KUZEY KIBRIS KAMPUSU
ODTÜ METU
NORTHERN CYPRUS CAMPUS

# CNG 352 : Database Management Systems

## Step 5: Relational Database Implementation

TEST YOURSELF

Yigit Kaleli - 2152007

Ziya Taner Kececi - 2152049

Yusuf Mete Kabakci - 2151983

# Description:

Our project "Test Yourself" is a web application that helps high school students to prepare for the exams. In our project Users have to register to the system to use it. In our project any User can register to the system in two ways. One way is registering as a Student and the second way is registering as an Instructor. If the user enters the system as a student, he/she can solve the instructor's uploaded tests and watch their solution videos. Also, Students can rate the instructors and comment on the solution videos. To be able to reach these features a student has to subscribe to that particular instructor which will be a one-time payment. Our project has a ticket system that is designed to help Instructors and Students. While a user is creating a ticket, he/she has to choose a category of the problem, write an explanation and/or a screenshot. To solve a problem admins are responding to the ticket and the status of the ticket will be updated to "solved". Also, users can see tickets they have created and follow the changes in ticket status. On the other hand, if the user wants to enter the system as an Instructor, he/she has to upload their Diploma(s) to the system. Instructors can upload tests to the system as well as solution videos of those tests they have uploaded. They can see the comments on their videos and reply to them. In addition to those features, our project generates statistical information such as overall success, historical improvement, correct/wrong answers, and those statistics can be examined by the students so that they can see the benefits of using this system.

# User:

## Data Requirements:

### Students:

In our Test Yourself project users can register to the system as a Student. While they are registering in this way, they are asked to give details such as Name, Surname, Email, Password, Grade, Bank Information (Optional), Date of Birth, Address, Phone Number, so that information can be processed.

### Instructors:

In our Test Yourself project users can register to the system as an Instructor. While they are registering in this way, they are asked to give details such as Name, Surname, Email, Password, Bank Information, Date of Birth, Address, Phone Number, Diploma, Subject (Expertise Field), so that information can be processed.

### Subscription information:

In our Test Yourself project, students can subscribe to instructors and we keep subscription data so that students can see whom they are subscribed to, and instructors can see who subscribed to them.

### Comments:

In our project students able to comment on the instructor's both tests and solution videos and instructors able to reply to those comments. These data will be collected to keep track of interactions in the system.

## Rates:

In our project students can rate instructors. So that instructors can be ranked and eventually this will be good feedback for them.

## Payment/Withdrawal Process:

When a student decided to subscribe to an instructor, the student has to give his/her credit card information for payment so that we kept their credit card information also, instructors can withdraw their money from the system.

## Payment/Withdrawal History:

After payment is proceeded students able to see their payment history logs. In addition, instructors can see their withdrawal history.

## Tickets:

In our ticket system students and instructors can create tickets to solve their problems. To create tickets, they should select problem type listed in the given form then they should write problem description as message. Problem types can be Payment Failure, Withdrawal Failure, Report Plagiarism, Subscription Failure and other types.

## Tests:

In our project Instructors are allowed to upload Tests that are in format of multiple-choice questions in the system. Instructors should type questions and they can upload related images to given questions. Instructors are free to upload as many tests as they want. The only rule is that not violating plagiarism rules. When an instructor uploads a test, they need to specify name of the test, and the test can cover multiple subjects which are determined by each distinct question subject. In addition, when a student completes a test, they are redirected to results page, in this page they can see the correct answers of questions and also, a video link of solutions if there is any.

## Questions:

When an instructor uploads a question first, they need to give the subject of that question and they need to type the body of the question and if they want, they can upload an image. Each question should have four option they can be either text or image. Every option has a “radio button” next to it and students can click only one of them, an instructor must decide which option is correct one before uploading that question.

## Solution Videos:

In our system, Instructors can also upload solution videos for their tests using third party website such as “Youtube” and this video will be embedded to solution page of the test and students can type comment to below the video.

# Transaction Requirements:

## Data Entry:

- Enter the details of new registered User.
- Enter the new comment. Such as “great video”.
- Enter the rate for instructor.
- Enter the details of ticket.
- Enter the details of new tests.
- Enter the solution videos.
- Enter the bank account details.

## Data Update/Delete:

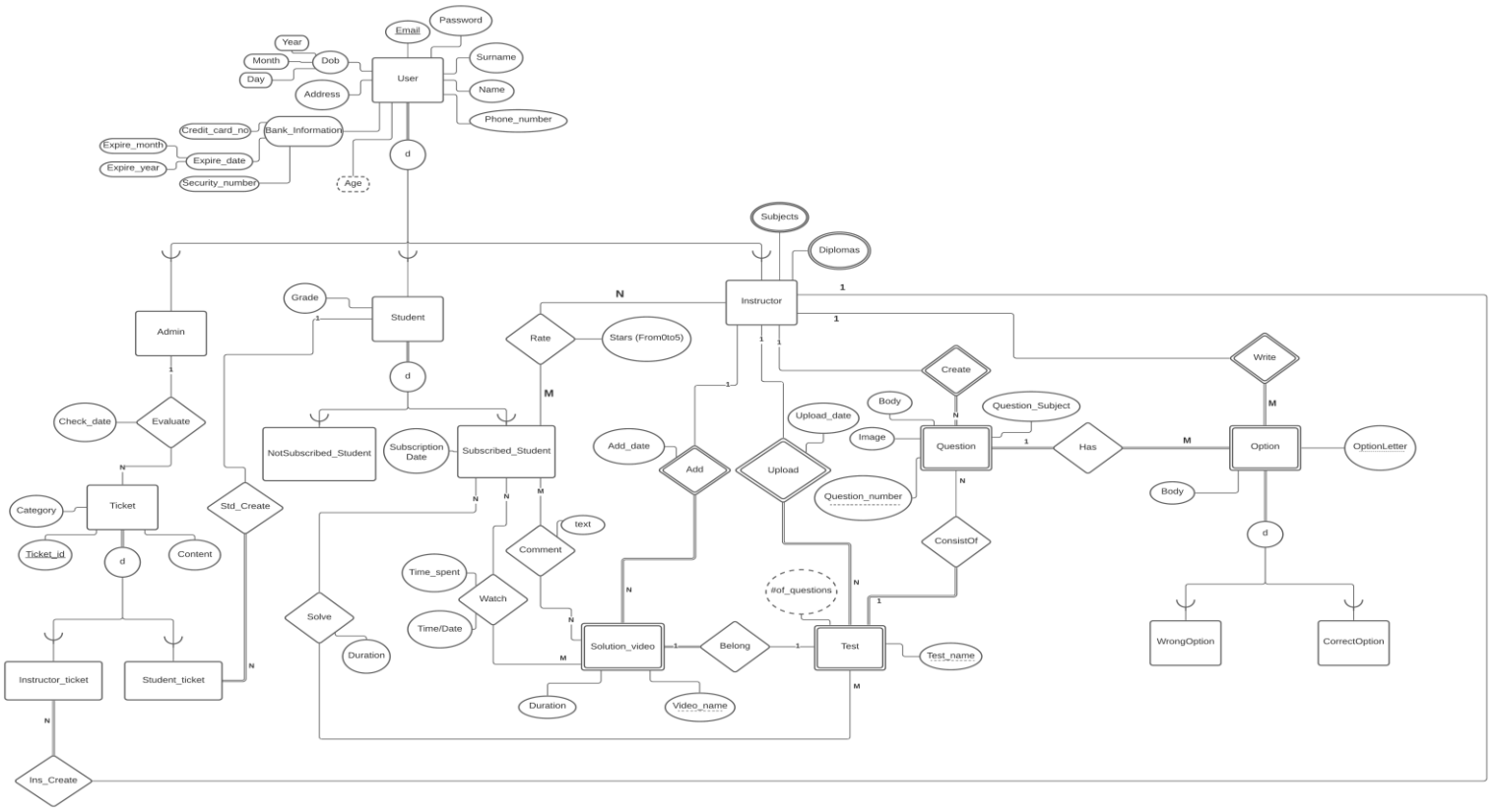
- Update and delete user information.
- Update and delete comments.
- Update and delete Rate for instructors.
- Update and delete bank information.
- Update and delete ticket information.
- Update and delete details of tests.
- Update and delete solution videos.

## Data Queries:

Examples of queries required by the Students/Instructors user views:

- List the details of students who live in given city and whose success rate is higher than 90.
- List the comments of a given student.
- List the tests of instructor whose rate is higher than 4.5(out of 5).
- List the Instructors who has more than 5000 subscribers.
- List the tickets which created by themselves.
- List the credit card details added by themselves.
- List the top 100 students whose stats are highest.
- List the tests added by himself/herself.
- List the students who subscribed himself/herself.
- List solution videos which added in a specific day.
- List the solutions videos which added by a given instructor.
- List the comments of a given solution videos.
- List the top 100 instructors who has most subscribers to see the most popular ones.

# EER DIAGRAM: Test Yourself



# Mapping:

- Step 1: We started mapping with regular entities which is only user entity which has primary key Email.

- Step 2: We mapped weak entities which are ticket, Solution_video, Test, Question. Ticket entity's partial key is ticket_id, Solution_video's partial key is Video_name, Test's partial key is Test_name, Question's partial key is Question_number.

  Step 3: We mapped our 1-to-1 relations which is only Belong. For this one we added "Test_name" to the Solution_video entity as foreign key.

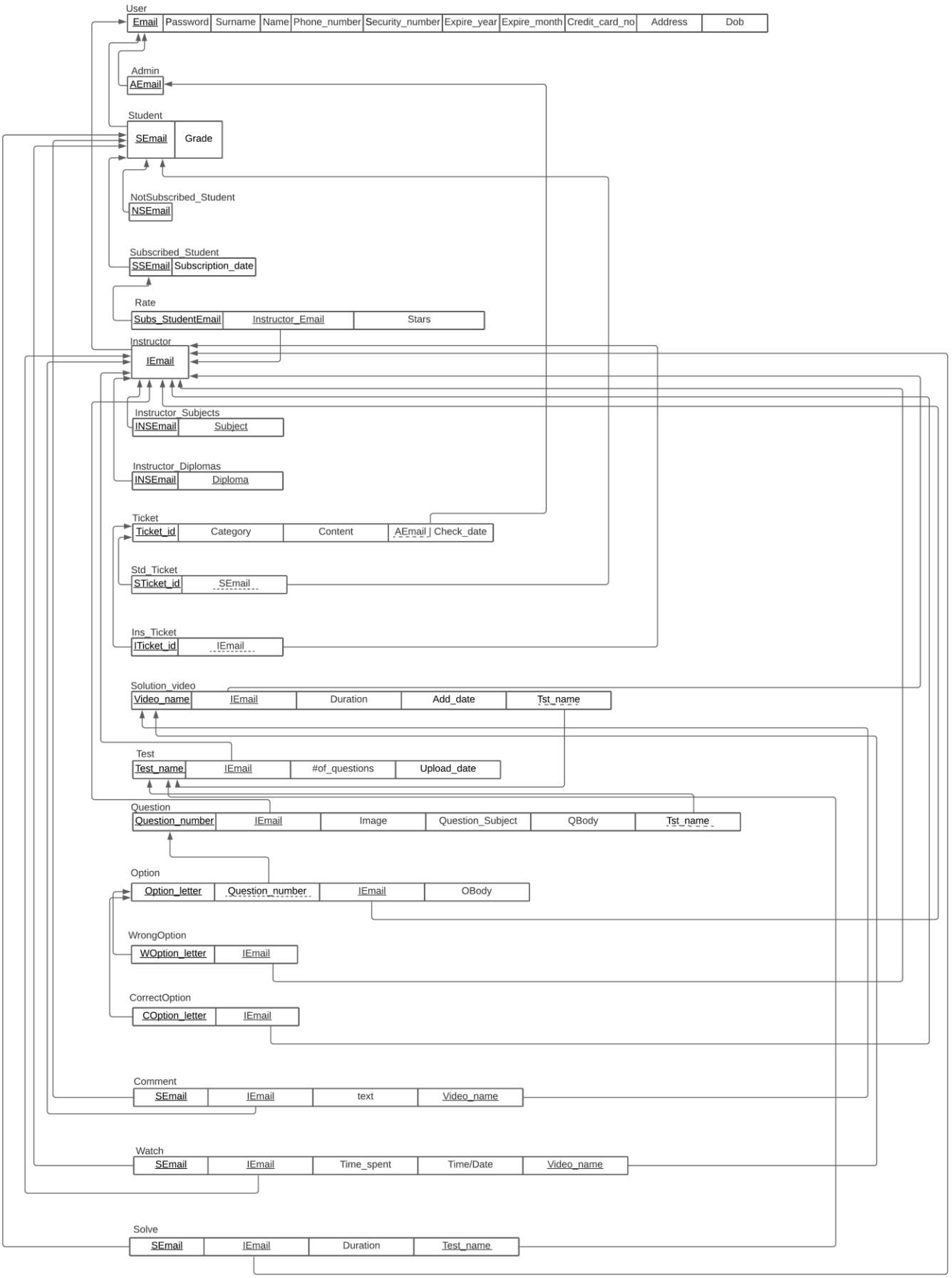
- Step 4: We mapped our 1-to-Many relations which are Evaluate, Add, Upload, Create, ConsistOf. For Add, Upload and Create relations we added "User's Email" to Solution_video, Test and Question as foreign key. For evaluate, we added "User's email" to Ticket entity as foreign key. For ConsistOf we added Test_name to Question entity as foreign key.

- Step 5: We mapped our many-to-many relations which are Solve, Watch, Comment, Subscribe. For these relations we created new entities and added participants entities primary keys.

- Step 6: We mapped our multi valued attributes which are Subjects, Diplomas, Subscribed_instructors, Wrong_options.

- Step 7: Because we had no N-ary relation we skipped this step.

- Step 8: We mapped our EER relations which are User, Student and Ticket. For these mappings we used "Multiple Relations-Superclass and Subclasses" method.


User
Email
Password
Surname
Name
Phone_number
Security_number
Expire_year
Expire_month
Credit_card_no
Address
Dob
Admin
AEmail
Student
SEmail
Grade
NotSubscribed_Student
NSEmail
Subscribed_Student
SSEmail
Subscription_date
Rate
Subs_StudentEmail
Instructor_Email
Stars
Instructor
IEmail
Instructor_Subjects
INSEmail
Subject
Instructor_Diplomas
INSEmail
Diploma
Ticket
Ticket_id
Category
Content
AEmail
Check_date
Std_Ticket
STicket_id
SEmail
Ins_Ticket
ITicket_id
IEmail
Solution_video
Video_name
IEmail
Duration
Add_date
Tst_name
Test
Test_name
IEmail
#of_questions
Upload_date
Question
Question_number
IEmail
Image
Question_Subject
QBody
Tst_name
Option
Option_letter
Question_number
IEmail
OBody
WrongOption
WOption_letter
IEmail
CorrectOption
COption_letter
IEmail
Comment
SEmail
IEmail
text
Video_name
Watch
SEmail
IEmail
Time_spent
Time/Date
Video_name
Solve
SEmail
IEmail
Duration
Test_name

# Normalization:

1NF) We checked if entity has multi valued attribute. If there, is it violates 1NF.

2NF) We checked entities with composite keys? If there is composite key than rest of the attributes must depends on that composite key (not any partial of it)

3NF) We checked if there is any transitive relation. If there is it violates 3NF.

BCNF) We checked if every entity has one functional dependency with super key. If there is any it violates BCNF.

User

| Email | Password | Surname | Name | Phone_number | Security_number | Expire_year | Expire_month | Credit_card_no | Address | Day | Month | Year | Account_type |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

FD1 : Email -----> Password, Surname, Name, Phone_number, Security_number, Expire_year, Expire_month, Credit_card_no, Address, Day, Month, Year, Account_type
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Admin

| AEmail |
|---|

No functional dependency for this entity
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Instructor

| IEmail |
|---|

No functional dependency for this entity
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Student

| SEmail | Grade | Student_type |
|---|---|---|

FD1: SEmail ---> Grade, Student_type
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Instructor_Subjects

| INSEmail | Subject |
|---|---|

FD1: INSEmail ---> Subject
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Instructor_Diplomas

| INSEmail | Diploma |
|---|---|

FD1: INSEmail ---> Subject
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

NotSubscribed_Student

| NSEmail |
|---|

No functional dependency for this entity
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Ticket

| Ticket_id | Category | Content | Ticket_type | AEmail \| Check_date |
|---|---|---|---|---|

FD1: Ticket_id ---> Category, Content, Ticket_type, AEmail, Check_date
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Subscribed_Student

| SSEmail |
|---|

No functional dependency for this entity
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Std_Ticket

| STicket_id | SEmail |
|---|---|

FD1: Sticket_id ---> SEmail
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Ins_Ticket

| ITicket_id | IEmail |
|---|---|

FD1: ITicket_id ---> IEmail
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Student_Subscriptions

| Subs_StudentEmail | Subscribed_Instructor_Email |
|---|---|

FD1: Subs_StudentEmail ---> Subscribed_Instructor_Email
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Solution_video

| Video_name | IEmail | Duration | Add_date | Tst_name |
|---|---|---|---|---|

FD1: Video_id, IEmail ----> Duration, Video_name, Tst_id, Add_date
This entity satisfies 1NF, 2NF, 3NF, BCNF

Test

| Test_name | IEmail | #of_questions | Upload_date |
|---|---|---|---|

FD1: Test_name, IEmail ----> $of_questions, upload_date
This entity satisfies 1NF, 2NF, 3NF, BCNF

Question

| Question_number | IEmail | Image | Question_Subject | Correct_option | Body | Tst_name |
|---|---|---|---|---|---|---|

FD1: Question_number, IEmail ----> Image, Question_Subject, Correct_option, Body, Tst_name
This entity satisfies 1NF, 2NF, 3NF, BCNF

Wrong_option

| WOQuestion_number | IEmail | Woption |
|---|---|---|

No functional dependency for this entity
This Entity, satisfies 1NF, 2NF, 3NF, BCNF

Subscribe

| SEmail | IEmail | Subs_date | Rate |
|---|---|---|---|

FD1: SEmail, IEmail ----> Subs_date, Rate
This entity satisfies 1NF, 2NF, 3NF, BCNF

Comment

| SEmail | IEmail | text | Video_name |
|---|---|---|---|

FD1: SEmail, IEmail, Video_name ----> text
This entity satisfies 1NF, 2NF, 3NF, BCNF

Watch

| SEmail | IEmail | Time_spent | Time/Date | Video_name |
|---|---|---|---|---|

FD1: SEmail, IEmail, Video_name ----> Time_spent, Time/Date
This entity satisfies 1NF, 2NF, 3NF, BCNF

Solve

| SEmail | IEmail | Duration | Video_name |
|---|---|---|---|

FD1: SEmail, IEmail, Video_Name ----> Duration
This entity satisfies 1NF, 2NF, 3NF, BCNF

# Queries

## Inserting 3 new users to the system

```
use test_yourself;

insert into `user` (email, password, surname, name, phone_number, security_number, expire_year, expire_month, credit_card_no, address, dob)
values
('taner@metu.edu.tr', 'tanerpassword123', 'Kececi', 'Ziya Taner', '123123123', 888, 2024, 01,123456, 'Antalya', '1997-03-10'),
('ayigit@metu.edu.tr', '123yigit123', 'Kaleli', 'Yigit', '333344444', 999, 2025, 02,1231234, 'istanbul', '1998-02-19'),
('mete@metu.edu.tr', '666mete666', 'Kabakci', 'Yusuf Mete', '555522222', 777, 2026, 03,557744, 'ankara', '1996-11-16');

select * from `user`;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| email | password | surname | name | phone_number | security_number | expire_year | expire_month | credit_card_no | address | dob |
|---|---|---|---|---|---|---|---|---|---|---|
| ayigit@metu.edu.tr | 123yigit123 | Kaleli | Yigit | 333344444 | 999 | 2025 | 2 | 1231234 | istanbul | 1998-02-19 |
| mete@metu.edu.tr | 666mete666 | Kabakci | Yusuf Mete | 555522222 | 777 | 2026 | 3 | 557744 | ankara | 1996-11-16 |
| taner@metu.edu.tr | tanerpassword123 | Kececi | Ziya Taner | 123123123 | 888 | 2024 | 1 | 123456 | Antalya | 1997-03-10 |
| useremail1 | userpasword1 | usersurname1 | username1 | usrphonenum1 | 111111 | 1111 | 1 | 11111 | useraddress1 | 1997-10-06 |
| useremail10 | userpasword10 | usersurname10 | username10 | usrphonenum10 | 111120 | 1120 | 10 | 11120 | useraddress10 | 1997-10-15 |
| useremail11 | userpasword11 | usersurname11 | username11 | usrphonenum11 | 111121 | 1121 | 11 | 11121 | useraddress11 | 1997-10-16 |
| useremail12 | userpasword12 | usersurname12 | username12 | usrphonenum12 | 111122 | 1122 | 12 | 11122 | useraddress12 | 1997-10-17 |
| useremail13 | userpasword13 | usersurname13 | username13 | usrphonenum13 | 111123 | 1123 | 1 | 11123 | useraddress13 | 1997-10-18 |
| useremail14 | userpasword14 | usersurname14 | username14 | usrphonenum14 | 111124 | 1124 | 2 | 11124 | useraddress14 | 1997-10-19 |
| useremail15 | userpasword15 | usersurname15 | username15 | usrphonenum15 | 111125 | 1125 | 3 | 11125 | useraddress15 | 1997-10-20 |

user 3 | Apply

## Deleting 3 users that we added

```
use test_yourself;

delete from `user` where user.email = 'ayigit@metu.edu.tr';
delete from `user` where user.email = 'taner@metu.edu.tr';
delete from `user` where user.email = 'mete@metu.edu.tr';

select * from `user`;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| email | password | surname | name | phone_number | security_number | expire_year | expire_month | credit_card_no | address | dob |
|---|---|---|---|---|---|---|---|---|---|---|
| useremail1 | userpasword1 | usersurname1 | username1 | usrphonenum1 | 111111 | 1111 | 1 | 11111 | useraddress1 | 1997-10-06 |
| useremail10 | userpasword10 | usersurname10 | username10 | usrphonenum10 | 111120 | 1120 | 10 | 11120 | useraddress10 | 1997-10-15 |
| useremail11 | userpasword11 | usersurname11 | username11 | usrphonenum11 | 111121 | 1121 | 11 | 11121 | useraddress11 | 1997-10-16 |
| useremail12 | userpasword12 | usersurname12 | username12 | usrphonenum12 | 111122 | 1122 | 12 | 11122 | useraddress12 | 1997-10-17 |
| useremail13 | userpasword13 | usersurname13 | username13 | usrphonenum13 | 111123 | 1123 | 1 | 11123 | useraddress13 | 1997-10-18 |
| useremail14 | userpasword14 | usersurname14 | username14 | usrphonenum14 | 111124 | 1124 | 2 | 11124 | useraddress14 | 1997-10-19 |
| useremail15 | userpasword15 | usersurname15 | username15 | usrphonenum15 | 111125 | 1125 | 3 | 11125 | useraddress15 | 1997-10-20 |
| useremail16 | userpasword16 | usersurname16 | username16 | usrphonenum16 | 111126 | 1126 | 4 | 11126 | useraddress16 | 1997-10-21 |
| useremail17 | userpasword17 | usersurname17 | username17 | usrphonenum17 | 111127 | 1127 | 5 | 11127 | useraddress17 | 1997-10-22 |
| useremail18 | userpasword18 | usersurname18 | username18 | usrphonenum18 | 111128 | 1128 | 6 | 11128 | useraddress18 | 1997-10-23 |
| useremail19 | userpasword19 | usersurname19 | username19 | usrphonenum19 | 111129 | 1129 | 7 | 11129 | useraddress19 | 1997-10-24 |
| useremail2 | userpasword2 | usersurname2 | username2 | usrphonenum2 | 111112 | 1112 | 2 | 11112 | useraddress2 | 1997-10-07 |

ser 3

# Updating 3 users

```sql
use test_yourself;
update `user` set user.`name` = 'yusuf yigit'
where user.email = 'ayigit@metu.edu.tr';

update `user` set user.`name` = 'TANER TANER'
where user.email = 'taner@metu.edu.tr';

update `user` set user.`name` = 'KETE METE'
where user.email = 'mete@metu.edu.tr';
select * from `user`;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

| email | password | surname | name | phone_number | security_number | expire_year | expire_month | credit_card_no | address | dob |
|---|---|---|---|---|---|---|---|---|---|---|
| ayigit@metu.edu.tr | 123yigit123 | Kaleli | yusuf yigit | 333344444 | 999 | 2025 | 2 | 1231234 | istanbul | 1998-02-19 |
| mete@metu.edu.tr | 666mete666 | Kabakci | KETE METE | 555522222 | 777 | 2026 | 3 | 557744 | ankara | 1996-11-16 |
| taner@metu.edu.tr | tanerpassword123 | Kececi | TANER TANER | 123123123 | 888 | 2024 | 1 | 123456 | Antalya | 1997-03-10 |
| useremail1 | userpasword1 | usersurname1 | username1 | usrphonenum1 | 111111 | 1111 | 1 | 11111 | useraddress1 | 1997-10-06 |
| useremail10 | userpasword10 | usersurname10 | username10 | usrphonenum10 | 111120 | 1120 | 10 | 11120 | useraddress10 | 1997-10-15 |

Table Creation queries and Data Population Queries were around 1200 lines thus we included them as sql file.

# Assumptions:

1- User can only be an Admin, Student, or Instructor.

2- Student users must enter their grade to the system.

3- Instructors must enter their diploma(s), and subject of expertise(s) to the system.

4- One ticket can be created by a single instructor or a student and can be evaluated by one admin while instructors and students create more than one ticket, and one admin can evaluate more than more ticket.

5- All students can subscribe to any instructor.

6- A student must subscribe to an instructor to solve his/her tests and watch his/her solution videos.

7- Tests consist of many questions; one question can be used only one test.

8- It is not a must for Students to watch solution videos of tests they have solved.

9- In this system an “not_subscribed_student” cannot solve tests and watch solution videos, but they can create tickets, subscribe to instructors.

10- While registering system only email must be unique any other attributes do not have to be unique.

11- Instructor cannot upload videos with the same name but, one instructor’s video’s name can be same with another instructor’s videos’ name.

12- Instructor cannot upload tests with the same name but, one instructor’s test’s name can be same with another instructor’s test’ name.