CS3710: DATABASE SYSTEM LAB

PROJECT REPORT

COURSE MANAGEMENT SYSTEM

Members

Prabal Vashisht (111501021)

Rajat Sharma(111501024)

28.02.2018

PHASE 1

Specifications/Requirements

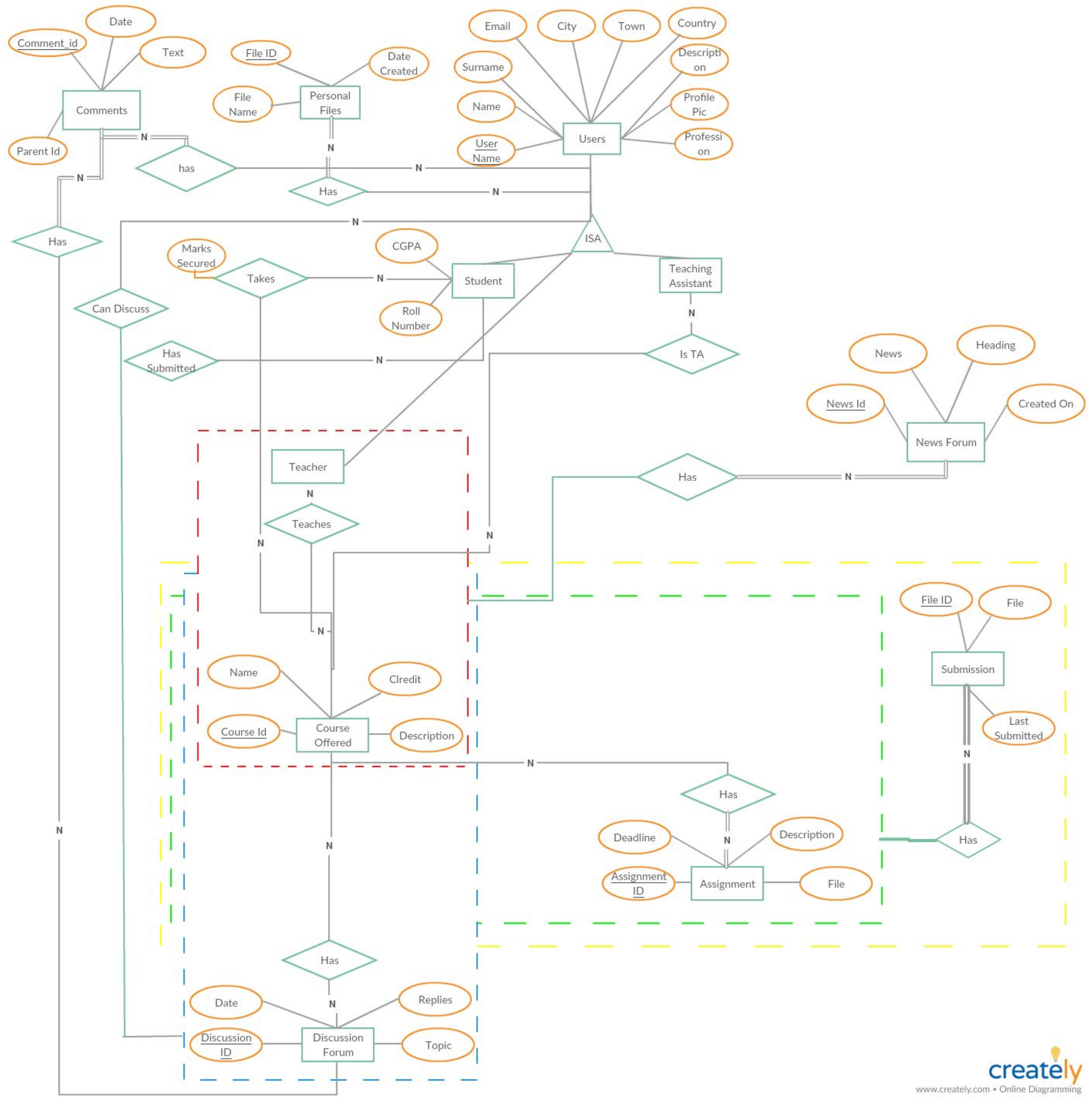
Specifications of the software are :

1. Each user can register through the GUI. Users can edit their profile once they are logged in. User's attributes includes:
    a. Unique username
    b. Name
    c. Surname
    d. Password
    e. Department he/she works
    f. Email address
    g. City
    h. Town
    i. Country
    j. Description
    k. Profile picture
2. Users can enter their profession in the GUI. Users are of three types:
    a. Students: Student's grades, comments on the forums, personal files, and submissions should be stored in the database.
    b. Teaching assistant: personal files, files uploaded, comments on forums should be stored in the database.
    c. Course instructors: personal files, files uploaded, comments on forums, admin rights to a user should be stored in the database.
3. Course instructors can create a course, add description of the course, register/deregister a student/TA , can create a discussion/news forum and add comments to it, upload assignments/lecture materials, grade a student, give limited access to TAs, create submission links and add comments to it.
4. Teaching Assistants(TA) of a particular course can add comments to a discussion/news forum, upload assignments/lecture materials(if permission is given by course instructor), grade a student(if permission is given by course instructor), create submission links and add comments to it(if permission is given by course instructor).
5. A student is enrolled in a set of courses. In a particular course he/she can submit assignments, get email notification everytime there is a an activity(relevant to the student) in a course, can view his grades, comment on the assignments that he/she

has submitted, can have a set of private files, gets updated when the deadline of submission is near, can create a discussion/news forum and add comments to it.

PHASE 2

The ERD of the database is given below.



PHASE 3

In this phase of the project we had to implement the following tasks (mentioned along with the task number) :
0. Make the database in 3NF form.

1. All tables and view are implemented.

2. All roles are created all privileged are created. Also decide default privilege a user will get.

3. At-least 10 triggers should be implemented to implement required integrity.

4. At-least 10 more function or procedure should be integrates which will make the database more consistent.

5. One procedure should take a periodic backup of your database. (COULD NOT BE IMPLEMENTED).

6. Insert some toy data which will help you to validate your trigger or procedure.

7. Submit description of functions, procedures and triggers in a report and also .sql file so that we can import it in our RDBMS and check.

The above mentioned tasks along with the results are mentioned hereby in the document.

TASK 0

Most of the tables in the database '*course_management_system*' were in 3NF except the tables '*teacher*', '*student*' and '*ta*' in which the attribute '*country*' depended on attribute '*city*' and '*town*' (and these two attributes are not candidate key). To convert the tables into 3NF we decomposed the tables into 4 tables: '*teacher*', '*student*' '*ta*' (without attribute country) and the table '*user_country*' with the attributes '*country*' , '*city*' and '*town*'.

TASK 1

All tables and views have been implemented.

TASK 2

There are **three** roles in the database:

1. Teacher
2. Teaching Assistant (TA)
3. Student

TASK 3

In total **11 triggers** have been created. Their functionality along with the code is mentioned below:

1.

Code : create trigger student_name before insert on student for each row begin insert into user_name values (NEW.username,NEW.username); end;

Description: The above trigger is used to maintain integrity between the tables *'student'* and *'user_name'.* We want this integrity because every single username present in *'student'* table should be present in *'user_name'* table and the table *'student'* refers to *'user_name'* table. The reverse needn't be true.

2.

Code : create trigger ta_name before insert on ta for each row begin insert into user_name values (NEW.username,NEW.username); end;

Description: The above trigger is used to maintain integrity between the tables *'ta'* and *'user_name'*. We want this integrity because every single username present in *'ta'* table should be present in *'user_name'* table and the table *'ta'* refers to *'user_name'* table. The reverse needn't be true.

3.

Code : create trigger teacher_name before insert on teacher for each row begin insert into user_name values (NEW.username,NEW.username); end;

Description: The above trigger is used to maintain integrity between the tables *‘teacher’* and *‘user_name’*. We want this integrity because every single username present in *‘teacher’* table should be present in *‘user_name’* table and the table *‘teacher’* refers to *‘user_name’* table. The reverse needn’t be true.

4.

Code : create trigger insert_city_st before insert on student for each row begin if not exists(select * from user_country where city=NEW.city and town=NEW.town) then insert into user_country values(NEW.city, NEW.town,NULL); end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘student’* and *‘user_country’*. We want this integrity because we want every single *‘city’* and *‘town’* pair to be present in *‘user_country’* table and if the city/town already exists in the table *’user_country*’ no need to insert it in the latter table. The table *‘student’* also refers to *‘user_country’*.

5.

Code : create trigger insert_city_ta before insert on ta for each row begin if not exists(select * from user_country where city=NEW.city and town=NEW.town) then insert into user_country values(NEW.city, NEW.town,NULL); end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘ta’* and *‘user_country’*. We want this integrity because we want every single *‘city’* and *‘town’* pair to be present in *‘user_country’* table and if the city/town already exists in the table *’user_country*’ no need to insert it in the latter table. The table *‘ta’* also refers to *‘user_country’*.

6.

Code : create trigger insert_city_te before insert on teacher for each row begin if not exists(select * from user_country where city=NEW.city and town=NEW.town) then insert into user_country values(NEW.city, NEW.town,NULL); end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘teacher’* and *‘user_country’*. We want this integrity because we want every single *‘city’* and *‘town’* pair to be present in *‘user_country’* table and if the city/town already exists in the table *’user_country*’ no need to insert it in the latter table. The table *‘teacher’* also refers to *‘user_country’*.

7.

Code : create trigger delete_city_st after delete on student for each row begin if not exists(select * from student where city=OLD.city and town=OLD.town) and not exists(select * from ta where city=OLD.city and town=OLD.town) and not exists(select * from teacher where city=OLD.city and town=OLD.town) then delete from user_country where city =OLD.city and town=OLD.town; end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘student’*, *‘teacher’* , *‘ta’* and *‘user_country’*. Every time a row is deleted from *‘student’*, if that city/town doesn’t exist in the other two tables(teacher and ta), it will be deleted from the table *‘user_country’*.

8.

Code :   create trigger delete_city_ta after delete on ta for each row begin if not exists(select * from student where city=OLD.city and town=OLD.town) and not exists(select * from ta where city=OLD.city and town=OLD.town) and not exists(select * from teacher where city=OLD.city and town=OLD.town) then delete from user_country where city =OLD.city and town=OLD.town; end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘student’*, *‘teacher’* , *‘ta’* and *‘user_country’*. Every time a row is deleted from *‘ta’*, if that city/town doesn’t exist in the other two tables(teacher and student), it will be deleted from the table *‘user_country’*.

9.

Code: create trigger delete_city_te after delete on teacher for each row begin if not exists(select * from student where city=OLD.city and town=OLD.town) and not exists(select * from ta where city=OLD.city and town=OLD.town) and not exists(select * from teacher where city=OLD.city and town=OLD.town) then delete from user_country where city =OLD.city and town=OLD.town; end if; end;

Description: The above trigger is used to maintain integrity between the tables *‘student’*, *‘teacher’* , *‘ta’* and *‘user_country’*. Every time a row is deleted from *‘teacher’*, if that city/town doesn’t exist in the other two tables(student and ta), it will be deleted from the table *'user_country'*.

10.

Code: create trigger check_parent_discid before insert on d_comments for each row begin if NEW.parent_id IS NOT NULL then begin declare temp int; select d_id into temp from d_comments where c_id = NEW.parent_id; if temp<> NEW.d_id then SIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=30001, MESSAGE_TEXT='Discussion id not of parent'; end if; end; end if; end;

Description: The above trigger is used to maintain integrity in the table *'d_comments'*. This integrity constraint arises from the fact that a reply's discussion id should be same as the it's parent's discussion id.

11.

Code: create trigger del_user before delete on user_name for each row begin delete from student where username=OLD.name; delete from teacher where username=OLD.name; delete from ta where username=OLD.name; end;

Description: This trigger is used to explicitly delete a username from *'student'*, *'teacher'* or *'ta'* whenever an entry is deleted from *'user_name'*.

TASK 4

In total **10 functions or procedures** have been created. Their functionality along with the code is mentioned below:

1.

Code: create function num_stu_enrolled (courseid varchar(50)) returns int begin return (select count(*) from takes where course_id=courseid); end;

Description: This function is used to find the number of students enrolled in a course.

2.

Code: create function num_ta_enrolled (courseid varchar(50)) returns int begin return (select count(*) from ista where course_id=courseid); end;

Description: This function is used to find the number of teaching assistants enrolled in a course.

3.

Code: create function num_te_enrolled (courseid varchar(50)) returns int begin return (select count(*) from teaches where course_id=courseid); end;

Description: This function is used to find the number of teachers teaching a course.

4.

Code: create function num_te_enrolled (courseid varchar(50)) returns int begin return (select count(*) from teaches where course_id=courseid); end;

Description: This function is used to find the number of teachers teaching a course.

5.

Code: create function num_replies (disc_id int) returns int begin return (select count(*) from d_comments where d_id = disc_id) ; end;

Description: This function is used to find the number of replies to a discussion forum.

6.

Code: create function num_cred_st (uname varchar(50)) returns int begin return (select sum(credits) from course natural join takes where username = uname); end;

Description: This function is used to find the number of credits student has taken.

7.

Code: create procedure num_course_taught(in t_name varchar(50),out num int) begin select count(*) into num from teaches where username=t_name; end;

Description: This procedure is used to find the number of courses taught by teacher.

8.

Code: create function num_replies_comment (cid int) returns int begin return (select count(*) from d_comments where parent_id=cid); end;

Description: This procedure is used to find the number of replies to a comment.

9.

Code: create function finish_deadline(asgnid int) returns bool begin declare a timestamp ; set a = (select deadline from assignment where assignment_id = asgnid); if (select timestampdiff(second,a,now()))<0 then return true; Else return false; end if; end;

Description: This function is used to find whether the assignment's deadline has expired or not.

10.

Code: create function calc_num_news(te_id varchar(50), c_id varchar(50)) returns int begin return (select count(*) from newsforum where username = te_id and course_id=c_id); end;

Description: This function is used to find the number of news forum a teacher has in a particular course.

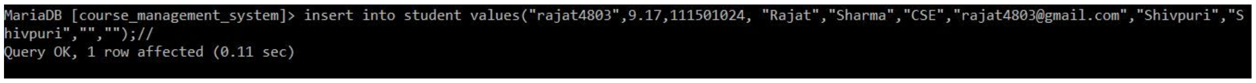
TASK 5

It couldn't be implemented.

TASK 6

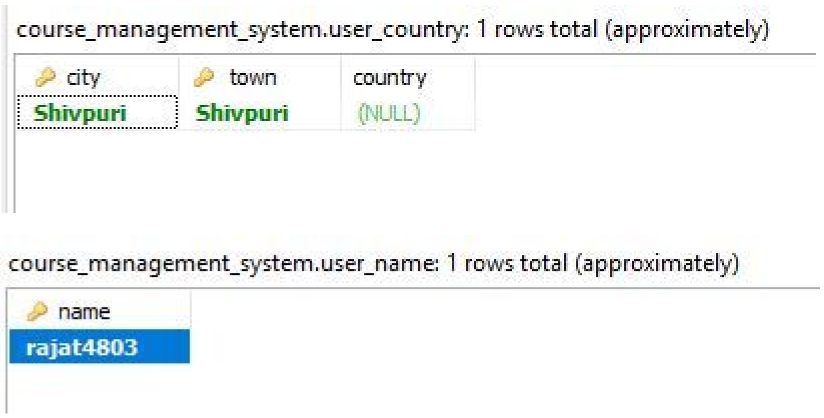
We inserted some toy data and checked whether the triggers and functions are working fine.

We first inserted data in *'student'* table corresponding to which the username got inserted in *'user_name'* table and the city/town got inserted in *'user_country'*. All of this is shown in FIG. 1 and FIG. 2 .



```
MariaDB [course_management_system]> insert into student values("rajat4803",9.17,111501024, "Rajat","Sharma","CSE","rajat4803@gmail.com","Shivpuri","S
hivpuri","","");//
Query OK, 1 row affected (0.11 sec)
```

FIG. 1



course_management_system.user_country: 1 rows total (approximately)

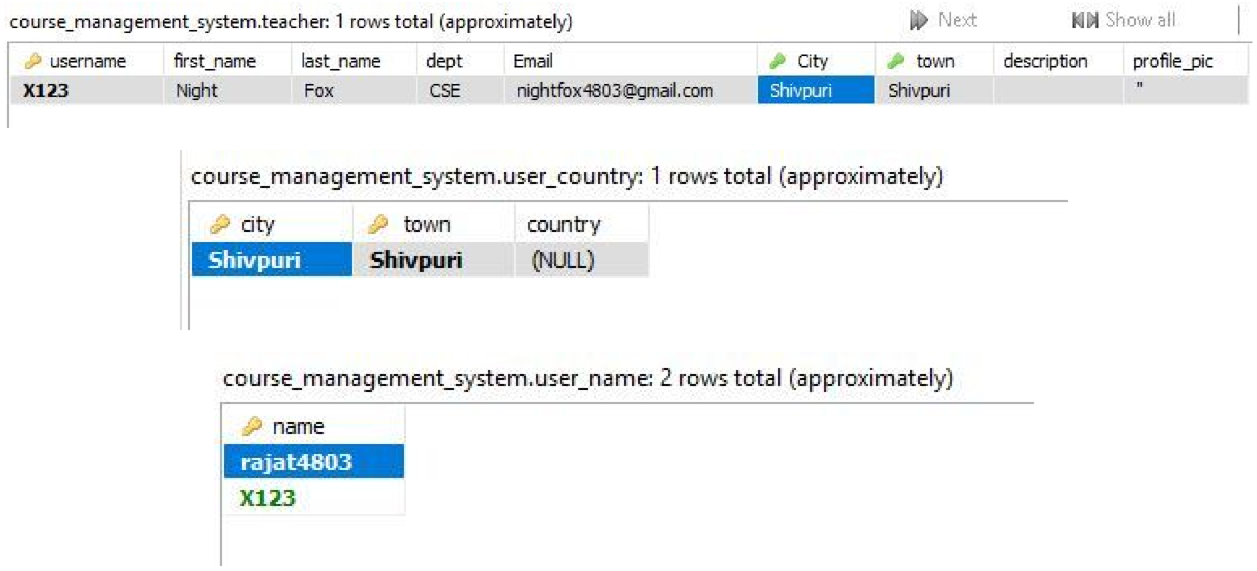
city	town	country
Shivpuri	Shivpuri	(NULL)

course_management_system.user_name: 1 rows total (approximately)

name
rajat4803

FIG. 2

Similarly we inserted in *'teacher'* table which triggered the insertion of username in *'user_name'* and city/town in *'user_country'*. This is shown in FIG 3 .



course_management_system.teacher: 1 rows total (approximately)

username	first_name	last_name	dept	Email	City	town	description	profile_pic
X123	Night	Fox	CSE	nightfox4803@gmail.com	Shivpuri	Shivpuri		"

course_management_system.user_country: 1 rows total (approximately)

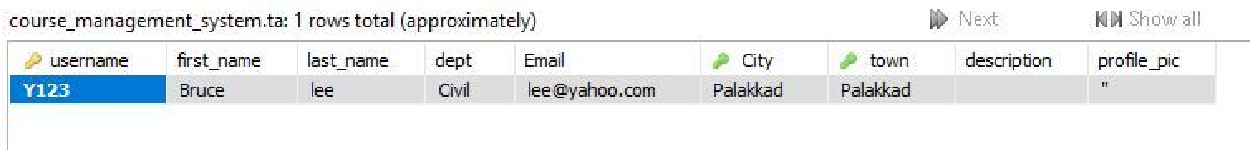
city	town	country
Shivpuri	Shivpuri	(NULL)

course_management_system.user_name: 2 rows total (approximately)

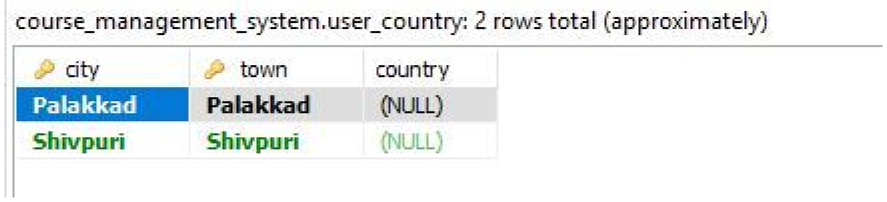
name
rajat4803
X123

FIG. 3

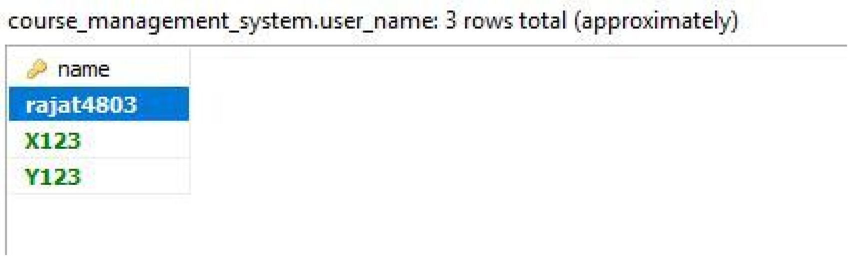
In the same way we inserted in ‘ta’ table which triggered the insertion of username in *‘user_name’* and city/town in *‘user_country’*. This is shown in FIG 4.


course_management_system.ta: 1 rows total (approximately)

username	first_name	last_name	dept	Email	City	town	description	profile_pic
Y123	Bruce	lee	Civil	lee@yahoo.com	Palakkad	Palakkad		''


course_management_system.user_country: 2 rows total (approximately)

city	town	country
Palakkad	Palakkad	(NULL)
Shivpuri	Shivpuri	(NULL)

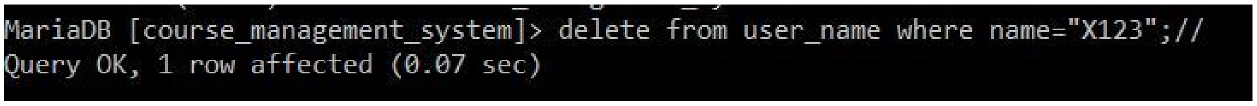

course_management_system.user_name: 3 rows total (approximately)

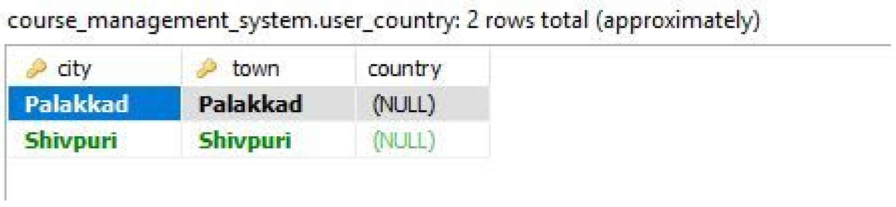
name
rajat4803
X123
Y123

FIG. 4

Inserted two users with same city/town and deleted one of the users from table *‘user_name’*. This didn’t result into the deletion of city/town from the table ‘user_country’.

This is shown in FIG. 5.

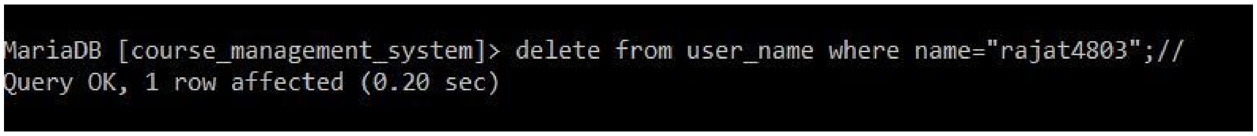

```
MariaDB [course_management_system]> delete from user_name where name="X123";//
Query OK, 1 row affected (0.07 sec)
```


course_management_system.user_country: 2 rows total (approximately)

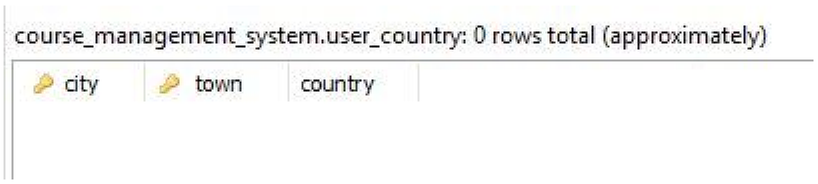
city	town	country
Palakkad	Palakkad	(NULL)
Shivpuri	Shivpuri	(NULL)

FIG. 5

Now since only one record will be left with a particular town/city, if we delete this record from *‘user_name’* this will delete the corresponding city/town from *‘user_country’*. This is shown in FIG 6.



```
MariaDB [course_management_system]> delete from user_name where name="rajat4803";//
Query OK, 1 row affected (0.20 sec)
```

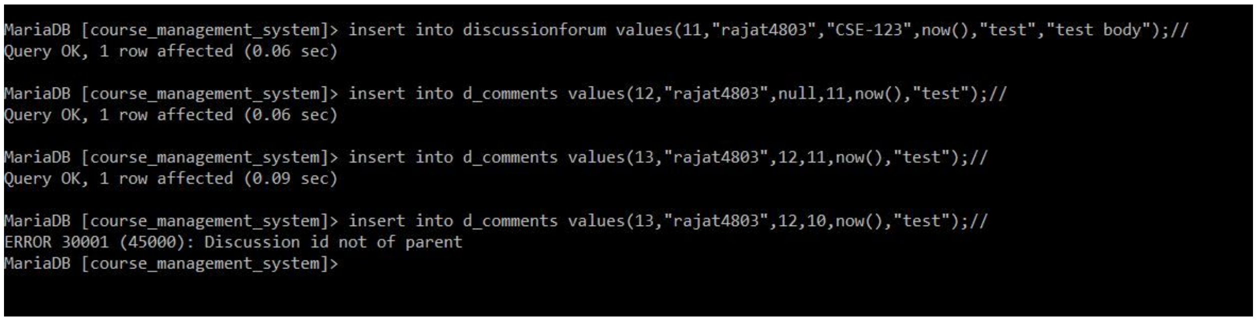


course_management_system.user_country: 0 rows total (approximately)

city	town	country

FIG. 6

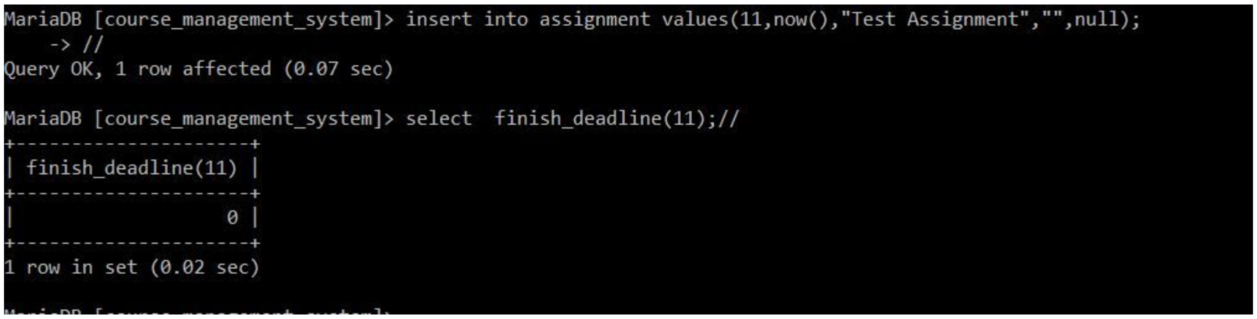
On inserting a reply to a comment with different discussion id as that of the parent comment leads to error. Shown in FIG. 7.



```
MariaDB [course_management_system]> insert into discussionforum values(11,"rajat4803","CSE-123",now(),"test","test body");//
Query OK, 1 row affected (0.06 sec)

MariaDB [course_management_system]> insert into d_comments values(12,"rajat4803",null,11,now(),"test");//
Query OK, 1 row affected (0.06 sec)

MariaDB [course_management_system]> insert into d_comments values(13,"rajat4803",12,11,now(),"test");//
Query OK, 1 row affected (0.09 sec)

MariaDB [course_management_system]> insert into d_comments values(13,"rajat4803",12,10,now(),"test");//
ERROR 30001 (45000): Discussion id not of parent
MariaDB [course_management_system]>
```

FIG. 7

Function ‘finish_deadline’ checks whether the deadline of the assignment is over or not by checking against the current time. This is shown in FIG. 8.



```
MariaDB [course_management_system]> insert into assignment values(11,now(),"Test Assignment","",null);
    -> //
Query OK, 1 row affected (0.07 sec)

MariaDB [course_management_system]> select  finish_deadline(11);//
+---------------------+
| finish_deadline(11) |
+---------------------+
|                   0 |
+---------------------+
1 row in set (0.02 sec)
```

FIG. 8

Rest all are trivial functions, and their outputs have been checked.

PHASE 4

USER INTERFACE

The UI for the "*Course Management System*" is made using Android Studio 3.0.1. Below subsections shows the working of the Android application.

Login

The login page has username and password field for the user to enter. The user can either be a Teacher, a TA or a student. Below figure shows the the login window on the app.

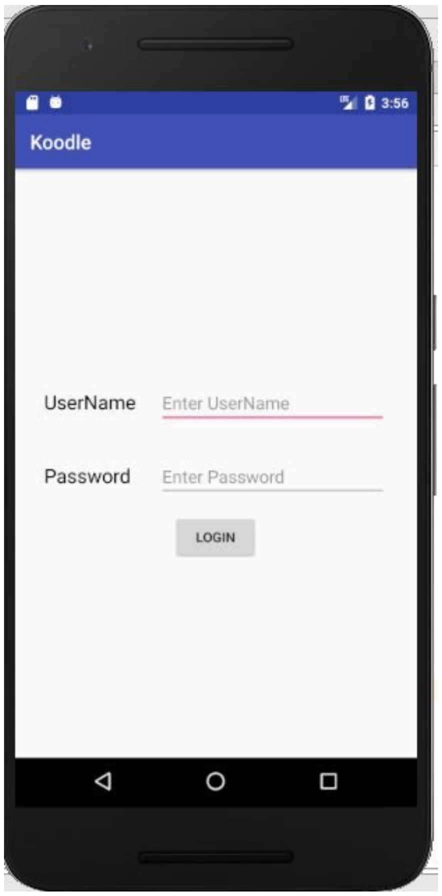
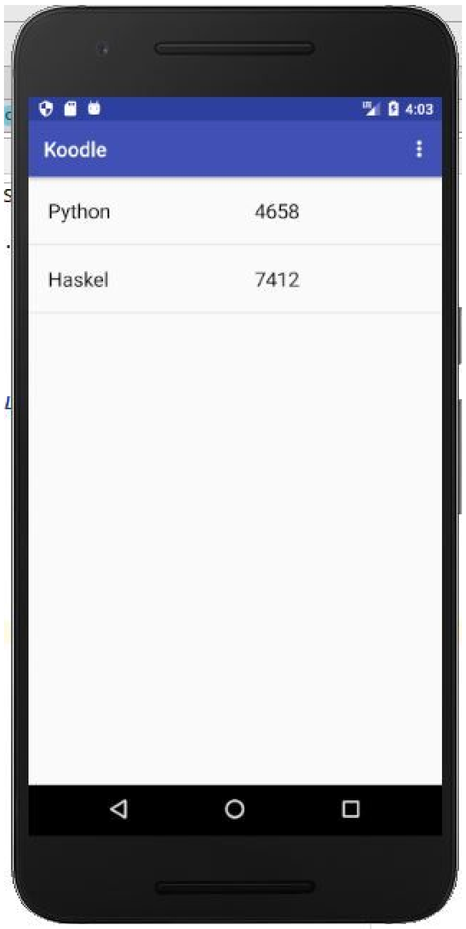


FIG. Login window

Course details

Course details of the courses offered(or taken) by the teacher(or the student) respectively, are shown on this window. The course_id along with the course_name is shown on the window. If the user clicks on the course viewholder, the user gets directed to the course's window. The activity related to the course details of a student is shown below.



Python	4658
Haskel	7412

FIG. Course details

The menu on the top right corner of the above figure is used to get the menu as shown in the below figure. This menu is meant for student.

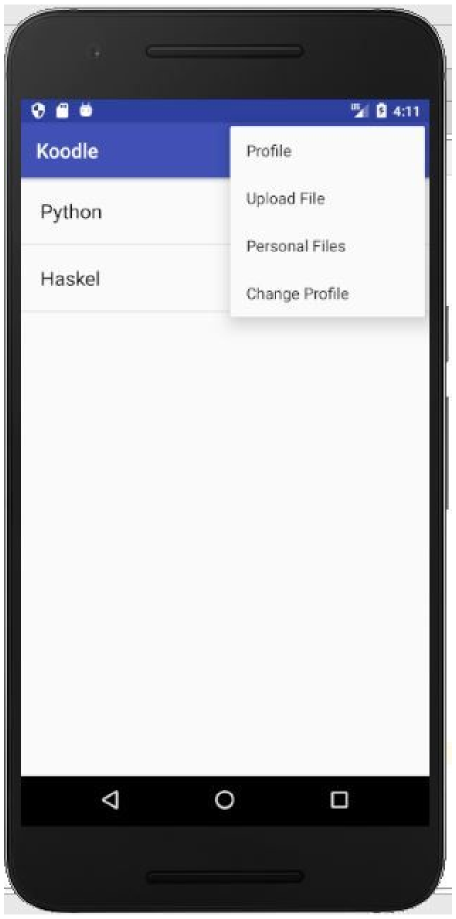
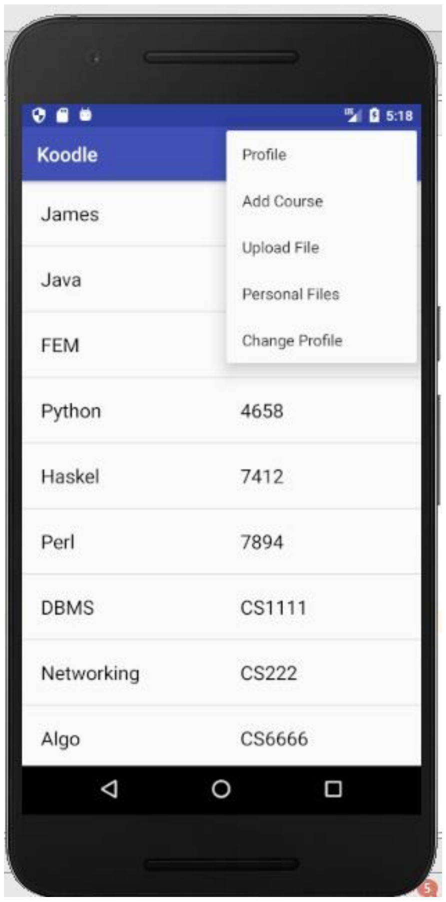


FIG. Menu for student

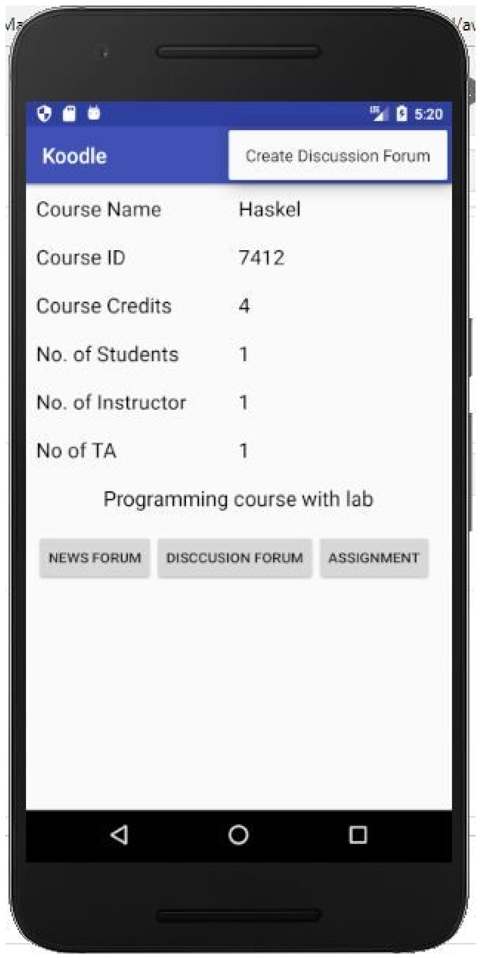
The menu for teacher is shown below.



James	
Java	
FEM	
Python	4658
Haskel	7412
Perl	7894
DBMS	CS1111
Networking	CS222
Algo	CS6666

FIG. Menu for teacher

The activity for a course taken by user consists of course id, course description, number of students enrolled in the course, number of TAs for the course and other details about the course. For the student, the activity will also have an option to see the discussion forums, the news forums and the assignments related to the course. The menu button on the top right corner, shows the menu which has an option to add a discussion forum. All these features are shown in the image below.

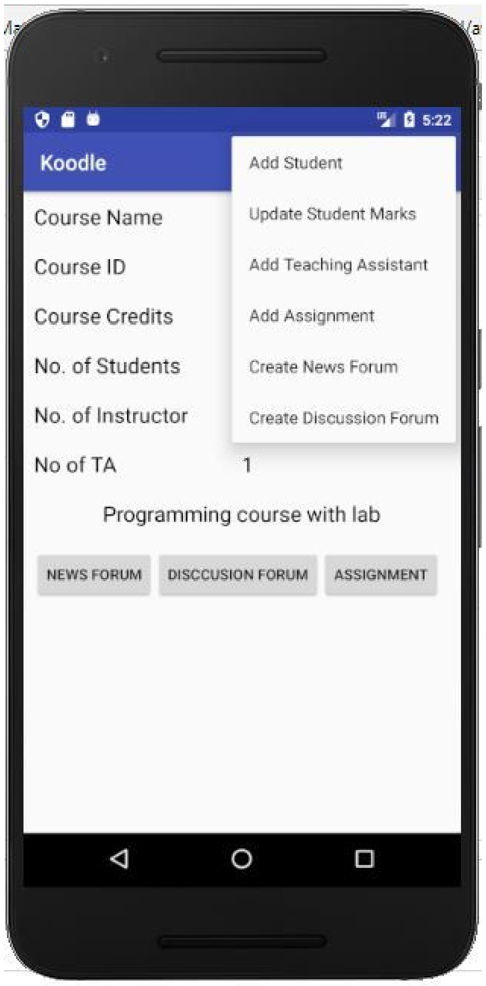


Course Name	Haskel
Course ID	7412
Course Credits	4
No. of Students	1
No. of Instructor	1
No of TA	1

Programming course with lab

FIG. Course description for student

For the teacher, the activity will also have an option to see the discussion forums and the news forums related to the course. The menu button on the top right corner, shows the menu which has an option to add a student to the course, update student's marks, add the teaching assistant to the course, add an assignment,



Course Name

Course ID

Course Credits

No. of Students

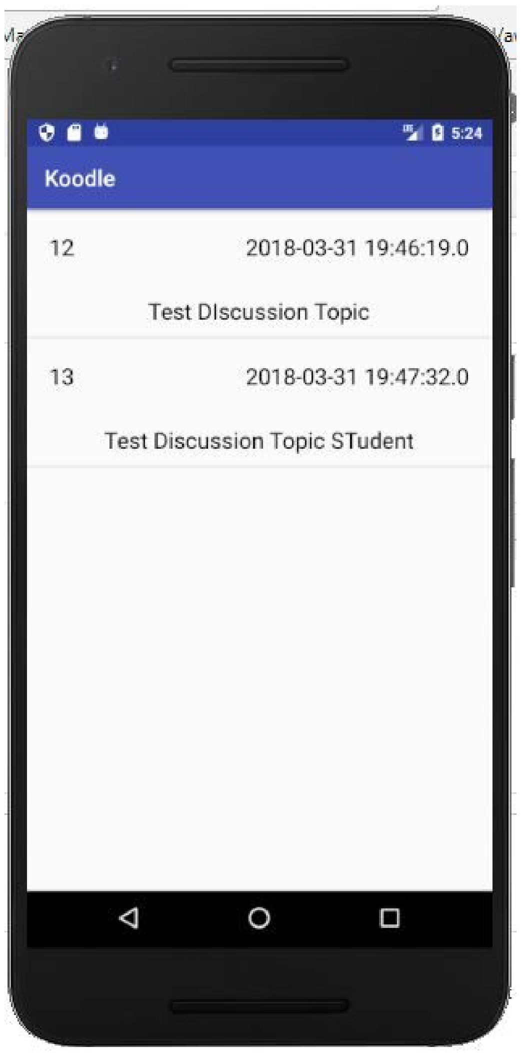
No. of Instructor

No of TA 1

Programming course with lab

FIG. Course description for teacher

An instance of a list of discussion forums is shown below.



12 2018-03-31 19:46:19.0

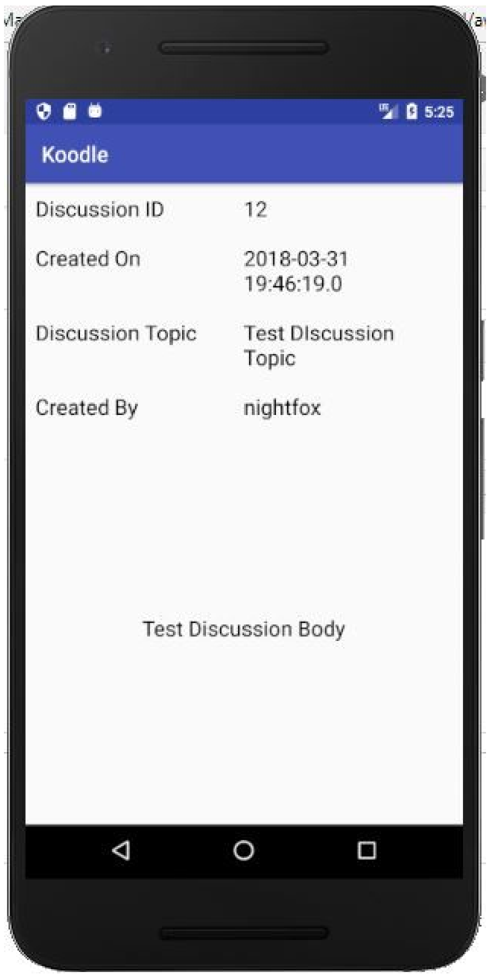
Test DIscussion Topic

13 2018-03-31 19:47:32.0

Test Discussion Topic STudent

FIG. Discussion forums list activity

On opening each discussion forum we get the following activity,

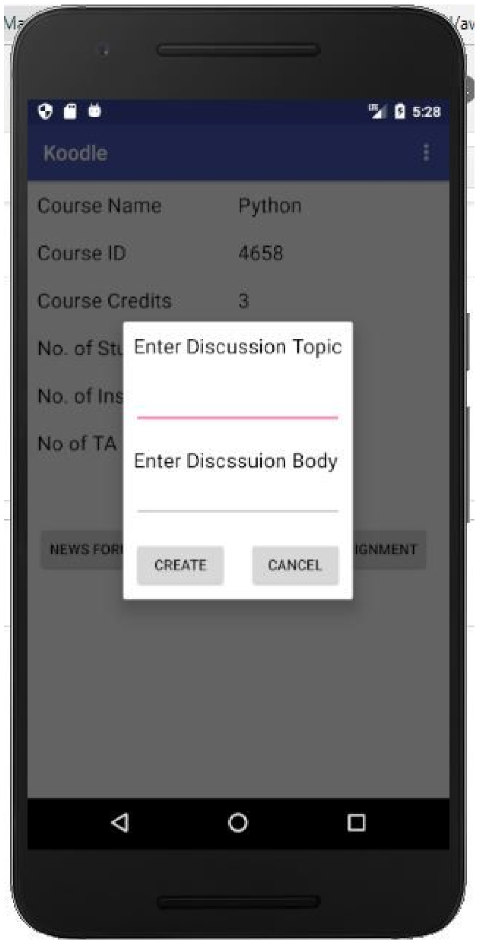


Discussion ID	12
Created On	2018-03-31 19:46:19.0
Discussion Topic	Test DIscussion Topic
Created By	nightfox

Test Discussion Body

FIG. Discussion forum description

The figure below shows the dialog box to add a discussion forum.


Koodle

Course Name Python

Course ID 4658

Course Credits 3

No. of St

No. of Ins

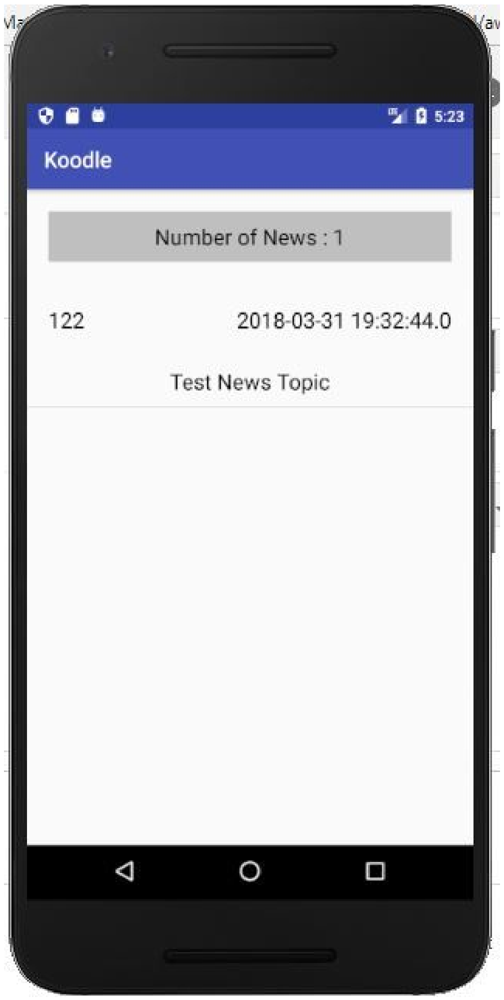
No of TA

Enter Discussion Topic

Enter Discssuion Body

FIG. Adding discussion forum

An instance of a list of news forums is shown below.


Koodle

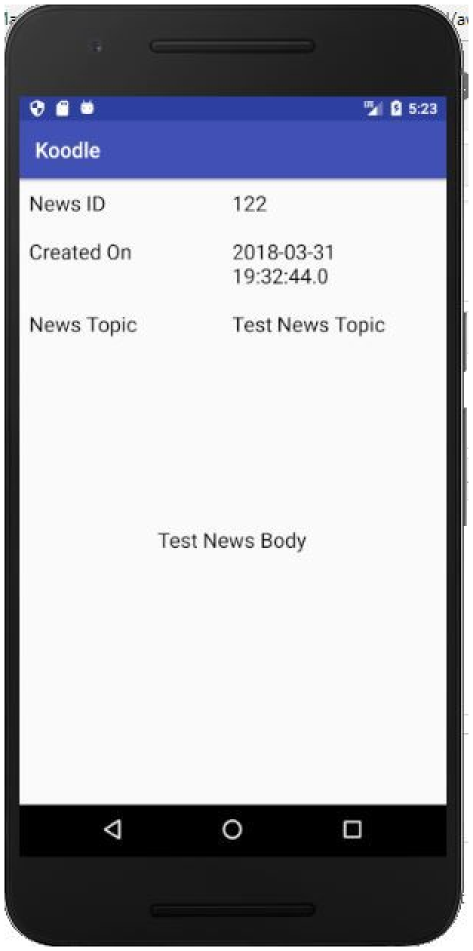
Number of News : 1

122 2018-03-31 19:32:44.0

Test News Topic

FIG. News forum list

On opening each news forum we get the following activity,


Koodle

News ID	122
Created On	2018-03-31 19:32:44.0
News Topic	Test News Topic

Test News Body

FIG. News forum description

The figure below shows the dialog box to add a news forum.

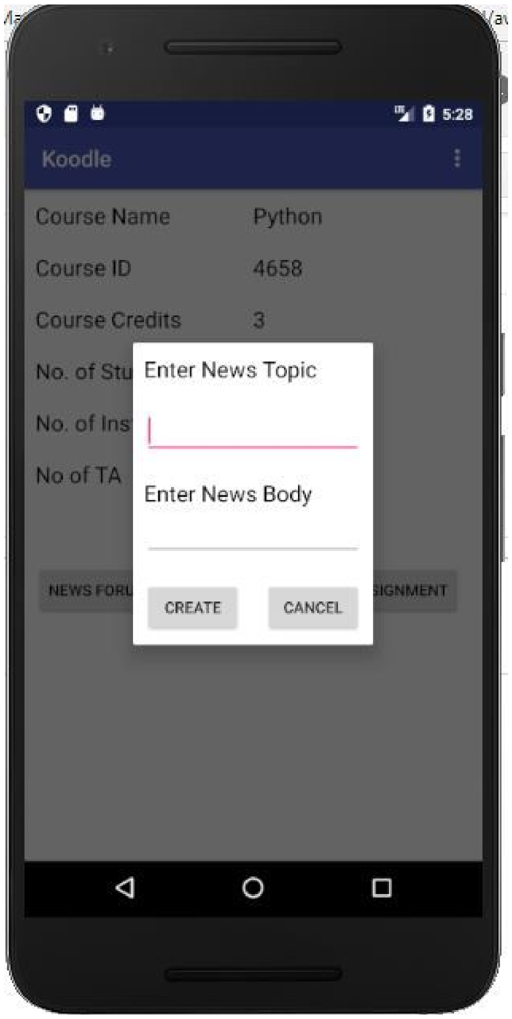
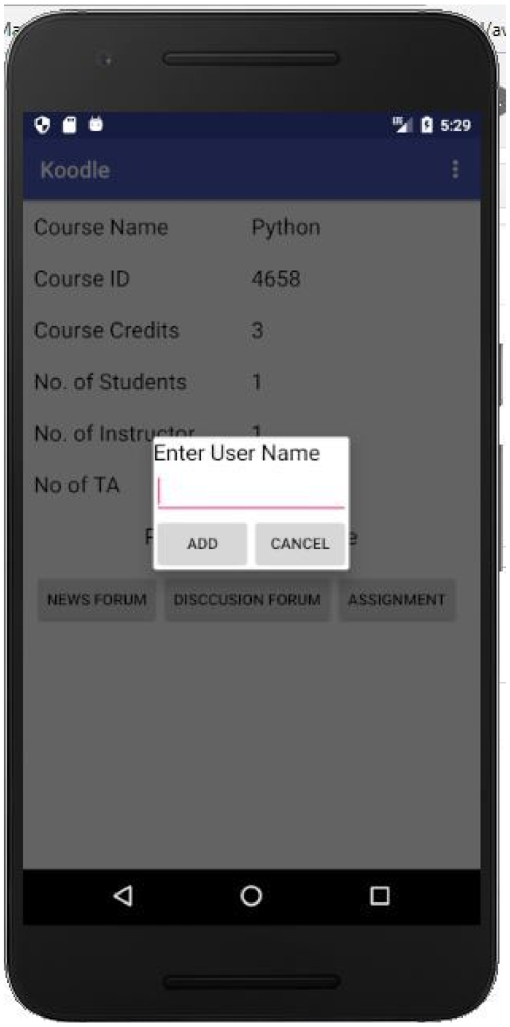


FIG. Adding news forum

Students can be added to a course by the teacher. Below figure shows this procedure.



Course Name	Python
Course ID	4658
Course Credits	3
No. of Students	1
No. of Instructor	1
No of TA	

FIG. Adding students

For updating the marks of a student, the teacher has the following dialog box,

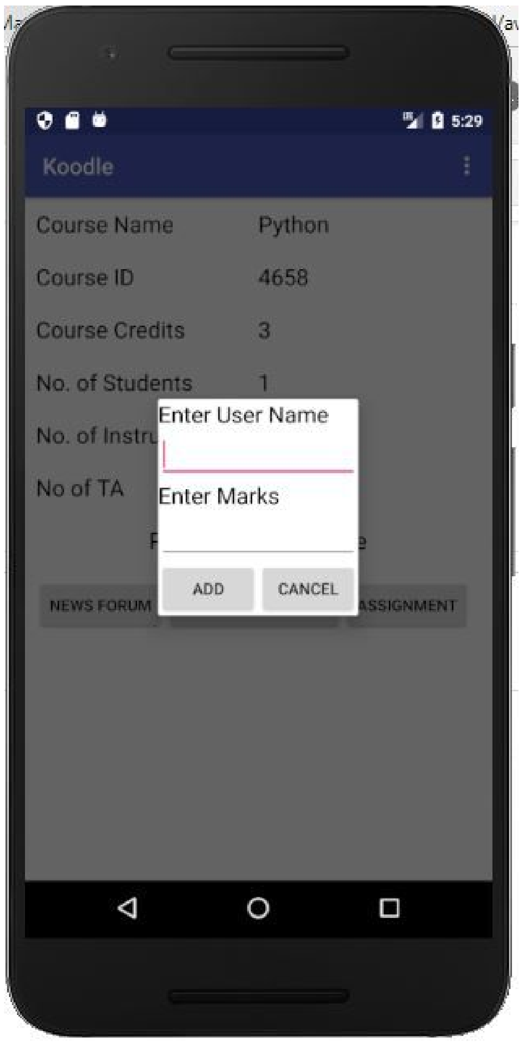
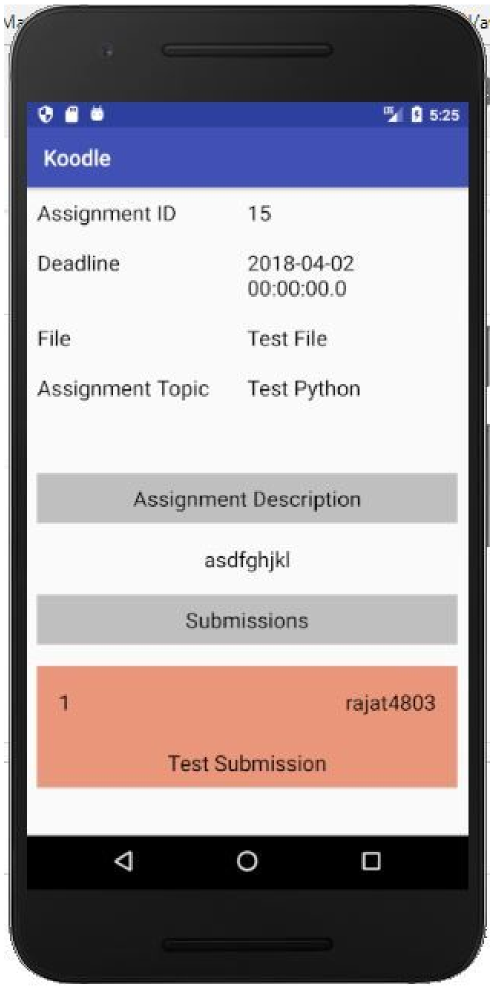


FIG. Updating marks of a student

Below figure shows the activity which shows the assignment's description.


Koodle

Assignment ID	15
Deadline	2018-04-02 00:00:00.0
File	Test File
Assignment Topic	Test Python

asdfghjkl

1 rajat4803

Test Submission

FIG. Assignment description

The marks can be added for an assignment by the TA or the teacher.

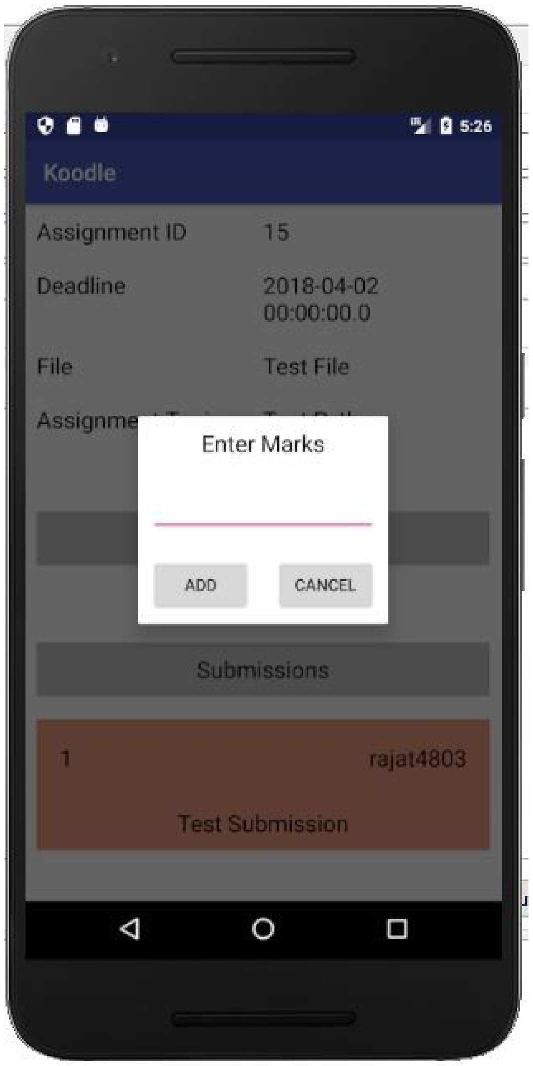
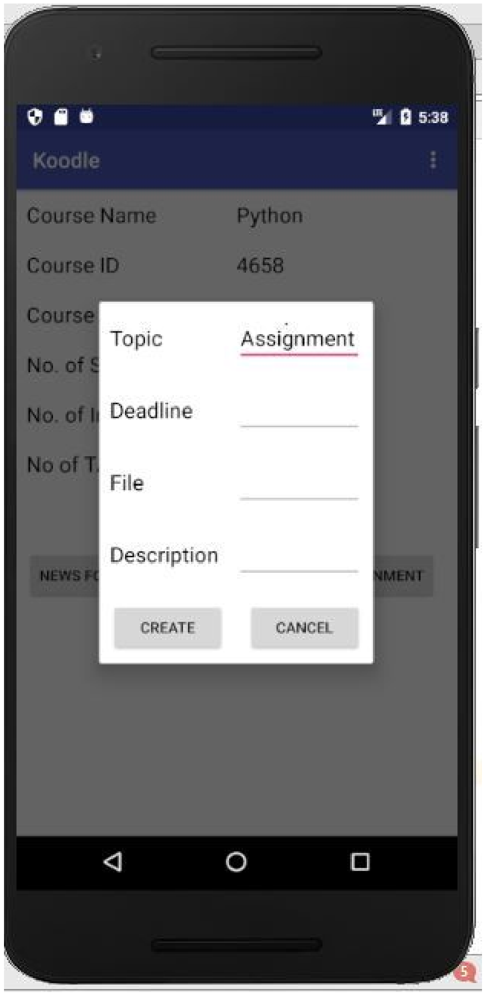


FIG. Adding marks for an assignment

For adding assignment we have the following activity,



Course Name Python

Course ID 4658

Topic Assignment

Deadline

File

Description

FIG. Adding assignment by teacher

TARGET AUDIENCE

This application can be used by universities who want to manage their courses online. This leads to less wastage of paper and also better governance of courses by teachers.