# **Student Registration Database Management System**

# **Bug Report and Lessons Learned**

# **IST659M003**

# **Dhwani Gandhi**

# **Karan Ashar**

# **Bugs and Solutions**

1. **Bug**: One of our Entities contained the college names. Earlier, we had set the datatype as VARCHAR (50), but after inserting the values we realised that the names were longer than the limit that we have set.

**Solution**: We fixed this by simply increasing the character limit to VARCHAR (100).

1. **Bug**: Initially, the Student table was connected to the College table and the Branch table individually. But we realised that since they were separately connected, we were not being able to maintain the consistency of the college-branch combinations.

**Solution**: We solved this issue by connecting the Student table with the College Branch table which contains all the combinations that are available in the database.

1. **Bug**: Initially in the Student table and the Student\_Registration table, the user had to manually select “No” as the value for Status\_1 & status\_2 and Student\_Status respectively. This drawback lead to the issue where there was uncertainty that the student would make a mistake and select a “Yes” instead.

**Solution**: To prevent this mishap we set the default value as “No” for all three statuses and removed this column from both the Student and the Student\_Registration forms where the user would have to input his information.

1. **Bug**: The Status\_1 in the Student table had to be manually changed to “Yes” once the student makes the payment and thus becomes a currently enrolled student. Also, the Student\_Status in the Student\_Registration had to be manually changed to “Yes” once the student makes the payment to confirm the payment made for that combination of college, branch, subject and semester. This needed a lot of time, accuracy and consistency.

**Solution**: Instead of wasting time and effort in doing so manually we created a trigger. Here as soon as an entry is made in the payment table, the corresponding entry in the Student\_Registration table and the student table are matched, and the respective status are changed to “Yes.” The trigger is as follows:

CREATE TRIGGER Update\_Status

ON Payment

FOR INSERT AS

BEGIN

UPDATE Student\_Registration

SET Student\_Registration.Student\_Status='Yes'

FROM Student\_Registration

INNER JOIN Payment ON

Student\_Registration.Semester=Payment.Semester AND

Student\_Registration.Subject\_ID=Payment.Subject\_ID AND

Student\_Registration.Branch\_ID=Payment.Branch\_ID AND

Student\_Registration.College\_ID=Payment.College\_ID AND

Student\_Registration.U\_Name=Payment.U\_Name

END;

CREATE TRIGGER Update\_Status1

ON Payment

FOR INSERT AS

BEGIN

UPDATE Student

SET Student.Status\_1='Yes'

FROM Student

INNER JOIN Payment ON

Student.U\_Name=Payment.U\_Name

END;

# **Lessons Learned**

1. We learned how to use SQL to create tables, insert values in those tables, create triggers, views and transactions for good data management, and run queries to find out the required information from the database.
2. We learned how to build a database management system from scratch. We first developed a logical data model depending on the business rules. We learned how to create an entity relationship model using Microsoft Visio. After this model we could easily create the tables for the database. We also learned to use Microsoft Access where we created forms for inserting information into the tables. We also learned to generate reports that would display meaningful insights needed by the users. We were able to build an efficient database management system from start to end.
3. We also learned how to connect Microsoft SQL Server to MS Access for easy and efficient manipulation of data.