**CHAPTER 1**

**INTRODUCTION**

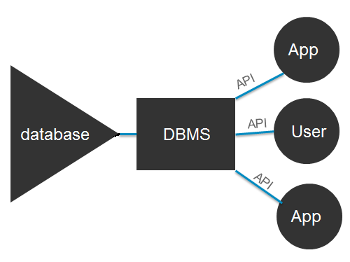
**1.1 Data, Database and DBMS**

Data is a known fact that can be recorded. It can also be defined as an information that has been translated into a form that is efficient for movement or processing. Eg: name, USN, address, crop name, etc. A datum is a unit of data. Meaningful data combines to form information. Hence, information is interpreted data i.e. data provided with semantics

Database is a collection of related data without an implicit meaning. It can also be defined as a structured set of data held in a computer, especially one that is accessible in various ways. Eg. student database, farmer database, employee database, etc.

Database management system (DBMS) is a collection of programs that enable users to create and maintain the database. It is a general purpose software which contains defining, constructing, manipulating, sharing, protecting the database. Basically, a DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, MySQL, Microsoft ACCESS, or EXCEL to store data in the form of database.

Database systems are meant to handle large collection of information. Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.

**1.2 Components of a DBMS**

*Fig.1.1 Components of a DBMS*

Users: Users may be of any kind, such as database administrators, system developers or database users.

Database application: Database application may be Departmental, Personal, Organizational and /or Internal.

DBMS: Software that allows users to create and manipulate database access.

Database: Collection of logical data as a single unit.

**1.3 Relational Data Model**

The relational model (RM) for [database](https://en.wikipedia.org/wiki/Database) management is an approach to managing [data](https://en.wikipedia.org/wiki/Data) using a structure and language consistent with [first-order predicate logic](https://en.wikipedia.org/wiki/First-order_logic), first described in 1969 by English computer scientist [Edgar F. Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd), where all data is represented in terms of [tuples](https://en.wikipedia.org/wiki/Tuple), grouped into [relations](https://en.wikipedia.org/wiki/Relation_(database)). A database organized in terms of the relational model is a [relational database](https://en.wikipedia.org/wiki/Relational_database).

The purpose of the relational model is to provide a [declarative](https://en.wikipedia.org/wiki/Declarative_programming) method for specifying data and queries: users directly state what information the database contains and what information they want from it, and let the database management system software take care of describing data structures for storing the data and retrieval procedures for answering queries.

Most relational databases use the [SQL](https://en.wikipedia.org/wiki/SQL) data definition and query language; these systems implement what can be regarded as an engineering approximation to the relational model. A table in an SQL database schema corresponds to a predicate variable; the contents of a table to a relation; key constraints, other constraints, and SQL queries correspond to predicates. However, SQL databases [deviate from the relational model in many details](https://en.wikipedia.org/wiki/Relational_model#SQL_and_the_relational_model), and Codd fiercely argued against deviations that compromise the original principles.

**1.4 E R Diagram**

The ER diagram shows how the tables in the database are connected to each other and how the control flows from one table to another when some action is triggered by the user. It also shows all the cardinality ratios and participation of the entities for all the relationships.

**1.5 ER to Relational Model Mapping algorithm**

This algorithm consists of 7 steps:

Mapping strong entity to relations:

For each strong entity type ‘E’ create a table or relation R that includes all the simple attributes of E. Since the strong entities will have their own primary keys, those attributes will become the keys for respective table.

Mapping weak entity to relations:

For each weak entity ‘W’ in the ER diagram, with owner entity type ‘E’, create a new table or relation and include all simple attributes of W as attributes of R. In addition, include primary key attribute of the relation that corresponds to owner entity as foreign key of R. The primary key of R is the primary key of owner entity and partial key of weak entity.

Mapping binary 1:1 relationship types:

For each binary 1:1 relationship type ‘R’ in the ER diagram, identify the relations S and T that corresponds to entity type participates in R. There are three approaches:

Foreign key approach: Choose of the relations S (total participation side) and include primary key of T as foreign key of S. Include all the simple descriptive attributes of 1:1 relationship types as the attributes of S.

Merge relationship approach: In this approach, merge the two entity types and relationship types into a single relation. This maybe appropriate when both participations are total.

Cross reference / relationship approach: In this approach, setup a new relation R for the purpose of cross-functioning the primary keys of S and T.

Mapping binary 1:N relationship types:

For each 1:N relationship type R, identify the relation S that represents the participating entity type at the N side of relationship type. Include primary key of S as he foreign key of T and also include all descriptive attributes.

Mapping binary M:N relationship types

For each M:N relationship R, create a new relation S to represent R. Include primary keys of the relations that represent the participating entity types as foreign keys of S and their combination will form the primary key for S and also include all the simple descriptive attributes of R.

Mapping multivalued attributes:

For each multivalued attribute ’A’, create a new relation and include an attribute corresponding to A and primary key ‘k’ of the participating entity or relationship. The primary key of new relation is combination of A and k.

Mapping of n-nary relationship type:

For n-nary relationship type R, n>2, create a new relation to represent R. Include primary keys of all participating entity types as foreign keys and also include all descriptive attributes of R, The primary key of newly created relation is the combination of all the foreign keys.

**1.6 Normalization**

It is a design for producing a set of good tables with all constraints incorporated as per the business constraints. Normalization process takes a relation schema through a series of tests to verify whether it satisfies a certain normal form. Normalization of relations is based on the functional dependencies and primary keys to achieve desirable properties of minimizing redundancy and minimizing update anomaly. Types of normal form are:

First Normal Form (1NF)

A relation schema R is in 1NF if every attribute of R takes only a single value. When a table contains multivalued attribute, we say that its not in first normal form. We identify multivalued attributes and remove using following techniques:

Use multiple tuples one per value

Use multiple columns one per value

Use a separate table

Second Normal Form (2NF)

A relation schema R is in 2NF if it satisfies 1NF and also if all non-prime attributes A in R should be fully functionally dependent on primary key in R. The test for 2NFinvolves testing for functional dependencies whose LHS attribute are part of primary key. If primary key contains a single attribute, the test need not be applied. If a relation schema is not in 2NF, it can be second normalized into a number of 2NF relations in which non-prime attributes are associated only with the part of the primary key on which they are fully functionally dependent.

Third Normal Form (3NF)

A relation schema R is in 3NF if it is in 2NF and no non-prime attributes of R is transitively dependent on primary key of R. A functional dependency X->Y is a transitive dependency if there’s a set of attributes Z that is not a subset of any key of R and both X->Z and Z->Y holds.

**1.7 OVERALL PROJECT DESCRIPTION**

* **Product perspective:**

The project **social networking site (.CONNECT)** is aimed towards storing the records of considerable users and needs assistance for managing records regarding the users. Project should be user-friendly and a reliable web site for above purpose.

**. CONNECT** is intended to be stand alone product and should not depend on the availability of other applications. The system will have an administrator who has full rights to perform all actions related to control and management of the project.

* **Product functions:**

There are two types of person who will be using this product, one is the admin who has access to the entire database of the product and the second one is the normal user who registers itself to this virtual world.

The features that are available for any user are:

**Register or sign up:** If the user is new to the application then before accessing any of the features he must sign up where he must provide his personal details containing a unique username.

**Login:** If the user has already signed up then he can directly go to login page, where he must produce his unique username and password to access his account.

**Upload:** User can share his or her photos or videos by uploading them on the site. He or she can also upload comments on the photos of other users who are connected to him or her.

**1.8 Triggers**

The MySQL trigger is a database object that is associated with a table. It will be activated when a defined action is executed for the table. The trigger can be executed when you run one of the following MySQL statements on the table: ***INSERT, UPDATE*** and ***DELETE*** and it can be invoked before or after the event.

The main requirement for running such MySQL Triggers is having MySQL ***SUPERUSER*** privileges.

**1.9 Stored Procedures**

A stored procedure is a subroutine like a subprogram in other computing languages, stored in database. A procedure has a name, a parameter list, and SQL statement(s). Almost all relational database system supports stored procedures.

The Stored Procedure are invoked using a ***CALL*** statement.

**1.10 APPLICATION**

Nowadays the requirement of advertising any product is very much necessary. Any user who wants to sell his product or there can be multinational companies who want to promote its product then this is the best platform and the best part is that through comments the other user can give reviews for the product.

.CONNECT also allows the user to have a to-do list in his account, user can maintain a list of tasks which he is supposed to do and later can work on them.

**CHAPTER 2**

**LITERATURE SURVEY**

Section mentions some of the project of the same domain. There are many sited which works on same domain and which have motivated us to work on our own social networking project.

Some of the projects studied are:

**2.1 Facebook.com (**http://www.facebook.com)

This is the most popular work produced by MARK ZUCKERBERG which mainly increased the craze of social networking in the world. User can upload pics, can message a friend and mainly he connect itself to entire world by making friends in it.

**2.2 Watsapp (** http://watsappweb.com)

This project was mainly developed for chatting and it gained user’s attention very soon and soon it became trending chatting app around the globe. Users can even make chatting groups in it to talk together. Later it introduced the concept of putting status in it where user can upload pics which generally user shares to show the world about his/her current situation.

**2.3 Instagram (**http://instagram.com)

This site was created mainly to upload pics and status, the concept of stories or status was first introduced in INSTAGRAM, later it also introduced chatting facility and there create friends concept in this but instead of that user can follow or unfollow the other users.

**2.4 Twitter (**<http://twitter.com>)

This website is basically used by the people to just write some stuffs and post it. This site was basically created to post the status, news or any other such thing. A lot of celebrities are found active on this site

**CHAPTER 3**

**DESIGN PHASE**

**3.1 Scope and purpose of Design**

The purpose of the design phase is to develop a clear understanding of what the developer wants people to gain from his/her project. As you the developer work on the project, the test for every design decision should be “Does this feature fulfil the ultimate purpose of the project?”

The purpose statement basically tells that what developer wants the project to do rather than describing the project itself. The design document will verify that the current design meets all the explicit requirements contained in the system model as well as the implicit requirements desired by the user.

* **Overall System Design Objectives**

The overall system design objective is to provide an efficient, modular design that will reduce the system’s complexity, facilitate change and result in easy implementation. In addition, this document will provide interface design models that are consistent user friendly and will provide straight forward transition through the various system functions.

**3.3 Structure of Design Document**

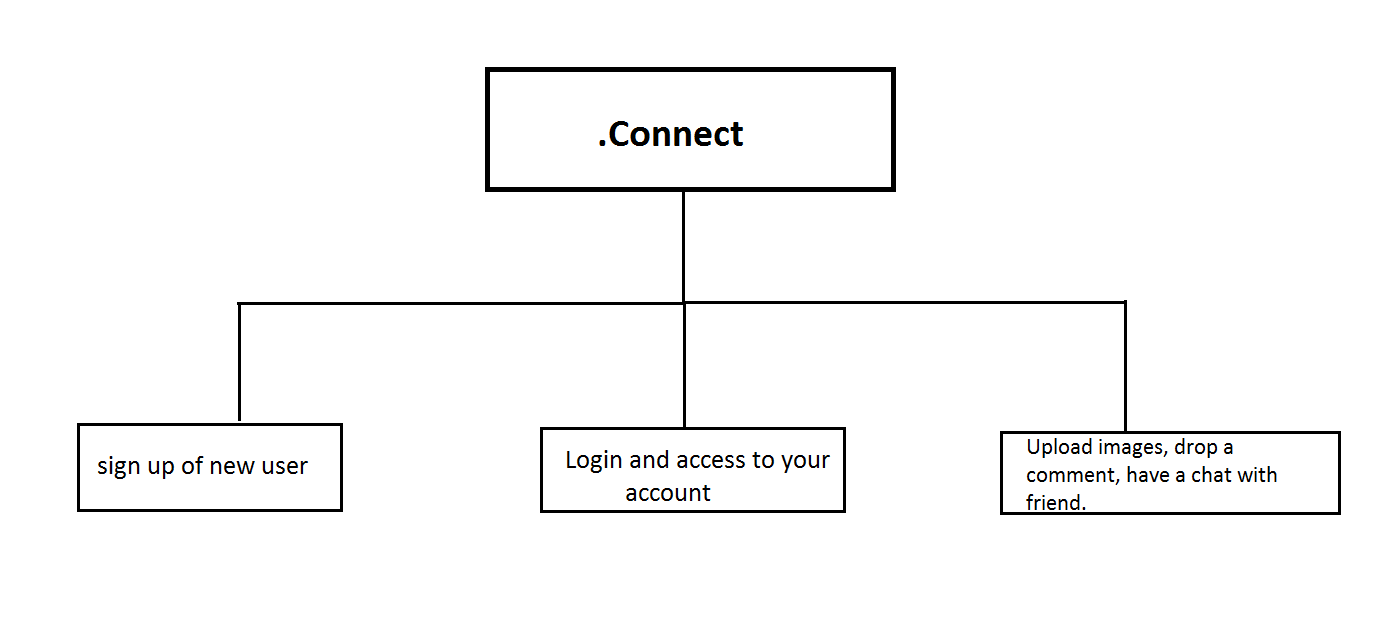
System architecture design contains mainly:

* The detailed diagram of the system, server and client.
* The relation schema which shows all information about the attributes stored in database.
* The entity relation (ER) diagram which shows all the cardinality ratios and participation of the entities.

**3.3.1System architecture design**

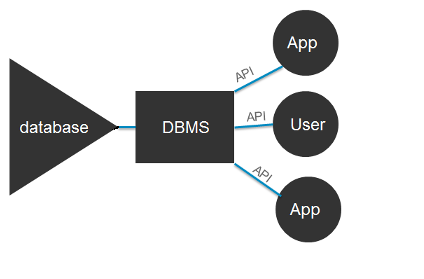
The **.connect** is a system which contain major part of DBMS which include: user details, triggers and procedures.

The user selects one of the available options as an input to the system. According to the input by the user, the system acts and rest of the functions are performed accordingly. The admin can operate on any user’s details, but cannot edit the username of the entered user in any table, since it is acting as the key constraint and as the referential key constraint.



*Fig 3.1: Architecture Diagram*

Fig 2.1 shows the basic structure of the project, what all the project provides and what all user has to do to access the project to its full.

**

*Fig 3.2: DBMS Architecture*

Fig 3.2 shows the architecture which is mainly followed in database connections, retrieval of data and how to APIs are used.

**3.3 Entities and Attributes**

Following are the list of entities and list of attributes used in the databse.

**3.3.1 List of Entities with description**

|  |  |
| --- | --- |
| **Register** | Used to store the initial information of any new user. |
| **Profile** | Used to store the complete information of the user. |
| **Todo** | Used to store the details regarding to-do list of the user. |
| **Feedback** | Used to store the information about the last feedback given by user. |
| **Friends** | Used to store the name of all friends with respect to a username. |
| **Uploads** | Used to store the details of any uploads done by the user. |
| **Comments** | used to store the all comments posted on a particular upload. |

*Table: 3.1*

**3.3.2 Entities with attribute’s description**

* **Register**

|  |  |  |
| --- | --- | --- |
| Email | Stores the email id of user | Varchar (30) |
| Username | Stores the unique username of the user | varchar (20) |
| Password | Stores the password corresponding to a username | Varchar (20) |

*Table: 3.2*

* **Profiles**

|  |  |  |
| --- | --- | --- |
| **Username** | Stores the unique username of the user | Varchar (20) |
| **Fullname** | Stores the full name of the user (foreign key). | Varchar (40) |
| **Profilepic** | Stores the url of the profile pic of user. | Varchar (300) |
| **Phone** | Stores the phone number of users. | Bigint (20) |
| **Address** | Stores the address of the user | Varchar (20) |
| **Gender** | Stores the gender of user. | Varchar (10) |
| **Placeborn** | Stores the birth place of user. | Varchar (20) |
| **Dateborn** | Stores the date of birth of user. | Date |
| **Job** | Stores the details of job of the user. | Varchar (20) |
| **Institution** | Stores the institution where the user studies or works. | Varchar (20) |

*Table: 3.3*

* **Todo**

|  |  |  |
| --- | --- | --- |
| **Id** | Stores the unique id of the to-do list. | Int (11) |
| **Username** | Stores the unique username of the user (foreign key). | Varchar (20) |
| **Message** | Stores the message of the to-do list. | Varchar (300) |

* **Feedback**

|  |  |  |
| --- | --- | --- |
| **Username** | Stores the unique username of the user (foreign key). | Varchar (30) |
| **Id** | Stores the unique id of all the feedback messages. | Int (11) |
| **feedback** | Stores the latest message given by the user as feedback. | Varchar (300) |
| **curdate** | Stores the time date and time when the feedback was given. | timestamp |

*Table 3.5*

* **Friends**

|  |  |  |
| --- | --- | --- |
| **Username** | Stores the unique username of the user (foreign key). | Varchar (20) |
| **friendid** | stores the unique id for each combination  (primary key) | int (11) |
| **Added** | Stores the unique username of the friend with respect to a username (foreign key). | Varchar (20) |

*Table: 3.6*

* **Uploads**

|  |  |  |
| --- | --- | --- |
| **Uploadid** | Stores the unique upload id of all the uploads. | Int (11) |
| **Username** | Stores the unique username of the user (foreign key). | Varchar (20) |
| **Title** | Stores the title of the pic being uploaded. | Varchar (30) |
| **Url** | Stores the url of the pic which was uploaded. | Varchar (300) |
| **Curtime** | Stores the time at which it was uploaded. | timestamp |

*Table:3.7*

* **Comments**

|  |  |  |
| --- | --- | --- |
| **Commentid** | Stores the unique id for all the comments. | Int (11) |
| **Username** | Stores the unique username of the user (foreign key). | Varchar (20) |
| **Message** | Stores the actual comment which is uploaded. | Varchar (300) |
| **Commenttime** | Stores the time at which comment was uploaded. | Timestamp |
| **Profilepic** | Stores the url of the profile pic of the username (foreign key). | Varchar (300) |
| **Uploadid** | Stores the uploadid of the pic on which comment was uploaded. | Int (11) |

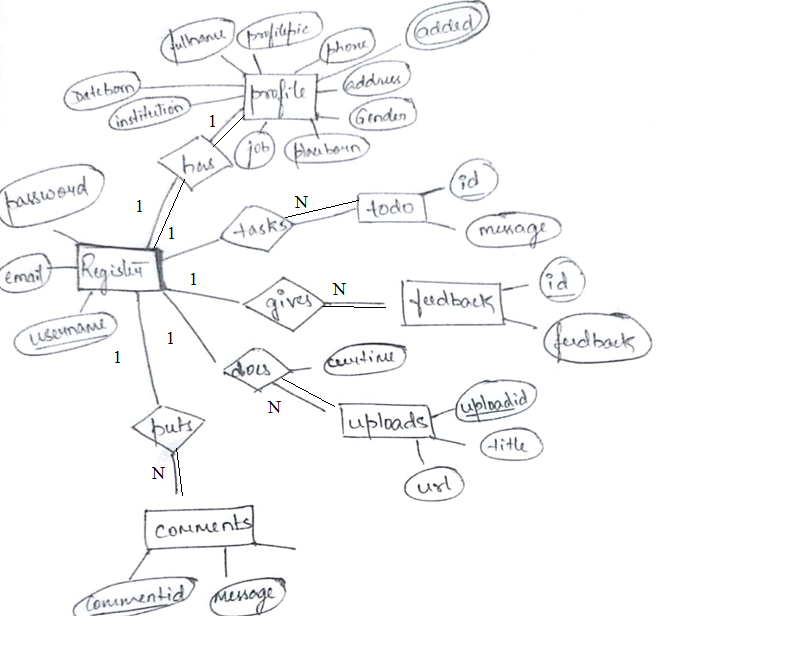
*Table: 3.8*

**3.4 Constraints on Relationships**

|  |  |  |
| --- | --- | --- |
| **Relationship** | **Cardinality Ratio** | **Participation** |
| Has (register : profile) | 1:1 | Total on both sides |
| Tasks (register : todo) | 1:N | Partial – register  Total – todo |
| Gives (register : feedback) | 1:N | Partial – register  Total – feedback |
| Does (register : uploads) | 1:N | Partial – register  Total – uploads |
| Puts (register : comments) | 1:N | Partial – register  Total – comments |
| Has (uploads : comments) | 1:N | Partial – uploads  Total - comments |

*Table: 3.9*

**3.5 The Entity Relation Diagram:** The ER diagram shows how the tables in the database are connected to each other and how the control flows from one table to another when some action is triggered by the user. It also shows all the cardinality ratios and participation of the entities for all the relationships.

*Fig: 3.3*

**3.6 The Relation Schema:** The schema diagram gives us the information about the attributes in the table of the database and how the given tables are related to each other.

Register

|  |  |  |
| --- | --- | --- |
| Email | Username | Password |

Profile

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| username | fullname | profilepic | Phone | Address | Gender | Placeborn | Dateborn | Job | institution |

Todo

|  |  |  |
| --- | --- | --- |
| id | Username | Message |

Feedback

|  |  |  |
| --- | --- | --- |
| Username | id | Feedback |

Friends

|  |  |
| --- | --- |
| Username | added |

Uploads

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Uploadid | Username | Title | url | Curtime |

Comments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| commentid | Username | Message | Commenttime | profilepic | Uploadid |

**3.7 Normalised Table:** Database normalisation is the process of organizing data into tables in such a way that the results of using the database are always unambiguous and as intended. Such normalisation is intrinsic to relational theory. It may have the effect of duplicating data within the database and often results in creation of additional tables

Register

|  |  |  |
| --- | --- | --- |
| email | Username | Password |

Profile

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| username | fullname | profilepic | Phone | Address | Gender | Placeborn | Dateborn | Job | institution |

Todo

|  |  |  |
| --- | --- | --- |
| id | Username | Message |

Feedback

|  |  |  |
| --- | --- | --- |
| Username | id | Feedback |

Friends

|  |  |
| --- | --- |
| Username | added |

Uploads

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| uploadid | Username | Title | url | Curtime |

Comments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| commentid | Username | Message | Commenttime | profilepic | Uploadid |

**CHAPTER 4**

**REQUIREMENTS**

**4.1 Technologies/Frameworks used in building the project**

* **DBMS (MySQL)**

MySQL database management system is used to store the user and product information which is used in the secondary storage device and can be altered anytime. Normalised and efficient schema is used to avoid redundancy and inconsistency. The data is uploaded in real time.

* **Node.js**

Node.js is used to implement the backend logic of the project, and runs on the server to process the requests from users. It interacts with the database to update the data ad process the error caught.

* **HTML**

HTML is used to serve the content to the user. It’s a mark up language

Used in the project to assign the arrangement and placement of content delivered to the user.

* **CSS**

CSS is used to beautify the contents and pages delivered to the user. It improves the UI and of the project.

* **Bootstrap**

Bootstrap is a CSS framework used in the project to beautify the web pages.

**4.3 Hardware Requirements**

* CPU with 512MB RAM and above
* Pentium processor

**4.2 Software Requirement Specification**

* **Server Side**

1. MySQL database Management System to store the data.
2. FTP accessibility to upload the content to the server or modify them.
3. Server running on Windows/Linux/Macintosh

* **Client Side**

1. Browser supporting HTML 5 and CSS 3.
2. 1336 x 768 screen size (recommended) for best experience.
3. Machine running Windows/Linux/Macintosh

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 MySQL**

It is known as s**tructured query language** basically carrying ouot operations on database. MySQL is an open source software provided where you can create your own database and then create all the relations which is needed in the database as shown in section 5.1.1 and can carry out all the operations like *INSERT, DELETE, UPDATE* etc. commands.

**5.1.1 CREATE table commands**

* **REGISTER**

CREATE TABLE register

(

email VARCHAR(30),

username VARCHAR(20) PRIMARY KEY,

password VARCHAR(20) NOT NULL

);

* **TODO**

CREATE TABLE todo(

id INT NOT NULL auto\_increment,

username VARCHAR(20) NOT NULL,

message VARCHAR(300),

FOREIGN KEY(username) REFERENCES register1(username) ON DELETE CASCADE,

PRIMARY KEY(id)

);

* **PROFILES**

CREATE TABLE profiles

(

username VARCHAR(20) NOT NULL,

fullname VARCHAR(40) NOT NULL,

profilepic VARCHAR(300) DEFAULT "/img/new-user-male-icon.jpg",

phone INT(12),

address VARCHAR(20) NOT NULL,

gender VARCHAR(10) NOT NULL,

placeborn VARCHAR(20),

dateborn DATE NOT NULL,

job VARCHAR(20),

institution VARCHAR(20),

FOREIGN KEY(username) REFERENCES register1(username) ON DELETE CASCADE,

PRIMARY KEY(username)

);

* **FEEDBACK**

CREATE TABLE feedback (

username VARCHAR(30) NOT NULL,

id INT NOT NULL auto\_increment,

feedback VARCHAR(300) NOT NULL,

FOREIGN KEY(username) REFERENCES register1(username) ON DELETE CASCADE,

PRIMARY KEY(id)

);

* **UPLOADS**

CREATE TABLE uploads

(

uploadid INT auto\_increment,

username VARCHAR(20) REFERENCES register1(username) ON DELETE CASCADE,

title VARCHAR(30),

url VARCHAR(300) NOT NULL,

curtime TIMESTAMP,

PRIMARY KEY(uploadid)

);

* **COMMENTS**

CREATE TABLE comments

(

commentid INT auto\_increment,

username VARCHAR(20) NOT NULL,

message VARCHAR(100),

uploadid INT auto\_increment,

curtime TIMESTAMP,

profilepic VARCHAR(300),

FOREIGN KEY (profilepic) REFERENCES profiles(profilepic),

FOREIGN KEY(username) REFERENCES register1(username) ON DELETE CASCADE,

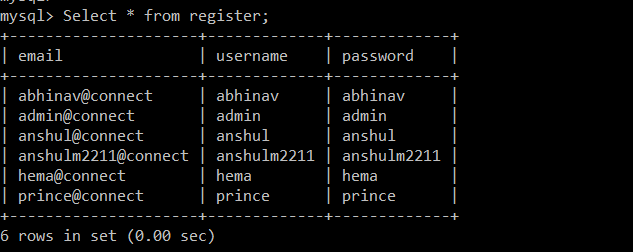
FOREIGN KEY (uploadid) REFERENCES uploads(uploadid) ON DELETE CASCADE,

PRIMARY KEY(commentid)

);

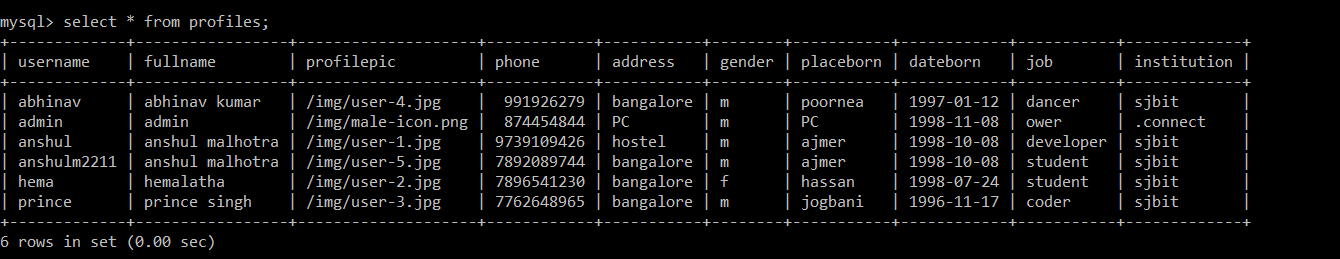
**5.2 TABLE CONTENTS**

* Register



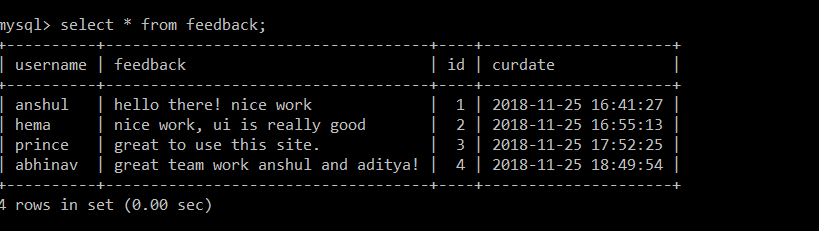
*Table:3.1*

* Profiles



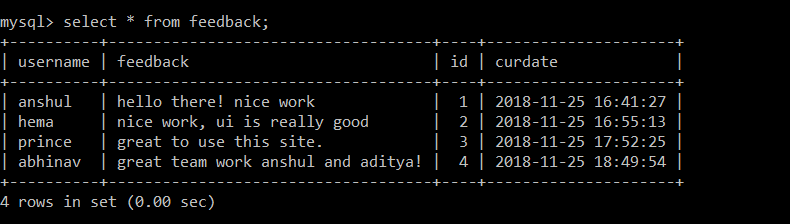
*Table:5.2*

* Feedback



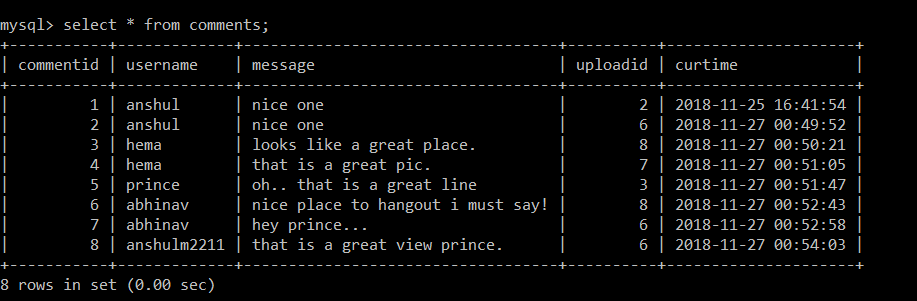
*Table:5.3*

* Uploads



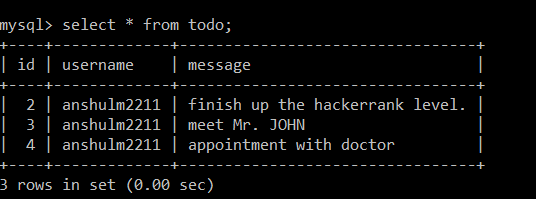
*Table:5.4*

* Comments



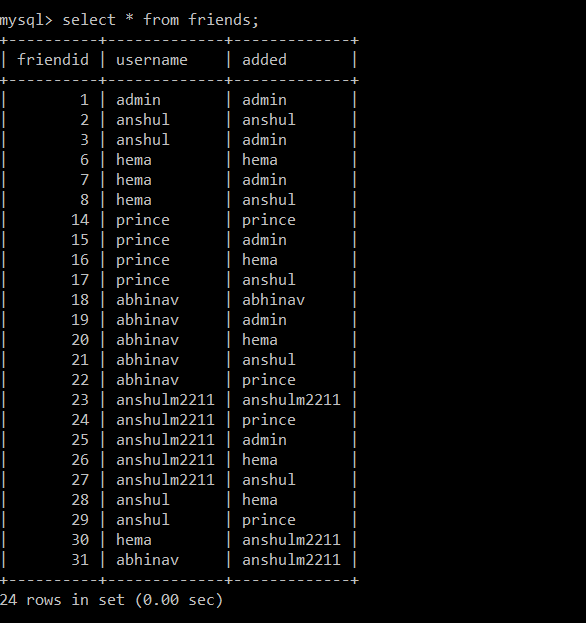
*Table:5.5*

* Todo



*Table:5.6*

* Friends



*Table:5.7*

* 1. **BACKEND CONNECTION**

**5.3.1 Node.js**

It  is an [open-source](https://en.wikipedia.org/wiki/Open-source_software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [JavaScript](https://en.wikipedia.org/wiki/JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system) that executes JavaScript code outside of a browser. Typically, JavaScript is used primarily for [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting), in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser.

**5.3.2 Express.js framework**

Express.js, or simply Express, is a [web application framework](https://en.wikipedia.org/wiki/Web_application_framework) for [Node.js](https://en.wikipedia.org/wiki/Node.js), released as [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software) under the [MIT License](https://en.wikipedia.org/wiki/MIT_License). It is designed for building [web applications](https://en.wikipedia.org/wiki/Web_application) and [APIs](https://en.wikipedia.org/wiki/API).[[3]](https://en.wikipedia.org/wiki/Express.js#cite_note-ExpressJS-3) It has been called the de facto standard server framework for Node.js.

**5.3.3 CONNECTION CODE**

var mysql = require('mysql');

var con = mysql.createConnection({

host: "localhost",

user: "root",

password: "1234",

database: "db"

});

con.connect(function(err) {

if (err) throw err;

console.log("Connected!");

});

* 1. **HTML & CSS**

HTML is used to create the user interface to display the project. CSS is used to providing styling to project like REGISTER page, LOGIN page, HOME page and many others.

* + 1. **REGISTER page**

The first page of the project where all the introduction to the project is given.

* + 1. **LOGIN page**

The login page is where user enters the username and password to get into the home page.

* + 1. **HOME page**

This is the actual virtual world for the user, here user has all links to access all the resources like uploads, follow, profile edit and otherwise logout.

**CHAPTER 6**

**TESTING**

Testing is the process used to help identify correctness, completeness, security and quality of developed software. This includes executing a program with the intent of finding errors. It is important to distinguish between faults and failures. Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors. It can be conducted as soon as executable software (even if partially complete) exists. Most testing occurs after system requirements have been defined and then implemented in testable programs.

**6.1** **TESTING PROCESS**

Testing is an integral part of software development. Testing process certifies whether the product that is developed compiles with the standards that it was designed to. Testing process involves building of test cases against which the product has to be tested.

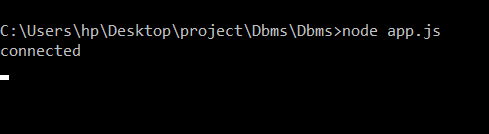
**6.2** **TESTING OBJECTIVES**

* The main objectives of testing process are as follows.
* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has high probability of finding undiscovered error.
* A successful test is one that uncovers the undiscovered error.

**CHAPTER 7**

**SNAPSHOTS**

**7.1 DATABASE connection**

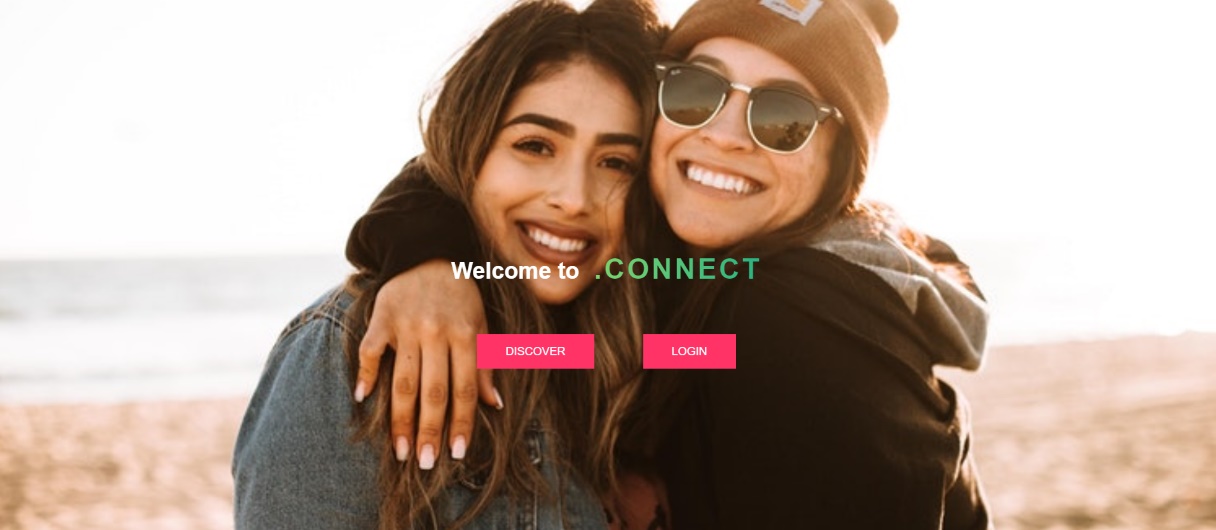
****

*Fig:7.1*

Code required to connect to database is mentioned in last chapter. The command is written is command prompt to execute the .js file where the actual code is written.

The line “connected” shows that there is no error in the code and it is executing correctly and the database is connected to the code.

**7.2 WELCOME PAGE**

****

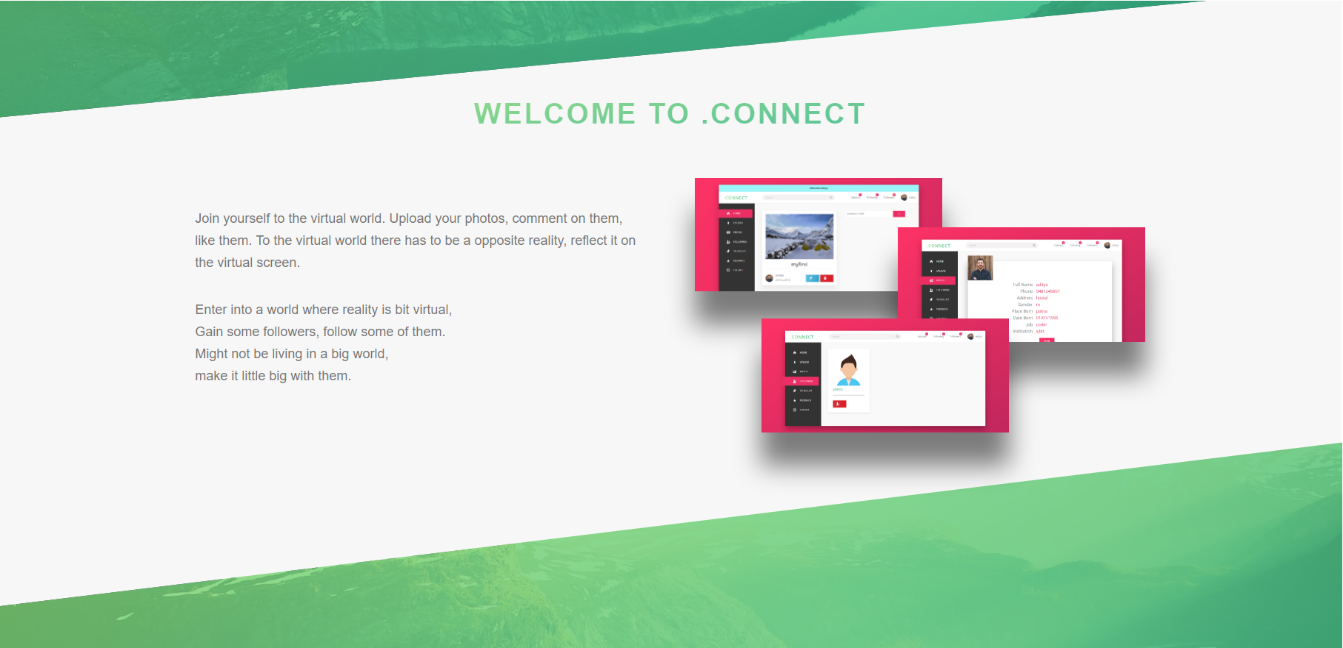
*Fig: 7.2*

This is the first page of project for which the localhost 3000 is created. There are two options for the user to proceed forward, if the user is new to the site then he can click on “discover” and can register himself or herself. If the user has already registered then he can directly click on “login” and can login to access his/her account.

**7.3 NEW USER**

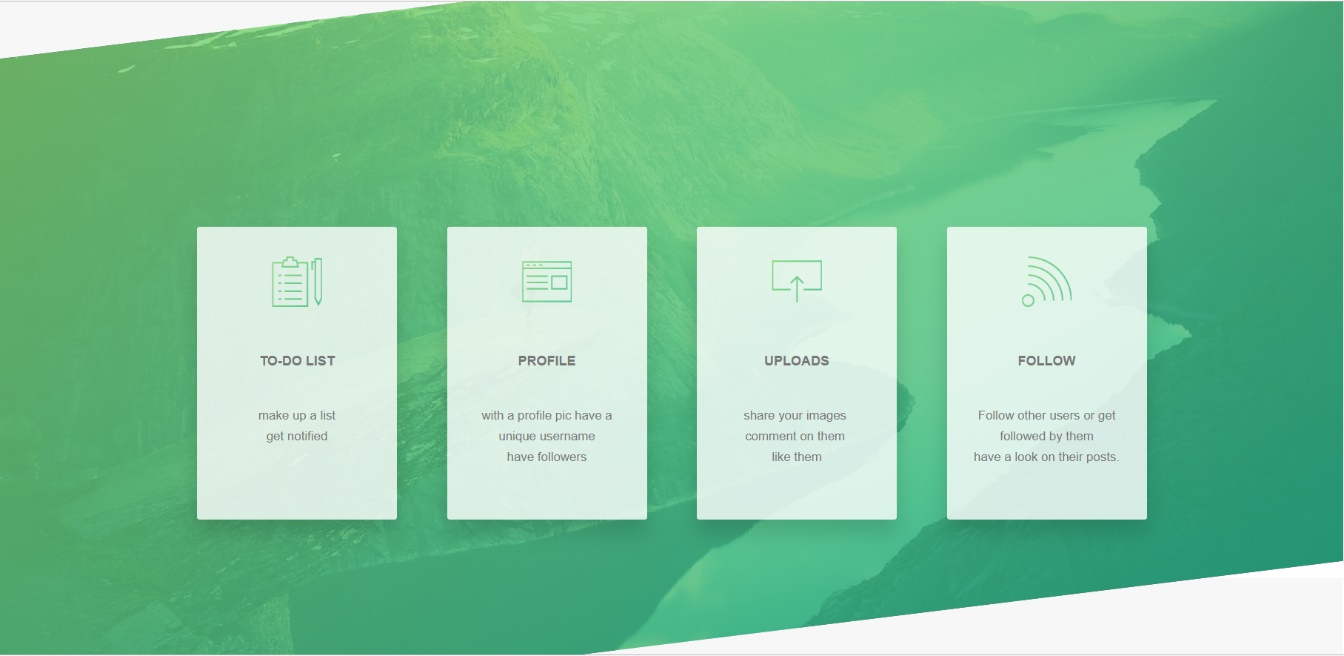
****

*Fig: 7.3*

****

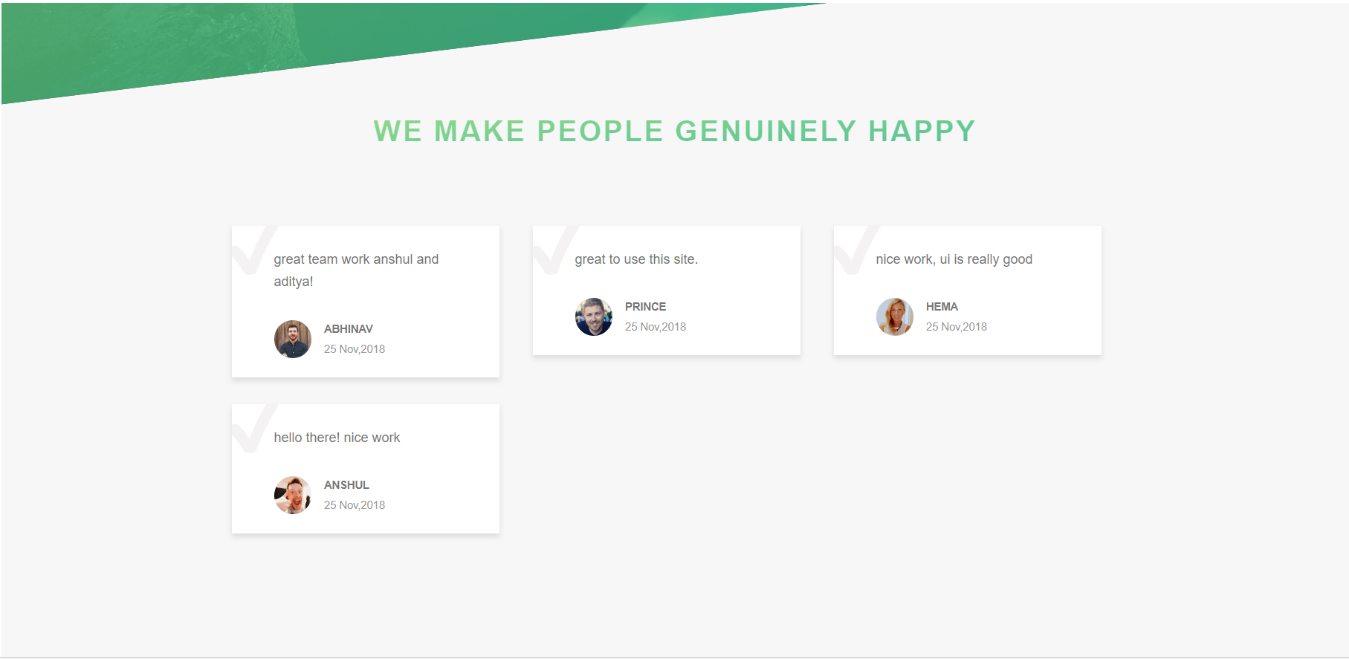
*Fig: 7.4*

Fig: 7.3 and 7.4 shows the introduction about the project and some screenshots of the home page is displayed with the description next to it.

****

*Fig: 7.5*

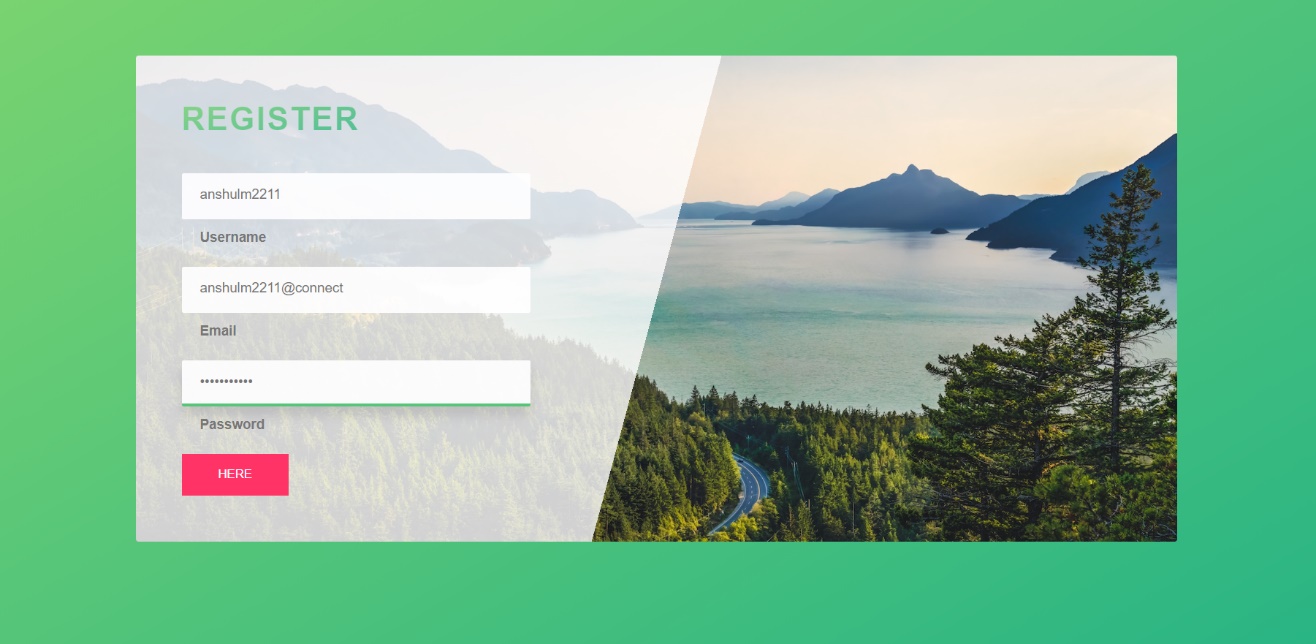
Shows some of the key features of the project.

****

*Fig: 7.6*

Fig 7.5 and 7.6 provides the introduction and information about the site which would attract the user to make an account and access the virtual world and it shows some of the feedbacks given by the users which are retrieved from the database.

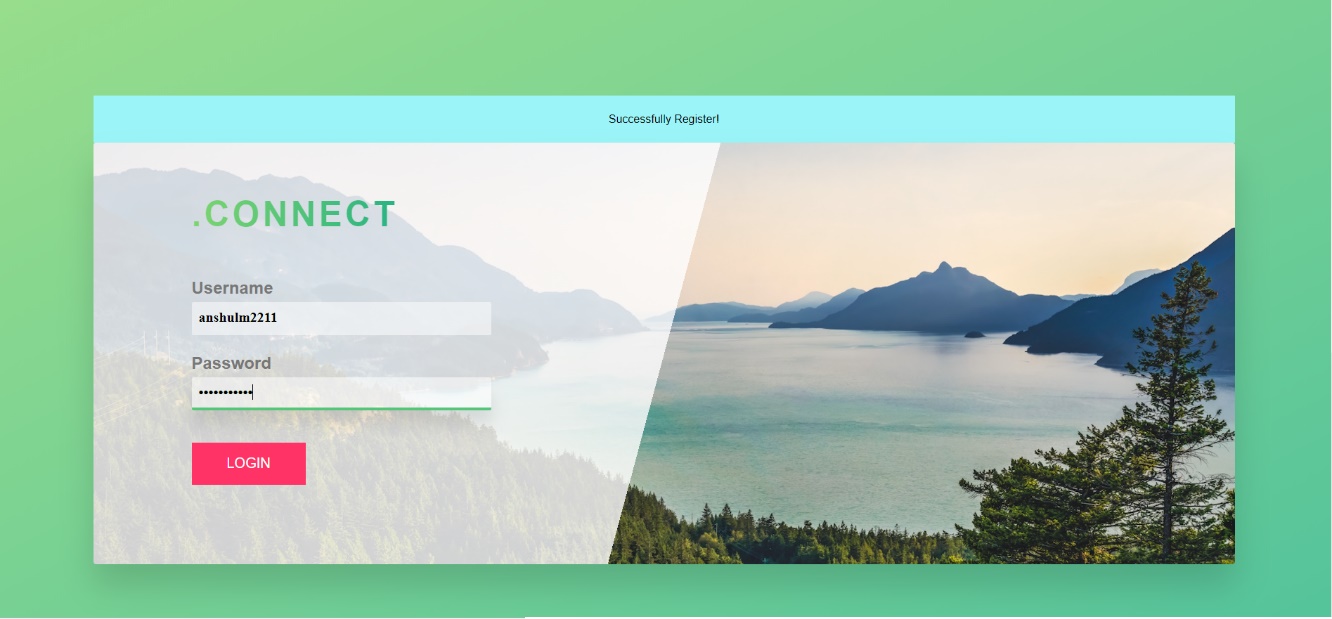
**7.4 REGISTER**

****

*Fig: 7.7*

The fig: 4.6 shows the first step to enter into the virtual world. Here user needs to provide username, email id and a password. Remember that the username has to be unique. After providing these three details, the user can click on “here” and this will redirect the user to login page.

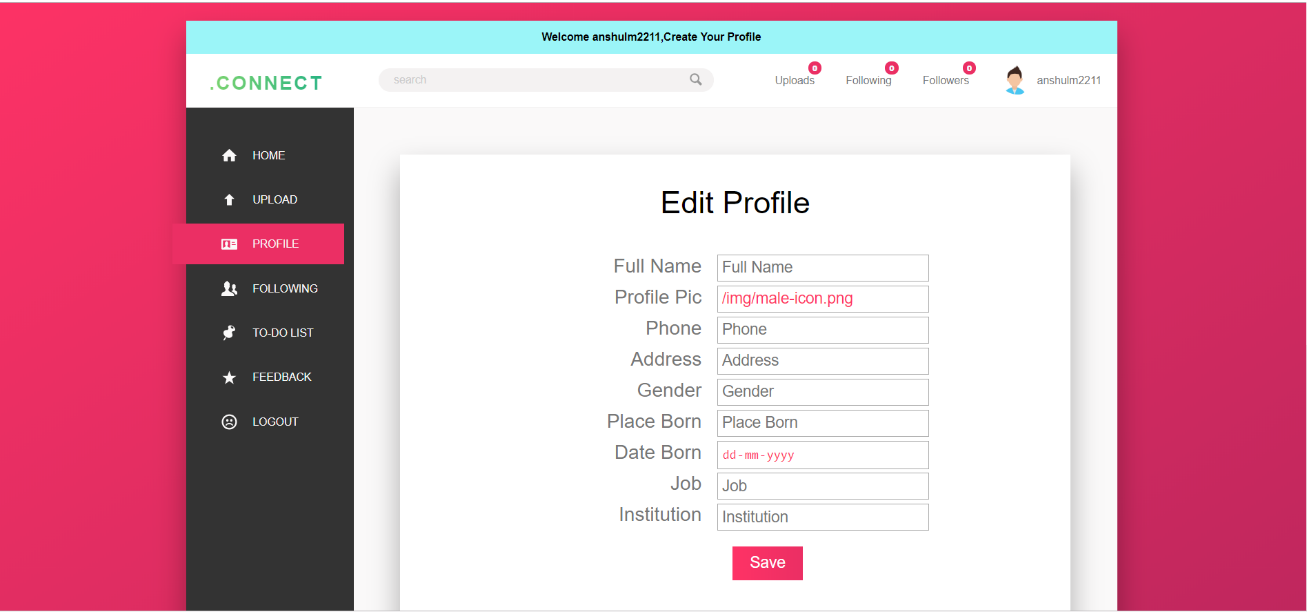
**4.5 LOGIN**

****

*Fig: 7.8*

Here user needs to provide his/her unique username and password. Once the combination matches in the database the user gets logged into his/her account. If the user is logging in for the first time then he needs to update his profile first.

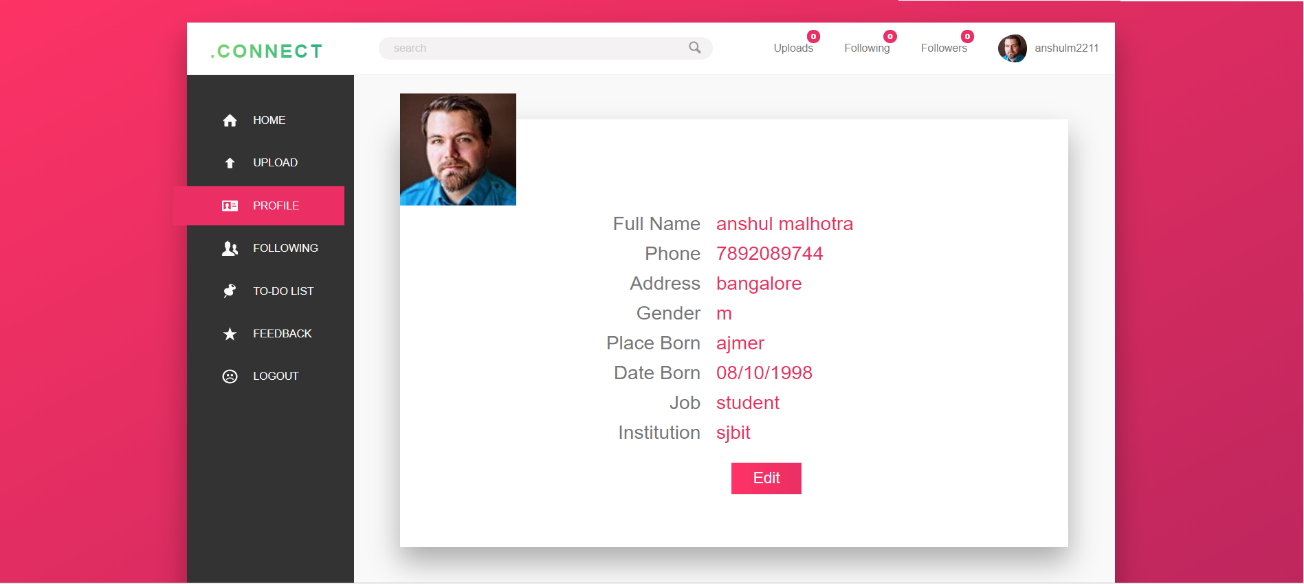
**7.6 PROFILE**

****

*Fig: 7.9*

User needs to enter the details which are required and yeah, all the details are mandatory.

Once the user has entered all the correct details, the page looks like,



*Fig: 7.10*

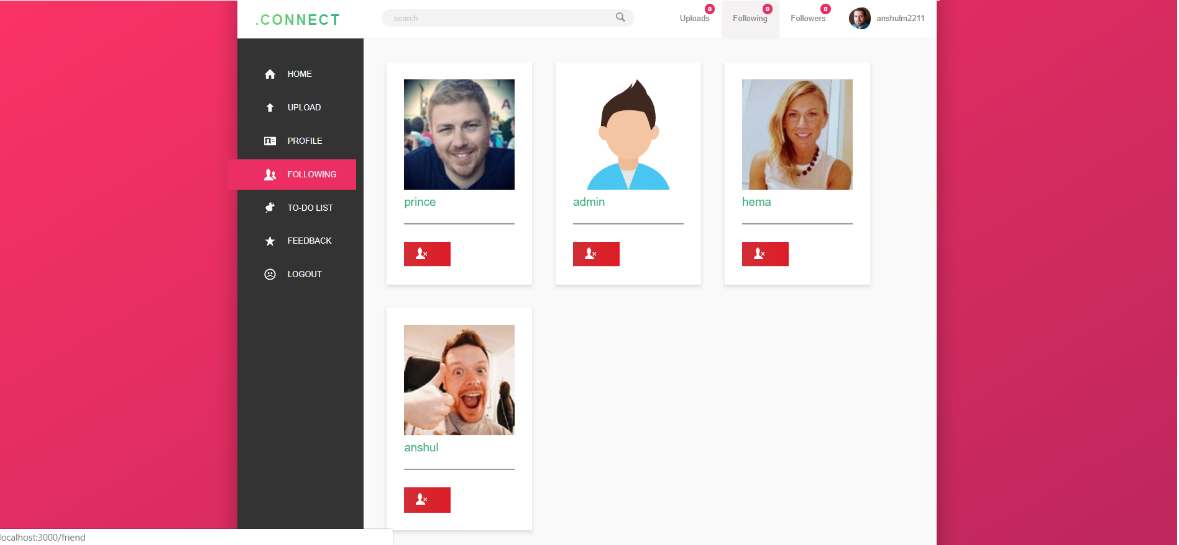
Once user’s profile is updated, user can access all the resources of the site.User can follow other users, can upload pictures, can comment on pictures.

**7.7 FOLLOW**

****

*Fig: 7.11*

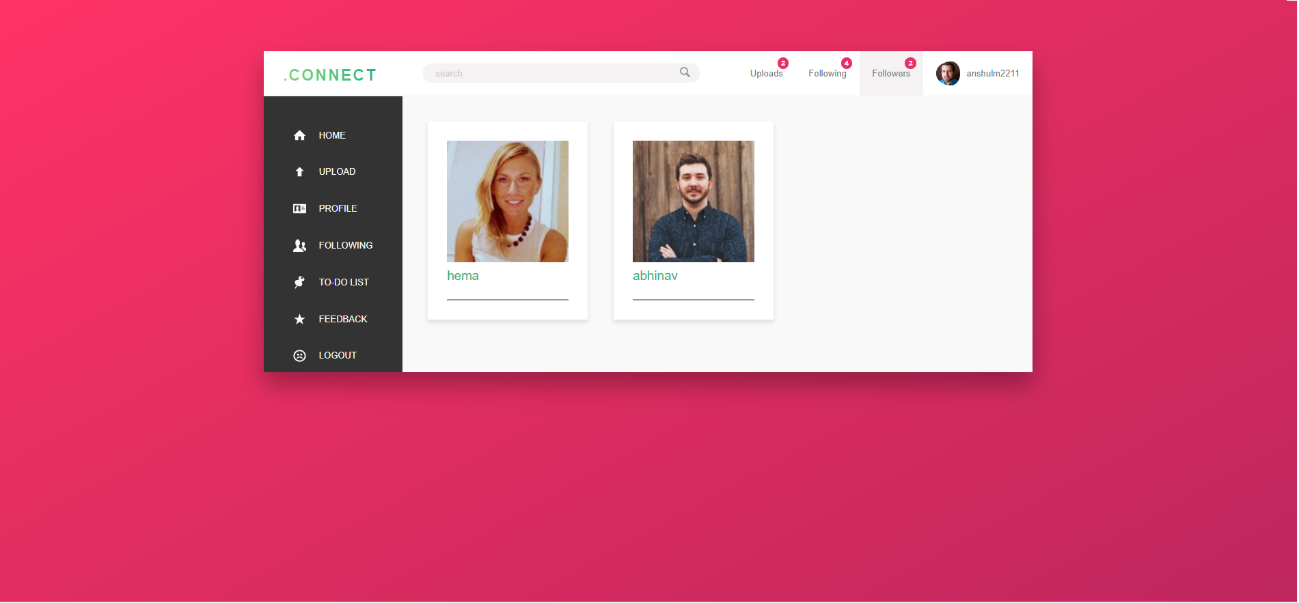
User can simply search a username in the search box and if the username exists then his name and profile pic will be shown, if user wants to follow him/her then he/she can simply click on “+” icon below and that person will automatically get added into the following list of the user.



*Fig: 7.12*

The fig: 4.11 shows some of the following usernames of the current user. If he/she wants to unfollow then he/she can click on the icon below the pic of the other username.

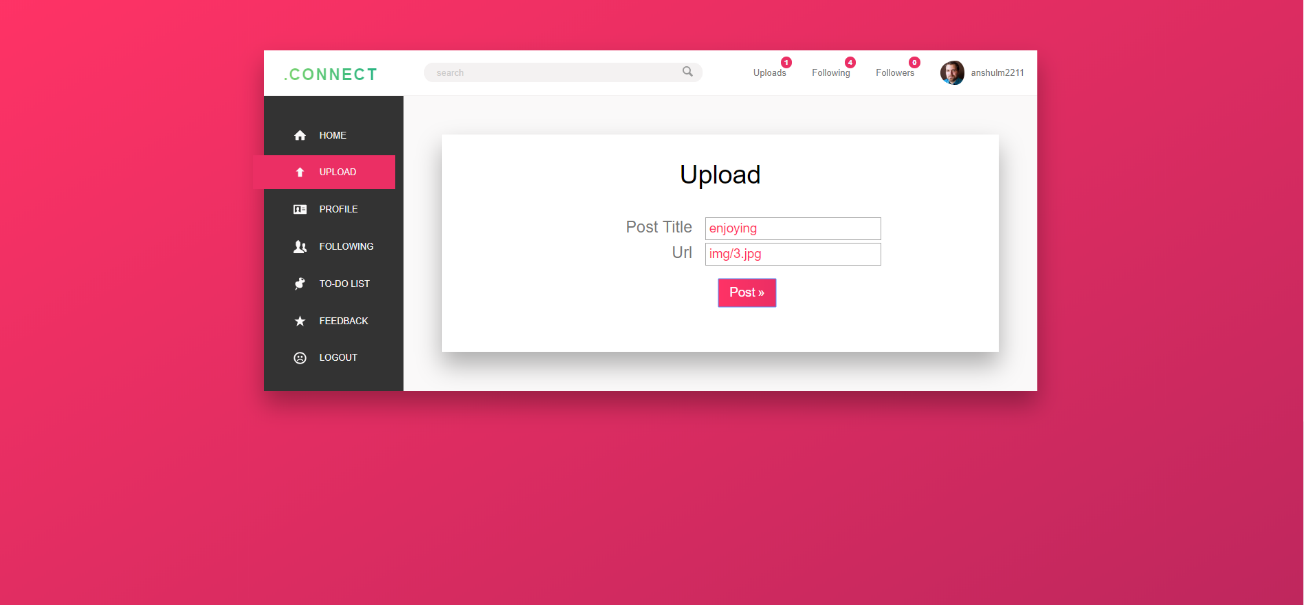
**7.8 FOLLOWERS**



*Fig: 7,13*

The fig: 4.12 shows the list of followers of the user.

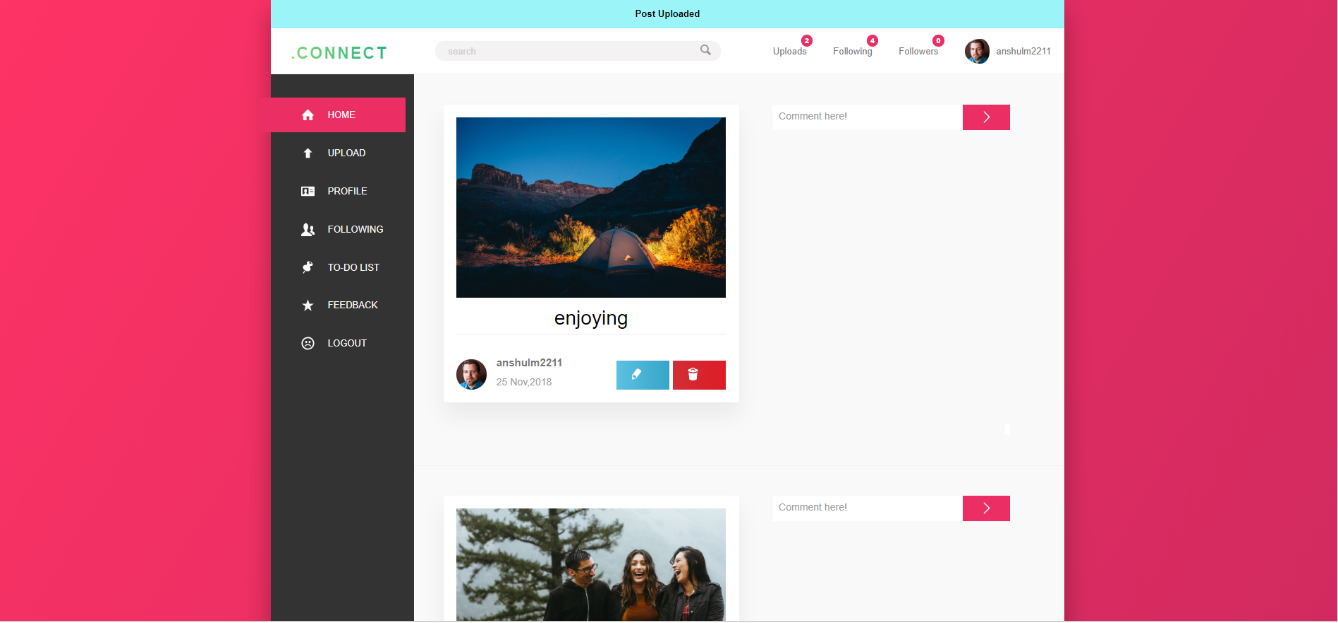
**7.9 UPLOADS**

****

*Fig: 7.14*

If a user wants to upload any image in .CONNECT then user can simply click on “upload” tab in the side bar, and then user can enter the “title” and “url” of the image which he/she wants to upload.

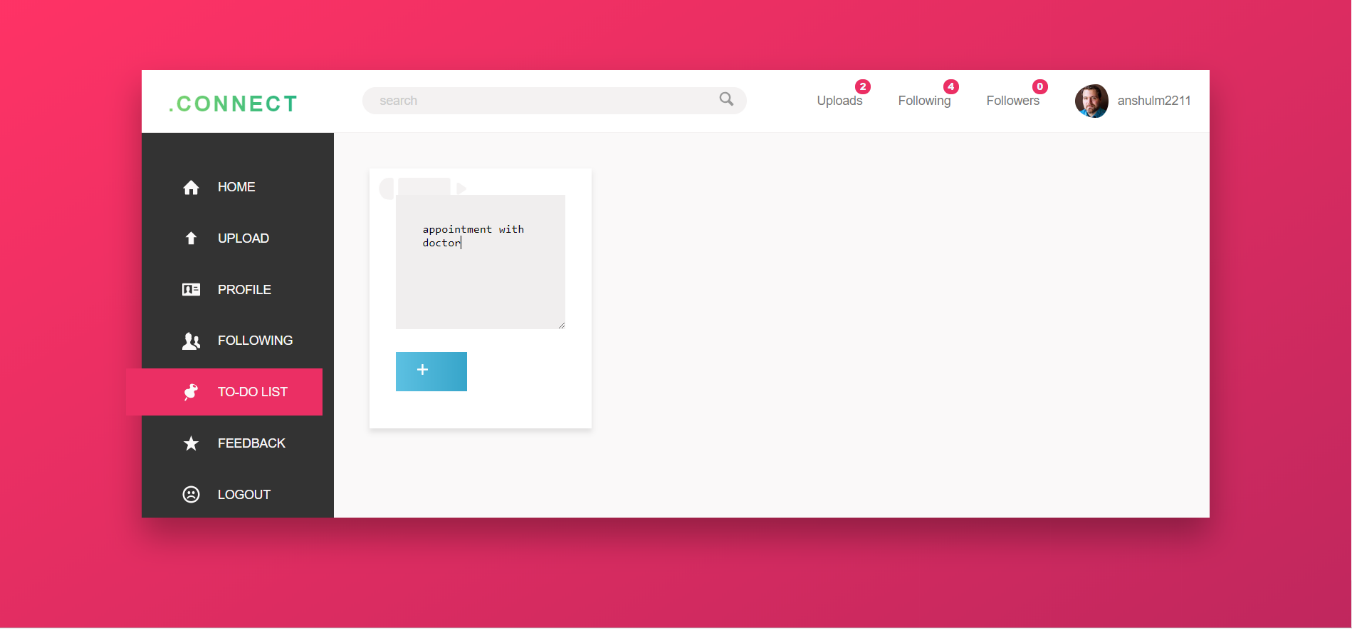
Once the user clicks the “post” then the next page opens which is shown in above snap where user’s post is shown in the home page.



*Fig: 7.15*

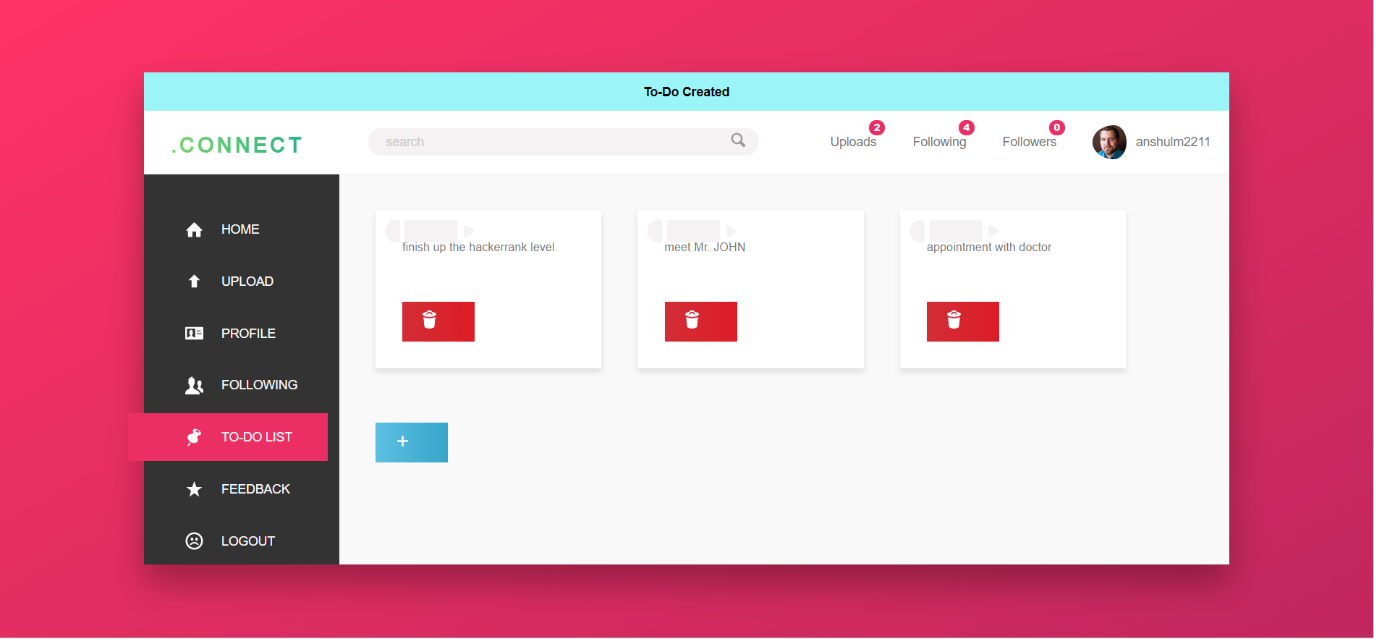
This home page is the place where all the uploads of the user, users whom current user follows are displayed in the order of their uploads.

**7.10 TO-DO LIST**

****

*Fig: 7.16*

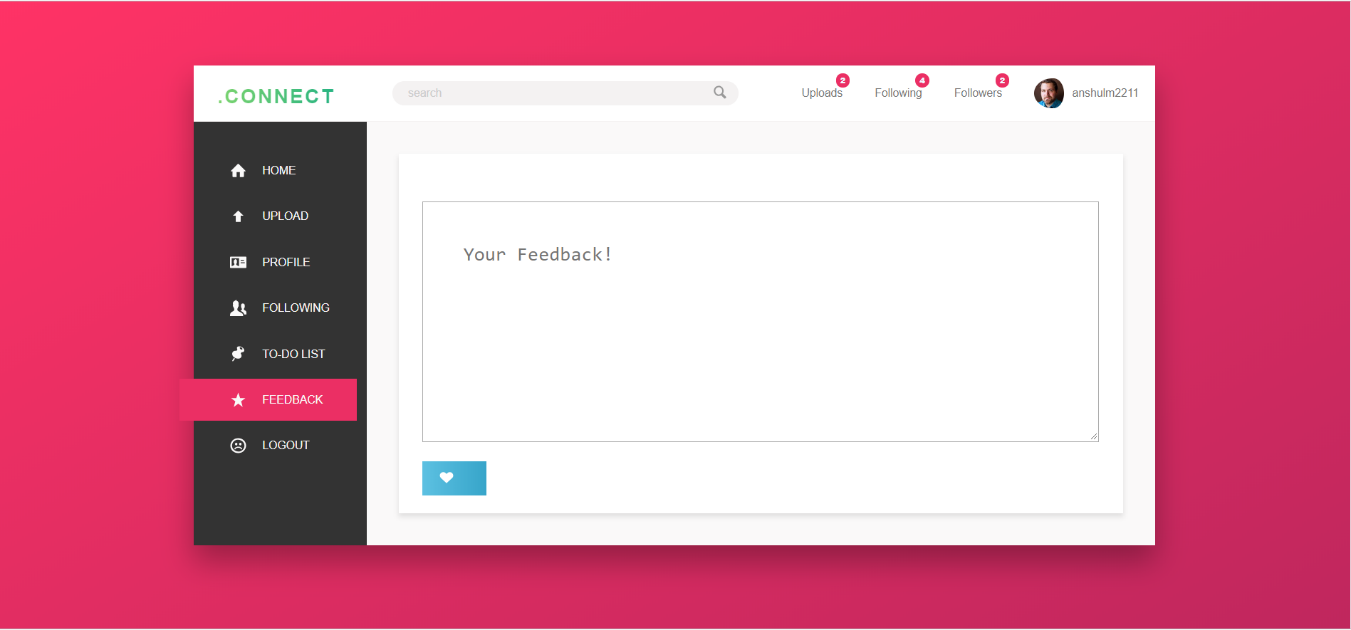
This is the special feature of this site where user can maintain a to-do list in his/her account where he can make up a list of all the tasks which user is supposed to do in coming days.



*Fig: 7.17*

The above snap shows some of the to-do list elements in the user’s account.

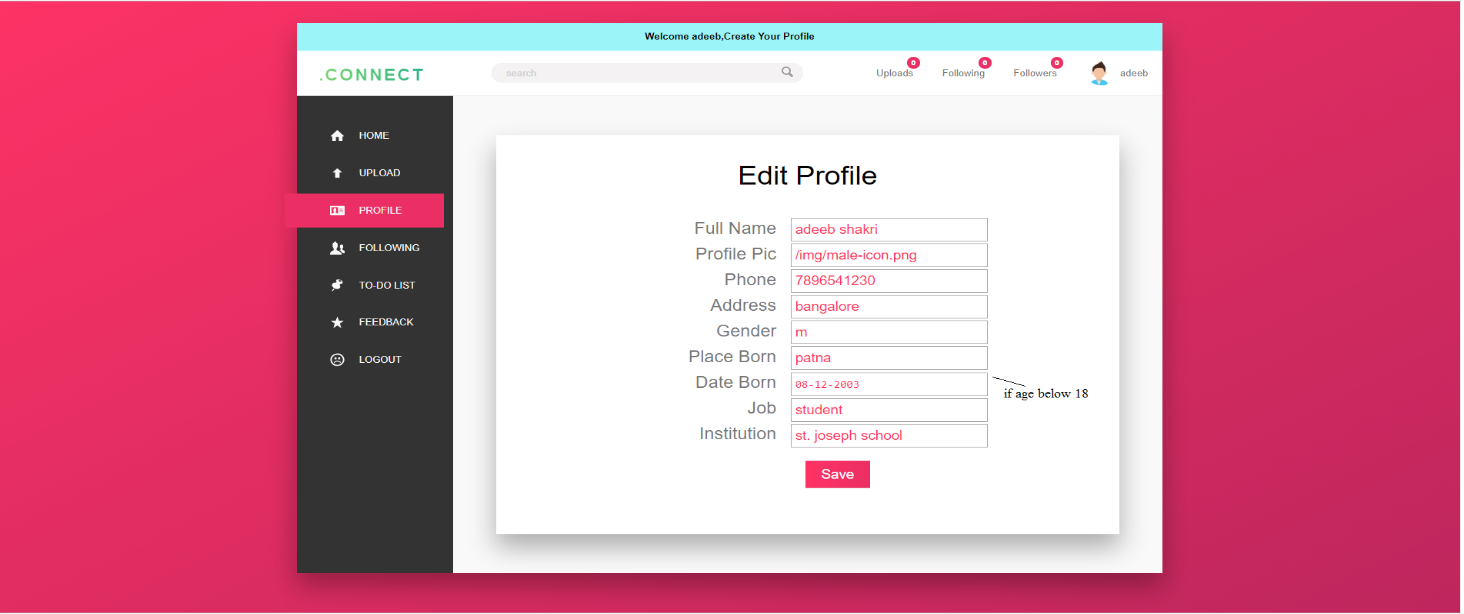
**7.11 FEEDBACK**

****

*Fig: 7.18*

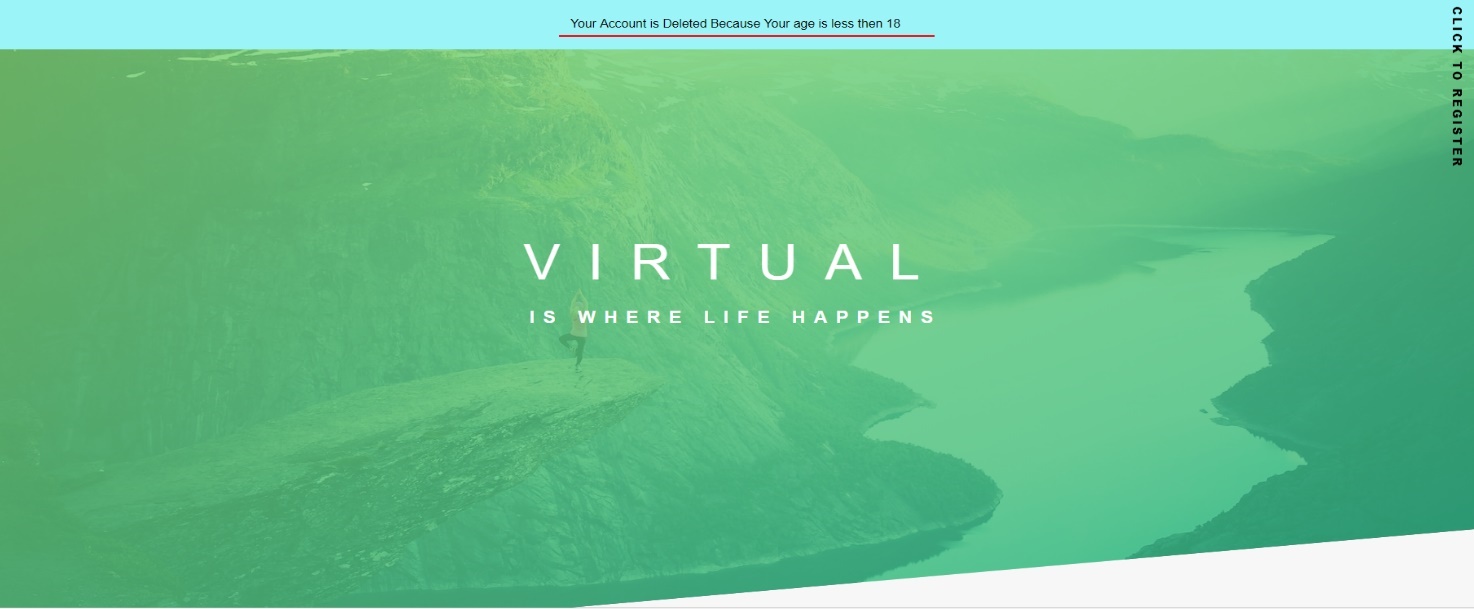
If a user wants to give any feedback regarding the site, then he can give it by following the above snap.

**7.12 CONCEPT OF TRIGGER**

****

*Fig: 7.19*

Consider if user has registered into the .CONNECT and while updating the profile user has given the date of birth which tells that user’s age is less than 18 then one of the rule which is “user’s must be an adult” is violated and the moment user clicks on “save” the following happens.



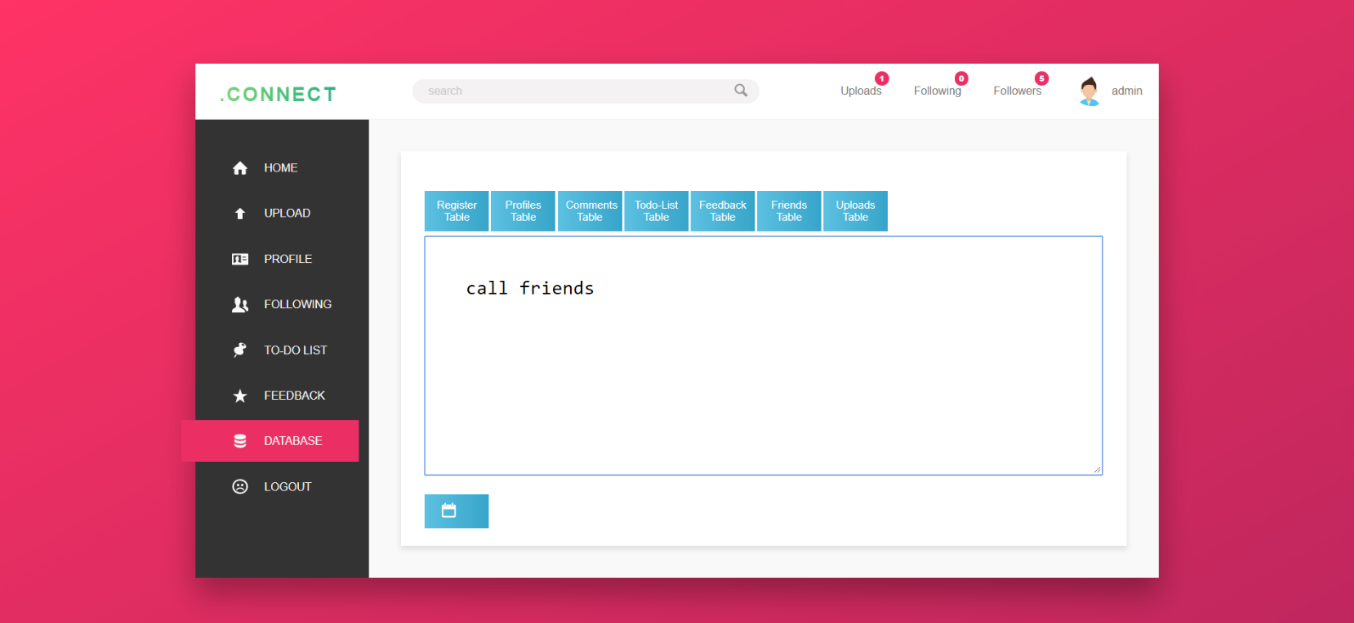
*Fig: 7.20*

The user is then redirected directly to discover page where he is given a message that your account is deleted since the age is less than 18. We used the concept of trigger to implement this in our project.

**7.13 CONCEPT OF STORED PROCEDURE**

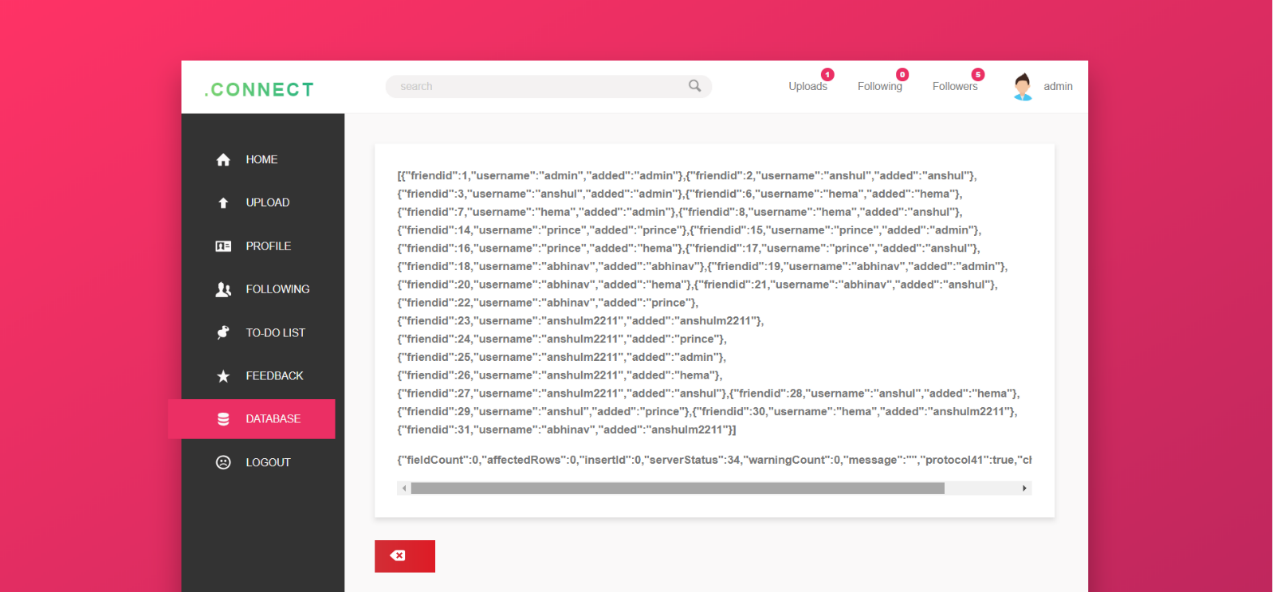
Stored procedures are the user defined procedure just like functions in other languages which can be used to call some queries in project which are repeatedly required.

Admin has some special features where he can see the entire database through the site where he can select the particular icon to see up the contents of the table, to retrieve the details from tables the stored procedure is used.



*Fig: 7.21*

Admin can enter the query in the text box, here the admin is calling the stored procedure ‘friends’, it displays all the contents of all the friends table.



*Fig: 7.22*

Otherwise user can also click on any one of the icon to retrieve the details, here also the stored procedures are used.

**CONCLUSION & FUTURE ENHANCEMENT**

A database was created for a user is interested in connecting itself to the world. Here resources like making a profile, uploading an image, following other users and many more stuffs are provided. Here we have concentrated more on the user interface because that is the thing which will attract the user more. Here we have used MySQL database system to store the database for the project.

For the future works on this project we plan to add chatting facility to provide user a chat box to have some endless conversation with another user. We have also planned to add a feature to provide a remainder for the to-do lists which user has created from its account and we are looking forward to add the tagging facility where user can tag another user in the pic which it uploads.

**REFERENCES**

* [www.google.com](http://www.google.com)
* [www.w3school.com](http://www.w3school.com) (for MySQL and database connectivity)
* [www.openclassroom.com](http://www.openclassroom.com) (for node.js tutorial)
* [www.stackoverflow.com](http://www.stackoverflow.com)
* YouTube videos on node.js
* Sumita Arora, Information Practices
* [www.wikipedia.com](http://www.wikipedia.com)
* Ramkrishnan and Gehrke, Datbase Management Systems, 3rd edition 2014