BİLKENT UNIVERSITY - ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CS 353 – Group 08

SPRING 2020

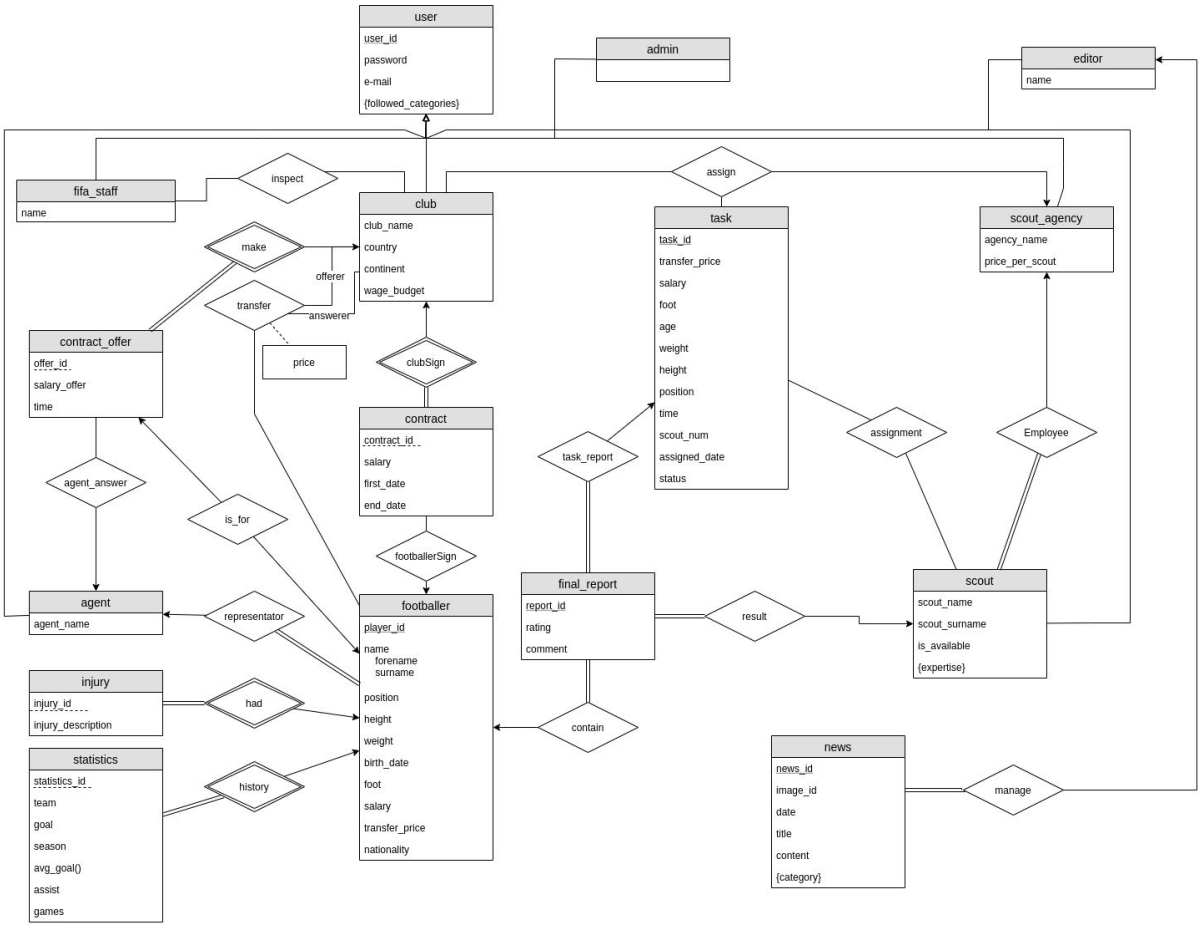
PROJECT DESIGN REPORT

Enes MERDANE
21600946
Mustafa Oğuz SATOĞLU
21702996
Emre Tolga AYAN
21703783
Şeyma Aybüke ERTEKİN
21602980

TA: Mustafa Çavdar

Table of Contents

1. Revised E/R Diagram



2. Table Schemas

- admin (<u>user_id</u>, password, email)
  - ✔ **primary key:** user_id
  - ✔ **candidate keys:** user_id, email
  - ✔ **functional dependencies:**

    user_id → password, email

    email → password, user_id
  - ✔ Both user_id and email are superkeys for the admin table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

```
create table admin(
        user_id         char(5),
        password        varchar(20) not null,
        email           varchar(50) not null,
        primary key (user_id)
);
```

- fifa_staff (<u>user_id</u>, password, email, name)
  - ✔ **primary key:** user_id
  - ✔ **candidate keys:** user_id, email
  - ✔ **functional dependencies:**

    user_id → password, email, name

    email → password, user_id, name

✓ Both user_id and email are superkeys for the fifa_staff table. Therefore, this relation schema is in BCNF and 3NF.

✓ **table definition:**

```
create table fifa_staff(
        user_id         char(5),
        password        varchar(20) not null,
        email           varchar(50) not null,
        name            varchar(80) not null,
        primary key (user_id)
);
```

- club (<u>user_id,</u> password, email, club_name, country, wage_budget)

✔ **primary key:** user_id

✔ **foreign key:** country *(to country_location table)*

✔ **candidate keys:** user_id, email, (club_name, country)

✔ **functional dependencies:**

user_id → club_name, country, wage_budget

email → club_name, country, wage_budget

club_name, country → user_id, wage_budget

✔ user_id, email, and (club_name, country) are superkeys for the club table. Therefore, this relation schema is in BCNF and 3NF.

✔ **table definition:**

```
create table club(
        user_id         char(5),
        password        varchar(20) not null,
```

email varchar(50) not null,

club_name varchar(50) not null,

country varchar(50),

wage_budget int,

primary key (user_id),

foreign key(country) references *country_location*

);

- country_location (<u>country</u>, continent)
  - ✓ **primary key:** country
  - ✓ **candidate keys:** country
  - ✓ **functional dependencies:**

    country → continent
  - ✓ country is a superkey for the country_location table. Therefore, this relation schema is in BCNF and 3NF.
  - ✓ **table definition:**

    create table country_location(

    country varchar(25),

    continent varchar(25),

    primary key (country)

    );
- scout_agency (<u>user_id,</u> password, email, agency_name, price_per_scout)
  - ✓ **primary key:** user_id
  - ✓ **candidate keys:** user_id, email, agency_name
  - ✓ **functional dependencies:**

user_id → password, email, agency_name, price_per_scout

agency_name → user_id, password, email, price_per_scout

email → user_id, password, agency_name, price_per_scout

✓ user_id, email, and agency_name are superkeys for the scout_agency table. Therefore, this relation schema is in BCNF and 3NF.

✓ **table definition:**

```
create table scout_agency(
        user_id          char(5),
        password         varchar(20) not null,
        email            varchar(50) not null,
        agency_name      varchar(50) not null,
        price_per_scout  int not null,
        primary key (user_id)
);
```

- agent (<u>user_id</u>, password, email, agent_name)

✓ **primary_key:** user_id

✓ **candidate keys:** user_id, email

✓ **functional dependencies:**

user_id → email, agent_name

email → user_id, agent_name

✓ Both user_id and email are superkeys for the agent table. Therefore, this relation schema is in BCNF and 3NF.

✓ **table definition:**

```
create table agent(
        user_id         char(5),
        password        varchar(20) not null,
        email           varchar(50) not null,
        agent_name      varchar(80) not null,
        primary key (user_id)
);
```

- editor (<u>user_id,</u> password, email, name)
  - ✓ **primary key:** user_id
  - ✓ **candidate keys:** user_id, email
  - ✓ **functional dependencies:**

    user_id □ password, email, name

    email □ user_id, password, name
  - ✓ Both user_id and email are superkeys for the editor table. Therefore, this relation schema is in BCNF and 3NF.
  - ✓ **table definition:**

```
create table editor(
        user_id         char(5),
        password        varchar(20) not null,
        email           varchar(50) not null,
        name            varchar(80) not null,
        primary key(user_id)
);
```

- user (<u>user_id</u>)

- ✓ **primary key:** user_id
- ✓ **candidate keys:** user_id
- ✓ **functional dependencies:** No functional dependency.
- ✓ **table definition:**

```
create table (
        user_id         char(5),
        primary key(user_id)
);
```

- followed_tags (<u>user_id, tag</u>)
  - ✓ **primary key:** (user_id, tag)
  - ✓ **candidate keys**: (user_id, tag)
  - ✓ **foreign key:** user_id *(to user table)*
  - ✓ **functional dependencies:** No functional dependency.
  - ✓ **table definition:**

```
create table followed_tags(
        user_id         char(5),
        tag             varchar(25),
        primary key(user_id, tag),
        foreign key(user_id) references user
);
```

- task (<u>task_id,</u> transfer_price, salary, foot, age, weight, height, position, time, scout_num, assigned_date, club_id, agency_id, status)
  - ✔ **primary key:** task_id
  - ✔ **candidate keys:** task_id

✔ **foreign_key:** club_id *(to club table)*, agency_id *(to scout_agency table)*

✔ **functional dependencies:**

task_id → transfer_price, salary, foot, age, weight, height, position, time, scout_num, status

✔ task_id is a superkey for the task table. Therefore, this relation schema is in BCNF and 3NF.

✔ **table definition:**

```
create table task(
        task_id             char(5),
        transfer_price      int,
        salary              int,
        foot                varchar(5) not null,
        age                 smallint not null,
        weight              smallint not null,
        height              smallint not null,
        position            varchar(10) not null,
        time                smallint not null,
        scout_num           smallint not null,
        assigned_date       varchar(10) not null,
        club_id             char(5) not null,
        agency_id           char(5) not null,
        status              varchar(10) not null,
        primary key (task_id),
        foreign key (club_id) references club,
```

foreign key (agency_id) references *scout_agency*

);

- contract_offer (<u>offer_id</u>, <u>club_id</u>, agent_id, salary_offer, time, player_id)
  - ✔ **primary_key:** (offer_id, club_id)
  - ✔ **candidate keys:** (offer_id, club_id)
  - ✔ **foreign key:** club_id *(to club table),* agent_id *(to agent table)*
  - ✔ **functional dependencies:**

    offer_id, club_id → salary_offer, time, agent_id

  - ✔ (offer_id, club_id) is a superkey for the contract_offer table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

```
create table contract_offer(
        offer_id            char(5),
        club_id             char(5),
        agent_id            char(5),
        salary_offer        int not null,
        time                smallint not null,
        player_id           char(5),
        primary key (offer_id, club_id),
        foreign key (club_id) references club,
        foreign key (agenct_id) references agent,
        foreign key (player_id) references footballer
);
```

- contract (<u>contract_id</u>, <u>club_id</u>, player_id, salary, first_date, end_date)
  - ✔ **primary_key:** (contract_id, club_id)
  - ✔ **candidate keys:** (contract_id, club_id)
  - ✔ **foreign_key:** club_id *(to club table),* player_id *(to footballer table)*
  - ✔ **functional dependencies:**

    contract_id, club_id → player_id, salary, first_date, end_date

  - ✔ (contract_id, club_id) is a superkey for the contract table. Therefore, this relation schema is in BCNF and 3NF.
- footballer (<u>player_id</u>, agent_id, forename, surname, position, height, weight, birth_date, foot, salary, transfer_price, nationality)
  - ✔ **primary_key:** player_id
  - ✔ **candidate keys:** player_id
  - ✔ **foreign_key:** agent_id *(to agent table)*
  - ✔ **functional dependencies:**

    player_id → agent_id, forename, surname, position, height, weight, birth_date, foot, salary, transfer_price, nationality

  - ✔ player_id is a superkey for the footballer table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

```
create table footballer(
        player_id       char(5),
        agent_id        char(5),
        forename        varchar(25) not null,
```

surname varchar(25) not null,

position varchar(10) not null,

height smallint not null,

weight smallint not null,

birth_date varchar(10) not null,

foot varchar(5) not null,

salary int,

transfer_price int,

nationality varchar(25)

primary key (player_id),

foreign key (agent_id) references *agent*

);

- statistics (<u>statistics_id</u>, <u>player_id</u>, team, goal, season, assist, games)
  - ✔ **primary_key:** (statistics_id, player_id)
  - ✔ **candidate keys:** (statistics_id, player_id)
  - ✔ **foreign key:** player_id *(to footballer table)*
  - ✔ **functional dependencies:**

    statistics_id, player_id → team, goal, season, assist, games

  - ✔ (statistics_id, player_id) is a superkey for the statistics table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

    create table statistics(

    statistics_id char(5),

    player_id char(5),

```
        team            varchar(25),
        goal            int not null,
        season          varchar(10) not null,
        assist          smallint not null,
        games           smallint not null,
        primary key (statistics_id, player_id),
        foreign key (player_id) references footballer
    );
```

- injury (injury_id, player_id, injury_description)
  - ✔ **primary_key**: (injury_id, player_id)
  - ✔ **candidate keys:** (injury_id, player_id)
  - ✔ **foreign_key:** player_id *(to footballer table)*
  - ✔ **functional dependencies:**

    injury_id, player_id → injury_description

  - ✔ (injury_id, player_id) is a superkey for the injury table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

```
create table injury(
        injury_id           char(5),
        player_id           char(5),
        injury_description  varchar(200) not null,
        primary key (injury_id, player_id),
        foreign key (player_id) references footballer
```

);

- final_report (report_id, player_id, task_id, scout_id, rating, comment)
  - ✔ **primary_key:** report_id
  - ✔ **candidate keys:** report_id
  - ✔ **foreign_key:** player_id *(to footballer table),* task_id *(to task table),* scout_id *(to scout table)*

    report_id → player_id, task_id, scout_id, rating, comment
  - ✔ report_id is a superkey for the final_report table. Therefore, this relation schema is in BCNF and 3NF.
  - ✔ **table definition:**

```
create table final_report(
        report_id           char(5),
        player_id           char(5),
        club_id             char(5),
        scout_id            char(5),
        rating              double-precision,
        comment             varchar(1000),
        primary key (report_id),
        foreign key (player_id) references footballer,
        foreign key (task_id) references task,
        foreign key (scout_id) references scout
);
```

- news (news_id, editor_id, date, title, content, image_id)

- ✔ **primary_key:** news_id
- ✔ **candidate keys:** news_id, (date, title, content, category)
- ✔ **foreign_key:** editor_id *(to editor table)*
- ✔ **functional dependencies:**

  news_id → editor_id, date, title, content, image_id

  date, title, content, category, image_id → news_id

- ✔ news_id and (date, title, content, category, image_id) are superkeys for the news table. Therefore, this relation schema is in BCNF and 3NF.
- ✔ **table definition:**

```
create table news(
        news_id          char(5),
        editor_id        char(5),
        date             date,
        title            varchar(25),
        content          varchar(1000),
        category         varchar(15),
        image_id         char(10),
        primary key (news_id),
        foreign key (editor_id) references editor
);
```

- categories (<u>news_id</u>, <u>category</u>)
  - ✔ **primary_key:** (news_id, category)
  - ✔ **candidate keys:** (news_id, category)

✔ **foreign_key:** news_id *(to news table)*

✔ **functional dependencies:**

news_id → → category

✔ *news_id → → category* is trivial, because news_id ∪ category = categories relation. Therefore, this relation schema is in BCNF and 4NF.

✔ **table definition:**

```
create table categories(
        news_id          char(5),
        category         varchar(100),
        primary key (news_id, category),
        foreign key (news_id) references news
);
```

- scout (<u>user_id</u>, password, email, agency_id, scout_name, scout_surname, is_available)

✔ **primary_key:** user_id

✔ **candidate keys:** user_id, email

✔ **foreign_key:** agency_id *(to scout_agency table),* task_id *(to task table)*

✔ **functional dependencies:**

user_id → password, email, agency_id, task_id, scout_name, scout_surname, is_available

email → password, email, agency_id, task_id, scout_name, scout_surname, is_available

✔ user_id and email are superkeys for the scout table. Therefore, this relation schema is in BCNF and 3NF.

✔ **table definition:**

```
create table scout(
        scout_id            char(5),
        password            varchar(20) not null,
        email               varchar(50) not null,
        agency_id           char(5),
        task_id             char(5),
        scout_name          varchar(25) not null,
        scout_surname       varchar(25) not null,
        is_available        boolean not null,
        primary key (scout_id),
        foreign key (agency_id) references scout_agency,
        foreign key (task_id) references task
);
```

- assignment (<u>scout_id, task_id</u>)

✔ **primary_key:** (scout_id, task_id)

✔ **candidate keys:** (scout_id, task_id)

✔ **functional dependencies:**

scout_id □□ task_id

✔ scout_id →→ task_id is trivial, because scout_id ∪ task_id = assignment relation schema. Therefore, this relation schema is in BCNF and 4NF.

✔ **table definition:**

```
create table scout_expertise(
        scout_id        char(5),
        task_id         char(20),
        primary key (scout_id, task_id),
        foreign key (scout_id) references scout
        foreign key (task_id) references task
);
```

- scout_expertise (<u>scout_id, expertise</u>)
  - ✔ **primary_key:** (scout_id, expertise)
  - ✔ **candidate keys:** (scout_id, expertise)
  - ✔ **functional dependencies:**

    scout_id □ □ expertise

  - ✔ scout_id → → expertise is trivial, because scout_id ∪ expertise = scout_expertise relation schema. Therefore, this relation schema is in BCNF and 4NF.
  - ✔ **table definition:**

```
create table scout_expertise(
        scout_id        char(5),
        expertise       varchar(20),
        primary key (scout_id, expertise),
        foreign key (scout_id) references scout
);
```

- inspect (<u>fifa_id, club_id</u>)
  - ✔ **primary_key:** (user_id, club_id)
  - ✔ **candidate keys:** (user_id, club_id)
  - ✔ **foreign_key:** fifa_id *(to fifa_staff table)*, club_id *(to club table)*
  - ✔ **functional dependencies:** *No functional dependency.*
  - ✔ **table definition:**

```
create inspect(
        fifa_id         char(5),
        club_id         char(5),
        primary key (fifa_id, club_id),
        foreign key (fifa_id) references fifa_staff,
        foreign key (club_id) references club
);
```

  - ✔

- transfer (<u>offerer_club_id, answerer_club_id</u>, <u>player_id</u>, price)
  - ✔ **primary_key:** (offerer_club_id, answerer_club_id, player_id)
  - ✔ **foreign_key:** offerer_club_id *(to club table)*, answerer_club_id *(to club table)*, player_id *(to footballer table)*
  - ✔ **candidate keys:** (offerer_club_id, answerer_club_id, player_id)
  - ✔ **functional dependencies:**

    offerer_club_id, answerer_club_id, player_id → price

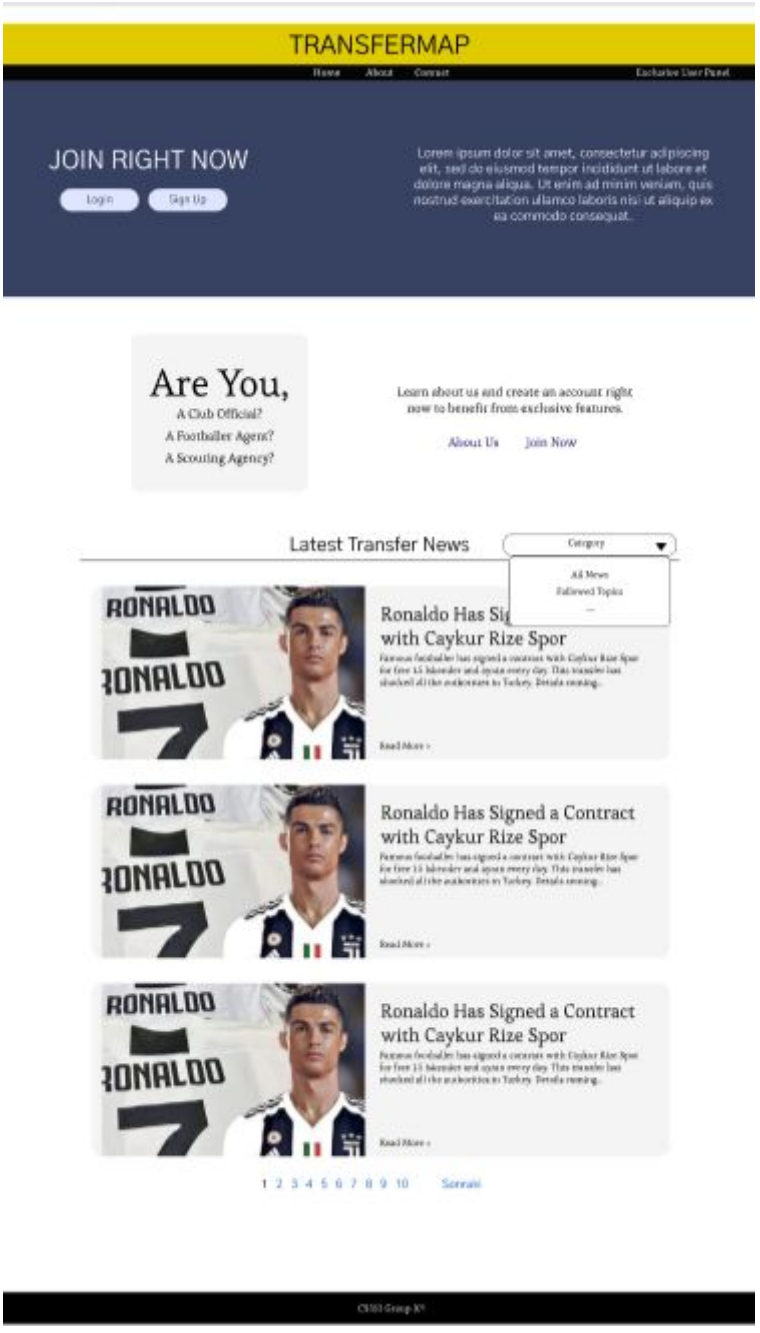
✔ *(offerer_club_id, answerer_club_id, player_id) is a superkey for the transfer table. Therefore, this relation schema is in BCNF and 3NF.*

✔ **table definition:**

```
create table transfer(
        offerer_club_id         char(5),
        answerer_club_id        char(5),
        player_id               char(5),
        price                   int,
        primary key (offerer_club_id, answerer_club_id, player_id),
        foreign key (offerer_club_id) references club,
        foreign key (answerer_club_id) references club,
        foreign key (player_id) references footballer
);
```

3. User Interface Design and SQL Statements

Main Page



In the news part, users will be able to see only a specific category. They will choose the category from the drop-down menu on the top right corner.

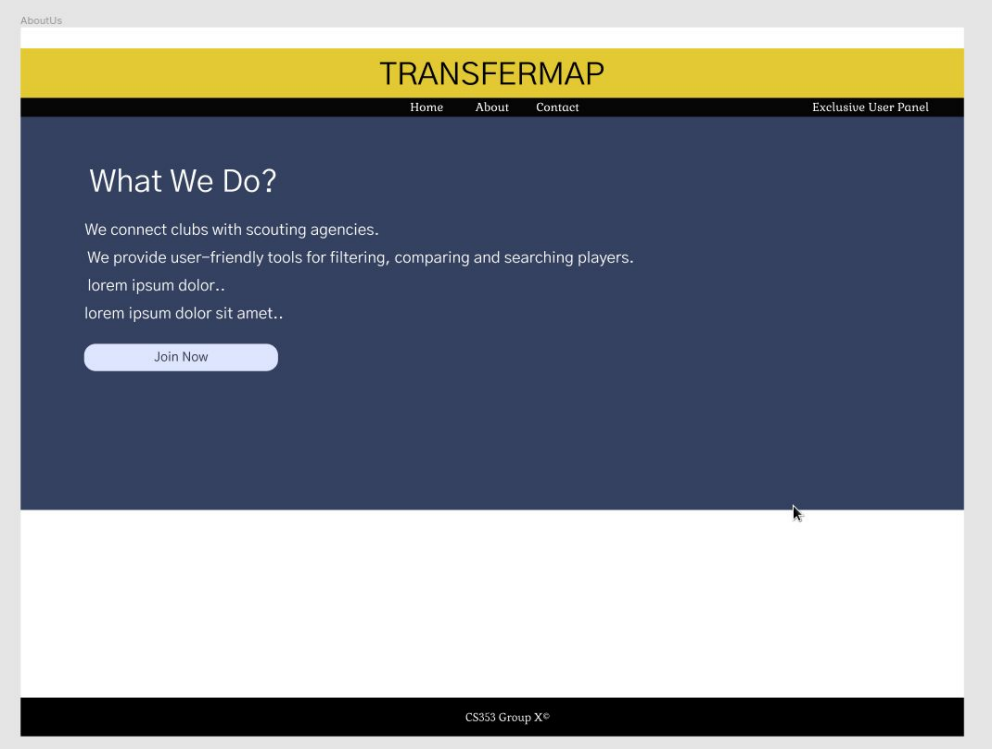
input: @category

SQL Statement:

To show news:

```
CREATE VIEW categorized_news AS
SELECT *
FROM news NATURAL JOIN categories
WHERE category = @category
ORDER BY date ASC;
```

About Us Page



Panel Menu for Managers

This page is for managers to navigate the page through different pages.

Panel Menu For Editors

This page is for editors to navigate the page through different pages.

Panel Menu For Clubs

This page is for clubs to navigate the page through different pages.

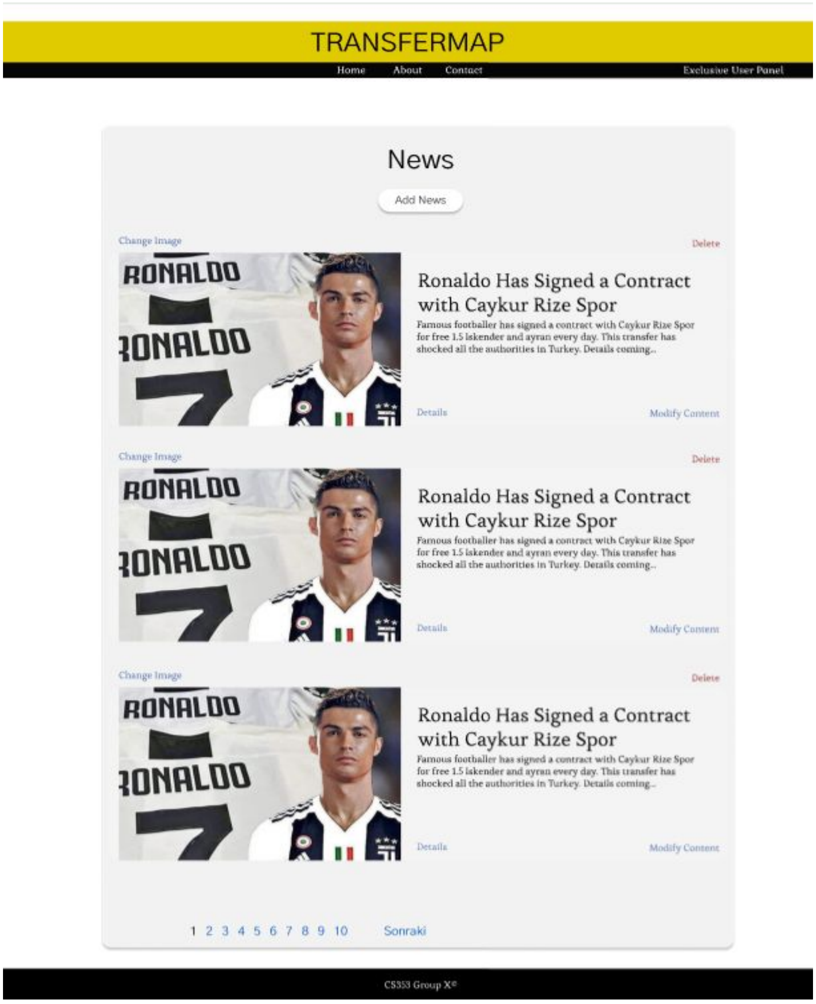
Panel Menu For Agencies

This page is for agencies to navigate the page through different pages.

Panel Menu For Scouts

This page is for scouts to navigate the page through different pages.

Editor Editing Page



If modify is clicked, the next page is opened filled, and data can be changed and saved. If add is clicked, next page is opened empty.

Input: @editor_id, @news_id

SQL Statements:

CREATE VIEW news_editor AS

SELECT *

FROM news

WHERE editor_id = @editor_id;

To delete news:

DELETE FROM news_editor

WHERE news_id = @news_id;

Editor Add News Page

input: @editor_id, @news_id, @title, @image_id, @content, @content, @category, @date

SQL Statement:

```
CREATE VIEW news_editor AS
SELECT *
FROM news
WHERE editor_id = @editor_id;
```

To add news:

```
INSERT INTO news_editor
VALUES (@news_id, @editor_id, @date, @title, @content, @image_id);

INSERT INTO categories
VALUES (@news_id, @category);
```

To update news:

```
UPDATE news_editor
SET date = @date
WHERE news_id = @news_id;

UPDATE news_editor
SET title = @title
WHERE news_id = @news_id;

UPDATE news_editor
SET content = @content
WHERE news_id = @news_id;

UPDATE news_editor
SET image_id = @image_id
WHERE news_id = @news_id;

INSERT INTO categories
VALUES(@news_id, @category);

DELETE FROM categories
WHERE news_id = @news_id AND category = @category;
```

My Account Page

This page's content varies with different types of users.

inputs: @name, @email, @password, @categories, @user_id, @price_per_scout

SQL Statements:

To update an editor account:

```
UPDATE editor
SET email = @email, password = @password, editor.name = @name
WHERE user_id = @user_id;
```

To update an agent account:

```
UPDATE agent
SET email = @email, password = @password, agent_name = @name
WHERE user_id = @user_id;
```

To update a club account:

```
UPDATE club
SET email = @email, password = @password, club_name = @name
WHERE user_id = @user_id;
```

To update a fifa account:

```
UPDATE fifa_staff
SET email = @email, password = @password, fifa_staff.name = @name
WHERE user_id = @user_id;
```

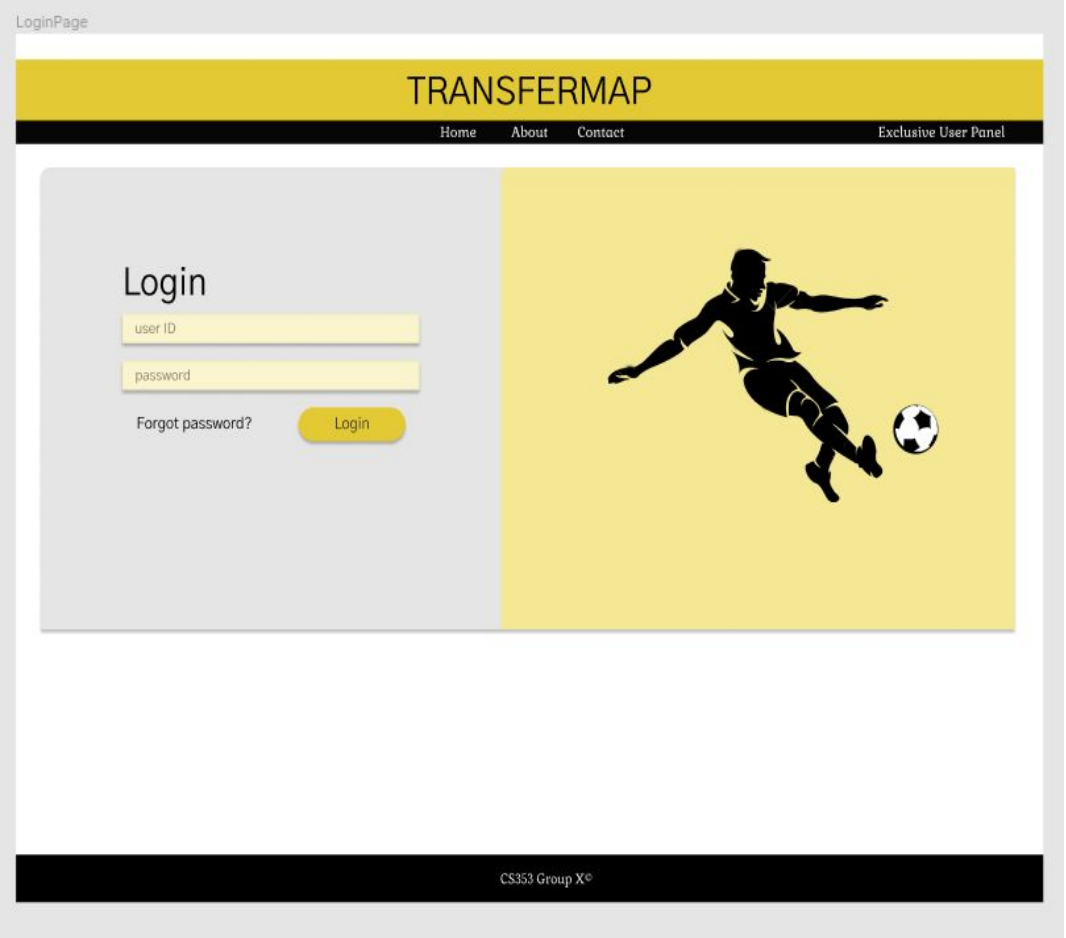
To update a scout account:

```
UPDATE scout
SET email = @email, password = @password, scout_name = @name
WHERE user_id = @user_id;
```

To update an agency account:

```
UPDATE scout_agency
SET email = @email, password = @password, agency_name = @name, price_per_scout =
@price_per_scout
WHERE user_id = @user_id;
```

Login Page



SQL Statement:

input: @user, @password

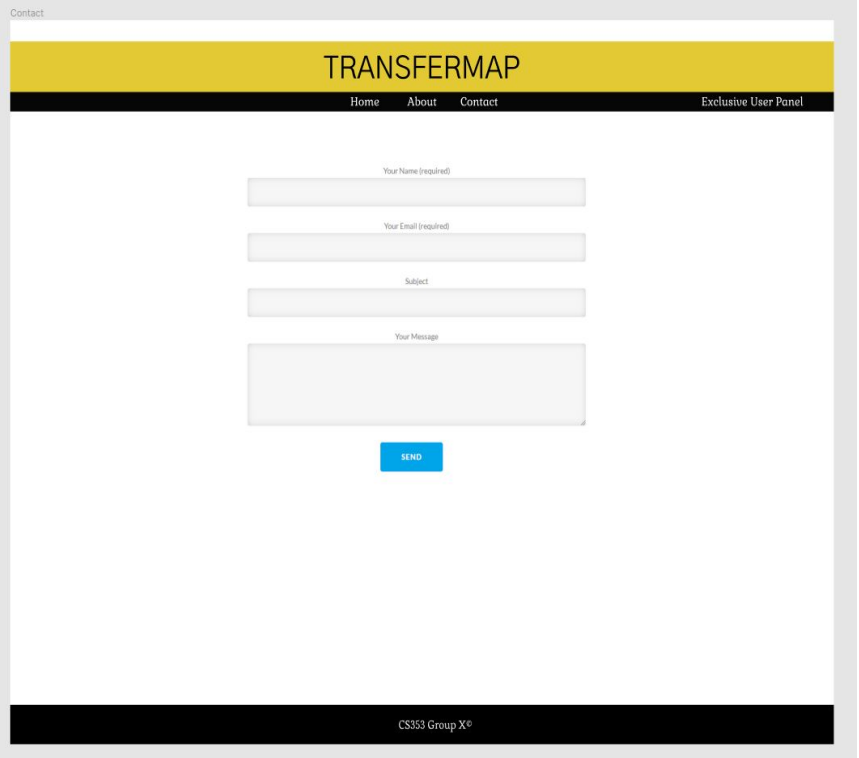
SELECT *

FROM user

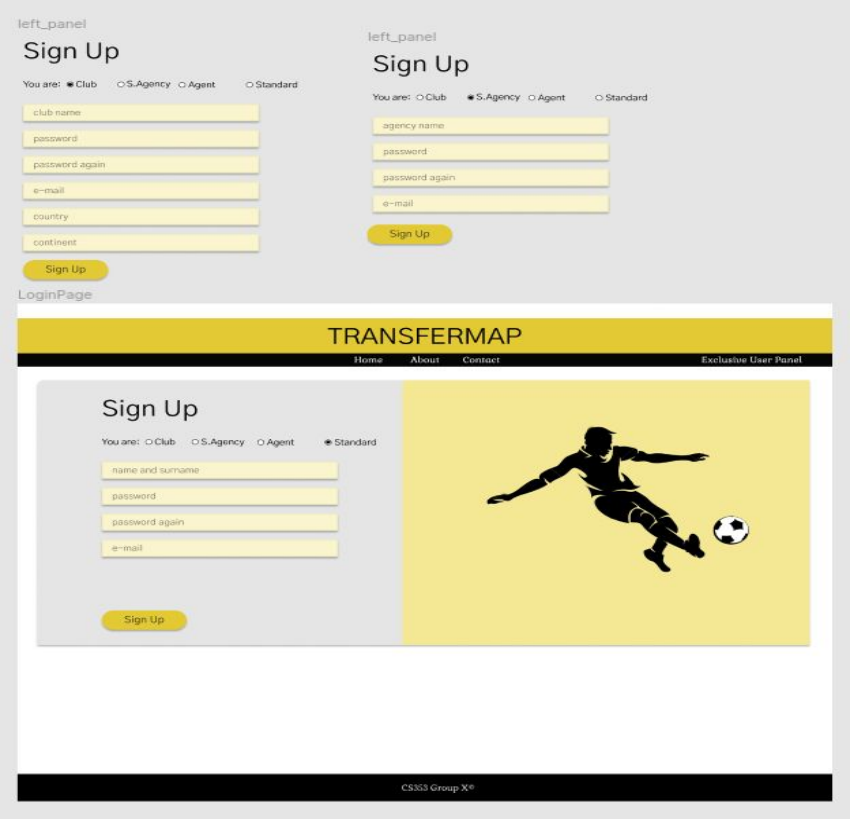
WHERE (user_id = @user OR email = @user)

AND password = @password;

Contact With Us Page



Sign Up page



That page is for signing up for the website.

Input: @user_id, @name_surname, @password, @e-mail, @country, @continent

SQL Statement:

To add a club account:

```
INSERT INTO club
VALUES (@user_id, @password, @email, @name_surname, @country);

INSERT_INTO country_location
VALUES(country, continent);
```

To add an agency account:

```
INSERT INTO scout_agency
VALUES (@user_id, @password, @email, @name_surname);
```

To add an agent account:

```
INSERT INTO agent
VALUES (@user_id, @password, @email, @name_surname);
```

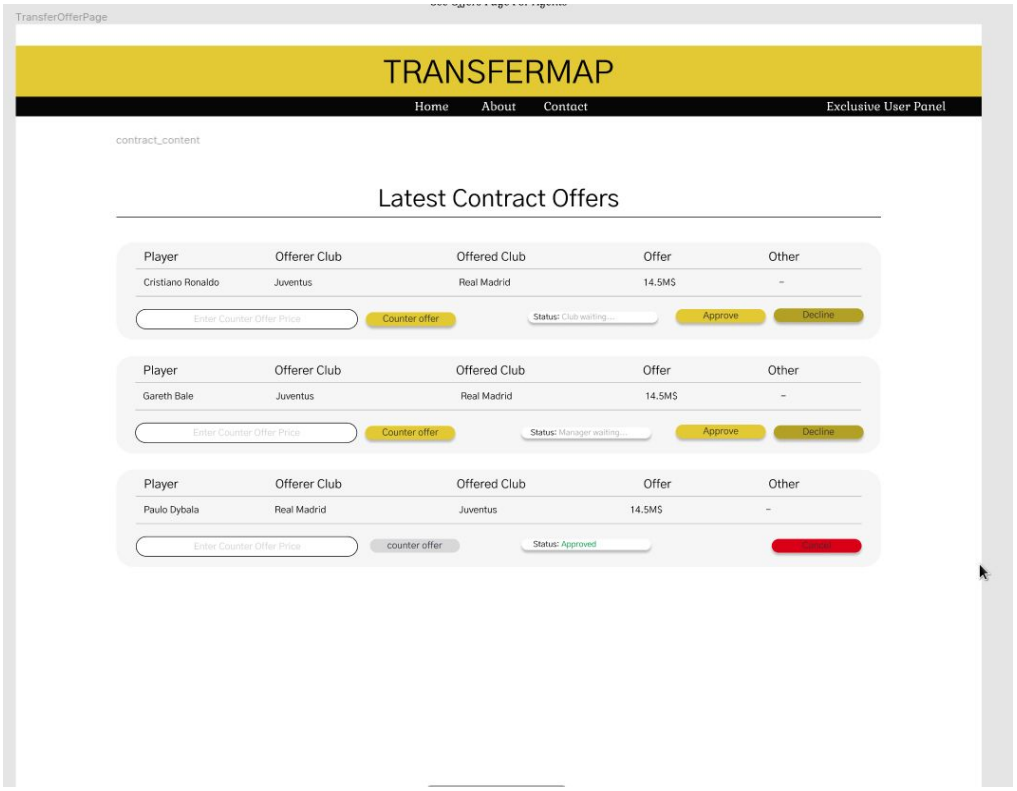
To add an editor account:

```
INSERT INTO editor
VALUES (@user_id, @password, @email, @name_surname);
```

To add an fifa account:

```
INSERT INTO fifa_staff
VALUES (@user_id, @password, @email, @name_surname);
```

Offers Page



Managers and clubs will see all the previous and current offers. Also the offers they made(for clubs). They can approve an offer, decline it, counteroffer, or cancel an offer.

inputs: @club_id, @agent_id

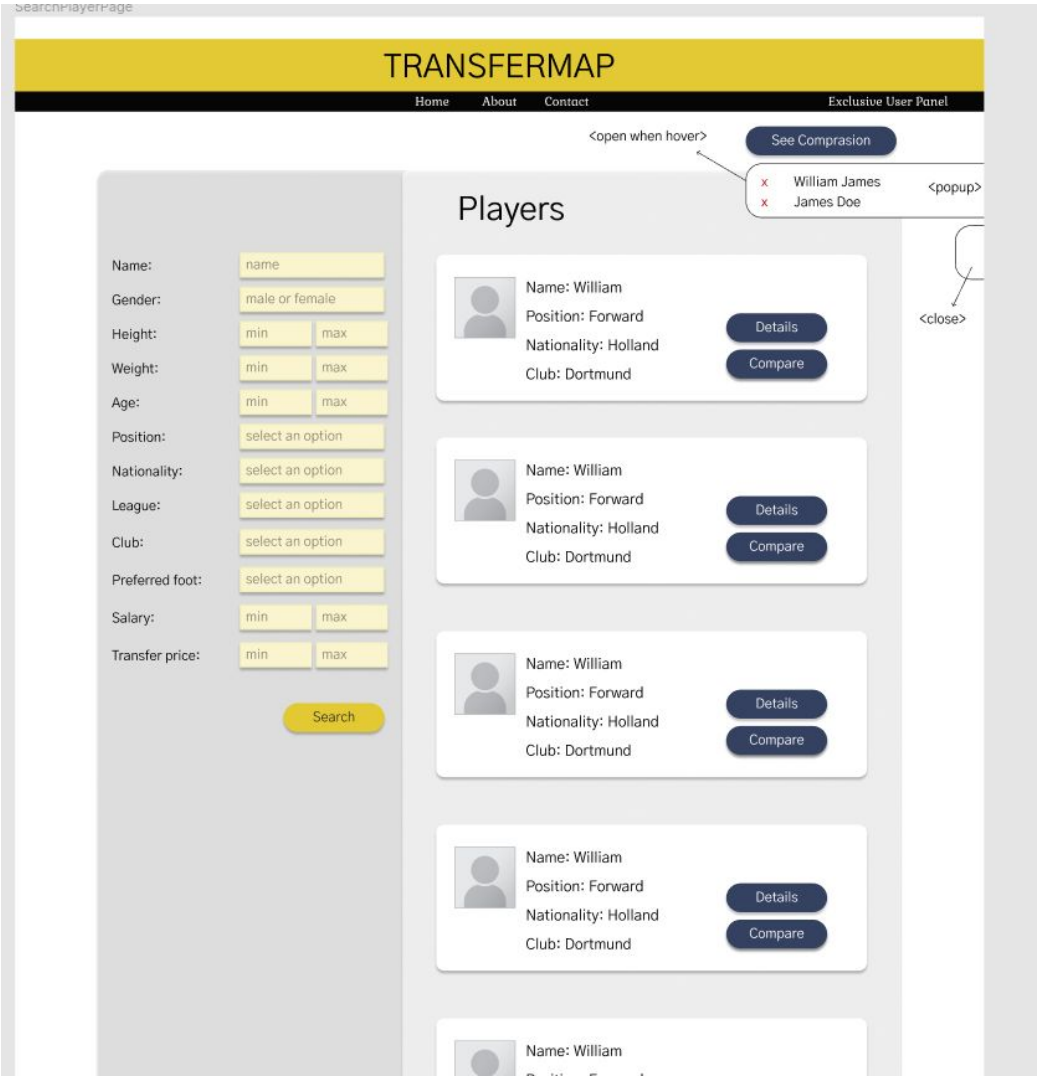
For clubs:

```
CREATE VIEW club_offers AS
SELECT *
FROM contract_offer NATURAL JOIN club
WHERE club_id = @club_id;
```

For agents:

```
CREATE VIEW agent_offers AS
SELECT *
FROM contract_offer NATURAL JOIN agent
WHERE agent_id = @agent_id;
```

Footballer Search Page



Clubs and scouts can make a search on this page. They can add players for comparison. Added players are shown in a popup menu in the top right corner that appears when hover. If the user clicks “see comparison”, the comparison page is opened for the players.

input: @player_id1, @player_id2, @id_for_details

SQL Statement:

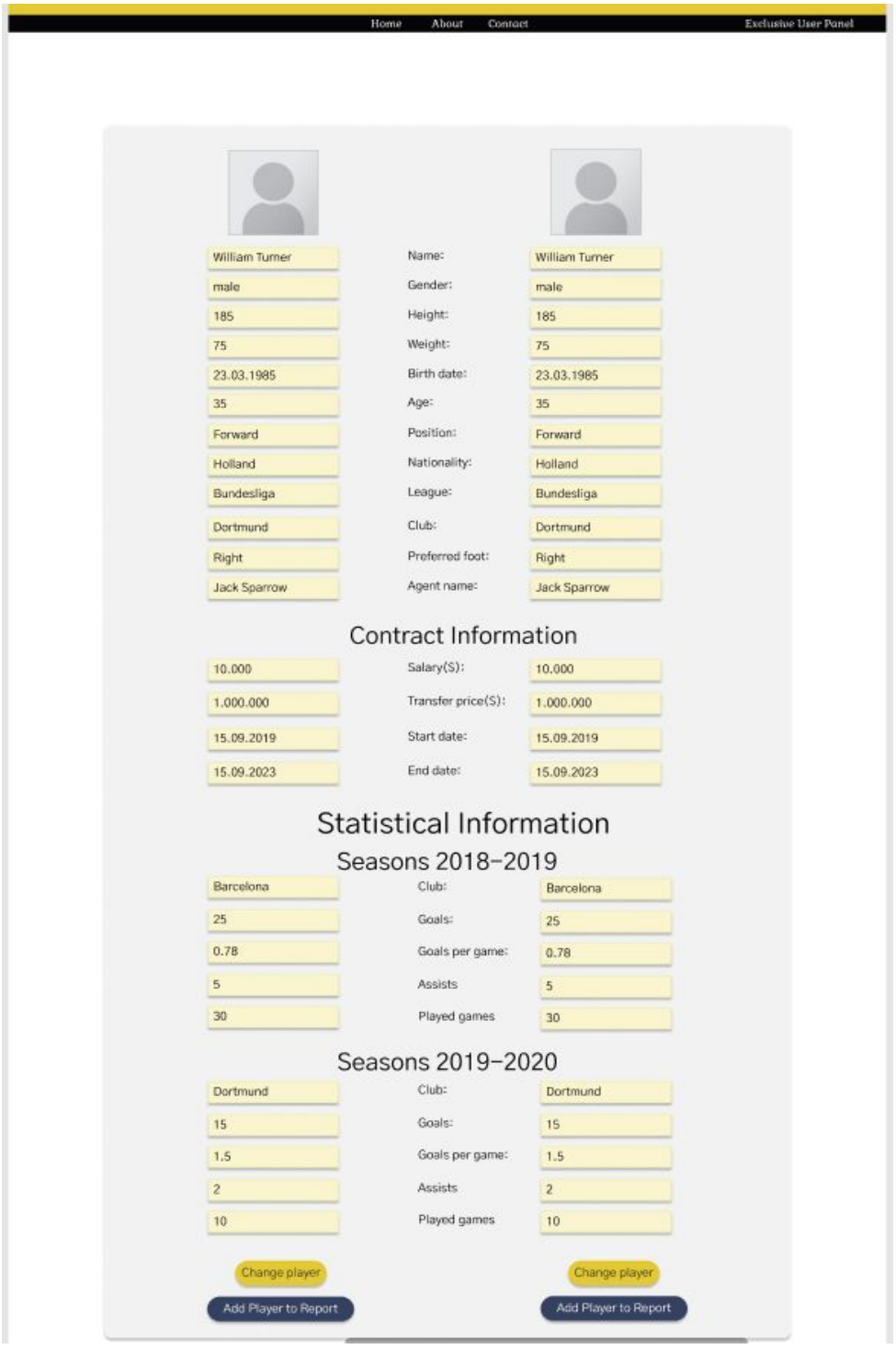
<filtering will be done in coding part by the website>

CREATE VIEW details AS

SELECT *

FROM footballer;

Comparison Page



If the comparison page is displayed by a scout, he can add the player instantly to his report. The button will not appear if the page is displayed by a club.

SQL Statement:

input: @report_id, @player_id, @scout_id, @club_id, @player_id1, @player_id2

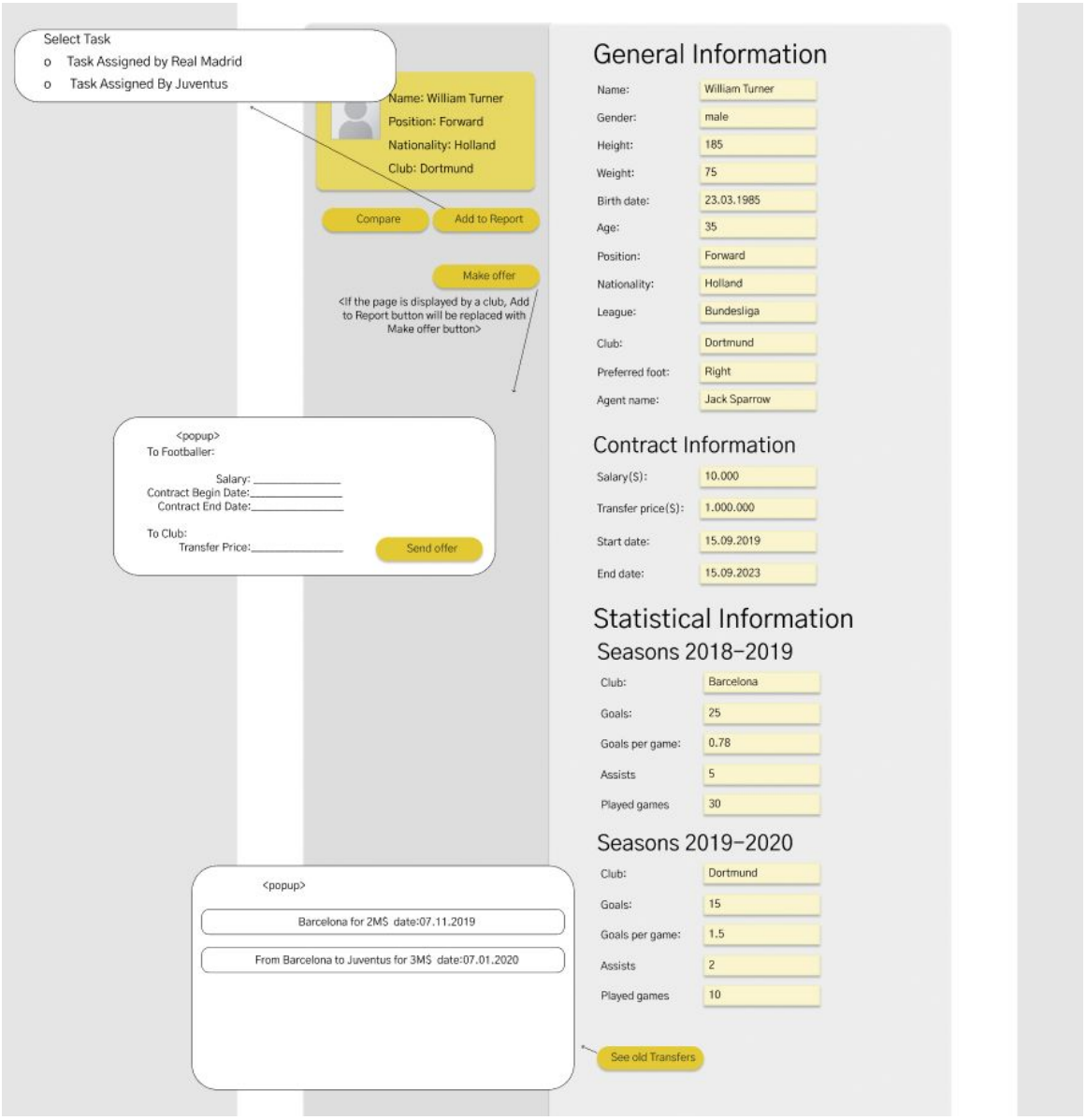
To insert a player into the report:

```
INSERT INTO final_report
VALUES(@report_id, @player_id, @club_id, @scout_id);
```

Display players:

```
CREATE VIEW comparison AS
(SELECT *  FROM (footballer NATURAL JOIN injury NATURAL JOIN statistics  NATURAL JOIN contract) WHERE player_id = @player_id1)
UNION
(SELECT * FROM (footballer NATURAL JOIN injury NATURAL JOIN statistics  NATURAL JOIN contract) WHERE player_id = @player_id2);
```

Footballer Details Page



On this page, all the info of footballer can be seen also the previous transfers. If it is displayed by a club, A “make offer button” will appear. When clicked, a popup is opened to enter salary, etc. and send an offer.

SQL Statement:

input: @report_id, @task_id, @offer_id, @scout_id

To make a contract offer to a footballer from the report:

```
WITH my_club(id) AS (SELECT club_id FROM final_report WHERE report_id = @report_id)
WITH my_agent(id) AS (SELECT agent_id
                      FROM   ( (SELECT player_id, agent_id  FROM footballer)
                                  NATURAL JOIN
                               (SELECT player_id FROM final_report WHERE report_id =
                              @report_id)))
INSERT INTO contract_offer
VALUES (@offer_id, my_club.id, my_agent.id);
```

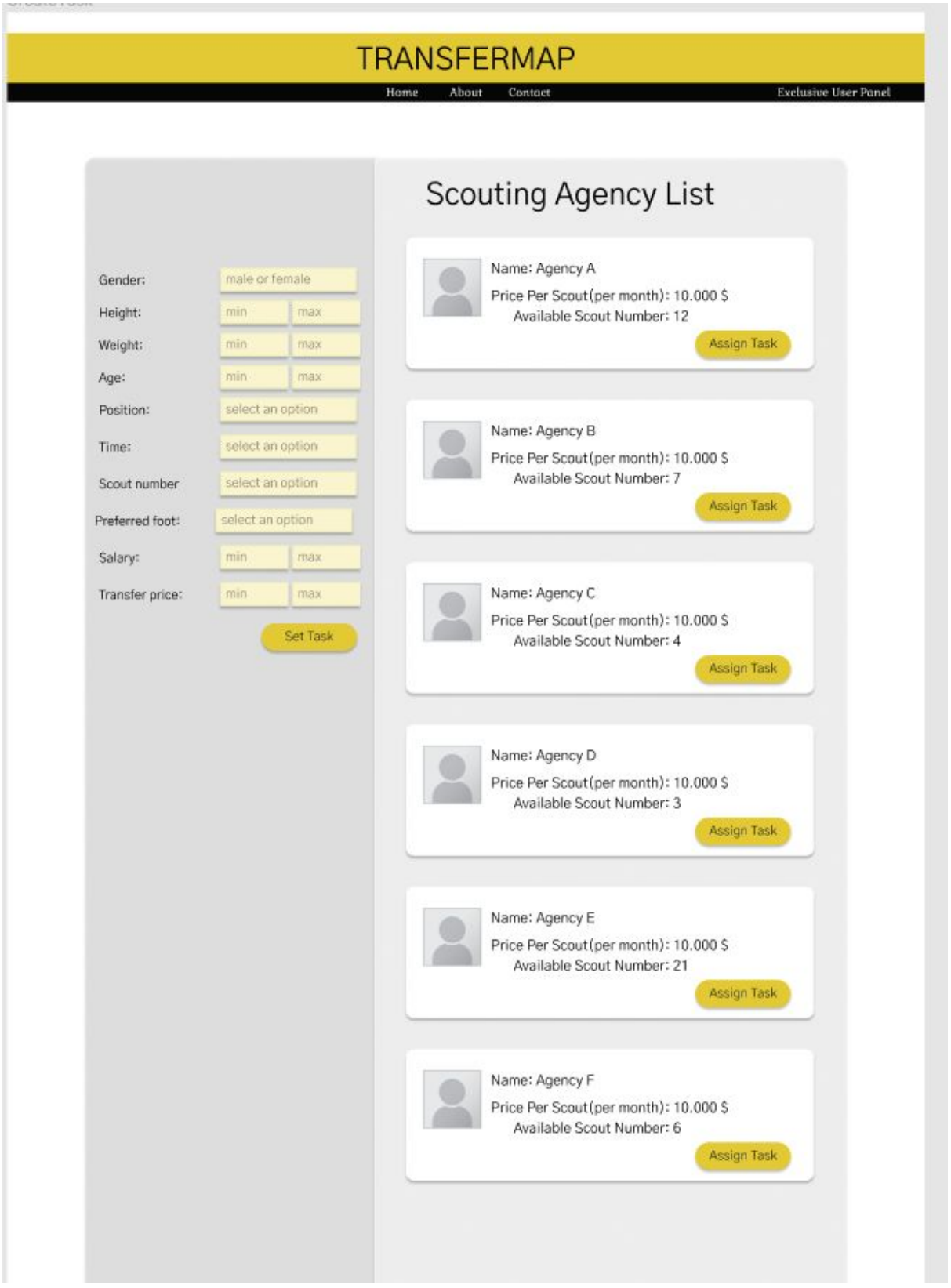
To display footballer details:

```
CREATE VIEW footballer_details AS
SELECT *
FROM footballer NATURAL JOIN injury NATURAL JOIN statistics  NATURAL JOIN contract
WHERE player_id = @player_id;
```

To add footballer to the report:

```
WITH my_player(id) AS (SELECT player_id
                       FROM footballer NATURAL JOIN final_report
                       WHERE report_id = @report_id)
INSERT INTO final_report
VALUES(@report_id, my_player.id, @task_id, @scout_id);
```

Create Task Page for Clubs



Clubs can enter task info and assign it to any available agency.

input: @task_id, @transfer_price, @salary, @foot, @age, @weight, @height, @position, @time, @scout_num, @assigned_date, @club_id

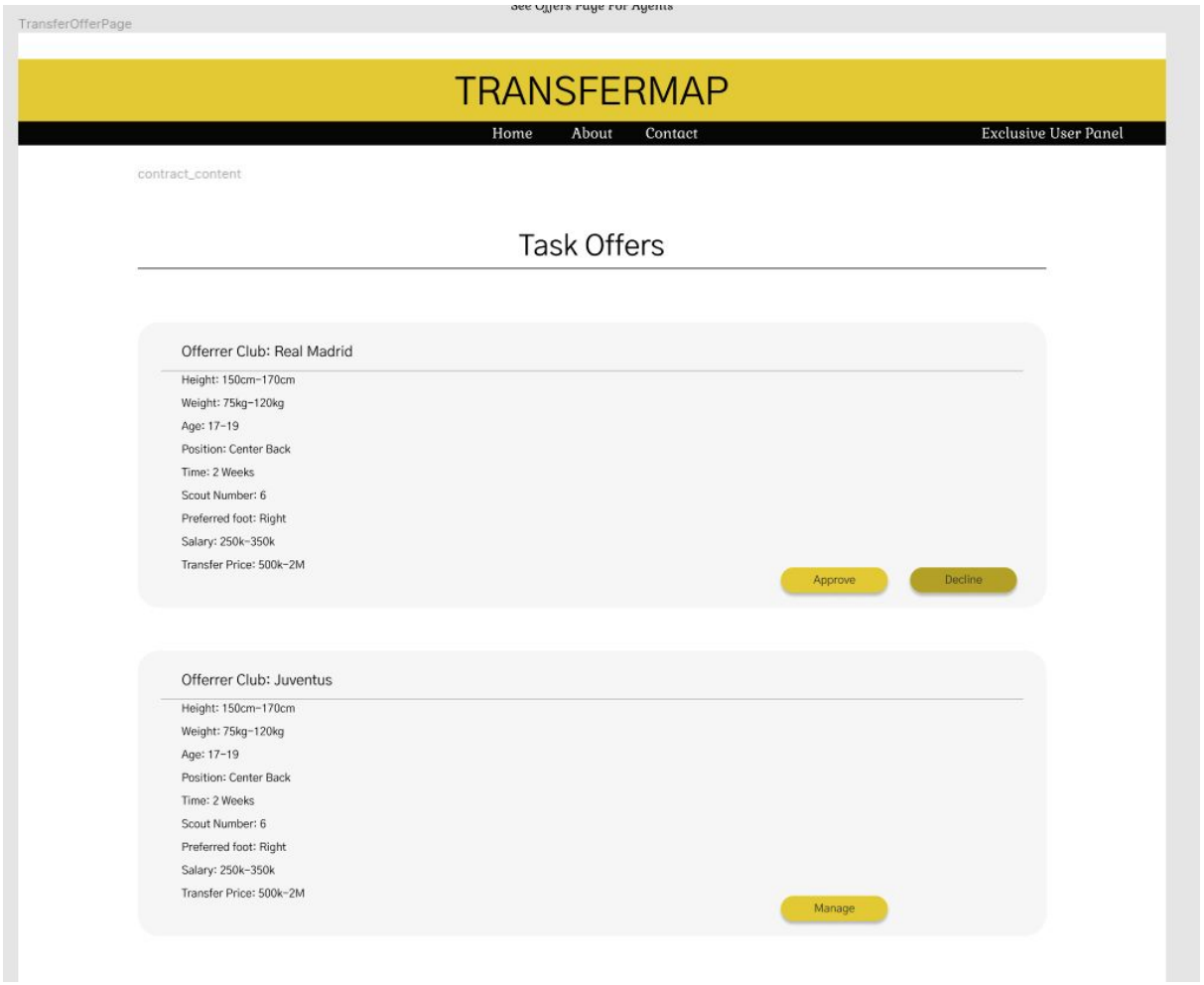
SQL Statement:

```
CREATE VIEW club_tasks AS
SELECT *
FROM task
WHERE club_id = @club_id;
```

To add a task:

```
INSERT INTO club_tasks
VALUES(@task_id, @transfer_price, @salary, @foot, @age, @weight, @height, @position, @time, @scout_num, @assigned_date, @club_id);
```

Tasks Page for Agency



Agencies can see tasks that offered, and also active tasks. Agencies can approve or decline a task. If approved, they can add scouts to the task by clicking the manage task button.

input: @agency_id, @task_id, @scout_id

SQL Statement:

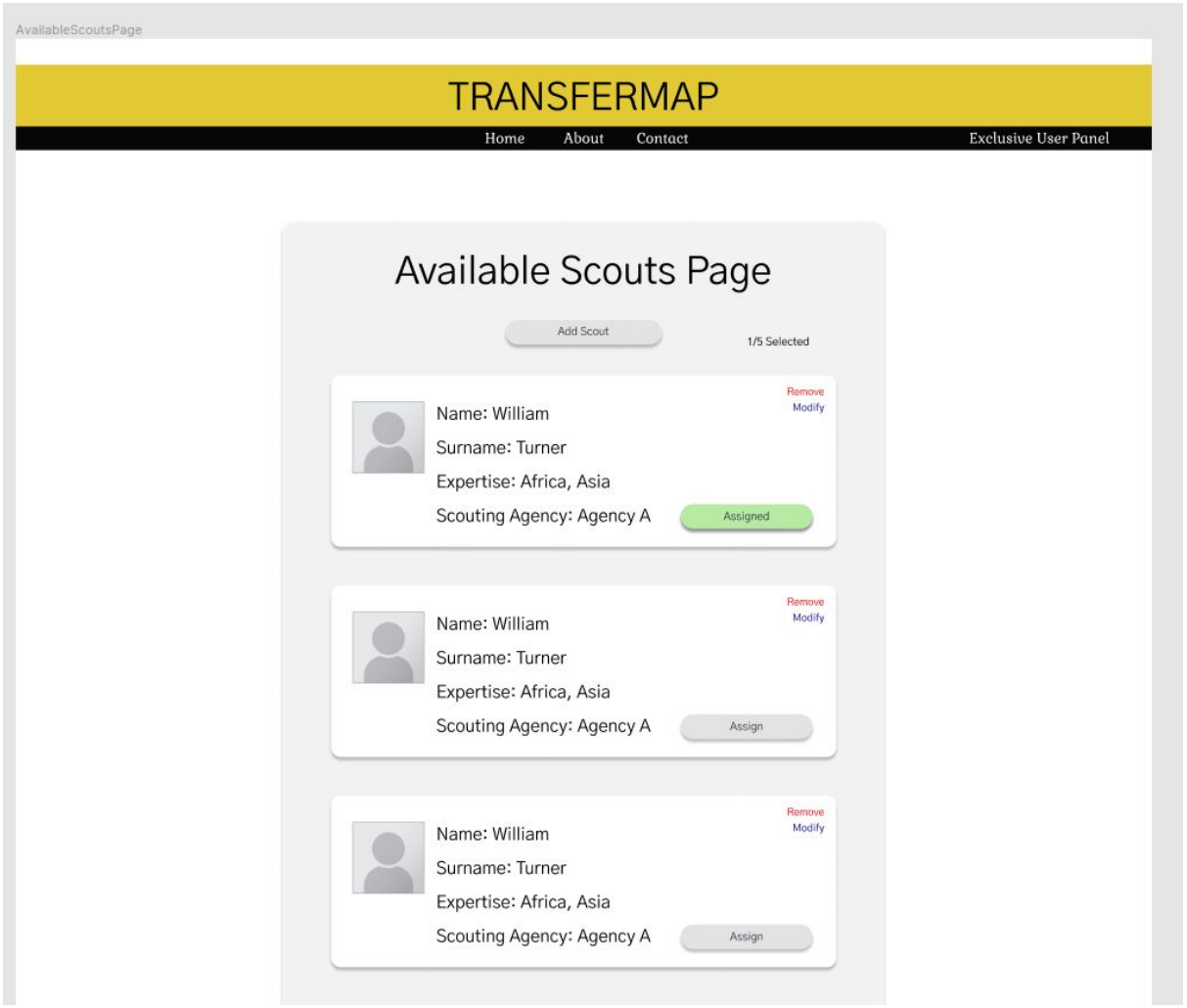
```
CREATE VIEW agency_tasks AS
SELECT *
FROM task
WHERE agency_id = @agency_id;
```

To approve a task:

```
UPDATE task
```

SET agency_id = @agency_id, status = ‘Approved’
WHERE task_id = @task_id;

Select Scout Page



Agencies can select scouts by clicking “Manage” for a task on the previous page. Also, they can add new scouts or remove/modify current ones. The page also says how many scouts are needed for the corresponding task.

input: @agency_id, @scout_id, @task_id, @name, @expertise

SQL Statement:

CREATE VIEW agency_scouts AS
SELECT *
FROM scout
WHERE agency_id = @agency_id;

To add a scout to the agency:

```
INSERT INTO agency_scouts
VALUES(scout_id);
UPDATE agency_scouts
SET name = @name
WHERE scout_id = @scout_id;
```

To update scout name:

```
UPDATE agency_scouts
SET name = @name
WHERE scout_id = @scout_id;
```

To add an expertise to the scout:

```
INSERT INTO scout_expertise
VALUES(scout_id, @expertise);
```

To delete an expertise to the scout:

```
DELETE FROM scout_expertise
WHERE scout_id = @scout_id;
```

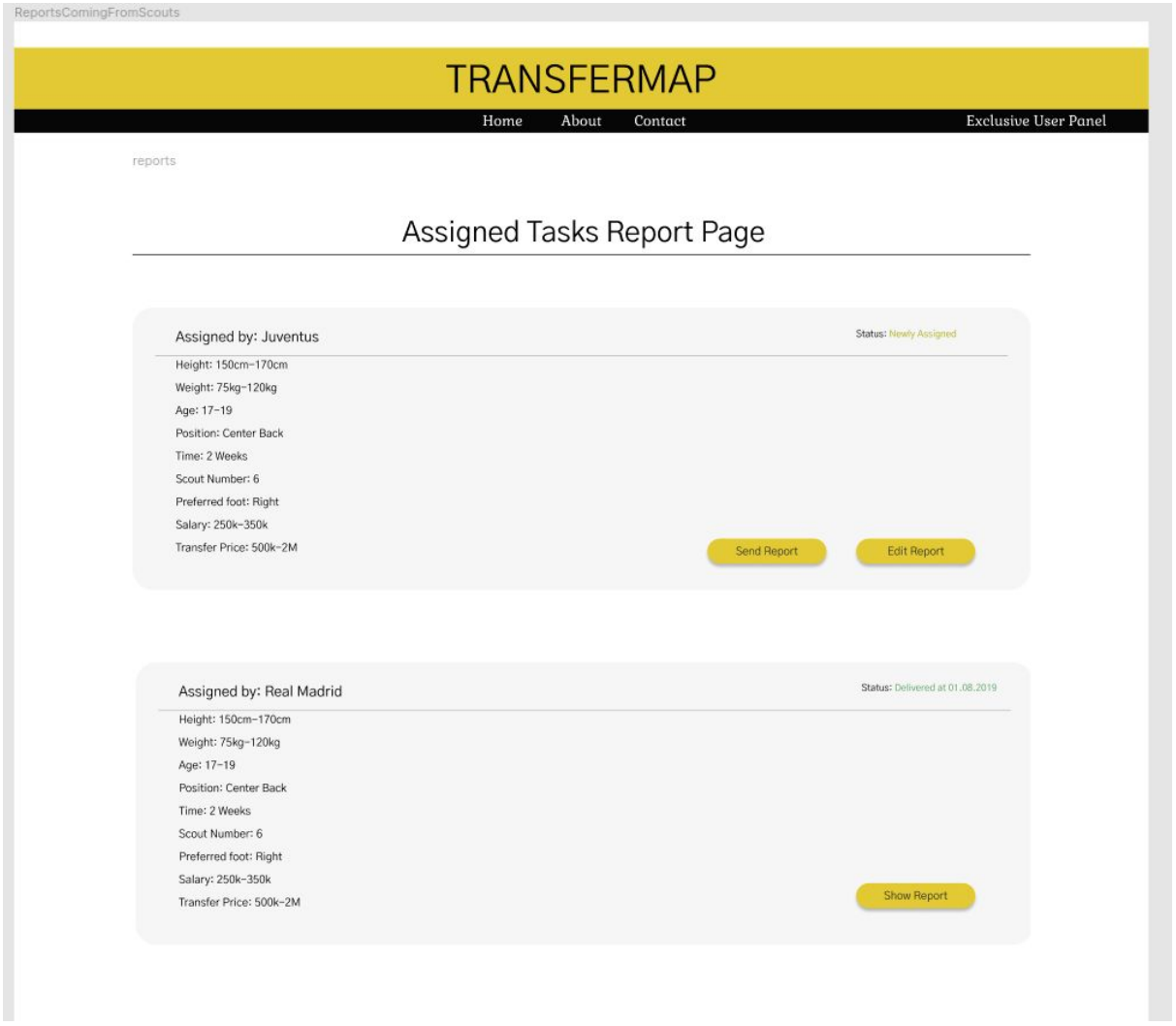
To assign a task to a scout:

```
INSERT INTO assignment
VALUES (@scout_id, @task_id);
```

To withdraw a task from a scout:

```
DELETE FROM assignment
WHERE scout_id = @scout_id AND task_id = @task_id;
```

Assigned Tasks Page for Scouts



When a scout is assigned, a new task is added to this page. Also, the scout is notified when a new task is added. Scouts can click edit reports and edit reports for footballers. Footballers must be added from the search page. When they are done, they click the send report button. They can also see previous tasks. Status info is displayed in the top right corner. Status info says if it is newly assigned or finished.

input: @s_id, @report_id, @task_id, @new_status

SQL Statement:

```
CREATE VIEW scout_tasks AS
SELECT *
FROM task NATURAL JOIN assignment
WHERE scout_id = @s_id;
```

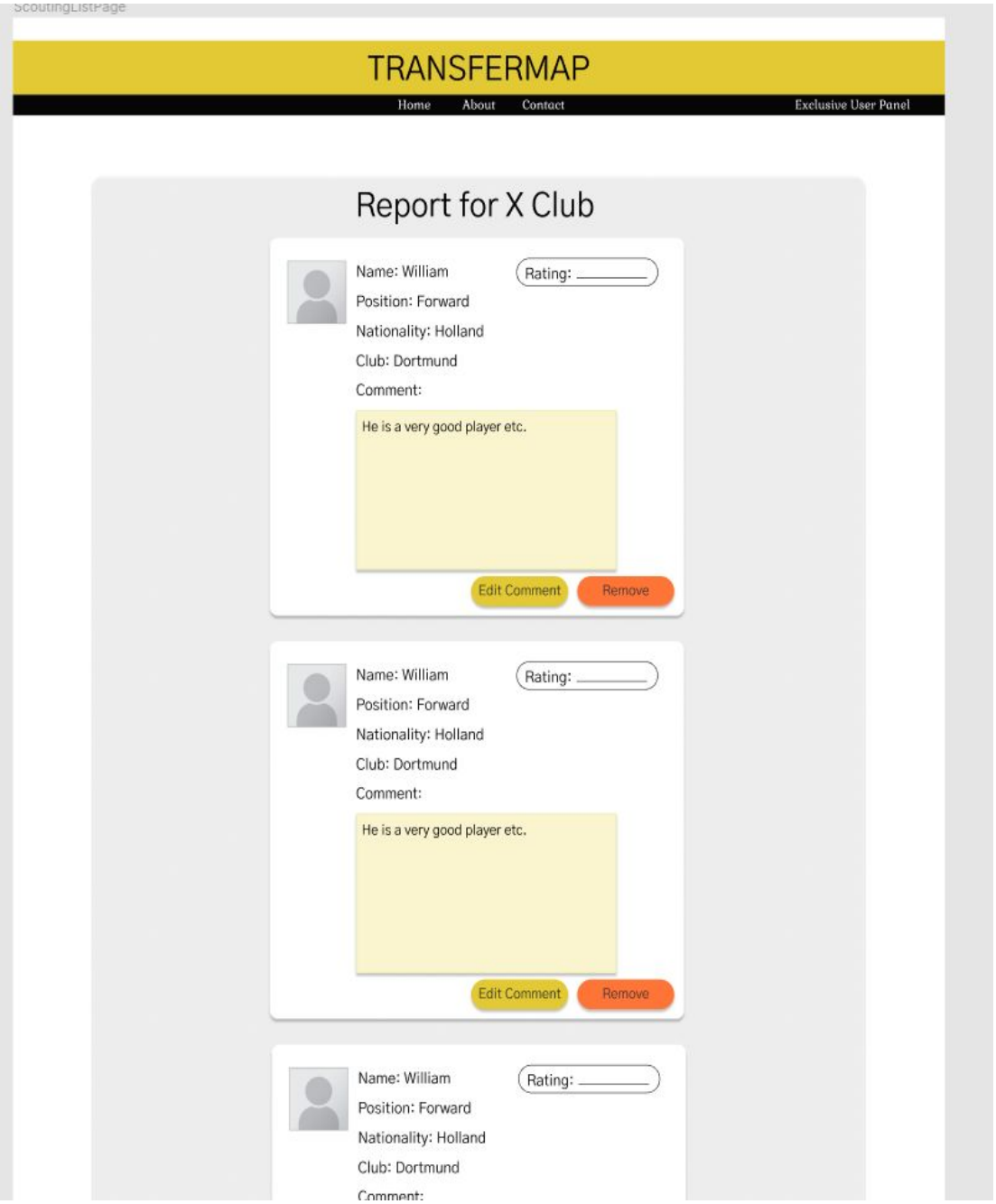
To update task status:

UPDATE task

SET status = ‘finished’

WHERE task_id = @task_id;

Scouts Report Page



Scouts can enter comments and a rating on selected footballers. Also, remove them from the report.

input: @report_id, @club_id, @player_id, @scout_id, @rating, @comment

SQL Statement:

```
CREATE VIEW scout_reports AS
SELECT *
FROM final_report
WHERE scout_id = @scout_id;
```

To add a footballer report:

```
INSERT INTO scout_reports
VALUES(@report_id, @player_id, @club_id, @scout_id, @rating, @comment);
```

To delete a footballer report:

```
DELETE FROM scout_reports
WHERE report_id = @report_id;
```

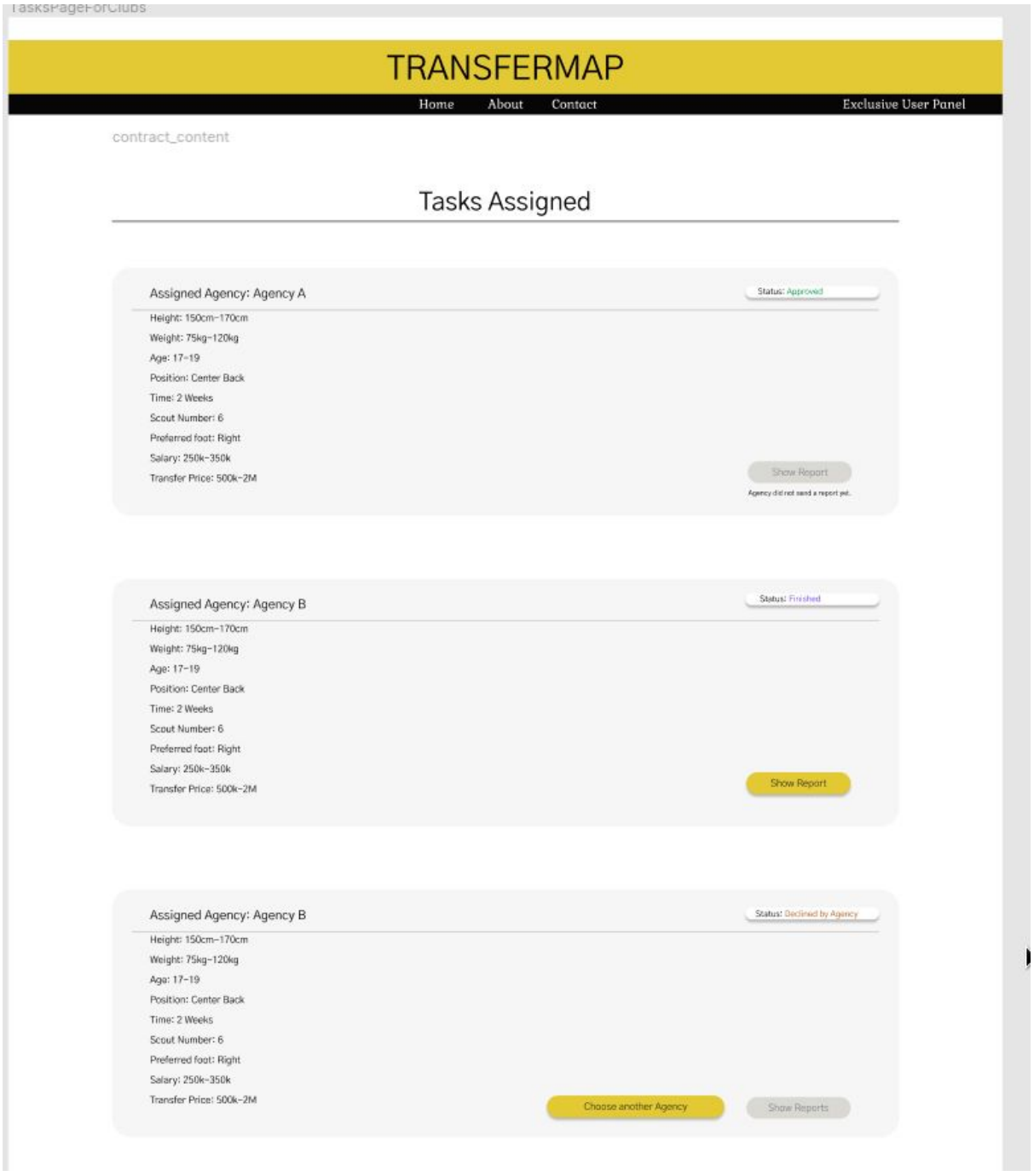
To update rating:

```
UPDATE scout_reports
SET rating = @rating
WHERE report_id = @report_id;
```

To update comment:

```
UPDATE scout_reports
SET comment = @comment
WHERE report_id = @report_id;
```

Task Page for Clubs



Clubs can see the tasks they create from this page. Also, they can see previous tasks. If a task is rejected by the agency, there is a button to assign the same task immediately to another agency. This will open the agency selection page again. If reports are sent by the scouts, Show report button turns yellow and clubs can see the reports.

input: @club_id

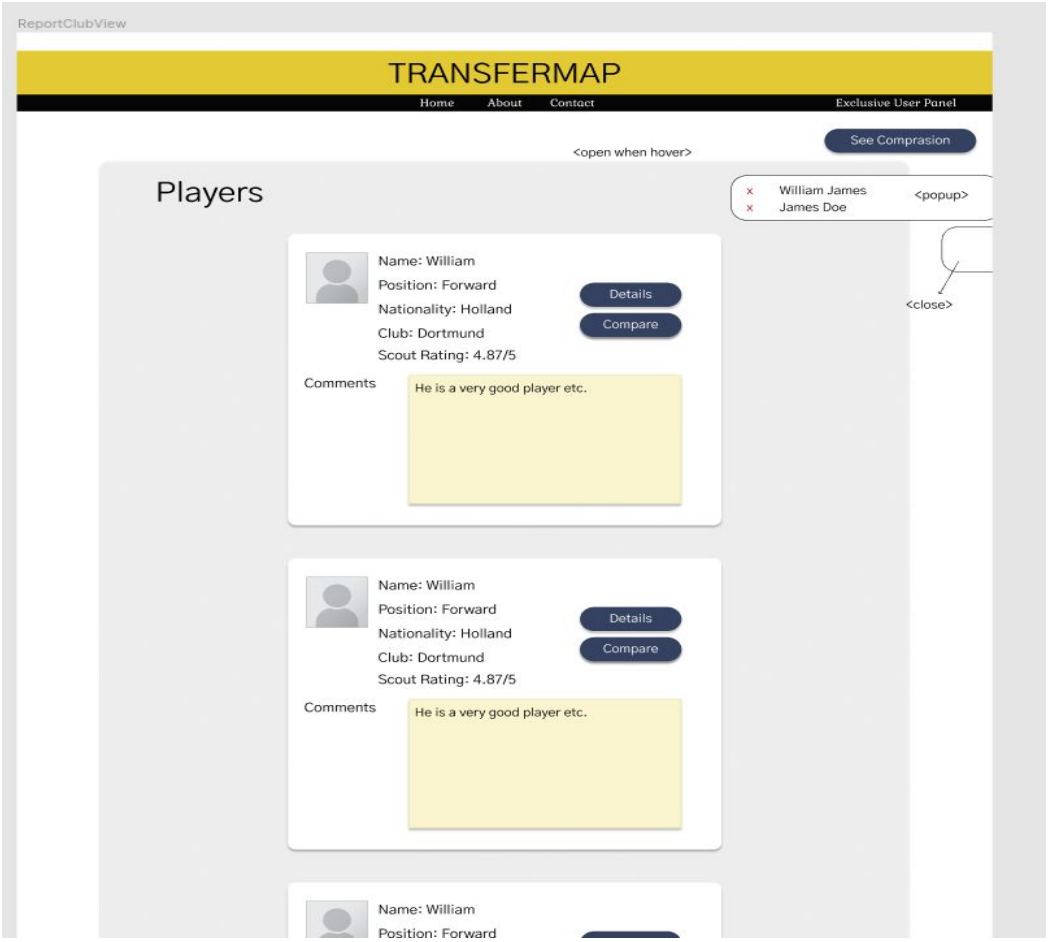
SQL Statement:

```
CREATE VIEW club_tasks AS
SELECT *
FROM task
WHERE club_id = @club_id;
```

To get task details:

```
CREATE VIEW task_reports AS
SELECT *
FROM final_report
WHERE task_id = @task_id;
```

Clubs Report Details Page



All the footballers that scouts sent are shown together here. A club can see details of the player by clicking details, they can see scout comments, also they can compare footballers. To make an offer, they click on details, then make an offer button on that page.

input: @report_id, @offer_id

SQL Statements:

```
CREATE VIEW report AS
SELECT *
FROM final_report NATURAL JOIN footballer
WHERE report_id = @report_id;
```

4. Website

After the feedback of the proposal, we have prepared a website using github.io to publish our reports.

- URL: https://aybukeertekin.github.io/Bilkent_CS353_Database_Project/