A project report on

USING NODEJS AND MONGODB TO BUILD A COLLEGE SOCIAL NETWORK APPLICATION

Submitted in partial fulfillment for the award of the degree of

Master of Computer Application (M.C.A)

by

ASHWANI GUPTA (16MCA0025)

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING (SITE)

JANUARY 2018

USING NODEJS AND MONGODB TO BUILD A COLLEGE SOCIAL NETWORK APPLICATION

Submitted in partial fulfillment for the award of the degree of

Master of Computer Application (M.C.A)

by

ASHWANI GUPTA (16MCA0025)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING (SITE)

JANUARY 2018

<u>DECLARATION</u>

I here by declare that the thesis entitled "USING NODEJS AND MONGODB TO BUILD A COLLEGE SOCIAL NETWORK APPLICATION" submitted by me, for the award of the degree of MASTER OF COMPUTER APPLICATION(M.C.A). VIT is a record of bonafide work carried out by me under the supervision of Prof. BRIJENDRA SINGH.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore Signature of the Candidate

Date :

CERTIFICATE

This is to certify that the thesis entitled "USING NODEJS AND MONGODB TO BUILD A COLLEGE SOCIAL NETWORK APPLICATION" submitted by ASHWANI GUPTA (16MCA0025) , SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING , VIT, for the award of the degree of "Master of Computer Application" is a record of bonafide work carried out by him under my supervision.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The Project report fulfils the requirements and regulations of VIT and in my opinion meets the necessary standards for submission.

Signature of the Guide Signature of the Hod

Internal Examiner External Examiner

Date: ………………..

CERTIFICATE BY THE EXTERNAL GUIDE

This is to certify that the project report entitled "VALUE STREAM MAPPING(VSM) GENRATOR" submitted by ASHWANI GUPTA(16MCA025) to Vellore Institute of Technology in partial fulfillment of the requirement for the award of the degree of MASTER OF COMPUTER APPLICATION in COMPUTER APPLICATION is a record of bonafide work carried out by him under my guidance. The project fulfills the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

<Signature of the External Supervisor>

<EXTERNAL GUID NAME>

Cerner Healthcare Solutions Private Ltd.
Ground Floor, Wing B, Block H2, Mountain Ash
Manyata Embassy Business Park Nagawara
Bangalore - 560 045, India

<SEAL OF THE COMPANY>

ABSTRACT

The demand for real-time, scalable and high performance application is ever increasing. NodeJS is a java script based framework for server side scripting on the web. Node (NodeJS) is considered a high performance and low memory consuming environment which aims to support long running applications. Traditional application used tabular relational database system which had strict consistency constraints. Complex queries usually involved relational joins to achieve the required data set. MongoDB is a NoSQL database system which is semi-structured in nature and offers a dynamic schema, NoSQL is best suitable for large scalable applications. This paper aims to support NodeJS and MongoDB as server side solution to build a college social network application which can be used by students and faculties.

CONTENTS

Chapter 1

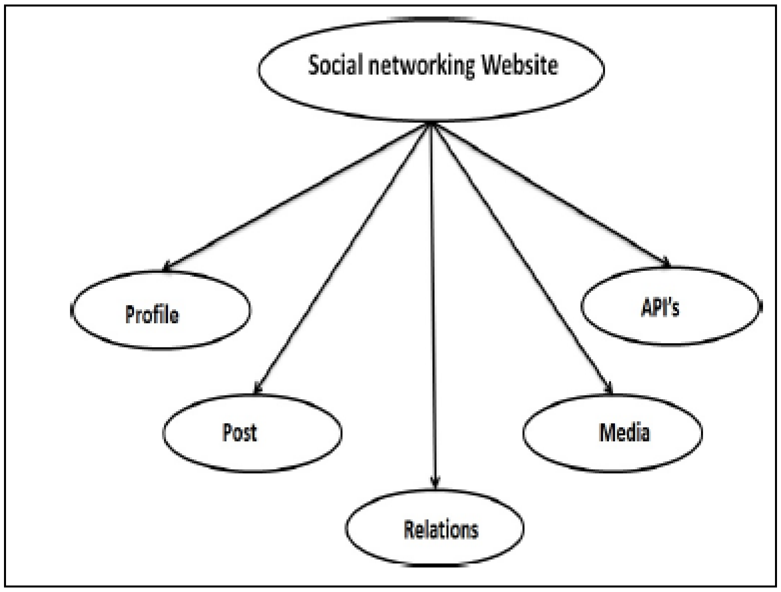
INTRODUCTION

1.1 SOCIAL NETWORKING WEBSITE

Social networking service is platform which gives peoples opportunity to build social network or social relations among people across the globe, who shares their thoughts, interests, activities or real life connections. Early startups have grabbed this opportunity for being successful due to the high popularity and interest among people. For discussing a good database for social networking website we must know the basic components of social networking website, will see these components in next part of this section.

1.2 COMPONENTS OF SOCIAL NETWORKING WEBSITES

Below figure represents the social networking website. As it involves the various activities of several users, the main components of these websites can be considered as People/Profile, Post/Blog, Media/Image/video/Audio, Relations/Interactions and API's.



OVERVIEW Fig - 1

1.2.1 PROFILE

This component collects user related information viz. biographical information (name, age, sex, living), educational information and professional information. These details give identity to user on the social networking website. This identity can be used by other users to form network among users. A profile can have connections with other profiles i.e. friendship, follow or subscribe.

1.2.2 POST

This component gives editing and publishing functions to users. For example user can post a blog, tweet etc. He can also publish some of his experiences.

1.2.3 MEDIA

Media includes various photos, audios and videos which a user wants to share with his connections. Media is the most important entity for a user.

1.2.4 RELATIONS

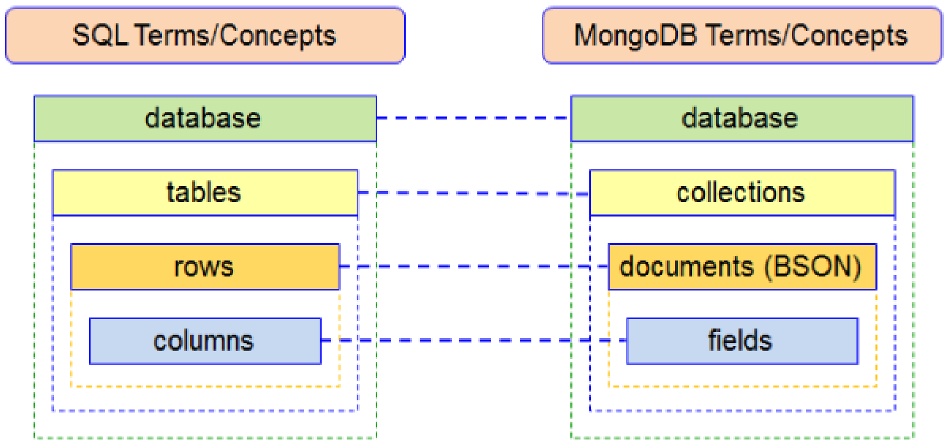
Relations or interaction can be the likes, shares and comments etc. which are received on a post or blog.

1.2.5 API's

API gives social networking websites a way to interact with the other websites or apps. API cannot be ignored as these are the most important entity responsible for the success of most social networking websites.

1.3 MONGODB

MongoDB is a scalable high performance open-source NoSQL database. MongoDB uses document store instead of two dimensional table structures. It was developed by the company 10gen as component of a planned platform as a service product . It was being adopted as backend software by major services companies including eBay, SourceForge, FourSquare and New York Times. The differences between MongoDB and normal Relational DBMS can be well represented in the following figure. The figure shows how Tables in RDBMS are replaced by Collections in MongoDB, Rows are replaced by JSON documents and Joins are replaced by Embedding and linking. MongoDB is good at scaling, uses Map Reduce and has descent geospatial integration



SQL terms/concepts in NoSQL Fig - 2

1.3.1 ADVANTEGES OF MONGODB

MongoDB has attracted many developers due to its features superior over the traditional databases. MongoDB has evolved as a new form of database which can be used by the developers. The reason why one should use MongoDB for Social Networking Websites are document oriented storage, indexes on any attribute, Replication and High Availability, Auto Sharding, Rich Queries, Fast in-place updates. These advantages are discussed in more details below. One of the first and most promising advantages which have attracted developers toward MongoDB is the less initial development efforts needed. As MongoDB follows procedures and methods like commands for interacting with it, developers don't have to waste much of the time in learning how to write code for MongoDB database. It sounds weird but it has been observed by many developers that writing MongoDB commands is more easier than written those long SELECT * FROM like queries.

One more advantage of MongoDB is its good at scaling. Scaling is achieved using Sharding in MongoDB. Sharding is the process of storing data records across multiple machine nodes and is MongoDB's approach to meeting the demands of data growth. With increasing size of data, a single machine may seem insufficient to store the data. Also it provides unacceptable read and write throughput. This problem of horizontal scaling is solved using sharding process. With Sharding, we add more machines to support data growth and demands of read and write operations.

MongoDB is Document Oriented Database in which one collection holds different documents, number of fields. Content and size of document can differ from one document to another. Even the user can change/add/delete any of these documents at any time, this feature makes MongoDB Schema-less.

Deep query ability of MongoDB is outstanding. MongoDB supports dynamic queries on documents using a document based query language that is nearly as powerful as SQL. This allows documents retrieval from database in short duration. Some research articles which were referenced while deciding the title.

Indexes support the efficient execution of queries in MongoDB. MongoDB defines indexes at the collection level and supports indexes on any field or sub-field of the documents in a MongoDB collection.

MongoDB's support for API development can be additional advantage in selecting MongoDB for developing a Social Networking Website [6]. Its document store structure suits the object structure that gets returned or processed by an API in response to GET or POST requests. Such structure is well used by Social Networking giant Facebook. Other advantages of using MongoDB include easy replication, MapReduce, clustering etc.

1.3.2 DISADVANTAGES OF MONGODB

Although we considered MongoDB as suitable for social networking website, there is another side of the coin that should be considered while selecting MongoDB for any social networking project. Many developers have realized this and shifted from using MongoDB to other databases within one year of their project startup. So going to the final decision regarding database selection without considering these cons would lead to risks. The cons are discussed below.

Performing JOIN query is one of the crucial points which have made developer think again before using MongoDB. Every application at some point needs to perform JOIN operations. But in case of Mon goDB which uses document like structure. If one need information from one table to filter the information from another table (i.e. a join), then MongoDB is going to work against us. In a SQL database we can easily get the info from multiple tables with a single JOIN query. In MongoDB we need multiple queries and join the data manually within our code (which may cause slow & ugly code, also reduced flexibility when the structure changes) to get data from multiple collections. Any speeds advantage that mongo gives us in pulling from a single collection will quickly be negated by making multiple round trips to the database

MongoDB supports indexes on any field or sub-field of the document; even it supports compound and multi-key indexes. But in some of the practical implementations it has been found that setting up indexes on many fields or attributes takes up a lot of RAM space. Actually these are B-tree indexes and if we have many, we might run out of system resources really fast.

There are some concurrency issues in MongoDB, when we perform a write operation in MongoDB it creates a lock on the entire database, not just the affected entries and not just for a particular connection, thus lock blocks not only write operation but also read operations.

Operations are not automatically treated as transactions by MongoDB. We have to manually choose to create a transaction, verify it manually, and then manually commit or rollback for ensuring data integrity upon create/update operations. Operations on a single document are always atomic with MongoDB databases; however, operations which involve multiple documents are not atomic, operations are often referred to as "transactions". Many practical use cases are supported by single-document atomicity as documents can be fairly complex and contain multiple "nested" documents.

Some other issues that can be considered at lower extent are less documentation, smaller community or experience with a document store database than relational database like MySQL.

1.4 NODEJS

NodeJS is an open source java script framework . It is a cross platform server side scripting environment. NodeJS resides on Google chrome's powerfull V8 engine. It is best suitable for I/O intensive applications such as streaming sites. Thousands of developers around the globe have already started using NodeJS as their back end tool for server side rendering. Being a single threaded application it is at the same time highly scalable. The same NodeJS program can take as much larger requests than compared to traditional apache HTTP servers. Instead of buffering the data it simply outputs the data in chunks. Some domains were NodeJS can prove to be a better choice are I/O bound applications, data streaming, JSON API, and traditional single page application.

1.5 OBJECTIVE

Social networking is a strategy to connect and discuss all sorts of ideas through individuals and communities. College social network is all about announcement and stories on those announcements (comments and trend). All the stakeholders of the organization will be able to use the platform for collaboration

1.6 SCOPE

Social networking has tremendous prospects in future. The bright future prospect of social networking is also proven with the fact that the technology is integrated in the mobile phones as well. Social networking has become an important tool of marketing in true sense of customer orientation.

CHAPTER 2

SURVEY

2.1 FINDINGS FORM RESEARCH

Kanoje, Sumitkumar, Varsha Powar, and Debajyoti Mukhopadhyay. [1] potray that MongoDB is good at some point for developing social networking website and somewhere it lags too. But that doesn’t means it should never be used, it might not the optimal solution in some situations. That doesn’t mean it don’t have a lot potential. Our database choice should be application specific.

Lei, Kai, Yining Ma, and Zhi Tan. This paper [2] shows that Node.js performs much better than the traditional technique PHP in high concurrency situation, no matter in benchmark tests or scenario tests. PHP handles small requests well, but struggles with large requests. Besides, Node.js prefers to be used in the IO-intensive situation, not compute-intensive sites. Python-Web is also not suitable for the computeintensive website.

Liang, Li, et al, in this paper [3] mainly introduces the system architecture, the data storage solutions based on MongoDB database and the statistical analysis solutions based on MapReduce. Finally, this paper presents user query and statistical analysis modules for the Express Supervision System.

Viknes Balasubramanee, Chathuri Wimalasena ,Raminder Singh, Marlon Pierce [4] discusses mainly about frameworks which is a combination of css classes and function related to javascript so as to get rid of using front end deveolpement It supports Model view controller,where the view is generaated in the browser using its Model which holds all the required data,where the controller takes care of the interactions between the html page and the model,here no server side scripting is used as the data is fetched from from the client site by using catched data.

Andrew John Poulter, Steven J. Johnston, Simon J. Cox [5] show us that MEAN is a free and open-source JavaScript software stack for building dynamic web sites and web applications. Best MEAN Stack developers utilise incredible features to implement across app and web development projects. Node JS to provide concurrent JavaScript environment for building scalable and fast web applications.

André Nitze [6] show that odularity helps the process of organizing the entire code base. So if modular approach is not applied then the structure will grow and become out of control. The modular approach in javascript programming helps to develop more structured code and make it easily maintainable.

Zhu Wei-ping,Li Ming-xin [7] Advancement of NOSQL,a non relational database management which supports database loose storage. SIMS is a modular application, the core offering covering storing basic school data with modules available to handle (among other things) legal registration, the recording of achievements and sanctions and the management and documentation of public examinations.

S. L. Bangare, S. Gupta, M. Dalal, A. Inamdar [8] potray that One of the more interesting developments recently gaining popularity in the server-side JavaScript space is Node.js. It's a framework for developing high-performance, concurrent programs that don't rely on the mainstream multithreading approach but use asynchronous I/O with an event-driven programming model.

SheilaA. McIlraith, Tran Cao Son, and Honglei Zeng [9] show that A semantic Web service, like conventional Web services, is the server end of a client–server system for machine-to-machine interaction via the World Wide Web. Semantic services are a component of the semantic Web because they use markup which makes data machine-readable in a detailed and sophisticated way (as compared with human-readable HTML which is usually not easily "understood" by computer programs).

Musangi Muthui [10] shows that Cloud computing is an information technology (IT) paradigm that enables ubiquitous access to shared pools of configurable system resources and higher-level services that can be rapidly provisioned with minimal management effort, often over the Internet. Cloud computing relies on sharing of resources to achieve coherence and economy of scale, similar to a utility.

2.2 LITRATURE SURVEY

S.No.	Title	Author/Year of Published	Research gap analysis
1	Using MongoDB for Social Networking Website Deciphering the Pros and Cons	Kanoje, Sumitkumar, Varsha Powar, and Debajyoti Mukhopadhyay. (ICIIECS), 2015 International Conference on. IEEE, 2015.	MongoDB is good at some point for developing social networking website and somewhere it lags too. But that doesn't means it should never be used, it might not the optimal solution in some situations. That doesn't mean it don't have a lot potential. Our database choice should be application specific.
2	Performance Comparison and Evaluation of Web Development Technologies in PHP, Python and Node.js	Lei, Kai, Yining Ma, and Zhi Tan. Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on. IEEE, 2014.	Node.js performs much better than the traditional technique PHP in high concurrency situation, no matter in benchmark tests or scenario tests. PHP handles small requests well, but struggles with large requests. Besides, Node.js prefers to be used in the IO-intensive situation, not compute-intensive sites. Python-Web is also not suitable for the computeintensive website.
3	Express Supervision System Based on NodeJS and MongoDB	Liang, Li, et al. Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on. IEEE,	mainly introduces the system architecture, the data storage solutions based on MongoDB database and the statistical analysis solutions based on MapReduce. Finally, this paper presents user query and

		2017.	statistical analysis modules for the Express Supervision System.
4	Twitter Bootstrap and Angular	Viknes Balasubramanee, Chathuri Wimalasena ,Raminder Singh, Marlon Pierce 2013	framework which is a combination of css classes and function related to javascript so as to get rid of using front end deveolpement It supports Model view controller,where the view is generaated in the browser using its Model which holds all the required data,where the controller takes care of the interactions between the html page and the model,here no server side scripting is used as the data is fetched from from the client site by using catched data.
5	Using the MEAN Stack to Implement a Restful Service for an Internet of Things Application	Andrew John Poulter, Steven J. Johnston, Simon J. Cox	It also establishes a secured path for better communicating with IOT devices by means of using pull-communication
6	Modularity of JavaScript Libraries and Frameworks in Modern Web Applications	André Nitze September 2014	current JS libraries and structures are assessed as far as their measured quality property.
7	Using Mongo DB to Implement Textbook Management System instead of MySQL	Zhu Wei-ping,Li Ming-xin	Advancement of NOSQL,a non relational database management which supports database loose storage
8	Node.js: Using JavaScript to Build High-Performance Network Programs	S. L. Bangare, S. Gupta, M. Dalal, A. Inamdar *19March 2016*	Use Of Nosql databse is preferred which includes MogoDb to build well developed backened
9	Semantic Web Services	SheilaA. McIlraith, Tran Cao Son, and Honglei Zeng March/April 2001	It will misuse clients' requirements and inclinations to help alter clients' solicitations or programmed Web benefit disclosure, execution, or

			sythesis and interoperation
10	Amazon Web Services Building dynamic capabilities for IT	Musangi Muthui, CGU Sisat February 28, 2013	As Infrastructure as a service the company has crossed the boundary in maintaining the top position by bewildering its own dynamic captivity so as to gain better uniformed it service.

CHAPTER 3

3.1 ARCHITECTURE DIAGRAMS

3.1.1 USE CASE DIAGRAM

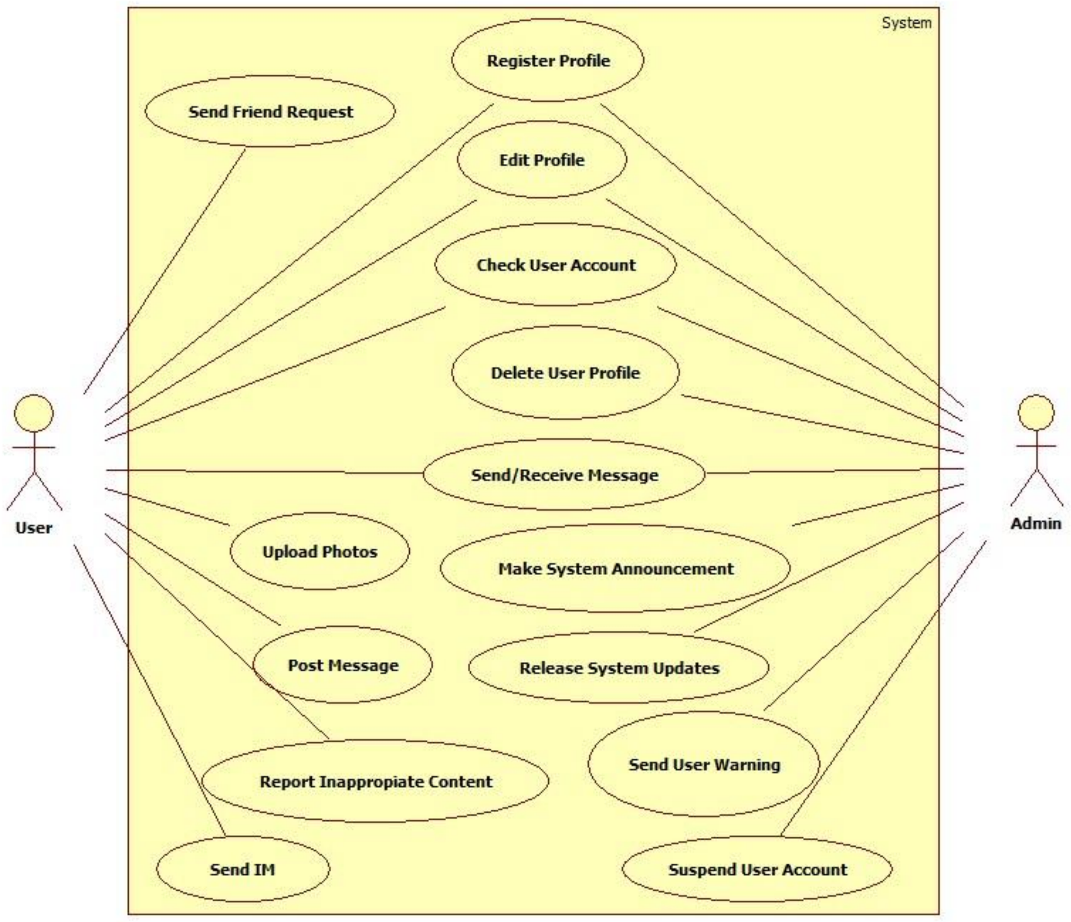


Fig - 3

Use case User: User is the primary actor of the system or the main person who shares their views on the discussions topics in the form of post and comments. The users have the capabilities to edit their individual profile. The users also get the capabilities to report inappropriate content and also send IMs in real time.

Use case Admin: admin takes care of monitoring reported tags/story and also has the capabilities to remotely suspend any user's account. The admin makes announcement and sends user warnings. Admin is too required register a profile with different access rights.

FUNCTIONAL REQUIREMENTS: must have a valid User ID and password to login. Who don't have their account in this site, can create a new account for signup. Should not be allowed to have more than one profile. can edit his/her profile and can post his/her views basing on the activities he/she desires, after the valid user login his/her account

3.1.2 CLASS DIAGRAM

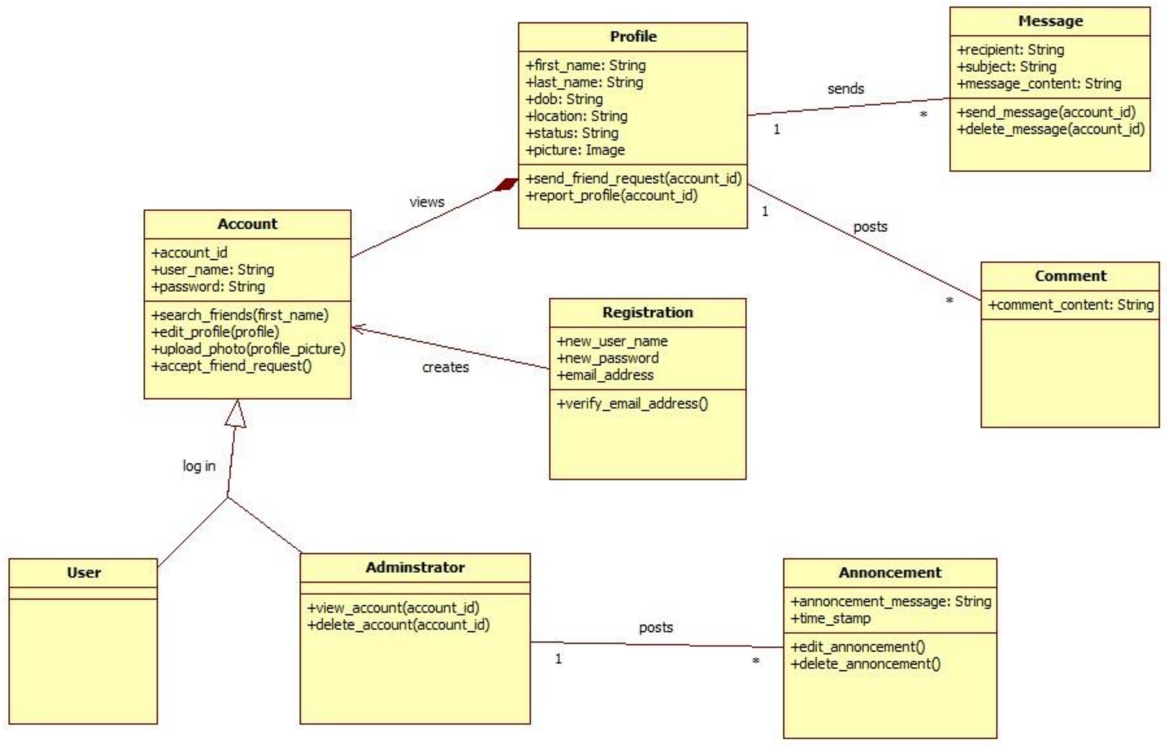


Fig - 4

3.1.3 DATABASE DIAGRAM

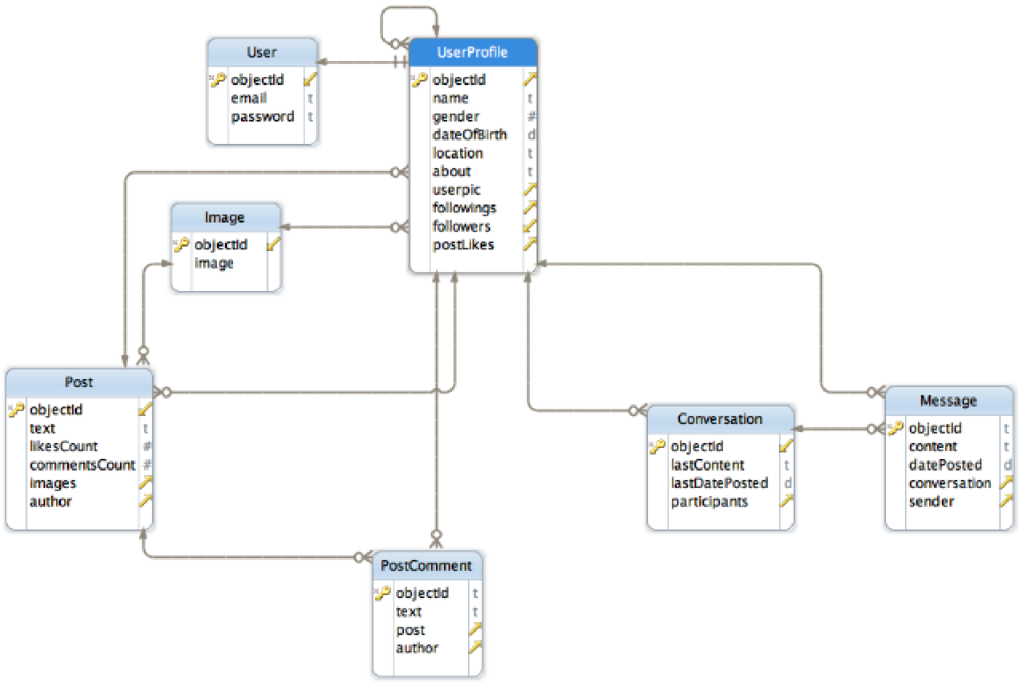


Fig - 5

Database Description

Description of Primary entities

User/UserProfile.- User/UserProfile To protect the user's email I've created another class for public user profile called UserProfile and add one-to-one relationship with User class. UserProfile is visible to anyone. User is visible only to the owner and the admin.

UserProfile - UserProfile has followings, followers and postLikes fields, which are one-to-many relationships to UserProfile and Post entity respectively. All of them

Image - Instead of storing images directly in PFFile field I've created a separate class for that. Therefore images can be stored in array and many-to-many relationships. This gives a better flexibility if we'll want to add, for example, like/unlike functionality to a photo.

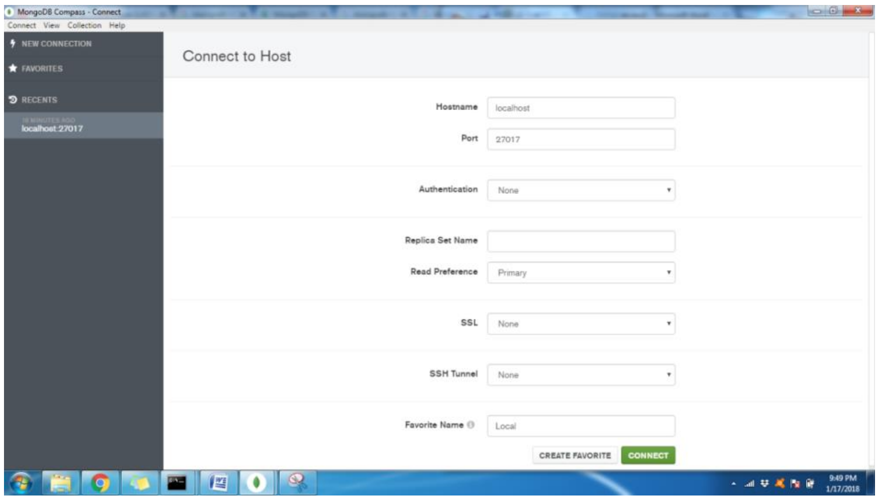
Chapter 4

4.1 IMPLIMENTATION(Design)

4.1.1 MONGODB COMPASS

MongoDB Compass is designed to allow users to easily analyze and understand the contents of their data collections within MongoDB and perform queries, without requiring knowledge of MongoDB query syntax.

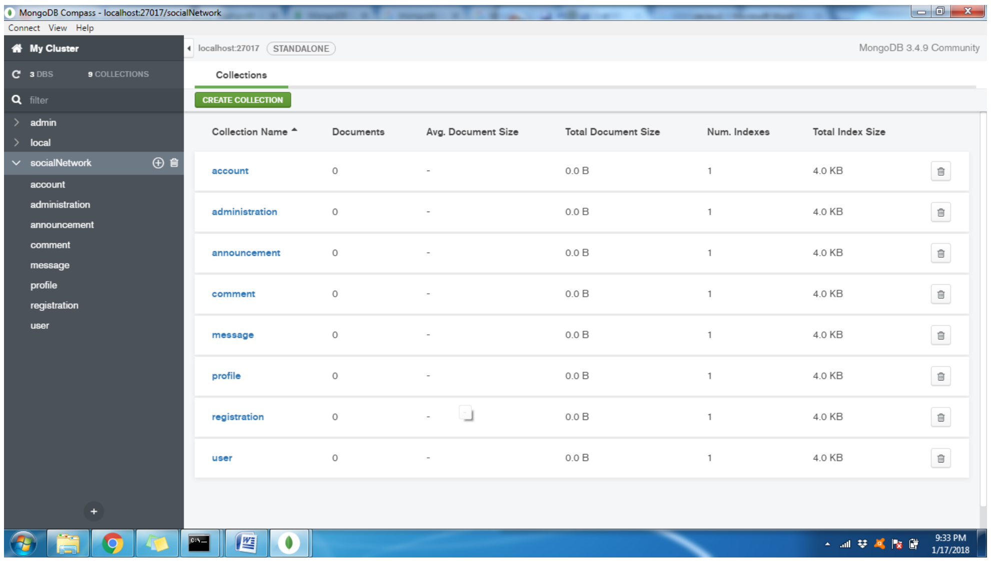
MongoDB Compass provides users with a graphical view of their MongoDB schema by randomly sampling a subset of documents from the collection. Sampling documents minimizes performance impact on the database and can produce results quickly.



Connect to Host

Hostname	localhost
Port	27017
Authentication	None
Replica Set Name	
Read Preference	Primary
SSL	None
SSH Tunnel	None
Favorite Name	Local

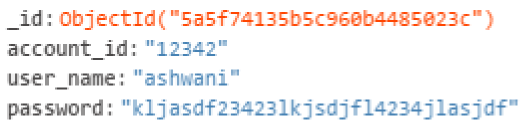
SCREENSHOT (login) Fig - 6



Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size
account	0	-	0.0 B	1	4.0 KB
administration	0	-	0.0 B	1	4.0 KB
announcement	0	-	0.0 B	1	4.0 KB
comment	0	-	0.0 B	1	4.0 KB
message	0	-	0.0 B	1	4.0 KB
profile	0	-	0.0 B	1	4.0 KB
registration	0	-	0.0 B	1	4.0 KB
user	0	-	0.0 B	1	4.0 KB

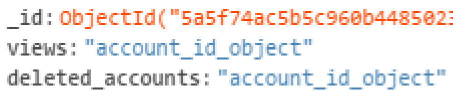
SCREENSHOT (Social Network Database) Fig - 7

4.1.2 SAMPLE DOCUMENT IN THE DATABASE WITH SCREENSHOTS



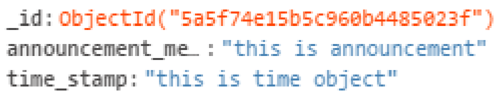
```
_id: ObjectId("5a5f74135b5c960b4485023c")
account_id: "12342"
user_name: "ashwani"
password: "kljasdf23423lkjsdjfl4234jlasjdf"
```

Account db



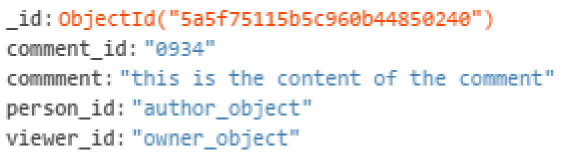
```
_id: ObjectId("5a5f74ac5b5c960b4485023e")
views: "account_id_object"
deleted_accounts: "account_id_object"
```

Administration db



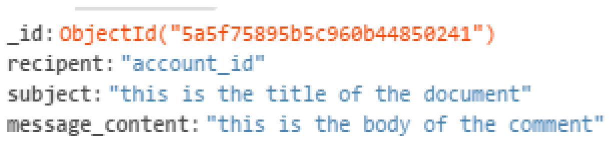
```
_id: ObjectId("5a5f74e15b5c960b4485023f")
announcement_me… : "this is announcement"
time_stamp: "this is time object"
```

Announcement db



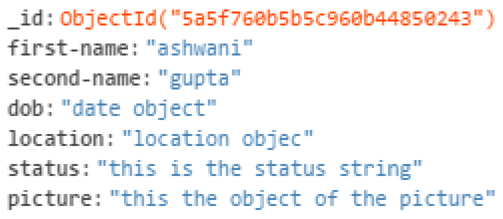
```
_id: ObjectId("5a5f75115b5c960b44850240")
comment_id: "0934"
commment: "this is the content of the comment"
person_id: "author_object"
viewer_id: "owner_object"
```

Comment db



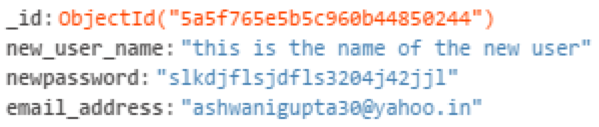
```
_id: ObjectId("5a5f75895b5c960b44850241")
recipent: "account_id"
subject: "this is the title of the document"
message_content: "this is the body of the comment"
```

Message db



```
_id: ObjectId("5a5f760b5b5c960b44850243")
first-name: "ashwani"
second-name: "gupta"
dob: "date object"
location: "location objec"
status: "this is the status string"
picture: "this the object of the picture"
```

Profile db



```
_id: ObjectId("5a5f765e5b5c960b44850244")
new_user_name: "this is the name of the new user"
newpassword: "slkdjflsjdfls3204j42jjl"
email_address: "ashwanigupta30@yahoo.in"
```

Registration db

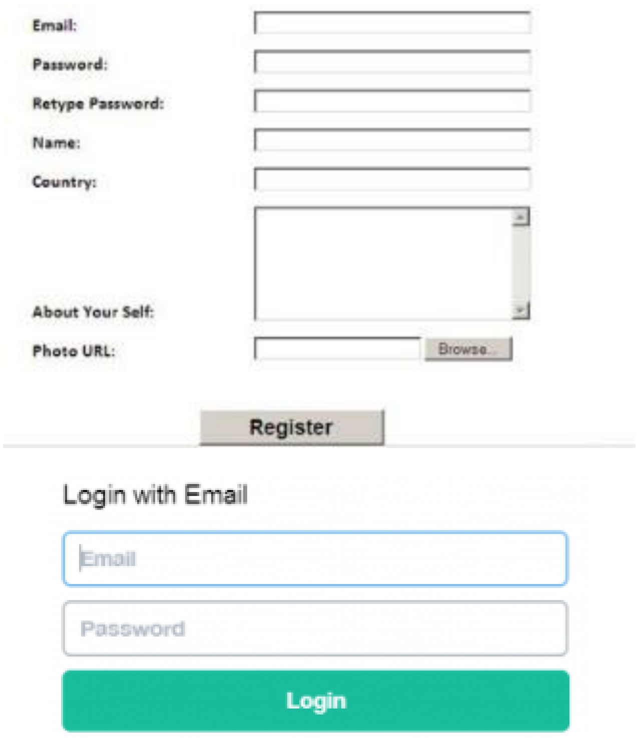
4.1.3 DATABASE CONNECTIVITY WITH NODEJS

```
var MongoClient = require('mongodb').MongoClient

var URL = 'mongodb://localhost:27017/mydatabase'

MongoClient.connect(URL, function(err, db) {

  if (err) return

}
```

4.1.4 CRUD OPERATIONS ON MONGODB USING EXPRESSJS(NODE)

CREATE OPERATION

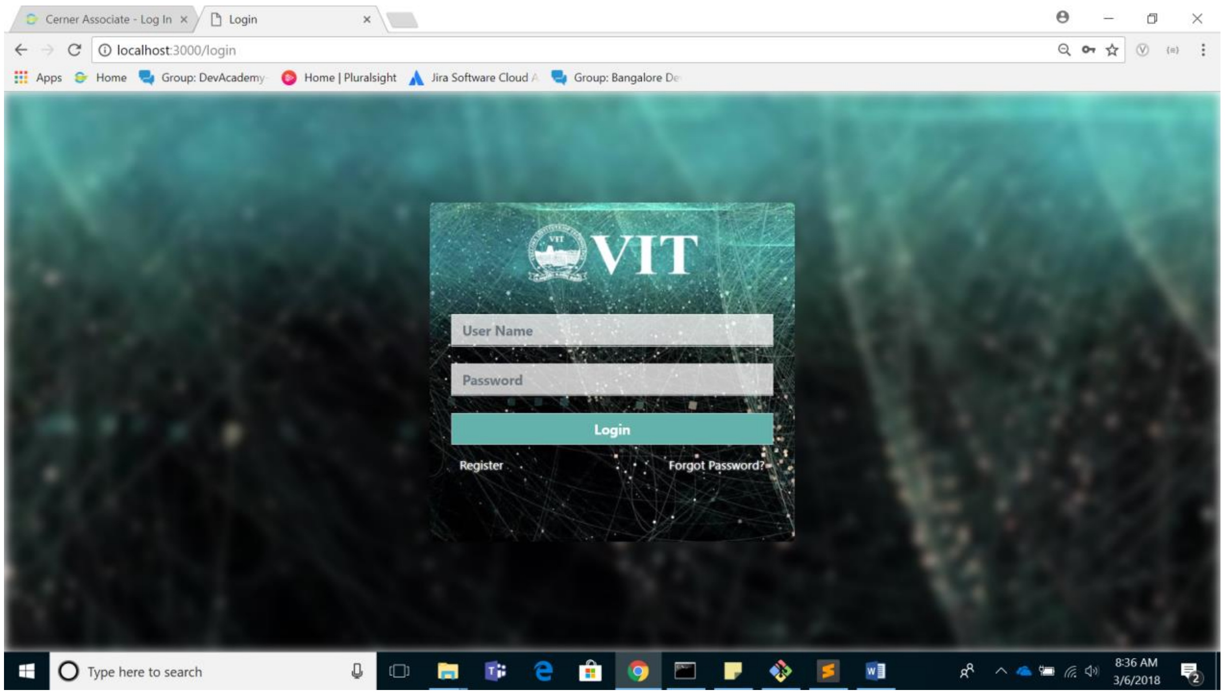
```
$http.post('/register',$scope.part). //$scope contains the JSON object form the registration page

                then(function(response) {

                   console.log("posted successfully");

                }).catch(function(response) {

                   console.error("error in posting");

                })

    }
```



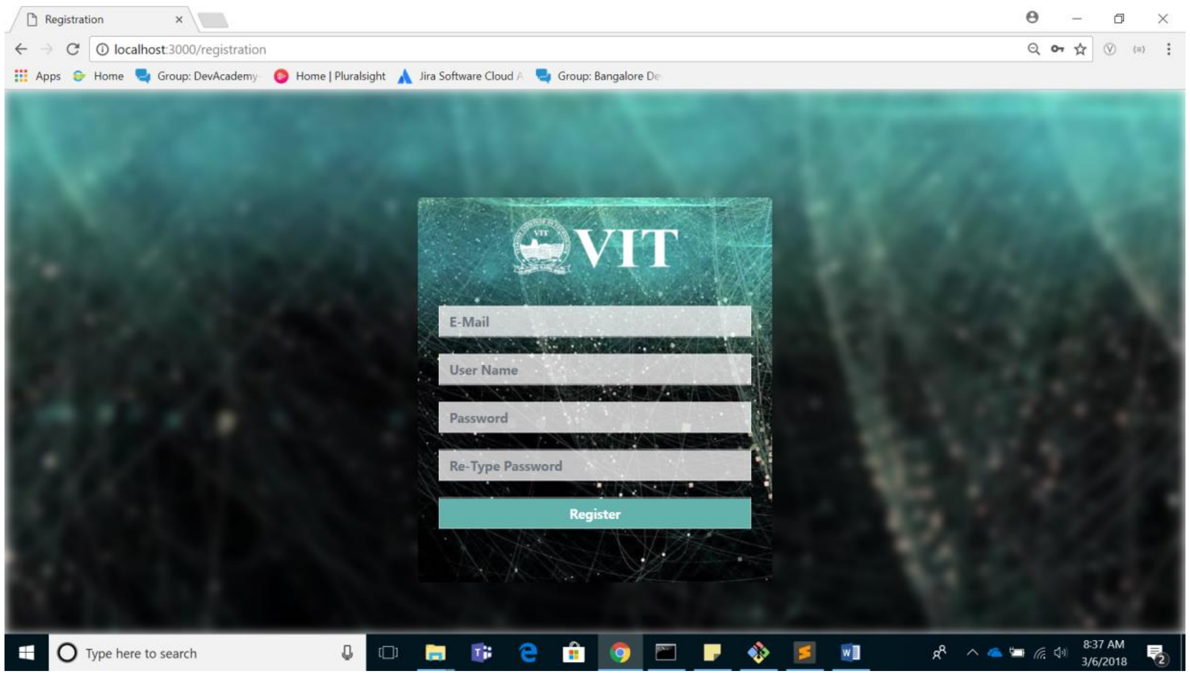
Registration and login page (sample design not final) Fig - 8

Chapter 5

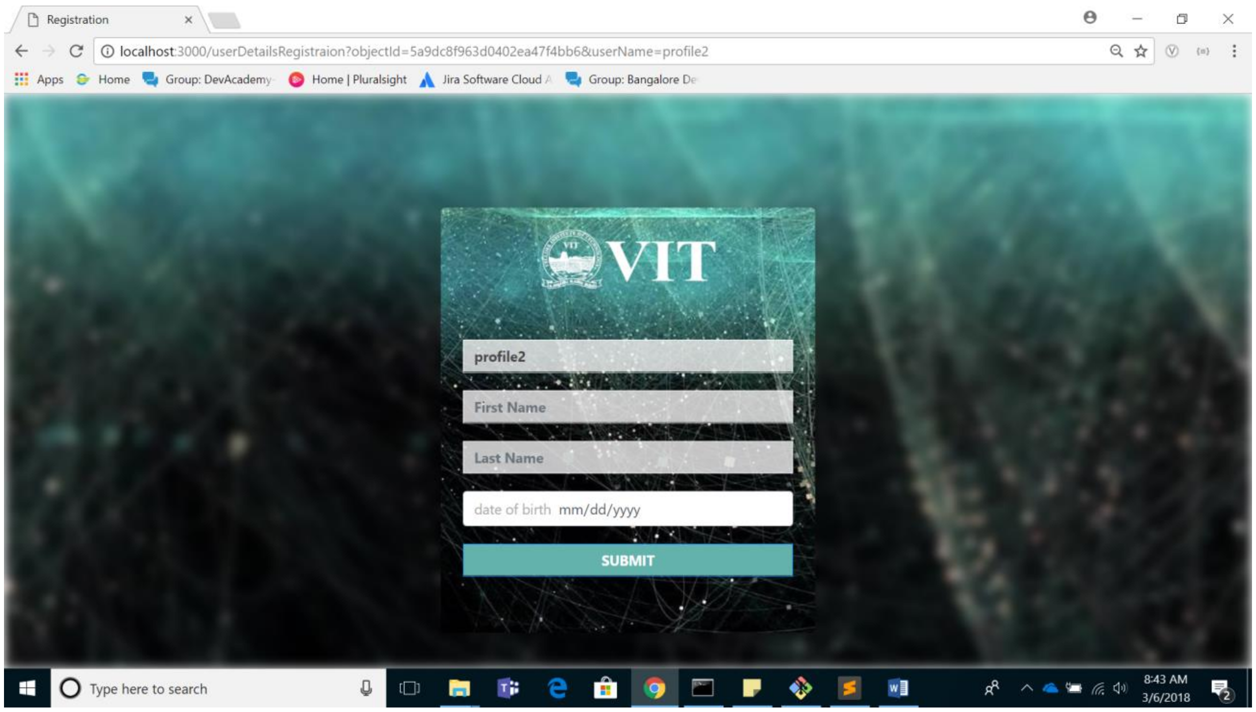
5.1 Implementation (Screen Shots)



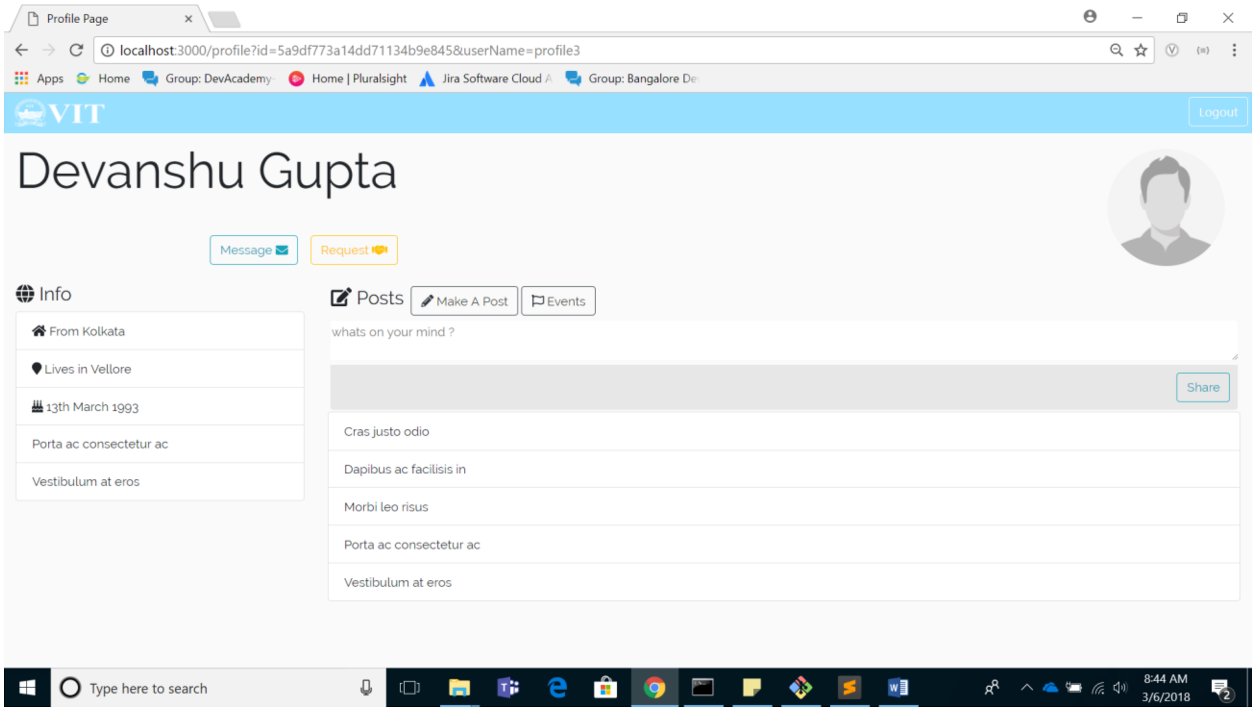
LOGIN PAGE Fig – 9



REGISTRATION PAGE Fig – 10


Registration
localhost:3000/userDetailsRegistraion?objectId=5a9dc8f963d0402ea47f4bb6&userName=profile2
VIT
profile2
First Name
Last Name
date of birth mm/dd/yyyy
SUBMIT

User Details Page – Fig 11


Profile Page
localhost:3000/profile?id=5a9df773a14dd71134b9e845&userName=profile3
VIT
Logout
Message
Request
Info
Posts
Make A Post
Events
Share

Devanshu Gupta

From Kolkata
Lives in Vellore
13th March 1993
Porta ac consectetur ac
Vestibulum at eros

whats on your mind ?

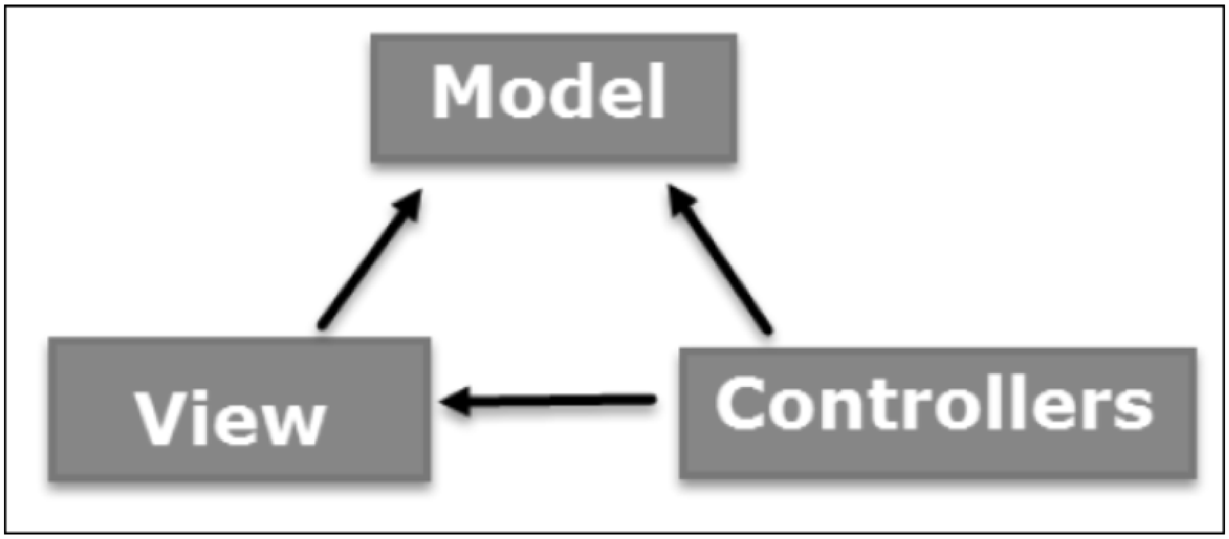
Cras justo odio
Dapibus ac facilisis in
Morbi leo risus
Porta ac consectetur ac
Vestibulum at eros

Profile Page - Fig 12

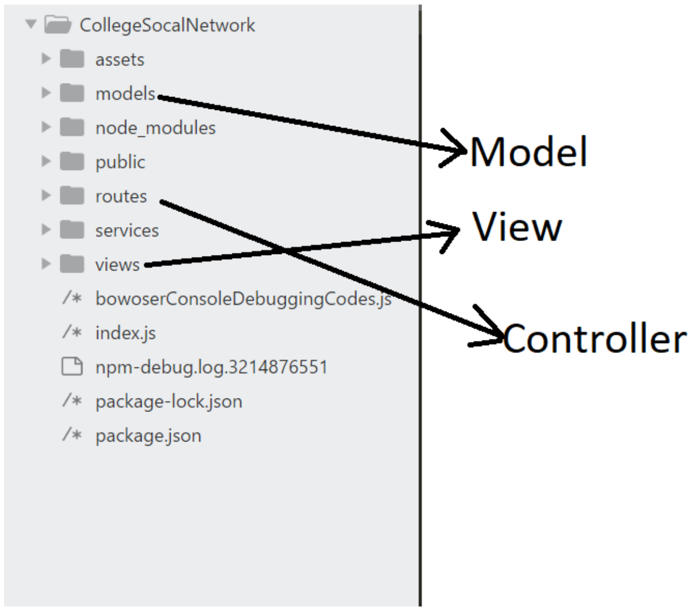
5.2 PROJECT STRUCTURE

5.2.1 Model View Controller

The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. MVC is one of the most frequently used industry-standard web development framework to create scalable and extensible projects.



Model View Controller Fig 13



Model View Controller Implemented in the current scenario Fig 14

Chapter 6

6 Implementation (Source Code)

6.1 MODELS

6.1.1 User Model

```
const mongoose = require("mongoose");
var Schema = mongoose.Schema;

var userSchema = new Schema({
            userName            :String,
            email               :String,
            password:String

});
module.exports = mongoose.model('Users',userSchema);
```

6.1.2.User Details

```
const mongoose = require("mongoose");
var Schema = mongoose.Schema;

var userDetailsSchema  = new Schema({
            _id                 :String,
            userName  :String,
            // email            :String,
            // password         :String,
            firstName  :String,
            lastName   :String,
            dateOfBirth         :String,
            friendList   :[{
                                            userName  :String,
                                            friendType :String,
                                 }],
            posts               :[{
                                            postId              :Number,
                                            description         :String
                                 }]
});

module.exports  = mongoose.model('UserDetails',userDetailsSchema);
```

6.2 VIEWS

6.2.1 Login Page

```
<!DOCTYPE html>
<html>
<head>
            <title>Login</title>
```

```
		<!-- bootstrap css -->
		<link rel="stylesheet" href="/css/bootstrap.css"/>
		<!-- custom css for this page -->
		<link rel="stylesheet" href="/css/login.css"/>
</head>
<body>
		<!-- for the blur background -->
		<div class="body-background"></div>

		<!-- container for login form -->
		<div class="container login-container">
			<div class="login-image-container">
				<img src="img/vitLogo2.png">
			</div>
			<form method="post" action="/login" class="">
				<div class="form-field-container">
					<input type="text" placeholder="User Name" name="username" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="password" placeholder="Password" name="password" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="submit" name="submit" value="Login" class="form-control">
					<p class="forgot-password-container"><span class="register-here"><a
href="/registration">Register</a></span><span class="forgot-password"><a href="forgotPassword"> Forgot Password? </a></span></p>
				</div>
			</form><!-- end of login form -->
		<div><!-- end of login container -->



		<!-- jquery js -->
		<script src="/js/jquery.min.js"></script>
		<!-- popper js -->
		<script src="/js/popper.js"></script>
		<!-- bootstrap js-->
		<script src="/js/bootstrap.js"></script>
</body>
</html>
```

6.2.2. Registration Page

```
<!DOCTYPE html>
<html>
<head>
		<title>Registration</title>
		<!-- bootstrap css -->
		<link rel="stylesheet" href="/css/bootstrap.css"/>
		<!-- toastr css -->
		<link rel="stylesheet" href="/css/toastr.css"/>
		<!-- custom css for this page -->
		<link rel="stylesheet" href="/css/registration.css"/>
</head>
<body>
		<!-- for the blur background -->
		<div class="body-background"></div>

		<!-- container for login form -->
		<div class="container login-container">
			<div class="login-image-container">
				<img src="img/vitLogo2.png">
			</div>
			<form method="post" action="/registration" onsubmit="return validateRegistrationForm()">
				<div class="form-field-container">
					<input type="text" id='email' placeholder="E-Mail" name="email" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="text" id='username' placeholder="User Name" name="username" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="Password" id='password' placeholder="Password" name="password" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="Password" id='repassword' placeholder="Re-Type Password" name="repassword" class="form-
control">
```

```
				</div>
				<div class="form-field-container">
					<input type="submit" id='submit' name="submit" value="Register" dass="form-control">
				</div>
			</form><!-- end of login form -->
		<div><!-- end of login container -->



		<!-- jquery js -->
		<script type="text/javascript" src="/js/jquery.min.js"></script>
		<!-- popper js -->
		<script type="text/javascript" src="/js/popper.js"></script>
		<!-- bootstrap js-->
		<script type="text/javascript" src="/js/bootstrap.js"></script>
		<!-- toastr js -->
		<script type="text/javascript" src="/js/toastr.min.js"></script>
		<!-- custom js files for this page -->
		<script type="text/javascript" src="/js/registration.js"></script>
</body>
</html>
```

6.2.3. Detail Entry Page

```
<!DOCTYPE html>
<html>
<head>
		<title>Registration</title>
		<!-- bootstrap css -->
		<link rel="stylesheet" href="/css/bootstrap.css"/>
		<!-- toastr css -->
		<link rel="stylesheet" href="/css/toastr.css"/>
		<!-- custom css for this page -->
		<link rel="stylesheet" href="/css/registration.css"/>
		<link rel="stylesheet" href="/css/userDetailsRegistration.css"/>
</head>
<body>
		<!-- for the blur background -->
		<div class="body-background"></div>

		<!-- container for login form -->
		<div class="container login-container">
			<div class="login-image-container">
				<img src="img/vitLogo2.png">
			</div>
			<form method="post" action="/userDetailsRegistraion" onsubmit="return validateUserDetailsForm()">
				<div class="form-field-container">
					<input type="text" id='objectId' name="objectId" class="form-control" hidden="">
				</div>
				<div class="form-field-container">
					<input type="text" id='userName' value="" name="userName" class="form-control">
				</div>
				<div class="form-field-container">
					<input type="text" id='firstName' placeholder="First Name" name="firstName" class="form-
control">
				</div>
				<div class="form-field-container">
					<input type="text" id='lastName' placeholder="Last Name" name="lastName" class="form-
control">
				</div>
				<div class="form-field-container">
					<input type="date" id='dateOfBirth' placeholder="date of birth" name="dateOfBirth" class="form-
control">
				</div>
				<div class="form-field-container">
					<input type="submit" id='submitbtn' name="submitbtn" class="form-control btn btn-primary"
value="SUBMIT">
				</div>
```

```
                    </form><!-- end of login form -->
        <div><!-- end of login container -->



        <!-- jquery js -->
        <script type="text/javascript" src="/js/jquery.min.js"></script>
        <!-- popper js -->
        <script type="text/javascript" src="/js/popper.js"></script>
        <!-- bootstrap js-->
        <script type="text/javascript" src="/js/bootstrap.js"></script>
        <!-- toastr js -->
        <script type="text/javascript" src="/js/toastr.min.js"></script>
        <!-- custom js files for this page -->
        <script type="text/javascript" src="/js/userDetailsRegistration.js"></script>
</body>
</html>
```

4. Profile Page

```
<!DOCTYPE html>
<html>
<head>
        <title>Profile Page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- bootstrap css -->
        <link rel="stylesheet" href="/css/bootstrap.css"/>
        <!-- custom css for this page -->
        <link href="CSS/profile.css" rel="stylesheet" type="text/css"/>
        <!-- railwy font -->
        <link href="https://fonts.googleapis.com/css?family=Raleway" rel="stylesheet">
</head>
<body>
        <div class="container-fluid header-bar">
          <div class="logout-button-holder">
            <a href="/logout" class="btn btn-outline-light logout-button">Logout</a>
          </div>
          <div class="logo-holder">
            <img src="img/vitLogo2.png" alt="vit logo"/>
          </div>
        </div>
        <div class="container-fluid page-body">
          <div class="person-name-pic-container">
            <div class="name-holder">
              <span id="firstName">Ashwani</span><span id='lastName'>Gupta</span>
              <div class="below-name-button-holder">
                <a href="/message" class="btn btn-outline-info">Message <i class="fas fa-envelope"></i></a>
                <a href="/request" class="btn btn-outline-warning">Request <i class="fas fa-handshake"></i></a>
              </div>
            </div>
            <div class="profile-pic-holder">
              <a href="/viweProfilePic">
                <img src="img/profilepic.jpg" alt=""/>
              </a>
            </div>
          </div>
          <div class="details-posts-container row justify-content-md-center">
            <div class="col-6 col-md-3 user-details-holder">
              <h3><i class="fas fa-globe"></i> Info</h3>
              <ul class="list-group">
                <li class="list-group-item"><i class="fas fa-home"></i> From Kolkata</li>
                <li class="list-group-item"><i class="fas fa-map-marker"></i> Lives in Vellore</li>
                <li class="list-group-item"><i class="fas fa-birthday-cake"></i> 13th March 1993</li>
                <li class="list-group-item">Porta ac consectetur ac</li>
                <li class="list-group-item">Vestibulum at eros</li>
              </ul>
```

```
                    </div>
                    <div class="col-6 col-md-9 user-posts-holder">
                        <div class="post-actions-holder">
                            <h3><i class="fas fa-edit"></i> Posts</h3>
                            <a href="/makeApost" class="btn btn-outline-dark"><i class="fas fa-pencil-alt"></i> Make A Post</a>
                            <a href="/events" class="btn btn-outline-dark"><i class="far fa-flag"></i> Events</a>
                        </div>
                        <div class="share-area-holder">
                            <textarea placeholder="whats on your mind ?"></textarea>
                            <div class="share-button-holder">
                                <a href="/share" class="btn btn-outline-info">Share</a>
                            </div>
                        </div>
                        <ul class="list-group">
                            <li class="list-group-item">Cras justo odio</li>
                            <li class="list-group-item">Dapibus ac facilisis in</li>
                            <li class="list-group-item">Morbi leo risus</li>
                            <li class="list-group-item">Porta ac consectetur ac</li>
                            <li class="list-group-item">Vestibulum at eros</li>
                        </ul>
                    </div>
                </div>
            </div>
            <!-- jquery js -->
            <script src="/js/jquery.min.js"></script>
            <!-- popper js -->
            <script src="/js/popper.js"></script>
            <!-- bootstrap js-->
            <script src="/js/bootstrap.js"></script>
            <!-- font awesome for awesome icons -->
            <script src="js/font-awesome.js" type="text/javascript"></script>
            <!-- custom js -->
            <script type="text/javascript" src="/js/profile.js"></script>
</body>
</html>
```

6.3 Route / Controllers

6.3.1 Login Route

```
//include all the required dependencies
const url = require('url');
var express = require('express');
var router = express.Router();

//include all the required models
var Users = require('../models/Users/');
var UserDetails = require('../models/UserDetails/');
router.get("/",function(req, res) {
   res.render('login');
});

router.post("/",function(req, res) {
        var obj_userAutenticationSearchQuery = {
                'userName'          :req.body.username,
                'password' :req.body.password
        }

        // print the requested body on the console
        console.log("req.body")
        console.log(req.body);

        //print the prepared search query on the console
```

```
		console.log("obj_userAutenticationSearchQuery");
		console.log(obj_userAutenticationSearchQuery);

		// search the mongodb database for the user
		Users.find(obj_userAutenticationSearchQuery,function(err,doc){
			if(err) console.log("error fetching data from Data base");

			// if user is found
			if(doc.length >= 1){
				console.log("user Schema");
				console.log(doc);

				var userObjectId = (doc[0]._id).toString();
				var userName = (doc[0].userName).toString();
				obj_userDetailsSearchQuery = {'userName' : obj_userAutenticationSearchQuery.userName};

				// search of userDetails in other database
				UserDetails.find(obj_userDetailsSearchQuery,function(err,doc){

					if(err) console.log("error fetching userDetails");
					else if(doc.length == 1){
						res.redirect(url.format({
					  pathname:"/profile",
					  query: {
					    "id": doc[0]._id,
					    "userName": doc[0].userName
					   }
					 }));
					}
					else{
						// res.redirect('/userDetailsRegistraion');
						console.log(userObjectId);
						res.redirect(url.format({
						  pathname:"/userDetailsRegistraion",
						  query: {
						    "objectId": userObjectId,
						    "userName": userName
						   }
						 }));
					}

				});
			}

			/if no documents are found
			else{
				console.log("login failed :rendering the login page again")
				res.render('login');
			}
		})
});
module.exports = router;
```

6.3.2 Registration Route

```
//include all the required dependencies
const url = require('url');
var express = require('express');
var router = express.Router();

//include all the required models
var Users = require('../models/Users/');
var UserDetails = require('../models/UserDetails/');
router.get("/",function(req, res) {
  res.render('registration');
});

router.post("/",function(req,   res) {
        data = {
                        'email'        :req.body.email,
                        'userName'                :req.body.username,
                        'password' :req.body.password
                };
                console.log(data);
                var userData = new Users(data);
                userData.save(function(err,doc){
                        if(err){
                                console.log('error  in storing  data');
                                console.log(err);
                        }
                        else{
                                res.render('login');
                        }
                });
});

module.exports  = router;
```

6.3.3 User Details Entry Route

```
//include all the required dependencies
const url = require('url');
var express = require('express');
var router = express.Router();

//include all the required models
var Users = require('../models/Users/');
var UserDetails = require('../models/UserDetails/');
router.get("/",function(req, res) {
   res.render('userDetailsRegistraion');
});

router.post("/",function(req, res) {
        console.log("req.body");
        console.log(req.body);

        console.log("prepared data for insertion");
        var data = {
                "_id"                         : req.body.objectId,
                "userName": req.body.userName,
                "firstName" : req.body.firstName,
                "lastName"  : req.body.lastName,
                "dateOfBirth"               : req.body.dateOfBirth,
                "friendList" :[],
                "posts"                                :[]
        };
        console.log(data);

        var thisUser = new UserDetails(data);
        thisUser.save(function(err,doc){
                if(err){
                        console.log('error in storing data');
                        console.log(err);
                }
                else{
                        console.log("doc after storing");
                        console.log(doc);
                        res.redirect(url.format({
                            pathname:"/profile",
                            query: {
                              "id": doc._id,
                              "userName": doc.userName
                             }
                        }));
                }
        });
});

module.exports = router;
```

6.3.4. Profile Route

```
//include all the required dependencies
const url = require('url');
var express = require('express');
var router = express.Router();

//include all the required models
var Users = require('../models/Users/');
var UserDetails = require('../models/UserDetails/');

//get route for profile page of each user
router.get("/",function(req, res) {
   res.render("profile");
});

module.exports = router;
```

6.5 VALIDATION

6.5.1 Registration Validation

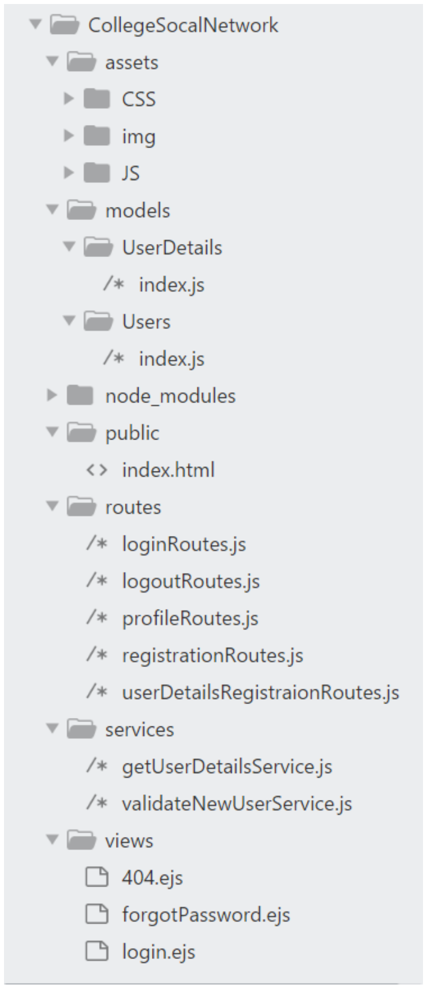
```
//flag to check if the use exists of not;
var newUser = 0;

function validateRegistrationForm() {
        console.log("call validateRegistrationForm()");
        var email = document.getElementById("email").value.toString();
        var username = document.getElementById("username").value.toString();
        var pass1 = document.getElementById("password").value.toString();
        var pass2 = document.getElementById("repassword").value.toString();

        //validate email id;
        if(!validateEmail(email)){
                toastr.error('Please Enter a valid Email Address');
                return false;
        }

        //validate username
        if(username == ""){
                toastr.error("Plase Enter your user name");
                return false;
        }

        //password fields cant be blank
        if(pass1=="" || pass2==""){
                toastr.error('Password field(s) cannot left blank');
                return false;
        }

        //check if the password and re-type password fields are the same;
        if(!(pass1 == pass2)){
                toastr.error('Passwords you Entered don\'t match');
                return false;
        }

        // function which changes the flag variable newUser;
        validateNewUser(email,username);

        // use the flag newUser to check server's response;
        if(newUser == 0){
                toastr.error("User Name / Email is already registered by us");
                return false;
        }
```

```
                    //dbugin purposes only set to false if dont want to submit form;
                    return true;
}

// function to validate email address
function validateEmail(email)
{
 if (/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/.test(email)){
    return true;
  }
  else{
                    return false;
  }
}

function validateNewUser(email,username){
                    // sync ajax request to check the database;
                    $.ajax({
                       url: '/validateNewUser',
                       async : false,
                       type: 'post',
                       data: {
                                    'email' : email,
                                    'username' : username
                       },
                       success: function(response){
                                    console.log("response is " + response);
                                                    if(response == 'taken' ) {
                                                                    newUser = 0;
                                    }
                                    else if(response == 'not_taken') {
                                                    newUser = 1;
                                    }
                       }
                    });
}
```

6.5.2 User Details Validation

```
function getJsonFromUrl() {
  var query = location.search.substr(1);
  var result = {};
  query.split("&").forEach(function(part) {
    var item = part.split("=");
    result[item[0]] = decodeURIComponent(item[1]);
  });
  // console.log(result);
  // return result;
  document.getElementById("objectId").value = result.objectId;
  document.getElementById("userName").value = result.userName;
}
getJsonFromUrl();

function validateUserDetailsForm(){
  var firstName = document.getElementById("firstName").value.toString();
  var lastName = document.getElementById("lastName").value.toString();
  var dateOfBirth = document.getElementById("dateOfBirth").value.toString();

  if(firstName == ""){
    toastr.error("Please Enter your first name");
    return false;
  }
  if(lastName == ""){
    toastr.error("Please Enter your last name");
    return false;
  }
  if(dateOfBirth == ""){
    toastr.error("Please  Enter your date of birth");
    return false;
  }
  return true;
}
```

6.6 The Full Project Structure


```
CollegeSocalNetwork
  assets
    CSS
    img
    JS
  models
    UserDetails
      index.js
    Users
      index.js
  node_modules
  public
    index.html
  routes
    loginRoutes.js
    logoutRoutes.js
    profileRoutes.js
    registrationRoutes.js
    userDetailsRegistraionRoutes.js
  services
    getUserDetailsService.js
    validateNewUserService.js
  views
    404.ejs
    forgotPassword.ejs
    login.ejs
```

Full project structure (expanded) Fig 15

Chapter 7

REFERENCES

[1]. Kanoje, Sumitkumar, Varsha Powar, and Debajyoti Mukhopadhyay. "Using MongoDB for social networking website deciphering the pros and cons." Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on. IEEE, 2015.

[2]. Lei, Kai, Yining Ma, and Zhi Tan. "Performance comparison and evaluation of web development technologies in php, python, and node. js." Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on. IEEE, 2014.

[3]. Liang, Li, et al. "Express supervision system based on NodeJS and MongoDB." Computer and Information Science (ICIS), 2017 IEEE/ACIS 16th International Conference on. IEEE, 2017.

[4]. Temere, Befekadu. "Responsive web application using Bootstrap and Foundation: Comparing Bootstrap and Foundation Frontend Frameworks." (2017).

[5]. Poulter, Andrew John, Steven J. Johnston, and Simon J. Cox. "Using the MEAN stack to implement a RESTful service for an Internet of Things application." Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on. IEEE, 2015.

[6]. Tilkov, Stefan, and Steve Vinoski. "**Node. js: Using JavaScript to build high-performance network programs**." IEEE Internet Computing 14.6 (2010): 80-83.

[7]. Wei-Ping, Zhu, L. I. Ming-Xin, and Chen Huan. "Using MongoDB to implement textbook management system instead of MySQL." Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on. IEEE, 2011.

[8]. Kanoje, Sumitkumar, Varsha Powar, and Debajyoti Mukhopadhyay. "**Using MongoDB for social networking website deciphering the pros and cons.**" Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on. IEEE, 2015.

[9]. Paudyal, Umesh. "Scalable **web application using node. js and couchdb.**" (2011).

[10]. Győrödi, Cornelia, et al. "A **comparative study: MongoDB vs. MySQL**." Engineering of Modern Electric Systems (EMES), 2015 13th International Conference on. IEEE, 2015.

[11]. Kanoje, Sumitkumar, Varsha Powar, and Debajyoti Mukhopadhyay. "**Using MongoDB for Social Networking Website.**" arXiv preprint arXiv:1503.06548 (2015).

[12]. Han, Jing, et al. "**Survey on NoSQL database**." Pervasive computing and applications (ICPCA), 2011 6th international conference on. IEEE, 2011.