**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,prerequisites\_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.