**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnanasangama”, Belagavi-590018, Karnataka**

****

**BANGALORE INSTITUTE OF TECHNOLOGY**

**K.R. Road, V.V.Puram, Bangalore-560 004**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**DATABASE MANAGEMENT SYSTEM MINI PROJECT**

**18CSL58**

**“STUDENT RESULT MANAGEMENT SYSTEM”**

**Submitted By**

**Bharath Gowda B 1BI19CS038**

**for the academic year 2021-22**

**Department of Computer Science & Engineering**

**Bangalore Institute of Technology**

**K.R. Road, V.V.Puram, Bangalore-560 004**

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnanasangama”, Belagavi-590018, Karnataka**

**BANGALORE INSTITUTE OF TECHNOLOGY**

**K.R. Road, V.V.Puram, Bangalore-560 004**

****

**Department of Computer Science & Engineering**

***Certificate***

This is to certify that the implementation of **DBMS MINI PROJECT** entitled   
“**STUDENT RESULT MANAGEMENT SYSTEM**” has been successfully completed by

**USN: 1BI19CS038 NAME: BHARATH GOWDA B**

of V semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in Computer Science & Engineering of the Visvesvaraya Technological University during the academic year 2021-2022.

**Lab In charge :**

**Prof. Tejashwini P S Dr. J Girija**

Assistant Professor Professor and Head

Dept. of CS&E Department of CS&E

Bangalore Institute of Technology Bangalore Institute of Technology

Bangalore Bangalore

Examiners: 1) 2)

**ACKNOWLEDGEMENT**

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to Head of Department Dr. J Girija for being kind enough to provide the necessary support to carry out the mini project. I am most humbled to mention the enthusiastic influence provided by the lab in-charges

Prof. Tejashwini P S, on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the staff members of Computer Science Department, BIT

**Bharath Gowda B**

**1BI19CS038**

**CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CH.NO.** |  | **TITLE** | **PAGE NO.** |
| 1. |  | Introduction |  |
| 1.1 | Overview | 1 |
| 1.2 | Problem Statement | 1 |
| 2. |  | Back End Design |  |
| 2.1 | Conceptual Database Design | 2 |
| 2.2 | Logical Database Design | 3 |
| 2.3 | Normalization | 4 |
| 3. |  | Front End Design |  |
| 3.1 | Html | 9 |
| 3.2 | CSS | 9 |
| 3.3 | Java script | 10 |
| 3.4 | Hardware and Software Configuration | 10 |
| 4. |  | Major modules | 11 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. |  | Implementation |  |
| 5.1 | Create Statements | 12 |
| 5.2 | Front End Code | 14 |
| 5.3 | Back End Code | 25 |
| 5.4 | SQL Queries | 41 |
| 6 | 6.1 | Snapshots | 43 |
| 7. | 7 | Applications | 48 |
| 8. | 8 | Conclusion | 49 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIG.NO.** | **NAME** | **PAGE NO.** |
| 2.1 | E R diagram | 2 |
| 2.2 | E R Mapping | 3 |
| 6.1 | Login Page | 43 |
| 6.2 | Home Page | 43 |
| 6.3 | Student - Professor Page | 44 |
| 6.4 | Student - Result Page | 44 |
| 6.5 | Faculty – Course Page | 45 |
| 6.6 | Faculty – Student List Page | 45 |
| 6.7 | Faculty – Update Marks Page | 45 |
| 6.8 | Admin – Create Account | 46 |
| 6.9 | Admin – Get Info Page | 47 |
| 6.10 | Admin – Course List Page | 47 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **NAME** | **PAGE NO.** |
| 1. | FACULTY | 5 |
| 2. | COURSE | 5 |
| 3. | DEPARTMENT-COURSE-LIST | 6 |
| 4. | DEPARTMENT COURSE | 6 |
| 5. | DEPARTMENT | 7 |
| 6. | COURSE-LIST | 7 |
| 7. | FACULTY TAKES | 7 |
| 8. | STUDENT | 8 |
| 9. | RESULTS | 8 |

**CHAPTER 1**

**INTRODUCTION**

**INTRODUCTION**

* 1. **OVERVIEW**

Student Result Management System is a web-based application that mainly focuses on providing the results to the student and the faculty. The student check their respective results using their University registered recognition id’s along with their grades and percentage of that particular semester.

The student accessing their results through college site is more convenient and the faculty can easily analyse the pass and failure of a particular subject. The system is divided into three modules- Student, Faculty and Administrator. The student using his roll number can view his results and the faculty using the joining year and the subject name and view the analysis of pass and failure count in the selected subject.

**1.2 PROBLEM STATEMENT**

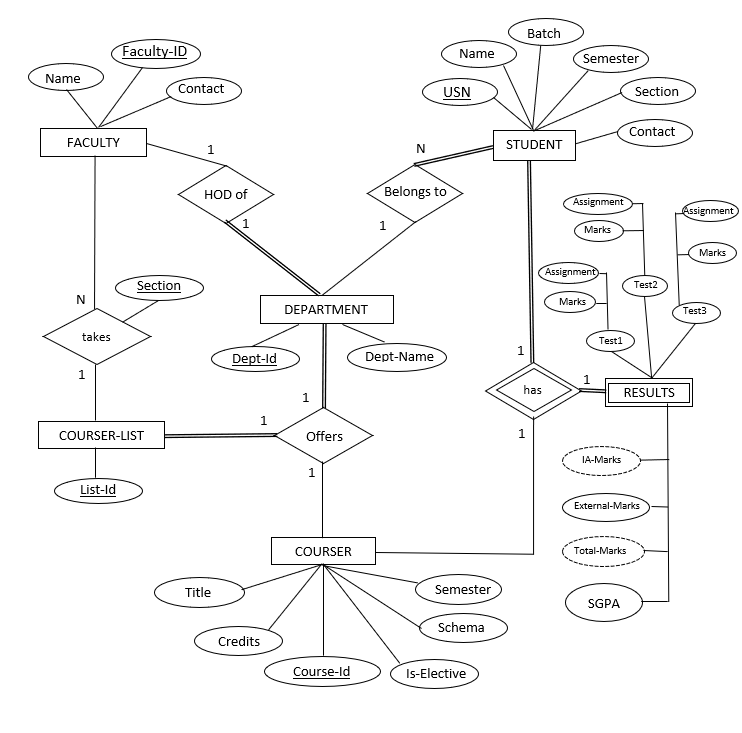
“To design and develop a system for managing the results of students.”

**CHAPTER 2**

**BACK END DESIGN**

**BACKEND DESIGN**

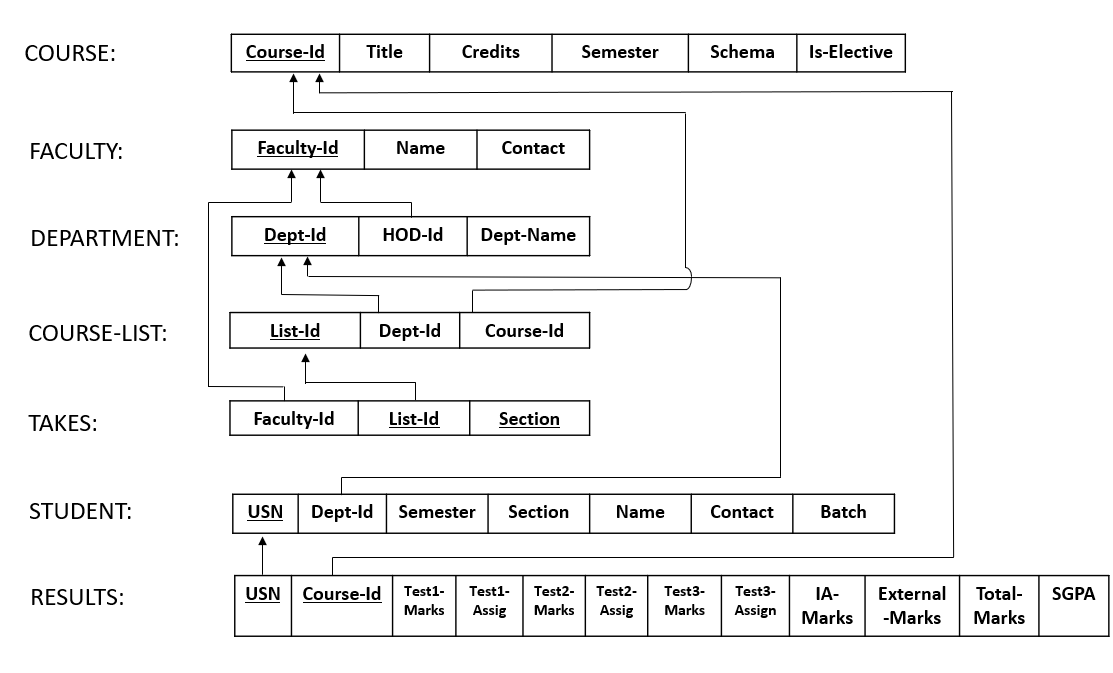
**2.1 CONCEPTUAL DATABASE DESIGN**



M

**2.1 ER DIAGRAM**

**2.2 LOGICAL DATABASE DESIGN**

****

**2.2 ER TO RELATIONAL MAPPING**

**2.3 NORMALISATION**

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

* Eliminating redundant(useless) data.
* Ensuring data dependencies make sense i.e. data id logically stored.

**FIRST NORMAL FORM (1NF):**

As per First Normal Form

1. There are no duplicated rows in the table.
2. Each cell is single valued or atomic.

**SECOND NORMAL FORM (2NF):**

As per Second Normal Form, a table is in 2NF iff it is in 1NF and every non prime attribute is not partially dependent on any key of the table.

**THIRD NORMAL FORM (3NF):**

Third Normal Form applies that every non-prime attribute of table must be dependent on primary key, or we can say that, there should not be the case that a non-prime attribute is determined by another non-prime attribute. So this *transitive functional* *dependency* should be removed from the table and also the table must be in the SecondNormal Form.

**2.3.1 NORMALISATION OF LOGIN**

**FACULTY:**

|  |  |  |
| --- | --- | --- |
| **Faculty-Id** | **Name** | **Contact** |
| F100 | Shobha | 9900783185 |
| F101 | Kiran | 9988075452 |
| F102 | Yashwanth | 9986451235 |
| F103 | Kishor | 7844561540 |
| F104 | Suma | 9845540044 |

* + FD = {Faculty-Id 🡪 {Name, Contact}}
  + The FACULTY relation is in 1NF since all columns have atomic and unique values.
  + The relation is in 2NF since every nonprime attribute in FACULTY is fully functionally dependent on the primary key **Faculty-Id**.
  + None of the non-prime attribute of FACULTY is transitively dependent on the primary key. So, it is in 3NF.

**COURSE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Couse-Id** | **Title** | **Credits** | **Semester** | **Schema** | **Is-Elective** |
| 18MAT11 | Math-I | 4 | 1 | 2018 | 0 (False) |
| 18CS54 | ATCI | 3 | 5 | 2018 | 0 (False) |
| 18CS641 | Advance-Java | 3 | 6 | 2018 | 1 (True) |
| 18CS642 | Image-Processing | 3 | 6 | 2018 | 1 (True) |
| 18EE78 | Adv-Electronics | 3 | 7 | 2018 | 0 (False) |

* FD = {Course-Id 🡪 {Title, Credits, Semester, Schema, Is-Elective}}
  + The COURSE relation is in 1NF since all columns have atomic and unique values.
  + The relation is in 2NF since every nonprime attribute in COURSE is fully functionally dependent on the primary key **Course-Id**.
  + None of the non-prime attribute of COURSE is transitively dependent on the primary key. So, it is in 3NF.
* **Consider coursers offered by a department including faculty in-charge for each course:**

**Department-Course-List: ( R )**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **List-Id** | **Dept-Id** | **Dept-Name** | **Hod-Id** | **Course-Id** | **Faculty-Id** | **Section** |
| L100 | D100 | CSE | F100 | 18CS54 | F102 | A |
| L100 | D100 | CSE | F100 | 18CS54 | F102 | B |
| L101 | D100 | CSE | F100 | 18CS641 | F103 | A |
| L102 | D100 | CSE | F100 | 18MAT11 | F104 | A |
| L103 | D101 | EEE | F101 | 18MAT11 | F104 | A |
| L104 | D101 | EEE | F101 | 18EE78 | F101 | A |
| L104 | D101 | EEE | F101 | 18EE78 | F101 | B |

FD: {

List-Id 🡪 {Dept-Id, Course-Id}

Dept-Id 🡪 {Dept-Name, Hod-Id}

{List -Id, Section} 🡪 Faculty-Id

}

Candidate key:

{List-Id, Section}+  = {Dept-Id, Dept-Name, Hod-Id, Course-ID, Faculty-Id}

Prime Attribute: {List-Id, Section}

Non-Prime Attribute: {Dept-Id, Dept-Name, Hod-Id, Course-ID, Faculty-Id}

* + The given relation R is in 1NF since all columns have atomic and unique values.
  + The relation is not in 2NF since Dept-Id and Course-Id is dependent only on List-Id which is a proper subset of Candidate Key.
  + Divide the Relation into two relations, one Department-Courses((List-Id, Course-Id, Dept-Id, Dept-Name, Hod-Id), another one Faculty-Takes(Faculty-Id, List-Id, Section}

**DEPARTMENT-COURSE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **List-Id** | **Course-Id** | **Dept-Id** | **Dept-Name** | **Hod-Id** |
| L100 | 18CS54 | D100 | CSE | F100 |
| L101 | 18CS641 | D100 | CSE | F100 |
| L102 | 18MAT11 | D100 | CSE | F100 |
| L103 | 18MAT11 | D101 | EEE | F101 |
| L104 | 18EE78 | D101 | EEE | F101 |

* + The relation is in 2NF since every nonprime attribute is fully functionally dependent on the primary key **List-Id**.
  + The non-prime attribute Dept-Name and Hod-Id are transitively dependent on List-Id. Hence the relation is not in 3NF
  + Divide the Relation into two relations, one Department(Dept-Id, Dept-Name, Hod-Id) another one Courses-List(List-Id, Course-Id, Dept-Id,}

**DEPARTMENT:**

|  |  |  |
| --- | --- | --- |
| **Dept-Id** | **Dept-Name** | **Hod-Id** |
| D100 | CSE | F100 |
| D101 | EEE | F101 |

* None of the non-prime attribute of DEPARTMENT is transitively dependent on the primary key. So, it is in 3NF.

**COURSE-LIST:**

|  |  |  |
| --- | --- | --- |
| **List-Id** | **Dept-Id** | **Course-Id** |
| L100 | 18CS54 | D100 |
| L101 | 18CS641 | D100 |
| L102 | 18MAT11 | D100 |
| L103 | 18MAT11 | D101 |
| L104 | 18EE78 | D101 |

* None of the non-prime attribute of STUDENT is transitively dependent on the primary key. So, it is in 3NF.

**FACULTY-TAKES:**

|  |  |  |
| --- | --- | --- |
| **Faculty-Id** | **List-Id** | **Section** |
| F102 | L100 | A |
| F102 | L100 | B |
| F103 | L101 | A |
| F104 | L102 | A |
| F104 | L103 | A |
| F101 | L104 | A |
| F101 | L104 | B |

* None of the non-prime attribute of FACULTY\_TAKES is transitively dependent on the primary key. So, it is in 3NF.

**STUDENT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **USN** | **Name** | **Contact** | **Dept-Id** | **Batch** | **Semester** | **Section** |
| 1BI18CS001 | Girish | 9988775461 | D100 | 2018 | 7 | A |
| 1BI19CS002 | Vivek | 9845678541 | D100 | 2019 | 5 | A |
| 1BI19CS058 | Hairsh | 9655874575 | D100 | 2019 | 5 | B |
| 1BI19CS161 | Suyog | 6988450210 | D101 | 2019 | 5 | B |

* + The STUDENT relation is in 1NF since all columns have atomic and unique values.
  + The relation is in 2NF since every nonprime attribute in STUDENT is fully functionally dependent on the primary key **USN**.
  + None of the non-prime attribute of STUDENT is transitively dependent on the primary key. So, it is in 3NF.

**RESULTS:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **USN** | **Course-Id** | **Test-1**  **Marks** | **Test-I**  **Ass.M** | **..** | **IA**  **Marks** | **External**  **Marks** | **Total**  **Marks** | **SGPA**  **Points** |
| 1BI19CS002 | 18MAT11 | 30 | 10 |  | 40 | 49 | 89 | 36 |
| 1BI19CS002 | 18CS54 | 29 | 8 |  | 35 | 45 | 80 | 27 |
| 1BI19CS058 | 18CS54 | 30 | 10 |  | 38 | 48 | 86 | 27 |
| 1BI19CS161 | 18MAT11 | 25 | 7 |  | 38 | 50 | 88 | 36 |
| 1BI19CS161 | 18EE78 | 15 | 8 |  | 30 | 35 | 75 | 24 |

* + The RESULTS relation is in 1NF since all columns have atomic and unique values.
  + The relation is in 2NF since every nonprime attribute in RESULTS is fully functionally dependent on the primary key **(USN, Course-Id)**.
  + None of the non-prime attribute of RESULTS is transitively dependent on the primary key. So, it is in 3NF.

**CHAPTER 3**

**FRONT END DESIGN**

**FRONT END DESIGN**

**3.1 HTML**

Hypertext Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets, within the web page content. HTML tags most commonly come in pairs like and , although some tags represent empty elements and so are unpaired. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. CSS Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can also be applied to any kind of XML document. CSS is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content.

**3.2 CASCADING STYLE SHEETS (CSS)**

**Cascading Style Sheets** (**CSS**) is a [style sheet language](https://en.wikipedia.org/wiki/Style_sheet_language) used for describing the [presentation](https://en.wikipedia.org/wiki/Presentation_semantics) of a document written in a [markup language](https://en.wikipedia.org/wiki/Markup_language) such as [HTML](https://en.wikipedia.org/wiki/HTML).[[1]](https://en.wikipedia.org/wiki/CSS#cite_note-1) CSS is a cornerstone technology of the [World Wide Web](https://en.wikipedia.org/wiki/World_Wide_Web), alongside HTML and [JavaScript](https://en.wikipedia.org/wiki/JavaScript). CSS is designed to enable the separation of presentation and content, including [layout](https://en.wikipedia.org/wiki/Page_layout), [colors](https://en.wikipedia.org/wiki/Color), and [fonts](https://en.wikipedia.org/wiki/Typeface). This separation can improve content [accessibility](https://en.wikipedia.org/wiki/Accessibility) provide more flexibility and control in the specification of presentation characteristics enable multiple [web pages](https://en.wikipedia.org/wiki/Web_page) to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content and enable the .css file to be [cached](https://en.wikipedia.org/wiki/Cache_(computing)) to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice and on [Braille-based](https://en.wikipedia.org/wiki/Braille_display) tactile devices. CSS also has rules for alternate formatting if the content is accessed on a [mobile device](https://en.wikipedia.org/wiki/Mobile_device).

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the [World Wide Web Consortium](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C). Internet media type ([MIME type](https://en.wikipedia.org/wiki/MIME_media_type))  is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free [CSS validation service](https://en.wikipedia.org/wiki/W3C_Markup_Validation_Service#CSS_validation) for CSS documents.

In addition to HTML, other markup languages support the use of CSS including [XHTML](https://en.wikipedia.org/wiki/XHTML), [plain XML](https://en.wikipedia.org/wiki/Plain_Old_XML), [SVG](https://en.wikipedia.org/wiki/Scalable_Vector_Graphics), and [XUL](https://en.wikipedia.org/wiki/XUL).

**3.3 JAVA SCRIPT**

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multiparadigm. It has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it for client-side page behaviour and all major web browsers have a dedicated JavaScript engine to execute it.

**3.4 HARDWARE AND SOFTWARE CONFIGURATION**

**Hardware Configuration**

1. Processor : Intel Pentium 4.0
2. Ram : 2GB
3. Hard disk : 500GB

**Software Configuration**

1. Operating system : windows 10
2. Frond-End : HTML, JAVA SCRIPT
3. Back-End : NodeJS, Express
4. Database : ORACLE

**CHAPTER 4**

**MAJOR MODULES**

**MAJOR MODULES**

* **Admin:**
  + Admin has complete access to all the resources. Admin creates new users (like faculty, student), update details of the user. Any complex operations required are done by admin.
* **Faculty:**
  + **Login:** Faculty can be both teaching and non-teaching staff.Admin creates account for new faculty and provides them with user-id and password. Faculty can login into portal using the same.
  + **Course-taken:** Faculty can take different course to different classes, Course-taken module gives details about it.
  + **Update-Marks:** This module allows a Faculty to retrieve, update marks of each students in a particular test. The output of this module will be in table form.
  + **CSV-Operation:** Thismodule allows a Faculty to export results of student in csv format. Which can be opened in Excel for any further operations.Faculty can import a csv file to update marks as long as the file follows required constraints.
  + **Maths-Operation:** This module allows a Faculty touse simple math operations (like addition, subtraction, average) for updating marks. Ex. Operation to find the final internal marks based on three tests average.
* **Department:**
  + **HOD:** For each department there is one Faculty as HOD. This module allows the HOD to monitor other faculties working for the department.
  + **Course-Offered:** The HOD can add or remove courses offered by the department for a particular semester based on university schema.
* **Student:** 
  + **Login:** Student can use there USN (or any Unique ID provided by the Institution) and password distributed by admin as user credentials.
  + **Check-Result:** This module allows theStudents to check their marks (updated by faculty in-charge). Students can also check their previous semester results.
  + **Contact-Faculty:** This moduleconnects Student with faculty in-charge for any querying. Student can only contact the faculty in-charge of a particular course in current semester of the student.

**CHAPTER 5**

**IMPLEMENTATION**

**IMPLEMENTATION**

**5.1 CREATE STATEMENTS**

Create table user\_login(

user\_id varchar(20) primary key,

password varchar(20) not null,

type number(1) default 2,

token varchar(20),

login\_fail number(2) default 0,

token\_fail number(5) default 0

);

create table course(

course\_id varchar(10) primary key,

title varchar(30) not null,

semester number(2),

schema number(4),

credits number(2),

isEelective number(1),

constraint ckc check(credits >=0 and semester > 0)

);

create table faculty(

faculty\_id varchar(20),

name varchar(20) not null,

contact number(10),

photo\_container varchar(10) default null,

constraint pkf primary key(faculty\_id),

constraint fkf foreign key(faculty\_id) references user\_login(user\_id) on delete cascade

);

create table department(

dept\_id varchar2(10),

hod\_id varchar2(20),

dept\_name varchar2(50) not null,

constraint pkd primary key(dept\_id),

constraint fkd foreign key(hod\_id) references faculty(faculty\_id) on delete set null

);

create table course\_list(

list\_id varchar(10),

course\_id varchar(10),

dept\_id varchar(10),

constraint pkcf primary key(list\_id),

constraint fkcf1 foreign key(dept\_id) references department(dept\_id) on delete cascade,

constraint fkcf2 foreign key(course\_id) references course(course\_id) on delete cascade,

constraint ukcf unique (dept\_id,course\_id)

);

create table takes(

faculty\_id varchar(20),

list\_id varchar(10),

section varchar(2),

constraint pkct primary key(list\_id,section),

constraint fkct1 foreign key(faculty\_id) references faculty(faculty\_id) on delete cascade,

constraint fkct foreign key(list\_id) references course\_list(list\_id) on delete cascade

);

create table student(

usn varchar(20),

name varchar(30),

dept\_id varchar(10),

batch number(4),

semester number(2) default 0,

section varchar(2),

contact number(10),

photo\_container varchar(10) default null,

constraint pkss primary key(usn),

constraint fks1 foreign key(dept\_id) references department(dept\_id) on delete cascade,

constraint fks2 foreign key(usn) references user\_login(user\_id) on delete cascade

);

create table results(

usn varchar(20),

course\_id varchar(20),

test1\_marks number(3) default -2,

test1\_assignment number(3) default -2,

test2\_marks number(3) default -2,

test2\_assignment number(3) default -2,

test3\_marks number(3) default -2,

test3\_assignment number(3) default -2,

IA\_marks number(3) default -2,

external\_marks number(3) default -2,

total number(3) default -2,

sgpa number(3) default -2,

constraint pkr primary key(usn,course\_id),

constraint fkr1 foreign key(usn) references student(usn) on delete cascade,

constraint fkr2 foreign key(course\_id) references course(course\_id) on delete cascade

);

**5.2 FRONT END CODE**

**5.2.1 LOGIN PAGE**

<html>

<head>

<title>

BIT-Result Portal

</title>

<script src='./static/JS/axios.min.js'></script>

<link rel="stylesheet" href='./static/CSS/login.css'>

</head>

<body>

<div id='home-page'>

<div class='left-home' onclick="window.open('https://bit-bangalore.edu.in')">

<img class='background' src='./static/images/background1.jpg'>

</div>

<div class='right-home'>

<div class='main-form'>

<form class='login-form' id='id-login-form'>

<h3> STUDENT / FACULTY</h3>

<div>

<input id='username' type="text" placeholder="username" required>

</div>

<div>

<input id='password' type="password" placeholder="password" required>

</div>

<div >

<button id='loginButton' type='submit'>

<span>LOGIN</span>

</button>

</div>

<span style='color:red;font-style: italic;font-size: 14px;' id='authentication-error'></span>

</form>

<div class='quick-link'>

<div>

<a href="https://vtu.ac.in/">

<img src="static/icons/vtulogo.png"/>

</a>

</div>

<div>

<a href='https://www.linkedin.com/school/bitsince1979/'>

<img src="static/icons/linkedin--v1.png"/>

</a>

</div>

<div>

<a href="https://www.facebook.com/bitsince1979">

<img src="static/icons/facebook--v1.png"/>

</a>

</div>

<div>

<a href="https://www.instagram.com/bitsince1979/">

<img src="static/icons/instagram-new--v2.png"/>

</a>

</div>

<div>

<a href="https://www.twitter.com/bitsince1979">

<img src="static/icons/twitter--v1.png"/>

</a>

</div>

<div>

<a href="https://www.youtube.com/bitsince1979">

<img src="static/icons/youtube-play.png"/>

</a>

</div>

<div>

<a href="https://goo.gl/maps/AsrhV4234NU3k8fa8">

<img src="static/icons/youtube-play.png"/>

</a>

</div>

</div>

</div>

</div>

</div>

<script>

document.getElementById('id-login-form').addEventListener('submit',function(e){

e.preventDefault();

login();

})

async function login(){

var user = {

'username' : document.getElementById('username').value,

'password' : document.getElementById('password').value

}

var res = await axios.post('/home/login',user)

console.log(res.data)

if(res.data.error){

document.getElementById('authentication-error').innerHTML = 'username or password incorrect'

document.getElementById('password').value = ''

}

else{

window.sessionStorage.setItem('username',user.username)

window.sessionStorage.setItem('token',res.data.token)

if(res.data.type.toLowerCase() == 'student'){

window.open('/student','\_self')

}

else if(res.data.type.toLowerCase() == 'admin'){

window.open('/admin','\_self')

}

else if(res.data.type.toLowerCase() == 'faculty'){

window.open('/faculty','\_self')

}

}

}

</script>

</body>

</html>

**5.2.2 STUDENT PAGE**

<html>

<head>

<title>

BIT-Result Portal

</title>

<script src='./static/JS/axios.min.js'></script>

<link rel="stylesheet" href='./static/CSS/navBar.css'>

<link rel="stylesheet" href='./static/CSS/faculty-home.css'>

<link rel="stylesheet" href='./static/CSS/table.css'>

</head>

<style>

#root{

min-height: 100vh;

}

.table-list th{

min-width: 80px;

}

</style>

<body>

<nav class="navbar">

<div class='nav-left'>

<div class='app-logo'><img src='./static/images/logo.png' class="medium-icon"></div>

<div class='app-name'> Bangalore Institute Of Technology </div>

</div>

<div class='nav-right'>

<ul>

<li class='active' id='main-options-home'><a>Home</a></li>

<li id='main-options-professors'><a >Professors</a></li>

<li id='main-options-results'><a >Results</a></li>

<li class='nav-userinfo'>

<label class="nav-userpic" for='upload-pic'>

<img id='user-icon' src='static/icons/thumnail.png'>

</label>

<input type='file' id='upload-pic' accept="image/\*" style="display: none;">

<div class='nav-username' id='username'></div>

</li>

</ul>

</div>

</nav>

<div id='home-render' style='min-height: 100vh;'>

</div>

<div id='root' style='display: none;'>

<style>

<div class='hod-panel' id='professors-hod-pannel' style='display: none;' >

<div id='hod-name' style='padding-left:15px;min-width:15%'></div>

<div id='dept-name'></div>

<div id='hod-contact' style='padding-right:15px;width:15%'></div>

</div>

<div id='main-table'>

</div>

</div>

<div >

<img style="margin:auto;width:100%;height:100vh;object-fit: cover;" src='./static/images/background1.jpg'>

</div>

</body>

<script src='static/JS/index.js'></script>

<script>

<script src="./static/JS/faculty-student.js"></script>

<script>

async function setInActive(){

document.getElementById('root').style.display = 'none'

document.getElementById('home-render').style.display = 'none'

document.getElementById('main-options-home').className = ''

document.getElementById('main-options-professors').className = ''

document.getElementById('main-options-results').className = ''

document.getElementById('professors-hod-pannel').style.display = 'none'

document.getElementById('results-header-options').style.display = 'none'

}

document.getElementById('main-options-home').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderHome()

document.getElementById('home-render').style.display = 'block'

window.history.replaceState({student: 'student'},'','home')

}

document.getElementById('main-options-professors').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderProfessors()

document.getElementById('root').style.display = 'block'

document.getElementById('professors-hod-pannel').style.display = 'flex'

window.history.replaceState({student: 'student'},'','professors')

}

document.getElementById('main-options-results').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

curInfo.semester = userInfo.semester

dropdown = document.getElementById('id-semester')

dropdown.innerHTML = ''

for(i=0;i<curInfo.semester;i++)

{

opt = document.createElement('option')

opt.value= i+1

opt.innerHTML= i+1

dropdown.append(opt)

}

dropdown.value = curInfo.semester

renderResult()

document.getElementById('root').style.display = 'block'

document.getElementById('results-header-options').style.display = 'block'

window.history.replaceState({student: 'student'},'','results')

}

</script>

<script>

const userInfo = {}

async function authenticate(){

var user={

'username': sessionStorage.getItem('username'),

'token' : sessionStorage.getItem('token')

}

let res = await axios.post('/student/authenticate',user)

if(res.data.error){

console.log('Authentication error')

alert('Authentication Failed')

document.body.innerHTML = ''

window.open('/','\_self')

}

else{

console.log(res.data)

userInfo['username'] = user.username

userInfo['token'] = user.token

userInfo['name'] = res.data.name

userInfo['photo\_container'] = res.data.photo\_container

userInfo['contact'] = res.data.contact

userInfo['semester'] = res.data.semester

userInfo['hod'] = res.data.hod

userInfo['hod\_contact'] = res.data.hod\_contact

userInfo['dept\_name'] = res.data.dept\_name

document.getElementById('username').innerHTML = userInfo.name

if(userInfo.photo\_container != null)

setProfilePic()

pageRender()

}

}

async function pageRender(){

console.log(window.location.pathname)

if(window.location.pathname == '/student/professors'){

document.getElementById('main-options-professors').childNodes[0].click()

}

else if(window.location.pathname == '/student/results'){

document.getElementById('main-options-results').childNodes[0].click()

}

else{

document.getElementById('main-options-home').childNodes[0].click()

}

}

authenticate()

</script>

</html>

**5.2.3 FACULTY PAGE**

<html>

<head>

<title>

BIT-Result Portal

</title>

<script src='./static/JS/axios.min.js'></script>

<link rel="stylesheet" href='./static/CSS/navBar.css'>

<link rel="stylesheet" href='./static/CSS/faculty-home.css'>

<link rel="stylesheet" href='./static/CSS/header.css'>

<link rel="stylesheet" href='./static/CSS/table.css'>

<link rel="stylesheet" href='./static/CSS/footer.css'>

</head>

<style>

</style>

<body>

<style>

</style>

<nav class="navbar">

<div class='nav-left'>

<div class='app-logo'><img src='./static/images/logo.png' class="medium-icon"></div>

<div class='app-name'> Bangalore Institute Of Technology </div>

</div>

<div class='nav-right'>

<ul>

<li class='active' id='main-options-home'><a>Home</a></li>

<li id='main-options-course'><a>Course</a></li>

<li id='main-options-students'><a>Students</a></li>

<li id='main-options-marks'><a>Update-Marks</a></li>

<li class='nav-userinfo'>

<label class="nav-userpic" for='upload-pic' style="border: none;background:transparent;">

<img id='user-icon' src='static/icons/thumnail.png'>

</label>

<input type='file' id='upload-pic' accept="image/\*" style="display: none;">

<div class='nav-username' id='username'></div>

</li>

</ul>

</div>

</nav>

<div id='home-render' style='min-height: 100vh;'>

</div>

<div id='root' style="display: none;">

<div class='header-option' style='display: none;' id='header-options'>

<div class='add-btn'>

<div>

<div class='input-field' style='visibility:hidden' style='display: none;' id='add-student'>

<input type='text' placeholder="Enter USN">

<div onclick="addStudent()" >

<img style='width:30px;height:100%;' src="static/icons/enter.png"/>

</div>

</div>

<div class='drop-down-options' style='visibility: visible;' id='drop-down-options'>

<select name="dept\_name" id="d\_dept\_name">

</select>

<select name="couse" id="d\_course\_id">

</select>

<select name="section" id="d\_section">

</select>

<div id='student-render' style='display:none' onclick="document.getElementById('main-table').innerHTML='';renderStudentList()" >

<img style='height:35px;width:35px;' src="static/icons/enter.png"/>

</div>

</div>

</div>

<div id='marks-add-button' style='display:none' class='add-student-button' onclick="console.log('to be implemented');" >

<img style='width:30px' src="static/icons/add.png"/>

<span style='color:purple;font-weight: bold;'>Add</span>

</div>

<div id='student-add-button' style='display:none' class='add-student-button' onclick="toggleVisibility(document.getElementById('add-student'));toggleVisibility(document.getElementById('drop-down-options'))" >

<img style='width:30px' src="static/icons/add.png"/>

<span style='color:purple;font-weight: bold;'>Add</span>

</div>

</div>

</div>

<div class='footer-option' style="display: none;" id='faculty-error-log'>

<div id="error-log" class='paragraph'>

<div style='color: red;'>Error Logs :</div>

</div>

</div>

</div>

<div id='home-pic'>

<img style="margin:auto;width:100%;height:100vh;object-fit: cover;" src='./static/images/background1.jpg'>

</div>

</body>

<script src='static/JS/faculty.js'></script>

<script src="./static/JS/faculty-student.js"></script>

<script src="./static/JS/faculty-results.js"></script>

<script src='static/JS/index.js'></script>

<script>

async function setInActive(){

document.getElementById('home-render').style.display = 'none'

document.getElementById('root').style.display = 'none'

document.getElementById('main-options-home').className = ''

document.getElementById('main-options-course').className = ''

document.getElementById('main-options-students').className = ''

document.getElementById('main-options-marks').className = ''

document.getElementById('faculty-error-log').style.display = 'none'

document.getElementById('home-pic').style.display = 'none'

document.getElementById('header-options').style.display = 'none'

document.getElementById('add-student').style.display = 'none'

document.getElementById('marks-add-button').style.display = 'none'

document.getElementById('student-add-button').style.display = 'none'

document.getElementById('marks-render').style.display = 'none'

document.getElementById('student-render').style.display = 'none'}

document.getElementById('main-options-home').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderHome()

document.getElementById('home-render').style.display = 'block'

document.getElementById('home-pic').style.display = 'block'

document.getElementById('error-log').innerHTML=''

window.history.replaceState({student: 'faculty'},'','home')

}

document.getElementById('main-options-course').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderCourseList()

document.getElementById('root').style.display = 'block'

document.getElementById('faculty-error-log').style.display = 'block'

document.getElementById('error-log').innerHTML=''

window.history.replaceState({student: 'faculty'},'','course')

}

document.getElementById('main-options-students').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderStudentList()

document.getElementById('root').style.display = 'block'

document.getElementById('faculty-error-log').style.display = 'block'

document.getElementById('header-options').style.display = 'block'

document.getElementById('student-render').style.display = 'block'

document.getElementById('student-add-button').style.display = 'flex'

document.getElementById('add-student').style.display = 'flex'

document.getElementById('error-log').innerHTML=''

window.history.replaceState({faculty:'faculty'},'','students')

}

document.getElementById('main-options-marks').childNodes[0].onclick = async function(e){

e.preventDefault()

setInActive()

e.target.parentNode.className = 'active'

renderResultList()

document.getElementById('root').style.display = 'block'

document.getElementById('faculty-error-log').style.display = 'block'

document.getElementById('drop-down-options').style.visibility = 'visible'

document.getElementById('add-student').style.visibility = 'hidden'

document.getElementById('header-options').style.display = 'block'

document.getElementById('marks-render').style.display = 'block'

document.getElementById('marks-add-button').style.display = 'flex'

window.history.replaceState({faculty:'faculty'},'','update-marks')

}

</script>

<script>

const userInfo = {}

async function pageRender(){

console.log(window.location.pathname)

if(window.location.pathname == '/faculty/update-marks'){

document.getElementById('main-options-marks').childNodes[0].click()

}

else if(window.location.pathname == '/faculty/students'){

document.getElementById('main-options-students').childNodes[0].click()

}

else if(window.location.pathname == '/faculty/course'){

document.getElementById('main-options-course').childNodes[0].click()

}

else{

document.getElementById('main-options-home').childNodes[0].click()

}

}

async function authenticate(){

var user={

'username': sessionStorage.getItem('username'),

'token' : sessionStorage.getItem('token')

}

let res = await axios.post('/faculty/authenticate',user)

if(res.data.error){

alert('Authentication Failed')

document.body.innerHTML = ''

window.open('/','\_self')

}

else{

userInfo['username'] = user.username

userInfo['token'] = user.token

userInfo['name'] = res.data.name

userInfo['photo\_container'] = res.data.photo\_container

userInfo['contact'] = res.data.contact

document.getElementById('username').innerHTML = userInfo.name

if(userInfo.photo\_container != null)

setProfilePic()

pageRender()

}

}

authenticate()

</script>

</html>

**5.2.4 ADMIN PAGE**

<html>

<head>

<title>

BIT-Result Portal

</title>

<script src='./static/JS/axios.min.js'></script>

<link rel="stylesheet" href='./static/CSS/navBar.css'>

<link rel="stylesheet" href='./static/CSS/login.css'>

<link rel="stylesheet" href='./static/CSS/table.css'>

<link rel="stylesheet" href='./static/CSS/faculty-home.css'>

</head>

<style>

#root{

min-height: 100vh;

width:100%;

margin-top: 16px;

}

.table-list {

width: fit-content;

}

</style>

<body>

</body>

<script src='static/JS/index.js'></script>

<script>

const adminInfo = {}

async function authenticate(){

var user={

'username': sessionStorage.getItem('username'),

'token' : sessionStorage.getItem('token')

}

let res = await axios.post('/admin/authenticate',user)

if(res.data.error){

console.log('Authentication error')

alert('Authentication Failed')

document.body.innerHTML = ''

window.open('/','\_self')

}

else{

console.log(res.data)

adminInfo['name'] = res.data.name

adminInfo['photo\_container'] = res.data.photo\_container

adminInfo['contact'] = res.data.contact

document.getElementById('username').innerHTML = adminInfo.name

if(adminInfo.photo\_container != null)

setProfilePic()

getDCLInfo()

//renderAdminPage()

}

}

authenticate()

</script>

</html>

**5.3 BACK END CODE**

**5.2.3 ORACLE RELATED FUNCTIONS**

async function connect(){

try{

db = await oracledb.getConnection({user : "dbms",password : "2310"});

isConnected=true;

}

catch(err){

}

}

//<------------------ User----------------------------->

async function authentication(user){

user.username = user.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

var result = await db.execute(`select password,type,login\_fail from user\_login where user\_id='${user.username}'`)

if(result.rows[0]){

let row = result.rows[0]

if(row[2] > manager.maxLoginFail ){

return({error: true,value:'AccountLocked'})

}

else if(row[0]==user.password){

res = {token:user.token,type:row[1],error:false}

db.execute(`update user\_login set token = '${user.token}',token\_fail=0 where user\_id='${user.username}'`)

}

else{

db.execute(`update user\_login set login\_fail = login\_fail + 1 where user\_id='${user.username}'`)

res = {token:'',type:'',error:true,value:'LoginFail'}

}

db.execute('commit')

}

else{

res = {error:true,value:'LoginFail'}

}

return res

}

async function tokenAuthentication(user){

user.username = user.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

var result = await db.execute(`select token,token\_fail,type from user\_login where user\_id='${user.username}'`)

if(result.rows.length < 1)

return {token:'',type:'',error:true,value:'NoRecord'}

let row = result.rows[0]

if(row[1] > manager.maxTokenFail ){

return({error: true,value:'AccountLocked'})

}

else if(row[0]==user.token){

res = {error:false,type:row[2]}

}

else{

res = {token:'',type:'',error:true,value:'TokenFail'}

db.execute(`update user\_login set token\_fail = token\_fail + 1 where user\_id='${user.username}'`)

db.execute('commit')

}

return res

}

async function getInfo(user){

user.username = user.username.toUpperCase()

if(!isConnected){

await connect();

}

var res = {}

if(manager.AccountType[user.type] == 'Student'){

let result = await db.execute(`select USN, s.Name as Name, Dept\_Name as Department, f.Name as HOD,

f.contact as HodContact, Batch, Semester, Section, s.Contact as Phone,s.photo\_container,s.dept\_id

from student s, department d, faculty f

where usn='${user.username}'

and s.dept\_id = d.dept\_id

and d.hod\_id = f.faculty\_id`)

res.name = result.rows[0][1]

res.dept\_name = result.rows[0][2]

res.hod = result.rows[0][3]

res.hod\_contact = result.rows[0][4]

res.batch = result.rows[0][5]

res.semester = result.rows[0][6]

res.section = result.rows[0][7]

res.contact = result.rows[0][8]

res.photo\_container = result.rows[0][9]

res.dept\_id= result.rows[0][10]

}

else{

result = await db.execute(`select name,photo\_container,contact from faculty where faculty\_id='${user.username}'`)

res['name'] = result.rows[0][0]

res['photo\_container'] = result.rows[0][1]

res['contact'] = result.rows[0][2]

}

return res

}

async function createPhotoContainer(username,type){

if(!isConnected){

await connect();

}

var res = manager.randomString(10)

try{

if(type==manager.AccountTypeInv['student']){

await db.execute(`update student set photo\_container='${res}' where usn='${username.toUpperCase()}'`)

await fs.mkdir(manager.path.private+'\\'+res,false,async (err) => {

if(err)

return true

});

}

else{

await db.execute(`update faculty set photo\_container='${res}' where faculty\_id='${username.toUpperCase()}'`)

await fs.mkdir(manager.path.public+'\\'+res,false,async (err) => {

if(err)

return true

});

}

await db.execute('commit')

}

catch(error){

//console.log(error)

//res = createPhotoContainer(username,type)

}

return res

}

//<------------------Admin---------------------------->

async function createAccount(info){

if(!isConnected){

await connect();

}

try{

if(info.type == 'student'){

photo = await createPhotoContainer(info.id.toUpperCase(),manager.AccountTypeInv['student'])

await db.execute(`insert into user\_login(user\_id,password,type,token)

values('${info.id.toUpperCase()}','${info.password}',2,'')`)

await db.execute(`insert into student(usn,name,dept\_id,batch,semester,section,contact,photo\_container)

values('${info.id.toUpperCase()}','${info.name}','${info.department.toUpperCase()}',

${info.batch},${info.semester},'${info.section}',${info.contact},'${photo}')`)

}

else if(info.type == 'faculty'){

photo = await createPhotoContainer(info.id.toUpperCase(),manager.AccountTypeInv['faculty'])

await db.execute(`insert into user\_login(user\_id,password,type,token)

values('${info.id.toUpperCase()}','${info.password}',1,'')`)

await db.execute(`insert into faculty(faculty\_id,name,contact,photo\_container)

values('${info.id.toUpperCase()}','${info.name}',${info.contact},'${photo}')`)

}

else if(info.type == 'admin'){

photo = await createPhotoContainer(info.id.toUpperCase(),manager.AccountTypeInv['faculty'])

await db.execute(`insert into user\_login(user\_id,password,type,token)

values('${info.id.toUpperCase()}','${info.password}',0,'')`)

await db.execute(`insert into faculty(faculty\_id,name,contact,photo\_container)

values('${info.id.toUpperCase()}','${info.name}',${info.contact},'${photo}')`)

}

else if(info.type == 'department'){

await db.execute(`insert into department(dept\_id,hod\_id,dept\_name)

values('${info.id.toUpperCase()}','${info.hod.toUpperCase()}','${info.name}')`)

}

else if(info.type == 'course'){

await db.execute(`insert into course(course\_id,title,semester,schema,credits,iselective)

values('${info.id.toUpperCase()}','${info.title}',${info.semester},${info.schema},${info.credits},${info.iselective})`)

}

else{

return {error:true,value: info.type + ' : Not valid'}

}

res = {error:false}

await db.execute('commit')

}

catch (error){

res = {error:true,value: info.type+' : Inertion was unsuccessful'}//,detail: error}

}

return res

}

async function getTableInfo(info){

if(!isConnected){

await connect();

}

let result

try{

if(info.type == 'student'){

result = await db.execute(`select usn,name,password,contact,batch,dept\_id,semester,section

from student s,user\_login a

where s.dept\_id='${info.department.toUpperCase()}' and s.semester=${info.semester} and s.section='${info.section.toUpperCase()}' and s.usn=a.user\_id

order by usn`)

}

else if(info.type == 'faculty'){

result = await db.execute(`select faculty\_id,name,password,contact

from faculty f,user\_login a

where type=1 and f.faculty\_id=a.user\_id

order by faculty\_id`)

}

else if(info.type == 'admin'){

result = await db.execute(`select faculty\_id,name,password,contact

from faculty f,user\_login a

where type=0 and f.faculty\_id=a.user\_id

order by faculty\_id`)

}

else if(info.type == 'department'){

result = await db.execute(`select dept\_id,dept\_name,hod\_id

from department

order by dept\_id`)

}

else if(info.type == 'course'){

result = await db.execute(`select course\_id,title,semester,credits,iselective,schema

from course where schema=${info.schema}

order by course\_id`)

}

else{

return {error:true,value: info.type + ' : Not valid'}

}

res = result

res['error'] =false

}

catch (error){

res = {error:true,value: info.type+' : Search was unsuccessful'}//,detail: error}

}

return res

}

async function updateTableInfo(info){

if(!isConnected){

await connect();

}

let res = {}

try{

if(info.type == 'student'){

await db.execute(`update user\_login set password='${info.password}',type=${manager.AccountTypeInv[info.type]} where user\_id='${info.usn}'`)

await db.execute(`update student set name='${info.name}', batch=${info.batch}, dept\_id='${info.dept\_id.toUpperCase()}' ,

semester=${info.semester}, section='${info.section.toUpperCase()}',contact = ${info.contact} where usn='${info.usn}'`)

}

else if(info.type == 'faculty'){

await db.execute(`update user\_login set password='${info.password}',type=${manager.AccountTypeInv[info.type]} where user\_id='${info.faculty\_id}'`)

await db.execute(`update faculty set name='${info.name}',contact=${info.contact} where faculty\_id='${info.faculty\_id}' `)

}

else if(info.type == 'admin'){

await db.execute(`update user\_login set password='${info.password}',type=${manager.AccountTypeInv[info.type]} where user\_id='${info.faculty\_id}'`)

await db.execute(`update faculty set name='${info.name}',contact=${info.contact} where faculty\_id='${info.faculty\_id}' `)

}

else if(info.type == 'department'){

await db.execute(`update department set dept\_name='${info.dept\_name}',hod\_id='${info.hod\_id.toUpperCase()}'

where dept\_id='${info.dept\_id}'`)

}

else if(info.type == 'course'){

await db.execute(`update course set title='${info.title}',semester=${info.semester}, credits=${info.credits}, iselective = ${info.iselective},schema=${info.schema}

where course\_id='${info.course\_id}'`)

}

else{

return {error:true,value: info.type + ' : Not valid'}

}

await db.execute('commit')

res['error'] =false

}

catch (error){

res = {error:true,value: info.type+' : update was unsuccessful'}//,detail: error}

}

return res

}

async function removeTableInfo(info){

if(!isConnected){

await connect();

}

let res = {}

try{

if(info.type == 'student'){

await db.execute(`delete from user\_login where user\_id='${info.id}'`)

}

else if(info.type == 'faculty'){

await db.execute(`delete from user\_login where user\_id='${info.id}'`)

}

else if(info.type == 'admin'){

await db.execute(`delete from user\_login where user\_id='${info.id}'`)

}

else if(info.type == 'department'){

await db.execute(`delete from department where dept\_id='${info.id}'`)

}

else if(info.type == 'course'){

await db.execute(`delete from course where user\_id='${info.id}'`)

}

else{

return {error:true,value: info.type + ' : Not valid'}

}

await db.execute('commit')

res['error'] =false

}

catch (error){

res = {error:true,value: info.type+' : Removal was unsuccessful'}//,detail: error}

}

return res}

async function getDepartment(){

if(!isConnected){

await connect();

}

let res = {}

try{

res.department = await db.execute(`select dept\_id,dept\_name,hod\_id from department order by dept\_id`)

res.course = await db.execute(`select course\_id,title from course order by course\_id`)

res.list = await db.execute(`select dept\_id,course\_id,title from course\_list inner join course using(course\_id) order by dept\_id,course\_id`)

res.faculty = await db.execute(`select faculty\_id,name from faculty order by faculty\_id`)

res.error = false

}

catch{

res.error = true

}

return res

}

async function getDeptCourse(info){

if(!isConnected){

await connect();

}

let res = {}

try{

res = await db.execute(`select list\_id,course\_id,title,credits

from (select course\_id,list\_id from course\_list where dept\_id = '${info.dept\_id.toUpperCase()}')

inner join (select \* from course where semester=${info.semester}) using(course\_id)

order by course\_id`)

res.error = false

}

catch{

res.error = true

res.value = 'No Course listed in department'

}

return res

}

async function removeDeptCourse(list\_id){

if(!isConnected){

await connect();

}

let res = {}

try{

res = await db.execute(`delete from course\_list where list\_id = '${list\_id}' `)

res.error = false

await db.execute('commit')

}

catch{

res.error = true

res.value = 'Removal of course-list was unsuccessful'

}

return res

}

async function addDeptCourse(info){

if(!isConnected){

await connect();

}

let res = {}

info.list\_id = manager.randomString(10)

try{

res = await db.execute(`insert into course\_list(list\_id,course\_id,dept\_id) values('${info.list\_id}','${info.course\_id.toUpperCase()}','${info.dept\_id}') `)

res.error = false

await db.execute('commit')

}

catch{

res.error = true

res.value = 'Insertion of course-list was unsuccessful'

}

return res

}

async function getFacultyCourse(info){

if(!isConnected){

await connect();

}

let result = {}

try{

result = await db.execute(`select list\_id,title,semester,section,faculty\_id,name

from (select list\_id,course\_id from course\_list where course\_id='${info.course\_id.toUpperCase()}' and dept\_id='${info.dept\_id.toUpperCase()}')

inner join takes using(list\_id) inner join faculty using(faculty\_id) inner join course using(course\_id) order by course\_id`)

result.error = false

}

catch{

result.error = true

result.value = 'Selected Course is not taught to any section'

}

return result

}

async function removeFacultyCourse(info){

if(!isConnected){

await connect();

}

let result = {}

try{

await db.execute(`delete from takes where list\_id='${info.list\_id}' and section='${info.section}'`)

result.error = false

await db.execute('commit')

}

catch{

result.error = true

result.value = 'Removal was unsucessful'

}

return result

}

async function addFacultyCourse(info){

if(!isConnected){

await connect();

}

let result = {}

try{

res = await db.execute(`select list\_id from course\_list where course\_id='${info.course\_id}' and dept\_id='${info.dept\_id}'`)

if(res.rows[0])

info.list\_id = res.rows[0][0]

else

return {error:true,value:'Department-Course combination is not listed in course-list'}

await db.execute(`insert into takes(list\_id,faculty\_id,section) values('${info.list\_id}','${info.faculty\_id}','${info.section.toUpperCase()}')`)

result.error = false

await db.execute('commit')

}

catch{

result.error = true

result.value = 'Insertion was unsucessful'

}

return result

}

//<------------------Student--------------------------->

async function getResults(info){

info.username = info.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

let result = await db.execute(`select \*

from ((select course\_id,semester as course\_Semester,isElective,Title

from course

where semester = ${info.semester})

inner join

(select \* from results

where usn = '${info.username}')

using (course\_id)

)

order by course\_id`)

if(result.rows[0]){

res = result

res.error = false

}

else{

res = {error:true,value:'ResultNotUpdated'}

}

return res

}

async function getProfessorContact(info){

info.username = info.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

let user = {}

let result = await db.execute(`select dept\_id,semester,section from student where usn='${info.username}'`)

user.dept\_id = result.rows[0][0]

user.semester = result.rows[0][1]

user.section = result.rows[0][2]

result = await db.execute(`(select name,contact,photo\_container,course\_id,title

from ((((select \* from course\_list where dept\_id = '${user.dept\_id}')

inner join (select course\_id,title from course where semester = ${user.semester}) using(course\_id))

inner join (select \* from takes where section = '${user.section}') using(list\_id))

inner join faculty using(faculty\_id)))

order by course\_id`)

if(result.rows[0]){

res = result

res.error = false

}

else{

res.error = true

res.value = 'NoProfessor'

}

return res

}

//<------------------Faculty-------------------------->

// info = (faculty\_id:'F100')

//return : faculty{...}

async function getCourseList(info){

info.username = info.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

let result = await db.execute(`select course\_id,title,dept\_name,semester,section

from ((((select \* from takes

where faculty\_id='${info.username}')

inner join course\_list

using (list\_id))

inner join department

using (dept\_id))

inner join course

using (course\_id))

order by course\_id,section`)

if(result.rows[0]){

res = result

res.error = false

}

else{

res = {error:true,value:'NoCourseTaught'}

}

return res

}

//info = faculty{...}

//return : {list\_id,dept\_id}

async function checkCourseList(info){

info.username = info.username.toUpperCase()

if(!isConnected){

await connect();

}

let res = {}

let result = await db.execute(`select \* from

(select list\_id,dept\_id,semester,title

from ((((select \* from course\_list where course\_id='${info.course\_id}')

inner join

(select \* from department where dept\_name='${info.dept\_name}')

using (dept\_id))

inner join

(select list\_id from takes

where faculty\_id = '${info.username}' and section='${info.section}')

using (list\_id))

inner join

(select course\_id,semester,title from course)

using (course\_id))

)

`)

if(result.rows[0]){

res.error = false

res.list\_id = result.rows[0][0]

res.dept\_id = result.rows[0][1]

res.semester = result.rows[0][2]

res.title = result.rows[0][3]

}

else{

res = {error:true}

}

////console.log(res)

return res

}

//info = faculty{...}+dept\_id+sem

async function getStudentBySem(info){

if(!isConnected){

await connect();

}

let res = []

let result = await db.execute(`select usn from student where semester = ${info.semester}

and section = '${info.section}' and dept\_id='${info.dept\_id}'

`)

////console.log(result)

if(result.rows[0]){

for(let i=0 ; i<result.rows.length;i++){

res.push(result.rows[i][0])

}

}

////console.log(236,res)

return res

}

//info = faculty{...}+dept\_id

async function getStudentResult(info){

if(!isConnected){

await connect();

}

let res = {}

let result = await db.execute(`select \*

from (select usn,name from student where semester = ${info.semester} and section = '${info.section}' and dept\_id='${info.dept\_id}')

inner join

(select \* from results where course\_id='${info.course\_id}')

using (usn)

order by usn

`)

if(result.rows[0]){

res=result

res.error = false

}

else{

res = {error:true,value:'NoCourseStudent'}

}

return res

}

//info = faculty{...}+dept\_id+{course\_id:'18CS54', newMarks=[usn,name,.....] }

async function updateResult(info){

if(!isConnected){ await connect();

}

try{

info.usn

}

catch{

return {error:false}

}

let res = {value:[]}

let student = await getStudentBySem(info)

if(student == []){

return ({error:true,value:'NoCourseStudent'})

}

let metaData=['TEST1\_MARKS','TEST1\_ASSIGNMENT','TEST2\_MARKS','TEST2\_ASSIGNMENT','TEST3\_MARKS','TEST3\_ASSIGNMENT','IA\_MARKS','EXTERNAL\_MARKS','TOTAL']

for(i = 0 ; i<info.newMarks.length;i++){

query = ''

s = info.newMarks[i]

if(student.includes(s[0])){

for(c=0 ; c<metaData.length ; c++)

query += metaData[c]+'='+ s[c+3]+', '

query = query.replace(/, $/,'')

try {

result = await db.execute(`update results set ${query} where usn='${s[0]}' and course\_id = '${info.course\_id}'`)

}

catch(error) {

res.value.push({usn:s[0],value:'Student Marks was not updated',detail:error})

}

}

else{

res.value.push({usn:s[0],value:'Student not listed in your class'})

}

}

db.execute('commit')

return res

}

//info = faculty{...}+dept\_id

async function getStudentFromCourse(info){

if(!isConnected){

await connect();

}

stud = await getStudentBySem(info)

s = ''

for(i=0;i<stud.length;i++){

s += `'${stud[i]}', `

}

s = s.replace(/,\s\*$/,'')

let res = {value:[]}

if(s== '')

return {error:true}

result = await db.execute(`select usn,name,semester,batch,contact,photo\_container

from (select \* from results where course\_id='${info.course\_id}' and usn in (${s}) )

inner join student using(usn) order by usn

`)

if(result.rows[0]){

res = result

res.error = false

res.studentBySem = stud

}

else{

res.error = true

}

return res

}

//info = {course\_id:'18CS54', usn : ['s1',s2'...]}

async function addStudentToCourse(info){

if(!isConnected){

await connect();

}

try{

info.usn

}

catch{

return {error:false}

}

let res = {value:[]}

for(let i=0 ; i<info.usn.length; i++){

try {

await db.execute(`insert into results(usn,course\_id) values('${info.usn[i]}','${info.course\_id}')`)

}

catch(error) {

res.value.push({usn:info.usn[i],value:'Student was not added',detail:error})

}

}

db.execute('commit')

return res}

//info = faculty{...}+dept\_id+{course\_id:'18CS54', usn : ['s1',s2'...]}

async function removeStudentFromCourse(info){

if(!isConnected){

await connect();

}

try{

info.usn

}

catch{

return {error:false}

}

let res = {value:[]}

let student = await getStudentBySem(info)

if(student == []){

return ({error:true,value:'NoCourseStudent'})

}

if(student.includes(info.usn)){

try {

await db.execute(`delete from results where usn='${info.usn}' and course\_id='${info.course\_id}'`)}

catch(error) {

res.value.push({usn:info.usn[i],value:'Removal of Student was unsucessful',detail:error})

}

}

else{

res.value.push({usn:info.usn[i],value:'Student not listed in your class'})

}

db.execute('commit')

return res

}

**5.4 SQL QUERIES**

<------Query: To retrive info of a student whose usn='1BI19CS058'-------------->

select USN, s.Name as Name, Dept\_Name as Department, f.Name as HOD,

f.contact as HodContact, Batch, Semester, Section, s.Contact as Phone

from student s, department d, faculty f

where usn='1BI19CS058' and s.dept\_id = d.dept\_id d.hod\_id = f.faculty\_id;

<-----Query: To retrive result of a particular semester say 7 of a student whose usn='1BI19CS161' ----------->

select \*

from ((select course\_id,semester as course\_Semester,isElective,Title from course

where semester

inner join

(select \* from results where usn = '1BI19CS161') using (course\_id)

);

<-----Query: To retrive professors contact info----------->

select name,contact,profile\_container

from faculty f

where f.faculty\_id in ((select faculty\_id

from takes

where section = 'A'

and list\_id in (select list\_id

from course\_list cl, course c

where cl.dept\_id='D100'

and cl.course\_id = c.course\_id

and c.semester = '6')

)

union

(select hod\_id from department

where dept\_id = 'D100'));

<------Query: To retrive all courses handled by a professor----------->

select course\_id,title,dept\_Name,Semester,section

from ((((select \* from takes

where faculty\_id = 'F101')

inner join course\_list

using (list\_id))

inner join department

using (dept\_id))

inner join course

using (course\_id));

<--------Query: To check is a faculty with faculty\_id='F102' teaches course course\_id='18CS54' to section='A' students belonging to dept\_name='CSE'---------->

select \* from

(select list\_id,dept\_id

from (select \* from course\_list where course\_id='18CS54')

inner join

(select \* from department where dept\_name='CSE') using (dept\_id)

)

inner join

(select list\_id from takes

where faculty\_id = 'F102' and section='A')

using (list\_id);

<-------Query: To retrive list of students taking a course='18MAT11' belonging to dept\_id='D100', semester=5 and section='A'-------------------->

select \*

from (select usn,name from student

where semester = 5 and section = 'A' and dept\_id='D100')

inner join

(select \* from results where course\_id='18MAT11')

using (usn)

;

<-------Query: To retrive usn list of student belonging to dept\_id = 'D100',semester=5,section='A'----->

select usn from student

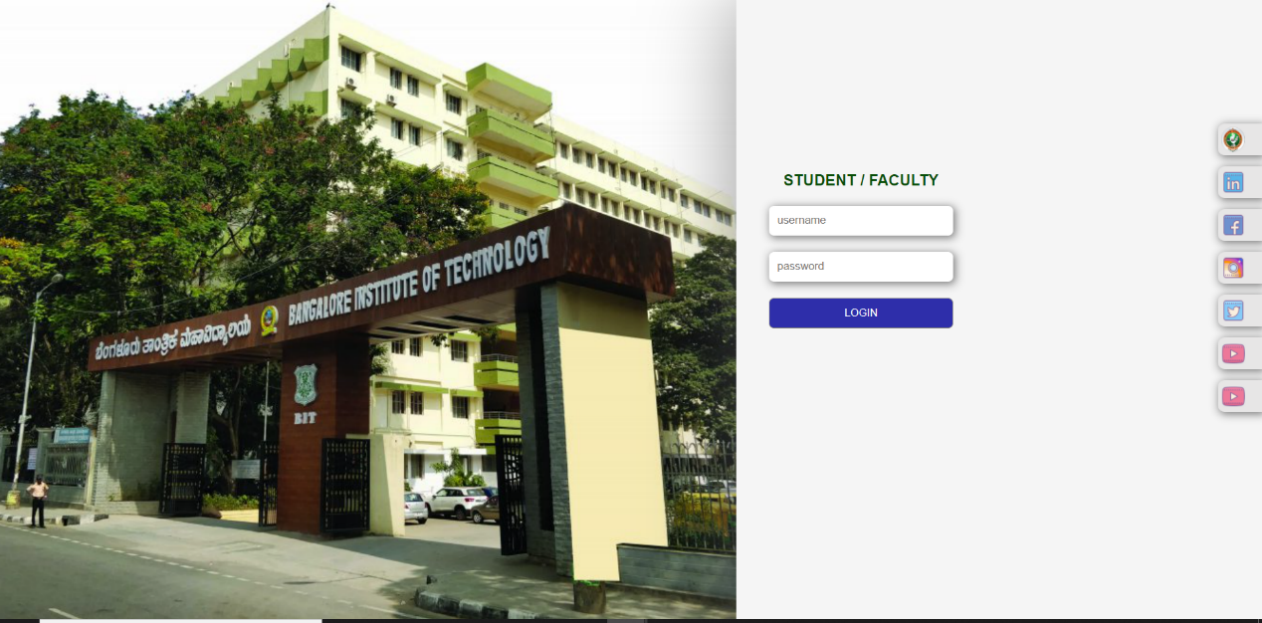
where dept\_id='D100' and semester=5 and section='A';

**CHAPTER 6**

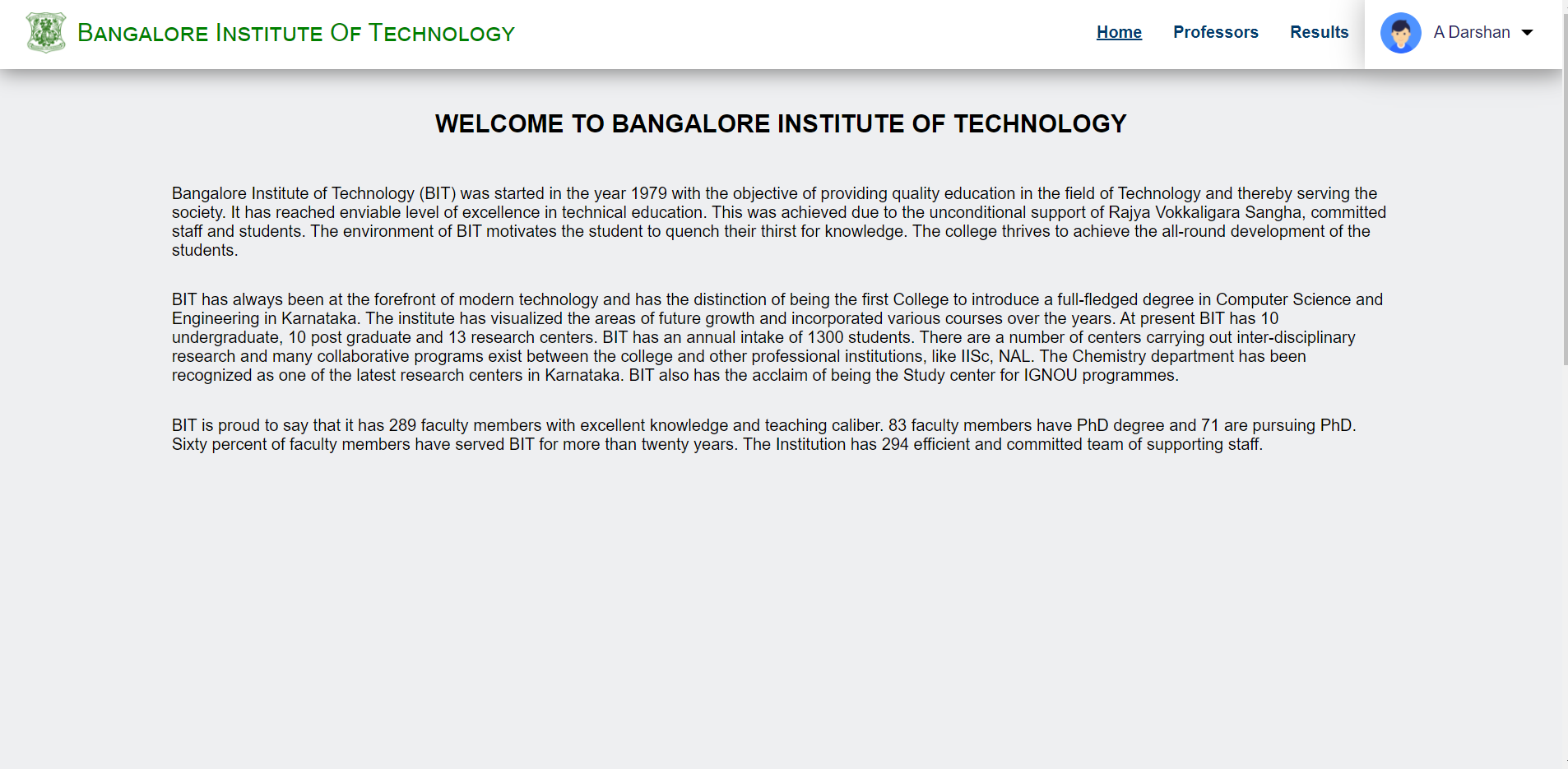
**SNAPSHOTS**

**SNAPSHOTS**

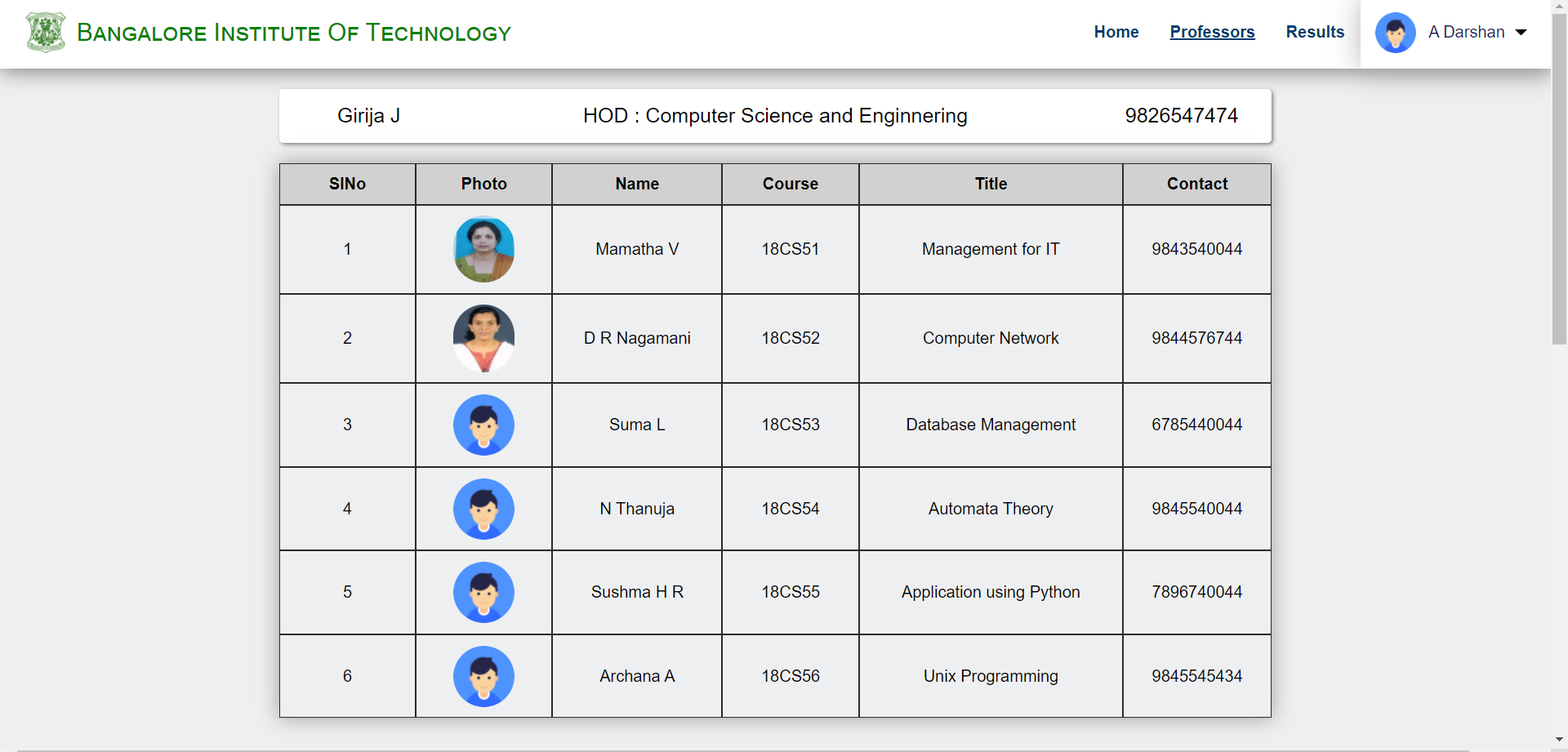
**6.1 LOGIN PAGE**

****

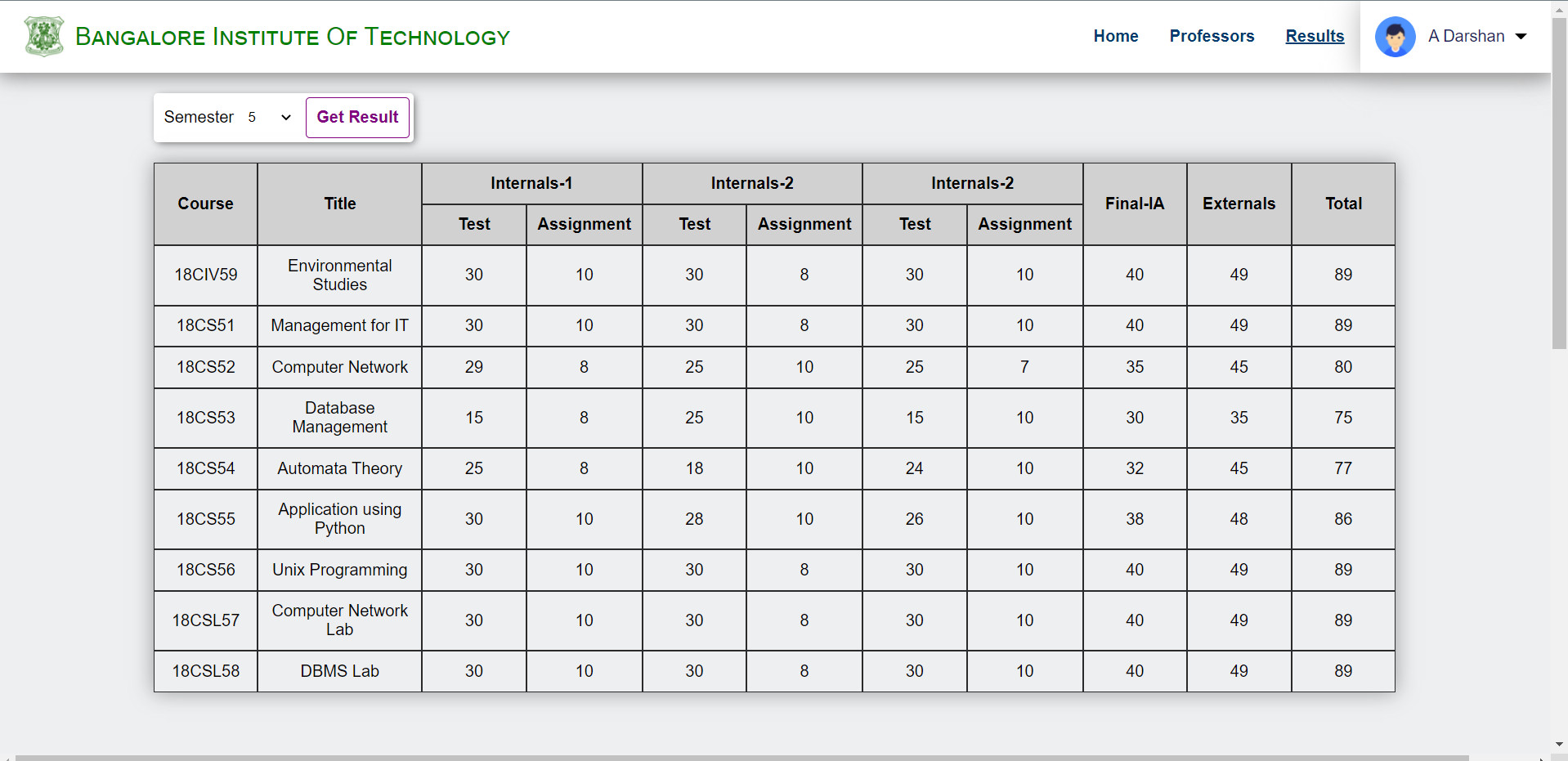
**6.2 HOME PAGE**

****

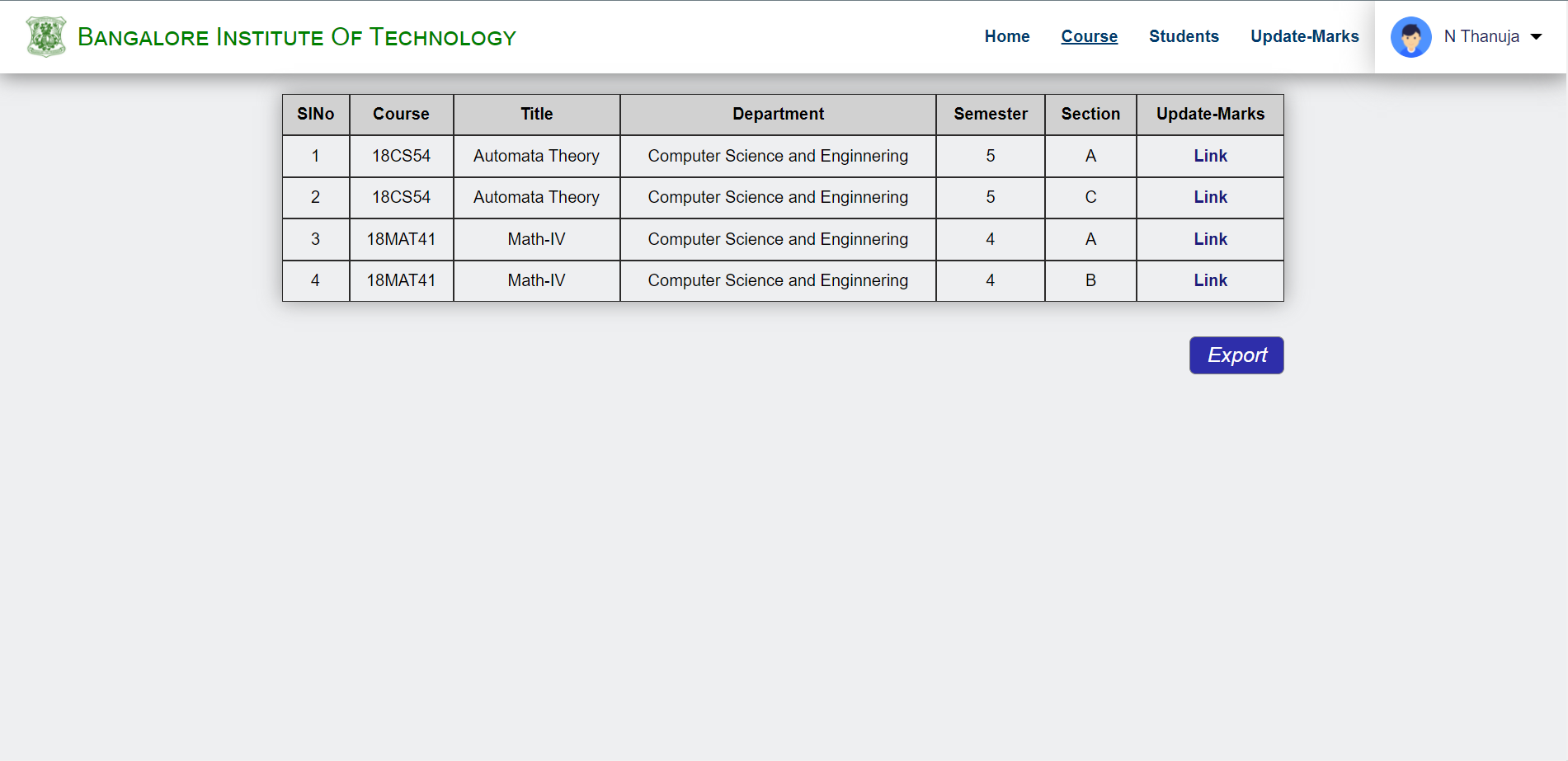
**6.3 STUDENT - PROFESSOR PAGE**

****

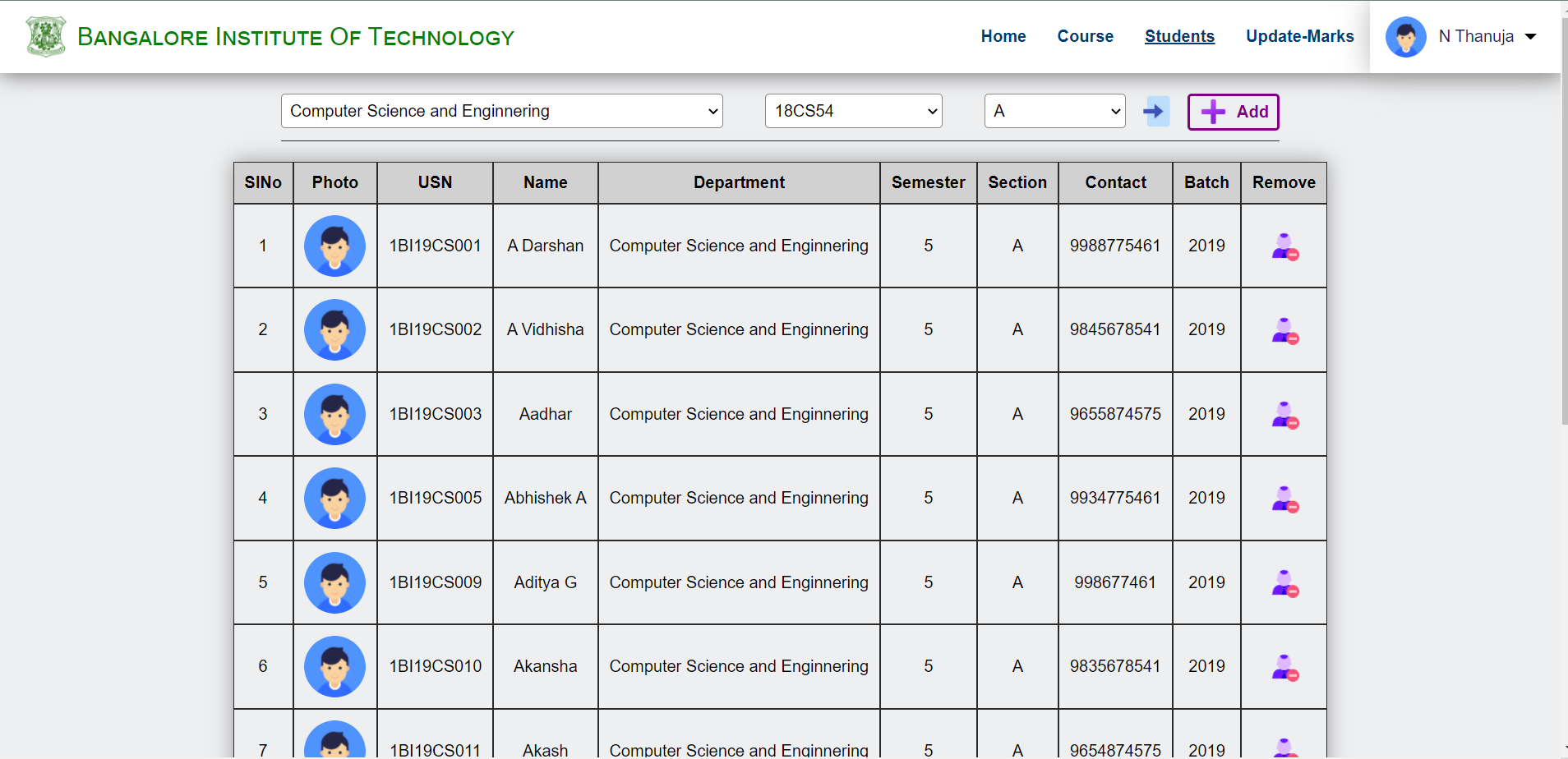
**6.4 STUDENT - RESULT PAGE**

****

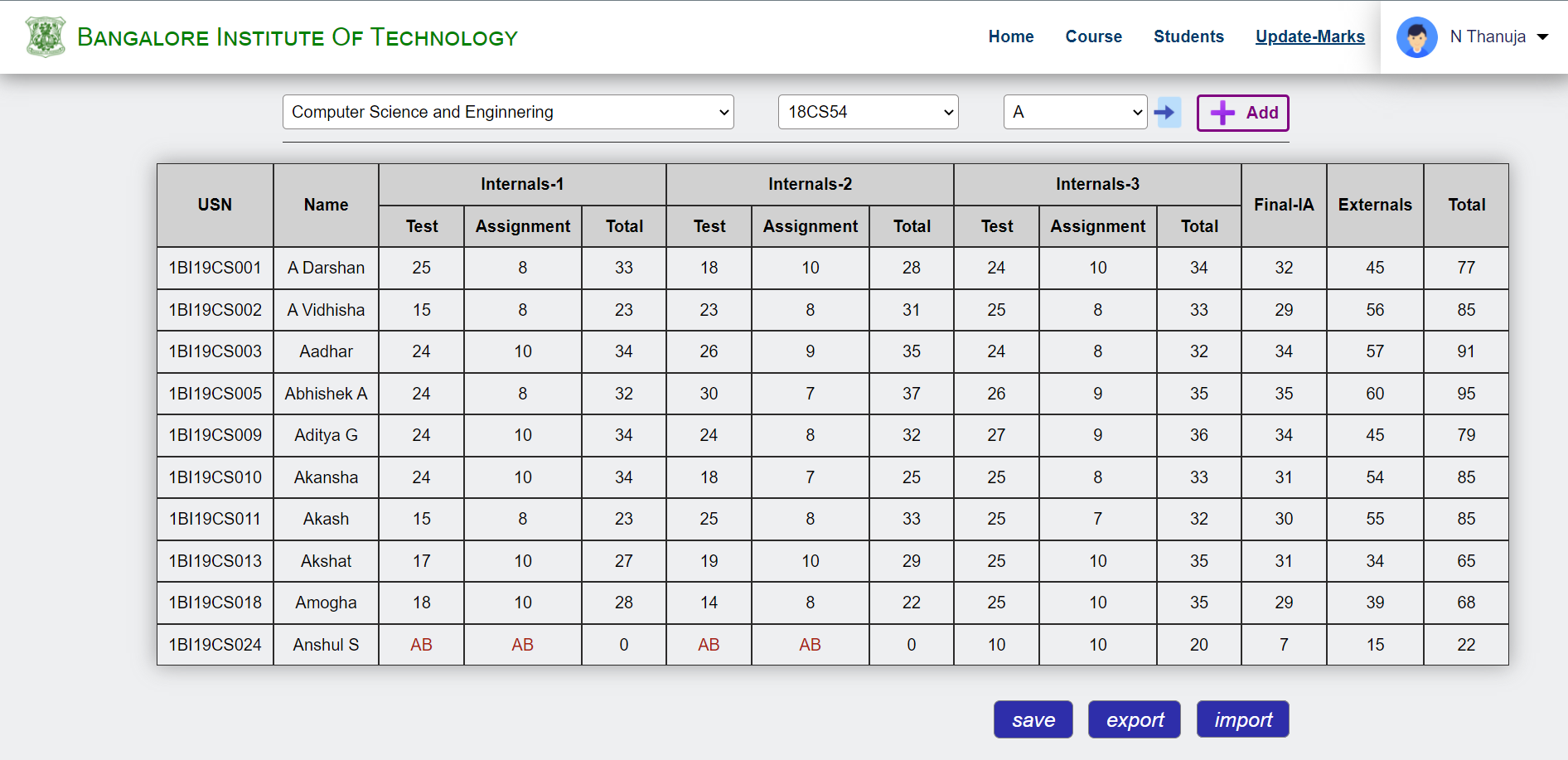
**6.5 FACULTY - COURSE PAGE**

****

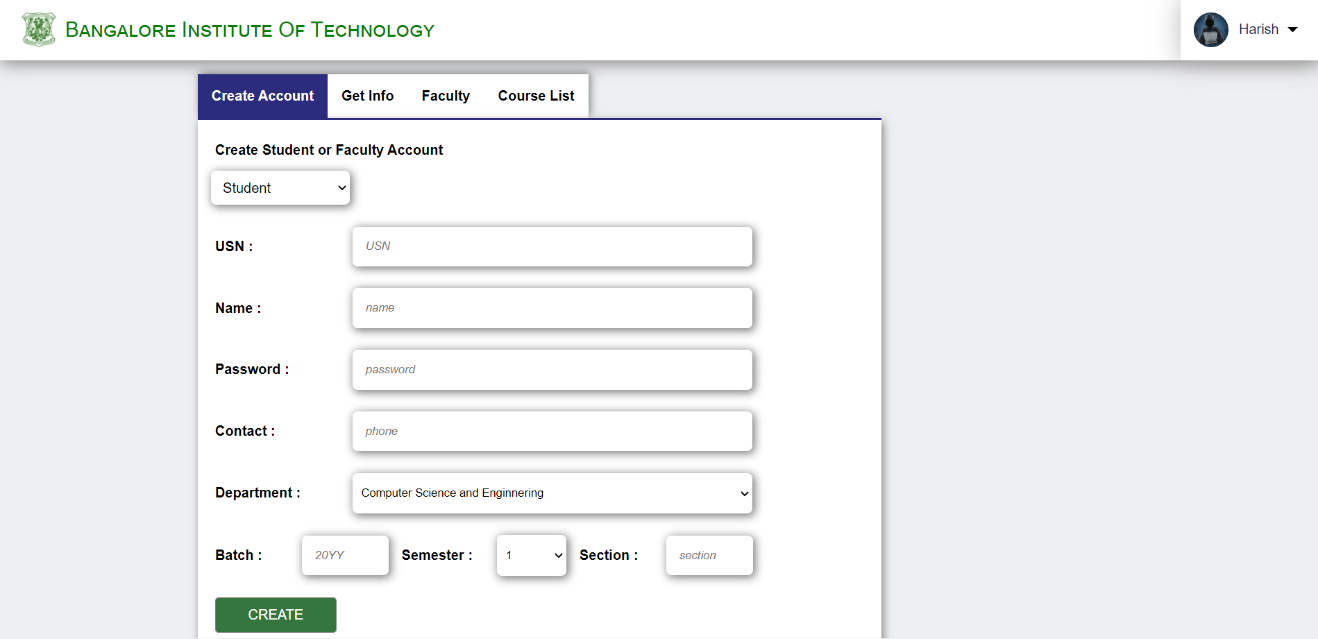
**6.6 FACULTY - STUDENT LIST PAGE**

****

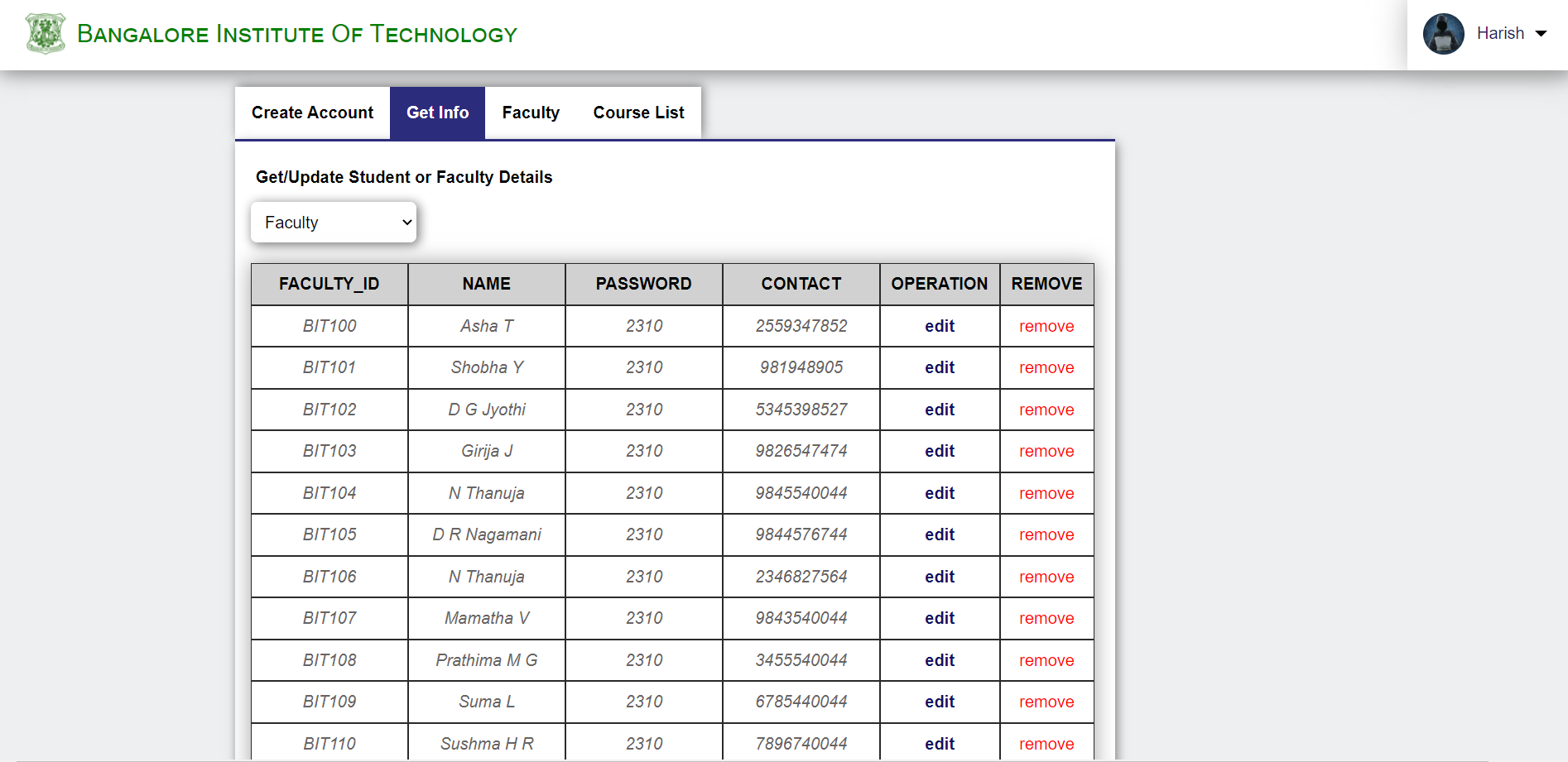
**6.7 FACULTY - UPDATE MARKS PAGE**

****

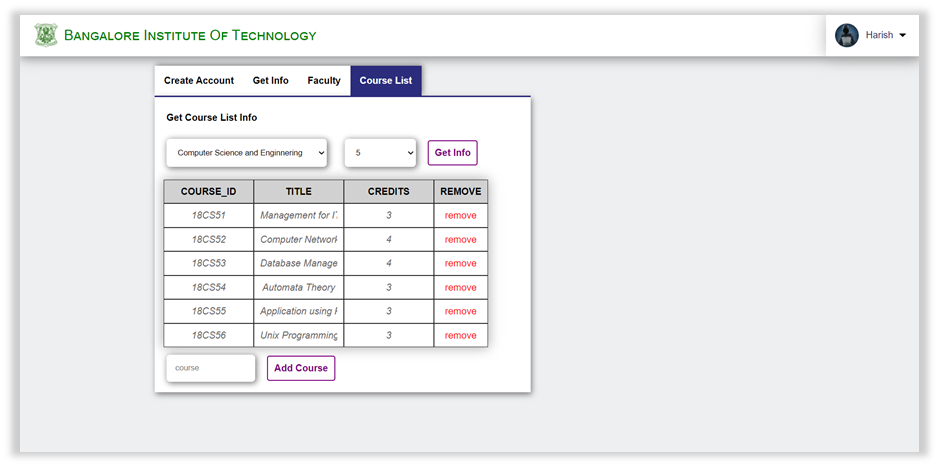
**6.8 ADMIN – CREATE ACCOUNT PAGE**

****

**6.9 ADMIN – GET INFO PAGE**

****

**6.10 ADMIN – COURSE LIST PAGE**

****

**CHAPTER 7**

**APPLICATIONS**

**APPLICATIONS**

* Collage result portal.
* Manage students result in school.
* Export Results as CSV file.
* Can be used to derive Student Management System

**CHAPTER 8**

**CONCLUSION**

**CONCLUSION**

* To conclude the description about the project: The project, developed using NODE JS and ORACLE is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement. The expanded functionality of today’s software requires an appropriate approach towards software development.
* Student result management system is an online website and can be used at any place, any time and by any student or faculty. This application will avoid the calculation and simplify the process of visualizing results by students as well as faculty.