VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI – 590 018

A Mini Project Report on

COLLEGE DATABASE MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirements as a part of the DBMS lab for the V semester of degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi

Submitted by

Prajwal B P Barlaya 1RN20IS108
Prajwal M 1RN20IS110

Under the Guidance of

Guide

Ms. Aishwarya G
Assistant Professor
Dept. of ISE, RNSIT

Lab In charge

Dr. T.S.Bhagavath Singh
Associate Professor
Dept. of ISE, RNSIT

ESTD : 2001
An Institute with a Difference

Department of Information Science and Engineering

RNS Institute of Technology

Channasandra, Dr. Vishnuvardhan Road, RR Nagar Post,
Bengaluru – 560 098

2022 – 2023

RNS INSTITUTE OF TECHNOLOGY

Channasandra, Dr.Vishnuvardhan Road, RR Nagar Post,
Bengaluru – 560 098

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini project report entitled **COLLEGE DATABASE MANAGEMENT SYSTEM** has been successfully completed by **PRAJWAL B P BARLAYA** bearing **USN 1RN20IS108** and **PRAJWAL M** bearing **USN 1RN20IS110**, presently V semester student of **RNS Institute of Technology** in partial fulfillment of the requirements as a part of the DBMS Laboratory for the award of the degree **Bachelor of Engineering in Information Science and Engineering** under **Visvesvaraya Technological University, Belagavi** during academic year 2022 – 2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements as a part of DBMS Laboratory for the said degree.

Ms. Aishwarya G	**Mr. T S Bhagavat Singh**	**Dr. Suresh L**
Assistant Professor	**Associate Professor**	**Professor and HOD**
Guide	Faculty In charge	

External Viva

Name of the Examiners	**Signature with date**
1. _____	_____
2. _____	_____

DECLARATION

We, **PRAJWAL B P BARLAYA [USN: 1RN20IS108**] and **PRAJWAL M [USN: 1RN20IS110],** students of V Semester BE, in Information Science and Engineering, RNS Institute of Technology hereby declare that the Mini project entitled **COLLEGE DATABASE MANAGEMENT SYSTEM** has been carried out by us and submitted in partial fulfilment of the requirements for the V Semester degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the academic year 2022-2023.

Place: Bengaluru

PRAJWAL B P BARLAYA
[1RN20IS108]

PRAJWAL M
[1RN20IS110]

ABSTRACT

College management is when a group of people and resources work together to plan, organize, strategize, and put in place structures to run an educational system. The Static website for this project “COLLEGE DATA BASE MANAGEMENT” is built using HTML , CSS and the backend for the project is done using Node Javascript , MySQL Server and the Angular CLI is used as framework for frontend .

College Data Management System gives a straightforward interface to support of understudy data, staff information, fee record. It very well may be utilized by instructive establishments or schools to keep up the records of understudies without any problem. The College Data Base Management System helps in registering the student details i.e Admission process . The student can enter his details and select the course in which he want to pursue his education. The details of student which he has entered can be can be viewed by admin after the registration Process has been completed. The details of students belonging to the particular branch can be viewed by the admin and thus can be sorted further.

ACKNOWLEDGEMENT

The fulfillment and rapture that go with the fruitful finishing of any assignment would be inadequate without the specifying the people who made it conceivable, whose steady direction and support delegated the endeavors with success.

We would like to profoundly thank **Management** of **RNS Institute of Technology** for providing such a healthy environment to carry out this project work.

We would like to express our thanks to our Principal **Dr. M K Venkatesha** for his support and inspired us towards the attainment of knowledge.

We wish to place on record our words of gratitude to **Dr. Suresh L**, Professor and Head of the Department, Information Science and Engineering, for being the enzyme and master mind behind our project work.

We would like to express our profound and cordial gratitude to our Faculty In charge **T S Bhagavath Singh**, Associate Professor, Department of Information Science and Engineering for his valuable guidance, constructive comments and continuous encouragement throughout the mini project work.

We would like to express our profound and cordial gratitude to our Guide **Ms. Aishwarya G**, Assistant Professor, Department of Information Science and Engineering for his valuable guidance in preparing Project report.

We would like to thank all other teaching and non-teaching staff of Information Science & Engineering who have directly or indirectly helped us to carry out the project work.

And lastly, we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in carrying out this project work.

PRAJWAL B P BARLAYA [1RN20IS108]

PRAJWAL M [1RN20IS110]

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The COLLEGE MANAGEMENT SYSTEM can be used to store student information like attendance, fees, and student result etc. admin can create report regarding any student any time using this system. Using this system you can register new student and their course details. You can submit students fees and can check fees details anytime. You can create exam result and submit in this system. Student can check their result online by logging to the system. You can also add new employee in the system and can check details of the employee easily. Student can also check course detail online from this system.

The main objective of college management system is to automate all functionalities of a college or university. Using this system you can manage all college management work like admission, fees submission, time table management and result declaration. Using this college management system you can view or update data and information about students and staff easily. This system helps in managing the activity like student admission, student registration, fees submission. Admin can also retrieve information of employee student.

Using this system you can manage all information of all aspects of a college, its students, faculties, Departments, marks and other curricular activities. College management system provides the easiest way to manage all functionalities of a college. This system facilitates colleges to maintain the functionality related to college employees and their students. College Data Management System gives a straightforward interface to support of study of data, staff information, attendance, admission, fee record. The college database management system needs tocreate the college database to organize student records and other information about the students. The admin can also register student to the database.

College Management System can store and manage all data of the various departments of a college like Administration, Attendance, Staff details etc. using this system user can retrieve any information related to student, teacher and fees. Using this system teacher can check student attendance anytime. This system also help teacher to announce the result. College administration can also manage college work easily. Admin can check leave, salary and other details of teacher any time. They can also create time table of classes from this system. The Library module is used for the data process of library and book accessing for students and staffs.

Chapter 2

LITERATURE SURVEY

2.1 Traditional File System

In the early days of computing, data management and storage were a very new concept for organizations. The traditional approach to data handling offered a lot of the convenience of the manual approach to business processes (e.g, handwritten invoices & account statements, etc.) as well as the benefits of storing data electronically.

The traditional approach usually consisted of custom-built data processes and computer information systems tailored for a specific business function. An accounting department would have their own information system tailored to their needs, where the sales department would have an entirely separate system for their needs.

Initially, these separate systems were very simple to set up as they mostly mirrored the business process that departments had been doing for years but allowed them to do things faster with less work. However, once the systems were in use for so long, they became very difficult for individual departments to manage and rely on their data because there was no reliable system in place to enforce data standards or management.

Separate information systems for each business function also led to conflicts of interest within the company. Departments felt a great deal of ownership for the data that they collected, processed, and managed which caused many issues among company-wide collaboration and data sharing.

2.1.1 Pros and Cons of the Traditional Approach

Pros

- **Simple**
  - Matched existing business processes and functions.
  - Companies were not as interested in funding complicated information systems
- **Initially low-cost**
  - Early computing was not viewed as beneficial for large funding.
  - Systems were designed to be cheap in order to save on cost.

Cons

- **Separated ownership**
  - Business functions had a high sense of data ownership.
  - Departments unwilling to share data for fear of minimizing their superiority.

- **Unmanaged redundancy**
  - Multiple instances of the same data appeared throughout various files, systems, and databases.
  - Information updated in one place was not replicated to the other locations.
  - Disk space was very expensive, and redundancy had a big impact on storage.
- **Data in consistency**
  - Redundant data stored in various locations was usually never stored the same way.
  - Formatting was not centrally managed.
- **Lack of data sharing**
  - Same data stored in multiple locations.
  - Caused unnecessary doubling of efforts for processing and managing data.
- **High costs in the long run**
  - Hiring data processors for each department was very expensive, and each position was typically working on the same thing just for a different area.
  - Doubling of work as well as excessive maintenance cost.

2.2 Downfall of Traditional Management System

Conceived in a relatively centralized era when software was deployed in static environments, legacy database architectures fail to support an increasingly mobile world where applications are accessed anytime, anywhere.

Today software users want consistent improvements in usability and expect SaaS vendors to deliver new features and functionalities needed to achieve their business objectives. However, legacy database technologies fall short. in serving the needs of todays distributedand cloud environments for the following reasons:

- Inadequate fail over capabilities
- Insufficient provisions during peak demands Latency issues
- Lack of high availability at all times Increasing operational costs
- Inability to meet the demands of global markets

For all of these reasons, traditional databases are unable to deliver results in a rapidly growing environment where the workload is geographically distributed across heterogeneous datacenters. Upgrading to a more distributed data model is costly and complicated and your DBAs can’t just sit back and give up on this situation. Hence, due to these various reasons, the downfall of the traditional system was inevitable.

2.3 Introduction to the Database Management System

A database management system (DBMS) refers to the technology for creating and managing databases. Basically, a DBMS is a software tool to organize (create, retrieve, update and manage) data in a database.

The main aim of a DBMS is to supply a way to store and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of Data. Meaningful data combine to form Information. Hence, information is interpreted data- data provided with semantics.MS ACCESS is one of the most common examples of database management software. Database systems are meant to handle large collection of information Management of data involves both defining structures for storage of information and providing mechanisms that can do the manipulation of those stored information. Moreover, the database system must ensure the safety of the information stored, despite system crash or attempts at unauthorized access.

2.4 Indicative areas for the use of a DBMS

- Airlines: reservations, schedules etc.
- Telecom : calls made, customer details, network usage etc. Universities : registration, results, grades, etc.
- Sales: products, purchases, customers etc. Banking: all transactions

2.5 Advantages of a DBMS

A Database Management System has many advantages over the traditional file system used in the earlier days, such as:

- **Data independence**: Application programs should be as free or independent as possible from details of data representation and storage. DBMS can supply an abstract view of the data for insulating application code from such facts.
- **Efficient data access**: DBMS utilize a mixture of sophisticated concepts and techniques for storing and retrieving data competently and this feature becomes important in cases where the data is stored on external storage devices.
- **Data integrity and Security** : If data is accessed through the DBMS , the DBMS can enforce integrity constraint on the data

- **Data administration**: When several users share the data, integrating the administration of data can offer major improvements. Experienced professionals understand the nature of the data being managed and can be responsible for organizing the data representation toreduce

redundancy and make the data to retrieve efficiently

2.6 Components of a DBMS

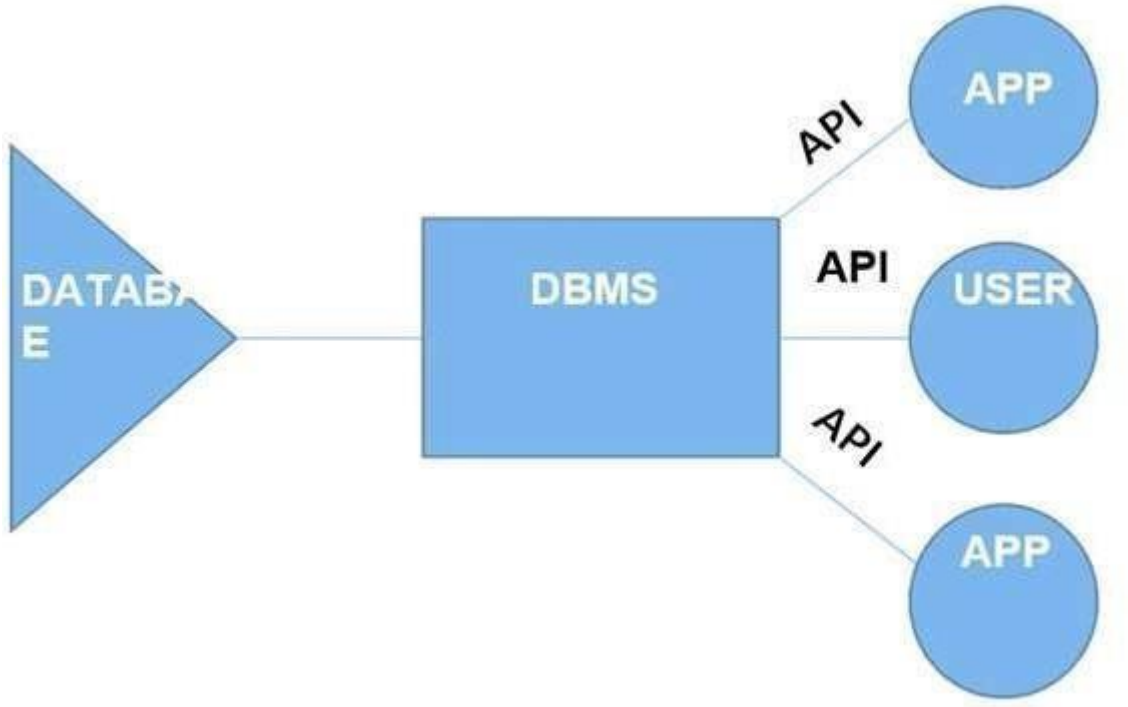


Figure 2.1 Components of DBMS

- **Users**: Users may be of any kind, such as data base administrators, system developers or database users.
- **Database application**: Database application may be Departmental ,Personal, Organizational and /or Internal
- **DBMS**: Software that allows users to create and manipulate database access.
- **Database**: Collection of logical data as a single unit.

Chapter 3

SYSTEM REQUIREMENTS

The main purpose of this SRS document is to illustrate the requirements of the project Student information System and is intended to help any organization to maintain and manage its student's personal data.

3.1 Hardware Requirements

- Processor: Intel Core i5
- RAM: 4GB
- Processor Speed: 1.2 GHz
- Storage space: 40GB

3.2 Software Requirements

Technologies Used:

- Front End: HTML, CSS
- Back-End: Nodejs
- Text Editor: VS CODE
- Server: Nodejs server
- Operating System: Windows 11
- Frame Work: Angular CLI
- Database: MySQL

Chapter 4

SYSTEM DESIGN

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.

4.1 Entity Relation Diagram

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database as shown in figure.

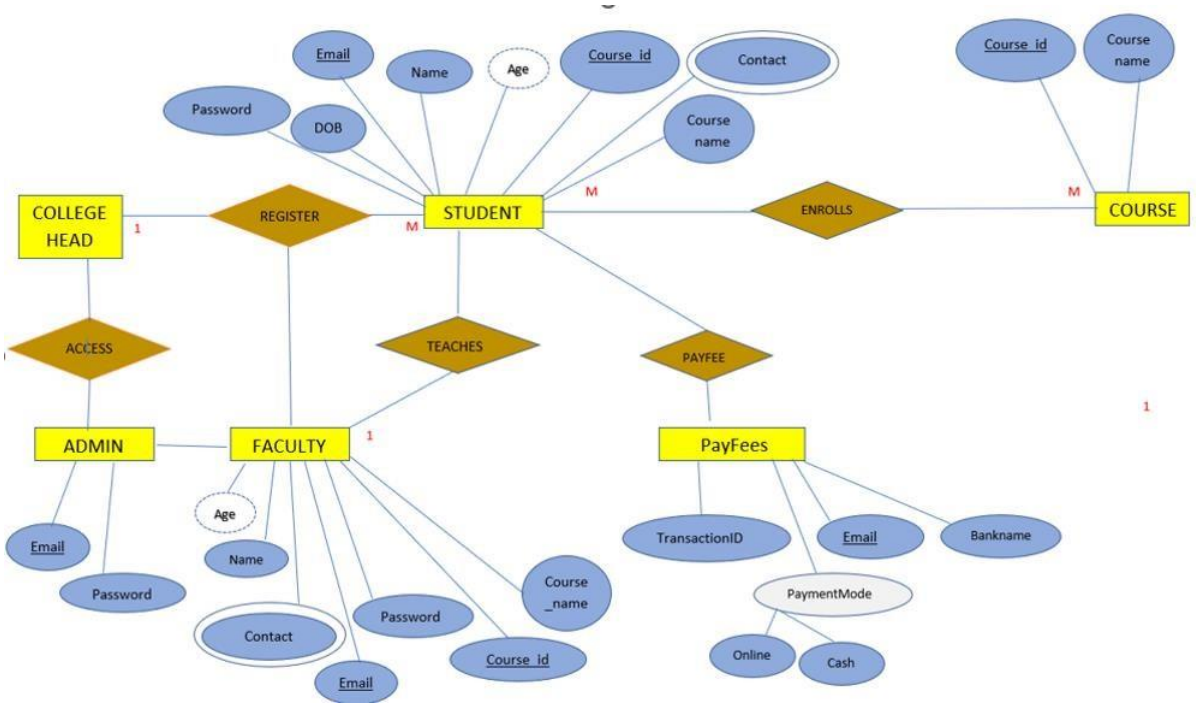


Figure 4.1 Entity Relational Diagram

4.2 Schema Diagram

A schema contains schema objects, which could be tables, columns, data types, store procedures, relationships, primary keys, foreign keys. A database schema can be represented in a visual diagram, which shows the database objects and their relationship with each other.

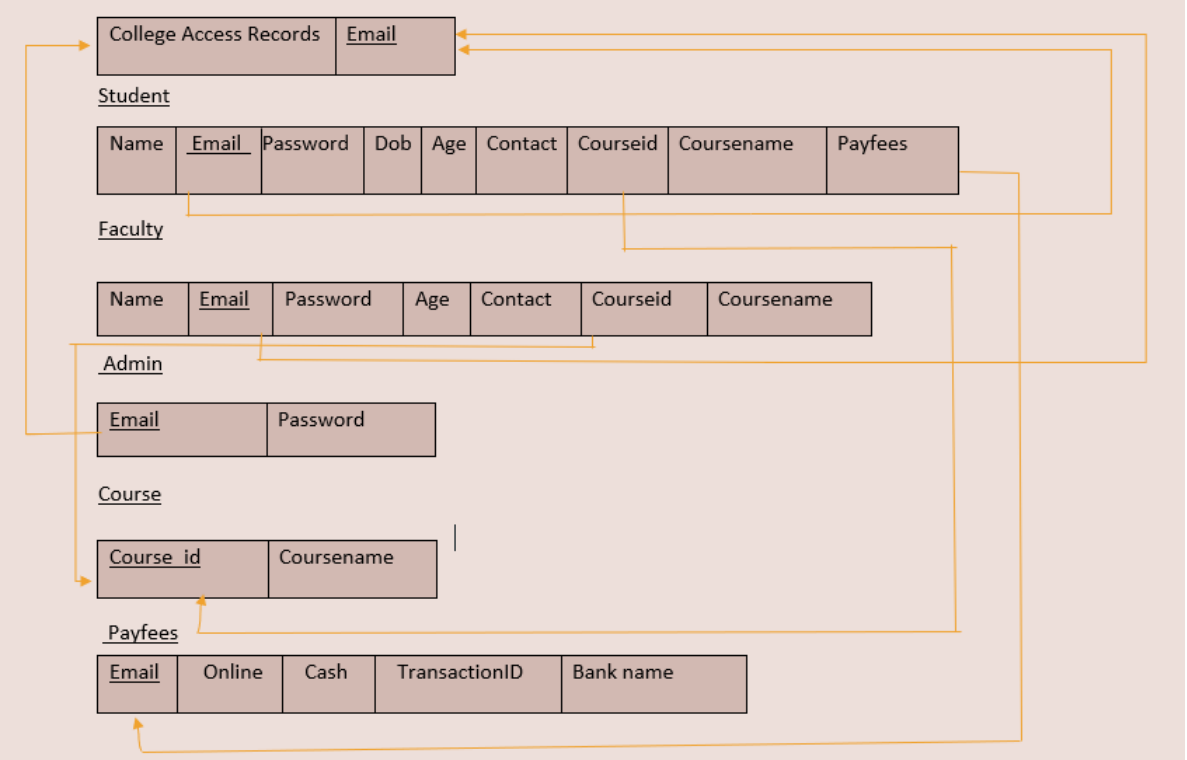


Figure 4.2 Schema diagram of College Data Base Management System

Chapter 5

IMPLEMENTATION

5.1 FRONT-END DEVELOPMENT

HTML: HTML is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current major version of the HTML standard. It was published in October 2014 by the World Wide Web Consortium (W3C) to improve the language with support for the latest multimedia, while keeping it both easily readable by humans and consistently understood by computers and devices such as web browsers, parsers, etc.HTML is intended to subsume not only HTML 4, but also XHTML 1 and DOM Level 2HTML.

CSS: Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. The first version of CSS was invented. 1998- CSS 2 was released and work on CSS 3 began. CSS 3 was very different from the other versions, for instead of being a single monolithic specification, it was published as a set of separate documents known as modules.

FRONT-END FRAMEWORK: Angular is an open-source, JavaScript framework written in TypeScript. Google maintains it, and its primary purpose is to develop single-page applications. As a framework, Angular has clear advantages while also providing a standard structure for developers to work with.

MYSQL: MySQL is a freely available open-source Relational Database Management System (RDBMS) that uses Structured Query Language (SQL). It is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons −It is released under an open-source license. MySQL uses a standard form of the well-known SQL data language. SQL is the most popular language for adding, accessing and managing content in a database. It works on many operating systemsand with many languages including PHP, PERL, C, C++, JAVA, etc. and works very quickly and works well even with large data sets.

VISUAL STUDIO: Microsoft Visual Studio is an integrated development environment (IDE) rom Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. Visual Studio supports 36 different programming languages and allows the code editor and

debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++, C++/CLI, Visual Basic.NET,C#, F#, JavaScript, Typescript, XML, XSLT, HTML, and CSS. Support for other languages such as Python, Ruby, Node.js, and M among others is available via plug-ins. Java (and J#) was supported in the past.

5.2 BACK-END DEVELOPMENT:

NODEJs: Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix, macOS, and more. Node.js is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser.

It is used for server-side programming, and primarily deployed for non-blocking, event-driven servers, such as traditional web sites and back-end API services, but was originally designed with real-time, push-based architectures in mind.

5.3 SQL (Structured Query Language)

SQL is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQL is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQL is an embedded SQL database engine. Unlike most other SQL databases, SQL does not have a separate server process. SQL reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. SQL is a compact library.The library size can be less than 750KiB, depending on the target platform and compiler optimization settings.

5.4 Code to Major Functionalities:

5.4.1. Student Registration

```
router.post('/signup', (req, res) => { let user = req.body;
query = "select name,dob,email,password,age,courseid,coursename, role,status from user where email=?"
connection.query(query, [user.email], (err, results) => {
if (!err) {

if (results.length <= 0) {
```

```
query = "insert into
user(name,dob,contactNumber,email,password,age,courseid,coursename,status,role)
values(?,?,?,?,?,?,?,?,'false','user')";
connection.query(query, [user.name, user.dob,user.contactNumber, user.email,
user.password,user.age,user.courseid,user.coursename], (err) => {
console.log(err);
if (!err) {
return res.status(200).json({ message: "Succesfully Registered ! You will receive a call from
BARLAYA INSTITUTE in next 2 days .Thank You " });
}
else {
return res.status(500).json(err);
}
})
}
else {
return res.status(400).json({ message: "Emaily Already Exist." }) }}
else {
return res.status(500).json(err);
}
})
})
```

5.4.2 . Admin Login :

```
router.post('/login', (req, res) => { let user = req.body;
query = "select name,dob,email,password,age,courseid,coursename, role,status from user where
email=?"
connection.query(query, [user.email], (err, results) => {
if (!err) {
if (results.length <= 0) {
query = "insert into
user(name,dob,contactNumber,email,password,age,courseid,coursename,status,role)
values(?,?,?,?,?,?,?,?,'false','user')";
connection.query(query, [user.name, user.dob,user.contactNumber, user.email,
user.password,user.age,user.courseid,user.coursename], (err) => {
console.log(err);
if (!err) {
```

```
return res.status(200).json({ message: "Succesfully Registered ! You will receive a call from
BARLAYA INSTITUTE in next 2 days .Thank You " });
}
else {
return res.status(500).json(err);
}
})
}
else {
return res.status(400).json({ message: "Emaily Already Exist." }) }}
else {
return res.status(500).json(err);
}
})
})
const user = req.body;
query = "select email,password,role,status from  user where email=?"
connection.query(query, [user.email], (err, results) => {
if (!err) {
if (results.length <= 0 || results[0].password != user.password) {
return res.status(401).json({ message: "Incorrect Username or Password" });
}
else if (results[0].status === 'false') {
return res.status(401).json({ message: "ONLY ADMIN CAN LOGIN HERE !.THANK YOU"
});
}
else if (results[0].password == user.password) {
const response = { email: results[0].email, role: results[0].role }
const accessToken = jwt.sign(response, process.env.ACCESS_TOKEN, { expiresIn: '8h' });
res.status(200).json({ token: accessToken });

}
else {
return res.status(400).json({ message: "Something Went Wrong . Please try in sometime" });
}}else {
return res.status(500).json(err);}})})
```

5.4.3. Fees Pay Form

```
const express = require('express');
const connection = require('../connection');
const router = express.Router();
const jwt = require('jsonwebtoken');
require('dotenv').config();
router.post('/Psignup', (req, res) => {
let payfee = req.body;
query = "select
name,contactNumber,email,paymentmode,transactionid,bankname,coursename,status,role from
payfee where email=?"
connection.query(query, [ payfee.email], (err, results) => {
if (!err) {
if (results.length <= 0) {
query = "insert into
payfee(name,contactNumber,email,paymentmode,transactionid,bankname,coursename,status,role
) values(?,?,?,?,?,?,?,'false','user')";
connection.query(query, [ payfee.name, payfee.contactNumber, payfee.email,
payfee.paymentmode, payfee.transactionid, payfee.bankname, payfee.coursename], (err) => {
console.log(err);
if (!err) {
return res.status(200).json({ message: "(: Your PAYMENT details has Registered :)" });
}
else {
return res.status(500).json(err);
}
})
}
else {
return res.status(400).json({ message: "You have already Updated your Payment Details" })} }
else {
return res.status(500).json(err);
}
})
})
router.post('/add',(req,res,next)=>{
```

```
let payfee= req.body;
query ="insert into  payfee
(name,contactNumber,email,paymentmode,transactionid,bankname,coursename)
values(?,?,?,?,?,?,?)";
connection.query(query,[ payfee.name, payfee.contactNumber, payfee.email,
payfee.paymentmode, payfee.transactionid, payfee.bankname,
payfee.coursename],(err,results)=>{
if(!err){
return res.status(200).json({message:"(: Fee PAYMENT details has Registered:)"});
} else{
return res.status(500).json(err);
}})})
router.get('/get',(req,res,next)=>{
var query =" select *from  payfee order by name"
connection.query(query,(err,results)=>{
if(!err){
return res.status(200).json(results);
} else{
return res.status(500).json(err);}})})
module.exports = router;
```

5.4.4. Router Connection

```
const express = require('express');
var cors = require('cors');
const connection = require('./connection');
const userRoute = require('./routes/user');
const facultyRoute = require('./routes/faculty');
const categoryRoute = require('./routes/category');
const productRoute = require('./routes/product');
const billRoute = require('./routes/bill');
const dashboardRoute = require('./routes/dashboard');
const payfeeRoute = require('./routes/payfee');
const app = express();
app.use(cors());
app.use(express.urlencoded({extended:true}));
app.use(express.json());
```

```
app.use('/user',userRoute);
app.use('/category',categoryRoute);
app.use('/faculty',facultyRoute);
app.use('/product',productRoute);
app.use('/bill',billRoute);
app.use('/dashboard',dashboardRoute);
app.use('/payfee',payfeeRoute);
app.listen(process.env.PORT, () => {
console.log("port is running in 8080");
})
module.exports = app;
```

5.4.5: Managing Branches Dashboard

```
<mat-card>
<b><span>MANAGE CATEGORY</span></b>
<button mat-flat-button color="primary" class="float-right" (click)="handleAddAction()">Add
Category</button>
</mat-card>
<hr>
<mat-card>
<mat-form-field appearance="fill">
<mat-label>Filter</mat-label>
<input matInput (keyup)="applyFilter($event)" #input>
</mat-form-field>
</mat-card>
<hr>
<!-- <mat-icon>edit</mat-icon> -->
</button></td></ng-container> <ng-container matColumnDef="edit">
        <th mat-header-cell *matHeaderCellDef>.</th>
        <td mat-cell *matCellDef="let element" class="action-link">
          <button mat-icon-button color="primary"
matTooltip="Edit"(click)="handleEditAction(element)">
          </button>
        </td><tr mat-header-row *matHeaderRowDef="displayedColumns; sticky:true"></tr>
<tr mat-row *matRowDef="let row;columns:displayedColumns"></tr>
</table></div>
```

5.5 STORED PROCEDURES:

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. Uses for stored procedures include data-validation (integrated into the database) or access-control mechanisms. Furthermore, stored procedures can consolidate and centralize logic that was originally implemented in applications. To save time and memory, extensive or complex processing that requires execution of several SQL statement scan be saved into stored procedures, and all applications call the procedures. One can use nested stored procedures by executing one storedprocedure from within another. In this project two stored procedures are used. One isfor hardware and other is for software.

```
router.post('/add',(req,res,next)=>{
  let user = req.body;
  query ="insert into user
(name,dob,contactNumber,email,password,age,courseid,coursename)
values(?,?,?,?,?,?,?,?)";
  connection.query(query,[user.name,user.dob,user.contactNumber,user.email,user.p
assword,user.age,user.courseid,user.coursename],(err,results)=>{
      if(!err){
          return res.status(200).json({message:"Student Added Successfully"});

      } else{
          return res.status(500).json(err);
      }
  })
})
```

5.6 Trigger

Trigger is a statement that a system executes automatically when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the actionto be performed when the trigger executes. Triggers are used to specify certain integrity constraint and referential constraints that cannot be specified using the constraint mechanism of SQL.

The above trigger is used to calculate the percentage of rating provided by user.

Whenever a new tuple is inserted, this trigger is also updated.

```
router.get('/get',(req,res,next)=>{
  var query =" select *from user order by name";
  connection.query(query,(err,results)=>{
     if(!err){
        return res.status(200).json(results);
 } else{
        return res.status(500).json(err); }})})
```

Chapter 6

SNAPSHOTS

6.1 HOME PAGE

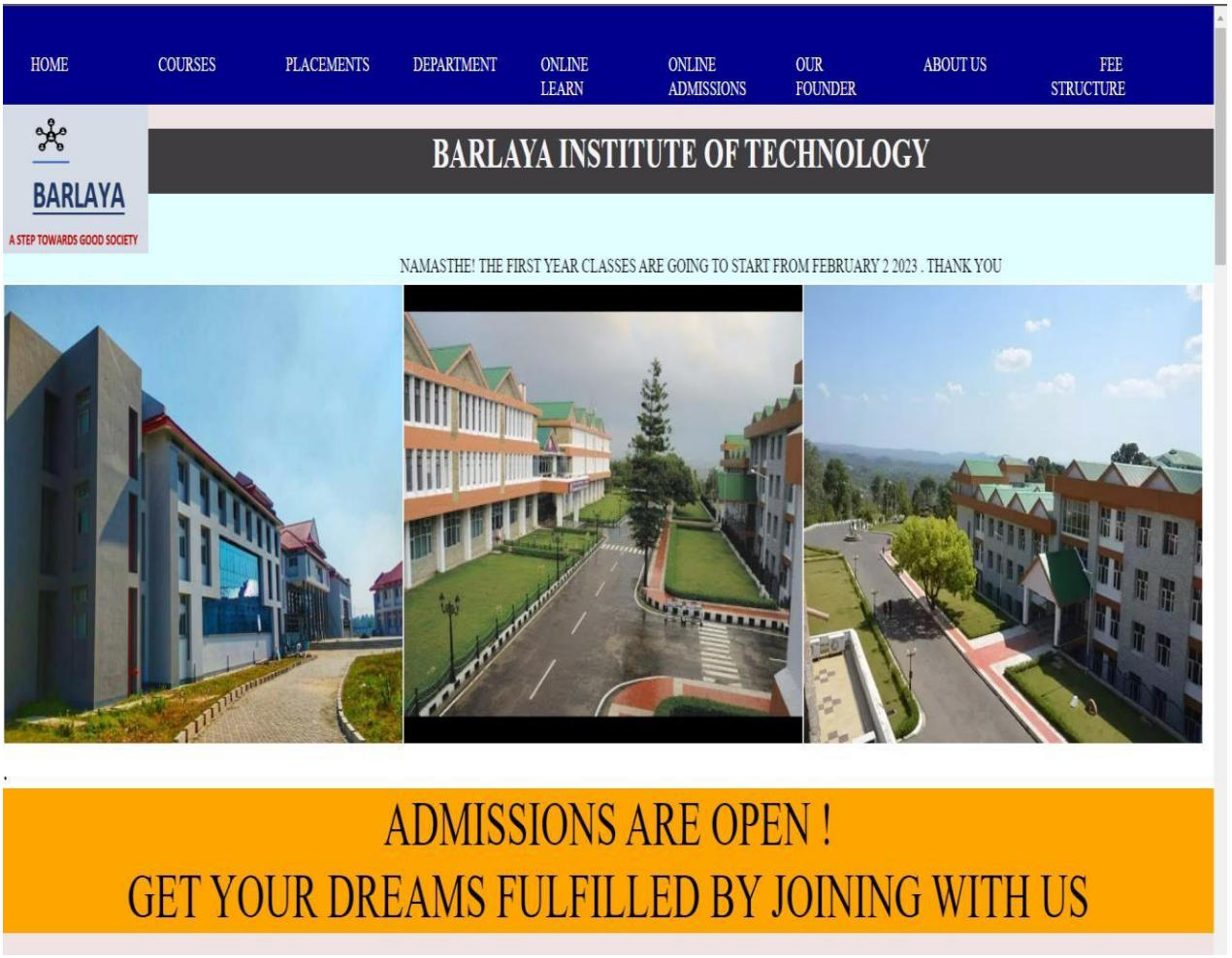
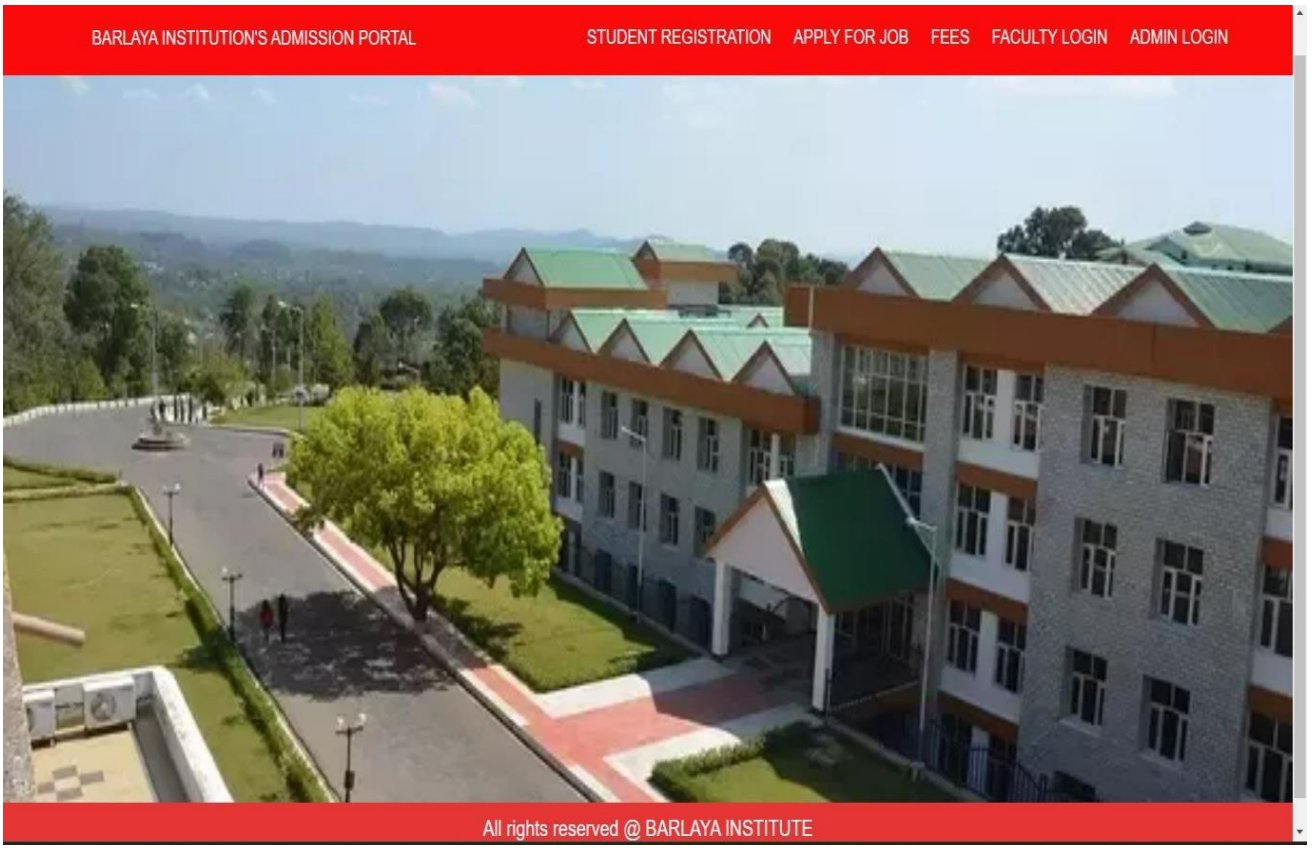


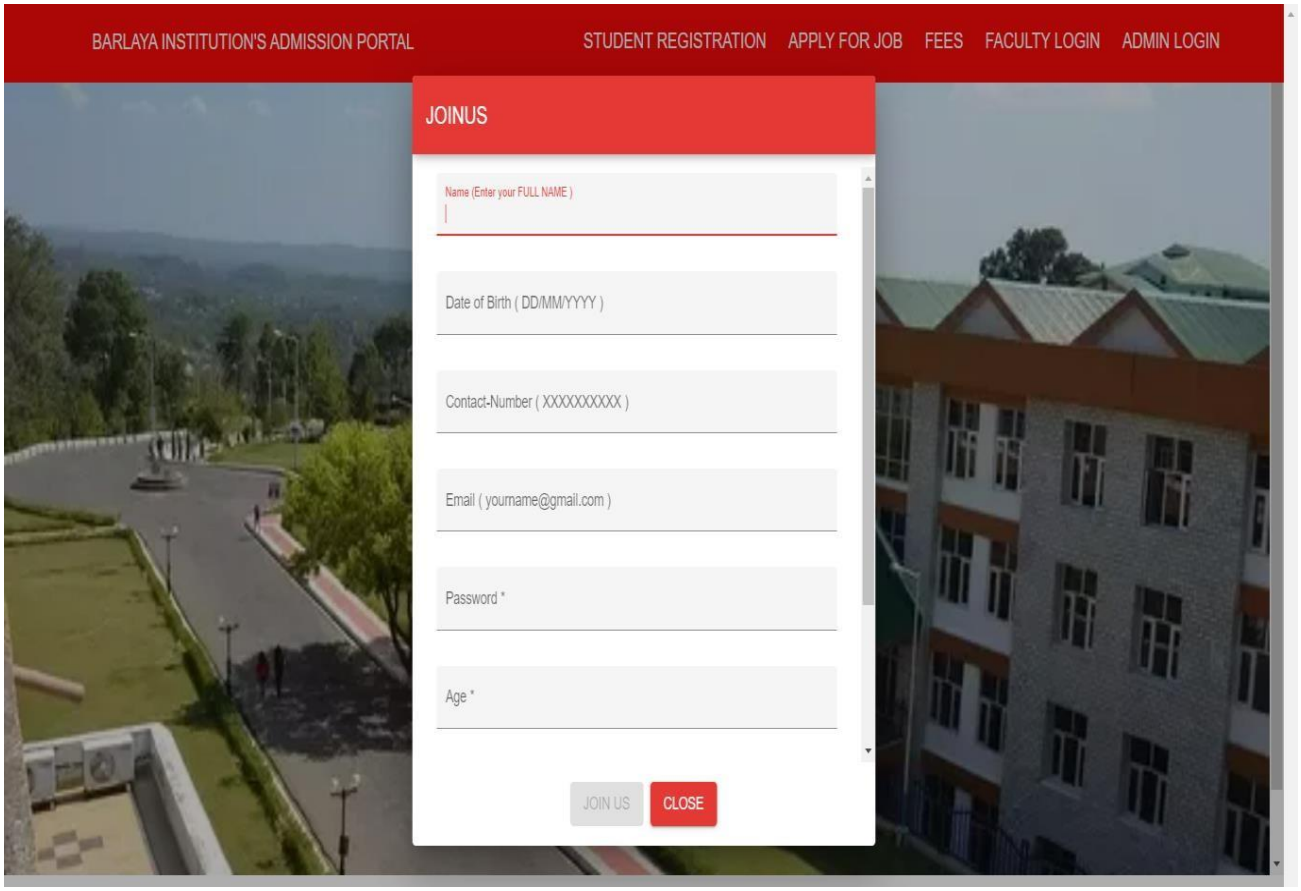
Figure 6.1 Home Page

This is the home page of institution where students can get into online admissions and can get registered.

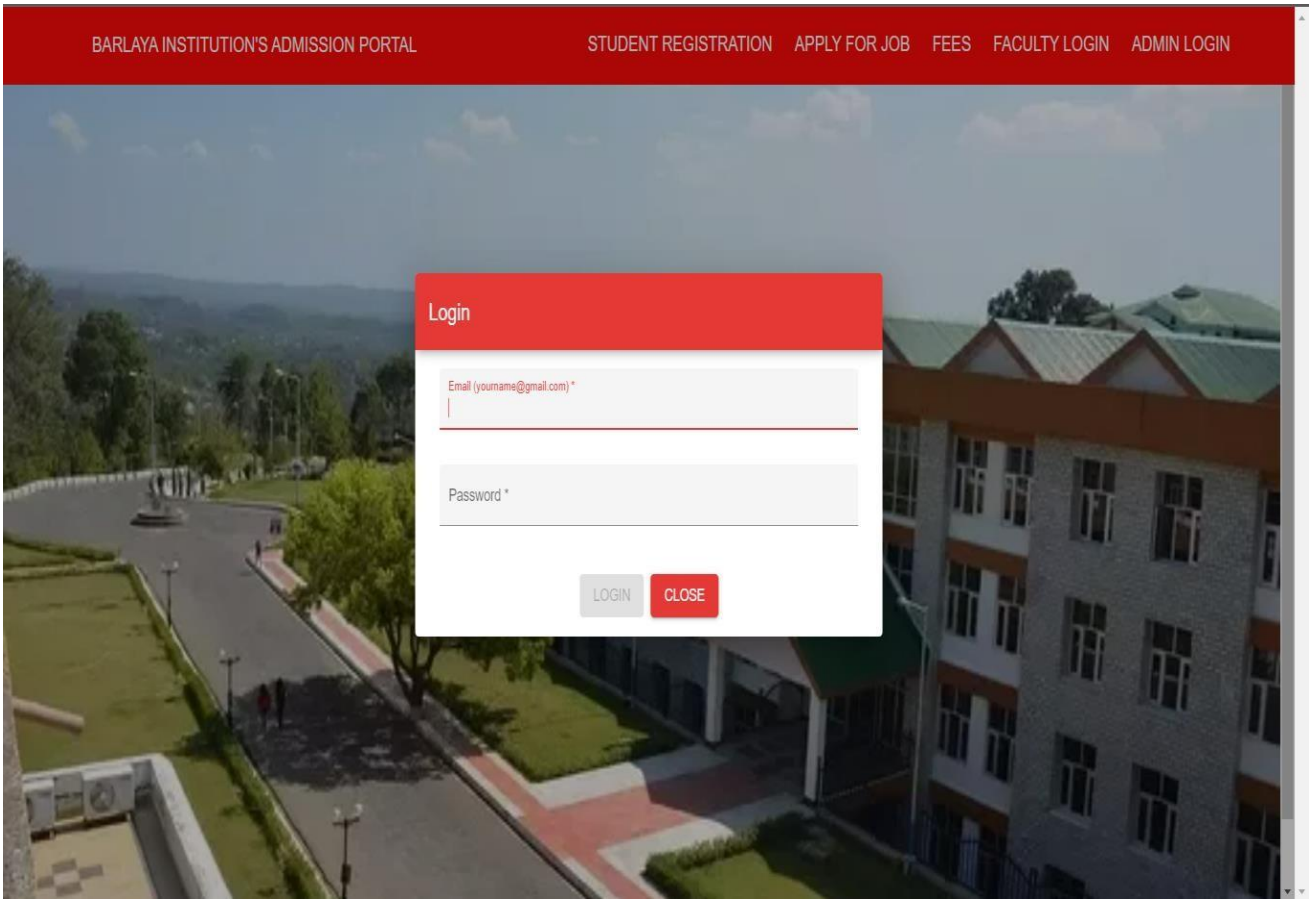
6.2 ADMISSION PORTAL



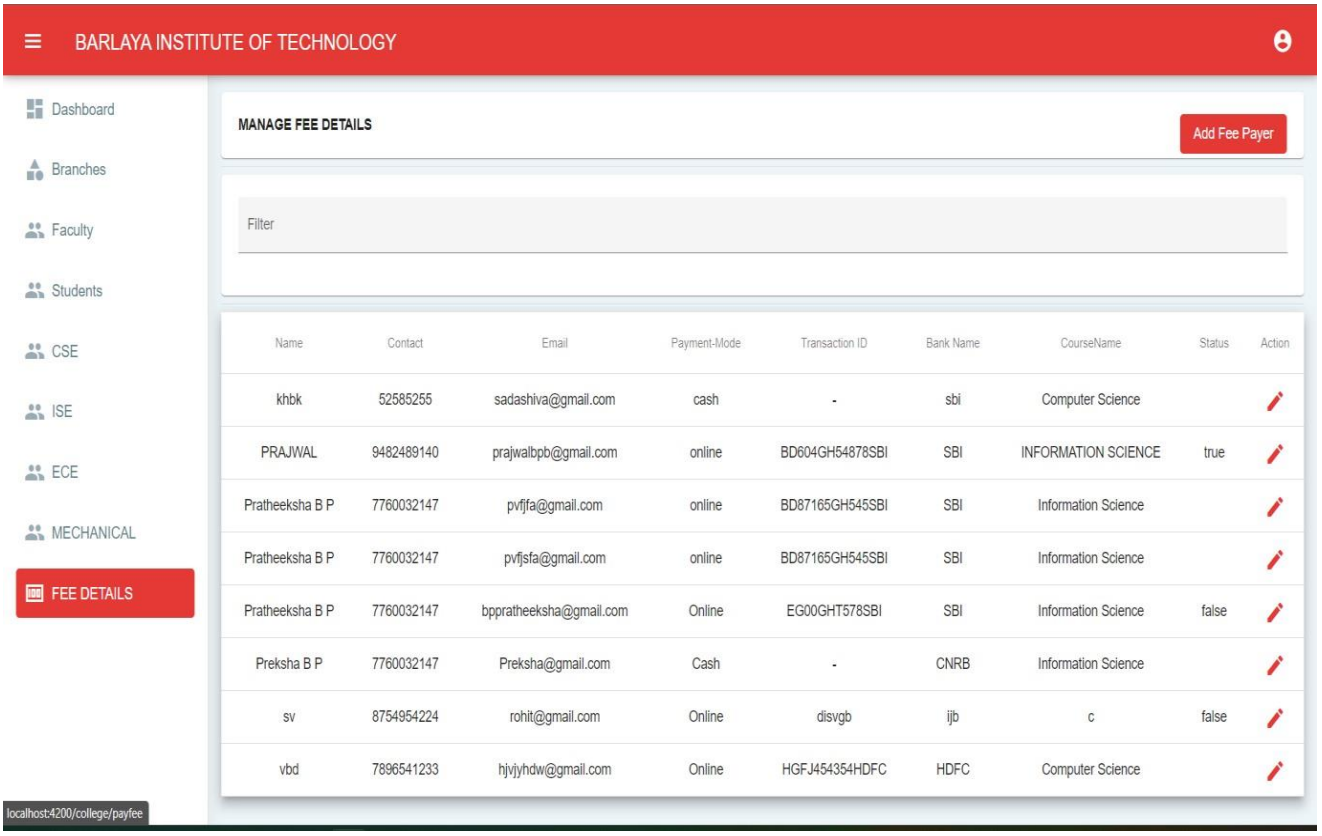
6.3 STUDENT SIGN UP PAGE



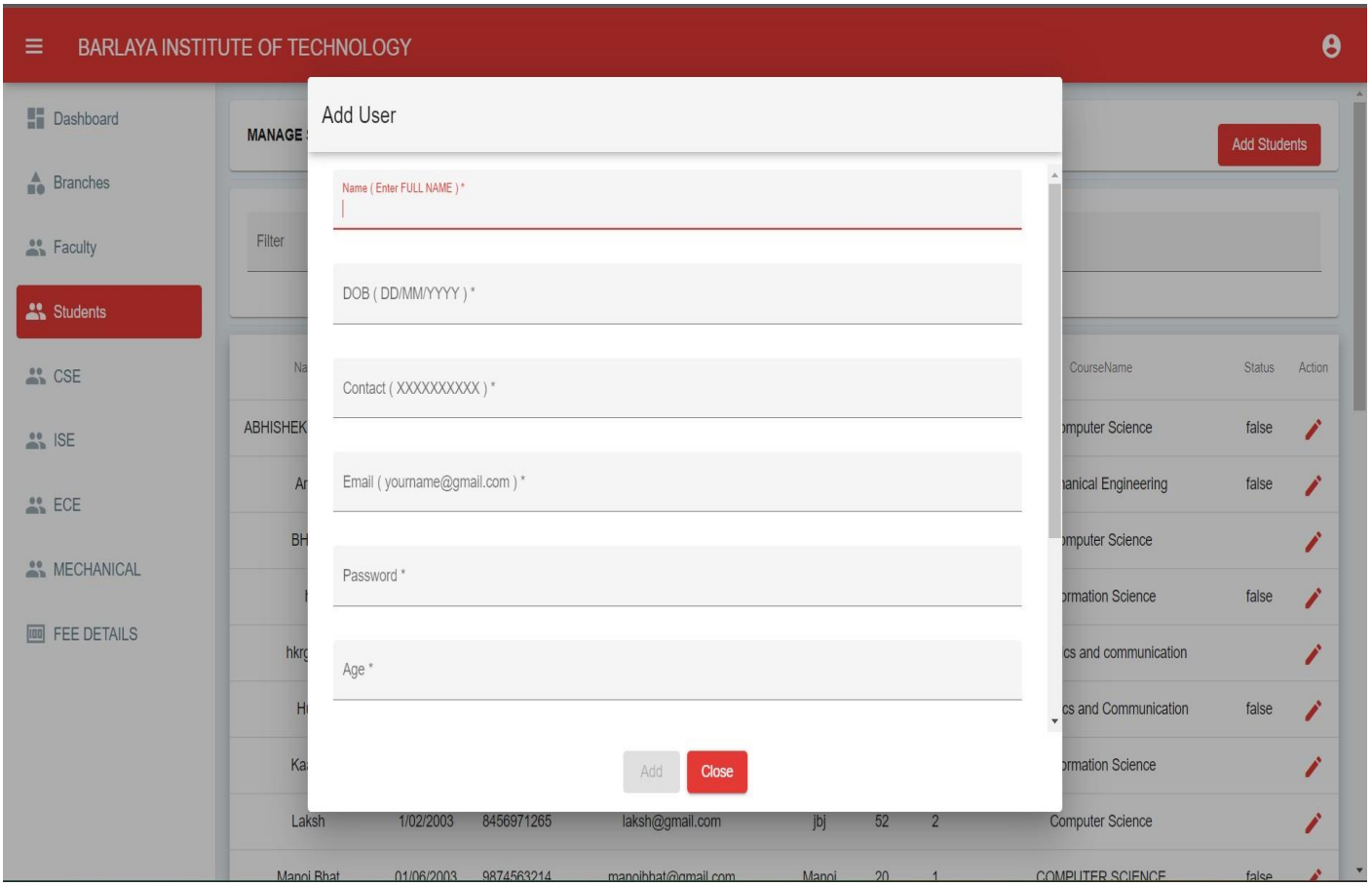
6.4 ADMIN LOGIN



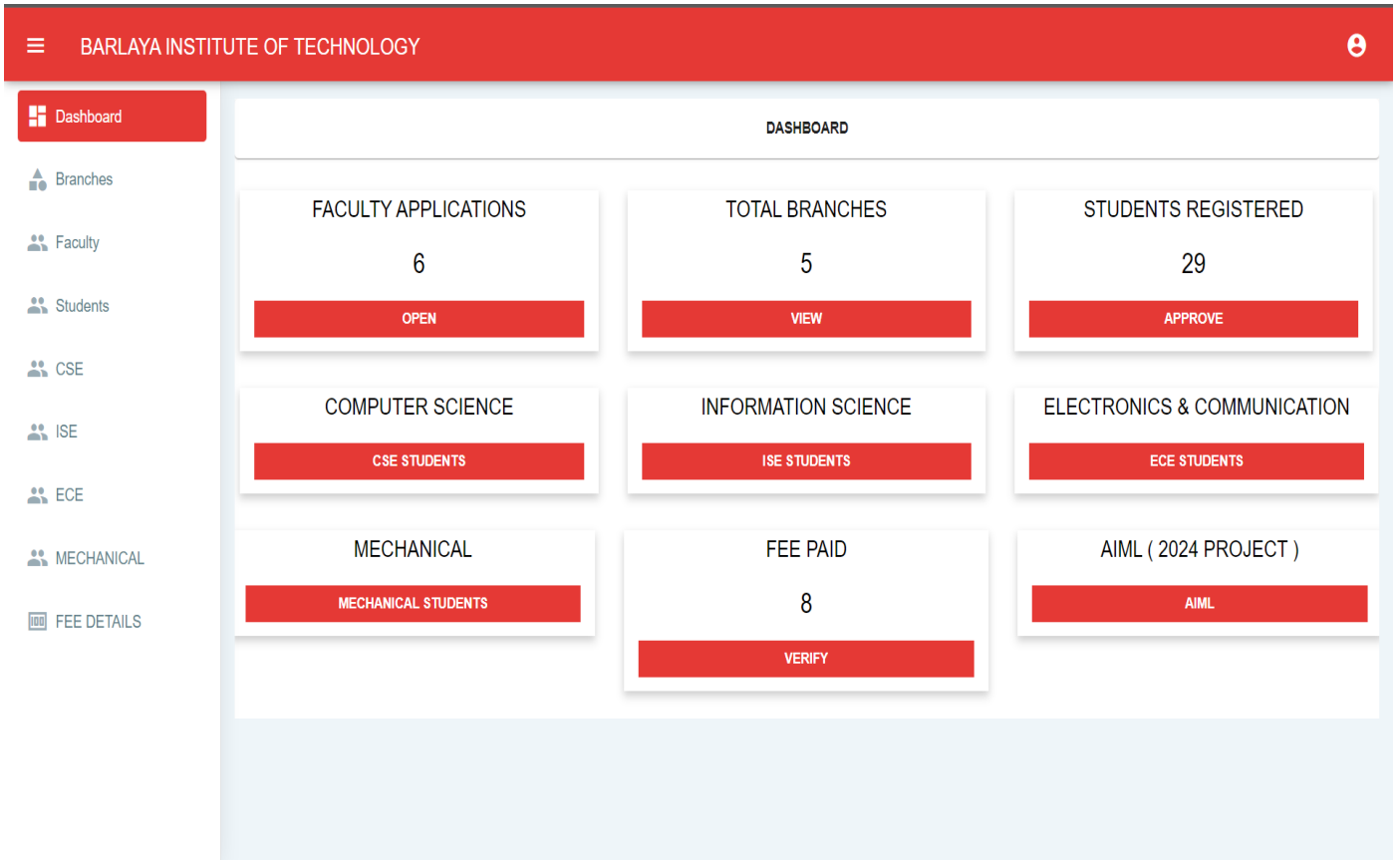
6.5 FEE DETAILS



6.6 Student Registration By Admin



6.7 Dashboard page



6.8 Table Structure

6.8.1 Table Name: Admin

```
mysql> desc admin;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| id       | int          | NO   | PRI | NULL    | auto_increment |
| email    | varchar(50)  | YES  | UNI | NULL    |                |
| password | varchar(250) | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

6.8.2 Table name: User / Student

```
mysql> desc user;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| name          | varchar(250) | YES  |     | NULL    |                |
| dob           | varchar(20)  | YES  |     | NULL    |                |
| contactNumber | varchar(20)  | YES  |     | NULL    |                |
| email         | varchar(50)  | YES  | UNI | NULL    |                |
| password      | varchar(250) | YES  |     | NULL    |                |
| age           | varchar(2)   | YES  |     | NULL    |                |
| courseid      | varchar(10)  | YES  |     | NULL    |                |
| coursename    | varchar(50)  | YES  |     | NULL    |                |
| status        | varchar(20)  | YES  |     | NULL    |                |
| role          | varchar(20)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
11 rows in set (0.00 sec)
```

6.8.3 Table Name: Courses

```
mysql> desc courses
    -> ;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| id         | int          | NO   | PRI | NULL    | auto_increment |
| courseid   | varchar(250) | YES  | UNI | NULL    |                |
| coursename | varchar(250) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)
```

6.8.4 Table Name: Payfees

```
mysql> desc payfee;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| name          | varchar(250) | YES  |     | NULL    |                |
| contactNumber | varchar(20)  | YES  |     | NULL    |                |
| email         | varchar(50)  | YES  | UNI | NULL    |                |
| paymentmode   | varchar(20)  | YES  |     | NULL    |                |
| transactionid | varchar(25)  | YES  |     | NULL    |                |
| bankname      | varchar(10)  | YES  |     | NULL    |                |
| coursename    | varchar(20)  | YES  |     | NULL    |                |
| status        | varchar(20)  | YES  |     | NULL    |                |
| role          | varchar(20)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

6.8.5 Table Name: Faculty

```
mysql> desc faculty;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| name          | varchar(250) | YES  |     | NULL    |                |
| contactNumber | varchar(20)  | YES  |     | NULL    |                |
| email         | varchar(50)  | YES  | UNI | NULL    |                |
| password      | varchar(250) | YES  |     | NULL    |                |
| age           | varchar(2)   | YES  |     | NULL    |                |
| courseid      | varchar(10)  | YES  |     | NULL    |                |
| coursename    | varchar(50)  | YES  |     | NULL    |                |
| status        | varchar(20)  | YES  |     | NULL    |                |
| role          | varchar(20)  | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

Chapter 7

CONCLUSION

The College Database Management has been designed to maintain the records of all the aspects of students with respect to branches selected during admission. This project was designed to model the working of a College website.

It contains all the details about the various branches for admission provided by the college for their students.

The database system project includes triggers that allow the admin to get the students details that are newly added by the admin or the student himself. It also includes a stored procedure to select a details of student's choice. The project provides simple retrieval techniques and easy adding operations thus helping in efficient maintenance of records.

FUTURE ENHANCEMENTS

Just like any other developer this project is the most basic website built using simple tools. We seek to increase the dynamic of the project by adding various other innovations to it. Such innovations would seem possible only with time, which we lack but regardless we strive to complete what we started.

We believe that apart from the present functionalities we can add:

"ONLINE PAYMENT GENERATING PDF" which will recommend various options for payment for student based on his convenience.

"FACILITY TO DOCUMENT UPLOAD DOCUMENTS" so that we can easily access to registered students.

Apart from these changes we are open to various suggestions and hope to implement them soon so that this website can be used by the students and faculties for their needs

REFERENCES

- Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, PEARSON, Fifth Edition.
- Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, McGRAW HILL, Third Edition.
- https://www.w3schools.com/html/
- https://www.w3schools.com/css/
- https://www.geeksforgeeks.org
- www.stackoverflow.com
- https://www.postman.com/
- https://youtu.be/ZHg2u4QNUdw