# Bilkent University

Computer Engineering

CS 353 - 03

## Project Design Report

Elnur Aliyev - 21600421

Imran Hajiyev - 21503442

Tunar Mahmudov -21603114

# National Judiciary Informatics System

https://windrunner21.github.io/CS353-Databases/

## Table of Contents

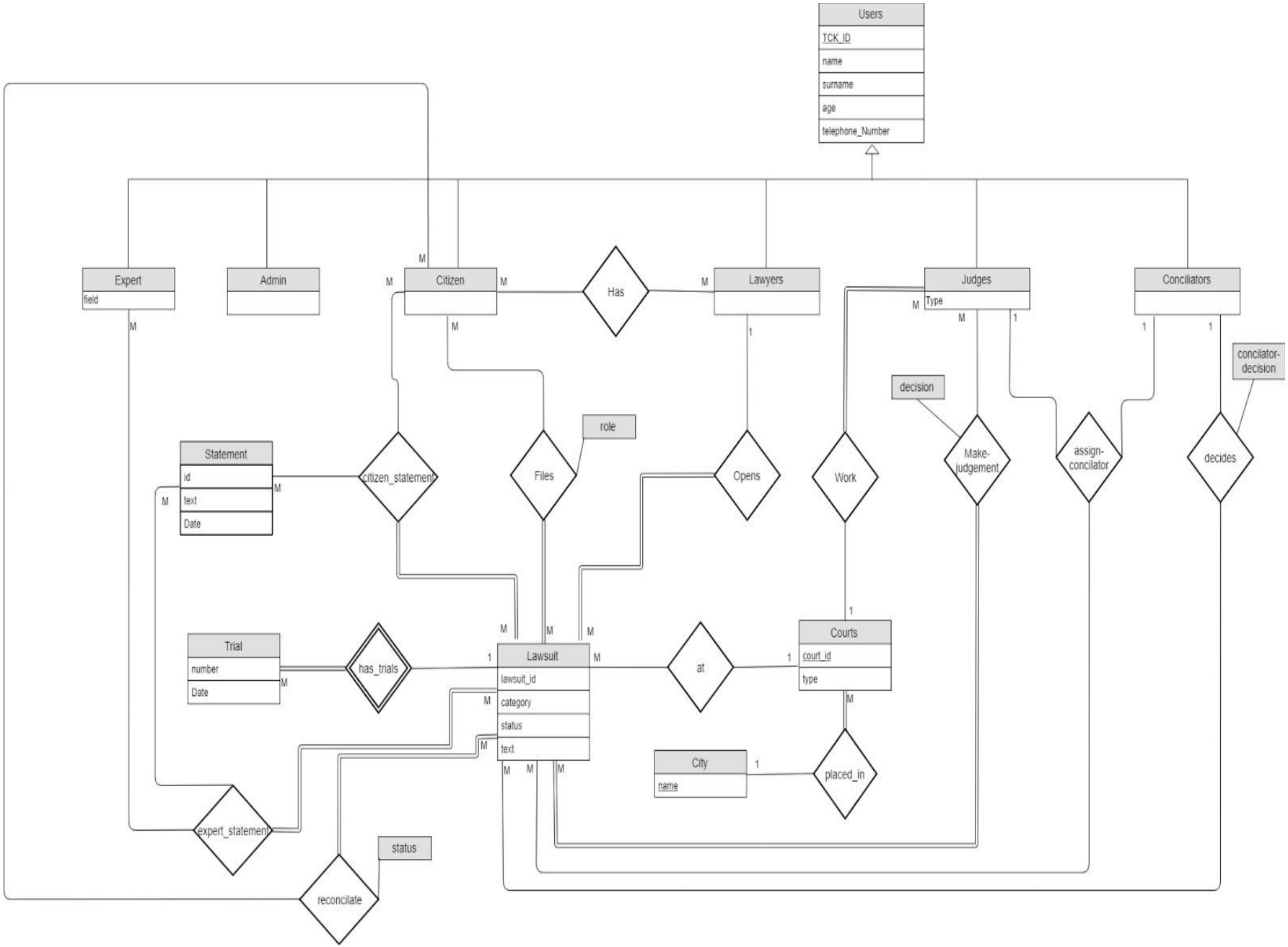
## 1. Revised E/R Model

According to teacher assistant's review, we revised or E/R model considering the feedback given in 5 points as follows:

1. TCK_ID attribute from Citizen entity was removed
2. New attributes to Lawsuit and Judge were added
3. Lawsuit's participation in Opens relationship was changed to total
4. Lawsuit's participation in Files relationship was changed to total.
5. Lawsuit's participation in Make_Judgement relationship was changed to total.
6. Lawsuit's participation in "At" relationship was changed to total.
7. Weak relationship between Court and City was removed and was set to normal one.
8. Multiplicity of Judges and Conciliator in Assign_Conciliator was changed to 1.
9. Work relationship between Courts and Judges was added. Judge's participation in work was set to total.
10. Weak entity Trial was added.
11. Weak relationship has_trials was added between Lawsuit and Trial. Trials' participation was changed to total.

Following changes in our E/R Diagram were made:

1. Two foreign keys (toWhom, byWhom) were removed from Files relationship. They were replaced by one foreign key(TCK_ID).
2. user_id was replaced by TCK_ID in Users entity.
3. trial_id and Date attributes were added to Trial.
4. statement entity was added. Lawsuit's participation in has_statement was set to total, has_statement relationship has statement attribute.
5. relationship (citizen_statement) between statement, citizen and lawsuit was added. participation of Lawsuit was set to total.
6. new user type – expert was added. It has field attribute.
7. relationship(expert_statement) between statement, expert and lawsuit was added. participation of Lawsuit was set to total.
8. relationship(reconciliate) between Citizen and Lawsuit with additional attribute status was added. participation of Lawsuit was set to total.


Users
TCK_ID
name
surname
age
telephone_Number
Expert
field
Admin
Citizen
Has
Lawyers
Judges
Type
Conciliators
Statement
id
text
Date
citizen_statement
Files
role
Opens
Work
decision
Make-judgement
assign-concilator
decides
conciator-decision
Trial
number
Date
has_trials
Lawsuit
lawsuit_id
category
status
text
at
Courts
court_id
type
City
name
placed_in
expert_statement
reconcilate
status
M
1

## 2. Relational Schemas

### 2.1. User

**Relational Model:**

User (tck_id, name, surname,age, telephone_number, password)

**Functional Dependencies:**

tck_id->name, surname,age, telephone_number, password

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE user
        (tck_id                 varchar(11) PRIMARY KEY,
        name                    varchar(20) NOT NULL,
        surname                 varchar(20) NOT NULL,
        age                     numeric(2,0),
        telephone_number        numeric(11,0)
        password                varchar(8) NOT NULL;
```

### 2.2. Citizen

**Relational Model:**

Citizen (tck_id, lawyer_tck_id, statement_id, status)

**Functional Dependencies:**

tck_id->lawyer_tck_id, statement_id, status

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE citizen
        (tck_id                     varchar(11) PRIMARY KEY,
        lawyer_tck_id               varchar(11),
        statement_id                varchar(11),
        status                      varchar(10) NOT NULL,
        FOREIGN KEY (lawyer_tck_id) REFERENCES lawyer (tck_id),
        FOREIGN KEY (tck_id)  REFERENCES user
        FOREIGN KEY (statement_id) REFERENCES statement (id),
        FOREIGN KEY (status) REFERENCES reconciliate);
```

### 2.3. Lawyer

**Relational Model:**

Lawyer (tck_id, lawsuit_id, citizen_tck_id)

**Functional Dependencies:**

tck_id->lawsuit_id, citizen_tck_id

**Candidate Keys:**

{ (tck_id, lawyer_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE lawyer
        (tck_id                     varchar(11) PRIMARY KEY,
        citizen_tck_id              varchar(11),
        lawsuit_id                  varchar(30),
        FOREIGN KEY (citizen_tck_id) REFERENCES citizen (tck_id),
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit,
        FOREIGN KEY (tck_id)  REFERENCES user);
```

### 2.4. Judge

**Relational Model:**

Judge (tck_id, court_id, conciliator_tck_id, lawsuit_id)

**Functional Dependencies:**

tck_id->court_id, conciliator_tck_id, lawsuit_id

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE judge
        (tck_id                     varchar(11) PRIMARY KEY,
        court_id                    varchar(20),
        conciliator_tck_id          varchar(11),
        lawsuit_id                  varchar(30),
        FOREIGN KEY (conciliator_tck_id) REFERENCES conciliator (tck_id),
        FOREIGN KEY (court_ id) REFERENCES court,
        FOREIGN KEY (tck_id)  REFERENCES user
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.5. Conciliator

**Relational Model:**

Conciliator (tck_id, lawsuit_id, judge_tck_id)

**Functional Dependencies:**

tck_id->lawsuit_id, judge_tck_id

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE conciliator
        (tck_id                     varchar(11) PRIMARY KEY,
        lawsuit_id                  varchar(30),
        judge_tck_id                varchar(11),
        FOREIGN KEY (judge_tck_id) REFERENCES judge (tck_id)
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit,
        FOREIGN KEY (tck_id)  REFERENCES user);
```

**2.6. Expert**

**Relational Model:**

Expert (tck_id, lawsuit_id, statement_id, field)

**Functional Dependencies:**

tck_id->lawsuit_id, statement_id, field

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE conciliator
        (tck_id                     varchar(11) PRIMARY KEY,
        lawsuit_id                  varchar(30),
        statement_id                varchar(11),
        field                       varchar(20),
        FOREIGN KEY (statement_id) REFERENCES statement (id),
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit,
        FOREIGN KEY (tck_id)  REFERENCES user);
```

### 2.7. Admin

**Relational Model:**

Admin (tck_id)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (tck_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE admin
        (tck_id                         varchar(11) PRIMARY KEY);
```

### 2.8. Trial

**Relational Model:**

Trial (trial_id, date, lawsuit_id)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (trial_id, date, lawsuit_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE trial
        (trial_id                       varchar(30),
        date                            varchar(10) NOT NULL,
        lawsuit_id                      varchar(30),
        PRIMARY KEY (trial_id, date, lawsuit_id),
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.9. Lawsuit

**Relational Model:**

Lawsuit (lawsuit_id, court_id, category, status, claims)

**Functional Dependencies:**

lawsuit_id->court_id, category, status, claims

**Candidate Keys:**

{ (lawsuit_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE lawsuit
        (lawsuit_id                 varchar(30) PRIMARY KEY,
        court_id                    varchar(20),
        category                    varchar(10) NOT NULL,
        status                      varchar(10) NOT NULL,
        claims                      varchar(1000) NOT NULL,
        FOREIGN KEY (court_id) REFERENCES court);
```

**2.10. Court**

**Relational Model:**

Court (court_id, type, city_name)

**Functional Dependencies:**

court_id->type, city_name

**Candidate Keys:**

{ (court_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE court
        (court_id                   varchar(20) PRIMARY KEY,
        type                        varchar(10) NOT NULL,
        city_name                   varchar(15),
        FOREIGN KEY (city_name) REFERENCES city);
```

### 2.11. City

**Relational Model:**

City (city_name)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (city_name) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE city
        (city_name                  varchar(15) PRIMARY KEY);
```

### 2.12. Statement

**Relational Model:**

Statement (statement_id, text, date, lawsuit_id)

**Functional Dependencies:**

statement_id->description, date, lawsuit_id

**Candidate Keys:**

{ (statement_id) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE statement
        (statement_id           varchar(11) PRIMARY KEY,
        text                    varchar(100) NOT NULL,
        date                    varchar(10) NOT NULL,
        lawsuit_id              varchar(30),
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.13. File

**Relational Model:**

file (tck_id, lawsuit_id, role)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (tck_id, lawsuit_id, role) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE file
        (tck_id                     varchar(11),
        lawsuit_id                  varchar(30),
        role                        varchar(30),
        PRIMARY KEY (tck_id, lawsuit_id, role),
        FOREIGN KEY (tck_id) REFERENCES citizen,
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.14. Make Judgment

**Relational Model:**

make_judgment (tck_id, lawsuit_id, decision)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (tck_id, lawsuit_id, decision) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE make_judgment
        (tck_id                     varchar(11),
        lawsuit_id                  varchar(30),
        decision                    varchar(10),
        PRIMARY KEY (tck_id, lawsuit_id, decision),
        FOREIGN KEY (tck_id) REFERENCES judge,
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.15. Decides

**Relational Model:**

decides (<u>tck_id, lawsuit_id, conciliator_decision</u>)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (tck_id, lawsuit_id, conciliator_decision) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE decides
        (tck_id                     varchar(11),
        lawsuit_id                  varchar(30),
        conciliator_decision        varchar(50),
        PRIMARY KEY (tck_id, lawsuit_id, conciliator_decision),
        FOREIGN KEY (tck_id) REFERENCES conciliator,
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

### 2.16. Reconciliate

**Relational Model:**

reconciliate (tck_id, lawsuit_id, status)

**Functional Dependencies:**

No dependencies.

**Candidate Keys:**

{ (tck_id, lawsuit_id, status) }

**Normal Form:**

BCNF

**Table Definition:**

```
CREATE TABLE decides
        (tck_id                 varchar(11),
        lawsuit_id              varchar(30),
        status                  varchar(10),
        PRIMARY KEY (tck_id, lawsuit_id, status),
        FOREIGN KEY (tck_id) REFERENCES citizen,
        FOREIGN KEY (lawsuit_id) REFERENCES lawsuit);
```

## 3. Functional Dependencies and Normalization of Tables

We followed good design principles and together with the feedback from the TA about our proposal report, we designed our database relations in the Boyce-Codd Normal Form. As it is BCNF, there is no data redundancy. Therefore, no further decomposition is required.

Boyce-Codd Normal Form is specified for each relation in the Relational Schemas Section above.

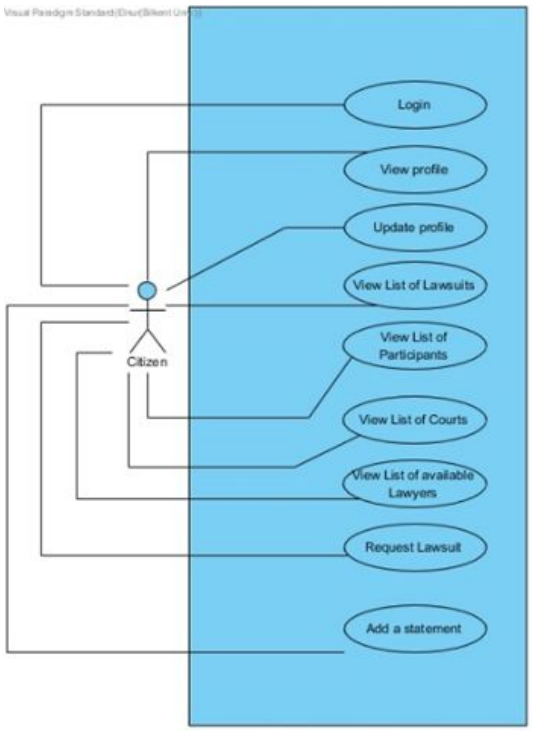
# 4. Functional Components

## 4.1. Use Cases/ Scenarios

In National Judiciary System 5 types of users exist. Admin, Citizen, Judge, Lawyer and Conciliator. The role of each user type has similarities. To use the system users must register at syste or login if they already have. Each user has its own limitation and availabilities.

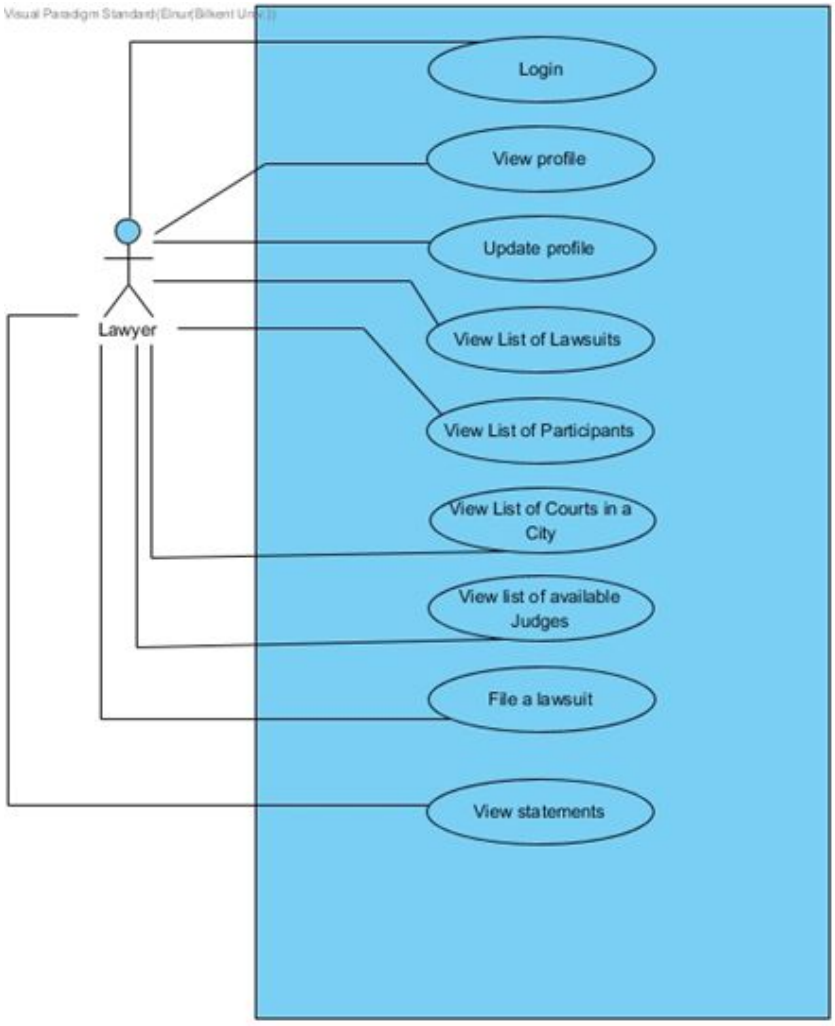
**Citizen:**

- Citizen can login to system with TCK_ID identity number and password.
- Citizen can access his profile page. He/she can look at his TCK identity number, name, age, mobile number.
- Citizen can change his/her password, age, mobile number.
- Citizen can access to the list of lawsuits in which citizen is involved.
- Citizen can view the list of all participants of a particular lawsuit and view their profiles. Citizen can view a lawyer, judge and conciliator of a particular lawsuit.
- Citizen can request a lawsuit to open. He can just select the court and a system will assign a lawyer automatically, or can select a particular lawyer.
- Citizen can view the list of courts in a particular city.
- Citizen can view the list of available lawyers in a particular court.
- Citizen can add a statement



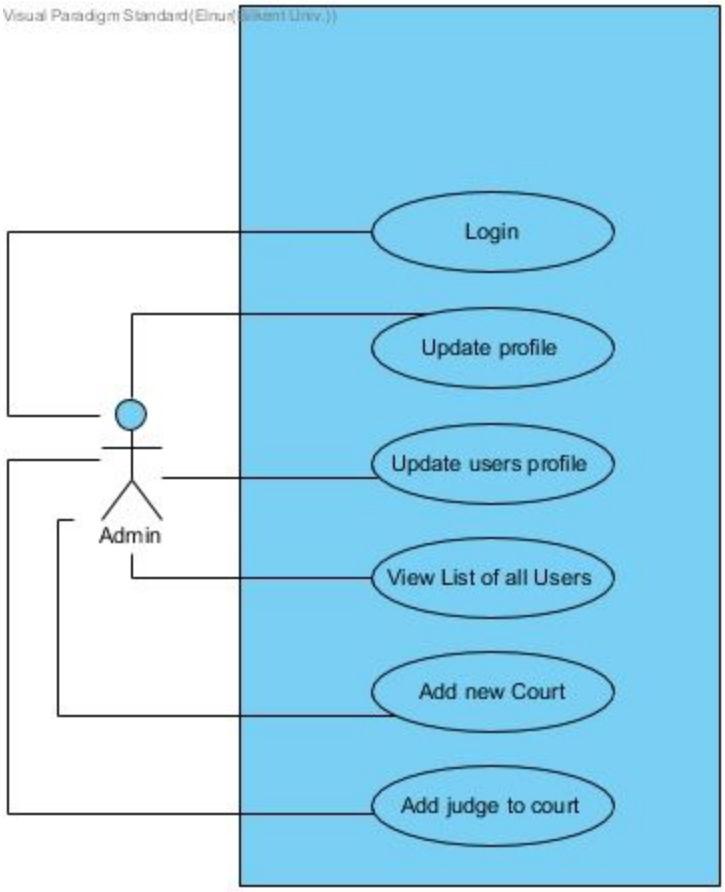
**Lawyer:**

- Lawyer can login to system with TCK_ID identity number and password.
- Lawyer can access his profile page. He/she can look at his TCK identity number, name, age, mobile number.
- Lawyer can change his/her password, age, mobile number.
- Lawyer can access to the list of lawsuits which they opened.
- Lawyer can view the list of all participants of a particular lawsuit and view their profiles.
- Lawyer can file a lawsuit. He can select the court in which process will proceed.
- Lawyer can view the list of courts in a particular city.
- Lawyer can view the list of available judges in a particular court.
- Lawyer can view the list of statements for a particular lawsuit



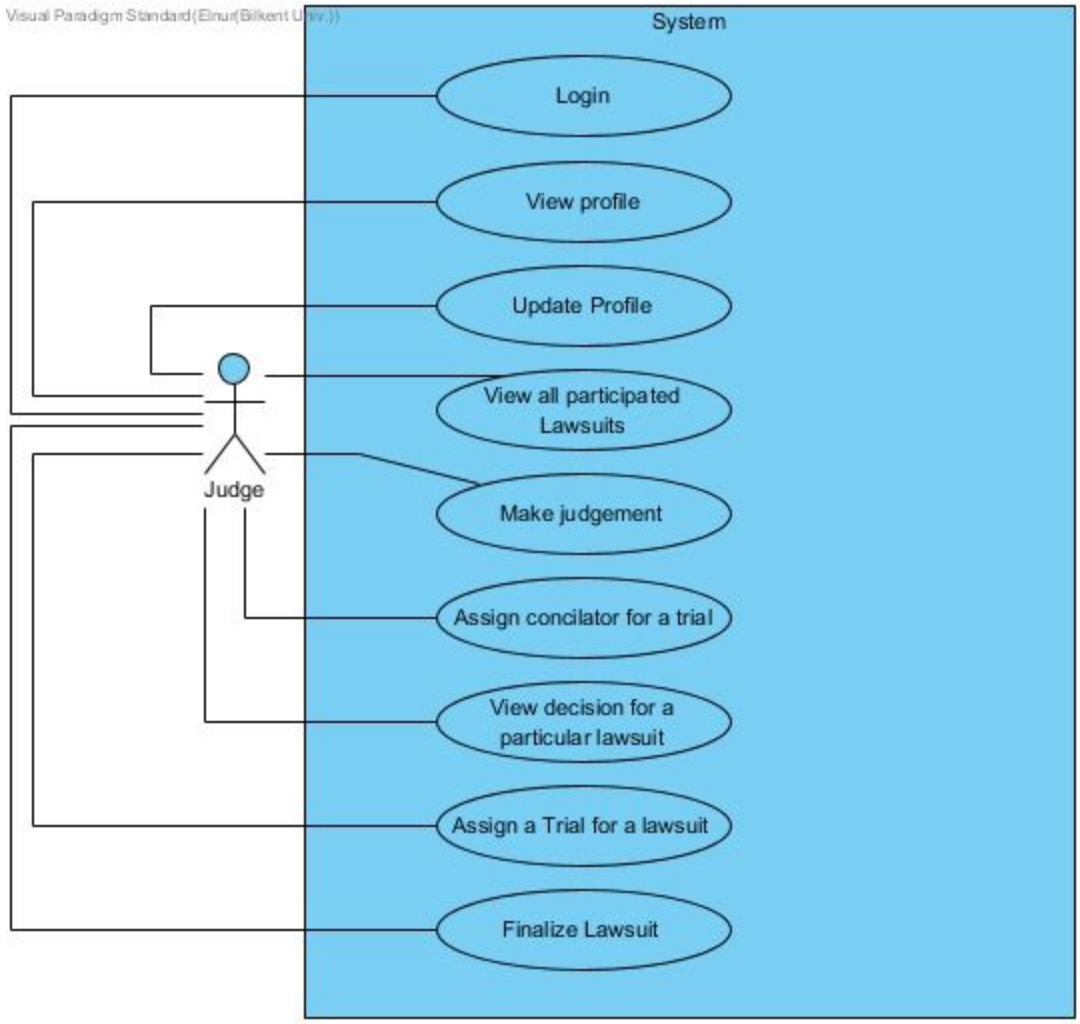
**Admin:**

- Admin can login to system with TCK_ID identity number and password.
- Admin can change his/her password
- Admin can update profiles of other users
- Admin can view the list of all users.
- Admin can add new court.
- Admin can add judge to court.



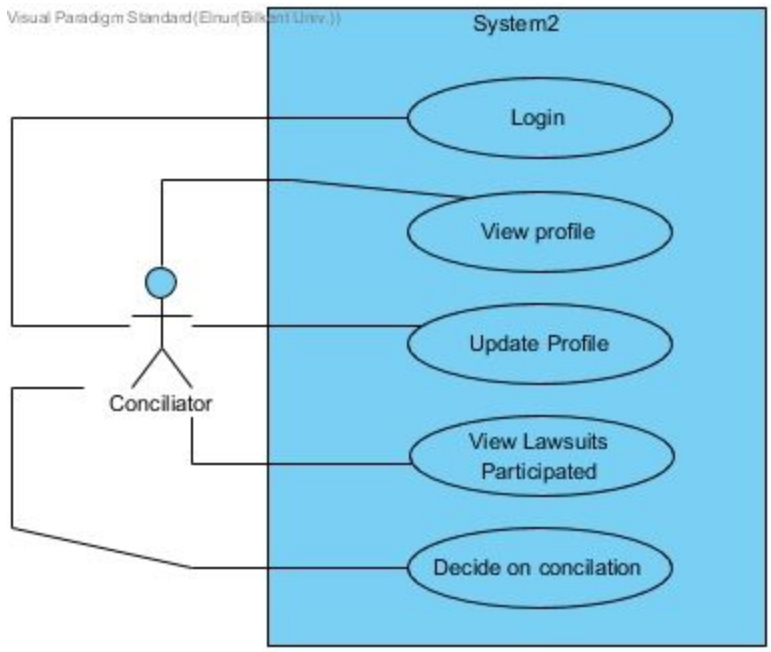
**Judge:**

- Judge can login to system with TCK_ID identity number and password.
- Judge can access his profile page. He/she can look at his TCK identity number, name, age, mobile number and type.
- Judge can change his/her password, age, mobile number and type.
- Judge can access to the list of lawsuits which he judges.
- Judge can make a judgement and give a decision.
- Judge can assign conciliator.
- Judge can view decision for a particular lawsuit.
- Judge can assign a trial for a lawsuit.
- Judge can finalize a lawsuit.



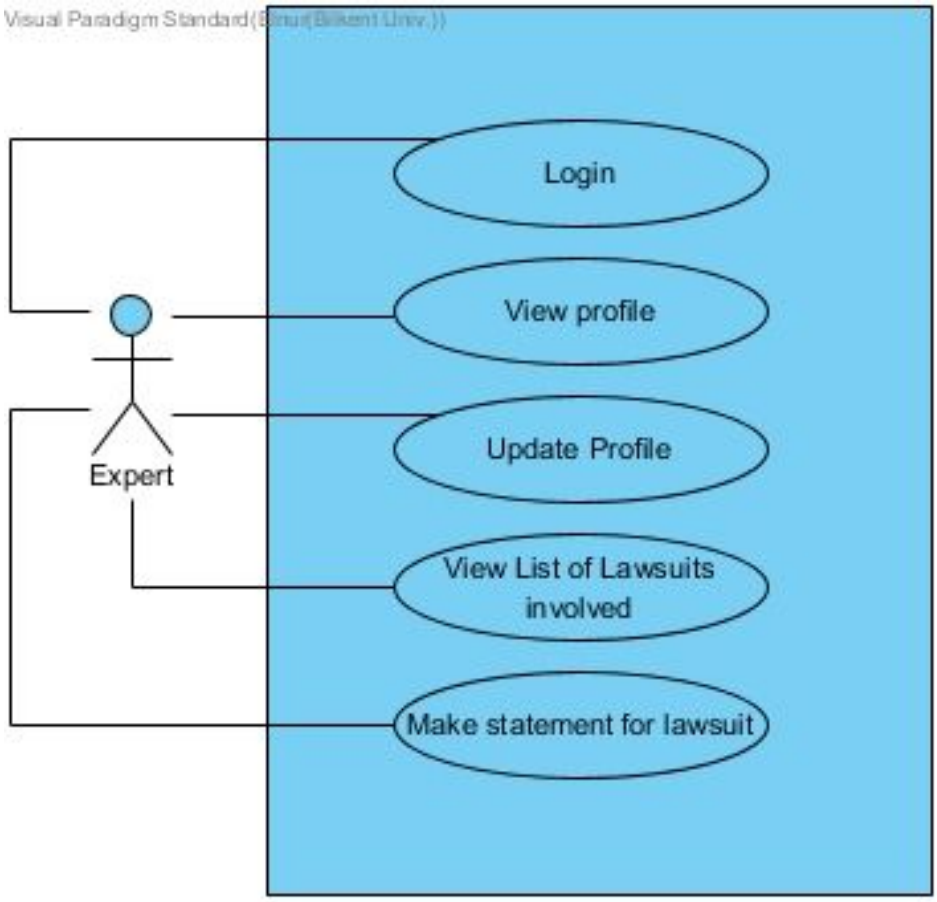
**Conciliator:**

- Conciliator can login to system with TCK_ID identity number and password.
- Conciliator can access his profile page. He/she can look at his TCK identity number, name, age mobile number and type.
- Conciliator can change his/her password, age, mobile number and type.
- Conciliator can access to the list of lawsuits which he concilates.
- Conciliator can make a judgement and give a decision.



**Expert:**

- Expert can login to system with TCK_ID identity number and password.
- Expert can access his profile page. He/she can look at his TCK identity number, name, age, mobile number and type.
- Expert can change his/her password, age, mobile number and type.
- Expert can view list of Lawsuits in which he/she involved
- Expert can make a statement for a lawsuit which he/she involved.



# 5. User Interface Design and Corresponding SQL Statements

## 5.1 Welcome Screen

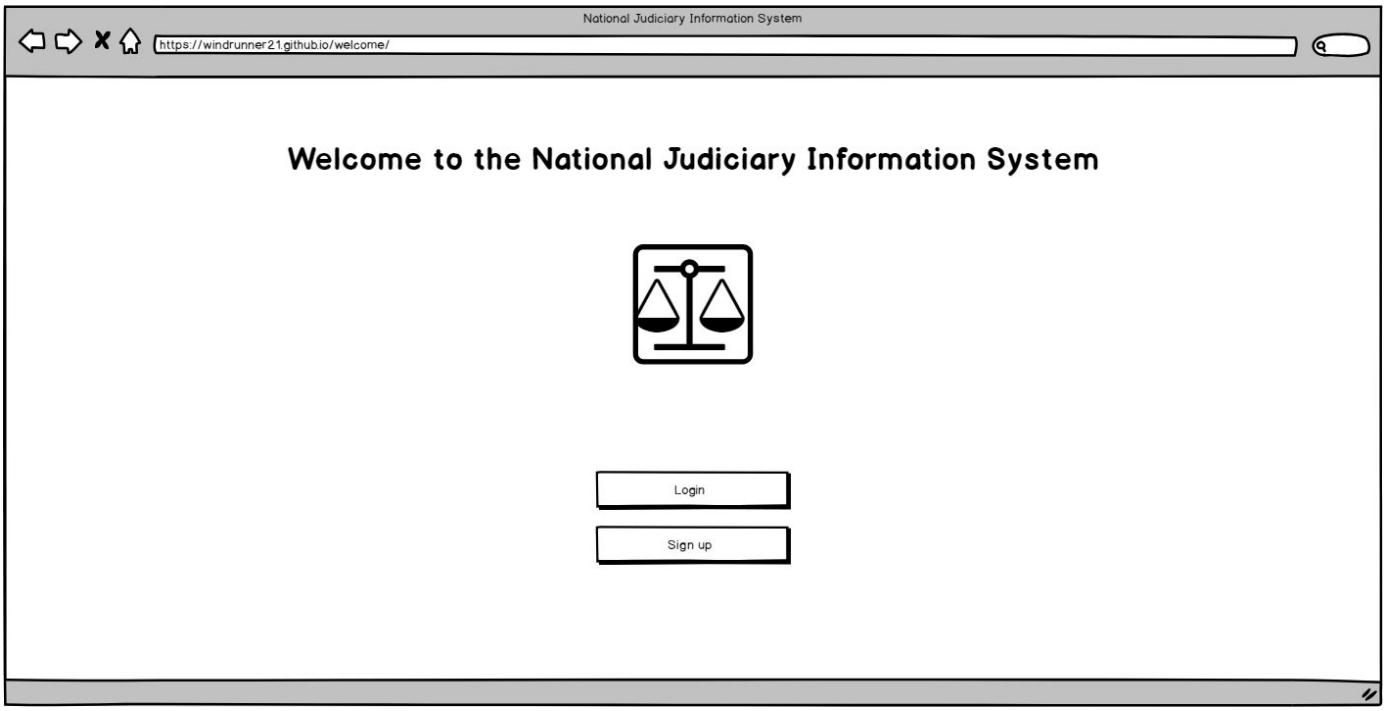



Figure 1: Welcome Screen

- Inputs: no input
- Process: This is the welcome screen of our system. User will sign up unless he/she already has an account. If the user has an account, he/she will be prompted to the login page
- SQL statement: no statement here

## 5.2 Sign-up Screen

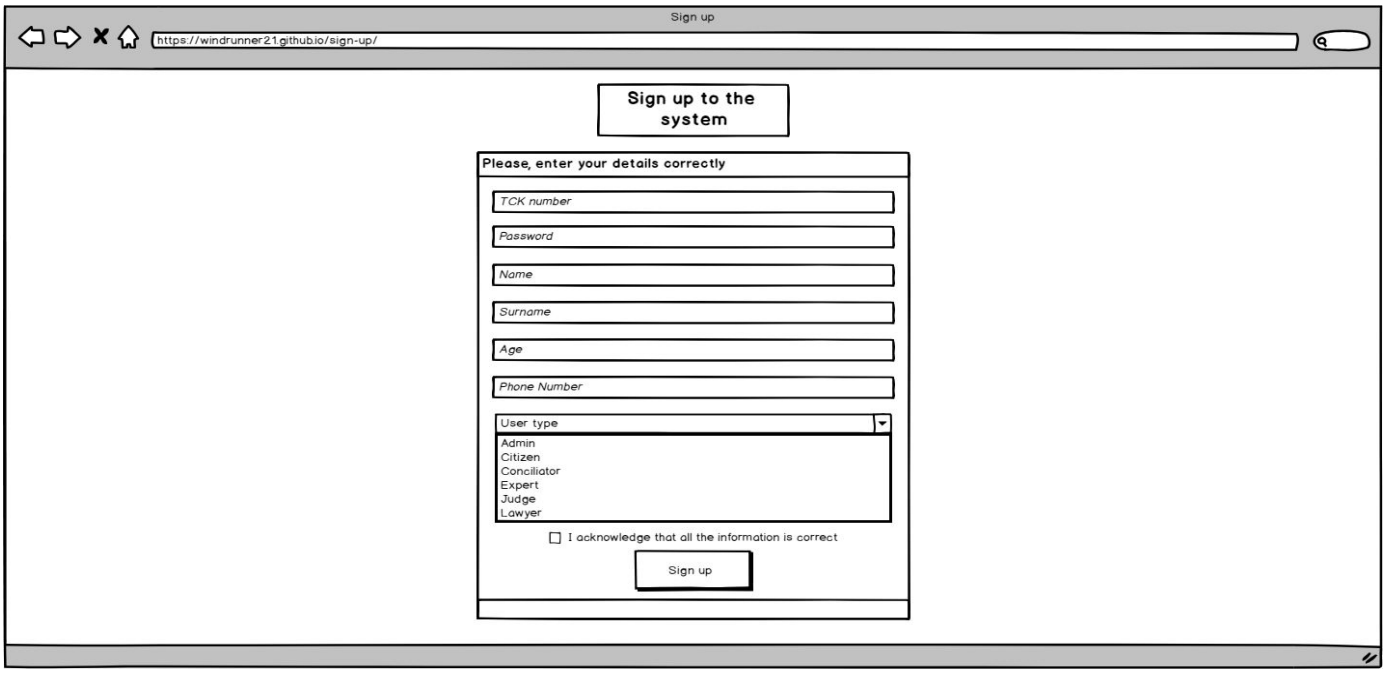



Figure 2: Sign up screen for citizens

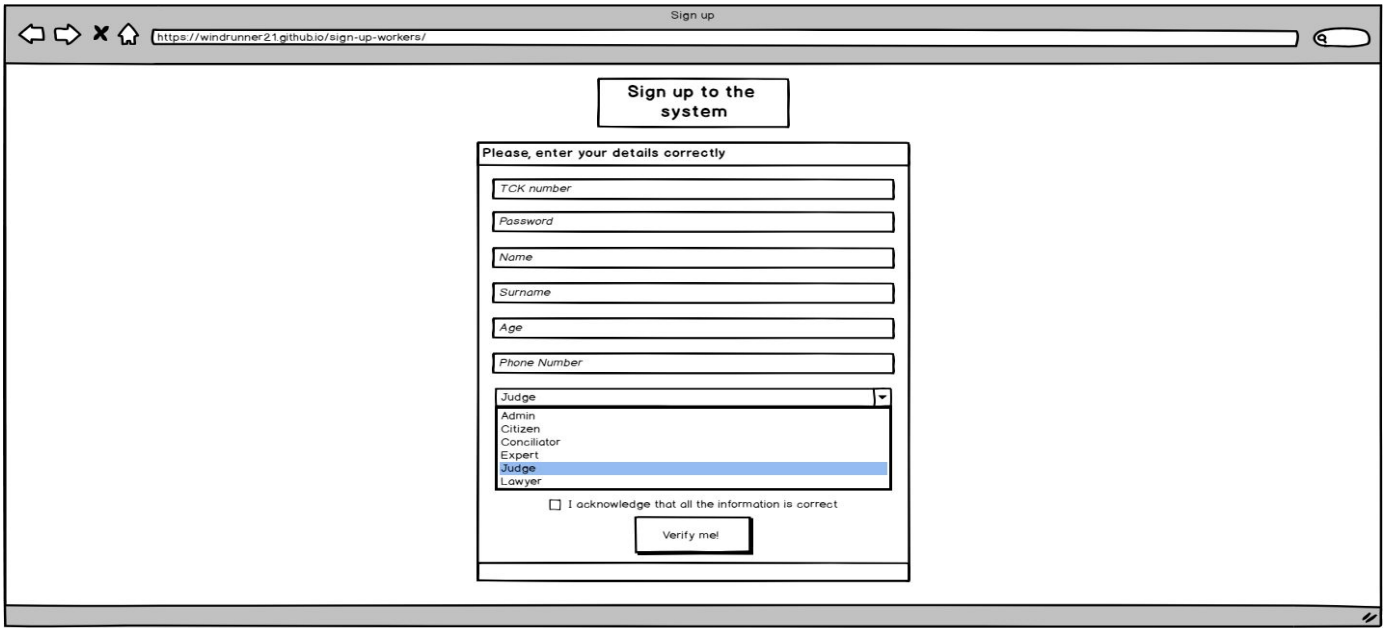



Figure 3:Sign up screen for workers of Judiciary

- **Inputs**: @TCK-id, @password, @name, @surname, @phone-number, @age, @user-type

- **Process**: The user will enter primarily information about himself such as TCK number, password, name, surname, phone number, age and will select the desired user type. If he has a profession of Judiciary, he will be prompted to select one of judge, lawyer, expert, conciliator, admin user types. Then as Figure 3 suggests, the system admin will verify the user and yield him to the homepage. Else if he is an ordinary citizen, he will select citizen user type showing as Figure 2

- **SQL statements:**
  - Adding a new user:

    insert into Citizen

    values (@TCK-id, @password, @name, @surname, @phone-number, @age)

### 5.3 Login Screen

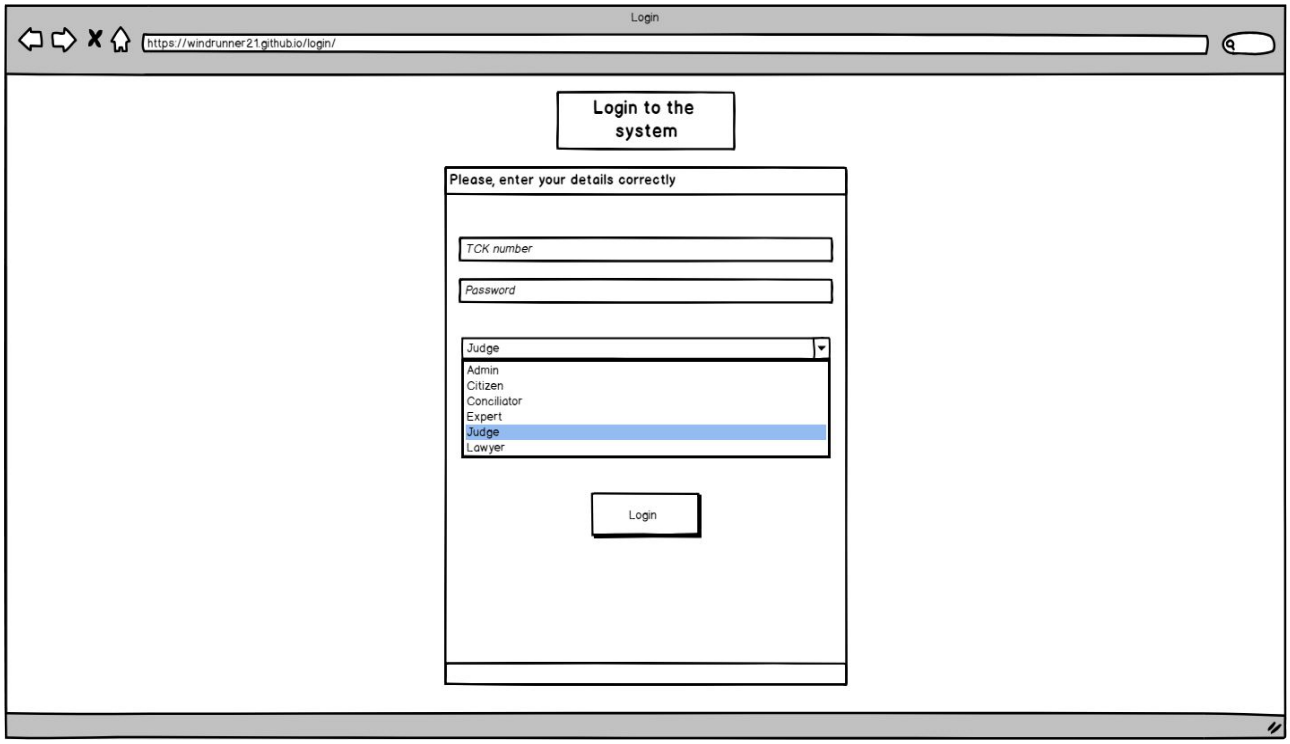



Figure 4: Login Screen

- **Inputs**: @TCK-id, @password, @user-type

- **Process**: The user will enter his TCK number and password that he has define in sign up page. Then he will choose his user type from the list

- **SQL statements:**
  - Login to the system:

    Select TCK-id, password

    From @user-type

    where TCK-id = @TCK-id and password = @password

## 5.4 Homepage for citizens and lawyers

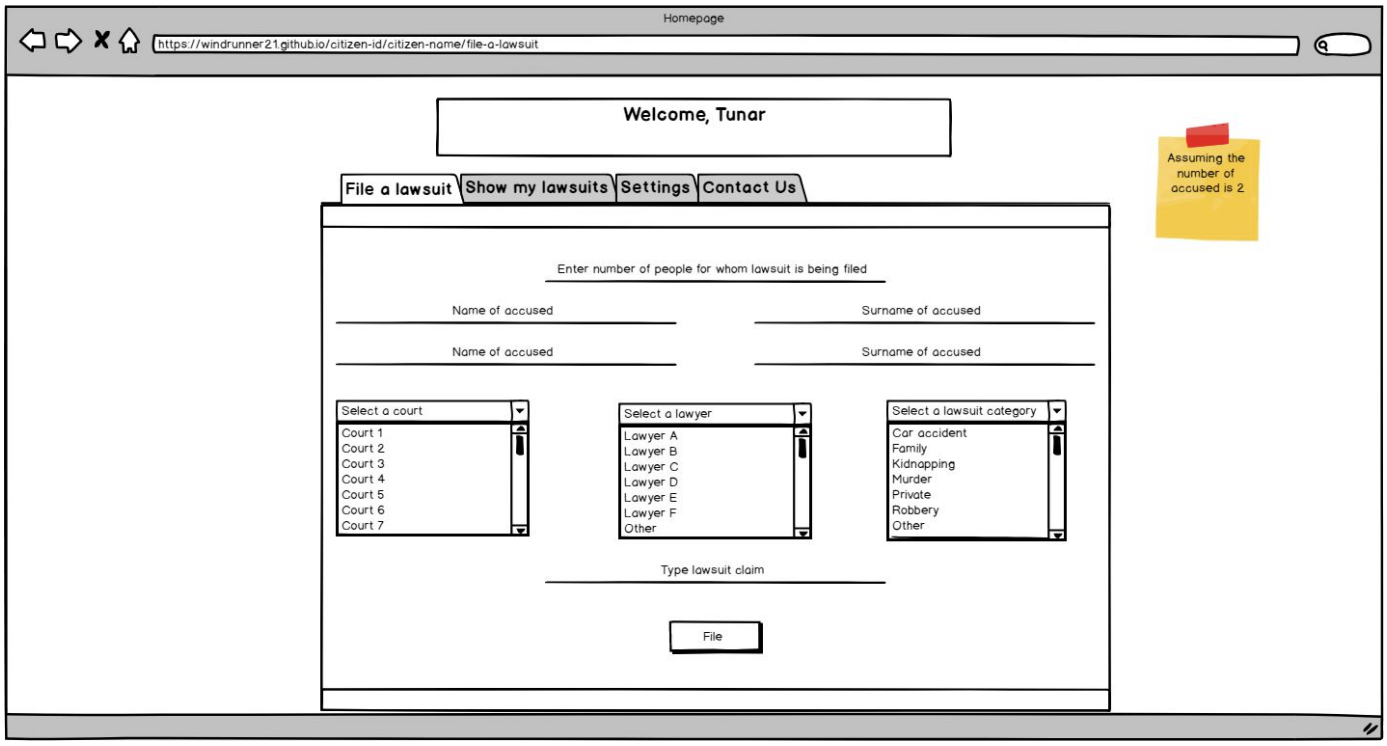



Figure 5 : File a lawsuit

- **Inputs**: @numOfAccused, @nameOfAccused, @surnameOfAccused, @court, @lawyer @lawsuitCategory, @lawsuitClaim

- **Process**: In order to file a lawsuit for a specific person or people, the user will be asked to enter the number of the accused people, then their names and surnames as well. Furthermore, he will select the desired court, lawyer and lawsuit category. At the end the user will write his claim and press file button to submit his lawsuit to the system

- **SQL statements:**
  - Add a lawsuit to the system:

```
Insert into Lawsuit
values(lawsuit-id, @lawsuitCategory, lawsuit-status, @lawsuitClaim)
```

Insert into Files

values(TCK-id, lawsuit-id, @numOfAccused, @nameOfAccused, @surnameOfAccused, @court, @lawyer)

### 5.5 Show Lawsuit for Citizens

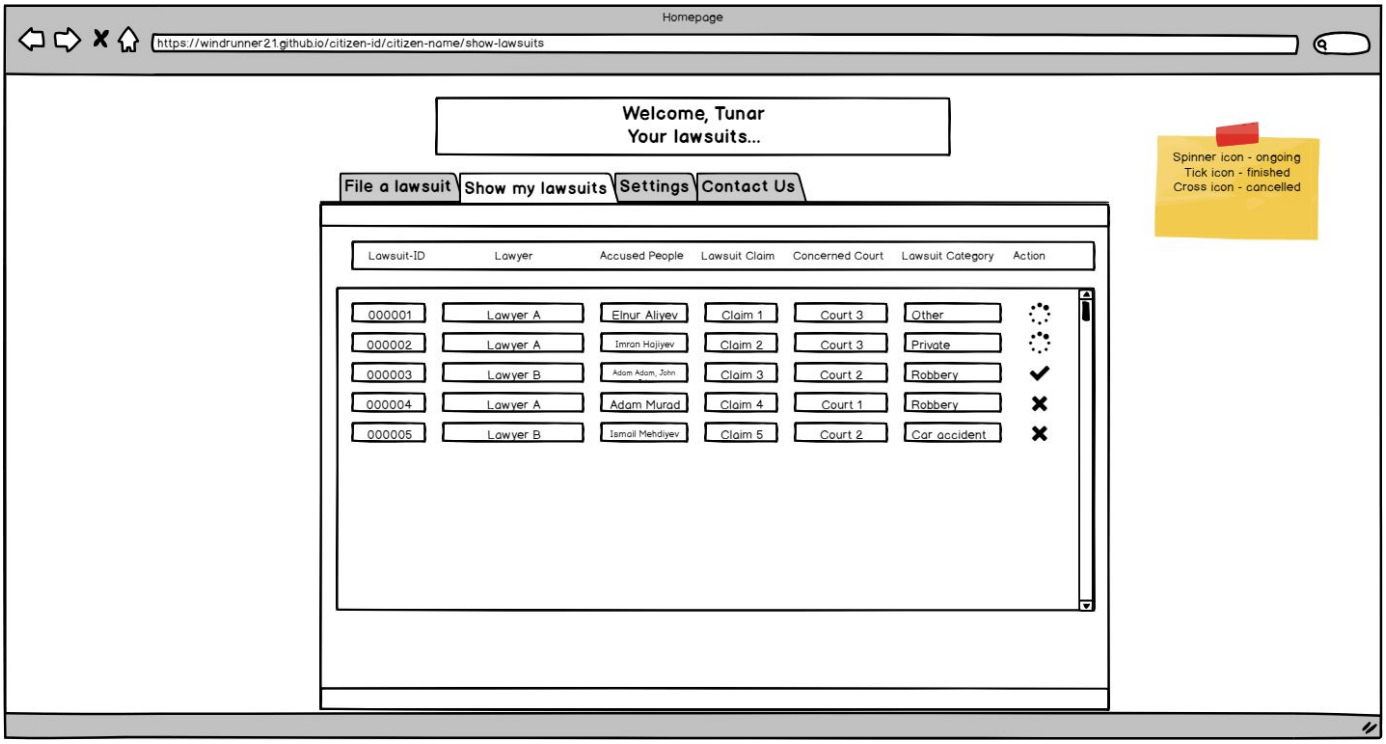



Figure 6 : Show citizen's lawsuits

- **Inputs**: no input

- **Process**: By pressing this tab of the window, the user can see his ongoing, cancelled or finished lawsuits in detail

➢ **SQL statements:**

❖ Show lawsuits of a specific citizen:

Select lawsuit-id, lawyer, accused-people, lawsuit-claim, court, lawsuit-category, status

From files natural join lawsuit

Where TCK-id = @TCK-id and password = @password

### 5.6 Settings Screen

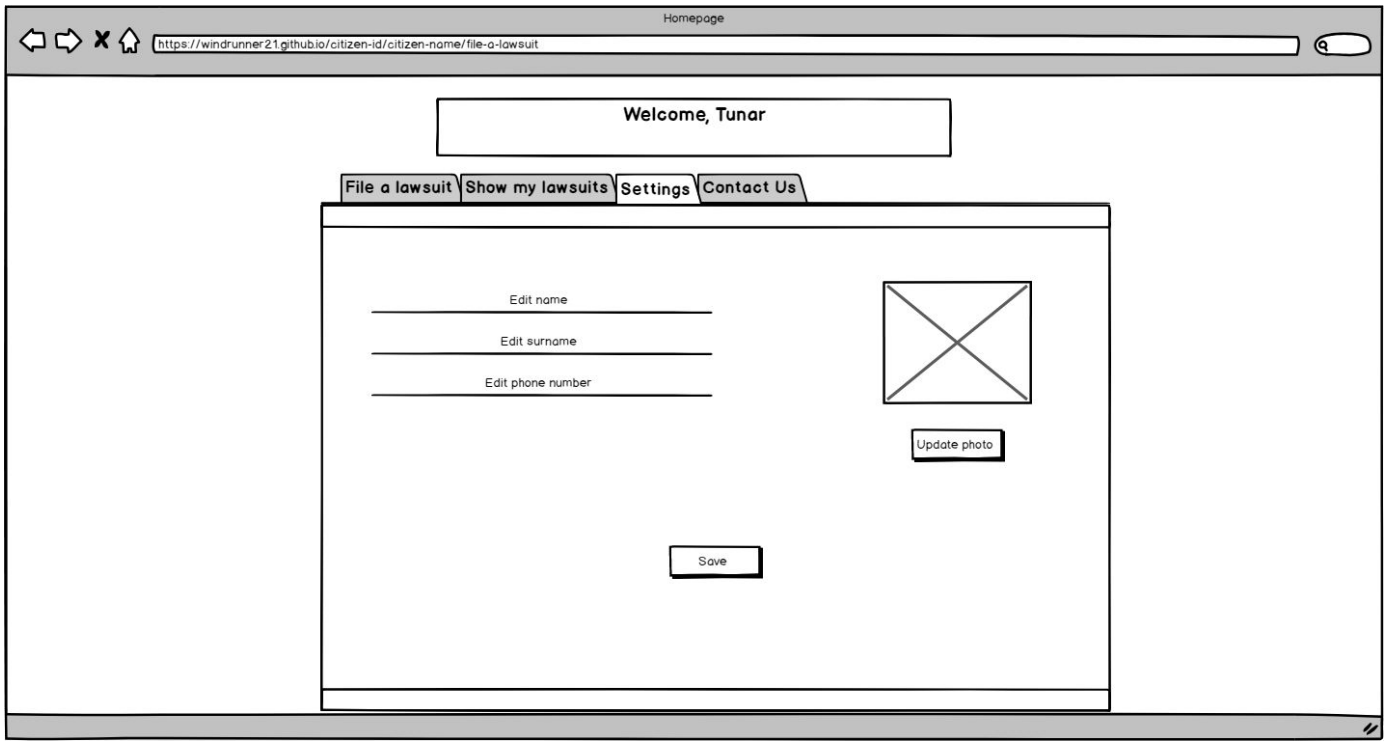



Figure 7: Settings

➢ **Inputs**: @name, @surname, @phone-number

➢ **Process**: This screen helps the user to change the information of himself

➢ **SQL statements:**

❖ Change name, surname and phone number of a citizen:

Update Citizen

Set name = @name, surname = @surname, phone-number = @phone-number

Where TCK-id = @TCK-id and password = @password

### 5.7 Contact Us Screen

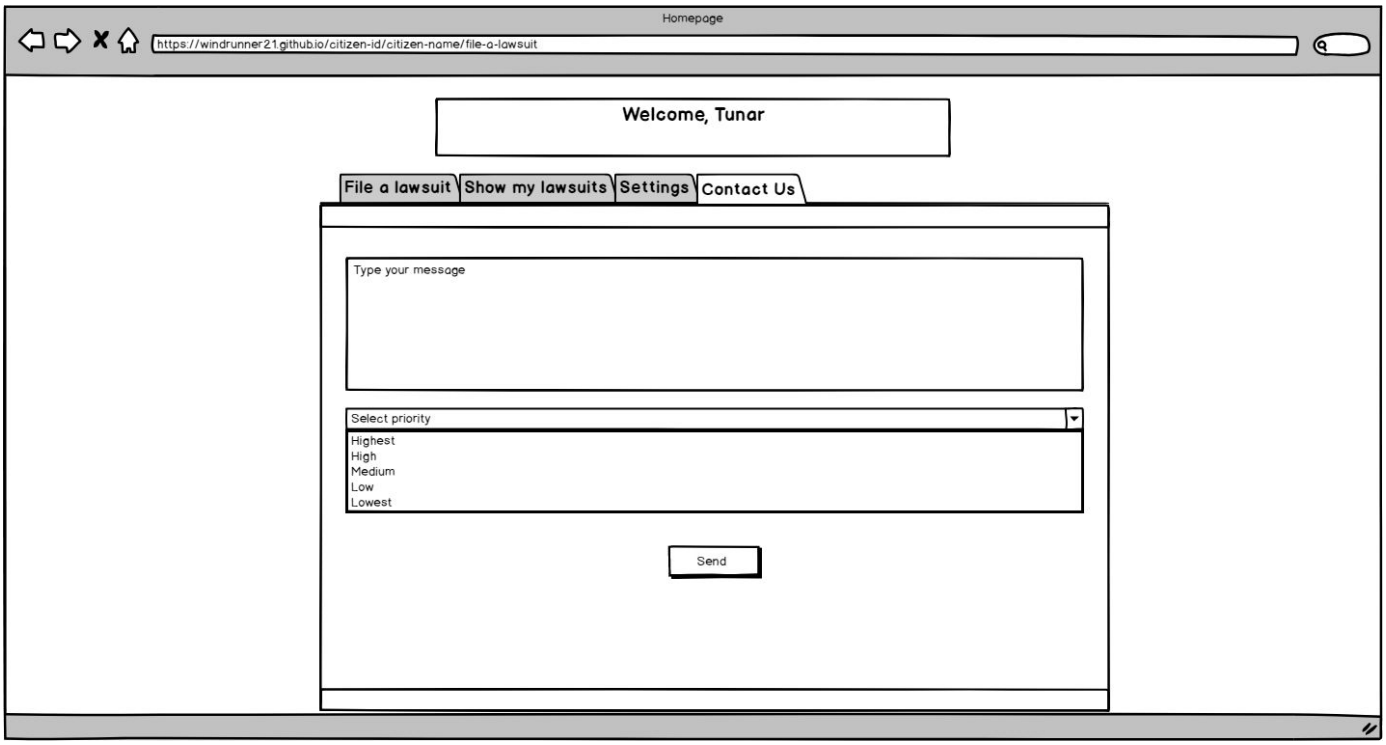



Figure 8 : Contact us

➢ **Inputs**: @message, @priority

➢ **Process**: User can send a message to the administrator using this tab. User messages WILL NOT be saved in our database and message system will be provided by mail services.

➢ **SQL statements:** no statement

### 5.8 Homepage for Judges

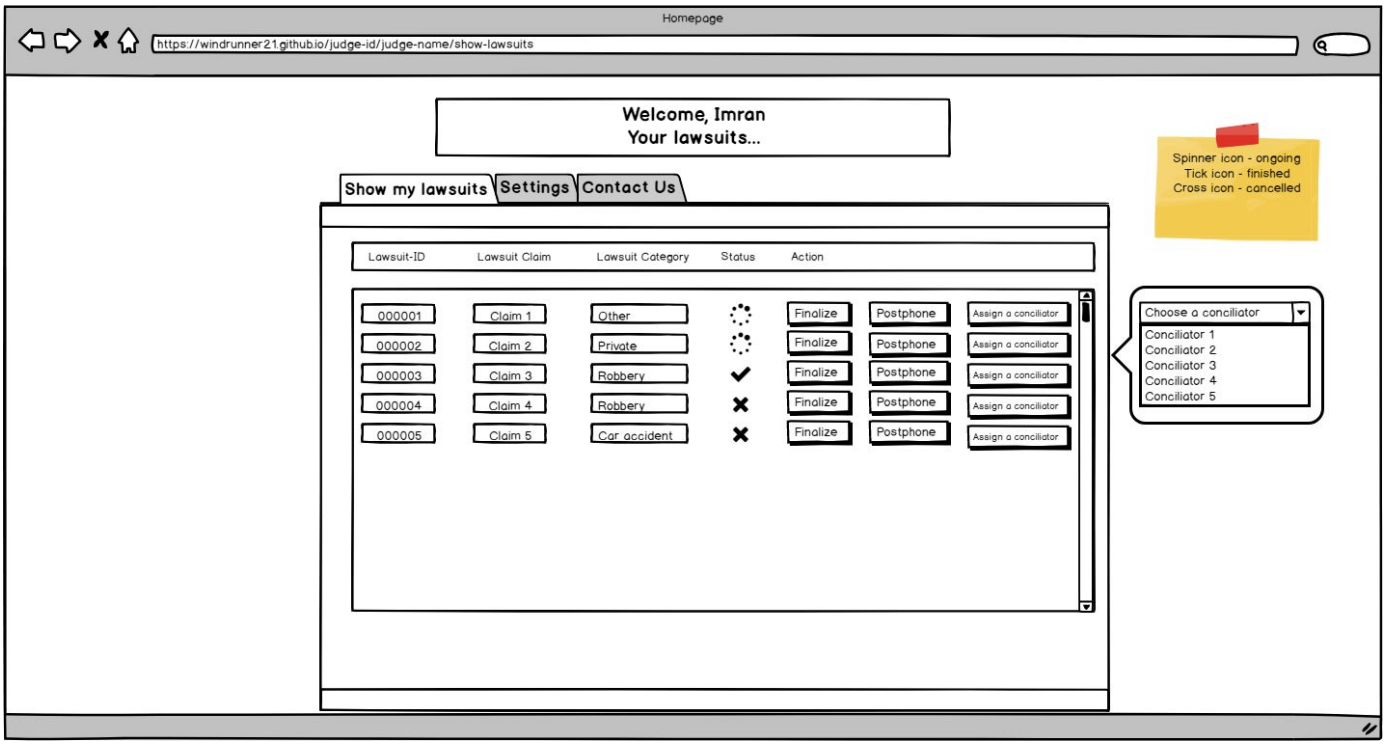



Figure 9 : Show Judge's lawsuits

- **Inputs**: no input

- **Process**: The judges of national judiciary system can see the lawsuits that are assigned/ongoing/finished to them using this tab. Furthermore, by the buttons next to each lawsuit, the judges can also finalize, postpone or assign conciliators to the cases as well

- **SQL statements:**
  - Show lawsuits of a judge:

```
Select lawsuit-id, lawsuit-claim, lawsuit-category, status
From lawsuit natural join Make-judgement
Where Make-judgement. TCK-id = @TCK-id
```

  - Assign a conciliator:

```
Insert into assign-conciliator
values(judge.TCK-id, conciliator.TCK-id, lawsuit-id)
```

  - Finalize a lawsuit:

```
Update Lawsuit
Set status = 'finalized'
Where lawsuit-id = (select lawsuit-id
                    Form Lawsuit natural join Make-judgement
                    Where Make-judgement.TCK-id = TCK-id)
```

  - Postpone a lawsuit:

```
Update Lawsuit
Set status = 'postponed'
Where lawsuit-id = (select lawsuit-id
                    Form Lawsuit natural join Make-judgement
                    Where Make-judgement.TCK-id = TCK-id)
```

## 5.9 Homepage for Conciliators

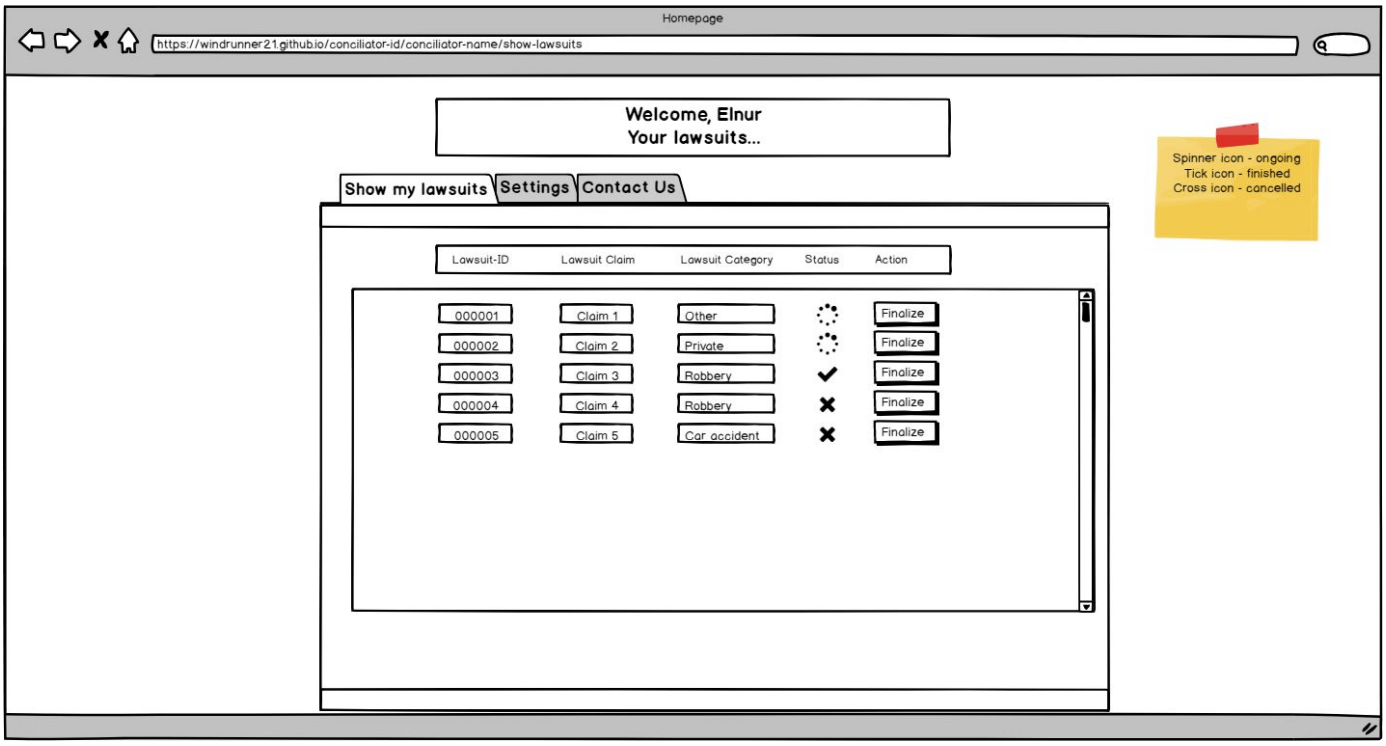



Figure 10: Lawsuits of a conciliator

- **Inputs**: no input

- **Process**: The conciliators of the national judiciary system can see the lawsuits that are assigned to them using this tab. Furthermore they can also finalize the cases as well

- **SQL statements:**
  - Show lawsuits of a conciliator:

    Select lawsuit-id, lawsuit-claim, lawsuit-category, status

    From lawsuit natural join assign-conciliator

    Where assign-conciliator. TCK-id = @TCK-id

  - Finalize a lawsuit:

    Update Lawsuit

    Set status = ‘finalized’

```
Where lawsuit-id = (select lawsuit-id
                    Form Lawsuit natural join assign-conciliator
                    Where assign-conciliator .TCK-id = TCK-id)
```

# 6. Advanced Database Components

## 6.1. Views

We wanted to limit the access of users to

## 6.2 Stored Procedures

## 6.3. Reports

We will generate reports every month on the number of lawsuits being completed (finalized) as well as the number of lawsuits being canceled and being on hold (ongoing).

### 6.3.1. The number of completed lawsuits.

```
select count (lawsuit.status)
from lawsuit
where lawsuit.status = “Finalized”
```

### 6.3.2. The number of canceled lawsuits.

```
select count (lawsuit.status)
from lawsuit
where lawsuit.status = “Canceled”
```

### 6.3.3. The number of ongoing lawsuits.

```
select count (lawsuit.status)
from lawsuit
where lawsuit.status = “Ongoing”
```

## 6.4. Triggers

- When a judge presses the *assign conciliator* button, it will add judge_id and lawsuit_id to the specified conciliator tuple. The trigger will assign the tck_id of the specified conciliator to the specified judge based on the lawsuit_id and judge_id, as well as assign the decision of the conciliator to the judge’s decision.
- When the lawsuit is canceled, trigger deletes the lawsuit related entries from every table that had a tuple with the canceled lawsuit_id.
- When a citizen files a lawsuit and chooses a specific court, the trigger randomly assigns free judge to the lawsuit, based on the court_id.

### 6.5. Constraints

- User cannot login without creating an account first.
- Non-citizen accounts cannot complete registration without the approval of the system admin.
- Non-citizen accounts will login using their TCKN and encrypted passwords generated from the admin side.
- Judges cannot assign a conciliator to the lawsuit without the approval of all the citizens involved in the lawsuit.
- Users cannot change their phone number and profile photos more than once in a month.
- Judge cannot postpone the lawsuit if it has been finalized.
- Judge cannot assign a conciliator if the lawsuit has been finalized.
- Judge cannot finalize the lawsuit if the conciliator has been assigned. In that case, only the conciliator can finalize the lawsuit.
- Citizen cannot choose a judge for the lawsuit they are filing.
- Citizen can file a lawsuit against any number of people.
- A group of citizens can file a lawsuit against another group of people.
- Lawsuits can be filed against the non-citizen person. In that case, non-citizen person will have to sign up as a citizen, and proceed to the lawsuit in a role of a citizen.
- Lawsuit cannot be filed without a claim from the victim side (citizen).
- Lawsuit can be filed with or without the lawyer.
- Lawsuit cannot be filed without a category.
- Lawsuit cannot be filed without a specified court.
- User cannot contact the support without specifying the priority of the message.
- Court assigns judge of needed type, who is available.
- Court cannot be added without specifying its type and the city it is located in.
- Only expert of the needed (correct) type can give the statement on the lawsuit claims.
- Lawsuit must have trials.

## 7. Implementation Plan

We decided to use the MySQL system for our relational database management system. MySQL is free, as it is open-source, easy to manage, as it has been in the market and usage for a long time and has good support. For the front-end and application part of our project, we decided to use React JavaScript library as it is a powerful tool in creating beautiful websites. Therefore, JavaScript is our implementation language for the application and front-end parts.