University Management System

Port Number : 8680

http://jedi.poly.edu:8680/

By

Jaya Amit Sai Gurrala (jg6660), Paul Sannihith Reddy Singareddy(ps4351), Puneeth Kasukurthi (pk2473)

Description:

Universities have a large amount of data every year which includes details about the schools which belong to it, the employees, buildings, students and so on. Students enroll for classes, they live in dorms and also make payments and requests which are handled by administrators who also oversee the professors who teach the classes. In order to make data storage and management simple and less time-consuming, an effective university management database system can be a one-stop application for all parties involved in the university.

Our application contains 4 main parts:
1.Administrators
2.Professors
3.Students
4.Universities

The user interface for the application is as follows (questions the user can ask/views available):

1.Administrators Module:

The Administrators can use the module to perform the following tasks:

- Registration: Administrators can add in their basic details to fill up the profile.
- Employee Payroll - The admins can register and update payments done to the professors.
- Accessibility Requests - After viewing the accessibility requests of the students, the admins can approve it and then send it to the concerned professor..
- Student Payment Activity - The admins can check the fees payment status of students and generate reports

2.Professors Module:

The Professors can use the module to perform the following tasks:

- Registration: Professors can add in their basic details to fill up the profile
- Information: The professors can view their classes and schools which employ them.
- Paychecks: Professors can check their paycheck status.
- Accessibility Requests : Professors can view approved accessibility requests submitted by the students.

3.Students Module:

The students can use the module to perform the following tasks:

- Registration: Professors can add in their basic details to fill up the profile
- Accessibility Requests: Students can submit accessibility requests based on their physical or mental health and give a description of their request and the Professor to whom the request should be sent.
- Request Status Update: Students can check the status of their request here.
- Fees Payments: Students can use this to pay their fees and also fetch reports of their payments.
- Classes: Students can see the details of the classes to which they are enrolled.
- Dorm Information: Students can see the details of their dorms here.

4.Universities:

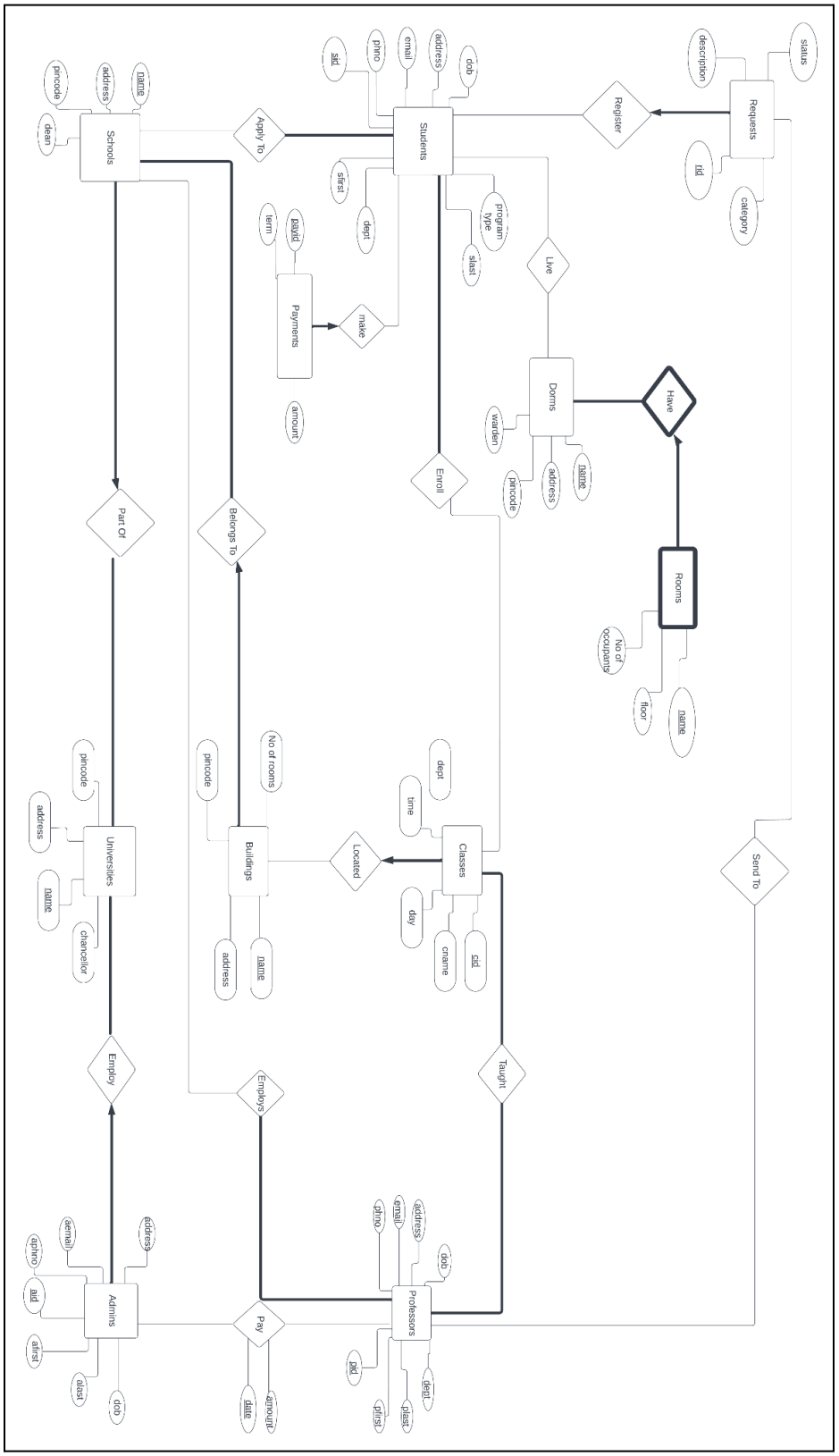
The user can perform the following tasks in this module:

- University Information: The user can see the complete details of both the schools which are part of the university as well as the buildings which belong to it.
- Buildings Information: The user can enter the name of a school and building along with a start and end time and we display the details of all the classes which take place in that building during that time frame.

Business Rules for the application are:

1. Each student should enroll in at least 1 class
2. Each student should apply to at least 1 school
3. Each payment carried out relates to a single student
4. Each accessibility request made relates to a single student
5. A student may raise a request and update the status to ‘completed’ once the request is directed to professor, else the status remains ‘in progress’
6. Each room is located in a single dorm. A dorm can have at least one room. If a dorm is destroyed, the rooms in it will also be destroyed too
7. Admins will carry out the salary payments to the professors. Each admin may or may not manage salary payment
8. Each accessibility request will be directed to the professor, if assigned
9. Building part of an university will belong to exactly one school, whereas a school can have at least more than 1 building
10. Universities will have more than 1 school and a given school will belong to exactly one university
11. Universities employs multiple admins whereas a given admin should work exactly in 1 university
12. A professor can work in different schools
13. Professor can take up multiple classes and in the same way a given class can be taught by multiple professors
14. A given class is located in exactly one building

ER Diagram of the application:



The application has the following entities and relationship sets:

1. Students: It contains the following information related to students - their sid (student id), first, last names, phone number, email, address, date of birth, department, program_type
2. Professors: It contains the following information related to professors - their pid (professor id), first, last names, phone number, email, address, date of birth, department
3. Payments: Students carry out their semester fees payments and this entity captures the information. It includes payid, term, amount and id of the student who carried out the payment
4. Universities: This entity captures the information related to students. It includes the name of the university, address, pincode, name of the chancellor
5. Schools_partof: Information related to different schools, part of different universities are stored in this entity. It has the following entities like name of the building, address, pincode, dean of the corresponding school, university name to which the given school belongs to
6. Buildings_belong_to: Contains the information of buildings which belong to different schools. It includes name, address, pincode, no of rooms part of the building, name of the school to which this building belongs to
7. Classes_located: Contains the following information about classes - cid (Course ID), cname (Course Name), day, time, department, name of the building to which the class belongs to
8. Requests: The student fills in a request related to accessibility for the exam which is captured in this table. It contains the following information- rid (Request ID), category of the request, description of the request, sid (Student ID) who is raising the request, status of the request, Professor ID to whom the request is being forwarded to
9. Dorms: This table captures the information related to dorms. It includes the name of the dorm, address, pincode and name of the warden
10. Rooms_have: Contains the information related to rooms - Room No, floor No, No of occupants in a given room, name of the dorm to which this building belongs to
11. Admins_employ: It contains the following information related to admins - aid (Admin ID), first, last names, phone number, email address, address, date of birth, name of the university at which the given admin works
12. Send_To: Student requests are assigned to the corresponding professor, so the table stores professor id, and the corresponding student request which was raised by the student
13. Enroll: This entity captures the information about the courses enrolled by the student. The table stores sid (Student ID), and the corresponding course ID's to which he was enrolled.
14. Live: It captures the information about the dorm where the student resides. It includes sid (Student ID), dorm name.
15. Taught: This entity captures the information about the courses taught by a professor. It includes the professor ID and course ID

16. Apply_to: It captures the information about the schools for which the given student has applied. Hence it includes sid (Student ID), School name to which the student has enrolled
17. Employs: Captures the information about the schools employing the professors. It includes pid (Professor ID), name of the school where the professor works
18. Pay: Contains the following information about professor salaries - Professor ID to whom the payment is made, Admin ID who carried out the payment, amount of the payment, date of the payment

Data Loading Procedure:

The data for the application was taken from various online sources and it was self constructed also. Initially we have filled up to 20 tuples for each table in the given CSV files.

The following commands were used to load the tables from the CSV Files for different tables.

```
cat final_project/Data/Admins_employ.csv | psql -U ps4351 -d ps4351_db -c “COPY Admins_employ from STDIN CSV HEADER”
cat final_project/Data/Apply_to.csv | psql -U ps4351 -d ps4351_db -c “COPY Apply_to from STDIN CSV HEADER”
cat final_project/Data/Buildings_belong_to.csv | psql -U ps4351 -d ps4351_db -c “COPY Buildings_belong_to from STDIN CSV HEADER”
cat final_project/Data/Classes_Located.csv | psql -U ps4351 -d ps4351_db -c “COPY Classes_Located from STDIN CSV HEADER”
cat final_project/Data/Dorms.csv | psql -U ps4351 -d ps4351_db -c “COPY Dorms from STDIN CSV HEADER”
cat final_project/Data/Employs.csv | psql -U ps4351 -d ps4351_db -c “COPY Employs from STDIN CSV HEADER”
cat final_project/Data/Enroll.csv | psql -U ps4351 -d ps4351_db -c “COPY Enroll from STDIN CSV HEADER”
cat final_project/Data/Live.csv | psql -U ps4351 -d ps4351_db -c “COPY Live from STDIN CSV HEADER”
cat final_project/Data/Pay.csv | psql -U ps4351 -d ps4351_db -c “COPY Pay from STDIN CSV HEADER”
cat final_project/Data/Payments.csv | psql -U ps4351 -d ps4351_db -c “COPY Payments from STDIN CSV HEADER”
cat final_project/Data/Professors.csv | psql -U ps4351 -d ps4351_db -c “COPY Professors from STDIN CSV HEADER”
```

cat final_project/Data/Requests.csv | psql -U ps4351 -d ps4351_db -c “COPY Requests from STDIN CSV HEADER”
cat final_project/Data/Rooms_have.csv | psql -U ps4351 -d ps4351_db -c “COPY Rooms_have from STDIN CSV HEADER”
cat final_project/Data/Schools_partof.csv | psql -U ps4351 -d ps4351_db -c “COPY Schools_partof from STDIN CSV HEADER”
cat final_project/Data/Send_to.csv | psql -U ps4351 -d ps4351_db -c “COPY Send_to from STDIN CSV HEADER”
cat final_project/Data/Students.csv | psql -U ps4351 -d ps4351_db -c “COPY Students from STDIN CSV HEADER”
cat final_project/Data/Taught.csv | psql -U ps4351 -d ps4351_db -c “COPY Taught from STDIN CSV HEADER”
cat final_project/Data/Universities.csv | psql -U ps4351 -d ps4351_db -c “COPY Universities from STDIN CSV HEADER”

Schema

```
drop table if exists Pay;
drop table if exists Rooms_have;
drop table if exists Live;
drop table if exists Taught;
drop table if exists Apply_to;
drop table if exists Payments;
drop table if exists Enroll;
drop table if exists Send_To;
drop table if exists Admins_employ;
drop table if exists Employs;
drop table if exists Professors;
drop table if exists Requests;
drop table if exists Classes_located;
drop table if exists Students;
drop table if exists Buildings_belong_to;
drop table if exists Schools_partof;
drop table if exists Dorms;
drop table if exists Universities;


create table Students (
  sid integer primary key,
  sfirst varchar(128),
```

```
  slast varchar(128),
  phno char(10),
  email varchar(64),
  address varchar(512),
  dob date,
  dept varchar(32),
  program_type varchar(32)
);

create table Professors(
  pid integer primary key,
  pfirst varchar(128),
  plast varchar(128),
  phno char(10),
  email varchar(64),
  address varchar(512),
  dob date,
  dept varchar(32)
);

create table Payments(
  payid serial primary key,
  term varchar(10),
  amount decimal,
  sid integer not null,
  foreign key( sid ) references Students( sid )
);

create table Universities(
  name varchar(128) primary key,
  address varchar(128),
  pincode integer,
  chancellor varchar(128)
);

create table Schools_partof(
  name varchar(128) primary key,
  address varchar(128),
  pincode integer,
  dean varchar(128),
```

```
  uname varchar(128) not null,
  foreign key(uname) references Universities(name)
);

create table Buildings_belong_to(
  name varchar(32) primary key,
  address varchar(128),
  pincode integer,
  no_of_rooms integer,
  schoolname varchar(32) not null,
  foreign key(schoolname) references Schools_partof(name)
);

create table Classes_located (
  cid varchar(32) primary key,
  cname varchar(128),
  day varchar(32),
  time time,
  dept varchar(32),
  bname varchar(32) not null,
  foreign key(bname) references Buildings_belong_to(name)
);

create table Requests(
  rid serial primary key,
  category varchar(32),
  description varchar(128),
  sid integer not null,
  status varchar(16),
  prof_id integer,
  foreign key( sid ) references Students( sid )
);

create table Dorms (
  name varchar(32) primary key,
  address varchar(128),
  pincode integer,
  warden varchar(32)
);
```

```
create table Rooms_have(
   rno integer,
   floor integer,
   No_of_occupants integer,
   dname varchar(32),
   primary key(rno,dname),
   foreign key(dname) references Dorms(name) on delete cascade
);


create table Admins_employ(
   aid integer primary key,
   afirst varchar(128),
   alast varchar(128),
   aphno char(10),
   aemail varchar(64),
   address varchar(512),
   dob date,
   uname varchar(32) not null,
   foreign key(uname) references Universities(name)
);

create table Send_To(
   pid integer,
   rid integer,
   primary key(pid,rid),
   foreign key (pid) references Professors(pid),
   foreign key (rid) references Requests(rid)
);

create table Enroll(
   sid integer,
   cid varchar(32),
   primary key (sid, cid),
   foreign key(sid) references Students(sid),
   foreign key(cid) references Classes_located(cid)
);

create table Live(
   sid integer,
```

```
  dname varchar(32),
  primary key(sid, dname),
  foreign key(sid) references Students(sid),
  foreign key(dname) references Dorms(name)
);

create table Taught(
  cid varchar(32),
  pid integer,
  primary key(cid, pid),
  foreign key(cid) references Classes_located(cid),
  foreign key(pid) references Professors(pid)
);

create table Apply_to(
  sid integer,
  sname varchar(32),
  primary key(sid, sname),
  foreign key(sid) references Students(sid),
  foreign key(sname) references Schools_partof(name)
);

create table Employs(
  pid integer,
  sname varchar(32),
  primary key(pid, sname),
  foreign key(pid) references Professors(pid),
  foreign key(sname) references Schools_partof(name)

);

create table Pay(
  pid integer,
  aid integer,
  amount integer,
  dates date,
  primary key(pid, aid, dates),
  foreign key(pid) references Professors(pid),
  foreign key(aid) references Admins_employ(aid)
);
```

```
alter sequence requests_rid_seq restart with 500;
alter sequence payments_pid_seq restart with 500;
```

The following constraints could not be captured:

- The participation constraint of Students in relationship Enroll
- The participation constraint of Students in relationship Apply To
- The participation constraint of Classes in relationship Taught
- The participation constraint of Professors in relationship Taught
- The participation constraint of Professors in relationship Employs