

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
JNANASANGAMA, BELAGAVI -590 018



A DBMS Mini Project Report
On

STUDENT MANAGEMENT SYSTEM

Submitted in partial fulfillment for the award of degree of

Bachelor of Engineering

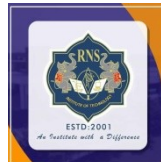
in

Artificial Intelligence and Machine Learning

Submitted by

MD IMTIYAZ ALAM

1RN20AI031



RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)
(UG programs – CSE, ECE, ISE, EIE and EEE are Accredited by NBA up to 30.6.2025)
Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

Department of AI & ML

2022 – 2023

RNS INSTITUTE OF TECHNOLOGY

(AICTE Approved, VTU Affiliated and NAAC 'A' Accredited)

(UG programs – CSE, ECE, ISE, EIE and EEE are Accredited by NBA up to 30.6.2025)

Channasandra, Dr. Vishnuvardhan Road, Bengaluru - 560 098

Department of AI & ML



CERTIFICATE

Certified that the Project entitled **Student Management System** carried out by Mr. **Md Imtiyaz Alam**, USN **1RN20AI031** a bonafide student of V Semester BE, **RNS Institute of Technology** in partial fulfillment for the Bachelor of Engineering in AI & ML ENGINEERING of the **Visvesvaraya Technological University**, Belagavi during the year 2022-23. It is certified that all corrections / suggestions indicated for Internal Assessment have been incorporated in the report. The Project report has been approved as it satisfies the academic requirements in respect of Database Management System with Mini Project Laboratory prescribed for the said Degree.

Course Teacher

Mrs. Sajitha N
Asst. Professor
Department of AI&ML
RNSIT, Bengaluru

HoD

Dr. Harsha S
Department of AI & ML
RNSIT, Bengaluru

Name & Signature

Examiner 1:

Examiner 2:

ABSTRACT

An automated student management system (SMS) is a software application designed to manage and organize student data for educational institutions. The system allows for the efficient and accurate tracking of student information, including personal details, enrollment and grades. It also provides a platform for communication between teachers and students. The system can be accessed through a web-based interface, allowing for remote access and collaboration

The SMS aims to streamline administrative tasks and improve the overall student experience by providing a centralized location for student information and communication. The system can also provide valuable data for decision-making and strategic planning, such as tracking student progress and identifying areas of improvement.

The system allows teachers to input and access student grades and other important information. It also allows administrators to track student enrollment and manage results of students.

One of the key benefits of an SMS is the ability to automate repetitive tasks, such as generating reports and sending reminders. This frees up staff time and resources, allowing them to focus on more important tasks. Additionally, the system can be customized to meet the specific needs of the institution, including integration with other software applications.

Overall, an SMS is a valuable tool for education institutions looking to improve their operations and support student success. It provides a centralized location for student information, automates repetitive tasks, and offers valuable data for decision-making. With the SMS, education institutions can improve communication, efficiency and support student's academic progress, and provide a better overall experience for students and their families.

ACKNOWLEDGEMENTS

At the very onset, we would like to place on record our gratitude to all those people who have helped us in making this project work a reality. Our Institution has played a paramount role in guiding us in the right direction.

I would like to profoundly thank **Sri. Satish R Shetty**, Chairman, RNS Group of Institutions, Bangalore for providing such a healthy environment for the successful completion of this project work.

I would also like to thank our beloved Principal, **Dr. M K Venkatesha**, for providing the necessary facilities to carry out this work.

I am extremely grateful to **Dr. Harsha S**, Head of the Department of Artificial Intelligence and Machine Learning, for having accepted to guide me in the right direction with all his wisdom.

I would like to express our sincere thanks to our Course Teacher **Mrs. Sajitha N** Assistant Professor, Department of Artificial Intelligence and Machine Learning for her constant encouragement that motivated us for the successful completion of this project work.

I would like to thank our guide **Ms. Pooja M** Assistant professor, Department of Artificial Intelligence and Machine Learning for her continuous guidance and constructive suggestions for this project.

Last but not the least, I am thankful to all the teaching and non-teaching staff members of the Artificial Intelligence and Machine Learning Department, parents, sister and friends for their encouragement and support throughout this work.

MD IMTIYAZ ALAM

1RN20AI031

TABLE OF CONTENTS

CONTENTS	Page No.
ABSTRACT	I
ACKNOWLEDGEMENT	II
1. INTRODUCTION	1
1.1 Overview of Database Management Systems	1
1.2 Problem Statement	2
1.3 Objectives	3
2. SYSTEM REQUIREMENTS	4
2.1 Hardware Requirements	4
2.2 Software Requirements	4
3. SYSTEM DESIGN	5
3.1 E R Diagram for Student Management System	5
3.2 Schema Diagram	6
3.3 Overview of GUI	7
3.4 Normalization	12
4. IMPLEMENTATION	15
4.1 Table Creation	15
4.2 Description of Table	17
4.3 Populated tables	22
4.4 SQL Triggers and Stored Procedures	25
4.5 Database Connectivity	26
4.6 Source Code (Front end)	27
5. RESULTS	39
6. CONCLUSION AND FUTURE ENHANCEMENTS	44
6.1 Conclusion	44
6.2 Future Enhancements	44
7. REFERENCES	46

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
3.1	E R Diagram	5
3.2	Schema Diagram	6
3.3	Layers of GUI	7
3.4	Types of Normal Forms	12
4.2.1	Description of admin table	18
4.2.2	Description of tblclasses table	18
4.2.3	Description of tblresult table	19
4.2.4	Description of tblstudents table	20
4.2.5	Description of tblsubjectcombinations table	21
4.2.6	Description of tblsubject table	21
4.3.1	Values of admin	22
4.3.2	Values of tblclasses	22
4.3.3	Values of tblstudents	23
4.3.4	Values of tblresult	23
4.3.5	Values of tblsubjectcombinations	24
4.3.6	Values of tblsubjects	24
5.1	Home page	39
5.2	Admin Dashboard	39
5.3	Classes page	40
5.4	Subjects pages	40
5.5	Students pages	41
5.6	Admin landing page(post login)	41

Figure No.	Name of Figure	Page No.
5.7	Create student class page	42
5.8	Subject creation page	42
5.9	Result page	43
5.10	Student Result	43
5.11	Logout page	44

Chapter 1

INTRODUCTION

1.1 Overview of Database Management Systems

Database is a collection of related data and data is a collection of facts and figures that can be processed to produce information. Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks. A database management system stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics:

- **Real-world entity** – A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behaviour and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- **Relation-based tables** – DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.
- **Isolation of data and application** – A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- **Less redundancy** – DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency** – Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of

leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- **Query Language** – DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties** – DBMS follows the concepts of Atomicity, Consistency, Isolation, and Durability (normally shortened as ACID). These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access** – DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.
- **Multiple views** – DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security** – Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

1.2 Problem Statement

"The college is in need of a student management database system that can effectively store, manage, and retrieve student information. The system should allow authorized staff members to easily access and update student records, track student progress and

enrollment, generate reports, and manage communication with parents and guardians. Additionally, the system should have strong security measures to ensure that sensitive student information is protected and accessible only to authorized individuals. The goal of the system is to streamline administrative tasks and improve the overall efficiency of student management for the college district."

1.3 Objectives

The main objective of a student management database project is to create a centralized, efficient, and secure system for storing, managing, and retrieving student information. The system should automate administrative tasks and improve the overall efficiency of student management for the college district.

In summary, the main objective of the student management database project is to improve the quality and effectiveness of student management by providing a reliable and easy-to-use system for managing and analyzing student data.

Chapter 2

SYSTEM REQUIREMENTS

2.1 Hardware Requirements

The Hardware requirements are very minimal and the program can be run on most of the machines.

Processor : i5 processor

Processor Speed : 1.2 GHz

RAM : 1 GB

Storage Space : 40 GB

Monitor Resolution : 1024*768 or 1336*768 or 1280*1024

2.2 Software Requirements

- Operating System used: Windows 11
- Technologies used: HTML, CSS, PHP, Bootstrap
- XAMPP Server: MySQL, PhpMyAdmin
- IDE used: Visual Studio Code
- Browser that supports HTML

Chapter 3

SYSTEM DESIGN

3.1 E R Diagram for Student Management

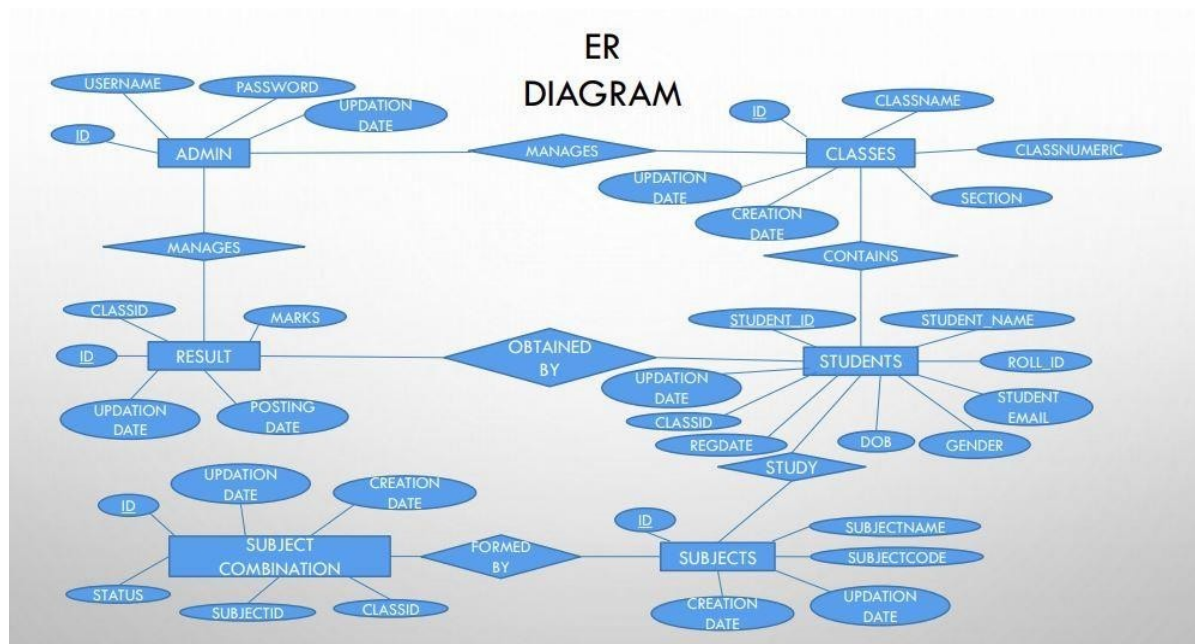


Figure 3.1: E R Diagram

Entity Relationship Diagram displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. Rectangles represent entities, ovals represent attributes and diamond shapes represent relationships.

As seen in the Figure 3.1, the Entity Relationship Diagram for Student Management System clearly specifies the way in which all tables are connected and we can also notice the cardinality ratio associated with each of the connected entities.

User specifies the address and it is seen that address is a weak entity as it is completely dependent on user. The user can select books which has different book genres. The rating is given for books and the user can view all the purchased books. The primary keys of all entities are underlined.

3.2 Schema Diagram



Figure 3.2: Schema Diagram

A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and the relations among them are associated. It formulates all the constraints that are to be applied on data. A database schema defines its entities and relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.

The Figure 3.2 represents schema diagram for Student Management System where foreign keys are referencing primary keys.

The main advantage of using schema diagram is it helps in organizing data into separate entities, making it easier to share a single schema within another database.

3.3 Overview of GUI

GUI is a Graphical Interface that is a visual representation of communication presented to the user for easy interaction with the machine. GUI means Graphical User Interface. It is the common user Interface that includes Graphical representation like buttons and icons, and communication can be performed by interacting with these icons rather than the usual text-based or command-based communication.

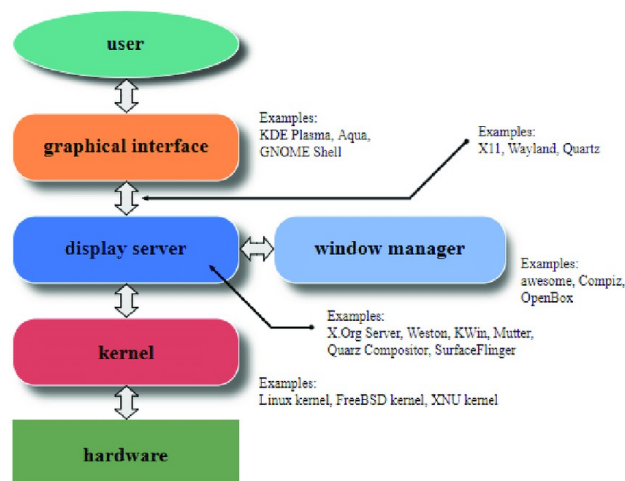


Figure 3.3: Layers of GUI

Abstraction is a major concept that has been used in a GUI operating system. Users can use the pointer to click on the icon, which initiates a series of actions. Normally an application or functionality will get started. Then the user will have to provide input or tasks to generate the desired action from the machine. The GUI actually translates user language, which comprises simple one-line commands, single click and double clicks to machine language or assembly_language. The machine understands machine language, and hence the machine responds to the task initiated, which is translated and communicated to the user via GUI.

A series of elements conforming a visual language have evolved to represent information stored in computers. This makes it easier for people with few computer skills to work with and use computer software. The most common combination of such elements in GUIs is the windows, icons, text fields, canvases, menus, pointer (WIMP) paradigm, especially in personal computers.

The WIMP style of interaction uses a virtual input device to represent the position of a pointing device's interface, most often a mouse, and presents information organized in windows and represented with icons. Available commands are compiled together in menus, and actions are performed making gestures with the pointing device. A window manager facilitates the interactions between windows, applications, and the windowing system. The windowing system handles hardware devices such as pointing devices, graphics hardware, and positioning of the pointer.

- **FRONTEND**

- **HTML :**

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from a local storage and render them to multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects like interactive forms can be embedded into the rendered page. It provides a way to create structured documents by denoting structural semantics for the text like headings, paragraphs, lists, links, quotes and other items. HTML elements are delimited by tags that are written within angle brackets. Tags such as `` and `<input />` introduce content into the page directly. Other tags such as `<p>...</p>` surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can also embed programs written in a scripting language such as JavaScript which affect the behaviour and content of web pages. Inclusion of CSS defines the look and layout of content.

- **CSS:**

Cascading Style Sheets (CSS) is a style sheet language which is used for describing the presentation of a document written in a markup language. Although most often its used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document, including plain XML, SVG and XUL, and is also applicable to rendering in speech, or on other media. Along with HTML and JavaScript, CSS is a cornerstone technology used by most websites to create visually engaging webpages, user interfaces for web applications, and user interfaces for many mobile applications. CSS is designed primarily to enable the separation of presentation and content, including aspects such as the layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple HTML pages to share the formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

- **PHP:**

PHP is a server-side scripting language designed primarily for web development but is also used as a general-purpose programming language. Originally created by Rasmus Lerdorf in 1994, the PHP reference implementation is now produced by The PHP Development Team. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Pre-processor.

PHP code can be embedded into HTML or HTML5 markup, or it can be used in combination with various web template systems, web content management systems and web frameworks. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server software combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code can also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

The standard PHP interpreter, powered by the Zend Engine, is a free software released under the PHP License. PHP has been widely ported and can be deployed on most web servers, on almost every operating system and platform, free of charge. The PHP language evolved without a written formal specification or standard until 2014, leaving the canonical PHP interpreter as a de facto standard. Since 2014 work has gone into creating a formal PHP specification. HP development began in 1995 when Rasmus Lerdorf wrote several Common Gateway Interface (CGI) programs in C, which he used in order to maintain his personal homepage. He extended them to work with web forms and to communicate with databases, and called this implementation "Personal Home Page/Forms Interpreter" or PHP/FI.

PHP/FI could help to build simple, dynamic web applications. To accelerate bug reporting and to improve the code, Lerdorf initially announced the release of PHP/FI as "Personal Home Page Tools (PHP Tools) version 1.0" on the Usenet discussion group on June 8, 1995. This release already had the basic functionality that PHP has as of 2013. This included Perl-like variables, form handling, and the ability to embed HTML. The syntax resembled that of Perl but was simpler, more limited and less consistent.

- **Backend**

- **MYSQL**

MySQL is a Relational Database Management System (RDBMS). MySQL server can manage many databases at the same time. In fact, many people might have different databases managed by a single MySQL server. Each database consists of a structure to hold onto the data itself. A data-base can exist without data, only a structure, be totally empty, twiddling its thumbs and waiting for data to be stored in it.

Data in a database is stored in one or more tables. You must create the data-base and the tables before you can add any data to the database. First

you create the empty database. Then you add empty tables to the database. Database tables are organized in rows and columns. Each row represents an entity in the database, such as a customer, a book, or a project. Each column contains an item of information about the entity, such as a customer name, a book name, or a project start date. The place where a particular row and column intersect, the individual cell of the table, is called a field. Tables in databases can be related. Often a row in one table is related to several rows in another table. For instance, you might have a database containing data about books you own. You would have a book table and an author table. One row in the author table might contain information about the author of several books in the book table. When tables are related, you include a column in one table to hold data that matches data in the column of another table.

MySQL, the most popular Open-source SQL database management system, is developed, distributed, and supported by MySQL AB. MySQL AB is a commercial company, founded by the MySQL developers. It is a second-generation Open-Source company that unites Open MySQL software is Open Source. Open-Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), to define what you may and may not do with the software in different situations.

The MySQL Database Server is very fast, reliable, and easy to use. MySQL Server was originally developed to handle large databases and has been successfully used in highly demanding production environments for several years. MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

○ XAMPP Server

Xampp server installs a complete, ready-to-use development environment. Xampp server allows you to fit your needs and allows you to setup a local server with the same characteristics as your production.

While setting up the server and PHP on your own, you have two choices for the method of connecting PHP to the server. For many servers, PHP has a direct module interface (also called SAPI). These servers include Apache, Microsoft Internet Information Server, Netscape and iPlanet servers. Many other servers support ISAPI, the Microsoft module interface (OmniHTTPd for example). If PHP has no module support for your web server, you can always use it as a CGI or FastCGI processor. This means you set up your server to use the CGI executable of PHP to process all PHP file requests on the server.

3.4 Normalization

Normalization is the process of organizing the data in any database. It is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies. Normalization divides the larger table and links them using relationships. The normal form of a relation refers to the highest normal form condition that it meets and hence the degree to which it has been normalized.

	1NF	2NF	3NF	4NF	5NF
Decomposition of Relation	R	R ₁₁ R ₁₂	R ₂₁ R ₂₂ R ₂₃	R ₃₁ R ₃₂ R ₃₃ R ₃₄	R ₄₁ R ₄₂ R ₄₃ R ₄₄ R ₄₅
Conditions	Eliminate Repeating Groups	Eliminate Partial Functional Dependency	Eliminate Transitive Dependency	Eliminate Multi-values Dependency	Eliminate Join Dependency

Figure 3.4: Types of Normal Forms

As seen in the Figure 3.4, Normal Forms are divided into 1NF, 2NF, 3NF, BCNF, 4NF, 5NF.

- **First Normal Form**

If a relation contain composite or multi-valued attribute, it violates first normal form or a relation is in first normal form if it does not contain any composite or multi-valued attribute. A relation is in first normal form if every attribute in that relation is singled valued attribute.

- **Second Normal Form**

To be in second normal form, a relation must be in first normal form and relation must not contain any partial dependency. A relation is in 2NF if it has No Partial Dependency, i.e., no non-prime attribute (attributes which are not part of any candidate key) is dependent on any proper subset of any candidate key of the table.

- **Third Normal Form**

Third normal form is based on the concept of transitive dependency. A relation schema R is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key. A relation schema R is in 3NF if every nonprime attribute of R meets both of these following conditions:

- It is fully functionally dependent on every key of R.
- It is non-transitively dependent on every key of R.

The relations used in this database are fully functionally dependent on its key attribute and does not hold any transitive dependencies. Hence all the relations are in 3NF.

- **Boyce Codd Normal Form**

Boyce normal form is states that on any given relation, the functional dependency frame should definitely be a candidate key or a super key. A table is in BCNF if every functional dependency $X \rightarrow Y$ where X is the super key of the table. We can say it is in Boyce Codd Normal Form or BCNF since the left part of all the functional dependencies is a super key.

Hence all the relations are in BCNF. However the following database will not be in Fourth Normal Form (4NF) since there exists multivalued dependency. For a relation to be in 4NF, the dependency $X \twoheadrightarrow Y$ if for a single value of X, multiple values of Y exist, then the relation has multivalued dependency. Every single table has all the attributes referring to one primary key. This means that 4NF condition fails for the given database.

Thus we can say that the given relational schema satisfies all the normal forms up to Boyce Codd Normal Form. Since our relation satisfies many of these normal forms, we can say that the overall redundancy of the data that will be inserted into the tables will be minimized and overall there will be less occurrences of insert, delete and update anomalies taking place. The higher the degree to which it is normalized the less error prone the application's database will be. In conclusion, normalization helps the application to be organized and minimize the redundancy of data.

Chapter 4

IMPLEMENTATION

4.1 Table Creation

```
CREATE TABLE `admin` (  
  
  `id` int(11) NOT NULL,  
  
  `UserName` varchar(100) NOT NULL,  
  
  `Password` varchar(100) NOT NULL,  
  
  `updationDate` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE  
  CURRENT_TIMESTAMP)  
  
ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `admin` (`id`, `UserName`, `Password`, `updationDate`) VALUES  
  
(1, 'admin', '21232f297a57a5a743894a0e4a801fc3', '2020-06-11 12:26:07');
```

```
CREATE TABLE `tblclasses` (  
  
  `id` int(11) NOT NULL,  
  
  `ClassName` varchar(80) DEFAULT NULL,  
  
  `ClassNameNumeric` int(4) NOT NULL,  
  
  `Section` varchar(5) NOT NULL,  
  
  `CreationDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  
  `UpdationDate` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON  
  UPDATE CURRENT_TIMESTAMP
```

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```
CREATE TABLE `tblresult` (  
  `id` int(11) NOT NULL,  
  `StudentId` int(11) DEFAULT NULL,  
  `ClassId` int(11) DEFAULT NULL,  
  `SubjectId` int(11) DEFAULT NULL,  
  `marks` int(11) DEFAULT NULL,  
  `PostingDate` timestamp NULL DEFAULT CURRENT_TIMESTAMP,  
  `UpdationDate` timestamp NULL DEFAULT NULL ON UPDATE  
  CURRENT_TIMESTAMP  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `tblstudents` (  
  `StudentId` int(11) NOT NULL,  
  `StudentName` varchar(100) NOT NULL,  
  `RollId` varchar(100) NOT NULL,  
  `StudentEmail` varchar(100) NOT NULL,  
  `Gender` varchar(10) NOT NULL,  
  `DOB` varchar(100) NOT NULL,  
  `ClassId` int(11) NOT NULL,  
  `RegDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  `UpdationDate` timestamp NULL DEFAULT NULL ON UPDATE  
  CURRENT_TIMESTAMP,
```

```
`Status` int(1) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `tblsubjectcombination` (

`id` int(11) NOT NULL,

`ClassId` int(11) NOT NULL,

`SubjectId` int(11) NOT NULL,

`status` int(1) DEFAULT NULL,

`CreationDate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,

`Updationdate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
CREATE TABLE `tblsubjects` (

`id` int(11) NOT NULL,

`SubjectName` varchar(100) NOT NULL,

`SubjectCode` varchar(100) NOT NULL,

`Creationdate` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,

`UpdationDate` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE

CURRENT_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

4.2 Description of Table

- **DESC admin;**

Table structure		Relation view							
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop
2	UserName	varchar(100)	latin1_swedish_ci		No	None			Change Drop
3	Password	varchar(100)	latin1_swedish_ci		No	None			Change Drop
4	updationDate	timestamp			No	0000-00-00 00:00:00		ON UPDATE CURRENT_TIMESTAMP()	Change Drop

Figure 4.2.1: Admin Desc table

- 1. id:** It is the primary key and has the datatype int.
- 2. userName:** It has the datatype char and specifies the full name of user/admin.
- 3. password:** It has the datatype char and is used during login.
- 4. updationDate:** It specifies the date and time of account updation.

- DESC tblclasses;**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop
2	ClassName	varchar(80)	latin1_swedish_ci		Yes	NULL			Change Drop
3	ClassNameNumeric	int(4)			No	None			Change Drop
4	Section	varchar(5)	latin1_swedish_ci		No	None			Change Drop
5	CreationDate	timestamp			No	current_timestamp()			Change Drop
6	UpdationDate	timestamp			No	0000-00-00 00:00:00		ON UPDATE CURRENT_TIMESTAMP()	Change Drop

Figure 4.2.2: tblclasses Desc table

- 1. 'id':** It is a Student number (miscellaneous entry and therefore, has no significance). It is a primary key just to make sure that something can be referenced to within the table if another primary key were to not be defined.
- 2. 'ClassName':** Name of class or branch.
- 3. 'ClassNameNumeric':** it is the name of semester in numeric form.
- 4. 'Section':** section is the another class of same branch.

5. **'CreationDate'**: it is the date of create class.

6. **'UpdationDate'** : it is the date of update class.

- **DESC tblresult;**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 StudentId	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	3 ClassId	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	4 SubjectId	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	5 marks	int(11)			Yes	NULL			Change Drop More
<input type="checkbox"/>	6 PostingDate	timestamp			Yes	current_timestamp()			Change Drop More
<input type="checkbox"/>	7 UpdationDate	timestamp			Yes	NULL		ON UPDATE CURRENT_TIMESTAMP()	Change Drop More

Figure 4.2.3: tblresult Desc table

1. **'id'**: It is a student number (miscellaneous entry and therefore, has no significance). It is a primary key just to make sure that something can be referenced to within the table if another primary key were to not be defined.

2. **'Student id'** : student id is the university seat number.it is a type of varchar.

3. **'Class id'** : it is the class id number.

4. **'Subject id'** : it is specific id number of a subject.

5. **'Marks'** : total marks received in a particular subject.

6.**'postingdate'** : it is the date of declaringdating result.

7.**'Updatingdate'** : it is the date of updating result.

- **DESC tblstudents;**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	StudentId	int(11)		No	None		AUTO_INCREMENT	Change Drop
<input type="checkbox"/>	2	StudentName	varchar(100)	latin1_swedish_ci	No	None			Change Drop
<input type="checkbox"/>	3	RollId	varchar(100)	latin1_swedish_ci	No	None			Change Drop
<input type="checkbox"/>	4	StudentEmail	varchar(100)	latin1_swedish_ci	No	None			Change Drop
<input type="checkbox"/>	5	Gender	varchar(10)	latin1_swedish_ci	No	None			Change Drop
<input type="checkbox"/>	6	DOB	varchar(100)	latin1_swedish_ci	No	None			Change Drop
<input type="checkbox"/>	7	ClassId	int(11)		No	None			Change Drop
<input type="checkbox"/>	8	RegDate	timestamp		No	current_timestamp()			Change Drop
<input type="checkbox"/>	9	UpdationDate	timestamp		Yes	NULL		ON UPDATE CURRENT_TIMESTAMP()	Change Drop
<input type="checkbox"/>	10	Status	int(1)		No	None			Change Drop

Figure 4.2.4: tblstudents Desc table

1. ‘Student ID’ : it is a unique identifier assigned to a student in a school or educational institution. It is typically a numeric or alphanumeric value that is used to track and manage student information in a database management system (DBMS) project.

2.‘StudentName’ : it is also an identifier which describe the name of students.

3. ‘RollId’ : it describe the roll number of student.

4. ‘StudentEmailId’ : it is an EmailI of a particular Student.

5. ‘Gender’ : Gender Describes the Gender it char form.

6. ‘DOB’ : it is the date of birth of a particular student which tells about Age.

7. ‘Class Id’ : it is the class the class Id of a particular branch.

8. ‘Reg Date’ : Reg date is the Admition date of the Student.

9. ‘Updation Date’ : it is the date of updating student details.

10. ‘Status’ : it declares that is student existing or dropped out to the college.

- DESC Subjectcombinations;**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 ClassId	int(11)			No	None			Change Drop More
<input type="checkbox"/>	3 SubjectId	int(11)			No	None			Change Drop More
<input type="checkbox"/>	4 status	int(1)			Yes	NULL			Change Drop More
<input type="checkbox"/>	5 CreationDate	timestamp			No	current_timestamp()			Change Drop More
<input type="checkbox"/>	6 Updationdate	timestamp			No	current_timestamp()			Change Drop More

Figure 4.2.5: tblsubjectcombination Desc table

1. 'id' : Same meaning as above.
2. 'ClassId' : it is the class id number.
3. 'Subject id' : it is specific id number of a subject.
4. 'Status' : Same meaning as above.
5. 'CreationDate' : it is the date of create subject.
6. 'UpdationDate' : date of updating subject details.

- DESC tblSubjects;**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop
2	SubjectName	varchar(100)	latin1_swedish_ci		No	None			Change Drop
3	SubjectCode	varchar(100)	latin1_swedish_ci		No	None			Change Drop
4	Creationdate	timestamp			No	current_timestamp()			Change Drop
5	UpdationDate	timestamp			No	0000-00-00 00:00:00		ON UPDATE CURRENT_TIMESTAMP()	Change Drop

Figure 4.2.6: tblsubject Desc table

1. 'id' : Same meaning as above.

2. **'SubjectName'** : it is the SubjectName of a particular subject.
4. **'Subjectcode'** : it is specific id number of a subject.
5. **'CreationDate'** : it is the date of create subject.
6. **'UpdationDate'** : date of updating subject details.

4.3 Populated tables

- SELECT * FROM admin;

				id	UserName	Password	updationDate
				1	admin	827ccb0eea8a706c4c34a16891f84e7b	2023-01-19 11:34:41
	<input type="checkbox"/> Check all	With selected:					

Figure 4.3.1: Values of admin

- SELECT * FROM tblclasses;

				id	ClassName	ClassNameNumeric	Section	CreationDate	UpdationDate
				1	AIML		5 A	2023-01-01 12:50:22	2023-01-12 02:11:30
				2	ISE		5 A	2023-01-12 02:16:55	0000-00-00 00:00:00
				3	CSE		5 A	2023-01-12 02:17:20	0000-00-00 00:00:00
				4	ECE		5 A	2023-01-12 02:17:40	0000-00-00 00:00:00
				5	AIML		5 a	2023-01-23 00:33:38	2023-01-23 00:33:38
				8	ISE		5 B	2023-01-14 11:44:33	2023-01-23 00:35:58

Figure 4.3.2: Values of tblclasses

- `SELECT * FROM tblstudents;`

	StudentId	StudentName	RollId	StudentEmail	Gender	DOB	ClassId	RegDate	UpdateDate	Status
Edit Copy Delete	1	Imtiyaz	1RN20AI031	imtiyaz@gmail.com	Male	2002-10-06	1	2023-01-01 12:53:30	2023-01-14 12:02:37	1
Edit Copy Delete	2	Abhishek S	1RN20AI004	abhishek@123	Male	2003-10-10	1	2023-01-14 11:49:42	NULL	1
Edit Copy Delete	3	Shresth	1RN20AI049	vatsal@gmail.com		2001-07-05	1	2023-01-05 23:09:23	2023-01-18 23:55:22	1
Edit Copy Delete	4	Prasoon	1RN20AI039	prasoon@gmail.com	Male	2000-01-01	1	2023-01-12 02:07:01	2023-01-14 12:03:27	1
Edit Copy Delete	5	Shubham	1RN20AI012	shubham@gmail.com	Male	2001-06-04	1	2023-01-12 02:07:59	NULL	1
Edit Copy Delete	9	sameer	1RN20IS130	sameer@123	Male	2004-01-02	2	2023-01-19 00:16:12	NULL	1
Edit Copy Delete	10	Akash	1RN20IS005	akash@123	Male	2001-10-08	2	2023-01-19 09:14:12	NULL	1

Figure 4.3.3: Values of tblstudents

- `SELECT * FROM tblresult;`

	id	StudentId	ClassId	SubjectId	marks	PostingDate	UpdateDate
Edit Copy Delete	11	1	1	4	98	2023-01-13 00:21:32	NULL
Edit Copy Delete	12	1	1	3	97	2023-01-13 00:21:32	NULL
Edit Copy Delete	13	1	1	6	99	2023-01-13 00:21:32	NULL
Edit Copy Delete	14	1	1	1	95	2023-01-13 00:21:32	NULL
Edit Copy Delete	15	1	1	5	100	2023-01-13 00:21:32	NULL
Edit Copy Delete	17	4	1	4	95	2023-01-14 11:42:16	NULL
Edit Copy Delete	18	4	1	3	95	2023-01-14 11:42:16	2023-01-19 10:03:22
Edit Copy Delete	19	4	1	6	85	2023-01-14 11:42:16	NULL
Edit Copy Delete	20	4	1	1	75	2023-01-14 11:42:16	NULL
Edit Copy Delete	21	4	1	5	98	2023-01-14 11:42:16	NULL
Edit Copy Delete	22	4	1	2	88	2023-01-14 11:42:16	NULL
Edit Copy Delete	23	3	1	4	75	2023-01-14 11:43:00	NULL
Edit Copy Delete	24	3	1	3	94	2023-01-14 11:43:00	NULL
Edit Copy Delete	25	3	1	6	70	2023-01-14 11:43:00	2023-01-19 11:12:57

Figure 4.3.4: Values of tblresult

- SELECT * FROM tblsubjectcombination;

<div><div>←</div><div>T</div><div>→</div></div>				<div>▼</div>	id	ClassId	SubjectId	status	CreationDate	Updationdate
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	13	2	3	1	2023-01-19 00:12:06	2023-01-19 00:12:06	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	14	2	4	1	2023-01-19 00:12:16	2023-01-19 00:12:16	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	18	2	9	1	2023-01-19 00:26:07	2023-01-19 00:26:07	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	19	1	1	1	2023-01-19 00:26:46	2023-01-19 00:26:46	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	20	1	2	1	2023-01-19 00:27:37	2023-01-19 00:27:37	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	21	1	3	1	2023-01-19 00:27:45	2023-01-19 00:27:45	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	22	1	4	1	2023-01-19 00:27:52	2023-01-19 00:27:52	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	23	3	1	1	2023-01-19 09:11:42	2023-01-19 09:11:42	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	24	3	7	1	2023-01-19 09:11:50	2023-01-19 09:11:50	
<div><div><input type="checkbox"/></div></div>	<div><div></div></div> Edit	<div><div></div></div> Copy	<div><div></div></div> Delete	25	3	3	1	2023-01-19 09:12:01	2023-01-19 09:12:01	

Figure 4.3.5: Values of tblsubjectcombination

- SELECT * FROM tblsubjects;

<div><div></div><div></div><div></div></div>		<div></div>	id	SubjectName	SubjectCode	Creationdate	UpdationDate	
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	1	MEI	18CS51	2023-01-01 12:54:55	2023-01-12 02:36:30
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	2	Python Programming	18AI52	2023-01-12 02:12:48	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	3	DBMS	18CS53	2023-01-12 02:13:11	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	4	ATC	18CS54	2023-01-12 02:13:42	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	5	Principle of AI	18AI55	2023-01-12 02:14:15	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	6	Mathematics for ML	18AI56	2023-01-12 02:14:48	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	7	computer networking	18CS52	2023-01-19 00:21:21	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	8	Unix Programing	18CS56	2023-01-19 00:21:52	0000-00-00 00:00:00
<div><div></div><div></div><div></div></div>	<div><div></div><div>Edit</div><div></div></div>	<div><div></div><div></div><div></div></div>	<div><div></div><div></div><div></div></div>	9	development using python	18CS55	2023-01-19 00:23:17	2023-01-19 00:23:55
<div><div></div><div></div><div></div></div> <div><div></div><div>Check all</div><div>With selected:</div><div><div></div><div>Edit</div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div></div><div></div></div><div><div></div><div>Export</div></div></div>								

Figure 4.3.6: Values of tblsubjects

4.4 SQL Triggers and Stored Procedures

- **Trigger**

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Trigger used in Student Management System:

```
DELIMITER $$  
INSERT INTO logs SET  
id=OLD.id,ClassName=OLD.ClassName,  
ClassNameNumeric=OLD.ClassNameNumeric,Section=OLD.Section;  
INSERT INTO logs SET  
id=OLD.id,ClassName=OLD.ClassName,  
ClassNameNumeric=OLD.ClassNameNumeric,Section=OLD.Section;  
$$  
DELIMITER;
```

The above trigger is used to store all the data which is updated or inserted from the table. Whenever a new tuple is updated, this trigger is also updated.

- **Stored Procedure**

A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, which are stored in a relational database management system (RDBMS) as a group, so it can be reused and shared by multiple programs. Stored procedures can access or modify data in a database, but it is not tied to a specific database or object, which offers a number of advantages.

A stored procedure provides an important layer of security between the user interface and the database. It supports security through data access controls because end users may enter or change data, but do not write procedures. A stored procedure preserves data integrity because information is entered in a consistent manner. It improves productivity because statements in a stored procedure only must be written once. Stored procedures offer advantages over embedding queries in a graphical user interface (GUI). Since stored procedures are modular, it is easier to troubleshoot when

a problem arises in an application. Stored procedures are also tunable, which eliminates the need to modify the GUI source code to improve its performance. It's easier to code stored procedures than to build a query through a GUI.

4.5 Database Connectivity

A Database connection is a facility in computer science that allows client software to talk to database server software, whether on the same machine or not. A connection is required to send commands and receive answers, usually in the form of a result set.

- **JDBC:**

Java Database Connectivity (JDBC) is an application programming interface (API) for the Java programming language, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT.

4.6 Source Code (Front end)

The source code of home page of Student management System is as given below:

```
<?php

session_start();

error_reporting(0);

include('includes/config.php');

if(strlen($_SESSION['alogin'])=="")

    {

        header("Location: index.php");

    }

else{

if(isset($_POST['submit']))

{

    $marks=array();

    $class=$_POST['class'];

    $studentid=$_POST['studentid'];

    $mark=$_POST['marks'];

    $stmt = $dbh->prepare("SELECT tblsubjects.SubjectName,tblsubjects.id FROM
tblsubjectcombination join tblsubjects on tblsubjects.id=tblsubjectcombination.SubjectId
WHERE tblsubjectcombination.ClassId=:cid order by tblsubjects.SubjectName");

    $stmt->execute(array(':cid' => $class));

    $sidl=array();

    while($row=$stmt->fetch(PDO::FETCH_ASSOC))
```

```
{

array_push($sid1,$row['id']);

}

for($i=0;$i<count($mark);$i++){

    $mar=$mark[$i];

    $sid=$sid1[$i];

    $sql="INSERT          INTO          tblresult(StudentId,ClassId,SubjectId,marks)
VALUES(:studentid,:class,:sid,:marks)";

    $query = $dbh->prepare($sql);

    $query->bindParam(':studentid',$studentid,PDO::PARAM_STR);

    $query->bindParam(':class',$class,PDO::PARAM_STR);

    $query->bindParam(':sid',$sid,PDO::PARAM_STR);

    $query->bindParam(':marks',$mar,PDO::PARAM_STR);

    $query->execute();

    $lastInsertId = $dbh->lastInsertId();

    if($lastInsertId)

    {

        $msg="Result info added successfully";

    }

    else

    {
```

```
$error="Something went wrong. Please try again";

}

}

}

?>

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>SMS Admin| Add Result </title>

    <link rel="stylesheet" href="css/bootstrap.min.css" media="screen" >

    <link rel="stylesheet" href="css/font-awesome.min.css" media="screen" >

    <link rel="stylesheet" href="css/animate-css/animate.min.css" media="screen" >

    <link rel="stylesheet" href="css/lobipanel/lobipanel.min.css" media="screen" >

    <link rel="stylesheet" href="css/prism/prism.css" media="screen" >

    <link rel="stylesheet" href="css/select2/select2.min.css" >

    <link rel="stylesheet" href="css/main.css" media="screen" >

    <script src="js/modernizr/modernizr.min.js"></script>

    <script>

function getStudent(val) {

  $.ajax({
```

```
type: "POST",

url: "get_student.php",

data:'classid='+val,

success: function(data){

    $("#studentid").html(data);

}

});

$.ajax({

    type: "POST",

    url: "get_student.php",

    data:'classid1='+val,

    success: function(data){

        $("#subject").html(data);

    }

});

}

</script>

<script>

function getresult(val,clid)

{
```

```
var clid=$("#clid").val();

var val=$("#stid").val();

var abh=clid+'$'+val;

//alert(abh);

$.ajax({

    type: "POST",

    url: "get_student.php",

    data:'studclass='+abh,

    success: function(data){

        $("#reslt").html(data);

    }

});

}

</script>
```

```
</head>
```

```
<body class="top-navbar-fixed">
```

```
<div class="main-wrapper">
```

```
<!-- ===== TOP NAVBAR ===== -->
```

```
<?php include('includes/topbar.php');?>
```

```
    <!-- ===== WRAPPER FOR BOTH SIDEBARS & MAIN CONTENT
===== -->
```

```
<div class="content-wrapper">
```

```
    <div class="content-container">
```

```
        <!-- ===== LEFT SIDEBAR ===== -->
```

```
        <?php include('includes/leftbar.php');?>
```

```
        <!-- /.left-sidebar -->
```

```
    <div class="main-page">
```

```
        <div class="container-fluid">
```

```
            <div class="row page-title-div">
```

```
                <div class="col-md-6">
```

```
                    <h2 class="title">Declare Result</h2>
```

```
                </div>
```

```
            <!-- /.col-md-6 text-right -->
```

```
        </div>
```

```
    <!-- /.row -->
```

```
    <div class="row breadcrumb-div">
```

```
<div class="col-md-6">

    <ul class="breadcrumb">

        <li><a href="dashboard.php"><i class="fa fa-home"></i>
Home</a></li>

        <li class="active">Student Result</li>

    </ul>

</div>

</div>

<!-- /.row -->

</div>

<div class="container-fluid">

    <div class="row">

        <div class="col-md-12">

            <div class="panel">

                <div class="panel-body">

<?php if($msg){?>

<div class="alert alert-success left-icon-alert" role="alert">

<strong>Well done!</strong><?php echo htmlentities($msg); ?>

</div><?php }
```



```
else if($error){?>

    <div class="alert alert-danger left-icon-alert" role="alert">

                                                <strong>Oh snap!</strong> <?php echo
htmlentities($error); ?>

                                </div>

                                <?php } ?>

                                <form class="form-horizontal" method="post">

<div class="form-group">

<label for="default" class="col-sm-2 control-label">Class</label>

<div class="col-sm-10">

        <select      name="class"      class="form-control      clid"      id="classid"
onChange="getStudent(this.value);" required="required">

<option value="">Select Class</option>

<?php $sql = "SELECT * from tblclasses";

$query = $dbh->prepare($sql);

$query->execute();

$results=$query->fetchAll(PDO::FETCH_OBJ);

if($query->rowCount() > 0)

{

foreach($results as $result)

{  ?>
```

```
<option value="<?php echo htmlentities($result->id); ?>"><?php echo  
htmlentities($result->ClassName); ?>&nbsp; Section-<?php echo htmlentities($result-  
>Section); ?></option>
```

```
<?php }} ?>
```

```
</select>
```

```
</div>
```

```
</div>
```

```
<div class="form-group">
```

```
<label for="date" class="col-sm-2 control-label
```

```
">Student Name</label>
```

```
<div class="col-sm-10">
```

```
<select name="studentid" class="form-control std"
```

```
id="studentid" required="required" onChange="getresult(this.value);">
```

```
</select>
```

```
</div>
```

```
</div>
```

```
<div class="form-group">
```

```
<div class="col-sm-10">
```

```
<div id="reslt">
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<div class="form-group">
```

```
                                <label for="date" class="col-sm-2 control-  
label">Subjects</label>
```

```
                                <div class="col-sm-10">
```

```
                                <div id="subject">
```

```
                                </div>
```

```
                                </div>
```

```
                                </div>
```

```
                                <div class="form-group">
```

```
                                <div class="col-sm-offset-2 col-sm-10">
```

```
                                <button type="submit" name="submit" id="submit"  
class="btn btn-primary">Declare Result</button>
```

```
                                </div>
```

```
                                </div>
```

```
                                </form>
```

```
                                </div>
```

```
                                </div>
```

```
                                </div>
```

```
                                <!-- /.col-md-12 -->
```

```
        </div>

    </div>

</div>

<!-- /.content-container -->

</div>

<!-- /.content-wrapper -->

</div>

<!-- /.main-wrapper -->

<script src="js/jquery/jquery-2.2.4.min.js"></script>

<script src="js/bootstrap/bootstrap.min.js"></script>

<script src="js/pace/pace.min.js"></script>

<script src="js/lobipanel/lobipanel.min.js"></script>

<script src="js/iscroll/iscroll.js"></script>

<script src="js/prism/prism.js"></script>

<script src="js/select2/select2.min.js"></script>

<script src="js/main.js"></script>

<script>

    $(function($) {

        $(".js-states").select2();

        $(".js-states-limit").select2({

            maximumSelectionLength: 2

        });

        $(".js-states-hide").select2({
```

minimumResultsForSearch: Infinity

});

});

</script>

Chapter 5

RESULTS

5.1 Home page

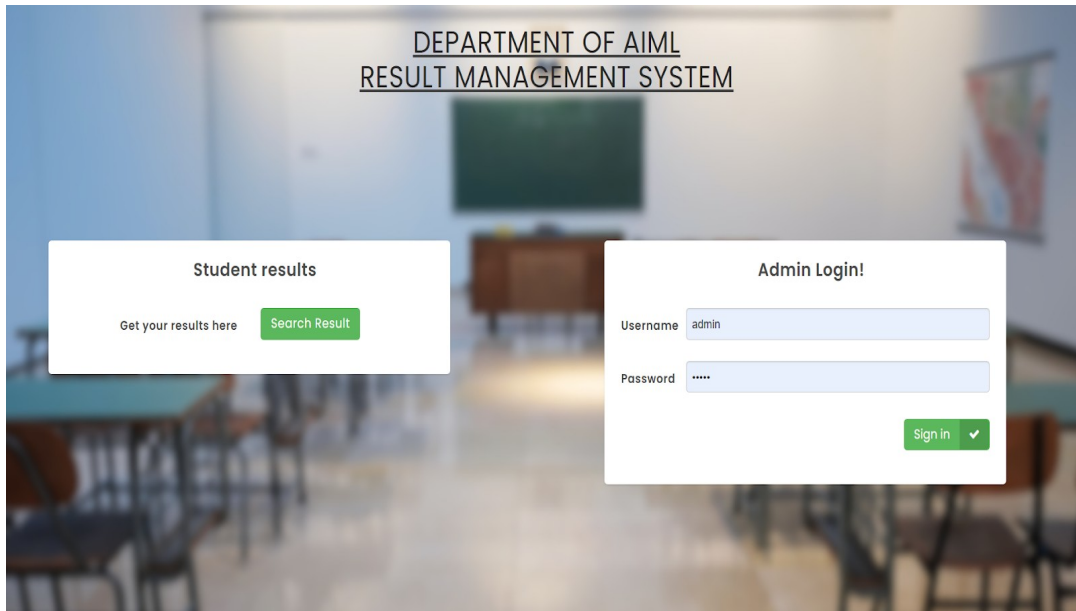


Figure 5.1: Home page of Student Result management System

Figure 5.1 represents home page of Student Management System. Admin Login and student also can check the result from here.

5.2 Admin Dashboard

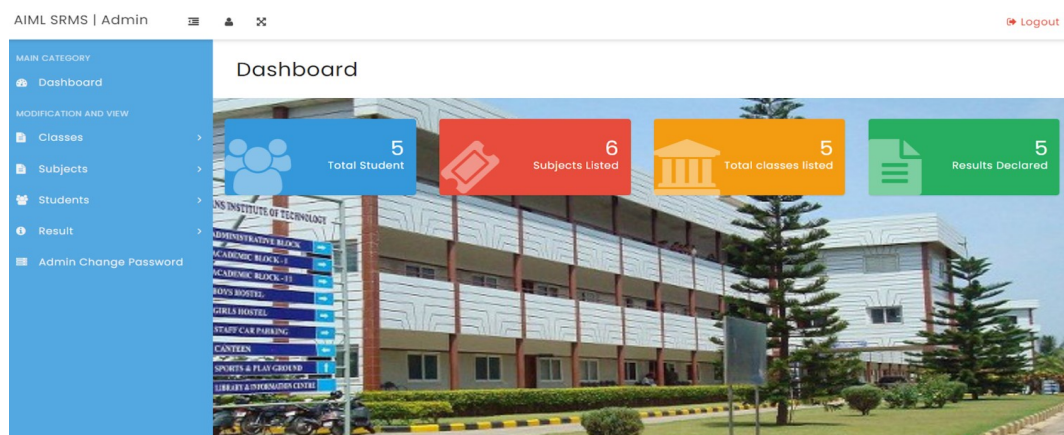


Figure 5.2: Admin Dashboard of Student management System

Figure 5.2 represents the Dashboard page where admin can filling the required details.

5.3 Classes Page

Manage Classes

Home / Classes / Manage Classes

View Classes Info

Show 10 entries Search:

#	Class Name	Semester	Section	Creation Date	Action
1	AIIML	5	A	2023-01-01 12:50:22	Edit
2	ISE	5	A	2023-01-12 02:16:55	Edit
3	CSE	5	A	2023-01-12 02:17:20	Edit
4	ECE	5	A	2023-01-12 02:17:40	Edit
5	ISE	5	A	2023-01-14 11:44:33	Edit

Showing 1 to 5 of 5 entries Previous 1 Next

Fig. 5.3 Classes Page of Student Management System

Figure 5.3 represents the class page where If a user doesn't have a account he can register himself as a customer by filling in username email and a password in the given form and can manage the classes.

5.4 Subjects Page

View Subjects Info

Show 10 entries Search:

#	Subject Name	Subject Code	Creation Date	Updation Date	Action
1	MEI	18CS51	2023-01-01 12:54:55	2023-01-12 02:36:30	Edit
2	Python Programming	18AI52	2023-01-12 02:12:48	0000-00-00 00:00:00	Edit
3	DBMS	18CS53	2023-01-12 02:13:11	0000-00-00 00:00:00	Edit
4	ATC	18CS54	2023-01-12 02:13:42	0000-00-00 00:00:00	Edit
5	Principle of AI	18AI55	2023-01-12 02:14:15	0000-00-00 00:00:00	Edit
6	Mathematics for ML	18AI56	2023-01-12 02:14:48	0000-00-00 00:00:00	Edit

Showing 1 to 6 of 6 entries Previous 1 Next

Fig. 5.4 Subjects Page of Student Management System

The Subjects page contains the details of all the subjects of a particular class.

5.5 Students Page

Home / Students / Manage Students

View Students Info

Show entries

Search:

#	Student Name	USN	Class	Reg Date	Status	Action
1	Imtiyaz	1RN20AI031	AIML(A)	2023-01-01 12:53:30	Active	✎
2	Abhishek	1RN20AI004	AIML(A)	2023-01-14 11:49:42	Active	✎
3	Shresth vatsal	1RN20AI049	AIML(A)	2023-01-05 23:09:23	Active	✎
4	Prasoon	1RN20AI039	AIML(A)	2023-01-12 02:07:01	Active	✎
5	Shubham	1RN20AI012	AIML(A)	2023-01-12 02:07:59	Active	✎

Showing 1 to 5 of 5 entries

Previous **1** Next

Fig 5.5 Student Page of Student Result Management System

Figure 5.5 represents student page where the details of students described.

5.6: Admin Landing Page (Post Login)

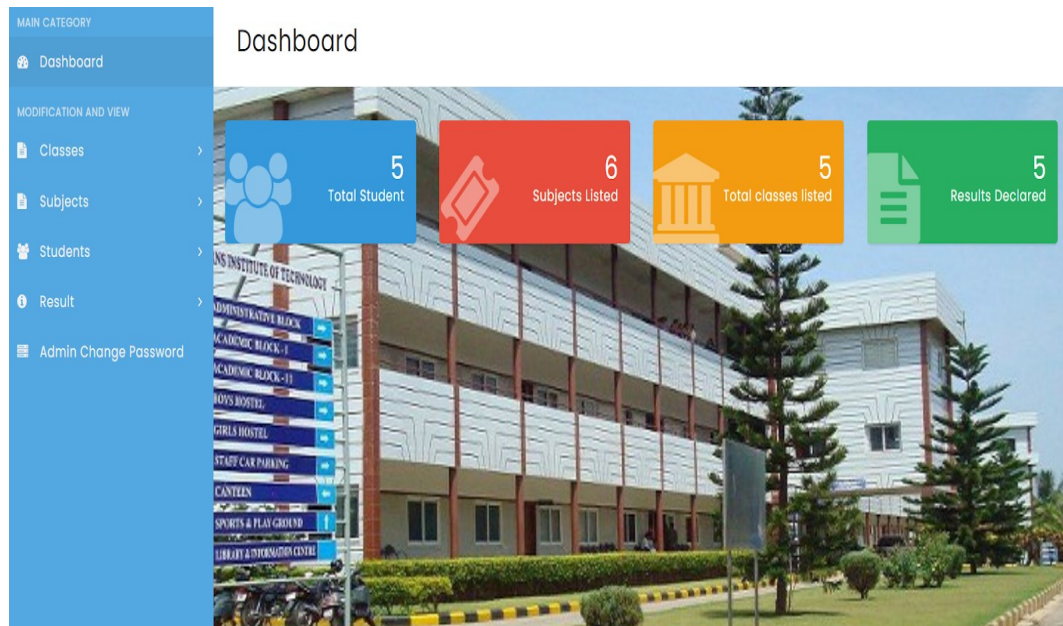


Fig 5.6 Admin page after login of student management System

Figure 5.6 represents Admin login where admin can see and perform a particular work.

5.7: Create Student class page

Fig 5.7 create student class page of student management system

Figure 5.7 represents the create student class page where admin can add details of a new students.

5.8: subject creation page (INSERT)

Fig 5.8 subject creation page of student management system

Figure 5.8 represents subject creation date where admin can add a new subject.

5.9: Result page

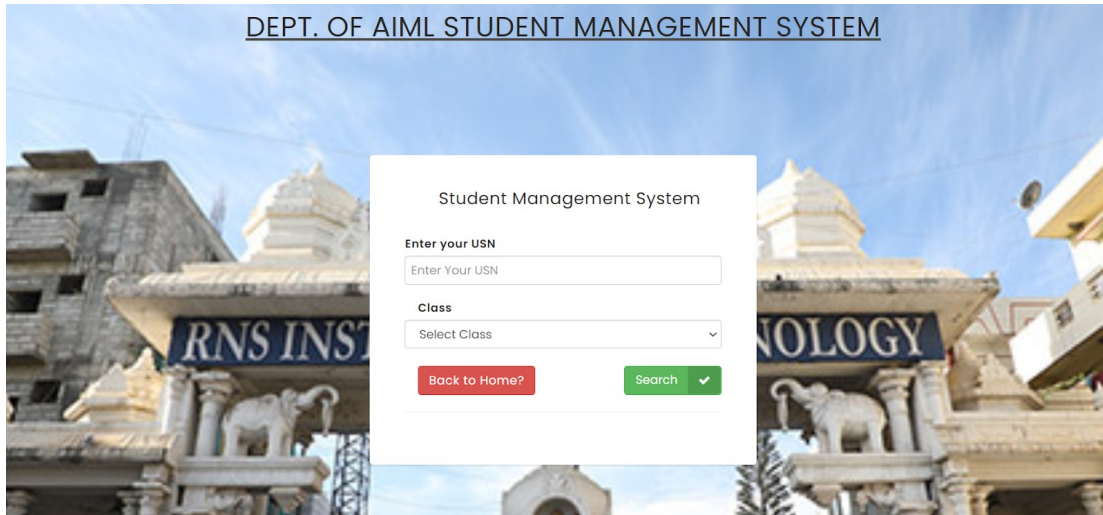


Fig 5.9 Result page of Student management system

Figure 5.9 represents the result page where student can see the result after entering USN and class name of the student.

5.10: Student Result

Student Name : Prasoon		
Student USN : 1RN20AI039		
Student Class: AIML(A)		
#	Subject	Marks
1	ATC	95
2	DBMS	95
3	Mathematics for ML	85
4	MEI	75
5	Principle of AI	98
6	Python Programming	88
Total Marks		536 out of 600
Percentage		89.333333333333 % passed

Fig 5.10 Result of a particular student

Figure 5.10 represents the markscard of a student where marks of all the subject written.

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

6.1 Conclusion

Student result management system lead to a better organization structure since the information management of the students is well structured and also lead to better as well as efficient utilization of resources. Student result management system can be used by education institutes to maintain the results of students easily. Achieving this objective is difficult using a manual system as the information is scattered, can be redundant and collecting relevant information may be very time consuming.

6.2 Future Enhancements

There are several ways to enhance a student management system. Here are a few examples:

- Implementing a search feature: This could allow users to search for students by name, ID, or other criteria, making it easier to find specific students within a large database.
- Adding a grades management feature: This could allow teachers to input and track student grades, and allow students and parents to view their grades online.
- Incorporating attendance tracking: This could allow teachers to take attendance and automatically calculate attendance percentage for each student.
- Adding a messaging system: This could allow teachers, students, and parents to communicate with each other directly within the system, rather than having to use external methods of communication.

- Implementing an online payment system: This could allow students and parents to pay school fees, tuition and other charges online.
- Incorporating a mobile app version: This could allow users to access the system from mobile devices, making it more convenient for those who are always on the go.
- Security and data protection: Implementing measures such as encryption, secure login and access controls to ensure the safety of sensitive student information.

Chapter 7

REFERENCES

- [1] Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems, PEARSON, Fifth Edition.
- [2] Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems, McGRAW HILL, Third Edition.
- [3] <https://www.w3schools.com/html/>
- [4] <https://www.w3schools.com/css/>
- [5] <https://www.geeksforgeeks.org>
- [6] <https://www.github.com>
- [7] <https://www.php.net/>
- [8] www.stackoverflow.com