Feature 1 :

Register an account with the system. The user needs to provide name, phone#, email, password, and type of user (faculty, staff, or student). The procedure should check whether the email already exists in user table. If so, please print a message saying the account exists. Otherwise create an account with input values and return a new user ID.

```
Create sequence new_user_id
 START WITH     11
 INCREMENT BY   1
 NOCACHE
 NOCYCLE;


Create or replace procedure Register_User(name in varchar,email_id in varchar, phone_num in integer, user_password in varchar, user_type in varchar) IS
Count_num integer;
Begin
Select count(*) into count_num from users where email=email_id;
If(count_num<=0)then
        Insert into users
values(new_user_id.nextval,name,email_id,phone_num,user_password,user_type);
        Dbms_output.put_line('New User Id is : s' || new_user_id.currval);
Else
Dbms_output.put_line('Account with same email-id already exists');
end if;
End;
set serveroutput on;

exec Register_User('Varun
Ramkrishna','varun.arkay@umbc.edu',4436538969,'abcd1234','student');\
```

Feature 2:

Allow a user to login by providing user id and password. Please check whether user id exists and password matches. If not, please print a message to indicate the error. Otherwise print a message to indicate user has logged on. The procedure should return a value 1 for success login and 0 for unsuccessful log in.

```
Create or replace
        PROCEDURE UserLogin (a in integer,b in varchar)AS
        counter integer;
BEGIN
        select count(*) into counter from users where user_id=a and password = b;
```

```
if counter = 0 then
   dbms_output.put_line('0 - User Login Failed');
elsif counter =1 then
   dbms_output.put_line('1 - User Login Successfull');
END if;
END;
```

Feature 3 :

Allow a user to read messages providing user id.

```
Create or replace PROCEDURE ReadMessage(userid in number) as
m_message varchar(200);
usercount integer;
Cursor C1 is select message from message where user_id=userid;

BEGIN
select count(*) into usercount from message where user_id= userid;
if (usercount >0) then
 Open C1;
 Loop
          fetch C1 into m_message;
          exit when C1%notfound;
          dbms_output.put_line(m_message);
 End loop;
 Close C1;
else
 dbms_output.put_line('UserId is Invalid');
end if;
END;
```

Feature 4 :

Create an event with title, description, start date and time, end date and time, location, an optional url, and organizer id. The procedure needs to check whether any other event at the same location has overlap duration with the new event. If so, print a message saying the event conflicts with an existing event. Otherwise, insert the event with input values into event table and print a message with new event ID.

```
create or replace procedure CheckClash(eid number, etitle varchar, edesc varchar, eurl varchar,
estart timestamp, eend timestamp, elocid number, eorgid number, estatus number) IS
count_clash number(1) := 0;
begin
select count(*) into count_clash
from events e1, events e2
where e1.location_id=e2.location_id and e1.start_datetime < eend and e1.end_datetime>estart
and e1.location_id = elocid and rownum=1;


if count_clash > 0 then
 dbms_output.put_line('Date range clashes with existing dates');
else
 insert into events values(eid,etitle,edesc,eurl,estart,eend,elocid,eorgid,estatus);
 dbms_output.put_line('The event has been created and the event id is ' || eid);
end if;
End;

set serveroutput on;
exec CheckClash(6,'Lecture','Lecture on LINUX','umbc.edu/IS651',timestamp '2015-9-1
10:00:00',timestamp '2015-9-1 10:30:00',2,5,1);
```

Feature 5 :

List people registered for an event by providing event id. The procedure prints out name of participants, their email addresses, and whether the participant is faculty, staff, or student.

```
Create or replace
PROCEDURE ConfirmedList (eid number) IS
Cursor c1 is select u.user_name, u.email, u.type, e.event_description
from events e, users u, registration r
where u.user_id=r.user_id and e.event_id=r.event_id and e.event_id=eid;
uname varchar(50);
uemail varchar(100);
utype varchar(50);
ename varchar(50);
BEGIN
Open c1;
Loop
fetch c1 into uname, uemail, utype, ename;
exit when c1%notfound;
```

```
dbms_output.put_line('Participant: ' || uname || ' with email id ' || uemail || ' is a ' || utype || ' and is registered for ' || ename);
End loop;
close c1;
END;

exec ConfirmedList(1);
```

Feature 6 :

List people on wait list of an event by providing the event id. The procedure prints out names of users on the wait list, their email addresses, and whether they are faculty, staff, or students

```
create or replace
procedure wait_list(eventid number)IS
cursor c1 is select user_name, email, type
from waitlist, users
where waitlist.user_id=users.user_id and event_id=eventid;
username varchar(100);
email_id varchar(50);
user_type varchar(50);
Begin
open c1;
Loop
fetch c1 into username,email_id,user_type;
exit when c1%notfound;
dbms_output.put_line('username is '||username||' email address '|| email_id || ' user type ' || user_type);
End Loop;
```

Feature 7 :

Return average rating of an event and total number of participants and number of people on wait list

```
Create or replace
        PROCEDURE EventRating (a in integer)AS
        event_flag integer;
        avg_rating integer;
        confirmed_count integer;
        waitlist_count integer;
BEGIN

        select count(*) into event_flag from events where event_id= a;
```

```
        if( event_flag = 0) then
         dbms_output.put_line('No event exists with that event ID');
        else

         select avg(rating) into avg_rating from review where event_id = a;
         If (avg_rating is null) Then
                dbms_output.put_line('No reviews available for this event');
         else
                dbms_output.put_line('Average rating :' || avg_rating);
         end if;

         select count(*) into confirmed_count from registration where event_id= a and
registration_status='C';
         dbms_output.put_line('Total number of participants :' || confirmed_count);

         select count(*) into waitlist_count from registration where event_id= a and
registration_status='W';
         dbms_output.put_line('Number of people on wait list :' || waitlist_count);

         end if;

END;
```

Feature 8 :

```
Create or replace procedure EventCancellation(eventid in integer)IS
cursor C1 is select user_id from registration where event_id = eventid;

message varchar(300);
event_title varchar(20);
userid integer;

begin
select TITLE into event_title from events where event_id= eventid and event_status=1;

        update events
        set event_status = 0
        where event_id = eventid;
   Open C1;
   Loop
        fetch C1 into userid;
        exit when C1%notfound;
```

```
        message := 'The event -' || event_title || ' has been cancelled';
        SendMessageToUsers(userid ,message);
    End loop;
    Close C1;

Exception
        when no_data_found then
        dbms_output.put_line('No active event exists with that event ID');
End;


exec EventCancellation(1);
```

Feature 9 :

Update the start and end date and time of an event by providing event ID and new start/end date and time. Update the event table if the event exists. If the event does not exist, print out a message saying wrong event ID. Please generate a message for each user who has registered or on wait list with the new date and time of the event.

```
Create or replace PROCEDURE UpdateEvent (a in integer ,startdatetime in
timestamp,enddatetime in timestamp ) IS
        Cursor C1 is Select user_id from registration where event_id= a and
REGISTRATION_STATUS ='C';
        event_title varchar(20);
        return_msg integer;
        userid integer;
        message varchar(300);

BEGIN
select TITLE into event_title from events where event_id= a;

  update events set START_DATETIME= startdatetime ,END_DATETIME= startdatetime where
event_id = a;

  Open C1;
  Loop
        fetch C1 into userid;
        exit when c1%notfound;
```

```
        message := 'The event -' || event_title || ' has been rescheduled. The event will now start
at ' || startdatetime || ' and end at ' || enddatetime;
        SendMessageToUsers(userid ,message);
  End loop;
  Close C1;

Exception
        when no_data_found then
        dbms_output.put_line('No event exists with that event ID');

END;



Create or replace PROCEDURE SendMessage2Users (userid in number, usermessage in
varchar)
IS
username varchar(20);
BEGIN
        select user_name into username from users where user_id = userid;
        dbms_output.put_line(username || ' :  ' || usermessage);
        insert into  message values(message_id.nextval,userid,usermessage,systimestamp);
End;

exec UpdateEvent(3,timestamp '2005-02-05 17:00:53.00',timestamp '2005-02-05 17:00:53.00')
```

Feature 10:

Search for events with a certain keyword in the title (you can use like), return start and end date and time, full event title, location, url, organizer name and email, and whether the event is full.

```
Create or Replace procedure SearchEvent(eventkeyword in varchar)
as
event_title varchar(50);
Cursor C1 is Select title from events where title like
'%'||eventkeyword||'%';//checks for title in events table for keyword
entered
Begin
Open C1;
Loop
Fetch C1 into event_title;
exit when C1%notfound;
eventlist(event_title);
```

```
End Loop;
Close C1;
exception when no_data_found then
Dbms_output.put_line('No Event found');
End;


Create or Replace Procedure eventlist(event_title in varchar)
as
EventId number;
EventDescription varchar(50);
EventUrl varchar(50);
Startdatetime timestamp;
Enddatetime timestamp;
OrganizerID number;
LocationId number;
Registration_count number;
Location_Capacity number;
Location_Name varchar(50);
Capacity_status Varchar(10);
Organizer_Name varchar(50);
Organizer_Email varchar(50);
Cursor C1 is Select
event_id,event_description,url,start_datetime,event_datetime,organizer
_id,location_id from events where title=event_title;
Begin
Open C1;
Loop
Fetch C1 into
EventId,EventDescription,EventUrl,Startdatetime,Enddatetime,OrganizerI
D,LocationId;
Exit when C1%notfound;
Select Count(user_id) into Registration_count from Registration where
event_id=EventId;
Select location_description,capacity into
Location_Name,Location_Capacity from Location where
location_id=LocationId;
Select user_name,email into Organizer_Name,Organizer_Email from users
where user_id=OrganizerID;
if(Registration_count=Location_Capacity) then//checks if capacity is
full
Capacity_status:='Full';
else
```

```
Capacity_status:='Not Full';
End if;
Dbms_output.put_line('EventID:'||EventId||' Event
Descrition:'||EventDescription||' Location:'||Location_Name||'
URL:'||EventUrl||' Organizer_Name:'||Organizer_Name||'
Email:'||Organizer_Email||' Start Date:'||Startdatetime||' End
Date:'||Enddatetime);
End Loop;
Close C1;
End;
```

Feature 11:

Feature 12:

```
Create or replace procedure cancelEventRegistration(eventid in number, userid in number)
as
message varchar(300);
event_title varchar(20);
UID number;
checkregistration number;
Begin

select REGISTRATION_ID into checkregistration where user_id= userid and event_id=eventid;

  delete from Registration where user_id= userid and event_id=eventid;

  Dbms_Output.Put_Line('Registration has been cancelled successfully');
  UID:=get_wluser(eventid);
   if(UID!=-1) then
     update Registration set registration_status='C' where user_id=userid and event_id=eventid;
     Delete from Waitlist where user_id=UID and event_id=eventid;
           select title into event_title from events where event_id=eventid;
     message := 'You have successfully enrolled for the event -' || event_title;
     SendMessageToUsers(userid ,message);
     Update Waitlist set waitlist_no=waitlist_no+1 where event_id= eventid;
   else
      Dbms_Output.Put_Line('User is not registered for this event');
End if;
Exception
 when no_data_found then
 Dbms_Output.Put_Line('Event does not exist');
```

end;

Feature : 13

Enter a review for an event. The user provides a numerical rating (1 to 5) as well as some comment. To prevent abuse, the feature needs to check that the user has registered for the event and only registered users can enter reviews.

```
Create or replace procedure EventReview(userid in integer,eventid in integer,rating_review in integer,comments in varchar)IS
registration_number integer;
Begin
        select count(*) into registration_number from registration where user_id=userid  and event_id=eventid ;
        if( (registration_number > 0) and (rating_review > 0 or rating_review < 6) )then
                insert into review(user_id,event_id,rating,review_comments)Values(userid,eventid,rating_review,comments);
                dbms_output.put_line('Thank you for your review');
        else
                dbms_output.put_line('Error : Either rating is not between 0 to 5 or the user has not registered for the event');
        end if;
End;
```

Feature 15

Print out the top K events with the most participants (only counting those registered), the top K locations with most number of events, and top K events with the highest average ratings. K is an input parameter.

```
create or replace procedure Top_k(k in number) as
eid number;
P_Count number;
E_Count number;
H_rating number;
Lid number;
location_name varchar(50);
event_name varchar(50);
cursor c1 is select event_id,count from (select event_id,count(user_id)as count from registration group by event_id order by count desc) where rownum<=k;
Cursor C2 is select location_id, count from (select Location_id,Count(Location_id) as count from events group by location_id order by count desc)where rownum<=k;
cursor c3 is select event_id, count from (select event_id,avg(rating)as count from review group by event_id order by count desc)where rownum<=k;

Begin
Dbms_Output.Put_Line('Top '||k||' events with most participants: ');
```

```
Open C1;
Loop
fetch C1 into eid,P_Count;
Exit when C1%notfound;
event_name:=get_eventname(eid);
Dbms_Output.Put_Line('Event ID: '||eid||' Event Name: '||event_name||' Participant
Count: '||P_Count);
end Loop;
Close C1;
Dbms_Output.Put_Line('Top '||k||' locations with most no of events: ');
Open C2;
Loop
fetch C2 into Lid,E_Count;
Exit when C2%notfound;
location_name:=get_locationname(Lid);
Dbms_Output.Put_Line('Location ID: '||Lid||' Location Name: '||location_name||' Event
Count: '||E_Count);
end Loop;
Close C2;
Dbms_Output.Put_Line('Top '||k||' events with highest average ratings: ');
Open C3;
Loop
fetch C3 into eid,H_rating;
Exit when C3%notfound;
event_name:=get_eventname(eid);
Dbms_Output.Put_Line('Event ID: '||eid||' Event Name: '||event_name||'  Average
Rating: '||H_rating);
end Loop;
Close C3;
End;

create or replace function get_eventname(eid in number) return varchar2
as
event_name varchar(50);
begin
select title into event_name from events where event_id=eid;
return event_name;
exception when no_data_found then
return 'No Event Found';
end;

create or replace function get_locationname(lid in number) return varchar2
as
location_name varchar(50);
begin
select location_description into location_name from location where location_id=lid;
return location_name;
exception when no_data_found then
return 'No Location Found';
end;
```

```
select * from users;
select * from message;
select * from events;
select * from location;
select * from registration;
select * from waitlist;
select * from review;


drop table review;
drop table waitlist;
drop table registration;
drop table events;
drop table location;
drop table message;
drop table users;

create table users
(
user_id int,
user_name varchar(20),
email varchar(20),
phone_number int,
password varchar(100),
type varchar(20),
primary key(user_id)
);

create table message
(
message_id int,
user_id int,
message varchar(300),
message_time timestamp,
primary key(message_id),
foreign key(user_id) references users(user_id)
);

create table location
(
```

```
location_id int,
location_description varchar(100),
capacity int,
primary key (location_id)
);

create table events
(
event_id int,
title varchar(20),
event_description varchar(100),
url varchar(100),
start_datetime timestamp,
end_datetime timestamp,
location_id int,
organizer_id int,
event_status varchar(20),
primary key(event_id),
foreign key(organizer_id) references users(user_id),
foreign key(location_id) references location(location_id)
);

create table registration
(
registration_id int,
user_id int,
event_id int,
registration_status varchar(20),
primary key(registration_id),
foreign key(user_id) references users(user_id),
foreign key(event_id) references events(event_id)
);

create table waitlist (
waitlist_id int,
user_id int,
event_id int,
registration_id int,
waitlist_no int,
foreign key(user_id) references users(user_id),
foreign key(event_id) references events(event_id),
foreign key(registration_id) references registration(registration_id));
```

```
create table review(
user_id int,
event_id int,
rating int,
review_comments varchar(100),
foreign key(user_id) references users(user_id),
foreign key(event_id) references events(event_id));



insert into users values(1,'Varun
Ramkrishna','jl28777@umbc.edu',4436538969,'abcd1234','student');
insert into users values(2,'Kevin Pelkey','pelkey@umbc.edu',4436538968,'efgh5678','faculty');
insert into users values(3,'Carlton Crabtree','cac1@umbc.edu',4436538967,'ijkl1234','faculty');
insert into users values(4,'Sumeet Dabhi','sdabhi@umbc.edu',4436538966,'lmno5678','staff');
insert into users values(5,'Charles Hogan','chogan@umbc.edu',4436538965,'pqrs1234','staff');
insert into users values(6,'Joel Jacob','joel@umbc.edu',4436538000,'joeljoel','student');
insert into users values(7,'Steve Smith','ss@umbc.edu',4436538001,'stevesteve','student');
insert into users values(8,'John Legend','john@umbc.edu',4436538002,'johnjohn','student');
insert into users values(9,'Ankita','ankita@umbc.edu',4436538003,'ankita00','student');
insert into users values(10,'Anuja','anuja@umbc.edu',4436538004,'anujaanuja','staff');


insert into location values(1,'Commons',5);
insert into location values(2,'Information Technology and Engineering',5);
insert into location values(3,'Sherman Hall',5);
insert into location values(4,'Physics Buidling',3);
insert into location values(5,'Public Policy Buidling',2);


insert into events values(1,'Fundamentals of IS','Lecture on Fundamentals of
IS','my.umbc.edu/IS601', timestamp '2015-8-20 16:30:00', timestamp '2015-8-20
19:00:00',3,3,1);
insert into events values(2,'LanLinux Management','Lecture on LINUX','my.umbc.edu/IS651U',
timestamp '2015-8-21 16:30:00', timestamp '2015-8-21 19:00:00',2,2,1);
insert into events values(3,'RLC Tutoring','JAVA tutoring','my.umbc.edu/JAVAtutor', timestamp
'2015-9-1 14:00:00', timestamp '2015-9-1 16:00:00',5,1,1);
insert into events values(4,'GSA meeting','Elect new president','my.umbc.edu/GSA', timestamp
'2015-9-1 14:00:00', timestamp '2015-9-1 16:00:00',4,4,1);
insert into events values(5,'CommonsFoodFestival','Free food event','my.umbc.edu/SWF',
timestamp '2015-9-1 9:00:00', timestamp '2015-9-1 10:00:00',5,5,1);
```

```
insert into registration values(1,1,3,'C');
insert into registration values(2,4,3,'C');
insert into registration values(3,5,3,'W');
insert into registration values(4,7,3,'W');
insert into registration values(5,3,1,'C');
insert into registration values(6,2,1,'C');
insert into registration values(7,4,1,'C');
insert into registration values(8,5,1,'C');
insert into registration values(9,6,1,'C');
insert into registration values(10,7,1,'W');
insert into registration values(11,8,1,'W');
insert into registration values(12,9,1,'W');
insert into registration values(13,10,1,'W');
```

```
Insert into waitlist values(1,5,3,3,1);
Insert into waitlist values(2,7,3,4,2);
Insert into waitlist values(3,7,1,10,1);
Insert into waitlist values(4,8,1,11,2);
Insert into waitlist values(5,9,1,12,3);
Insert into waitlist values(6,10,1,13,4);
```