# Physiotherapist’s Clinic Database

**Project Report**
**May 2025**

**Lecture**
**BM2223 Database Management Systems**

**Authors**
Malik UĞUR
Adem BURAN
Ahmet Başar ALÜZÜM
Vahdettin OKUR
Metehan AKSAKAL
Arif Can EKİNCİ

**Asst. Prof. Dr. ARZUM KARATAŞ**

# Contents

## 1. Authors

| Author | Number | Scen ario | ERD | Database Schema | SQL Table Creation | Data Entry | Queries | Report |
|---|---|---|---|---|---|---|---|---|
| Malik Uğur | 2311504304 | YES | YES | YES | YES | YES | YES | YES |
| Adem Buran | 2211504033 | YES | YES | YES | YES | | | YES |
| Ahmet Başar Alüzüm | 2211504017 | YES | YES | YES | YES | YES | YES | YES |
| Vahdettin Okur | 2211504006 | YES | YES | YES | YES | | | |
| Metehan Aksakal | 2211504020 | YES | YES | YES | YES | | | |
| Arif Can Ekinci | 2111504014 | YES | YES | YES | YES | YES | YES | |

## 2. General Description of the Project

A detailed description of the project titled **”Physiotherapist’s Clinic Database”**, prepared to meet the requirements of the Database Management Systems Course. This report includes the scenario, ER diagram, schema design, program screenshots, all queries and their results and populated database tables. During the implementation phase of the project, we used Oracle SQL Developer 24.3.1 to model and test the clinic’s operational workflow in a digital environment.

### 2.1 The Core Functions Of The Physiotherapist’s Clinic Database Can Be Listed As Follows

a. **Registering new patients into the database with personal details** such as name, surname, contact information, birth date, and treatment status.

b. **Recording and tracking treatment processes**, including treatment titles, start and end dates, diagnosis details, and associated methods and medicines.

c. **Viewing a patient's complete treatment history**, along with all medicines used and medical methods applied during each treatment.

d. **Generating custom queries and reports** based on patient treatment status, medicine usage, and method application counts.

# 3. The Scenario

As the clinic manager, I want a system that helps us keep track of our patients and the treatments they receive. When a patient comes in, we want to store their **name, birthday, email,** and **phone number**.

We also want to know whether they have started treatment or not. Each treatment has a **start** and **end** date and includes a **diagnosis**. We'd like to add a short title for the **treatment** and also a note about what the doctor thinks the issue might be.

During treatment, we sometimes use different medicines, so the system should help us see which medicines were used and how much. The system should also track how much medicine we have in **stock**, like how many **units** or packages are left.

Lastly, treatments may involve certain methods (like physiotherapy or surgery), and I want the system to show how many times a method was used during a treatment. That way, we want to see implementation time.

## 3.1. DESCRIPTION OF THE SCENARIO

The clinic requires a relational database management system to efficiently store and manage treatment information. The **Client** entity stores key demographic and contact information including name, surname, birthday, email, and phone number, along with treatment status and registration date.

Each **Treatment** entity is associated with one client (client_id as foreign key) and includes additional metadata such as treatment_start, treatment_end, treatment_header, and diagnosis_header. A single client may have multiple treatments over time.

The **Treatment** entity is linked to **Medicine** via a many-to-many relationship, where the usage context (i.e., quantity of the medicine used) is stored in the associative relationship "use" with the attribute amount_of_usage. Each medicine is described by its medicine_name, unit, and available stock.

Treatments may also involve multiple **Methods** (e.g., procedures or therapies). This relationship is captured through the "include" relationship between **Treatment** and **Method**, also many-to-many, which records implement_count. Each method includes method_name and implement_time.

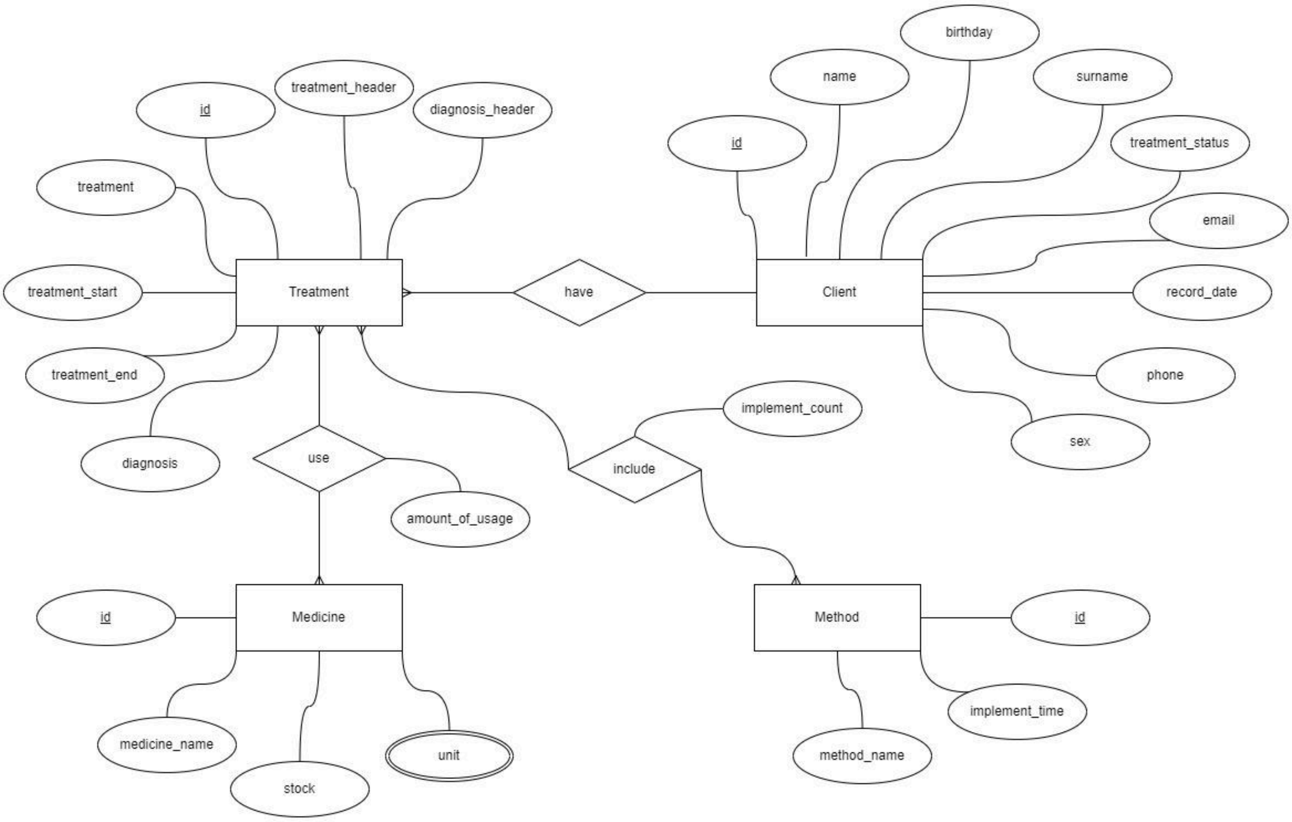
**Relations:**

a) **Treatment – Client:** (N: 1) A single client may have multiple treatments over time.
b) **Treatment – Method:** (N: M) Many treatments may include many methods.
c) **Treatment – Medicine:** (N: M) In many treatments may use many medicines.

**The entire model aims to track:**

a) Client treatment histories
b) Medicine usage and inventory
c) Applied methods per treatment and frequency

## 4. The ER Diagram



## 4.1. Entities: Treatment, Client, Medicine, and Method

## 4.2. Attributes:

a) **Treatment :** id , client_id, treatment_header, diagnosis_header, treatment, treatment_start, treatment_end, diagnosis.

b) **Medicine:** id, medicine_name, stock, unit

c) **Method :** id, implement_time, method_name

d) **Client :** id, name, surname, birthday, treatment_status, email, record_date, phone, sex.

## 4.3. Relations :

| | | |
|---|---|---|
| a) | **Treatment – Client:** | N : 1 |
| b) | **Treatment – Method:** | N : M |
| c) | **Treatment – Medicine:** | N : M |

## 5. The Relational Model

### **Step 1:** Mapping of Regular (Strong) Entity Types

Treatment:

| Id | treatment_header | Treatment | treatment_start | treatment_end | diagnosis | diagnosis_header |
|---|---|---|---|---|---|---|

Client:

| Id | Surname | Name | Birthday | Phone | Email | Sex | record_date | treatment_status |
|---|---|---|---|---|---|---|---|---|

Medicine:

| id | stock | unit | medicine_name |
|---|---|---|---|

Method

| id | implement_time | method_name |
|---|---|---|

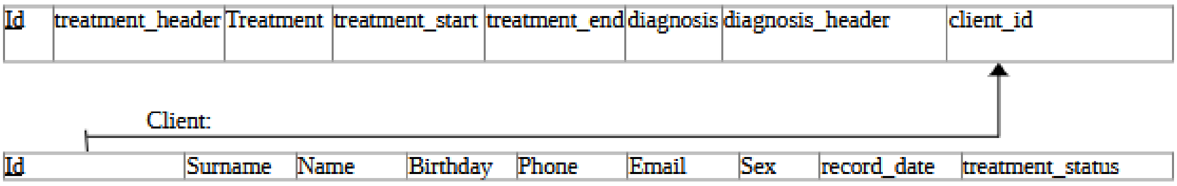
### **Step 2:** Mapping of Weak Entity Types

-There is no weak entities in this scenario.

### **Step 3:** Mapping of Binary 1:1 Relationship Types

-There is no such a relation.

### **Step 4:** Mapping of Binary 1:N Relationship Types



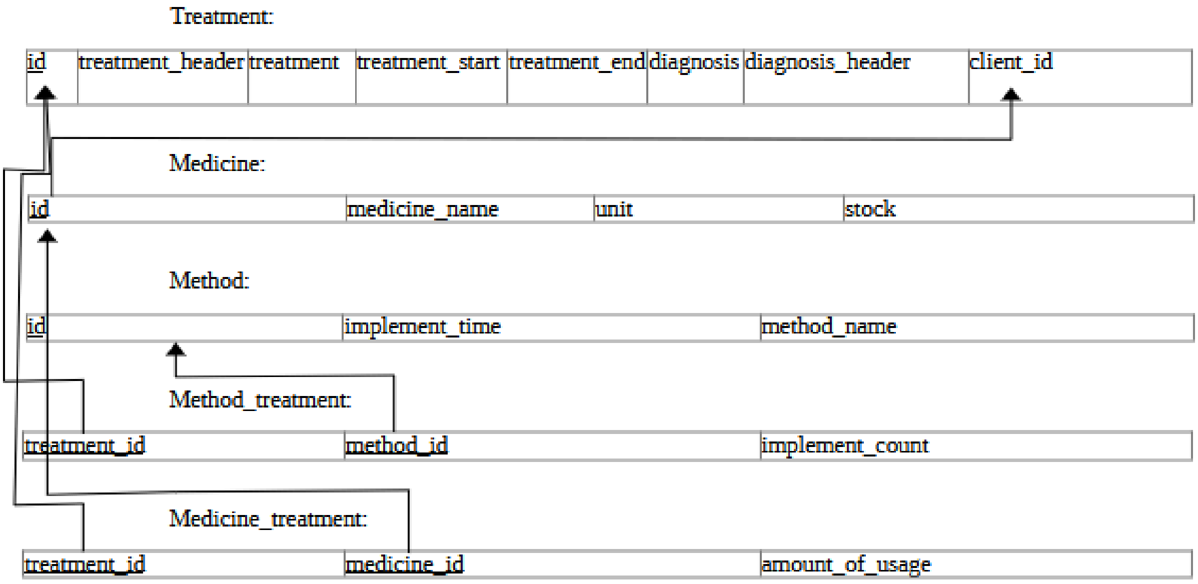
Treatment:

| Id | treatment_header | Treatment | treatment_start | treatment_end | diagnosis | diagnosis_header | client_id |
|---|---|---|---|---|---|---|---|

Client:

| Id | Surname | Name | Birthday | Phone | Email | Sex | record_date | treatment_status |
|---|---|---|---|---|---|---|---|---|

**Step 5:** Mapping of Binary M:N Relationship Types



Treatment:

| id | treatment_header | treatment | treatment_start | treatment_end | diagnosis | diagnosis_header | client_id |
|---|---|---|---|---|---|---|---|

Medicine:

| id | medicine_name | unit | stock |
|---|---|---|---|

Method:

| id | implement_time | method_name |
|---|---|---|

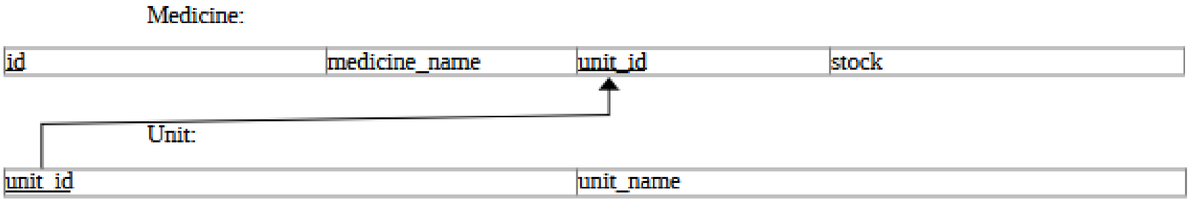
Method_treatment:

| treatment_id | method_id | implement_count |
|---|---|---|

Medicine_treatment:

| treatment_id | medicine_id | amount_of_usage |
|---|---|---|

**Step 6:** Mapping of Multivalued attributes



Medicine:

| id | medicine_name | unit_id | stock |
|---|---|---|---|

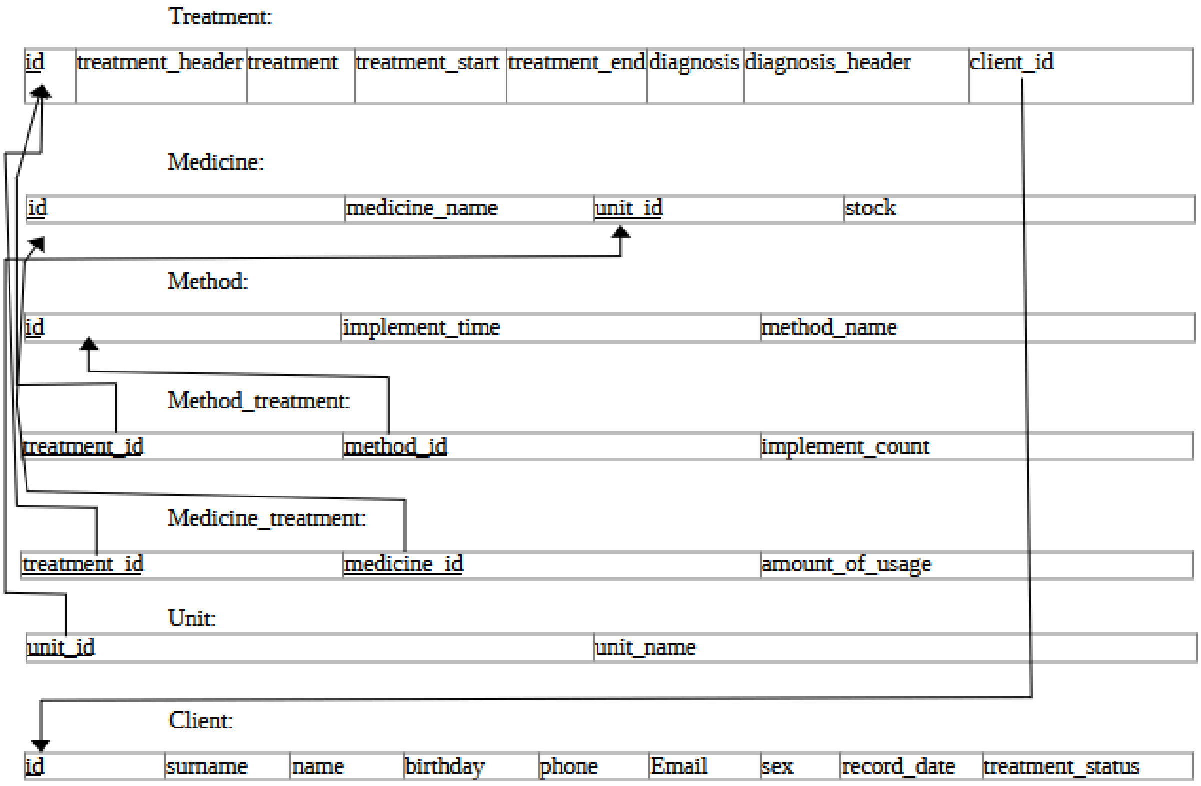
Unit:

| unit_id | unit_name |
|---|---|

**Step 7:** Mapping of N-ary Relationship Types

-There is no N-ary relationship.

## After Mapping



Treatment:

| id | treatment_header | treatment | treatment_start | treatment_end | diagnosis | diagnosis_header | client_id |
|---|---|---|---|---|---|---|---|

Medicine:

| id | medicine_name | unit_id | stock |
|---|---|---|---|

Method:

| id | implement_time | method_name |
|---|---|---|

Method_treatment:

| treatment_id | method_id | implement_count |
|---|---|---|

Medicine_treatment:

| treatment_id | medicine_id | amount_of_usage |
|---|---|---|

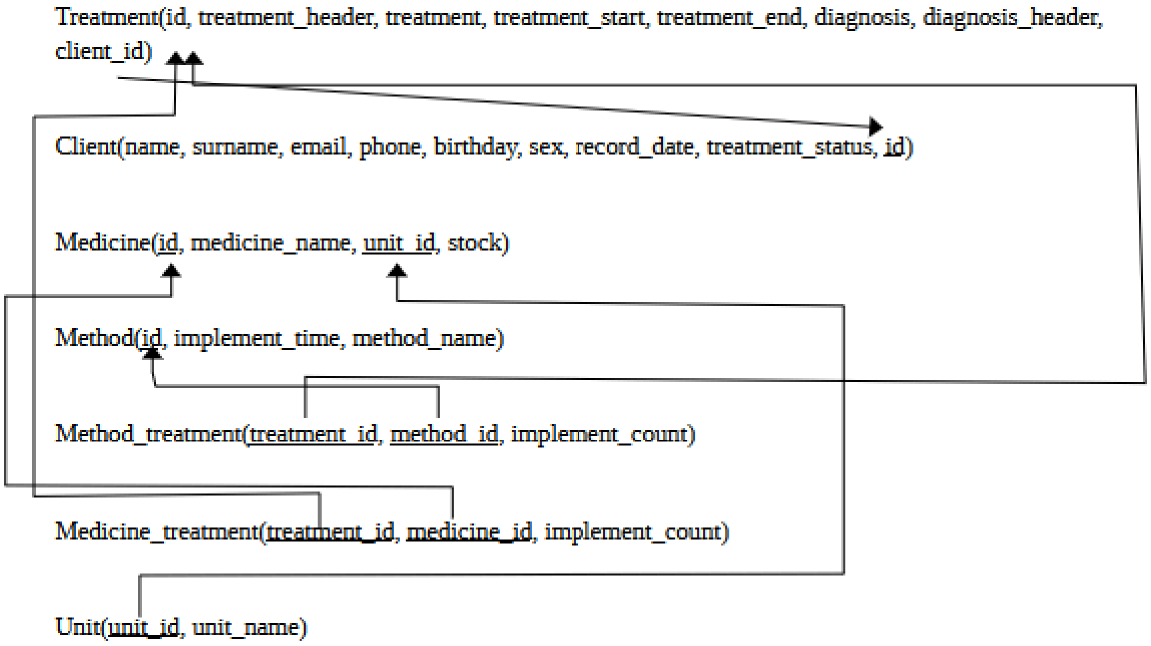
Unit:

| unit_id | unit_name |
|---|---|

Client:

| id | surname | name | birthday | phone | Email | sex | record_date | treatment_status |
|---|---|---|---|---|---|---|---|---|

## Relation Schema of the Final Mapping



Treatment(id, treatment_header, treatment, treatment_start, treatment_end, diagnosis, diagnosis_header, client_id)

Client(name, surname, email, phone, birthday, sex, record_date, treatment_status, id)

Medicine(id, medicine_name, unit_id, stock)

Method(id, implement_time, method_name)

Method_treatment(treatment_id, method_id, implement_count)

Medicine_treatment(treatment_id, medicine_id, implement_count)

Unit(unit_id, unit_name)

# 6. Normalization

## a) First Normal Form

First normal form checks every attribute is single values.In our final tables after mapping there is no need to take action for atomic attributes.

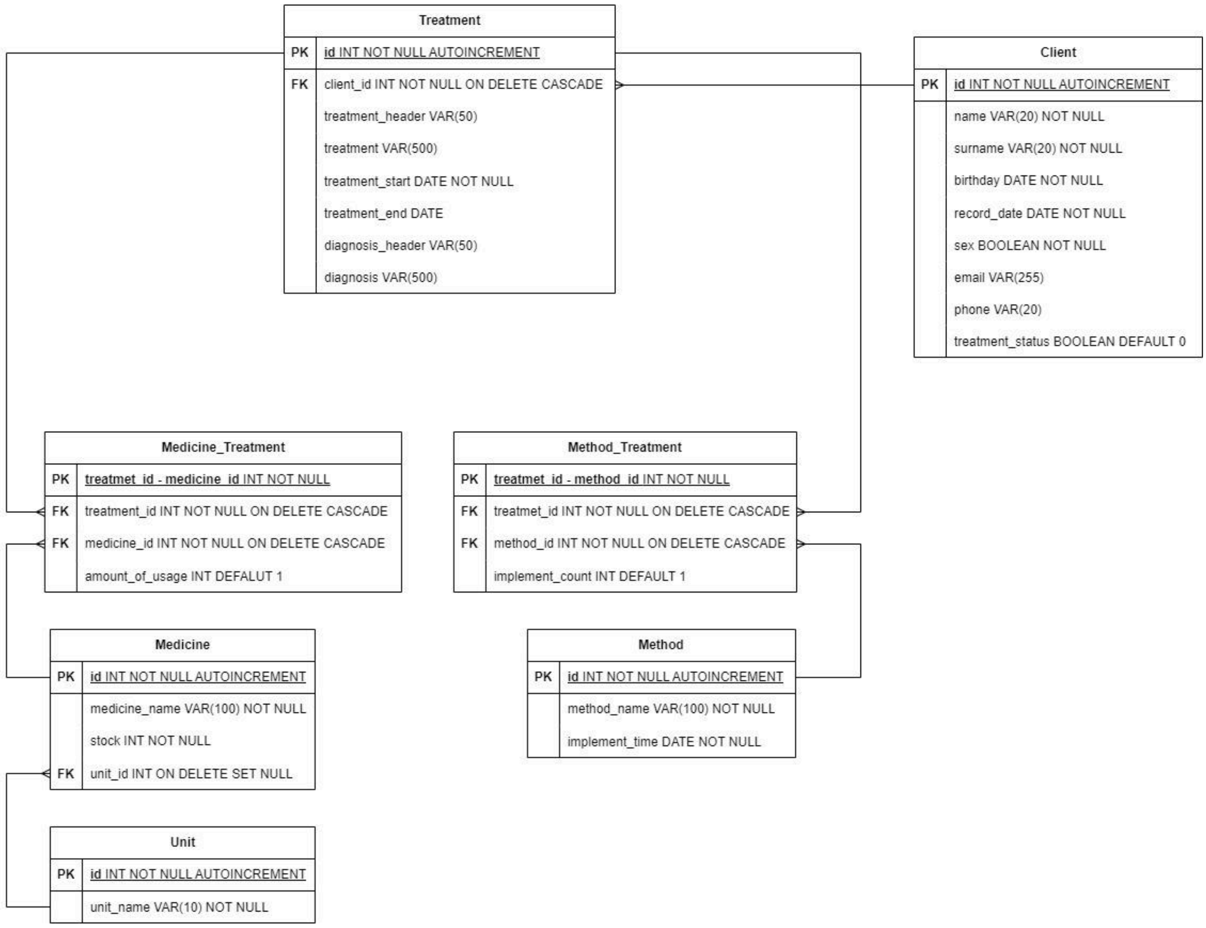
## b) Second Normal Form

Second normal form checks for partial dependencies. In our tables there There is no partial dependencies because mapping helps tables about redundancy, and efficiency and with these things our tables are divided with their n-m relations so our tables with mapping go in a transition.

## c) Third Normal Form

Third normal form checks tables for transitive dependencies. Transitive dependencies may be lost in our database after some deletion operation which data we don't want loose.In our tables there is no such a situation.

# 7. The Database Schema



| | Treatment |
|---|---|
| PK | id INT NOT NULL AUTOINCREMENT |
| FK | client_id INT NOT NULL ON DELETE CASCADE |
| | treatment_header VAR(50) |
| | treatment VAR(500) |
| | treatment_start DATE NOT NULL |
| | treatment_end DATE |
| | diagnosis_header VAR(50) |
| | diagnosis VAR(500) |

| | Client |
|---|---|
| PK | id INT NOT NULL AUTOINCREMENT |
| | name VAR(20) NOT NULL |
| | surname VAR(20) NOT NULL |
| | birthday DATE NOT NULL |
| | record_date DATE NOT NULL |
| | sex BOOLEAN NOT NULL |
| | email VAR(255) |
| | phone VAR(20) |
| | treatment_status BOOLEAN DEFAULT 0 |

| | Medicine_Treatment |
|---|---|
| PK | treatmet_id - medicine_id INT NOT NULL |
| FK | treatment_id INT NOT NULL ON DELETE CASCADE |
| FK | medicine_id INT NOT NULL ON DELETE CASCADE |
| | amount_of_usage INT DEFALUT 1 |

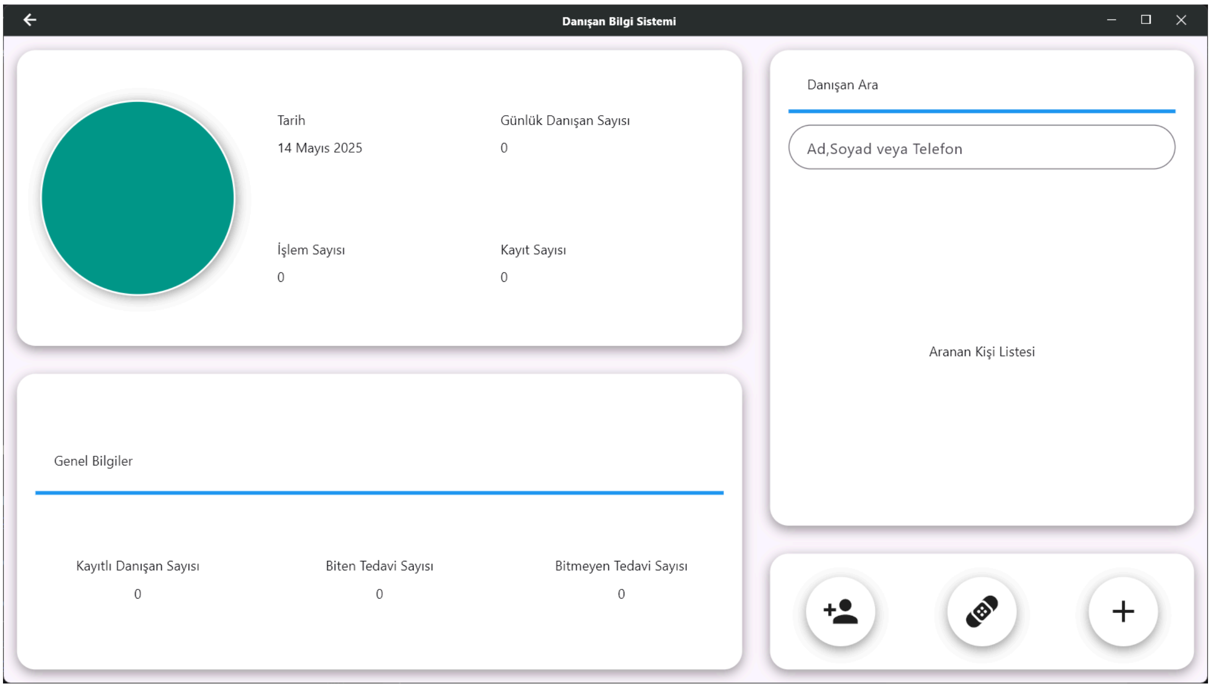
| | Method_Treatment |
|---|---|
| PK | treatmet_id - method_id INT NOT NULL |
| FK | treatmet_id INT NOT NULL ON DELETE CASCADE |
| FK | method_id INT NOT NULL ON DELETE CASCADE |
| | implement_count INT DEFAULT 1 |

| | Medicine |
|---|---|
| PK | id INT NOT NULL AUTOINCREMENT |
| | medicine_name VAR(100) NOT NULL |
| | stock INT NOT NULL |
| FK | unit_id INT ON DELETE SET NULL |

| | Method |
|---|---|
| PK | id INT NOT NULL AUTOINCREMENT |
| | method_name VAR(100) NOT NULL |
| | implement_time DATE NOT NULL |

| | Unit |
|---|---|
| PK | id INT NOT NULL AUTOINCREMENT |
| | unit_name VAR(10) NOT NULL |

# 8. Screenshots



Danışan Bilgi Sistemi

Tarih
14 Mayıs 2025

Günlük Danışan Sayısı
0

İşlem Sayısı
0

Kayıt Sayısı
0

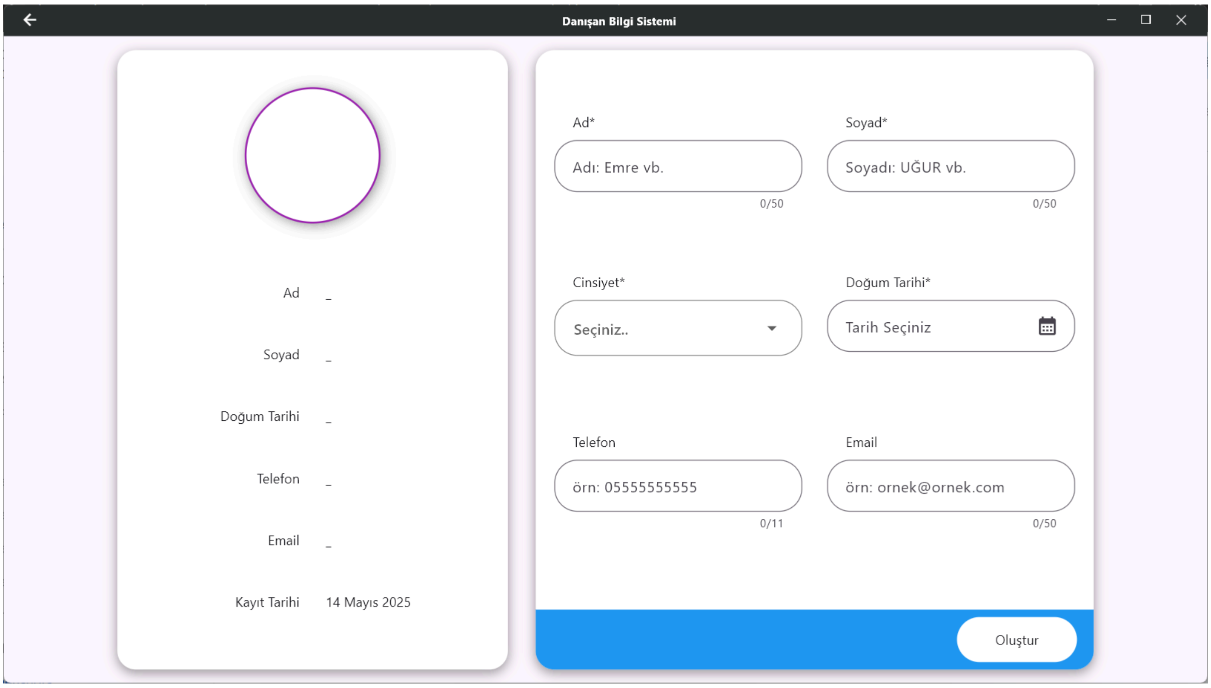
Genel Bilgiler

| Kayıtlı Danışan Sayısı | Biten Tedavi Sayısı | Bitmeyen Tedavi Sayısı |
|---|---|---|
| 0 | 0 | 0 |

Danışan Ara

Ad,Soyad veya Telefon

Aranan Kişi Listesi



Danışan Bilgi Sistemi

| | |
|---|---|
| Ad | _ |
| Soyad | _ |
| Doğum Tarihi | _ |
| Telefon | _ |
| Email | _ |
| Kayıt Tarihi | 14 Mayıs 2025 |

Ad*
Adı: Emre vb.
0/50

Soyad*
Soyadı: UĞUR vb.
0/50

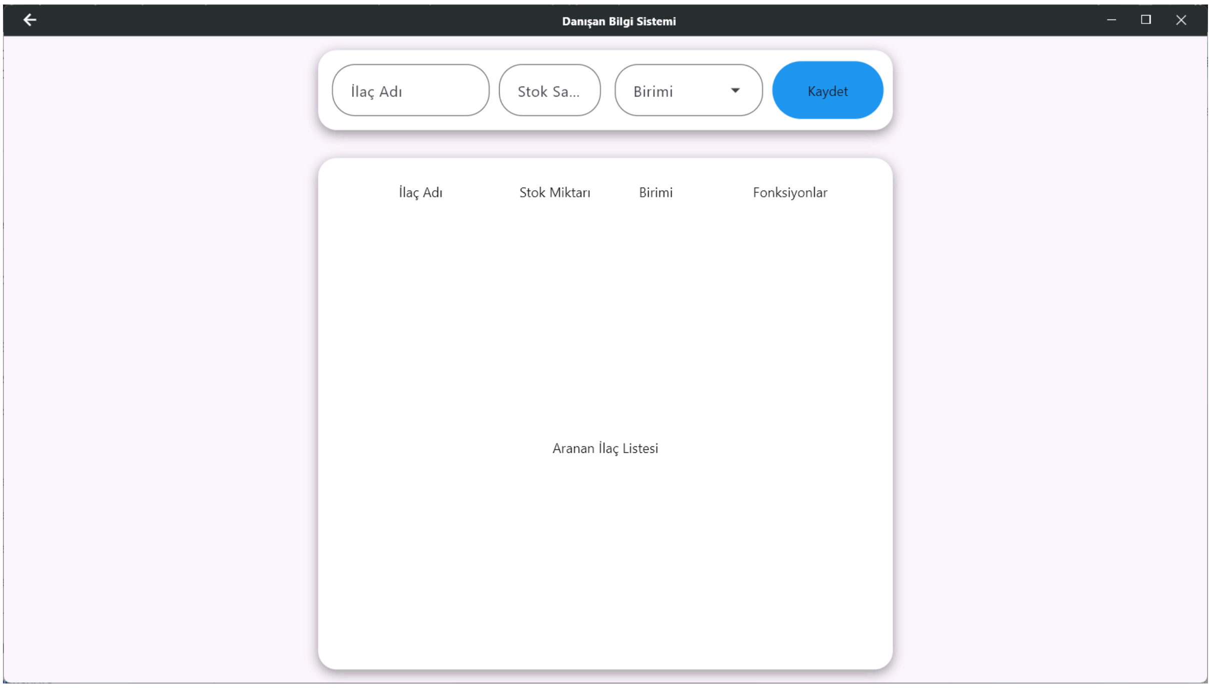
Cinsiyet*
Seçiniz..

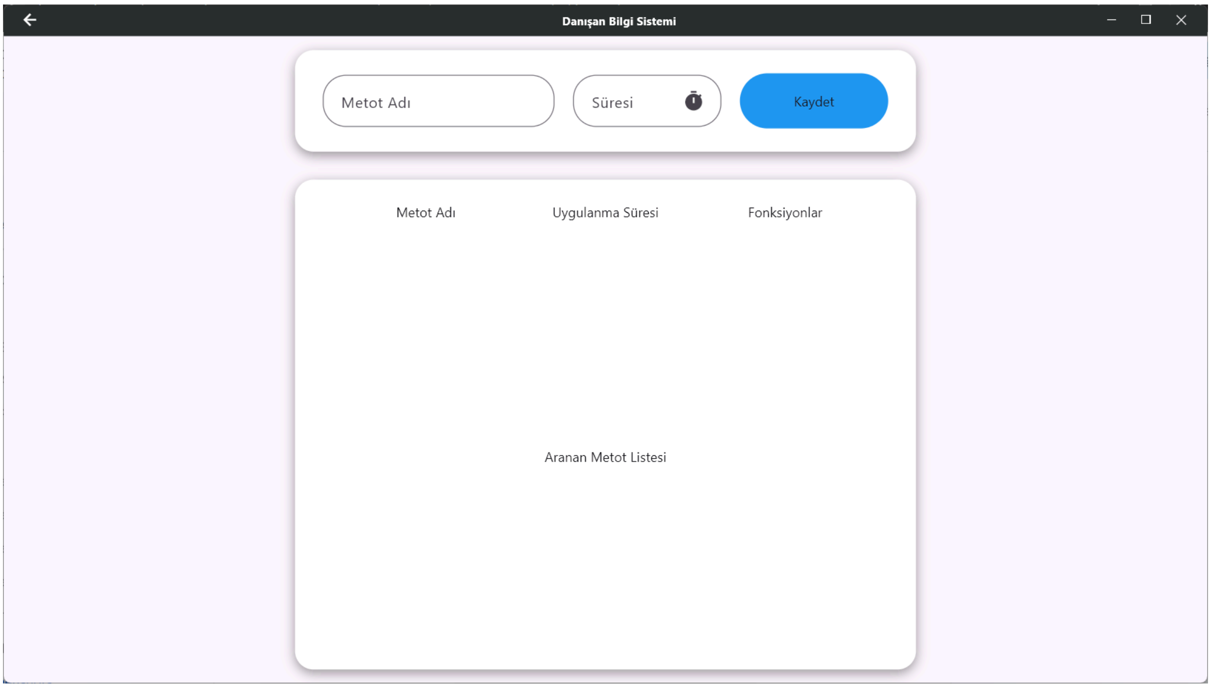
Doğum Tarihi*
Tarih Seçiniz

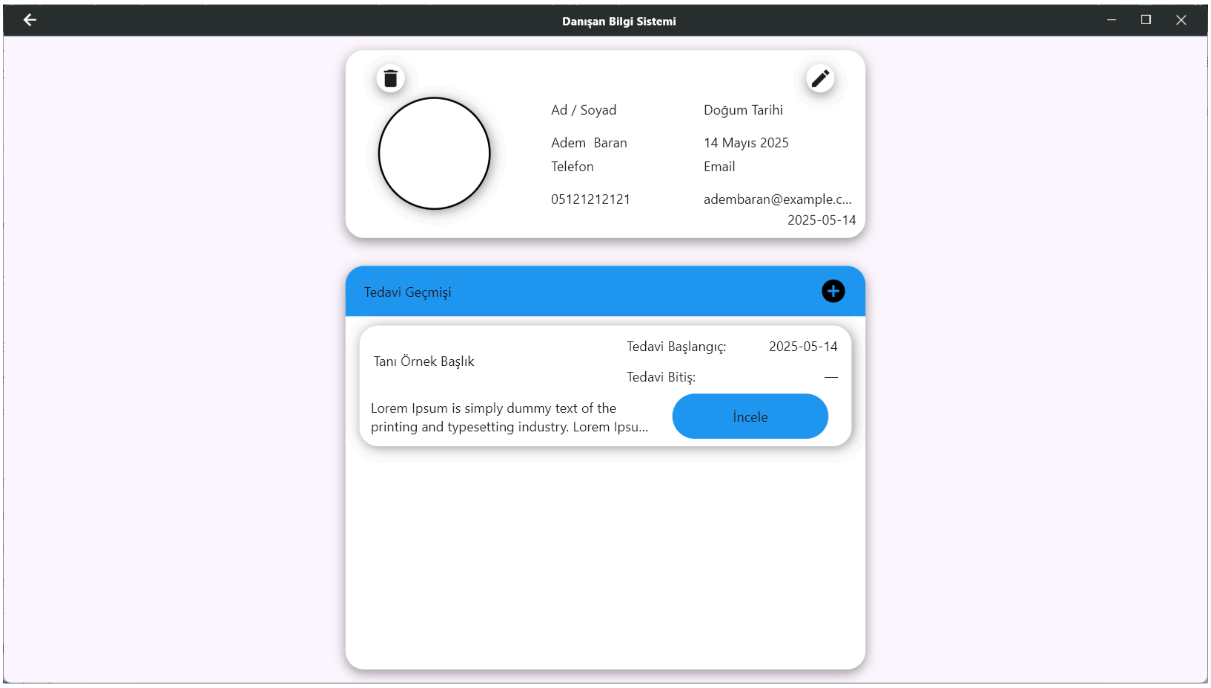
Telefon
örn: 05555555555
0/11

Email
örn: ornek@ornek.com
0/50

Oluştur

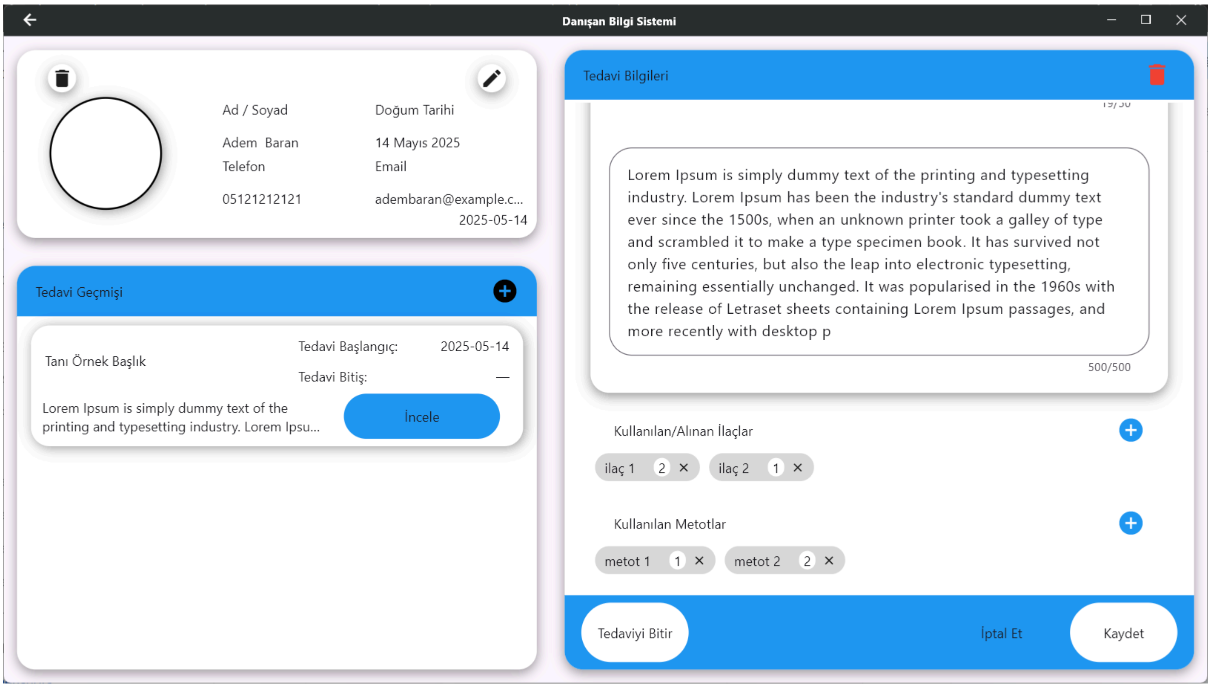

Danışan Bilgi Sistemi
İlaç Adı
Stok Sa...
Birimi
Kaydet
İlaç Adı
Stok Miktarı
Birimi
Fonksiyonlar
Aranan İlaç Listesi


Danışan Bilgi Sistemi
Metot Adı
Süresi
Kaydet
Metot Adı
Uygulanma Süresi
Fonksiyonlar
Aranan Metot Listesi



Ad / Soyad: Adem Baran
Doğum Tarihi: 14 Mayıs 2025
Telefon: 05121212121
Email: adembaran@example.c...
2025-05-14

Tanı Örnek Başlık
Tedavi Başlangıç: 2025-05-14
Tedavi Bitiş: —
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsu...



Ad / Soyad: Adem Baran
Doğum Tarihi: 14 Mayıs 2025
Telefon: 05121212121
Email: adembaran@example.c...
2025-05-14

Tanı Örnek Başlık
Tedavi Başlangıç: 2025-05-14
Tedavi Bitiş: —
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsu...

19/50

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop p

500/500

Kullanılan/Alınan İlaçlar: ilaç 1 (2) ✕, ilaç 2 (1) ✕

Kullanılan Metotlar: metot 1 (1) ✕, metot 2 (2) ✕

## 9. Sample Queries

```sql
SELECT name, surname, treatment_status
FROM Client;
```

```sql
SELECT t.treatment_header, t.treatment_start, t.treatment_end
FROM Treatment t
JOIN Client c ON t.client_id = c.id
WHERE c.name = 'Name_1' AND c.surname = 'Surname_1';
```

```sql
SELECT medicine_name, stock
FROM Medicine;
```

```sql
SELECT t.treatment_header, m.medicine_name
FROM Treatment t
JOIN Medicine_Treatment mt ON t.id = mt.treatment_id
JOIN Medicine m ON mt.medicine_id = m.id
WHERE m.medicine_name = 'Medicine_1';
```

```sql
SELECT m.method_name, mt.implement_count
FROM Method m
JOIN Method_Treatment mt ON m.id = mt.method_id
JOIN Treatment t ON mt.treatment_id = t.id
WHERE t.treatment_header = 'Method_1';
```

```sql
SELECT name, surname, treatment_status
FROM Client
WHERE treatment_status = 1;
```

```sql
DELETE FROM Treatment
WHERE id = 101;
```

```sql
UPDATE Medicine
SET stock = stock - 10
WHERE medicine_name = 'Ibuprofen';
```

```sql
SELECT u.unit_name, SUM(m.stock) AS total_stock
FROM Medicine m
JOIN Unit u ON m.unit_id = u.id
GROUP BY u.unit_name;
```

```
SELECT t.treatment_header, t.diagnosis_header, t.diagnosis
FROM Treatment t;
```

```
SELECT name, surname, email, phone
FROM Client;
```

```
SELECT c.name, c.surname, t.treatment_header
FROM Client c
JOIN Treatment t ON c.id = t.client_id
JOIN Method_Treatment mt ON t.id = mt.treatment_id
JOIN Method m ON mt.method_id = m.id
WHERE m.method_name = 'Method_10';
```

```
SELECT treatment_id, COUNT(DISTINCT medicine_id)
FROM Medicine_Treatment
GROUP BY treatment_id
HAVING COUNT(DISTINCT medicine_id) >= 3;
```

```
DELETE FROM Medicine
WHERE medicine_name = 'Medicine_5';
```

```
SELECT t.treatment_header, COUNT(mt.method_id) AS method_count
FROM Treatment t
LEFT JOIN Method_Treatment mt ON t.id = mt.treatment_id
GROUP BY t.treatment_header;
```