# STUDENT ALUMNI PORTAL

A Report for the Final Evaluation of Project

**Submitted By**

**Vidhi Aggarwal (2000290140131)**
**Vaibhav Gupta (2000290140128)**
**Raj Vardhan Chauhan (2000290140095)**

**Submitted in partial fulfilment of the**
**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATIONS**

**Under the Supervision of**
**Prof. Neelam Rawat (Assistant Professor)**
**Prof. Ankit Verma (Assistant Professor)**

**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**DR. APJ ABDUL KALAM TECHNICAL UNIVERSITY LUCKNOW**
**(Formerly Uttar Pradesh Technical University, Lucknow)**

**KIET Group of Institutions, Ghaziabad**
**Uttar Pradesh-201206**
**(JUNE 2022)**

# CERTIFICATE

Certified that **Vidhi Aggarwal (200029014005816), Vaibhav Gupta (200029014005813), Raj Vardhan Chauhan (200029014005780) have** carried out the project work having "**Student Alumni Portal**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:01/06/2022**

**Vidhi Aggarwal (2000290140131)**
**Vaibhav Gupta (2000290140128)**
**Raj Vardhan Chauhan (2000290140095)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Date:01/06/2022**

**Prof. Neelam Rawat (Assistant Professor)**
**Prof. Ankit Verma (Assistant Professor)**

**Department of Computer Applications**
**KIET Group of Institutions, Ghaziabad**

**Signature of Internal Examiner** **Signature of External Examiner**

**Dr. Ajay Shrivastava**
**Head, Department of Computer Applications**
**KIET Group of Institutions, Ghaziabad**

## ABSTRACT

The main purpose of the website is to allow old, new students and faculties of our MCA department to communicate with each other. This allows students to know about each other and their current activities. Many institutions colleges and universities maintain the information manually about present and past students. This does not allow efficient data management and retrieval process. A student willing to get information about others has to approach the college and obtain the details. Often the information may not be available and misleading. It will provide an easy way for the students to talk and interact with each other and their alumni and hold discussions about different topics. They can talk about academic, non-academic and placement related themes. They can even chat about other issues that are common to most of them and orate their minds about other atypical topics. This website allows students to register and then search the data based on different criteria. Also it has the benefit of having a centralize database and up to date information. A user can easily obtain information about their alumni and other registered users. This could also come in handy for asking referrals and other industry related. So, the website has a lot of possibilities to turn into a useful platform and be a comfortable communication podium for present students and their alumni. Initial description of the various functionalities and services provided by the website. The document will also serve the basis for acceptance testing by the user. The website enables the students of the MCA department to sign-up using the email ID using which they can login and use the website further. Authorized users are provided with some facilities such as:

- Administrator module
- One time Email sending facility to Alumni members.
- Testimonials by Alumni.
- Present student module

# ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Prof. Neelam Rawat** and **Prof. Ankit Verma** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Vidhi Aggarwal**

**Vaibhav Gupta**

**Raj Vardhan Chauhan**

# TABLE OF CONTENTS

# INTRODUCTION

The primary reason for the site is to permit old, new understudies and resources of our MCA office to speak with one another. This permits understudies to be aware of one another and their present exercises. Numerous foundations schools and colleges keep up with the data physically about present and past understudies. This doesn't permit proficient information the board and recovery process. An understudy ready to get data about others needs to move toward the school and acquire the subtleties. Frequently the data may not be accessible and misdirecting. It will give a simple way to the understudies to talk and cooperate with one another and their graduated class and have conversations about various points. They can discuss scholarly, non-scholastic and arrangement related subjects. They might talk about different issues that are normal to the greater part of them and speak their brains about other abnormal subjects. This site permits understudies to enlist and afterward search the information in view of various standards. Likewise, it has the advantage of having a bring together data set and exceptional data. A client can undoubtedly get data about their graduated class and other enlisted clients. This could likewise prove to be useful for asking references and other industry related. In this way, the site has a ton of potential outcomes to transform into a valuable stage and be an agreeable correspondence platform for present understudies and their graduated class. Introductory portrayal of the different functionalities and administrations given by the site. The record will likewise serve the reason for acknowledgment testing by the client. The site empowers the understudies of the MCA office to join utilizing the email ID utilizing which they can login and utilize the site further. Approved clients are given a few offices, for example,

- Manager module
- Once Email sending office to Alumni individuals.
- Tributes by Alumni.
- Present under by student

## 1.1 Purpose:

This system can be used as an application for the student alumni portal to manage the college information and student’s information. The system is an online application that can be accessed throughout the organization and outside customers as well with proper login provided, which will give better service to the customers. The graduated students can easily find some information such as contact details for Another graduate of college using the application, which means that they can easily contact any person that is a member of the alumni web site. Making it possible for the former students to keep in touch is not the only functionality that the application will offer. The former students can easily contact the alumni officer if some help is required. Furthermore, the former students will have access to some news that are related to college

## 1.2 Scope:

This system can be used as the Office of Alumni and College Relations seeks to protect the privacy of its alumni and friends, and thus, endeavours to safeguard the use of information in its custody. To that end, the Office of Alumni and College Relations provides constituent information to requestors only under the conditions. The main scope of this project is to make a chain of connection between the alumni and the college from which they got graduation.

➢ To build up a sense of belongingness to the college in which they studied.

➢ To provide a way that keep up both alumni and the college updated about the events

## 1.3 Overview:

Overall description consists of background of the entire specific requirement. It also gives explanation about actor and function which is used. It gives explanation about architecture diagram and it also gives what we are assumed and dependencies. It also supports specific requirement and also it support functional requirement, supplementary requirement other than actor which is used. It also gives index and appendices. It also gives explanation about any doubt and queries. Once a student graduates from the institute, his/her professional life or career begins, with higher education playing an important role in establishing himself/herself in the profession. In respect of college, it has been our experience that from the very beginning, the alumni have maintained personal contacts with one another, rather than use the channel of Alumni Association. The advancements in information technology have certainly helped in creating new resources such as alumni web pages, list servers etc., so as to permit greater interactions between the alumni.

## 1.4. Hardware / Software used in Project:

### 1.4.1 Hardware Specification:

- HDD 20 GB Hard Disk space and above
- 4 GB RAM
- 2.8 GHz Processor and above

### 1.4.2 Software Specification:

- Operating System Windows
- IDE: - Visual Studio Code
- Database: - MongoDB
- Backend: - Express JS, Node JS
- Frontend: - React

# 2. FEASIBILITY STUDY

A feasibility study involves information assessment (identify information required to answer the three questions), information collection (question information sources to discover the answers), and report writing (make a recommendation about whether the system development should continue; propose changes to the scope, budget, and schedule; suggest further high-level requirements for the system).

The relevant information sources for acquiring the necessary information may include personals from organization where the system will be used, developers who are familiar with the type of system that is proposed, technology experts, end-users of the system.

To obtain a picture of how often feasibility studies are considered an integral part of the process in web development an informal survey was carried out. We run the survey through our friends and some other students from the college to know is the website that we are going to develop, and design is needed, or the idea is worth to be worked. And after getting positive feedback we started our work. First, we talked with students regarding the problems they were facing and after that we moved to the depth of the problems and found out how we can eliminate the problems from students' life.

## 2.1 Technical Feasibility:

This involves questions such as whether the technology needed for the system exists, how difficult it will be to build, and whether the firm has enough experience using that technology. The assessment is based on outline design of system requirements in terms of input, processes, output, fields, programs and procedures. This can be qualified in terms of volume of data, trends, frequency of updating in order to give an introduction to the technical system. The application is the fact that it has been developed on windows XP platform and a high configuration of 1GBRAM on Intel Pentium Dual core processor. This is technically feasible. The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system.

## 2.2 Operational Feasibility:

Operational feasibility is the measure of how well a proposed system solves the problems, and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility assessment focuses on the degree to which the proposed development projects fits in with the existing business environment and objectives with regard to development schedule, delivery date, corporate culture and existing business processes. To ensure success, desired operational outcomes must be imparted during design and development.

These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability and others. These parameters are required to be considered at the early stages of design if desired operational behaviours are to be realised. A system design and development require appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters.

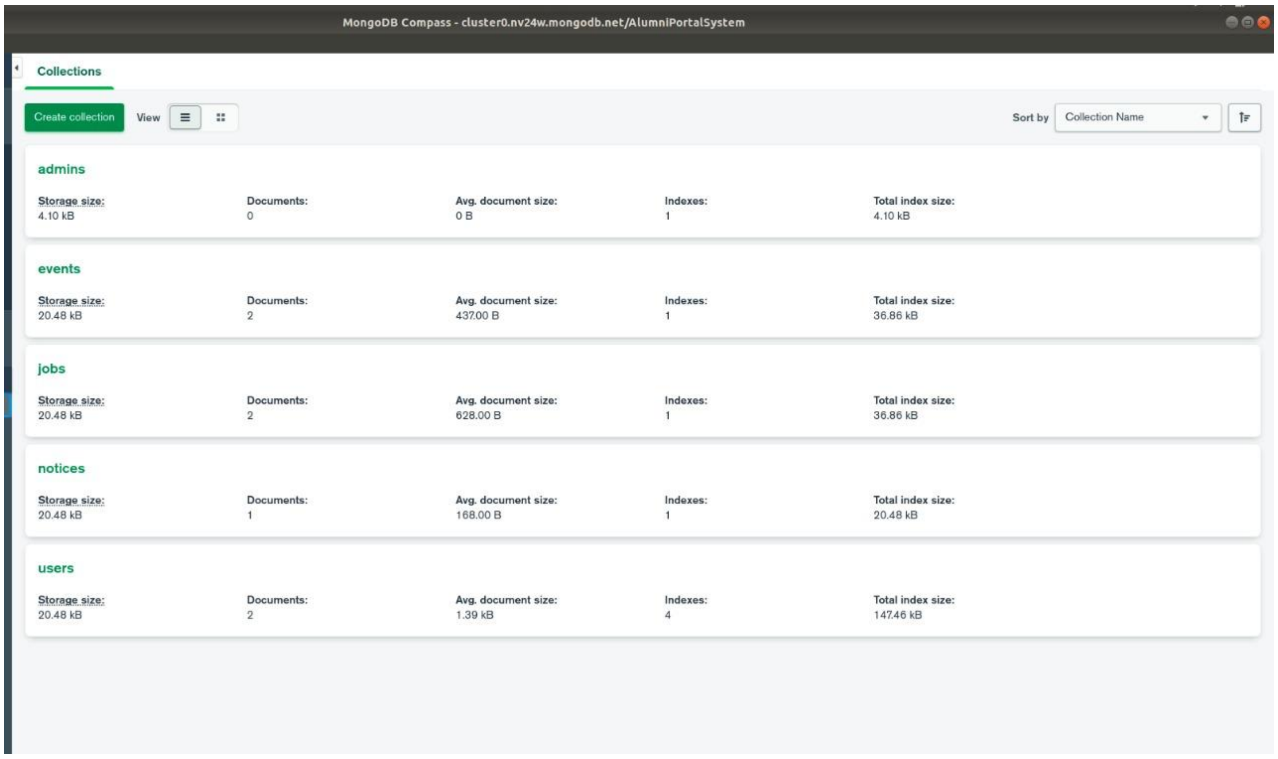
A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that needs to be an integral part of the early design phases.

## 2.3 Behavioural Feasibility:

Establishing the cost-effectiveness of the proposed system i.e., if the benefits do not outweigh the costs, then it is not worth going ahead. In the fast-paced world today there is a great need of online social networking facilities. Thus, the benefits of this project in the current scenario make it economically feasible.

The purpose of the economic feasibility assessment is to determine the positive economic benefits to the organization that the proposed system will provide. It includes quantification and identification of all the benefits expected. This assessment typically involves a cost/benefits analysis.

## 3. DATABASE DESIGN

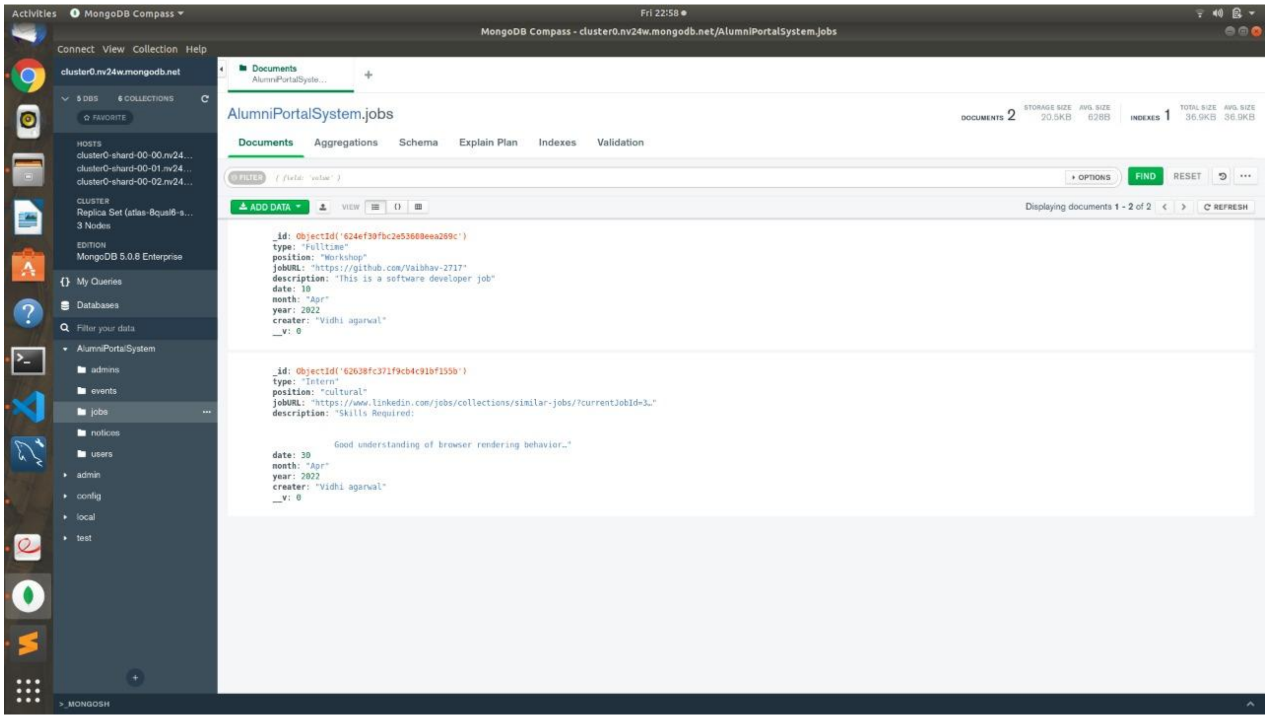


MongoDB Compass - cluster0.nv24w.mongodb.net/AlumniPortalSystem

Collections

| Collection | Storage size: | Documents: | Avg. document size: | Indexes: | Total index size: |
|---|---|---|---|---|---|
| admins | 4.10 kB | 0 | 0 B | 1 | 4.10 kB |
| events | 20.48 kB | 2 | 437.00 B | 1 | 36.86 kB |
| jobs | 20.48 kB | 2 | 628.00 B | 1 | 36.86 kB |
| notices | 20.48 kB | 1 | 168.00 B | 1 | 20.48 kB |
| users | 20.48 kB | 2 | 1.39 kB | 4 | 147.46 kB |

### 3.1 Jobs Collection:

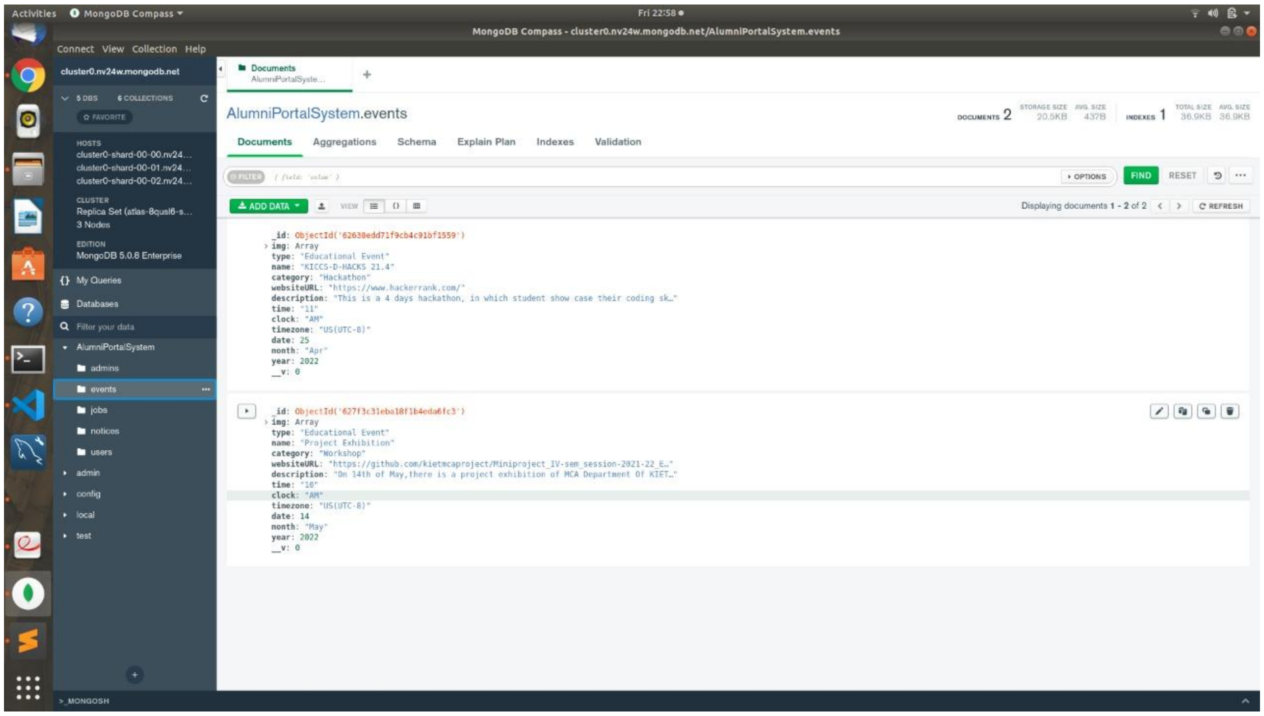


```
_id: ObjectId('624ef30fbc2e53608eea269c')
type: "Fulltime"
position: "Workshop"
jobURL: "https://github.com/Vaibhav-2717"
description: "This is a software developer job"
date: 10
month: "Apr"
year: 2022
creater: "Vidhi agarwal"
__v: 0
```

```
_id: ObjectId('62638fc371f9cb4c91bf155b')
type: "Intern"
position: "cultural"
jobURL: "https://www.linkedin.com/jobs/collections/similar-jobs/?currentJobId=3…"
description: "Skills Required:


        Good understanding of browser rendering behavior…"
date: 30
month: "Apr"
year: 2022
creater: "Vidhi agarwal"
__v: 0
```

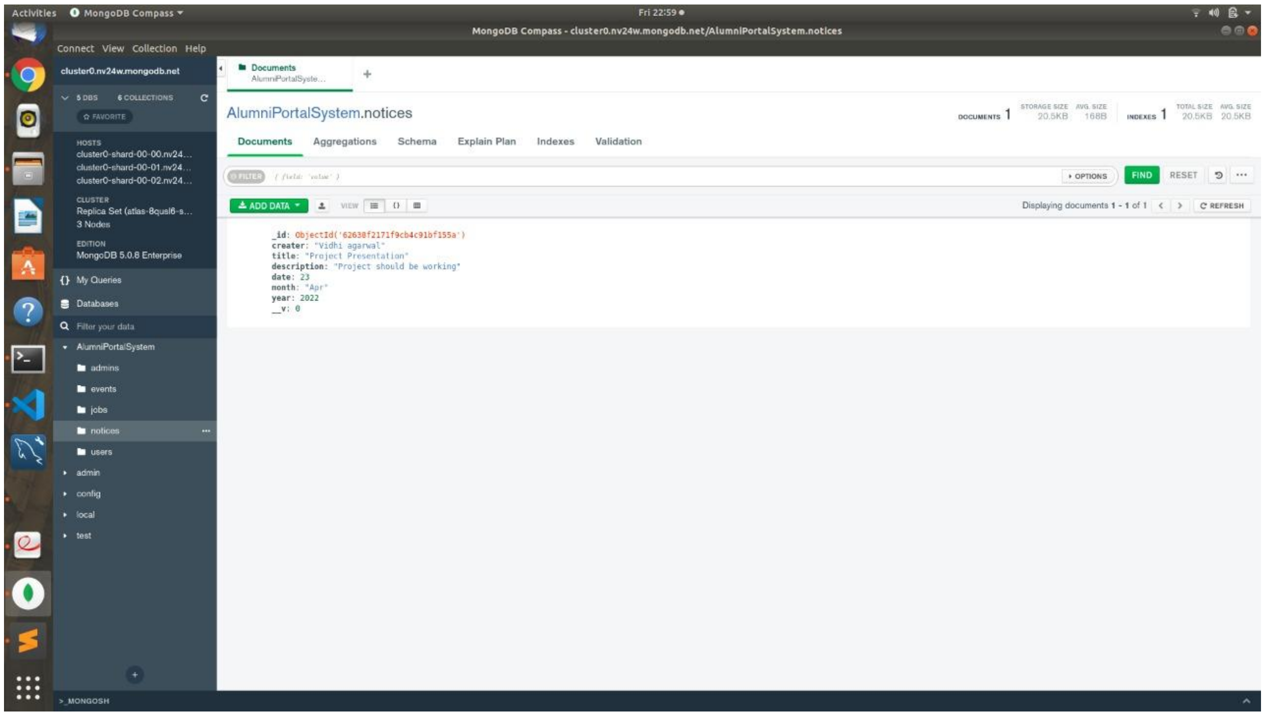
## 3.2 Events Collection:



```
_id: ObjectId('62638edd71f9cb4c91bf1559')
> img: Array
  type: "Educational Event"
  name: "KICCS-D-HACKS 21.4"
  category: "Hackathon"
  websiteURL: "https://www.hackerrank.com/"
  description: "This is a 4 days hackathon, in which student show case their coding sk…"
  time: "11"
  clock: "AM"
  timezone: "US(UTC-8)"
  date: 25
  month: "Apr"
  year: 2022
  __v: 0
```

```
_id: ObjectId('627f3c31eba18f1b4eda6fc3')
> img: Array
  type: "Educational Event"
  name: "Project Exhibition"
  category: "Workshop"
  websiteURL: "https://github.com/kietmcaproject/Miniproject_IV-sem_session-2021-22_E…"
  description: "On 14th of May,there is a project exhibition of MCA Department Of KIET…"
  time: "10"
  clock: "AM"
  timezone: "US(UTC-8)"
  date: 14
  month: "May"
  year: 2022
  __v: 0
```

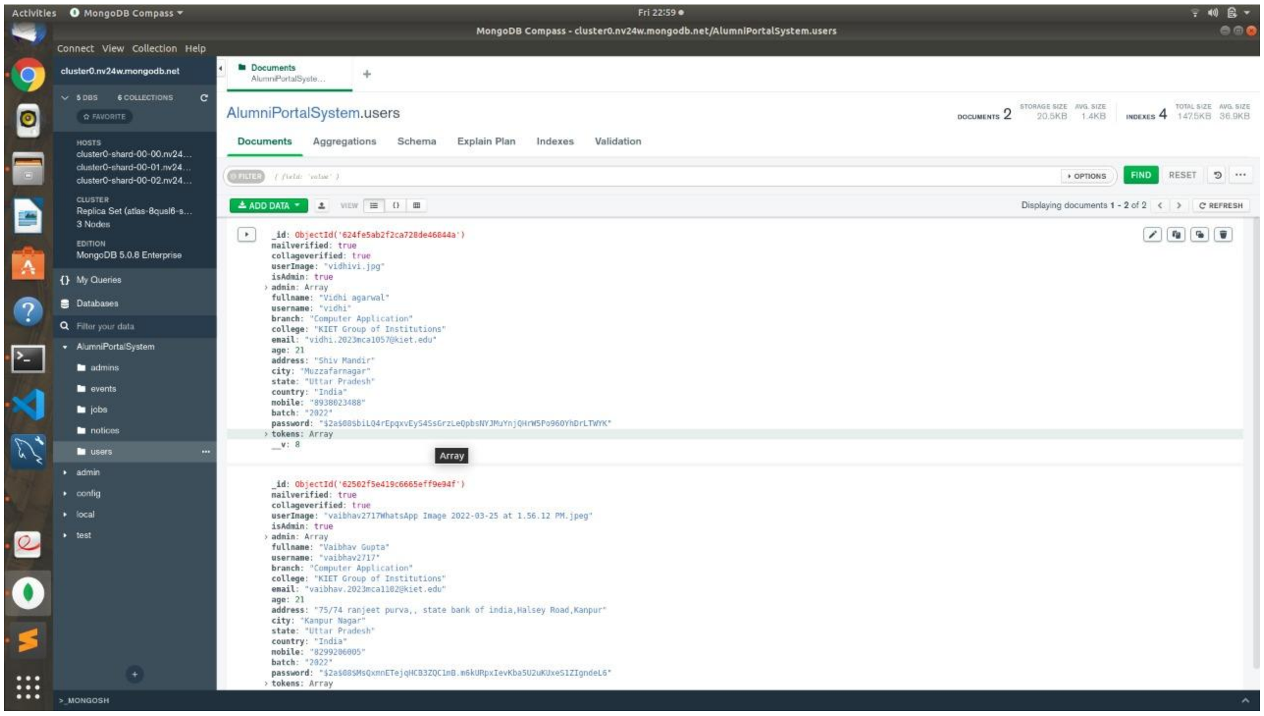
## 3.3 Notices Collection:



MongoDB Compass - cluster0.nv24w.mongodb.net/AlumniPortalSystem.notices
AlumniPortalSystem.notices
Documents
Aggregations
Schema
Explain Plan
Indexes
Validation
ADD DATA
FIND
RESET
Displaying documents 1 - 1 of 1
REFRESH

```
_id: ObjectId('62630f2171f9cb4c91bf155a')
creater: "Vidhi agarwal"
title: "Project Presentation"
description: "Project should be working"
date: 23
month: "Apr"
year: 2022
__v: 0
```

## 3.3 Users Collection:

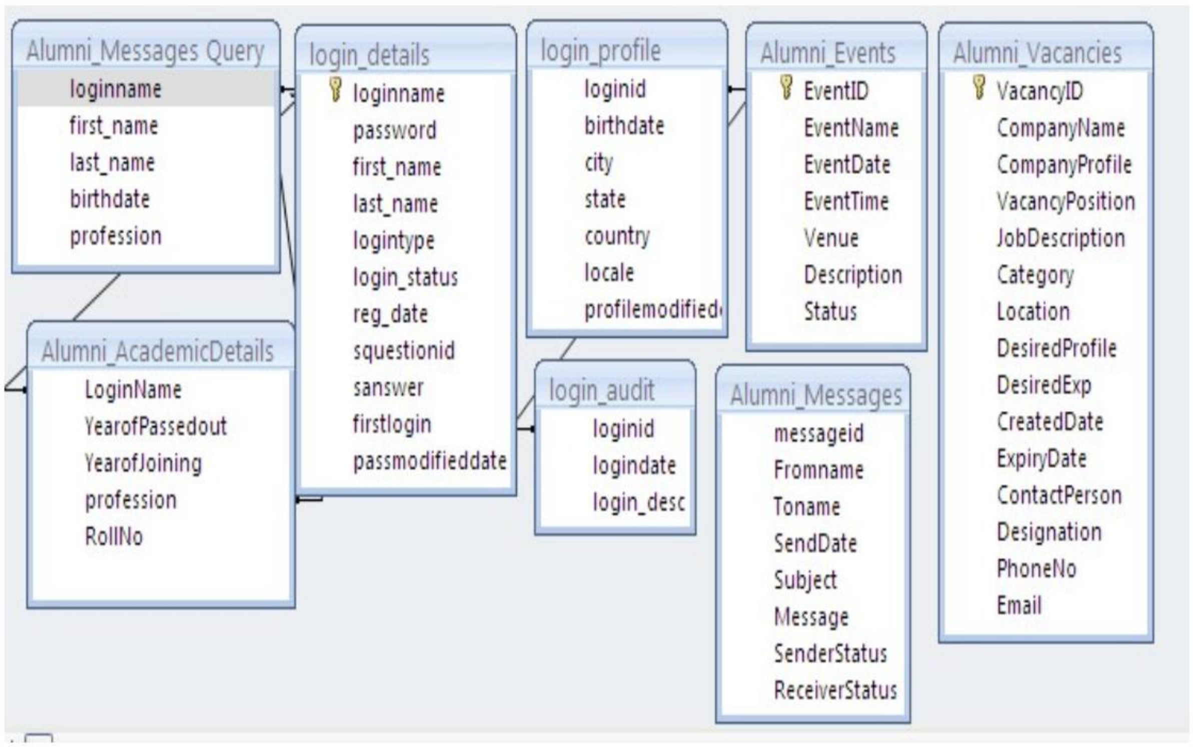


```
_id: ObjectId('624fe5ab2f2ca728de46844a')
mailverified: true
collageverified: true
userImage: "vidhivi.jpg"
isAdmin: true
> admin: Array
fullname: "Vidhi agarwal"
username: "vidhi"
branch: "Computer Application"
college: "KIET Group of Institutions"
email: "vidhi.2023mca1057@kiet.edu"
age: 21
address: "Shiv Mandir"
city: "Muzzafarnagar"
state: "Uttar Pradesh"
country: "India"
mobile: "8938023488"
batch: "2022"
password: "$2a$08$biLQ4rEpqxvEyS4SsGrzLeQpbsNYJMuYnjQHrW5Po960YhDrLTWYK"
> tokens: Array
__v: 8

_id: ObjectId('62502f5e419c6665eff9e94f')
mailverified: true
collageverified: true
userImage: "vaibhav2717WhatsApp Image 2022-03-25 at 1.56.12 PM.jpeg"
isAdmin: true
> admin: Array
fullname: "Vaibhav Gupta"
username: "vaibhav2717"
branch: "Computer Application"
college: "KIET Group of Institutions"
email: "vaibhav.2023mca1102@kiet.edu"
age: 21
address: "75/74 ranjeet purva,, state bank of india,Halsey Road,Kanpur"
city: "Kanpur Nagar"
state: "Uttar Pradesh"
country: "India"
mobile: "8299206005"
batch: "2022"
password: "$2a$08$MsQxmnETejqHCB3ZQC1mB.m6kURpxIevKba5U2uKUxeS1ZIgndeL6"
> tokens: Array
```

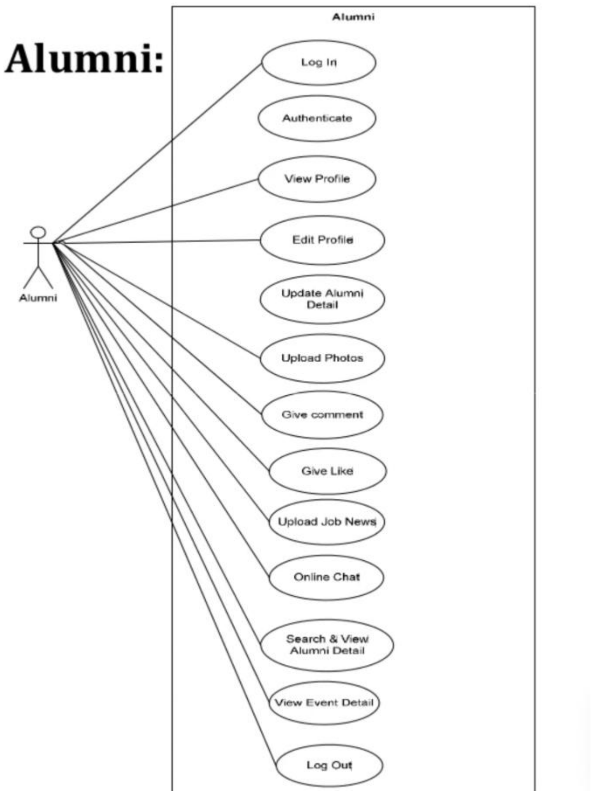
# 4. DIAGRAMS

## 4.1 E-R Diagram

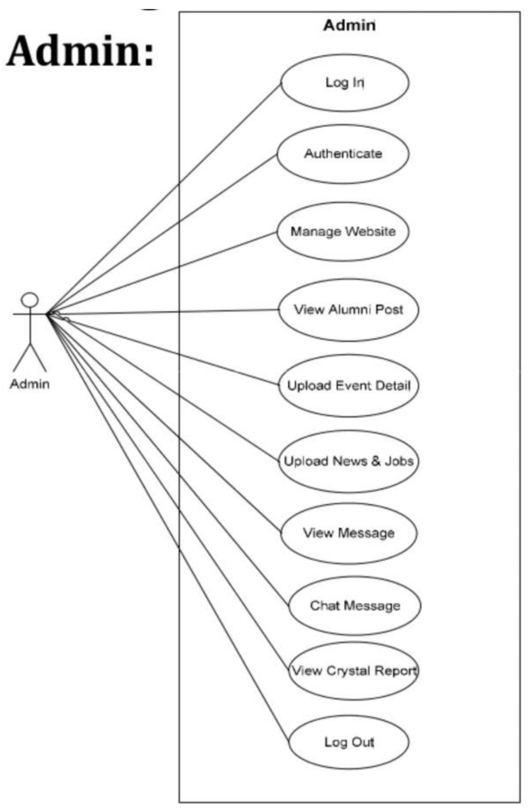


## 4.2 Use-Case Diagrams

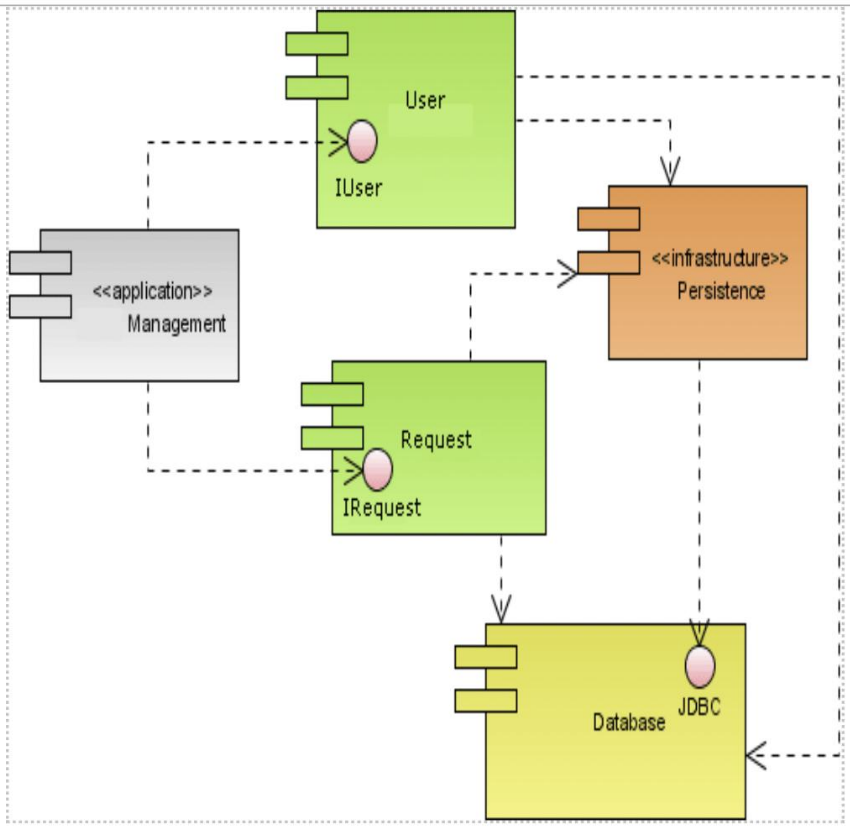
### Use Case Diagram for Alumni:



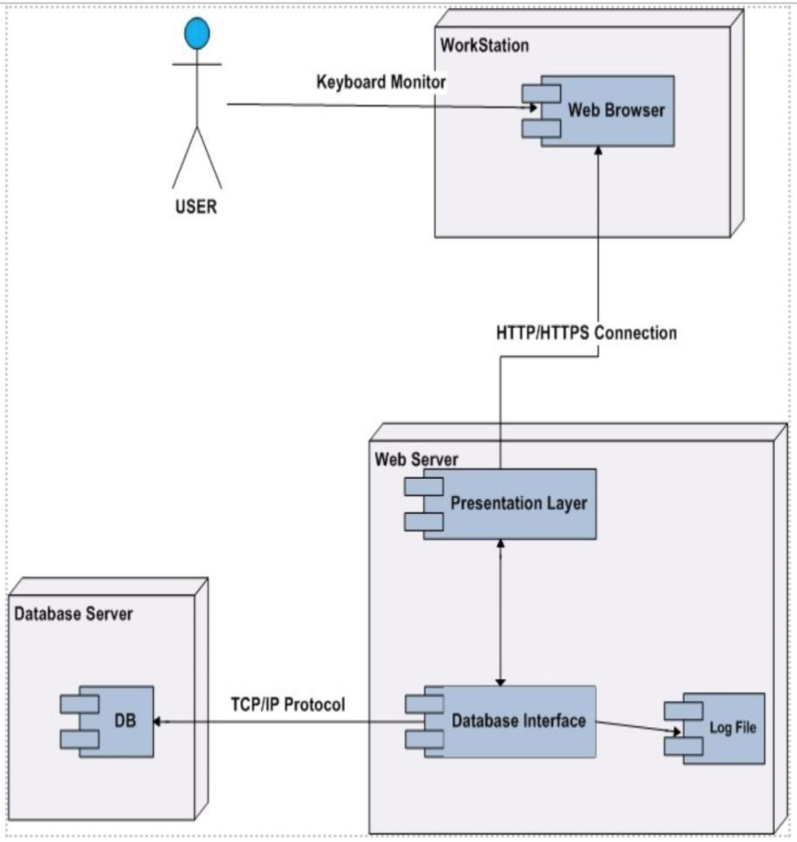
## ‣ Use Case Diagram for Admin:



## 4.3 Component Diagram:

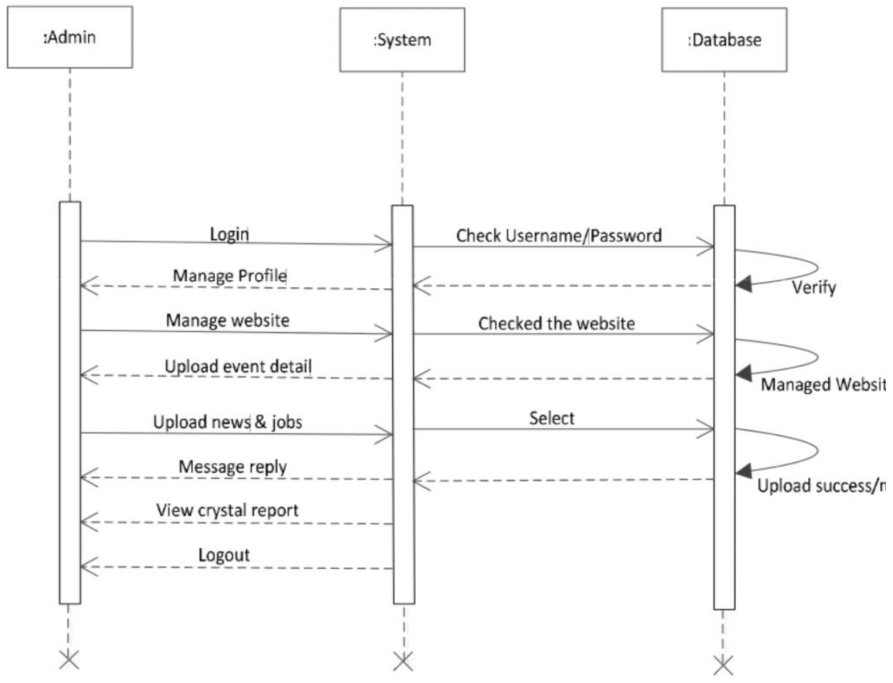


## 4.4 Deployment Diagram:

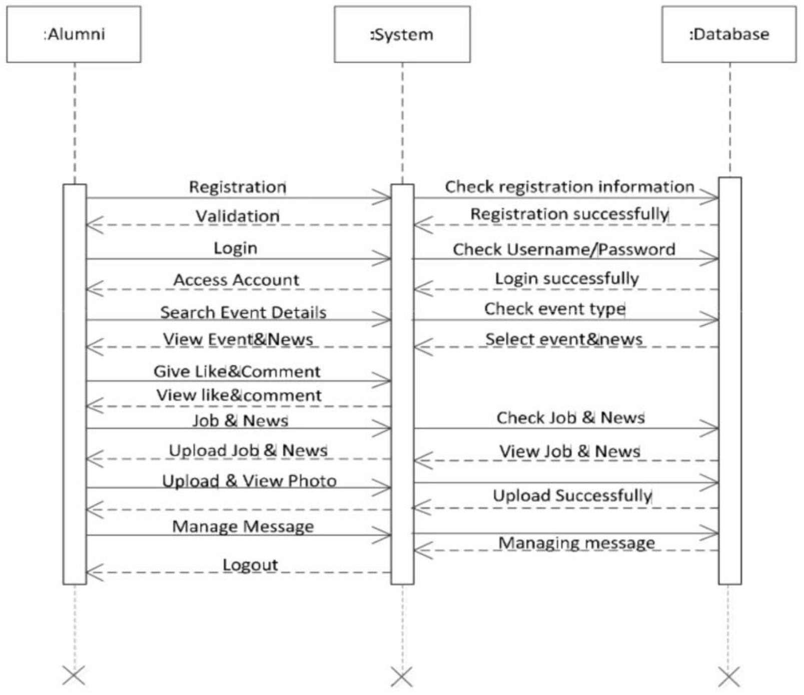


## 4.5 Sequence Diagram

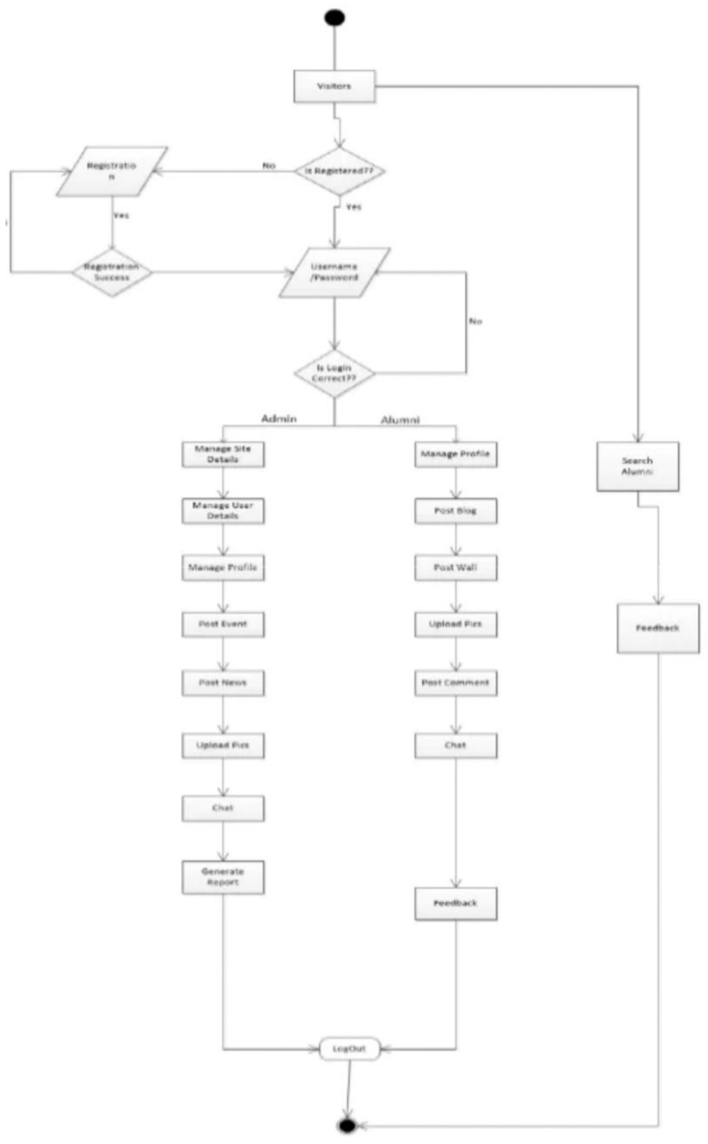
### ‣ Sequence Diagram for Admin:



### ‣ Sequence Diagram for Alumni:

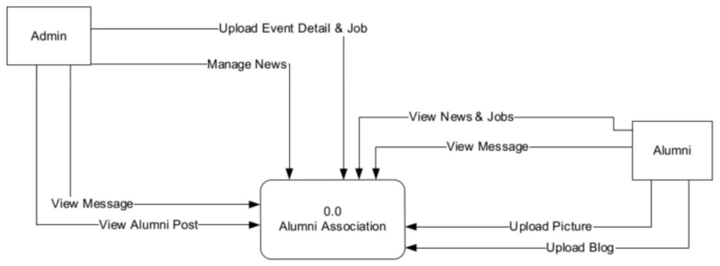


## 4.6 Activity Diagram

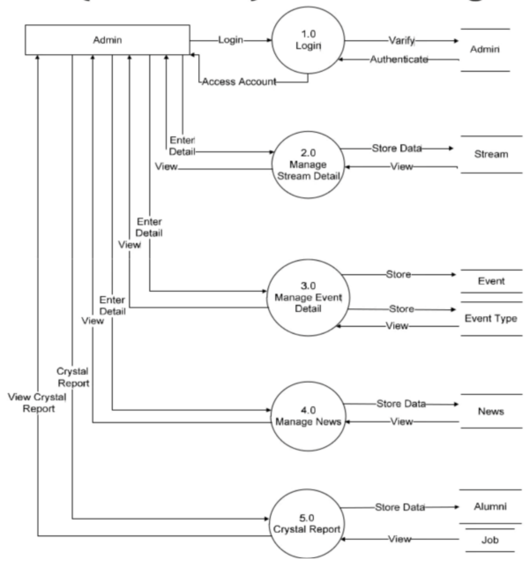


## 4.7 Data Flow Diagrams

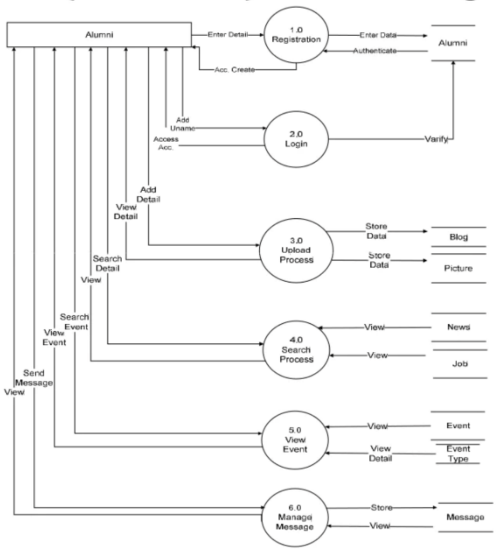
### Context Level (Level 0) Data Flow Diagram:



### ➢ First Level (Admin side) Data Flow Diagram:



### ➢ First Level (Alumni side) Data Flow Diagram:



# 5. SYSTEM DESIGN

## 5.1 Objectives

✓ To bring together all the old students and the faculty of MCA department to share their experiences with each other

✓ To maintain and update the data base of all the alumni of the college and to interact with them

✓ To utilize the rich experiences of old students of the college for the benefit and progress of the present students

✓ To provide guidance to the present students in their endeavour for better employment and higher studies.

✓ To promote the campus placements through the old students working in reputed industries in India and abroad

✓ To get the valuable advices of the Alumni in the overall development of the department

✓ To arrange seminars, debates, workshops and also to arrange cultural and social welfare programs

✓ To arrange teaching and training classes to the students studying in the college and also to the members to upgrade technical and general skills.

✓ To gather and maintain database of student information and to assist the members in securing suitable jobs

✓ To involve the members in the overall development of the MCA department

## 5.2 Technology

### 5.2.1 User Interfaces

The website should work and be tested against IE, Firefox, Google Chrome, and Netscape.

### 5.2.2 Hardware Interfaces

There are no special hardware interface requirements. The website will work good on mobile phone, any specification laptop with internet access, or any other device that have a screen and can open a web browser.

### 5.2.3 Software Interfaces

Software requirements of the system are very nominal, and no other special requirement is there hence it is economically feasible. The website will be able to run on any decent web browser. Technology that will be used in making of the discussion forum are:

**HTML**

The Hypertext Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

**CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

**JavaScript**

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g., functional programming) styles. Read more about JavaScript. 8

**React**

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, react is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

**MongoDB**

MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).

**Express**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework − • Allows to set up middleware to respond to HTTP Requests.
• Defines a routing table which is used to perform different actions based on HTTP Method and URL. Allows to dynamically render HTML Pages based on passing arguments to templates

## 5.3 Functional Description:

a. Login to the system through the first page of the application.

b. Change the password after login to the application.

c. See his/her details and change it.

d. Help from the system.

## 5.4 Module description:

Our project mainly consists of 3 modules. They are;

- Student module
- Admin module
- Alumni module

Totally 3 master pages are used for the modules. Before login the general masterpage comes into view. After login a separate master page for admin will come inview if he logged in; else master page of alumni comes in view. The descriptionof various modules is given below.

### 5.4.1 Student module:

The general module is the starting webpage of our project which gives muchinformation about the college, project associates along with general menus. The menus present in the general webpage are,

**Home**

**About**

**Our team**

**News**

The home menu is the first menu pointed in the general webpage,  the menu home is linked to the homepage of the general master page. The students must enter to thehomepage for login to the alumni module.

The next menu is the about menu which is included in the general master page for viewing the information about the project which is used by the publics as well as students. The about menu is linked to the about webpage.

Our team menu is the next menu, that is present in general master page. This page contains the project associators information and it is provided for publics as well asstudents. Our team menu is linked to our team webpage.

The final menu news is the menu of the general master page. The news sent from the admin are displayed in this page and this page is provided for students as well aspublics. The menu news is linked to the updated news webpage.

### 5.4.2 Admin module:

The admin is the one who looks after all the changes and developments that takes place. There may be one or more than one admin. This module is strictly dedicated to the administrator of the system. In the admin module the following menus comes into view;

- *Alumni registration*
- Alumni list
- *Enquiry*
- News
- *Logout*

The alumni registration menu is the first menu in the admin module the various attributes that are included in registration webpage are the Alumni name, username, register number, password, dob, gender, address, mobile number, email address, pass out year, company name, designation.

There is a button at the end which provides the provision for insertion of values intothe database. After entering the details in the text boxes the insert button is clicked for submission. If there exists any blank text box the system displays an error message to enter the values to the particular field.

The next alumni list menu is the menu of the admin master page this admin list page is used to display the registered alumni students. In this page there is a grid view that will displays the lists of the alumni stored in the database.

The menu enquiry is the next menu of the admin master page this enquiry page is used to send any notification news about the college to the alumni students. The admin sends the college information to the alumni time to time through this webpage. This enquiry menu is linked to the enquiry webpage of the admin module.

The menu news is the next menu of the admin master page, this page is used to see various queries that are passed by the alumni and this page consists the grid view that will displays the queries sent from the alumni. The grid view is linked to the database. This menu news is linked to the news webpage of the admin master page. The final menu is logout menu which is used by the admin to get logged out. This menu redirects the current logout page to homepage of the system. The menu is linked to the general homepage.

## 5.4.3 Alumni module:

The alumni module is used by the alumni. This alumni module displayed in the webpage when the alumni logged in from the homepage of the general module. This module is provided only for the registered members of alumni or students. The alumni module consists following menus, they are;

- *User home*
- Profile setting
- *Enquiry*
- Logout

The menu user home is the menu of the alumni master page this user home page is the page where the alumni may view the news uploaded by the admin. This page consists the grid view that will displays the news sent from the admin. The grid viewis linked to the database. The menu user home is linked to the user homepage of thealumni module.

The profile setting menu is the menu of the alumni master page this profile setting page will be having the same fields of few that are present in the alumni registrationpage of the admin module. But here the alumni will be having a provision to edit and update the changes in his/her profile. There are two buttons edit and update which is helpful in editing updating the profile.

The menu enquiry is the menu of the alumni module, the alumni send the messageto the admin to get information about the college or any student by this page. This page consists two text boxes named subject and description at the end of the page there is a send button this will sends the messages.

The final menu is logout menu which is used by the admin to get logged out. This menu redirects the current logout page to homepage of the system. The menu is linked to the general homepage.

## 5.5 Database:

To move with the times, we need to keep track of information. The success of business enterprise today depends much on how quickly it can retrieve the desired information needed for decision making and routine operation.

Databases are store-houses of information. For retrieving any information, wefall back on databases. Herein comes the role of database management system which helps in systematic organization of information. Amongst the many databases management system available in the market. Access is quite useful and popular because of its unique feature.

A database has many advantages. Many of the useful databases are those which can provide latest information's. The information in a database shouldbe organized in such a way that it is easy to update. And only then, it can quickly provide the information.

Any stored information or a database, whether electronic or manual, helps usby keeping track of our inventories, payroll invoicing or anything else that weconsider important.

### Defining a Database:

A database is an organized collection of related information about a particularsubject or purpose; for example, information regarding the price of a book or the profile of an individual in any organization, consider for instances, the example of a boxful of index cards carrying the names and prices of the booksof a particular author. Every time we juggle through the index cards to set them in an alphabetical order or to look up the price of a particular book, weend up managing the cards.

In computer, a database is like a box containing index cards. The only difference is that instead of index cards, a computer database is filled with **records**. While each index card in the box contains information written in many lines, a record in a database contains all the information related to an item in just one line. Information in a database is stored in **rows** (records) and**columns** (fields).

- *Objects of a relational database:*

**Tables:**

A Table is a collection of information on specific topic. In a table, the

information is stored in rows and columns. We can store different types ofdata in different tables.
Components that need to make up a table are:

➢ *Colum: Each column represents a field. A field stores only a specific category* of information.

- *Row: Each row in a table is called a record and it consists of a number of related*
- *Fields: Each field contains some bits of data about the record.*
- *Domain: The maximum and minimum values a field can have, is called its domain.*
- *Primary key: A primary key refers to one or more fields in a table that uniquely identify each record in the table i.e., it gives a distinct identity to a record.*
- *Foreign key: A foreign key refers to primary key of another table.*

# 6. TESTING

‘Software testing is the process used to help identify the correctness, completeness, Security, and quality of developed computer software. Testing is a process of executing aProgram or application with the intent of finding errors with that in mind, testing can never completely establish the correctness of arbitrary computer software. In other words, testing is Comparisons testing should be distinguished from the separate discipline of software quality Assurance, which encompasses all business process areas, not testing. Type of tests done in our projectare as follows;

## 6.1 Acceptance testing:

User acceptance of a system is the key factor for the success of any system. The System under the consideration is tested for user acceptation by constantly keeping in touch with prospective system users at the time of developing thee making changes whenever required

.

## 6.2 Compatibility Testing:

Testing is to ensure compatibility of an application. Compatibility testing can be Performed manually software developed using these tools and it’s running in windows; in this way the compatibility testing is conducted.

## 6.3 System Testing:

Testing conducted on a complete, integrated system to evaluate the system’s Compliance with its specified requirements. System testing requires no knowledge of the Inner design of the code or logic.

## 6.4 Unit Testing:

Unit testing focuses on the verification of the smallest unit of software design using the unit plans prepared during the design phase of the system, error within theboundary of the module are uncovered in this testing phase, each and sub module were found to be working satisfactorily. In unit testing the entire system is divided into small part or units according to roughly. So many errors in a unit if any are debugged right there, thus saving the time. In this project all the views are tested toensure that they operate correctly.

- In our project forms, views and reports are readable and abbreviations areavoided as much as possible.
- List boxes, radio buttons, check boxes are used so that users can easily selectthe available options.
- Proper validations are done so as to avoid any errors related to database.

## 6.5 Integration testing:

All modules were combined in this testing step. Then the program was testedas a whole. Using integrated test plans prepared in the design phase of the system development as a guide, the integration was carried out. All the errors found in the system were corrected. In our project

- All the modules are checked for correctness and made sure that is works in the way intended after integrating with one another.

- All the links are made sure to point at the right location.

## 6.6 Validation testing:

This test succeeds when the software function in a manner that can be reasonable expected by the user. After the validation test has been conducted one of the following two conditions exists.

- The function or the performance characteristics confirmed to the specifications and are accepted.
- A deviation from specification is uncovered and appropriate messagesare given.
  This testing is used in our project to our project to test each page, form, view and report is checked against the requirement document so as to ensure that it’s desired.

## Validation:

The basic objective of the validation activity is to ensure that the SRS reflects the Actual requirements accurately and clearly. Total three types of validation have beenused in this project as mentioned in the admin module. They are;

### Required Field Validator:

This type of validation is used so that the fields present in the webpage are not left out blank. Some of the fields present in the webpage must be filled without leavingblank. For this purpose, it is used. It is used. It can be used for textbox, radio buttonlist, dropdown list etc. The control to validate that is to which field the control of thevalidator must be given is to be specified.

There will be provision for entering message where relevant error message must be entered. The validation group must be assigned ‘0’ because there must not be any difficulties for moving through the web pages. The buttons (usuallyinsert and update) that will be functioning for the fields present must also be assignedthe validation group.

### Range Validator:

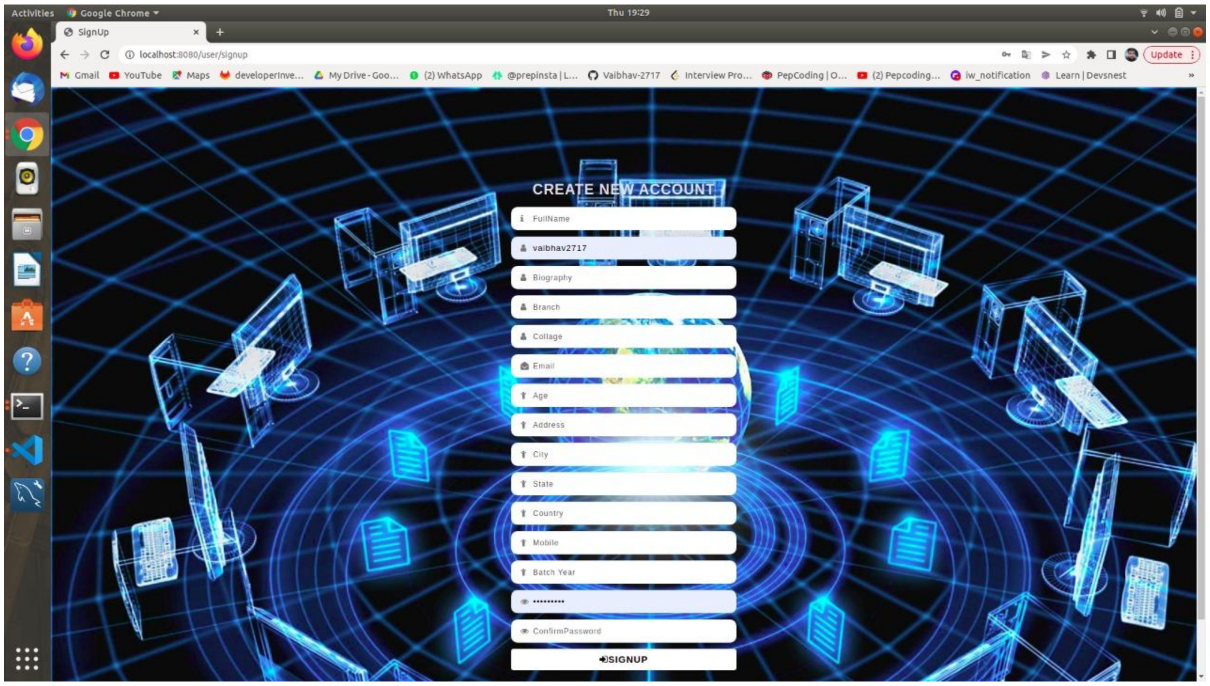
Next type of validator is the range validator which is used for entering the specific values with a particular range. When the range that is specified is crossed, an error message appears on the screen. This validator is usually used for the fields containing the numbers. The control to validate, error message and the validate group must be specified within which the number must be present.

**Regular Expression Validator:**

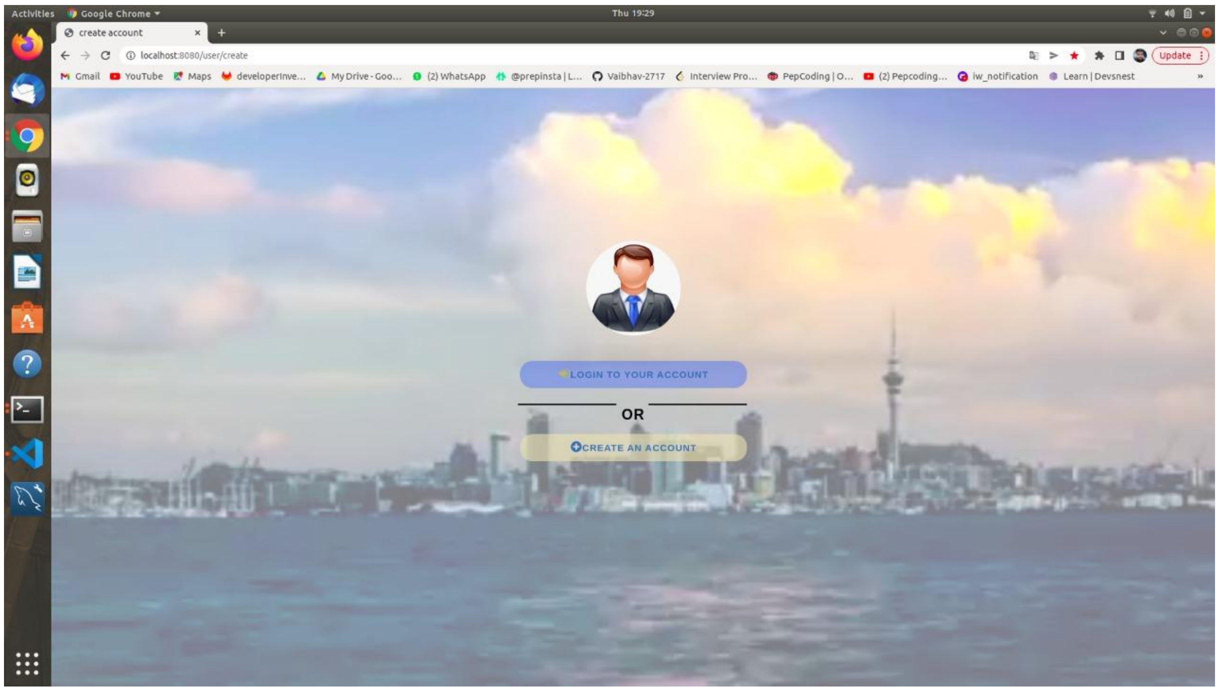
This type of validator is used when a specific type of expression is not entered whena specific type (character, integer, phone number or e-mail id) is not given to the field present, an error message appears on the screen, here also the control to validate, error message and the validate group must be specified. Along with these,the validation expression is specified where the expression for character type, integertype, contact number and e-mail id format is entered.

## 7. OUTPUTS:

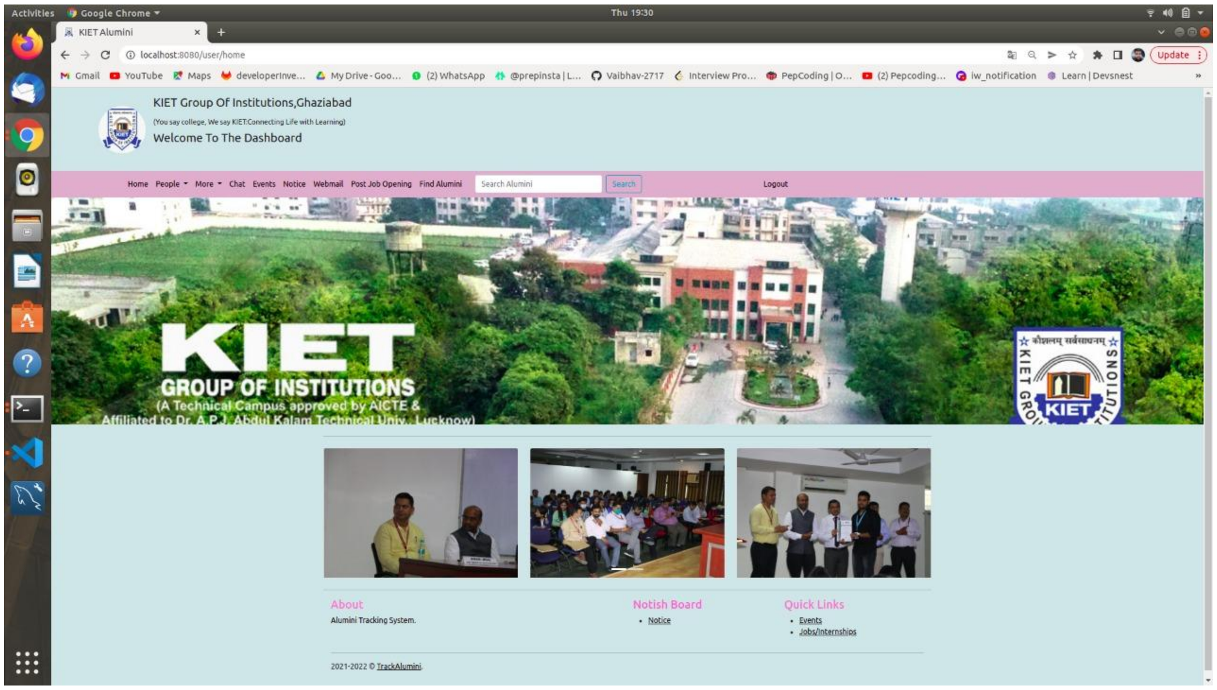
### 7.1 Sign Up



### 7.2 Login



## 7.3 Home Page



KIET Group Of Institutions,Ghaziabad

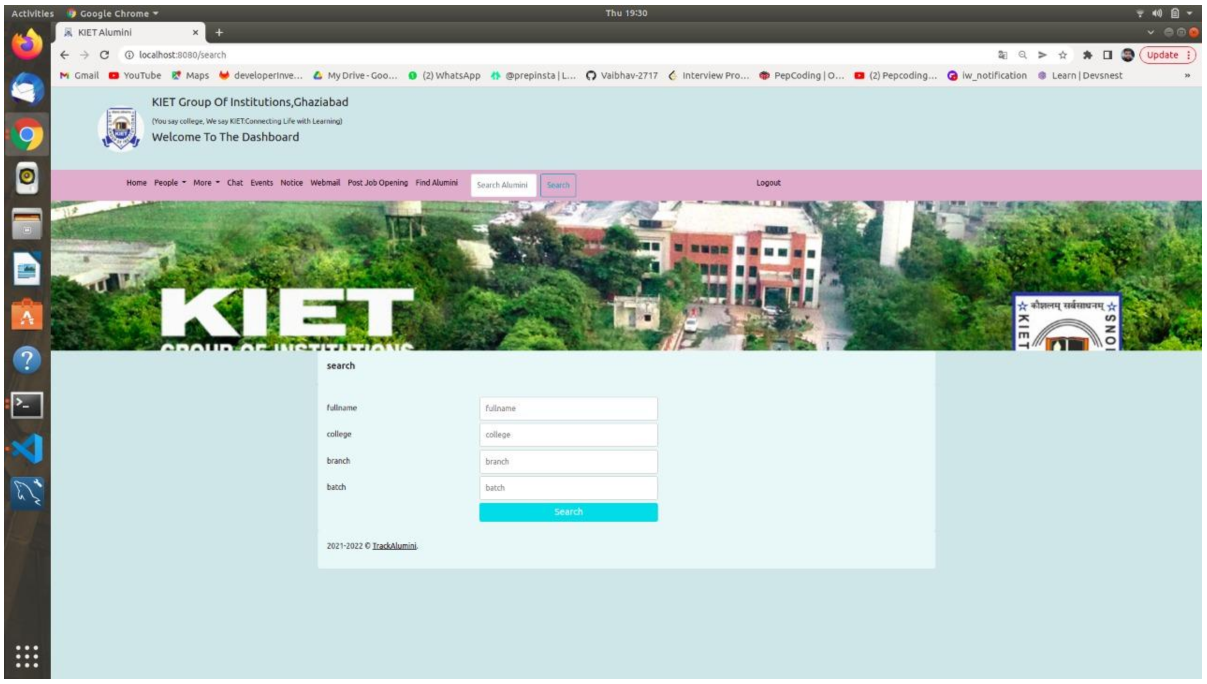
(You say college, We say KIET.Connecting Life with Learning)

Welcome To The Dashboard

Alumini Tracking System.

2021-2022 © TrackAlumini.

## 7.4 Find Alumini



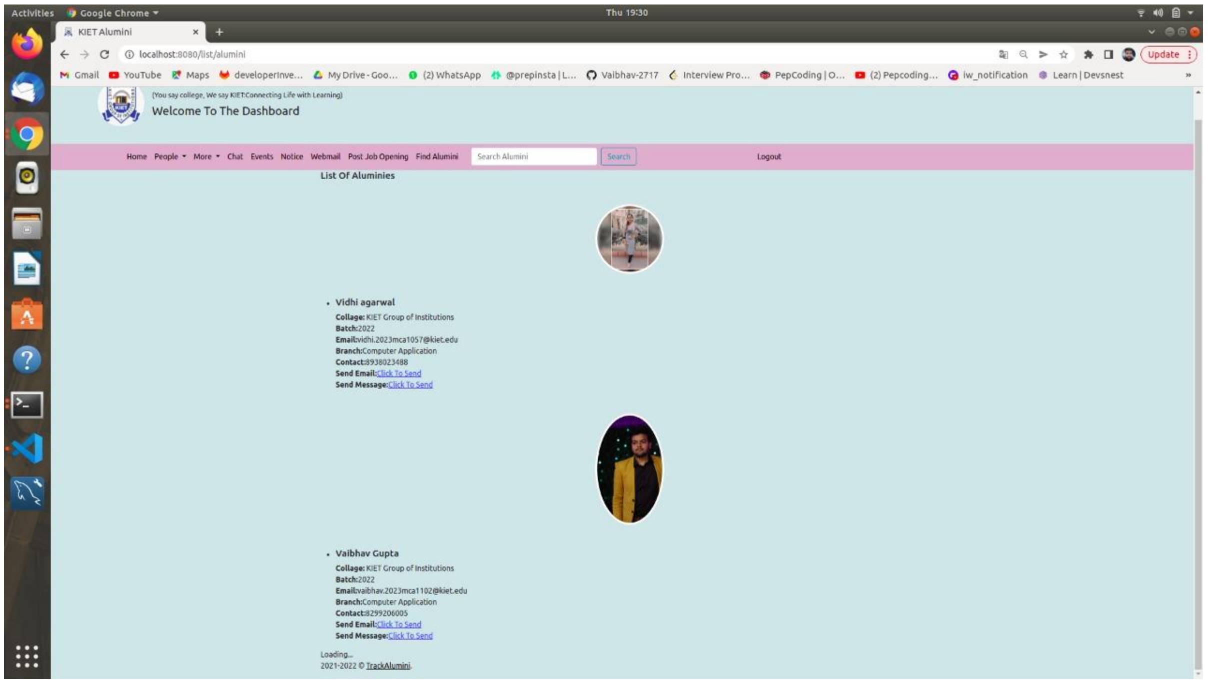
KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET.Connecting Life with Learning)

Welcome To The Dashboard

2021-2022 © TrackAlumini.

## 7.5 All Alumini



localhost:8080/list/alumini

(You say college, We say KIET:Connecting Life with Learning)

Welcome To The Dashboard

Home People More Chat Events Notice Webmail Post Job Opening Find Alumini Search Alumini Search Logout

List Of Aluminies

- **Vidhi agarwal**

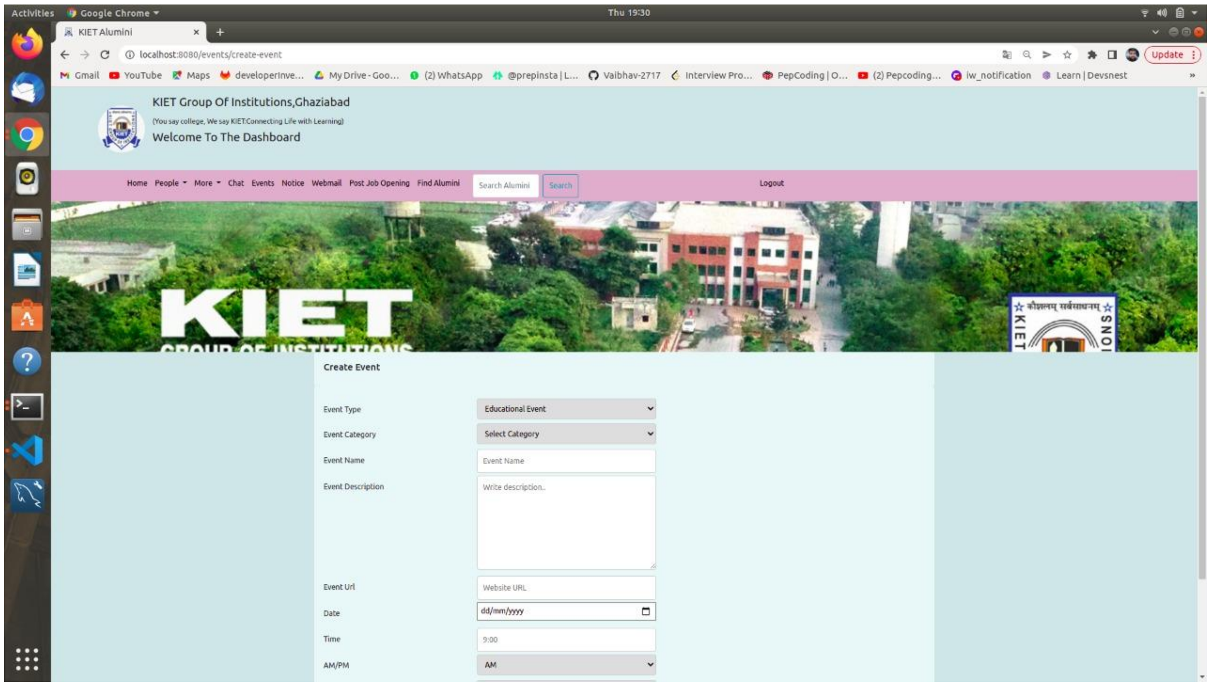
  **Collage:** KIET Group of Institutions
  **Batch:**2022
  **Email:**vidhi.2023mca1057@kiet.edu
  **Branch:**Computer Application
  **Contact:**8938023488
  **Send Email:**Click To Send
  **Send Message:**Click To Send

- **Vaibhav Gupta**

  **Collage:** KIET Group of Institutions
  **Batch:**2022
  **Email:**vaibhav.2023mca1102@kiet.edu
  **Branch:**Computer Application
  **Contact:**8299206005
  **Send Email:**Click To Send
  **Send Message:**Click To Send

Loading...
2021-2022 © TrackAlumini.

## 7.6 Create Event



localhost:8080/events/create-event

KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET:Connecting Life with Learning)

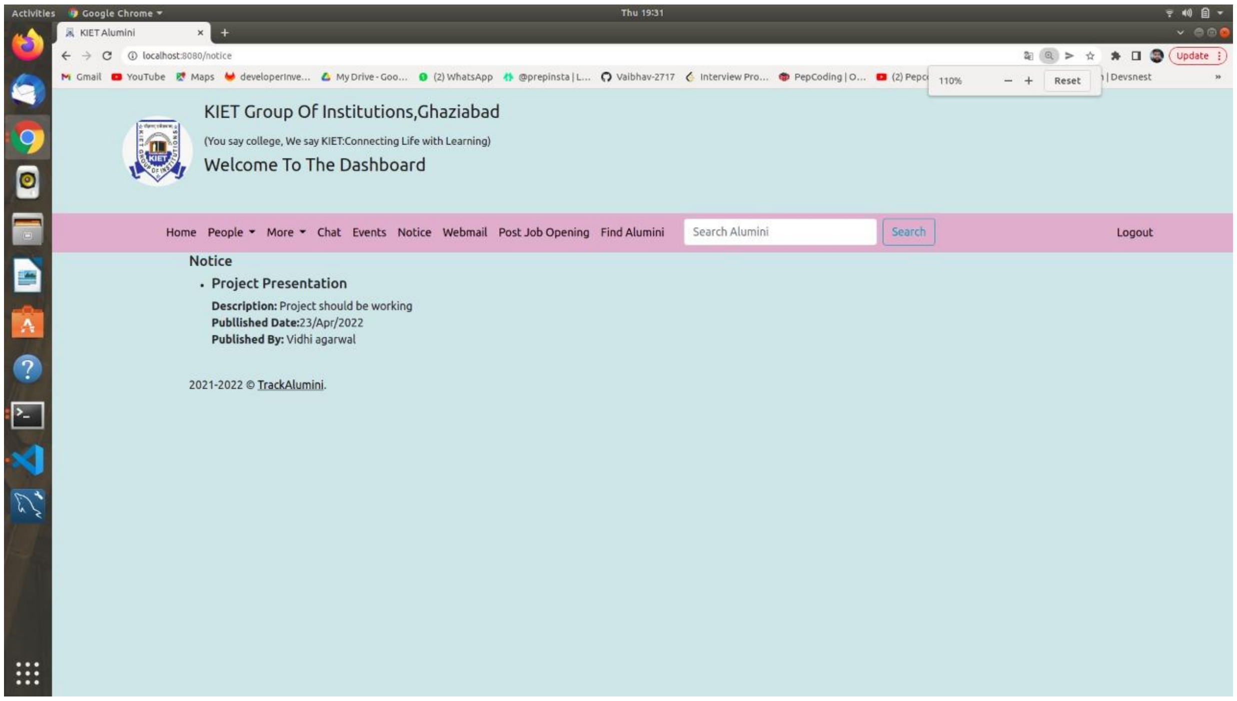
Welcome To The Dashboard

Home People More Chat Events Notice Webmail Post Job Opening Find Alumini Search Alumini Search Logout

Create Event

| | |
|---|---|
| Event Type | Educational Event |
| Event Category | Select Category |
| Event Name | Event Name |
| Event Description | Write description.. |
| Event Url | Website URL |
| Date | dd/mm/yyyy |
| Time | 9:00 |
| AM/PM | AM |

## 7.7 Notice Board



KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET:Connecting Life with Learning)

Welcome To The Dashboard

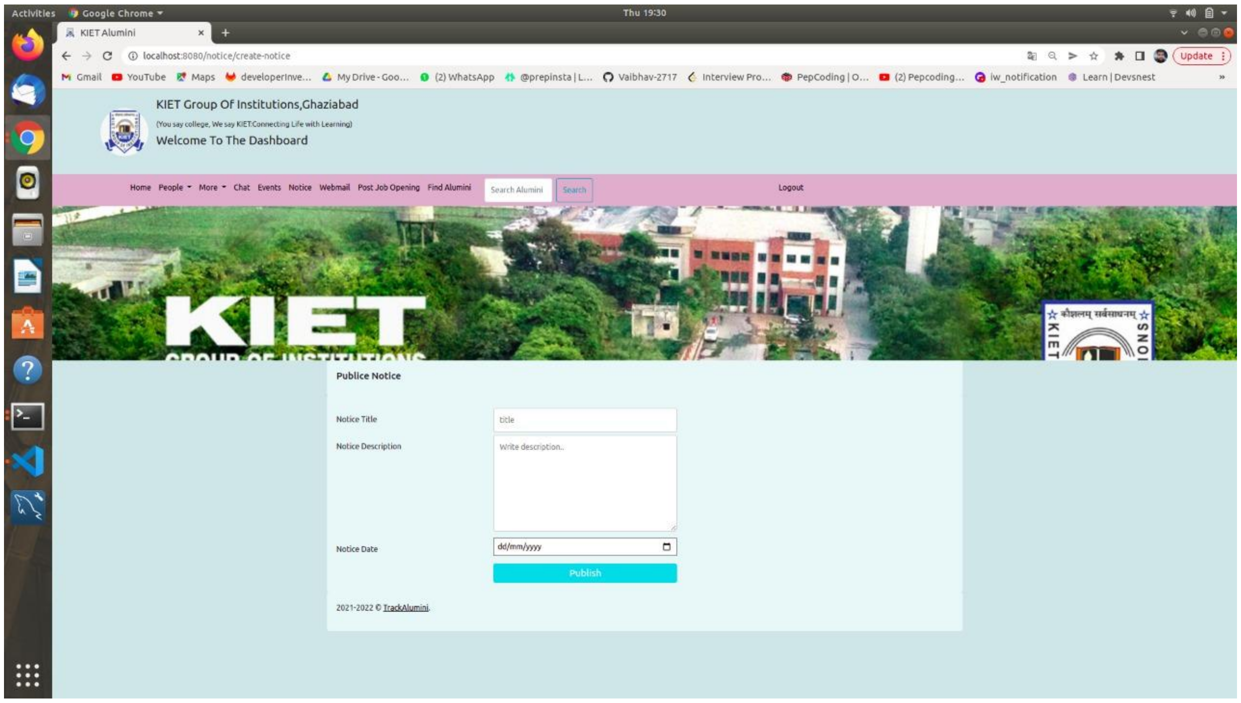
Notice

- Project Presentation

**Description:** Project should be working
**Publlished Date:**23/Apr/2022
**Published By:** Vidhi agarwal

2021-2022 © TrackAlumini.

## 7.8 Create Notice



KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET:Connecting Life with Learning)

Welcome To The Dashboard

Publice Notice

Notice Title: title

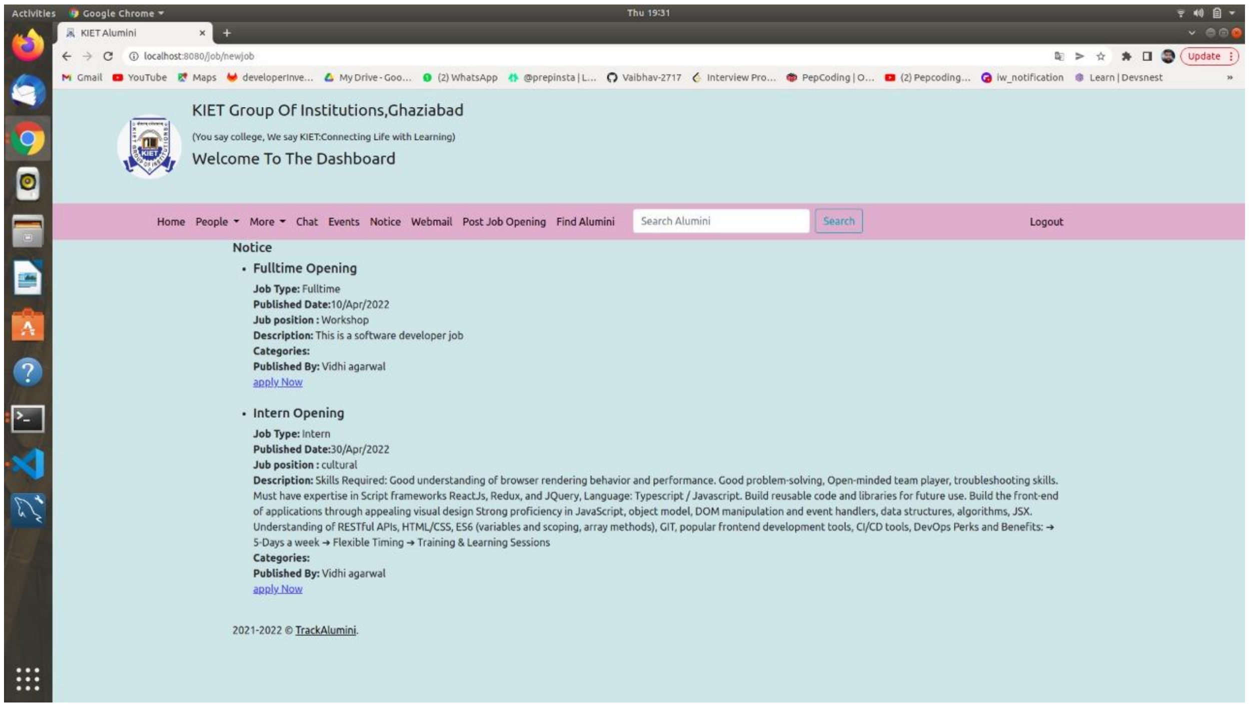
Notice Description: Write description..

Notice Date: dd/mm/yyyy

Publish

2021-2022 © TrackAlumini.

## 7.9 Jobs and Internships



KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET:Connecting Life with Learning)

Welcome To The Dashboard

Notice

- **Fulltime Opening**

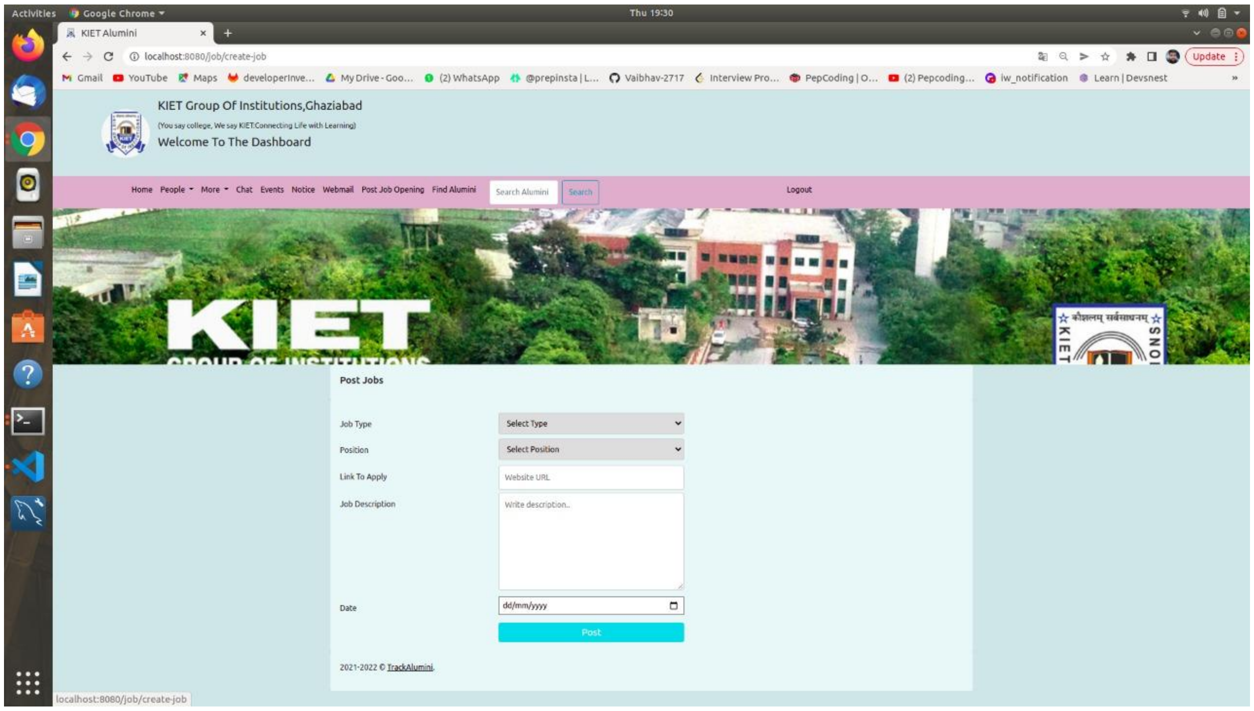
  **Job Type:** Fulltime
  **Published Date:**10/Apr/2022
  **Job position :** Workshop
  **Description:** This is a software developer job
  **Categories:**
  **Published By:** Vidhi agarwal
  apply Now

- **Intern Opening**

  **Job Type:** Intern
  **Published Date:**30/Apr/2022
  **Job position :** cultural
  **Description:** Skills Required: Good understanding of browser rendering behavior and performance. Good problem-solving, Open-minded team player, troubleshooting skills. Must have expertise in Script frameworks ReactJs, Redux, and JQuery, Language: Typescript / Javascript. Build reusable code and libraries for future use. Build the front-end of applications through appealing visual design Strong proficiency in JavaScript, object model, DOM manipulation and event handlers, data structures, algorithms, JSX. Understanding of RESTful APIs, HTML/CSS, ES6 (variables and scoping, array methods), GIT, popular frontend development tools, CI/CD tools, DevOps Perks and Benefits: → 5-Days a week → Flexible Timing → Training & Learning Sessions
  **Categories:**
  **Published By:** Vidhi agarwal
  apply Now

2021-2022 © TrackAlumini.

## 7.10 Create Job



KIET Group Of Institutions,Ghaziabad

(You say college, We say KIET:Connecting Life with Learning)

Welcome To The Dashboard

Post Jobs

| | |
|---|---|
| Job Type | Select Type |
| Position | Select Position |
| Link To Apply | Website URL |
| Job Description | Write description.. |
| Date | dd/mm/yyyy |

2021-2022 © TrackAlumini.

## 8. FUTURE ENHANCEMENTS:

- Attractive User interface will be provided.
- The student can easily send queries without confusion.
- System makes overall project management much easier and flexible.
- Vast amount of data will be stored.
- No risk of data mismanagement at any level.
- High level of security is provided

# 9. MODULE WISE CODES:

## 9.1 Authentication/User:

**Authentiction.js**

```
const jwt=require('jsonwebtoken')
const User=require('../../models/user/user')

const auth= async(req,res,next)=>{
    try{
        const token = req.cookies['auth_token_2']
        const decoded=jwt.verify(token,'thisismyjwtsecret2')

        const user=await User.findOne({_id:decoded._id,'tokens.token':token})
        if(!user){
            throw new Error()
        }
        req.user=user
        req.token=token
        next()
    }catch(e){
        res.redirect('/')

    }
}

module.exports=auth
```

## 9.2 Db:

**Mongoose.js**

```
const mongoose = require("mongoose");
dbConnect()
async function dbConnect(){

    try {
        await
mongoose.connect('mongodb+srv://vaibhav:vaibhav@cluster0.nv24w.mongodb.net/Alum
niPortalSystem' ,
                    {useCreateIndex:true   ,useNewUrlParser:true,useUnifiedTopology:   true
});
        console.log('Mongo DB Connection success')
    } catch (error) {
        console.log('Mongo DB Connection failed')
    }
}

module.exports = mongoose
```

## 9.3 Mail verification:

Mailverification.js

```
const nodemailer=require('nodemailer')
const jwt=require('jsonwebtoken')

const mailverification=(emailid,id)=>{
   console.log('yes')
   const transporter=nodemailer.createTransport({
      service:'gmail',
      auth:{
         user:'vaibhav.2023mca1102@kiet.edu',
         pass:'princessanu@27'
      }
   })

   const token=jwt.sign({_id:id,type:'mailverification'},'thisismyjwtsecret')
    const
url=`https://aluminitrackingsystem.herokuapp.com/user/mailverification?token=${token}`
   const mailOption={
      from:'vaibhav.2023mca1102@kiet.edu',
      to:emailid,
      subject:'Verify your gmail for kiet Mca Alumini user.',

      html:`<p>Thanks For Resistering With Us Click <a href=${url}>here</a> to verify
your email</p>`
   }

   transporter.sendMail(mailOption,(err,data)=>{
      if(err){
         console.log('Mail not Sent for verify')
      }else{
         console.log('Mail sent for verify!')
      }
   })
```

```
}




const resetpassword=async(emailid)=>{
    const transporter=nodemailer.createTransport({
        service:'gmail',
        auth:{
            user:'vaibhav.2023mca1102@kiet.edu',
            pass:'princessanu@27'
        }
    })

    const token=jwt.sign({emailid,type:'resetpassword'},'thisismyjwtsecret2')
   const                                url=`https://aluminitrackingsystem.herokuapp.com/user/reset-
password?token=${token}`
    const mailOption={
        from:'vaibhav.2023mca1102@kiet.edu',
        to:emailid,
        subject:'Reset your passowrd',
        html:`<p>Click <a href=${url}>here</a> to reset your password</p>`
    }



    transporter.sendMail(mailOption,(err,data)=>{
        if(err){
            console.log('Mail not sent for passowrd!!')
        }else{
            console.log('Mail sent for passowrd!!!')
        }
    })
}
//collage verification
const collageverification=(emailid,id,name,batch)=>{
    console.log('yes')
    const transporter=nodemailer.createTransport({
        service:'gmail',
        auth:{
            user:'vaibhav.2023mca1102@kiet.edu',
            pass:'princessanu@27'
        }
    })

    const token=jwt.sign({_id:id,type:'collageverification'},'thisismyjwtsecret')
     const
url=`https://aluminitrackingsystem.herokuapp.com/user/collageverification?token=${toke
n}`
    const mailOption={
        from:'vaibhav.2023mca1102@kiet.edu',
        to:emailid,
        subject:'Verify Alumini',

        html:`<p> student named ${name} batch ${batch } want to register <a
href=${url}>here</a> to verify your alumini</p>`,

    }

    transporter.sendMail(mailOption,(err,data)=>{
        if(err){
            console.log('Mail not Sent for collageverification')
        }else{
            console.log('Mail sent for collageverification!')
        }
    })
}
module.exports={
    mailverification,
    resetpassword,
    collageverification
}
```

## 9.4 Emails:

### 9.4.1 Users

**admin.js**

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

const eventSchema = new Schema({
   name:
   {type:String},

})

module.exports = mongoose.model("Admin" , eventSchema);
```

**user.js**

```
const mongoose = require("mongoose")
const validator =require('validator')
const bcrypt=require('bcryptjs')
const jwt=require('jsonwebtoken')

const userSchema = new mongoose.Schema({
     fullname:{
               type:String,
               required:true,
               trim:true
     },
     username:{
               type:String,
               required:true,
               unique:true,
               trim:true
     },
     email:{
               type:String,
               required:true,
               unique:true,
               validate(value){
                       if(!validator.isEmail(value)){
                               throw new Error('Email not valid')
                       }
               }
     },
     age:{
               type:Number,
               required:true
     },
     password:{
               type:String,
               required:true,
               trim:true,
               validate(value){
                       if(value.length<6){
                               throw new Error()
                       }
               }
     },

     avatar:{
               type:String
     },
     avatarId: String,
     tokens:[{
               token:{
                       type:String,
                       required:true
               }
     }],
     mailverified:{
                 type:Boolean,
                 default:false
     },
     collageverified:{
                 type:Boolean,
                 default:false
     },
     gender:{
                 type:String,
     },
     country:{
                 type:String,
     },



       userImage : {
                 type:String,
                 default:'mak.jpg'
     },



     isAdmin:{
                 type:Boolean,
                 default:false
     },
     address:{
                 type:String,

     },
     city:{
                 type:String,

     },
     mobile:{
                 type:String,
                required:true,
                unique:true,
                trim:true

        },
        branch: String,
        batch: String,
        college: String,
      state: String,
        country: String,
        admin: [
                {
                  type: mongoose.Schema.Types.ObjectId,
                  ref: "Admin"

                }
          ]




});

userSchema.methods.generatingauthtoken=async function(){
        const user=this
        const token=jwt.sign({_id:user._id.toString()},'thisismyjwtsecret2')
        user.tokens=user.tokens.concat({token})
      await user.save()
        return token
}

userSchema.pre('save',async function(next){
        const user=this
        if(user.isModified('password')){
                user.password=await bcrypt.hash(user.password,8)
        }
```

```
    next()
})

module.exports = mongoose.model("User",userSchema);
```

**events.js:**

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

const eventSchema = new Schema({
   type : {type: String},
   name : {type: String},
   category : {type: String },
   img: [{
      type: String
   }
   ],
   websiteURL : {type: String},
   description : {type: String},
   time : {type: String},
   clock : {type: String},
   timezone: {type: String},
   date :{type: Number},
   month: {type:String},
   year: {type: Number}
})

module.exports = mongoose.model("Events" , eventSchema);
```

**jobs.js:**

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

const eventSchema = new Schema({
   type : {type: String},
   position : {type: String},
   category : {type: String },
   jobURL : {type: String},
   description : {type: String},
   date :{type: Number},
   month: {type:String},
   year: {type: Number},
   creater:{type: String}
})

module.exports = mongoose.model("Jobs" , eventSchema);
```

**notice.js**

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

const noticeSchema = new Schema({

   title: String,
   description : {type: String},


   date :{type: Number},
   month: {type:String},
   year: {type: Number},
   creater:{type: String,default:'HOD'}
})

module.exports = mongoose.model("Notice" , noticeSchema);
```

## 9.5 Routes:

**loginroute.js**

```
 const express = require("express");
var formidable = require('formidable');
const router = express.Router();
const crypto=require('crypto');
const path = require("path");



const multer = require("multer");
const User = require("../../models/user/user.js");
const Admin = require("../../models/user/admin.js");




const auth=require('../../authentication/user/auth')
const            {mailverification,resetpassword,collageverification}           =
require("../../emails/mailverification");
const bcryptjs=require('bcryptjs')
const jwt=require('jsonwebtoken');

const bodyParser= require("body-parser")

fs = require('fs-extra')

router.use(bodyParser.urlencoded({extended: true}))




router.get("/signup",(req,res)=>{
        res.render("signup.ejs");
});
```

```
router.get('/create',async(req,res)=>{
	res.render("create.ejs")
})




router.get("/signin",(req,res)=>{
	res.render("index.ejs")
})
router.get("/reset",(req,res)=>{
	res.render("reset.ejs")
})

router.get("/forget",(req,res)=>{
	res.render("forget.ejs")
})






// //<-------------->for photo upload<---------------->













// //<
// //<-------------->TO READ THE USER PROFILE<---------------->
```

```
//<------------> USER profile<---------------->




router.get('/home',auth,async(req,res)=>{
        res.render('Mak.ejs')
})


router.get('/notish',auth,async(req,res)=>{
        res.render('notish.ejs')
})

router.get('/form',auth,async(req,res)=>{
        Admin.find({ }, function(err, alumni) {

      if (err) {
        console.log(err);
        console.log("OOPS there's an error");

      } else {

        alumni.forEach(function(alumni_) {
                                console.log(alumni_.name );


        });

        //res.render("list_alumini.ejs", { alumini: alumni });
  //res.send({ alumni: alumni })
      }

  });

        res.render('form.ejs')
```

```
})

// USER profile

//<------------>TO LOGOUT THE USER<---------------->
router.get('/logout',auth,async(req,res)=>{
        try{
                req.user.tokens=req.user.tokens.filter((token)=>{
                        return req.token!==token.token
                })
                await req.user.save()
                res.redirect('/')
        }catch(e){
                res.redirect('/')
        }
})
//new field

router.post('/admin',auth,async(req,res)=>{
const user = await User.findById(req.user._id)

                  //const Admin=new Admin();
                  new Admin({

                        name: req.body.name

                        }).save(async(err, doc)=>{
                        if(err) res.json(err);
                        else {

                                res.redirect('/user/home')
                        }

                        });
                        //user.admin.push(Admin._id);
                        //await user.save();
                        //console.log(Admin._id)
})



//new field


//Verify the mail id


router.get("/mailverification", async (req, res) => {
        try {
          const token = req.query.token;
          const decode = jwt.verify(token, "thisismyjwtsecret");

          var message = null,
                error = null;
          if (decode.type !== "mailverification") error = "Wrong token";

          const user = await User.findById({ _id: decode._id });
          if (!user) error = "Invalid user";

          if (error === null) {
                user.mailverified = true;
                if(user.username=='vaibhav2717')
                {
                        user.isAdmin = true;
                }
                await user.save();
                message = "Mail verified";
          }
         /* if(user.username=='kanhaiya12')
                        {
                                var fullname=user.username;
                                User.findOneAndUpdate(

                                        {"isAdmin" :fullname },
                                function(err, result){
                                        if(err) {
                                                console.log(err);
                                        }
                                });
                        }*/
                        if(user.collageverified==true && user.mailverified==true)
                {
                        res.redirect('/user/signin');
                }
                else{
                        //res.send('You Verified you email wait for your collage to verify your
identity,Again Click Your gmail link Once your collage verified you')
                        res.render('verify_student')
                }
                        //res.redirect('/user/signin')
                        //res.send('wait For your collage to verify Your identity')

        } catch (e) {
                res.redirect("/user/signup");
        }
 });



 //collage verify
 router.get("/collageverification", async (req, res) => {
        try {
          const token = req.query.token;
          const decode = jwt.verify(token, "thisismyjwtsecret");

          var message = null,
                error = null;
          if (decode.type !== "collageverification") error = "Wrong token";

          const user = await User.findById({ _id: decode._id });
          if (!user) error = "Invalid user";
```

```
            if (error === null) {
                user.collageverified = true;
                if(user.username=='vaibhav2717')
                {
                    user.isAdmin = true;
                }
                await user.save();
                message = "Mail verified by collage";
            }
           /* if(user.username=='kanhaiya12')
                    {
                        var fullname=user.username;
                        User.findOneAndUpdate(

                            {"isAdmin" :fullname },
                        function(err, result){
                            if(err) {
                                console.log(err);
                            }
                        });
                    }*/
                    //res.redirect('/user/signin')
                    //res.send('Thank You,You verified You alumini')
                    res.render('verify_collage')

           } catch (e) {
                res.redirect("/user/signup");
           }
  });

//=========================================
//
//                     POST  ROUTES      (CHANGES  WITH  DATABASE  AND
AUTHORIZATION)
//
//=========================================
//server.js




router.get("/postme" ,(req,res) =>{
        res.render('uploadYourpic')
  })
router.post('/postme',auth, function(req, res) {
        var form =new formidable.IncomingForm();
        form.parse(req);
        let reqPath= path.join(__dirname, '../../../');
        let newfilename;
        form.on('fileBegin', function(name, file){
                file.path = reqPath+ 'public/upload/'+ req.user.username + file.name;
                newfilename= req.user.username+ file.name;
        });
        form.on('file', function(name, file) {
                User.findOneAndUpdate({
                        username: req.user.username
                },
                {
                        'userImage': newfilename
                },
                function(err, result){
                        if(err) {
                                console.log(err);
                        }
                });
        });
        req.flash('success_msg', 'Your profile picture has been uploaded');
        res.redirect('/user/home')
});

router.post("/signup",async(req,res)=>{
        //let test=new Object
        try{
                console.log("i m here");
                const email=await User.findOne({email:req.body.email})
                const username=await User.findOne({username:req.body.username})

                console.log('yes')
                if(email)
                {
                        console.log('yes')
                        req.flash('error','Email is already register!')
                        res.redirect('/user/signup')
                }
                else if(username)
                {
                        console.log('yes')
                        req.flash('error','Username is already register!')

                        res.redirect('/user/signup')
                }
                else
                {
                        console.log('yes')
                        const user=new User(req.body)
                        await user.save()

                        mailverification(user.email, user._id);
                        collageverification('vaibhav.2023mca1102@kiet.edu',
user._id,user.fullname,user.batch);
                        req.flash('error','Email is sent Verify email to login!')
                        res.redirect('/user/signup')



                }
        }catch(e){
                console.log(e)
                res.send(e)
        }
```

```
});




//   CORRECT IT!!!!!!!!
router.post('/signin',async (req,res)=>{
        try{
                const                                                              user=await
User.findOne({username:req.body.username,mobile:req.body.mobile})
                //const user1=await User.findOne({mobile:req.body.mobile})
                //console.log(user1)
                if(!user){
                        req.flash('error','Username/mobile is not valid')
                        res.redirect('/user/signin')
                }

                else
                {
                        const                                                   isMatch=await
bcryptjs.compare(req.body.password,user.password)
                        if(!isMatch){
                                req.flash('error','Invalid password')
                                res.redirect('/user/signin')
                        }
                        else
                        {
                                const token=await user.generatingauthtoken()
                                res.cookie('auth_token_2',token)

                        console.log(user.username)
                        if(user.isAdmin == true)
                        {
                                res.redirect("/user/home");
                        }
                        else{
                                console.log('you are not an admin')
```

```
                                        //res.redirect("/otp");
                                        res.redirect("/user/home");
                                }

                                }
                        }




                }catch(e){
                        res.send('server error')
                }
        })

        router.post('/forget-password',async(req,res)=>{
                try{
                        var message=null,error=null
                        const user=await User.findOne({email:req.body.email})
                        if(user === null)
                        error='Email is not registered'

                        if(error === null)
                        {
                                resetpassword(req.body.email);
                                message='Check you emailid and reset your password.'
                        }
                        res.redirect("/user/forget")
                        console.log('reset link send');
                }catch(e){
                        //res.render('message-reset.ejs',{message:null,error:'Server error'})
                        console.log('something is wrong');
                }
        })
        //My new routes

        router.get("/reset-password", async (req, res) => {
            res.render("reset")
})
//my new routes

router.post("/reset-password", async (req, res) => {
            try {
              const token = req.body.token;
              const decode = jwt.verify(token, "thisismyjwtsecret2");

              var message = null,
                        error = null;
              if (decode.type !== "resetpassword") error = "Wrong token!!";

              if (error === null) {
                        const user = await User.findOne({ email: decode.emailid });
                        const password = req.body.password;
                        user.password = password;
                        await user.save();
                        message = "Password changed sucessfully";
              }
              //res.render('reset',{token:token})
              res.redirect('/user/signin')
            } catch (e) {
                        res.send('oops')
            }
  });

  router.get('/alumini',auth,async(req,res)=>{
            let newUser = {};

            newUser = (await User.findById(req.user._id))
            res.render('alumini.ejs',{user:newUser})
})
```

```
/*router.get('/search',auth,async(req,res)=>{
    try{
        const all_products_ids=await User.find({})
        const product_ids=all_products_ids[0][req.query.tag]
        if(product_ids !== 'undefined')
        {
            var all_products=[]
            for(var i=0;i<product_ids.length;i++)
            {
                console.log(product_ids[i])
                if(product_ids[i].user_id)
                {
                    const product=await User.findById({_id:product_ids[i].user_id})
                    all_products=[...all_products,product]
                }
            }
            res.json({products:all_products})
            //res.render('search.ejs',{data:all_products.reverse(),user:req.user})
        }
    }catch(e){
        res.send(e)
    }
})
router.post('/filters',async(req,res)=>{
    try {
        var ans=[];
        const filter_tags=await User.find({});
        const all_tags=filter_tags[0];
        console.log(all_tags);
        const filters=req.body.filters;//to be applied tags
        console.log(filters);
        const products_for_first_filter=all_tags[filters[0][0]];
        products_for_first_filter.forEach(async (user_id)=>{
            var prod=await User.findById({_id:user_id});
            for(var i=0;i<filters.length;i++)               // looping through tags (color,size)
            {
                for(var  j=0;j<filters[i].length;j++)                           //  looping  through  tags  value
(blue,red,black)
                {
                    if(prod[filters[i]].equals(filters[i][j]))
                    ans.push(prod);
                }
            }
        })
        console.log(ans);
        res.send(ans);

    } catch (error) {
        console.log(error);
    }
})*/
//create search get and pot routes

module.exports = router;
```

**searchroute.js:**

```
const express = require("express");
var formidable = require('formidable');
const router = express.Router();
const crypto=require('crypto');
const path = require("path");

const multer = require("multer");
const tags = require("../../models/user/query.js");
const bodyParser= require("body-parser")
const User = require("../../models/user/user.js");

fs = require('fs-extra')
const mak=User.find({})



console.log(mak)
//tags.push()
router.get("/search",(req,res)=>{
        res.render("search.ejs")
})
module.exports = router;
```

**otp.js:**

```
 const express = require("express")
const app =  express();
const router = express.Router();
//var exphbs  = require('express-handlebars');
var bodyParser = require('body-parser');

// Load configuration from .env file




// Load and initialize MesageBird SDK
var messagebird = require('messagebird')('FB3bcFcJkII2I1kYIEdP1AiBC');
//var messagebird = require('messagebird')('a6ThMKcuPyw00AXQP2qva3ZKk');
router.use(bodyParser.urlencoded({ extended : true }));

// Display page to ask the user for their phone number
router.get('/', function(req, res) {
    res.render('step1');
});

// Handle phone number submission
router.post('/step2', function(req, res) {
    var number = req.body.number;

    // Make request to Verify API
    messagebird.verify.create(number, {
        originator : 'Code',
        template : 'Your verification code is %token.'
    }, function (err, response) {
        if (err) {
            // Request has failed
            console.log(err);
            res.render('step1', {
                error : err.errors[0].description
            });
        } else {
            // Request was successful
            console.log(response);
            res.render('step2', {
                id : response.id
            });
        }
    })
});

// Verify whether the token is correct
router.post('/step3', function(req, res) {
    var id = req.body.id;
    var token = req.body.token;

    // Make request to Verify API
    messagebird.verify.verify(id, token, function(err, response) {
        if (err) {
            // Verification has failed
            console.log(err);
            res.render('step2', {
                error: err.errors[0].description,
                id : id
            });
        } else {
            // Verification was successful
            console.log(response);
            res.redirect('/user/home');
        }
    })
});
module.exports = router;
```

**Sendmessages.js**

```
const express = require("express")
const app =  express();
const router = express.Router();
const mongoose = require("mongoose");
const User = require("../models/user/user.js");

const fast2sms = require('fast-two-sms')





const fs = require('fs')
const path = require('path')



app.use(express.json())

router.get("/alumni/:id/message", function(req, res) {
    User.findById(req.params.id, function(err, foundalumni) {
        if (err) {
            console.log(err);
        } else {
            //render show template with that campground
            console.log(foundalumni);
            res.render("message", {
                alumni: foundalumni
            });


        }
    });


});
router.post("/alumni/:id/message", function(req, res) {
    var message = req.body.text;


    User.findById(req.params.id, function(err, foundalumni) {
        if (err) {
            console.log(err);
        } else {
            res.redirect("/search/");
            receiver = foundalumni.mobile;
           var options = {authorization :
"e68rANJXD3ITGQsHlaP79ytuEWdbf5ZwMFYOxVvk0jRKzL1Bc20cX4smN1Z6b7v2THe
o5nLRgpkOiP9W" , message : message ,  numbers : [receiver]} ;
                fast2sms.sendMessage(options) .then(response=>{
      console.log(response)
    }) .catch((error) => {
                    console.log(error);
                }); //Asynchronous Function.




        }
    })


});

module.exports = router;

//
PYohrtHOD2wNcjubslMBWRq0gLiUQGIKS1zVeFd946C8nxv35AIN325JKHhSWnuariEM
fLcCtvesR0XT
```

## Profile

```
// for profile pic upload
$(document).ready(function() {

    var readURL = function(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();

            reader.onload = function (e) {
                $('.profile-pic').attr('src', e.target.result);
            }

            reader.readAsDataURL(input.files[0]);
        }
    }

    $(".file-upload").on('change', function(){
        readURL(this);
    });

    $(".upload-button").on('click', function() {
       $(".file-upload").click();
    });
});
// for message
$(document).ready(function(){
    $('[data-toggle="tooltip"]').tooltip();
});
// for photo upload
$('.upload-wrap input[type=file]').change(function(){
    var id = $(this).attr("id");
    var newimage = new FileReader();
    newimage.readAsDataURL(this.files[0]);
    newimage.onload = function(e){
      $('.uploadpreview.' + id ).css('background-image', 'url(' + e.target.result + ')' );
    }
  });
```

**Sign Up**

```
$(document).ready(function() {


    var readURL = function(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();


            reader.onload = function (e) {
                $('.profile-pic').attr('src', e.target.result);
            }


            reader.readAsDataURL(input.files[0]);
        }
    }


    $(".file-upload").on('change', function(){
        readURL(this);
    });


    $(".upload-button").on('click', function() {
       $(".file-upload").click();
    });
});


var password = document.getElementById("password")
  , confirm_password = document.getElementById("confirm_password");


function validatePassword(){
  if(password.length != 6){
    confirm_password.setCustomValidity("Passwords length must be 6");
  }else if(password.value != confirm_password.value) {
    confirm_password.setCustomValidity("Passwords Don't Match");
  } else {
    confirm_password.setCustomValidity('');
  }
}

password.onchange = validatePassword;
confirm_password.onkeyup = validatePassword;
```

## JSON File

```
{
  "name": "walson-socailmedia_nj-kanhaiya",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "nodemon src/app.js",
    "dev": "nodemon src/app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "accepts": "^1.3.7",
    "array-flatten": "^3.0.0",
    "axios": "^0.21.1",
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "cloudinary": "^1.23.0",
    "connect-flash": "^0.1.1",
    "content-disposition": "^0.5.3",
    "content-type": "^1.0.4",
    "cookie-parser": "^1.4.5",
    "dotenv": "^8.2.0",
    "ejs": "^3.1.5",
    "express": "^4.17.1",
    "express-session": "^1.17.1",
    "fast-two-sms": "^2.1.0",
    "formidable": "^1.2.2",
    "fs-extra": "^9.0.1",
    "gridfs-stream": "^1.1.1",
    "hits": "^2.0.2",
    "http": "0.0.1-security",
    "jshint": "^2.12.0",
    "jsonwebtoken": "^8.5.1",
    "messagebird": "^3.5.0",
    "method-override": "^3.0.0",
    "moment": "^2.29.1",
    "mongoose": "^5.10.17",
    "multer": "^1.4.2",
    "multer-gridfs-storage": "^4.2.0",
    "nodemailer": "^6.4.16",
    "nodemon": "^2.0.15",
    "path": "^0.12.7",
    "pusher": "^4.0.2",
    "socket.io": "^3.0.4",
    "textmagic-rest-client": "^1.0.11",
    "twilio": "^3.54.0",
    "uuid": "^8.3.1",
    "uuidv1": "^1.6.14",
    "validator": "^13.1.17"
  }
}
```

## 10. REFERENCES

1. Students on placement: a comparative study

2. Job search on the internet and its outcome

3. A Study on Job Description and its Effect on Employee Performance: Case of Some Selected Manufacturing Organizations in the City of Pune, India

4. A Quick Introduction to Version Control with Git and GitHub

5. A Study in Relationship Orientation and Prioritization of Alumni Association Preferences with College Seniors in Higher Education

6. An Online Job portal management system

7. Node.js Challenges in Implementation

8. Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development

9. New technologies for web development

10. Analysis of Campus Recruitment Parameters in an Indian Context