CS353 DESIGN REPORT

National Judiciary System

Spring 2019 Group 21

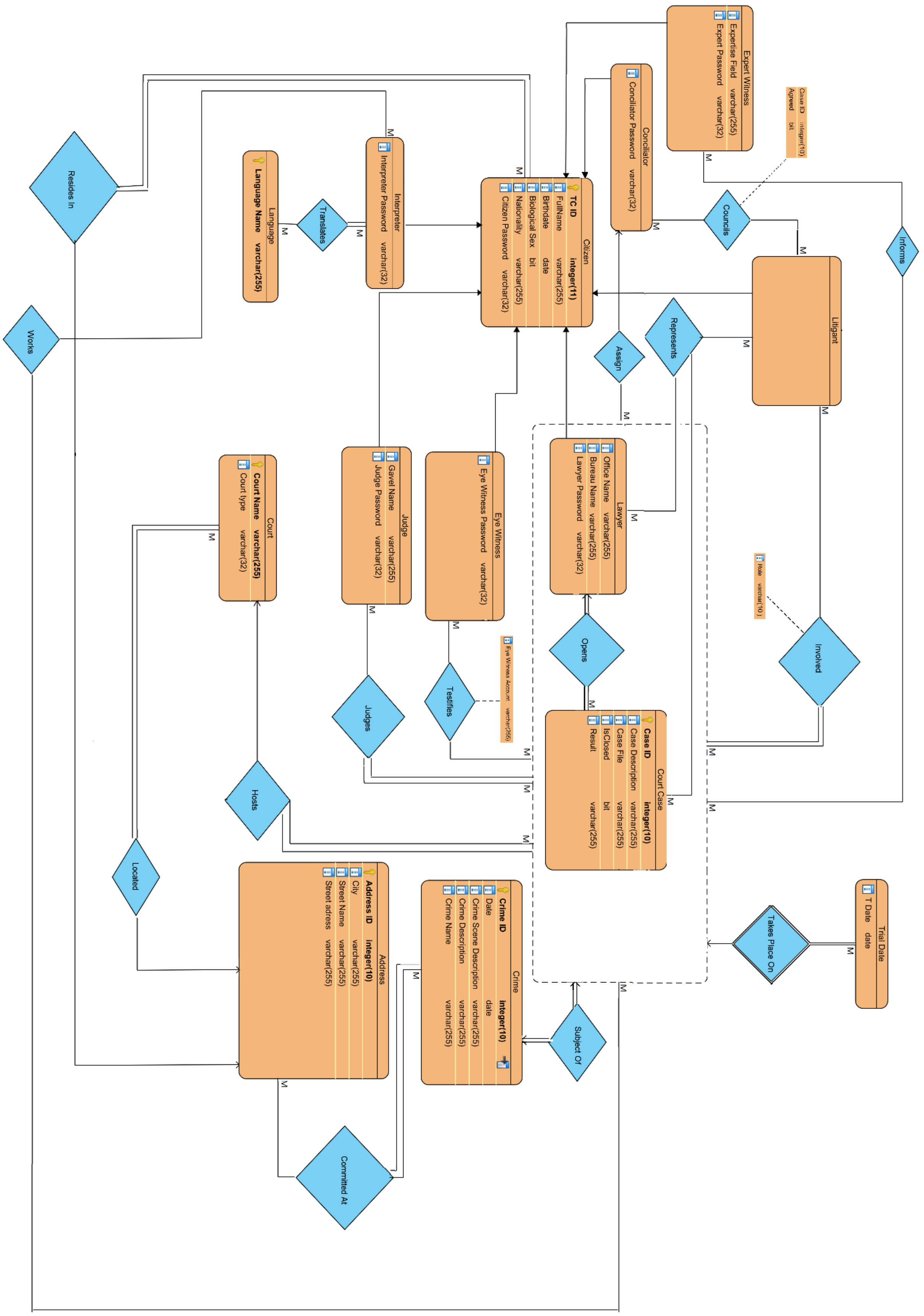
Mert Soydinç	21601321	Section 3
Faruk Oruç	21601844	Section 3
Ünsal Öztürk	21601565	Section 2
Doğa Oruç	2160202	Section 2

Teaching Assistant: Arif Usta

1 Table of Contents

[This space has been intentionally left blank]

1. Revision of E/R Model and E/R Diagram Revisions



The E/R diagram provided in the project proposal was revised according to the feedback. The initial changes made on the E/R diagram were about reducing the scope of the project, as it was deemed to be too large to be implemented. Therefore, the E/R diagram was rehauled. Two actors, the public prosecutor and the guardian, along with their corresponding relations – opens, and acts on behalf – were removed from the system. Crime type entity and is type of relation were also removed for simplification purposes. A similar approach was taken for the Court Type Relation and Court Type, and Court Type was made to be an attribute of the Court entity. To store trial dates for a particular court case, we introduced a new weak entity called trial date, which has a many to many relationship with a new aggregate entity that we propose, governing the court case and related entities. To increase dynamic interaction in the model, the lawyer entity now has a relationship between itself and Court Case, called Opens, and the three entries in the model form an aggregate relationship. The remaining actors now have relationships with this aggregate entity. Namely, the Judge, Testifies, Involved, Takes Place On, Assign, and Works relations now connect to this aggregated structure.

The represents relation was reworked, and it is now a ternary relationship between Litigant, Lawyer, and Court Case.

Another type of dynamic interaction between actors we introduced in the revised E/R model is regarding the assignment of a conciliator to a particular court case. Judges, through the aggregate court case entity, may now request a conciliator, via the "Assign" relation, which stores the conciliator to court case mapping. The Councils relation has also been reworked, which now has two extra attributes, Case ID in reference to the court case, and it also stores information about whether a party has accepted the terms offered by the conciliator.

The other major modification in the E/R model is the introduction of password attributes to all actors. Each person in the database will have access to a citizen account, and other legislative staff (e.g. judges and lawyers) will have access to their accounts via the password they set (in addition to their citizen accounts), and the information regarding these passwords will be stored in the database using a SHA256 hashing scheme. No password is a candidate key for any relation, since two distinct people may have the same password. The password field was added to all actors, i.e. to Expert Witness, Citizen, Lawyer, Eye Witness, Judge, and Interpreter.

The remaining changes regarding the introduction and removal of attributes are minor changes: explicit keys were removed from children entities of Citizen for the accuracy of E/R model notation. The entities Lawyer and Judge no longer have their Work ID attributes, and instead derive their key from the parent entity, Citizen. Lastly, the eye witness account attribute was moved to the Testifies relation.

2. Relational Model, Table Schemas, and Normalization

2.1 Address

Relational Model: Address(<u>Address_ID</u>, City, Street_Name, Street_Adress)

Functional Dependencies: Address_ID → City, Street_Name, Street_Adress

Candidate Key: {(Adress_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Address(
        Adress_ID INTEGER(10) NOT NULL,
        City VARCHAR(255) NOT NULL,
        Street_Name VARCHAR(255) NOT NULL,
        Street_Adress VARCHAR(255) NOT NULL,
        PRIMARY KEY(Address_ID)
);
```

2.2 Citizen

Relational Model: Citizen(<u>TC_ID</u>, FullName, Birthdate, Biological_Sex, Nationality ,Address_ID, Citizen_Password)

Foreign Key: Address_ID references Address

Functional Dependencies: TC_ID → FullName, Birthdate, Biological_Sex, Nationality, Address_ID, Citizen_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Citizen(
        TC_ID INTEGER(10) NOT NULL,
        FullName VARCHAR(255) NOT NULL,
         Birthdate DATE NOT NULL,
         Biological_Sex BIT NOT NULL,
        Nationality VARCHAR(255) NOT NULL,
        Adress_ID INTEGER(10),
        Citizen_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID)
        FOREIGN KEY (Address_ID) REFERENCES Address,
);
```

2.3 Litigant

Relational Model: Litigant(<u>TC_ID</u>)

Foreign Key: TC_ID references Citizen

Functional Dependencies: None

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Litigant(
        TC_ID INTEGER(10) NOT NULL,
        PRIMARY KEY(TC_ID)
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.4 Lawyer

Relational Model: Lawyer(<u>TC_ID</u>, Office_Name,Bureau_Name, Lawyer_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID → Office_Name,Bureau_Name, Lawyer_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Lawyer(
        TC_ID INTEGER(10) NOT NULL,
        Office_Name VARCHAR(255) NOT NULL,
        Bureau_Name VARCHAR(255) NOT NULL,
        Lawyer_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID)
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.5 Interpreter

Relational Model: Interpreter(<u>TC_ID</u>, Interpreter_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID → Interpreter_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Interpreter(
        TC_ID INTEGER(10) NOT NULL,
        Interpreter_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID),
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.6 Expert_Witness

Relational Model: Expert_Witness(<u>TC_ID</u>, Expertise_Field, Expert_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID →Expertise_Field, Expert_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Expert_Witness(
        TC_ID INTEGER(10) NOT NULL,
        Expertise_Field VARCHAR(255) NOT NULL,
        Expert_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID),
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.7 Eye_Witness

Relational Model: Eye_Witness(<u>TC_ID</u>, Eye_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID →Eye_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Eye_Witness(
        TC_ID INTEGER(10) NOT NULL,
        Eye_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID),
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.8 Judge

Relational Model: Judge(<u>TC_ID</u>, Gavel_Name, Judge_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID → Gavel_Name, Judge_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Judge(
        TC_ID INTEGER(10) NOT NULL,
        Gavel_Name VARCHAR(255) NOT NULL,
        Judge_Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID),
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.9 Conciliator

Relational Model: Conciliator(<u>TC_ID</u>,Conciliator_Password)

Foreign Key: TC_ID references Citizen

Functional Dependencies: TC_ID → Conciliator_Password

Candidate Key: {(TC_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Conciliator(
        TC_ID INTEGER(10) NOT NULL,
        Conciliator _Password VARCHAR(32) NOT NULL,
        PRIMARY KEY(TC_ID)
        FOREIGN KEY (TC_ID) REFERENCES Citizen
);
```

2.10 Language

Relational Model: Language(<u>Language_Name</u>)

Functional Dependencies: None

Candidate Keys: {(Language_Name)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Language(
        Language_Name VARCHAR(255) NOT NULL,
        PRIMARY KEY(Language_Name)
);
```

2.11 Court

Relational Model: Court(<u>Court_Name, </u>Court_Type, Adress_ID)

Foreign Key: Address_ID references Address

Functional Dependencies: Court_Name → Court_Type,Adress_ID

Candidate Key: {(Court_Name)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Court(
        Court_Name VARCHAR(255) NOT NULL,
        Adress_ID INTEGER(10) NOT NULL,
        Court_Type VARCHAR(32) NOT NULL,
        PRIMARY KEY(Court_Name),
        FOREIGN KEY (Address_ID) REFERENCES Address,
);
```

2.12 Court_Case

Relational Model:
Court_Case(<u>Case_ID</u>,Case_Description,Case_file,IsClosed,Result,Initiator_Lawyer,Court_Name, Conciliator_ID, Crime_ID)

Foreign Key: Initiator_Lawyer references Lawyer(TC_ID)

Foreign Key: Court_Name references Court

Foreign Key: Conciliator_ID references Conciliator(TC_ID)

Foreign Key: Crime_ID references Crime

Functional Dependencies: Case_ID → Case_Description, Case_file, IsClosed, Result, Initiator_Lawyer, Court_Name, Conciliator_ID, Crime_ID

Candidate Key: {(Court_Name)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Court_Case (

        Case_ID INTEGER(10) NOT NULL,

        Case_Description VARCHAR(255) NOT NULL,

        Case_file VARCHAR(255) NOT NULL,

        IsClosed bit NOT NULL,

        Result VARCHAR(255) NOT NULL,

        Initiator_Lawyer INTEGER(10) NOT NULL,

        Court_Name VARCHAR(255) NOT NULL,

         Conciliator_ID INTEGER(10),

         Crime_ID INTEGER(10) NOT NULL,

         PRIMARY KEY(Case_ID),

        FOREIGN KEY (Initiator_Lawyer) REFERENCES Lawyer(TC_ID),

        FOREIGN KEY (Court_Name) REFERENCES Court,

        FOREIGN KEY (Conciliator_ID) REFERENCES Conciliator(TC_ID),

        FOREIGN KEY (Crime_ID) REFERENCES Crime

);
```

2.13 Crime

Relational Model: Crime(<u>Crime_ID</u>,Date,Crime_Scene_Description,Crime_Description,Crime_Name)

Functional Dependencies: Crime_ID → Date, Crime_Scene_Description, Crime_Description, Crime_Name

Candidate Key: {(Crime_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Crime (
        Crime_ID INTEGER(10) NOT NULL,
        Date DATE NOT NULL,
        Crime_Scene_Description VARCHAR(255) NOT NULL,
        Crime_Description VARCHAR(255) NOT NULL,
        Crime_Name VARCHAR(255) NOT NULL,
         PRIMARY KEY(Crime_ID),
);
```

2.14 Translates

Relational Model: Translates(<u>TC_ID, Language_Name</u>)

Foreign Key:TC_ID references Interpreter

Foreign Key: Language_Name references Language

Functional Dependencies: None

Candidate Key: {(TC_ID, Language_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Translates (
        TC_ID INTEGER(10) NOT NULL,
        Language_Name VARCHAR(255) NOT NULL,
         PRIMARY KEY(TC_ID, Language_Name),
        FOREIGN KEY (TC_ID) REFERENCES Interpreter,
        FOREIGN KEY (Language_Name) REFERENCES Language
);
```

2.15 Works

Relational Model: Works(<u>TC_ID,Case_ID</u>)

Foreign Key:TC_ID references Interpreter

Foreign Key: Case_ID references Court_Case

Functional Dependencies: None

Candidate Key: {(TC_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Translates (
        TC_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
        PRIMARY KEY(TC_ID, Case_ID),
        FOREIGN KEY (TC_ID) REFERENCES Interpreter,
        FOREIGN KEY (Case_ID) REFERENCES Court_Case
);
```

2.16 Councils

Relational Model: Councils(Conciliator_ID, Litigant_ID, Case_ID, Agreed)

Foreign Key: Conciliator_ID references Conciliator(TC_ID)

Foreign Key:Case_ID references Case

Foreign Key:Litigant_ID references Litigant(TC_ID)

Functional Dependencies: Conciliator_ID, Litigant_ID, Case_ID → Agreed

Candidate Keys: {(Conciliator_ID, Litigant_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Councils (
        Councilator_ID INTEGER(10) NOT NULL,
        Litigant_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
        Agreed BIT NOT NULL,
        PRIMARY KEY(Conciliator_ID, Litigant_ID, Case_ID),
        FOREIGN KEY (Councilator_ID) REFERENCES Conciliator(TC_ID),
        FOREIGN KEY (Litigant_ID) REFERENCES Litigant (TC_ID),
        FOREIGN KEY (Case_ID) REFERENCES Court_Case
);
```

2.17 Represents

Relational Model: Represents(<u>Lawyer_ID,TC_ID, Case_ID</u>)

Foreign Key:Lawyer_ID references Lawyer(TC_ID)

Foreign Key:TC_ID references Litigant(TC_ID)

Foreign Key:Case_ID references Case

Functional Dependencies: None

Candidate Key: {(Lawyer_ID,TC_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Represents (
        TC_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
        Lawyer_ID INTEGER(10),
         PRIMARY KEY(Lawyer_ID, TC_ID, Case_ID),
        FOREIGN KEY (Lawyer_ID) REFERENCES Lawyer(TC_ID),
        FOREIGN KEY (TC_ID) REFERENCES Litigant(TC_ID),
        FOREIGN KEY (Case_ID) REFERENCES Court_Case
);
```

2.18 Informs

Relational Model: Informs(<u>TC_ID,Case_ID</u>)

Foreign Key:TC_ID references Expert_Witness

Foreign Key: Case_ID references Court_Case

Functional Dependencies: None

Candidate Key: {(TC_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Informs(
        TC_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
         PRIMARY KEY(TC_ID, Case_ID),
        FOREIGN KEY (TC_ID) REFERENCES Expert_Witness,
        FOREIGN KEY (Case_ID) REFERENCES Court_Case
);
```

2.19 Testifies

Relational Model: Testifies(<u>TC_ID,Case_ID, Witness_Account</u>)

Foreign Key:TC_ID references Eye_Witness

Foreign Key: Case_ID references Court_Case

Functional Dependencies: None

Candidate Key: {(TC_ID,Case_ID, Witness_Account)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Testifies (
        TC_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
        Witness_Account VARCHAR(255) NOT NULL,
         PRIMARY KEY(TC_ID, Case_ID, Witness_Account),
        FOREIGN KEY (TC_ID) REFERENCES Litigant(TC_ID),
        FOREIGN KEY (Case_ID) REFERENCES Court_Case
);
```

2.20 Judges

Relational Model: Judges(<u>TC_ID,Case_ID</u>)

Foreign Key:TC_ID references Judge

Foreign Key: Case_ID references Court_Case

Functional Dependencies: None

Candidate Key: {(TC_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

```
CREATE TABLE Judges (
        TC_ID INTEGER(10) NOT NULL,
        Case_ID INTEGER(10) NOT NULL,
         PRIMARY KEY(TC_ID, Case_ID),
        FOREIGN KEY (TC_ID) REFERENCES Judge,
		FOREIGN KEY (Case_ID) REFERENCES Court_Case
	);
```

2.21 Committed_At

Relational Model: Committed_At(Crime_Id, Address_Id)

Foreign Key:Crime_ID references Crime

Foreign Key:Address_ID references Address

Functional Dependencies: None

Candidate Key: {(Crime_Id, Address_Id)}

Normal Form: 3NF

Table Definition:

```
	CREATE TABLE  Committed_At (
		Address_ID INTEGER(10) NOT NULL,
		Crime_ID INTEGER(10) NOT NULL,
		 PRIMARY KEY(Crime_Id, Address_Id),
		FOREIGN KEY (Address_ID) REFERENCES Address,
		FOREIGN KEY (Crime_ID) REFERENCES Crime
	);
```

2.22 Trial_Date

Relational Model: Trial_Date(Case_ID, T_Date)

Foreign Key: Case_ID references Court_Case

Functional Dependencies: None

Candidate Key: {(Case_ID, T_Date)}

Normal Form: 3NF

Table Definition:

```
	CREATE TABLE  Committed_At (
		T_Date Date NOT NULL,
		Case_ID INTEGER(10) NOT NULL,
		 PRIMARY KEY(Case_ID, T_Date),
		FOREIGN KEY (Case_ID) REFERENCES Court_Case
	);
```

2.23 Involved

Relational Model: Involved(Litigant_ID, Case_ID, Role)

Foreign Key: Case_ID references Court_Case

Foreign Key: Litigant_ID references Litigant(TC_ID)

Functional Dependencies: None

Candidate Key: {(Litigant_ID, Case_ID)}

Normal Form: 3NF

Table Definition:

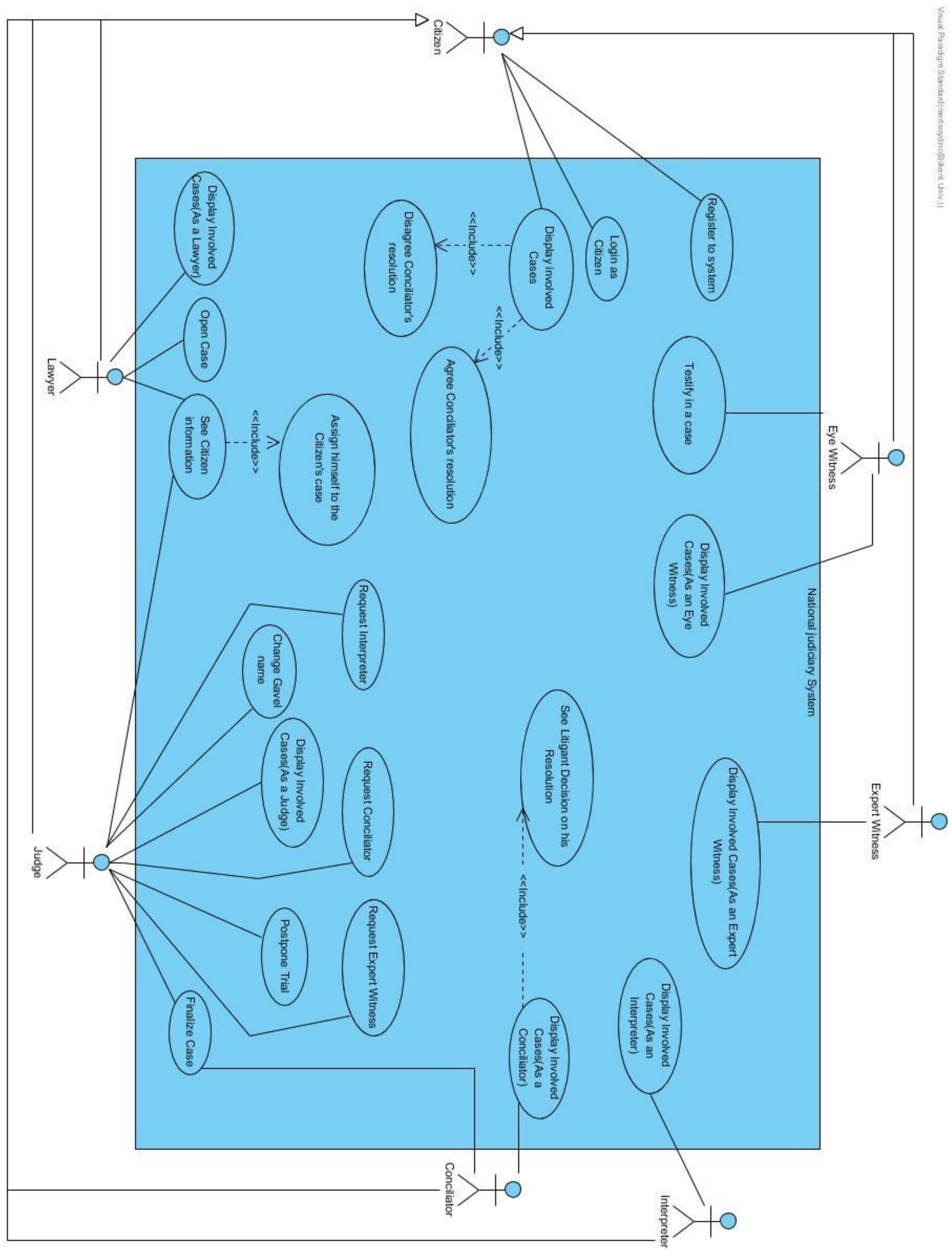
```
CREATE TABLE  Committed_At (

        Litigant_ID INTEGER(10) NOT NULL,

        Role VARCHAR(10) NOT NULL,

        Case_ID INTEGER(10) NOT NULL,

        PRIMARY KEY(Litigant_ID, Case_ID),

        FOREIGN KEY (Case_ID) REFERENCES Court_Case,

        FOREIGN KEY (Litigant_ID) REFERENCES Litigant(TC_ID)

);
```

3. Other Functional Dependencies, Normalization of Tables and Notes

All of the relations are in at least the third normal form, i.e. all non-key attributes are atomic, all non-key attributes depend on the whole key, and all non-key attributes do not depend anything other than the key. It is also worth noting that most, if not all, of these relations are also in Boyce-Codd Normal Form, i.e. the left-hand side of all dependencies is either trivial or is a super-key for the relation. The tables in this design are quite normalized, and there is no need to normalize them any further, as anomalies are easily avoided.

We excluded the trivial functional dependencies while writing the functional dependencies present in the tables above, i.e. we did not state that an attribute determines itself in the functional dependencies, and assumed that they are always implied. If a functional dependency in the tables above is listed as “None”, for instance, there is actually still a functional dependency in the table, and that dependency is the determination of the key by itself.

4. Use Cases


National judiciary System
Citizen
Register to system
Login as Citizen
Display involved Cases
<<Include>>
Disagree Conciliator's resolution
<<Include>>
Agree Conciliator's resolution
Testify in a case
Eye Witness
Display Involved Cases(As an Eye Witness)
Display Involved Cases(As a Lawyer)
Open Case
See Citizen information
<<Include>>
Assign himself to the Citizen's case
Lawyer
Request Interpreter
Change Gavel name
Display Involved Cases(As a Judge)
Request Conciliator
Postpone Trial
Request Expert Witness
Finalize Case
Judge
See Litigant Decision on his Resolution
<<Include>>
Display Involved Cases(As an Expert Witness)
Expert Witness
Display Involved Cases(As a Conciliator)
Display Involved Cases(As an Interpreter)
Conciliator
Interpreter

Login as Citizen:

Every citizen can log in with their account information that is assigned by the Government.

Display involved Cases:

Every citizen can see the cases in which they take part in as a suspect or a victim.

Disagree Conciliator's resolution:

Every citizen can disagree with the case resolution proposed by the Conciliator (if one exists).

Agree Conciliator's resolution.

Every citizen can agree with the case resolution proposed by the Conciliator (if one exists).

Display Involved Cases (As a Lawyer):

Every Lawyer can see the (past and ongoing) cases that they take a part in as a lawyer.

Open Case:

Every Lawyer can open a case according to the wishes of the client.

See Citizen Information:

Every Lawyer can see the information of citizens by using their ID’s.

Assign himself to the Citizen's case:

Every Lawyer can assign himself (if a lawyer is not already assigned) to the Citizen’s case.

Testify in a case:

Every Eye Witness can enter their Witness Account into the system.

Display Involved Cases(As an Eye Witness):

Every Eye Witness can see the (past and ongoing) cases that they take a part in as a Witness.

Display Involved Cases(As an Expert Witness):

Every Expert Witness can see the (past and ongoing) cases that they take a part in as a Witness.

Request Interpreter:

Every Judge can request an Interpreter to be assigned to a case if needed.

Change Gavel name:

Every Judge can change the name of his precious gavel.

Display Involved Cases (As a Judge):

Every Judge can see the (past and ongoing) cases that they take a part in as a Judge.

Request Conciliator:

Every Judge can request a Conciliator to be assigned to a case if needed.

Postpone Trial:

Every Judge can postpone the trial to a later date if needed.

Request Expert Witness:

Every Judge can request an Expert Witness to be assigned to a case if needed.

Finalize Case:

Every Judge can finalize a case.

Display Involved Cases (As a Conciliator) :

Every Conciliator can see the (past and ongoing) cases that they take a part in as a Conciliator.

See Litigant Decision on his Resolution:

Every Conciliator can see the Litigants decision on his proposed resolution to the case.

Display Involved Cases (As an Interpreter)

Every Interpreter can see the (past and ongoing) cases that they take a part in as an Interpreter.

[This space has been intentionally left blank]

5. Algorithms

5.1 Assignment Algorithms for Judges, Interpreters, Conciliators, and Expert Witnesses

The database management system will keep track of the number of cases a judge, an interpreter, conciliator, and an expert witness is currently involved in. As per the limitations in the project proposal, the number of cases that a judge, interpreter and expert witness can have is bounded from above. Also, to ensure that the process of the assignment of a case to one of these actors is carried out as fair as possible, that is to say that, for instance, a judge should not be able to "pick" a case in which a close acquaintance of the said judge is present (to present judicial corruption), a judge will *randomly* be assigned to a new case. Similarly, those who are eligible for assignment to a case, i.e. judges, interpreters, conciliators, and expert witnesses, are randomly assigned to the court case. Judges are automatically assigned to the court case by the system, and the rest of the aforementioned actors may be assigned to the case if the judge thinks it is necessary for the actor to take place in the court case. In this case, the system randomly assigns an available/eligible interpreter (based on the language that needs to be translated), conciliator (if the judge thinks the case may be settled through negotiation), or an expert witness (if the judge thinks the case may only be resolved through expert knowledge) based on the number of cases they are currently taking part in, and based on their address. The algorithm will consider the proximity of the current address of that actor by comparing the *city* attribute of the current address of that user to the *city* attribute of the address of the court in which the trial will take place. The algorithm will also try not to assign a user a court case if the number of current court cases for that user is five. Due to extreme demand or the distribution of the actors with respect to the cities they live in, the algorithm may over-assign some work to some of the actors to prevent starvation of court case requests.

The following is an example query that the algorithm will use as an input to find an appropriate assignment of a court case to a judge. It finds the judges who live in the same city as the city in which the court case will be held, and who currently have less than six court cases assigned to them. The algorithm will request this data from the database and then will pick one judge to assign to a case. Similar queries will be written for other actors. (i.e. instead of joining the Judges and Citizen table, the tables corresponding to the other tables will be joined with citizen to filter out people who live in other cities).

```
SELECT Judges.TC_ID, COUNT(Judges.Case_ID) AS Case_Count

FROM (SELECT *

        FROM Judges NATURAL JOIN Citizen

        WHERE Judges.Case_ID = &CASE_ID) NATURAL JOIN Address

WHERE Address.City IN (
        SELECT A.City
        FROM Court_Case, Court, Address AS A
        WHERE Court_Case.Court_Name = Court.Court_Name AND Court.Address_ID =
A.Address_ID
)
GROUP BY Judges.TC_ID
HAVING COUNT(Judges.CASE_ID) < 6
```

5.2 Algorithms for Trial Date Scheduling

The database management system will take care of the scheduling of the trial dates for a given court case. It is intended that if a judge decides to postpone a trial, the system will automatically try to find a time slot at the same court which is at least three months later than the latest scheduled trial in the database. The following is the query that returns the latest scheduled time slot for a given trial for a court case.

```
(SELECT T_Date
FROM Trial_Date
WHERE Trial_Date.Case_ID = &CASE_ID)
EXCEPT
(SELECT T1.T_Date
FROM Trial_Date AS T1, Trial_Date AS T2
WHERE T1.Case_ID = T2.Case_ID AND T1.Case_ID = &CASE_ID AND T1.T_Date < T2.T_Date)
```

The following query will then be run by the algorithm to fetch the dates on which a case has been scheduled at the same court, and the dates that are returned are at least three months after the latest date on which a trial for this particular court case will be held.

```
SELECT Trial_Date.T_Date
FROM Trial_Date
WHERE Trial_Date.Case_ID IN(
        SELECT Court_Case.Case_ID
        FROM Court NATURAL JOIN Court_Case
        WHERE Court.Court_Name = Court_Case.Court_Name
) AND Trial_Date.T_Date > &LATEST_TRIAL + 90
```

The scheduler will then find a date which is NOT included in the query above and which is later than &LATEST_TRIAL + 90. And then, the following query will be run:

```
INSERT INTO Trial_Date

VALUES(&CASE_ID, &TRIAL_DATE)
```

For the case of the initial designation of a trial date, the system will automatically insert a tuple into the Trial_Date tuple when a judge is appointed to the case. The following query will be run

```
INSERT INTO Trial_Date

VALUES(&CASE_ID, &JUDGE_APPOINTMENT_DATE + &OFFSET)
```

Which means that there will be a trial after in at least &OFFSET days after a judge has been appointed. &OFFSET will be a value larger than 90 days. The program will run the previous queries to fetch the dates on which there are trials, and will avoid scheduling the trial on those days.

5.3 Lawyer Related Algorithms

There are two algorithms that the system will employ regarding the possible actions of lawyers can take in NJIS2. The first algorithm is the algorithm which generates a court case for a set of victims, identified by their TC IDs. The system takes in all information required to start a case: Victim and Suspect TC IDs, a description of the case, the location of the case file in terms of building/room as the case file attribute, and full information about the crime: Crime name, date on which the crime has happened, a description of the crime, a description of the crime scene, the address at which the crime took place (city name, address). This will add all the involved people to the case by inserting tuples into the involved table. The role of the suspects will be marked as "Suspect" and the role of the victims are marked as "Victim". A judge is automatically assigned to the case, as described in the previous algorithm. The lawyer will also be inserted into the represents table with the appropriate victim TC ID to lawyer TC ID mapping.

The other algorithm which involves lawyers is the "get case" algorithm. A lawyer may look up a specific case by running a search over the TC IDs of citizens. The lawyer may then choose to represent a suspect in a case. This will only insert to the represents table.

5.4 Judge Related Algorithms

Judges can do many things in the system: Most importantly, they can finalize the court case to which they are assigned. This is done via providing a "verdict" for the court case. This simply updates the court case tuple, and marks it as a closed case. If the judge thinks that a verdict has not yet been reached, the judge may postpone the trial, the algorithm for which works as above. The judge may also request other actors to be involved in the court case, such as interpreters, expert witnesses, and conciliators. This will insert to the related tables of the E/R model, i.e. the Testifies table, the Works table, the Informs table, and the Councils table.

The judge may also store the name of the gavel employed in the judicial process. This can be done so by requesting to change the gavel name through the user interface.

5.5 Conciliator Related Algorithms

Conciliators are appointed by the system to a particular court case if the judge of that case decides that it is necessary for a conciliator to be a part of the court case. The conciliator tries to establish negotiations between the suspects and the victims. Each person involved in the case with the roles "Suspect" and "Victim" are inserted in the councils table. They all have an agreement bit assigned to them. They may change their status to "agreed" if they come to terms with the other side. Once everybody agrees, the conciliator may finalize the case with the click of a button. For this to happen, all the agreed bits of a particular court case must be one, and the system checks this before it allows the finalization of the case by the conciliator.

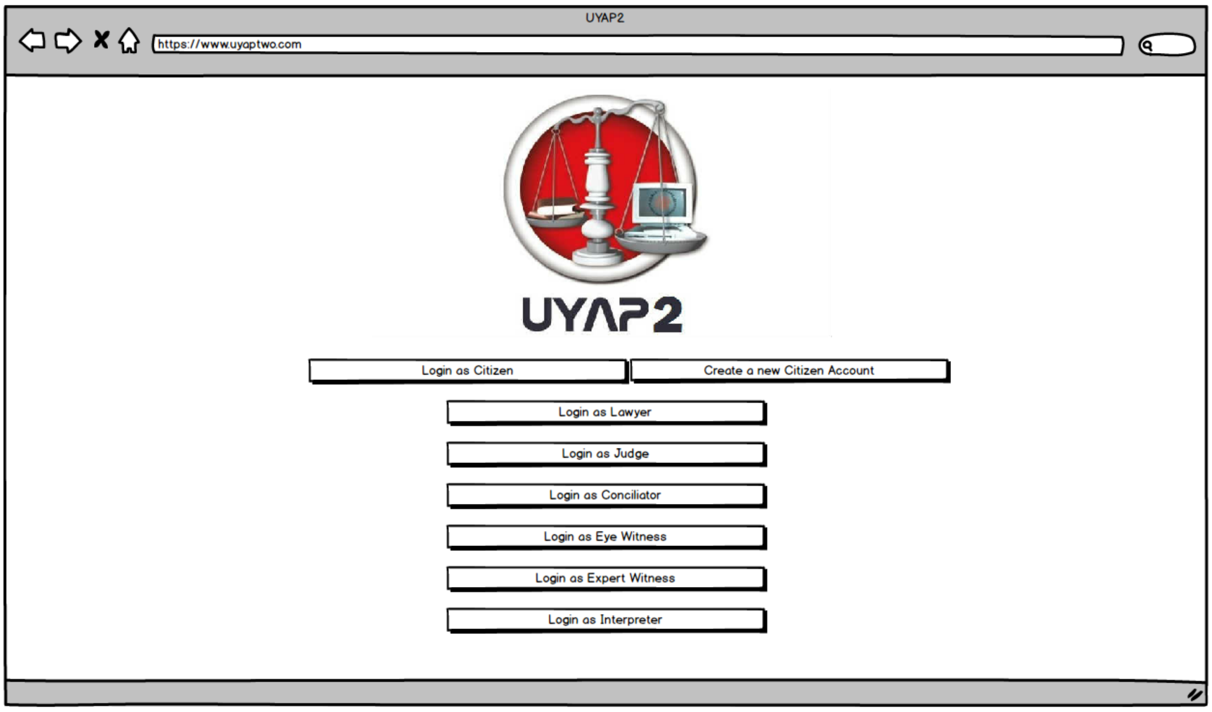
5.6 Other Algorithms for Logical Desiderata

The system will run some batch logical checks daily to detect any inconsistencies in the database. These checks will include determining the amount of "overworked" judicial staff, get report updates for OLAP purposes. Also, this miscellaneous category of algorithms will check whether two distinct cases have the same date by accident, whether caused by concurrency or availability issues, and will fix them.

[This space has been intentionally left blank]

6. User Interface Design and Corresponding SQL Statements

6.1 Main Page



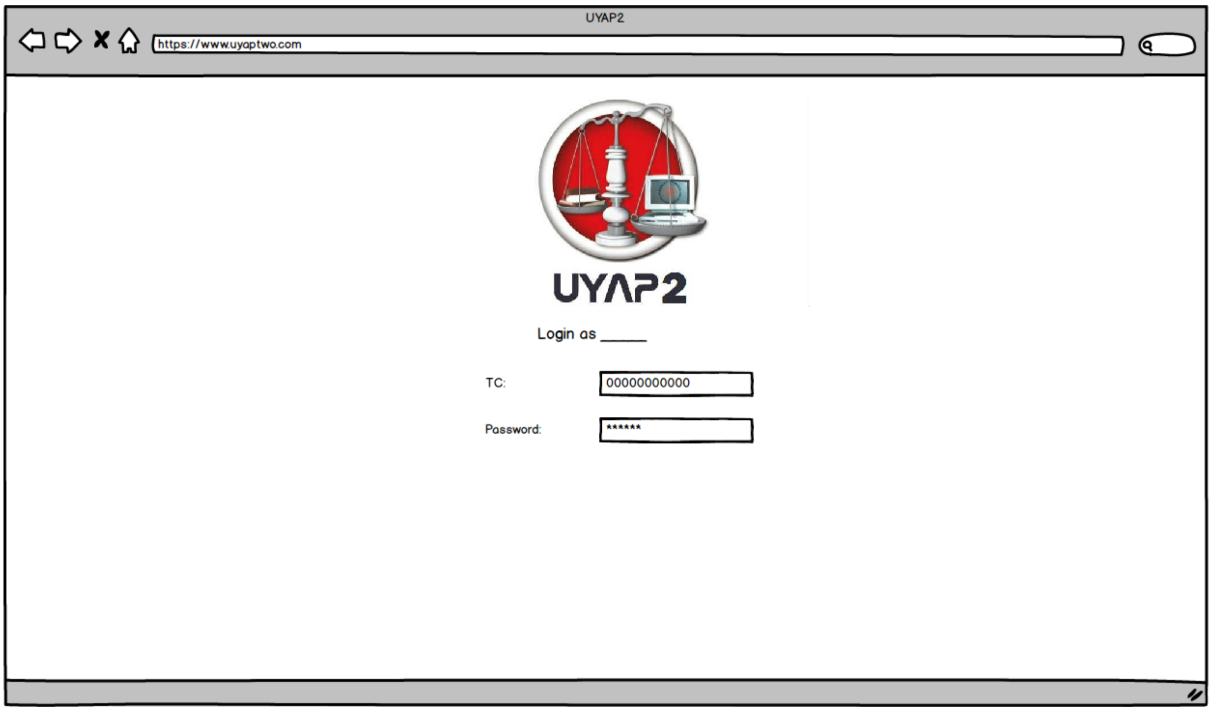
Inputs: -

Process: This screen will allow any user to login to a particular portal of the system, just like in the original UYAP. Each person has a different login for a different "portal". For instance, a judge has a judge account and a citizen account. A regular citizen only has a citizen account. Other roles have their corresponding accounts, along with their corresponding citizen accounts. All people registered with the Ministry of Internal Affairs should be present in the system, however, since this is merely a simulation of the original system, by choosing the option to register to the system, citizens may register.

SQL Statements: None.

[This space has been intentionally left blank]

6.2 Login



Login as ______

TC: 00000000000

Password: ******

Inputs: &TC_ID, &Password

Process: The user enters a TC_ID and a password. The query that will be executed depends on the portal to which the user wants to log in to. If the user wants to login to the lawyer portal, the system checks the Lawyer table for the password, if the user wants to login to the citizen portal, the system checks the Citizen table, and so on. The database management system does not directly store the passwords, but it rather uses a "salted" version of the SHA256 hash of the password for security reasons. System admin has a TC_ID of 0, and is registered in the Citizen table.

SQL Statements:

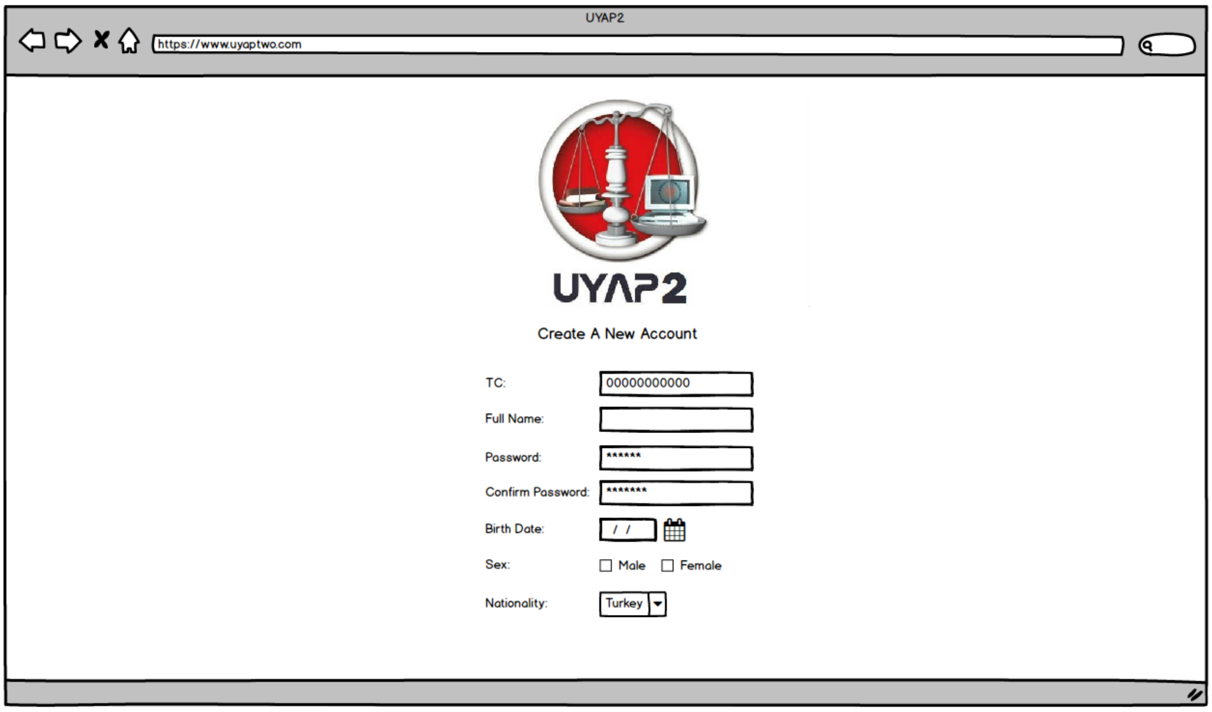
List of possible queries depending on the login portal:

```
SELECT TC_ID, Citizen_Password

FROM Citizen

WHERE Citizen.TC_ID = &TC_ID AND Citizen_Password = &Password;


SELECT TC_ID, Lawyer_Password

FROM Lawyer

WHERE Lawyer.TC_ID = &TC_ID AND Lawyer _Password = &Password;


SELECT TC_ID, Eye_Password

FROM Eye_Witness

WHERE Eye_Witness.TC_ID = &TC_ID AND Eye _Password = &Password;
```

```
SELECT TC_ID, Judge_Password

FROM Judge

WHERE Judge.TC_ID = &TC_ID AND Judge _Password = &Password;


SELECT TC_ID, Expert_Password

FROM Expert_Witness

WHERE Expert_Witness.TC_ID = &TC_ID AND Expert _Password = &Password;


SELECT TC_ID, Conciliator_Password

FROM Conciliator

WHERE Conciliator.TC_ID = &TC_ID AND Conciliator_Password = &Password;


SELECT TC_ID, Citizen_Password AS Admin_Password

FROM Citizen

WHERE &TC_ID = 0 AND Citizen_Password = &Password;
```

[This space has been intentionally left blank]

6.3 Citizen Sign Up



Create A New Account

TC:	00000000000
Full Name:	
Password:	******
Confirm Password:	*******
Birth Date:	/ /
Sex:	☐ Male ☐ Female
Nationality:	Turkey

Inputs: &TC_NO, &NAME, &PASSWORD, &BIRTHDATE, &SEX, &NATIONALITY, &ADDR_ID

Process: Any user may open a citizen account in the system by providing their information. This information is later added to the system. The user has to fill in the necessary fields, which corresponds to the fields given in the mockup. Not all fields are present in the mockup due to space constraints. The address ID is automatically generated.

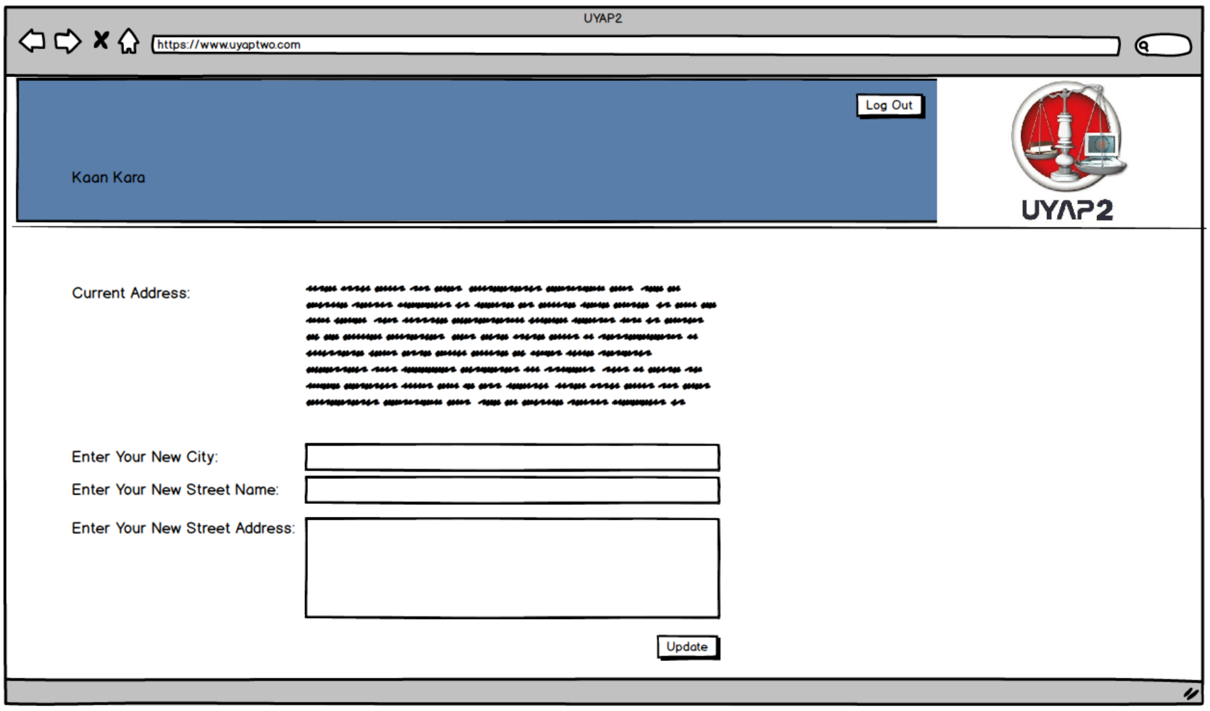
SQL Statements:

Sign Up:

INSERT INTO Citizen

VALUES(&TC_ID, &NAME, &BIRTHDATE, &SEX, &NATIONALITY, &ADDR_ID, &PASSWORD)

6.4 Citizen Address Update



Kaan Kara

Current Address:

Enter Your New City:

Enter Your New Street Name:

Enter Your New Street Address:

Update

Input: &TC_ID, &CITY, &STREET_NAME &STREET_ADDR

Process: The citizen may update their address information by providing the required fields to the system and clicking on update. Finally, the user can log out.

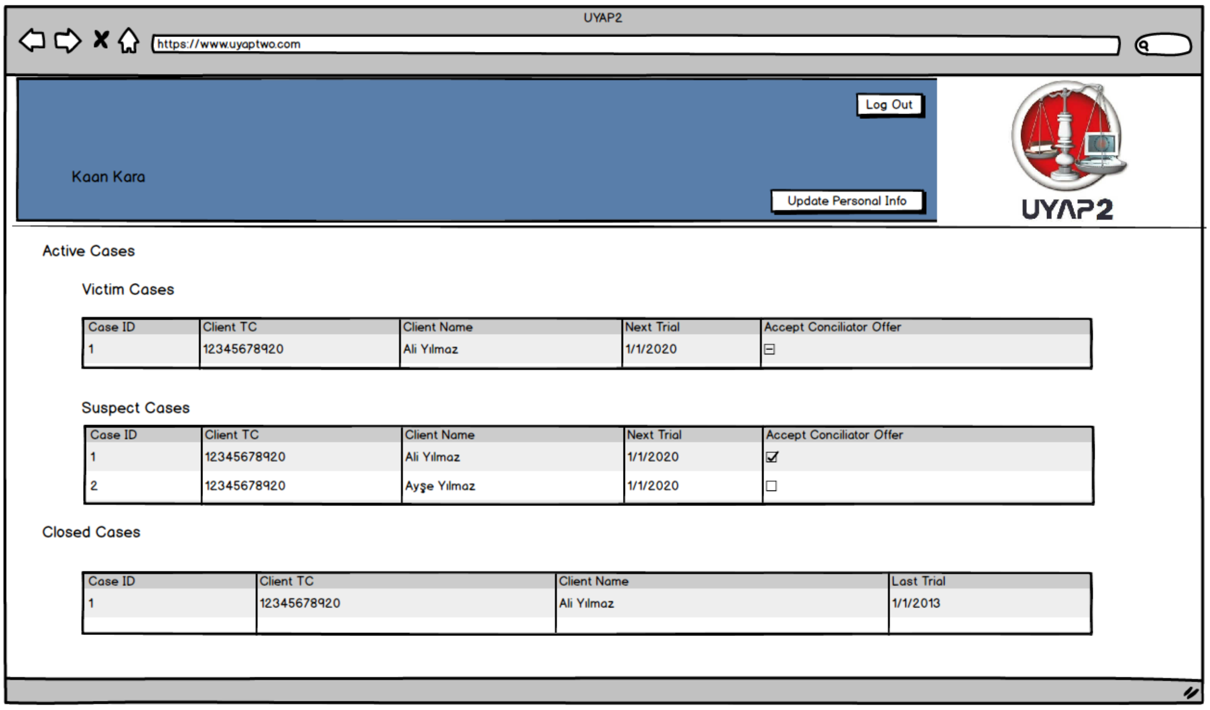
SQL Statements:

Address Update

```
WITH Address_Citizen AS (
        SELECT Address_ID
        FROM Adress NATURAL JOIN Citizen
        WHERE Citizen.TC_ID = &TC_ID
)
IF((&CITY, &STREET_NAME, &STREET_ADDR) NOT EXISTS (SELECT City, Street_Name, Street_Address FROM Address))
BEGIN
INSERT INTO Adress
        VALUES(&Address _ID, City = &CITY, Street_Name = &STREET_NAME, Street_Address = &STREET_ADDR)
END
UPDATE Citizen
```

```
SET Citizen.Address_ID = &Address_ID

WHERE Citizen.Adress_ID = Address_Citizen
```

6.5 Citizen Main View



Kaan Kara

Active Cases

Victim Cases

Case ID	Client TC	Client Name	Next Trial	Accept Conciliator Offer
1	12345678920	Ali Yılmaz	1/1/2020	⊟

Suspect Cases

Case ID	Client TC	Client Name	Next Trial	Accept Conciliator Offer
1	12345678920	Ali Yılmaz	1/1/2020	☑
2	12345678920	Ayşe Yılmaz	1/1/2020	☐

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2013

Inputs: &TC_ID, &CASE_ID

Process: Every citizen has a unique TC ID and has a page on which the citizen may view the current cases in which the citizen is involved either as a Victim or a Suspect. The citizen is also able to view the closed cases, cases they were once a part of. The citizen is displayed a summary of the cases on this page. The Case ID, TC ID of the involved party, the name of the involved party, and the next trial date is displayed to the user. The citizen may also accept the offer by a conciliator appointed to the case by the judge by marking the checkbox next to the summary of a case. From this page, the user may either log out or update their address information. Finally, the user can update his personal info(only address) and log out.

SQL Statements

Active Suspect Case Summary:

```
WITH Next_Trial AS (
                    (SELECT Case_ID, T_Date
                    FROM Trial_Date)
                    EXCEPT
                    (SELECT T1.Case_ID, T2.T_Date
                    FROM Trial_Date AS T1, Trial_Date AS T2
                    WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
          )
          SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
```

```
            FROM Involved, Citizen, Next_Trial, Court_Case
            WHERE Involved.Litigant_ID Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND
Involved.Role = “Suspect” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 0
AND Citizen.TC_ID = &TC_ID
```

Active Victim Case Summary:

```
WITH Next_Trial AS (
                        (SELECT Case_ID, T_Date
                        FROM Trial_Date)
                        EXCEPT
                        (SELECT T1.Case_ID, T2.T_Date
                        FROM Trial_Date AS T1, Trial_Date AS T2
                        WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
)
SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
FROM Involved, Citizen, Next_Trial, Court_Case
WHERE Involved.Litigant_ID Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND
Involved.Role = “Victim” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 0
AND Citizen.TC_ID = &TC_ID
```

Closed Suspect Case Summary

```
WITH Next_Trial AS (
                        (SELECT Case_ID, T_Date
                        FROM Trial_Date)
                        EXCEPT
                        (SELECT T1.Case_ID, T2.T_Date
                        FROM Trial_Date AS T1, Trial_Date AS T2
                        WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
)
SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
FROM Involved, Citizen, Next_Trial, Court_Case
```

```
WHERE Involved.Litigant_ID Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND Involved.Role = “Suspect” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 1 AND Citizen.TC_ID = &TC_ID
```

Closed Victim Case Summary

```
WITH Next_Trial AS (
            (SELECT Case_ID, T_Date
            FROM Trial_Date)
            EXCEPT
            (SELECT T1.Case_ID, T2.T_Date
            FROM Trial_Date AS T1, Trial_Date AS T2
            WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
)
SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
FROM Involved, Citizen, Next_Trial, Court_Case
WHERE Involved.Litigant_ID Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND Involved.Role = “Victim” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 1 AND Citizen.TC_ID = &TC_ID
```

Citizen Account Info

```
SELECT TC_ID, FullName, BIrthDate, Biological_Sex, Nationality
FROM Citizen
WHERE TC_ID = &TC_ID
```

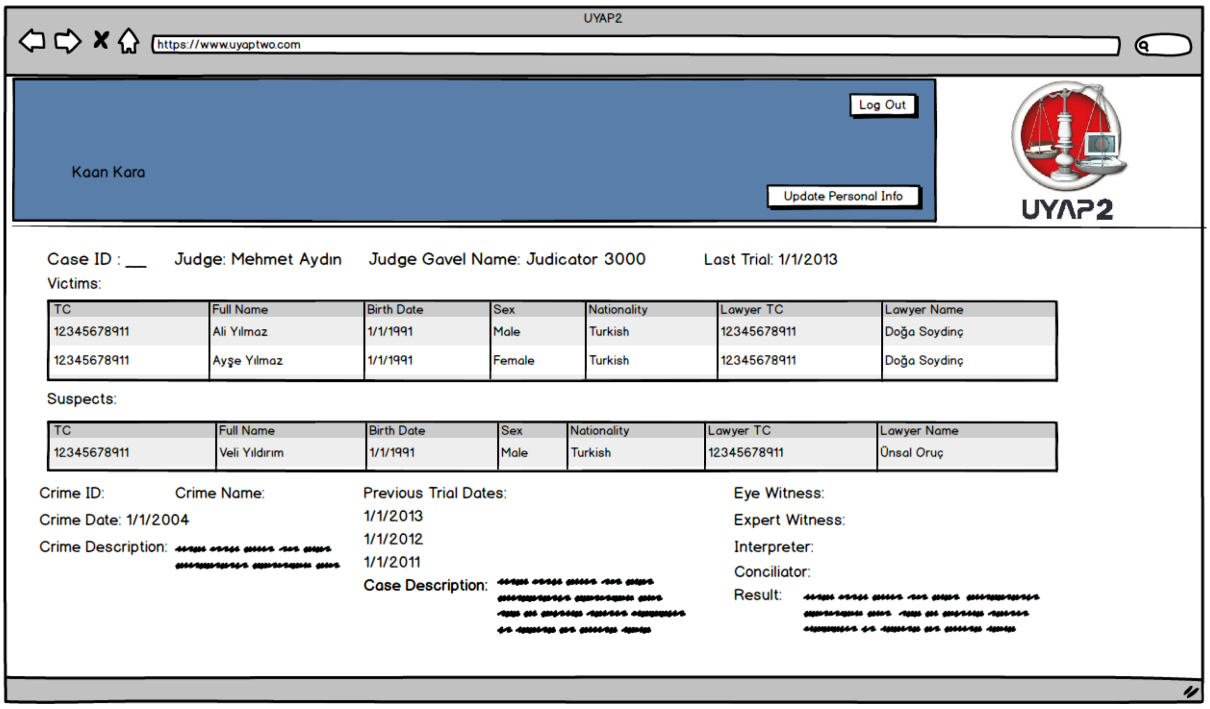
Conciliation Accept Update

```
UPDATE Councils
SET Agreed = 1
WHERE Litigant_ID = &TC_ID AND Case_ID = &CASE_ID
```

Conciliation Deslect Update

```
UPDATE Councils
SET Agreed = 0
WHERE Litigant_ID = &TC_ID AND Case_ID = &CASE_ID
```

6.6 Case View



Kaan Kara

Case ID : __ Judge: Mehmet Aydın Judge Gavel Name: Judicator 3000 Last Trial: 1/1/2013

Victims:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Ali Yılmaz	1/1/1991	Male	Turkish	12345678911	Doğa Soydinç
12345678911	Ayşe Yılmaz	1/1/1991	Female	Turkish	12345678911	Doğa Soydinç

Suspects:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Veli Yıldırım	1/1/1991	Male	Turkish	12345678911	Ünsal Oruç

Crime ID:
Crime Name:
Crime Date: 1/1/2004
Crime Description:

Previous Trial Dates:
1/1/2013
1/1/2012
1/1/2011
Case Description:

Eye Witness:
Expert Witness:
Interpreter:
Conciliator:
Result:

Inputs: &CASE_ID (ID of the current case)

Process: This is a generic view of detailed information about a given court case. Users are related to the case are able to see nearly the entirety of the details about the case present in the database. The users are able to view the ID of the case, the judge of the case, the name of the judge's gavel, the latest trial, the victims and the suspects, information related to the crime that has been committed, the assigned eye witness, conciliator, expert witness, and the interpreter, along with the final verdict of the judge. Finally, the user can update his personal info(only address) and the user can log out.

SQL Statements:

Judge Information:

SELECT Court_Case.Case_ID AS CASE_ID, Judge.TC_ID, Citizen.FullName, Citizen.BirthDate, Citizen.Biological_Sex, Citizen.Nationality, Judge.Gavel_Name

FROM Citizen, Judge, Judges, Court_Case

WHERE Citizen.TC_ID = Judges.TC_ID AND Judge.TC_ID = Judges.TC_ID AND Judges.Case_ID = Court_Case.CASE_ID AND CASE_ID = &CASE_ID

Court and Case Information:

SELECT Court_Case.Case_ID, Court_Case.Description, Court_Case.Case_file, Court_Case.IsClosed, Court_Case.Result, Court_Case.Court_Name, Court.Court_Type, Address.Street_Name, Address.City, Address.Street_Address

FROM Court_Case, Court, Address

```
WHERE Court.Court_Name = Court_Case.Court_Name AND Court.Address_ID = Address.Address_ID
AND CASE_ID = &CASE_ID
```

Victim Information:

```
SELECT Involved.Case_ID AS CASE_ID, C1.TC_ID, C1.FullName, C1.BirthDate, C1.Biological_Sex,
C1.Nationality, C2.TC_ID, C2.FullName

FROM Involved, Citizen AS C1, Citizen AS C2, Represents

WHERE Involved.TC_ID = C1.TC_ID AND Represents.TC_ID = Involved.TC_ID AND Involved.Role =
“Victim” AND C2.TC_ID = Represents.Lawyer_ID AND Involved.Case_ID = Represents.Case_ID AND
CASE_ID = &CASE_ID
```

Suspect Information:

```
SELECT Involved.Case_ID AS CASE_ID, C1.TC_ID, C1.FullName, C1.BirthDate, C1.Biological_Sex,
C1.Nationality, C2.TC_ID, C2.FullName

FROM Involved, Citizen AS C1, Citizen AS C2, Represents

WHERE Involved.TC_ID = C1.TC_ID AND Represents.TC_ID = Involved.TC_ID AND Involved.Role =
“Suspect” AND C2.TC_ID = Represents.Lawyer_ID AND Involved.Case_ID = Represents.Case_ID AND
CASE_ID = &CASE_ID
```

Crime Information:

```
SELECT Crime_ID, Crime_Date, Crime_Scene_Description, Crime_Description, Crime_Name

FROM Court_Case NATURAL JOIN Crime

WHERE Case_ID = &CASE_ID
```

Trial Date Information:

```
SELECT  T_Date

FROM Trial_Dates

WHERE Case_ID = &CASE_ID

ORDER BY T_Date
```

Interpreter Information:

```
SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID

FROM Citizen, Works

WHERE Citizen.TC_ID = Works.TC_ID AND Case_ID = &CASE_ID
```

Eye Witness Information

```
SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID

FROM Citizen, Testifies

WHERE Citizen.TC_ID = Testifies.TC_ID AND CASE_ID = &CASE_ID
```

Conciliator Information:

```
SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID

FROM Citizen, Councils

WHERE Citizen.TC_ID = Councils.TC_ID AND Case_ID = &CASE_ID
```

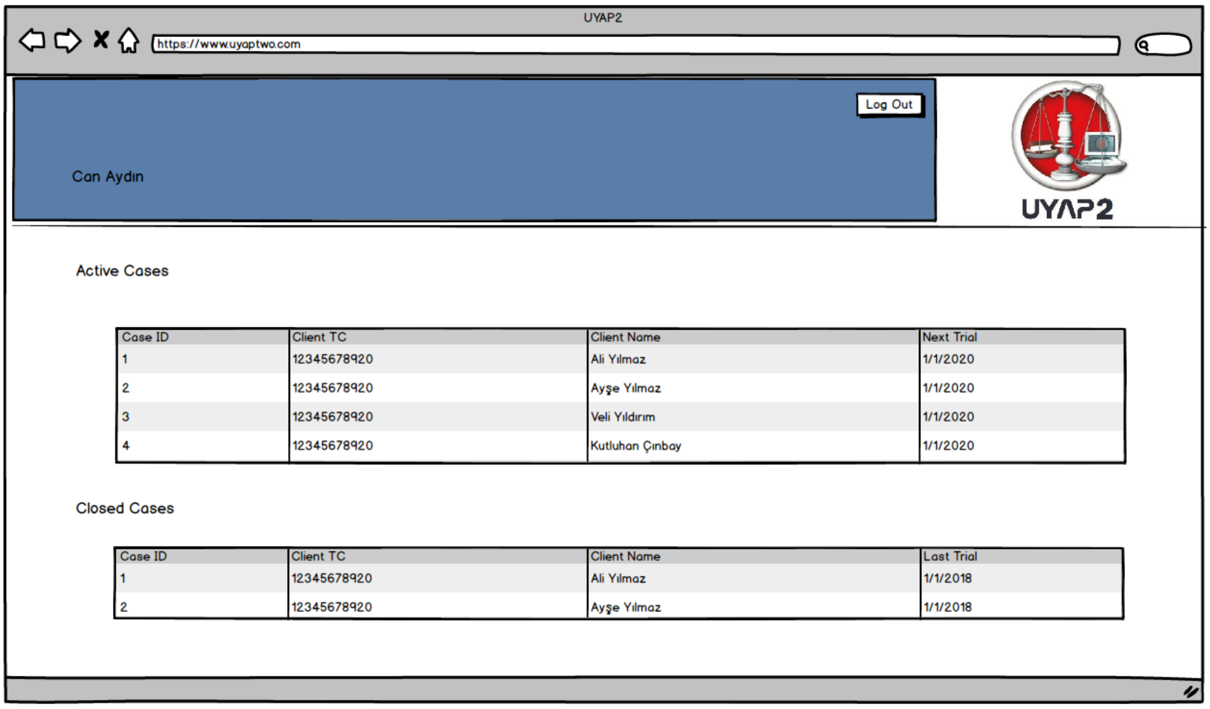
Expert Witness Information:

```
SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality

FROM Citizen, Informs

WHERE Citizen.TC_ID = Informs.TC_ID AND Case_ID = &CASE_ID
```

Note: In practice, these queries will be executed over several views using stored procedures. See the advanced SQL section for the details.

[This space has been intentionally left blank]

6.7 Conciliator Portal Main Screen



Active Cases

Case ID	Client TC	Client Name	Next Trial
1	12345678920	Ali Yılmaz	1/1/2020
2	12345678920	Ayşe Yılmaz	1/1/2020
3	12345678920	Veli Yıldırım	1/1/2020
4	12345678920	Kutluhan Çınbay	1/1/2020

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2018
2	12345678920	Ayşe Yılmaz	1/1/2018

Input: &TC_ID

This screen is mostly the same as the citizen portal main view, however, since conciliators are neither victims nor suspects, their display is slightly different. They only see their active and closed cases, and the client names are written. Finally, the user can log out.

SQL Statements:

Active Cases

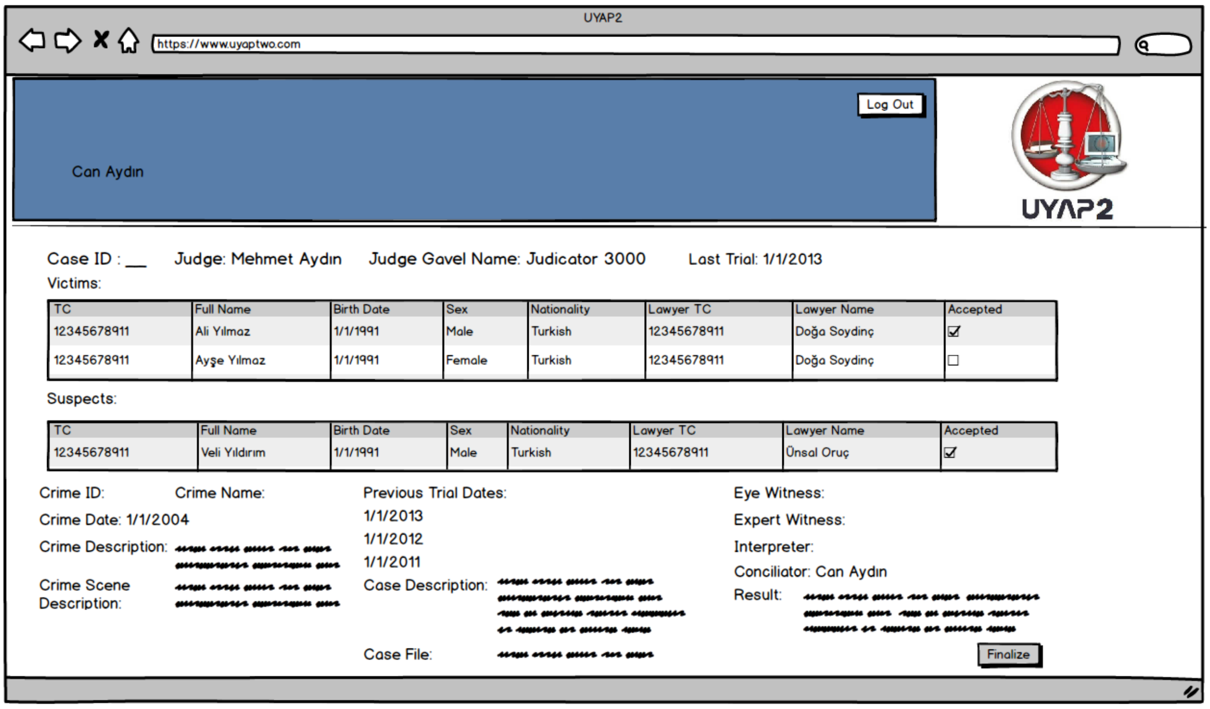
```
WITH Next_Trial AS (
            (SELECT Case_ID, T_Date
            FROM Trial_Date)
            EXCEPT
            (SELECT T1.Case_ID, T2.T_Date
            FROM Trial_Date AS T1, Trial_Date AS T2
            WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
)
SELECT Councils.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
FROM Councils, Citizen, Next_Trial, Court_Case
WHERE Court_Case.Initiator_Lawyer = Citizen.TC_ID AND Councils.TC_ID = &TC_ID AND
Next_Trial.Case_ID = Councils.Case_ID AND Court_Case.IsClosed = 0
```

Closed Cases

```
WITH Next_Trial AS (
        (SELECT Case_ID, T_Date
        FROM Trial_Date)
        EXCEPT
        (SELECT T1.Case_ID, T2.T_Date
        FROM Trial_Date AS T1, Trial_Date AS T2
        WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
)
SELECT Councils.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date
FROM Councils, Citizen, Next_Trial, Court_Case
WHERE Court_Case.Initiator_Lawyer = Citizen.TC_ID AND Councils.TC_ID = &TC_ID AND
Next_Trial.Case_ID = Councils.Case_ID AND Court_Case.IsClosed = 1
```

[This space has been intentionally left blank]

6.8 Conciliator Case Finalization



Can Aydın

Case ID : __ Judge: Mehmet Aydın Judge Gavel Name: Judicator 3000 Last Trial: 1/1/2013

Victims:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name	Accepted
12345678911	Ali Yılmaz	1/1/1991	Male	Turkish	12345678911	Doğa Soydinç	☑
12345678911	Ayşe Yılmaz	1/1/1991	Female	Turkish	12345678911	Doğa Soydinç	☐

Suspects:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name	Accepted
12345678911	Veli Yıldırım	1/1/1991	Male	Turkish	12345678911	Ünsal Oruç	☑

Crime ID: Crime Name:

Crime Date: 1/1/2004

Crime Description:

Crime Scene Description:

Previous Trial Dates:
1/1/2013
1/1/2012
1/1/2011

Case Description:

Case File:

Eye Witness:

Expert Witness:

Interpreter:

Conciliator: Can Aydın

Result:

Input: &TC_ID, &CASE_ID (Conciliator)

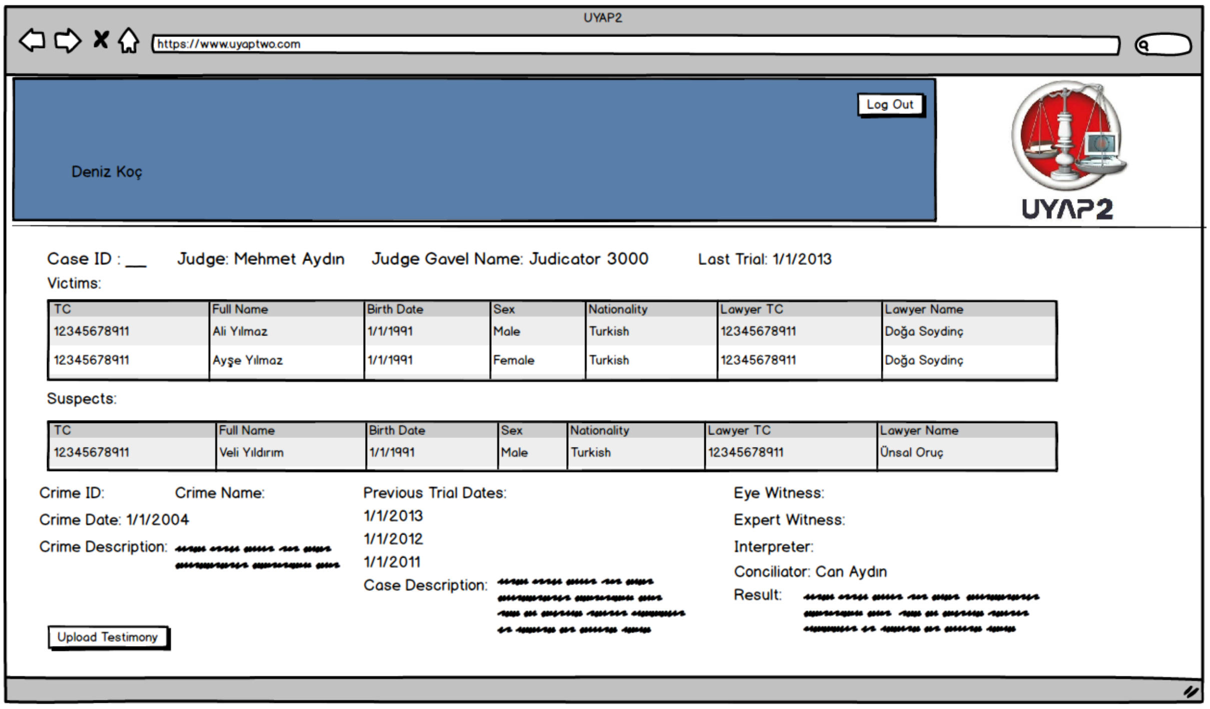
Process: This screen shows the current state for the conciliation process for a court case with a conciliator present. There are no SQL statements to handle this case other than fetching the information present in the case. A trigger is fired upon an update in the Councils table, and it checks for a given case whether every single suspect and victim has agreed to drop the case. If everybody has accepted, the trigger enables the button and the conciliator may click on the finalize button to close the case. Finally, the user can log out.

SQL Statements:

Same statements as Case View

[This space has been intentionally left blank]

6.9 Eye Witness Case View



Deniz Koç

Case ID : __ Judge: Mehmet Aydın Judge Gavel Name: Judicator 3000 Last Trial: 1/1/2013

Victims:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Ali Yılmaz	1/1/1991	Male	Turkish	12345678911	Doğa Soydinç
12345678911	Ayşe Yılmaz	1/1/1991	Female	Turkish	12345678911	Doğa Soydinç

Suspects:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Veli Yıldırım	1/1/1991	Male	Turkish	12345678911	Ünsal Oruç

Crime ID: Crime Name:
Crime Date: 1/1/2004
Crime Description:

Previous Trial Dates:
1/1/2013
1/1/2012
1/1/2011
Case Description:

Eye Witness:
Expert Witness:
Interpreter:
Conciliator: Can Aydın
Result:

Input: &TC_ID, &CASE_ID ,&Testimony, &RANDOM_PASSWORD (Eye Witness)

Process: This screen shows the current state for a court case. There are no SQL statements to handle this case other than fetching the information present in the case and inserting the Eye Witness Testimony into the system. Finally, the user can log out.

SQL Statements:

Eye Witness Creation by an External Actor

INSERT INTO Eye_Witness

VALUES(&TC_ID, &RANDOM_PASSWORD)

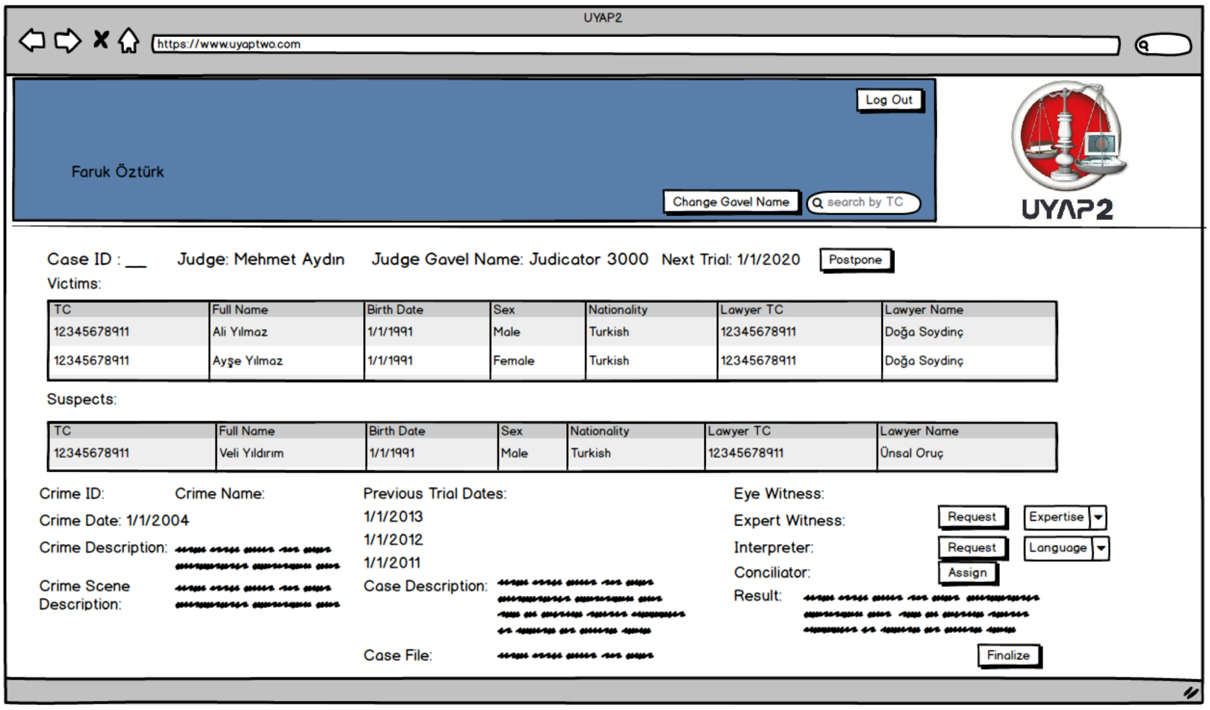
Information Fetching is the same as Case View

Upload Testimony:

INSERT INTO Testify VALUES(&TC_ID,&CASE_ID,&Testimony) ;

[This space has been intentionally left blank]

6.10 Judge Case View



Faruk Öztürk

Case ID : __ Judge: Mehmet Aydın Judge Gavel Name: Judicator 3000 Next Trial: 1/1/2020 Postpone

Victims:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Ali Yılmaz	1/1/1991	Male	Turkish	12345678911	Doğa Soydinç
12345678911	Ayşe Yılmaz	1/1/1991	Female	Turkish	12345678911	Doğa Soydinç

Suspects:

TC	Full Name	Birth Date	Sex	Nationality	Lawyer TC	Lawyer Name
12345678911	Veli Yıldırım	1/1/1991	Male	Turkish	12345678911	Ünsal Oruç

Crime ID: Crime Name:

Crime Date: 1/1/2004

Crime Description:

Crime Scene Description:

Previous Trial Dates:
1/1/2013
1/1/2012
1/1/2011

Case Description:

Case File:

Eye Witness:

Expert Witness: Request Expertise

Interpreter: Request Language

Conciliator: Assign

Result:

Finalize

Input: &TC_ID, &CASE_ID, &VERDICT, &CON_TC(Randomy Picked), &INTR_TC(Randomly Picked), &EW_TC(Randomly Picked), &LANGUAGE, &EXPERTISE, &GAVEL_NAME

Process: This screen shows the current state for a court case. This view requires fetching the information present in the case. Judges can request Expert Witnesses, Interpreter and can assign Conciliators to this specific case. Also the Judge can finalize the case. The user can also search the database using a TC ID and can log out.

SQL Statements:

Same Statements as Case View

Judge Case Finalization

UPDATE Court_Case

SET Court_Case.Result=&VERDICT, IsClosed = 1

WHERE Court_Case.Case_ID = &CASE_ID

Conciliator Request, Picked Randomly From the Pool

INSERT INTO Councils

VALUES(&CON_TC_ID, &TC_ID, &CASE_ID, 0)

Judge Interpreter Request

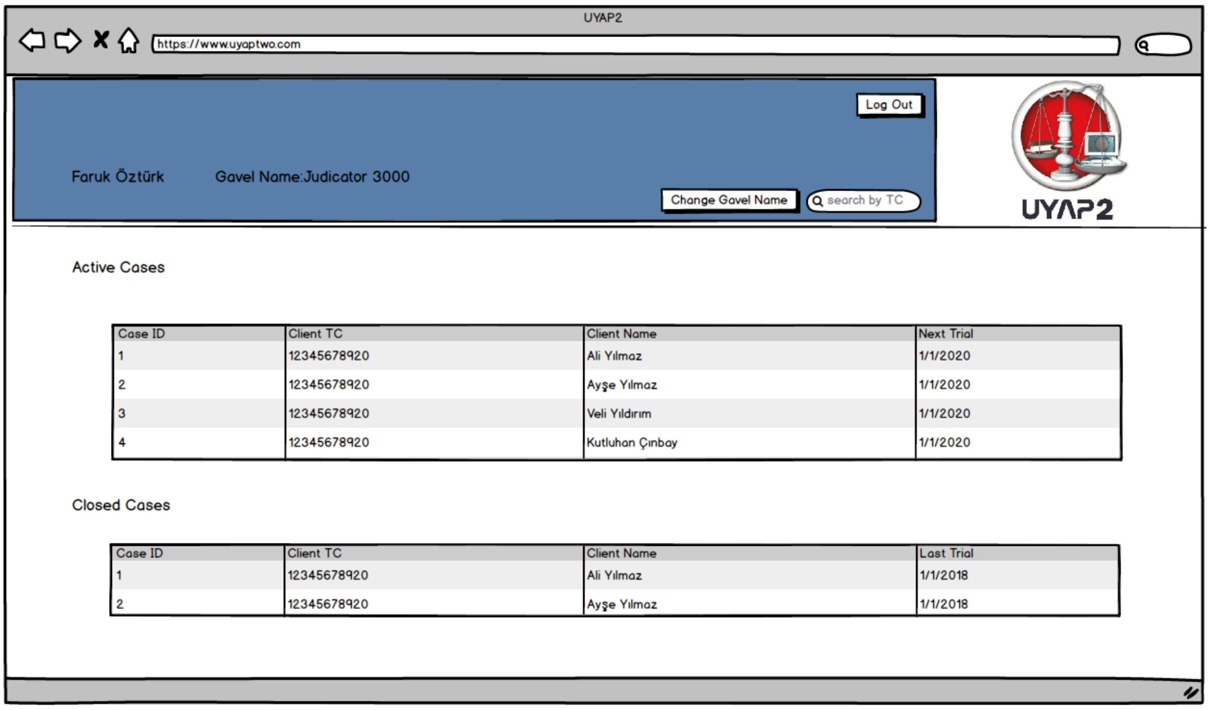
```
WITH Some_Translator AS (
	SELECT *
	FROM Translates
	WHERE Translates.Language_Name = &LANGUAGE AND rownum = 1
)
INSERT INTO Works
VALUES(Some_Translator.TC_ID, &CASE_ID)
```

Judge Expert Witness Request

```
WITH Some_Expert AS (
	SELECT *
	FROM Expert_Witness
	WHERE Expertise_Field = &EXPERTISE AND rownum = 1
)
INSERT INTO Informs
VALUES(Some_Expert.TC_ID, &CASE_ID)
```

[This space has been intentionally left blank]

6.11 Judge Portal Main Screen



Faruk Öztürk Gavel Name:Judicator 3000

Active Cases

Case ID	Client TC	Client Name	Next Trial
1	12345678920	Ali Yılmaz	1/1/2020
2	12345678920	Ayşe Yılmaz	1/1/2020
3	12345678920	Veli Yıldırım	1/1/2020
4	12345678920	Kutluhan Çınbay	1/1/2020

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2018
2	12345678920	Ayşe Yılmaz	1/1/2018

Input: &TC_ID(Judge), &GAVEL_NAME

Process: This screen shows the Court cases affiliated with the Judge. This view requires fetching the information present for this particular Judge. Judges can change the name of their gavels from this page. The user can also search the database using a TC ID and can log out.

SQL Statements:

The query is similar to the other main screens of other actors. Alternatively the following stored procedure may be used to fetch Judge summary.

EXEC Fetch_Open_Judge_Summary(&TC_ID)

EXEC Fetch_Closed_Judge_Summary(&TC_ID)

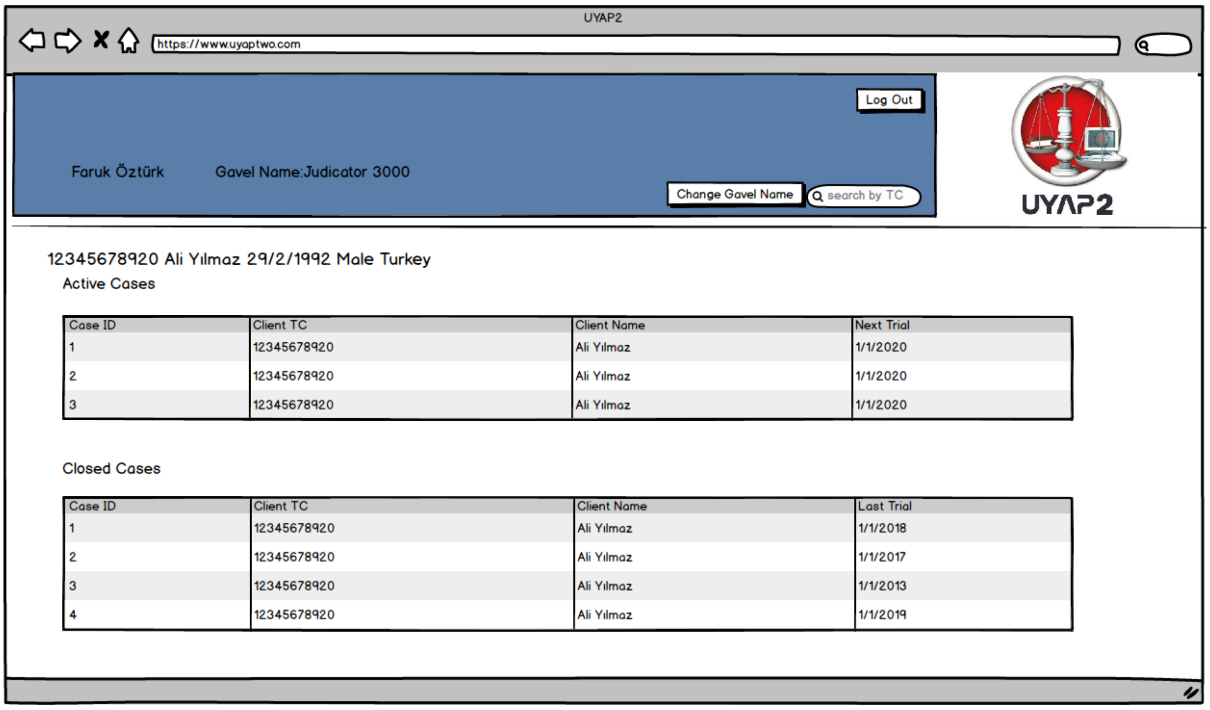
Gavel Name Update

UPDATE Judge

SET Gavel_Name = &GAVEL_NAME

WHERE &TC_ID = Judge.TC_ID

6.12 Search View



12345678920 Ali Yılmaz 29/2/1992 Male Turkey

Active Cases

Case ID	Client TC	Client Name	Next Trial
1	12345678920	Ali Yılmaz	1/1/2020
2	12345678920	Ali Yılmaz	1/1/2020
3	12345678920	Ali Yılmaz	1/1/2020

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2018
2	12345678920	Ali Yılmaz	1/1/2017
3	12345678920	Ali Yılmaz	1/1/2013
4	12345678920	Ali Yılmaz	1/1/2019

Input: &TC_ID(Judge), &TC_ID(Citizen), &GAVEL_NAME

Process: This screen shows the Court cases affiliated with the searched TC_ID. This view requires fetching the information present for this particular Citizen. Judges can change the name of their gavels from this page. The user can also search the database using a TC ID and can log out.

SQL Statements:

The query is similar to the other main screens of other actors. Alternatively the following stored procedure may be used to fetch summary of a particular citizen.

EXEC Fetch_Open_Citizen_Summary(&TC_ID)

EXEC Fetch_Closed_ Citizen _Summary(&TC_ID)

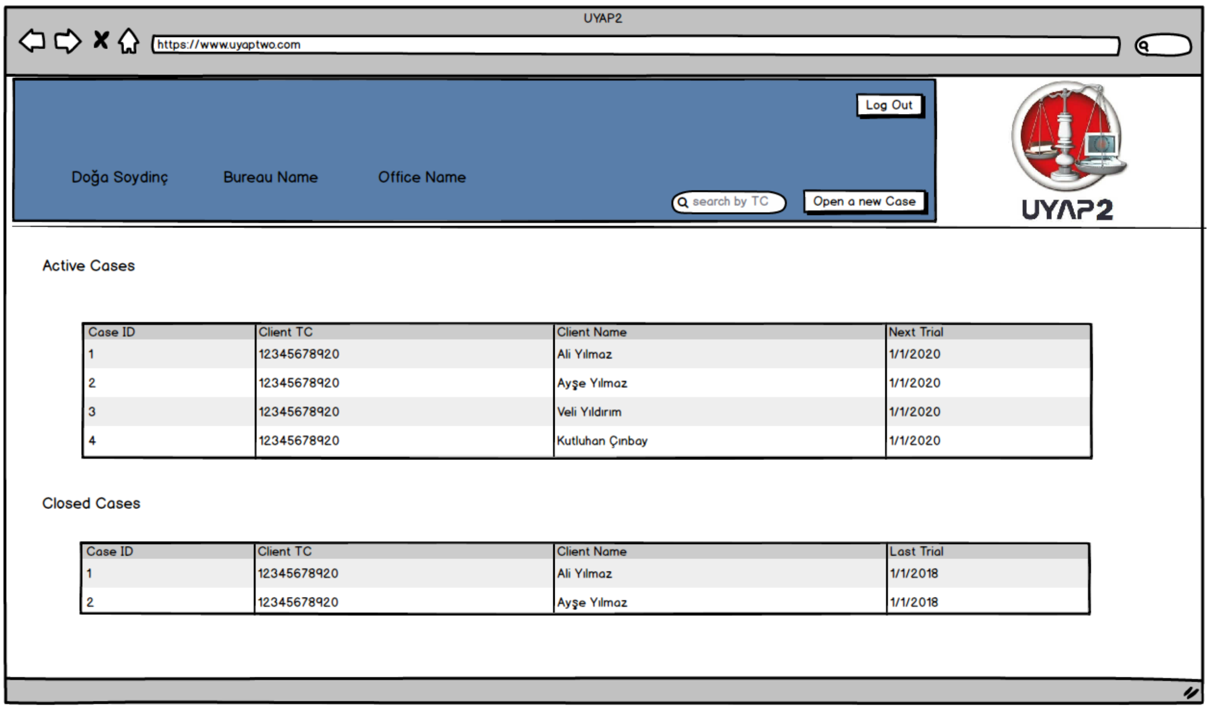
Gavel Name Update

UPDATE Judge

SET Gavel_Name = &GAVEL_NAME

WHERE &TC_ID = Judge.TC_ID

6.13 Lawyer Portal Main Screen



Active Cases

Case ID	Client TC	Client Name	Next Trial
1	12345678920	Ali Yılmaz	1/1/2020
2	12345678920	Ayşe Yılmaz	1/1/2020
3	12345678920	Veli Yıldırım	1/1/2020
4	12345678920	Kutluhan Çınbay	1/1/2020

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2018
2	12345678920	Ayşe Yılmaz	1/1/2018

Input: &TC_ID(Lawyer)

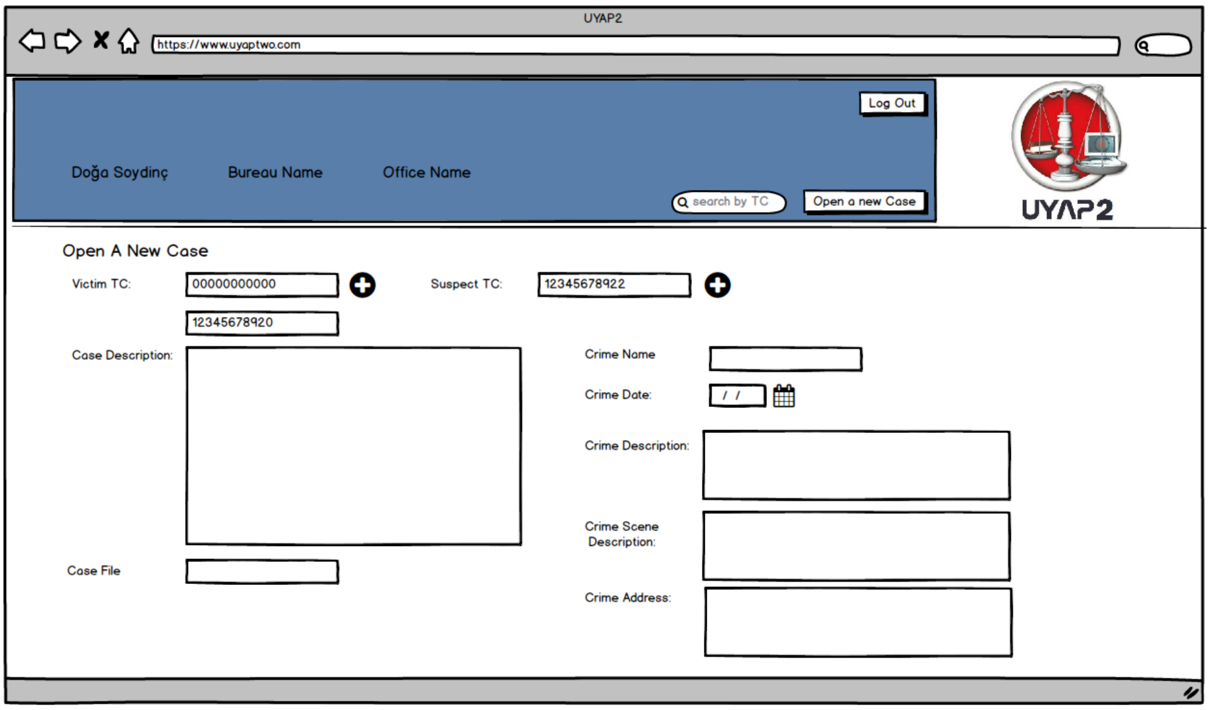
Process: This screen is almost identical to the Judge main screen except that the Lawyers have an additional permission to open a new case. They also cannot change the name of their gavel since they don't have one.

SQL Statements:

Same queries as the generic summary screen.

[This space has been intentionally left blank]

6.14 Lawyer New Case View



Open A New Case

Victim TC: 00000000000
12345678920

Suspect TC: 12345678922

Case Description:

Case File

Crime Name

Crime Date: / /

Crime Description:

Crime Scene Description:

Crime Address:

Input: &TC_ID(Lawyer)

Process: In this page, Lawyers can open new cases by inputting the TC ID's of victims which they represent. They must input the appropriate information in order to successfully open a new case.

They also can search for Citizens using their TC ID's and can also log out of the system.

SQL Statements:

BEGIN

SAVEPOINT INITIAL_STATE

INSERT INTO Court_Case

VALUES(&NEW_CASE_ID, &CASE_DESC, &CASE_FILE, 'FALSE', '-', &LAWYER_ID, &COURT_NAME, NULL, &NEW_CRIME_ID)

INSERT INTO Involved

VALUES(&LITIGANT_TC, &NEW_CASE_ID, "Victim")

INSERT INTO Represents

(&LAWYER_ID, &LITIGANT_ID, &NEW_CASE_ID)

INSERT INTO Involved

VALUES(&LITIGANT_TC, &NEW_CASE_ID, "Victim")

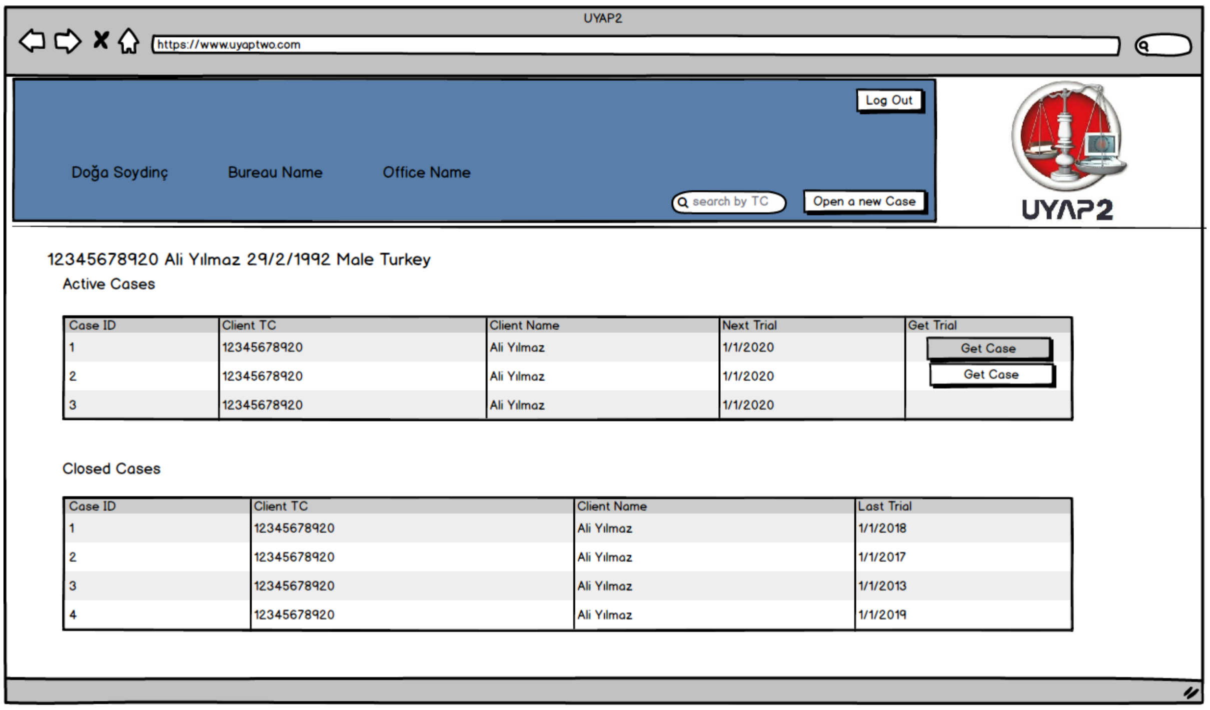
INSERT_INTO Crime

VALUES(&NEW_CRIME_ID, &CRIME_DATE, &CRIME_SCENE_DESC, &CRIME_DESC, &CRIME_NAME)

```
        INSERT INTO Address
        VALUES(&CRIME_ADDRESS_ID, &CITY, &STREET_NAME, &STREET_ADDRESS)
        INSERT INTO Commited_At
        VALUES(&NEW_CRIME_ID, &CRIME_ADDRESS_ID)
EXCEPTION
        WHEN OTHERS THEN
        ROLLBACK TO INITIAL_STATE
        RAISE;
END;
```

[This space has been intentionally left blank]

6.15 Lawyer Search View



12345678920 Ali Yılmaz 29/2/1992 Male Turkey

Active Cases

Case ID	Client TC	Client Name	Next Trial	Get Trial
1	12345678920	Ali Yılmaz	1/1/2020	Get Case
2	12345678920	Ali Yılmaz	1/1/2020	Get Case
3	12345678920	Ali Yılmaz	1/1/2020	

Closed Cases

Case ID	Client TC	Client Name	Last Trial
1	12345678920	Ali Yılmaz	1/1/2018
2	12345678920	Ali Yılmaz	1/1/2017
3	12345678920	Ali Yılmaz	1/1/2013
4	12345678920	Ali Yılmaz	1/1/2019

Input: &TC_ID(Lawyer),TC_ID(Citizen)

Process: In this page, Lawyers can see the cases affiliated with the searched TC_ID and Get the case to represent the person in question. They also can search for Citizens using their TC ID's, Open a new case and can also log out of the system.

SQL Statements:

Become a Lawyer for the Case for a Victim

INSERT Represents

VALUES(&TC_ID, &LITIGANT_TC_ID, &CASE_ID)

[This space has been intentionally left blank]

7. Advanced Database Components

7.1 Citizen Case Summary Views

A citizen should not be allowed access to the entirety of the database, and the citizen in question should only be able to see the cases that are directly related to them. Therefore, when a citizen logs in the system, the following views will be used.

Inputs: &TC_ID (TC_ID of the currently logged in user)

7.1.1 Ongoing Case Suspect View

```
CREATE VIEW Open_Suspect_Citizen_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        )

        SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date

        FROM Involved, Citizen, Next_Trial, Court_Case

        WHERE Involved.Litigant_ID Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND
Involved.Role = “Suspect” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 0
```

[This space has been intentionally left blank]

7.1.2 Ongoing Case Victim View

```
CREATE VIEW Open_Victim_Citizen_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        )

        SELECT Involved.Case_ID, Citizen.TC_ID AS TC, Citizen.FullName, Next_Trial.T_Date

        FROM Involved, Citizen, Next_Trial, Court_Case

        WHERE Involved.Litigant_ID AND Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID
AND Involved.Role = “Victim” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed
= 0
```

7.1.3 Closed Case Suspect View

```
CREATE VIEW Closed_Suspect_Citizen_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        )

        SELECT Involved.Case_ID, Citizen.TC_ID, Citizen.FullName, Next_Trial.T_Date

        FROM Involved, Citizen, Next_Trial, Court_Case

        WHERE Involved.Litigant_ID = Citizen.TC_ID AND Involved.Case_ID = Next_Trial.Case_ID AND
Involved.Role = “Suspect” AND Court_Case.Case_ID = Involved.Case_ID AND Court_Case.IsClosed = 1
```

7.1.4 Closed Case Victim View

```
CREATE VIEW Closed_Victim_Citizen_Summary AS
        WITH Next_Trial AS (
                (SELECT Case_ID, T_Date
                FROM Trial_Date)
                EXCEPT
                (SELECT T1.Case_ID, T2.T_Date
                FROM Trial_Date AS T1, Trial_Date AS T2
                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)
        )
        SELECT Involved.Case_ID, Citizen.TC_ID AS TC, Citizen.FullName, Next_Trial.T_Date
        FROM Involved, Citizen, Next_Trial, Court_Case
        WHERE Involved.Litigant_ID = &TC_ID AND Citizen.TC_ID = &TC_ID AND Involved.Case_ID =
Next_Trial.Case_ID AND Involved.Role = "Victim" AND Court_Case.Case_ID = Involved.Case_ID AND
Court_Case.IsClosed = 1
```

7.2 Generic Case Summary Views

The following views also attempt to limit the summaries certain roles have access to in the system. The only difference is that this view will be available to all roles except the citizen, since all other roles can not be marked as "Suspects" or "Victims".

Lawyers have a slightly different view: They see their client in the TC_ID and Name fields of the summary.

The rest of the actors have the same generic view of the court case summary, but with the name of one of the victims in the TC_ID and the Name fields.

[This space has been intentionally left blank]

7.2.1 Lawyer Case Summary Views

Current Cases

```
CREATE VIEW Open_Lawyer_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        )

        SELECT Represents.Case_ID, Represents.Lawyer_ID as TC, Citizen.TC_ID, Citizen.FullName,
Next_Trial.T_Date

        FROM Represents, Next_Trial, Citizen, Court_Case

        WHERE Represents.TC_ID = Citizen.TC_ID AND Next_Trial.Case_ID = Represents.Case_ID
AND Court_Case.Case_ID = Represents.Case_ID AND Court_Case.IsClosed = 0;
```

7.2.2 Past Cases

```
CREATE VIEW Closed_Lawyer_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        )

        SELECT Represents.Case_ID, Represents.Lawyer_ID as TC, Citizen.TC_ID, Citizen.FullName,
Next_Trial.T_Date

        FROM Represents, Next_Trial, Citizen, Court_Case

        WHERE Represents.TC_ID = Citizen.TC_ID AND Next_Trial.Case_ID = Represents.Case_ID
AND Court_Case.Case_ID = Represents.Case_ID AND Court_Case.IsClosed = 1;
```

7.2.3 Generic Case Summary Views

```
CREATE VIEW Open_Judge_Summary AS

        WITH Next_Trial AS (

                (SELECT Case_ID, T_Date

                FROM Trial_Date)

                EXCEPT

                (SELECT T1.Case_ID, T2.T_Date

                FROM Trial_Date AS T1, Trial_Date AS T2

                WHERE T1.Case_ID = T2.Case_ID AND T1.T_Date < T2.T_Date)

        ),

        SELECT Judges.TC_ID AS TC, Judges.Case_ID, MAX(Citizen.TC_ID), Citizen.FullName,
Next_Trial.T_Date

        FROM Judges, Citizen, Involved, Next_Trial, Court_Case

        WHERE Judges.Case_ID = Court_Case.Case_ID AND Involved.TC_ID = Citizen.TC_ID AND
Involved.Case_ID = Court_Case.Case_ID AND Next_Trial.Case_ID = Court_Case.Case_ID AND
Court_Case.IsClosed = 0

        GROUP BY Judges.Case_ID, Citizen.FullName, Next_Trial.T_Date
```

There will be similar views to this summary, with different actors, and for open/closed cases. The only difference in the CREATE VIEW expressions for those views is that the information comes from different tables. For eye witnesses, the information comes from the Testifies table, for expert witnesses, it comes from the Informs table, and for Interpreters, the information comes from the Works table.

For closed cases, the check for closedness is Court_Case.IsClosed = 1. In total, the remaining 4 actors (judges, eye witnesses, interpreters, expert witnesses) will have 8 views in total, resembling the one above.

[This space has been intentionally left blank]

7.3 Detailed Case Views

The detailed case view will be available to all users who are currently participating in a court case. Since this is a very common query to the database, and thus it is practical to have it as a view. It is even more practical to have all of these as “modular” views, and request them in a stored procedure operation from the database. The entirety of these views will be used to obtain detailed information from the database about a particular case.

Inputs: &CASE_ID (Case ID of a particular court case)

```
CREATE VIEW Judge_Info AS (

        SELECT Court_Case.Case_ID AS CASE_ID, Judge.TC_ID, Citizen.FullName, Citizen.BirthDate,
Citizen.Biological_Sex, Citizen.Nationality, Judge.Gavel_Name

        FROM Citizen, Judge, Judges, Court_Case

        WHERE Citizen.TC_ID = Judges.TC_ID AND Judge.TC_ID = Judges.TC_ID AND Judges.Case_ID
= Court_Case.CASE_ID

)
```

```
CREATE VIEW Court_Case_Info AS (

        SELECT Court_Case.Case_ID, Court_Case.Description, Court_Case.Case_file,
Court_Case.IsClosed, Court_Case.Result, Court_Case.Court_Name, Court.Court_Type,
Address.Street_Name,  Address.City, Address.Street_Address

        FROM Court_Case, Court, Address

        WHERE Court.Court_Name = Court_Case.Court_Name AND Court.Address_ID =
Address.Address_ID

)
```

```
CREATE VIEW Victims AS (

        SELECT Involved.Case_ID AS CASE_ID, C1.TC_ID, C1.FullName, C1.BirthDate,
C1.Biological_Sex, C1.Nationality, C2.TC_ID, C2.FullName

        FROM Involved, Citizen AS C1, Citizen AS C2, Represents

        WHERE Involved.TC_ID = C1.TC_ID AND Represents.TC_ID = Involved.TC_ID AND
Involved.Role = “Victim” AND C2.TC_ID = Represents.Lawyer_ID AND Involved.Case_ID =
Represents.Case_ID

)
```

```
CREATE VIEW Suspects AS (

        SELECT Involved.Case_ID AS CASE_ID, C1.TC_ID, C1.FullName, C1.BirthDate,
C1.Biological_Sex, C1.Nationality, C2.TC_ID, C2.FullName

        FROM Involved, Citizen AS C1, Citizen AS C2, Represents

        WHERE Involved.TC_ID = C1.TC_ID AND Represents.TC_ID = Involved.TC_ID AND
Involved.Role = “Suspect” AND C2.TC_ID = Represents.Lawyer_ID AND Involved.Case_ID =
Represents.Case_ID

)


CREATE VIEW Crime_Info AS (

        SELECT Case_ID, Crime_ID, Crime_Date, Crime_Scene_Description, Crime_Description,
Crime_Name

        FROM Court_Case NATURAL JOIN Crime

)


CREATE VIEW Trial_Dates AS (

        SELECT Case_ID, T_Date

        FROM Trial_Dates

        ORDER BY T_Date

)


CREATE VIEW Interpreter_Info AS (

        SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID

        FROM Citizen, Works

        WHERE Citizen.TC_ID = Works.TC_ID
)


CREATE VIEW Eye_Witness_Info AS (

        SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID

        FROM Citizen, Testifies

        WHERE Citizen.TC_ID = Testifies.TC_ID
)
```

```
CREATE VIEW Conciliator_Info AS (
        SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID
        FROM Citizen, Councils
        WHERE Citizen.TC_ID = Councils.TC_ID
)


CREATE VIEW Expert_Info AS (
        SELECT TC_ID, FullName, BirthDate, Biological_Sex, Nationality, Case_ID
        FROM Citizen, Informs
        WHERE Citizen.TC_ID = Informs.TC_ID
)
```

7.4 Stored Procedures

7.4.1 Stored Procedures for Suspect and Victim Insertion

Stored procedures will come in handy for suspect and victim insert operations to the Involved table, since this procedure will be repeated many times, not only for different cases, but also multiple times for a single case as well. A procedure will be used to loop over all the insertions into the database when a case is being created by a lawyer. The insertion procedure is the same and common for all case creations. Having this as a stored procedure will increase the efficiency in the system by reducing the number of times the database system and the client communicates by having the insertions executed as a quick batch job instead of continually querying the database management system for the insertion of new tuples, hence reducing the latency for large cases.

7.4.2 Stored Procedures for Detailed Case View

The detailed case view simply fetches multiple views, and the views fetched for this purpose are always the same. Therefore, having this fetch operation as a stored procedure will also increase the efficiency of data flow and allow us to reduce our cognitive load during implementation. Also, while fetching the tables in practice, the following procedures will be used instead of the queries provided in the UI part. Given a good trigger implementation, this approach will allow good caching of joined tables, which will decrease latency significantly.

The following are the procedures for detailed case views in Oracle SQL syntax:

```
CREATE PROCEDURE Fetch_Judge_Info(&CASE_ID IN NUMBER) AS
SELECT *
FROM Judge_Info
WHERE CASE_ID = &CASE_ID
GO;
```

```
CREATE PROCEDURE Fetch_Victim_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Victims

WHERE CASE_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Suspect_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Suspects

WHERE CASE_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Crime_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Crime_Info

WHERE Case_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Trial_Dates(&CASE_ID IN NUMBER) AS

SELECT *

FROM Trial_Dates

WHERE Case_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Interpreter_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Interpreter_Info

WHERE Case_ID = &CASE_ID

GO;
```

```
CREATE PROCEDURE Fetch_Eye_Witness_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Eye_Witness_Info

WHERE Case_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Conciliator_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Conciliator_Info

WHERE Case_ID = &CASE_ID

GO;


CREATE PROCEDURE Fetch_Expert_Info(&CASE_ID IN NUMBER) AS

SELECT *

FROM Expert_Info

WHERE Case_ID = &CASE_ID

GO;
```

7.4.3 Stored Procedures for Summary Views

```
CREATE PROCEDURE Fetch_Open_Suspect_Summary(&TC_ID IN NUMBER) AS

SELECT *

FROM Open_Suspect_Citizen_Summary

WHERE TC = &TC_ID

GO;


CREATE PROCEDURE Fetch_Open_Victim_Summary (&TC_ID IN NUMBER) AS

SELECT *

FROM Open_Victim_Citizen_Summary

WHERE TC = &TC_ID

GO;
```

```
CREATE PROCEDURE Fetch_Closed_Victim_Summary (&TC_ID IN NUMBER) AS
SELECT *
FROM Closed_Victim_Citizen_Summary
WHERE TC = &TC_ID
GO;

CREATE PROCEDURE Fetch_Closed_Suspect_Summary (&TC_ID IN NUMBER) AS
SELECT *
FROM Closed_Suspect_Citizen_Summary
WHERE TC = &TC_ID
GO;

CREATE PROCEDURE Fetch_Open_Lawyer_Summary(&TC_ID IN NUMBER) AS
SELECT *
FROM Open_Lawyer_Summary
WHERE TC = &TC_ID
GO;

CREATE PROCEDURE Fetch_Closed_Lawyer_Summary(&TC_ID IN NUMBER) AS
SELECT *
FROM Closed_Lawyer_Summary
WHERE TC = &TC_ID
GO;

CREATE PROCEDURE Fetch_Open_Judge_Summary(&TC_ID IN NUMBER) AS
SELECT *
FROM Open_Judge_Summary
WHERE TC = &TC_ID
GO;
```

7.5 Reports

We will provide the following reports in our system.

7.5.1 Crimes per capita for a given Year

Input = @Year

```
WITH (SELECT YEAR(date) AS Year ,Count(Crime_ID)
        FROM Crime
        GROUP BY Year
        ) AS Crimes_Per_Year(Year,Count);
Select*
FROM Crimes_Per_Year,(SELECT Count(TC_ID) AS Capita
                        FROM (SELECT*
                                FROM Citizen
                                WHERE YEAR(Birthdate)  < @Year))
WHERE Year = @Year;
```

7.5.2 Average case per lawyer

```
WITH(SELECT Lawyer_ID AS TC_ID, COUNT(*) AS Count
        FROM Represents
        GROUP BY Lawyer_ID) AS Case_Per_Lawyer(TC_ID,Count);
SELECT *
FROM (SELECT COUNT(TC_ID)
        FROM Case_Per_Lawyer), (SELECT SUM(Count)
                                FROM Case_Per_Lawyer);
```

7.5.3 Average case per judge

```
WITH(SELECT TC_ID,COUNT(*) AS Count
        FROM Judges
        GROUP BY TC_ID) AS Case_Per_Judge(TC_ID,Count);
SELECT *
FROM (SELECT COUNT(TC_ID)
        FROM Case_Per_Judge), (SELECT SUM(Count)
                                FROM Case_Per_Judge);
```

7.5.4 Distribution of Crimes by Crime Name

```
SELECT *
FROM (SELECT COUNT(*) AS Total_Crime FROM Crime), (SELECT Crime_Name, COUNT(Crime_ID)
                                                   FROM Crime
                                                   GROUP BY Crime_Name);
```

7.5.5 Crimes by Nationality

```
SELECT *
FROM (SELECT COUNT(*) AS Total_Crime FROM Crime), (SELECT Nationality, Count(Case_ID)
                                                   FROM Involved NATURAL JOIN Citizen
                                                   WHERE Involved.role = Suspect
                                                   GROUP BY Nationality);
```

7.5.6 Crimes by Age Groups

```
SELECT *
FROM (SELECT COUNT(*) AS Total_Crime FROM Crime), (SELECT YEAR(Birthdate), Count(Case_ID)
                                                   FROM Involved NATURAL JOIN Citizen
                                                   WHERE Involved.role = Suspect
                                                   GROUP BY YEAR(Birthdate));
```

7.6 Triggers

- Triggers will be employed to update cached views. If a table is ever updated, the corresponding view is to be updated. If the RDBMS handles these cases, this trigger may not be needed.
- A trigger fires whenever someone changes their accepted status when trying to reconcile via a conciliator. If all parties agree, the system lets the conciliator to finalize the case. The detection of a consensus will be implemented via a trigger.

7.7 Constraints

- The system may not be used without a login
- The username for any user in any portal is their TC ID.
- The number of daily cases that a lawyer may start is 3, maximum.
- The number of cases that a non-citizen actor may be involved in is no more than 5, simultaneously.
- No tuple may ever be removed from the database. All past information about past court cases must be kept for legal reasons.
- A citizen may only see cases in which they are involved.

- A lawyer may not start a court case in which the said lawyer is a victim or a suspect.
- All citizens of the state are actually in the database. (Every citizen has an account whether they like it or not, and sign-up is a part of the project just because we had to simulate it)
- Eye witnesses are registered to the eye witness portal externally, and they are registered when they provide their testimony.
- Interpreters and Expert Witnesses have no extra actions that they can take in the system. They can only attend the trial.
- Interpreters and Expert Witnesses are assumed to be added externally to the system, presumably by the Ministry of Justice, with their respective attributes of fields of expertise and languages spoken.
- A conciliator may not finalize a case before all parties agree to the conciliation process.

7.8 Secondary Indexing

A secondary indexing is to be introduced for the views that were discussed in the Views section of this report. Whenever possible, TC_ID will be the primary index for the views, and where applicable, a secondary index on CASE_ID may be constructed. This will speed up the updates, queries, and searches on these views due to the speedup provided by the indices.

8. Implementation Plan

For our system functionalities and UI, we intend to employ HTML, CSS, JavaScript, and possibly PHP for our front-end implementation. The back-end implementation of the project will be carried out with Java Server Pages. For the RDBMS, we intend to use Oracle SQL.

9. Website

Project Webpage https://uensalo.github.io/Spring-2019-CS353-Group-21/