CHAPTER 1

INTRODUCTION

"Pt. Sadhu Maharaj owns a Music School for students interested in Hindustani Classical Music. There are around 15 students in his school. The music school is linked with few agencies to provide opportunities for students by setting up events.

The music school holds the record of students with their details such as Name, Registration Number (in yyHINrrr format where yy stands for year, rrr for student number), Current Level, Contact Number, Age, Date Joined, Admission Fees and Monthly Fees(varying).

Hindustani classical music has six levels named praveshikaa, madhyamaa, vishaarad pratham, vishaarad puurna, alankar pratham and alankar puurna. If a new student joins the class he or she will be on the first level i.e. praveshikaa. Each level of Hindustani classical music has different Raags. For example, praarambhik Raag has some Raags like Kalyaan and Bhoop, Durga, Khamaaj, etc. Each Raags have their own properties Aaroh and Avaroh.

Music School conducts an exam once a year. Students will have to clear every level in order to qualify for the next level. Music class wants to keep track of the details like Student Registration Number, Status and Date of Examination. Panditji wants to keep track of details of fees payment like Student Registration Number, Date of fees payment and Amount.

Music School is associated with a number of agencies that provide an opportunity for students to perform live. Panditji would also like to keep track of the same with the date of the event, place of the event, a student Registration Number and the name of the agency.

MySQL

MySql is the most popular and freely available open source Relational Database Management System that uses Structured Query Language (SQL). SQL is the most popular language for accessing and managing content in a database.

Python

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Flask

Flask is a micro web framework written in Python. It is classified as microframework because it does not require particular tools or libraries. It is designed to make getting started quick and easy, with the ability to scale up to complex applications.

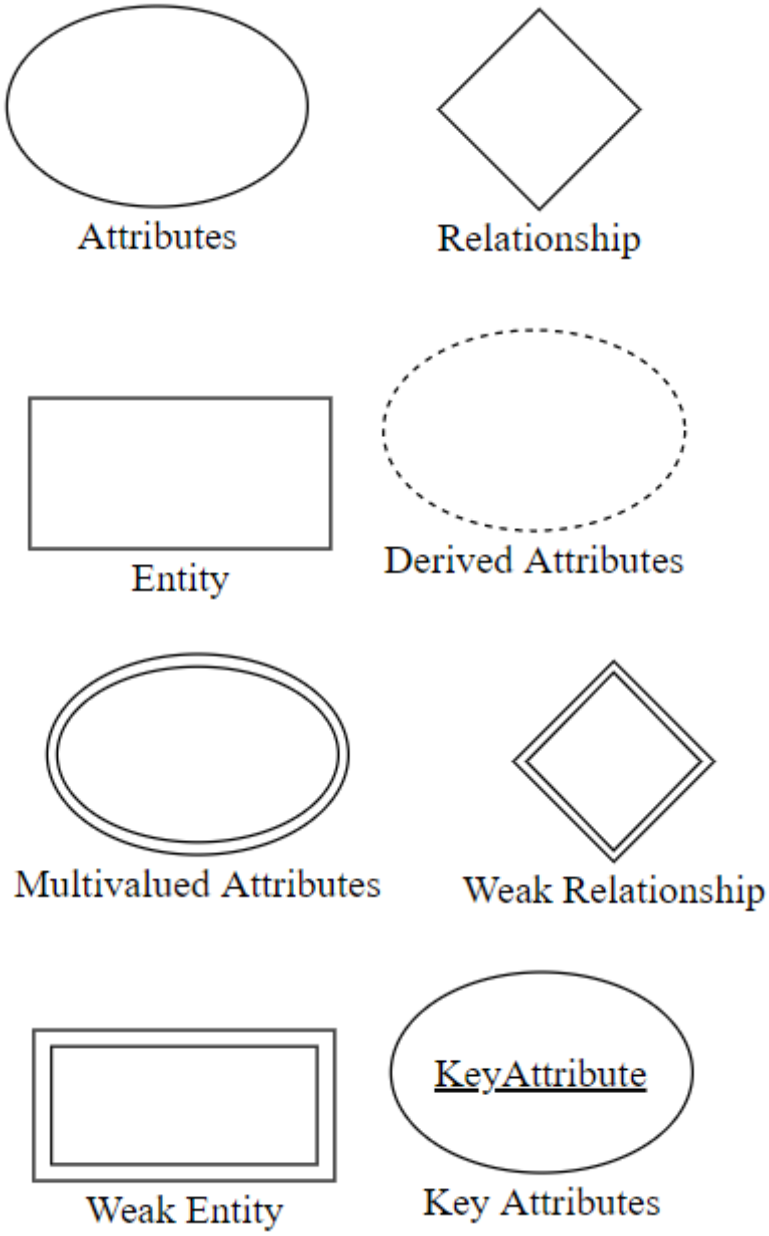
CHAPTER 2

DESIGN

2.1 ER DIAGRAM

An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. The ER Diagram of our project is shown in the figure:2.1

Symbols in Entity Relationship:



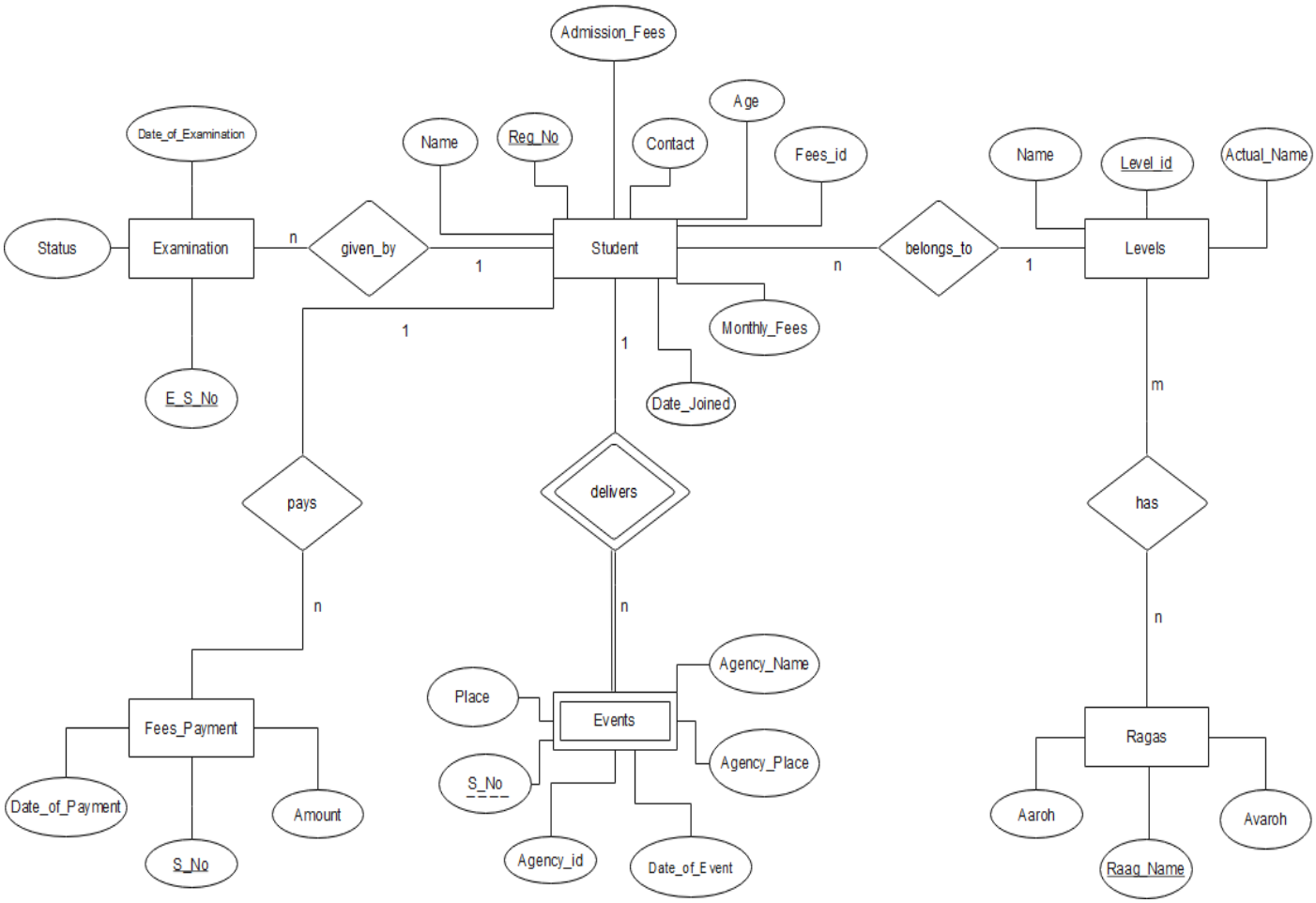


Figure 2.1: E-R Diagram

2.2 RELATIONAL SCHEMA

1. Mapping of Regular Entity Type

For every regular entity in our entity relationship diagram, we have created a separate relation. These created relations contain the respected attributes and respected primary key.

Level

Level_id	Name	Actual_Name

Raga

Raga_Name	Aaroh	Avaroh

Student

Name	Reg_No	Contact	Date_Joined	Place	Fees_id	Monthly_fees	Admission_Fees	Age

Examination

E_S_No	Date_of_Examination	Status

Fees_Payment

S_No	Date_of_Payment	Amount

Figure 2.2.1: Mapping of Regular Entities

2. Mapping of Weak Entity Types

When mapping weak entity types along with other attributes the partial key and primary key of parent entity together will form their primary key of the new relation.

Events

S_No	Student_Reg_No	Place	Date_of_Event	Agency_id	Agency_Name	Agency_Place

Figure 2.2.2 : Mapping of Weak Entities

3. Mapping of Binary 1:1 Relationship Types

Since our ER Diagram has no binary relationship this step is ignored for our project.

4. Mapping of Binary 1: N Relationship Types

In this step, we map binary 1:N relationships and the primary key of the entity on 1 side is made as foreign key in the entity present on the N side of relationship.

Student

Name	Reg_No	Level	Contact	Date_Joined	Place	Fees_id	Monthly_fees	Admission_Fees	Age

Examination

E_S_No	Student_Reg_No	Date_of_Examination	Status

Fees_Payment

S_No	Student_Reg_No	Date_of_Payment	Amount

Figure 2.2.3 : Mapping of 1:N Relationship Types

5. Mapping of M:N Relationship Types

In this step, we map M:N relationships. Here we map the primary keys of two table and create another entity with the relation name.

Level_Ragas

Level_id	Raga_Name

Figure 2.2.4 : Mapping of M:N Relationship Types

6. Mapping of Multivalued Attributes

Since our ER Diagram has no multivalued attributes this step is ignored for our project.

7. Mapping of N-ary Relationship Types

Since our ER Diagram has N-ary relationship types this step is ignored for our project.

2.3 SCHEMA DIAGRAM

A schema diagram is a diagram which contains entities and the attributes that defines a schema. A schema diagram only shows us the database design. Relation diagram for our project can be seen in figure 2.3.1.

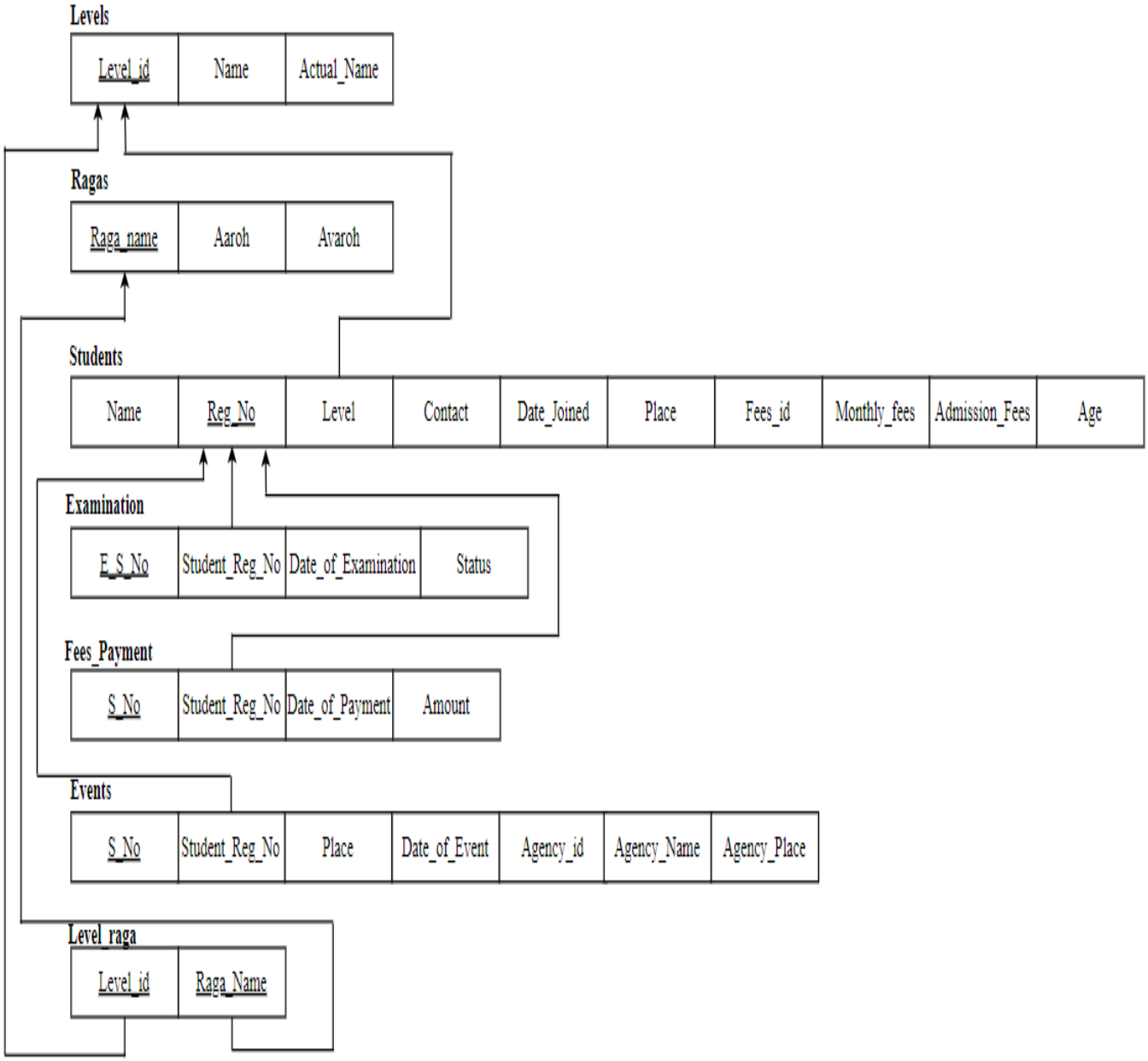
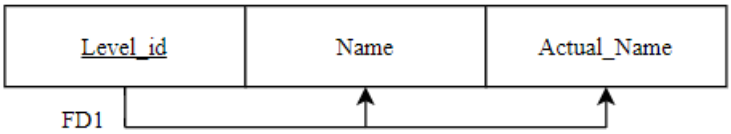


Figure 2.3.1: Schema Diagram

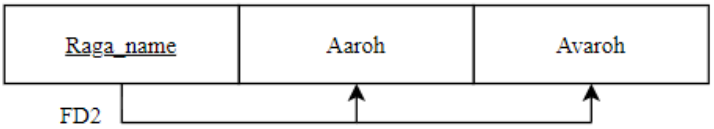
2.4 NORMALIZATION

Levels



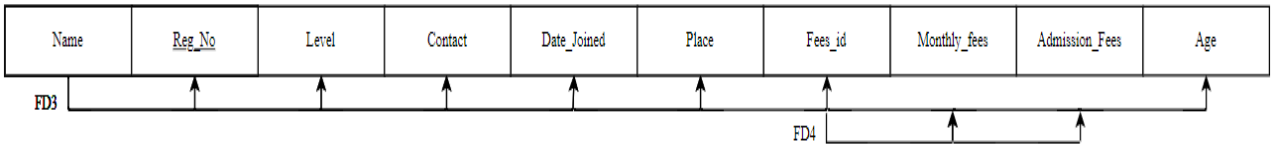
- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is in 3NF because there is no transitive dependency.

Ragas

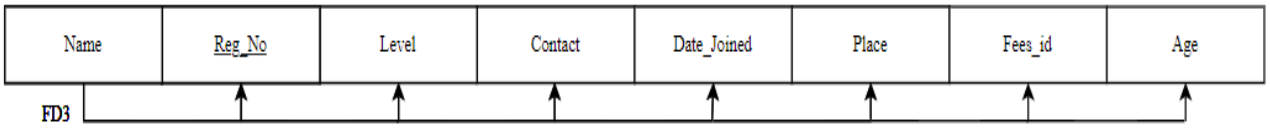


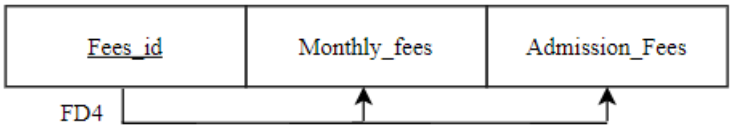
- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is in 3NF because there is no transitive dependency.

Students



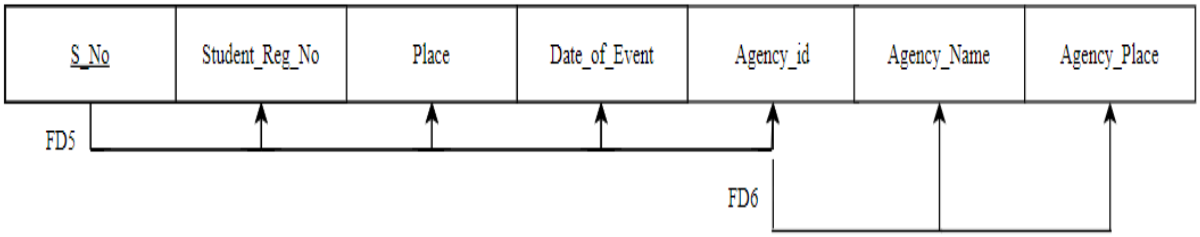
- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is not in 3NF because there is a transitive dependency.
- Hence, we split Students into two tables to obtain 3NF.



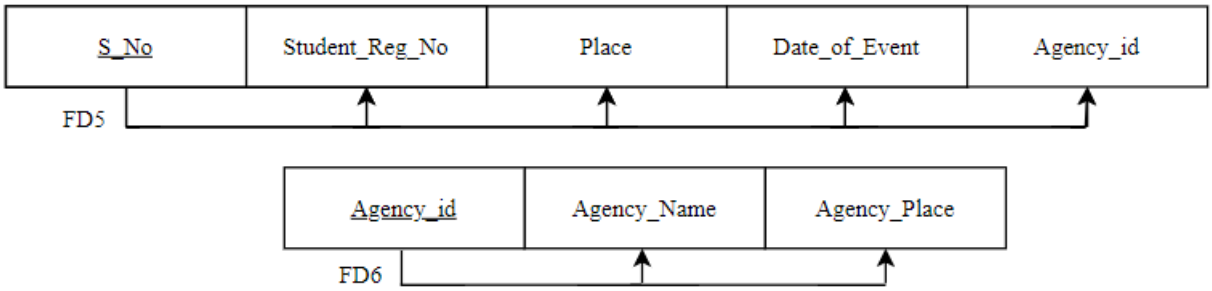


- The above obtained functional dependencies are now in 3NF.

Events

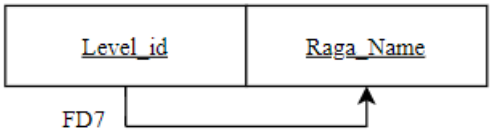


- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is not in 3NF because there is a transitive dependency.
- Hence, we split Events into two tables to obtain 3NF.



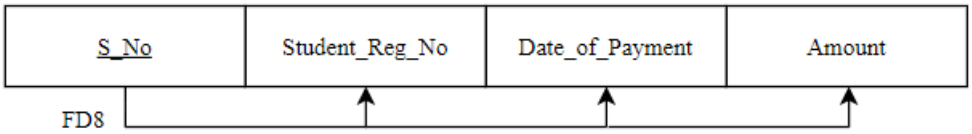
- The above obtained functional dependencies are now in 3NF.

Level_Ragas



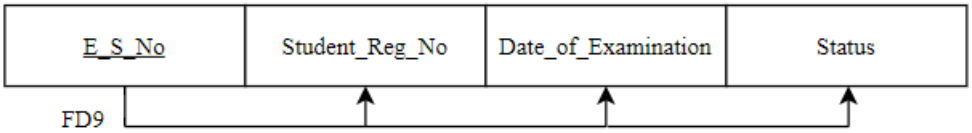
- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is in 3NF because there is no transitive dependency.

Fees_Payment



- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is in 3NF because there is no transitive dependency.

Examination



- The above relation is in 1NF because there are no multivalued attributes in the relational schema.
- The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
- The above relation is in 3NF because there is no transitive dependency.

CHAPTER 3

IMPLEMENTATION

3.1 System Specification

Operating System: Windows XP/7/8/10/MacOS/Linux.

Memory: Minimum of 1GB of RAM, Minimum of 2GB hard disk space.

Backend: MySQL Workbench 8.0 CE

Frontend: Flask (framework of Python)

3.2 Table Structure

3.2.1 Levels

CREATE TABLE LEVELS(

LEVEL_ID INT PRIMARY KEY,

NAME VARCHAR(15),

ACTUAL_NAME VARCHAR(20));

```
mysql> desc levels;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Level_id    | int         | NO   | PRI | NULL    |       |
| Name        | varchar(15) | YES  |     | NULL    |       |
| Actual_Name | varchar(20) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

3.2.2 Ragas

CREATE TABLE RAGAS(

RAGA_NAME VARCHAR(15) PRIMARY KEY,

AAROH VARCHAR(60),

AVAROH VARCHAR(60));

```
mysql> desc ragas;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Raga_Name | varchar(15) | NO   | PRI | NULL    |       |
| Aaroh     | varchar(60) | YES  |     | NULL    |       |
| Avaroh    | varchar(60) | YES  |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

3.2.3 Students

CREATE TABLE STUDENTS (
NAME VARCHAR(15),
REG_NO CHAR(8) PRIMARY KEY CHECK(REG_NO LIKE '__HIN___',
LEVEL INT,
CONTACT BIGINT,
DATE_JOINED DATE,
FEES_ID INT,
AGE INT,
FOREIGN KEY(LEVEL) REFERENCES LEVELS(LEVEL_ID),
FOREIGN KEY(FEES_ID) REFERENCES FEES(FEES_ID));

```
mysql> desc students;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| Name        | varchar(50) | YES  |     | NULL    |       |
| Reg_No      | char(8)     | NO   | PRI | NULL    |       |
| Level       | int         | YES  | MUL | NULL    |       |
| Contact     | bigint      | YES  |     | NULL    |       |
| Date_Joined | date        | YES  |     | NULL    |       |
| Fees_id     | int         | YES  | MUL | NULL    |       |
| Age         | int         | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

3.2.4 Fees

CREATE TABLE FEES(
FEES_ID INT PRIMARY KEY,
MONTHLY_FEES INT,
ADMISSION_FEES INT);

```
mysql> desc fees;
+----------------+------+------+-----+---------+-------+
| Field          | Type | Null | Key | Default | Extra |
+----------------+------+------+-----+---------+-------+
| Fees_id        | int  | NO   | PRI | NULL    |       |
| Monthly_Fees   | int  | YES  |     | NULL    |       |
| Admission_Fees | int  | YES  |     | NULL    |       |
+----------------+------+------+-----+---------+-------+
```

3.2.5 Agencies

CREATE TABLE AGENCIES(

AGENCY_ID INT PRIMARY KEY,

AGENCY_NAME VARCHAR(20),

AGENCY_PLACE VARCHAR(20));

```
mysql> desc agencies;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Agency_Id    | int         | NO   | PRI | NULL    |       |
| Agency_name  | varchar(20) | YES  |     | NULL    |       |
| Agency_Place | varchar(20) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

3.2.6 Events

CREATE TABLE EVENTS(

S_NO INT PRIMARY KEY,

AGENCY_ID INT,

STUDENT_REG_NO CHAR(8),

PLACE VARCHAR(30),

DATE_OF_EVENT DATE,

FOREIGN KEY(STUDENT_REG_NO) REFERENCES STUDENTS(REG_NO) ON DELETE CASCADE,

FOREIGN KEY(AGENCY_ID) REFERENCES AGENCIES(AGENCY_ID) ON DELETE SET NULL);

```
mysql> desc events;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| S_No           | int         | NO   | PRI | NULL    |       |
| Agency_Id      | int         | YES  | MUL | NULL    |       |
| Student_Reg_No | char(8)     | YES  | MUL | NULL    |       |
| Place          | varchar(30) | YES  |     | NULL    |       |
| Date_of_Event  | date        | YES  |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
```

3.2.7 Level_Ragas

CREATE TABLE LEVEL_RAGAS(

LEVEL_ID INT,

RAGA_NAME VARCHAR(15),

PRIMARY KEY(LEVEL_ID, RAGA_NAME),

FOREIGN KEY(LEVEL_ID) REFERENCES LEVELS ON DELETE CASCADE,

FOREIGN KEY(RAGA_NAME) REFERENCES RAGAS ON DELETE SET CASCADE);

```
mysql> desc level_ragas;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Level_id  | int         | NO   | PRI | NULL    |       |
| Raga_name | varchar(15) | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

3.2.8 FEES_PAYMENT

CREATE TABLE FEES_PAYMENT(

S_NO INT PRIMARY KEY,

STUDENT_REG_NO CHAR(8),

DATE_OF_PAYMENT DATE,

AMOUNT INT,

FOREIGN KEY(STUDENT_REG_NO) REFERENCES STUDENTS ON DELETE SET CASCADE);

```
mysql> desc fees_payment;
+-----------------+---------+------+-----+---------+-------+
| Field           | Type    | Null | Key | Default | Extra |
+-----------------+---------+------+-----+---------+-------+
| S_No            | int     | NO   | PRI | NULL    |       |
| Student_Reg_No  | char(8) | YES  | MUL | NULL    |       |
| Date_of_Payment | date    | YES  |     | NULL    |       |
| Amount          | int     | YES  |     | NULL    |       |
+-----------------+---------+------+-----+---------+-------+
```

3.2.9 EXAMINATION

CREATE TABLE EXAMINATION(

E_S_NO INT PRIMARY KEY,

STUDENT_REG_NO CHAR(8),

STATUS CHAR(10),

DATE_OF_EXAMINATION DATE,

FOREIGN KEY(STUDENT_REG_NO) REFERENCES STUDENTS ON DELETE SET CASCADE);

```
mysql> desc examination;
+---------------------+----------+------+-----+---------+-------+
| Field               | Type     | Null | Key | Default | Extra |
+---------------------+----------+------+-----+---------+-------+
| E_S_No              | int      | NO   | PRI | NULL    |       |
| Student_Reg_No      | char(8)  | YES  | MUL | NULL    |       |
| Status              | char(10) | YES  |     | NULL    |       |
| Date_of_Examination | date     | YES  |     | NULL    |       |
+---------------------+----------+------+-----+---------+-------+
```

3.3 Functionalities

3.3.1 Connecting to Database

The “Inventory Management System of Music School” has been developed Python and its Framework i.e., Flask. It uses MySql database for storing the data and it is connected by the use of library of Python mysql-connector. A block of code suggesting the connection is shown as follows:

```
Import mysql.connector
Mydb = mysql.connector.connect(host="localhost", user="root", passwd="7019252847",
database="sadhu")
c = Mydb.cursor()
c.execute("USE SADHU; ")
```

3.3.2 Insert

Insert operation is used to add student information, levels, ragas, level_ragas, agencies, examination, events, fees and fees_payment details to the database. A snippet for the same:

```
c.execute("""
        INSERT INTO STUDENTS(NAME, REG_NO, LEVEL, CONTACT, DATE_JOINED,
FEES_ID, AGE) VALUES(%s,%s,%s,%s.%s,%s,%s,%s)
        """, (NAME, REG_NO, LEVEL, CONTACT, DATE_JOINED, FEES_ID, AGE))
Mydb.commit()
```

flash(“Successfully inserted data into Students”)

Value for NAME, REG_NO, LEVEL, CONTACT, DATE_JOINED, FEES_ID, AGE are received from add_students route using POST method of Python.

3.3.3 Delete

Delete operation is used to remove unwanted data from the table, A basic code for the same is as follows:

```
c.execute("""
        DELETE FROM STUDENTS WHERE REG_NO = %s
        """,(REG_NO,))
Mydb.commit()
```

Value for REG_NO is received from delete_student_button route using POST method of Python and the changes can be seen in the database.

3.3.4 Update

Update operation is used to update the wrong entries or to updated status of examination in the table, A basic code for the same is as follows:

```
c.execute("""
        UPDATE EXAMINATION SET STATUS = %s, STUDENT_REG_NO = %s WHERE
E_S_NO = %s
        """,(STATUS, STUDENT_REG_NO, E_S_NO,))
Mydb.commit()
```

Value for STATUS, STUDENT_REG_NO and E_S_NO is obtained from the route Update_student_button using the POST method of Python.

3.3.5 Trigger

Trigger is used to update the student level in the database on updating the value of STATUS in the table EXAMINATION. Trigger automatically updates the value of LEVEL present in STUDENT table by LEVEL+1 when the value of STATUS is set to "PASSED".

```
delimiter#
CREATE TRIGGER EXAMINATION_AFTER_UPDATE
AFTER UPDATE ON EXAMINATION
FOR EACH ROW
BEGIN
        IF (new.STATUS = "Passed”) THEN
                UPDATE STUDENTS SET LEVEL = LEVEL+1 WHERE REG_NO =
                        (SELECT STUDENT_REG_NO FROM EXAMINATION WHERE
                        STUDENT_REG_NO = NEW.STUDENT_REG_NO);
        END IF;
END;
END#
```

3.3.6 Stored Procedure

A procedure is created to calculate the total amount of fees paid by the students till date. The implementation of the same can be seen below.

```
delimiter#
CREATE PROCEDURE FEES_HISTORY(IN STUD_REG_NO CHAR(8))
BEGIN
        SELECT STUDENT_REG_NO, SUM(|AMOUNT) FROM FEES_PAYMENT WHERE
        STUDENT_REG_NO = STUD_REG_NO;
END;
END#
```

3.3.7 Stored Function

A procedure is created here to calculate the student association years with the music school. The code for the same can be seen below:

```
CREATE FUNCTION NO_OF_YEARS (DATE1 DATE)
RETURNS INTEGER DETERMINISTIC
BEGIN
        DECLARE DATE2 DATE;
        SELECT CURRENT_DATE() INTO DATE2;
RETURN YEAR(DATE2)-YEAR(DATE1);
END;
```

3.3.8 Views

Views are created to obtain the events given by students, the code snippet for the same can be seen below:

```
CREATE VIEW STUDENT_EVENT_DETAILS AS
SELECT NAME, REG_NO, DATE_OF_EVENT
FROM STUDENTS, EVENTS
WHERE STUDENT.REG_NO = EVENTS.STUDENT_REG_NO
GROUP BY REG_NO;
```

Similarly, another view as been created to obtain student association details using the function stored function NO_OF_YEARS.

```
CREATE VIEW STUDENT_ASSOCIATION AS
SELECT NAME, NO_OF_YEARS(DATE_JOINED)
FROM STUDENTS
ORDER BY REG_NO;
```

CHAPTER 4

RESULTS

The Inventory Management System of Music School helps the administrator of the music school to keep track of records of music school in a systematic way and retrieve and manipulate the same when required.

4.1 Snapshots

4.1.1 Home Page

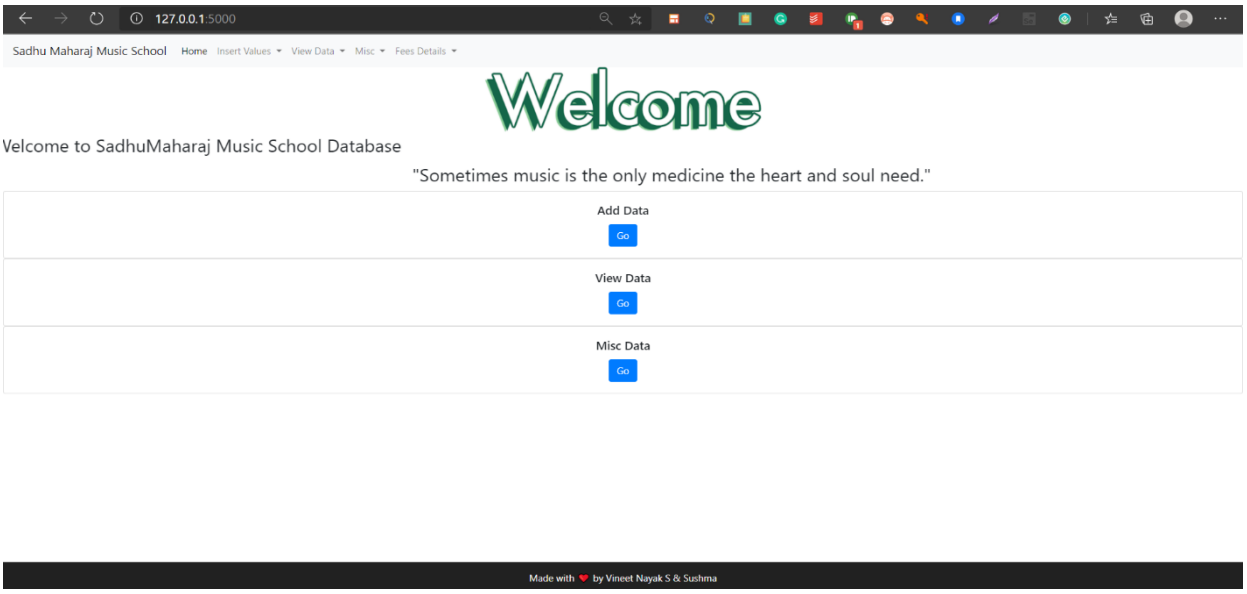


Figure 4.1.1: Home Page

The above shown snapshot is a Home page which connects all the key components used.

4.1.2 Add Data

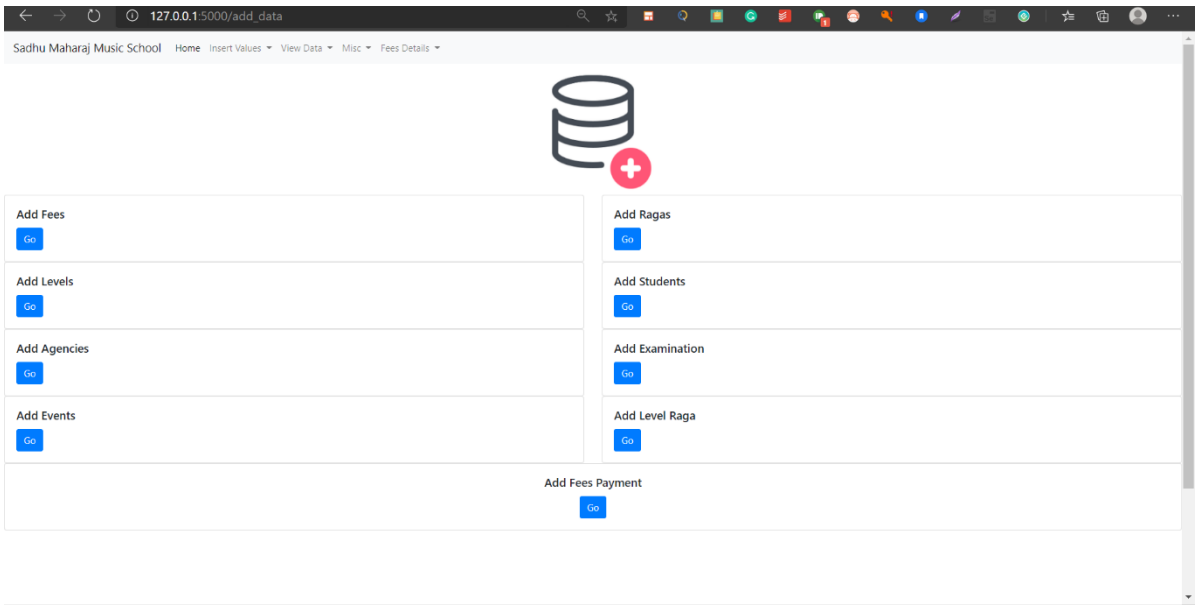


Figure 4.1.2 Add Data Page

4.1.2.1 Add Level Data

In this page admin can add details of Levels.

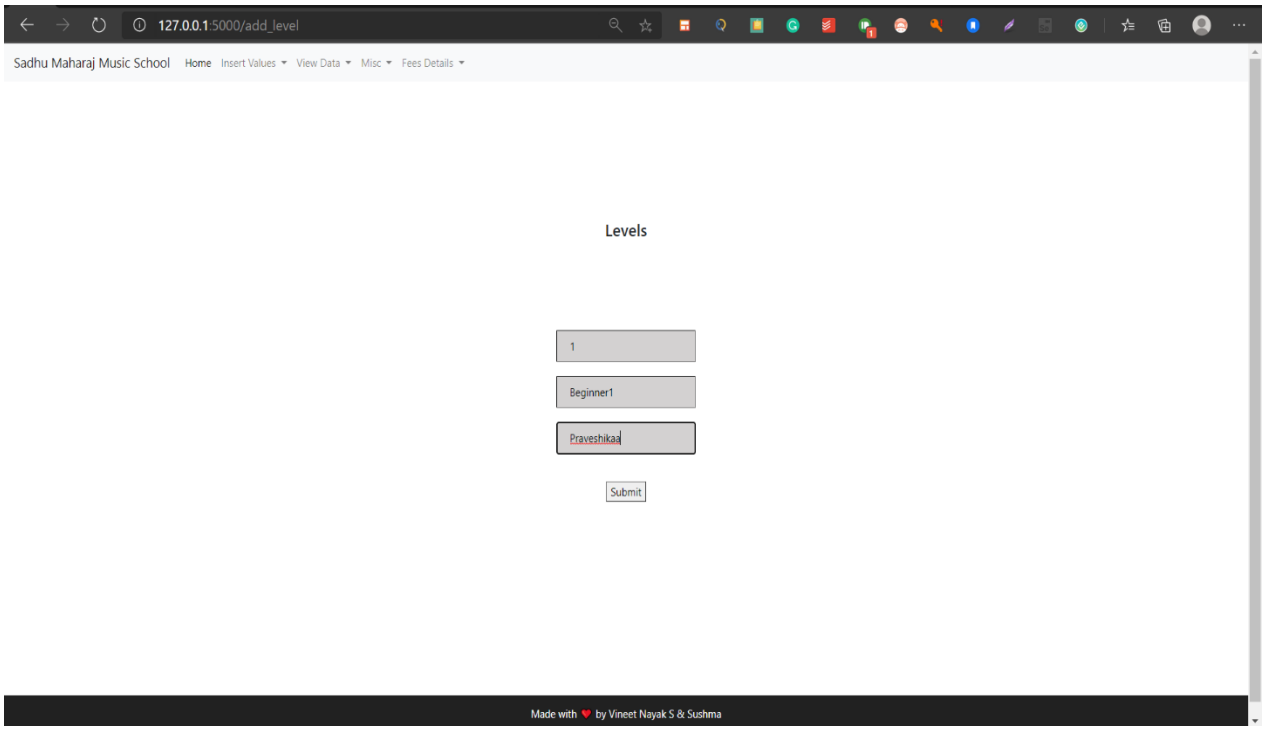


Figure 4.1.2.1: Add Levels Page

4.1.2.2 Add Ragas Data

In this page admin can add details of Ragas.

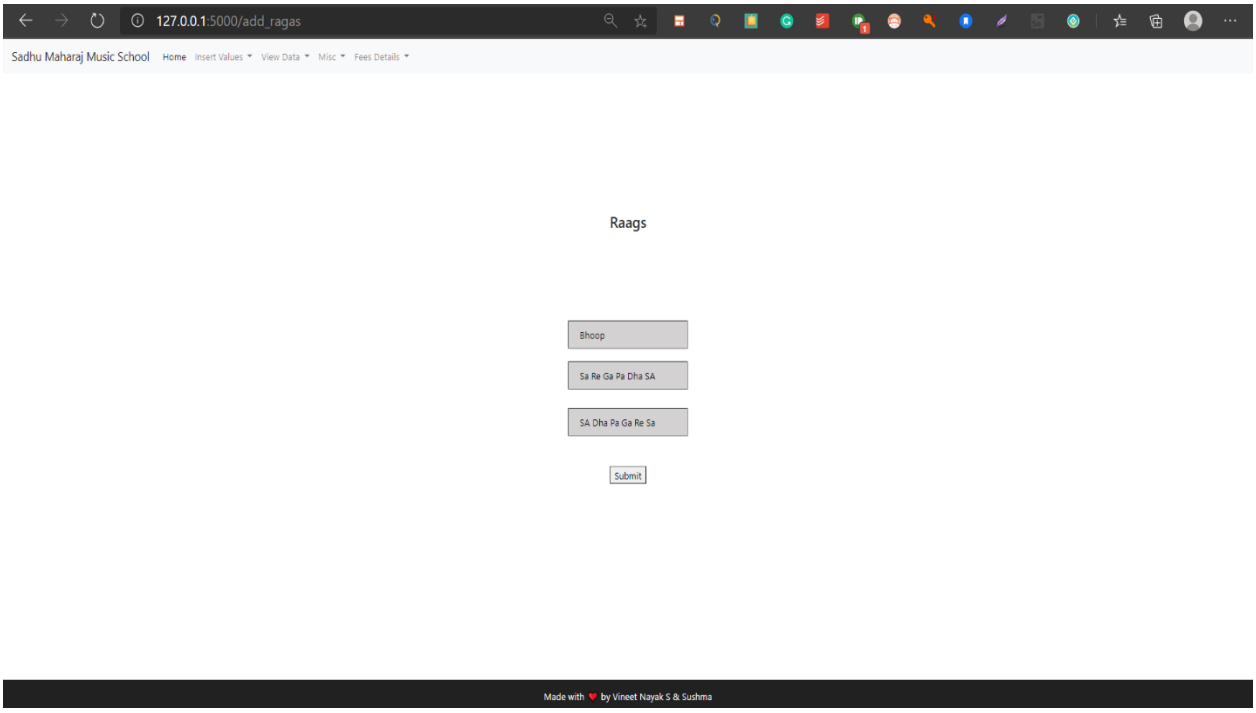


Figure 4.1.2.2: Add Ragas Page

4.1.2.3 Add Student Data

In this page admin can add details of students.

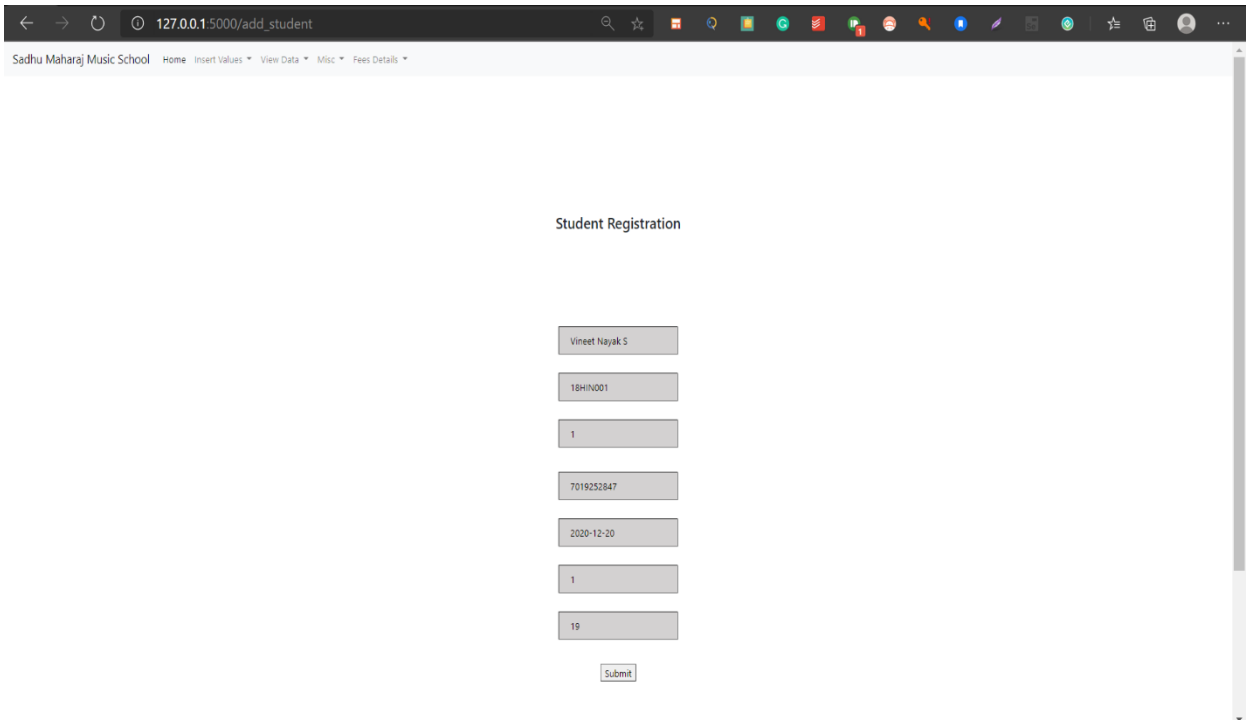


Figure 4.1.2.3: Add Student Details Page

4.1.2.4 Add Level_Ragas Data

In this page admin can add details of Level_Ragas.

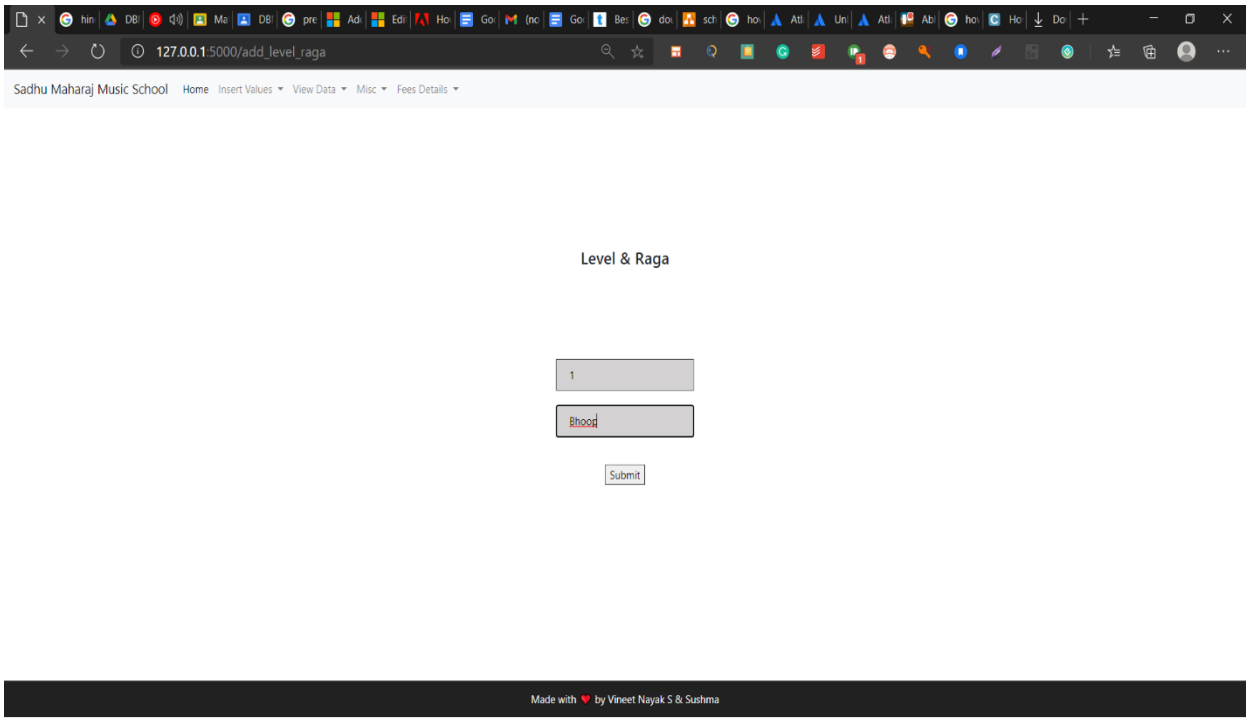


Figure 4.1.2.4: Add Level_Ragas Page

4.1.2.5 Add Events Data

In this page admin can add details of events.

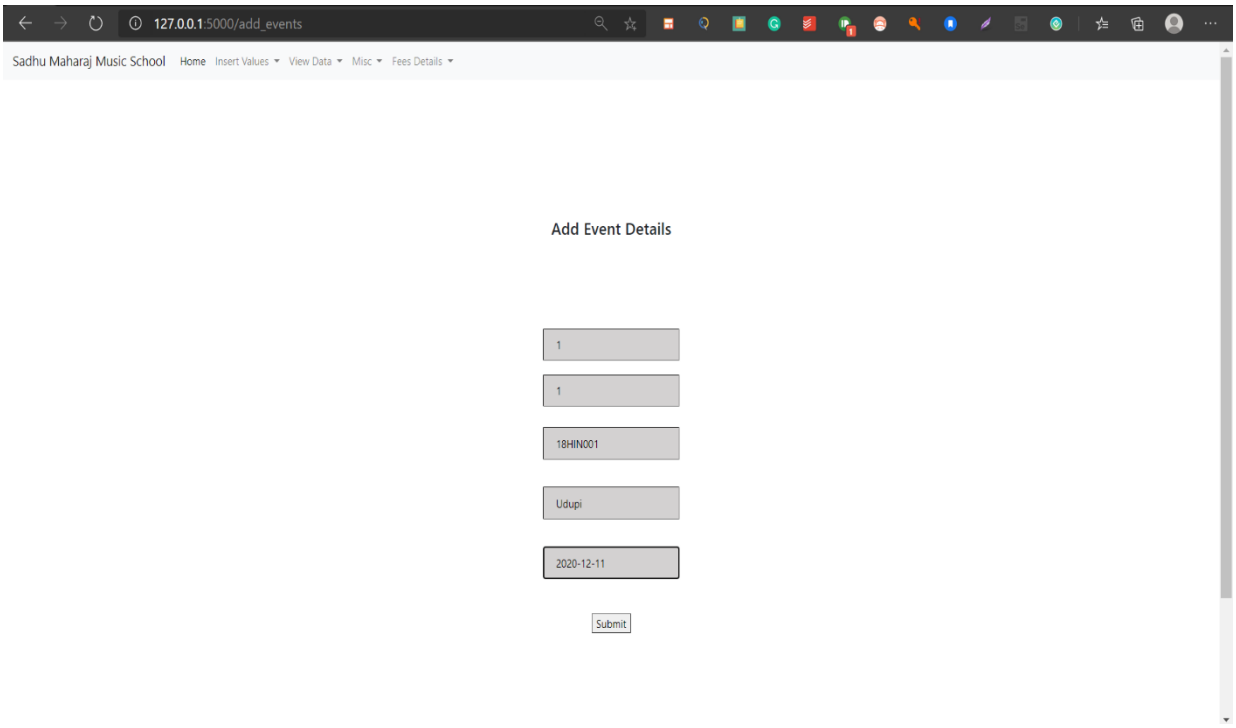


Figure 4.1.2.5: Add Event Details Page

4.1.2.6 Add Examinations Data

In this page admin can add details of examination.

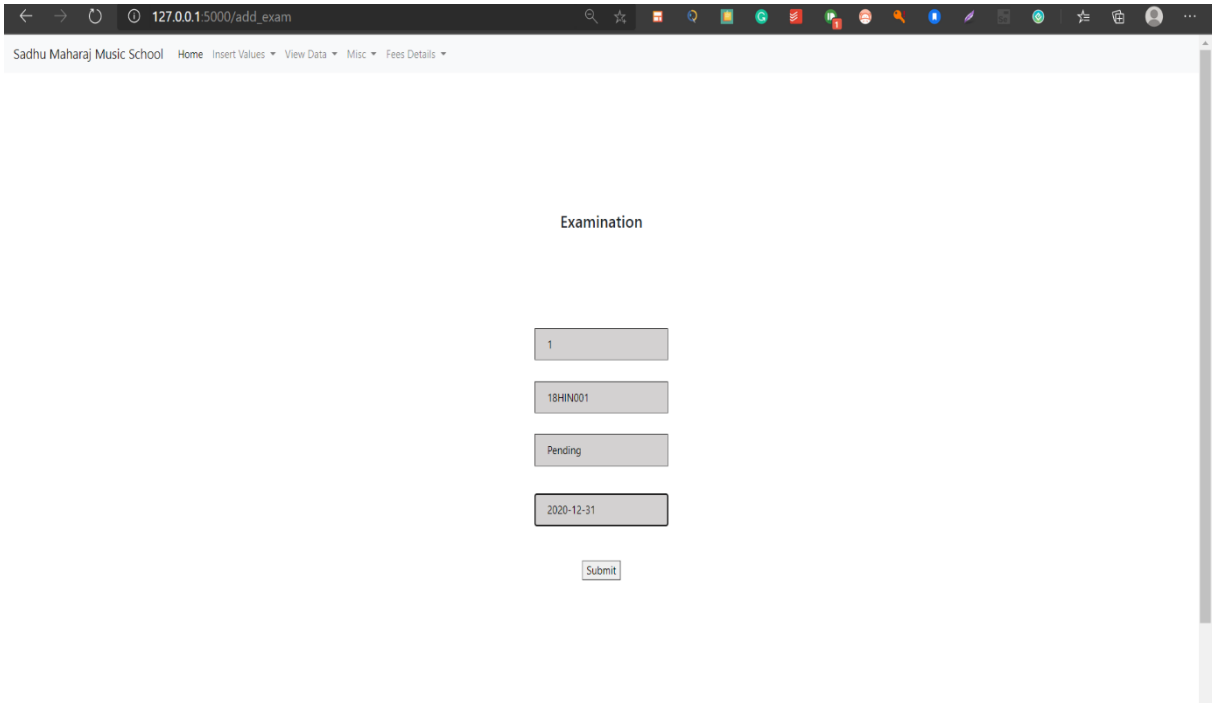


Figure 4.1.2.6: Add Examination Details Page

4.1.2.7 Add Agency Data

In this page admin can add details of agencies.

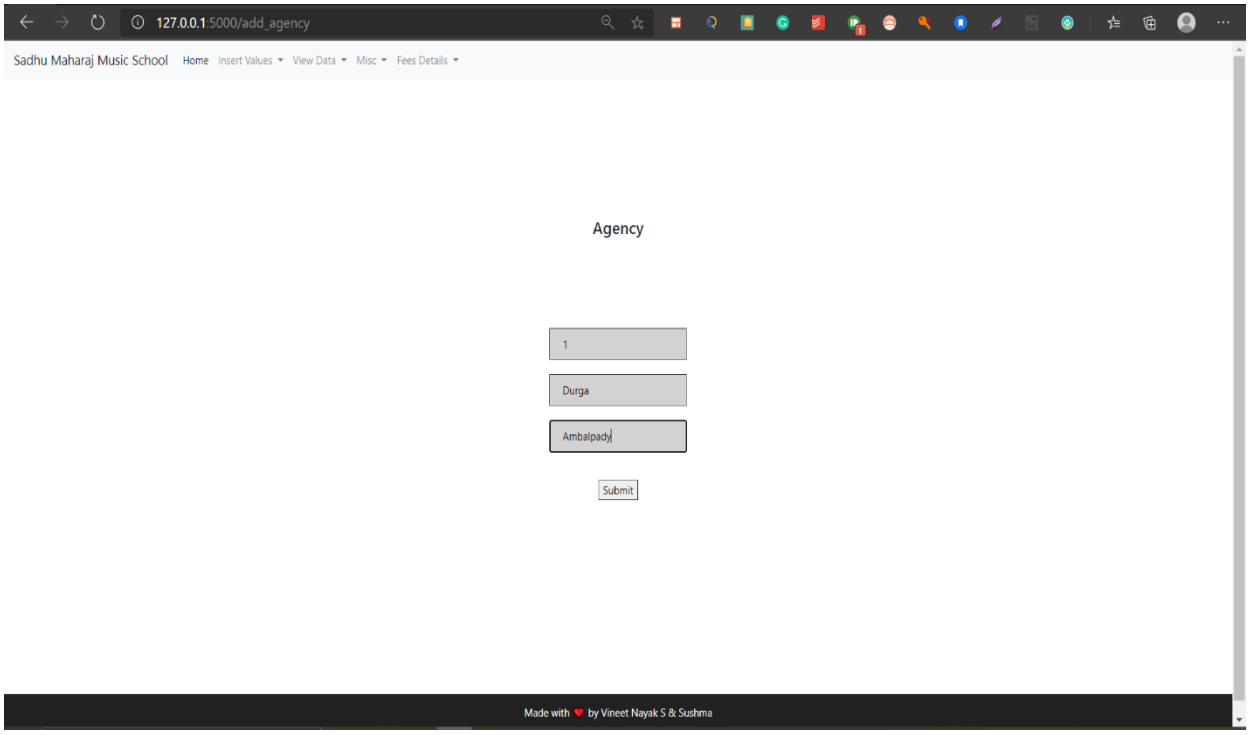


Figure 4.1.2.7: Add Agency Details Page

4.1.2.8 Add Fees Payment Data

In this page admin can add fees payment details.

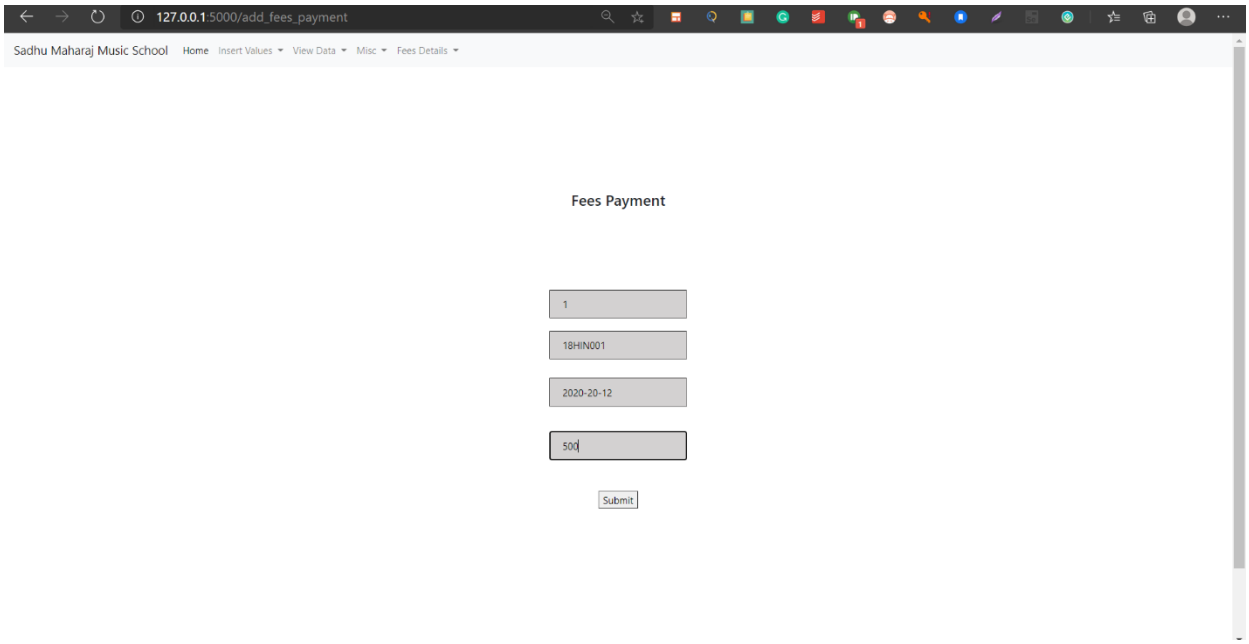


Figure 4.1.2.8: Add Fees Payment Details Page

4.1.2.9 Add Fees Data

In this page admin can add fees details.

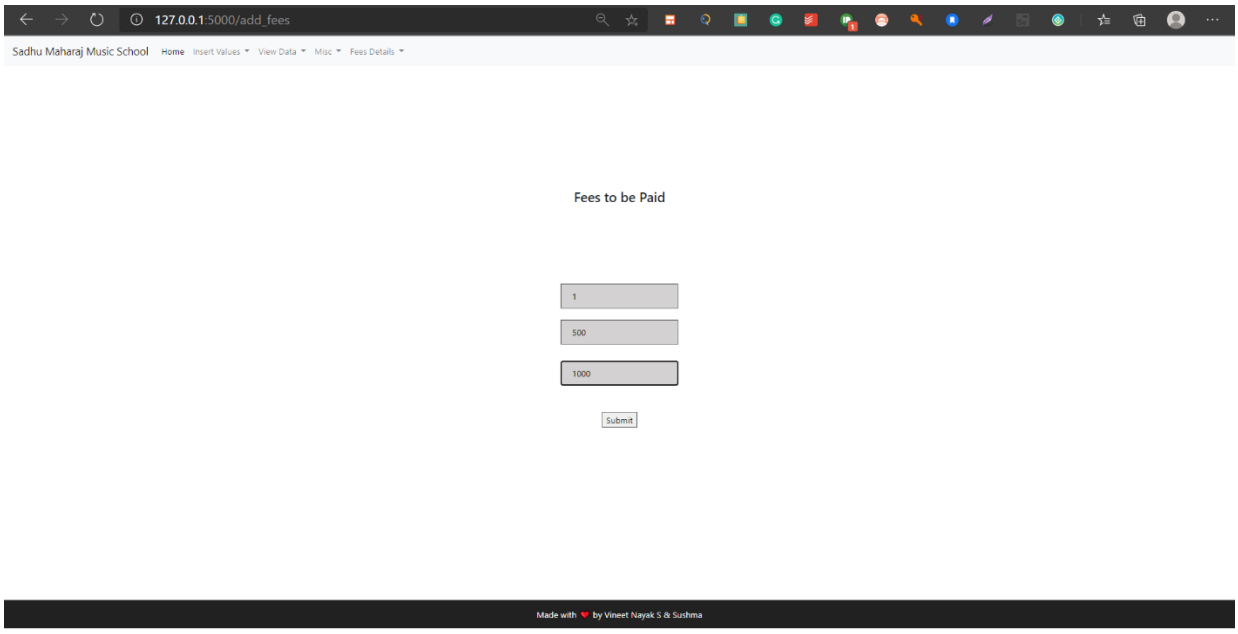


Figure 4.1.2.9: Add Fees Details Page

4.1.3 View Data

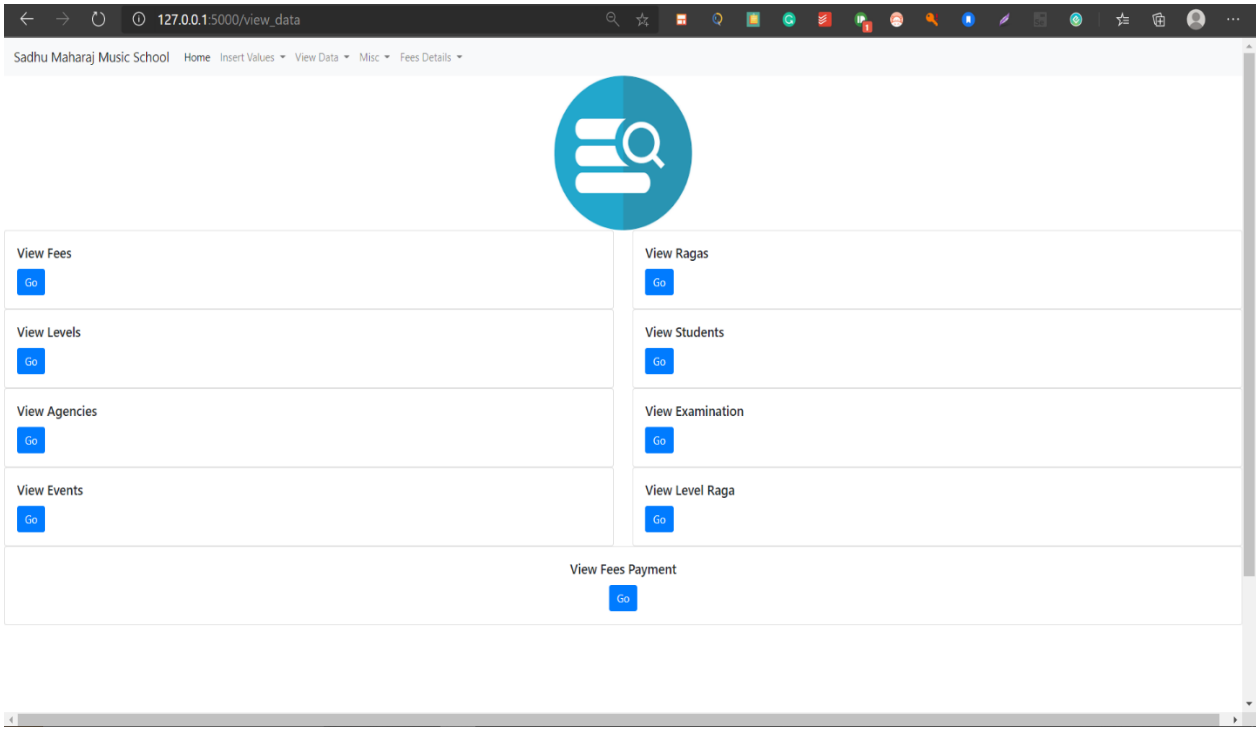
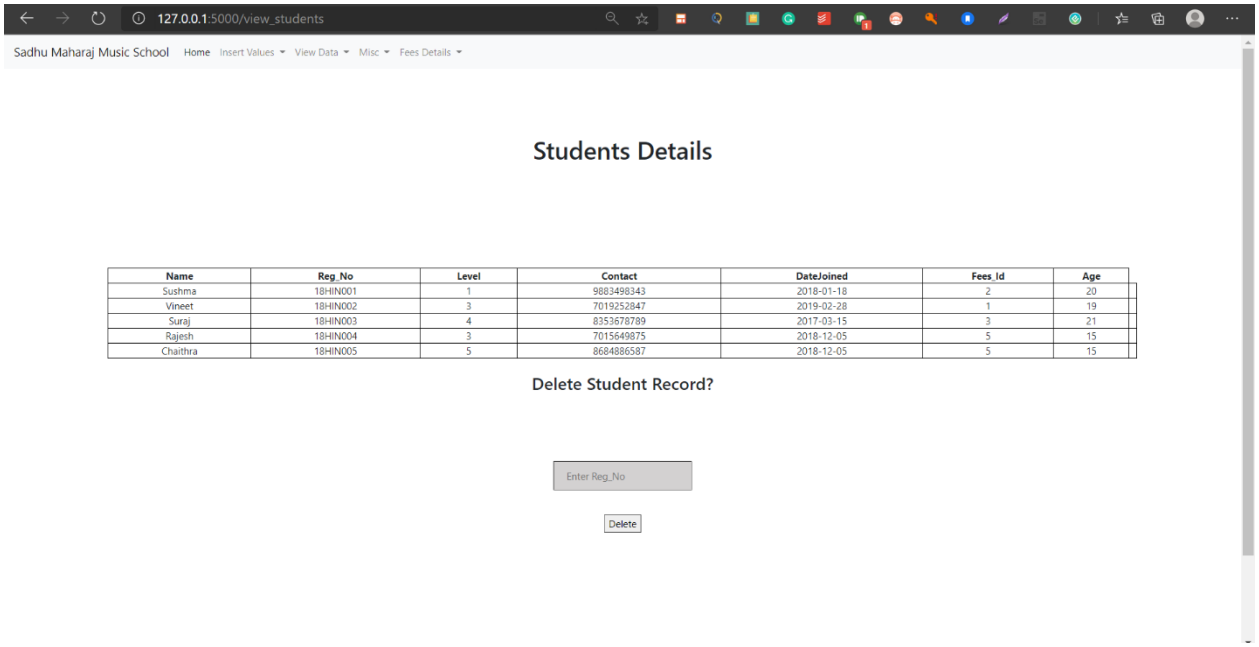


Figure 4.1.3: View Details Page

4.1.3.1 View & Delete Student Data

In this page admin can view and delete student details.

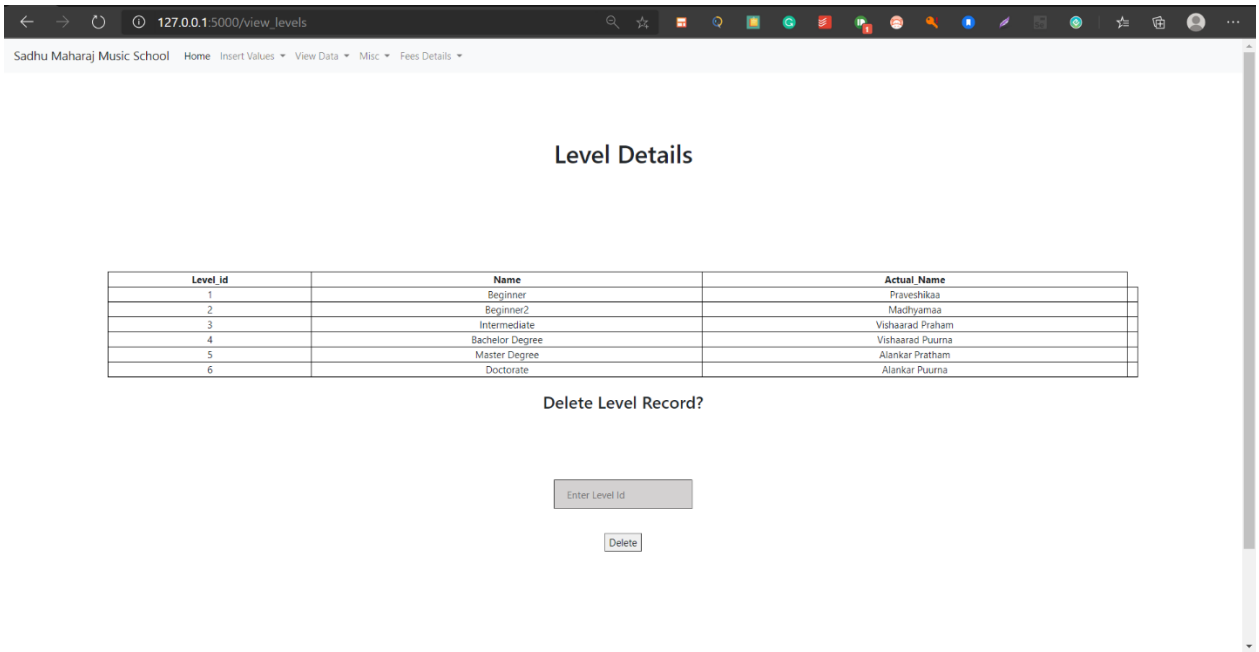


Name	Reg_No	Level	Contact	DateJoined	Fees_Id	Age
Sushma	18HIN001	1	9883498343	2018-01-18	2	20
Vineet	18HIN002	3	7019252847	2019-02-28	1	19
Suraj	18HIN003	4	8353678789	2017-03-15	3	21
Rajesh	18HIN004	3	7015649875	2018-12-05	5	15
Chaithra	18HIN005	5	8684886587	2018-12-05	5	15

Figure 4.1.3.1: View & Delete Student Details Page

4.1.3.2 View & Delete Levels Data

In this page admin can view and delete level details.



Level_id	Name	Actual_Name
1	Beginner	Praveshikaa
2	Beginner2	Madhyamaa
3	Intermediate	Vishaarad Praham
4	Bachelor Degree	Vishaarad Puurna
5	Master Degree	Alankar Pratham
6	Doctorate	Alankar Puurna

Figure 4.1.3.2: View & Delete Levels Page

4.1.3.3 View & Delete Ragas Data

In this page admin can view and delete ragas details.

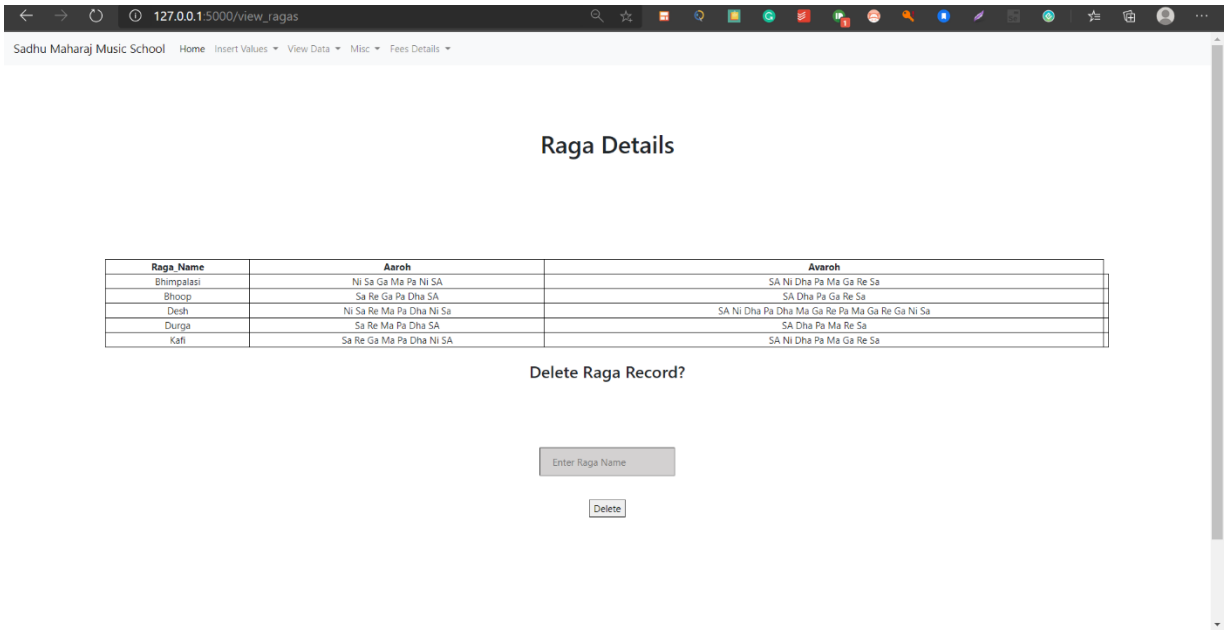


Figure 4.1.3.3: View & Delete Ragas Page

4.1.3.4 View & Delete Events Data

In this page admin can view and delete event details.

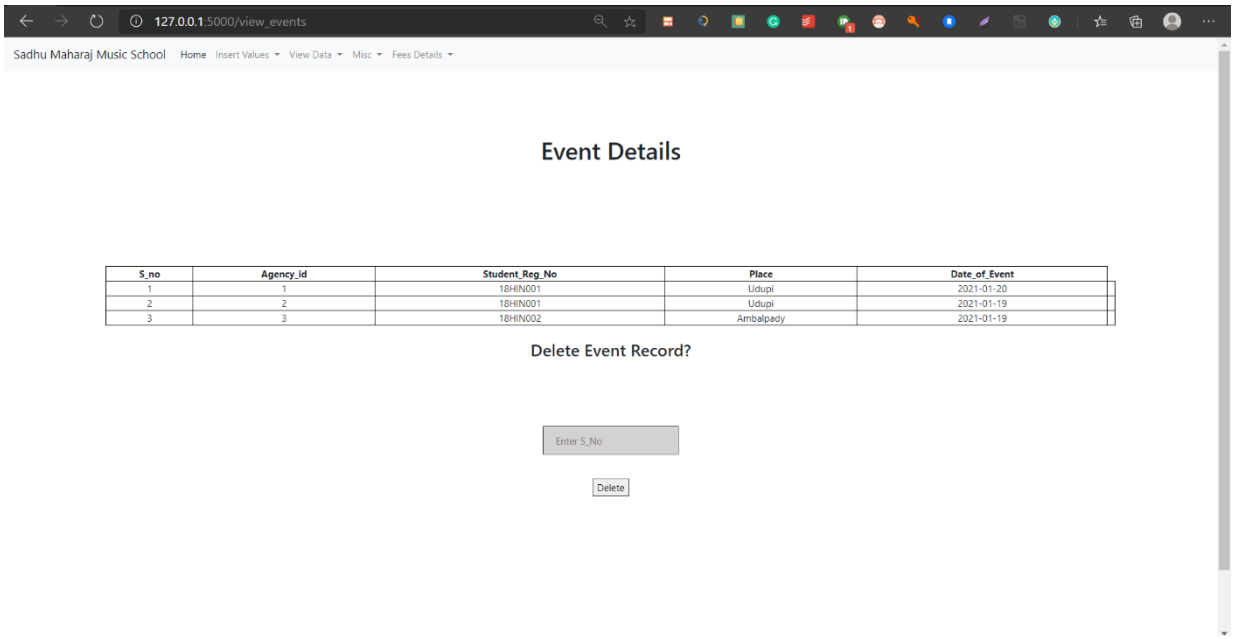


Figure 4.1.3.4: View & Delete Events Page

4.1.3.5 View, Update & Delete Exam Data

In this page admin can view, update & delete exam details.

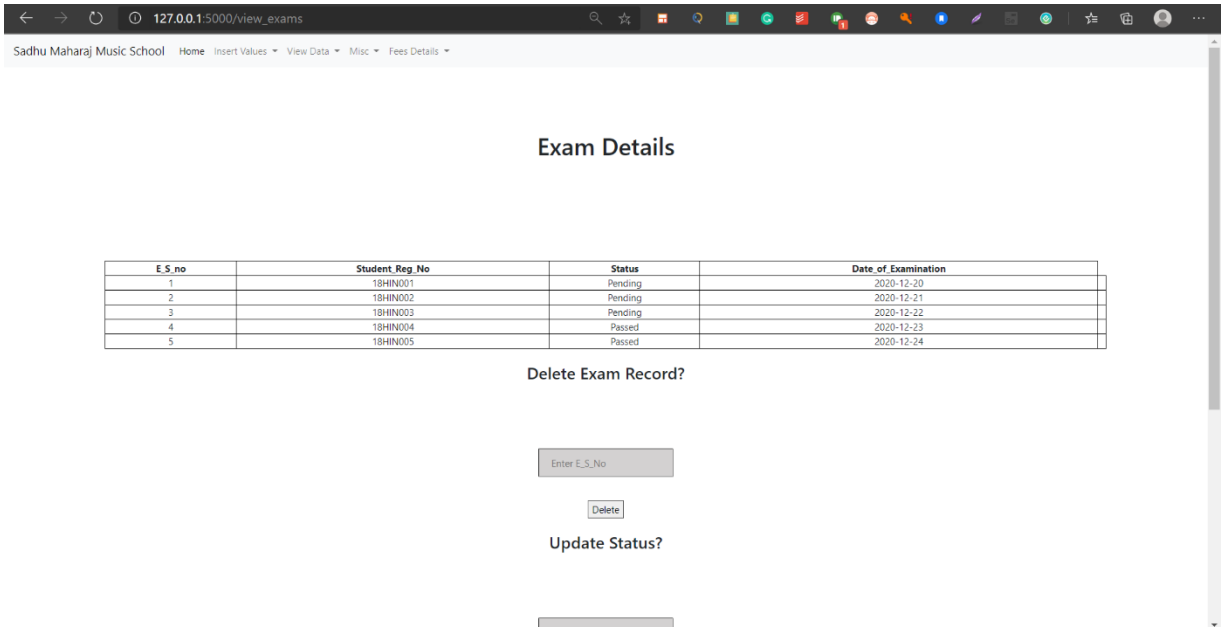


Figure 4.1.3.5.1: View, Update & Delete Exam Page

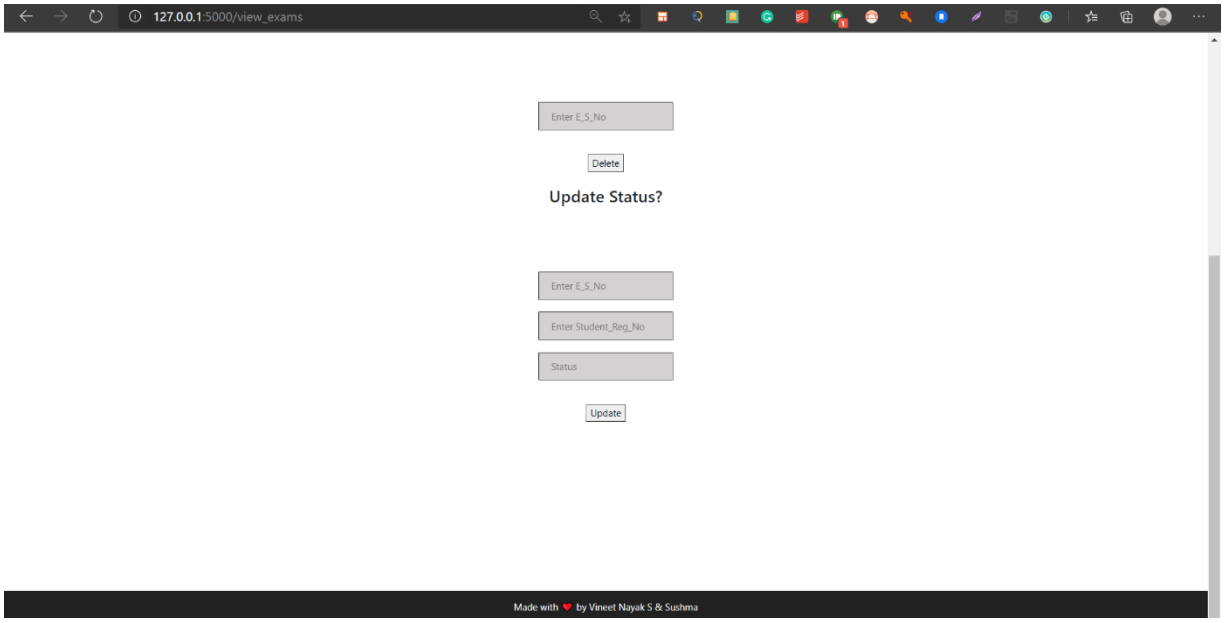


Figure 4.1.3.5.2: View, Update & Delete Exam Page

4.1.3.6 View & Delete Agency Data

In this page admin can view & delete Agency Data.

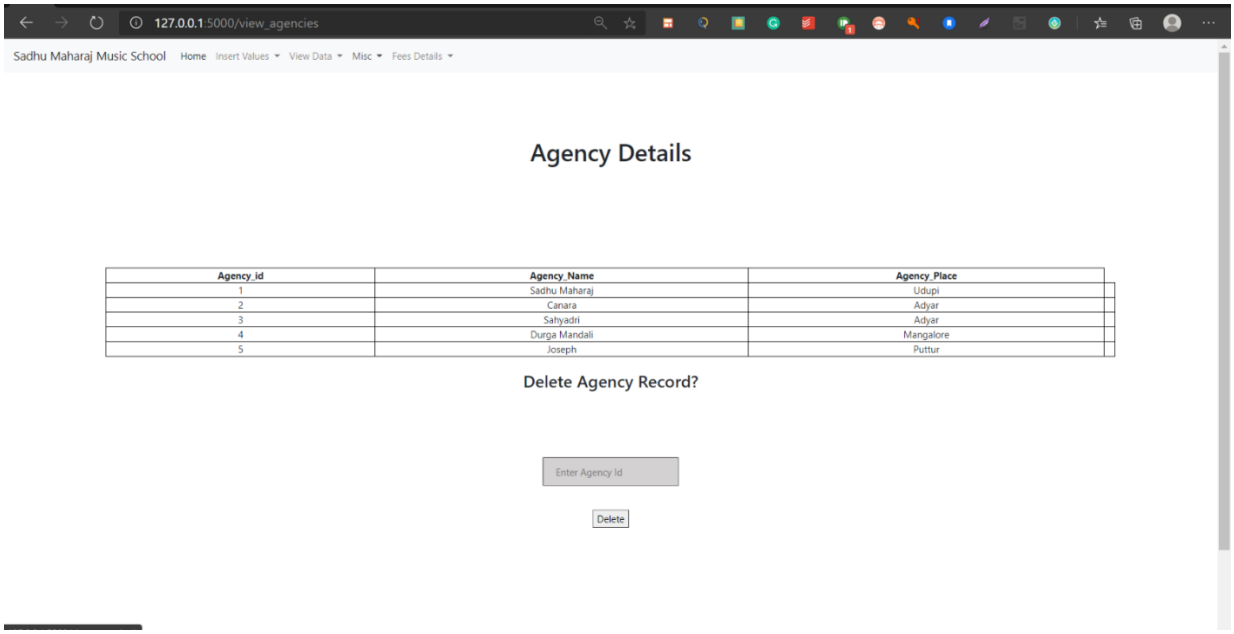


Figure 4.1.3.6: View & Delete Agency Details Page

4.1.3.7 View & Delete Fees Payment Data

In this page admin can view & delete fees payment data and can retrieve individual fees payment history.

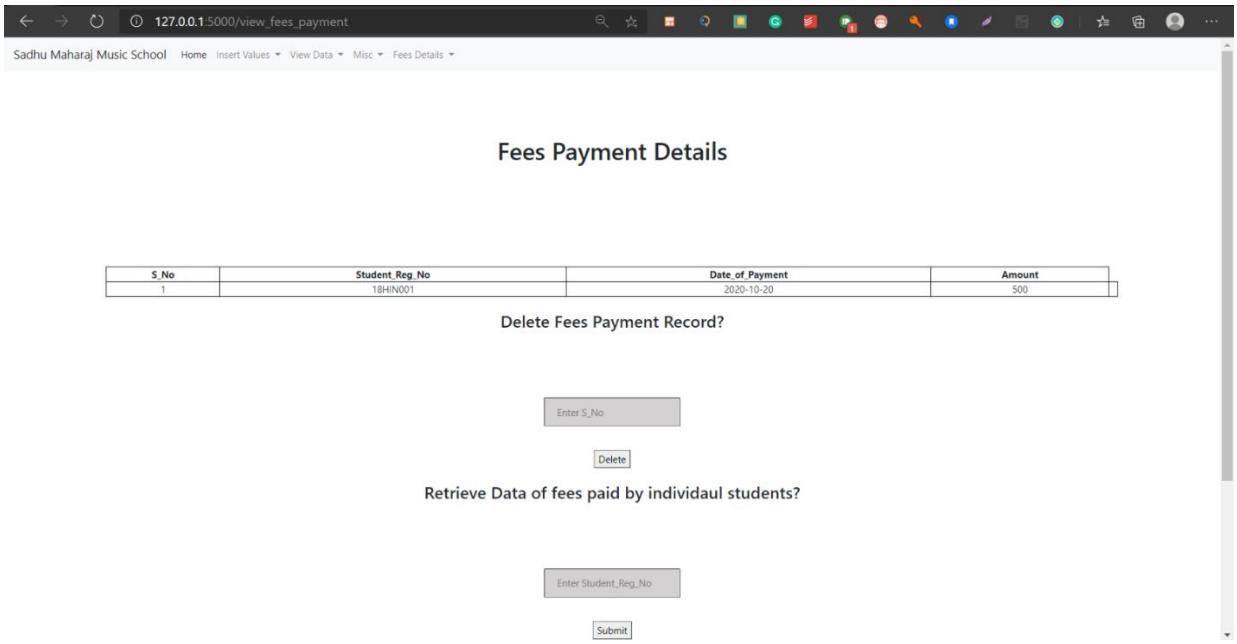


Figure 4.1.3.7: View & Delete Fees Payment Page

4.1.3.8 View & Delete Fees Payment Data

In this page admin can view, update and delete fees details.

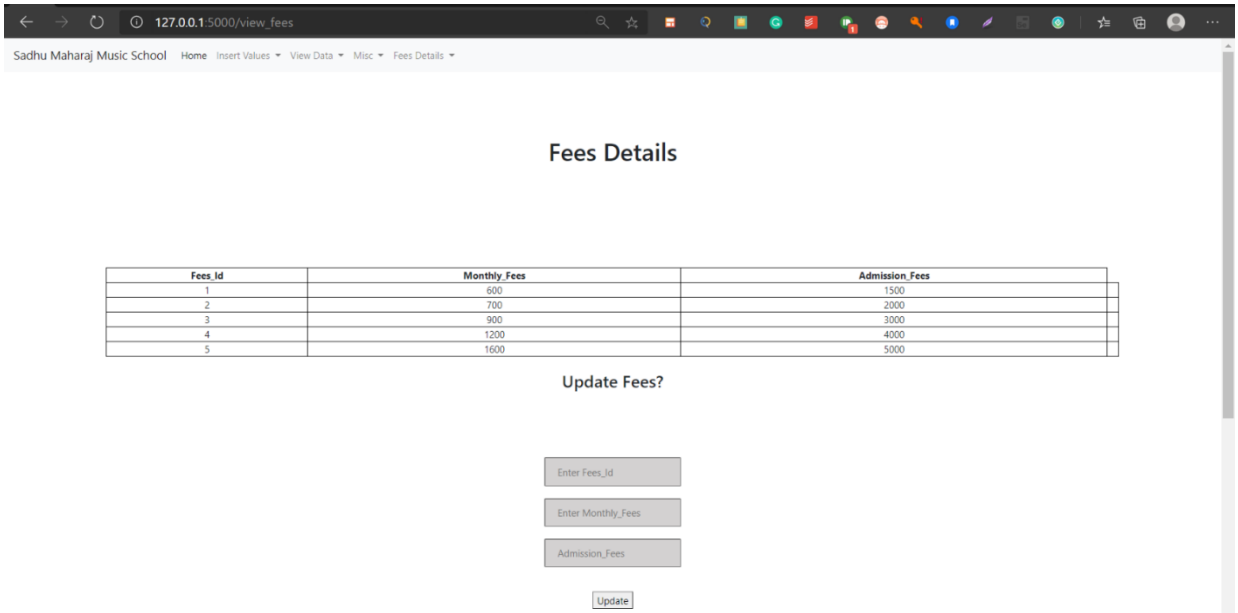


Figure 4.1.3.8.1: View, Update & Delete Fees Payment Page

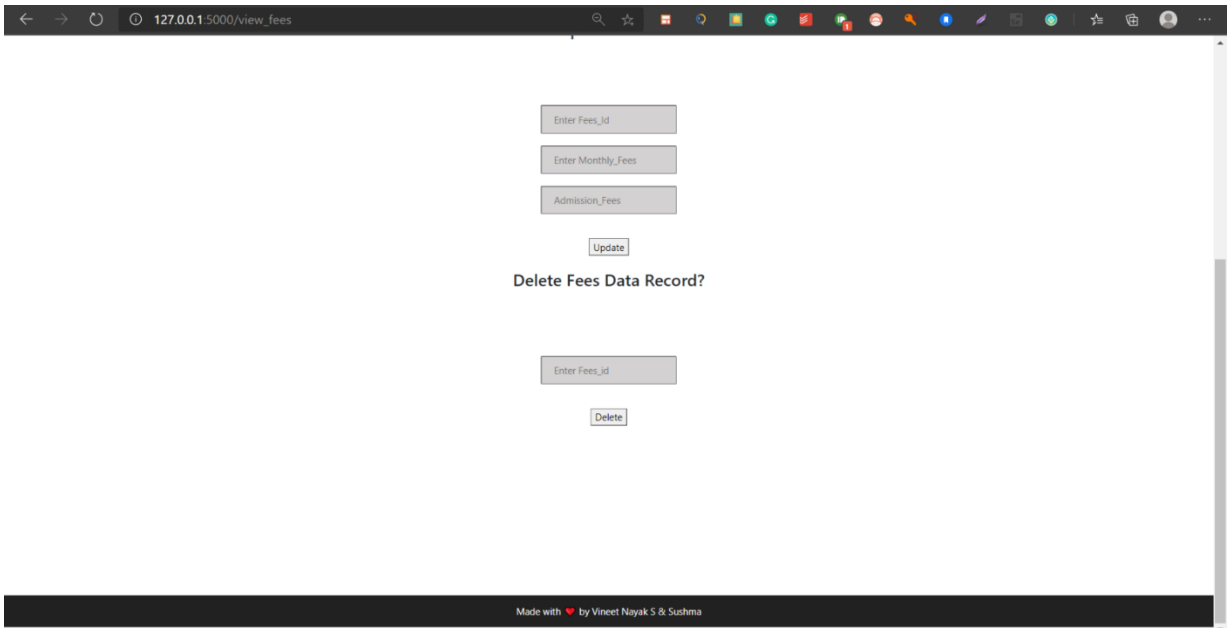


Figure 4.1.3.8.2: View, Update & Delete Fees Payment Page

4.1.4 Miscellaneous Data

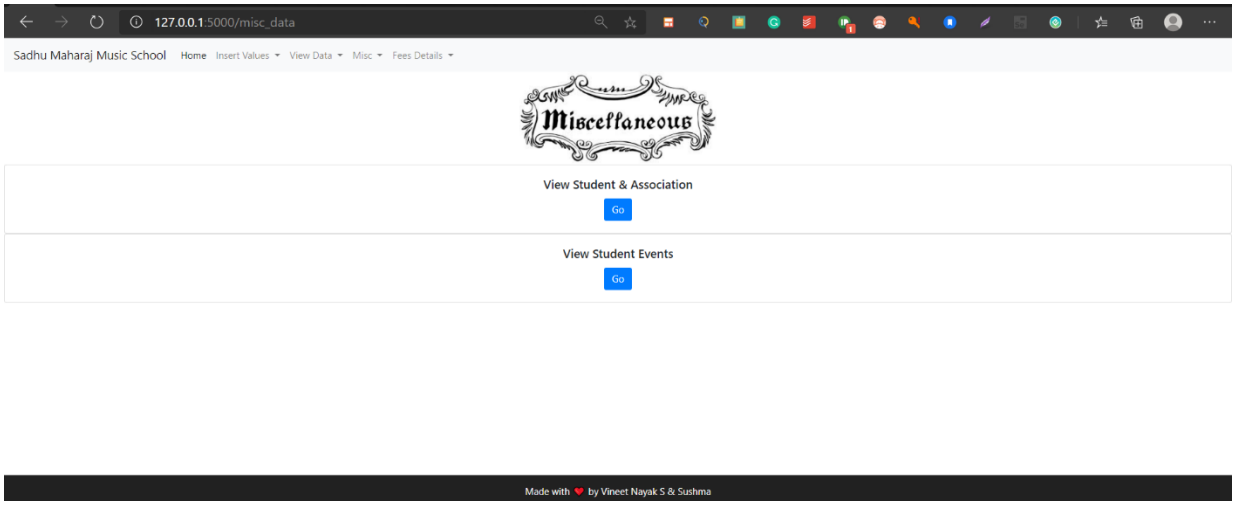
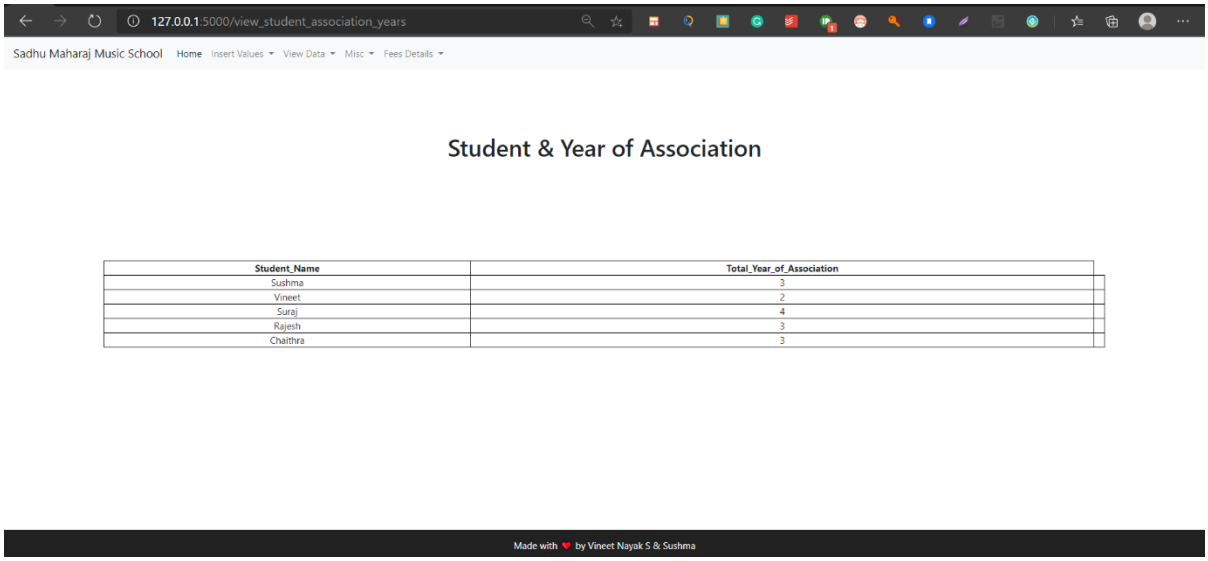


Figure 4.1.4: Miscellaneous Page

4.1.4.1 Student & Association View

Student & Association is the view that is been created in the project. Admin can view the students journey year with the music school.



Student_Name	Total_Year_of_Association
Sushma	3
Vineet	2
Suraj	4
Rajesh	3
Chaithra	3

Figure 4.1.4.1: Student & Association View Page

4.1.4.2 Student & Events

Admin can see the details of the events given by students after joining the music school.

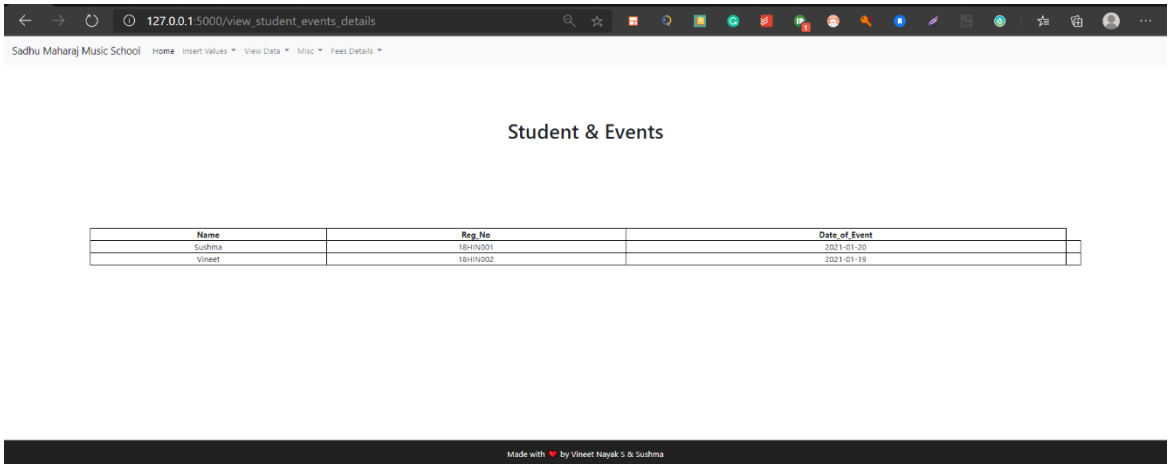


Figure 4.1.4.2: Student & Events View Page

4.1.4.3 Output Status Page

Admin will receive the status for each CRUD operations specially for Insertion, Deletion and Update.

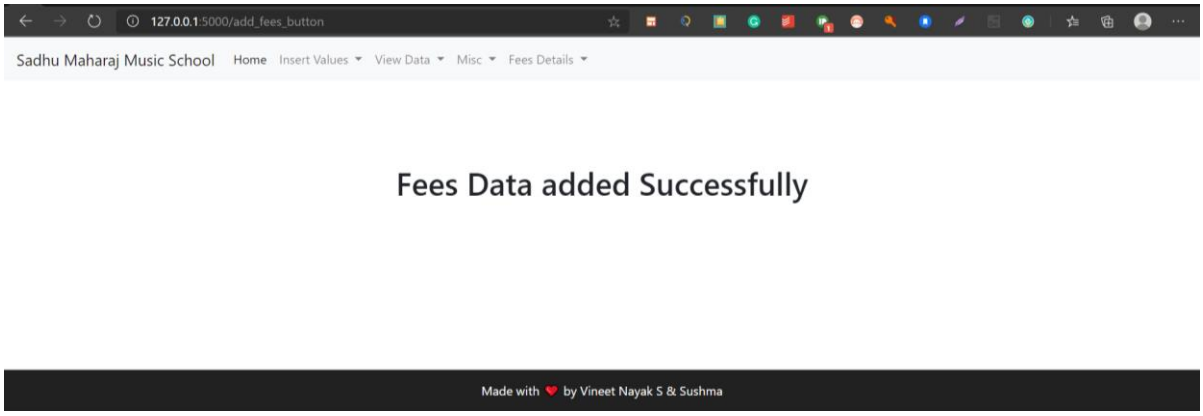


Figure 4.1.4.3.1: Success Status Page

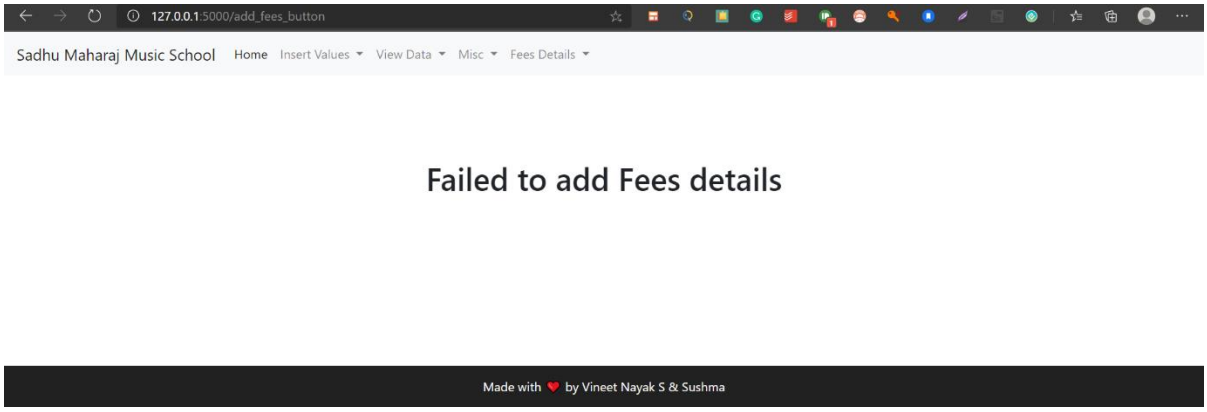


Figure 4.1.4.3.2: Failure Status Page

CONCLUSION

Our project helps administrator of a music school to store the details of students, ragas & levels and retrieve it whenever required. In addition to that it provides a slot to keep track of fees payment history of the students along with event and examination details. While developing the project we have learnt a lot about Flask, MySql and programming the database, and have made it user friendly by hiding the complicated components of it from users. In future this project can be improved by creating a portal for students to register by themselves with implementation of online fees payment.

REFERENCES

1. MySql - (6) MySQL Tutorial for Beginners [Full Course] - YouTube
2. Flask - (6) Python Flask Tutorial: Full-Featured Web App Part 1 - Getting Started - YouTube
3. Python - https://bit.ly/3nQwdZo
4. Frontend - https://bit.ly/39FJ6Aq
5. Flask & MySql - https://bit.ly/3qo9Z2L
6. Reference Slides - https://bit.ly/3nOkMSi