



**L** OVELY  
**P** ROFESSIONAL  
**U** NIVERSITY

---

## **SCHOOL OF COMPUTER APPLICATIONS**

Report on Capstone Project

# **E-Course Management System**

### **Supervised By**

Mr. Balraj Kumar

Assistant Professor & Assistant Dean

### **Submitted By**

Abhijeet Yadav - 11800671

Sachin Kumar - 11811027

Vivek Prasad - 11800669

Rohit Kumar Chauhan - 11808466

## **Table of Content**

1. Introduction
2. Problem Identification
  - 2.1 Formal Problem Definition and its Description
3. Feasibility Study
  - 3.1 Technical Feasibility
  - 3.2 Schedule Feasibility
  - 3.3 Legal Feasibility
  - 3.4 Resource Feasibility
  - 3.5 Economic Feasibility
4. Outcome Statement of the Project
5. Software Requirement Specifications
  - 5.1 Purpose
  - 5.2 Scope
  - 5.3 External Interface Requirements
6. System Design
  - 6.1 Objective of Data Flow Diagram
  - 6.2 Zero Level DFD of E-Course Management System
  - 6.3 First Level DFD of E-Course Management System
  - 6.4 Second Level DFD of E-Course Management System

## 7. User Case Diagram

7.1 User Authentication and Roles

7.2 Course Review/Feedback

7.3 Assignment Upload

7.4 Assigning of Grades

## 8. Database Design

8.1 User

8.2 Profile

8.3 Courses

8.4 Event

8.4 Relationship between Entities

# 1. Introduction

The aim of this capstone project is to create a full stack website which will provide a fully working online solution where users/ students can get information about various courses being offered by Engineer's Gurukul and get enrolled in any of the courses/ events which interests them.

Enrolled students/ mentors would be able login to their account, check their course/Event schedule, upload/ download assignments, provide reviews on the courses, post their queries and complaints, edit their profile, and receive notifications on any important announcements.

Admin Panel will have the ability to check and manage all the events and courses being offered as of now. Admin can add new courses/ events, update existing one or delete the one not available anymore. Admin can also check the courses purchased and conformation of payments for the courses. He/ She can also post announcements on all the happenings and upcoming events.

This website will provide a very user friendly, fully engaging experience to the users and is going to be fully responsive on all devices. Website will be using the latest and well established technologies currently available in the market.

## 2. Problem Identification

Engineer's Gurukul is an Educational Institute which provides training and courses to college students on various computer science subjects such as web development, cyber security, machine learning, IOT etc.

In every summer and winter training sessions, Engineer's Gurukul gets admission with hundreds of new audiences and hence they have to manually handle all the tasks such as

- Keeping track of students being enrolled
- Payment status of enrolled students
- Schedule of different batches of Courses or Events
- Conducting of the MCQ's and quizzes
- Handling student's assignments and grades
- Queries and complaints of the students
- Posting Notification on either whatsapp group or gmail.
- Rating, Feedback and FAQ's

## **2.1 Formal Problem Definition and Description**

We are going to create a full stack website which will provide a fully working online solution for managing various courses, events, enrolled students of engineer's gurukul. Currently they are manually handling all the management of their students, courses. which sometimes get very hectic and problematic. By using this website they will be able to automate the tasks of handling each and every student's information such as their registration number, course enrolled, fees status, pay fees online, check grade, student doubts.

Objectives of the project :-

- Automate the process of record keeping and tracking of all the enrolled student's name, registration no, courses, payment status etc.
- Implementing a cashless transaction feature through razor pay gateway.
- Creating a dashboard where teachers can manage and view their scheduled classes and where students can also see their timetable, faculty, course details etc.
- Creating a dashboard to handle all the assignments and grades where both student and mentor can check the assignment status is a more formal approach.
- Also a dashboard where students can post their queries/complaints and the teacher/faculty can look into the problem later on.
- Creating a section where students can give their ratings and feedback about the courses they are pursuing and also read answers to some frequently asked questions.

## **3. Feasibility Study**

### **3.1 Technical Feasibility -**

Our project is a complete web based application. The main technologies and tools associated with this project are HTML, CSS, JavaScript, React, Node.js, Express.js, MongoDB, Stripe Payment Gateway.

### **3.2 Schedule Feasibility -**

Being a full stack project the website is expected to be finished in 4 months. Depending on code, design and effort, the deadline is quite reasonable.

In the very first month, all the work related to documentation, SRS, data flow diagrams will be carried out.

In the second and third months, we will be building the core front end and back end functionalities of the web application and connect it with the database to make it fully functional for testing.

Last month will be fully dedicated for deployment of the project and testing it through different test cases and debug if any problem will occur.

### **3.3 Legal Feasibility -**

Our project will not be indulging in any activities which can violate any legal course of action, hence, legal feasibility is not applicable for our project.

### **3.4 Resource Feasibility -**

Resources that are required to build this project are Full stack web developer, Programming devices, Hosting server, Payment gateway service, Programming Tools.

### **3.5 Economic Feasibility -**

Being a web application, it will have an associated hosting cost. Since the system doesn't consist of any multimedia data transfer, bandwidth required for the operation of the application is very low. The system will follow freeware software standards. No cost will be charged from the potential user of the application. Bug fixes and maintaining tasks will have an associated cost later on.



## **4. Outcome Statement of the Project**

At the end, we are going to make Engineering Gurukul's task much easier by providing them with a fully working online solution for managing the students through the learning process and integrating it into their coursework.

Students will be able to browse through all the courses on the website, read reviews and purchase desired courses online. There will be a module for handling student's announcements, assignments and grades, queries and complaint handling modules, online quiz module, Feedback module and FAQ's about the frequently asked questions.

### **4.1 Users of the project -**

#### **4.1.1 Student**

- Login/ register with their account.
- Enroll into new course/ events
- Pay their fees
- Check their scheduled classes.
- Check announcement, pending assignment and grades
- Post their queries and complaints
- Read and post course ratings

#### **4.1.2 Teacher**

- Login/ register with their account.
- Check their scheduled classes.
- Upload assignment, announcements for students
- Upload student's marks and grades
- Read the student's doubt from the student's queries section.

### **4.1.3 Admin**

- User management
- Category Management
- Course management
  - Edit/Delete/Add
- Event Management
  - Add/ Edit/ Delete
- Order management
  - List courses purchased
  - Confirm payment
  - View Order details
- General Reports

## **5. Software Requirement Specifications**

### **5.1 PURPOSE**

The purpose of developing this “student management system” is to make Engineering Gurukul's task much easier by providing them with a fully working online solution for managing the students through the learning process and integrating it into their coursework.

### **5.2 SCOPE**

This web application will be designed and developed as a simple and intuitive system which shall cater the academic needs of students and faculty of Engineers Gurukul. Through this web application, students will be able to get enrolled in their desired courses , make online payment, check their course/Event schedule, upload/ download assignments, provide reviews on the courses, and give online tests etc.

### **5.3 External Interface Requirements**

#### **5.3.1 User Interface Requirements -**

##### Registration Form

This will allow users to create their account and register themselves with the website, after their registration, they will be able to login with their registered email and password.

##### Login Form

This is for logging in the students who are already registered with the website.

#### Course detail Page

In this page, different categories of courses being offered will be shown along with their details and reviews.

#### Payment Gateway

From this interface, users can pay for the courses online and get themselves enrolled.

#### Upload and download of assignment

This interface is for both the students and faculty for the purpose of uploading and downloading the assignment assigned to the students.

#### Notification Interface

This is basically for sending notifications to the students for conveying any important message.

#### Timetable Interface

This interface will be used by both teachers and students for checking their timetable of the day for their enrolled courses.

### **5.3.2 Hardware Interface**

The application would use the best web generation tools and tested technologies. The basic infrastructural requirements are:

#### **a. Client's Hardware**

Intel Pentium III 300 MHz or faster

At least 1GB RAM

At least 200 MB freed hard disk space

**For Mobile -**

At least 1GB RAM, Android 4.4(or newer) or ios 8(or newer), 100MB disk space

**b. Web**

Any web browser (Google chrome recommended)

**c. Developer's Hardware**

Core i5 or higher

4GB RAM or higher

At least 500GB Hard Disk space

### **5.3.3 Functional Requirements**

Functional requirements define the basic system behaviour. Essentially, they are what the system does or must not do, and can be thought of in terms of how the system responds to inputs. Functional requirements usually define if/then behaviours and include calculations, data input, and business processes.

#### **5.3.3.1 Login Requirement**

This is for logging in the students who are already registered with the website. 'Table 1' lists all the details of the same.

Purpose	User logs in to the system using an existing profile
User	Any user with an existing profile

Input Data	Username and password (through keyboard and mouse)
Output Data	Correct input will result in next page i.e. Dashboard
Pre-conditions Post-conditions	Input profile exists in the database, User password matches profile Page data is appropriate for the selected profile
Alternative Flow(s)	Invalid password or invalid username
Business Rule	This allows users to log in to their profile from anywhere.

Table 1

### 5.3.3.2 Uploading File

This interface is for both the students and faculty for the purpose of uploading and downloading the assignment assigned to the students. Below given 'Table 2' gives a detailed description.

Purpose	User wants to share data (pdf / ppt / others)
User	A legitimate user logged into the application
Input Data	File to be shared
Output Data	File ready to be downloaded by other intended users
Pre-conditions Post-conditions	User is logged in, file exists on user's computer The intended user is able to download the file

Table 2

### 5.3.3.3 Notifications

This is basically for sending notifications to the students for conveying any important message. 'Table 3' below mentions the details.

Purpose	Getting notifications of assignments and other important messages.
User	Any user of application
Input Data	User enables the notifications from settings
Output Data	Notifications
Pre-conditions	User has chosen to receive notifications.
Post-conditions	After enabling the notifications, the user is receiving them.

Table 3

#### 5.3.3.4 Time Table

This interface will be used by both teachers and students for checking their timetable of the day for their enrolled courses. 'Table 4' gives a description on the time table module.

Purpose	User wants to access the time table
User	A legitimate user logged into the application
Input Data	-
Output Data	Time table
Pre-conditions	User is logged in
Post-conditions	Correct time table is shown

Table 4

#### 5.3.4 Performance Requirements

The system will be able to support at least 1000 users. It is important that a substantial number of users be able to access the system at the same time, since an academic portal is important to the courses that employ it. The

times when the system will be under the most stress are likely during assignment submissions.

### **5.3.5 Database Requirements**

We are going to use noSQL database MongoDB. All data will be saved in the database: user accounts and profiles, uploaded files, messages etc. The database will allow concurrent access and will be kept consistent at all times.

### **5.3.6 Non Functional Requirements**

While functional requirements define what the system does or must not do, non-functional requirements specify how the system should do it. Non-functional requirements do not affect the basic functionality of the system (hence the name, non-functional requirements).

### **5.3.7 Availability**

The system should be available at all times, meaning the user can access it using a web browser, only restricted by the down time of the server on which the system runs. In case of a hardware failure or database corruption, a replacement page will be shown. Also in case of a hardware failure or database corruption, backups of the database should be retrieved with the MySQL server and saved by the administrator.

### **5.3.8 Security**

Passwords will be saved encrypted in the database in order to ensure the user's privacy. The user's IP will be logged and the system will be protected against vulnerabilities such as SQL injection attacks.

### **5.3.9 Portability**

The application will be practically independent of the OS-system and the end user can use any web browser to be able to use the features of the application.



## **6. System Design**

System design is basically a procedure of describing various components of the system such as, overall architecture, various modules, interface and data used in the system based on the user requirements. It is the process of describing, developing and designing the system in such a way, so that it can fulfill the requirements of the business organization.

### **6.1 Objective of Data Flow Diagram**

Data flow diagram shows graphical representation of the system. It shows how information(data) travels between different modules/processes. It generally starts with 0 level representing context level diagram which represents the whole system. Level 1 diagram shows module and level 2 diagram further refines working of various modules.

### **6.2 Zero Level DFD of E-Course Management System**

The Zero level data flow diagram elaborates the high level processes. It is a basic overview of the whole e-course management system being made.

The figure below describes processes taking place when various user roles are interacting with the website such as admin can manage courses, CMS, enrolled users. While users can browse courses, get enrolled, give reviews and receive messages and teachers can upload/ download assignments, send messages and check their schedule.

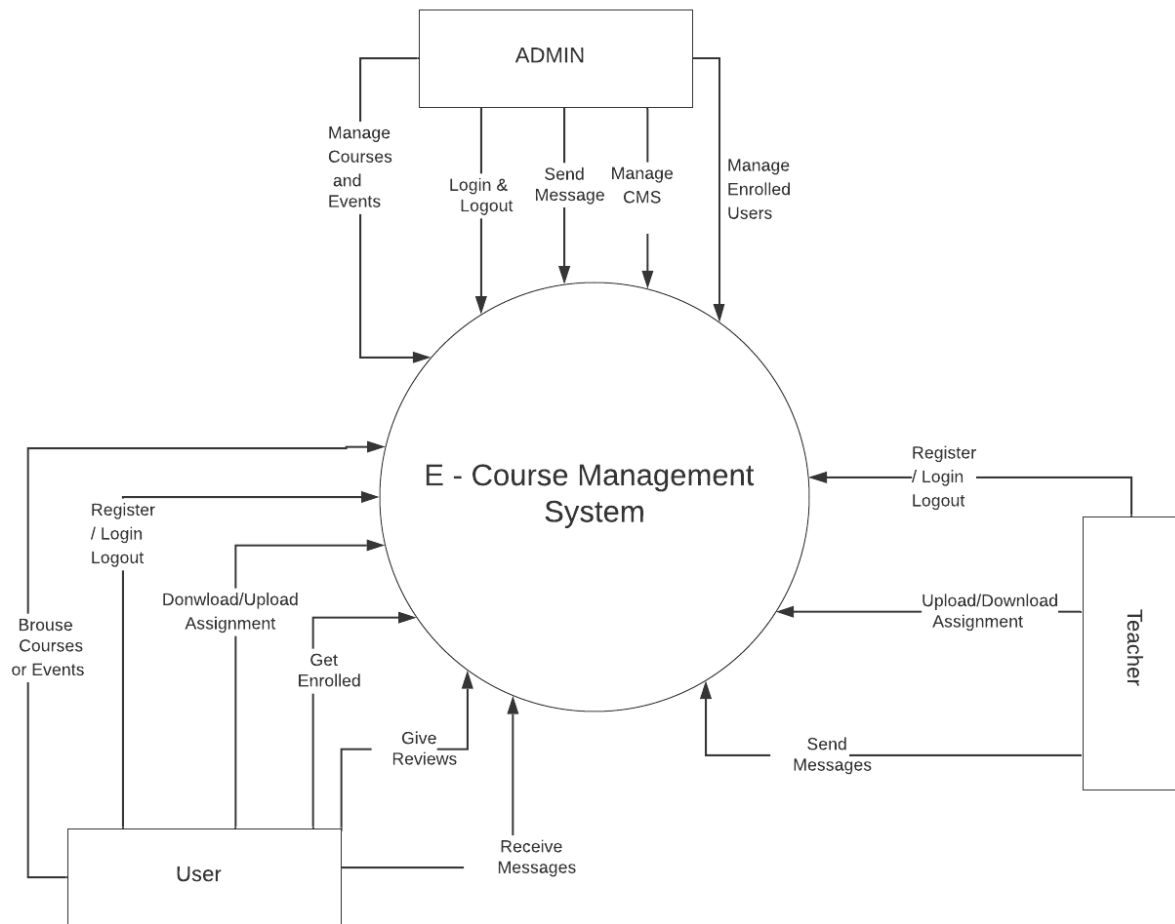


Figure1 : Zero Level DFD

### **6.3 First Level DFD of E-Course Management System**

First level data flow diagram goes one step deeper and describes the modules which are being used for management of the system.

It shows how the Login component is responsible for authentication of the user who is trying to sign in the website. Fees Management is responsible for secure payment of the fees though payment gateway. And also the functionality of course and event management and various user roles.

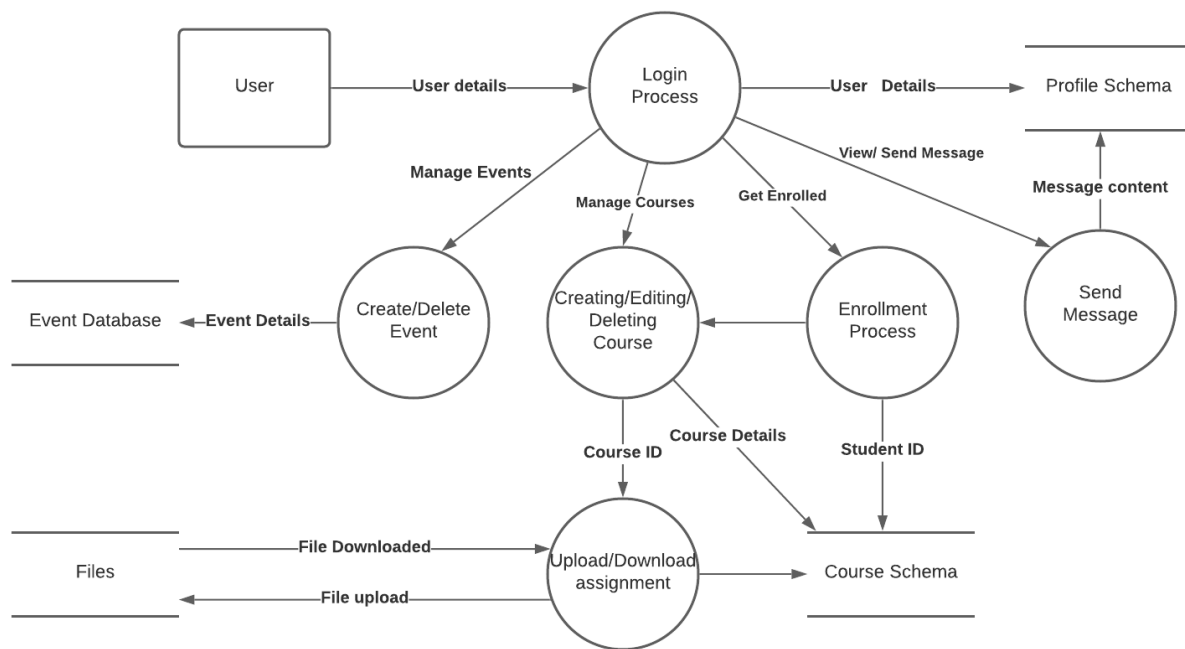


Figure2 : First Level DFD

## **6.4 Second Level DFD of E-Course Management System**

Second level DFD includes a detailed explanation and working of each module mentioned in first level DFD. It can be used for further planning or recording the module functionality and use case.

It shows how each module is responsible for various other small processes such as managing courses under admin, where admin can add new courses, update existing courses or delete courses. Or under student role, students can get enrolled into the courses and give reviews or submit assignments etc.

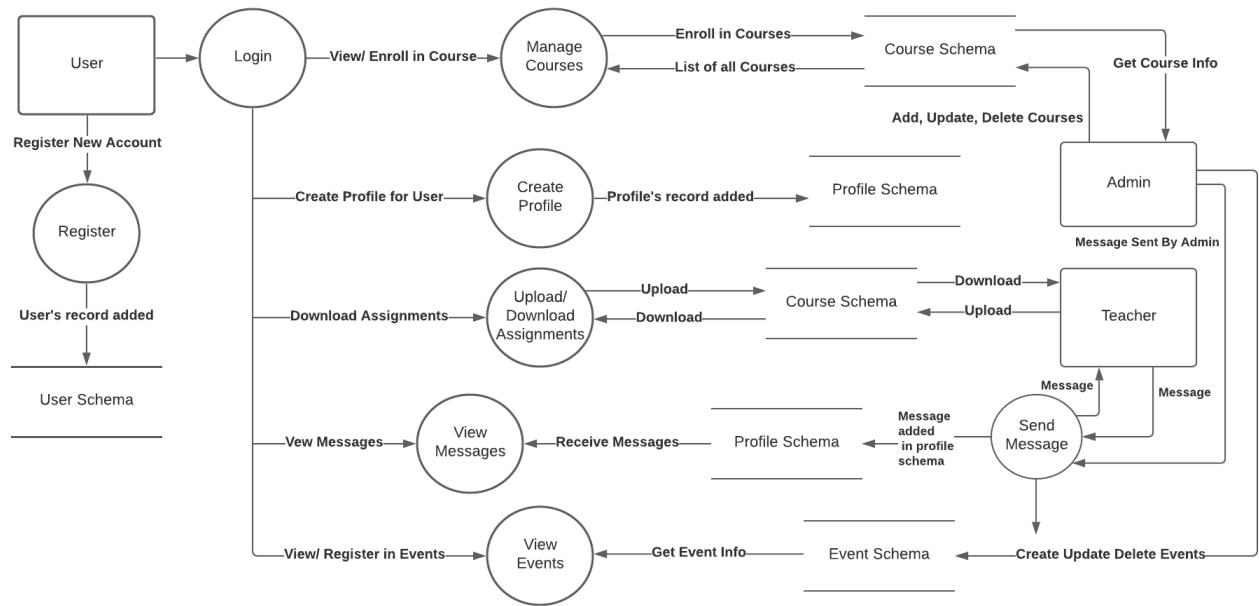


Figure 3: Second Level DFD

## 7. User Case Diagram

### 7.1 User Authentication and Roles

This flow chart describes the working of the Login page and authentication system. If the user is legitimate, then he will be given the access according to his role, else the option for registering will be shown.

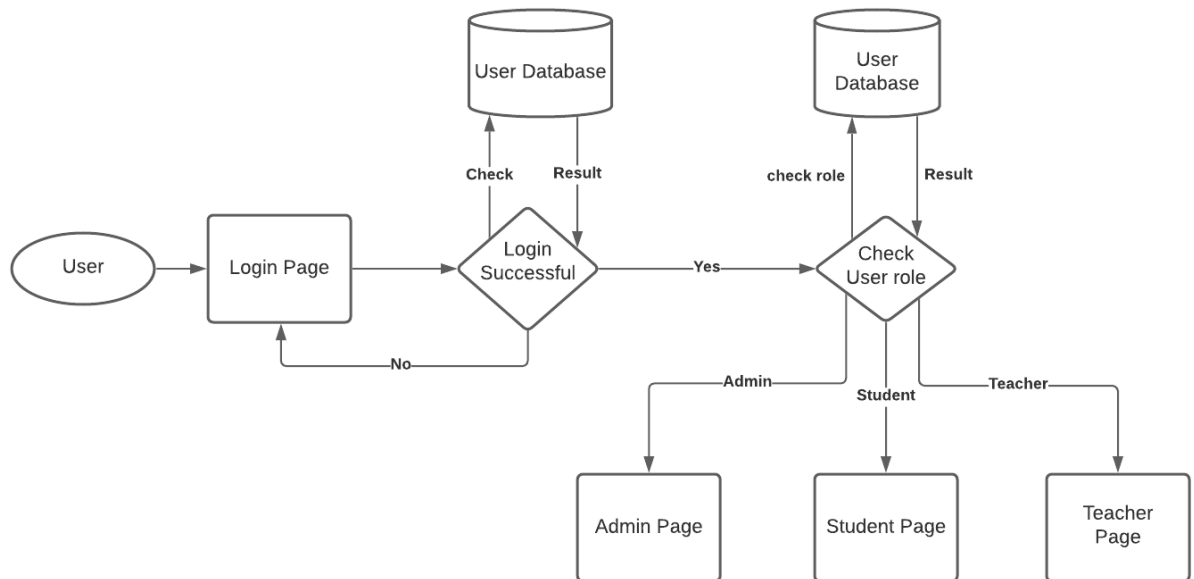


Figure 3: Flow chart of Login and Authentication

### 7.2 Course Review/Feedback

This flow chart describes the working of the feedback page. After successful login into the system, if the user is enrolled in any course, he can provide feedback.

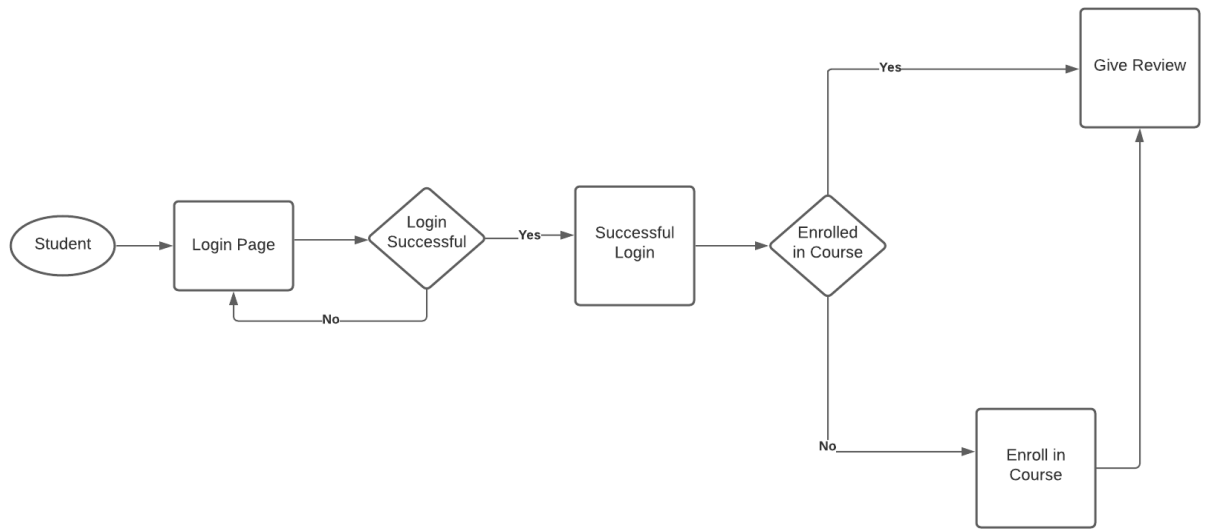


Figure 4: Flow chart for Course Reviews

### 7.3 Assignment Upload

The figure below describes the process of assigning the assignment by the faculty to the students. After successful login, the teacher can upload the assignment that students need to undertake and the system will send notification to all the enrolled students.

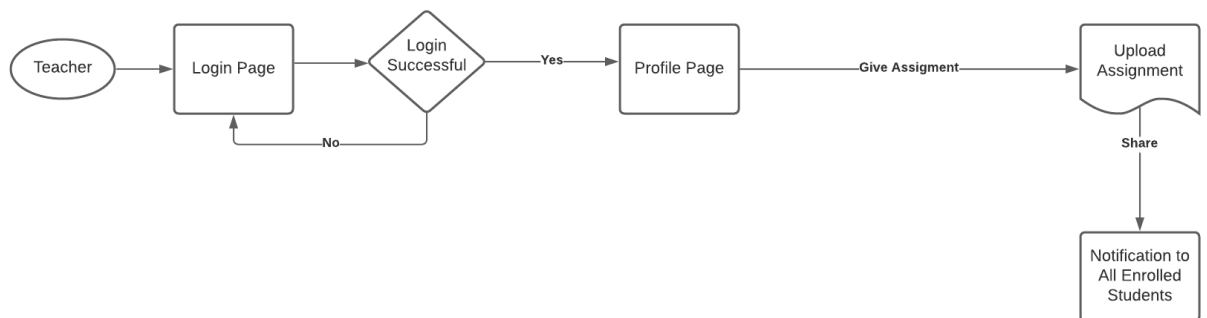


Figure 5: Flow Chart for uploading assignment

## 7.4 Assigning of Grades

This flow chart describes the process of assigning marks to the students.

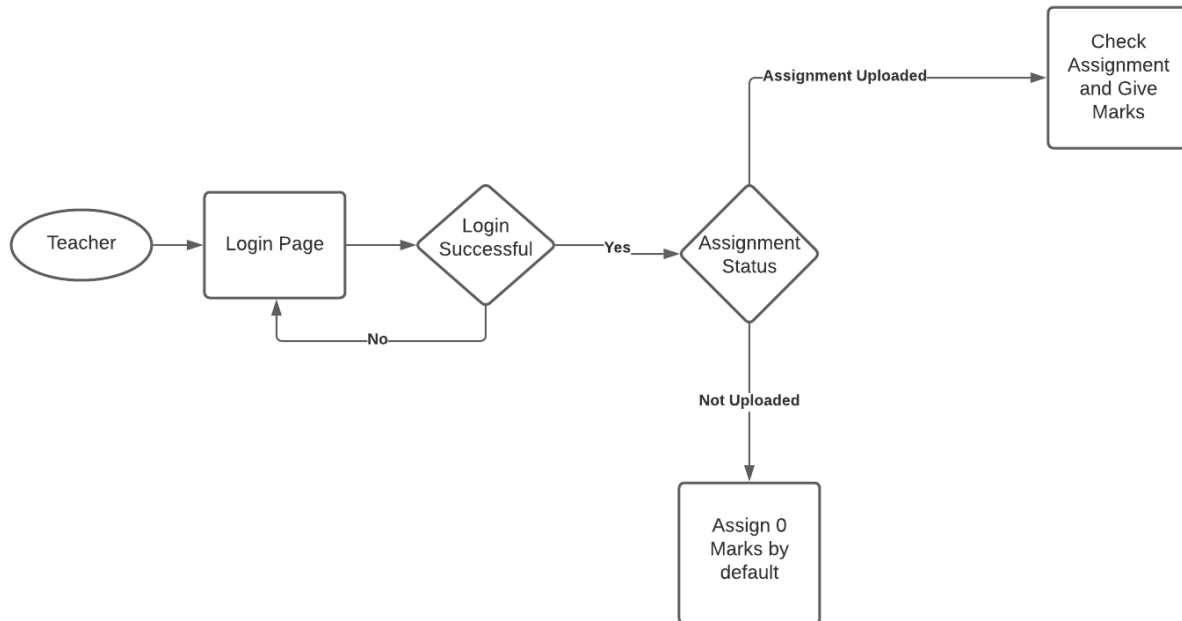


Figure 6: Flow Chart for assigning marks

## 8. Database Design

Database is basically an organized collection of data. In this, data is collected electronically in a form of structure in our computer system or cloud. A database is usually controlled by a database management system(DBMS). For this project we are using mongodb which is a NoSQL database. Mongodb uses collections and documents instead of tables as in traditional databases.

Following are the entities we are going to use for this website -

### 8.1 User

This entity will contain various information about the users such as their name, registered email address, password, role and we will use this information later at the time of logging in the user.

Field Name	Type	Description
Name	String	This contains name of the user
Email	String	This contains Email of the user
Password	String	This contains password of the user
Role	String	This will contains one of the role for user(student,teacher,admin)

Table 5

### 8.2 Profile

All the registered users will have an option for creating their own profile on the website. Having a profile will allow users to get enrolled with the courses and receive assignments, messages, notifications, and display their information to others such as github username, bio, about etc.

Field Name	Type	Description
User	Reference user from User table	This contain id from user table



enrolledCourses	Array of String	This contain list of all enroll courses
assignment	Array of Object	This contain list of all assignment
skills	Array of String	This contain list of skills
bio	String	This contain Information entered by user
githubusername	String	This contain github user Name
education	Array Of Object	Contain information related to education
messages	Array of Object	Contain list of all previous message
date	Date	Contain Date of profile created
social	Array of Object	Contain link related to social network

Table 6

### 8.3 Courses

This entity is for storing all the information regarding the courses being offered to students. The main purpose of this entity is to provide information about the courses such as name, content, description, prices etc.

Field Name	Type	Description
title	String	Contain title of course

description	String	Contain brief description of course
content	String	Contain List of content in the course
teacher	String	Contain name of the teacher
startDate	Date	Contain starting date of course
price	number	Contain price of the course
endDate	Date	Contain Ending date of course
prerequisite	Array of String	Contain skill require to take course
enrolledStudent	Array of string	Contain list of all student taking this course
review	Array of string	Contain review given by student

Table 7

### 8.3 Events

This entity is for storing all the information about the upcoming events. The main purpose of this entity is to provide basic information about the events such as title, description, link,dates etc.

Field Name	Type	Description
title	String	Contain title of course
description	String	Contain brief description of course
endDate	Date	Contain Ending date of course
link	String	Contain link for a external website/Form

## **8.4 Relationship between Entities**

1. When a registered user will create a new profile, a new schema of profile will be created associated with that user and this schema will also hold information about that user such as id, name and role of the user.
2. When
  - a. Users will enroll in a course, the course id of that course will be added in enrolledStudent(Course Schema) and enrolledCourse(Profile Schema).
  - b. When the user will review a course the data will be added in the review column of Course Schema and Profile Schema.
  - c. When the user (teacher) will give assignment in any specific course, that assignment will be added in the assignment column of Course and Profile Schema.

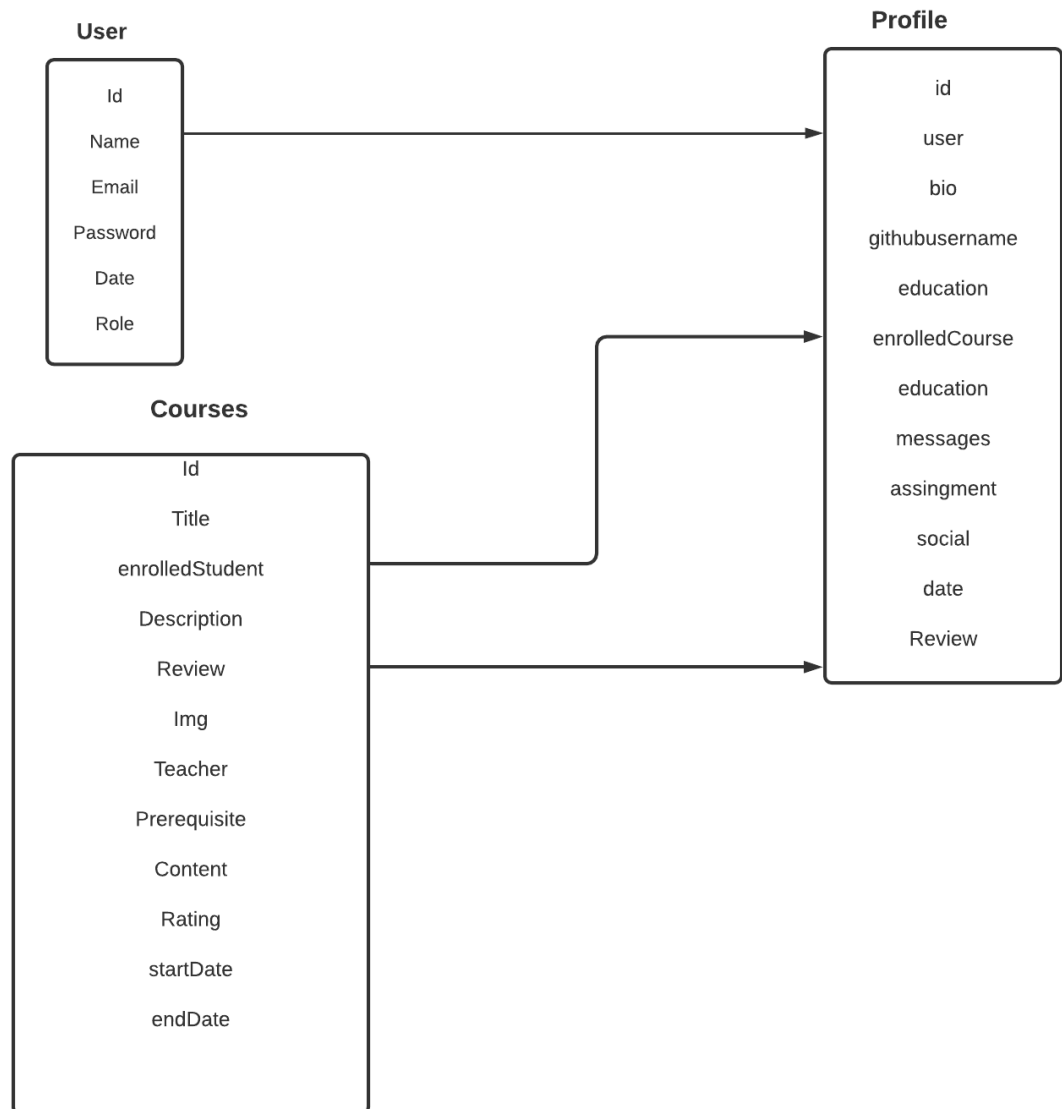


Figure 7: Database structure and relation

## 9. Screen Shots with coding

### 9.1 Code and Screenshot -

#### 9.1.1 - Assignment Module

##### 9.1.1.1 List of Course enroll by student

```
router.get('/courselist', async (req, res) => {

  try {

    var directoryPath = path.join(

      __dirname + '\\..' + '\\..',

      '/uploads',

      '/teacher'

    );

    fs.readdir(directoryPath, function (err, files) {

      //handling error

      if (err) {

        return console.log('Unable to scan directory: ' + err);

      }

      //listing all files using forEach

      files.forEach(function (file) {

        // Do whatever you want to do with the file

        console.log(file);

      });

      res.json(files);

    });

  }

});
```

```

    });


    } catch (err) {

        console.error(err.message);

    }

});

```

<div>  <div> <a href="#">Dashboard</a> <a href="#">Courses</a> <a href="#">Assignment</a> <a href="#">Message</a> <a href="#">View Events</a> <a href="#">Logout</a> </div> </div>			
Subjects For Assignments -			
#	Subject Assignment	Taught By	Assignments
→	MERN Stack	Dr. Sanjeev Singh	4
→	Networking	Dr. Sanjeev Singh	15

### 9.1.1.2 List of assignment in a particular course

```

router.get('/course-list/:courseID', async (req, res) => {

    try {

        var courseID = req.params.courseID;

        var directoryPath = path.join(

            __dirname + '\\..' + '\\..',

            '/uploads',

            '/teacher',

            `/${courseID}`

```

```
);

fs.readdir(directoryPath, function (err, files) {

    //handling error

    if (err) {

        return console.log('Unable to scan directory: ' + err);

    }

    //listing all files using forEach

    files.forEach(function (file) {

        // Do whatever you want to do with the file

        console.log(file);

    });

    res.json(files);

});

} catch (err) {

    console.error(err.message);

}

});
```

## List of Assignments

#	Assignment Title	Download	Upload	Due Date
→	Login Exercise.pdf	<a href="#">Upload Assignment</a>	<a href="#">Download</a>	
→	Mern Stack Assignment 1	<a href="#">Upload Assignment</a>	<a href="#">Download</a>	
→	wdawedeawd.pdf	<a href="#">Upload Assignment</a>	<a href="#">Download</a>	

### 9.1.1.3 Upload Assignment

```
router.post('/student/:courseID/:assignID/:username/:email', (req, res) =>
{
  if (req.files) {
    var name = req.params.username + '_' + req.params.email;
    var file = req.files.file;
    file.name = name + '.pdf';
    var filename = file.name;
    var courseID = req.params.courseID;
    var assignID = req.params.assignID;
    var paths = path.join(__dirname + '\\..' + '\\..', '/uploads',
'/student');

    fs.mkdir(
      path.join(paths, `${courseID}`, `${assignID}`),
      { recursive: true },
```



```
function (err) {

    if (err) {

        console.log(err);

    } else {

        console.log('New directory successfully created.');
```

```

    } else {

        console.log('No file');

    }

});

```

ENGINEERS  
gurukul

Dashboard Courses Assignment Message View Events Logout

### Upload Your Assignment Here

Assignment Due Date Over!

Choose File Browse

Upload

## 9.2 Database Tables -

### User -

```

_id: ObjectId("603cbcaacf0b20090c3114b")
role: "student"
name: "Vivek"
email: "vv@gmail.com"
password: "$2a$10$o8SbRmp1eYmrhfxl4v81y.MUMw1ITJbb3lZTTUpXjc6Dfph2KDyzC"
date: 2021-03-01T10:06:34.251+00:00
__v: 0

```

### Profiles -

```

    _id: ObjectId("603e202fafd0fa17a40afae")
  > enrolledCourse: Array
  > assignment: Array
  > notification: Array
  > skills: Array
    user: ObjectId("603cbf08d2d72c26d05ecdb5")
    bio: "Technical Head at Engineers Gurukul"
    githubusername: "EngGurukul"
  > education: Array
  > messages: Array
    date: 2021-03-02T11:23:27.317+00:00
    __v: 83
  > social: Object

```

## Events -

```

_id: ObjectId("606abe3c26dd8a11cccbc7e7")
title: "Seminar on how to write good code."
description: "Software engineering is not just all about learning a language and bui..."
startDate: "2021-04-06"
endDate: "2021-04-21"
link: "https://docs.google.com/forms/u/0/"
__v: 0

```

## Course -

```

_id: ObjectId("6064a0fde0cd122cc8d2df87")
  > prerequisite: Array
  > enrolledStudent: Array
    teacher: "Dr. Sanjeev Singh"
    title: "MERN Stack"
    content: "MongoDB, Express, React, Node"
    description: " MERN is one of several variations of the MEAN stack (MongoDB Express ..."
    startDate: "2021-03-17"
    endDate: "2021-03-27"
  > review: Array
    __v: 24
  > assignment: Array

```

## 10. Validation Check

In this project Express Validator is used for validating input in node js.

If there is any error in the input we send the error message in form of array and in react we dispatch error message using showAlert action

## Sending Error Message -

```
const errors = validationResult(req);

if (!errors.isEmpty()) {

  return res.status(400).json({ errors: errors.array() });

}
```

## Receiving error message in react -

```
const errors = err.response.data.errors;

if (errors) {

  errors.forEach((error) => dispatch(setAlert(error.msg, 'danger')));

}
```

## setAlert action -

```
import { v1 as uuid } from 'uuid';

import { SET_ALERT, REMOVE_ALERT } from './types';

export const setAlert = (msg, alertType, timeout = 2000) => (dispatch) => {

  const id = uuid();
```

```

dispatch({

  type: SET_ALERT,

  payload: { msg, alertType, id },

});

  setTimeout(() => dispatch({ type: REMOVE_ALERT, payload: id }),
timeout);

};

```

## 10.1 Validation check in sign up

```

router.post(

  '/',

  [

    check('name', 'Name is required').not().isEmpty(),

    check('email', 'Please include a valid email').isEmail(),

    check(

      'password',

      'Please Enter a password with 6 or more characters'

    ).isLength({ min: 6 }),

  ],

  async (req, res) => {

    const errors = validationResult(req);

    if (!errors.isEmpty()) {

```

```

    return res.status(400).json({ errors: errors.array() });

  }

}

```



## Sign Up

Name

Email

Password

Confirm Password

## 10.2 Validation in login

```

router.post(

  '/',

  [

    check('email', 'Please include a valid email').isEmail(),

    check('password', 'Password is required').exists(),

  ],

  async (req, res) => {

    const errors = validationResult(req);

```

```

    if (!errors.isEmpty()) {

        return res.status(400).json({ errors: errors.array() });

    }


}


```



## Login

---

 Email

 Password

## 10.3 Course

```

router.post(

    '/',

    [

        auth,

        [

            check('teacher', 'Teacher Name is Required').not().isEmpty(),

            check('title', 'Course Title is Required').not().isEmpty(),

            check('description', 'Description name is
Required').not().isEmpty(),

```

```

        check('content', 'Course Content is required').not().isEmpty(),

        check('startDate', 'Start date of course is
Required').not().isEmpty(),

        check('endDate', 'Ending date of course is
Required').not().isEmpty(),

        check('prerequisite', 'Prerequisite is Required').not().isEmpty(),

    ],

],

async (req, res) => {

    const errors = validationResult(req);


    if (!errors.isEmpty()) {

        return res.status(400).json({ errors: errors.array() });

    }

}
}

```



[Dashboard](#)
[Message](#)
[Courses](#)
[Users](#)
[Events](#)
[Logout](#)

Teacher Name is Required

Course Content is required

Start date of course is Required

Ending date of course is Required

## Create New Course

Create your course here

Course Title

Description

Description of District