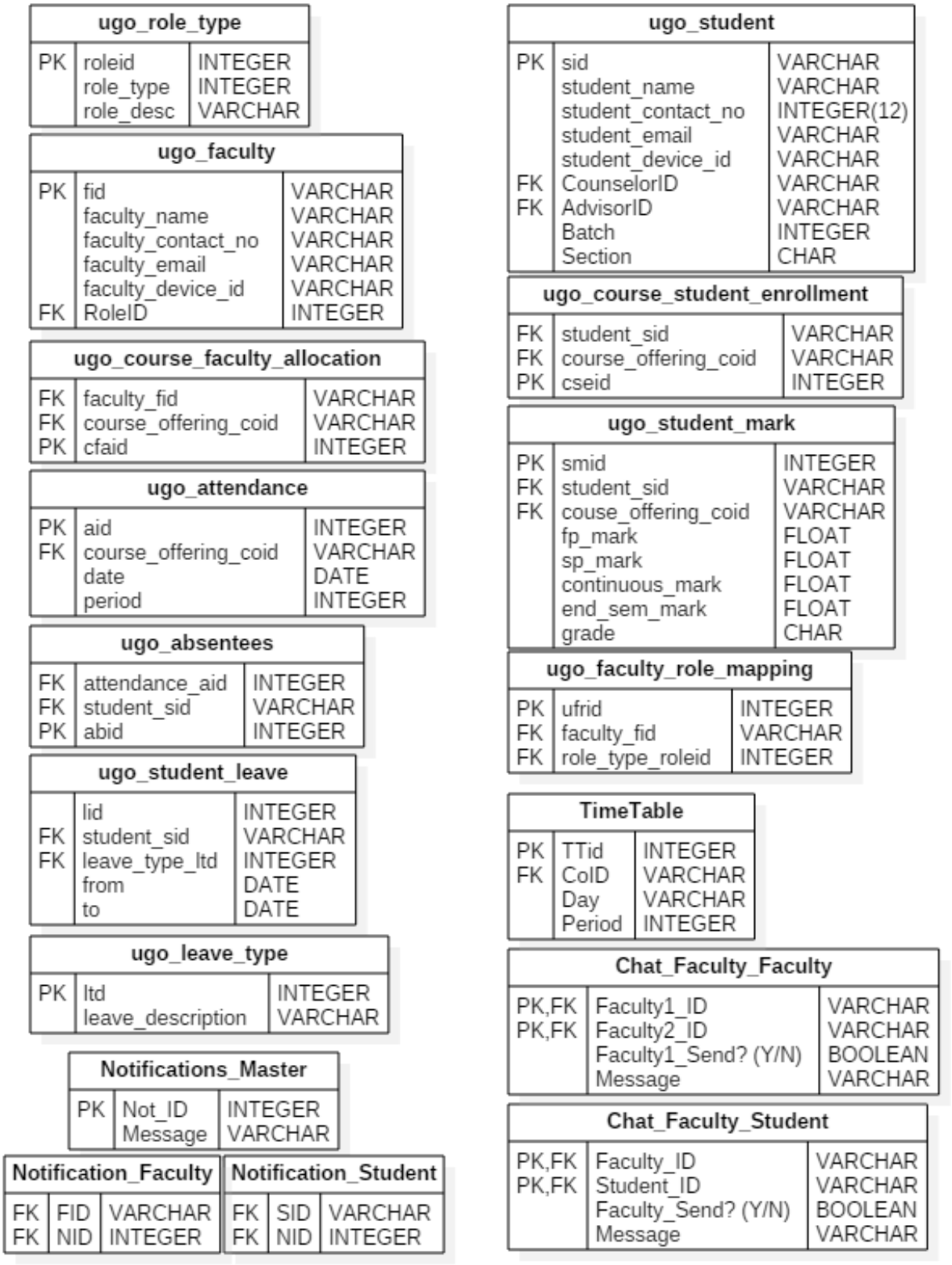
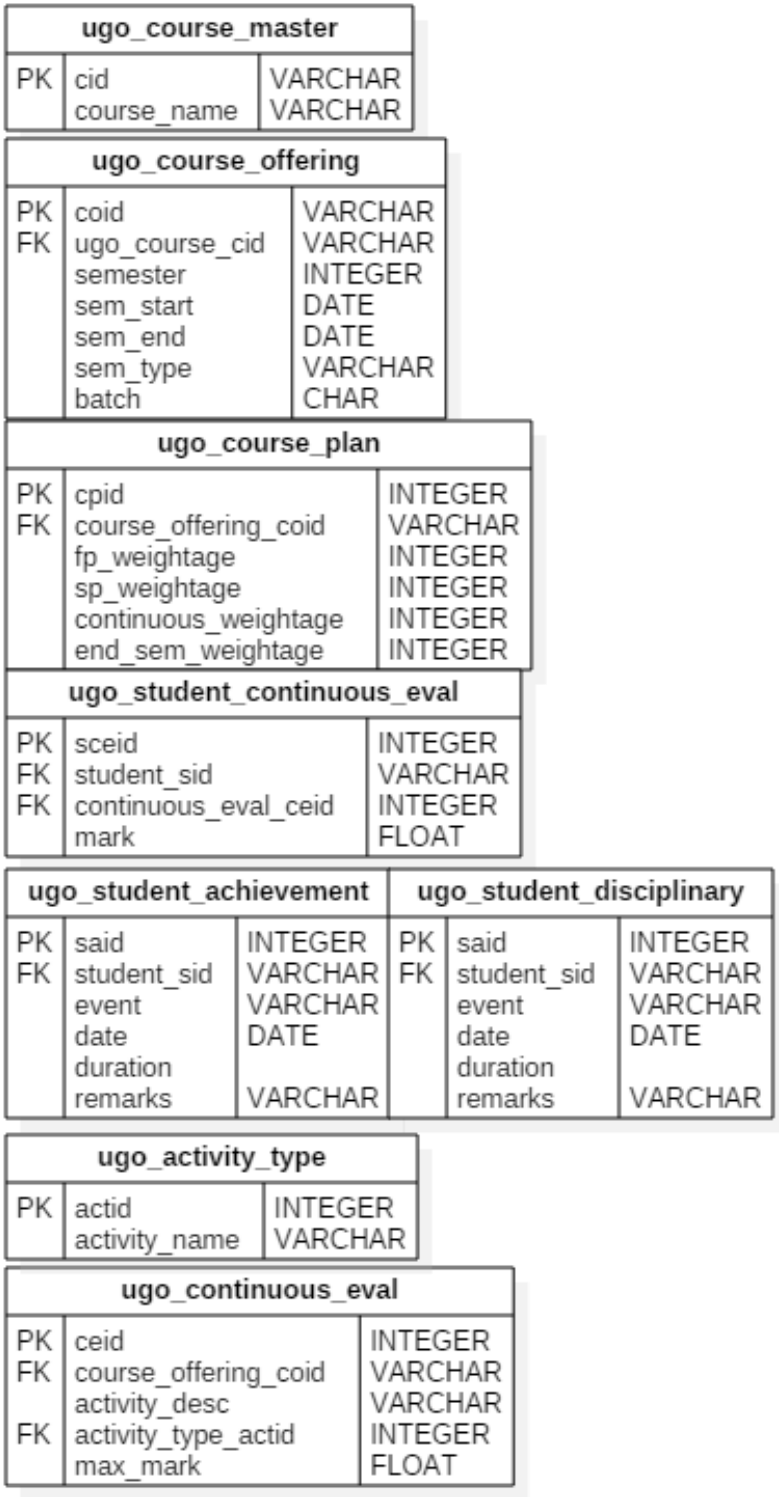
Database Team Project Report

INDEX

UML Diagram of Tables




ugo_course_master
PK cid VARCHAR
course_name VARCHAR
ugo_course_offering
PK coid VARCHAR
FK ugo_course_cid VARCHAR
semester INTEGER
sem_start DATE
sem_end DATE
sem_type VARCHAR
batch CHAR
ugo_course_plan
PK cpid INTEGER
FK course_offering_coid VARCHAR
fp_weightage INTEGER
sp_weightage INTEGER
continuous_weightage INTEGER
end_sem_weightage INTEGER
ugo_student_continuous_eval
PK sceid INTEGER
FK student_sid VARCHAR
FK continuous_eval_ceid INTEGER
mark FLOAT
ugo_student_achievement
PK said INTEGER
FK student_sid VARCHAR
event VARCHAR
date DATE
duration
remarks VARCHAR
ugo_student_disciplinary
PK said INTEGER
FK student_sid VARCHAR
event VARCHAR
date DATE
duration
remarks VARCHAR
ugo_activity_type
PK actid INTEGER
activity_name VARCHAR
ugo_continuous_eval
PK ceid INTEGER
FK course_offering_coid VARCHAR
activity_desc VARCHAR
FK activity_type_actid INTEGER
max_mark FLOAT

E R DIAGRAM HERE!

Data Dictionary

(Needs to Be updated)

Tables (Done):

Faculty
Student

Course Master
Currently Running Courses
Course Faculty Allocation
Course Student Allocation
Weightage
Student Grading

Notification Master
Student Notification
Faculty Notification

Master Attendance
Leave Table
Absentee Table
Time Table

Chat Log (Faculty and Faculty)
Chat Log (Faculty and Student)

Tables (Pending):

Role Allocation
Access Privileges
Login/Authentication Table

Faculty and Student

Faculty

FID	Name	Phone. Number	Type	Other Details

FID : Id of Faculty (PK)
Name : Name of Faculty
Ph. Number : Phone Number
Type : Type of Faculty (Restricted, ex H for HoD, C for Counsellor, etc.)
Other Det : More columns acc to requirements (mail id, etc.)

Student

SID	Name	Email id	Couns. ID	Adv. ID	Other Details

SID	:	Student ID (PK)
Name	:	Name of Student
Email ID	:	email id of student
Couns ID	:	ID of counsellor (FK, refers to FID in Faculty table, type must be C)
Adv. ID	:	ID of advisor (FK, refers to FID in faculty table, type must be A)
Other Details	:	As required by project (Parent name, address, photo, etc.)

Courses and Evaluation

Course_Master

Year	Course Code	Category	Credits	L T P	Title

Year	:	Year of syllabus publication (Composite PK)
Course Code	:	Course Code, eg: CSE360 (Composite PK)
Category	:	H for Humanities, E for Engineering, etc.
Credits	:	Credits allocated for this course
L T P	:	Three values, giving (Lecture hrs, Tutorial hrs, Practical hrs)

Currently Running Courses

Course Number	(Year, Course Code)	ES (??)

Course Number	:	Unique ID (PK)
(Year, Course Code)	:	FK refers Course Master
ES	:	Mentioned in Course book and excel sheet, no description… (??)

Course Faculty Allocation

C. No	FID

C. No	:	Course Number (FK, refers Currently Running Courses) (Comp. PK)
FID	:	Faculty assigned to this course (FK refers Faculty Table) (Comp. PK)

Course Student Allocation

C. No	SID

C. No : Course Number (FK, refers Currently Running Courses) (Comp. PK)
SID : Student assigned to this course (FK refers Student Table) (Comp. PK)

Weightage

Year, CID	Periodical I	Periodical II	Continuous Eval.	End Semester

Year, CID : (FK refers Master Course Table) (PK)
Periodical I : Weightage of Periodical I
Periodical II : Weightage of Periodical II
Continuous Eval: Weightage of Continuous Eval.
End Semester : Weightage of End Semester Exams

Student_grading

SID	C No	Periodical I	Periodical II	C. Eval	End Sem	Grade

SID : Student ID (FK refers Student Table) (Composite PK)
C No : Course Number (FK refers currently running courses) (Composite PK)
Periodical I : Marks (weighted in Periodical I)
Periodical II : Marks (weighted in Periodical II)
Contin. Eval : Marks (weighted) in Contin. Eval
End Semester : Marks (weighted) in End Sem
Grade : Final Grade (Initially NULL)

Notifications

Notification Master

NID	Message	Timestamp

NID : Notification ID (PK)
Message : Message of the Notification
Timestamp : Time of adding notification

Faculty_Notification

FID	NID

FID : Faculty ID (FK refers Faculty Table) (Composite PK)
NID : Notification ID (FK refers Notification Master) (Composite PK)

Student_Notification

SID	NID

SID : Student ID (FK refers Student Table) (Composite PK)
NID : Notification ID (FK refers Notification Master) (Composite PK)

Attendance and Time Table

Master Attendance

AID	Course No	Date	Period

AID : Attendance ID (Primary Key)
Course No : Which course (Foreign Key, refers Currently Running Courses)
Date : On which date
Period : Which Period

Leave Table

LID	SID	Type	Details	From	To

LID : Leave ID (Primary Key)
SID : Student ID (Foreign Key, refers Student Table)
Type : Type of Leave granted (Duty, Medical, etc)
From : Start Period of Leave (Date, Period)
To : End Period of Leave (Date, Period)

Absentee Table

AID	SID	LID

AID : Attendance ID (Foreign Key, refers Master Attendance) (Composite Primary Key)

SID : Student ID (Foreign Key, refers Student Table) (Composite Primary Key)

LID : Leave ID (Foreign Key, refers Leave Table)

Time Table

Day	Period	Course No

Day : Day of the week (Monday, Tuesday, etc.) (Composite PK)
Period : Which Period of the Day (Composite PK)
Course No : Course allocated (FK refers Curr. Running Courses) (Composite PK)

Note :- No need of semester etc, as a course may be allocated to multiple semesters in same hour, as well as the fact that a batch's time table can be extracted by the courses attended by any student in that batch.

Chat Log

Faculty to Faculty

Fac. ID 1	Fac. ID 2	User1 Send?(Y/N)	Message	Timestamp

Fac ID 1 : ID of faculty 1 (Refers Faculty Table)
Fac ID 2: ID of faculty 2 (refers Faculty Table)
User1 Send?: Is the sender of msg Fac. ID 1? Yes or No
Message: The chat message
Timestamp: The timestamp of the message

Faculty to Student

Fac. ID	Stud. ID	User1 Send?(Y/N)	Message	Timestamp

Fac ID: ID of faculty (Refers Faculty Table)
Stud. ID: ID of student (refers Student Table)
User1 Send?: Is the sender of msg Fac. ID? Yes or No
Message: The chat messageTimestamp: The timestamp of the message

REQUIREMENTS

WITH DESCRIPTION ON HOW DB HANDLES EACH

I. Attendance

Students :-

- **Should be able to view their own attendance (both the overall percentage for a given course and also a detailed report for each class).**

  ugo_attendance gives the details of each class for a course on a date. We can then check whether the student is absent for a class from ugo_absentee table where attendance_id is a foreign key from the ugo_attendance table.

- **Should know whether they are in risk of attendance shortage (notifications).**

  We can calculate the attendance for a student for a course as stated above. If his/her attendance falls below the minimum percentage we can create a notification and store in the Notification_Master table which can be distinguished using the notification id(Not_Id). The Notification_Student contains the Not_Id as foreign key and the student_id as foreign key. The Student_Notification can be used to send notifications.

Faculty :-

- **Should be able to enter, edit and track attendance for all the students that he/she teaches for the course that they take.**

  Faculty must enter the details of the class taken in the ugo_attendance table. Then faculty must enter the students absent for that class in the ugo_absentee table.

- **Should be able to sort through and view those students who are having attendance shortages.**

  The number of classes for each course can be counted from ugo_attendance table. The ugo_absentee table can be searched and the number of classes for which a particular student is absent can be known. Hence we can keep track of all students who have attendance shortage.

- **Should not be able to edit or view attendance for subjects that he/she doesn't teach.**

  ugo_course_faculty_allocation contains the details of the faculty allotted to each course. A faculty can edit the attendance only if the faculty_id matches with the faculty_id allotted to a particular course from the ugo_course_faculty_allocation table.

Advisors/Councilors :-

- **Should be able to view the attendance of all the students that he/she has been allotted to for all the courses that they study.**

  The ugo_student table contains both Counselor and Advisor foreign keys which are assigned based on the primary key of ugo_faculty_role_mapping table. The primary key is assigned based on the role_type_id which is provided for each faculty using ugo_role_type model.

- **Should be able to sort through and view those students having attendance shortages.**

  From the ugo_absentees table Counselor can check for attendance of those students assigned to them in a particular course from ugo_absentee table and check the number of classes taken from ugo_attendance table which will help in calculating those having attendance shortage.

II. Students' Profile.

Students :-

- **Should be able to view their own profile.**

  Student can only view or edit his profile which is done by the ugo_student table. Once a student logs in, the login student_id should match with that present in the ugo_student table so that he can only view or edit only his own profile.

Faculty :-

- **Should be able to view the profiles of the students that he/she is teaching.**

  Faculty can match the course assigned to him/her (using ugo_course_faculty_allocation table) and the course in which the students have

enrolled in (using ugo_course_student_enrollment table) and they can view the profiles of these students.

Advisors :-

- **Should be able to view the profiles of the students that he/she is allotted to.**

  In the ugo_student table, once the advisor ID's are assigned, the advisor can view those student profiles assigned to them.

- **Can edit details of the profiles of their students if necessary (adding achievements, change in contact details, etc.)**

  In the ugo_students table, once the Advisor id's are assigned, the advisor can edit the profiles of those students assigned to them.

Achievements.

- **Counselor can add an achievement and that will be updated in the student profile.**

  In the ugo_students table, once the Counselor id's are assigned, the Counselor can add achievements, to those students assigned to them, in the ugo_student_achievement table.

Any Disciplinary actions or past negative remarks.

Councilor can add any negative remark or a disciplinary action taken against a student (like getting caught copying in an exam) and that will be updated in the student profile. In the ugo_students table, once the Counselor id's are assigned, the Counselor can add the negative remarks, to those students assigned to them, in the ugo_student_disciplinary table.

Basic Details (like name, address, contact number, email, photo, etc.).

Basic details are to be filled by the student after the first login thereafter he/she view or edit their details.

Academic Details (Past SGPA's and grades as well as the current CGPA, graphs to indicate progress).

CGPA and SGPA mapping not yet covered.

Attendance Details.

Ugo_attendance table deals with the number of classes taken for a particular course and ugo_absentee table monitors whether a student is present or absent for a particular period.

Parents' Details.

Parent table not yet covered.

III. Marks and Grading.

Students :-

- **Should be able to view their internal marks (periodical marks, test result, assignment score) and grades.**

  This can be done using the ugo_student_mark table. This table takes the student id (student_sid) and the course id (course_offering_coid) to give all the academic results of the student.

- **Should be able to obtain their current progress (graphs).**

  This can also be obtained using the ugo_student_mark table. This will show the details of all the marks. But, in order to get the current progress, the marks of the student (student_sid) until the current examinations can be viewed and mapped into a graph if needed.

- **Should be able to view class average marks, highest marks in class (only the marks will be seen, not the one who scored).**

  This can be done using the ugo_student_mark table. From this table, all the marks are scanned for every (student_sid) and the maximum out of them, average, least can be identified.
- **Should be alerted when the grades and marks are published (notifications).**

  This can be done by using the Notification_Student table. Using this, the student has to be notified whenever a new entry is made to the table I.e whenever a new NID is added to that table.

Faculty :-

- **Should be able to decide the weightage for internal exams, continuous evaluation, etc.**

This can be done using the ugo_course_plan table. Using this table through the (course_offering_coid) attribute which acts a foreign key taken from ugo_course_offering table.

- **Should be able to decide how many assignments or tests are to be conducted.**

  This is done using the ugo_continuous_eval table. This table gives the details of all the tests and assignments for that particular course using the (course_offering_coid) attribute. The activity is identified using the activity_type_actid attribute taken from the ugo_activity_type table.

- **Should be able to enter, edit and view marks/grades of all students studying the course that he/she teaches.**

  The marks can be entered/viewed using the ugo_student_mark table. The marks of a particular student for a particular course is identified by the (course_offering_coid) attribute and the student_sid attribute which are foreign keys taken from ugo_student and ugo_course_master respectively.

Advisors/Councilors :-

- **Should be able to view marks/grades of all the students allotted to him/her.**

  The councilor can view the list of all students under him/her using his/her id in the Counselor_id attribute from the ugo_student table. Then, using the student_id generated, these id's can be used in the ugo_student_mark table and ugo_student_continuous_eval table to get the marks/grades.

IV. Communication.

1:1 Chat.

- **Faculty – Faculty**

  Can be implemented using the Chat_Faculty_Faculty table. This is done by entering the Faculty1_Id and Faculty2_ID to communicate. These IDs are referenced to the ugo_faculty table.

- **Student – Student**

  N.A

- **Student – Faculty**

  This is implemented using the Chat_Faculty_Student table. The Student_ID and the Faculty_ID are referenced to ugo_student and ugo_faculty respectively. The message is saved in the Message attribute.

- **Faculty – Student**

  This is also implemented using the Chat_Faculty_Student table. The Student_ID and the Faculty_ID are referenced to ugo_student and ugo_faculty respectively. The message is saved in the Message attribute.

Notifications.

- **Faculty should be able to post notification for a student/group of students/batch/semester/department.**

  This can be done by adding the notification message to the Notifications_Master table and the NID must be added to the Notifications_Student table along with all the SIDs of the students. The SIDs of the required students have to be retrieved from the ugo_student and ugo_course_student_enrollment table.

V. Leave Monitoring.

Advisors/Councilors :-

- **A councilor can enter details about the leave of students when they submit their leave forms (medical leave / duty leave).**

  This is done using the ugo_student_leave table. This table is filled by the counselor and it contains all the details of the leave. The leave_type_id attribute is taken from the ugo_leave_type table which contains the description of the type of leave.

- **When that happens, a notification is triggered for the concerned faculty and they can update the attendance of the student.**

  Whenever a new entry is made to the ugo_student_leave table, using the (student_sid) attribute, all the faculty (faculty_fid) that are undertaking the courses (course_offering_coid) registered by the student for that semester have to be

notified. The message is added to the Notifications_Master table and the NID is added along with all the FID in the Notification_Faculty table.

Project Report

Challenges faced

Faced several technical difficulties while modelling the database, such as inconsistencies, and lack of normalization during the design phase

- Solved by iterating through several database models which allowed us to pinpoint the inconsistencies and iteratively correct them, while identifying normalization faults, and attempting to correct them.

Distinguishing between dated entries and active ones

- Decided to implement active/inactive column for every entry to denote whether the entry is currently active or not. Outdated entries will be marked as inactive.

Differentiate courses for different sections/branches/years/etc.

- We assign different course_number to classes being taken for the same course id for different sections, branches, etc. with a separate allocation table containing the information pertaining to which student is attending which course_number class.

Keeping track of syllabus changes (Having a different course for every new syllabus, etc.)

- We have a course_code associated with the year in which that version of the course code was revised as the primary key, so {CSE320, 2014} is distinct from {CSE320, 2016}, if 2014 had some syllabus, which was revised in 2016.

Figuring out how to handle redo students and courses

- In the table student_allocation, we specify which student is attending which course_number, and whether it's a regular course for him, a redo, etc.

Figuring out how to handle absentees and attendance

- We realized that in order to keep track of who skipped which class and when, we need to also keep track of which course_no was taken for which hour on which day, and for each such period, have a separate set of absentees. Using this information, we can figure out all related information such as number of classes taken for that course_no in a semester. The attendance records of a student who is taking that course_number, etc.

Dense E-R Design

- With the number of tables and data required for an app of this scope, rendering that amount of information in an E R diagram, leads to a very confusing diagram. We decided that it was best to split the information. Two diagrams could convey the information more clearly, one defining the entities and their attributes, and another defining the core entities (such as students faculty) and the relation between them, which uses various tables, without mentioning the attributes. The result is still very complex. However, it is understandable, when compared to a pure E-R diagram.

Converting hierarchical design to relational design

- We were faced with the possibility that the hierarchical structure that is inherent in Institutes, such as a flow of information from HoD - > Counselors - > Class Reps -> Students, among others, would not be reflected in a relational model. However, upon further analysis, we came to realize that, in fact, this could be taken care of by a relational design, if we define the relations properly.

Populating the Database

- We did not have sufficient information to populate the database, as the information is not publically available. However, we used the information that was available (namely students from 2014 CSE Batch), to populate those tables, and our team created an imaginary scenario of few faculty, courses, etc. to be able to enter dummy structured data into the database.

Work Completed

- Modelling the Database
- Entity Relationship Diagram
- Data Dictionary
- Database Creation
- Partial Population of Database

Work Pending

- Authentication and Access privileges model
- Queries according to service requirement
- Completely Populating the Database