VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Santhibastawad Road, Machhe

Belagavi - 590018, Karnataka, India

DBMS LABORATORY WITH MINI PROJECT (18CSL58) REPORT

ON

"Student Marks and Activity Database Management System"

Submitted in the partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

For the Academic Year 2022-2023

Submitted by

Al Aqmar Damana **1JS20IS011**

Under the Guidance of

Dr. Sowmya KN

Associate Professor, Dept. of ISE, JSSATEB

2022-2023

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060

JSS MAHAVIDYAPEETHA, MYSURU

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that DBMS LABORATORY WITH MINI PROJECT (18CSL58) Report entitled **"Student Marks and Activity Database Management System"** is a Bonafede work carried out by **Al Aqmar Damana [1JS20IS011]** in partial fulfillment for the award of degree of Bachelor of Engineering in Information Science and Engineering of Visvesvaraya Technological University Belagavi during the year 2022- 2023.

Signature of the Guide	**Signature of the HOD**
Dr. Sowmya K N	**Dr.Rekha P M**
Associate Professor,	**Professor & HOD,**
Dept. of ISE,	**Dept. of ISE,**
JSSATEB	**JSSATEB**

TABLE OF CONTENTS

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible. So with gratitude, we acknowledge all those whose guidance and encouragement crowned our efforts with success.

First and foremost, we would like to thank his **Holiness Jagadguru Sri Shivarathri Deshikendra Mahaswamiji** and **Dr. Bhimasen Soragaon**, Principal, JSSATE, Bangalore for providing an opportunity to carry out the Project Work as a part of our curriculum in the partial fulfilment of the degree course.

We express our sincere gratitude for our beloved Head of the department, **Dr. Rekha P. M**, for her co-operation and encouragement at all the moments of our approach.

It is our pleasant duty to place on record our deepest sense of gratitude to our respected guide **Dr. Sowmya K N**, Associate Professor for the constant encouragement, valuable help and assistance in every possible way.

We would like to thank all ISE Department Teachers, **Mrs. Sowmya G** ma'am our laboratory instructor and non-teaching staff for providing us with their valuable guidance and for being there at all stages of our work.

Al Aqmar Damana [1JS20IS011]

ABSTRACT

This abstract describes a proposed database management system for tracking student marks and activities in an educational system. The system would allow for the storage and organization of student information, including demographics, grades, and participation in activities. The proposed system would improve efficiency and accuracy in the management of student records, and would provide valuable insights into student performance and engagement. Overall, this database management system would serve as a valuable tool for educators, administrators, and researchers in the field of education.

The proposed student marks and activity database management system would be a comprehensive solution for tracking student information in an educational system. Additionally, the system would provide a variety of reporting and analysis tools, allowing educators to generate customized reports and analyze student performance data.

One of the key features of the system would be its ability to track student performance over time. This would provide educators with valuable insights into student progress, and would allow them to identify areas where students are excelling or struggling. The system would also allow educators to compare student performance across different groups, such as by grade level or gender.

The system would also be designed to be user-friendly and easy to navigate. This would make it accessible to educators of all levels of technical expertise, and would ensure that the system is widely adopted and used effectively. Additionally, the system would be designed to be secure, with robust data protection and privacy controls in place to ensure that student data is kept confidential.

In summary, the proposed student marks and activity database management system would provide a comprehensive, user-friendly, and secure solution for tracking student information in an educational setting. It would improve efficiency and accuracy in the management of student records, and would provide valuable insights into student performance and engagement. This database management system would serve as a valuable tool for educators, administrators, and researchers in the field of education.

CHAPTER1
INTRODUCTION

INTRODUCTION TO FILE STRUCTURES

A File Structure is a combination of representations for data in files and of operations for accessing the data. A File Structure allows applications to read, write and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order.

Why Study File Structures?

1. Data Storage

Computer Data can be stored in three kinds of locations

- Primary Storage ==> Memory [Computer Memory]
- Secondary Storage [Online Disk/ Tape/ CDROM that can be accessed by the computer]
- Tertiary Storage ==> Archival Data [Offline Disk/Tape/ CDROM not directly available to the computer.]

Out of all these, Primary and Secondary storage have been out main focus.

2. Memory Versus Secondary Storage

- Secondary storage such as disks can pack thousands of megabytes in a small physical location.
- Computer Memory (RAM) is limited.
- However, relative to Memory, access to secondary storage is extremely slow
- [E.g., getting information from slow RAM takes 120 X 10^{-9} seconds(= 120 nanoseconds) while getting information from Disk takes 30 X 10^{-3} seconds (= 30 milliseconds)]

3. How Can Secondary Storage Access Time be Improved?

By improving the File Structure.

Since the details of the representation of the data and the implementation of the operations determine the efficiency of the file structure for particular applications, improving these details can help improve secondary storage access time.

HISTORY OF DBMS

I. Early Work

Early Work assumed that files were on tape. Access was sequential and the cost of access grew in direct proportion to the size of the file.

II. The emergence of Disks and Indexes

As files grew very large, unaided sequential access was not a good solution. Disks allowed for direct access. Indexes made it possible to keep a list of keys and pointers in a small file that could be searched very quickly. With the key and pointer, the user had direct access to the large, primary file.

III. The emergence of Tree Structures

As indexes also have a sequential flavor, when they grew too much, they also became difficult to manage. The idea of using tree structures to manage the index emerged in the early 60's. However, trees can grow very unevenly as records are added and deleted, resulting in long searches requiring many disk accesses to find a record.

IV. Balanced Trees

In 1963, researchers came up with the idea of AVL trees for data in memory. AVL trees, however, did not apply to files because they work well when tree nodes are composed of

single records rather than dozens or hundreds of them. In the 1970's came the idea of B-Trees which require an O (logk N) access time where N is the number of entries in the file and k, the number of entries indexed in a single block of the B-Tree structure --> B-Trees can guarantee that one can find one file entry among millions of others with only 3 or 4 trips to the disk.

V. Hash Tables

Retrieving entries in 3 or 4 accesses is good, but it does not reach the goal of accessing data with a single request. From early on, Hashing was a good way to reach this goal with files that do not change size greatly over time. Recently, Extendible Dynamic Hashing guarantees one or at most two disk accesses no matter how big a file becomes.

OBJECTIVES

The Objectives of **STUDENT MARKS AND ACTIVITY DATABASE MANAGEMENT SYSTEM** are:

1. The main objective of our project is to provide an easy interface for the faculty of any college to *integrate* and *organise* all the data related to students activities conducted in college.
2. This project also aims at *providing access* to students data across various departments in the university.
3. It also aids in the *preservation of students data* for years to come. In case any faculty wants to *retrieve information* of previously enrolled students, this mini-project will provide an appropriate solution.
4. This project will provide major assistance to inter-departmental data sharing, because it provides a *single platform*.

ORGANIZATION OF THE REPORT

Chapter 1 provides the information about the basics of file structures, the history og dbms and the objectives. In **chapter 2** we discuss about the literature survey and the normalization. In **Chapter 3**, we discuss the software and hardware requirements to run the above applications. **Chapter 4** gives the idea of the system design. **Chapter 5** gives a clear picture about the project and its actual implementation. In **Chapter 6** we discuss about the system testing. **Chapter 7** discusses about the results and discussions of the program. **Chapter 8** concludes by giving the direction for future enhancement and the **Chapter 9** includes the references.

CHAPTER 2

LITERATURE SURVEY

INTRODUCTION

A database management system (DBMS) refers to the technology for creating and managing databases. Basically, DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient. By data, we mean known facts that can be recorded and that have embedded meaning. Normally people use software such as DBASE IV or V, Microsoft ACCESS, or EXCEL to store data in the form of database. A datum is a unit of data. Meaningful data combined to form information. Hence, information is interpreted data – data provided with semantics. MS. ACCESS is one of the most common examples of database management software. The name indicates what database is. Database is one of the important components for many applications and is used for storing a series of data in a single set. In other words, it is a group / package of information that is put in order so that it can be easily access, manage and update.

SQLlite

SQLite is an in-process library that implements a self-contained, server-less, zero-configuration, transactional SQL database engine. The source code for SQLite is in the public domain.

SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine. SQLite is the most used database engine in the world. SQLite is built into all mobile phones and most computers and comes bundled inside countless other applications that people use every day.

The SQLite file format is stable, cross-platform, and backwards compatible and the developers pledge to keep it that way through the year 2050. SQLite database files are commonly used as containers to transfer rich content between systems.

Flask

Flask is a web framework, it's a Python module that lets you develop web applications easily. It's has a small and easy-to-extend core: it's a microframework that doesn't include an ORM (Object Relational Manager) or such features.

It does have many cool features like url routing, template engine. It is a WSGI web app framework. Flask has become popular among Python enthusiasts. As of October 2020, it has second most stars on Github among Python web-development frameworks, only slightly behind Django, and was voted the most popular web framework in the Python Developers Survey 2018, 2019, 2020 and 2021.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

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

HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into 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 such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <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 embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The

World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Bootstrap is an HTML, CSS and JS library that focuses on simplifying the development of informative web pages (as opposed to web applications). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light- and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components which do not require other libraries like jQuery. They provide additional user interface elements such as dialog boxes, tooltips, progress bars, navigation drop-downs, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code.

Windows 11

Windows 11 is the latest major release of Microsoft's Windows NT operating system, released in October 2021. It is a free upgrade to its predecessor, Windows 10 (2015), and is available for any Windows 10 devices that meet the new Windows 11 system requirements.

Windows 11 features major changes to the Windows shell influenced by the canceled Windows 10X, including a redesigned Start menu, the replacement of its "live tiles" with a separate "Widgets" panel on the taskbar, the ability to create tiled sets of windows that can be minimized and restored from the taskbar as a group, and new gaming technologies inherited from Xbox Series X and Series S such as Auto HDR and DirectStorage on compatible hardware. Internet Explorer (IE) has been replaced by the Chromium-based Microsoft Edge as the default web browser, like its predecessor, Windows 10, and Microsoft Teams is integrated into the Windows shell. Microsoft also announced plans to allow more flexibility in software that can be distributed via the Microsoft Store and to support Android apps on Windows 11 (including a partnership with Amazon to make its app store available for the function).

Mac OS X

Mac OS X was originally presented as the tenth major version of Apple's operating system for Macintosh computers; until 2020, versions of macOS retained the major version number "10". The letter "X" in Mac OS X's name refers to the number 10, a Roman numeral, and Apple has stated that it should be pronounced "ten" in this context. However, it is also commonly pronounced like the letter "X". Previous Macintosh operating systems (versions of the classic Mac OS) were named using Arabic numerals, as with Mac OS 8 and Mac OS 9.

NORMALIZATION

Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data. Normalization split a large table into smaller tables and define relationships between them to increases the clarity in organizing data.

Database normalization types

First Normal Form (1NF)

• First normal form (1NF) deals with the `shape' of the record type

• A relation is in 1NF if, and only if, it contains no repeating attributes or groups of attributes.

• Example: The Student table with the repeating group is not in 1NF

Second Normal Form (2NF)

• A relation is in 2NF if, and only if, it is in 1NF and every non-key attribute is fully functionally dependent on the whole key.

Third Normal Form (3NF)

• A relation is in 3NF if, and only if, it is in 2NF and there are no transitive functional dependencies.

• Transitive functional dependencies arise:

• when one non-key attribute is functionally dependent on another non-key attribute

• FD: non-key attribute -> non-key attribute

• and when there is redundancy in the database

Boyce-Codd Normal Form (BCNF)

• When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.

• 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys

• i.e. composite candidate keys with at least one attribute in common.

• BCNF is based on the concept of a determinant.

Fourth Normal Form (4NF)

• It is a normal form used in database normalization Introduced by Ronald Fagin in 1977, 4NF is the next level of normalization after Boyce–Codd normal form (BCNF). Whereas the second, third, and Boyce–Codd normal forms are concerned with functional dependencies, 4NF is concerned with a more general type of dependency known as a multivalued dependencies

Fifth Normal Form (5NF)

• It is also known as project-join normal form (PJ/NF) is a level of database normalization designed to reduce redundancy in relational databases recording multi-valued facts by isolating semantically related multiple relationships. A table is said to be in the 5NF if and only if every nontrivial join dependency in that table is implied by the candidate keys.

CHAPTER 3

REQUIREMENT SPECIFICATIONS

SOFTWARE SPECFICATION

- Project Type: Web-Based Application
- Front-end Tech: HTML, CSS, BOOTSTRAP
- Database Tool: SQLlite
- Back-end Tech: Flask
- OS: Windows 11 and above, Linux and Mac compatible.
- Browser: Internet explorer, Chrome, Firefox, or Safari

HARDWARE SPECFICATION

- Processor: x86 compatible processor with 1.7 GHz Clock Speed
- RAM: 512 MB or greater
- Hard Disk: 20 GB or grater
- Monitor: VGA/SVGA
- Keyboard: 104 keys standard
- Mouse: 2/3 button. Optical/Mechanical.

USER CHARACTERISTICS

Every user:

- Should be aware of the marks distribution criteria.
- Should be aware of the USN, Subject Code formats.
- Should be comfortable with basic working of the computer.
- Must have basic knowledge of English.

CHAPTER 4

SYSTEM DESIGN

INTRODUCTION TO SYSTEM DESIGN

System is a collection of an interrelated components that works together to achieve a purpose. System analysis is referred to the systematic examination or detailed study of a system in order to identify problems of the system, and using the information gathered in the analysis stage to recommend improvements or solution to the system.

System design is an abstract representation of a system component and their relationship and which describe the aggregated functionality and performance of the system. System design is also the overall plan or blueprint for how to obtain answer to the question being asked. The design specifies which of the various type of approach.

Systems design is the process or art of defining the architecture, components modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. Properly designed database are easy to maintain, improves data consistency and are cost effective in terms of disk storage space. The database designer decides how the data elements correlate and what data must be stored.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

ATTRIBUTES

Attributes define the properties of a data object and take on one of three different characteristics. They can be used to:

- Name an instance of data object.
- Describe the instance.

The following figure shows the Relational model for Student Marks and Activity database management.

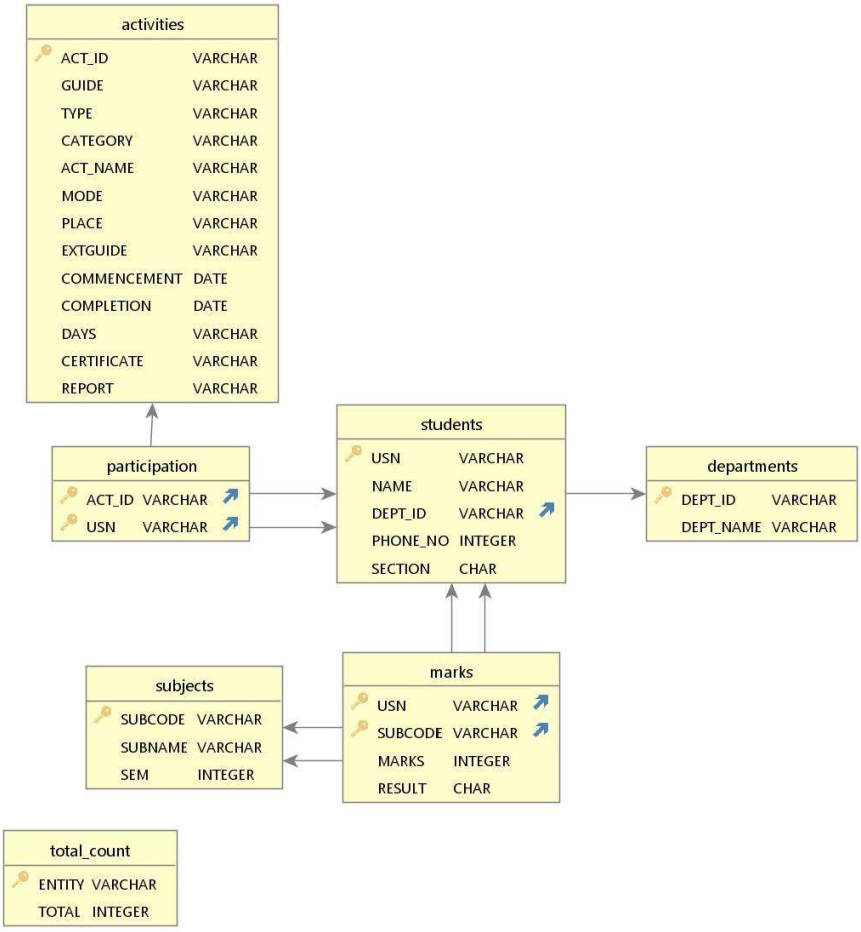


Fig1:- Relational model for Student Marks and Activity database management.

Relational Model was proposed by E.F. Codd to model data in the form of relations or tables. After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDBMS languages like SQLlite, MySQL etc. So we will see what Relational Model is. Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).

SCHEMA DIAGRAM

A **database schema** is the skeleton structure that represents the logical view of the entire database. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams.

A **schema diagram** contains entities and the attributes that will define that **schema**. It only shows us the database design. It does not show the actual data of the database. **Schema** can be a single table or it can have more than one table which is related.

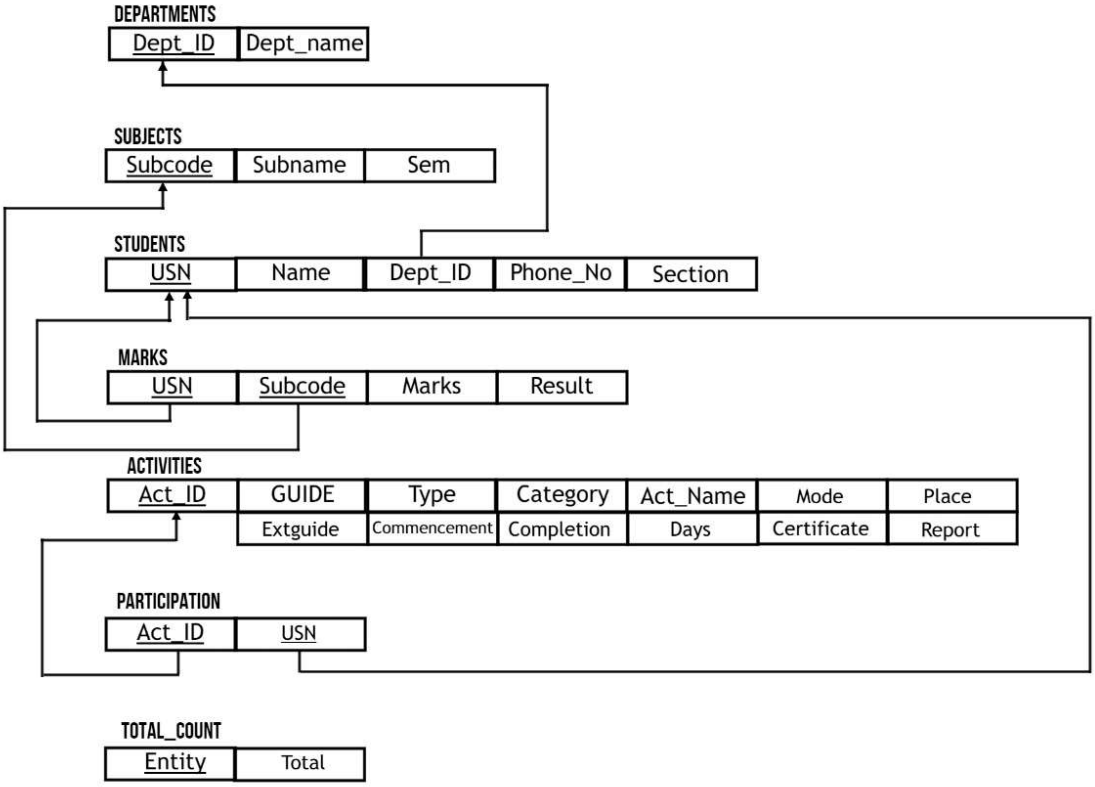


Fig2:- Schema Diagram for Student Marks and Activity database management

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. These integrity constraints ensure compatibility between parts of the schema. All

constraints are expressible in the same language. A database can be considered a structure in realization of the database language.

The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real-world entities are modeled in the database. All the various table used are described in the following schema. The necessary Primary key's and the corresponding Foreign keys are also represented.

ER DIAGRAM

An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**. An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

The ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.

Here are the geometric shapes and their meaning in an E-R Diagram.

Rectangle:Represents Entity sets.

Ellipses: Attributes

Diamonds: Relationship Set

Lines: They link attributes to Entity Sets and Entity sets to Relationship Set

Double Ellipses: Multivalued Attributes

Dashed Ellipses: Derived Attributes

Double Rectangles: Weak Entity Sets

Double Lines: Total participation of an entity in a relationship set

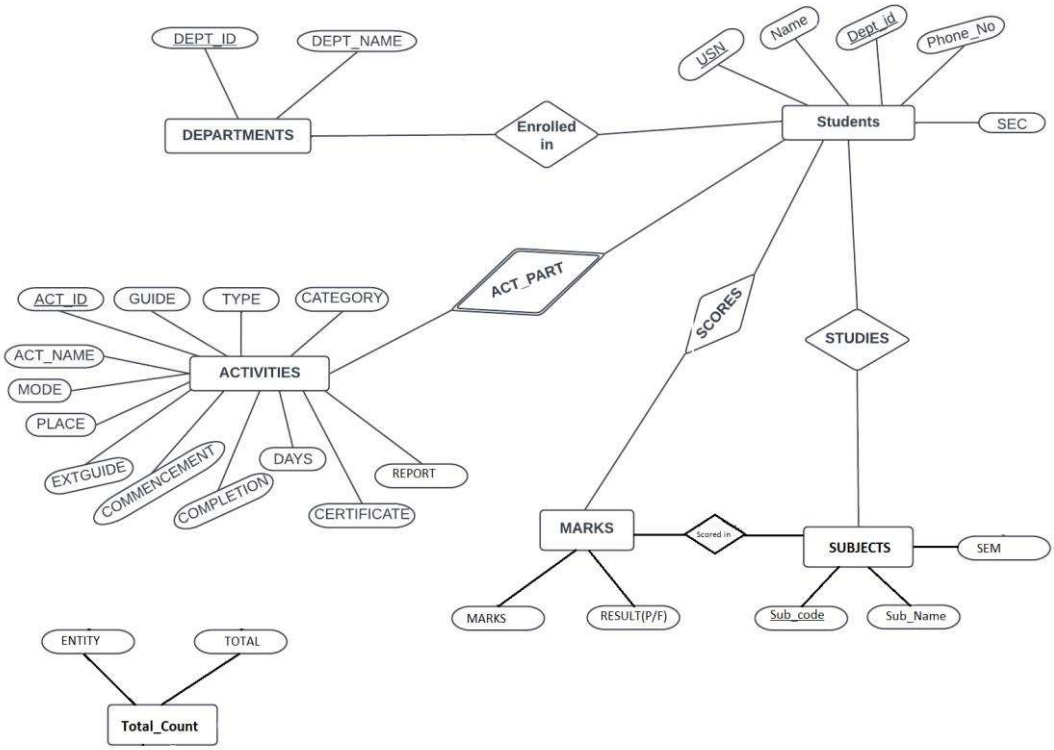


Fig3:- ER Diagram for Student Marks and Activity database management

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

WIREFRAME

A wireframe is a two-dimensional illustration of a page's interface that specifically focuses on space allocation and prioritization of content, functionalities available, and intended behaviours. For these reasons, wireframes typically do not include any styling, colour, or graphics. Wireframes also help establish relationships between a website's various templates.

Whenever the project is launched in the local host the homepage will be rendered in the web which will allow the user to login/sign up, upon login the student details can be entered through the different buttons available in the home page where the particular button will render the different forms which allow the user to interact with the database as shown in the figure below.

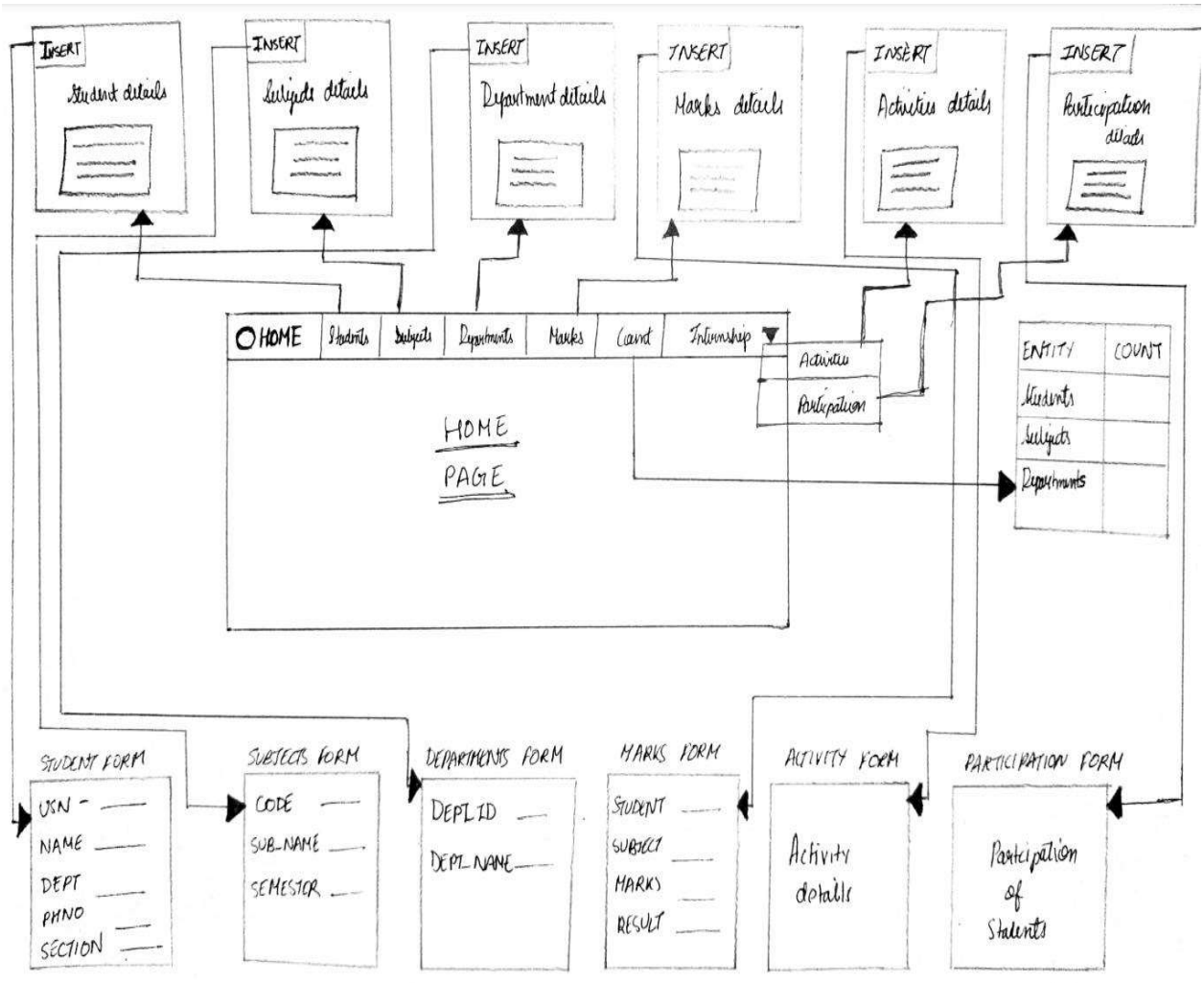


Fig4:- Wireframe for the Student marks and activity database management system

CHAPTER 5

PROJECT IMPLEMENTATION

INTRODUCTION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigating of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The following codes will ensure the complete implementation of our design and the project.

CREATING TABLES

TABLE STUDENTS

```
CREATE TABLE STUDENTS (

        USN VARCHAR(15) PRIMARY KEY,
        NAME VARCHAR(20),
        DEPT_ID VARCHAR(10),
        PHONE_NO INTEGER(10),
        SECTION CHAR(10)
);
```

TABLE DEPARTMENTS

```
CREATE TABLE DEPARTMENTS
(
        DEPT_ID VARCHAR(10) PRIMARY KEY,
        DEPT_NAME VARCHAR(10)
);
```

TABLE SUBJECTS

```
CREATE TABLE SUBJECTS
(
        SUBCODE VARCHAR(10) PRIMARY KEY,
        SUBNAME VARCHAR(25),
        SEM INTEGER(1)
);
```

TABLE MARKS

```
CREATE TABLE MARKS
(
        USN VARCHAR(15),
        SUBCODE VARCHAR(10),
        MARKS INTEGER(3),
        RESULT CHAR(6),
        FOREIGN KEY (USN) REFERENCES STUDENTS(USN),
        FOREIGN KEY (SUBCODE) REFERENCES SUBJECTS(SUBCODE),
        PRIMARY KEY(USN,SUBCODE)
);
```

TABLE ACTIVITIES

```
CREATE TABLE ACTIVITIES
(
        ACT_ID VARCHAR(10) PRIMARY KEY,
        GUIDE VARCHAR(10),
        TYPE VARCHAR(10),
        CATEGORY VARCHAR(10),
        ACT_NAME VARCHAR(10),
        MODE VARCHAR(10),
        PLACE VARCHAR(10),
        EXT_GUIDE VARCHAR(10),
        COMMENCEMENT DATE,
        COMPLETION DATE,
        DAYS INTEGER(2),
        CERTIFICATE VARCHAR(10),
        REPORT VARCHAR(10)
);
```

TABLE PARTICIPATION

```
CREATE TABLE PARTICIPATION
(
        ACT_ID VARCHAR(10),
        USN VARCHAR(10),
        FOREIGN KEY (ACT_ID) REFERENCES AICTE_ACT(ACT_ID),
        FOREIGN KEY (USN) REFERENCES STUDENTS(USN)
);
```

CREATION OF TABLES WITH DICTIONARIES

```python
import sqlite3

class DBObject:
    tables = ["students", "subjects", "departments", "marks", "activities", "participation", "total_count"]
    schemas = {"departments" : "departments(DEPT_ID VARCHAR(10) PRIMARY KEY,DEPT_NAME VARCHAR(10))",
    "students":"students(USN VARCHAR(15) PRIMARY KEY, NAME VARCHAR(20), DEPT_ID VARCHAR(10) REFERENCES departments(DEPT_ID), PHONE_NO INTEGER(10), SECTION CHAR(10))",
    "subjects" : "subjects(SUBCODE VARCHAR(10) PRIMARY KEY,SUBNAME VARCHAR(25),SEM INTEGER(1))",
    "marks" : "marks(USN VARCHAR(15),SUBCODE VARCHAR(10),MARKS INTEGER(3),RESULT CHAR(6),FOREIGN KEY (USN) REFERENCES STUDENTS(USN),FOREIGN KEY (SUBCODE) REFERENCES
    SUBJECTS(SUBCODE),PRIMARY KEY(USN,SUBCODE))",
    "activities" : "activities(ACT_ID VARCHAR(10) PRIMARY KEY, GUIDE VARCHAR(100), TYPE VARCHAR(100), CATEGORY VARCHAR(100), ACT_NAME VARCHAR(100), MODE VARCHAR(100),
    PLACE VARCHAR(100), EXTGUIDE VARCHAR(100), COMMENCEMENT VARCHAR(100), COMPLETION VARCHAR(100), DAYS VARCHAR(100), CERTIFICATE VARCHAR(100), REPORT VARCHAR(100))",
    "participation" : "participation(ACT_ID VARCHAR(10),USN VARCHAR(10),FOREIGN KEY (ACT_ID) REFERENCES activities(ACT_ID),FOREIGN KEY (USN) REFERENCES STUDENTS(USN),
    PRIMARY KEY(ACT_ID, USN))",
    "total_count" : "total_count(ENTITY VARCHAR(20) PRIMARY KEY, TOTAL INTEGER)"}

    table_headings = {"students" : ["USN", "NAME", "DEPT_ID", "PHONE NO.", "SEC"],
    "subjects": ["SUB_CODE","SUB_NAME","SEM"],
    "departments": ["DEPT_ID","DEPT_NAME"],
    "marks": ["USN","SUB_CODE","MARKS","RESULT"],
    "activities":["ActID", "Guide", "Type", "Category", "ActName", "Mode", "Place", "ExtGuide", "Commencement","Completion", "Days", "Certification", "Report"],
    "participation": ["ACT_ID","USN"],
    "total_count": ["ENTITY", "COUNT"]}

    rows_for_total = ['Students', 'Subjects', 'Departments']

    #for creation of tables
    def __init__(self, fname):
        self.conn = sqlite3.connect(fname, check_same_thread=False)
        self.curr = self.conn.cursor()
        self.existing_tables = set(tname[0] for tname in self.curr.execute("SELECT name FROM sqlite_master WHERE TYPE='table'"))
        for table in self.tables:
            if table not in self.existing_tables:
                self.curr.execute(f"CREATE TABLE {self.schemas[table]}")
        self.conn.commit()
        for row in self.rows_for_total:
            self.curr.execute(f"INSERT OR IGNORE INTO total_count VALUES('{row}', 0)")
        self.conn.commit()
        self.make_triggers()
        self.conn.commit()
```

Code Snippet for creation of tables using dictionaries

QUERIES

TABLE STUDENTS

```
def insert_students(self, obj):
    if obj["usn"] == "":
        return False
    try:
        self.curr.execute(f"INSERT INTO students VALUES ('{obj['usn']}', '{obj['name']}', '{obj['dept_id']}', '{obj['ph']}', '{obj['sec']}')")
        self.conn.commit()
        return True
    except:
        return False

def get_students(self):
    return self.curr.execute("SELECT * FROM students").fetchall()

def del_students(self, usn):
    try:
        self.curr.execute(f"DELETE FROM students WHERE usn='{usn}'")
        # delete marks and participation records
        self.del_marks_by_usn(usn)
        self.del_participation_by_usn(usn)
        self.conn.commit()
        return True
    except:
        return False
```

Code Snippet for insertion, deletion and retrieval for students table

TABLE SUBJECTS

```
    def insert_subjects(self, obj):
        if obj["code"] == "":
            return False
        try:
            self.curr.execute(f"INSERT INTO subjects VALUES ('{obj['code']}', '{obj['name']}', '{obj['sem']}')")
            self.conn.commit()
            return True
        except:
            return False

    def get_subjects(self):
        return self.curr.execute("SELECT * FROM subjects").fetchall()

    def del_subjects(self, code):
        try:
            self.curr.execute(f"DELETE FROM subjects WHERE subcode='{code}'")
            # delete marks records
            self.del_marks_by_subject(code)
            self.conn.commit()
            return True
        except:
            return False
```

Code Snippet for insertion, deletion and retrieval for subjects table

TABLE DEPARTMENTS

```python
def insert_departments(self, obj):
    if obj["id"] == "":
        return False
    try:
        self.curr.execute(f"INSERT INTO departments VALUES ('{obj['id']}', '{obj['name']}')")
        self.conn.commit()
        return True
    except:
        return False

def get_departments(self):
    return self.curr.execute("SELECT * FROM departments").fetchall()

def del_departments(self, id):
    try:
        self.curr.execute(f"DELETE FROM departments WHERE dept_id='{id}'")
        # delete students under this department
        usns = self.curr.execute(f"SELECT usn FROM students WHERE dept_id='{id}'").fetchall()
        for usn in usns:
            self.del_students(usn[0])
        self.conn.commit()
        return True
    except:
        return False
```

Code Snippet for insertion, deletion and retrieval for departments table

TABLE MARKS

```python
def insert_marks(self, obj):
    try:
        print(obj)
        self.curr.execute(f"INSERT INTO marks VALUES ('{obj['usn']}', '{obj['code']}', '{obj['marks']}', '{obj['result']}')")
        self.conn.commit()
        return True
    except:
        return False

def get_marks(self):
    return self.curr.execute("SELECT * FROM marks").fetchall()

def del_marks(self, str):
    try:
        usn, subcode = str.split()
        self.curr.execute(f"DELETE FROM marks WHERE usn='{usn}' AND subcode='{subcode}'")
        self.conn.commit()
        return True
    except:
        return False

def del_marks_by_usn(self, usn):
    try:
        self.curr.execute(f"DELETE FROM marks WHERE usn='{usn}'")
        self.conn.commit()
        return True
    except:
        return False

def del_marks_by_subject(self, subcode):
    try:
        self.curr.execute(f"DELETE FROM marks WHERE subcode='{subcode}'")
        self.conn.commit()
        return True
    except:
        return False
```

Code Snippet for insertion, deletion and retrieval for marks table

TABLE PARTICIPATION

```
def insert_participation(self, obj):
    try:
        self.curr.execute(f"INSERT INTO participation VALUES ('{obj['id']}', '{obj['usn']}')")
        self.conn.commit()
        return True
    except:
        return False

def get_participation(self):
    return self.curr.execute("SELECT * FROM participation").fetchall()

def get_participation_details(self):
    part = self.get_participation()
    details = list()
    for i in range(len(part)):
        details.append([part[i][0], i+1])
        stu_details = self.curr.execute(f"select name from students where usn='{part[i][1]}'").fetchone()
        details[i] += [part[i][1], stu_details[0]]
        act_details = self.curr.execute(f"select * from activities where act_id='{part[i][0]}'").fetchone()
        details[i] += list(act_details[1:])
    return details

def get_participation_headings(self):
    return ["S. No.", "USN", "Name"]+self.get_headings('activities')[1:]

def del_participation(self, str):
    try:
        id, usn = str.split()
        self.curr.execute(f"DELETE FROM participation WHERE act_id='{id}' AND usn='{usn}'")
        self.conn.commit()
        return True
    except:
        return False

def del_participation_by_usn(self, usn):
    try:
        self.curr.execute(f"DELETE FROM participation WHERE usn='{usn}'")
        self.conn.commit()
        return True
    except:
        return False

def del_participation_by_act_id(self, act_id):
    try:
        self.curr.execute(f"DELETE FROM participation WHERE act_id='{act_id}'")
        self.conn.commit()
        return True
    except:
        return False
```

Code Snippet for insertion, deletion and retrieval for participation table

TABLE ACTIVITIES

```
def insert_activities(self, obj):
    try:
        self.curr.execute(f"INSERT INTO activities VALUES ('{obj['id']}', '{obj['guide']}', '{obj['type']}', '{obj['cat']}', '{obj['name']}', '{obj['mode']}', '{obj
        ['place']}', '{obj['ext']}', '{obj['start']}', '{obj['end']}', '{obj['days']}', '{obj['cert']}', '{obj['report']}')")
        self.conn.commit()
        return True
    except:
        return False

def get_activities(self):
    return self.curr.execute("SELECT * FROM activities").fetchall()

def del_activity(self, id):
    try:
        self.curr.execute(f"DELETE FROM activities WHERE act_id='{id}'")
        # delete participation records
        self.del_participation_by_act_id(id)
        self.conn.commit()
        return True
    except:
        return False
```

Code Snippet for insertion, deletion and retrieval for activities table

TRIGGERS

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.

Syntax:

create trigger [trigger_name]

[before | after]

{insert | update | delete}

on [table_name]

[for each row]

[trigger_body]

Trigger for Count of Students

In this trigger we aim to keep track of the number of students that are in the database. Here, we have used AFTER INSERT which will increment the count of students into a separate table as shown below.

```
CREATE TRIGGER stui

AFTER INSERT ON students
BEGIN
UPDATE total_count SET total = total + 1 WHERE entity = 'Students';
END;

CREATE TRIGGER stud
AFTER INSERT ON students
BEGIN
   UPDATE total_count SET total = total - 1 WHERE entity = 'Students';
END;
```

Trigger for Count of Subjects

In this trigger we aim to keep track of the number of subjects that are in the database. Here, we have used AFTER INSERT which will increment the count of subjects into a separate table as shown below.

```
CREATE TRIGGER subi
AFTER INSERT ON subjects
```

```
BEGIN
   UPDATE total_count SET total = total + 1 WHERE entity = 'Subjects';
END;


CREATE TRIGGER subd
AFTER INSERT ON subjects
BEGIN
   UPDATE total_count SET total = total - 1 WHERE entity = 'Subjects';
END;
```

Trigger for Count of Subjects

In this trigger we aim to keep track of the number of subjects that are in the database. Here, we have used AFTER INSERT which will increment the count of subjects into a separate table as shown below.

```
CREATE TRIGGER subi
AFTER INSERT ON subjects
BEGIN
   UPDATE total_count SET total = total + 1 WHERE entity = 'Subjects';
END;

CREATE TRIGGER subd
AFTER INSERT ON subjects
BEGIN
   UPDATE total_count SET total = total - 1 WHERE entity = 'Subjects';
END;
```

IMPLEMENTATION OF TRIGGERS IN SQLite3

```
    trig_names = ['stui', 'stud', 'subi', 'subd', 'depi', 'depd']
    triggers = {'stui':'''CREATE TRIGGER stui
AFTER INSERT ON students
BEGIN
    UPDATE total_count SET total = total + 1 WHERE entity = 'Students';
END;''',
    'stud':'''CREATE TRIGGER stud
AFTER DELETE ON students
BEGIN
    UPDATE total_count SET total = total - 1 WHERE entity = 'Students';
END;''',
    'subi':'''CREATE TRIGGER subi
AFTER INSERT ON subjects
BEGIN
    UPDATE total_count SET total = total + 1 WHERE entity = 'Subjects';
END;''',
    'subd':'''CREATE TRIGGER subd
AFTER DELETE ON subjects
BEGIN
    UPDATE total_count SET total = total - 1 WHERE entity = 'Subjects';
END;''',
    'depi':'''CREATE TRIGGER depi
AFTER INSERT ON departments
BEGIN
    UPDATE total_count SET total = total + 1 WHERE entity = 'Departments';
END;''',
    'depd':'''CREATE TRIGGER depd
AFTER DELETE ON departments
BEGIN
    UPDATE total_count SET total = total - 1 WHERE entity = 'Departments';
END;'''}

    def make_triggers(self):
        existing_trigs = self.curr.execute("SELECT name from sqlite_master WHERE TYPE='trigger'").fetchall()
        existing_trigs = set([trig[0] for trig in existing_trigs])
        for trig in self.trig_names:
            if trig not in existing_trigs:
                self.curr.execute(self.triggers[trig])
        self.conn.commit()
```

Code Snippet for implementation of Triggers using dictionaries

PSEUDO CODE

Pseudocode is an informal high-level description of the operating principle of a computer program or other algorithm. It uses the structural conventions of a normal programming language, but is intended for human reading rather than machine reading.

Algorithm for Table Display

Step1: BEGIN

Step 2: Establish connection with the database using the username and password of the database.

Step 3: Define the select query to retrieve all the values from the DBMS

Step 4: Define Default Table Model for the table and fetch the details from the database.

Step 5: END

Algorithm for Insert

Step 1: BEGIN

Step 2: Get all the necessary values required for insertion into variable defined in the method.

Step 3: Define the query for insertion as stated above.

Step 4: Execute the Query using the **(Select * from)** the required table.

Step 5: END

Algorithm for Update

Step 1: BEGIN

Step 2: Get all the necessary values required for updating the values into the variable defined in the method.

Step 3: UPDATE table name

SET column1 = value1, column2 = value2, ...

WHERE condition;

Step 4: Define the Query for Updating as stated above.

Step 5: Execute the Query using the method defined.

Step 6: END

Algorithm for Delete

Step 1: BEGIN

Step 2: Get the model number of the instrument which is to be deleted into a variable defined in the method.

Step 3: Delete from table_name where condition;

Step 4: Define the Query for deleting as stated above.

Step 5: Execute the Query using the **executeUpdate()** method defined.

Step 6: END

Algorithm for Joins

Nested Loop Join

In the nested loop join algorithm, for each tuple in outer relation, we have to compare it with all the tuples in the inner relation then only the next tuple of outer relation is considered. All pairs of tuples which satisfy the condition are added in the result of the join.

for each tuple t_R in T_R **do**
 for each tuple t_s in T_s **do**
  compare (t_R, t_s) if they satisfies the condition
  add them in the result of the join
 end
end

Block Nested Loop Join:

In block nested loop join, for a block of outer relation, all the tuples in that block are compared with all the tuples of the inner relation, then only the next block of outer relation is considered. All pairs of tuples which satisfy the condition are added in the result of the join.

for each block b_R in B_R **do**
 for each block b_s in B_S **do**
  for each tuple t_R in T_R **do**
   for each tuple t_s in T_s **do**
    compare (t_R, t_s) if they satisfies the condition
    add them in the result of the join

```
  end
 end  end
end
```

Algorithm for Triggers

Step 1: create trigger [trigger_name]: Creates or replaces an existing trigger with the trigger_name.

Step 2: [before | after]: This specifies when the trigger will be executed.

Step 3: {insert | update | delete}: This specifies the DML operation.

Step 4: on [table_name]: This specifies the name of the table associated with the trigger.

Step 5: [for each row]: This specifies a row-level trigger, i.e., the trigger will be executed for each row being affected.

Step 6: [trigger_body]: This provides the operation to be performed as trigger is fired

HTML CODE

index.html

```
{% extends 'base.html' %}
{% block title %}
Index Page
{% endblock title %}

{% block body %}
<div class="container-fluid container-md my-5 pt-5" style="max-width:667px">
    <h1 class="text-white" style="text-align: center; margin-bottom: 12px;">Student Marks and Activity</h1>
    <h1 class="text-white" style="text-align: center; margin-bottom: 100px;">Database Management System</h1>
    <h4 class="text-white" style="text-align: center; margin-bottom: 50px;">Created by</h4>
    <h3>
        <table class="table table-borderless">
            <thead>
                <tr>
                    <td class="text-white" style="text-align: center;">Name</td>
                    <td class="text-white" style="text-align: center;">USN</td>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td class="text-white" style="text-align: center;">Al Aqmar Damana</td>
                    <td class="text-white" style="text-align: center;">1JS20IS011</td>
                </tr>
                <tr>
                    <td class="text-white" style="text-align: center;">Aman Kumar</td>
                    <td class="text-white" style="text-align: center;">1JS20IS012</td>
                </tr>
            </tbody>
        </table>
    </h3>
</div>
{% endblock body %}
```

code snippet of index.html

base.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title>{% block title %}{% endblock title %}</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="stylesheet">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body style="background: url(/bg);">
  <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
    <div class="container-fluid">
      <a href="/" class="navbar-brand">
        <img src="/logo" alt="JSSATE-B" id="logo" style="width: 40px;">
      </a>
      <ul class="navbar-nav">
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/">Home</a>
        </li>
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/students">Students</a>
        </li>
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/subjects">Subjects</a>
        </li>
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/departments">Departments</a>
        </li>
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/marks">Marks</a>
        </li>
        <li class="nav-item" style="margin-right: 36px;">
          <a class="nav-link" href="/count">Count</a>
        </li>
        <li class="nav-item dropdown" style="margin-right: 36px;">
          <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown">Internship</a>
          <ul class="dropdown-menu">
            <li><a class="dropdown-item" href="/activities">Activities</a></li>
            <li><a class="dropdown-item" href="/participation">Participation</a></li>
          </ul>
        </li>
        {% block insert_button %}
        {% endblock insert_button %}
      </ul>
    </div>
  </nav>

  {% block body %}
  {% endblock body %}
</body>

</html>
```

Code snippet for base.html

CHAPTER 6

SYSTEM TESTING

INTRODUCTION

Testing plays a vital role in the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Once program code has been developed, testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals, that is conducted tests to uncover errors and ensure that defined input will produce actual results that agree with required results. Broadly speaking, there are at least three levels of testing: unit testing, integration testing, and system testing.

TYPES OF TESTING

Unit testing

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to ensure that the building blocks of the software work independently from each other.

Unit testing is a software development process that involves a synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Unit testing aims to eliminate construction errors

before code is promoted to additional testing; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development process.

Sl. No.	Test Cases	Pass	Fail
1	Database	✓	
2	Database Updation	✓	
3	Foreign Key Constraints	✓	
4	Front-end View	✓	
5	Main Program	✓	
6	Procedures	✓	
7	Referential IntegrativeConstraints	✓	
8	Relational Schema	✓	
9	Tokens	✓	
10	Triggers	✓	
11	Validation of Inputs	✓	

Integration Testing

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

Students Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	i. The user needs to fill all the details that are compulsory. “Item successfully inserted” message should be displayed.	Successful
		ii. Student USN and Name fields are compulsory and USN should be in the correct format. “Invalid format” message should be displayed otherwise.	Successful

Subjects Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	i. The user needs to fill all the details that are compulsory. “Item successfully inserted” message should be displayed.	Successful
		ii. Subject code field is compulsory and should be in the correct format. “Invalid format” message should be displayed otherwise.	Successful

Department Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	The user needs to fill all the details that are compulsory. "Item successfully inserted" message should be displayed else "Unable to insert item" message should be displayed if any compulsory field(s) is left empty.	Successful

Marks Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	The user needs to fill all the details that are compulsory. "Item successfully inserted" message should be displayed.	Successful

Activity Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	The user needs to fill all the details that are compulsory. "Item successfully inserted" message should be displayed else "Unable to insert item" message should be displayed if any compulsory field(s) is left empty.	Successful

Participation Form page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking submit	The user needs to fill all the details that are compulsory. "Item successfully inserted" message should be displayed else "Unable to insert item" message should be displayed if any compulsory field(s) is left empty.	Successful

System Testing

System testing tests a completely integrated system to verify that the system meets its requirements. For example, a system test might involve testing a logon interface, then creating and editing an entry, plus sending or printing results, followed by summary processing or deletion (or archiving) of entries, then logoff.

Count page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking count from Navigation bar	Count for the number of students, subjects, departments should be displayed.	Successful

Student Details display page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking students from Navigation bar	The details entered through the student form should be displayed and the count of student should be incremented in the count table for each entry.	Successful
2	On clicking delete	The student entry should get deleted and the marks entries associated with that student should also be deleted.	Successful

Subject Details display page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking subject from Navigation bar	The details entered through the subject form should be displayed and the count of subject should be incremented in the count table for each entry.	Successful
2	On clicking delete	The subject should get deleted and the marks entries associated with that subject should also be deleted.	Successful

Department Details display page

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking department from Navigation bar	The details entered through the department form should be displayed and the count of department should be incremented in the count table for each entry.	Successful
2	On clicking delete	The department should get deleted and the student entries associated with that department should also be deleted.	Successful

Navigation bar

Sl. No.	Test Case	Excepted Result	Test Result
1	On clicking Students from Navigation bar	The details of all students should be displayed and a new student detail can be added by clicking on the “insert new student” button which will redirect to Students Form page.	Successful
2	On clicking Subjects from Navigation bar	The details of all Subjects should be displayed and a new subject detail can be added by clicking on the “insert new subject” button which will redirect to Subject Form page.	Successful
3	On clicking Departments from Navigation bar	The details of all Departments should be displayed and a new Departments detail can be added by clicking on the “insert new	Successful

		Departments" button which will redirect to Departments Form page.	
4	On clicking Marks from Navigation bar	The details of all Marks should be displayed and a new Marks detail can be added by clicking on the "insert new Marks" button which will redirect to Marks Form page.	Successful
5	On clicking Activities from "Internship" dropdown in Navigation bar	The details of all Activities should be displayed and a new Activity detail can be added by clicking on the "insert Activity Details" button which will redirect to Activity Form page.	Successful
6	On clicking Participation from "Internship" dropdown in Navigation bar	The details of all Participation should be displayed and a new Participation detail can be added by clicking on the "insert Participation Details" button which will redirect to Participation Form page.	Successful

CHAPTER 7

RESULTS AND DISCUSSIONS

Home page of STUDENT MARKS AND ACTIVITY DATABASE MANAGEMENT SYSTEM

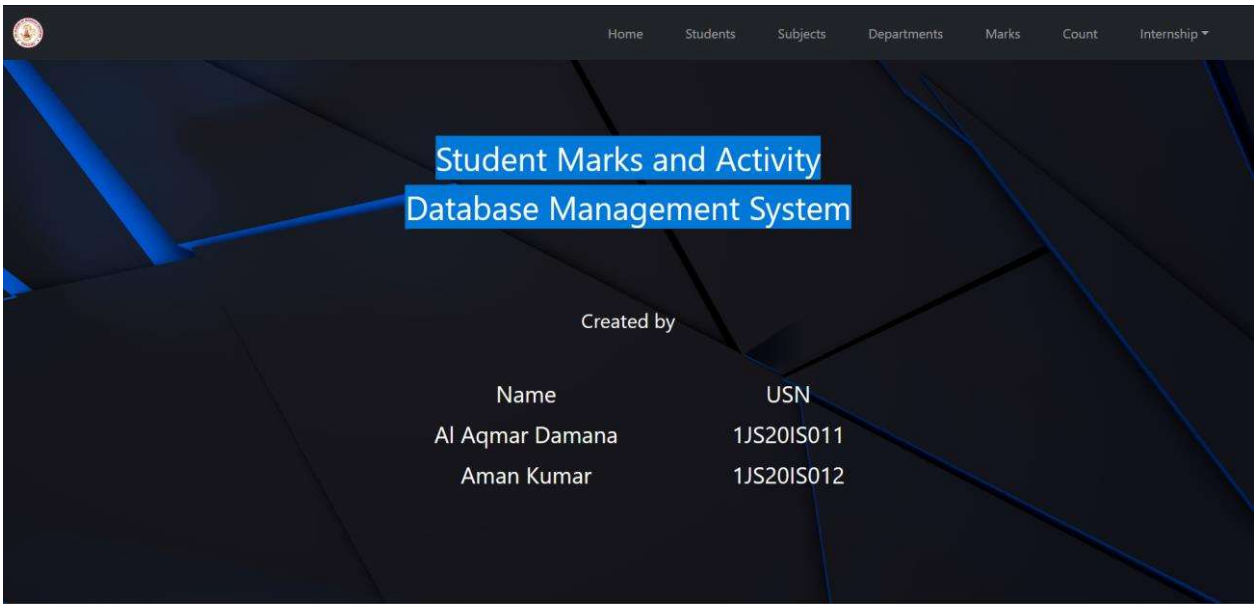
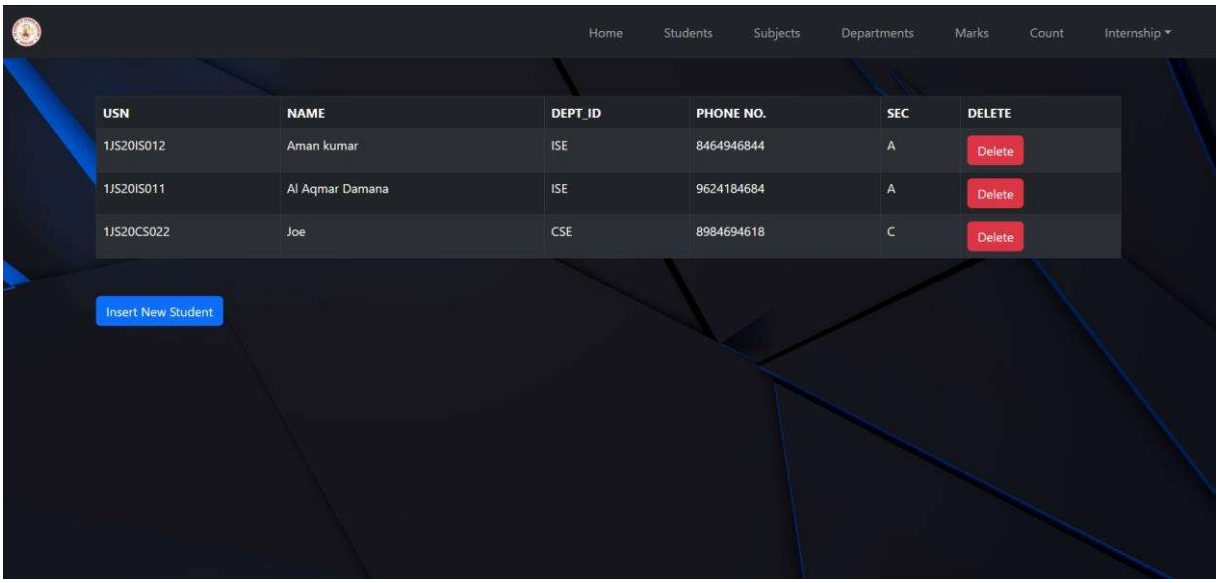
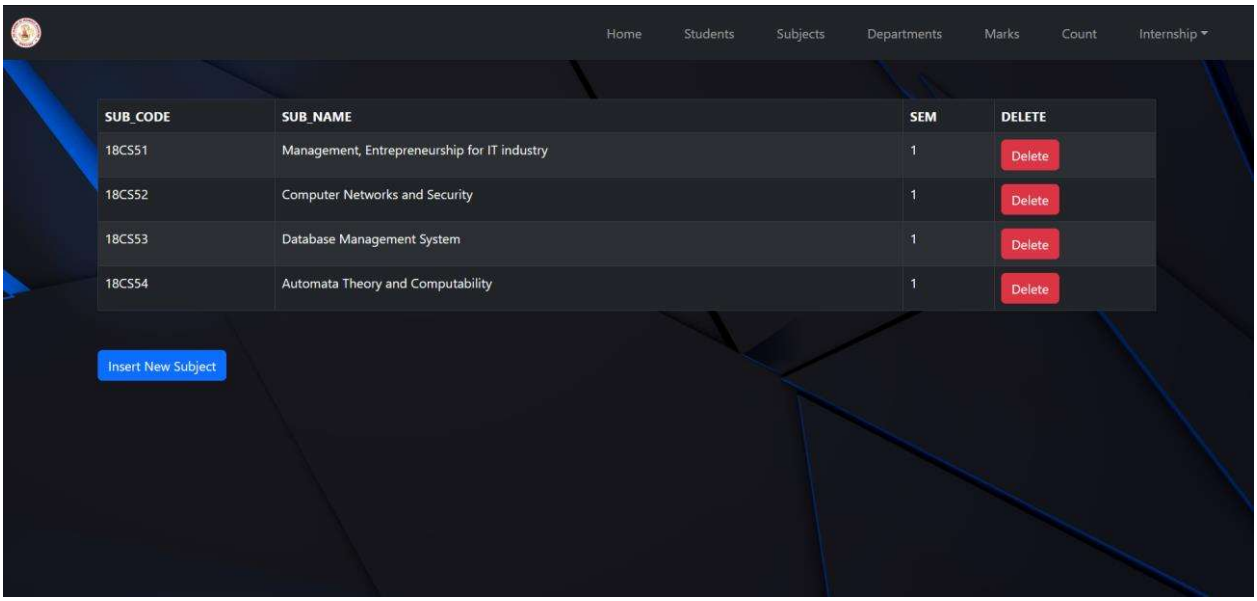


Fig4:- Home Page of Student Marks And Activity Database Management

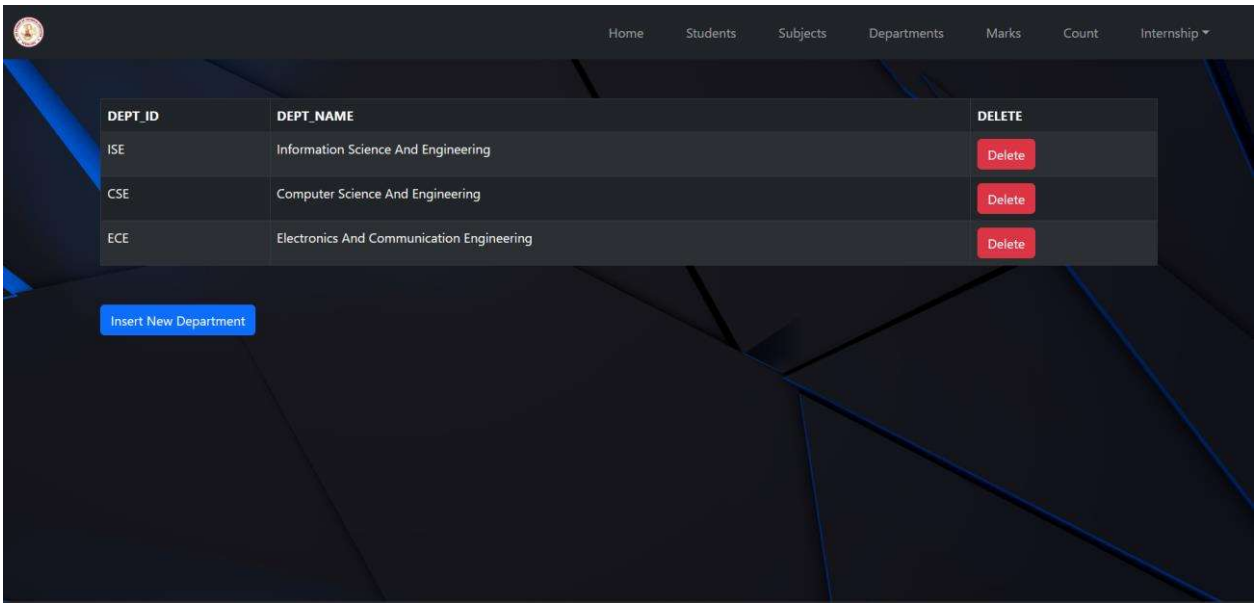
Student Details page



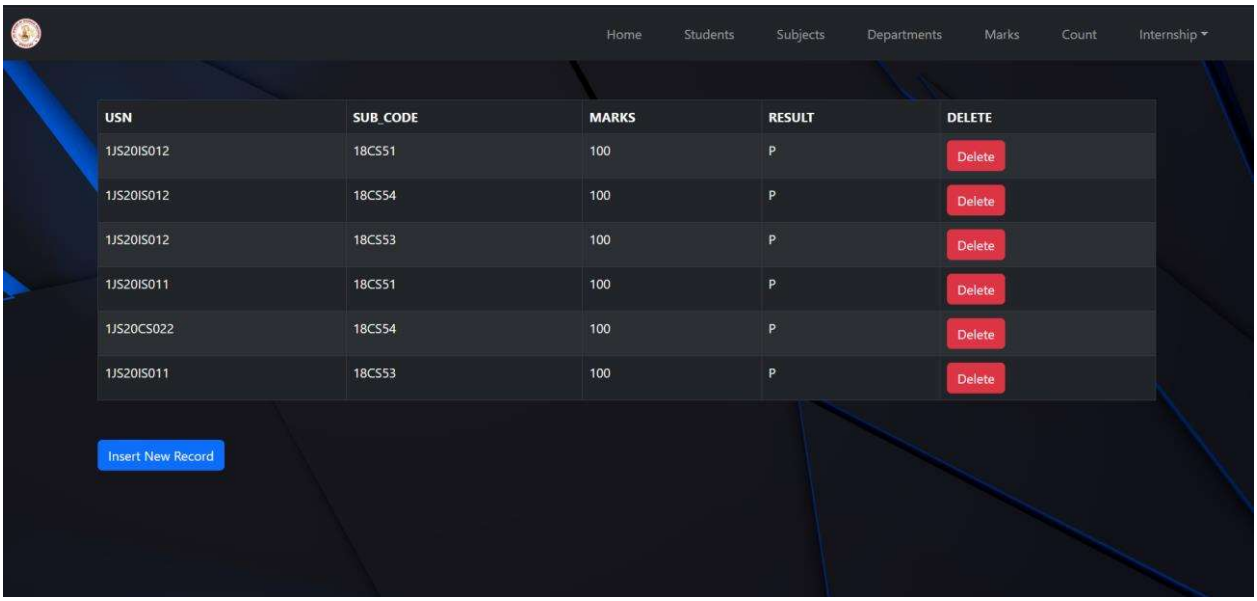
Details of all the Subjects



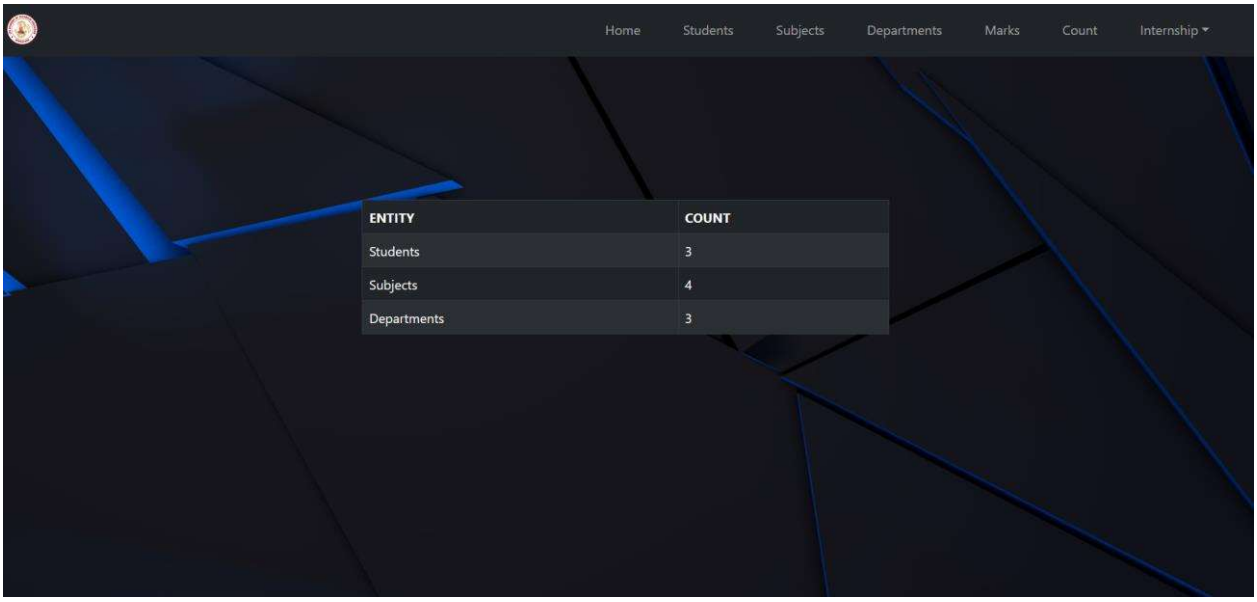
Webpage to display Departments



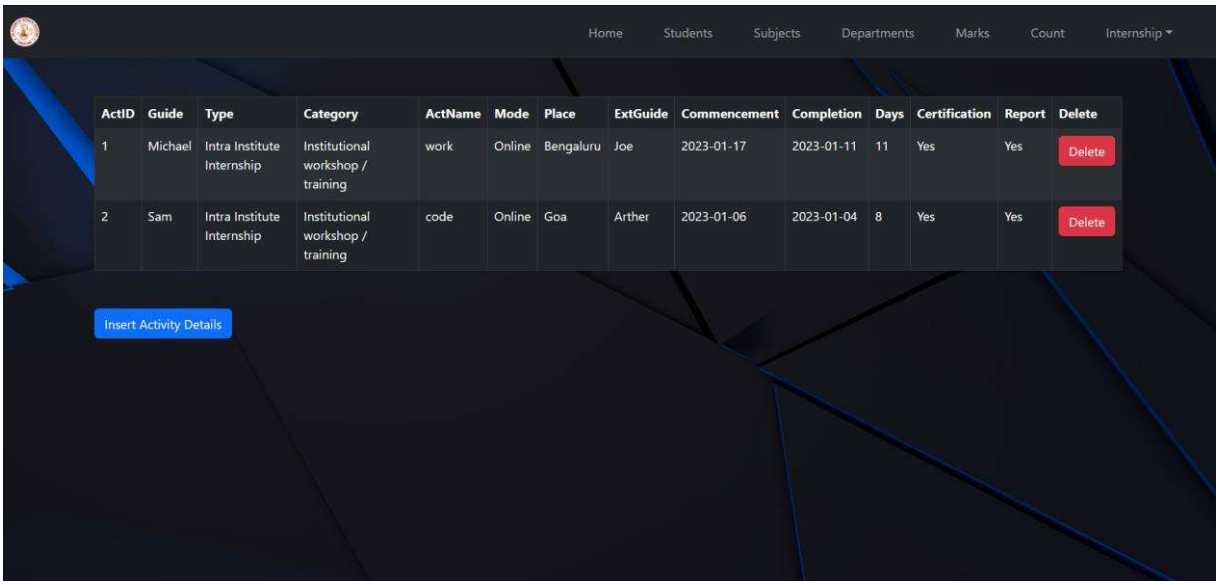
Webpage to display Marks of Students



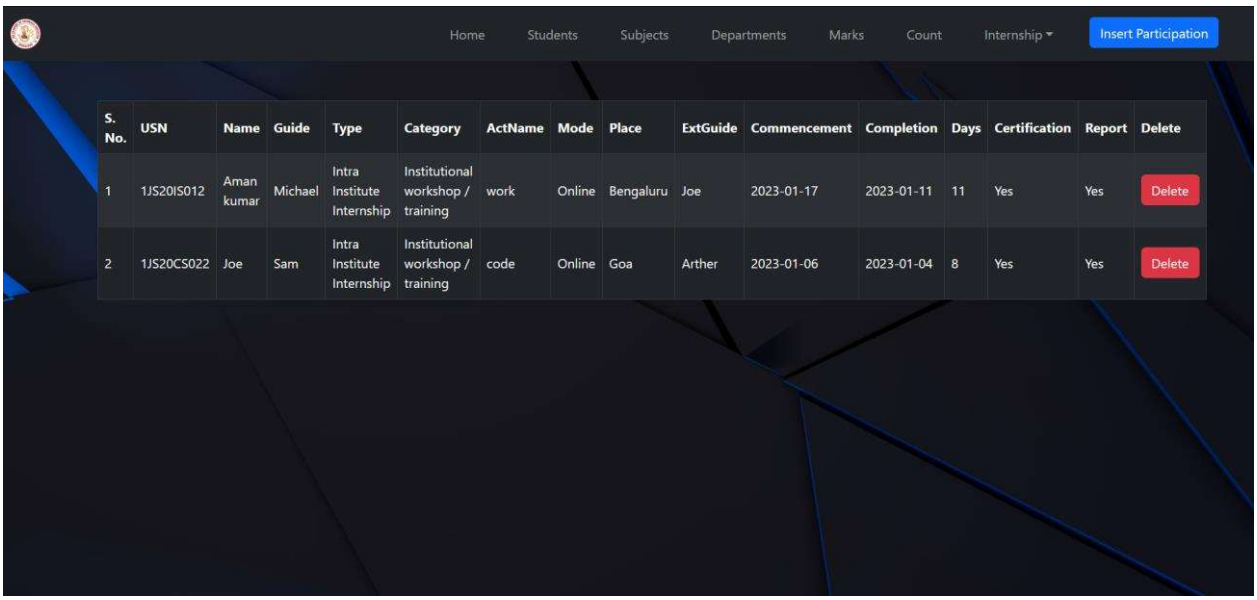
Webpage to display Counts of Subjects,Students and Departments



Details of All Activities



Details about Participation of Students



CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

The project was developed to nurture the needs of faculty members concerning Students Marks and Activities data management. This easy-to-use computerized version of student data will not only help the faculty but will also help in ease of administration.

In this entire process, the project ensures that data stored is easy to access and available at all times. The security and encapsulation of data are provided by the triggers, as the triggers help in keeping a count of the data.

This is a small prototype of a management application for student data in college. The limitation of the application is that it lacks enough features to be implemented in a real-life situation. Still it is very useful to keep a track of all the activities performed by the students throughout their college.

FUTURE ENHANCEMENT

1. The scope for this project is to store the data of students and keep a record of their marks and activities to help the teachers in keeping a track of student's performance.
2. The platform can be hosted on online servers to make it accessible worldwide.
3. Distribute the loads of the system.
4. More updates can be made related to student activities and their attendance status.

CHAPTER 9

REFERENCES

BOOK REFERENCES

1. Database systems Models, Languages, Design and Application Programming, Ramez
2. Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
3. An Introduction to Database Systems" by Bipin Desai.
4. Database System Concepts" by Abraham Silberschatz and S Sudarshan.
5. "Database Management Systems" by Raghu Ramakrishnan.
6. Automate the Boring stuff with Python by Al Sweigart.

WEB REFERENCES

1. www.geeksforgeeks.org
2. https://stackoverflow.com/questions
3. www.tutorialspoint.com
4. www.javatpoint.com
5. https://flask.palletsprojects.com/en/2.2.x/tutorial/
6. https://www.sqlite.org/docs.html
7. W3schools