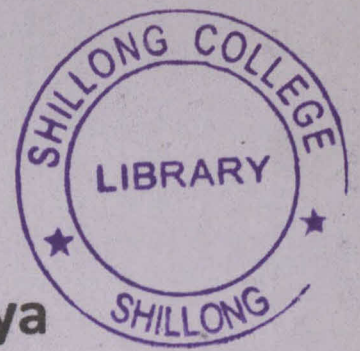


SHILLONG COLLEGE

Shillong-793001, Meghalaya



Project on

"Back To My Village"

**Submitted for the partial fulfilment for the award of the degree of Bachelor of
Computer Applications**

By

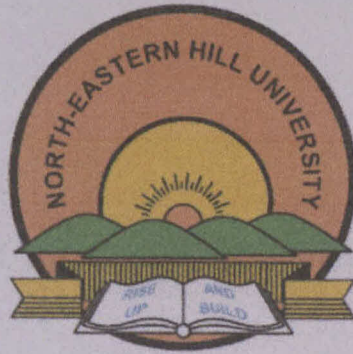
Nicholas Syngkli

Roll No: P1500074

Regn No: 14537 of 2013-14

Department of Computer Science and Applications

Shillong College, Shillong-793001



NORTH EASTERN HILL UNIVERSITY

Certified that this is a bonafide of the project

Entitled

"Back To My Village"

Submitted for the partial fulfilment for the award of the degree of Bachelor of
Computer Applications

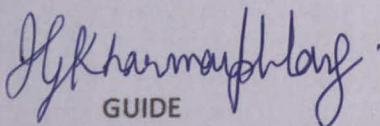
Submitted

By

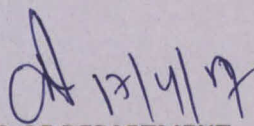
Nicholas Syngkli

Roll No: P1500074

Regn No: 14537 of 2013-14

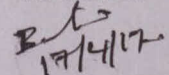

GUIDE

Smt. I.G. Kharmawphlang


HEAD OF DEPARTMENT

Mrs. A. Mitri

EXAMINER


17/4/17

Department of Computer Science and Applications

Shillong College, Shillong-793001



ACKNOWLEDGEMENT

At every outset I express my gratitude to almighty lord for showering his grace and blessings upon me to complete this project.

Although our name appears on the cover of this book, many people had contributed in some form or the other form to this project Development. I could not done this project without the assistance or support of each of the following I thank you all.

I wish to place on my record my deep sense of gratitude to my project guide, **Smt. Ibameda.G.Kharmawphlang Software Solutions**, for her constant motivation and valuable help through the project work. Express my gratitude to **Mrs. Aiom M. Mitri** , Head of the **Department Computer Sciences and Application**, Shillong College for her valuable suggestions and advices throughout the year course. I also extend my thanks to other Faculties for their Cooperation during my Course.

Finally I would like to thank my friends for their cooperation to complete this project.

*****NICHOLAS SYNGKLJ*****

ABSTRACT

Back to My Village is a charity group of professionals those want to voluntarily contribute in their village/town's development. Issues like Primary education, people's health, government policies awareness and availability of basic facilities/infrastructure are on main focus among others.

Through the website group want to help their members collaborate, to plan, assess and implement different activities and learn with others experience/feedbacks/suggestions. Group also wants to encourage others to join their initiatives and recognize their contributions.

Preface

Contents

1. INTRODUCTION

- 1.1 INTRODUCTION TO PROJECT
- 1.2 PURPOSE OF THE SYSTEM
- 1.3 PROBLEMS IN EXISTING SYSTEM

2. SYSTEM ANALYSIS

- 2.1 INTRODUCTION
- 2.2 STUDY OF THE SYSTEM
- 2.3 SYSTEM REQUIREMENT SPECIFICATIONS
- 2.4 PROPOSED SYSTEM
- 2.5 INPUT AND OUTPUT
- 2.6 PROCESS MODULES USED WITH JUSTIFICATION

3. FEASIBILITY REPORT

- 3.1 TECHNICAL FEASIBILITY
- 3.2 OPERATIONAL FEASIBILITY
- 3.3 ECONOMICAL FEASIBILITY

4. SOFTWARE REQUIREMENT SPECIFICATIONS

- 4.1 FUNCTIONAL REQUIREMENTS
- 4.2 PERFORMANCE REQUIREMENTS

5. SELECTED SOFTWARE

- 5.1 INTRODUCTION Python using Bootstrap
Server PyCharm
- 5.2 SQLite Server

6. SYSTEM DESIGN

6.1 INTRODUCTION

6.2 E-R DIAGRAM

6.3 DATA FLOW DIAGRAMS

6.4 DATA DICTIONARY

6.5 UML DIAGRAMS

7. SYSTEM TESTING AND IMPLEMENTATION

8. CONCLUSION

9. BIBLOGRAPHY



Introduction

1.1 INTRODUCTION TO PROJECT

The Back to My Village is a web based project.

This project is aimed to develop for charity group professionals those who wants to help their own city/village.

Through the website group want to help their members collaborate, to plan, assess and implement different activities and learn with others experience/feedbacks/ suggestions. Group also wants to encourage others to join their initiatives and recognize their contributions.

1.2 PURPOSE OF THE PROJECT

The project is fully integrated with Customer Relationship Management (CRM) solution and developed in a manner that is easily manageable, time saving and relieving one form semi-automated.

Back To My Village is a charity group of professionals those want to voluntarily contribute in their village/town's development. Issues like Primary education, people's health, government policies awareness and availability of basic.

Through the website group want to help their members collaborate, to plan, assess and implement different activities and learn with others experience/feedbacks/ suggestions. Group also wants to encourage others to join their initiatives and recognize their contributions.

1.3 EXISTING SYSTEM

The existing system is a semi-automated at where the information is stored in the form of excel sheets in disk drives. The information sharing to the Volunteers, Group members, etc. is through mailing feature only. The information storage and maintenance is more critical in this system. Tracking the member's activities and progress of the work is a tedious job here. This system cannot provide the information sharing by 24x7 days.

PROPOSED SYSTEM

The development of this new system objective is to provide the solution to the problems of existing system. By using this new system, we can fully automate the entire process of the current system. The new system would like to make as web-enabled so that the information can be shared between the members at any time using the respective credentials. To track the status of an individual process, the status update can be centralized using the new system. Being a web-enabled system, the process can be accessed across the world over net.

This system also providing the features like Chatting, Mailing between the members; Images Upload - Download via the web site; updating the process status in centralized location; generated reports can also be exporting to the applications like MS-Excel, PDF format, etc. In this new system, the members like Donors can give their valuable feedback to the Volunteers so that the Volunteers can check their progress of the tasks.

The entire process categorized as different modules like Admin module, Volunteer module, etc. at where we can classify the functionality as an individual process.

Using the new system entering into Admin module we can perform....

In this new system using the Volunteer module we can do....

In the Reports module we can generate reports like Weekly Status Report.

System Analysis

2.1 INTRODUCTION

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

2.2 STUDY OF THE SYSTEM

In the flexibility of the uses the interface has been developed a graphics concept in mind, associated through a browser interface. The GUI'S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The interfaces help the administrations with all the transactional states like Data insertion, Data deletion and Data updating along with the extensive data search capabilities.

The operational or generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities

NUMBER OF MODULES

The system after careful analysis has been identified to be presented with the following modules:

The Modules involved are

1. Web Administration
2. Group Member
3. Anonymous Users
4. Member Communications
5. Registration
6. Reports
7. Authentication

Web Administration

Administrator is a super user treated as owner of this site. He can have all the privileges. Administrator can register members directly, and delete the information of a registered member.

He verifies the information uploads into the system by the members or voluntaries. If any mismanagement done by the member immediately he can delete the information uploaded earlier by the member.

Basic and advance admin facilities add, update members, backup, recovery of data, generating various reports etc. The system provides an interface to the admin to change static web contents. He can track the member activities and progress.

Group Member

User is nothing but a registered member. Through the website group want to help their members collaborate, to plan, assess and implement different activities and learn with others experience/feedback/suggestions.

With the help of online questionnaires, members need to access the mature ness of primary education, health facilities etc. and based on the assessment need to categorize and chalk out a plan of actions by choosing from system suggested activities.

Group members arrange online information and face to face meetings with everyone to motivate them to contribute in this system. Members advised to develop a volunteers group in the village so that they can monitor, stabilize the change and reports to the members group.

Members need to provide facilitating like communications discussion forum, chat, mail etc. to the users of this system.

Anonymous Users

Anonymous users means a normal visitor of this system simply called as guest. By visiting this site he can get the information about this system, and also he can post his suggestions, queries, and doubts to the member group after that member need to reply sufficient response to those queries. If a an anonymous users want to became a register person means he needs to fill the registration form and became a member.

Member Communications

The member group need to collaborate, to plan, assess and implement different activities and learn with others experience, feedbacks, and suggestions. Here the facilitating communication means, the system provides discussion forum, chat, mail etc.

To find the assessment of current situation the system provides online questionnaires, members need to access the mature ness of primary education, health facilities etc.

Each plan of action would be shared with other members before execution so that they can share their experiences, feedback and suggestions.

Registration

The system has a process of registration. Every member need to submit his complete details in the form of registration. Whenever a user registration completed automatically user can get a user id and password. By using that user id and password member can log into the system.

Reports

Different kind of reports is generated by the system.

- State, Area wise list of adopted villages and their timely progress report
- Activities list and plan of action
- Magazine Subscription statistics
- Member list and their activities

2.3 System Requirement Specifications

Hardware Requirements:

RAM 512MB and Above

HDD 40gb Hard Disk Space and Above

Software Requirements:

Python Web Server Gateway Interface v1.0(WSGI) Django

Database (SQLite)

Project Category:

Language used

Front End: Bootstrap(html, css and javascripts)

Back End: Python

2.4 PROPOSED SYSTEM

To debug the existing system, remove procedures those cause data redundancy, make navigational sequence proper. To provide information about users on different level and also to reflect the current work status depending on organization. To build strong password mechanism.

NEED FOR COMPUTERIZATION

We all know the importance of computerization. The world is moving ahead at lightning speed and everyone is running short of time. One always wants to get the information and perform a task he/she/they desire(s) within a short period of time and too with amount of efficiency and accuracy. The application areas for the computerization have been selected on the basis of following factors:

- Minimizing the manual records kept at different locations.
- There will be more data integrity.
- Facilitating desired information display, very quickly, by retrieving information from users.
- Facilitating various statistical information which helps in decision-making?
- To reduce manual efforts in activities that involved repetitive work.

Updating and deletion of such a huge amount of data will become easier.

FUNCTIONAL FEATURES OF THE MODEL

As far as the project is developed the functionality is simple, the objective of the proposal is to strengthen the functioning of Audit Status Monitoring and make them effective and better. The entire scope has been classified into five streams known as Coordinator Level, management Level, Auditor Level, User Level and State Web Coordinator Level. The proposed software will cover the information needs with respect to each request of the user group viz. accepting the request, providing vulnerability document report and the current status of the audit.

2.5 INPUT AND OUTPUT

The major inputs and outputs and major functions of the system are follows:

Inputs:

- Admin enter his user id and password for login.
- User enters his user id and password for login.
- User Create new folder for personnel usage.
- Admin enter user id or date for track the user login information
- New user gives his completed personnel, address and phone details for registration.
- Admin gives different kind of user information for search the user data.
- User gives his user id, hint question, answer for getting the forgotten password.
- Administrator giving information to generate various kinds of reports.

Outputs:

- Admin can have his own home page.
- Users enter their own home page.
- The user defined folders can store in the centralized database.
- Admin will get the login information of a particular user.
- The new user's data will be stored in the centralized database.
- Admin get the search details of different criteria.
- User can get his forgot password.
- Different kind of reports is generated by administrator.

2.6 PROCESS MODEL USED WITH JUSTIFICATION

- **ACCESS CONTROL FOR DATA WHICH REQUIRE USER AUTHENTICAIION**
- The following commands specify access control identifiers and they are typically used to authorize and authenticate the user (command codes are shown in parentheses)
- **USER NAME (USER)**
- The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the control connections are made (some servers may require this).
- **PASSWORD (PASS)**
- This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress type out.

Feasibility Report

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operational Feasibility
- Economical Feasibility

3.1. TECHNICAL FEASIBILITY

The technical issue usually raised during the feasibility stage of the investigation includes the following:

- Does the necessary technology exist to do what is suggested?

- Do the proposed equipments have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of 'Secure Infrastructure Implementation System'. The current system developed is technically feasible. Thus it provides an easy access to the users. The database's purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hardware requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

3.2. OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and 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 would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

3.3. ECONOMICAL FEASIBILITY

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

SOFTWARE REQUIREMENT SPECIFICATION

The software, Site Explorer is designed for management of web sites from a remote location.

INTRODUCTION

Purpose: The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope: This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

DEVELOPERS RESPONSIBILITIES OVERVIEW:

The developer is responsible for:

- Developing the system, which meets the SRS and solving all the requirements of the system?
- Demonstrating the system and installing the system at client's location after the acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

- Conducting any user training that might be needed for using the system.
- Maintaining the system for a period of one year after installation.

4.1. FUNCTIONAL REQUIREMENTS

OUTPUT DESIGN

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization.
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

OUTPUT DEFINITION

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

- Will decimal points need to be inserted
- Should leading zeros be suppressed.

Output Media:

In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

- The suitability for the device to the particular application.
- The need for a hard copy.
- The response time required.
- The location of the users
- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are:

The outputs were needed to be generated as a hard copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

INPUT DESIGN

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

INPUT STAGES:

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission

- Data validation
- Data correction

INPUT TYPES:

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

INPUT MEDIA:

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive.

As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

ERROR AVOIDANCE

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

ERROR DETECTION

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

DATA VALIDATION

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

USER INTERFACE DESIGN

It is essential to consult the system users and discuss their needs while designing the user interface:

USER INTERFACE SYSTEMS CAN BE BROADLY CLASSIFIED AS:

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

USER_INITIATED INTERFACES

User initiated interfaces fall into two approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

COMPUTER-INITIATED INTERFACES

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

ERROR MESSAGE DESIGN:

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a

system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

4.2. PERFORMANCE REQUIREMENTS

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

SELECTED SOFTWARE

5.1 INTRODUCTION TO Pythonusing Bootstrap

Requirements

- A computer
- Internet

Language Introduction

Python is a dynamic, interpreted (bytecode-compiled) language. There are no type declarations of variables, parameters, functions, or methods in source code. This makes the code short and flexible, and you lose the compile-time type checking of the source code. Python tracks the types of all values at runtime and flags code that does not make sense as it runs.

One unusual Python feature is that the whitespace indentation of a piece of code affects its meaning. A logical block of statements such as the ones that make up a function should all have the same indentation, set in from the indentation of their parent function or "if" or whatever. If one of the lines in a group has a different indentation, it is flagged as a syntax error.

Python's use of whitespace feels a little strange at first, but it's logical and I found I got used to it very quickly. Avoid using TABs as they greatly complicate the indentation scheme (not to mention TABs may mean different things on different platforms). Set your editor to insert spaces instead of TABs for Python code. Since Python variables don't have any type spelled out in the source code, it's extra helpful to give meaningful names to your variables to remind yourself of what's going on. So use "name" if it's a single name, and "names" if it's a list of names, and "tuples" if it's a list of tuples. Many basic Python errors result from forgetting what type of value is in each variable, so use your variable names (all you have really) to help keep things straight.

5.2 Database (SQLite Server)

DB Browser for SQLite is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

It is for users and developers wanting to create databases, search, and edit data. It uses a familiar spreadsheet-like interface, and you don't need to learn complicated SQL commands.

Controls and wizards are available for users to:

- Create and compact database files
- Create, define, modify and delete tables
- Create, define and delete indexes
- Browse, edit, add and delete records
- Search records
- Import and export records as text
- Import and export tables from/to CSV files
- Import and export databases from/to SQL dump files
- Issue SQL queries and inspect the results
- Examine a log of all SQL commands issued by the application

What is Not?

This program is not a visual shell for the sqlite command line tool. It does not require familiarity with SQL commands. It is a tool to be used both by developers and by end users, and it must remain as simple to use as possible in order to achieve its goals.

SYSTEM DESIGN

6.1. INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirements have been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

6.2 E-R Diagrams

The relation upon the system is structure through a conceptual ER-Diagram, which not only specifies the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.

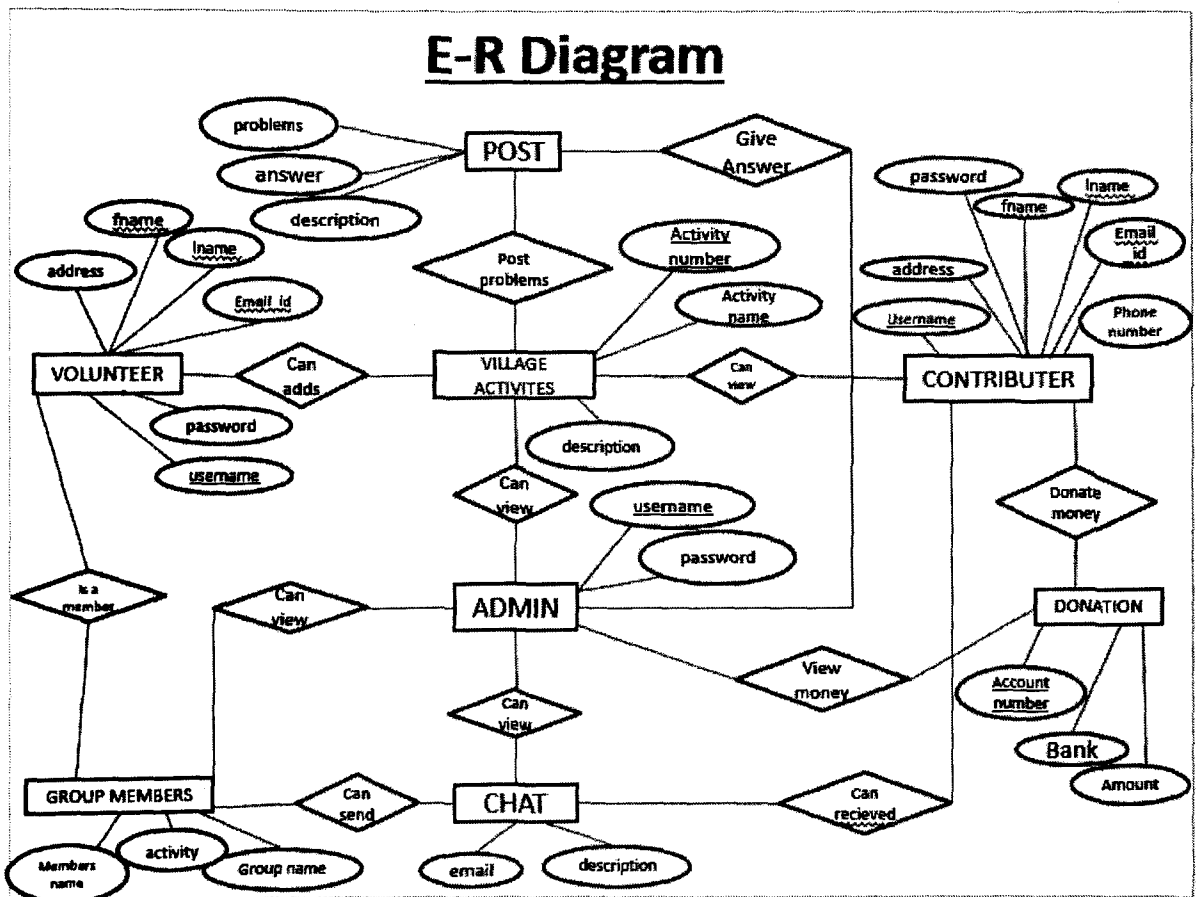
The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the data modeling

activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.

The set of primary components that are identified by the ERD are

- Data object
- Relationships
- Attributes
- Various types of indicators.

The primary purpose of the ERD is to represent data objects and their relationships.



6.3 DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The *lop-level diagram* is often called *context diagram*. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

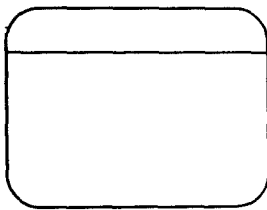
A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.



DFD SYMBOLS:

In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.



Source or Destination of data



Data flow



Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each word capitalized.

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SILENT FEATURES OF DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used

to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

- 1) No process can have only outputs.
- 2) No process can have only inputs. If an object has only inputs than it must be a sink.
- 3) A process has a verb phrase label.

DATA STORE

- 1) Data cannot move directly from one data store to another data store, a process must move data.
- 2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store

3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and /or destination of data.

- 1) Data cannot move directly from a source to sink it must be moved by a process
- 2) A source and /or sink has a noun phrase label

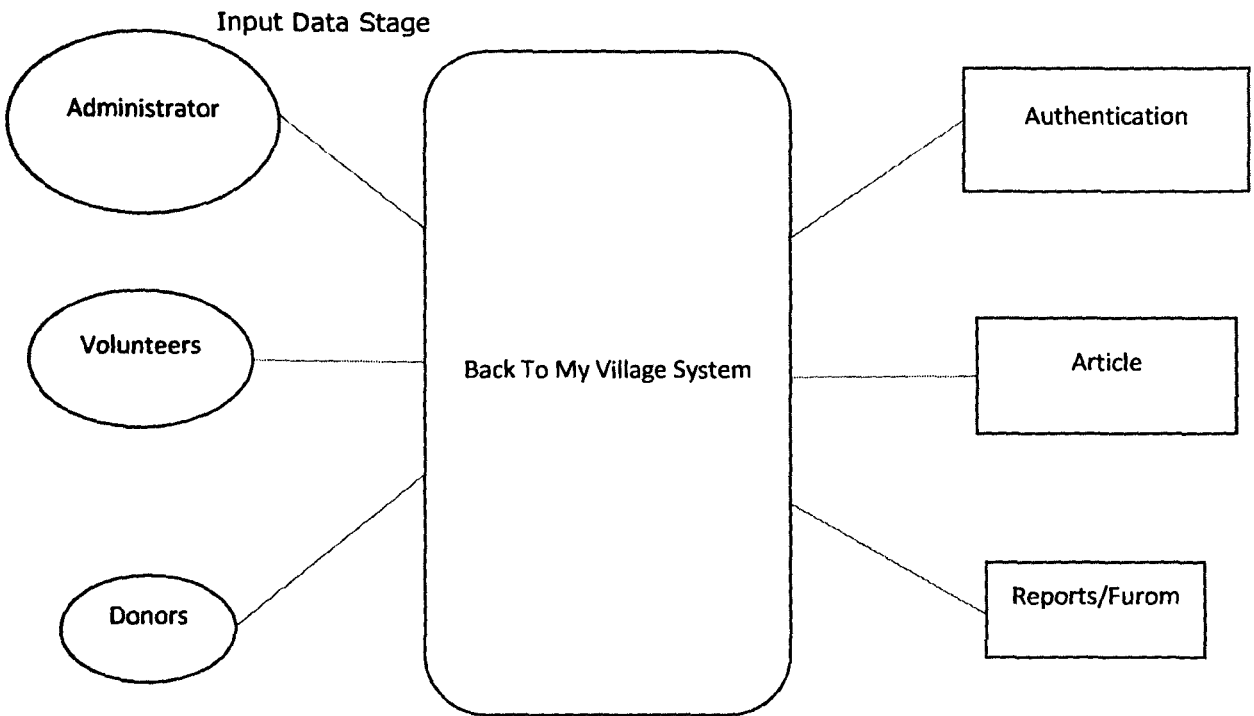
DATA FLOW

- 1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated however by two separate arrows since these happen at different times.
- 2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
- 3) A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
- 4) A Data flow to a data store means update (delete or change).
- 5) A data Flow from a data store means retrieve or use.

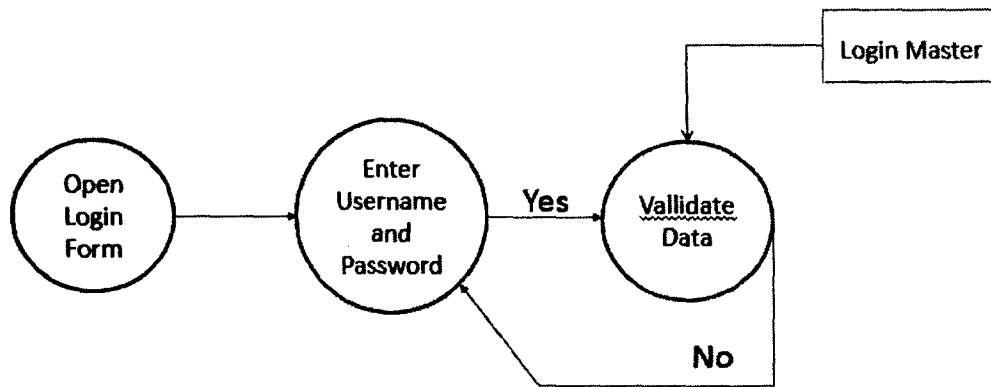
A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

DFD Diagrams:

Context Level (0th Level) DFD

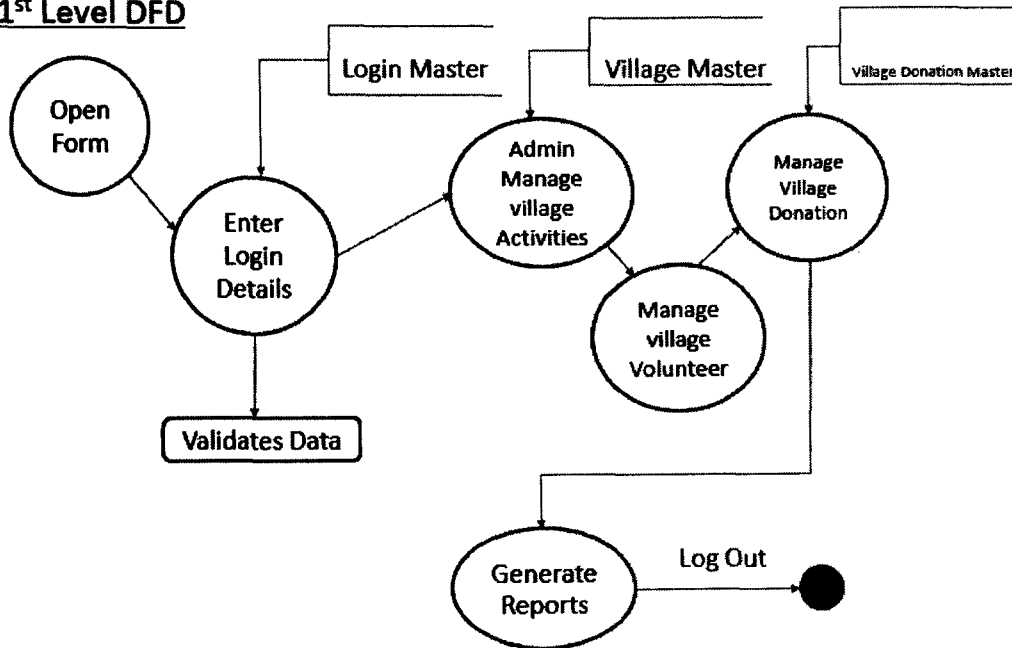


Login DFD Diagram

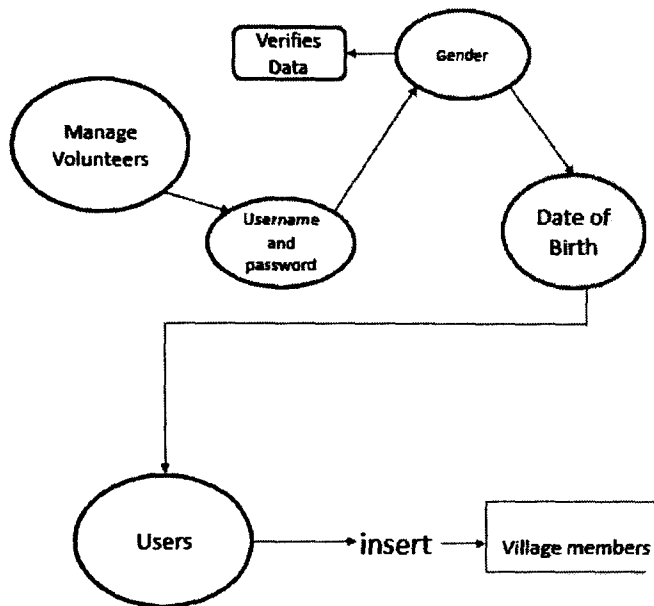


Admin Details Data Flow

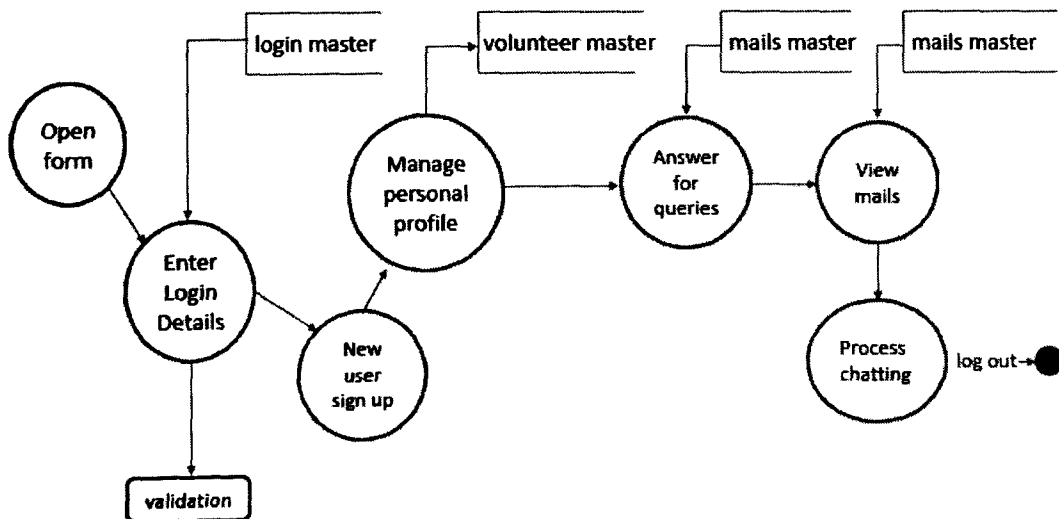
1st Level DFD



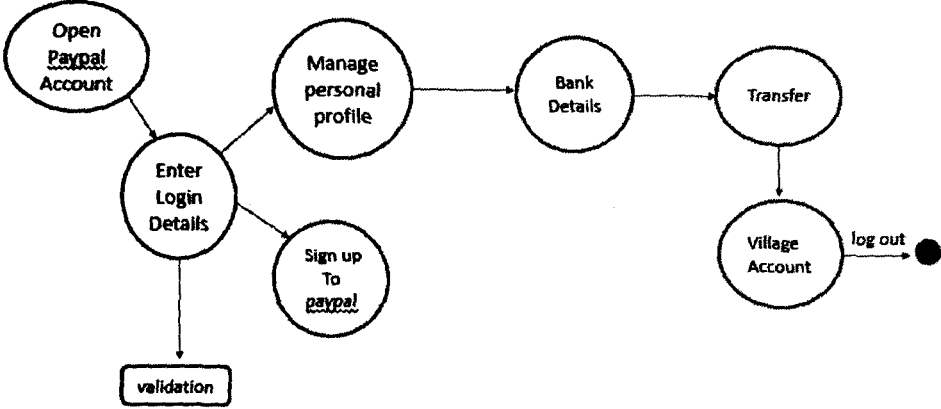
2nd Level DFD Manage Volunteers



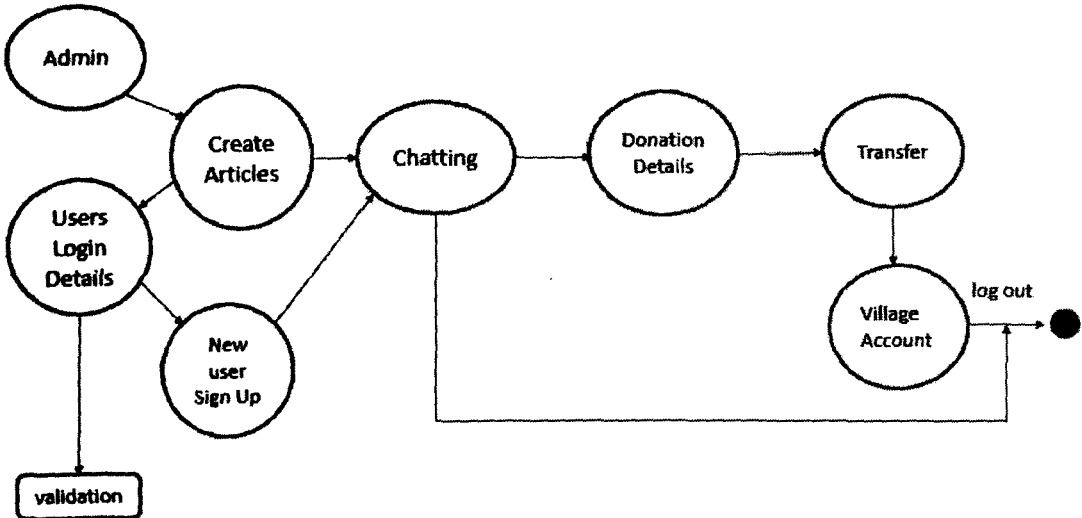
Donor Details Data Flow 1st Level DFD



Donor Details Data Flow 2nd Level DFD



DFD for Forum Level 0th



6.4 DATA DICTIONARY

After carefully understanding the requirements of the client the entire data storage requirements are divided into tables. The below tables are normalized to avoid any anomalies during the course of data entry.

Table Content

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
name	varchar(100)	NOT NULL
app_label	varchar(100)	NOT NULL
model	varchar(100)	NOT NULL

django migrations

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
app	varchar(255)	NOT NULL
name	varchar(255)	NOT NULL

auth_group_permissions

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id KEY	integer	NOT NULL PRIMARY AUTOINCREMENT
group_id	integer	NOT NULL REFERENCES

auth_group

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id KEY	integer	NOT NULL PRIMARY AUTOINCREMENT
name	varchar(80)	NOT NULL UNIQUE)

auth_user_groups

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id KEY	integer	NOT NULL PRIMARY AUTOINCREMENT
user_id	integer	NOT NULL
group_id	integer	NOT NULL



auth user user permissions

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>	
id	integer		NOT NULL PRIMARY KEY AUTOINCREMENT
user_id	integer		NOT NULL

auth user

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>	
permission_id	integer		NOT NULL

forum forum old

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>	
id	integer		NOT NULL PRIMARY KEY AUTOINCREMENT

blog article

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>	
id	integer		NOT NULL PRIMARY KEY AUTOINCREMENT
title	varchar(100)		NOT NULL
detail	text		NOT NULL
created_ondatetime			NOT NULL
article_img	varchar(100)		NOT NULL

auth user

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
password	varchar(128)	NOT NULL
last_login	datetime	NULL
is_superuser	bool	NOT NULL
first_name	varchar(30)	NOT NULL
last_name	varchar(30)	NOT NULL
email	varchar(254)	NOT NULL
is_staff	bool	NOT NULL
is_active	bool	NOT NULL
date_joined	datetime	NOT NULL
username	varchar(150)	NOT NULL UNIQUE

forum forum

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
topic_name	varchar(100)	NOT NULL
describe	text	NOT NULL
created_on	datetime	NOT NULL
user_id	integer	NOT NULL

auth user

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
---------------------------	------------------------	--------------------------

total_post	integer	NOT NULL
------------	---------	----------

forum forumpost

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
--------------------	-----------------	-------------------

id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
Post	text	NOT NULL
latest_post	datetime	NOT NULL
forum_id	integer	NOT NULL
user_id	integer	NOT NULL

postman message

<u>Column Name</u>	<u>Datatype</u>	<u>Allow Null</u>
--------------------	-----------------	-------------------

id	integer	NOT NULL PRIMARY KEY AUTOINCREMENT
subject	varchar(120)	NOT NULL
body	text	NOT NULL
email	varchar(254)	NOT NULL
sent_at	datetime	NOT NULL
read_at	datetime	NULL
replied_at	datetime	NULL
sender_archived	bool	NOT NULL
recipient_archived	bool	NOT NULL
sender_deleted_at	datetime	NULL
recipient_deleted_at	datetime	NULL
moderation_status	varchar(1)	NOT NULL
moderation_date	datetime	NULL
moderation_reason	varchar(120)	NOT NULL

register reguser

Column Name

DatatypeAllow Null

id	integer	NOT NULL PRIMARY KEY	AUTOINCREMENT
Birthday	date	NOT NULL	
gender	varchar(10)	NOT NULL	
user_id	integer	NOT NULL UNIQUE	

django session

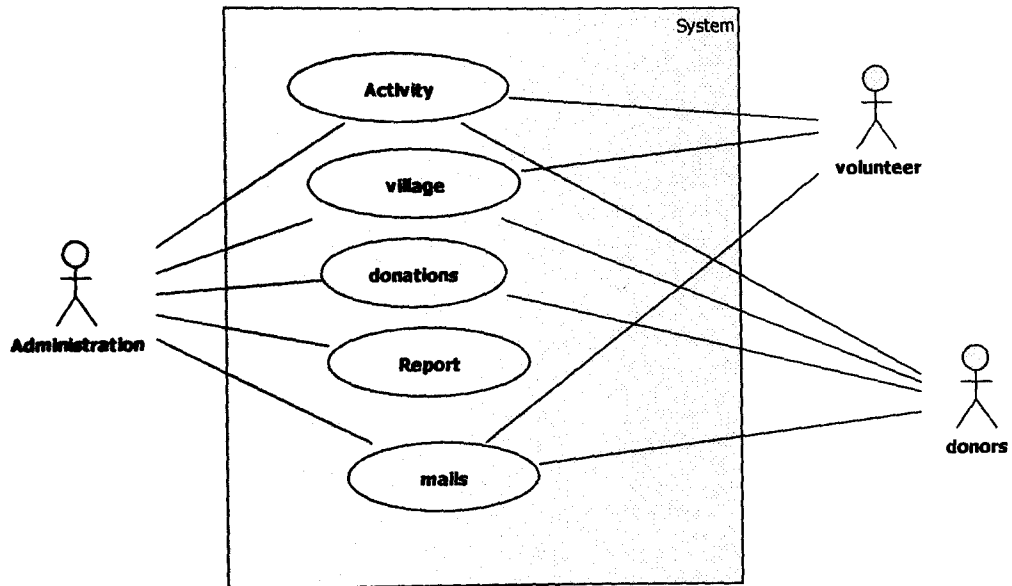
Column Name

DatatypeAllow Null

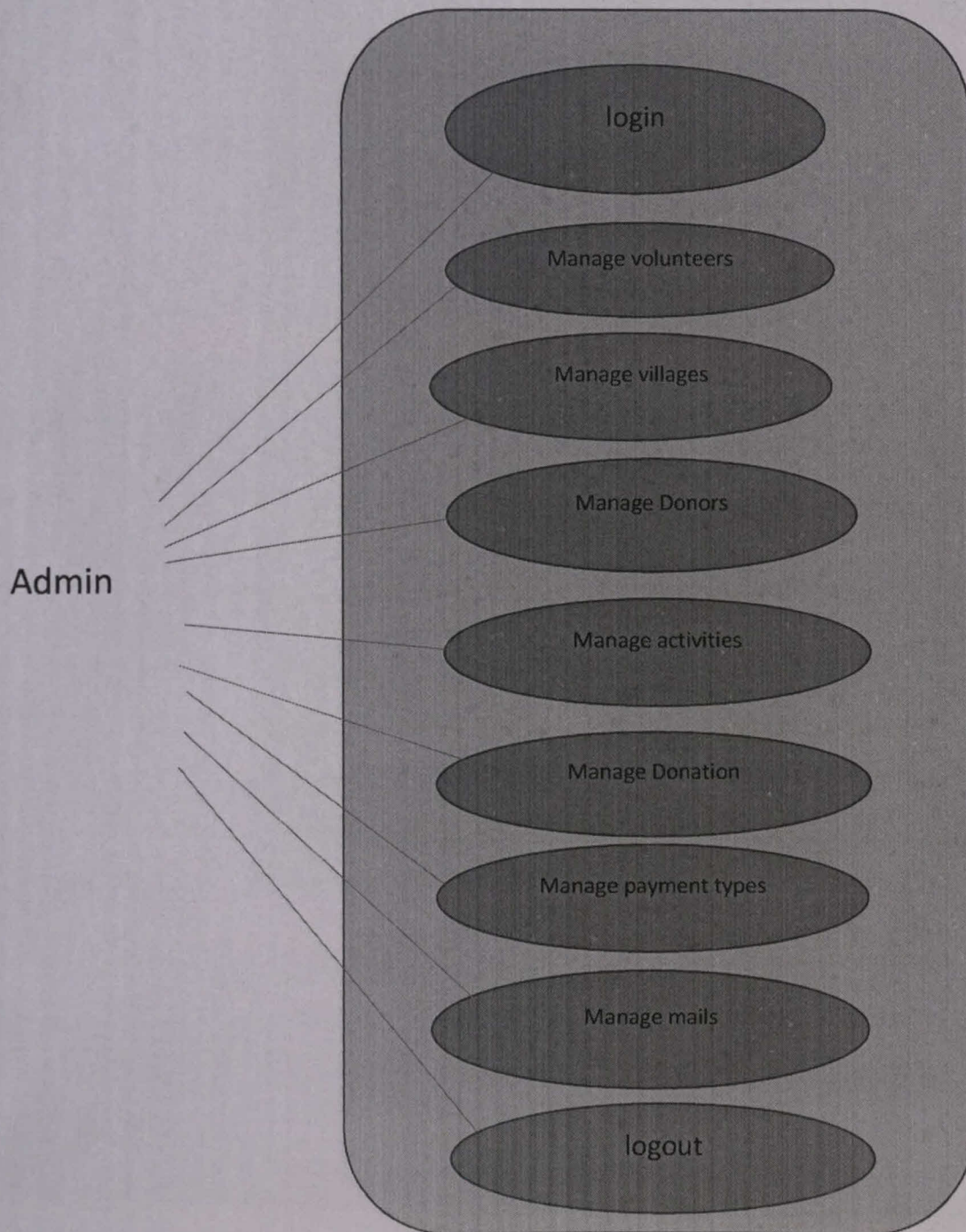
session_key	varchar(40)	NOT NULL PRIMARY KEY	
session_data	text	NOT NULL	

6.5 UML DIAGRAMS

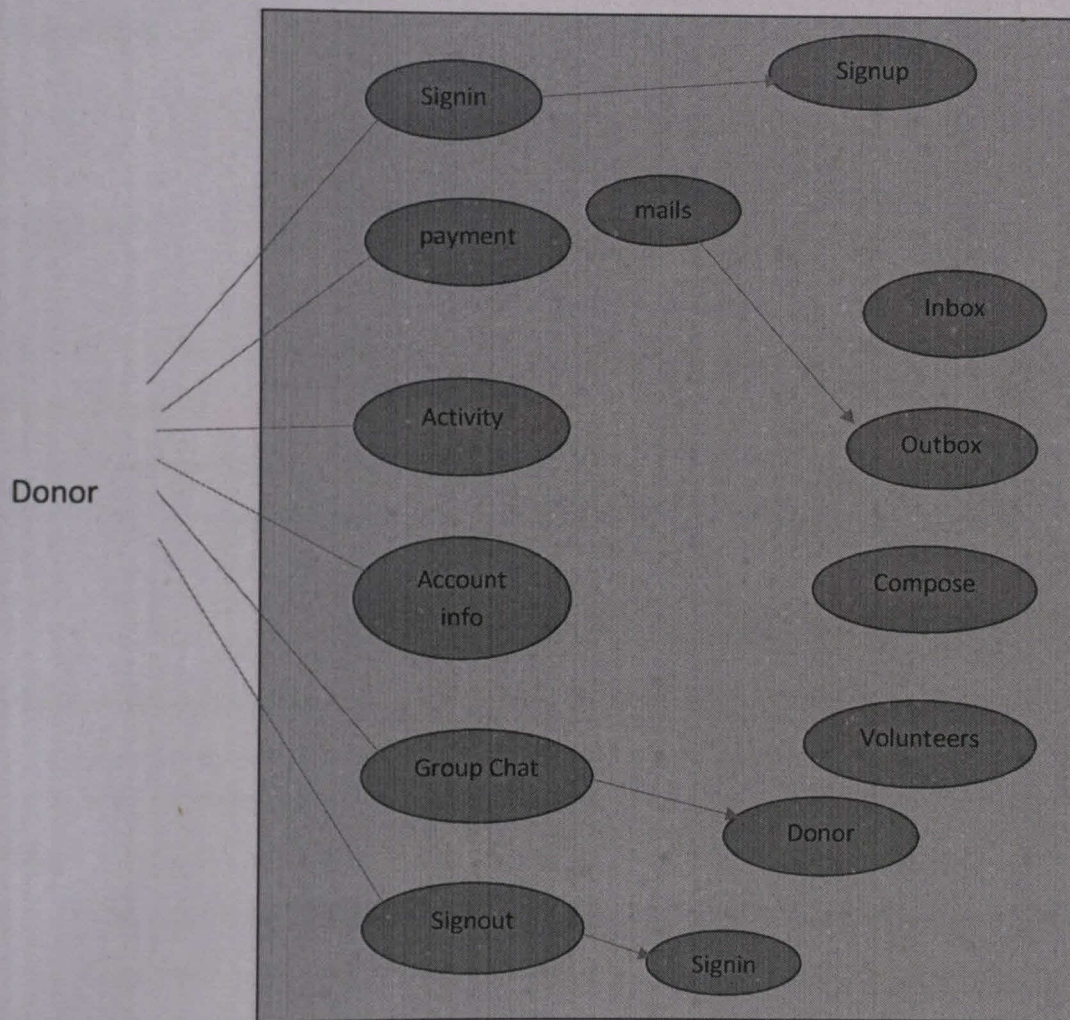
Use Case Diagram:



Admin Use Case Diagram:



Donor Use case Diagram:



BTMV Project Coding

7.SYSTEM TESTING AND IMPLEMENTATION

PythonSource Code

▼Btmv

▶blog

▼migrations

blog.html

```
{% extends 'blog/base.html' %}
```

```
{% block body %}
```

```
    <div class="container">
```

```
        <div class="row">
```

```
            <div class="col-lg-9">
```

```
                <div class="panel panel-default">
```

```
                    <div class="panel-heading">
```

```
                        <h4>{{ article.title }} <small id="small"><span class="label label-default">{{  
article.created_on }}</span></small></h4>
```

```
                    </div>
```

```
                    <div class="panel-body">
```

```
                        
```

```
                        <p>{{ article.detail|safe }}</p>
```

```
                    </div>
```

```
                    <div class="panel-footer">
```

```
                        <p align="right">By {{ user.get_username }}</p>
```

```
                    </div>
```

```
                </div>
```

</div>

<div class="col-lg-3">

<div class="list-group">

{% for item in article_side | slice:"0:5" %}

<h4 class="list-group-item-heading">{{ item.title | slice:"0:20" | safe }}</h4>

<p class="list-group-item-text">{{ item.detail | slice:"0:100" | safe }}</p>

{% endfor %}

</div>

</div>

</div>

</div>

{% endblock %}



blog\0001_initial.py

```
from __future__ import unicode_literals
```

```
from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    initial = True
```

```
    dependencies = [
    ]
```

```
    operations = [
```

```
        migrations.CreateModel(
```

```
            name='Article',
```

```
            fields=[
```

```
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False,
verbose_name='ID')),
```

```
                ('title', models.CharField(max_length=100)),
```

```
                ('detail', models.TextField()),
```

```
                ('created_on', models.DateTimeField(auto_now_add=True)),
```

```
                ('article_img', models.FileField(upload_to='')),
```

```
            ],
```

```
        ),
```

```
    ]
```

▼ static

blog\admin.py

```
from django.contrib import admin
from .models import Article
```

```
admin.site.register(Article)
```

blog\apps.py

```
from django.apps import AppConfig
```

```
class BlogConfig(AppConfig):
    name = 'blog'
```

blog\forms.py

```
from django import forms
from .models import Article
```

```
class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        fields = ('title', 'detail', )
```

blog\models.py

```
from django.db import models
from django.core.urlresolvers import reverse
```

```
class Article(models.Model):
    title=models.CharField(max_length=100)
    detail=models.TextField()
    created_on= models.DateTimeField(auto_now_add=True)
    article_img=models.FileField()
```

```
def get_absolute_url(self):
    return reverse('blog:index',kwargs={'pk': self.pk})
```

blog\urls.py

```
from django.conf.urls import url
from . import views

app_name = 'blog'
urlpatterns = [

    url(r'^$', views.home, name='home'),
    url(r'^about/$', views.about, name='about'),
    url(r'^contact/$', views.contact, name='contact'),
    url(r'^add/$', views.ArticleCreate.as_view(), name='add_article'),
    url(r'^(?P<pk>[0-9]+)$', views.index, name='index'),
    url(r'^adds/$', views.CreateArticle, name='adds_article')
]
```

Blog\views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponseRedirect, HttpRequest, HttpResponseRedirect
from django.contrib.auth import authenticate, login
from django.views.generic import View
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from .models import Article
from django.views.generic.edit import CreateView, UpdateView, DeleteView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.decorators import login_required
from .forms import ArticleForm

def index(request, pk):
    article= Article.objects.get(pk=pk)
    article_side= Article.objects.all()
    article_side=article_side.order_by('-created_on')

    return render(request, 'blog/blog.html', {'article': article, 'article_side': article_side})

def about(request):
    return render(request, 'blog/about.html')

def contact(request):
```

```
return render(request, 'blog/contact.html')
```

```
class ArticleCreate(CreateView):  
    model= Article  
    fields= ['title', 'detail','article_img']
```

```
def home(request):  
    article= Article.objects.all()  
    article= article.order_by('-created_on')  
    paginator=Paginator(article, 6)  
    page=request.GET.get('page')  
    try:  
        post=paginator.page(page)  
    except PageNotAnInteger:  
        post=paginator.page(1)  
    except EmptyPage:  
        post=paginator.page(paginator.num_pages)  
    return render(request,'blog/home.html',{'post': post, 'page':page})
```

```
def CreateArticle(request,):  
    if request.method=='POST':  
        form=ArticleForm(request.POST)  
        if form.is_valid():  
            article=form.save(commit=False)  
            article.save()  
            return HttpResponseRedirect('/blog')  
        else:  
            form=ArticleForm()  
            return render(request,'blog/test.html',{'form':form})
```

► Btmv

settings.py

"""

Django settings for btmv project.

Generated by 'django-admin startproject' using Django 1.10.4.

For more information on this file, see

<https://docs.djangoproject.com/en/1.10/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/1.10/ref/settings/>

"""

```
import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/1.10/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = '#+)kwsizdtrr%o%kx--1%tlfm=!y19s@w@bnlz#2rk&at^yyt*'
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
AUTH_PROFILE_MODULE='reguser.Reguser'
```

```
# Application definition
```

```
POSTMAN_AUTO_MODERATE_AS = True
```

```
LOGIN_URL='/register/login/'
```

```
INSTALLED_APPS = [
```

```
    'blog.apps.BlogConfig',
```

```
    'forum.apps.ForumConfig',
```

```
    'django.contrib.admin',
```

```
    'django.contrib.auth',
```

```
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
```

```
    'django.contrib.messages',
```

```
'django.contrib.staticfiles',
'register.apps.RegisterConfig',
'crispy_forms',
'django_wysiwyg',
'tinymce',
'postman',
'pytz'
]
```

```
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'btmv.urls'
```

```
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [os.path.join(BASE_DIR, 'templates')]
,
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
],
]
```

```
WSGI_APPLICATION = 'btmv.wsgi.application'
```

```
# Database
# https://docs.djangoproject.com/en/1.10/ref/settings/#databases
```

```
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3',
```



```
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),  
}  
}
```

Password validation

<https://docs.djangoproject.com/en/1.10/ref/settings/#auth-password-validators>

AUTH_PASSWORD_VALIDATORS = [

```
{  
    'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
},  
{  
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',  
},  
{  
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',  
},  
{  
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',  
},  
]
```

Internationalization

<https://docs.djangoproject.com/en/1.10/topics/i18n/>

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/1.10/howto/static-files/>

STATIC_URL = '/static/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

MEDIA_URL = '/media/'

CRISPY_TEMPLATE_PACK='bootstrap3'

btmv\urls.py

```
"""btmv URL Configuration
```

The `urlpatterns` list routes URLs to views. For more information please see:
<https://docs.djangoproject.com/en/1.10/topics/http/urls/>

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to `urlpatterns`: `url(r'^$', views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to `urlpatterns`: `url(r'^$', Home.as_view(), name='home')`

Including another `URLconf`

1. Import the `include()` function: `from django.conf.urls import url, include`
2. Add a URL to `urlpatterns`: `url(r'^blog/', include('blog.urls'))`

```
"""
```

```
from django.conf.urls import url,include
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
urlpatterns = [
```

```
url(r'^admin/', admin.site.urls),
url(r'^blog/',include('blog.urls')),
url(r'^forum/',include('forum.urls')),
url(r'^register/',include('register.urls')),
url(r'^tinymce/',include('tinymce.urls')),
url(r'^messages/',include('postman.urls',namespace="postman")),
]
```

```
if settings.DEBUG:
```

```
urlpatterns += static(settings.STATIC_URL,document_root=settings.STATIC_ROOT)
urlpatterns += static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```


wsgi.py

"""

WSGI config for btmv project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

<https://docs.djangoproject.com/en/1.10/howto/deployment/wsgi/>

"""

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "btmv.settings")
```

```
application = get_wsgi_application()
```

▶ forum

▼ migrations

0001_initial.py

```
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2017-01-20 18:14
from __future__ import unicode_literals

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion

class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Forum',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('topic_name', models.CharField(max_length=100)),
                ('describe', models.TextField()),
                ('created_on', models.DateTimeField(auto_now_add=True)),
                ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
        migrations.CreateModel(
            name='ForumPost',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('post', models.TextField()),
                ('latest_post', models.DateTimeField(auto_now_add=True)),
                ('forum', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE, to='forum.Forum')),
            ],
        ),
    ]
```

```
        ('user', models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to=settings.AUTH_USER_MODEL)),
    ],
),
]
```

0002_forum_total_post.py

```
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2017-01-23 04:34
from __future__ import unicode_literals

from django.db import migrations, models
```

```
class Migration(migrations.Migration):
```

```
    dependencies = [
        ('forum', '0001_initial'),
    ]
```

```
    operations = [
        migrations.AddField(
            model_name='forum',
            name='total_post',
            field=models.IntegerField(default=0),
        ),
    ]
```

forum\admin.py

```
from django.contrib import admin
from .models import Forum, ForumPost
admin.site.register(Forum)
admin.site.register(ForumPost)
```

forum\apps.py

```
from django.apps import AppConfig
```

```
class ForumConfig(AppConfig):
    name = 'forum'
```

forum\forms.py

```
from django import forms
from .models import ForumPost, Forum
```

```
class ForumForm(forms.ModelForm):
    class Meta:
        model=Forum
        fields=('topic_name','describe',)
```

```
class ForumPostForm(forms.ModelForm):
    class Meta:
        model=ForumPost
        fields= ('post',)
```

models.py

```
from django.db import models
from django.core.urlresolvers import reverse
from django.contrib.auth.models import User
```

```
class Forum(models.Model):
    topic_name= models.CharField(max_length=100)
    describe= models.TextField()
    created_on= models.DateTimeField(auto_now_add=True)
    user=models.ForeignKey(User)
    total_post=models.IntegerField(default=0)
```

```
def __str__(self):
    return 'Forum_id-'+str(self.id)
```

```
class ForumPost(models.Model):
    forum= models.ForeignKey(Forum,on_delete=models.CASCADE)
    user=models.ForeignKey(User)
    post= models.TextField()
    latest_post=models.DateTimeField(auto_now_add=True)
```

```
def __str__(self):
    return 'Forumpost_id-'+str(self.id)
```

urls.py

```
from django.conf.urls import url
from . import views

app_name = 'forum'
urlpatterns = [

    url(r'^$', views.home, name='home'),
    url(r'(?P<pk>[0-9]+)$', views.detail, name='detail'),
    url(r'^create/$', views.createForum, name='create'),
    url(r'^delete/(?P<pk>[0-9]+)/(?P<pk_1>[0-9]+)/$', views.deletePost, name='delete'),
    url(r'^edit/(?P<pk>[0-9]+)/(?P<pk_1>[0-9]+)/$', views.editPost, name='edit'),
    url(r'^userprofile/(?P<pk>[0-9]+)/$', views.userProfile, name='userprofile')

]
```

views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpRequest
from django.contrib.auth import authenticate, login
from django.views.generic import View
from django.core.paginator import Paginator, EmptyPage, PageNotAnInteger
from .models import Forum, ForumPost
from django.contrib.auth.models import User
from django.views.generic.edit import CreateView, UpdateView, DeleteView
from django.contrib.auth.mixins import LoginRequiredMixin
from .forms import ForumPostForm, ForumForm
from django.db.models import Count

def home(request):
    forum= Forum.objects.all()
    f=Forum.objects.annotate(num_post=Count('forumpost')).order_by('-num_post')
    paginator=Paginator(forum, 8)
    page=request.GET.get('page')
    try:
        post=paginator.page(page)
    except PageNotAnInteger:
        post=paginator.page(1)
    except EmptyPage:
```

```
post=paginator.page(paginator.num_pages)
return render(request, 'forum/home.html', {'page': page,'post':post,'f':f})
```

```
def detail(request,pk):
    forum=Forum.objects.get(pk=pk)
    post_set=forum.forumpost_set.all()
    paginator=Paginator(post_set, 8)
    page=request.GET.get('page')
    try:
        post=paginator.page(page)
    except PageNotAnInteger:
        post=paginator.page(1)
    except EmptyPage:
        post=paginator.page(paginator.num_pages)
    if request.method=='POST':

        form=ForumPostForm(request.POST)
        if form.is_valid():
            forumpost=form.save(commit=False)
            forumpost.forum=forum
            forumpost.user=request.user
            forumpost.save()
            return redirect('forum:detail',pk=pk)
        else:
            form=ForumPostForm()
    return
    render(request,'forum/detail.html',{'form':form,'post':post,'page':page,'forum':forum})
```

```
def createForum(request):
    forum=Forum.objects.all()
    if request.method=='POST':
        form=ForumForm(request.POST)
        if form.is_valid():
            forum_topic=form.save(commit=False)
            forum_topic.user=request.user
            forum_topic.save()
            return redirect('forum:home')
        else:
            form=ForumForm()
    return render(request,'forum/create.html',{'form':form})
```

```
def deletePost(request,pk,pk_1):
    forum=Forum.objects.get(pk=pk_1)
```

```
forum=forum.forumpost_set.filter(pk=pk).delete()
return redirect('forum:detail',pk_1)
```

```
def editPost(request,pk,pk_1):
instance=Forum.objects.get(pk=pk_1)
instance=instance.forumpost_set.get(pk=pk)
form = ForumPostForm(request.POST or None, instance=instance)
if form.is_valid():
form.save()
return redirect('forum:detail',pk_1)
return render(request,'forum/edit.html',{'form':form})
```

```
def userProfile(request,pk):
user=User.objects.get(pk=pk)
return render(request,'register/profile.html',{'user':user})
```

▼register

►migrations

register\0001_initial.py

```
# -*- coding: utf-8 -*-
# Generated by Django 1.10 on 2017-01-23 14:45
from __future__ import unicode_literals

from django.conf import settings
from django.db import migrations, models
import django.db.models.deletion

class Migration(migrations.Migration):

    initial = True

    dependencies = [
        migrations.swappable_dependency(settings.AUTH_USER_MODEL),
    ]

    operations = [
        migrations.CreateModel(
            name='Reguser',
            fields=[
                ('id', models.AutoField(auto_created=True, primary_key=True, serialize=False, verbose_name='ID')),
                ('birthday', models.DateField()),
                ('gender', models.CharField(max_length=10)),
                ('user', models.OneToOneField(on_delete=django.db.models.deletion.CASCADE, to=settings.AUTH_USER_MODEL)),
            ],
        ),
    ]
```


register\admin.py

```
from django.contrib import admin
from .models import Reguser
admin.site.register(Reguser)
```

register\apps.py

```
from django.apps import AppConfig
```

```
class RegisterConfig(AppConfig):
    name = 'register'
```

register\forms.py

```
from django import forms
from django.contrib.auth.models import User
from django.forms import ModelForm
from register.models import Reguser
```

```
class RegistrationForm(ModelForm):
    username= forms.CharField(label='User Name')
    email= forms.EmailField(label='Email Address')
    password=
    forms.CharField(label='Password',widget=forms.PasswordInput(render_value=False))
    password1=
    forms.CharField(label='Password1',widget=forms.PasswordInput(render_value=False))
```

```
class Meta:
    model=Reguser
    exclude=('user',)
```

```
def clean_username(self):
    username= self.cleaned_data['username']
    try:
        User.objects.get(username=username)
    except User.DoesNotExist:
        return username
    raise forms.ValidationError("The username is already taken")
```

```
def clean(self):
    password = self.cleaned_data['password']
    password1 = self.cleaned_data['password1']
    if self.cleaned_data['password'] != self.cleaned_data['password1']:
```

```
raise forms.ValidationError('The password did not match')
return self.cleaned_data
```

```
class LoginForm(forms.Form):
    username= forms.CharField(label='Username')
    password=
    forms.CharField(label='password',widget=forms.PasswordInput(render_value=False))
```

register\models.py

```
from django.db import models
from django.contrib.auth.models import User
#from django.db.models.signals import post_save
```

```
class Reguser(models.Model):
    user= models.OneToOneField(User)
    birthday= models.DateField()
    gender=models.CharField(max_length=10)
```

```
def __unicode__(self):
    return self.name
```

```
#def create_reguser_user_callback(sender,instance,**kwargs):
#    # reguser, new=Reguser.objects.get_or_create(user=instance)
#    # post_save.connect(create_reguser_user_callback,User)
```

register\urls.py

```
from django.conf.urls import url
from . import views
```

```
app_name = 'register'
urlpatterns = [
    url(r'^$', views.ReguserRegistration, name='register'),
    url(r'^login/$', views.LoginRequest, name='login'),
    url(r'^logout/$', views.LogoutRequest, name='logout'),
    url(r'^profile/$', views.UserProfile, name='profile'),
    url(r'^password/$', views.change_password, name='change_password'),
]
```

register\views.py

```
from django.core.paginator import Paginator, PageNotAnInteger, EmptyPage
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.http import HttpResponseRedirect
from django.shortcuts import render_to_response
from django.template import RequestContext

from blog.models import Article
from register.forms import RegistrationForm, LoginForm
from register.models import Reguser
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from django.contrib.auth import update_session_auth_hash
from django.contrib.auth.forms import PasswordChangeForm

def ReguserRegistration(request):
    if request.user.is_authenticated():
        return HttpResponseRedirect('/register/profile/')
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            user =
            User.objects.create_user(username=form.cleaned_data['username'], email=form.cleaned_data['email'],
            password=form.cleaned_data['password'])
            user.save()
            reguser =
            Reguser(user=user, birthday=form.cleaned_data['birthday'], gender=form.cleaned_data['gender'])
            reguser.save()
            return HttpResponseRedirect('profile/')
        else:
            return render(request, 'register/register.html', {'form': form})
        else:
            form = RegistrationForm()
            context = {'form': form}

            return render(request, 'register/register.html', context)

def LoginRequest(request):
    article = Article.objects.all()
```

```

article= article.order_by('-created_on')
paginator=Paginator(article, 6)
page=request.GET.get('page')
try:
post=paginator.page(page)
except PageNotAnInteger:
post=paginator.page(1)
except EmptyPage:
post=paginator.page(paginator.num_pages)
if request.user.get_username():
return render(request,'blog/home.html',{'post':post})
if request.method=='POST':
form=LoginForm(request.POST)
if form.is_valid():
username = form.cleaned_data['username']
password = form.cleaned_data['password']
reguser=authenticate(username=username,password=password)
if reguser is not None:
login(request,reguser)
return render(request,'blog/home.html',{'post':post})
else:
invalid='Invalid username or password'
return render(request,'register/login.html',{'form':form,'invalid':invalid})
else:
return render(request,'register/login.html',{'form':form})
else:
form=LoginForm()
context = {'form':form}
return render(request,'register/login.html',context,{'post':post})

def LogoutRequest(request):
logout(request)
return HttpResponseRedirect('/register/login')

def UserProfile(request):
return render(request,'register/profile.html')

def editUser(request,pk):
user=User.objects.get(pk=pk)
form=RegistrationForm(request.POST or None, instance=user)
if form.is_valid():

```

```
form.save()
return redirect('register:login')
return render(request, 'register/register.html', {'form': form})
```

```
def change_password(request):
if request.method == 'POST':
form = PasswordChangeForm(request.user, request.POST)
if form.is_valid():
user = form.save()
update_session_auth_hash(request, user)
messages.success(request, 'Your password was successfully updated!')
return redirect('register:profile')
else:
messages.error(request, 'Please correct the error below.')
else:
form = PasswordChangeForm(request.user)
return render(request, 'register/change_password.html', {'form': form})
```

HTML Files

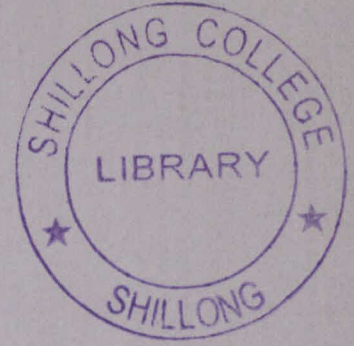
▼ templates

► blog

blog\home.html

```
{% extends 'blog/base.html' %}
{% block body %}
<div class="container-fluid" >
<div class="col-md-10">
    {% for item in post %}
<div class="row">
<div class="panel panel-default" style="margin-left: 15%" >
<div style="margin-left: 5%">
<div class="title" align=left>
<h3>{{ item.title }}</h3>
        <h6 align="right" style="color: black" ><span class="label label-default"><i
        class="glyphicon glyphicon-calendar">
        </i>{{ item.created_on|slice:"0:10" }}</span></h6>
<div class="jumbotron">
        
</div>

</div>
<div class="created-on" align=left>
<p>{{item.detail|slice:"0:500"|safe}}</p>
</div>
<div align="right">
        <a href="{% url 'blog:index' item.pk %}"><span class="label label-
        primary">Read more</span></a>
</div>
</div>
</div>
</div>
        {% endfor %}
<div align="right">
        {% include 'blog/paginator.html' with page=post %}
</div>
</div>
<div class="col-md-2" style="margin-left: 20px">
<div class="list-group" style="background-color: orange">
        {% for item in article %}
<a href="{% url 'blog:index' item.id %}" class="list-group-item" style="background-color:
azure">
```



```
<h4 class="list-group-item-heading">{{ item.title }}</h4>
```

```
<p class="list-group-item-text">{{item.detail | slice:"0:200" | safe}}</p>  
</a>
```

```
{% endfor %}
```

```
</div>
```

```
</div>
```

```
</div>
```

```
{% endblock %}
```



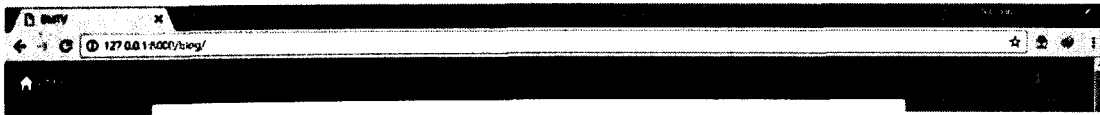
base.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>BMTV</title>
  {% load staticfiles %}
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script
src="http://maxcdn.bootstrapcdn.com/bootstrap/3.3.4/js/bootstrap.min.js"></script>
<script type="text/javascript" src="{% static 'js/tiny_mce/tiny_mce.js' %}"></script>
<style type="text/css">
  .navbar-inverse{
margin: 0px;
  }
  .created-on{
margin: 5px;
  }

</style>
<script type="text/javascript">
tinyMCE.init({
mode:"textareas"
});
</script>
</head>
<body style="background-color:lightslategrey">
  <nav class="navbar navbar-inverse" style="">
    <div class="container-fluid">
      <!--logo-->
      <div class="navbar-header">
        <a href="{% url 'blog:home' %}" class="navbar-brand"><span
class="glyphicon glyphicon-home"></span>BTMV </a>
      </div>
      <!--menu items-->
      <div>
        <ul class="nav navbar-nav">
          <li class="#"><a href="{% url 'blog:add_article' %}"><span class="glyphicon
glyphicon-pencil"></span> Article </a></li>
          <li class="#"><a href="{% url 'forum:home' %}" ><span class="glyphicon
glyphicon-comment"></span> Forum</a></li>
          <li class="#"><a href="{% url 'blog:about' %}"><span class="glyphicon
glyphicon-globe"></span> About</a></li>
        </ul>
      </div>
    </div>
  </nav>
</body>
</html>
```



```
<li class="#"><a href="{% url 'blog:contact' %}"><span class="glyphicon glyphicon-  
phone"></span> Contact Us</a></li>  
</ul>  
  <ul class="nav navbar-nav navbar-right">  
    {% if user.is_authenticated %}  
<li class="#"><a href="{% url 'register:profile' %}"><span class="glyphicon glyphicon-  
user"></span>{{request.user.get_username}}</a></li>  
    {% else %}  
<li class="#"><a href="{% url 'register:login' %}">Sign up</a></li>  
    {% endif %}  
  </ul>  
</div>  
</div>  
</nav>  
{% block body %}  
  
{% endblock %}
```





blog.html

```
{% extends 'blog/base.html' %}
{% block body %}
    <div class="container">
        <div class="row">
            <div class="col-lg-9">
                <div class="panel panel-default">
<div class="panel-heading">
    <h4>{{ article.title }} <small id="small"><span class="label label-default">{{
    article.created_on }}</span></small></h4>
</div>
    <div class="panel-body">
        
        <p>{{ article.detail | safe }}</p>
    </div>
<div class="panel-footer">
<p align="right">By {{ user.get_username }}</p>
</div>
    </div>
</div>
    <div class="col-lg-3">
        <div class="list-group">
            {% for item in article_side | slice:"0:5" %}
                <a href="{% url 'blog:index' item.id %}" class="list-group-item">
<h4 class="list-group-item-heading">{{ item.title | slice:"0:20" | safe }}</h4>
<p class="list-group-item-text">{{ item.detail | slice:"0:100" | safe }}</p>
                </a>
            {% endfor %}
        </div>
    </div>
</div>
</div>
</div>
{% endblock %}
```

BMTV Select user to change: Nicholas

127.0.0.1:8000/blog/28

BMTV Article Forum About Contact us Nicholas

Implementing Different Activities to Village Area April 4, 2017, 7:25 AM



Implementing Different
The main thrust of the women's development activities would be to assist women in the sustainable

Governments Policies
Education is one of the most important means of empowering women with the knowledge, skills and

Jermanal LP School
When it comes to the safety of school children, the multi-purpose hall also works as a gym for th

Khuswai face poor dr
Drinking water supply and sanitation in India continue to be inadequ

Sohtad village, Umko
A slum is a heavily populated urban informal settlement characterized by substandard housing and

127.0.0.1:8000/blog/

Windows taskbar: e, P, W, Ps, X, ENG 14:37 10-04-2017

Article_form.html

```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container">
<div class="panel panel-default" style="padding: 10px">

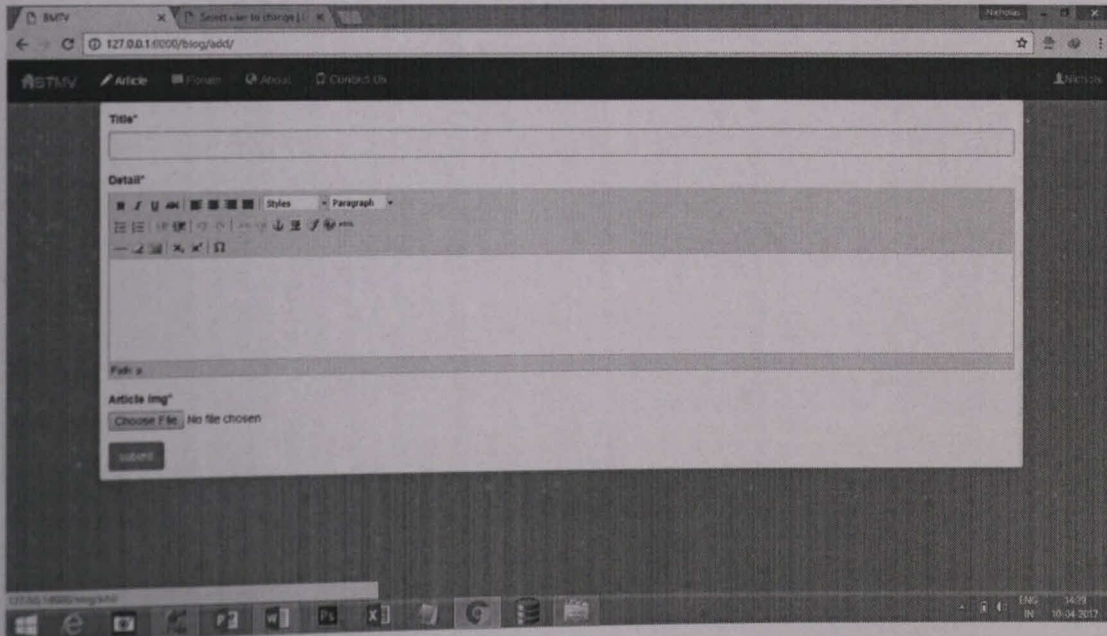
<form method="post" novalidate action="" enctype="multipart/form-data">
    {% csrf_token %}
    {{ form|crispy }}
<button class="btn btn-success">submit</button>

</form>
</div>
</div>

{% endblock %}
```

form_template.html

```
{% for field in form %}
<br><label>{{ field.label_tag }}</label><br>
    {{ field }}
{% endfor %}
```



panigator.html

```
<nav aria-label="page navigation" style="align-content: flex-end">
  {% if page.has_other_pages %}
<ul class="pagination">
  {% if page.has_previous %}
<li><a aria-label="Previous" href="?page={{ page.previous_page_number }}">&laquo;
</a></li>
  {% else %}
<li class="disabled"><span>&laquo;</span></li>
  {% endif %}
  {% for pages in page.paginator.page_range %}
  {% if page.number == pages %}
<li class="active"><span>{{ pages }}<span class="sr-only">(current)</span></span></li>
  {% else %}
<li><a href="?page={{ pages }}">{{ pages }}</a></li>
  {% endif %}
  {% endfor %}
  {% if page.has_next %}
<li><a href="?page={{ page.next_page_number }}">&raquo; </a></li>
  {% else %}
<li class="disabled"><span>&raquo;</span></li>
  {% endif %}
</ul>
  {% endif %}
</nav>
```

about.html

```
{% extends 'blog/base.html' %}
{% block body%}
<div class="container">
<div class="col-md-12">
<div class="panel panel-default">
<div class="wrapper">

<div>
<p>The project is a charity group of professionals those want to voluntarily contribute in
their village/town's development. Issues like Primary education, people's health,
government policies awareness and availability of basic facilities/infrastructure are on main
focus among others. Through the website group want to help their members collaborate, to
plan, assess and implement different activities and learn with others
experience/feedbacks/suggestions. Group also wants to encourage others to join their
initiatives and recognize their contributions. The existing system is a semi-automated at
where the information is stored in the form of excel sheets in disk drives. The information
sharing to the Volunteers, Group members, etc. is through mailing feature only. The
information storage and maintenance is more critical in this system. Tracking the member's
activities and progress of the work is a tedious job here.</p>
</div>
<div>
<p> This system cannot provide the information sharing by 24x7 days. The development of
this new system objective is to provide the solution to the problems of existing system. By
using this new system, we can fully automate the entire process of the current system. The
new system would like to make as web-enabled so that the information can be shared
between the members at any time using the respective credentials.</p>
</div>
<div>
<p> To track the status of an individual process, the status update can be centralized using
the new system. Being a web-enabled system, the process can be accessed across the world
over net. This system also providing the features like Chatting, Mailing between the
members; Images Upload – Download via the web site; updating the process status in
centralized location; generated reports can also be exporting to the applications like MS-
Excel, PDF format, etc. In this new system, the members like Donors can give their valuable
feedback to the Volunteers so that the Volunteers can check their progress of the tasks. The
entire process categorized as different modules like Admin module, Volunteer module,
etc. at where we can classify the functionality as an individual process. Using the new system
entering into Admin module we can perform. In this new system using the Volunteer module
we can do. In the Reports module we can generate reports like Weekly Status Report.</p>
</div>
<div>
```

The project is fully integrated with Customer Relationship Management (CRM) solution and developed in a manner that is easily manageable, time saving and relieving one form semi automated. Back To My Village is a charity group of professionals those want to voluntarily contribute in their village/town's development. Issues like Primary education, people's health, government policies awareness and availability of basic. Through the website group want to help their members collaborate, to plan, assess and implement different activities and learn with others experience/feedbacks/ suggestions. Group also wants to encourage others to join their initiatives and recognize their contributions.

</div>

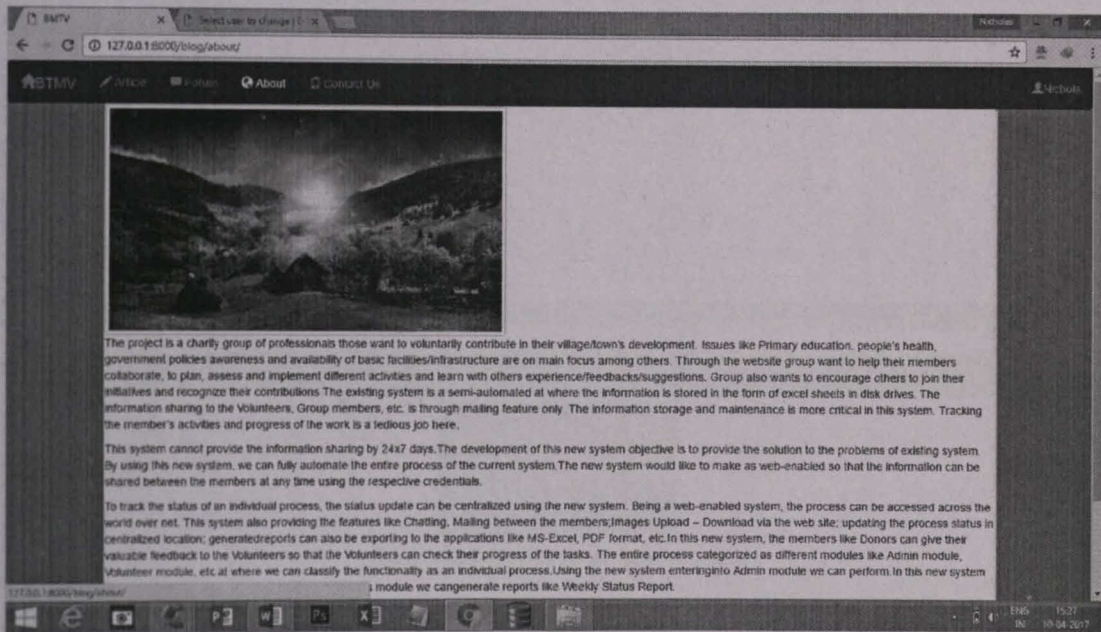
</div>

</div>

</div>

</div>

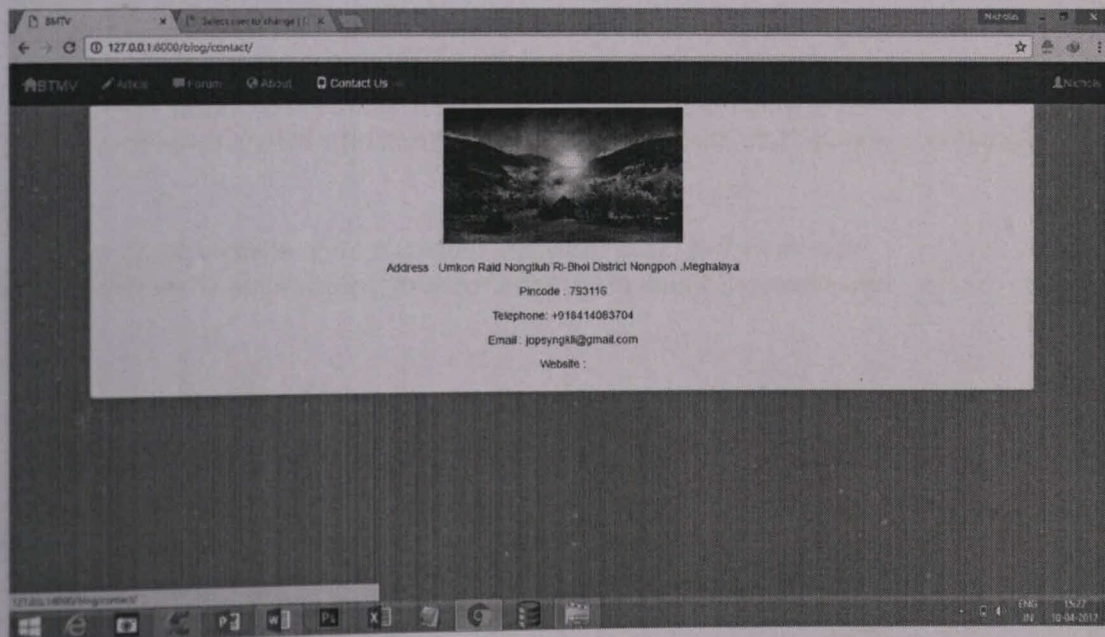
{% endblock %}



contact.html

```
{% extends 'blog/base.html' %}
{% block body %}
<div class="container">
<div class="row" id="box-search">
<div class="thumbnail text-center">

<div class="caption">
<p>
<p>Address : Umkon Raid Nongtluh Ri-Bhoi District Nongpoh ,Meghalaya</p>
<p>Pincode : 793116</p>
<p> Telephone: +918414083704</p>
<p> Email : jopsyngkli@gmail.com</p>
<p>Website :</p>
</p>
</div>
</div>
</div>
</div>
{% endblock %}
```



► forum

forum\home.html

```
{% extends 'blog/base.html' %}
{% block body %}
<div class="container">
<div class="row">
<div class="col-lg-2">
    <a href="{% url 'forum:create' %}"><button class="btn btn-default"><small>New
    Thread</small></button></a>
</div>
</div>
<div class="row">
<div class="col-lg-9">
<div class="panel panel-default">
<table class="table">
<thead>
<th>Sl.no</th>
<th>Topic for Discussion</th>
<th>read</th>

</thead>
    {% for item in post %}
<tr>
<td>{{ item.id }}</td>
<td>
<p><a href="{% url 'forum:detail' item.id %}"> {{ item.topic_name }}</a></p>
<p>created by-<a href="{% url 'forum:userprofile' item.user.id%}">{{ item.user }}</a></p>
</td>
<td>
<p align="right"><small>{{ item.created_on|slice:"0:11" }}</small></p>
<p align="right"><small>Post-{{ item.forumpost_set.count }}</small></p>
</td>
</tr>
    {% endfor %}
</table>
</div>
<div align="right">
    {% include 'blog/paginator.html' with page=post %}
</div>
</div>
<div class="col-lg-3" style="margin-bottom: 10px">
<p style="color: white">Most Post Thread</p>
    {% for item in f %}
<div class="panel panel-default" style="margin-bottom: 0px">
```

```
<a href="{% url 'forum:detail' item.id %}" style="color:orangered">{{ item.topic_name }}</a>
```

```
</div>
```

```
{% endfor %}
```

```
{% if user.is_authenticated %}
```

```
<div class="panel panel-default" style="margin-top: 20px" >
```

```
<form action="https://www.paypal.com/cgi-bin/webscr" method="post" target="_top">
```

```
<input type="hidden" name="cmd" value="_s-xclick">
```

```
<input type="hidden" name="hosted_button_id" value="X4TXWBRL2TP7N">
```

```
<table><tr><td><input type="hidden" name="on0" value="Donate
```

```
Options">DONATE</td></tr><tr><td><select name="os0">
```

```
<option value="Option 1"> $1.00 USD </option>
```

```
<option value="Option 2"> $3.00 USD </option>
```

```
<option value="Option 3"> $5.00 USD </option>
```

```
<option value="Option 4"> $8.00 USD </option>
```

```
<option value="Option 5"> $10.00 USD </option>
```

```
<option value="Option 6"> $13.00 USD </option>
```

```
<option value="Option 7"> $15.00 USD </option>
```

```
<option value="Option 8"> $17.00 USD </option>
```

```
<option value="Option 9"> $20.00 USD </option>
```

```
<option value="Option 10"> $23.00 USD</option>
```

```
<option value="Option 11"> $25.00 USD </option>
```

```
<option value="Option 12"> $28.00 USD </option>
```

```
<option value="Option 13"> $30.00 USD</option>
```

```
<option value="Option 14"> $35.00 USD </option>
```

```
<option value="Option 15"> $40.00 USD </option>
```

```
<option value="Option 16"> $45.00 USD </option>
```

```
<option value="Option 17"> $50.00 USD </option>
```

```
</select></td></tr>
```

```
</table>
```

```
<input type="hidden" name="currency_code" value="USD">
```

```
<input type="image" src="http://www.pngall.com/wp-content/uploads/2016/05/PayPal-Donate-Button-PNG-File.png" border="0" name="submit" alt="PayPal – The safer, easier way to pay online!">
```

```

```

```
</form>
```

```
</div>
```

```
{% endif %}
```

```
</div>
```

```
{% endblock %}
```

BMV Select user to change | Nichols

127.0.0.1:8000/forum/

Forum

New Thread

Sl.no	Topic for Discussion	read
7	Government's Policies Awareness Program to Differents Villages created by-Nichols	April 5, 2017, 5:18 a.m. Post-7
8	Implementing Differents Activities created by-Nichols	April 5, 2017, 6:20 a.m. Post-7

Most Read Thread

Government's Policies Awareness Program to Differents Villages
Implementing Differents Activities

DONATE
\$1.00 USD

Donate

127.0.0.1:8000/forum/

Windows Taskbar: File Explorer, Word, PowerPoint, Photoshop, Xcode, Chrome, Firefox, VLC, Steam, Spotify, Skype, OneDrive, Mail, Calendar, Photos, Settings, Task View, Search, Start, Network, Volume, Battery, Date/Time: EN 15:34 10-04-2017

BMV Select user to change | Nichols

127.0.0.1:8000/forum/

Forum

New Thread

Sl.no	Topic for Discussion	read
7	Government's Policies Awareness Program to Differents Villages created by-Nichols	April 5, 2017, 5:18 a.m. Post-7
8	Implementing Differents Activities created by-Nichols	April 5, 2017, 6:20 a.m. Post-7

Most Read Thread

Government's Policies Awareness Program to Differents Villages
Implementing Differents Activities

DONATE

- \$1.00 USD
- \$1.00 USD
- \$3.00 USD
- \$5.00 USD
- \$8.00 USD
- \$10.00 USD
- \$13.00 USD
- \$15.00 USD
- \$17.00 USD
- \$20.00 USD
- \$23.00 USD
- \$25.00 USD
- \$28.00 USD
- \$30.00 USD
- \$35.00 USD
- \$40.00 USD
- \$45.00 USD
- \$50.00 USD

ate

127.0.0.1:8000/forum/

Windows Taskbar: File Explorer, Word, PowerPoint, Photoshop, Xcode, Chrome, Firefox, VLC, Steam, Spotify, Skype, OneDrive, Mail, Calendar, Photos, Settings, Task View, Search, Start, Network, Volume, Battery, Date/Time: EN 15:34 10-04-2017

Billing Information - PayPal | Select user to charge | C... X

PayPal, Inc. [US] | https://www.paypal.com/cgi-bin/webscr

jopsyngkli@gmail.com

Billing Information PayPal Secure Payments

* Required

Description	Terms	Amount
Donate Option: Option 1	\$3.00 USD for each year	\$3.00 USD

Choose a Payment Method

You need a PayPal account for this purchase.

PayPal I already have a PayPal account.

I need to create a PayPal account (where available). [Learn more](#)

Country

*Country:

Credit or Debit Card Information

*First Name:

(as it appears on card)

*Last Name:

(as it appears on card)

*Card Type:

*Card Number:

*Expiration Date:

*Card Security Code:

Billing Information - PayPal | Select user to charge | C... X

PayPal, Inc. [US] | https://www.paypal.com/cgi-bin/webscr

*Card Security Code: [What's this?](#)

Billing Address

*Address line 1:

Address line 2:

*City:

*State:

*ZIP code:

*Is this your shipping address? Yes, it is the same as my shipping address

No

Contact Information

This information will only be used to contact you regarding your payment, if needed.

*Email Address:

*Home Telephone:

[Privacy](#)

Security Check

Type characters as shown in the box. [Help](#)

*Enter the characters:

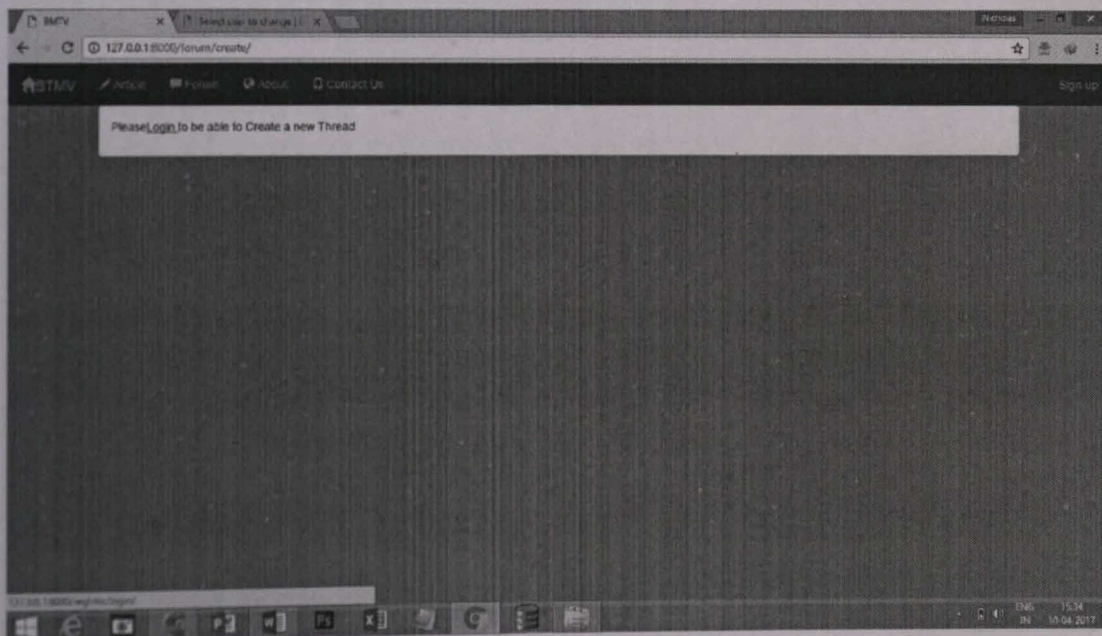
(not case sensitive)

[Need help?](#)

[Continue](#)

create.html

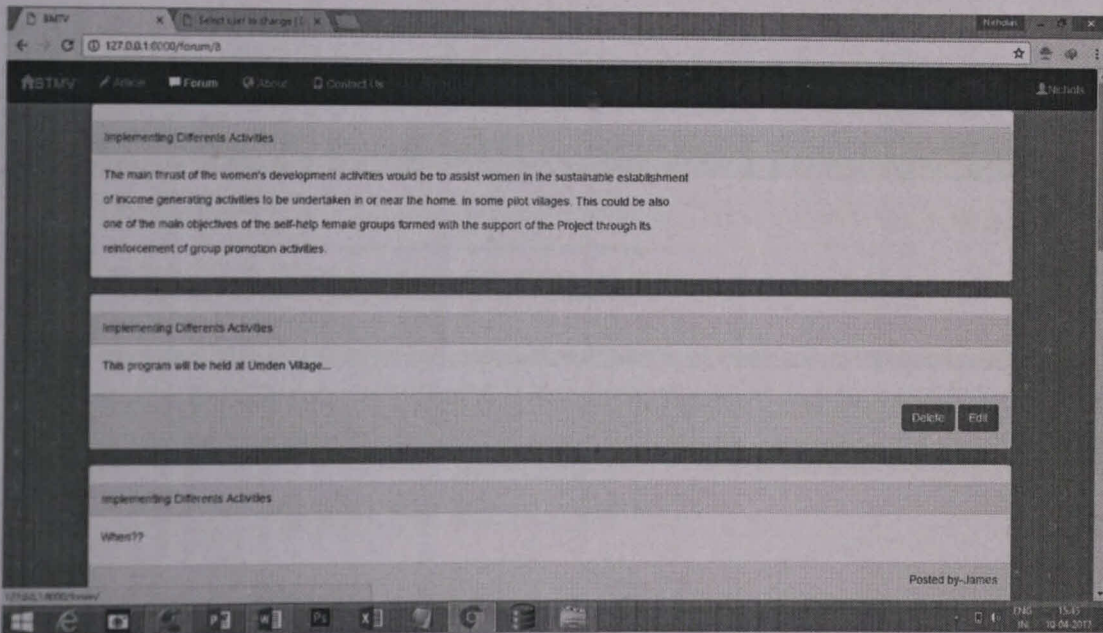
```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container">
<div class="panel panel-default">
<div class="panel-body">
    {% if user.is_authenticated %}
<form action="" novalidate method="post">
    {% csrf_token %}
    {{ form|crispy }}
<button class="btn btn-success">submit</button>
</form>
    {% else %}
<p> Please <a href="{% url 'register:login' %}">Login </a> to be able to Create a new
Thread</p>
    {% endif %}
</div>
</div>
</div>
{% endblock %}
```



edit.html

```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container">
<form action="" method="post">
    {% csrf_token %}
    {{ form|crispy }}
<button class="btn btn-success" type="submit">Submit</button>
</form>
</div>

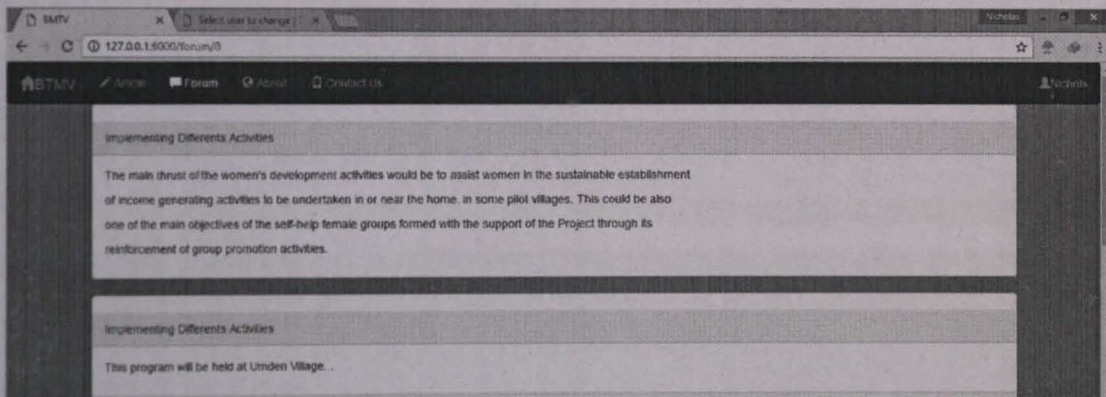
{% endblock %}
```



details.html

```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container">
<form action="" method="post">
    {% csrf_token %}
    {{ form|crispy }}
<button class="btn btn-success" type="submit">Submit</button>
</form>
</div>

{% endblock %}
```



▶ register

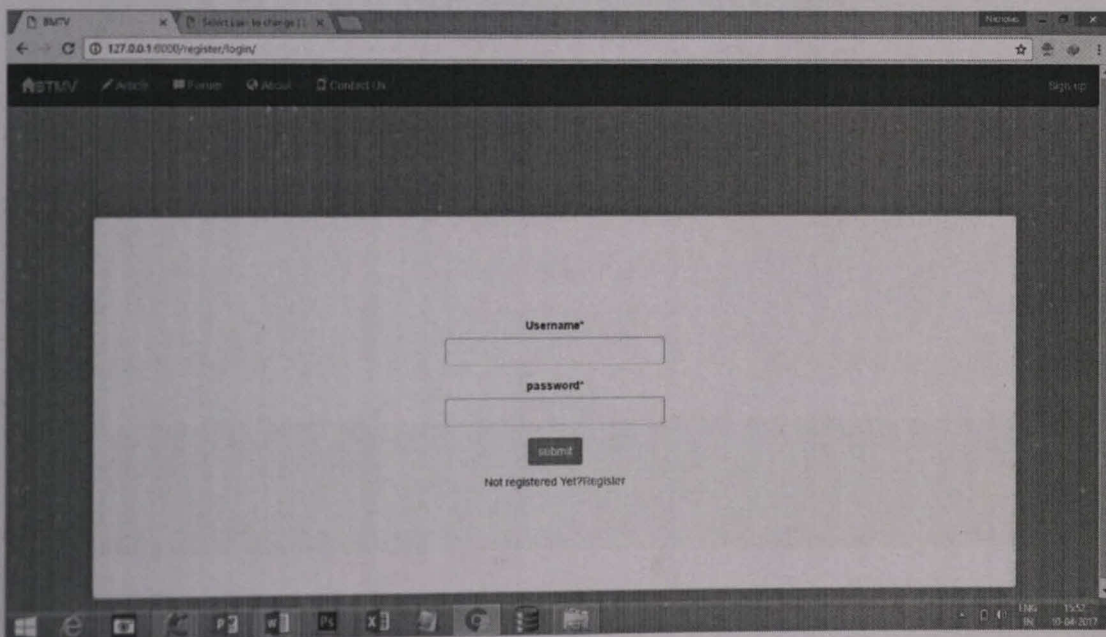
login.html

```
{% extends 'blog/home.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container" align="center" style="margin-top: 10%">
<div class="panel panel-default" style="padding: 10%">
<div style="width: 30%">
<form action="" method="post">
    {% csrf_token %}
<p class="text-danger"> {% if invalid%}{{ invalid }} {% endif %}</p>
    {{ form | crispy }}
<p><input type="submit" value="submit" class="btn btn-success"></p>
    {% if invalid%}Not registered Yet?<a href="{% url 'register:register'
%}">Register</a>
    {% endif %}
<p>Not registered Yet?<a href="{% url 'register:register' %}">Register</a></p>
</div>

</div>

</div>

{% endblock %}
```

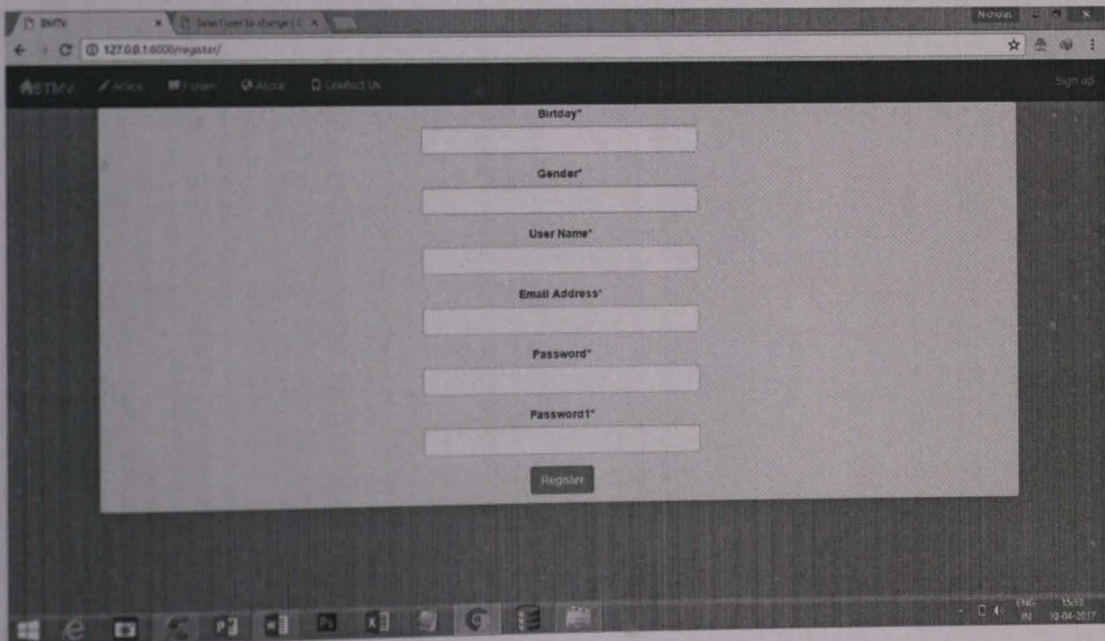


register.html

```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container" align="center">
<div class="panel panel-default" style="background-color:aliceblue">
<div style="width: 30%">
<form action="" method="post">
    {% csrf_token %}
    {% if form.errors %}Please {% endif %}
    {{ form|crispy }}
<p><input type="submit" value="Register" class="btn btn-success"></p>
</form>
</div>
</div>

</div>

{% endblock %}
```



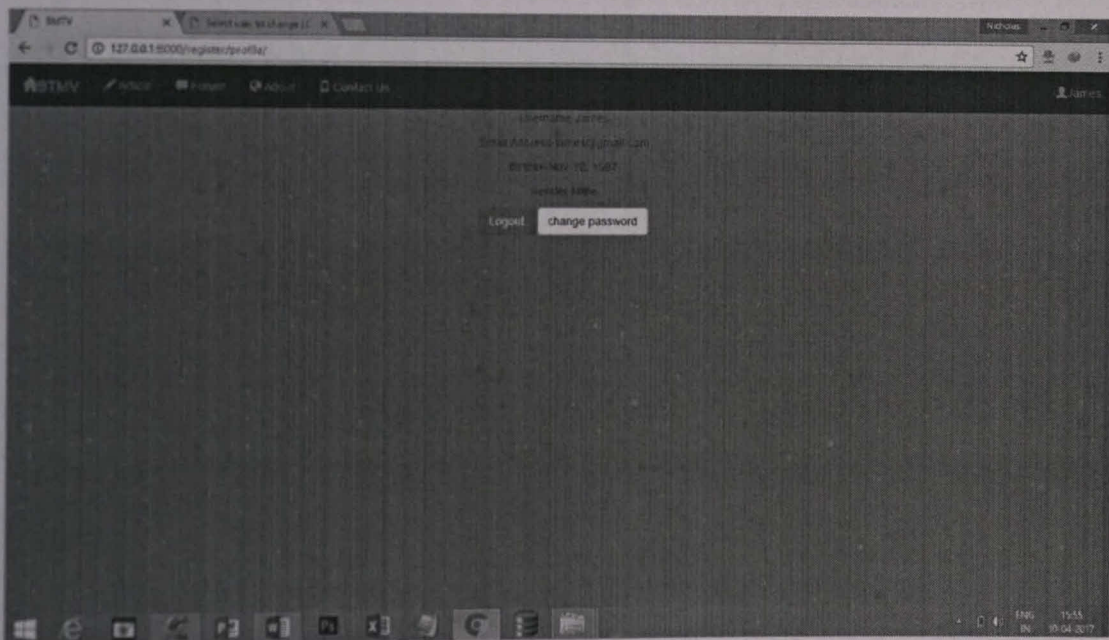
The screenshot shows a web browser window with the URL `127.0.0.1:8000/register/`. The page displays a registration form with the following fields:

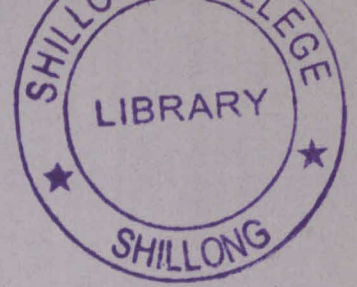
- Birthday*
- Gender*
- User Name*
- Email Address*
- Password*
- Password1*

At the bottom of the form is a button labeled "Register". The browser's taskbar at the bottom shows the Windows logo, several application icons, and the system tray with the date and time "10-04-2017 15:33".

profile.html

```
{% extends 'blog/base.html' %}
{% block body %}
<div class="container" align="center">
<p>Username-{{ user.get_username }}</p>
<p>Email Address-{{ user.email }}</p>
<p>Birtday-{{ user.reguser.birtday }}</p>
<p>Gender-{{ user.reguser.gender }}</p>
    {% if user.id == request.user.id %}
<p><a href="{% url 'register:logout' %}"><button class="btn btn-danger" value="Logout"
>Logout</button></a>
<a href="{% url 'register:change_password' %}"><button class="btn btn-default"
value="Logout" >change password</button></a></p>
    {% endif %}
</div>
{% endblock %}
```

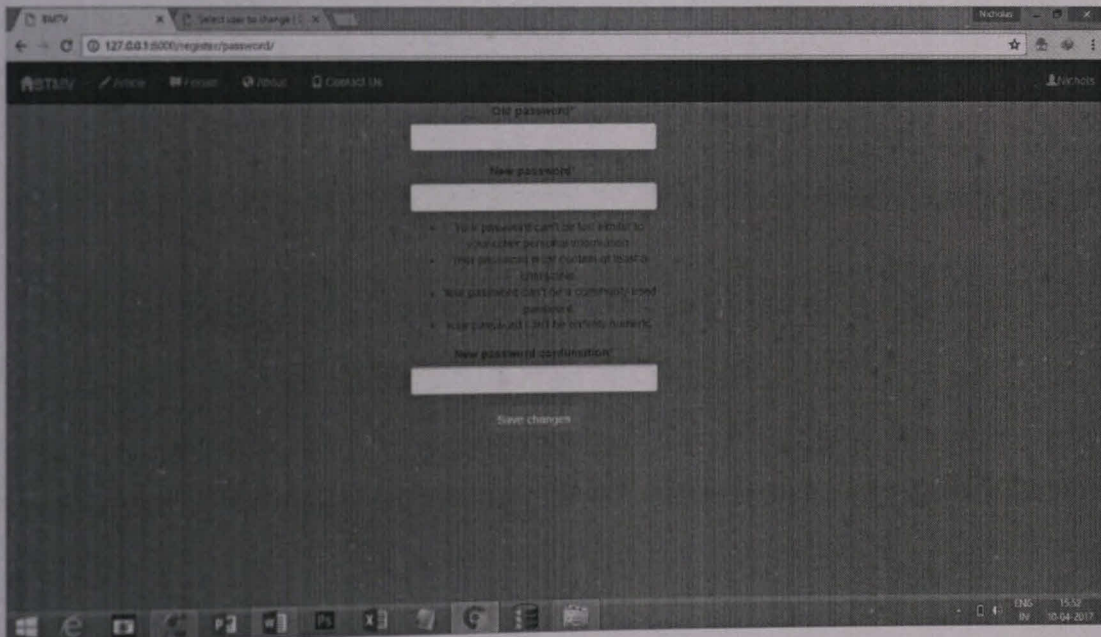




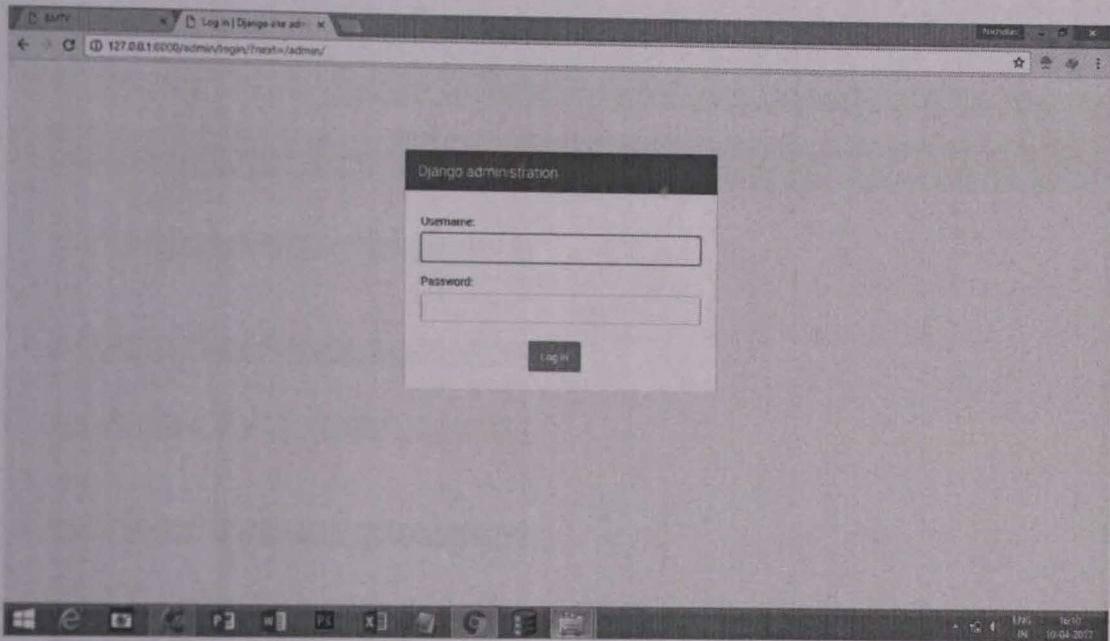
change_password.html

```
{% extends 'blog/base.html' %}
{% load crispy_forms_tags %}
{% block body %}
<div class="container-fluid">
<div class="col-lg-3" align="center" style="margin-left: 35%">
<form method="post">
    {% csrf_token %}
    {{ form|crispy }}
<button type="submit" class="btn btn-success">Save changes</button>
</form>
</div>
</div>

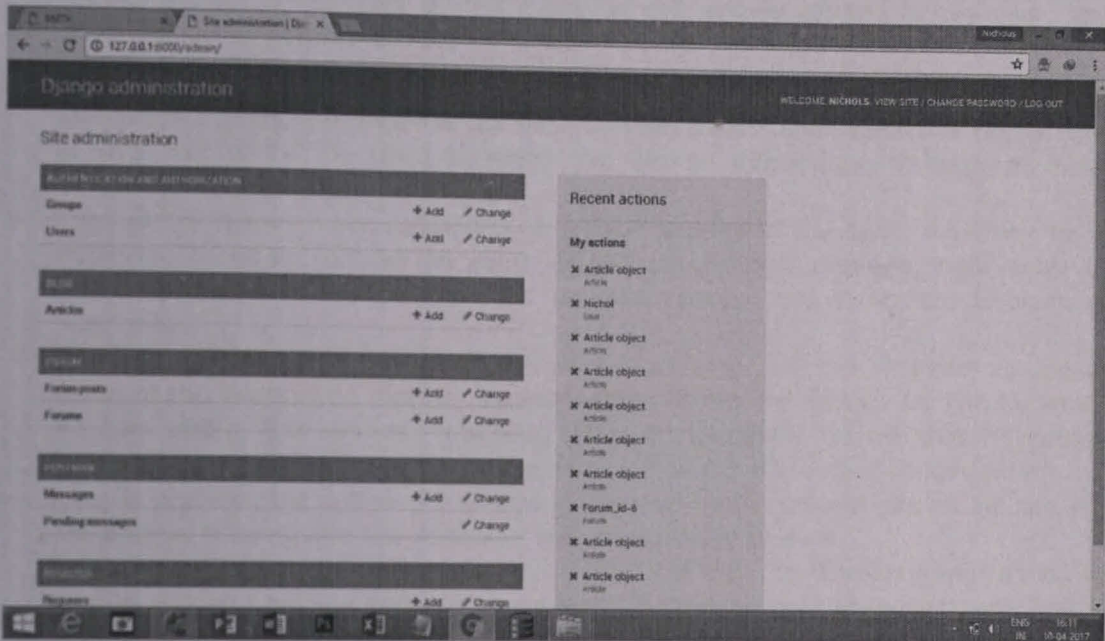
{% endblock %}
```



Admin Login



Admin Home Page



8. Conclusion

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in Python using Bootstrap (includes Html, css and javascripts) web based application and no some extent Windows Application and SQLITE Server, but also about all handling procedure related with "**Back To My Village**". It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

BENEFITS:

The project is identified by the merits of the system offered to the user. The merits of this project are as follows: -

- It's a web-enabled project.
- This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
- The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
- Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.
- User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
- From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is, we can sat that the project is user friendly which is one of the primary concerns of any good project.
- Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
- Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
- Allocating of sample results becomes much faster because at a time the user can see the records of last years.
- Easier and faster data transfer through latest technology associated with the computer and communication.
- Through these features it will increase the efficiency, accuracy and transparency,

LIMITATIONS:

- The size of the database increases day-by-day, increasing the load on the database back up and data maintenance activity.
- Training for simple computer operations is necessary for the users working on the system.



9. BIBLIOGRAPRY

- ✓ For Installation PyCharm 5.0 – JetBrains

<https://www.jetbrains.com/pycharm>

- ✓ SQLite Server

www.tutorialspoints.com

- ✓ Django Framework (Bootstrap)

www.w3school.com

Html, css and Javascripts