Bangalore Institute of Technology

K.R. ROAD, BENGALURU-560004.

Department of Computer Science and Engineering

DBMS Mini Project Synopsis

Student Result Management System

USN	NAME	BATCH	PHONE NO	E-MAIL
1BI19CS038	Bharath Gowda B	A2	9900783185	bharathvishnubharath@gmail.com
1BI19CS058	Harish V	A4	9880948479	harish00411@gmail.com

Problem Domain: Education

Lab In-Charges: Prof. Prathima M.G, Prof. Kavitha K

29-12-2021

2021-2022

For office use only:

Accepted : □

To be modified : □

Rejected : □

Signature of the Lab In-Charges

PROBLEM STATEMENT:

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

This system is built for students to check results and gather information about courses offered by department. The faculty can view the overall performance of the students in the semester examinations subject-wise. The student accessing their results through college site is more convenient and the faculty can easily analyse pass and fail of a particular subject. This system is built to check marks for different semester.

FRONT-END TOOL:

- HTML
- CSS
- JAVASCRIPT
- BOOTSTRAP5

HTML: HTML is the standard markup language for documents designed to be displayed in a web browser.

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.

JAVASCRIPT: JavaScript is a powerful and flexible programming language. It can execute on a web browser that allows us to make interactive webpages.

BOOTSTRAP5: Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development.

BACK-END TOOL:

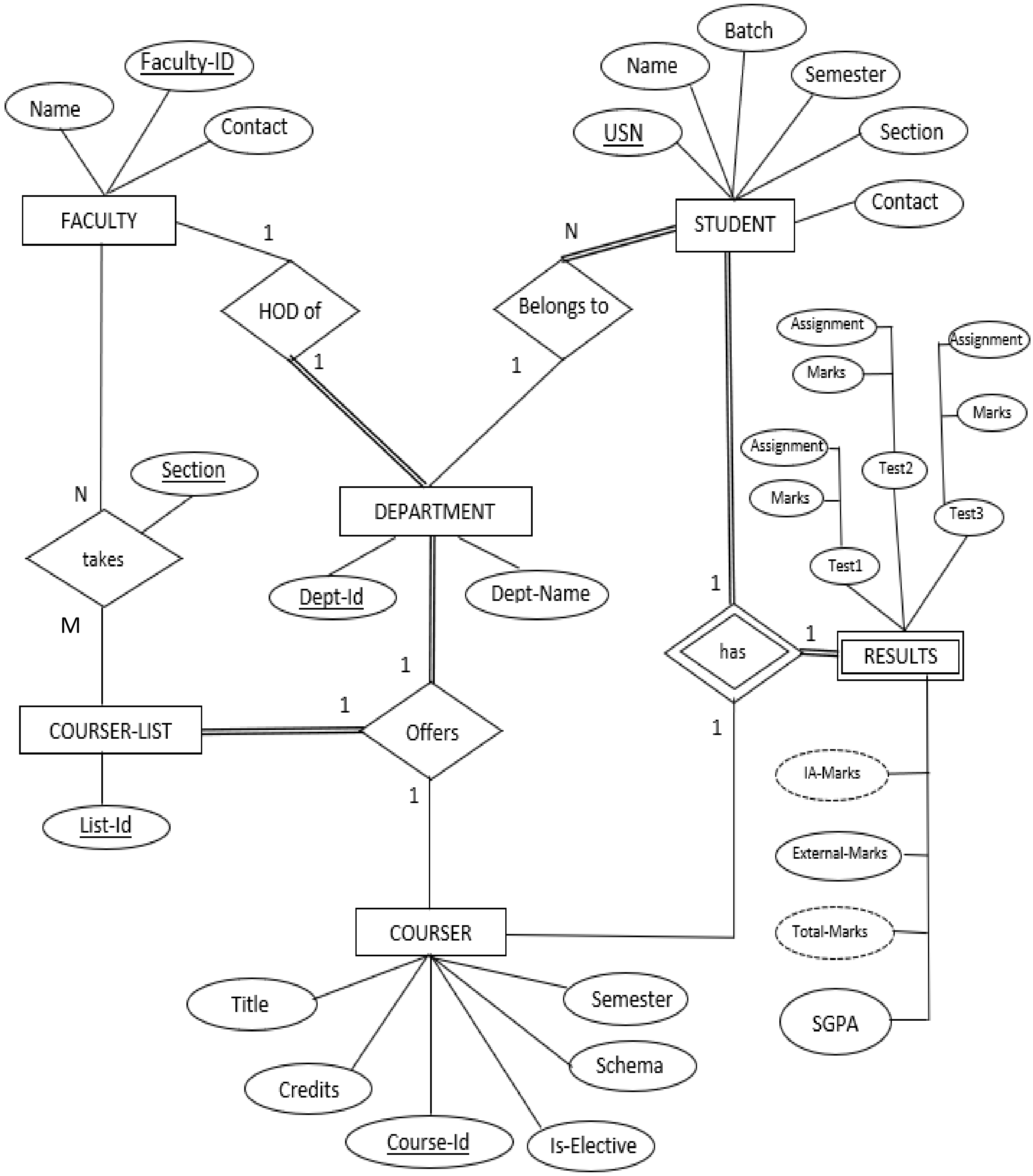
- Python
- NodeJS
- Oracle

Python: Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

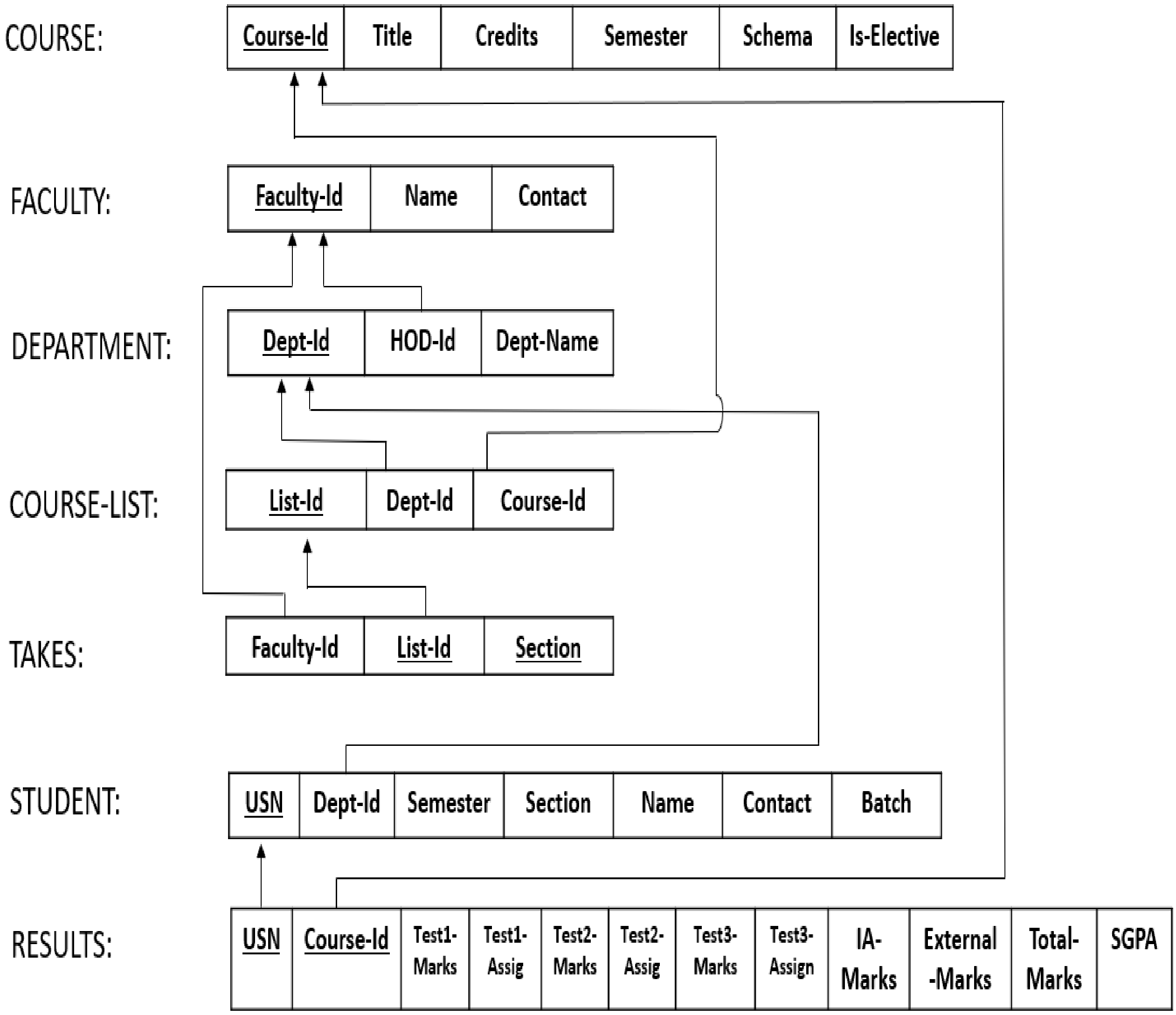
NodeJS: Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

Oracle: Oracle is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database.

ER-Diagram:


FACULTY
Name
Faculty-ID
Contact
HOD of
1
1
N
takes
Section
M
COURSER-LIST
List-Id
DEPARTMENT
Dept-Id
Dept-Name
Offers
1
1
1
1
COURSER
Title
Credits
Course-Id
Is-Elective
Schema
Semester
STUDENT
Name
Batch
Semester
USN
Section
Contact
Belongs to
N
1
has
1
1
1
RESULTS
Assignment
Marks
Test1
Assignment
Marks
Test2
Assignment
Marks
Test3
IA-Marks
External-Marks
Total-Marks
SGPA

ER to Relational mapping:



NORMALIZATION:

❖ 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
17MAT11	Math-I	4	1	2017	0 (False)
17PHY12	Physics	4	1	2017	0 (False)
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	**Course-Id**	**Dept-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:**

<u>**USN**</u>	**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:**

<u>**USN**</u>	<u>**Course-Id**</u>	**Test-1 Marks**	**Test-I Assig.**	**Test-2 Marks**	**..**	**IA Marks**	**External Marks**	**Total Marks**	**SGPA Points**
1BI19CS002	18MAT11	30	10	30		40	49	89	36
1BI19CS002	18CS54	29	8	25		35	45	80	27
1BI19CS058	18CS54	30	10	28		38	48	86	27
1BI19CS161	18MAT11	25	7	20		32	50	82	36
1BI19CS161	18EE78	15	8	25		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.

MODULE DESCRIPTION:

❖ **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:** This module 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 to use 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 the Students to check their marks (updated by faculty in-charge). Students can also check their previous semester results.
    - **Contact-Faculty:** This module connects 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.

APPLICATIONS:

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