Bilkent University

Department of Computer Engineering

CS 353: DATABASE SYSTEMS

Group 3: Online Course Platform

Project Design Report

Group Members:

Işık Özsoy 21703160 Section: 2

Defne Betül Çiftci 21802635 Section: 3

Şebnem Uslu 21802068 Section: 2

Melike Fatma Aydoğan 21704043 Section: 2

Instructors: Özgür Ulusoy, Uğur Güdükbay
Teaching Assistant: Mustafa Can Çavdar

Design Report

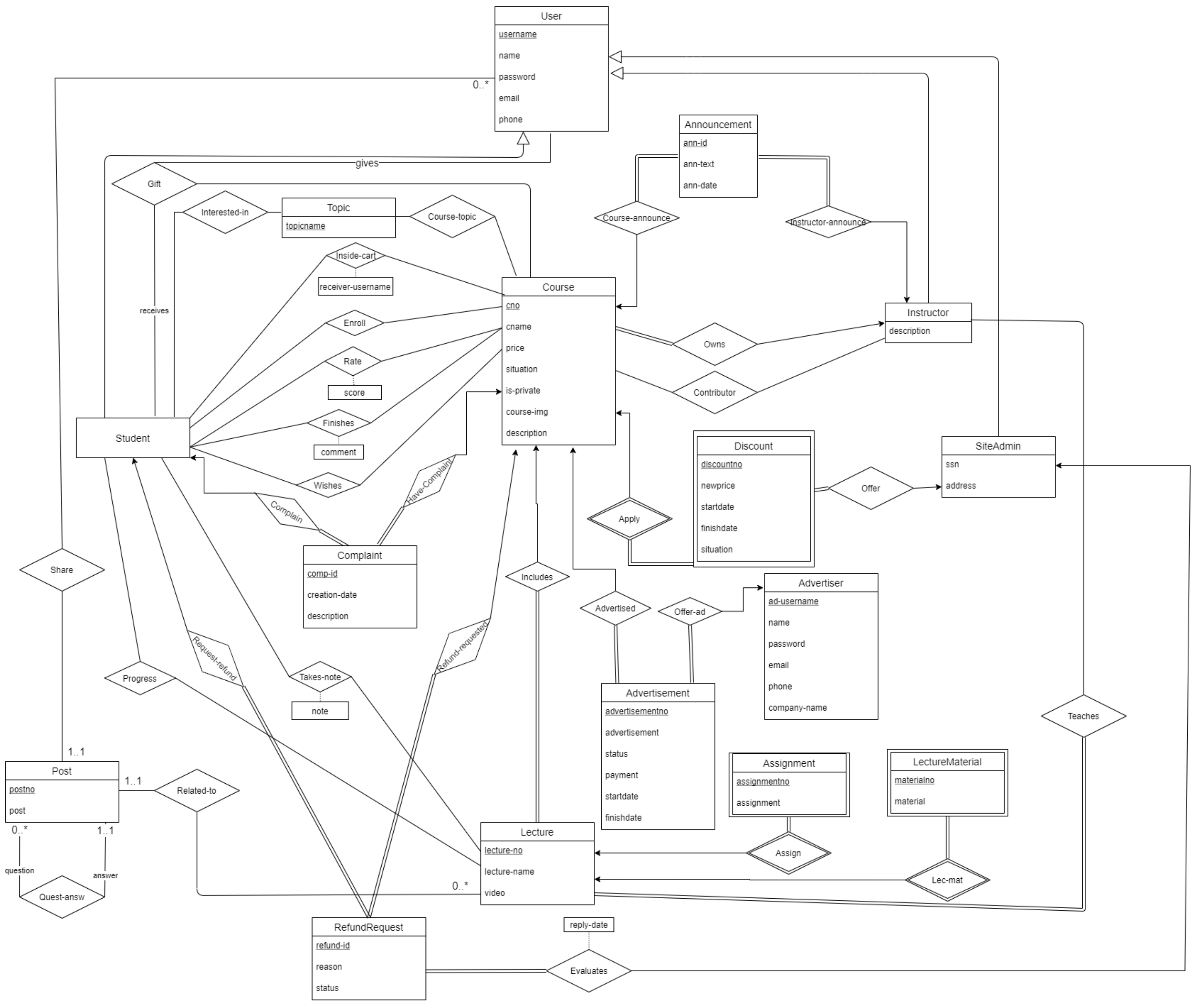
April 2, 2021

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Database Systems course CS353.

Table of Contents

I. Revised E/R Model

Revised E/R Diagram



Changes Made In E/R Diagram

- Primary key for RefundRequest (refund-id) was added.
- Participation of RefundRequest in the Request-refund relation became total.
- Advertiser is now not a User, but a separate entity with mostly same attributes except for company-name and ad-username.
- Lecture became a normal entity instead of a weak entity.
- Evaluates relation was added with attribute reply-date which shows the date that Admin evaluates the request. This relation demonstrates which request is evaluated by who and when.
- Complain relation was transformed from a three-way relation into two different binary relations named “Complain” that connects Complaint and Student and “Have-Complain” that connects Course and Complaint.

- Offer-ad relation was transformed from a three-way relation into two different binary relations named “Offer-ad” that connects Advertisement and Advertiser and “Advertised” that connects Advertisement and Course.
- Request-refund relation was transformed from a three-way relation into two different binary relations named “Request-refund” that connects Student and RefundRequest and “Refund-requested” that connects Course and RefundRequest.
- Takes-note is now not connected to Course but connected to Lecture.
- A many-to-many “In-cart” relation is added between Student and Course that has an attribute named “receiver-email”.
- Participation of Announcement to Course and Instructor became total.

II. Relation Schemas

User

Relational Model: User (username, name, password, email, phone)
Candidate Key:

username
email

Primary Key: username
Table Definition:

```
create table User (
        username char (50),
        name char (50),
        password char(50),
        email char (50),
        phone char (50),
        primary key (username)
)
engine=InnoDB;
```

Student

Relational Model: Student (username)
Candidate Key:

username

Primary Key: username
Foreign Key:

username referencing User

Table Definition:

```
create table Student (
        username char (50),
        primary key (username),
        foreign key (username) references User
)
engine=InnoDB;
```

Instructor

Relational Model: Instructor (<u>username</u>, description)
Candidate Key:
    username
Primary Key: username
Foreign Key:
    username referencing User
Table Definition:
```
create table Instructor (
	username char(50),
	description char(1000),
	primary key (username),
	foreign key (username) references User
)
engine=InnoDB;
```

SiteAdmin

Relational Model: SiteAdmin (<u>username</u>, ssn, address)
Candidate Key:
    username
    ssn
Primary Key: username
Foreign Key:
    username referencing User
Table Definition:
```
create table SiteAdmin (
	username char (50),
	ssn char (20),
	address char (100),
	primary key (username),
	foreign key (username) references User
)
engine=InnoDB;
```

Advertiser

Relational Model: Advertiser (<u>ad-username</u>, name, password, email, phone, company-name)
Candidate Key:
    ad-username
    email
Primary Key: username
Table Definition:
```
create table Advertiser (
	ad-username char (50),
	name char (50),
	password char (50),
	email char (50),
        phone char (50),
        company-name char (100),
        primary key (ad-username)
)
engine=InnoDB;
```

Course

Relational Model: Course (<u>cno</u>, owner-username, cname, price, situation, is-private, course-img, description)
Candidate Key:
    cno
Primary Key: cno
Foreign Key:
    owner-username referencing Instructor (username)
Table Definition:

```
create table Course (
        cno int,
        owner-username char (50),
        cname char (50),
        price numeric (6,2),
        situation smallint,
        is-private smallint,
        course-img varchar(512),
        description varchar (1000),
        primary key (cno),
        foreign key (owner-username) references Instructor (username)
)
engine=InnoDB;
```

Gift

Relational Model: Gift (<u>sender-username, receiver-username, cno</u>)
Candidate Key:
    sender-username, receiver-username, cno
Primary Key: sender-username, receiver-username, cno
Foreign Key:
    sender-username referencing User (username)
    receiver-username referencing Student (username)
    cno referencing Course
Table Definition:

```
create table Gift (
        sender-username int,
        receiver-username int,
        cno int,
        primary key (sender-username, receiver-username, cno),
        foreign key (sender-username) references User (username),
        foreign key (receiver-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Complaint

Relational Model: Complaint (comp-id, s-username, cno, creation-date, description)
Candidate Key:
comp-id
Primary Key: comp-id
Foreign Key:
s-username referencing Student (username)
cno referencing Course
Table Definition:

```
create table Complaint-made (
        comp-id int,
        s-username char(50),
        cno int,
        creation-date date,
        description char(1000),
        primary key (comp-id),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Takes-note

Relational Model: Takes-note (s-username, lecture-no, note)
Candidate Key:
s-username, lecture-no
Primary Key: s-username, lecture-no
Foreign Key:
s-username referencing Student (username)
cno referencing Course
Table Definition:

```
create table Takes-Note (
        s-username char (50),
        lecture-no int,
        note varchar (4000),
        primary key (s-username, lecture-no),
        foreign key (s-username) references Student (username)
)
engine=InnoDB;
```

Wishes

Relational Model: Wishes (s-username, cno**)**
Candidate Key:

s-username, cno

Primary Key: s-username, cno

Foreign Key:

s-username referencing Student (username)

cno referencing Course

Table Definition:

```
create table Wishes (
        s-username char(50),
        cno int,
        primary key (s-username, cno),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Finishes

Relational Model: Finishes (<u>s-username, cno</u>, comment)

Candidate Key:

s-username, cno

Primary Key: s-username, cno

Foreign Key:

s-username referencing Student (username)

cno referencing Course

Table Definition:

```
create table Finishes (
        s-username char(50),
        cno int,
        comment varchar (150),
        primary key (s-username, cno),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Rate

Relational Model: Rate (<u>s-username, cno</u>, score)

Candidate Key:

s-username, cno

Primary Key: s-username, cno

Foreign Key:

s-username referencing Student (username)

cno referencing Course

Table Definition:

```
create table Rate (
        s-username char (50),
        cno int,
        score int,
        primary key (s-username, cno),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Enroll

Relational Model: Enroll (<u>s-username, cno</u>)
Candidate Key:
    s-username, cno
Primary Key: s-username, cno
Foreign Key:
    s-username referencing Student (username)
    cno referencing Course
Table Definition:

```
create table Enroll (
        s-username char (50),
        cno int,
        primary key (s-username, cno),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Announcement

Relational Model: Announce (<u>ann-id</u>, cno, i-username, ann-tex, ann-date)
Candidate Key:
    ann-id
Primary Key: ann-id
Foreign Key:
    cno referencing Course
    username referencing Instructor
Table Definition: create table Announcement (

```
        ann-id char(20),
        s-username char(50),
        cno int,
        ann-text varchar(1000),
        ann-date date,
        primary key (ann-id),
        foreign key (i-username) references Instructor (username),
        foreign key (cno) references Course
)
engine=InnoDB;
```

Contributor

Relational Model: Contributor (<u>cno, i-username</u>)
Candidate Key:
  cno, i-username
Primary Key: cno, i-username
Foreign Key:
  cno referencing Course
  i-username referencing Instructor (username)
Table Definition:
```
create Contributor (
	cno int,
	i-username char(50),
	primary key (cno, i-username),
	foreign key (cno) references Course,
	foreign key (i-username) references Instructor (username)
)
engine=InnoDB;
```

Lecture

Relational Model: Lecture (<u>lecture-no</u>, lecture-name, video, cno)
Candidate Key:
  lecture-no
Primary Key: lecture-no
Foreign Key:
  cno referencing Course
Table Definition:
```
create table Lecture (
	lecture-no int,
	lecture-name char (200),
	video char (100),
	cno int,
	primary key (lecture-no),
	foreign key (cno) references Course (cno)
)
engine=InnoDB;
```

Progress

Relational Model: Progress (<u>s-username, lecture-no</u>)
Candidate Key:
  s-username, lecture-no
Primary Key: s-username, lecture-no
Foreign Key:
  s-username referencing Student (username)
  lecture-no referencing Lecture
Table Definition:
```
create table Progress (
        s-username char(50),
        lecture-no int,
        primary key (s-username, lecture-no),
        foreign key (s-username) references Student (username),
        foreign key (lecture-no) references Lecture (lecture-no)
)
engine=InnoDB;
```

Teaches

Relational Model: Teaches (<u>i-username, lecture-no</u>)
Candidate Key:
    i-username, lecture-no
Primary Key: i-username, lecture-no
Foreign Key:
    lecture-no referencing Lecture
    i-username referencing Instructor (username)
Table Definition:

```
create table Teaches (
        i-username char(50),
        lecture-no int,
        primary key (i-username, lecture-no),
        foreign key (i-username) references Instructor (username),
        foreign key (lecture-no) references Lecture (lecture-no)
)
engine=InnoDB;
```

Topic

Relational Model: Topic (<u>topicname</u>)
Candidate Key:
    topicname
Primary Key: topicname
Table Definition:

```
create table Topic(
        topicname char(100),
        primary key ( topicname )
)
engine=InnoDB;
```

Course-topic

Relational Model: Course-topic (<u>cno, topicname</u>)
Candidate Key:
    cno, topicname
Primary Key: cno, topicname
Foreign Key:
    cno referencing Course
    topicname referencing Topic

Table Definition:
```
create table Course-topic(
        cno int,
        topicname char(100),
        primary key( cno, topicname ),
        foreign key (cno) references Course (cno),
        foreign key ( topicname ) references Topic (topicname)
)
engine=InnoDB;
```

Interested-in

Relational Model: Interested-in (<u>s-username, topicname</u>)
Candidate Key:
 s-username, topicname
Primary Key: s-username, topicname
Foreign Key:
 s-username referencing Student (username)
 topicname referencing Topic
Table Definition:
```
create table Interested-in(
        s-username char (50),
        topicname char (100),
        primary key ( s-username, topicname ),
        foreign key ( s-username ) references Student (username),
        foreign key ( topicname ) references Topic (topicname)
)
engine=InnoDB;
```

Discount

Relational Model: Discount (<u>discountno</u>, newprice, startdate, finishdate, situation, cno, admin-username)
Candidate Key:
 discountno
Primary Key: discountno
Foreign Key:
 cno referencing Course
 admin-username referencing SiteAdmin (username)
Table Definition:
```
create table Discount(
        discountno int,
        newprice numeric(6,2),
        startdate Date,
        finishdate Date,
        situation smallint,
        cno int,
        admin-username char(50),
        primary key ( discountno ),
        foreign key ( cno ) references Course (cno),
        foreign key ( admin-username ) references SiteAdmin (username)
)
engine=InnoDB;
```

Post

Relational Model: Post (postno, lecture-no, post, username)
Candidate Key:
postno
Primary Key: postno
Table Definition:

```
create table Post (
        postno int,
        lecture-no int,
        post char(200),
        username char(50),
        primary key (postno),
        foreign key (username) references User (username),
        foreign key (lecture-no) references Lecture (lecture-no)
)
engine=InnoDB;
```

Quest-answ

Relational Model: Quest-answ (answer-no, question-no)
Candidate Key:
answer-no
Primary Key: answer-no
Foreign Key:
answer-no referencing Post (postno)
question-no referencing Post (postno)
Table Definition:

```
create table Quest-answ (
        answer-no int,
        question-no int,
        primary key ( answer-no ),
        foreign key (answer-no) references Post (postno),
        foreign key (question-no) references Post (postno)
)
engine=InnoDB;
```

Advertisement

Relational Model: Advertisement (advertisementno, ad-username, cno, advertisement, status, payment, startdate, finishdate)
Candidate Key:
advertisementno

Primary Key: advertisementno
Foreign Key:
 ad-username referencing Advertiser (username)
 cno referencing Course
Table Definition:

```
create table Advertisement (
        advertisementno int,
        ad-username char (50),
        cno int,
        advertisement varchar(512),
        status smallint,
        payment numeric(20,2),
        startdate Date,
        finishdate Date,
        primary key (advertisementno),
        foreign key (ad-username) references Advertiser (username),
        foreign key (cno) references Course (cno)
)
engine=InnoDB;
```

RefundRequest

Relational Model: RefundRequest (<u>refund-id</u>, s-username, cno, reason, status)
Candidate Key:
 refund-id
Primary Key: refund-id, s-username, cno
Foreign Key:
 s-username referencing Student (username)
 cno referencing Course
Table Definition:

```
create table RefundRequest (
        refund-id int,
        s-username char (50),
        cno int,
        reason char (500),
        status smallint default 0,
        primary key ( refund-id),
        foreign key (s-username) references Student (username),
        foreign key (cno) references Course (cno)
)
engine=InnoDB;
```

Evaluates

Relational Model: Evaluates (<u>refund-id</u>, admin-username, reply-date)
Candidate Key:
 refund-id
Primary Key: refund-id

Foreign Key:
	admin-username referencing SiteAdmin
	refund-id referencing RefundRequest
Table Definition:
```
create table Evaluates (
	refund-id int,
	admin-username char(50),
	reply-date Date,
	primary key (refund-id),
	foreign key ( admin-username) references SiteAdmin ( username ),
	foreign key ( refund-id ) references RefundRequest ( refund-id )
)
engine=InnoDB;
```

Assignment

Relational Model: Assignment (<u>assignmentno</u>, assignment, lecture-no)
Candidate Key:
	assignmentno
Primary Key: assignmentno
Foreign Key:
	lecture-no referencing Lecture
Table Definition:
```
create table Assignment(
	assignmentno int,
	assignment longblob,
	lecture-no int,
	primary key (assignmentno),
	foreign key (lecture-no) references Lecture (lecture-no)
)
engine=InnoDB;
```

LectureMaterial

Relational Model: LectureMaterial (<u>materialno</u>, material, lecture-no)
Candidate Key:
	materialno
Primary Key: materialno
Foreign Key: lecture-no referencing Lecture
Table Definition:
```
create table LectureMaterial (
	materialno int,
	material longblob,
	lecture-no int,
	primary key ( materialno ),
	foreign key ( lecture-no ) references Lecture (lecture-no)
)
engine=InnoDB;
```

Inside-cart

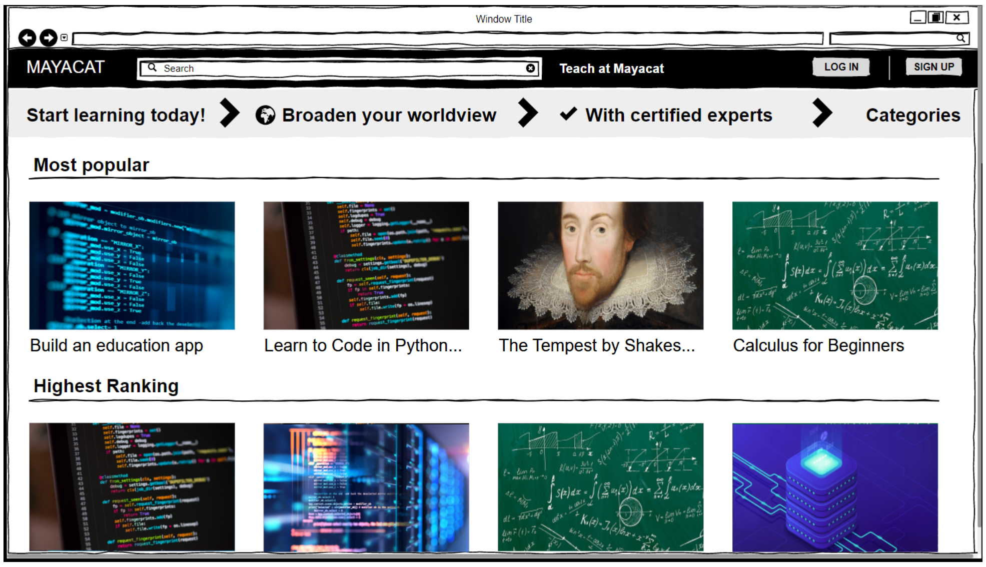
Relational Model: Inside-cart (cno, username, receiver-username)
Candidate Key:
    cno, username
Primary Key: cno, username
Foreign Key:
    cno referencing Course
    cart-id referencing Cart
    receiver-username referencing Student (username)
Table Definition:

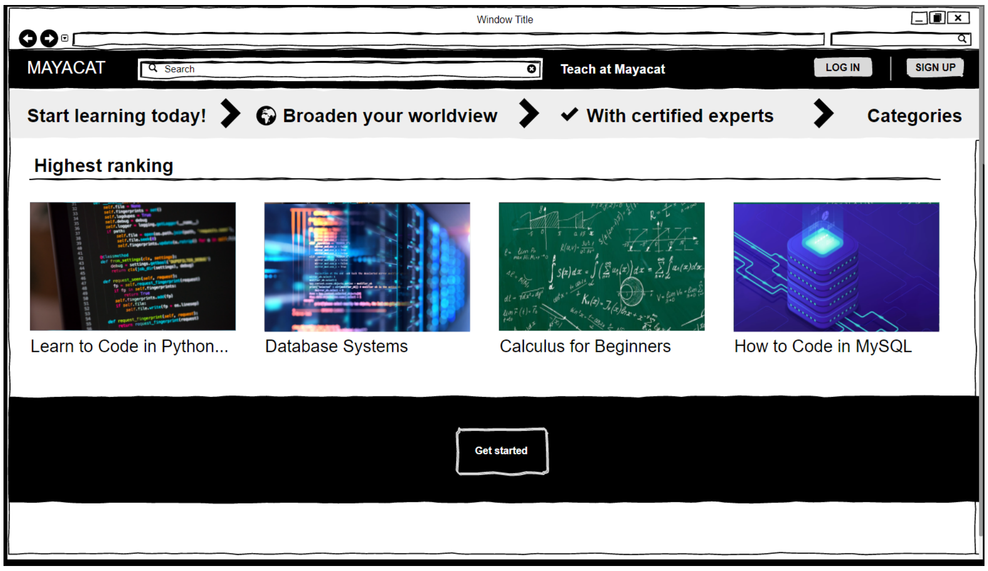
```
create table Inside-cart (
	cno int,
	username char (50),
	receiver-username char (50),
	primary key (cno, username),
	foreign key (cno) references Course (cno),
	foreign key (username) references User (username),
	foreign key (receiver-username) references Student (username)
)
engine=InnoDB;
```

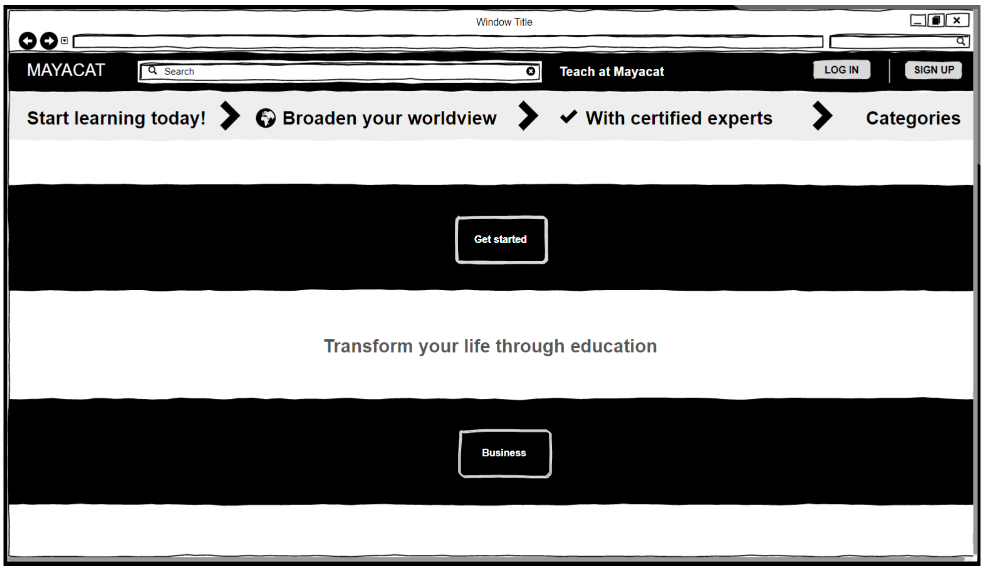
III. Interface Designs and Corresponding SQL Statements

As some of the mockups were not obligatory and, in our report, is there mostly for reference purposes, we did not write the queries for those. All the other necessitated designs are placed alongside their queries.

Main Page Before Login (Scroll Down)







SQL Statement:

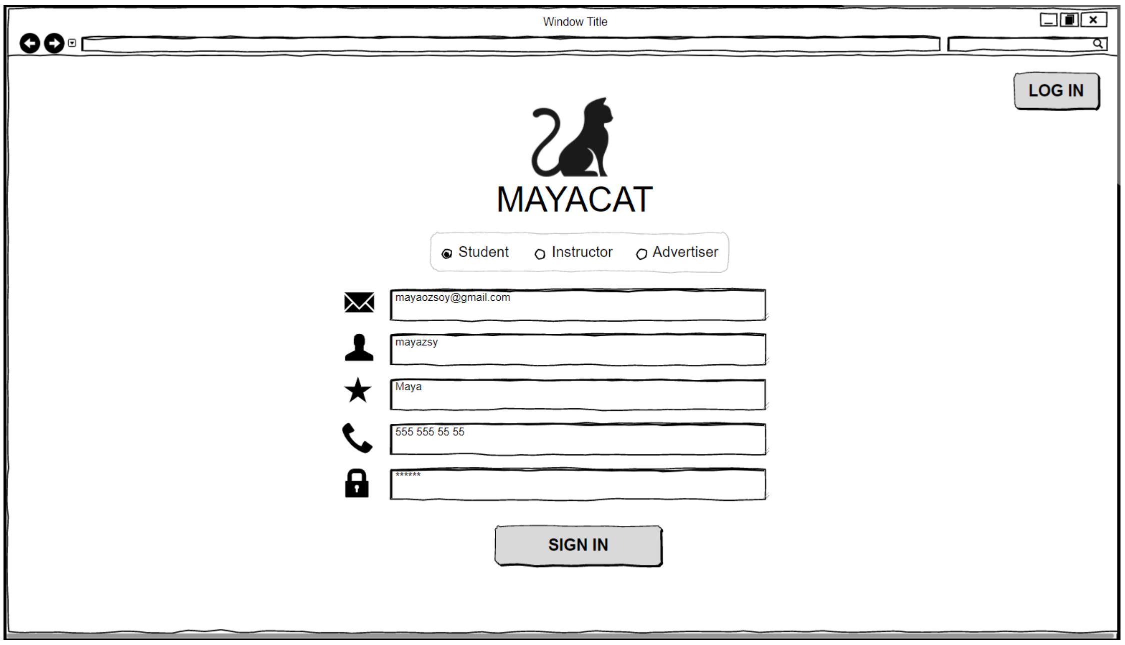
- Used to demonstrate the highest ranked courses.

```
select c.cno, c.cname
from Course as c
natural join (
        select r.cno
        from Rate as r
        group by r.cno
        order by avg(r.score)) as t
limit 4;
```

- Used to demonstrate the most popular courses.

```
select c.cno, c.cname
from Course as c
natural join (
        select e.cno
        from Enroll as e
        group by e.cno
        order by count(e.s-username)) as t
limit 4;
```

Student Signup Page



[✓ Student] Instructor Advertiser

mayaozsoy@gmail.com

mayazsy

Maya

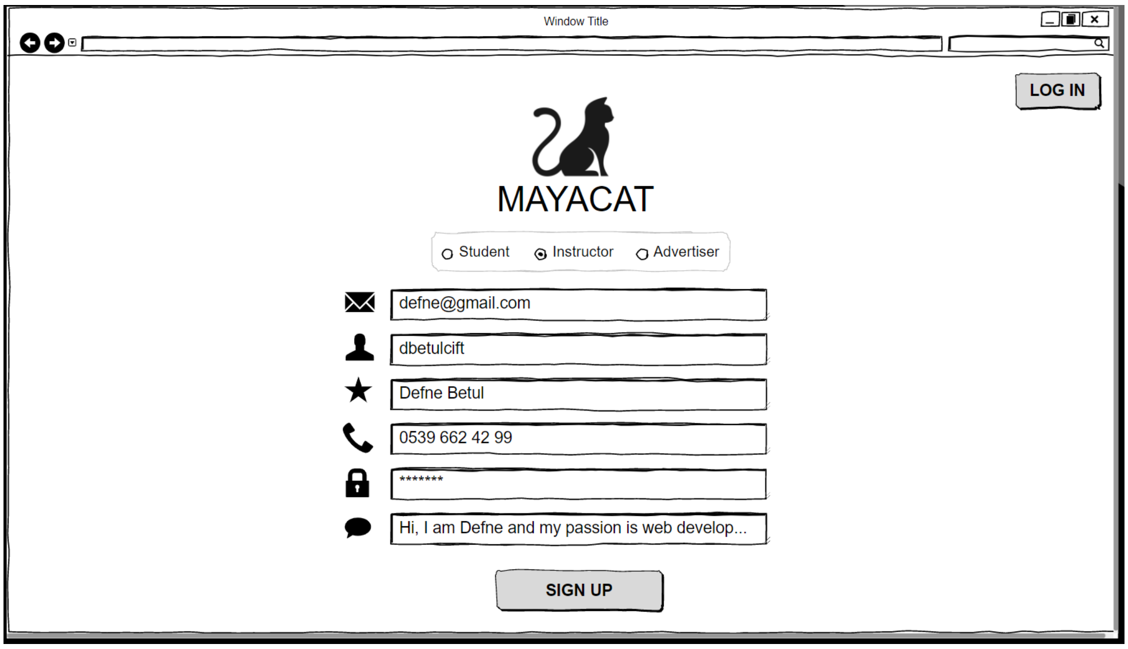
555 555 55 55

******

SQL Statement:

insert into User values (“mayazsy”, “Maya”, “123456”, “mayaozsoy@gmail.com”, “05555555555”);
insert into Student (“mayazsy”);

Instructor Signup



defne@gmail.com

dbetulcift

Defne Betul

0539 662 42 99

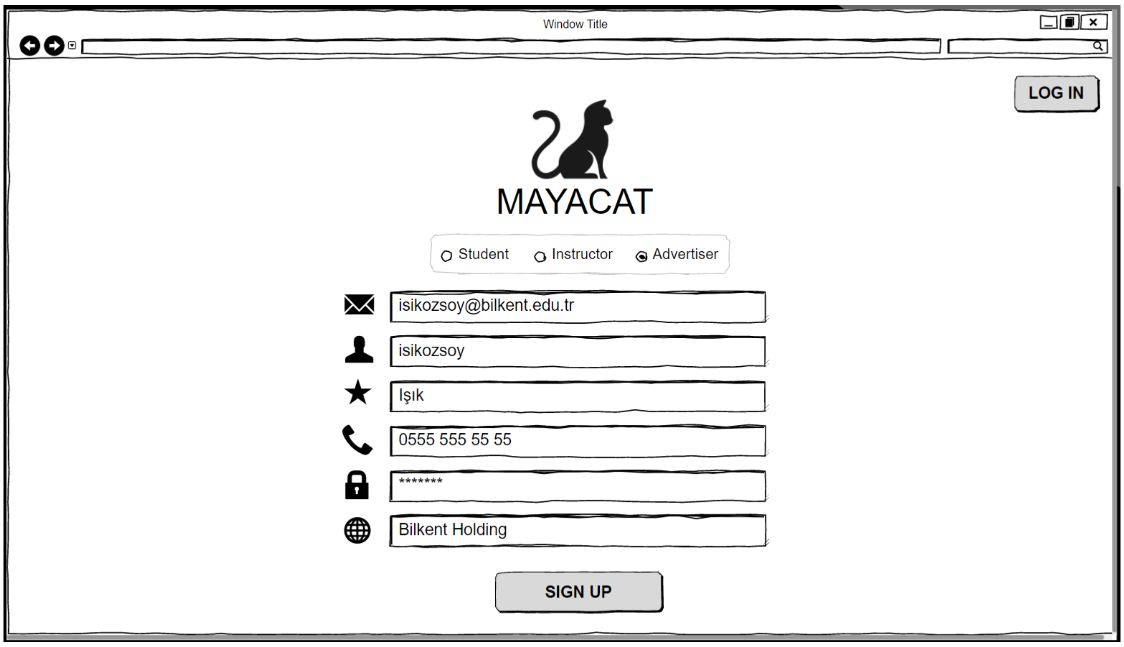
****

Hi, I am Defne and my passion is web develop...

SQL Statement:

insert into User values (“dbetulcift”, “Defne Betul”, “Db1234”, “defne@gmail.com”, “05396624299”);
insert into Instructor values (“dbetulcift”, “Hi, I am Defne and my passion is web development. I have been teaching it for 15 years.”);

Advertiser Signup



isikozsoy@bilkent.edu.tr

isikozsoy

Işık

0555 555 55 55

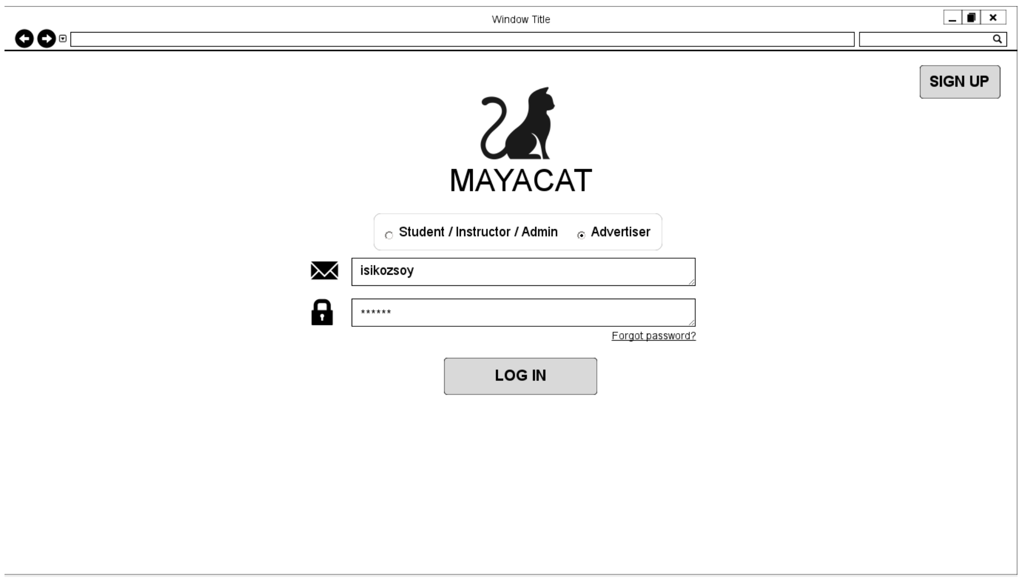
****

Bilkent Holding

SQL Statement:

insert into Advertiser values (“isikozsoy”, “Isik”, “mm19kk”, “isikozsoy@bilkent.com”, “05459554545”, “Bilkent Holding”);

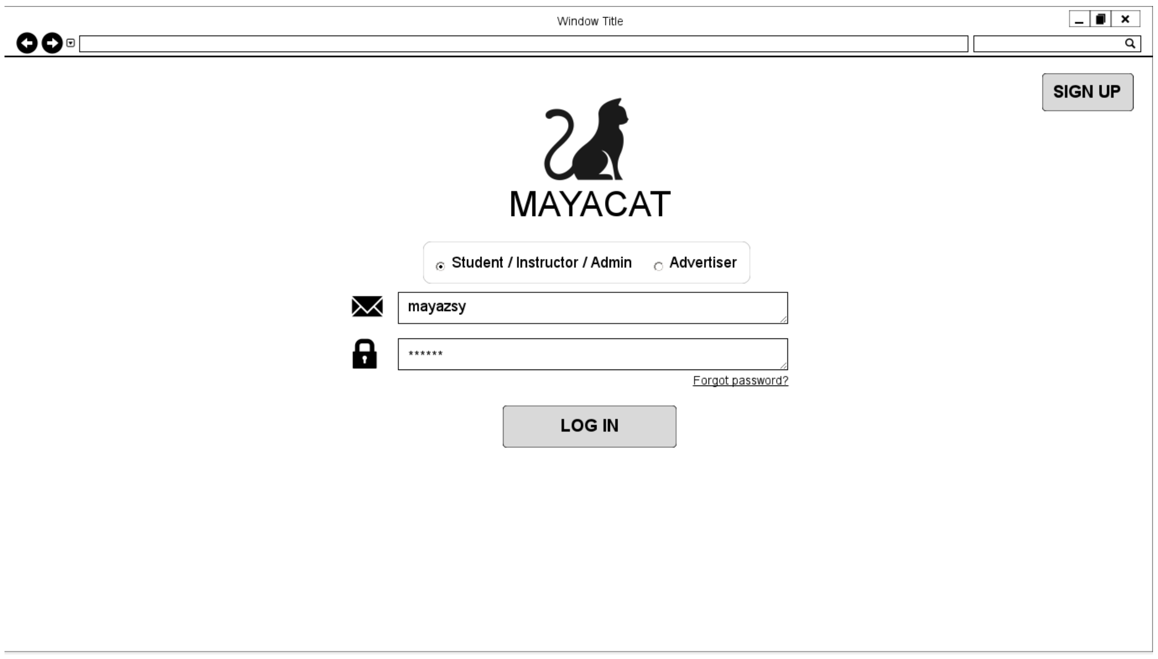
Login Page for Advertiser



SQL Statement:

select password
from Advertiser
where ad-username =”isikozsoy”;

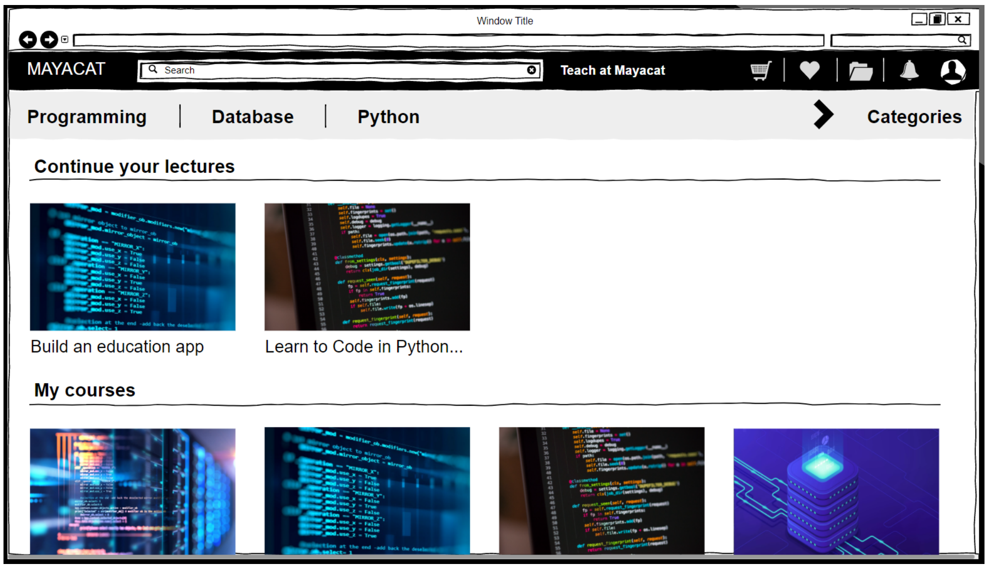
Login Page for Other User Types



SQL Statement:

select password
from User
where username =”mayazsy”;

Main Page of Student After Login (Scroll Down)



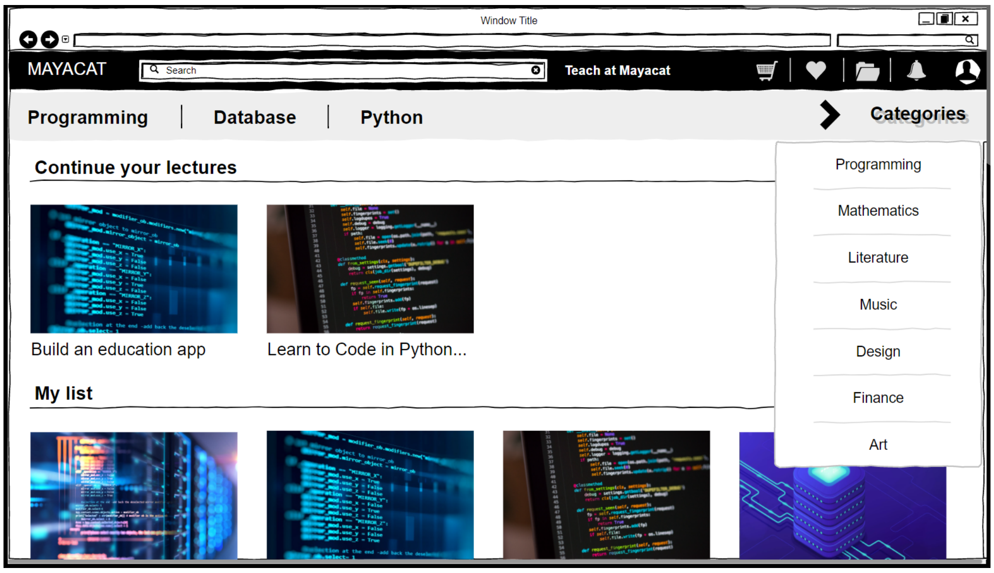
SQL Statement:

- Used to demonstrate the courses that are not finished yet.

select cno, cname, course-img
from Enroll
where s-username="mayazsy" and cno not in (
select cno
from Finishes
where s-username="mayazsy");

- Used to demonstrate all the courses of the student.

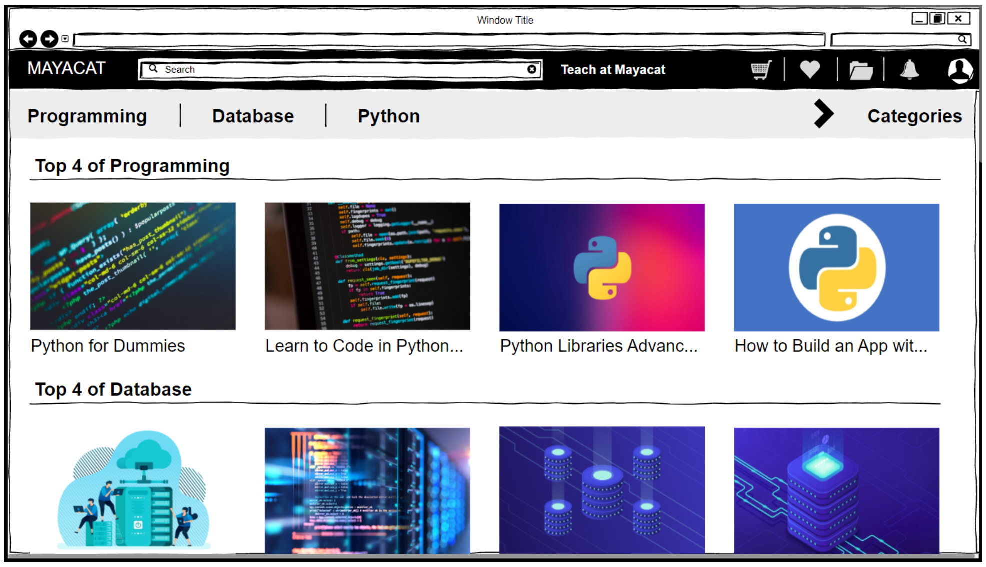
select cno, cname, course-img
from Enroll
where s-username="mayazsy";



- Categories show all the topics in the platform. At the top of the page, the topics "Programming", "Database" and "Python" are selected.

SQL Statement:

select distinct topicname
from Topic;

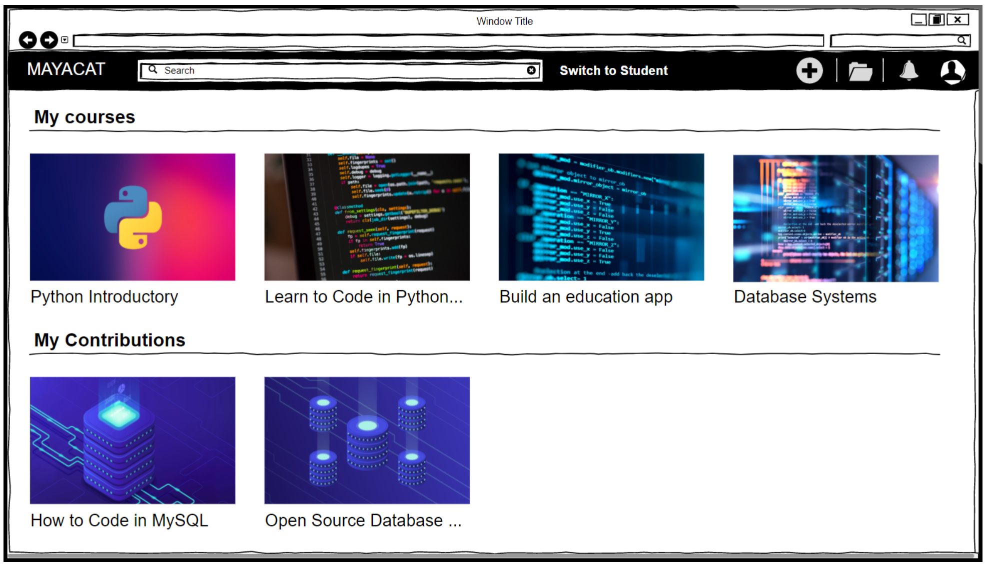


- When the student scroll downs the page, s/he sees the courses, which belong to the selected topics, separately.

SQL Statement:

```
with specified-courses (cno) as
   (select cno
    from Course-topic
    where topicname = "Programming")
select cno, cname, course-img
from (select cno, sum (score) as tot-rate
        from specified-courses natural join Rate
        group by cno) natural join Course;

with specified-courses (cno) as
   (select cno
    from Course-topic
    where topicname = "Database")
select cno, cname, course-img
from (select cno, sum (score) as tot-rate
        from specified-courses natural join Rate
        group by cno) natural join Course;

with specified-courses (cno) as
   (select cno
    from Course-topic
    where topicname = "Python")
select cno, cname, course-img
from (select cno, sum (score) as tot-rate
         from specified-courses natural join Rate
         group by cno) natural join Course;
```

Main Page of Instructor After Login



My courses

Python Introductory

Learn to Code in Python...

Build an education app

Database Systems

My Contributions

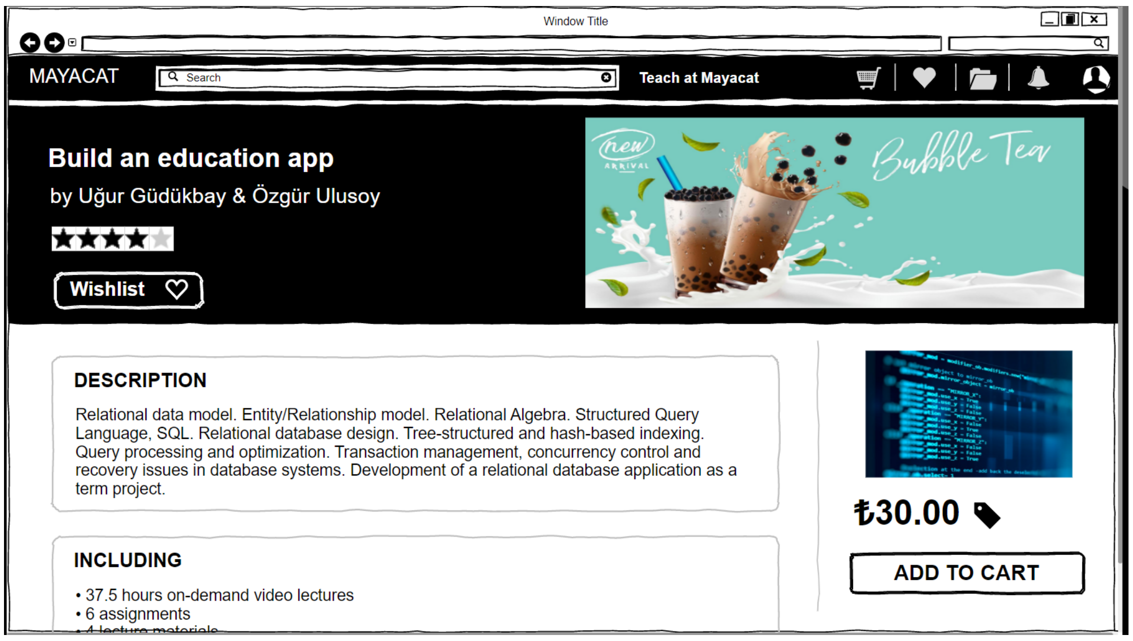
How to Code in MySQL

Open Source Database ...

SQL Statement:

```
select cno, cname, course-img
from Course
where owner-username = "dbetulcift";

select cno, cname
from Course natural join Contributor
where i-username = "dbetulcift";
```

Course Info Page



Build an education app

by Uğur Güdükbay & Özgür Ulusoy

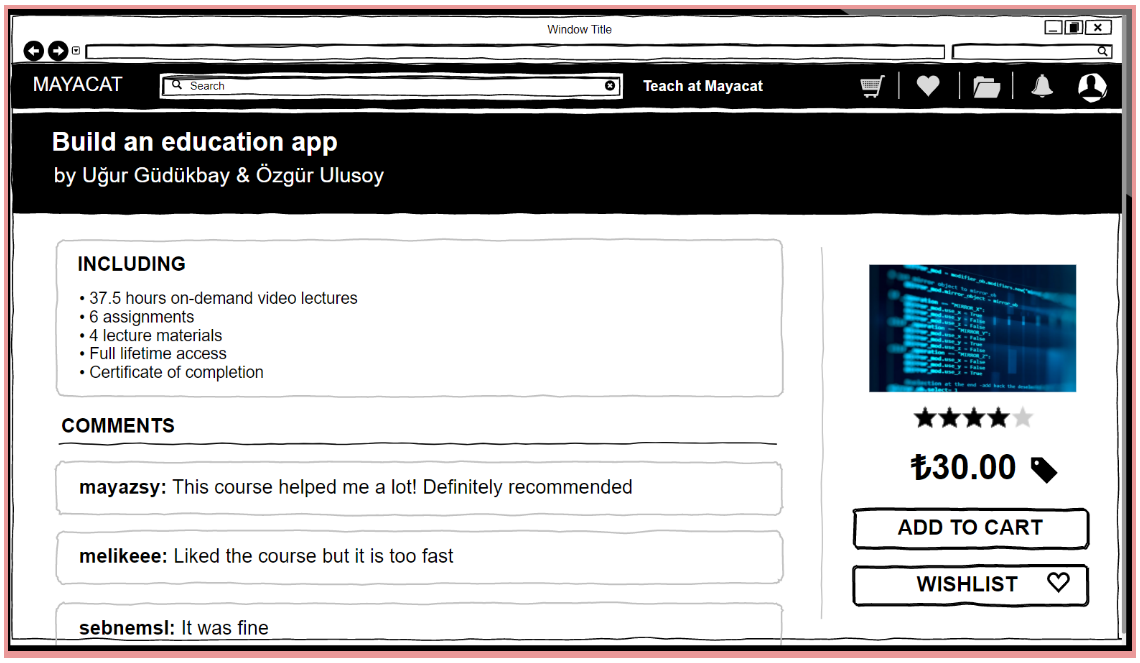
DESCRIPTION

Relational data model. Entity/Relationship model. Relational Algebra. Structured Query Language, SQL. Relational database design. Tree-structured and hash-based indexing. Query processing and optimization. Transaction management, concurrency control and recovery issues in database systems. Development of a relational database application as a term project.

INCLUDING

- 37.5 hours on-demand video lectures
- 6 assignments

₺30.00



Build an education app

by Uğur Güdükbay & Özgür Ulusoy

INCLUDING

- 37.5 hours on-demand video lectures
- 6 assignments
- 4 lecture materials
- Full lifetime access
- Certificate of completion

COMMENTS

mayazsy: This course helped me a lot! Definitely recommended

melikeee: Liked the course but it is too fast

sebnemsl: It was fine

₺30.00

SQL Statement:

- Assume that the cno of the course named Build an education app is 1.

select cname, owner-username, price, description, name
from Course, User
when cno = 1;

select i-username, name
from Contributor, User
where cno = 1 and i-username = ‘mayazsy’;

```
--To demonstrate the comments made by the student who has finished the course.
select comment
from Finishes
where cno=15;

select cno, avg(score) as avg-rate
from Rate
where cno = 1;
```

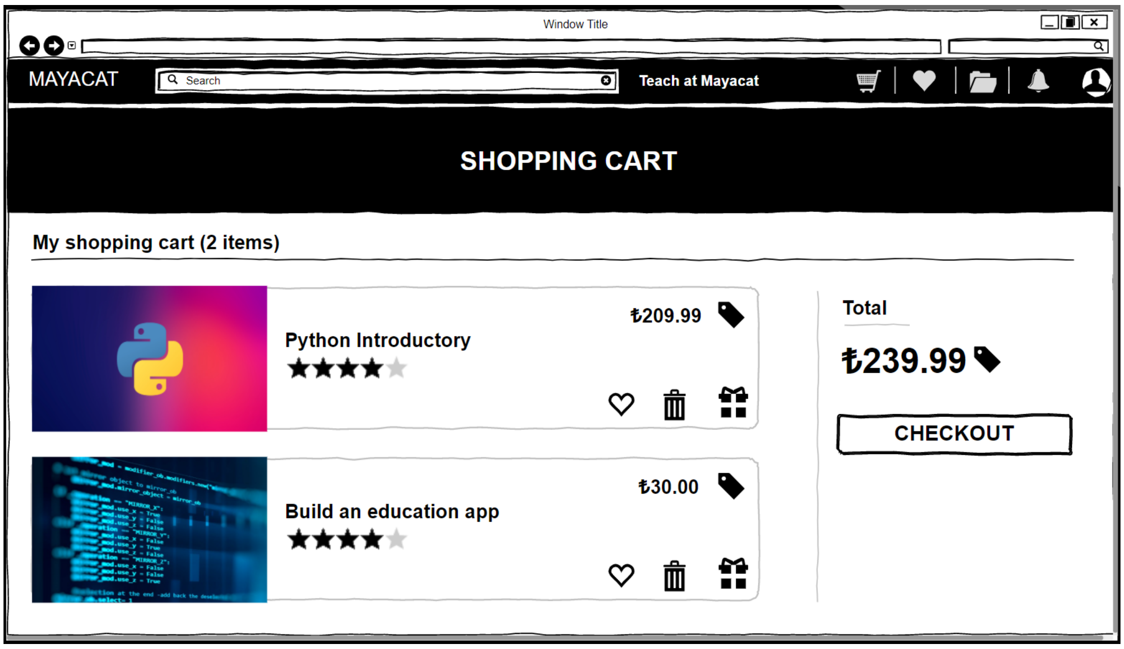
- If the student adds the item to the cart, the following query will be used (assume that the course no is 9). Null represents the username of the student that will receive the course as a gift.

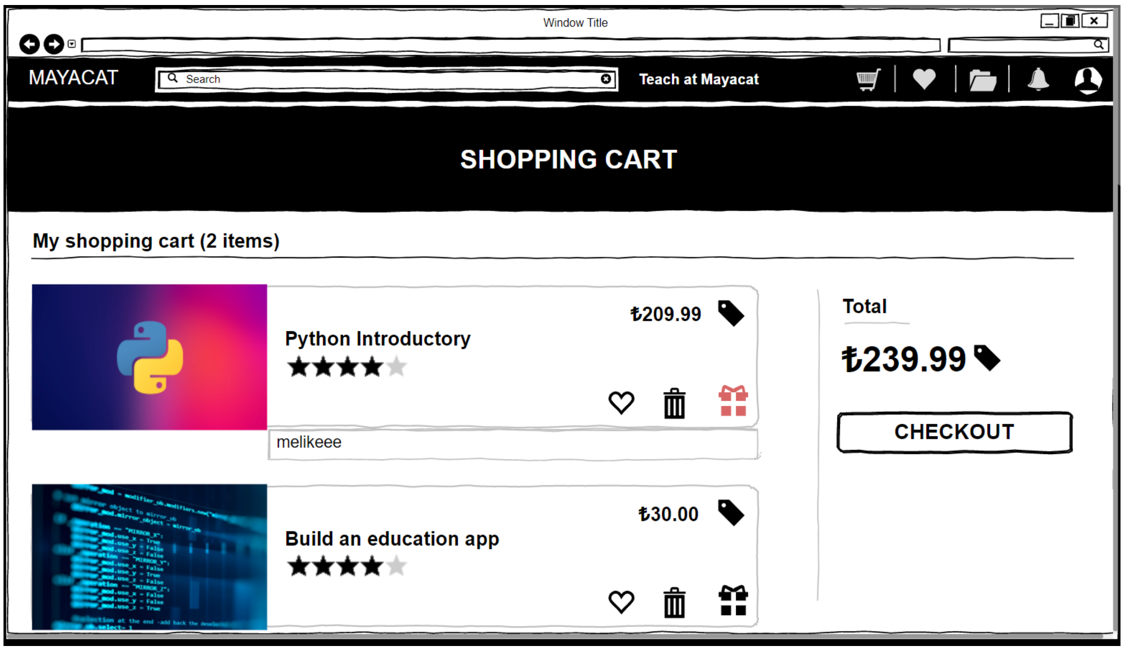
```
insert into Inside-cart values (9, 'mayaozsy', null);

select advertisementno, advertisement
from Advertisement
where cno = 1 and curdate() between startdate and finishdate and status = 1;
```

- If the student adds the course into wishlist by clicking on the heart icon;

```
insert into Wishes values ('mayazsy', 9);
```

Shopping Cart Page and with Gift Property Page





SHOPPING CART

My shopping cart (2 items)

Python Introductory ₺209.99

melikeee

Build an education app ₺30.00

Total

₺239.99

CHECKOUT

SQL Statement:

- When the student enters the shopping cart page,

select cno, cname, price, receiver-email
from Course as C, Inside-cart as I
where C.cno = I.cno and I.username = "mayazsy";

select cno, avg(score)
from Rate
where cno in (select cno
                from In-cart as I
                where I.username = "mayazsy")
group by cno;

- In the mockups above, the user selected the gift option for the first course in the cart, then provided the username of the receiver. Assume that the cno of the first course is 5 and the second course is 15.

update Inside-cart
set recevier-username = 'melikeee'
where cno = 9 and username = 'mayazsy';

- Considering the second page, if the student buys the courses;

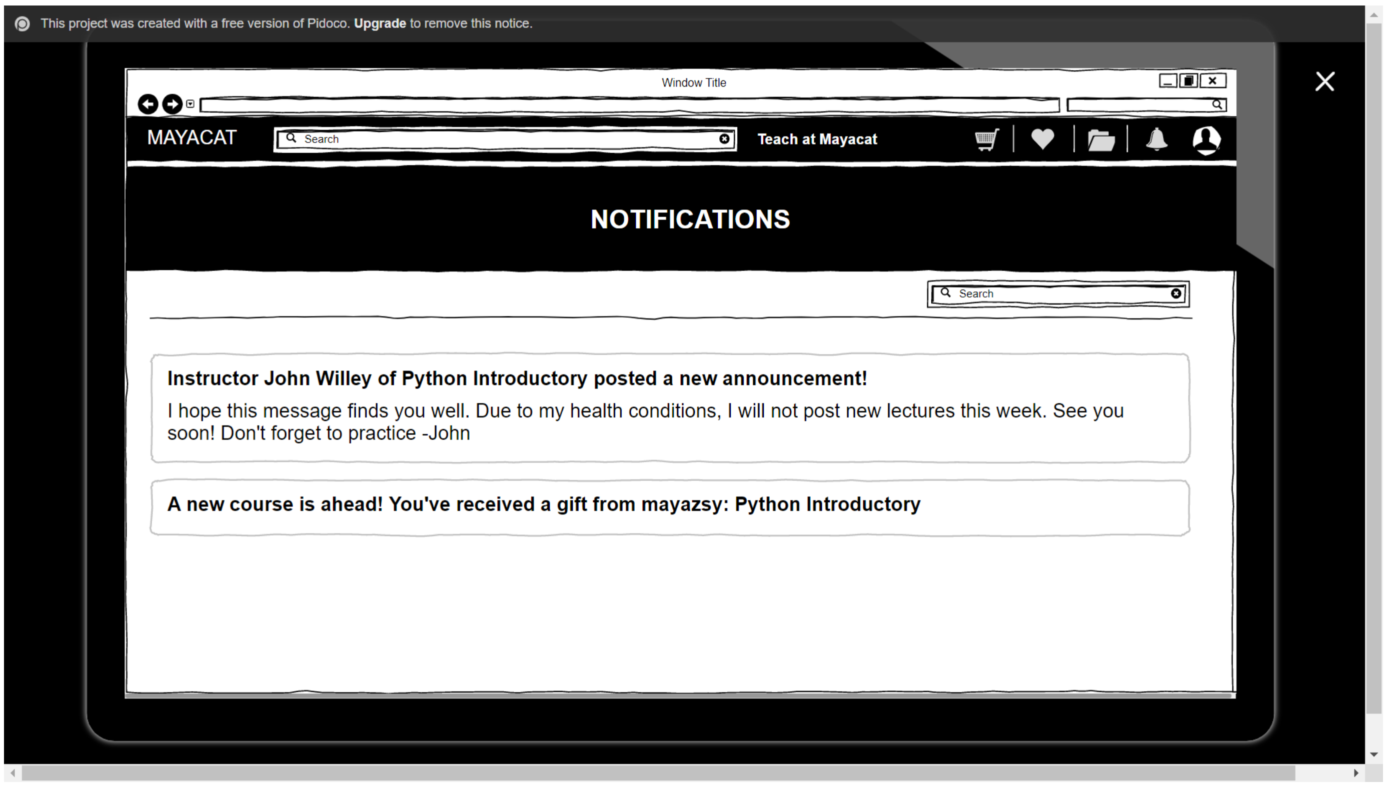
--for the course that is a gift
insert into Gift ('mayazsy', 'melikeee', 9);
insert into Enroll ('melikeee', 9);

--for the course that is not a gift
insert into Enroll values("mayazsy", 15);

- If the student selects remove option for the second course;

delete from Inside-cart
wherecno = 15 and username = ‘mayazsy’;

Notifications


This project was created with a free version of Pidoco. Upgrade to remove this notice.
Window Title
MAYACAT
Search
Teach at Mayacat
NOTIFICATIONS
Search

Instructor John Willey of Python Introductory posted a new announcement!

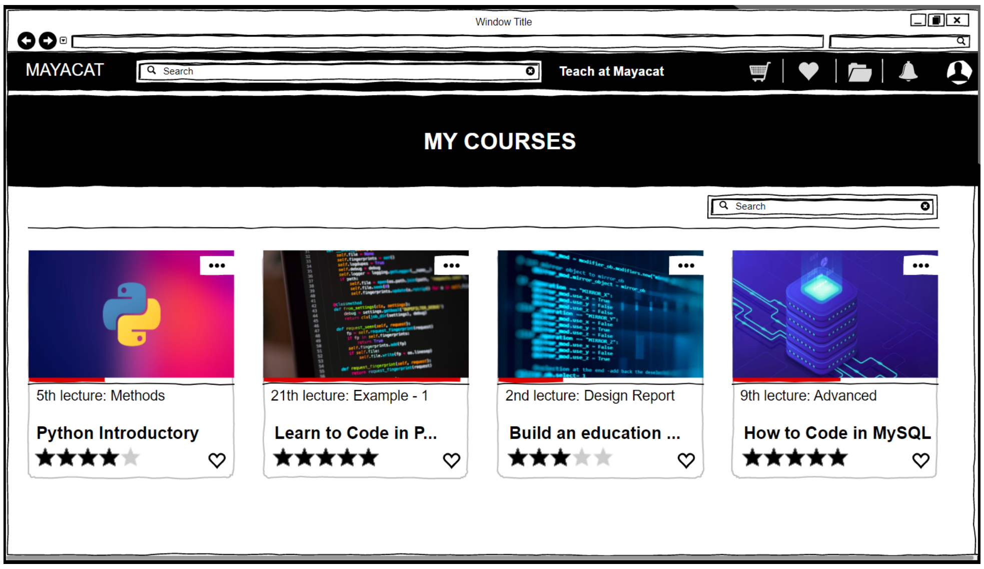
I hope this message finds you well. Due to my health conditions, I will not post new lectures this week. See you soon! Don't forget to practice -John

A new course is ahead! You've received a gift from mayazsy: Python Introductory

SQL Statement:

-- for listing the announcements of enrolled courses
select ann-tex, cname, ann-id, ann-date, owner-username
from Course, Announce, Enroll
where Enroll.s-username = “mayazsy” and Enroll.cno = Course.cno and Announce.cno = Course.cno;
-- for the list of received gifts
select g.sender-username, c.cname
from Gift as g, Course as c
where g.cno = c.cno;

Student’s Courses Page



SQL Statement:

select cno, cname, course-img
from Course, Enroll
where Course.cno = Enroll.cno and Enroll.s-username = “mayazsy”;

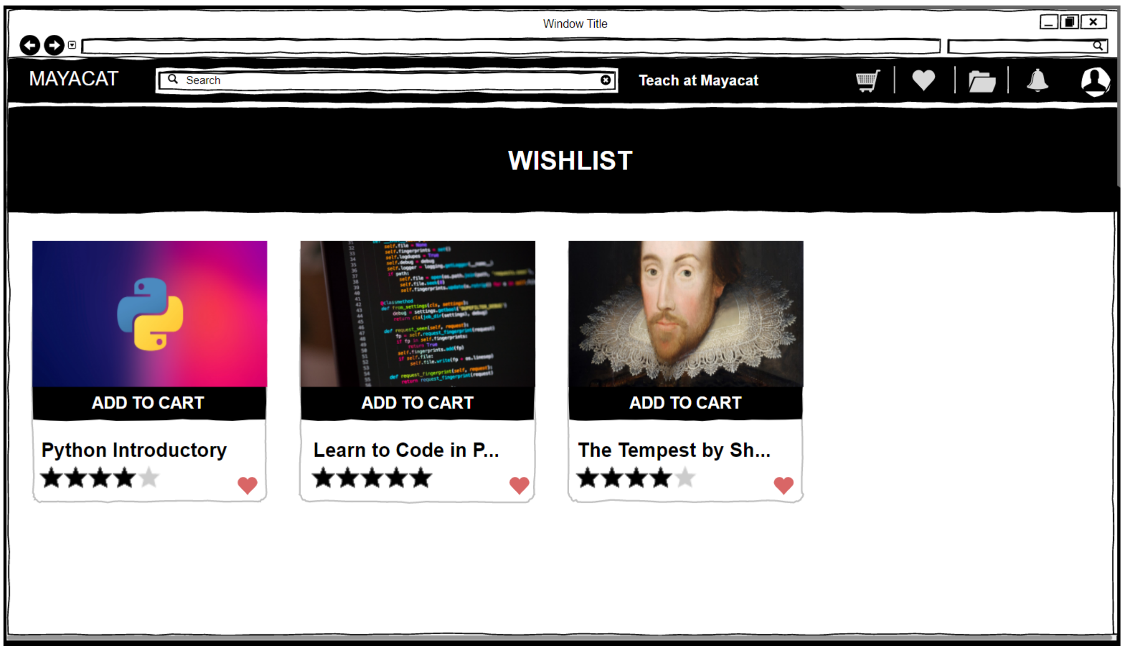
select cno,avg(score)
from Rate
where cno in (select cno
                    from Enroll as E
                    where E.s-username = “mayazsy”)
group by cno;

- The red lines behind the course images demonstrate the general progress for the course. In order to find the progress rate, the number of the completed lectures will be divided into the total number of the lectures of the given course. Assume that the cno of Python Introductory course is 5.

--The number of lectures of the Python Introductory course is found.
select count (lecture-no) as tot-lec-count
from Lecture
where cno = 5;

-- The number of completed lectures of the Python Introductory course is found.
select count (lecture-no) as comp-lec-count
from Lecture as L, Progress as P
where L.cno = 5 and L.cno = P.cno and username=’mayazsy’;

Wishlist Page



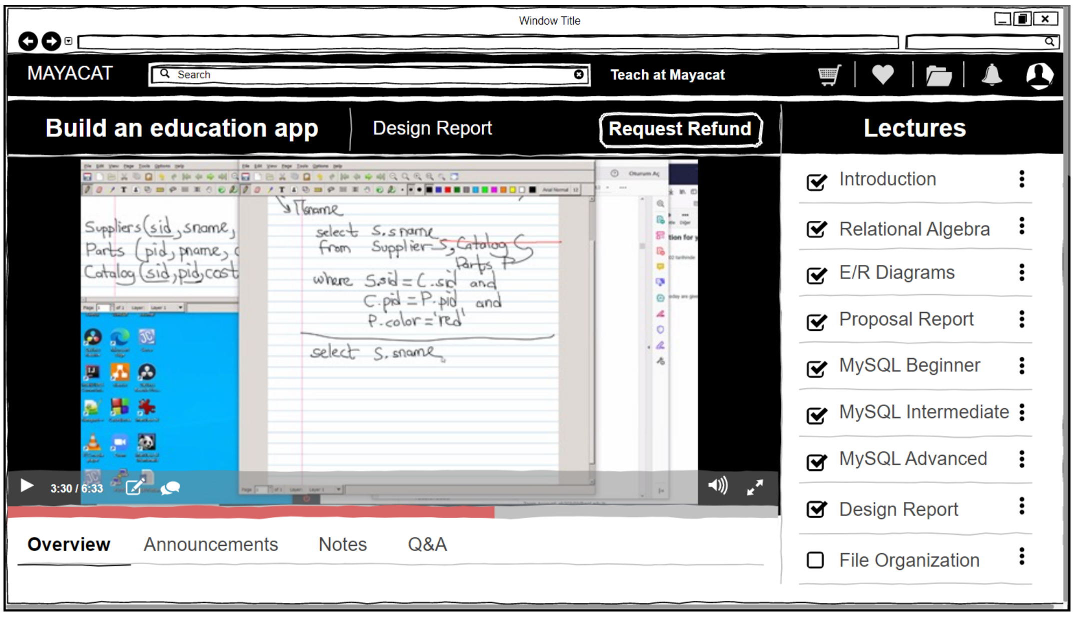
SQL Statement:

-- listing the courses inside the wishlist
select cno, cname, course-img
from Course, Wishes
where Wishes.cno = Course.cno and Wishes.s-username = “mayazsy”;

- If the student adds the course called The Tempest by Shekspare to cart (Assume that the cno is 22);

insert into Inside-cart values (22, ‘mayazsy’, null);

Watching Lecture Page



SQL Statement:

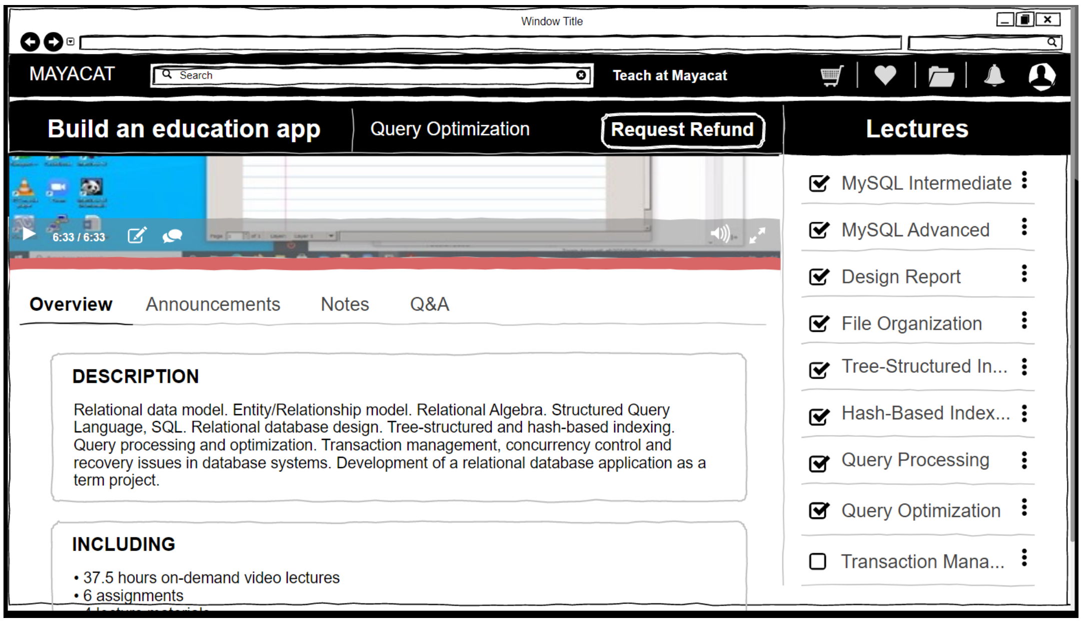
- It was assumed that the cno of Build an education app is 15 and the lecture no of the current lecture (Design Report) is 1432.

select lecture-no, lecture-name, video
from Lecture as L
where L.cno = 15;

- When the lecture is opened, it is added into the Progress table and considered as completed.

insert into Progress (“mayazsy”, 1432);

Course Overview Page



DESCRIPTION

Relational data model. Entity/Relationship model. Relational Algebra. Structured Query Language, SQL. Relational database design. Tree-structured and hash-based indexing. Query processing and optimization. Transaction management, concurrency control and recovery issues in database systems. Development of a relational database application as a term project.

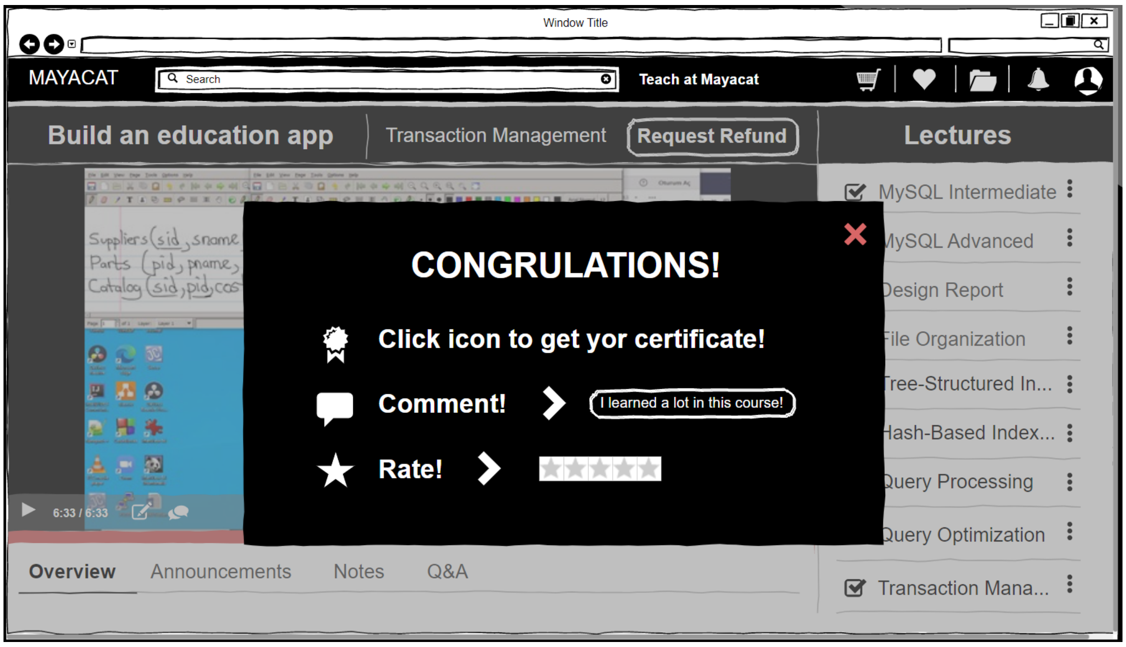
INCLUDING

• 37.5 hours on-demand video lectures
• 6 assignments

SQL Statement:

-- list course contents
select cno, description, including, owner-username
from Course
where cno = 15;

Completing a Course Page



SQL Statement:

- Assume that the cno of the Build an education app is 15

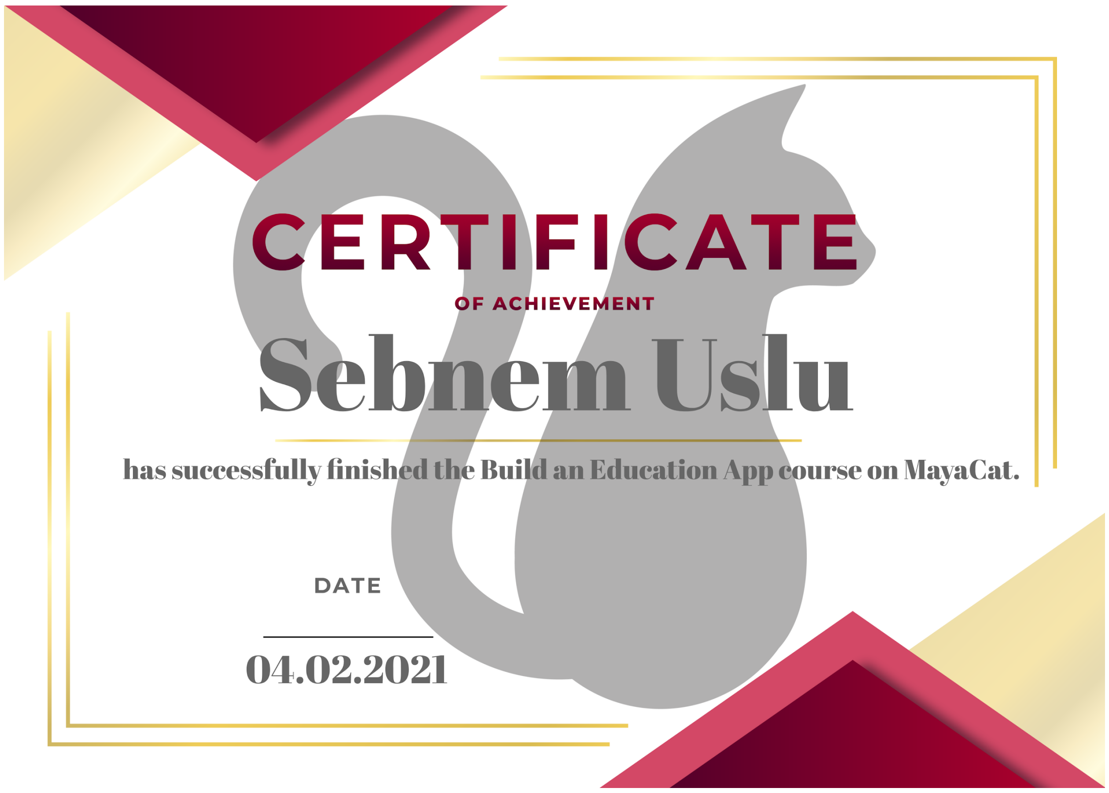
select count(lecture-no) as finished-lec-cnt
from Progress natural join Lecture
where s-username = “mayazsy” and cno = 15;

select count(lecture-no) as lecture-cnt
from Lecture
where cno = 15;

- If finished-lec-cnt == lecture-cnt;

insert into Finishes values (“mayazsy”, lecture-no, “I learned a lot in this course!”);

select cno, cname, s-username, comment
from Finishes natural join Course
where s-username = “mayazsy” and cno = 15;


CERTIFICATE

OF ACHIEVEMENT

Sebnem Uslu

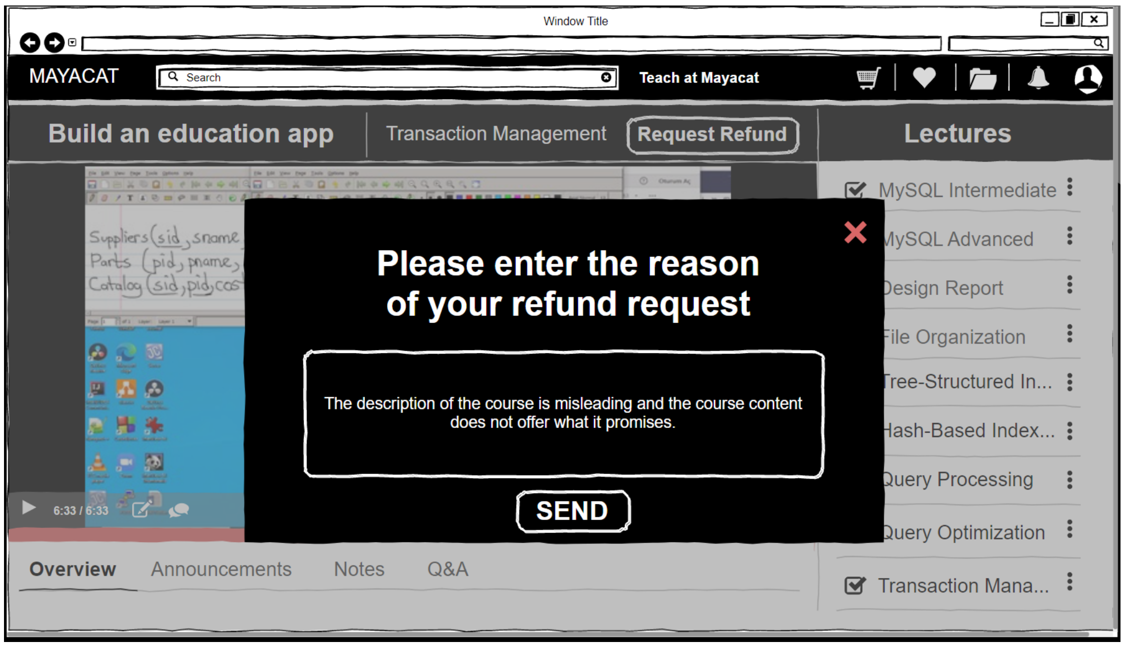
has successfully finished the Build an Education App course on MayaCat.

DATE

04.02.2021

- A mockup of the certificate pdf template can be seen above.

Course Refund Request



Please enter the reason of your refund request

The description of the course is misleading and the course content does not offer what it promises.

SQL Statement:

- The student requested a refund for the course called Build and education app (cno: 15)

insert into RefundRequest values (512, “mayazsy”, 15, ‘The description of the course is misleading and the course content does not offer what it promises’, 0);
--it automatically enters to the database as 0, meaning not evaluated yet

- If an admin whose username is marythead selects and evaluates the request as approved;

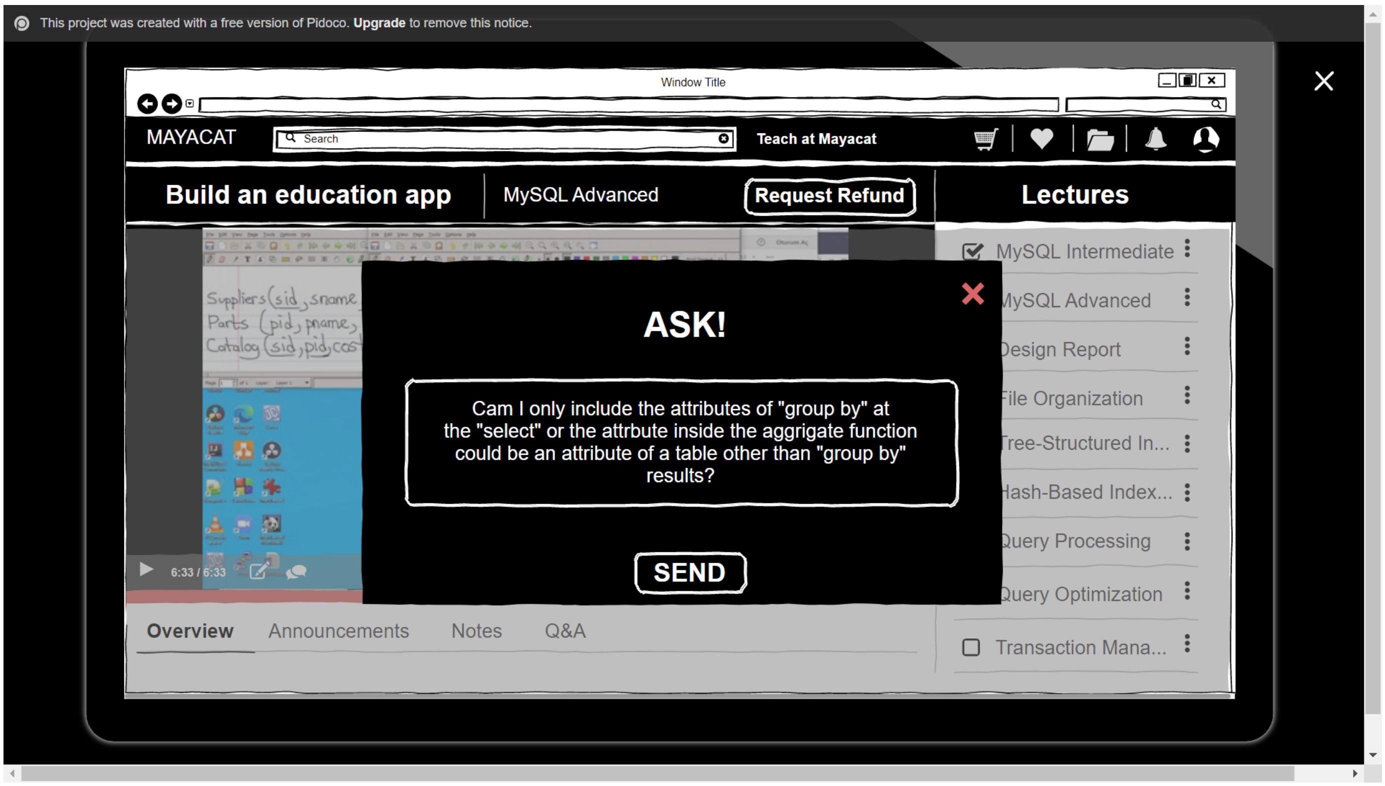
insert into Evaluates values (512, ‘marythead’, ‘02/04/2020’);

update RefundRequest
set status = 1
where refund-id = 512;

- If an admin whose username is marythead selects and evaluates the request as rejected;

select 'marythead', refund_id, curdate()
from RefundRequest
where status = -1;

Course Ask Question



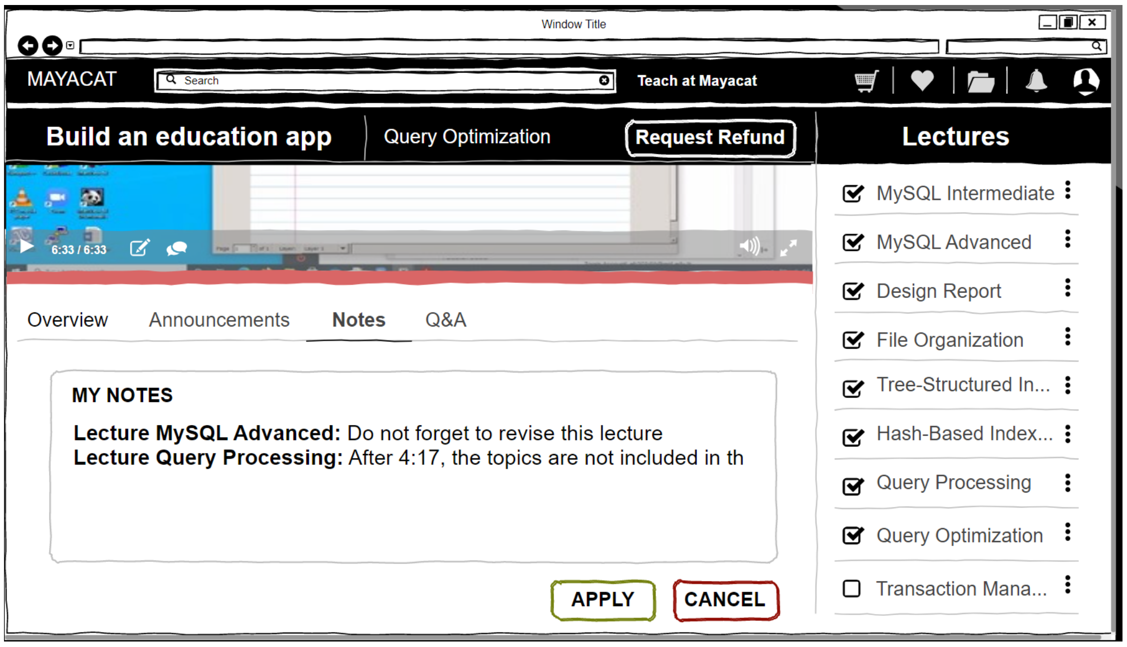
Cam I only include the attributes of "group by" at the "select" or the attrbute inside the aggrigate function could be an attribute of a table other than "group by" results?

SQL Statement:

-- 1028 is the id of the post, while 15 is the id of the course the post is inside of
-- inserts the comment seen above as a discussion post to the forum of the course

insert into Post values (1028, 15, ‘Can I only include the attributes of “group by” at the “select” or the attribute inside the aggregate function could be an attribute of a table other than “group by” results?’, “mayazsy”);

Note Page



MY NOTES

Lecture MySQL Advanced: Do not forget to revise this lecture
Lecture Query Processing: After 4:17, the topics are not included in th

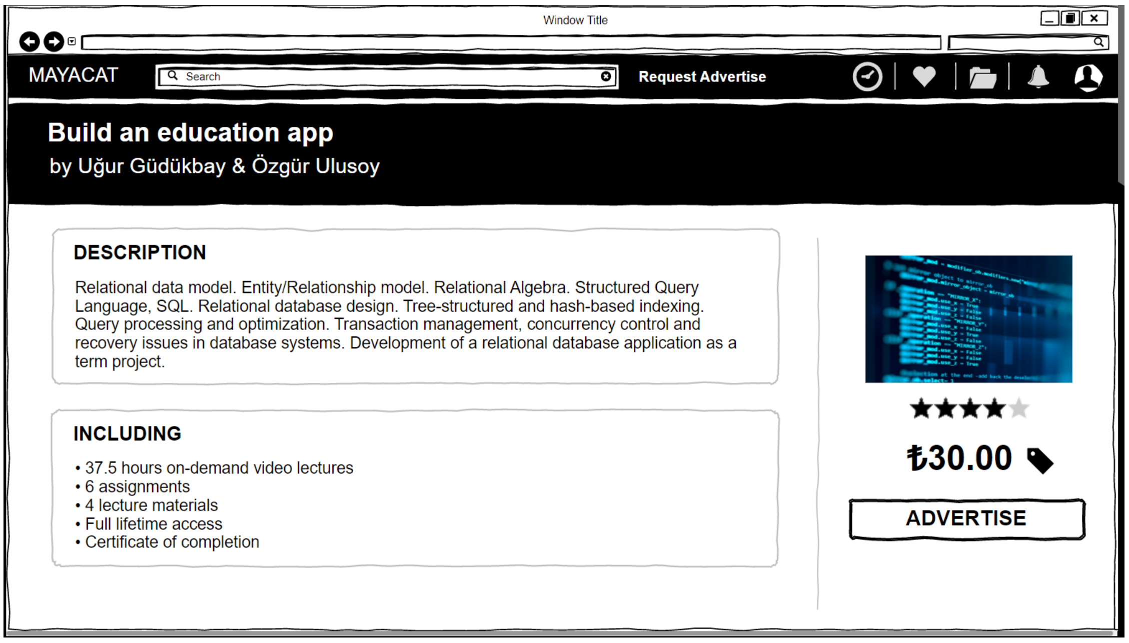
- ✓ MySQL Intermediate
- ✓ MySQL Advanced
- ✓ Design Report
- ✓ File Organization
- ✓ Tree-Structured In...
- ✓ Hash-Based Index...
- ✓ Query Processing
- ✓ Query Optimization
- ☐ Transaction Mana...

SQL Statement:

- A student can take a note for the current lecture.

insert into Takes-note (“mayazsy”, 14321, “Do not forget to revise this lecture”);
insert into Takes-note (“mayazsy”, 10283, “After 4:17, the topics are not included in the recommended textbooks.”);

Course Page for Advertisers - Additional Functionality



Build an education app
by Uğur Güdükbay & Özgür Ulusoy

DESCRIPTION

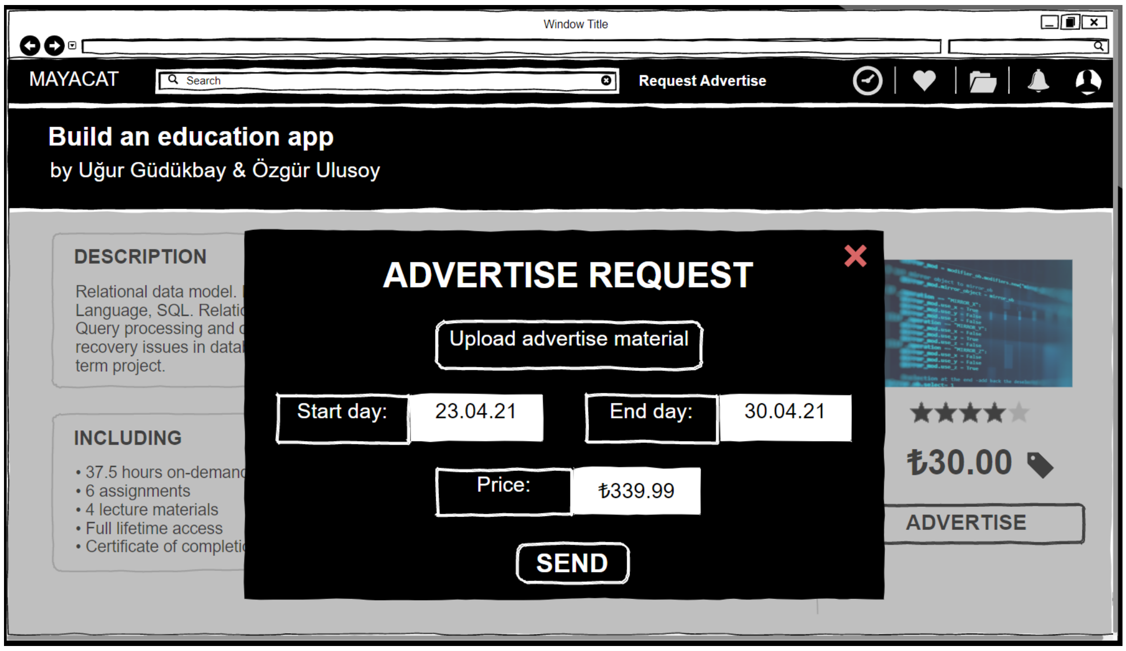
Relational data model. Entity/Relationship model. Relational Algebra. Structured Query Language, SQL. Relational database design. Tree-structured and hash-based indexing. Query processing and optimization. Transaction management, concurrency control and recovery issues in database systems. Development of a relational database application as a term project.

INCLUDING

- 37.5 hours on-demand video lectures
- 6 assignments
- 4 lecture materials
- Full lifetime access
- Certificate of completion

-- list course contents (assume that cno is 15)
select cno, description, including, owner-username
from Course
where cno = 15;

Advertise Page for Advertisers - Additional Functionality



Build an education app
by Uğur Güdükbay & Özgür Ulusoy

DESCRIPTION
Relational data model. Language, SQL. Relatic Query processing and c recovery issues in datal term project.

INCLUDING
- 37.5 hours on-demand
- 6 assignments
- 4 lecture materials
- Full lifetime access
- Certificate of completi

ADVERTISE REQUEST
Upload advertise material
Start day: 23.04.21
End day: 30.04.21
Price: ₺339.99
SEND

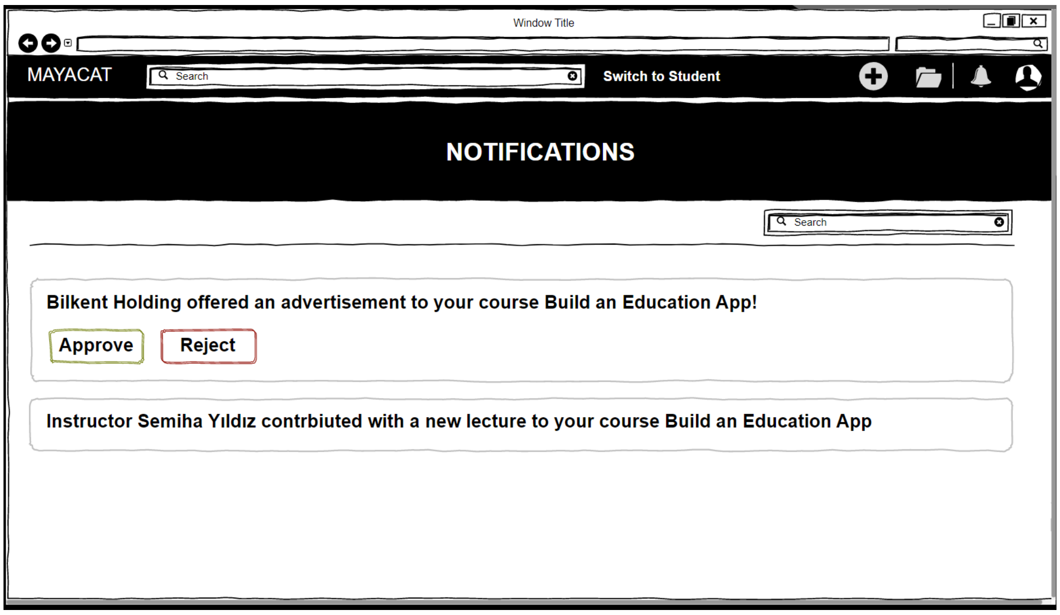
₺30.00
ADVERTISE

SQL Statement:

--status = 0 means it is not reviewed yet

insert into Advertisement (836, “isikozsoy”, 1, “ad-836-img.php”, 0, 339.99, ‘2021-04-21’, ‘2021-04-30’);

Notification Page for Instructors that Shows the Advertisement Offer - Additional Functionality



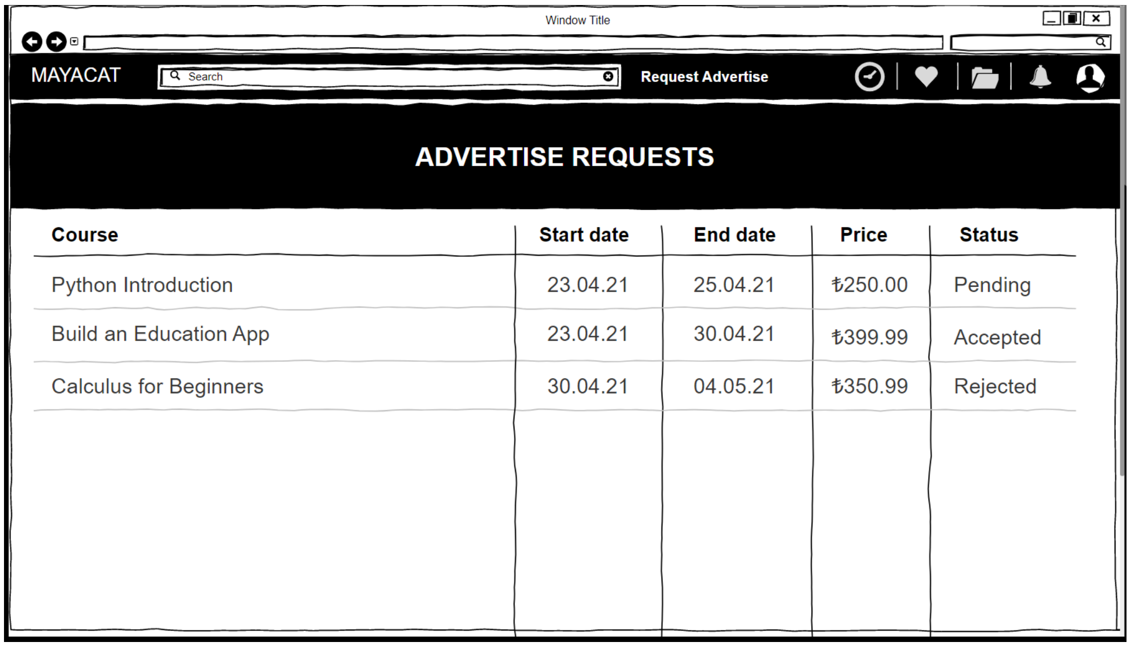
Bilkent Holding offered an advertisement to your course Build an Education App!

Instructor Semiha Yıldız contrbiuted with a new lecture to your course Build an Education App

SQL Statement:

```
select advertisementno, advertisement, company-name, cno
from Advertisement natural joins Advertiser, Course
where Advertisement.cno = Course.cno and Course.owner-username = “mayazsy”;

select cno, cname, i-username, name
from Contributor natural join Course, User
where Course.owner-username = “mayazsy” and User.username = name;
```

Advertisement Requests for Advertisers - Additional Functionality

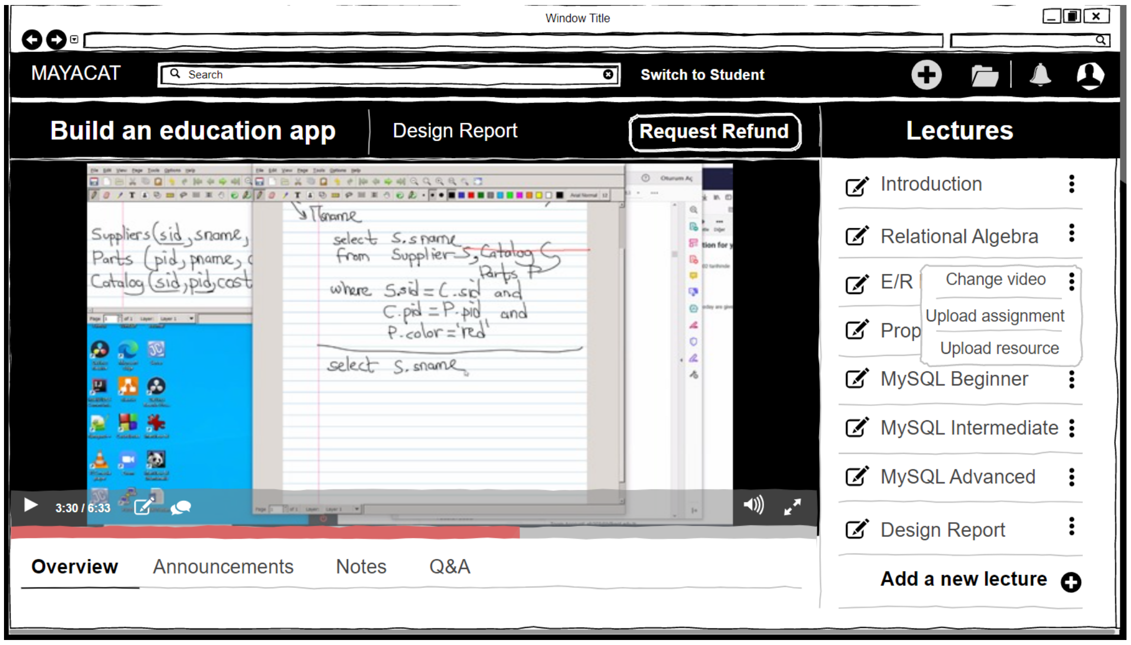


Course	Start date	End date	Price	Status
Python Introduction	23.04.21	25.04.21	₺250.00	Pending
Build an Education App	23.04.21	30.04.21	₺399.99	Accepted
Calculus for Beginners	30.04.21	04.05.21	₺350.99	Rejected

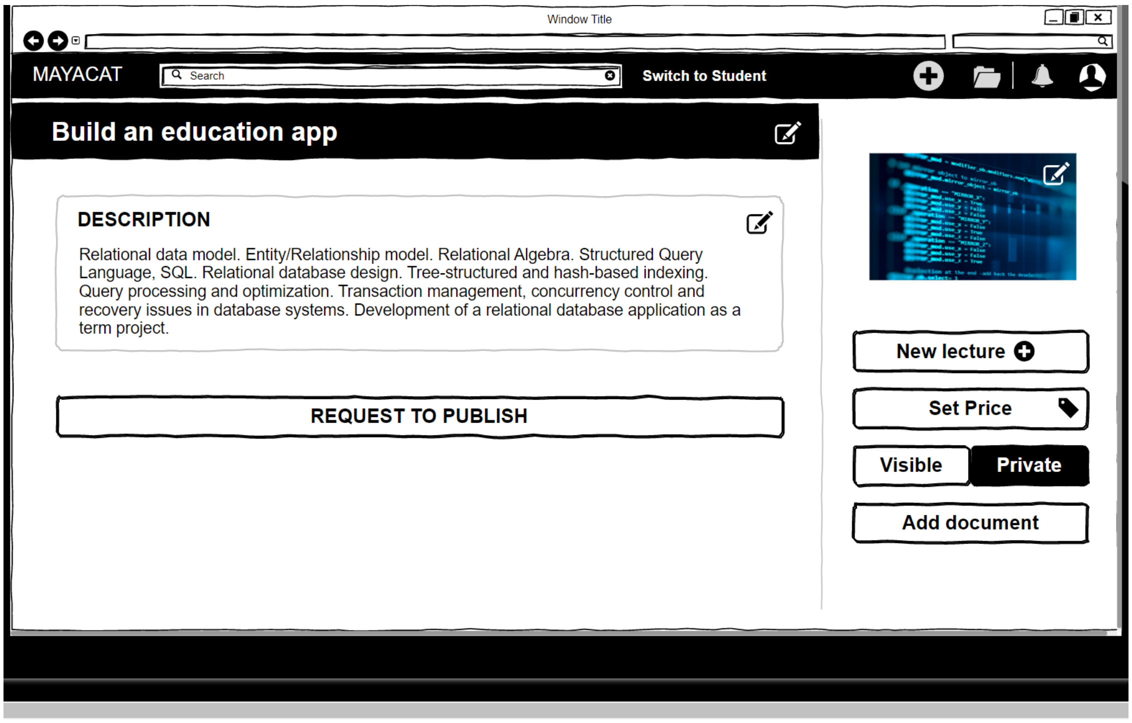
SQL Statement:

select advertisementname, startdate, finishdate, price, status
from Advertisement
where ad-username = 'isikozsoy'; --isikozsoy is an Advertiser

Extra Pages Made for Instructor Which Are Unnecessary for Design Report



```
Suppliers(sid, sname,
Parts (pid, pname,
Catalog (sid, pid, cost

select S.sname
from Suppliers S, Catalog C
                  Parts P
where S.sid = C.sid and
      C.pid = P.pid and
      P.color = 'red'

select S.sname
```



Window Title

MAYACAT

Search

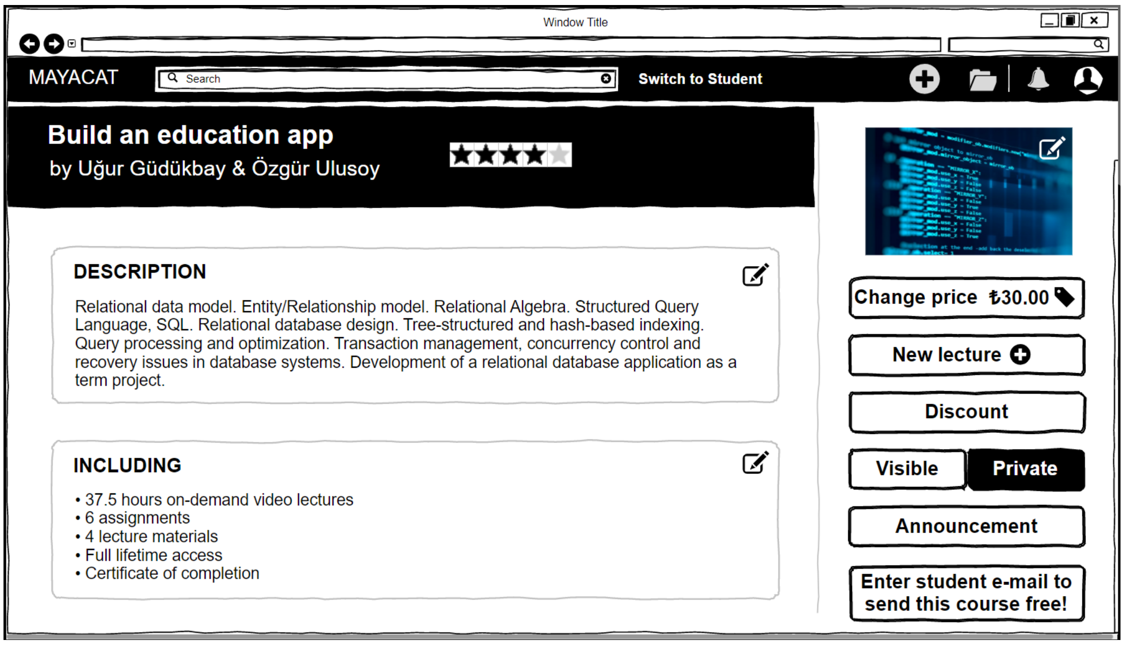
Switch to Student

Build an education app

DESCRIPTION

Relational data model. Entity/Relationship model. Relational Algebra. Structured Query Language, SQL. Relational database design. Tree-structured and hash-based indexing. Query processing and optimization. Transaction management, concurrency control and recovery issues in database systems. Development of a relational database application as a term project.

REQUEST TO PUBLISH



Window Title

MAYACAT

Search

Switch to Student

Build an education app

by Uğur Güdükbay & Özgür Ulusoy

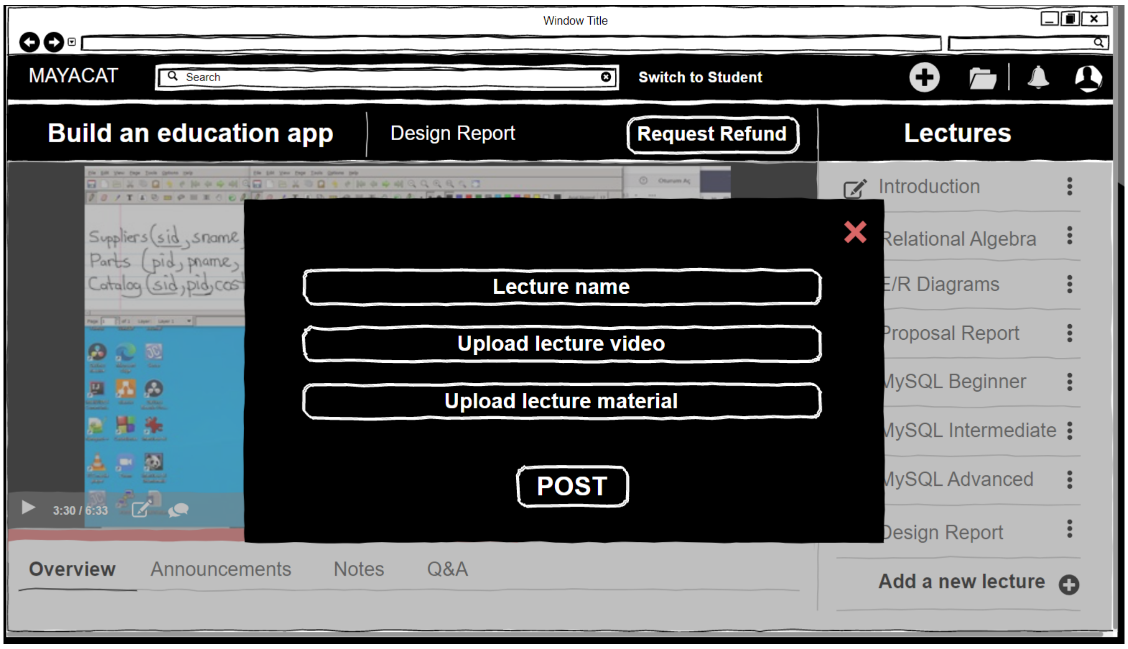
DESCRIPTION

Relational data model. Entity/Relationship model. Relational Algebra. Structured Query Language, SQL. Relational database design. Tree-structured and hash-based indexing. Query processing and optimization. Transaction management, concurrency control and recovery issues in database systems. Development of a relational database application as a term project.

INCLUDING

- 37.5 hours on-demand video lectures
- 6 assignments
- 4 lecture materials
- Full lifetime access
- Certificate of completion

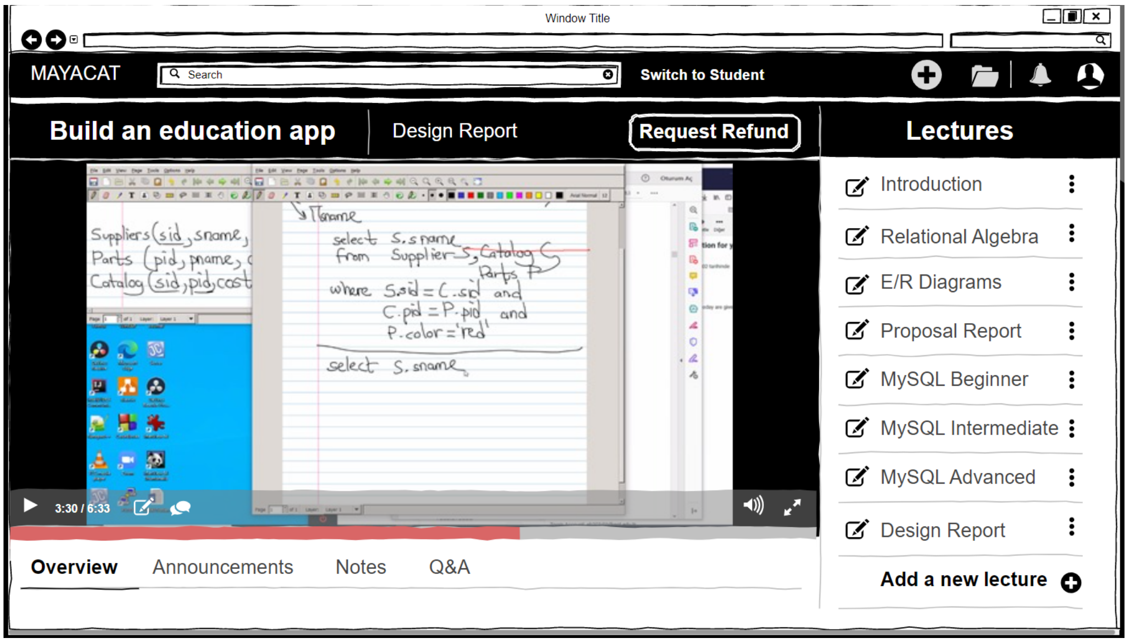
Add Lecture Material Page



SQL Statement:

--1432 is the number of the lecture Indexing in the Build an education app and an additional material is added to that lecture.

insert into LectureMaterial values (100321, “100321.php” ,1432);



IV. Systems and Technologies

We have decided to use JavaScript, HTML, and PHP for website design and functionalities and MySQL for the database management system.