**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.

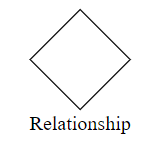
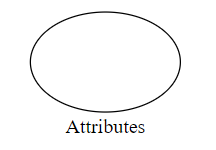
**CHAPTER 2**

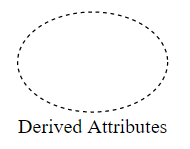
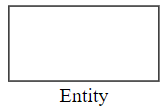
**DESIGN**

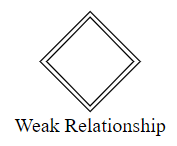
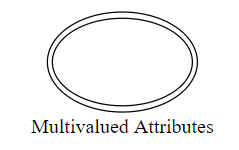
**2.1 ER DIAGRAM**

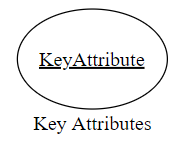
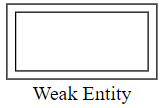
An entity–relationship model describes interrelated things of interest in a specific domain of knowledge. The ER Diagram of our project is shown in the figure:2.1

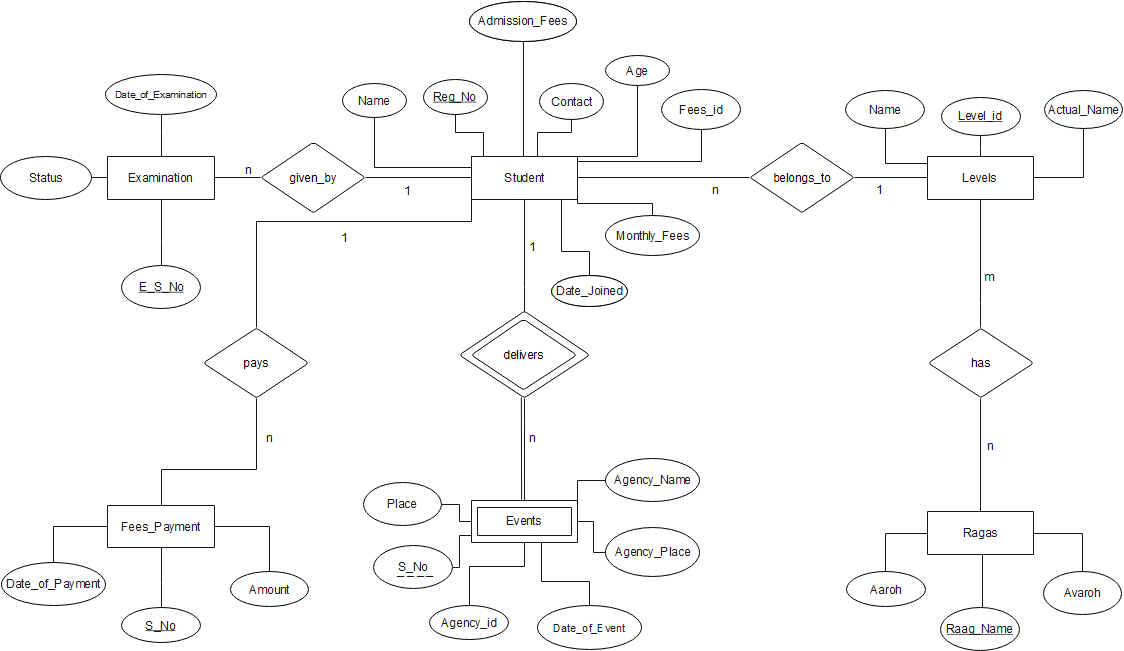
**Symbols in Entity Relationship:**









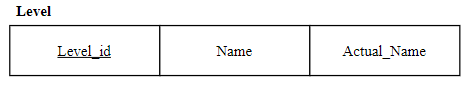


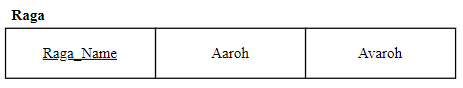
**Figure 2.1:** E-R Diagram

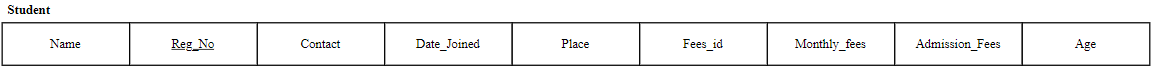
**2.2 RELATIONAL SCHEMA**

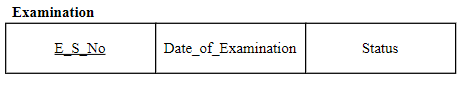
**1. Mapping of Regular Entity Type**

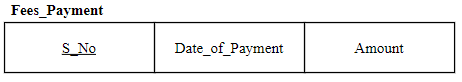
For every regular entity in our entity relationship diagram, we have created a separate relation. These created relations contain the respected attributes and respected primary key.







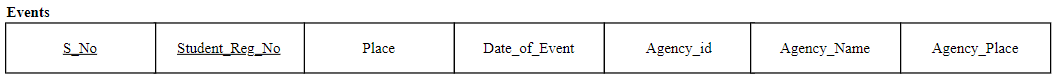




**Figure 2.2.1:** Mapping of Regular Entities

**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.



**Figure 2.2.2 :** Mapping of Weak Entities

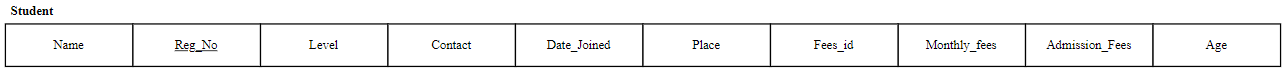
1. **Mapping of Binary 1:1 Relationship Types**

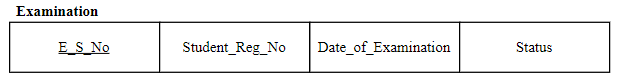
Since our ER Diagram has no binary relationship this step is ignored for our project.

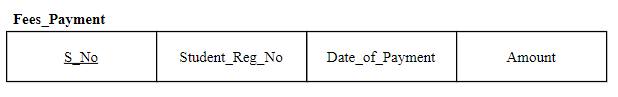
1. **Mapping of Binary 1: N Relationship Types**

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.





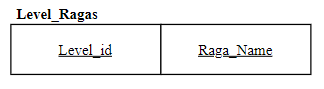


**Figure 2.2.3 :** Mapping of 1:N Relationship Types

1. **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.



**Figure 2.2.4 :** Mapping of M:N Relationship Types

1. **Mapping of Multivalued Attributes**

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

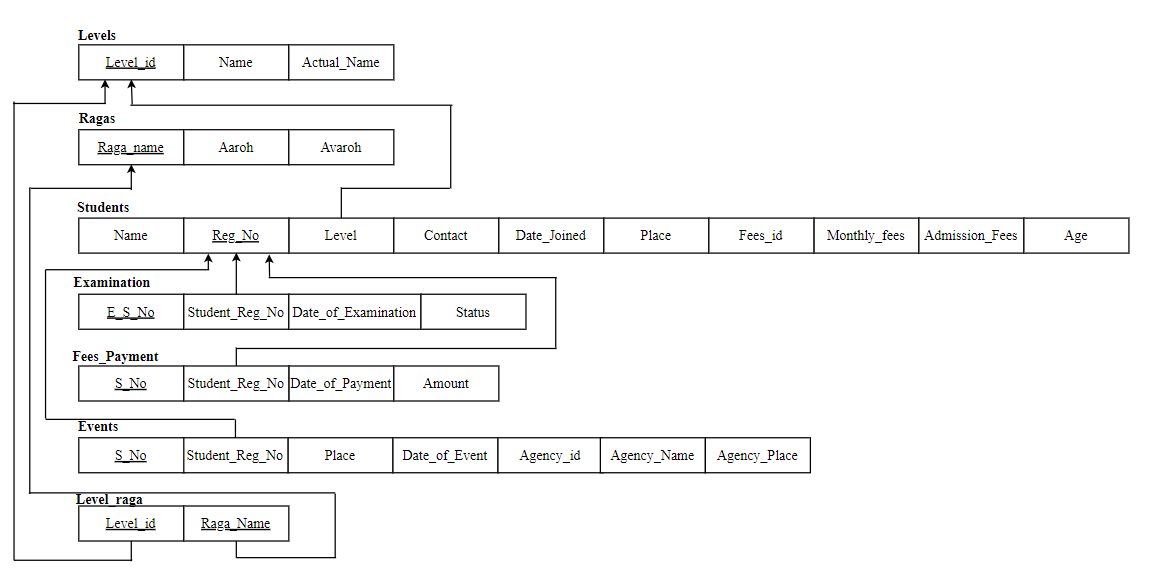
1. **Mapping of N-ary Relationship Types**

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

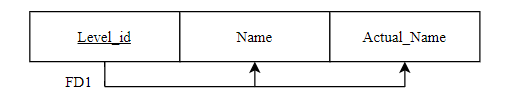
schema diagram only shows us the database design. Relation diagram for our project can be seen in figure 2.3.1.



**Figure 2.3.1:** Schema Diagram

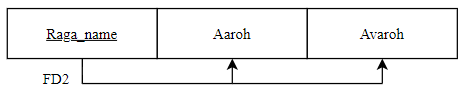
**2.4 NORMALIZATION**

**Levels**



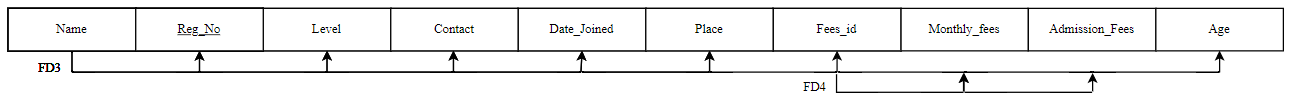
* The above relation is in 1NF because there are no multivalued attributes in the relational schema.
* The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
* The above relation is in 3NF because there is no transitive dependency.

**Ragas**

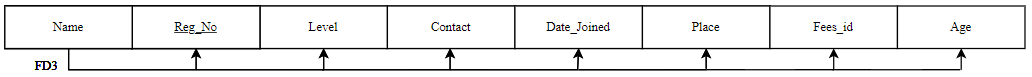


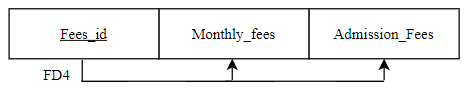
* The above relation is in 1NF because there are no multivalued attributes in the relational schema.
* The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
* The above relation is in 3NF because there is no transitive dependency.

**Students**



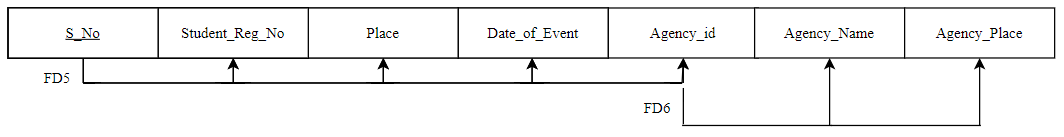
* The above relation is in 1NF because there are no multivalued attributes in the relational schema.
* The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
* The above relation is not in 3NF because there is a transitive dependency.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



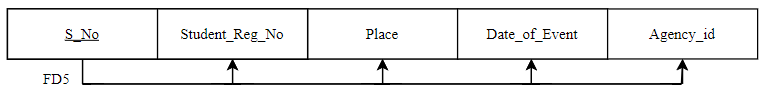


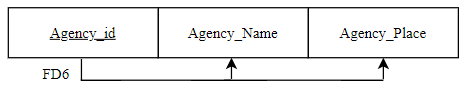
* The above obtained functional dependencies are now in 3NF.

**Events**



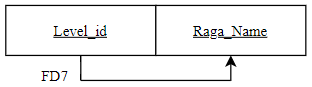
* The above relation is in 1NF because there are no multivalued attributes in the relational schema.
* The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
* The above relation is not in 3NF because there is a transitive dependency.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





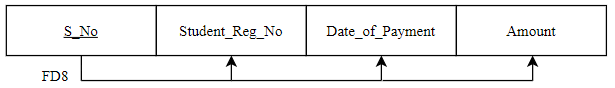
* The above obtained functional dependencies are now in 3NF.

**Level\_Ragas**



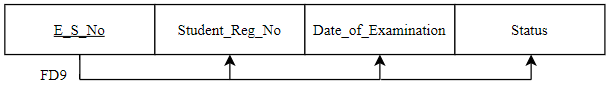
* The above relation is in 1NF because there are no multivalued attributes in the relational schema.
* The above relation is in 2NF because all the attributes in the relational schema are fully functional dependent on the primary key.
* The above relation is in 3NF because there is no transitive dependency.

**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**

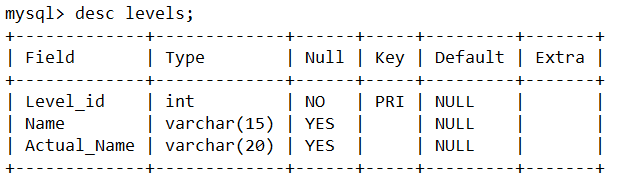
**3.2.1 Levels**

CREATE TABLE LEVELS(

LEVEL\_ID INT PRIMARY KEY,

NAME VARCHAR(15),

ACTUAL\_NAME VARCHAR(20));



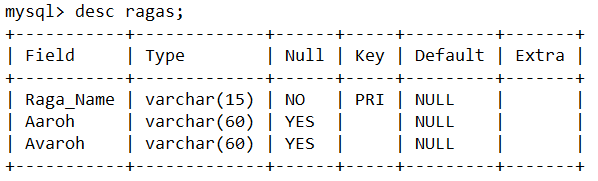
**3.2.2 Ragas**

CREATE TABLE RAGAS(

RAGA\_NAME VARCHAR(15) PRIMARY KEY,

AAROH VARCHAR(60),

AVAROH VARCHAR(60));



**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,

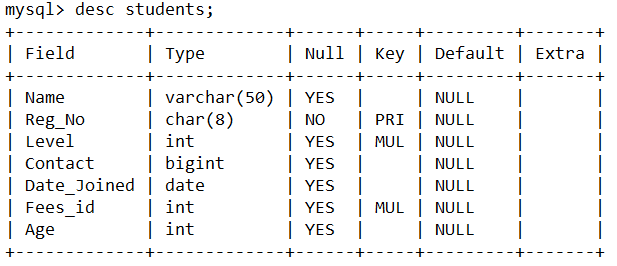
DATE\_JOINED DATE,

FEES\_ID INT,

AGE INT,

FOREIGN KEY(LEVEL) REFERENCES LEVELS(LEVEL\_ID),

FOREIGN KEY(FEES\_ID) REFERENCES FEES(FEES\_ID));



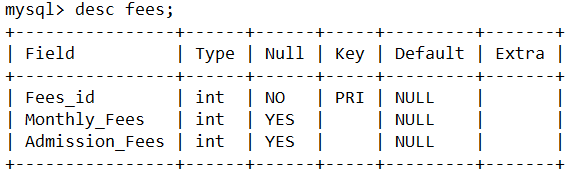
**3.2.4 Fees**

CREATE TABLE FEES(

FEES\_ID INT PRIMARY KEY,

MONTHLY\_FEES INT,

ADMISSION\_FEES INT);



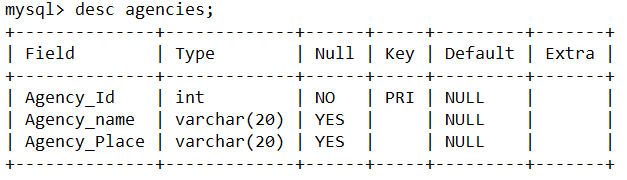
**3.2.5 Agencies**

CREATE TABLE AGENCIES(

AGENCY\_ID INT PRIMARY KEY,

AGENCY\_NAME VARCHAR(20),

AGENCY\_PLACE VARCHAR(20));



**3.2.6 Events**

CREATE TABLE EVENTS(

S\_NO INT PRIMARY KEY,

AGENCY\_ID INT,

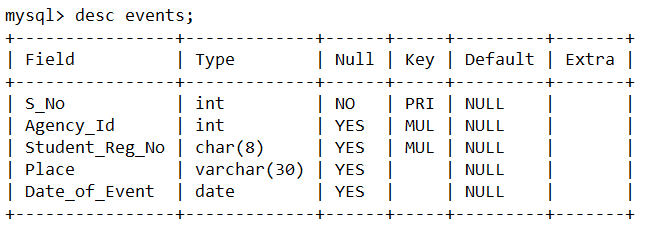
STUDENT\_REG\_NO CHAR(8),

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);



**3.2.7 Level\_Ragas**

CREATE TABLE LEVEL\_RAGAS(

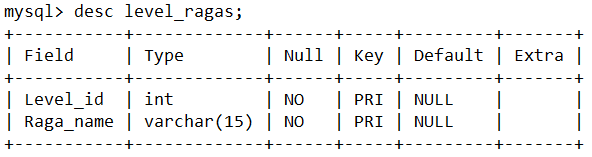
LEVEL\_ID INT,

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);



**3.2.8 FEES\_PAYMENT**

CREATE TABLE FEES\_PAYMENT(

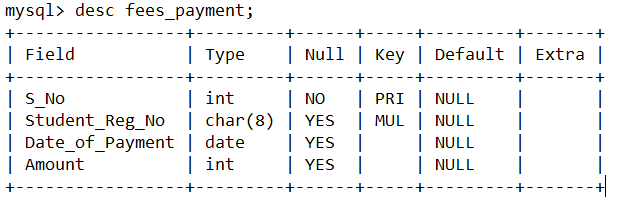
S\_NO INT PRIMARY KEY,

STUDENT\_REG\_NO CHAR(8),

DATE\_OF\_PAYMENT DATE,

AMOUNT INT,

FOREIGN KEY(STUDENT\_REG\_NO) REFERENCES STUDENTS ON DELETE SET CASCADE);



**3.2.9 EXAMINATION**

CREATE TABLE EXAMINATION(

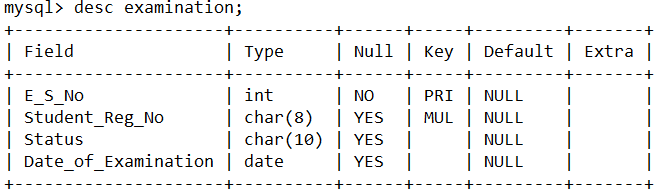
E\_S\_NO INT PRIMARY KEY,

STUDENT\_REG\_NO CHAR(8),

STATUS CHAR(10),

DATE\_OF\_EXAMINATION DATE,

FOREIGN KEY(STUDENT\_REG\_NO) REFERENCES STUDENTS ON DELETE SET CASCADE);



**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**

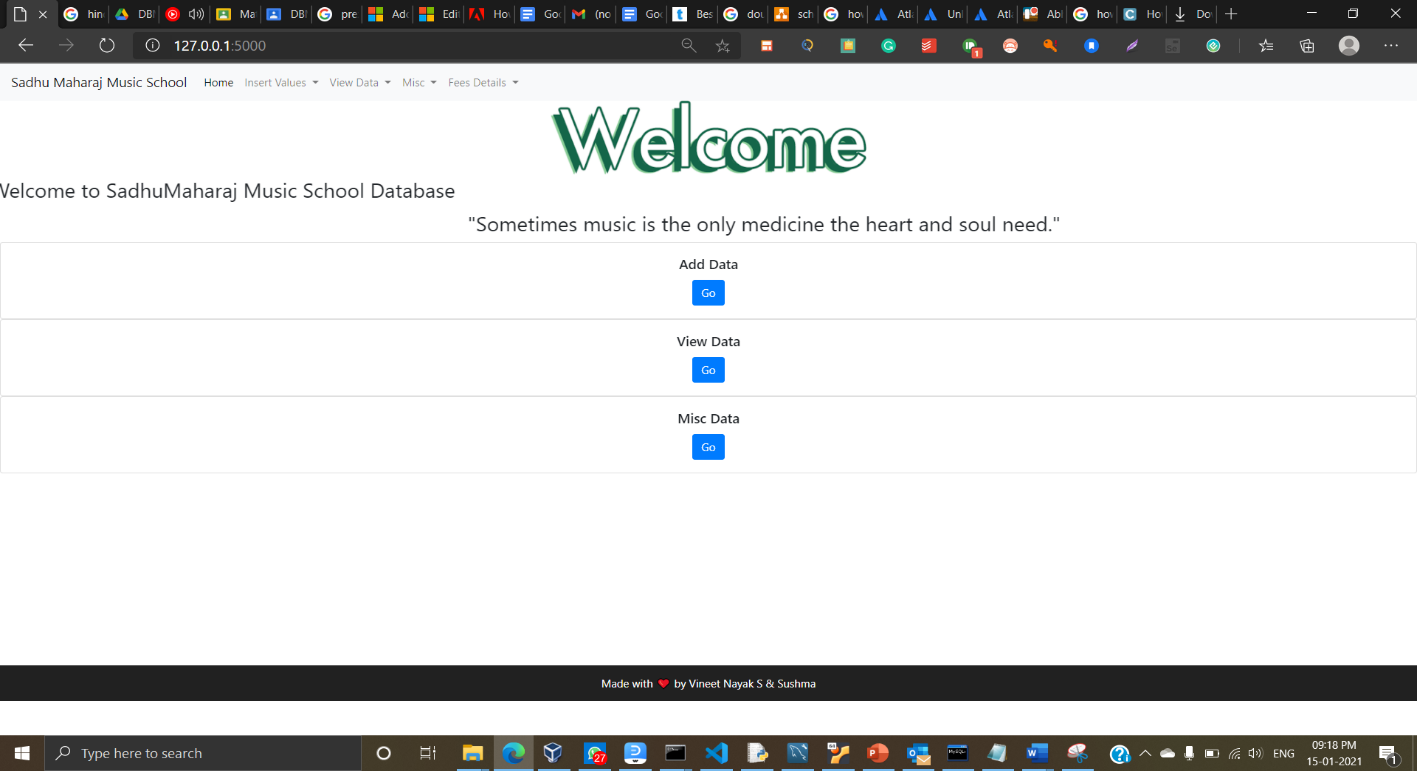


Figure 4.1.1: Home Page

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

**4.1.2 Add Data**

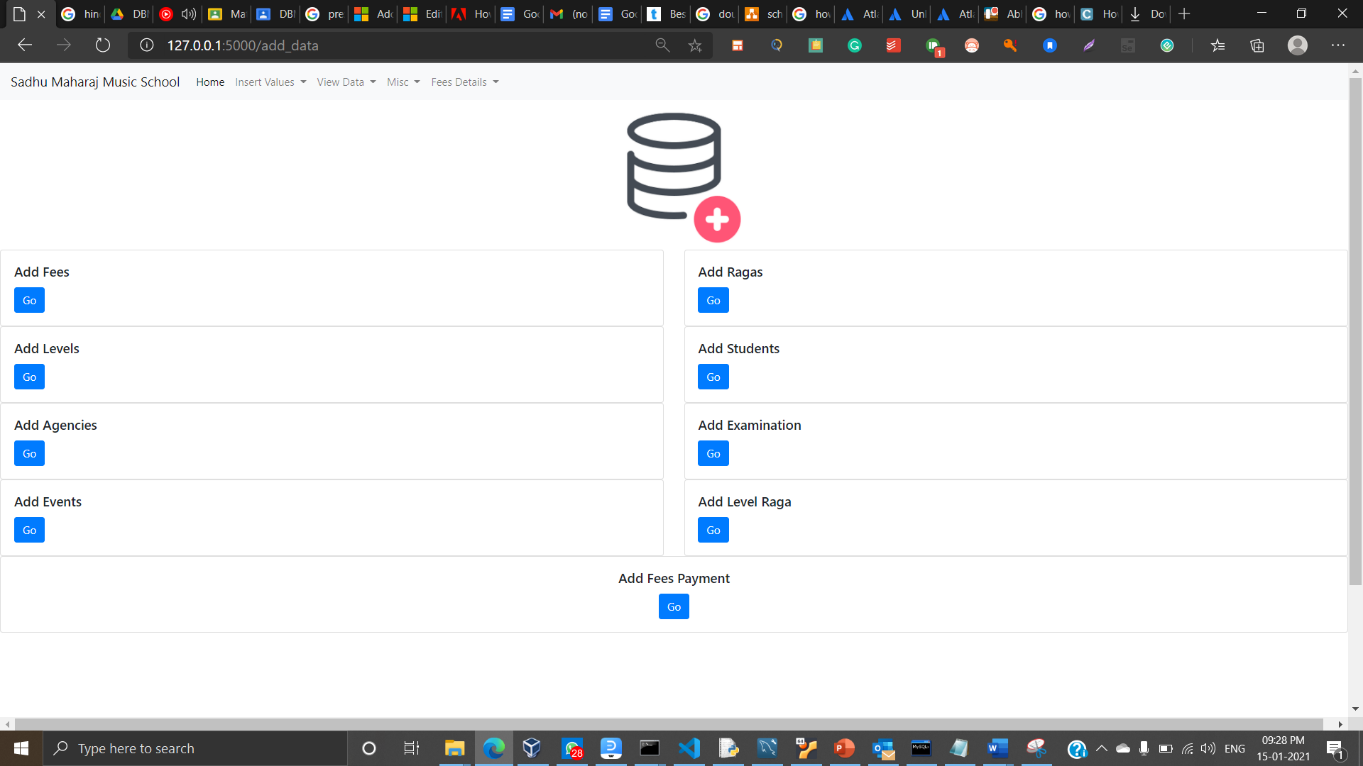


Figure 4.1.2 Add Data Page

**4.1.2.1 Add Level Data**

In this page admin can add details of Levels.

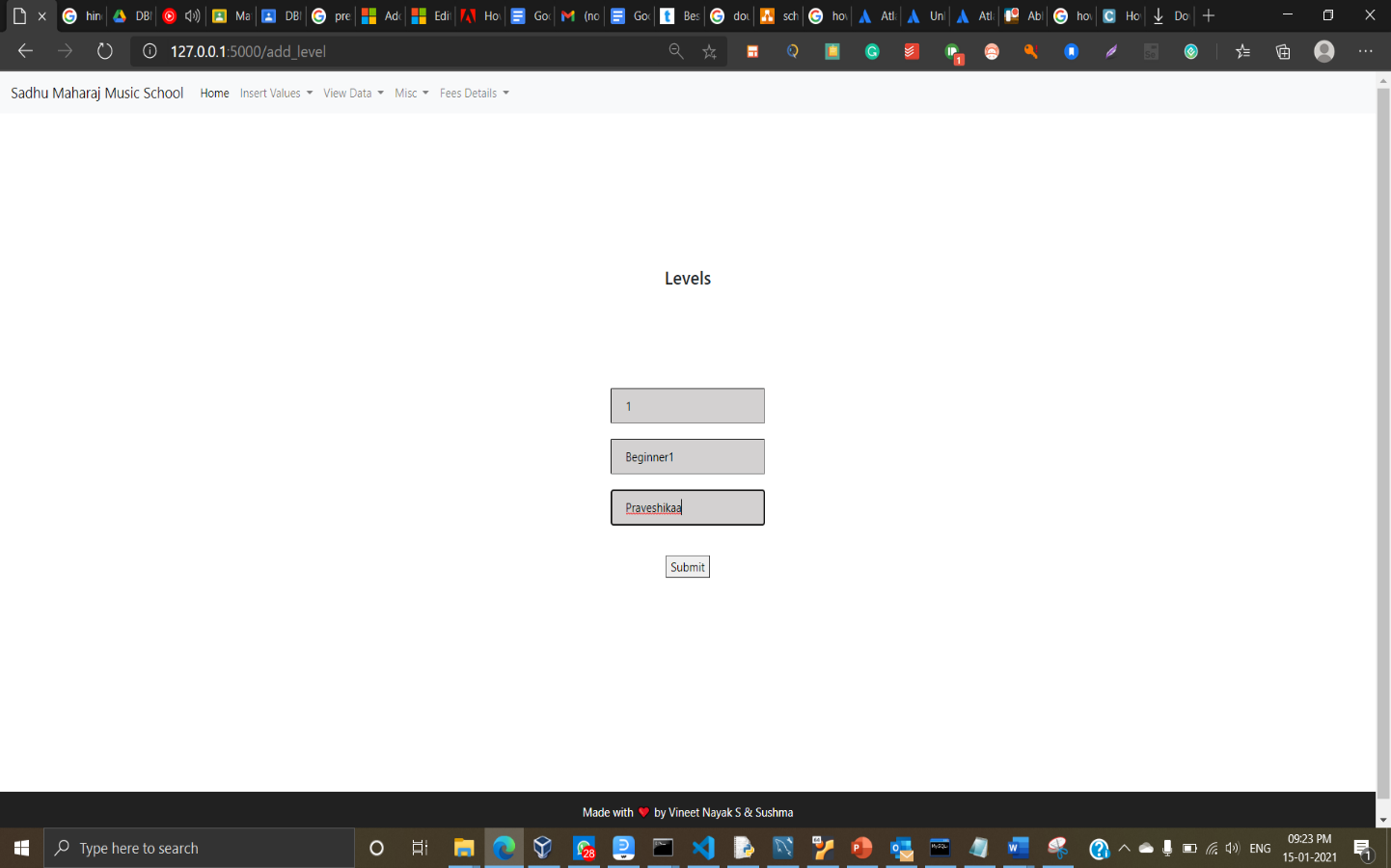


Figure 4.1.2.1: Add Levels Page

**4.1.2.2 Add Ragas Data**

In this page admin can add details of Ragas.

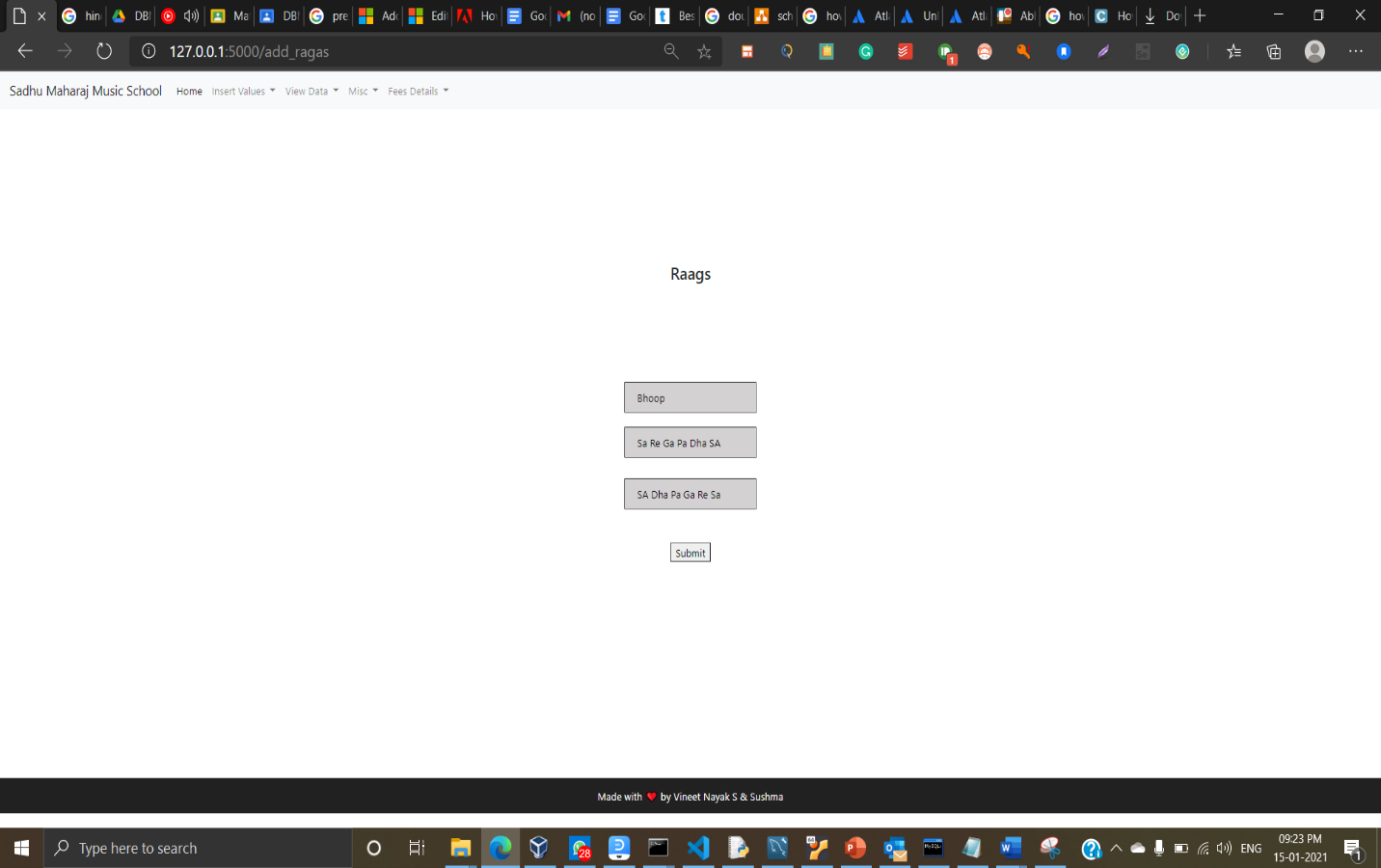


Figure 4.1.2.2: Add Ragas Page

**4.1.2.3 Add Student Data**

In this page admin can add details of students.

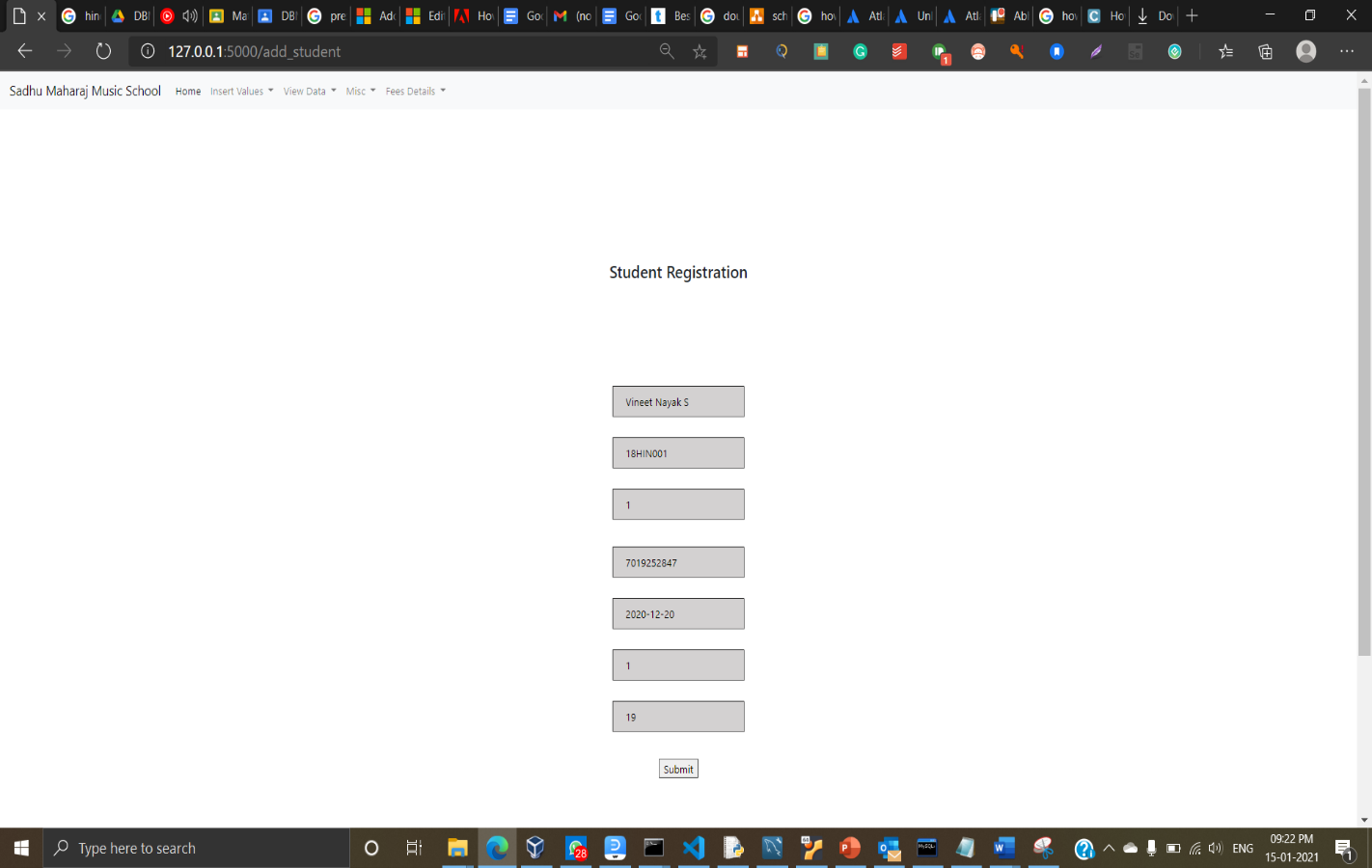


Figure 4.1.2.3: Add Student Details Page

**4.1.2.4 Add Level\_Ragas Data**

In this page admin can add details of Level\_Ragas.

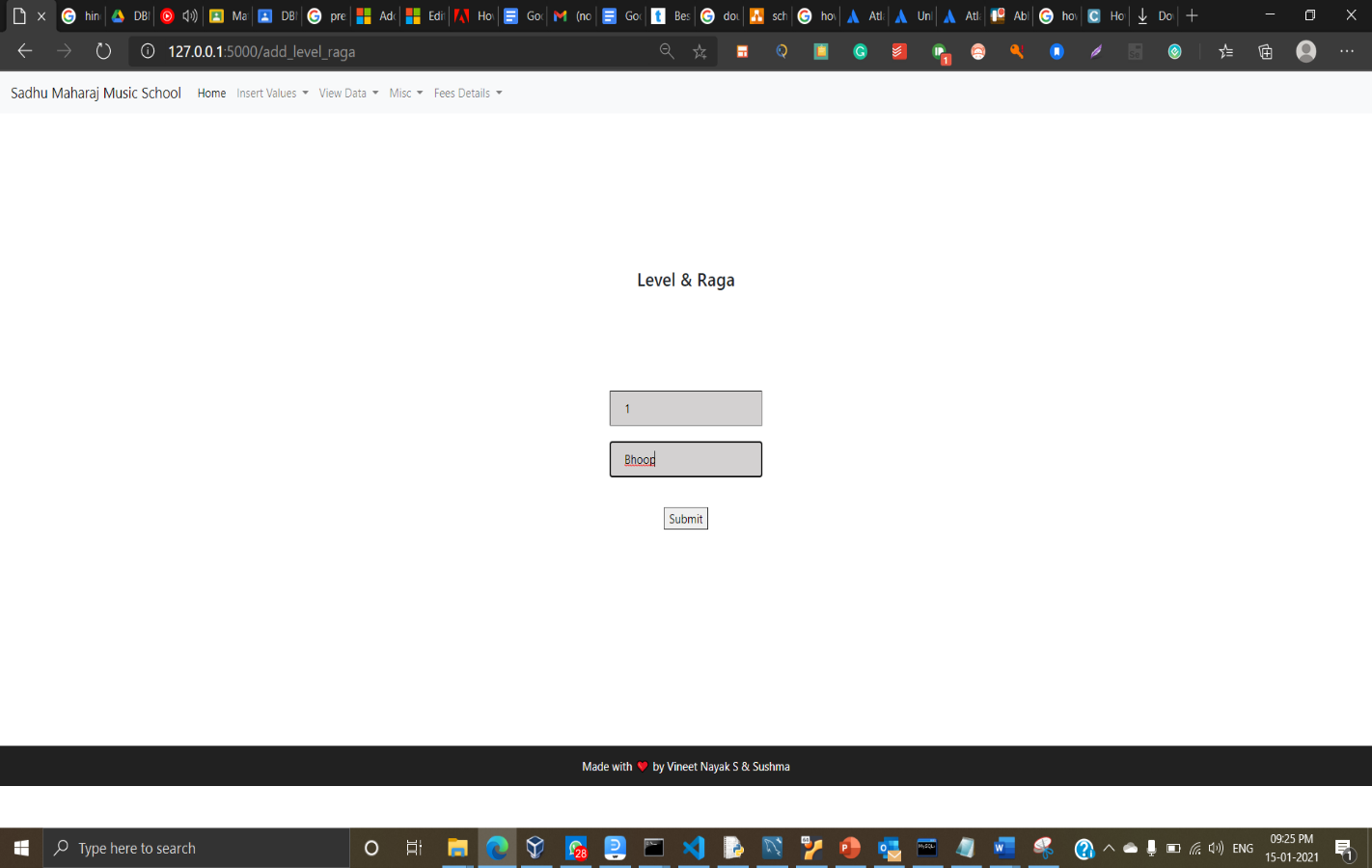


Figure 4.1.2.4: Add Level\_Ragas Page

**4.1.2.5 Add Events Data**

In this page admin can add details of events.

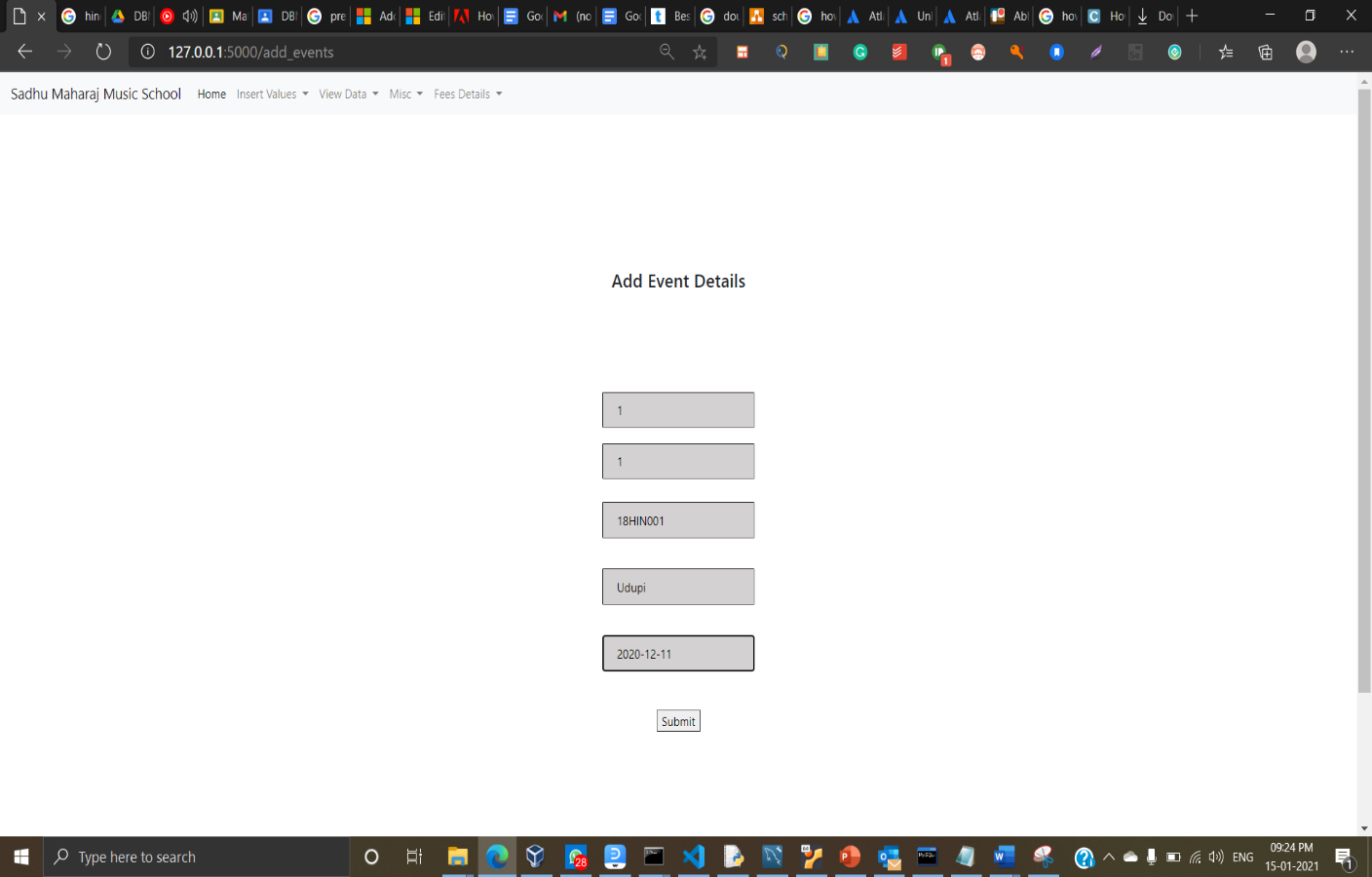


Figure 4.1.2.5: Add Event Details Page

**4.1.2.6 Add Examinations Data**

In this page admin can add details of examination.

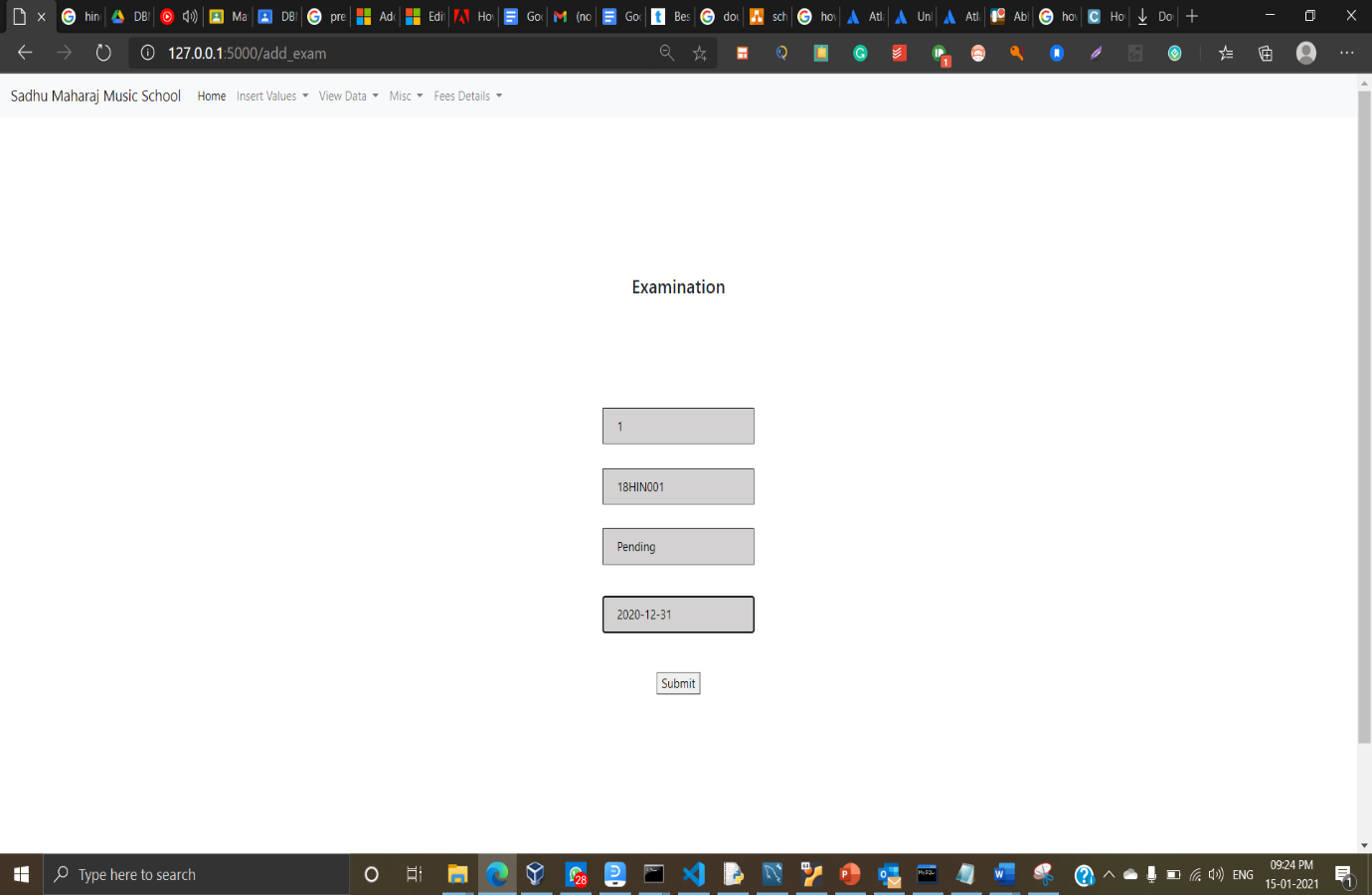


Figure 4.1.2.6: Add Examination Details Page

**4.1.2.7 Add Agency Data**

In this page admin can add details of agencies.

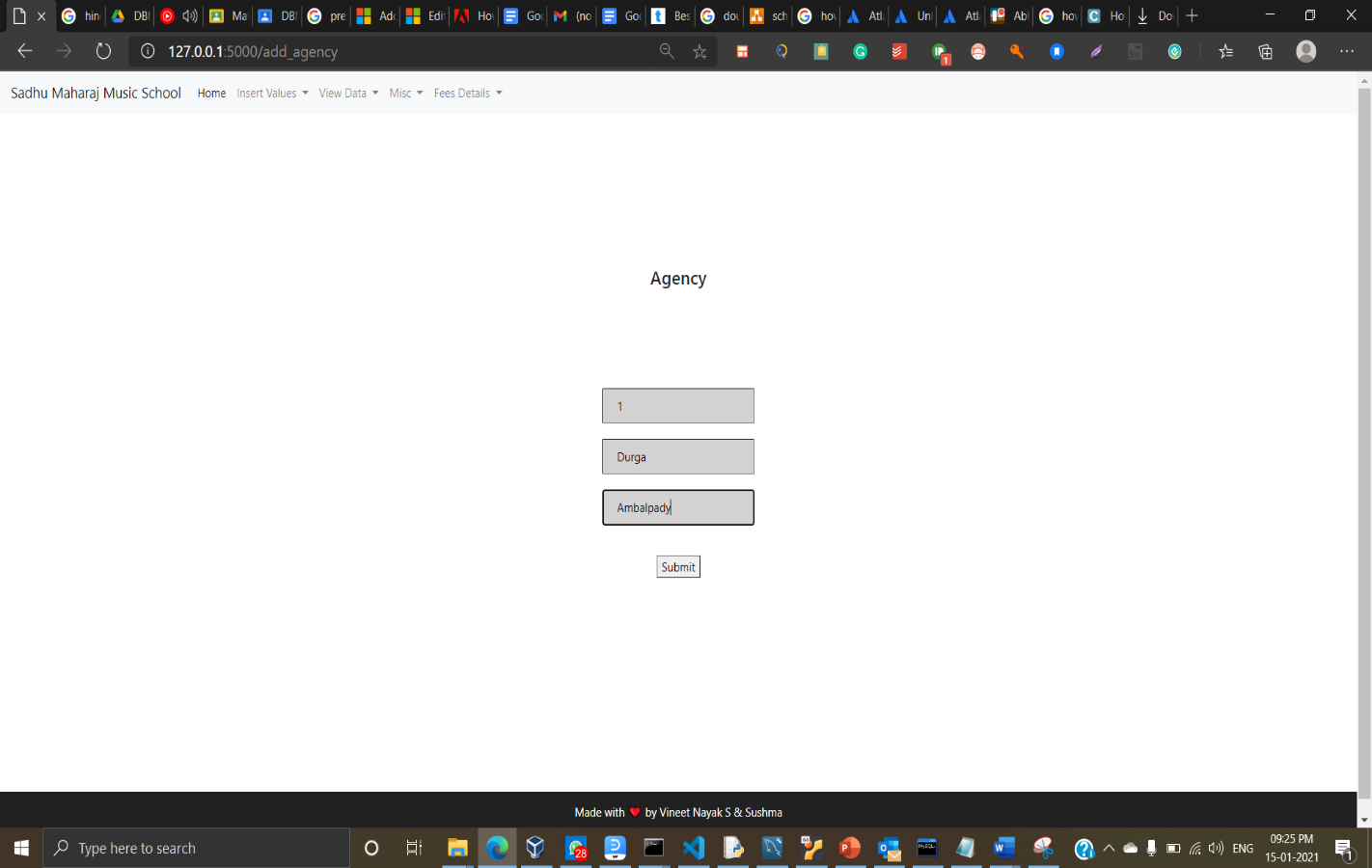


Figure 4.1.2.7: Add Agency Details Page

**4.1.2.8 Add Fees Payment Data**

In this page admin can add fees payment details.

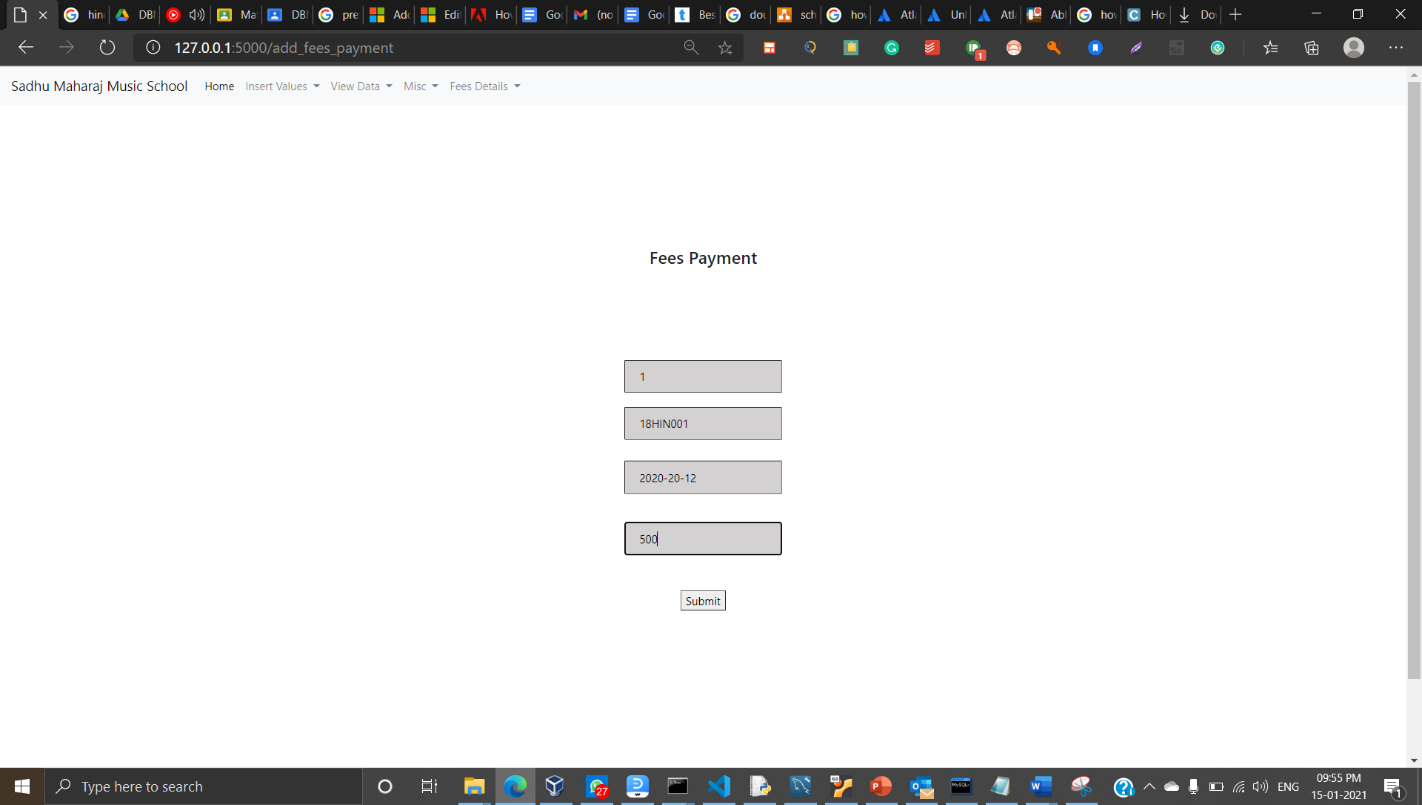


Figure 4.1.2.8: Add Fees Payment Details Page

**4.1.2.9 Add Fees Data**

In this page admin can add fees details.

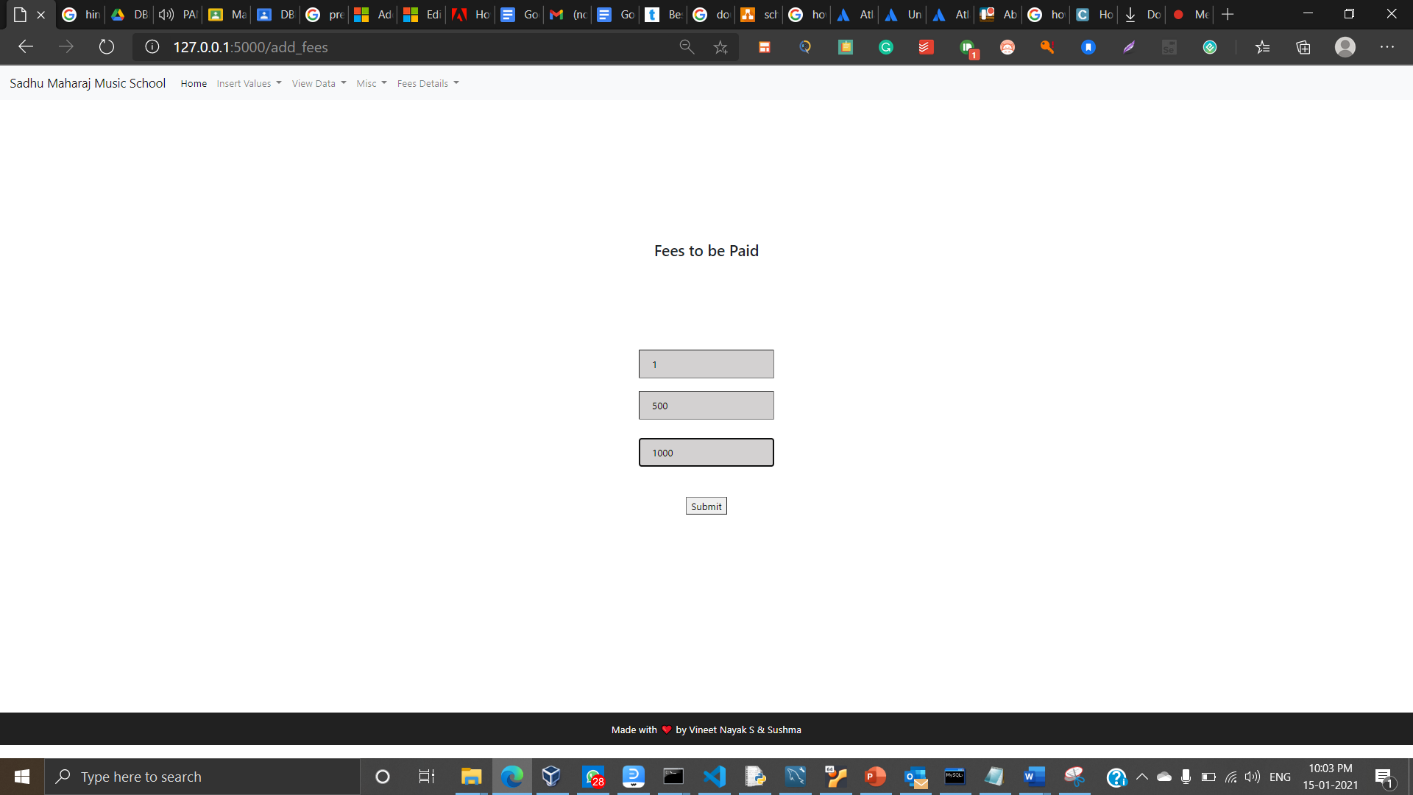


Figure 4.1.2.9: Add Fees Details Page

**4.1.3 View Data**

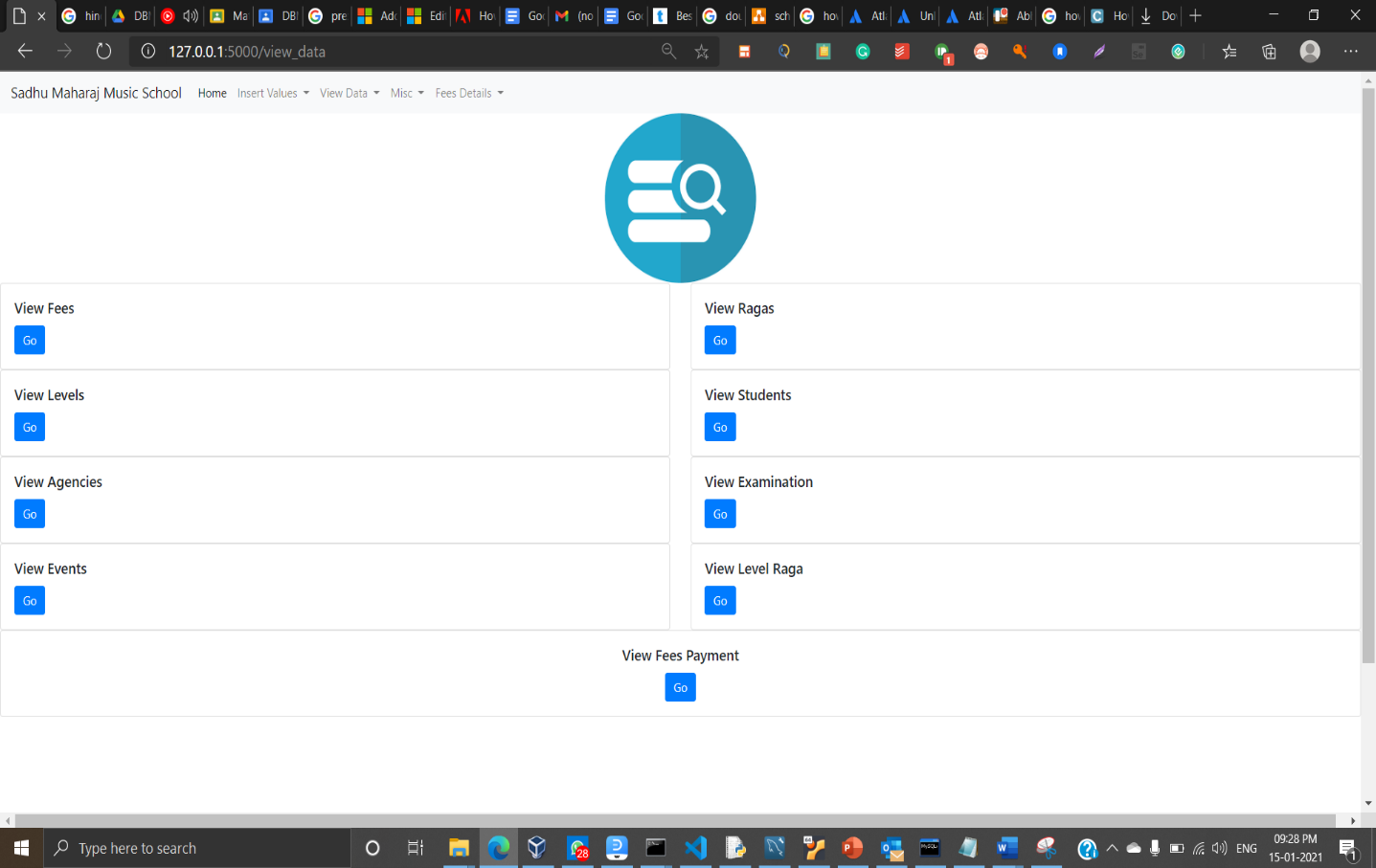


Figure 4.1.3: View Details Page

**4.1.3.1 View & Delete Student Data**

In this page admin can view and delete student details.

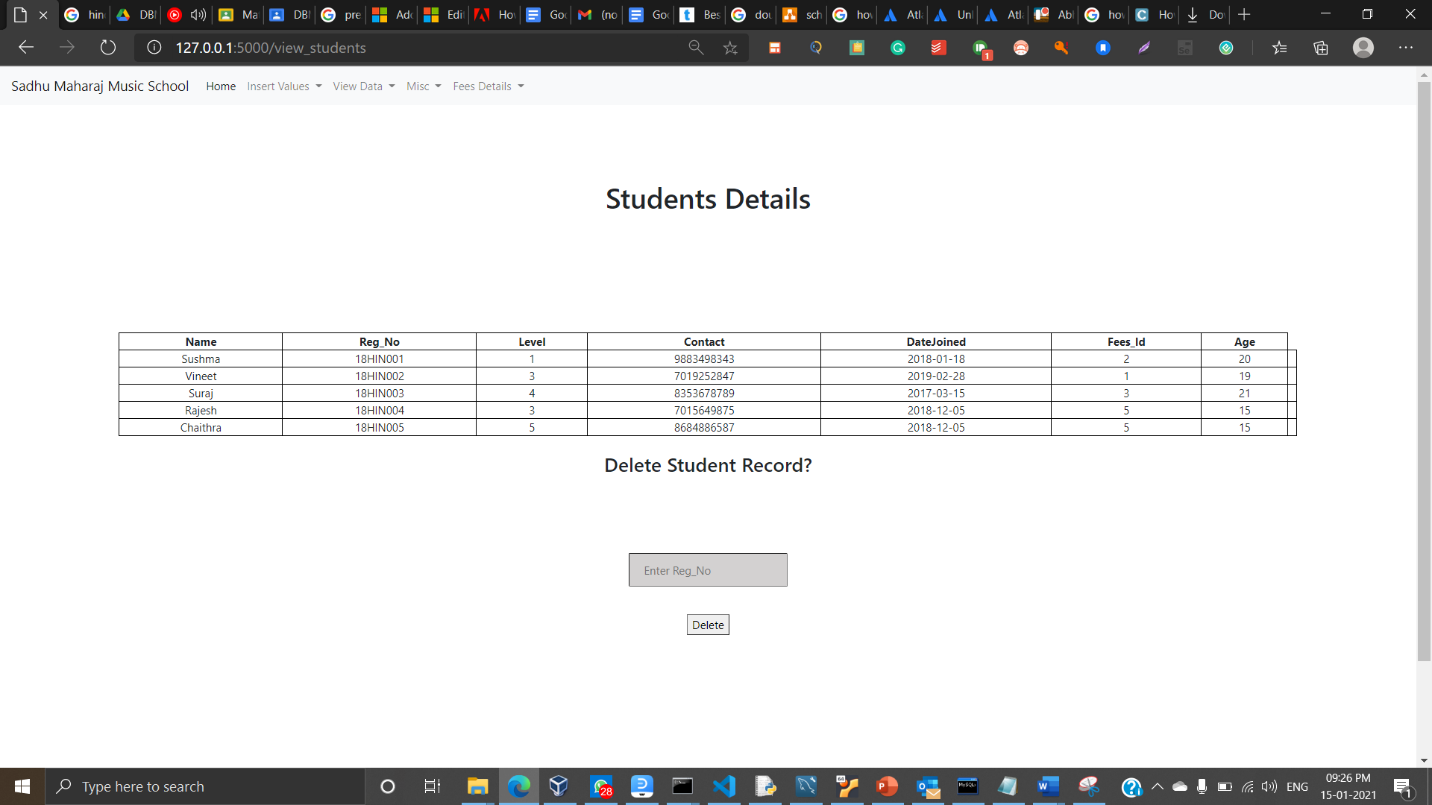


Figure 4.1.3.1: View & Delete Student Details Page

**4.1.3.2 View & Delete Levels Data**

In this page admin can view and delete level details.

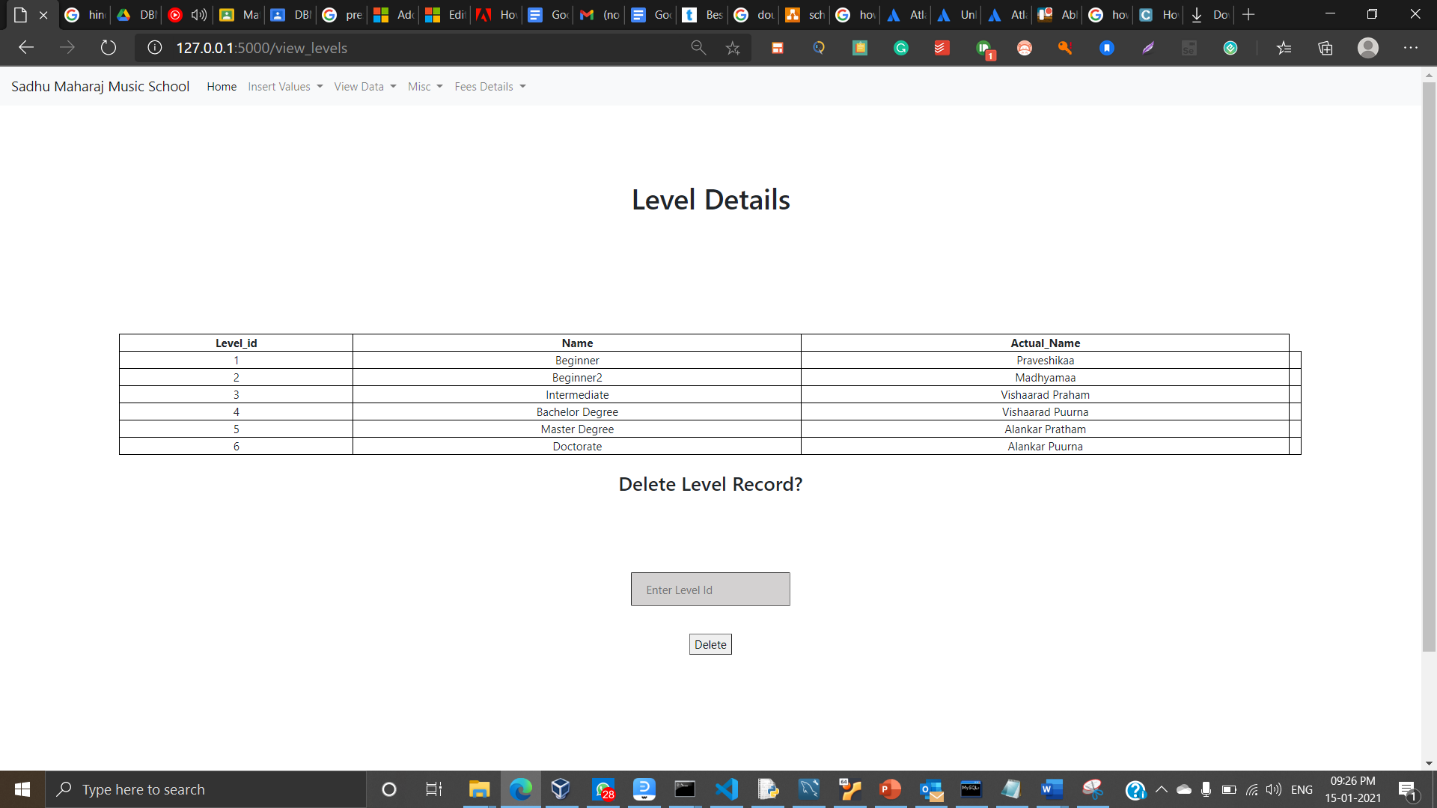


Figure 4.1.3.2: View & Delete Levels Page

**4.1.3.3 View & Delete Ragas Data**

In this page admin can view and delete ragas details.

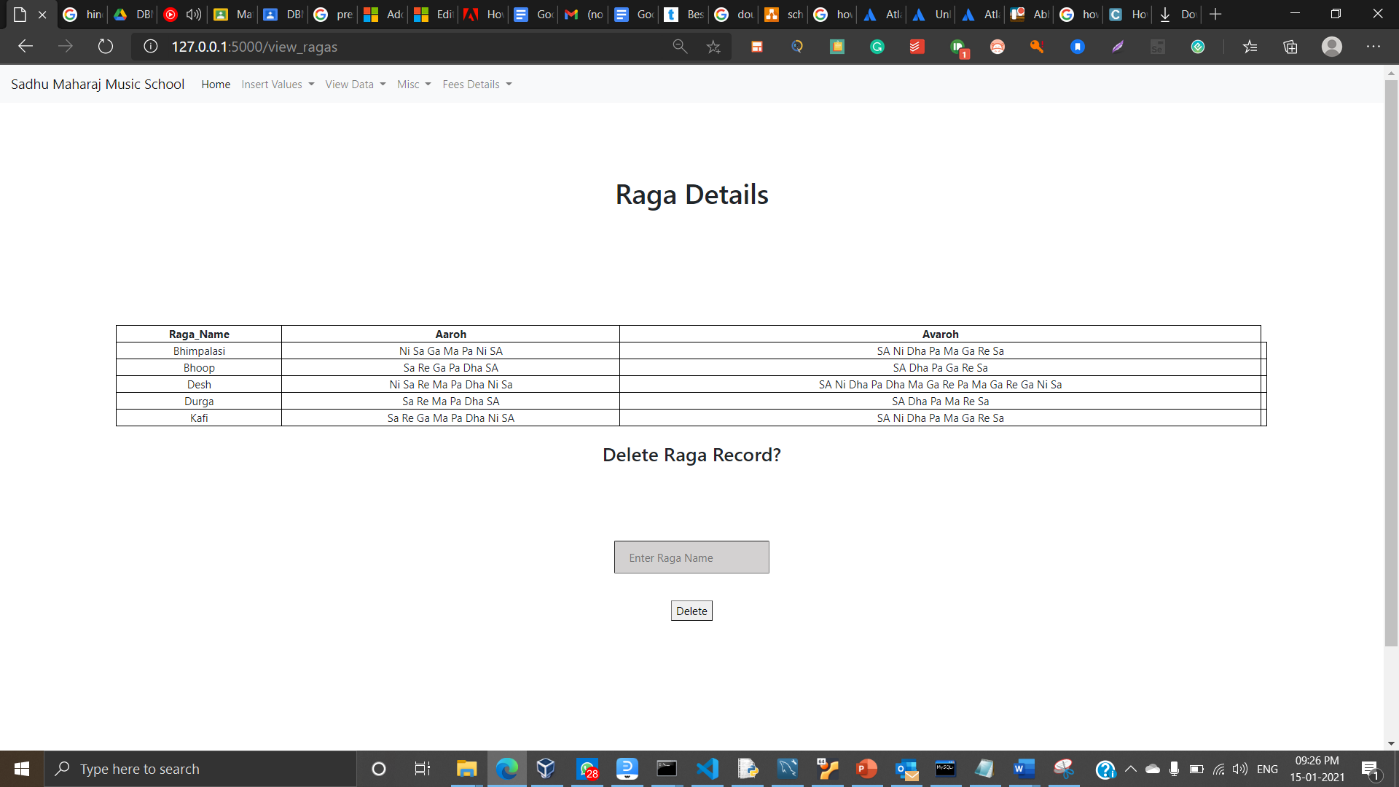


Figure 4.1.3.3: View & Delete Ragas Page

**4.1.3.4 View & Delete Events Data**

In this page admin can view and delete event details.

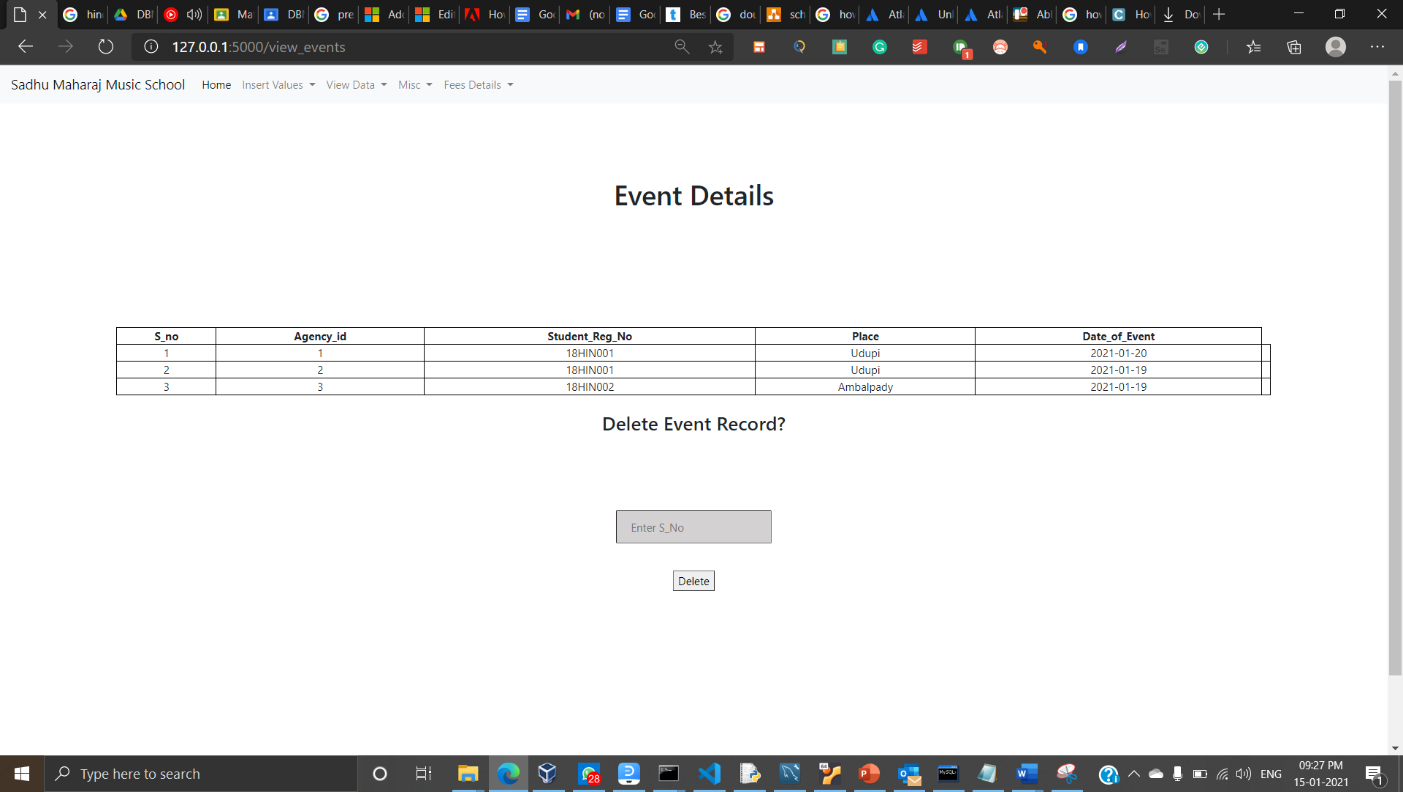


Figure 4.1.3.4: View & Delete Events Page

**4.1.3.5 View, Update & Delete Exam Data**

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

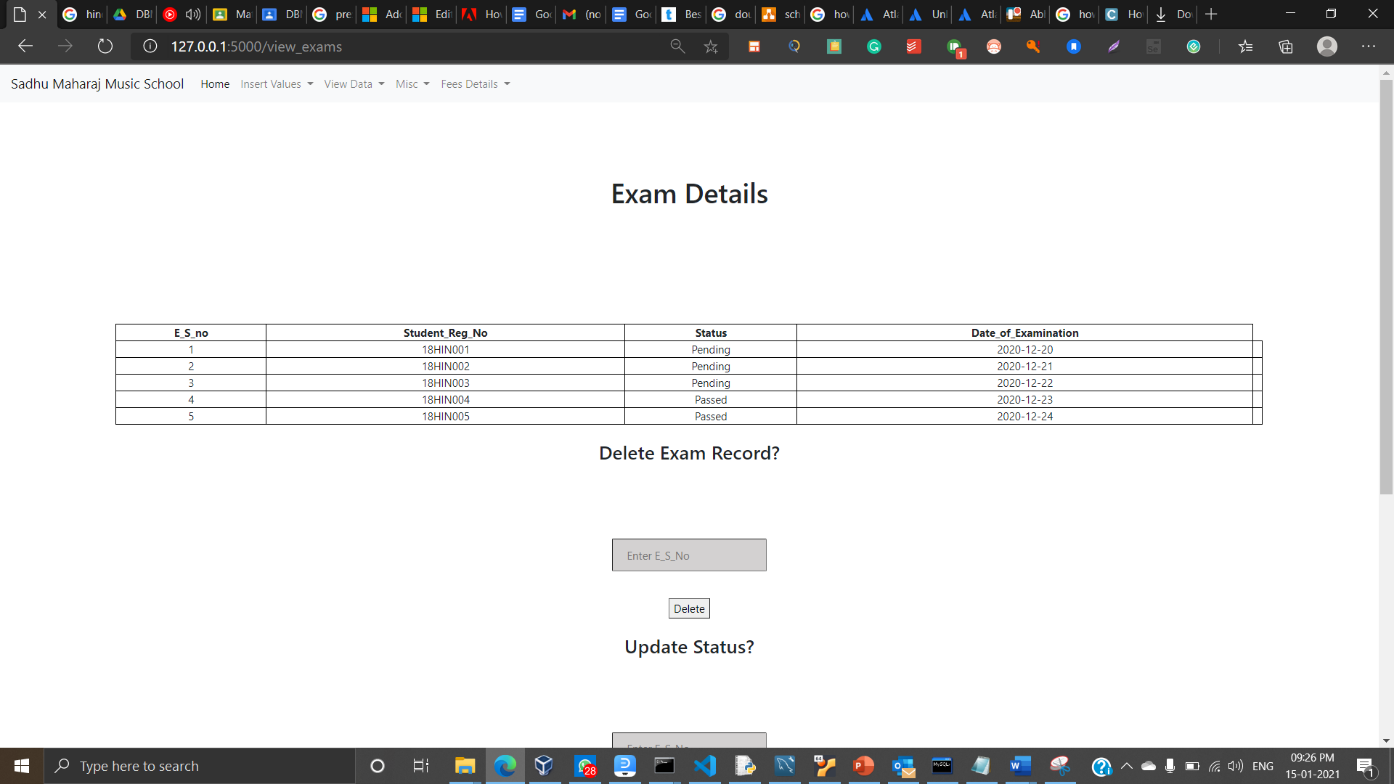


Figure 4.1.3.5.1: View, Update & Delete Exam Page

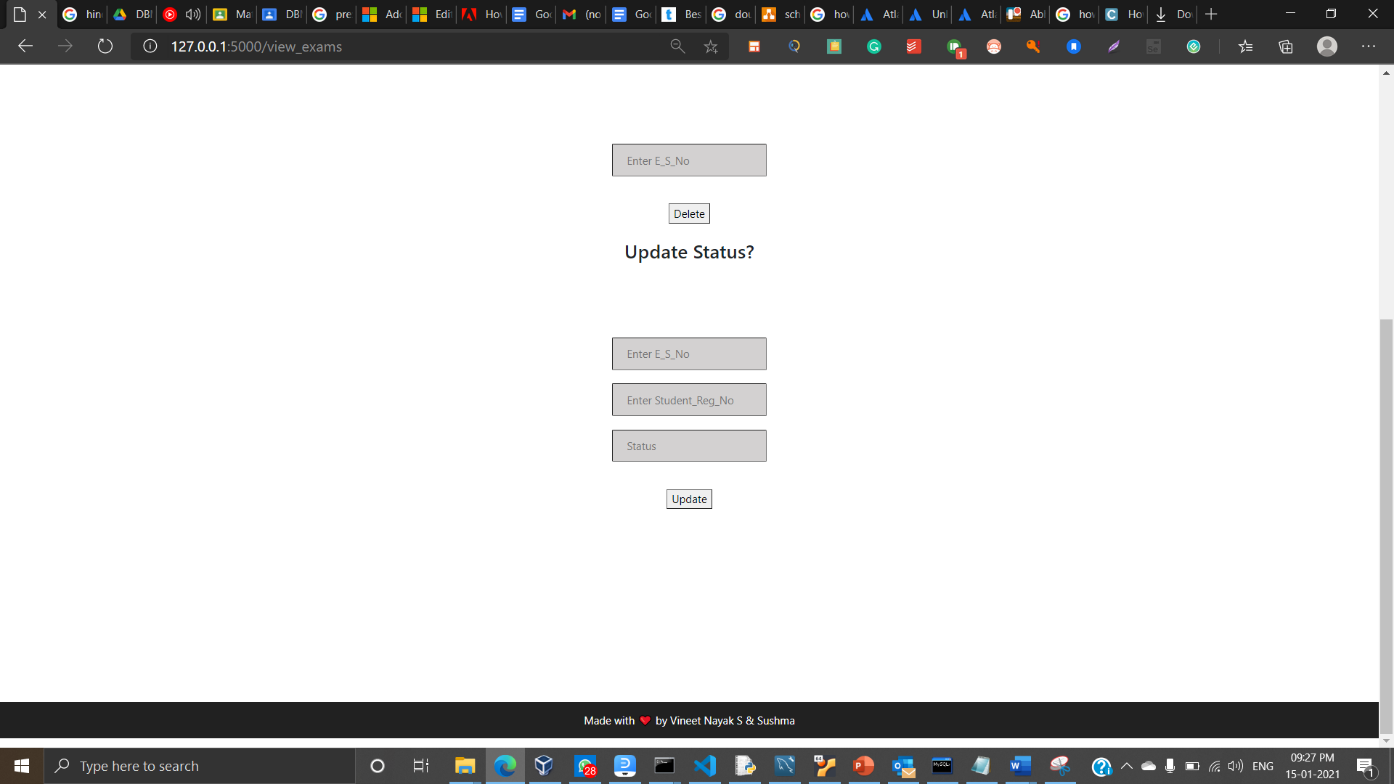


Figure 4.1.3.5.2: View, Update & Delete Exam Page

**4.1.3.6 View & Delete Agency Data**

In this page admin can view & delete Agency Data.

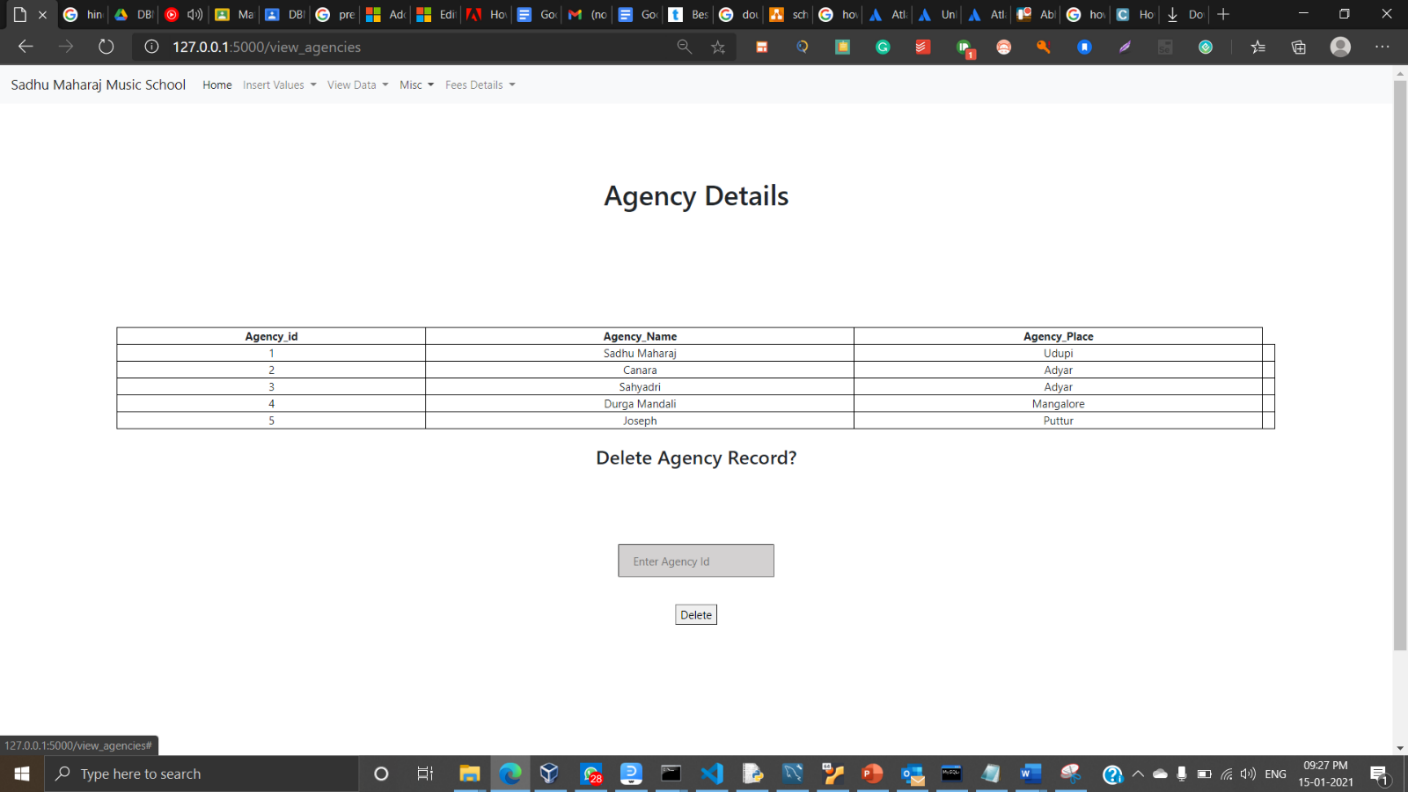


Figure 4.1.3.6: View & Delete Agency Details Page

**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.

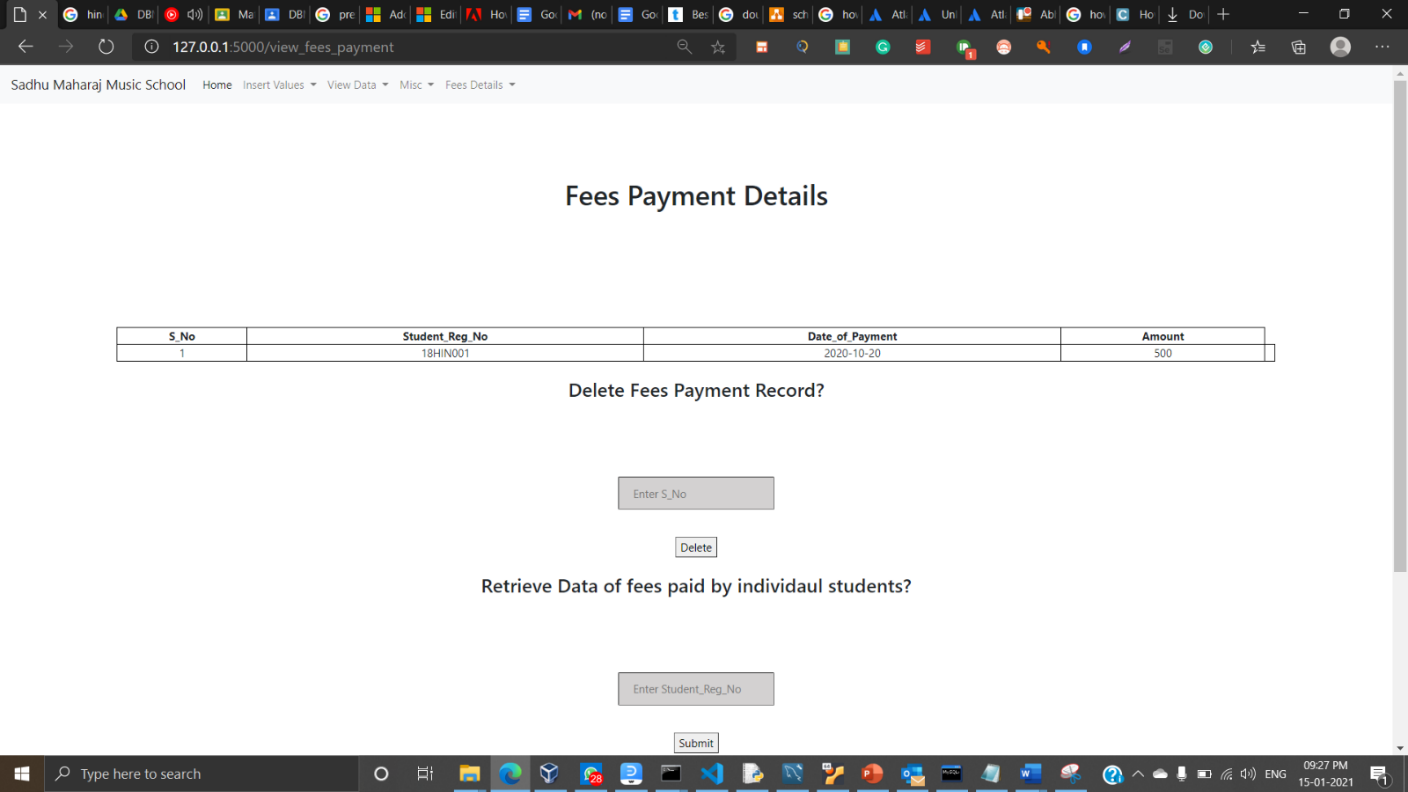


Figure 4.1.3.7: View & Delete Fees Payment Page

**4.1.3.8 View & Delete Fees Payment Data**

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

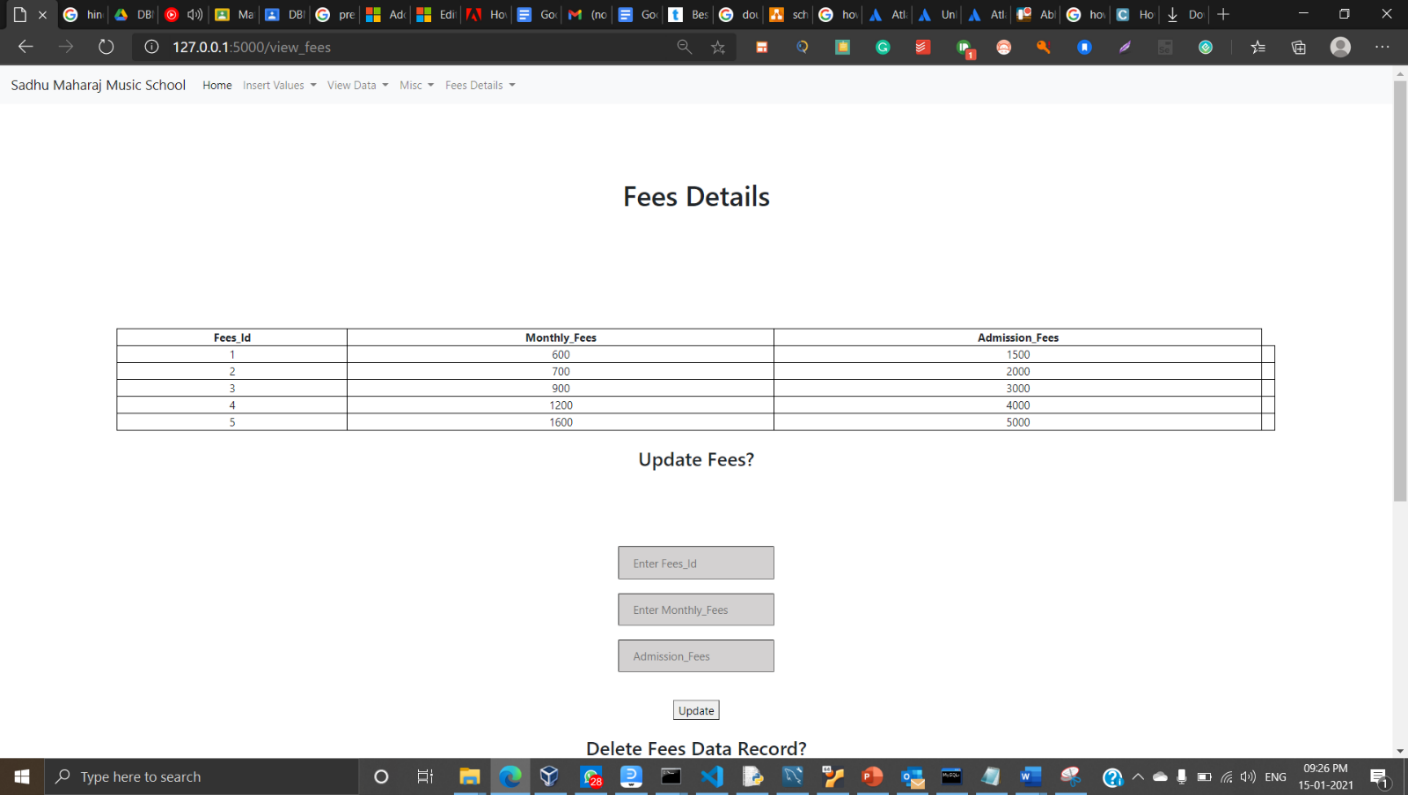


Figure 4.1.3.8.1: View, Update & Delete Fees Payment Page

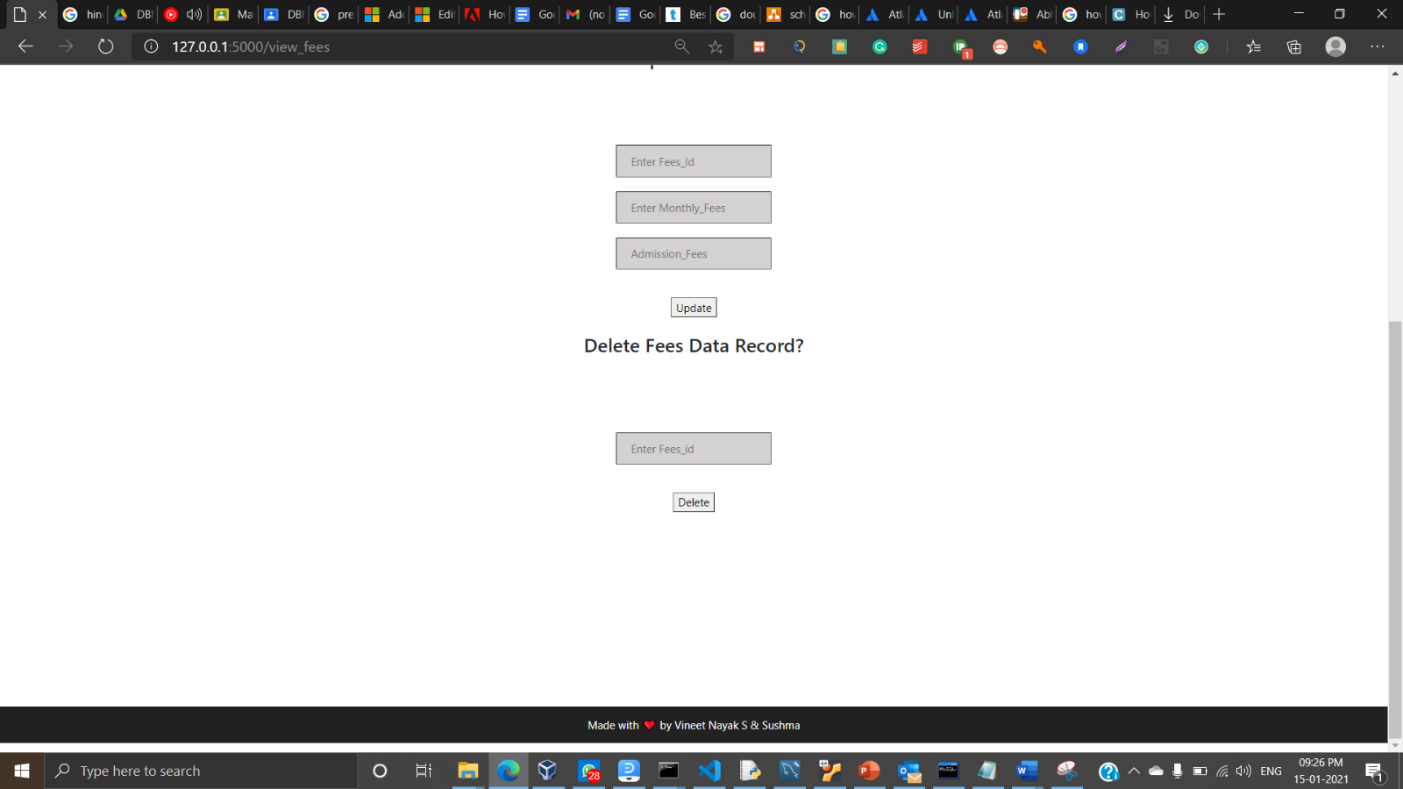


Figure 4.1.3.8.2: View, Update & Delete Fees Payment Page

**4.1.4 Miscellaneous Data**

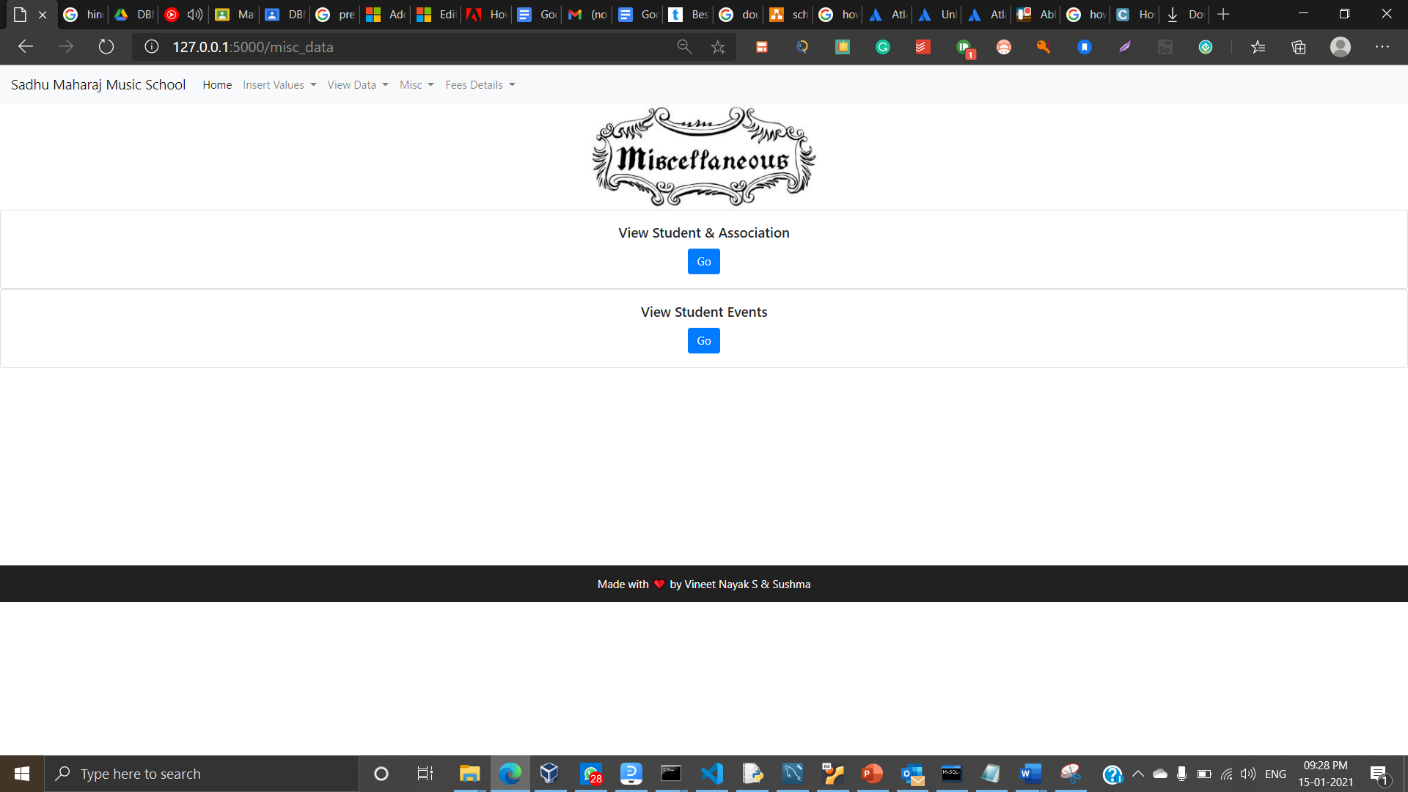


Figure 4.1.4: Miscellaneous Page

**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.

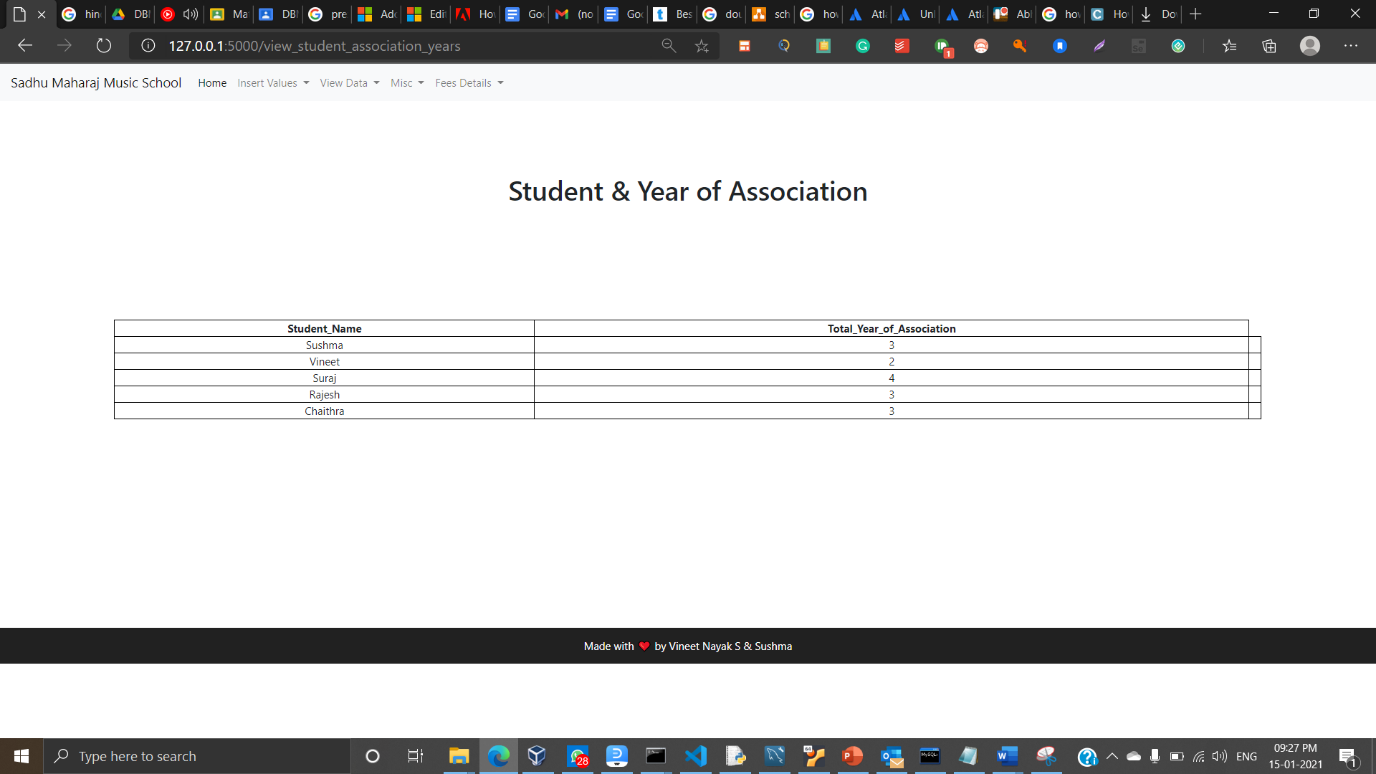


Figure 4.1.4.1: Student & Association View Page

**4.1.4.2 Student & Events**

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

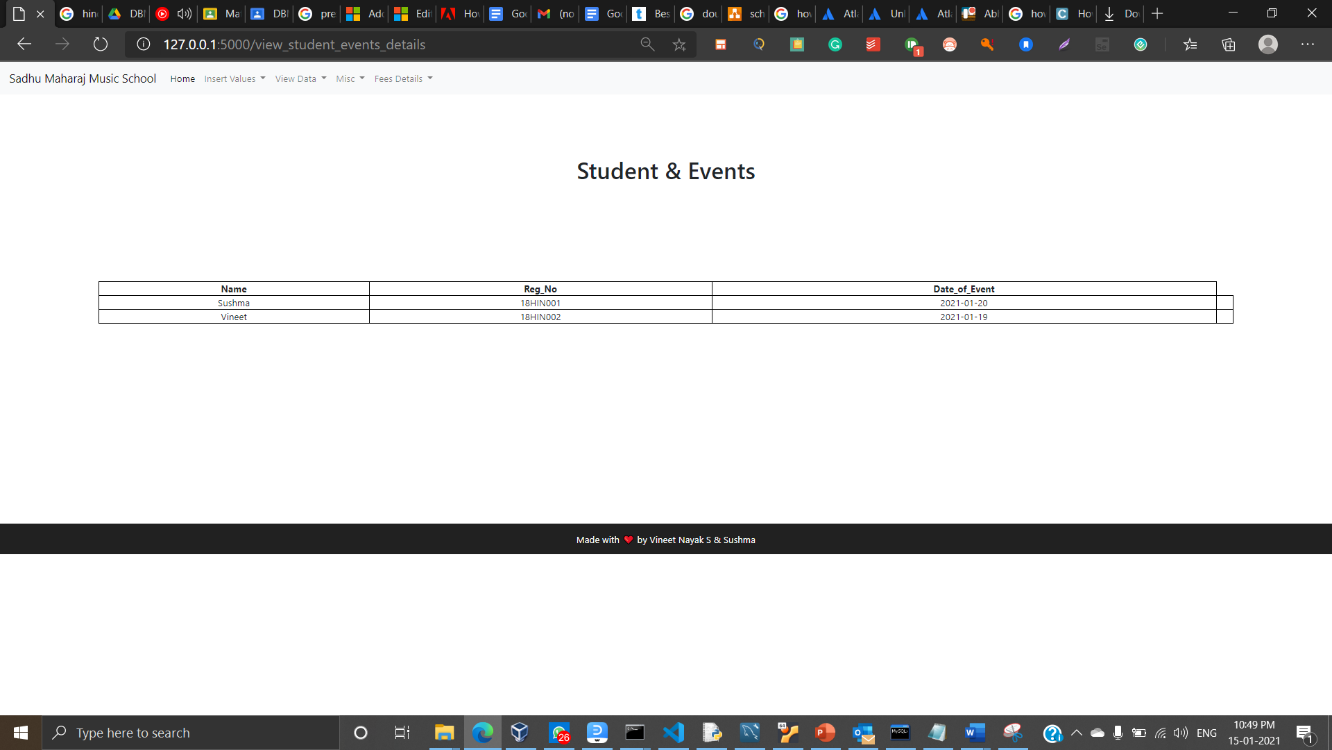


Figure 4.1.4.2: Student & Events View Page

**4.1.4.3 Output Status Page**

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

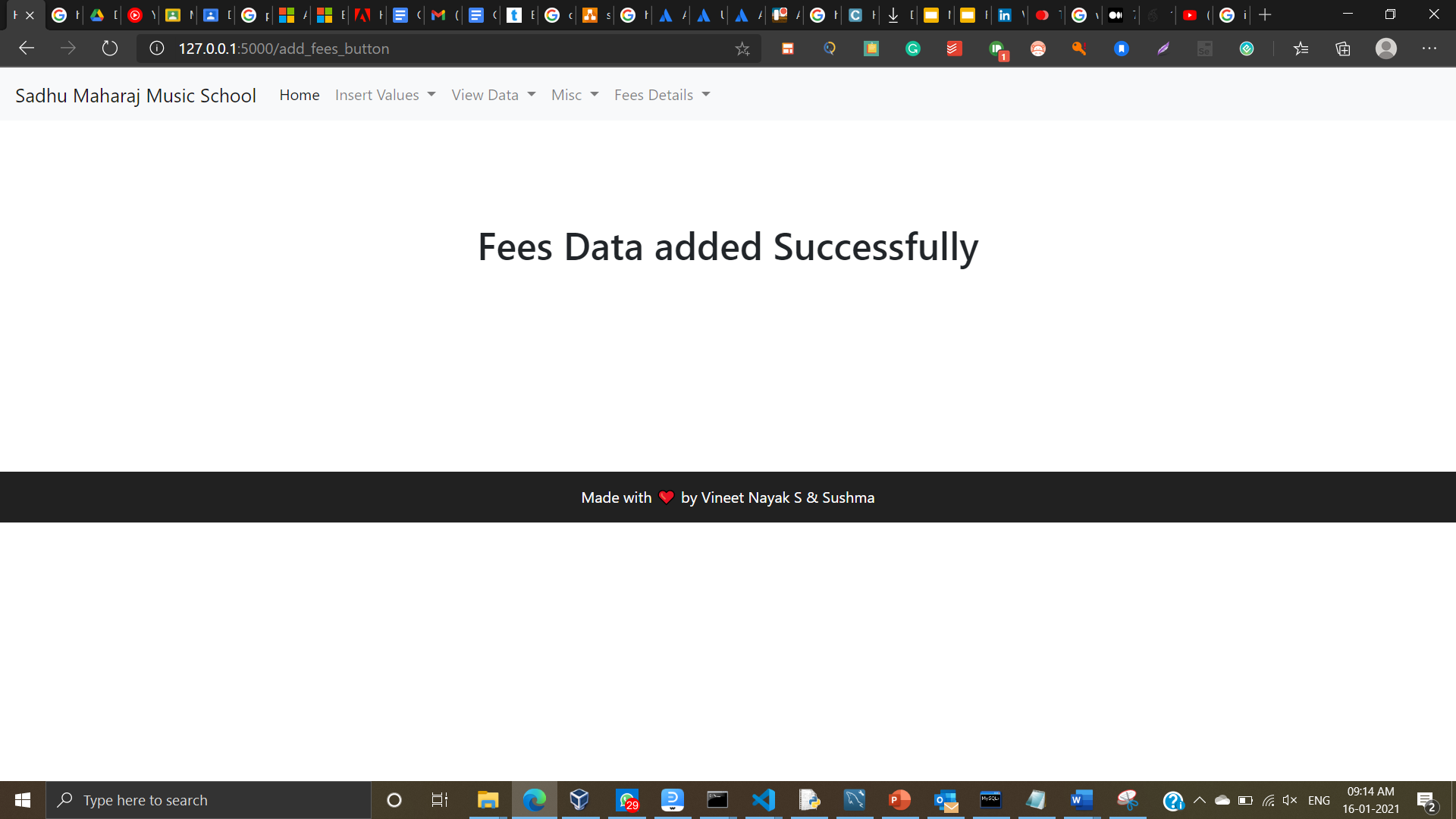


Figure 4.1.4.3.1: Success Status Page

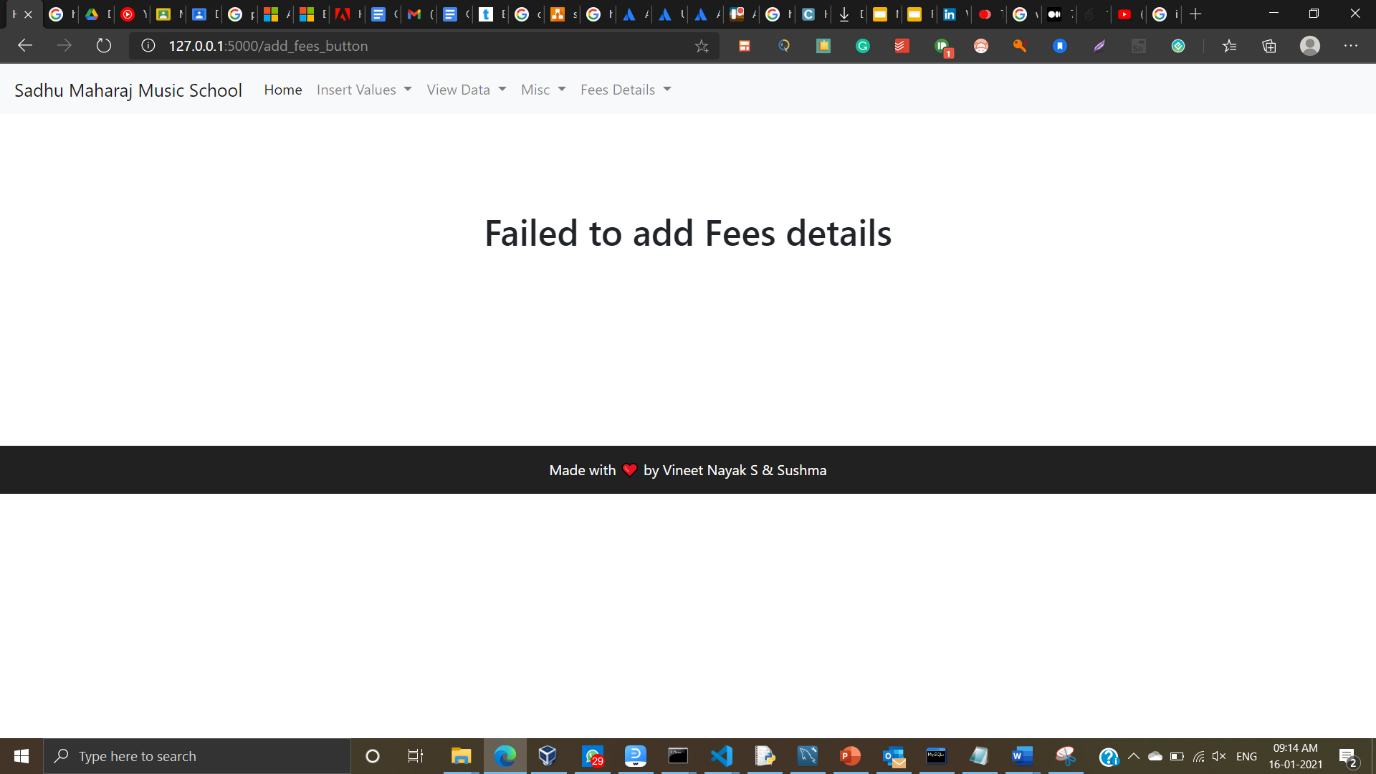


Figure 4.1.4.3.2: Failure Status Page

**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](https://www.youtube.com/watch?v=7S_tz1z_5bA)
2. Flask - [(6) Python Flask Tutorial: Full-Featured Web App Part 1 - Getting Started - YouTube](https://www.youtube.com/watch?v=MwZwr5Tvyxo)
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>