**Team Report**

**Team Members:**

Name: Harshada Rode

BNumber: B00743407

Bmail: [hrode1@binghamton.edu](mailto:hrode1@binghamton.edu)

Harshada Rode

**Plan for completing the project:**

|  |  |  |
| --- | --- | --- |
| **Date** | **Planned Tasks** | **Actually Completed Tasks** |
| 11/10 | 1. GUI Plan 2. Setting up of execution environment | 1. GUI plan created 2. Setting up of execution environment completed |
|  |  |  |
| 11/11 | 1. GUI implementation 2. Query 1, 2, 3 implementations | 1. GUI implementation started 2. Query 1, 2, 3 implementations completed |  |
|  |  |  |  |
| 11/17 | 1. GUI implementation for Query 4,5,6 and implementation of queries. | 1. GUI implementation for Query 4,5,6 and implementation of queries completed. |
|  |  |  |
| 11/18 | 1. GUI implementation for Query 7, 8 and implementation of queries. | 1. GUI implementation for Query 7, 8 and implementation of queries completed. |
|  |  |  |
| 11/22 | 1. Testing and report generation | 1. Testing and report generation completed. |

**Procedures implemented in project:**

* procedure displayStudents(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays students table.
* procedure displayTAs(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays tas table.
* procedure displayCourses(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays courses table.
* procedure displayClasses(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays classes table.
* procedure displayEnrollments(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays enrollments table.
* procedure displayPrerequisites(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays prerequisites table.
* procedure displayLogs(c\_dbcursor OUT SYS\_REFCURSOR);
* Displays logs table.
* procedure get\_TA\_of\_class(class\_id in classes.classid%type,c\_dbcursor OUT SYS\_REFCURSOR,status out number);
* Returns B#, first\_name, last\_name of TA of the class.

* procedure getPreReq(deptCode in prerequisites.dept\_code%type, courseNum in prerequisites.course#%type, c\_dbcursor OUT SYS\_REFCURSOR, status out number);
* Returns dept\_code and course# of prerequisite courses for a given course with dept\_code and course# as inputs.
* procedure checkIfStudentIsPresent(bnum in students.B#%type, status out number);
* Checks if a student with input B# is present in students table.
* procedure checkIfClassIsPresent(class\_Id in classes.classid%type, status out number);
* Checks if a class with input classid is present in classes table.
* procedure getClassSemNYear(class\_Id in classes.classid%type, semester1 out classes.semester%type, year1 out classes.year%type, limit1 out classes.limit%type, class\_size1 out classes.class\_size%type);
* Returns class information for a class specified with input classid.
* procedure getEnrollments (bnum in enrollments.B#%type, class\_Id in enrollments.classid%type, status out number);
* Checks if a student is enrolled in a class.
* B# and classid are inputs.
* procedure isStudentOverloaded (bnum in enrollments.B#%type, cnt out number);
* Returns no. of classes a student is enrolled in.
* Input is B#.
* procedure checkStudentEnrollment(bnum in enrollments.B#%type,class\_Id in enrollments.classid%type, status out number);
* Checks if a student is enrolled in a specific class.
* B# and classid are input parameters.
* procedure prereqViolation(bnum in enrollments.B#%type, class\_Id in enrollments.classid%type, status out number);
* Used to check if dropping the student from the class would cause a violation of the prerequisite requirement for another class.
* Used while dropping a student from a class.
* procedure totalEnrollmentForStudent(bnum in enrollments.B#%type, status out number);
* Returns total classes taken by student.
* B# is the input parameter.
* procedure dropStudentFromClass(bnum in enrollments.B#%type, class\_Id in enrollments.classid%type);
* Delete a student entry from enrollments table.
* B# and classid are input parameters.
* procedure deleteStudentFromStudentTable(bnum in enrollments.B#%type);
* Deletes a student entry from students table.
* B# is the input parameter.
* procedure checkPreReqCondition(bnum in enrollments.B#%type, class\_Id in enrollments.classid%type, status out number);
* Used to check if the student has completed the required prerequisite courses with at least a grade C.
* Used while enrolling a student into a class.
* procedure enrollStudentInClass(bnum in enrollments.B#%type, class\_Id in enrollments.classid%type);
* Inserts a student and class in enrollment table with lgrade as null.
* Used to enroll a student into a class.
* B# and classid are provided as inputs.
* procedure checkIfClasshasTA(class\_Id in classes.classid%type, status out number);
* Checks if a class provided as input has a TA.
* Classid is input.

**Triggers implemented:**

* enroll\_student\_trigger
* Adds on entry into logs
* Increments class\_size by 1.
* drop\_from\_class\_trigger
* Adds on entry into logs
* Decrements class\_size by 1.
* delete\_student\_trigger
* Adds on entry into logs
* Deletes student records for a specific student from enrollments table.

**Java Pages implemented:**

* Login:
* Verifies user credentials and logs in.
* MyFrame
* Provides options to the user to perform various operations.
* Display
* Displays various tables based on input table name
* TAofClass
* Gives TA information based on input classid.
* Prerequisites
* Gives prerequisites of a course based on input course# and dept\_code
* EnrollStudentIntoClass
* Allows user to enroll a student in the class if required conditions are satisfied.
* DropStudentFromClass
* Allows user to drop a student from a class if required conditions are satisfied.
* DeleteStudentFromStudentT
* Allows user to delete a student from students table and subsequently deletes student’s records from enrollments table for that specific student.