Student Registration System

Design Document

Submitted By-
Saurabh Chaudhari
Triveni Banpela

Project Description:
The main objective of the project is to create Student Registration System by using Oracle's PL/SQL and JDBC. The system should proceed with course enrollment and student registration according to requirement document.

Design description of the front end methods and the backend procedures:
JAVA Methods:

Method Name	**Details**
deleteStudent	Delete student details
dropEnrollment	Delete enrollment details
enrollStudent	Enroll student
getPrerequisites	Get prerequisites details
printTableData	Print table data as per user's input
getChoice	Get user's input
printMenu	Show the options available to user on screen
getClassInfo	Get Class information
getStudentInfo	Get Student Information
InsertStudentRecord	Insert Student information

Main Procedures:

Procedure	Description
show_logs	Display logs table data
show_students	Display student table data
show_courses	Display courses table data
show_enrollments	Display enrollments table data
show_classes	Display classes table data
show_PREREQUISITES	Display Prerequisites table data
insert_student	Insert details in Student table
get_student_info	Displays student information
get_prerequisites	Gives the list of prerequisites of couse
get_class_info	Gives information about the class
enrollment_student	Enroll a student into a class.
drop_enrollment	Drop a student from a class
delete_student	Delete student record from

Helper Procedures Description:

These Procedures are called within Main procedure for various validity check.

Procedure	Description
student_validity_check	Check for duplicated sid
emailid_validity_check	Check for duplicate email id
convert_boolean	Convert Boolean to int
class_validity_check	Check if classid already present or not
class_space_availability_check	Check for class_size<limit
enrollment_check	Check if student already enrolled
enrollment_count_check	Check total classes that student is registered current semester
CHECK_PREREQUISITES_GRADES	Check If student has completed all prerequisites with min grade D
prerequiste_voilation_check	Check if current course is prerequisite of anther course he is registered
last_class_check	Check is student is not enrolled in any class.
last_student_check	Check if class has no students enrolled

Triggers Defined:

Trigger Name	Details
ENROLLMENT_INSERT	Its invoked After insert on Enrollments table
ENROLLMENT_INSERT_LOG	Its invoked After insert on Enrollments table
ENROLLMENT_DELETE	Its invoked After delete on Enrollments table
ENROLLMENT_DELETE_LOG	Its invoked before delete on Enrollments table
STUDENT_INSERT_LOG	Its invoked After insert on Students table
STUDENT_DELETE_LOG	Its invoked After delete on Students table

Sequence:

Log_seq: This creates logid from 1000

Methodology:
It’s a Menu driven Interface.
System waits for user’s input from screen and accordingly processes the request provided by user.
User needs to enter any number between 1 to 9 (Valid Inputs).

User Input	Function Called
1	Print Menu and print table data of user’s choice
2	Insert Student Record
3	Get Student Information
4	Get Prerequisites
5	Get Class Information
6	Enroll Student details
7	Drop Enrollment details
8	Delete Student details
9	Exit System

Code:
Package:
Package Name: Database Project:

```
create or replace package databaseproject is

PROCEDURE show_logs(logs_cur OUT SYS_REFCURSOR);
PROCEDURE show_students(students_cur OUT SYS_REFCURSOR);
PROCEDURE show_courses(courses_cur OUT SYS_REFCURSOR);
PROCEDURE show_enrollments(enrollments_cur OUT SYS_REFCURSOR);
PROCEDURE show_classes(classes_cur OUT SYS_REFCURSOR);
PROCEDURE show_PREREQUISITES(PREREQUISITES_cur OUT SYS_REFCURSOR);

PROCEDURE student_validity_check(sidIn IN students.sid%TYPE,
                                                      flag OUT boolean);
PROCEDURE emailid_validity_check(emailIn IN students.email%TYPE,
                                                      flag OUT boolean);
PROCEDURE convert_boolean (flagIn IN boolean, flagOut OUT number);
PROCEDURE insert_student (sidIn IN students.sid%type,
                                    firstnameIn IN students.firstname%type,
                                    lastnameIn IN students.lastname%type,
                                    statusIn IN students.status%type,
                                    gpaIn IN students.gpa%type,
                                    emailIn IN students.email%type,eflag OUT number,
                                    sflag OUT number);


PROCEDURE get_student_info(sidIn IN students.sid%TYPE,
                                         students_cur OUT SYS_REFCURSOR,
                                         classes_cur OUT SYS_REFCURSOR);

procedure get_prerequisites(dept_codeIn in prerequisites.dept_code%type,
                                      course_noIn in prerequisites.course_no%type,
                                      prerequisites_cur out SYS_REFCURSOR);


PROCEDURE get_class_info(classidIn IN classes.classid%TYPE,
                                      class_cur OUT SYS_REFCURSOR,
                                      students_cur OUT SYS_REFCURSOR);


PROCEDURE class_validity_check(classidIn IN classes.classid%TYPE,
                                                  flag OUT boolean);

PROCEDURE class_space_availability_check(classidIn IN classes.classid%TYPE,
                                                              flag OUT boolean);


PROCEDURE enrollment_check(sidIn IN students.sid%TYPE,
                                            classidIn IN classes.classid%TYPE,
                                            flag OUT boolean);


PROCEDURE enrollment_count_check(sidIn IN students.sid%TYPE,
                                            classidIn IN classes.classid%TYPE,
                                            current_enrollment_count OUT number);


procedure CHECK_PREREQUISITES_GRADES(sidIn IN students.sid%type,
                                                        classidIn IN
classes.classid%type,
                                                        flag OUT boolean);

PROCEDURE enrollment_student(sidIn IN students.sid%TYPE,classidIn IN classes.classid%TYPE,
                                            status OUT Number, enrollement_count OUT
number);

procedure prerequiste_voilation_check(sidIn IN students.sid%TYPE,
                                                              classidIn IN
classes.classid%TYPE,
                                                              flag OUT boolean);

procedure last_class_check(sidIn IN students.sid%TYPE,
                                            status OUT Number);

procedure last_student_check(classidIn IN classes.classid%TYPE,
                                            status OUT Number);

procedure drop_enrollment(sidIn IN students.sid%TYPE,
                                      classidIn IN classes.classid%TYPE,
                                            status OUT Number, last_class OUT Number,
                                            last_student OUT Number);


procedure delete_student(sidIn IN students.sid%TYPE,
                                            status OUT Number);

end databaseproject;
/

create or replace package body databaseproject is

PROCEDURE show_logs(logs_cur OUT SYS_REFCURSOR)
IS
BEGIN
   open logs_cur for
   select * from logs;
END show_logs;


PROCEDURE show_students(students_cur OUT SYS_REFCURSOR)
IS
BEGIN
   open students_cur for
   select * from students;
END show_students;


PROCEDURE show_courses(courses_cur OUT SYS_REFCURSOR)
IS
BEGIN
   open courses_cur for
   select * from courses;
END show_courses;


PROCEDURE show_enrollments(enrollments_cur OUT SYS_REFCURSOR)
IS
BEGIN
   open enrollments_cur for
   select * from enrollments;
END show_enrollments;


PROCEDURE show_classes(classes_cur OUT SYS_REFCURSOR)
IS
BEGIN
   open classes_cur for
```

```
    select * from classes;
END show_classes;

PROCEDURE show_PREREQUISITES(PREREQUISITES_cur OUT SYS_REFCURSOR)
IS
BEGIN
    open PREREQUISITES_cur for
    select * from PREREQUISITES;
END show_PREREQUISITES;

PROCEDURE emailid_validity_check(emailIn IN students.email%TYPE, flag OUT boolean)
As
i number;
Begin
select COUNT(*) into i from students where email = emailIn;
if (i != 0) then
  flag := true;
else
  flag := false;
end if;
end emailid_validity_check;


PROCEDURE student_validity_check(sidIn IN students.sid%TYPE,flag OUT boolean)
As
i number;
Begin
select COUNT(*) into i from students where sid = sidIn;
if (i != 0) then
  flag := true;
else
  flag := false;
end if;
end student_validity_check;


PROCEDURE convert_boolean
(flagIn IN boolean,
flagOut OUT number)
as
Begin
if (flagIn = true)
  then flagOut := 1;
  else
  flagOut:=0;
  end if;
end convert_boolean;
```

```
PROCEDURE insert_student
(sidIn IN students.sid%type,
firstnameIn IN students.firstname%type,
lastnameIn IN students.lastname%type,
statusIn IN students.status%type,
gpaIn IN students.gpa%type,
emailIn IN students.email%type,
eflag OUT number,
sflag OUT number)
AS
email_flag boolean;
sid_flag boolean;
BEGIN
student_validity_check(sidIn,sid_flag);
EMAILID_VALIDITY_CHECK(emailIn,email_flag);
if (sid_flag = false) then
  if (email_flag = false) then
    Insert into SYSTEM.STUDENTS (SID,FIRSTNAME,LASTNAME,STATUS,GPA,EMAIL) values
      (SIDIn,FIRSTNAMEIn,LASTNAMEIn,STATUSIn,GPAIn,EMAILIn);
      commit;
  end if;
end if;
CONVERT_BOOLEAN(sid_flag,sflag);
CONVERT_BOOLEAN(email_flag,eflag);
END insert_student;


PROCEDURE get_student_info(sidIn IN students.sid%TYPE,
                                                  students_cur OUT SYS_REFCURSOR,
                                                  classes_cur OUT SYS_REFCURSOR)
As
Begin
  open students_cur for
  select sid,lastname,status from STUDENTS where sid = sidIn;
  open classes_cur for
  select cl.classid, concat(cl.dept_code,cl.course_no) as course_id,co.title,cl.year,cl.semester
  from classes cl, courses co
  where cl.dept_code = co.DEPT_CODE and cl.course_no=co.course_no;
end get_student_info;

procedure get_prerequisites(dept_codeIn in prerequisites.dept_code%type,
course_noIn in prerequisites.course_no%type, prerequisites_cur out SYS_REFCURSOR )
IS
    cursor pre_req_cursor is
    select pre_dept_code, pre_course_no from prerequisites
    where dept_code = dept_codeIn and course_no =course_noIn;
```

```
            prerequisites_row pre_req_cursor%rowtype;

            begin
                    insert into temp select pre_dept_code,pre_course_no from prerequisites
                            where dept_code=dept_codeIn and course_no=course_noIn;
                    OPEN pre_req_cursor;
                    LOOP
                            fetch pre_req_cursor into prerequisites_row;
                            EXIT when pre_req_cursor%NOTFOUND;

            get_prerequisites(prerequisites_row.pre_dept_code,prerequisites_row.pre_course_no,prerequi
sites_cur);
   END LOOP;
   OPEN prerequisites_cur FOR
     select * from temp;
   close pre_req_cursor;
END get_prerequisites;


PROCEDURE get_class_info(classidIn IN classes.classid%TYPE,
                                              class_cur OUT SYS_REFCURSOR,
                                              students_cur OUT SYS_REFCURSOR)
As
Begin
  open class_cur for
  select cl.classid,co.title,cl.semester,cl.year from CLASSES cl,COURSES co
  where classid = classidIn  and cl.DEPT_CODE = co.DEPT_CODE and cl.COURSE_NO = co.COURSE_NO;
  open students_cur for
  select e.sid,s.lastname
  from students s, enrollments e
  where e.CLASSID = classidIn and e.sid = s.sid;
end get_class_info;



PROCEDURE class_validity_check(classidIn IN classes.classid%TYPE, flag OUT boolean)
As
i number;
Begin
select COUNT(*) into i from classes where classid = classidIn;
if (i != 0) then
  flag := true;
else
  flag := false;
end if;
end class_validity_check;
```

```
PROCEDURE class_space_availability_check(classidIn IN classes.classid%TYPE, flag OUT boolean)
As
class_size_var number;
limit_var number;
begin
select class_size,limit into class_size_var,limit_var from classes where classid = classidIn;
if(class_size_var < limit_var) then
flag := true;
else
flag := false;
end if;
end class_space_availability_check;


PROCEDURE enrollment_check(sidIn IN students.sid%TYPE,classidIn IN classes.classid%TYPE, flag OUT
boolean)
AS
enrollment_count number;
begin
select count(*) into enrollment_count from enrollments where sid = sidIn and classid = classidIn;
IF enrollment_count>0
then
flag := true;
else
flag := false;
end if;
end enrollment_check;



PROCEDURE enrollment_count_check(sidIn IN students.sid%TYPE,classidIn IN classes.classid%TYPE,
                                                                current_enrollment_count OUT
number)
As
current_semester classes.semester%type;
current_year classes.year%type;
begin
select count(*) into current_enrollment_count from enrollments where sid = sidIn and classid in
(select classid from classes where (semester,year) in
(select semester,year from classes where classid = classidIn));
end enrollment_count_check;


procedure CHECK_PREREQUISITES_GRADES(sidIn IN students.sid%type,
                                                                classidIn IN classes.classid%type,
                                                                flag OUT boolean)
AS
i INT;
```

```
j INT;
begin
select count(*) into i from  prerequisites  where (DEPT_CODE,COURSE_NO)in
(select dept_code,COURSE_NO from classes where classid  = classidIn);
select count(classid) into j from ENROLLMENTS where lgrade <= 'D' and sid = sidIn and classid in(
select classid from classes where (DEPT_CODE,COURSE_NO) in
(select PRE_DEPT_CODE,PRE_COURSE_NO from prerequisites  where (DEPT_CODE,COURSE_NO)in
            (select dept_code,COURSE_NO from classes where classid  = classidIn)));
if (i=j) then
  flag := true;
else
  flag := false;
end if;
end CHECK_PREREQUISITES_GRADES;




PROCEDURE enrollment_student(sidIn IN students.sid%TYPE,classidIn IN classes.classid%TYPE,
status OUT Number,
enrollement_count OUT number)
as
valid_sid  boolean;
valid_classid  boolean;
valid_class_size  boolean;
enrollement_check  boolean;
prerequisites_check  boolean;

begin
student_validity_check(sidIn,valid_sid);
class_validity_check(classidIn,valid_classid );
class_space_availability_check(classidIn,valid_class_size);
enrollment_check(sidIn,classidIn,enrollement_check);
enrollment_count_check(sidIn,classidIn,enrollement_count);
CHECK_PREREQUISITES_GRADES(sidIn,classidIn,prerequisites_check);
if(valid_sid = true) then
  if(valid_classid = true) then
    if(valid_class_size = true) then
        if(enrollement_check = false) then
          if(enrollement_count < 3) then
            if(prerequisites_check = true ) then
                insert into ENROLLMENTS (SID,CLASSID) values (sidIn,classidIn);
                        commit;
                status :=1;
              else
              status :=2;
              end if;
          else
          status :=3;
        end if;
      else
        status :=4;
      end if;
  else
    status :=5;
  end if;
 else
   status :=6;
 end if;
else
 status :=7;
end if;
end enrollment_student;


procedure prerequiste_voilation_check(sidIn IN students.sid%TYPE,
classidIn IN classes.classid%TYPE,
flag OUT boolean)
As
i int;
begin
select count(*) into i from enrollments where sid = sidIn and  classid in(
select classid from classes where (dept_code,course_no) in
(select dept_code,course_no from PREREQUISITES where (pre_dept_code,pre_course_no) in
(select dept_code,course_no from classes where classid = classidIn)));
if (i>0) then
flag := true;
else
flag := false;
end if;
end prerequiste_voilation_check;



procedure last_class_check(sidIn IN students.sid%TYPE, status OUT Number)
as
i int;
begin
SELECT count(*) into i FROM enrollments where sid  = sidIn;
if(i = 0) then
status := 1;
else
status := 0;
end if;
end last_class_check;
```

```
procedure last_student_check(classidIn IN classes.classid%TYPE, status OUT Number)
as
i int;
begin
SELECT count(*) into i FROM enrollments where classid  = classidIn;
if(i = 0) then
status := 1;
else
status := 0;
end if;
end last_student_check;


procedure drop_enrollment(sidIn IN students.sid%TYPE,
classidIn IN classes.classid%TYPE,
status OUT Number,
last_class OUT Number,
last_student OUT Number)
As
valid_sid  boolean;
valid_classid  boolean;
enrollement_check boolean;
prerequiste_check boolean;
Begin
last_class := 0;
last_student := 0;
student_validity_check(sidIn,valid_sid);
class_validity_check(classidIn,valid_classid );
enrollment_check(sidIn,classidIn,enrollement_check);
prerequiste_voilation_check(sidIn,classidIn,prerequiste_check);
if(valid_sid = true) then
  if(valid_classid = true) then
    if(enrollement_check = true) then
      if(prerequiste_check = false) then
        delete from enrollments where sid = sidIn and classid = classidIn;
              commit;
        last_class_check(sidIn,last_class);
        last_student_check(classidIn,last_student);
        status:=1;
      else
        status:=2;
      end if;
    else
      status:=3;
    end if;
  else
    status:=4;
  end if;
else
  status:=5;
end if;
end drop_enrollment;




procedure delete_student(sidIn IN students.sid%TYPE, status OUT Number)
as
valid_sid  boolean;
begin
valid_sid := false;
student_validity_check(sidIn,valid_sid);
if(valid_sid = true) then
  delete from students WHERE sid = sidIn;
            commit;
  status := 1;
else
  status := 2;
end if;
end delete_student;


end databaseproject;
/
```

Triggers:

```
Drop trigger ENROLLMENT_INSERT;
Drop trigger ENROLLMENT_INSERT_LOG;
Drop trigger ENROLLMENT_DELETE;
Drop trigger ENROLLMENT_DELETE_LOG;
Drop trigger STUDENT_INSERT_LOG;
Drop trigger STUDENT_DELETE_LOG;


create or replace TRIGGER ENROLLMENT_DELETE
AFTER DELETE ON Enrollments
for each row
BEGIN
  update classes set class_size = class_size-1 where classid =:new.classid;
END;
/

create or replace TRIGGER ENROLLMENT_DELETE_lOG
Before DELETE ON Enrollments
for each row
```

```
BEGIN
Insert into LOGS (LOGID,WHO,TIME,TABLE_NAME,OPERATION,KEY_VALUE) values
  (logs_seq.nextval,(select user from dual),sysdate,'Enrollments','delete',concat(concat(:old.sid,'
'),:old.classid));
  END;
  /

  create or replace TRIGGER ENROLLMENT_INSERT
AFTER INSERT ON Enrollments
for each row
BEGIN
  update classes set class_size = class_size+1 where classid =:new.classid;
END;
/

create or replace TRIGGER ENROLLMENT_INSERT_LOG
AFTER INSERT ON Enrollments
for each row
BEGIN
  Insert into LOGS (LOGID,WHO,TIME,TABLE_NAME,OPERATION,KEY_VALUE) values
  (logs_seq.nextval,(select user from dual),sysdate,'Enrollments','insert',concat(concat(:new.sid,'
'),:new.classid));
END;

/

create or replace TRIGGER STUDENT_DELETE_LOG
AFTER DELETE ON Students
for each row
BEGIN
  Insert into LOGS (LOGID,WHO,TIME,TABLE_NAME,OPERATION,KEY_VALUE) values
  (logs_seq.nextval,(select user from dual),sysdate,'Students','delete',:old.sid);
delete from enrollments where sid = :old.sid;
END;
/


create or replace trigger STUDENT_INSERT_LOG
after insert on students
for each row
begin
insert into Logs(logid,who,time,table_name,operation,key_value)values
  (logs_seq.NEXTVAL,(Select user from dual), sysdate,'students','insert',:new.sid);
end;

/
```

Sequence:

```
Drop sequence logs_seq;
create sequence logs_seq
MINVALUE 1000
MAXVALUE 9999
START with 1000
INCREMENT by 1;
```

Extra Hepler table: (To store temp results in get_prerequisutes procedure)

```
create table temp (dept_code varchar2(4) not null,course_no number(3) not null);
```

JAVA CODE:

```
//package demo;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.Types;
import oracle.jdbc.OracleCallableStatement;
import oracle.jdbc.OracleTypes;
import oracle.jdbc.pool.OracleDataSource;

public class Driver {
        public static void main(String args[]) throws SQLException
        {
                try
                {
                        int choice;
                        OracleDataSource ds = new oracle.jdbc.pool.OracleDataSource();
                  ds.setURL("jdbc:oracle:thin:@//localhost:1522/oracle");
                  Connection conn = ds.getConnection("SYSTEM", "Saurabh123");
//                String createTable = "create table temp (pre_dept_code varchar2(4),pre_course_no
number(3))";
//                      Statement stmt1 = conn.createStatement();
//                      stmt1.executeQuery(createTable);
                  while(true)
                  {
                         printMenu(0);
                         choice = getChoice(9);
                         switch(choice)
                         {
                                 case 1:
```

```
		{
			printMenu(1);
			choice = getChoice(7);
			printTableData(choice,conn);
			break;
		}
		case 2:
		{
			InsertStudentRecord(conn);
			break;
		}
		case 3:
		{
			getStudentInfo(conn);
			break;
		}
		case 4:
		{
			getPrerequisites(conn);
			break;
		}
		case 5:
		{
			getClassInfo(conn);
			break;
		}
		case 6:
		{
			enrollStudent(conn);
			break;
		}
		case 7:
		{
			dropEnrollment(conn);
			break;
		}
		case 8:
		{
			deleteStudent(conn);
			break;
		}
		case 9:
		{
			System.exit(1);
			break;
		}
	}
```

```
                        }
                    }
                    catch (Exception e) {
                            e.printStackTrace();
                            System.exit(1);
                    }
            }
            public static void deleteStudent(Connection conn) {
                    try
                    {
                            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                            System.out.println("Student ID: ");
                            String sid = br.readLine();
                            CallableStatement stmt = conn.prepareCall("BEGIN
    databaseproject.delete_student(?,?); END;");
                            stmt.setString(1,sid);
                            stmt.registerOutParameter(2, Types.INTEGER);
                            stmt.execute();
                            int status = stmt.getInt(2);
                            switch (status)
                            {
                                    case 1:
                                    {
                                            System.out.println("Student "+sid+" deleted from database.");
                                            break;
                                    }
                                    case 2:
                                    {
                                            System.out.println("The sid is invalid");
                                            break;
                                    }
                            }
                    }
                    catch (Exception e)
                    {
                            e.printStackTrace();
                            System.exit(1);
                    }
            }
            public static void dropEnrollment(Connection conn) {
                    try
                    {
                            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                            System.out.println("Student ID: ");
                            String sid = br.readLine();
                            System.out.println("Enter Class ID: ");
                            String classid = br.readLine();
```

```
                                CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.drop_enrollment(?,?,?,?,?); END;");
                                stmt.setString(1,sid);
                                stmt.setString(2,classid);
                                stmt.registerOutParameter(3, Types.INTEGER);
                                stmt.registerOutParameter(4, Types.INTEGER);
                                stmt.registerOutParameter(5, Types.INTEGER);
                                stmt.execute();
                                int status = stmt.getInt(3);
                                int last_class = stmt.getInt(4);
                                int last_student = stmt.getInt(5);
                                switch(status)
                                {
                                        case 1:
                                        {
                                                System.out.println("Student "+sid+" is Dropped from class "+
classid);
                                                if(last_class==1)
                                                {
                                                        System.out.println("This student is not enrolled in any
classes");
                                                }
                                                if(last_student==1)
                                                {
                                                        System.out.println("The class now has no students.");
                                                }
                                                break;
                                        }
                                        case 2:
                                        {
                                                System.out.println("The drop is not permitted because another
class uses it as a prerequisite.");
                                                break;
                                        }
                                        case 3:
                                        {
                                                System.out.println("The student is not enrolled in the class");
                                                break;
                                        }
                                        case 4:
                                        {
                                                System.out.println("The classid is invalid.");
                                                break;
                                        }
                                        case 5:
                                        {
                                                System.out.println("The sid is invalid");
                                                break;
```

```
					}
				}

			}
			catch (Exception e)
			{
				e.printStackTrace();
				System.exit(1);
			}
		}
		public static void enrollStudent(Connection conn) {
			try
			{
				BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
				System.out.println("Student ID: ");
				String sid = br.readLine();
				System.out.println("Enter Class ID: ");
				String classid = br.readLine();
				CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.enrollment_student(?,?,?,?); END;");
				stmt.setString(1,sid);
				stmt.setString(2,classid);
				stmt.registerOutParameter(3, Types.INTEGER);
				stmt.registerOutParameter(4, Types.INTEGER);
				stmt.execute();
				int status = stmt.getInt(3);
				int count = stmt.getInt(4);
				switch(status)
				{
					case 1:
					{
						System.out.println(sid+" is Enrolled in class "+classid);
						if(count == 2)
						{
							System.out.println("You are overloaded");
						}
						break;
					}
					case 2:
					{
						System.out.println("Prerequisite courses have not been
completed.");
						break;
					}
					case 3:
					{
						System.out.println("Students cannot be enrolled in more than
three classes in the same semester.");
```

```
					break;
				}
				case 4:
				{
					System.out.println("The student is already in the class.");
					break;
				}
				case 5:
				{
					System.out.println("The class is closed.");
					break;
				}
				case 6:
				{
					System.out.println("The classid is invalid");
					break;
				}
				case 7:
				{
					System.out.println("The sid is invalid");
					break;
				}
			}
		}
		catch (Exception e)
		{
			e.printStackTrace();
			System.exit(1);
		}


	}
	public static void getPrerequisites(Connection conn) {
		try
		{
			BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
			System.out.println("Enter Dept Code: ");
			String dept_code = br.readLine();
			System.out.println("Enter Course No: ");
			String course_no = br.readLine();





			CallableStatement cs = conn.prepareCall("begin
databaseproject.get_prerequisites(?,?,?); end;");
			cs.setString(1,dept_code);
```

```
                    cs.setInt(2, Integer.parseInt(course_no));
                    cs.registerOutParameter(3,OracleTypes.CURSOR);
                    cs.execute();

                    ResultSet preRequistes = ((OracleCallableStatement)cs).getCursor(3);
                    if(preRequistes.getFetchSize() == 0)
                    {
                            System.out.println("Course "+dept_code+course_no+" does not have
any Prerequistes");
                    }
                    else
                    {
                            System.out.println("Prerequisite Courses for "+dept_code+course_no+":
");
                            //System.out.format("%-4s\t%-3s","dept_code","course_no");
                            //System.out.println("\n------------------------");
                            while(preRequistes.next())
                            {
                                    //System.out.format("%-4s\t\t%-
3d\n",preRequistes.getString(1),preRequistes.getInt(2));

    System.out.println(preRequistes.getString(1)+preRequistes.getInt(2));
                            }
                    }
                    //preRequistes.close();
                    //cs.close();
                    //stmt.executeQuery(dropTable);
                    //stmt.close();
                    //stmt1.close();
                    String truncateTable = "TRUNCATE table temp";
                    Statement stmt = conn.createStatement();
                    stmt.executeQuery(truncateTable);
            }
            catch (Exception e)
            {
                    e.printStackTrace();
                    System.exit(1);
            }
    }
    public static void printTableData(int tableChoice, Connection conn)
    {
            switch(tableChoice)
            {
                    case 1:
                    {
                            try
                            {
```

```
                                        CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_students(?); END;");
                                        stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
                                    stmt.execute();
                                    ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);

                                    while (rs.next())
                                    {
                                         System.out.format("%-4s --> %-15s --> %-15s --> %-10s --> %.2f
--> %-20s\n",
                                                      rs.getString(1),
rs.getString(2),rs.getString(3),rs.getString(4),rs.getDouble(5),rs.getString(6));
                                    }
                                    rs.close();
                                }
                                catch (SQLException e)
                                {
                                        e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }
                        case 2:
                        {
                                try
                                {
                                        CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_courses(?); END;");
                                        stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
                                    stmt.execute();
                                    ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
                                    while (rs.next())
                                    {
                                         System.out.format("%-4s --> %-3d --> %-20s\n",
                                                      rs.getString(1), rs.getInt(2),rs.getString(3));
                                    }
                                    rs.close();
                                    stmt.close();
                                }
                                catch (SQLException e)
                                {
                                        e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }
```

```
				case 3:
				{
					try
					{
						CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_classes(?); END;");
						stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
				        stmt.execute();
				        ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
				        while (rs.next())
				        {
				            System.out.format("%-5s --> %-4s --> %-3d --> %-2d --> %-4d --
> %-6s --> %-3d --> %-3d\n",
				            			rs.getString(1),
rs.getString(2),rs.getInt(3),rs.getInt(4),rs.getInt(5),rs.getString(6),rs.getInt(7),rs.getInt(8));
				        }
				        rs.close();
					}
					catch (SQLException e)
					{
						e.printStackTrace();
						System.exit(1);
					}
					break;
				}
				case 4:
				{
					try
					{
						CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_prerequisites(?); END;");
						stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
				        stmt.execute();
				        ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
				        while (rs.next())
				        {
				            System.out.format("%-4s --> %-3d --> %-4s --> %-3d\n",
				            			rs.getString(1), rs.getInt(2),
rs.getString(3),rs.getInt(4));
				        }
				        rs.close();
					}
					catch (SQLException e)
					{
						e.printStackTrace();
						System.exit(1);
```

```
                    }
                    break;
                }
                case 5:
                {
                    try
                    {
                        CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_enrollments(?); END;");
                        stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
                      stmt.execute();
                      ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
                      while (rs.next())
                      {
                         System.out.format("%-5s --> %-4s --> %-1s\n",
                                     rs.getString(1), rs.getString(2),rs.getString(3));
                      }
                      rs.close();
                    }
                    catch (SQLException e)
                    {
                        e.printStackTrace();
                        System.exit(1);
                    }
                    break;
                }
                case 6:
                {
                    try
                    {
                        CallableStatement stmt = conn.prepareCall("BEGIN
databaseproject.show_logs(?); END;");
                        stmt.registerOutParameter(1, OracleTypes.CURSOR); //REF
CURSOR
                      stmt.execute();
                      ResultSet rs = ((OracleCallableStatement)stmt).getCursor(1);
                      while (rs.next())
                      {
                         System.out.format("%-4d --> %-10s --> %-10s --> %-15s --> %-
6s --> %-14s\n",
                                     rs.getInt(1), rs.getString(2),
rs.getString(3),rs.getString(4),rs.getString(5),rs.getString(6));
                      }
                      rs.close();
                    }
                    catch (SQLException e)
                    {
```

```
                                        e.printStackTrace();
                                        System.exit(1);
                                }
                                break;
                        }
                        case 7:
                        {
                                printMenu(0);
                                break;
                        }
                }
        }
        public static int getChoice(int screen)
        {
                int choice = 0;
                try
                {
                        BufferedReader input_reader = new BufferedReader(new
InputStreamReader(System.in));
                        do
                        {
                                System.out.println("Enter Choice");
                                choice = Integer.parseInt(input_reader.readLine());
                        }while(choice < 1 || choice > screen);
                }
                catch (Exception e) {
                        System.out.println("getChoice Exeption");
                        System.exit(1);
                }
                return choice;
        }
        public static void printMenu(int screen)
        {
                switch(screen)
                {
                        case 0:
                        {
                                System.out.println();
                                System.out.println("*****Main Menu*****");
                                System.out.println("1.View Table Data:");//#2
                                System.out.println("2.Add Student:");//#3
                                System.out.println("3.View Studnet info:");//#4
                                System.out.println("4.View Course Prerequisites:");//#5
                                System.out.println("5.View Class Info");//#6
                                System.out.println("6.Enroll a Student in Class:");//#7
                                System.out.println("7.Drop a Student from Class:");//#8
                                System.out.println("8.Delete a Student:");//#9
                                System.out.println("9.Exit");
```

```
					break;
				}
				case 1:
				{
					System.out.println();
					System.out.println("***Select Table***");
					System.out.println("1.Students\n"
							+ "2.Courses\n"
							+ "3.Classes\n"
							+ "4.Prerequisites\n"
							+ "5.Enrollments\n"
							+ "6.Logs\n"
							+ "7.Back to Main Menu");
					break;
				}
			}
		}
		public static void getClassInfo(Connection conn)
		{
			try
			{
				BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
				System.out.println("Enter Classid: ");
				String classid = br.readLine();
				CallableStatement cs = conn.prepareCall("begin
databaseproject.get_class_info(:1,:2,:3); end;");
				cs.setString(1,classid);
				cs.registerOutParameter(2,OracleTypes.CURSOR);
				cs.registerOutParameter(3,OracleTypes.CURSOR);
				cs.execute();
				ResultSet classInfo = ((OracleCallableStatement)cs).getCursor(2);
				ResultSet students = ((OracleCallableStatement)cs).getCursor(3);
				if(classInfo.getFetchSize() == 0)
				{
					System.out.println("The cid is invalid");
				}
				else
				{
					classInfo.next();
					System.out.println("Class ID: "+classInfo.getString(1));
					System.out.println("Title: "+classInfo.getString(2));
					System.out.println("Semester: "+classInfo.getString(3));
					System.out.println("Year: "+classInfo.getInt(4));
					if(students.getFetchSize() == 0)
					{
						System.out.println("No student is enrolled in the class.");
					}
					else
```

```
                        {
                                System.out.println("Enrolled Students: ");
                                System.out.format("%-4s --> %-15s\n------------------------\n",
                                                "SID","Lastname");
                                while(students.next())
                                {
                                        System.out.format("%-4s --> %-15s\n",

        students.getString(1),students.getString(2));
                                }
                        }
                }
        }
        catch (Exception e)
        {
                System.err.println("Exception in get Student Info.");
                e.printStackTrace();
                System.exit(1);
        }
    }
    public static void getStudentInfo(Connection conn)
    {
        try
        {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Enter Sid: ");
                String sid = br.readLine();
                CallableStatement cs = conn.prepareCall("begin
databaseproject.get_student_info(:1,:2,:3); end;");
                cs.setString(1,sid);
                cs.registerOutParameter(2,OracleTypes.CURSOR);
                cs.registerOutParameter(3,OracleTypes.CURSOR);
                cs.execute();
                ResultSet student = ((OracleCallableStatement)cs).getCursor(2);
                ResultSet classes = ((OracleCallableStatement)cs).getCursor(3);
                if(student.getFetchSize() == 0)
                {
                        System.out.println("The SID is invalid");
                }
                else
                {
                        student.next();
                        System.out.println("SID: "+student.getString(1));
                        System.out.println("Lastname: "+student.getString(2));
                        System.out.println("Status: "+student.getString(3));
                        if(classes.getFetchSize() == 0)
                        {
                                System.out.println("The student has not taken any course");
```

```
					}
					else
					{
						System.out.println("Enrolled Classes: ");
						while(classes.next())
						{
							System.out.format("%-5s --> %-7s --> %-20s --> %-4d -->
%-6s\n",

	classes.getString(1),classes.getString(2),classes.getString(3),classes.getInt(4),classes.getString(5))
;
						}
					}
				}
			}
			catch (Exception e)
			{
				System.err.println("Exception in get Student Info.");
				e.printStackTrace();
				System.exit(1);
			}
		}
		public static void InsertStudentRecord(Connection conn)
		{
			try
			{
				BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
				System.out.println("Enter Sid: ");
				String sid = br.readLine();
				System.out.println("Enter Firstname: ");
				String firstname = br.readLine();
				System.out.println("Enter Lastname: ");
				String lastname = br.readLine();
				System.out.println("Enter Status: ");
				String status = br.readLine();
				System.out.println("Enter GPA: ");
				Double gpa = Double.parseDouble(br.readLine());
				System.out.println("Enter Email ID: ");
				String email = br.readLine();
				CallableStatement cs = conn.prepareCall("begin
databaseproject.insert_student(:1,:2,:3,:4,:5,:6,:7,:8); end;");
				cs.setString(1,sid);
				cs.setString(2,firstname);
				cs.setString(3,lastname);
				cs.setString(4,status);
				cs.setDouble(5, gpa);
				cs.setString(6,email);
				cs.registerOutParameter(7, Types.INTEGER);
```

```
                        cs.registerOutParameter(8, Types.INTEGER);
                        cs.executeQuery();

                        int sid_validity = cs.getInt(7);
                        int email_validity = cs.getInt(8);

                        if(sid_validity==0)
                        {
                                if(email_validity==0)
                                {
                                        System.out.println("Student Record Inserted Successfully.");
                                }
                                else
                                {
                                        System.out.println("SID already Exists.");
                                }
                        }
                        else
                        {
                                System.out.println("Email ID already Exists.");
                        }

                }
                catch (Exception e)
                {
                        System.err.println("Exception in Insert Student.");
                        e.printStackTrace();
                        System.exit(1);
                }
        }
}
```

Conclusion:
Project Design for the stored procedures were created as per the requirement document. Team designed the menu interface accordingly to meet those requirements.