**=**Follow**::**join('people','people.fu\_conter','=','follows.fu\_conter')

**->**where('people.job\_id',$id)

**->**where('follows.job\_id',$id)

**->**orderBy('people.fu\_conter','desc')

**->**get();

        $job**=**job**::**findOrfail($id);

**if**(Auth**::**user()**->***type***==**1){

**return** view('backend.job.report',compact(['job','merges','person']));

        }

**return** view('frontend.job.report',compact(['job','merges','person']));

    }

*// to delete followup*

*public* function destroy(**$id**)

    {

        $job**=**job**::**findOrfail($id);

        $job**->**delete();

        \Illuminate\Support\Facades\Request**::**session()**->**flash('alert-success', 'Successful Deleted');

**return** redirect('admin/job/view');

    }

*// to search specific followup*

*public* function search(Request **$request**)

    {

**if**($request**->***search***==**''){

            $request**->**session()**->**flash('alert-success', 'No Search to display');

**return** redirect('job/view');

        }

        $url**=**'job/'**.**$request**->***search***.**'/addFollowup';

        $jobs**=**job**::**all();

        $org**=**job**::**findOrfail($request**->***search*);

        $show**=**1;

**return** view('frontend.job.follow',compact(['jobs','show','url','org']));

    }

*//to add followup*

*public* function store(Request **$request** ,**$id**)

    {

        $*this***->**validate($request,[

      'p\_mobile'**=>**'required|min:10|numeric',

      'p\_name'**=>**'required',

      'p\_email'**=>**'required',

      'p\_post'**=>**'required',

      'images' **=>** 'mimes:jpeg,jpg,png|required|max:10000',

      'f\_pitch'**=>**'required',

      'f\_remarks'**=>**'required',

      'f\_response'**=>**'required'

     ]);

*//to update organizational details*

*public* function update(Request **$request**, **$id**)

    {

        $*this***->**validate($request,[

      'name'**=>**'required',

      'address'**=>**'required',

      'landline'**=>**'required|min:9|numeric',

      'email'**=>**'required'

     ]);

*//to update representative and followup*

*public* function updateFollow(Request **$request**, **$id**)

    {

        $*this***->**validate($request,[

      'p\_mobile'**=>**'required|min:10|numeric',

      'p\_name'**=>**'required',

      'p\_email'**=>**'required',

      'p\_post'**=>**'required',

      'f\_pitch'**=>**'required',

      'f\_remarks'**=>**'required',

      'f\_response'**=>**'required'

     ]);

*//Implementing linear search*

*public* function search(Request **$request**)

    {

**if**($request**->***search***==**''){

            $request**->**session()**->**flash('alert-success', 'No Search to display');

**return** redirect('job/view');

        }

        $url**=**'job/'**.**$request**->***search***.**'/addFollowup';

        $jobs**=**job**::**all();

        $org**=**job**::**findOrfail($request**->***search*);

        $show**=**1;

**return** view('frontend.job.follow',compact(['jobs','show','url','org']));

    }

*//to generate notification*

*public* function index()

    {

        $thisdate**=** date("Y-m-d 00:00:00");

        $now**=new** *\DateTime*();

**if**(Auth**::**user()**->***type***==**1){

            $extend**=**'backend.layout';

            $admin**=**\App\Notification**::**where('admin\_status',0)

**->**where(function(**$query**)**use**($thisdate){

**return** $query

**->**where('date','=',$thisdate)

**->**orWhere('followup\_date','=',$thisdate);

            })

**->**where('remind\_me', '>=', ($now**->**format('Y-m-d H:i')))

**->**where('type','=',1)

**->**orderBy('remind\_me', 'ASC')

**->**get();

**return** view('notification',compact('admin','extend','thisdate'));

        }

        $user**=**\App\Notification**::**where([

            'user\_id'**=>**\Illuminate\Support\Facades\Auth**::**user()**->***id*,

            'user\_status'**=>**0,

            'type'**=>**1

        ])

**->**where(function(**$query**)**use**($thisdate){

**return** $query

**->**where('date','=',$thisdate)

**->**orWhere('followup\_date','=',$thisdate);

            })

**->**where('remind\_me', '>=', ($now**->**format('Y-m-d H:i')))

**->**orderBy('remind\_me', 'ASC')

**->**get();

        $extend**=**'backend.layoutFront';

**return** view('notification',compact('user','extend','thisdate'));

    }

*public* function index()

    {

        $jobs**=**job**::**where('user\_id',Auth**::**user()**->***id*)**->**get();

        $show**=**0;

**return** view('frontend.job.follow',compact(['jobs','show']));

    }

*/\*\**

*\* Show the form for creating a new resource.*

*\**

*\* @return \Illuminate\Http\Response*

*\*/*

*//to show followup report*

*public* function search(Request **$request**)

    {

**if**($request**->***search***==**''){

            $request**->**session()**->**flash('alert-success', 'No Search to display');

**return** redirect('job/view');

        }

        $url**=**'job/'**.**$request**->***search***.**'/addFollowup';

        $jobs**=**job**::**all();

        $org**=**job**::**findOrfail($request**->***search*);

        $show**=**1;

**return** view('frontend.job.follow',compact(['jobs','show','url','org']));

*//*

    }

*/\*\**

*\* Store a newly created resource in storage.*

*\**

*\* @param  \Illuminate\Http\Request  $request*

*\* @return \Illuminate\Http\Response*

*\*/*

*//to update followup*

*public* function store(Request **$request** ,**$id**)

    {

        $*this***->**validate($request,[

      'p\_mobile'**=>**'required|min:10|numeric',

      'p\_name'**=>**'required',

      'p\_email'**=>**'required',

      'p\_post'**=>**'required',

      'images' **=>** 'mimes:jpeg,jpg,png|required|max:10000',

      'f\_pitch'**=>**'required',

      'f\_remarks'**=>**'required',

      'f\_response'**=>**'required'

     ]);

     $c**=**1;

     $j**=**person**::**where('job\_id',$id)**->**orderBy('fu\_conter','desc')**->**first(['fu\_conter']);

     $k**=**job**::**where('id',$id)**->**first();

*//increment of counter variable*

**if**($j**->***fu\_conter***!=**0){

         $c**=++**$j**->***fu\_conter*;

         $k**->***counter***=**$k**->***counter***+**1;

         $k**->**update();

     }

        $person**=new** person();

        $follow**=new** Follow();

        $person**->***p\_name***=**$request**->***p\_name*;

        $person**->***p\_post***=**$request**->***p\_post*;

        $person**->***p\_email***=**$request**->***p\_email*;

        $person**->***p\_mobile***=**$request**->***p\_mobile*;

**if** ($request**->**hasfile('images'))

            {

                 $file**=**$request**->**file('images');

                 $extesnion**=**$file**->**getClientOriginalExtension();

                 $filename**=**time()**.**'.'**.**$extesnion;

                 $file**->**move('uploads/images/',$filename);

                 $person**->***images***=**$filename;

            }

**else**

            {

**return** $request;

                $person **->***images***=**'';

            }

        $person**->***fu\_conter***=**$c;

        $person**->***job\_id***=**$id;

        $person**->**save();

        $follow**->***f\_pitch***=**$request**->***f\_pitch*;

        $follow**->***p\_remarks***=**$request**->***p\_remarks*;

        $follow**->***f\_response***=**$request**->***f\_response*;

        $follow**->***f\_remarks***=**$request**->***f\_remarks*;

        $follow**->***f\_status***=**$request**->***f\_status*;

**if**($request**->***f\_status***==**1){

            $notify**=**Notification**::**where('job\_id',$id)**->**first();

**if**($notify){

                $notify**->**delete();

            }

            $job**=**job**::**findOrfail($id);

            $follodate**=** date('Y-m-d H:i:s', strtotime("$request**->***followup\_date*"));

            $follow**->***followup\_date***=**$follodate;

            $time**=**date('Y-m-d h:i', strtotime("$request**->***followup\_date*"));

            $data**=**[

                'message'**=>**'You have a FollowUp at '**.**$time**.**' for the job:<b> '**.**$job**->***name***.**'-'**.**$job**->***remarks***.**'</b>',

                'messageAdmin'**=>**Auth**::**user()**->***name***.**' have a FollowUp at '**.**$time**.**' for the job:<b> '**.**$job**->***name***.**'-'**.**$job**->***remarks***.**'</b>',

                'fu\_conter'**=>**$c,

                'followup\_date'**=>** date('Y-m-d 00:00:00', strtotime("$request**->***followup\_date*")),

                'remind\_me'**=>**$follodate,

                'date'**=>**date('Y-m-d', strtotime('-1 day', strtotime($request**->***followup\_date*))),

                'admin\_status'**=>**0,

                'user\_status'**=>**0,

                'type'**=>**1,

                'user\_id'**=>**Auth**::**user()**->***id*,

                'job\_id'**=>**$id

            ];

            Notification**::**create($data);

        }

**else**{

            $notify**=**Notification**::**where('job\_id',$id)**->**first();

**if**($notify){

            $notify**->**delete();

            }

        }

        $follow**->***fu\_conter***=**$c;

        $follow**->***job\_id***=**$id;

        $follow**->**save();

        $request**->**session()**->**flash('alert-success', 'Successful ADDED Followup');

**return** redirect('job/'**.**$id**.**'/edit');

    }

*/\*\**

*\* Display the specified resource.*

*\**

*\* @param  int  $id*

*\* @return \Illuminate\Http\Response*

*\*/*

*public* function show(**$id**)

    {

*//*

    }

*/\*\**

*\* Show the form for editing the specified resource.*

*\**

*\* @param  int  $id*

*\* @return \Illuminate\Http\Response*

*\*/*

*//to edit followup*

*public* function edit(**$id**)

    {

        $jobs**=**job**::**findOrfail($id);

        $show**=**1;

**return** view('frontend.job.job',compact('show','jobs'));

    }

*public* function editFollow(**$id**)

    {

    $categories**=**[

      "1"**=>**"Next Followup",

        "2"**=>**"Success",

        "3"**=>**"Fail",

    ];

        $show**=**20;

$person**=**person**::**findOrfail($id);

        $job**=**$person**->***job\_id*;

      $jobs**=**$person**->**join('follows','follows.job\_id','=','people.job\_id')

**->**where('follows.job\_id',$job)

**->**where('follows.fu\_conter',$person**->***fu\_conter*)

**->**where('people.id',$id)

**->**first();

**return** view('frontend.job.job',compact('jobs','categories','show'));

    }

*//to update added followup*

*public* function update(Request **$request**, **$id**)

    {

        $*this***->**validate($request,[

      'name'**=>**'required',

      'address'**=>**'required',

      'landline'**=>**'required|min:9|numeric',

      'email'**=>**'required'

     ]);

       $jobs**=**job**::**findOrfail($id);

        $data**=**$request**->**all();

        $jobs**->**update($data);

        $request**->**session()**->**flash('alert-success', 'Successful Updated Job');

**return** redirect('job/'**.**$id**.**'/edit');

    }

*public* function updateFollow(Request **$request**, **$id**)

    {

        $*this***->**validate($request,[

      'p\_mobile'**=>**'required|min:10|numeric',

      'p\_name'**=>**'required',

      'p\_email'**=>**'required',

      'p\_post'**=>**'required',

      'f\_pitch'**=>**'required',

      'f\_remarks'**=>**'required',

      'f\_response'**=>**'required'

     ]);

        $follow**=**Follow**::**findOrfail($id);

        $jobid**=**$follow**->***job\_id*;

        $fuconter**=**$follow**->***fu\_conter*;

        $person**=**person**::**where('job\_id',$jobid)**->**where('fu\_conter',$fuconter)**->**first();

        $person**->***p\_name***=**$request**->***p\_name*;

        $person**->***p\_post***=**$request**->***p\_post*;

        $person**->***p\_email***=**$request**->***p\_email*;

        $person**->***p\_mobile***=**$request**->***p\_mobile*;

**if** ($request**->**hasfile('images'))

            {

              File**::**delete(public\_path('uploads/images/'**.**$person**->***images*));

                 $file**=**$request**->**file('images');

                 $extesnion**=**$file**->**getClientOriginalExtension();

                 $filename**=**time()**.**'.'**.**$extesnion;

                 $file**->**move('uploads/images/',$filename);

                 $person**->***images***=**$filename;

            }

**else**

            {

**return** $request;

                $person **->***images***=**'';

            }

        $person**->**save();

        $follow**->***f\_pitch***=**$request**->***f\_pitch*;

        $follow**->***p\_remarks***=**$request**->***p\_remarks*;

        $follow**->***f\_response***=**$request**->***f\_response*;

        $follow**->***f\_remarks***=**$request**->***f\_remarks*;

        $follow**->***f\_status***=**$request**->***f\_status*;

**if**($request**->***f\_status***==**1){

            $notify**=**Notification**::**where('job\_id',$jobid)**->**first();

**if**($notify){

                $notify**->**delete();

            }

            $job**=**job**::**findOrfail($jobid);

            $follodate**=** date('Y-m-d H:i:s', strtotime("$request**->***followup\_date*"));

            $follow**->***followup\_date***=**$follodate;

            $time**=**date('Y-m-d h:i', strtotime("$request**->***followup\_date*"));

            $data**=**[

                'messageAdmin'**=>**Auth**::**user()**->***name***.**' have a FollowUp at '**.**$time**.**' for the job:<b> '**.**$job**->***name***.**'-'**.**$job**->***remarks***.**'</b>',

                'message'**=>**'You have a FollowUp at '**.**$time**.**' for the job:<b> '**.**$job**->***name***.**'-'**.**$job**->***remarks***.**'</b>',

                'fu\_conter'**=>**$fuconter,

                'followup\_date'**=>** date('Y-m-d 00:00:00', strtotime("$request**->***followup\_date*")),

                'remind\_me'**=>**$follodate,

                'date'**=>**date('Y-m-d', strtotime('-1 day', strtotime($request**->***followup\_date*))),

                'admin\_status'**=>**0,

                'user\_status'**=>**0,

                'type'**=>**1,

                'user\_id'**=>**Auth**::**user()**->***id*,

                'job\_id'**=>**$jobid

            ];

            Notification**::**create($data);

        }

        $follow**->***fu\_conter***=**$fuconter;

        $follow**->***job\_id***=**$jobid;

        $follow**->**save();

        $request**->**session()**->**flash('alert-success', 'Successful Updated ')

**return** redirect('job/'**.**$jobid**.**'/edit');

    }

# APPENDIX II: PROJECT SNAPSHOT

**Some relevant Snapshots:**

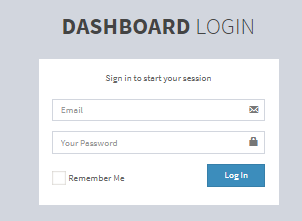
****

Figure 12: Snapshot of Login

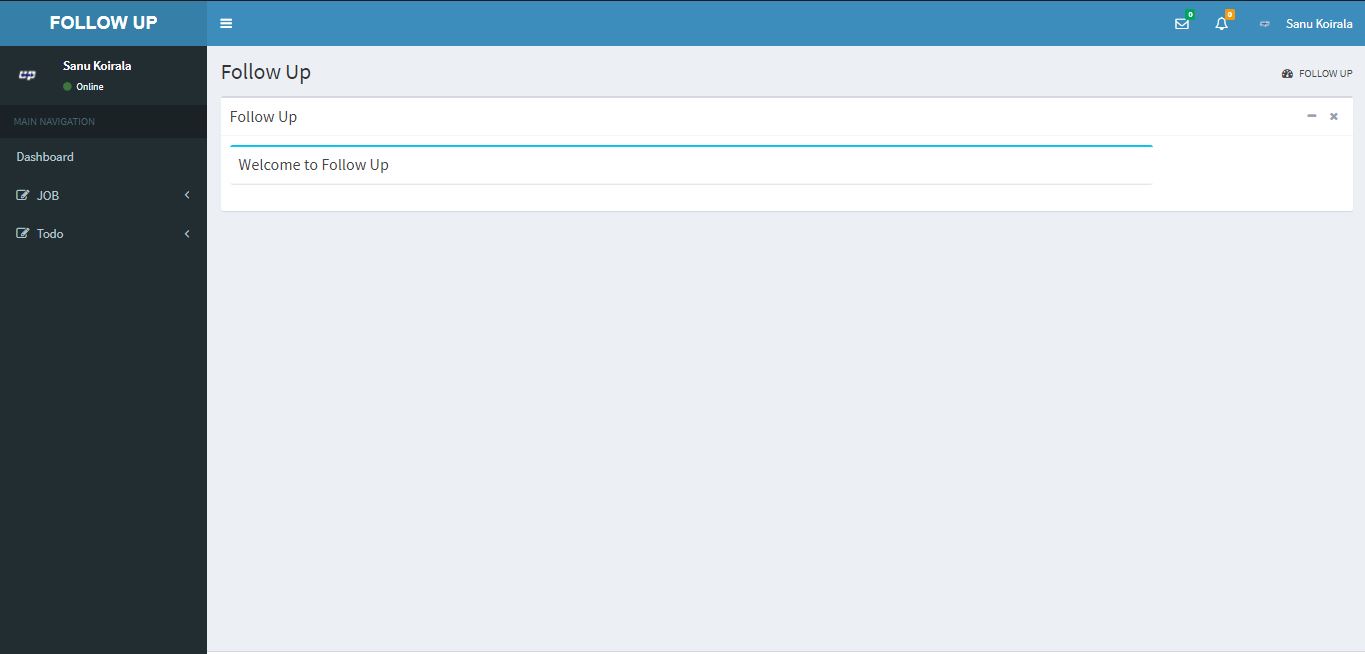


Figure 13: Snapshot of User Dashboard

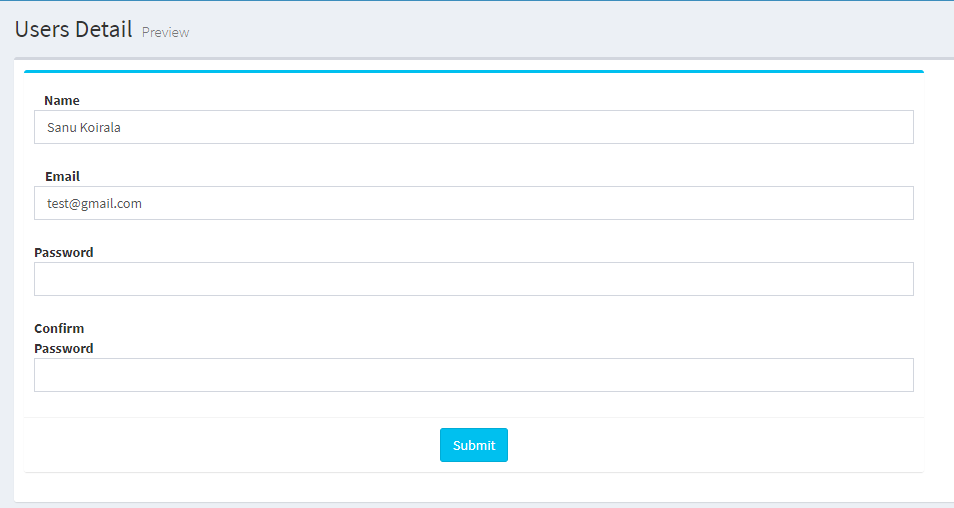


Figure 14: Snapshot of Editing User

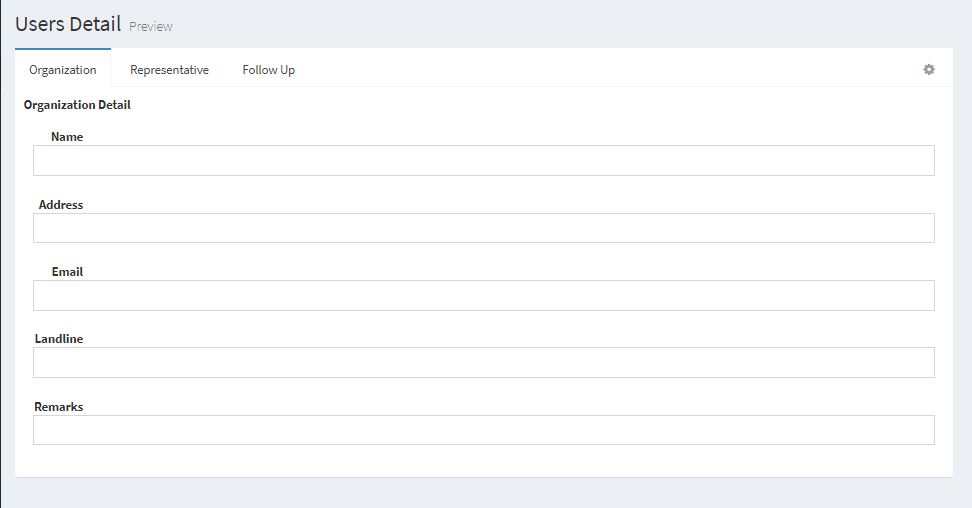


Figure 15: Snapshot of Adding Follow up

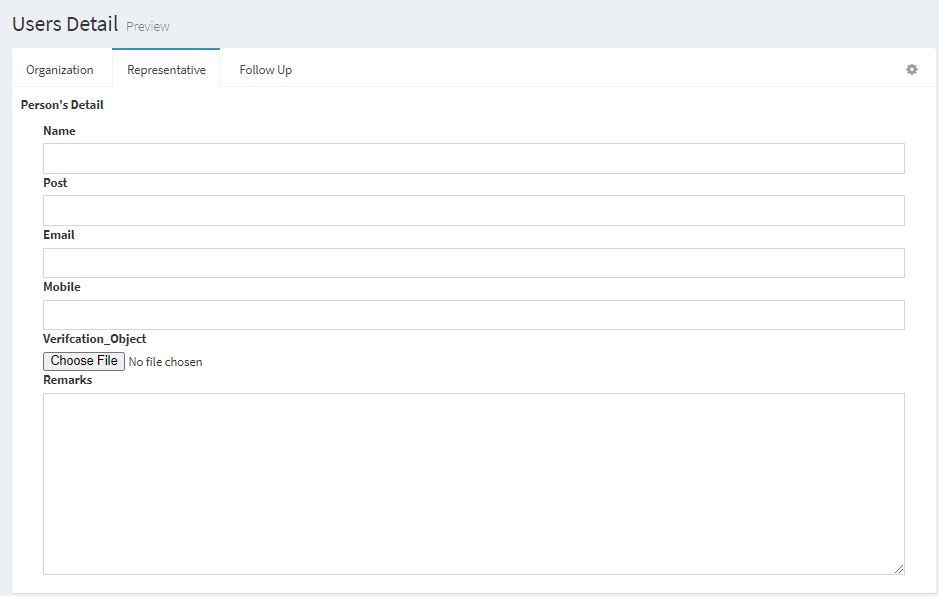


Figure 16: Snapshot of Adding Follow up

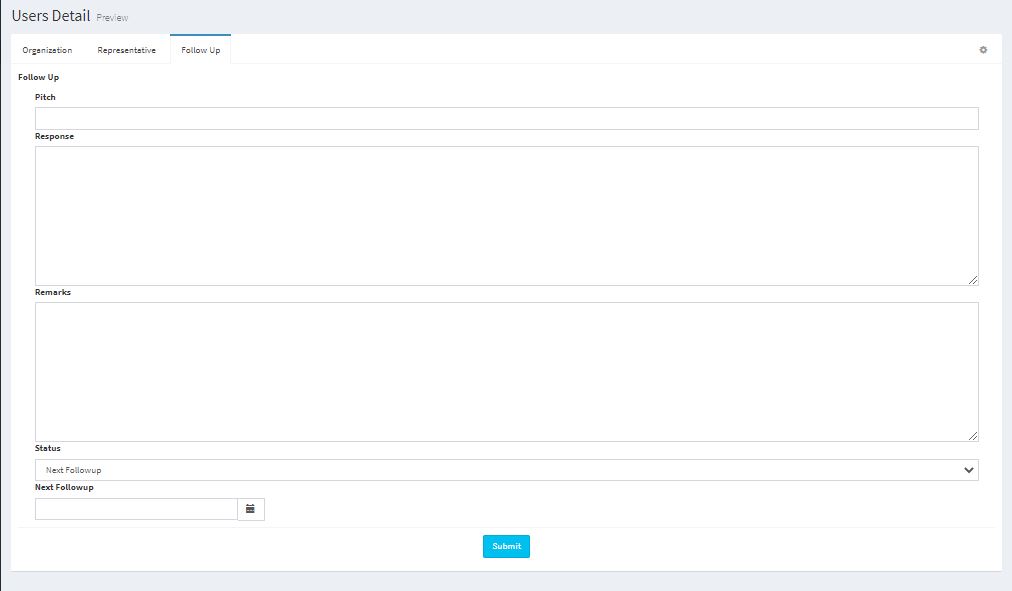


Figure 17: Snapshot of Adding Follow up

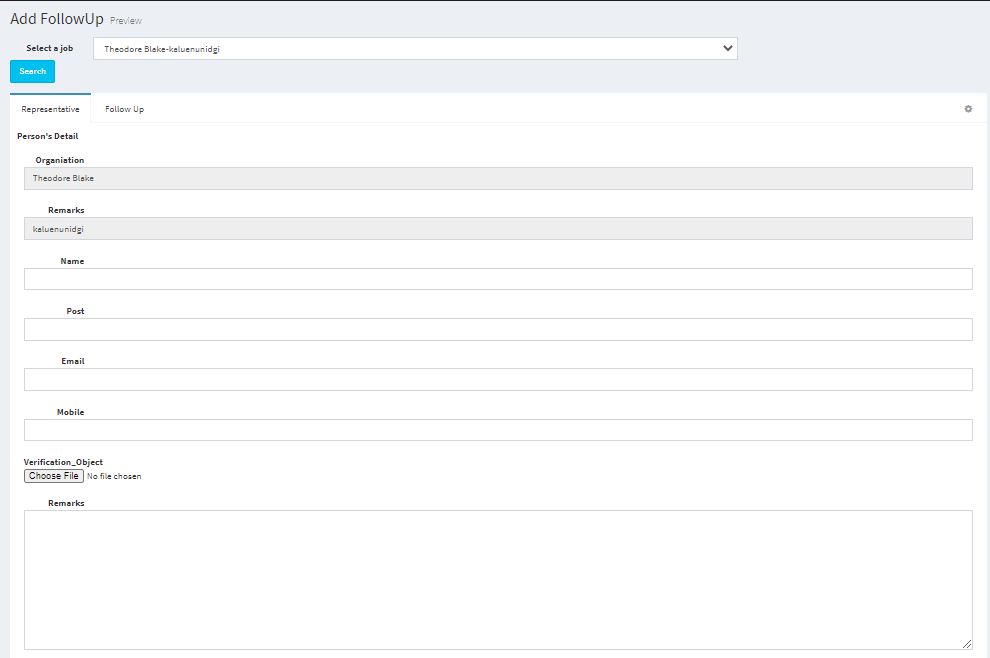


Figure 18: Snapshot of Adding new Representative

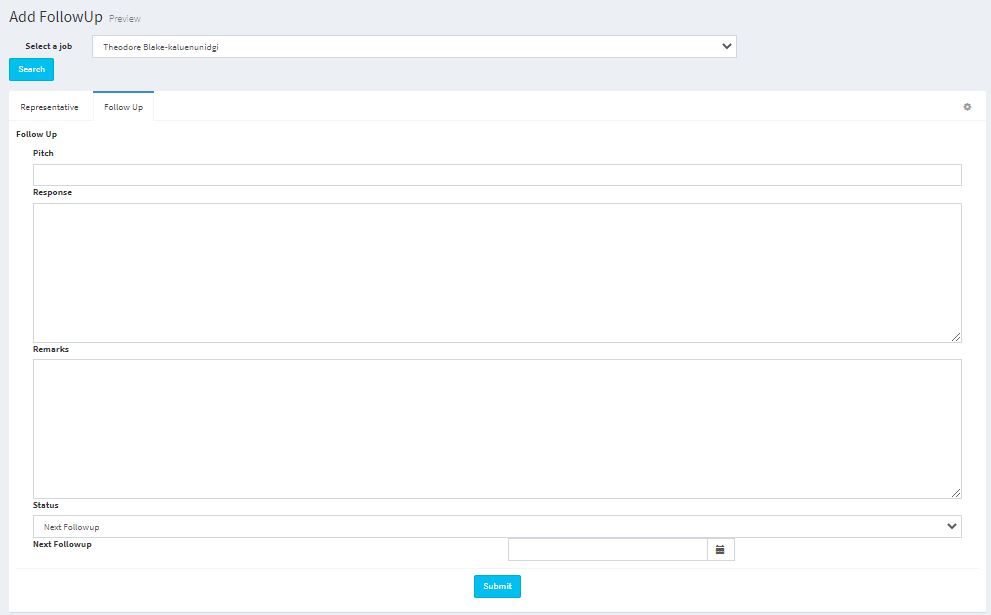
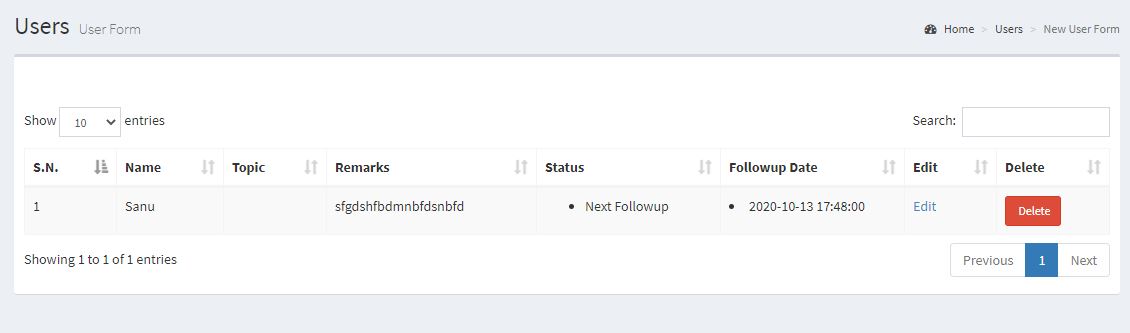


Figure 19: Snapshot Adding new Follow up

****

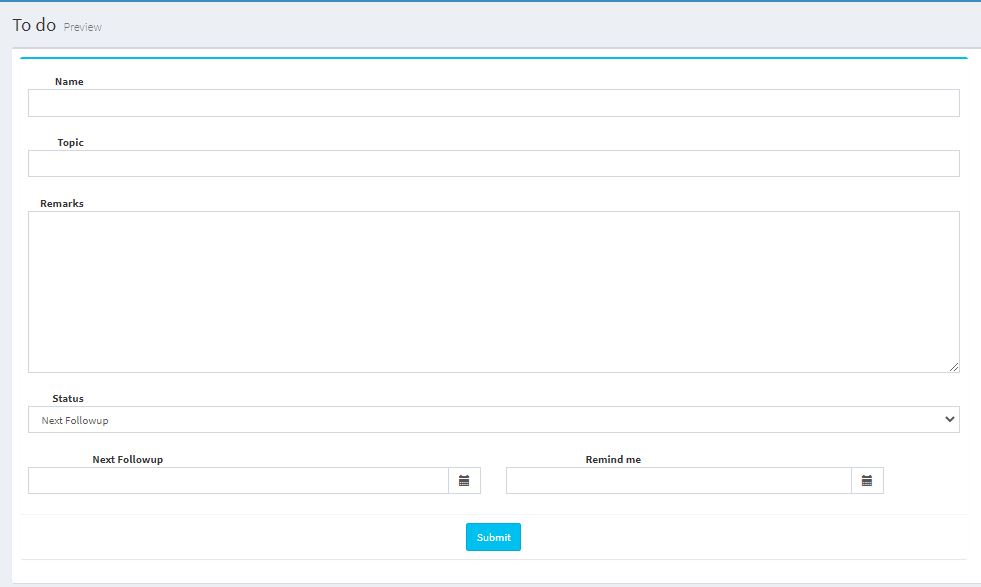
Figure 20: Snapshot of User's To-do list

Figure 21: Snapshot of adding to-do list