

# **CHAPTER 1**

## **INTRODUCTION**

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Overview of Database Management System**

Database is the collection of related data. A software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications is known as Database Management System.

During the 1960s, the first Database Management System was developed by IBM and was known as Information Management System or the IMS. The IMS was originally created for NASA's Apollo Program.

Charles Bachman was an Industrial Researcher who developed a Database Management System known as Integrated Database Store or the IDS. The system was standardized by Committee on Data Systems Languages or the CODASYL.

The IDS and the IMS are known as the precursors of navigational databases. But the main problem of these two systems was they were not easy to use. Therefore, during 1970s, a man named Edward Codd thought of an easy way to use. He wrote a paper entitled, "A Relational Model for Large Shared Data Banks", in which he proposed replacing these current systems with that of a table which consisted of rows & columns. This concept would become Relational Database Management System or the RDBMS.

Today, the RDBMS is used everywhere in the world. It is also very easy to create Database Management Systems using this concept.

### **1.2 Problem Statement**

It is very important to maintain and handle information of a school in a user-friendly software. This application provides a way to record and access information in a simple way.

### **1.3 Objectives**

The main objective of this application is to:

1. Create a user-friendly database for school management.

## **School Database Management System**

---

2. Show the different subjects belonging to different departments taught in the school.
3. Manage the students' information which includes Student ID, name, date of birth and ID of the subject currently studying.
4. Manage the teachers' information which includes Teacher ID, name and the subject he/she teaches.
5. Manage the information of the report card of the student which has the grade points of all the students from every single department.
6. Use an appropriate GUI to display all the stored and newly inserted, updated data or deleted data.
7. Trigger events whenever updating of subject takes place in the student database so as to track the presence of the student.
8. Show execution of queries from the front-end.
9. Show normalization.

# **CHAPTER 2**

## **SYSTEM REQUIREMENTS**

## **CHAPTER 2**

### **SYSTEM REQUIREMENTS**

#### **2.1 Software and hardware requirements**

1. Front-End Tool – Sublime Text Editor 3 will be used to write the front-end code in HTML and PHP as HTML is widely used and PHP is easy to write and execute codes in it plus it also offers many different functionalities that is useful to connect to the back-end software, phpMyAdmin.

The front-end software is the software from which the database can be retrieved or accessed from the back-end tool. PHP is much simpler programming language when compared with other languages such as JavaScript.

2. Back-End Tool – phpMyAdmin in WAMP server 3.1.0 64-bit version will be used to create the back-end of the application with WAMP server. MySQL version 5.7.19 is already present in WAMP (Windows Apache MySQL PHP & other programming languages) which is used to create the tables and again uses PHP or any other programming language to connect to MySQL database. PHP already has the required facilities to connect to MySQL databases.

A Back-End software, in this case, is a software that is used to create the database. MySQL is one of the most popular applications for creating and designing database applications. MySQL has a very user-friendly interface through which the databases can be created easily. It serves as the perfect back-end tool.

These are the final software requirements in order to create the application.

Hardware – PC used is Dell Inspiron 5577 having following hardware specifications:

1. Intel core i7 7<sup>th</sup> Gen processor.
2. Windows 10 Home Operating System.
3. 8GB RAM.
4. 8GB Graphics Processing Unit.
5. 128GB SSD.

# **CHAPTER 3**

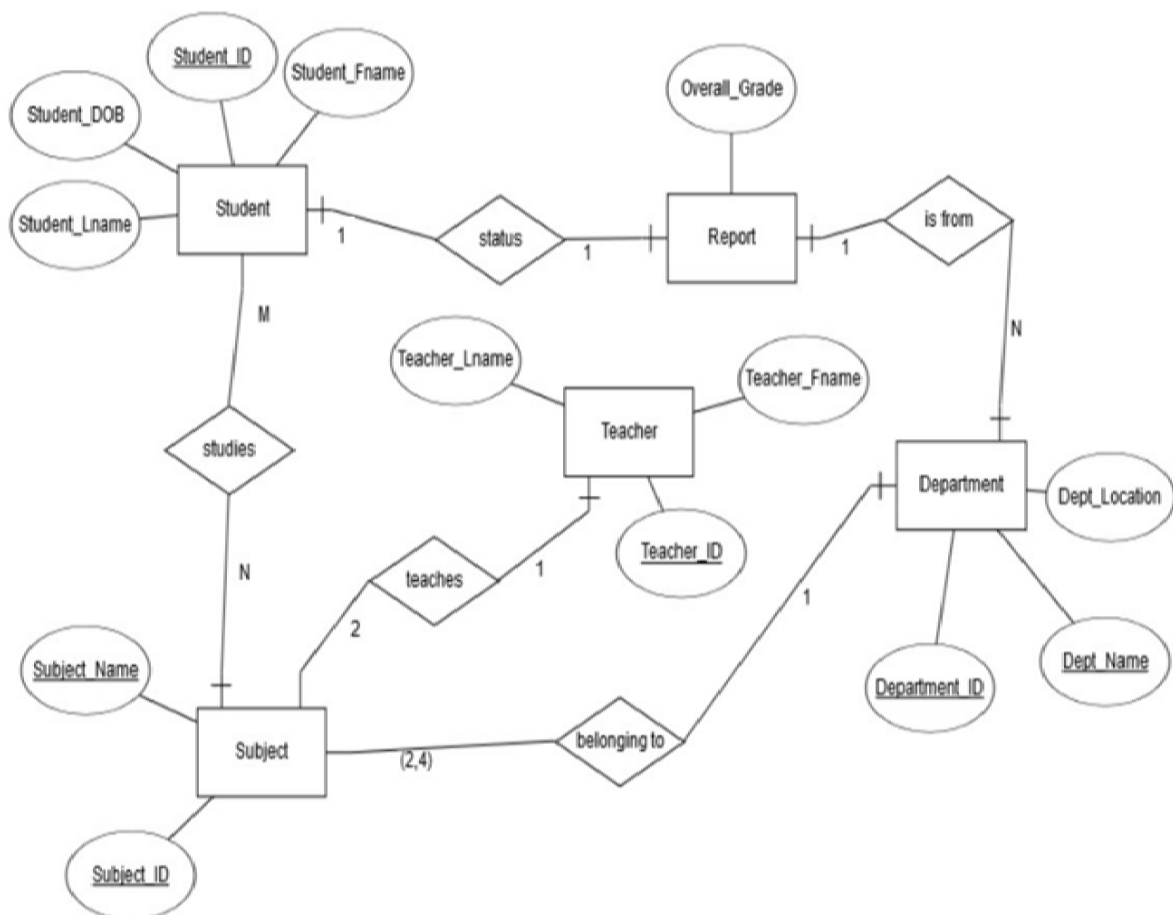
## **SYSTEM DESIGN**

## CHAPTER 3

### SYSTEM DESIGN

The system design shows the way how the application is designed. This application design is in the form of Entity Relationship diagram and Schema Diagram.

#### 3.1 Entity Relationship Diagram



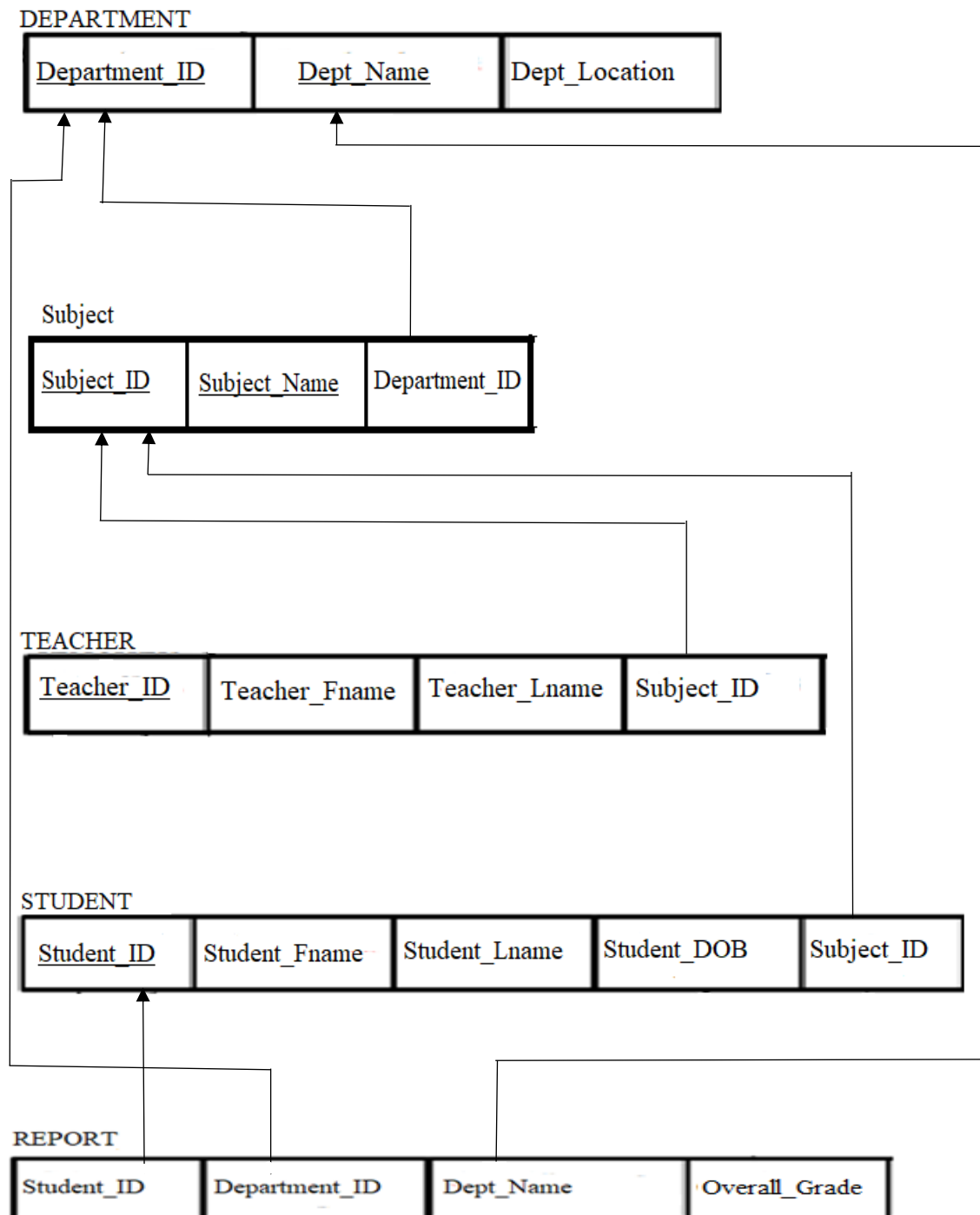
**Figure 3.1 – E R Diagram for the given application**

Figure 3.1 gives the relation between different entities to form the user-friendly application.

The ER Diagram gives the relationship and also the cardinality ratio involved in it.

"A picture is worth 100 words". This ER Diagram shows the pictorial or graphical representation of the database application.

## 3.2 Schema Diagram



**Figure 3.2 – Schema Diagram**

Figure 3.2 - Schema Diagram, shows all attributes with their constraints and are useful to determine relations. The schema diagram shown in the figure shows how different entities of the school database management application are related based on primary/unique key and foreign key constraints.



### 3.3 Overview of Graphical User Interface

Xerox PARC's Computer Scientist Alan Kay and others along with him developed the Alto Computer, the first revolutionary computer to feature the use of Bit-Mapped screen, and was the first computer to demonstrate the desktop metaphor and graphical user interface (GUI). It used icons, windows and also first pop-up menus. Xerox's work was further extended by Apple, when it built the first commercial GUI-based PC with drop-down menus in it, which was called the LISA (Low Integrated System Architecture), directly inspired from the Alto computer. Then it extended to the Macintosh, which featured 128k of macOS version 1.

Later, Microsoft also built their own GUI-based OS, the Windows, which currently dominates in the field of computing and games. Windows is today the best Open-Source OS.

This project features the use of GUI using the web browsers such as Internet Explorer, Mozilla Firefox, Microsoft Edge etc. The back-end software is MySQL. For the front-end, Sublime Text Editor 3 is used to create the HTML and PHP files. These files are executed in Microsoft Edge, which acts like the GUI to the application. These files create pages and links through which traversing from one page to another is made easier using buttons. Ex: - We can traverse from the Student Details page to inserting the new data and deleting any data present in the database.

The GUI serves as the core part of any application on a computer, through which many functions can be performed easily. One of the major GUIs is the web browser. HTML, PHP, CSS etc. are executed on the web browser.

### 3.4 Normalization

Normalization is a database design technique which organizes tables in a manner that reduces redundancy and dependency of data.

It divides larger tables to smaller tables and links them using relationships. There are 3 types of Normal forms that will be discussed here -

1. 1NF
2. 2NF
3. 3NF

### 3.4.1 1NF (First Normal Form)

The rules of the First Normal Form are:-

1. Attributes of the relation should not be multivalued.
2. Attributes of the relation should not be a composite attribute.
3. 1NF allows only single atomic values.

Take example of student table.

The relation can be given as

R(Student\_ID, Student\_Fname, Student\_Lname, Student\_DOB, Subject\_ID)

Functional Dependency (FD) which violated 1NF is

Student\_Name  $\longrightarrow$  Student\_Fname Student\_Lname .....(FD 3.1)

FD 3.1 violates 1NF as the attribute is composite attribute. Therefore, there is a need to resolve it into individual components to avoid 1NF violation.

Therefore, the new FD is – Student\_Name  $\longrightarrow$  Student\_Fname, Student\_Lname. This is shown in Figure 3.1 E R Diagram.

### 3.4.2 2NF (Second Normal Form)

The rules of 2NF are: -

1. Non-Prime attribute should determine Non-Prime attribute.
2. If AB is the candidate key of the relation R(A, B, C, D, E) and if AB as a whole determines any of the non-prime attributes, then it is not a violation. If only A or only B determines any of the non-prime attributes, then it is a 2NF violation.
3. Prime attribute should never determine any of the non-prime attribute.

An example from the given application –

R(Department\_ID, Dept\_Name, Dept\_Location, Subject\_ID, Subject\_Name)

Here, the FDs can be given as –

Department\_ID  $\longrightarrow$  Dept\_Name, Dept\_Location .....(FD 3.2)

Subject\_ID  $\rightarrow$  Subject\_Name, Department\_ID .....(FD 3.3)

Here, Subject\_ID is the candidate key as well as the only prime attribute as it can derive all the attributes in the given relation. It is because –

1. From FD 3.3 –

$(\text{Subject\_ID})^+ = (\text{Subject\_ID}, \text{Subject\_Name}, \text{Department\_ID})$  .....(CLOSURE 3.1)

2. From FD 3.2 –

$(\text{Department\_ID})^+ = (\text{Department\_ID}, \text{Dept\_Name}, \text{Dept\_Location})$  .....(CLOSURE 3.2)

Where ‘+’ as a superscript represents closure of the attribute.

Therefore, from CLOSURE 3.1 and CLOSURE 3.2,

$(\text{Subject\_ID})^+ = (\text{Subject\_ID}, \text{Subject\_Name}, \text{Department\_ID}, \text{Dept\_Name}, \text{Dept\_Location})$  .....(CLOSURE 3.3)

CLOSURE 3.3 indicates that Subject\_ID can derive all attributes when closure is applied and so, it is a candidate key.

Here, FD 3.3 violates 2NF as Prime attribute determines non-prime attributes but the FD 3.2 does not violate the rules of 2NF.

Therefore, there is a need to convert these FDs into relations.

The relations can be broken down as: -

(i) R1(Department\_ID, Dept\_Name, Dept\_Location)

(ii) R2(Subject\_ID, Subject\_Name, Department\_ID)

### 3.4.3 3NF (Third Normal Form)

The rule of 3NF is: -

Non-Prime attribute/s should not determine any non-prime attribute/s.

Again taking the relation between Subject table and Department table, the relation is: -

R(Department\_ID, Dept\_Name, Dept\_Location, Subject\_ID, Subject\_Name)

From the above relation, FDs formed are FD 3.2 and FD 3.3. Therefore, the candidate key and the only prime attribute is Subject\_ID.

FD 3.2 violates 3NF as non-prime attribute Department\_ID derives two other non-prime attributes which are Dept\_Name and Dept\_Location.

Therefore, there is a need to decompose the relation R into 2NF and 3NF relations.

Firstly, decompose those which violates 3NF and then those which violates 2NF.

Resolving the FDs into relations –

R(Department\_ID, Dept\_Name, Dept\_Location, Subject\_ID, Subject\_Name) is already in 1NF Form.

On progressive normalization of relation R into 3NF design, the relations obtained are: -

1. R1(Subject\_ID)
2. R2(Subject\_ID, Subject\_Name, Department\_ID)
3. R3(Department\_ID, Dept\_Name, Dept\_Location)

Both R1 and R2 are in 2NF.

R3 is in 3NF.

# **CHAPTER 4**

## **IMPLEMENTATION**

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 Table Creation

1. create table Department(Department\_ID integer primary key, Dept\_Name varchar(20) UNIQUE, Dept\_Location varchar(20));
2. create table Subject(Subject\_ID integer primary key, Subject\_Name varchar(20) UNIQUE, Department\_ID integer references Department(Department\_ID));
3. create table Student(Student\_ID integer primary key, Student\_Fname varchar(20), Student\_Lname varchar(20), Student\_DOB Date, Subject\_ID integer references Subject(Subject\_ID));
4. create table Teacher(Teacher\_ID integer primary key, Teacher\_Fname varchar(20), Teacher\_Lname varchar(20), Subject\_ID integer references Subject(Subject\_ID));
5. create table Report(Student\_ID integer references Student(Student\_ID), Department\_ID integer references Department(Department\_ID), Dept\_Name varchar(20) references Department(Dept\_Name), Overall\_Grade varchar(2));

#### 4.2 Table Description

1. desc Student;

Field	Type	Null	Key	Default
Student_ID	int(11)	NO	PRI	NULL
Student_Fname	varchar(20)	YES		NULL
Student_Lname	varchar(20)	YES		NULL
Student_DOB	date	YES		NULL
Subject_ID	int(11)	YES		NULL

**Figure 4.1 – Description of Student Table**

2. desc Teacher;

Field	Type	Null	Key	Default
Teacher_ID	int(11)	NO	PRI	NULL
Teacher_Fname	varchar(20)	YES		NULL
Teacher_Lname	varchar(20)	YES		NULL
Subject_ID	int(11)	YES		NULL

**Figure 4.2 – Description of Teacher table**

3. desc Subject;

Field	Type	Null	Key	Default
Subject_ID	int(11)	NO	PRI	NULL
Subject_Name	varchar(20)	YES	UNI	NULL
Department_ID	int(11)	YES		NULL

**Figure 4.3 – Description of Subject table**

4. desc Department;

Field	Type	Null	Key	Default
Department_ID	int(11)	NO	PRI	NULL
Dept_Name	varchar(20)	NO	UNI	NULL
Dept_Location	varchar(20)	YES		NULL

**Figure 4.4 – Description of Department table**

5. desc Report;

Field	Type	Null	Key	Default
Student_ID	int(11)	YES		NULL
Department_ID	int(11)	YES		NULL
Dept_Name	varchar(20)	YES		NULL
Overall_Grade	varchar(2)	YES		NULL

**Figure 4.5 – Description of Report Table**

## 4.3 Populated Tables

1. select \*from Teacher;

Teacher_ID	Teacher_Fname	Teacher_Lname	Subject_ID
1000	Shruti	Shenoy	111
1100	Prashant	Mitra	112
1200	Malavika	A	112
1300	Mohammad	Salman	221
1400	Daisy	Singh	332
1500	Marshall	G	441
1600	Megha	Vasant	552

**Figure 4.6 – Teacher Table**

2. select \*from Department;

Department_ID	Dept_Name	Dept_Location
11	Mathematics	Ground Floor
22	Science	First Floor
33	Computer Science	Second Floor
44	Hindi	Third Floor
55	English	Fourth Floor

**Figure 4.7 – Department Table**

3. select \*from Subject;

Subject_ID	Subject_Name	Department_ID
111	Arithmetic	11
112	Algebra	11
113	Geometry	11
221	Physics	22
222	Chemistry	22
223	Biology	22
331	C Programming	33
332	Web Programming	33
333	Visual BASIC	33
441	Hindi Grammar	44
442	Hindi Literature	44
551	English Grammar	55
552	English Literature	55

**Figure 4.8 – Subject Table**



4. select \*from Report;

Student_ID	Department_ID	Dept_Name	Overall_Grade
1	11	Mathematics	O
2	22	Science	B
11	33	Computer Science	B
20	44	Hindi	A
60	55	English	C
70	11	Mathematics	O

**Figure 4.9 – Report table**

5. select \*from Student;

Student_ID	Student_Fname	Student_Lname	Student_DOB	Subject_ID
1	Ashish	Choudhary	1997-09-12	441
2	Avinash	Patil	1997-10-23	111
11	Farhan	Thackeray	1997-01-26	222
20	Indhu	Verma	1997-04-02	222
35	Lavanya	Verma	1997-04-30	333
45	Parikshith	Verma	1997-04-04	442
60	Rohan	M	1998-12-16	551

**Figure 4.10 – Student table**

## **4.4 SQL Triggers & Stored Procedures**

### **4.4.1 SQL Triggers**

A Trigger is a stored procedure which runs automatically when various events such as insertion, updating or deleting one or more multiple data in the given table. Most triggers are defined to run when changes are made to a table's data. Triggers can be defined to run instead of or after DML (Data Manipulation Language) actions such as INSERT, UPDATE, and DELETE.

In this database, the triggers are used to record events after updating the 'Subject\_ID' attribute inside the Student table. The SQL Query is as shown below: -

```
CREATE TABLE student_audit(id int AUTO_INCREMENT PRIMARY KEY,  
Student_ID int references Student(Student_ID), Subject_ID int references  
Subject(Subject_ID), changedat datetime, action varchar(50));
```

The trigger is as shown below and the code is based on the syntax of phpMyAdmin.

```
DROP TRIGGER IF EXISTS `after_student_update`;

CREATE DEFINER = `root`@`localhost` #User defined here is root@localhost

TRIGGER `after_student_update`

AFTER UPDATE

ON `student`

FOR EACH ROW

BEGIN

INSERT INTO student_audit

SET action = 'update', Student_ID = OLD.Student_ID, Subject_ID=NEW.Subject_ID,

changedat=NOW();

END
```

First, create student\_audit table and then, create the trigger as shown above. Before creating the trigger, check whether the given trigger exists or not. If so, then delete it. After updating the Subject\_ID for a given student, the student\_audit table displays the time and date at which it is changed and this is possible by the NOW() function. This function displays the current date and time of updating the Subject\_ID. action = 'update' shows that we are updating the table, changedat attribute shows the date and time at which updating takes place. id autoincrements after every update.

The table is as shown after updating of Subject\_ID.

id	Student_ID	Subject_ID	changedat	ACTION
1	1	552	2017-11-16 23:47:46	update
2	1	552	2017-11-16 23:48:53	update
3	1	222	2017-11-16 23:49:52	update
4	1	222	2017-11-16 23:58:58	update

**Figure 4.11 – student\_audit table after updating.**

### 4.4.2 SQL Stored Procedure

A stored procedure is a user defined function consisting of a piece of code written in the local version of SQL, which may return a value that is invoked by calling it

explicitly. One of the most beneficial reasons to use stored procedures is the added layer of security that can be placed on the database from the calling application. In this application, the stored procedure is used to display the contents of the 'report' table. The code shown below is based on the syntax of phpMyAdmin.

```
DROP PROCEDURE `p1`; #Delete procedure p1 if it already exists
```

```
CREATE DEFINER=`root`@`localhost`
```

```
PROCEDURE `p1`()
```

```
NOT DETERMINISTIC # It says that the procedure is not deterministic
```

```
READS SQL DATA SQL SECURITY DEFINER /*procedure reads sql data and also  
there is a security definer*/
```

```
SELECT Student_ID, Department_ID, Dept_Name, Overall_Grade
```

```
FROM report
```

It creates p1 procedure which when called by using "call p1()", it displays the data of the report table. Stored procedures can be invoked anytime the user wants instead of writing the whole query again and again. It saves a lot of time.

```
call p1();
```

Student_ID	Department_ID	Dept_Name	Overall_Grade
1	11	Mathematics	O
2	22	Science	B
11	33	Computer Science	B
20	44	Hindi	A
60	55	English	C
70	11	Mathematics	O

**Figure 4.12 – Stored Procedure for displaying report contents**

## 4.5 Database Connectivity

The main front-end tool used to connect the back-end tool WAMP Server's phpMyAdmin is PHP. PHP is one of the simplest tools to create front-end that is, the GUI of the application.

### 4.5.1 Procedure for connection to the MySQL Database in phpMyAdmin

The instruction for connecting the database is: -

//Choose the database for which connection should be made. Variable initialized for //this purpose is \$database (in PHP, \$ declares that the character/string following it is a variable)

```
$database = "studentdatabase";
```

//Declare variable \$con to connect to the database using the in-built PHP function //and its signature given by - mysqli\_query(host name, username, password)

```
$con = mysqli_query("localhost", "root", "airbusa380800");
```

//Select database \$database using built-in function and its signature //given by - mysqli\_select\_db(connection, database name) or die(mysqli\_error()). //die(mysqli\_error()) is used to show the error in connectivity.

```
mysqli_select_db($con, $database) or die(mysqli_error());
```

//Now, declare \$result as variable used to execute the given query using built-in //function and its signature given by – mysqli\_query(connection, query)

```
$result = mysqli_query($con, "SELECT Student_ID, Student_Fname, Student_Lname, Student_DOB, Subject_ID FROM Student");
```

The above code snippet shows how the MySQL Database is connected using the PHP as the front-end tool and because of this, instead of writing queries and modifying the data, we can easily do it in front end by just giving the query once in the program.

## **CHAPTER 5**

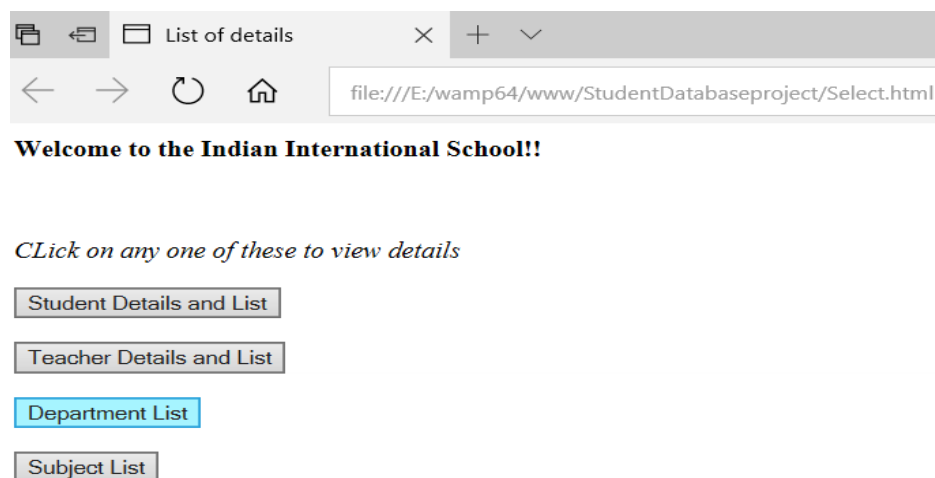
### **RESULTS**

## CHAPTER 5

### RESULTS

Figure 5.1 shows the Select page where there are 4 buttons. On clicking the "Department List" button, it leads to the page as shown in Figure 5.2.

The Figure 5.2 shows the list of all the departments present in the school and their respective locations.

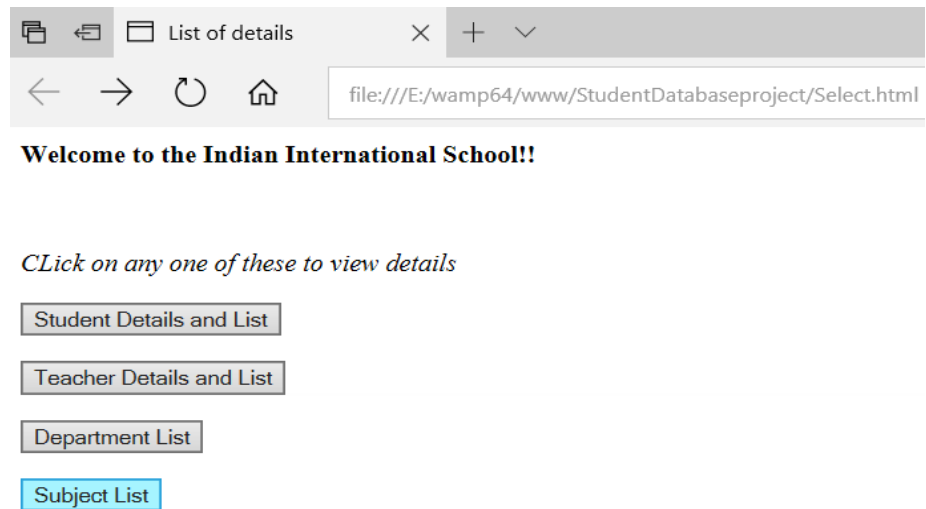


**Figure 5.1 – Select Page**

Department ID	Department Name	Department Location
11	Mathematics	Ground Floor
22	Science	First Floor
33	Computer Science	Second Floor
44	Hindi	Third Floor
55	English	Fourth Floor

Figure 5.2 is a screenshot of a web browser displaying the 'Department List' page. The browser window has a title bar that says 'localhost'. The address bar shows the URL 'localhost/StudentDatabaseproject/Dept.php'. Below the browser window is a table with three columns: 'Department ID', 'Department Name', and 'Department Location'. The table contains five rows of data.

**Figure 5.2 – The Department List**



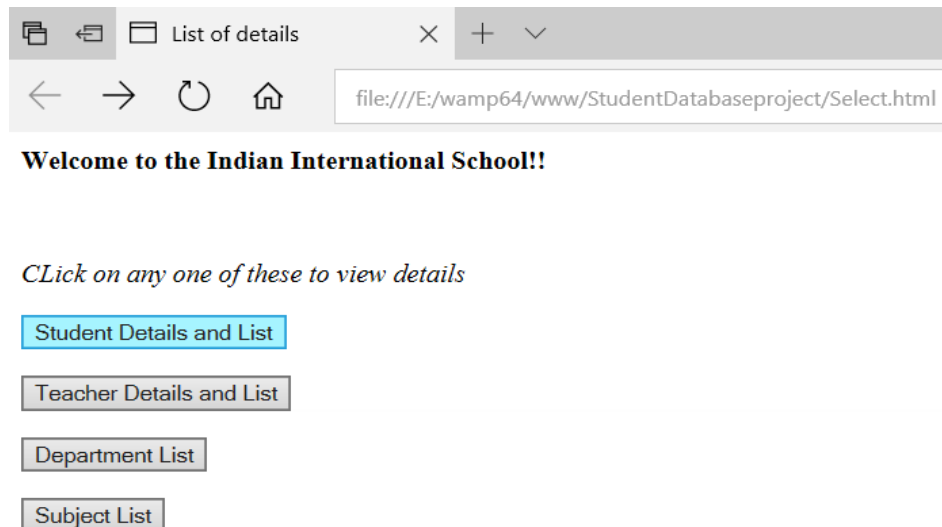
**Figure 5.3 – Selecting the ‘Subject List’**

localhost/StudentDatabaseproject/Subject.php

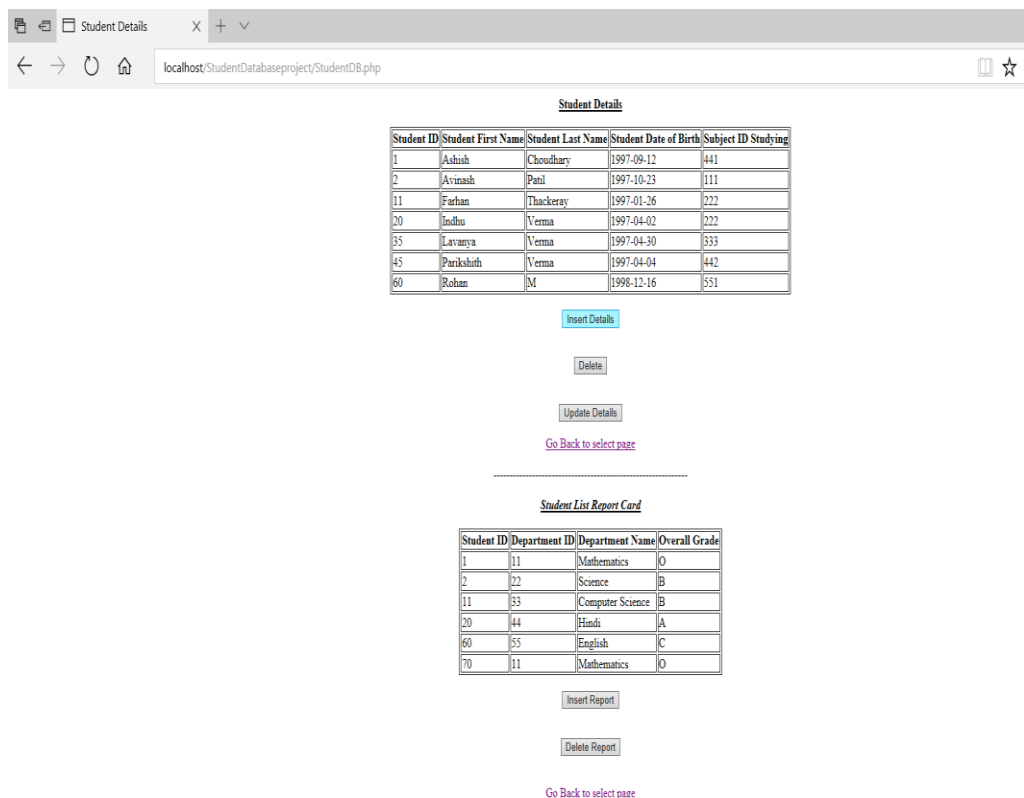
Subject ID	Subject Name	Department ID
111	Arithmetic	11
112	Algebra	11
113	Geometry	11
221	Physics	22
222	Chemistry	22
223	Biology	22
331	C Programming	33
332	Web Programming	33
333	Visual BASIC	33
441	Hindi Grammar	44
442	Hindi Literature	44
551	English Grammar	55
552	English Literature	55

**Figure 5.4 – Subject List**

On selecting the "Subject List" button as shown in Figure 5.3, it leads to the page shown in Figure 5.4, where the subject list and the respective Department IDs are displayed.



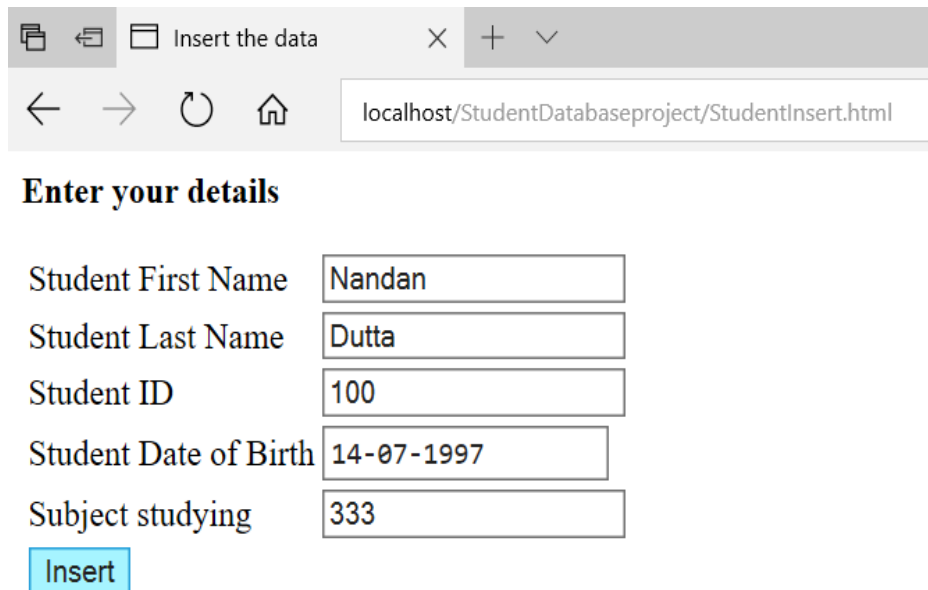
**Figure 5.5 – Selecting Student Details and List.**



**Figure 5.6 – Student Details page.**

Figure 5.5 shows the selection of "Student List and Details" button. On pressing the button, it leads to page as shown in Figure 5.6, which shows the Student Details table and also, the Student List Report Card table. In Figure 5.6, on pressing the "Insert Details" button below the Student Details, it leads to the page as shown in Figure 5.7.



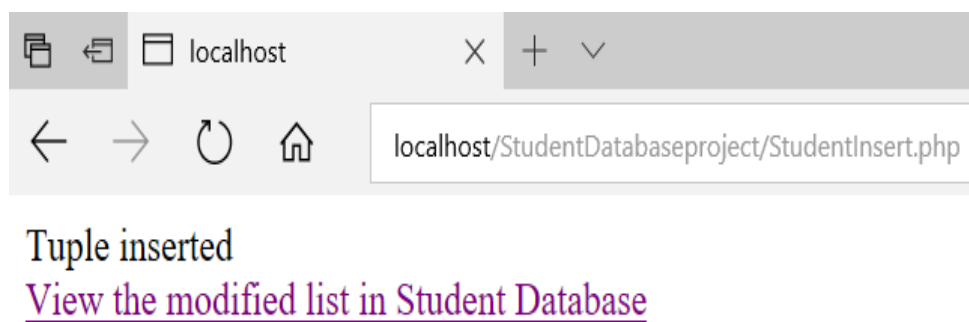


The screenshot shows a web browser window with the address bar displaying 'localhost/StudentDatabaseproject/StudentInsert.html'. The page title is 'Insert the data'. Below the title, there is a form titled 'Enter your details'. The form contains five input fields: 'Student First Name' with the value 'Nandan', 'Student Last Name' with the value 'Dutta', 'Student ID' with the value '100', 'Student Date of Birth' with the value '14-07-1997', and 'Subject studying' with the value '333'. Below the input fields is a blue 'Insert' button.

Student First Name	Nandan
Student Last Name	Dutta
Student ID	100
Student Date of Birth	14-07-1997
Subject studying	333

[Insert](#)

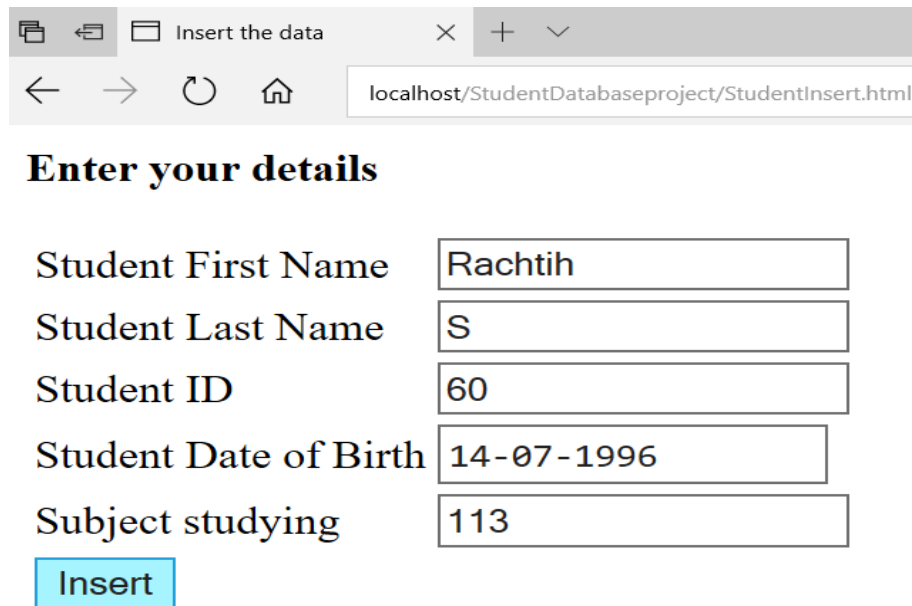
**Figure 5.7 – The Student Detail Insertion.**



**Figure 5.8 – Success Message**

In Figure 5.7, the student details are inserted and on pressing the "Insert" button, it leads to page as shown in Figure 5.8, which says "Tuple inserted". On clicking the blue link "View the modified list in Student Database", it leads to the page as shown in the Figure 5.11, which shows modified table with the new inserted data.

The Figures 5.9 and 5.10 shows the errors in inserting the student with a different name but an already existing Student ID.

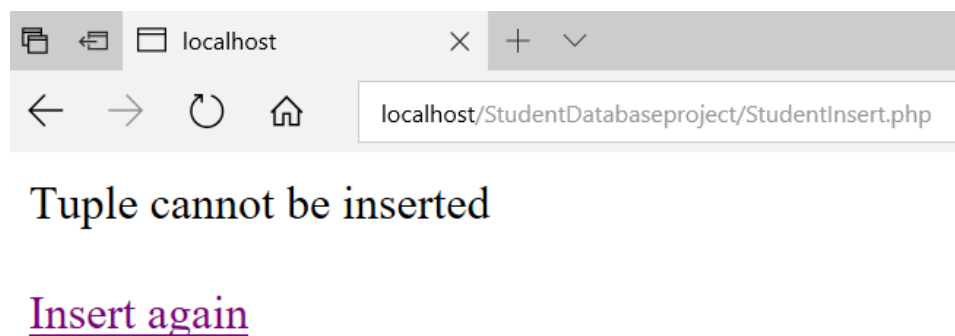


The screenshot shows a web browser window with the address bar displaying 'localhost/StudentDatabaseproject/StudentInsert.html'. The page title is 'Insert the data'. Below the title, there is a form titled 'Enter your details'. The form contains five input fields: 'Student First Name' with the value 'Rachtih', 'Student Last Name' with the value 'S', 'Student ID' with the value '60', 'Student Date of Birth' with the value '14-07-1996', and 'Subject studying' with the value '113'. Below these fields is a blue button labeled 'Insert'.

Student First Name	Rachtih
Student Last Name	S
Student ID	60
Student Date of Birth	14-07-1996
Subject studying	113

[Insert](#)

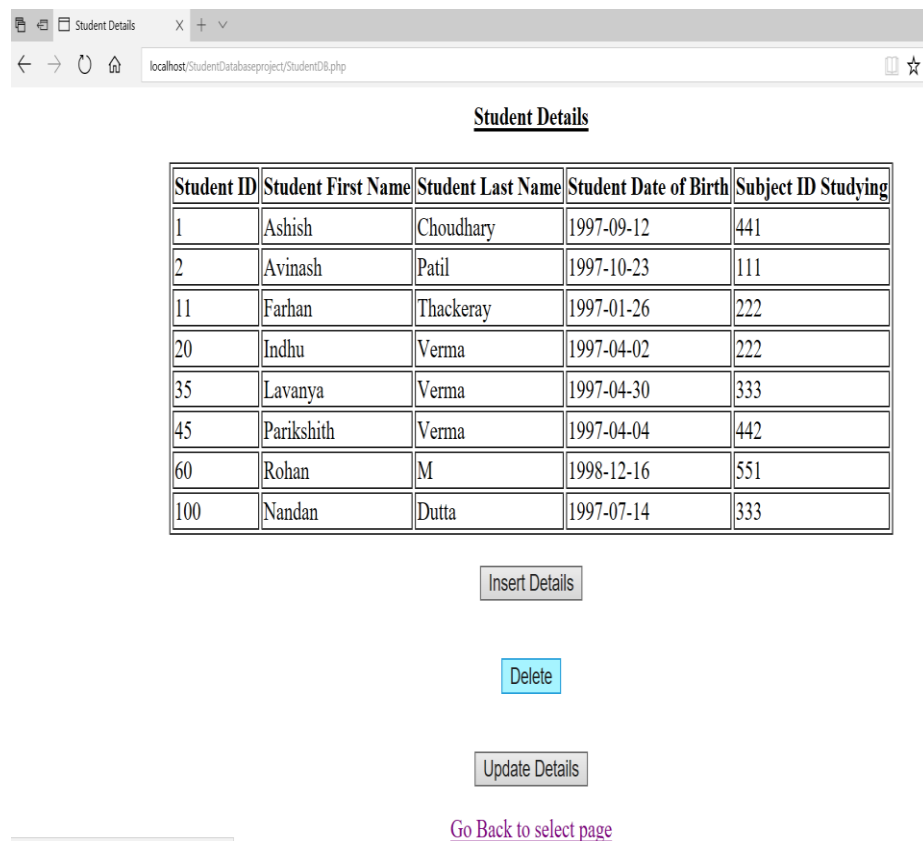
**Figure 5.9 – Inserting student details.**



**Figure 5.10 – Error Message**

In Figure 5.9, a new student is being inserted for the Student Details list table but the main problem here is that the new student has the Student ID which is same as that of Rohan's. This violates primary key constraint. Therefore, due to this, the data will not be inserted and the message displayed is as shown in Figure 5.10.

The user must click on the blue link shown below the error message "Tuple cannot be inserted" in Figure 5.10 and insert the appropriate data.



Student Details

Student ID	Student First Name	Student Last Name	Student Date of Birth	Subject ID Studying
1	Ashish	Choudhary	1997-09-12	441
2	Avinash	Patil	1997-10-23	111
11	Farhan	Thackeray	1997-01-26	222
20	Indhu	Verma	1997-04-02	222
35	Lavanya	Verma	1997-04-30	333
45	Parikshith	Verma	1997-04-04	442
60	Rohan	M	1998-12-16	551
100	Nandan	Dutta	1997-07-14	333

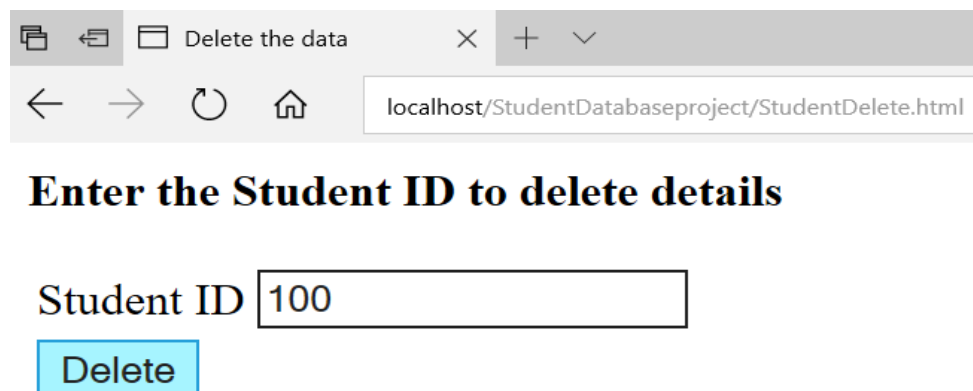
Insert Details

Delete

Update Details

[Go Back to select page](#)

**Figure 5.11 – The modified list of student details.**



Delete the data

localhost/StudentDatabaseproject/StudentDelete.html

**Enter the Student ID to delete details**

Student ID

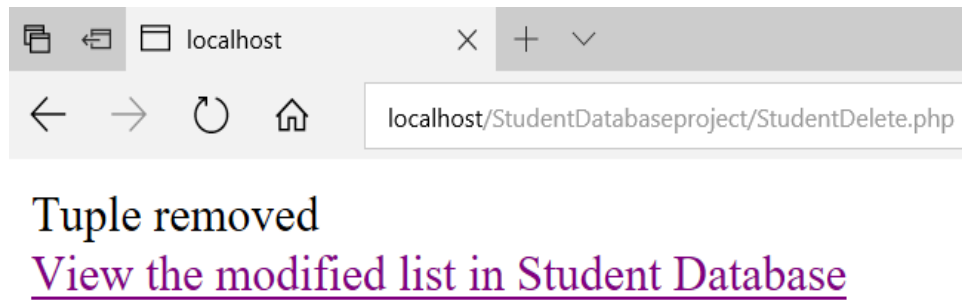
Delete

**Figure 5.12 – Deletion of Student whose ID = 100.**

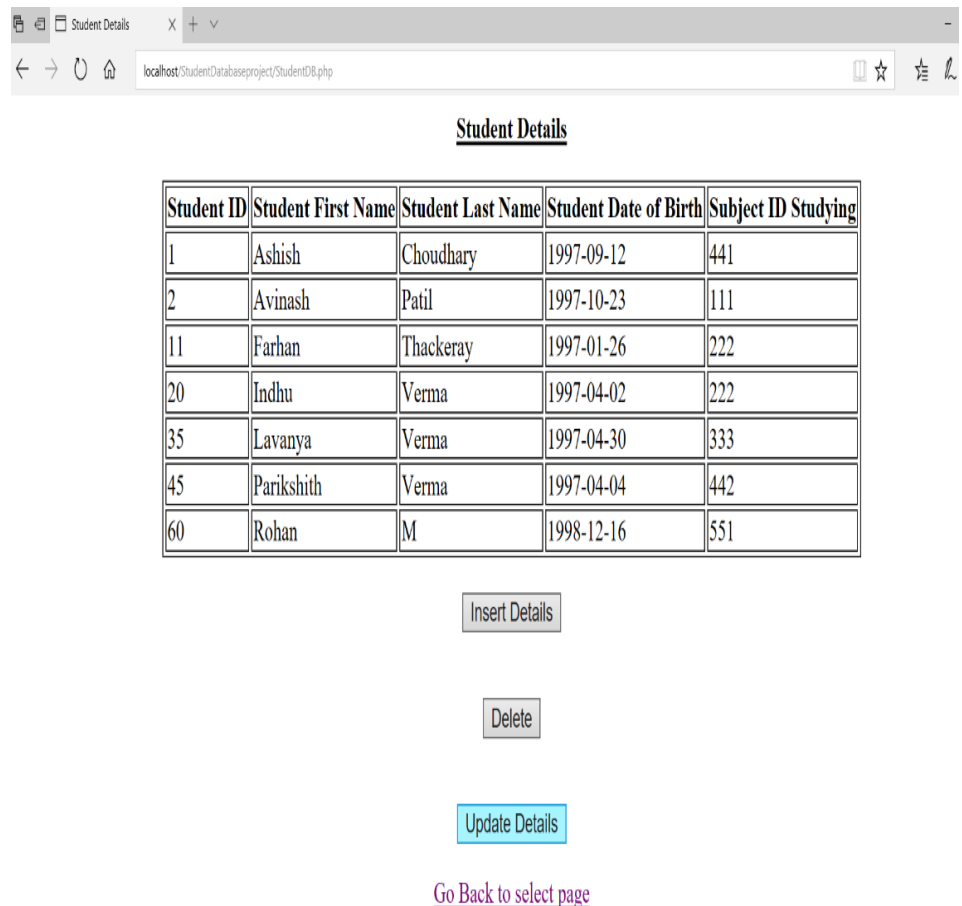
The Figure 5.11 shows the "Delete" button being selected below the modified Student Details list. Whenever the blue link "Go Back to select page" is selected, it will go back to the select page as shown in Figure 5.1.

On selection of the "Delete" button, it leads to the page as shown in Figure 5.12. In this, the student's information to be deleted is done just by entering the Student ID.

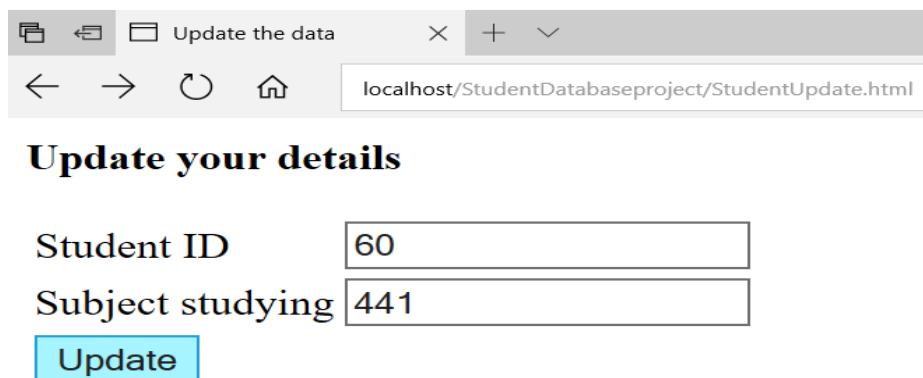
On entering the Student ID and on pressing "Delete" button, Figure 5.13 is shown.



**Figure 5.13 – Data removed message.**

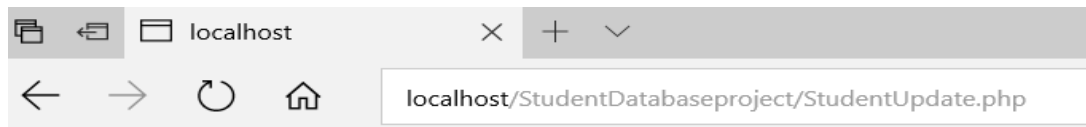


**Figure 5.14 – Modified list after deletion.**



**Figure 5.15 – Updating Subject ID.**

On pressing the blue link as shown in Figure 5.13, it leads to page as shown in Figure 5.14 where the row containing Student with Student ID=100 is deleted. In the same page, select the "Update Details" button to update the Student Details. It leads to page as shown in Figure 5.15. Here, the Student ID and the new Subject ID is entered. On pressing "Update" button, it leads to page as shown in Figure 5.16.



**Tuple updated**

[View the modified list in Student Database](#)

**Trigger table**

ID	Student ID	Subject ID	changedat	action
1	1	552	2017-11-16 23:47:46	update
2	1	552	2017-11-16 23:48:53	update
3	1	222	2017-11-16 23:49:52	update
4	1	222	2017-11-16 23:58:58	update
5	1	112	2017-11-17 00:00:18	update
6	1	112	2017-11-17 00:00:34	update
7	1	332	2017-11-17 13:24:17	update
8	1	441	2017-11-17 13:45:55	update
9	70	111	2017-11-17 17:07:24	update
10	70	111	2017-11-17 17:08:46	update
11	100	551	2017-11-17 17:56:02	update
12	60	441	2017-11-18 06:42:21	update
13	60	441	2017-11-18 06:43:08	update
14	60	441	2017-11-18 06:52:36	update
15	60	441	2017-11-18 06:54:16	update

**Figure 5.16 - Success message and the trigger table.**

Figure 5.16 - The trigger table shows the time and date at which the update is done. On clicking the blue link below "Tuple updated" message, page shown will be Figure 5.17.

## School Database Management System

Student Details

localhost/StudentDatabaseproject/StudentDB.php

**Student Details**

Student ID	Student First Name	Student Last Name	Student Date of Birth	Subject ID Studying
1	Ashish	Choudhary	1997-09-12	441
2	Avinash	Patil	1997-10-23	111
11	Farhan	Thackeray	1997-01-26	222
20	Indhu	Verma	1997-04-02	222
35	Lavanya	Verma	1997-04-30	333
45	Parikshith	Verma	1997-04-04	442
60	Rohan	M	1998-12-16	441

Insert Details

Delete

Update Details

[Go Back to select page](#)

**Figure 5.17 – Updated List of Student Details.**

Figure 5.17 shows the updated list of the of Student where Subject ID of Rohan is updated to 441. Scroll down the same page where Figure 5.18 can be seen.

Student Details

localhost/StudentDatabaseproject/StudentDB.php

**Student List Report Card**

Student ID	Department ID	Department Name	Overall Grade
1	11	Mathematics	O
2	22	Science	B
11	33	Computer Science	B
20	44	Hindi	A
60	55	English	C
70	11	Mathematics	O

Insert Report

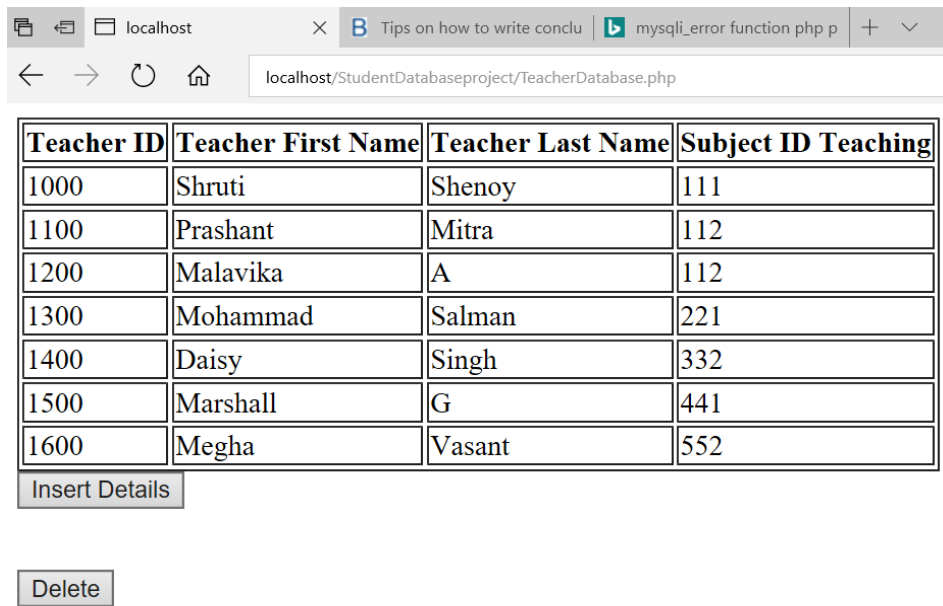
Delete Report

[Go Back to select page](#)

**Figure 5.18 – The Student Report Card List.**

Figure 5.18 shows the Student Report Card List table and Figure 5.19 shows Teacher List and Details. Going back to the Select Page and selecting "Teacher List and Details" buttons, Figure 5.19 is displayed.

## School Database Management System



Teacher ID	Teacher First Name	Teacher Last Name	Subject ID Teaching
1000	Shruti	Shenoy	111
1100	Prashant	Mitra	112
1200	Malavika	A	112
1300	Mohammad	Salman	221
1400	Daisy	Singh	332
1500	Marshall	G	441
1600	Megha	Vasant	552

Insert Details

Delete

[Go Back to select page](#)

**Figure 5.19 – The teacher list and details.**

The Figure 5.19 shows about the details of the teacher.

## **CHAPTER 6**

### **CONCLUSION & FUTURE ENHANCEMENTS**



## **CHAPTER 6**

### **CONCLUSION & FUTURE ENHANCEMENTS**

This database application is able to deliver all the required objectives. The application is able to retrieve, update, insert and update the data in the table in a simple way. The GUI also is designed to be simple so that the user can do important user-friendly actions and events in a short span of time. Hence, the use of Relational DBMS is much more efficient than the traditional file management system.

In future, enhancements can be done to make sure that the whole Relational Database can be in fully normalized form upto BCNF with much more simpler features in GUI. The application can also be made to have separate GUIs for students also. This application is done with keeping only teachers and other important school staff in mind. So, in future, this can be extended to students also.