CS301 : Databases

Course Project (Phase A) document

Group 7

Team members:

1. Shivam Pandey (2019CHB1055)
2. Deepan Maitra (2019CSB1044)

Table of Contents:

A. TABLES and SCHEMAS

(Primary keys: underlined)

Main tables:

1. **Student** (st_id, first_name, last_name, department, batch, degree)
   Has details of all the students in the institute

2. **Instructor** (ins_id, first_name, last_name, department, age, position)
   Has details of all faculty or instructors of the institute

3. **Course_catalogue** (course_id, course_name, department, L, T, P, S, C)
   Has the details of all the courses of the institute, out of which some are offered every semester

4. **Course_offering** (course_id, section, year, semester, ins_id, timetable_slot , prerequisites, min_cgpa, eligible_batches, eligible_departments)
   Has the details of the courses being offered in the current semester, including the faculty offering the course, the section he/she is teaching, and the timetable slot of the course. This table is a relationship between the Instructor and the Course_catalogue table

[assumption: the current semester is **year 2022, semester 1**. We are dealing with the functionality of this current semester only]

Course tables (one table created for each course offered by a faculty, for each section)

5. **[Course_id]_section[section_no]_[year]_[semester]**
   (st_id, first_name, last_name, grade)
   When a faculty floats a course in the current semester for a section, a table is dynamically created for that course_id and section, being taught in a particular year and semester. Each such table has the details of all the students who successfully register for that course. The instructor uploads the grades of all students taking that course in this table (Assumption: a teacher teaches only *one section* of a course, but he/she might teach multiple courses)

Student tables (for each student, we have 'transcript' table and current semester 'takes' table)

6. **[student first name]_[st_id]_takes** (course_id, section, timetable_slot)
   For each student, this table has the list of the courses which he/she is doing *this semester*, along with the section he/she is in and the timetable slot of the course offering. This table has only the ongoing course details of the student.

7. **[student first name]_[st_id]_transcript** (course_id, year, semester, credits, grade)
   For each student, this table will hold all the courses which he/she has done previously (course history) as well as the courses being done in the current semester. All the courses will have the grades allotted. For the courses being done currently, grades will be fed by the Dean afterwards from the course tables.

Ticket tables (for the ticket generation, there will be separate ticket tables for each student, each faculty, each branch advisor and one for the Dean Academics)

8. **[st_id]_ticket** (ticket_id, course_id, section)
   When a student violates the 1.25 rule of credit limit while registering for a course, a ticket ID is generated and added in this table. For each student, his/her ticket table will have all the tickets raised by him/her.

9. **[instructor first name]_[ins_id]_ticket**
   (ticket_id, st_id, course_id, section, instructor_decision)
   Each instructor goes through all the student tickets and collects the tickets pertaining to the course(s) being currently taught by him/her. For each instructor, this table will have the tickets which he/she has to respond to.

10. **advisor_[department]_ticket**
    (ticket_id, st_id, course_id, section, advisor_decision)
    Each branch advisor goes through all the student tickets and collects the tickets raised by students of his/her advising branch. For each branch advisor, this table will have the tickets which he/she has to respond to.

11. **dean_ticket**
    (ticket_id, st_id, course_id, section, instructor_decision, advisor_decision, dean_decision)
    The ticket table of the Dean Academics has all the tickets of every student, since the Dean has to respond to all tickets. If the Dean responds 'Yes' to a ticket after the respective Instructor and Advisor has responded, the student is allowed to take the course (above the credit limit).

Other tables:

12. **timetable_slots** : this has all the available timetable slots (to be uploaded by Dean)
13. **departments**: this has all the department names in the institute

B. ROLES and GRANT ACCESSES to TABLES

The following roles are there in the database:

1. **dean_academics:** This is the role for the Dean Academics / the academic section.

2. **Student roles:** There is a separate role for each student in the format of **[st_id]**
   (For example, *2019MEB1077* and *2019EEB1666* are separate roles corresponding to the student)

3. **Instructor roles:** Each instructor has separate roles assigned in the format of **[instructor first name]_[ins_id].** (For example, *Karishma_2* and *Swati_6* are separate roles for two instructors)

4. **Branch advisor roles:** Each branch advisor has separate roles assigned in the format of **advisor_[department].** (For example, *advisor_CS* and *advisor_ME* are roles for the advisors of the CS and ME branch)

The tables and their respective grant accesses are explained below:

1. **Student** and **Instructor** table: all roles have VIEW access, only Dean has EDIT access.
2. **Course_catalogue** table: all roles have VIEW access, only Dean has EDIT access.
3. **Course_offering** table: Students and advisors have VIEW access. All the Instructors and the Dean have EDIT access.
4. **Course tables** (for each course being offered) : Only the Instructor who will teach a particular section of a course, will have EDIT (insert and update) access to the table for that course offering. Other instructors will not have access to the course tables which he/she is not teaching. All the advisors and Dean have VIEW access to all the course tables. No student has any access (not even VIEW) to any of the course tables (this is where the grades will be uploaded, so security issues will turn up if a student can see others grades)

5. **Student tables (takes** table and **transcript** table): A student has INSERT access to only his 'takes' table. He only has VIEW access to his 'transcript' table. (the 'takes' table is only a list of courses being done by the student in the ongoing semester but doesn't have a grades column. The 'transcript' table has the grades of previous and current courses, hence no Edit access is given) Instructors will have VIEW access to all student 'transcript' and 'takes' table. Dean has EDIT access to all the student 'transcript' and 'takes' tables (for transferring grades from the course tables and to insert the course in the student's 'takes' table after validation using the trigger)

6. **Tickets table:**

    - **Student tickets** tables: each student will have INSERT access to only his/her own tickets table. All the instructors, advisors and Dean will have VIEW access to all the student ticket tables.
    - **Instructor Ticket** table: each Instructor will have INSERT and UPDATE access to only his/her own tickets table. Dean will have VIEW access.
    - **Advisor ticket** table: each advisor will have INSERT and UPDATE access to only his/her own tickets table. Dean will have VIEW access.
    - **Dean ticket** table: Only the Dean will have INSERT and UPDATE access to his ticket table. Instructors and Students will have the VIEW access.

C. STORED PROCEDURES :

Some preliminary procedures/functions for bulk usage:

a) Procedures to create users:
    - create_student_user( ) : creates all the student roles
    - create_instructor_user( ) : creates all the instructor roles
    - create_advisor_user( ): creates all advisor roles for each department

b) Procedures to grant access to users as specified above:
   - give_access_to_dean( )
   - give_access_to_advisor( )
   - give_access_to_instructor( )
   - give_access_to_student( )

c) Make various tables/collection of tables:
   - create_student_transcripts( ) : makes the 'takes' and 'transcript' table for each student (*[student first name]_[st_id]_transcript* and *[student first name]_[st_id]_takes*)
   - make_student_ticket_tables( ): makes the individual ticket table of each student.
   - make_instructor_ticket_tables( ): makes the individual ticket table of each instructor.
   - make_advisor_ticket_tables( ): makes the individual ticket table of each advisor.
   - make_dean_ticket( ): makes the Dean ticket table

d) **upload_student_history( ) :** this procedure is used to fill the transcripts of each student with their respective course history (grades of courses previously done by him). It uploads CSV files for each student (from a local computer location) and copies the course details with the grades into the student transcript table.

e) **offer_course** (*course_id, section, year, semester, slot, prereq, min_cgpa, batches, depts*)

This procedure is called by each faculty using their respective login access. He/she must give necessary arguments like the course_id and section of the course he/she will be teaching and other constraints like the time-table slot, prerequisites, minimum CGPA criteria, eligible batches and eligible departments for the course. Checks will be done if the course exists in the course_catalogue and whether the timetable slot of this course is clashing with another course the same instructor is teaching.

f) **register_course** (*course_id, section*)

This procedure is called by each student using his/her own login to register for a course being offered. While registering, several checks will be performed:

- if this course is actually being offered in the desired section
- if the student already has registered for a course in the same time-table slot as this new course
- if the student is from the eligible department and batch as mentioned in course_offering
- if the student has previously cleared the mentioned prerequisites for this course
- if the current CGPA of the student is satisfying the minimum CGPA criteria
- The 1.25 credit limit: check will be performed if the student is violating the 1.25 credit limit based on the credits earned in the previous two semesters. If yes, then ticket generation procedure generate_ticket( ) will be called.

If all these criteria are satisfied, then the course is added to the 'takes' table of the student.

g) **generate_ticket**(*ticket_id, st_id, course_id, section*) : This is the ticket generation routine that gets called within the register_course( ) procedure if the 1.25 rule is violated. It adds the ticket_id to the tickets table of that particular student.

h) **copy_to_course_tables( ):** After all the students have registered for all their desired courses, the Dean Academics will execute this procedure to go to each student's 'takes' table, loop through all the registered courses and then add that student details to the course table of that particular course.

i) **upload_grades**(*course_id*, *section, year, semester*) : This will be called by each instructor using their respective login access, to upload grades for a course they are teaching. The grades will be uploaded from a CSV file and the respective course table will be filled.

j) **copy_grades_to_transcript**( ) : this will be executed by the Dean once to go to all the course tables, and for each student registered for each course, the respective grade will be copied to the student's transcript table. The student has only VIEW access to his/her own transcript table, and so he/she can see the grades.

k) **report_generation**(*st_id*): will be used by the Dean Academics office to print the transcripts of any student. This will also call the CGPA calculation function to display the CGPA of the student.

l) **calculate_cgpa**(st_id) : utility function to calculate the CGPA of a student.

m) **grade_conversion**(grade): utility function that converts a letter grade (A, B- etc.) to the numeric grade (10,8 etc.)

n) **upload_timetableslots**( ) : called by the academic section at the beginning of the semester to enter the available timetable slots from a CSV file and fill the timetable_slots table.

Ticket propagation functions:

a) **extract_ticket_instructor**( ): called by each instructor. Goes through all the student ticket tables and collects the tickets pertaining to the courses being taught by that instructor,to add to his own ticket table.

b) **extract_ticket_advisor**( ): called by each advisor. Each branch advisor goes through all the student tickets and collects the tickets raised by students of his/her advising branch.

c) **extract_ticket_dean**( ): called by Dean Academics. Adds all generated tickets to his own ticket table.

d) **instructor_ticket_response**(*ticket_id, response*): will be called by each instructor to give his/her decision for each ticket in his/her own ticket table.

e) **advisor_ticket_response**(*ticket_id, response*): will be called by each advisor to give his/her decision for each ticket in his/her own ticket table.

f) **dean_copy_response**( ) : Dean will execute this only once, after all instructors and advisors have given their decision responses to each of their respective tickets. For each ticket of any student, this copies the instructor and advisor decision pertaining to that particular ticket and puts that in the Dean's ticket table instructor_decision and advisor_decision column.

g) **dean_ticket_response**(*ticket_id, response*): will be called by Dean for each ticket to provide his final decision response.

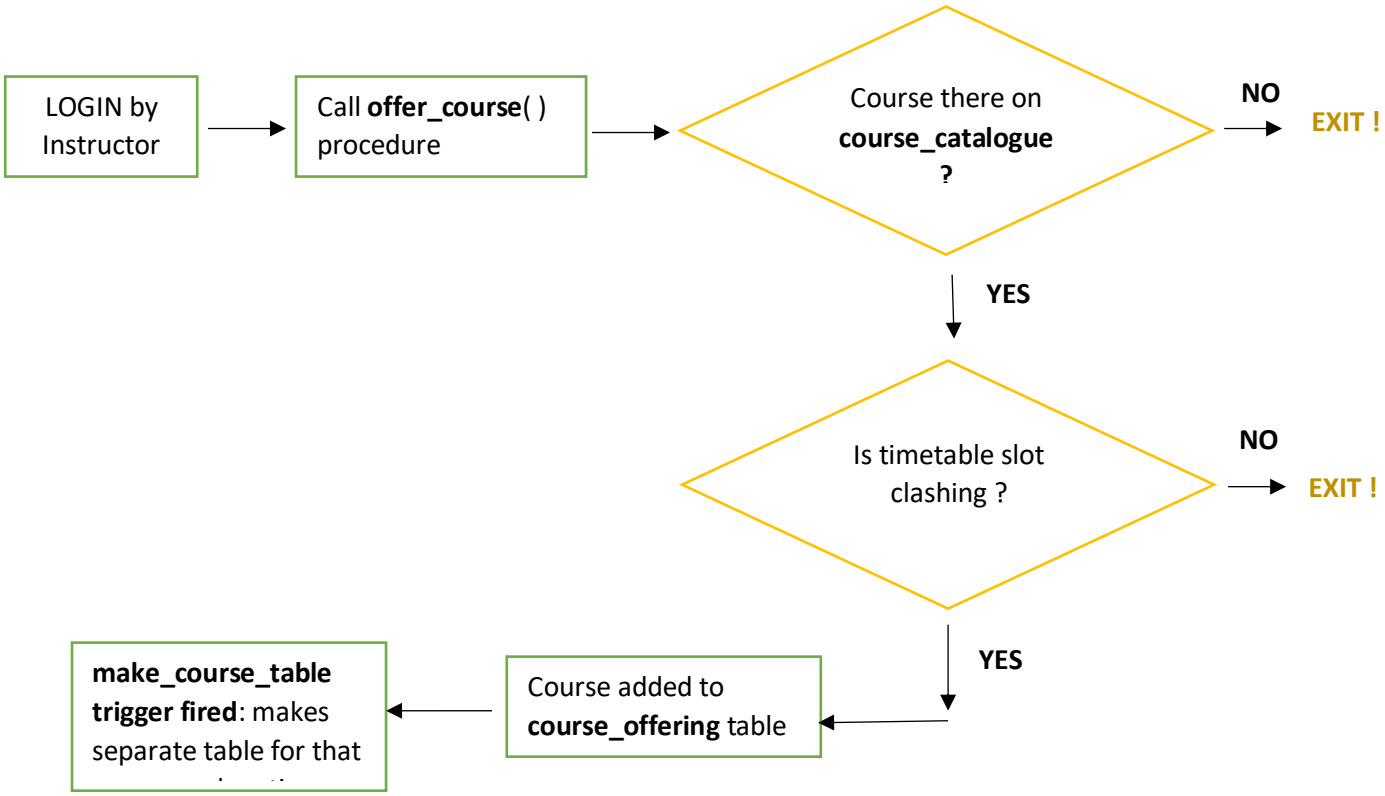
D. TRIGGERS:

a) **make_course_table** : whenever a course is offered by a faculty (added to the course_offering) this trigger is fired which creates a course table for that course and section. (in the format: **[Course_id]_section[section_no]_[year]_[semester]**)

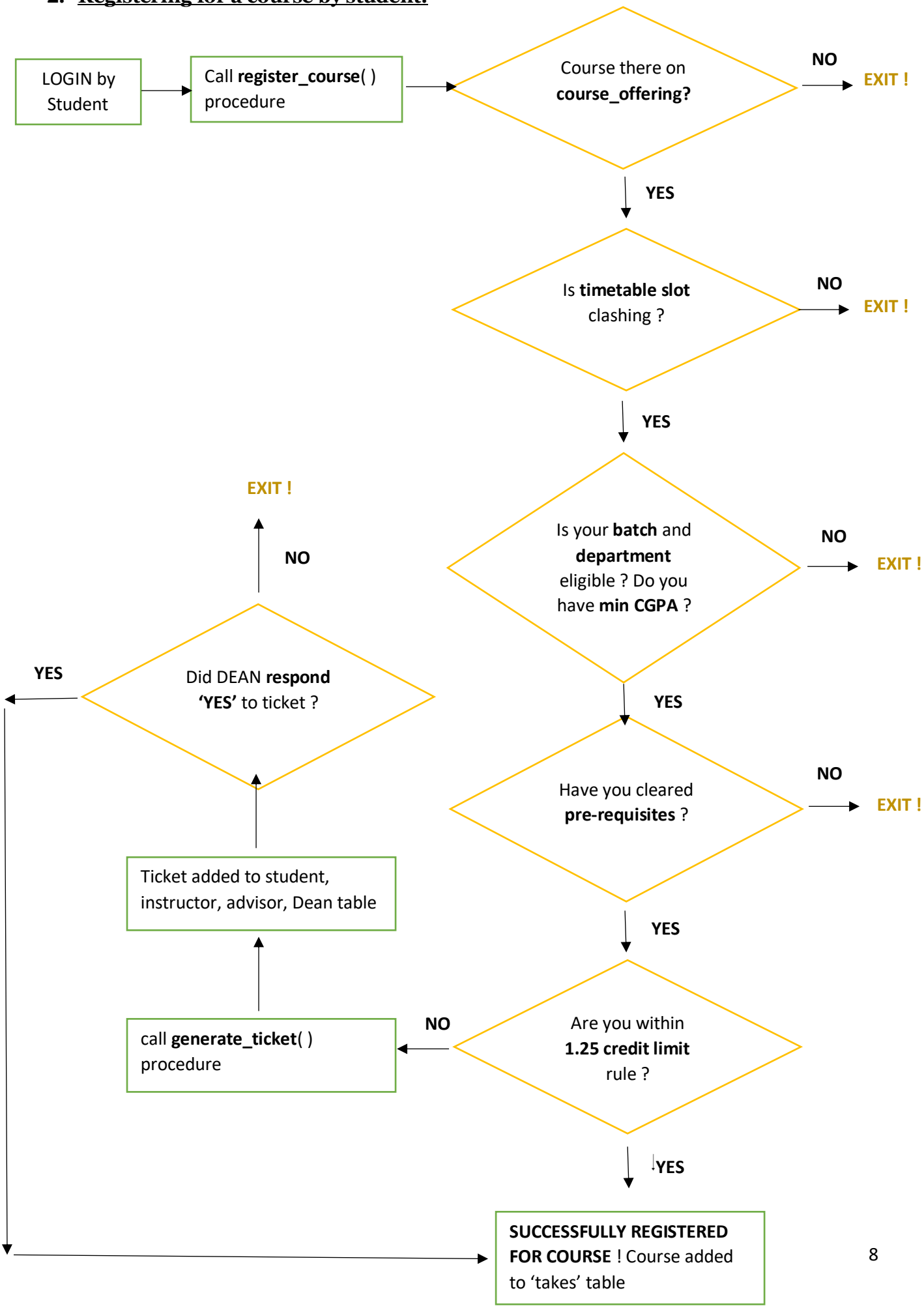
b) **dean_decision** : this trigger is fired when a Dean responds 'Yes' to any of the generated tickets. When this happens, a tuple is added to the 'takes' table of that particular student **[student first name]_[st_id]_takes** (signifying that the student has successfully registered for the course)

E. FUNCTIONALITES:

1. Offering a course by faculty

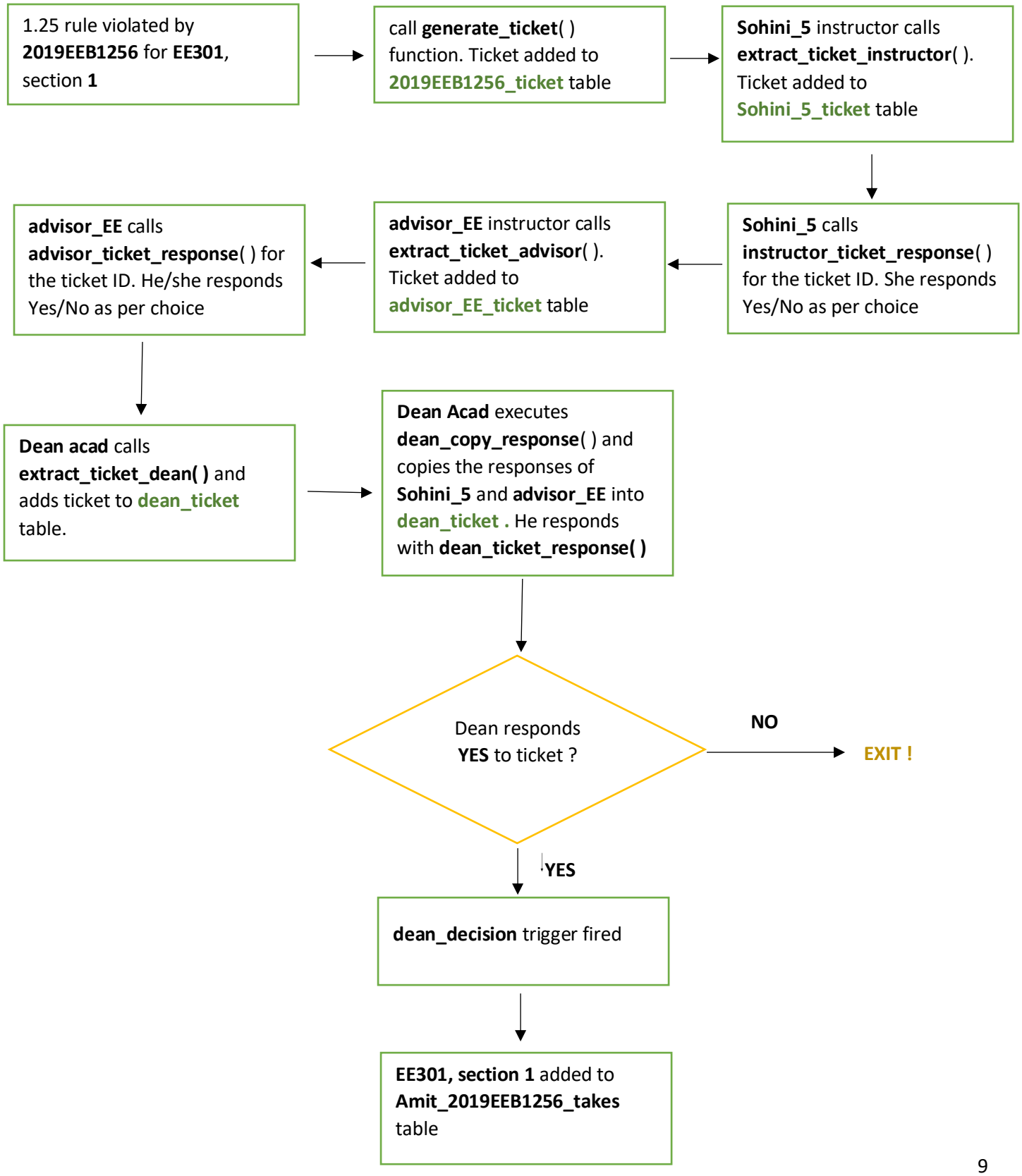


2. Registering for a course by student:



3. Ticket generation and Propagation:

For instance, suppose student *2019EEB1256* (*Amit Das*) raises a ticket for course *EE301*, *section 1* which is being taught by instructor role *Sohini_5*. The corresponding branch advisor will be *advisor_EE*



Assumptions in project:

1. An instructor can teach multiple courses, but only a single section of any course
2. We are considering *year 2022, semester 1* as current semester. The whole logic is based upon this (so 2021 semester 1 and 2021 semester 2 are the last two semesters)
3. Since all Instructors have INSERT access to the same course_offering table, we have assumed that an instructor will only float courses with his/her own instructor ID and not pose as another instructor.
4. As mentioned in class, when a ticket is generated, the corresponding instructor and advisor can respond Yes/No (responses are stored too) However, the final verdict is of the Dean Academics (if a scenario arises where both instructor and advisor have responded with NO, but Dean has responded with YES, then also the course can be taken by the student)
5. The ticket propagation is assumed to be happening in stages. The first stage is ticket generation, where all tickets are generated by students. After this phase, all the instructors and the advisors fill their own ticket tables at once. After they've all responded to their corresponding tickets, the Dean Academics copies their responses to his own ticket table. He then gives Yes/No to all tickets at once.