Department of Computer Engineering

Bilkent University

CS 353 Term Project

Group 5 - Social Betting Platform

Final Report

İdil YILMAZ - 21703556
Doğukan Ertunga KURNAZ - 21702331
Turgut Alp EDİS - 21702587
Utku GÖKÇEN - 21703746

Instructor: Prof. Dr. Özgür Ulusoy
Teaching Assistant(s): Mustafa Can Çavdar

May 17, 2022

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Database System course CS353.

Table of contents

Table of contents 2

Project Description 4

2. Final E/R Model 5

3. Final List of Tables

3.1 User 6

3.2 Normal User 6

3.3 Normal User Friend 6

3.4 Normal User Follows 6

3.5 Editor 6

3.6 Editor Request 7

3.7 Slip Creator 7

3.8 Admin 7

3.9 Bet 7

3.10 Bet Slip 7

3.11 Comments 8

3.12 Item Coupon 8

3.13 Matches 8

3.14 Results 8

3.15 Voleyball Results 8

3.16 Basketball Results 8

3.17 Football Results 9

3.18 LOL Results 9

3.19 Esports Team 9

3.20 Team 9

3.21 Sports 9

3.22 Contest 10

3.23 Competitors 10

3.24 Match Comment 10

3.25 Bet Slip Comment 10

3.26 Competitor Contest 10

3.27 Banned Users 10

3.28 Banned Editors 11

3.29 Plays 11

3.30 Like Comment 11

3.31 Share Bet Slip 11

3.32 Edit Bet 11

3.33 Buys 12

3.34 Has 12

3.35 Creates 12

Implementation Details 12

Advanced Database Features 14

5.1 Reports 14

5.1.1 Find Ended Bet Slips 14

5.1.2 Check MBN Condition 14

5.1.3 Filter Bets 14

5.1.4 Find Editor Win Count 15

6. User's Manual 16

6.1 Register Page 16

6.2 Login Page 16

6.3 Home Page 17

6.4 Social Page 19

6.5 Raffle Page 19

Project Web Page 20

1. Project Description

This project is a web-based application that proposes to work as a betting site which also has social features. It works on a database. Its design supports certain famous sports in the world, such as football, basketball, tennis as well as e-sports, such as League of Legends (LOL).

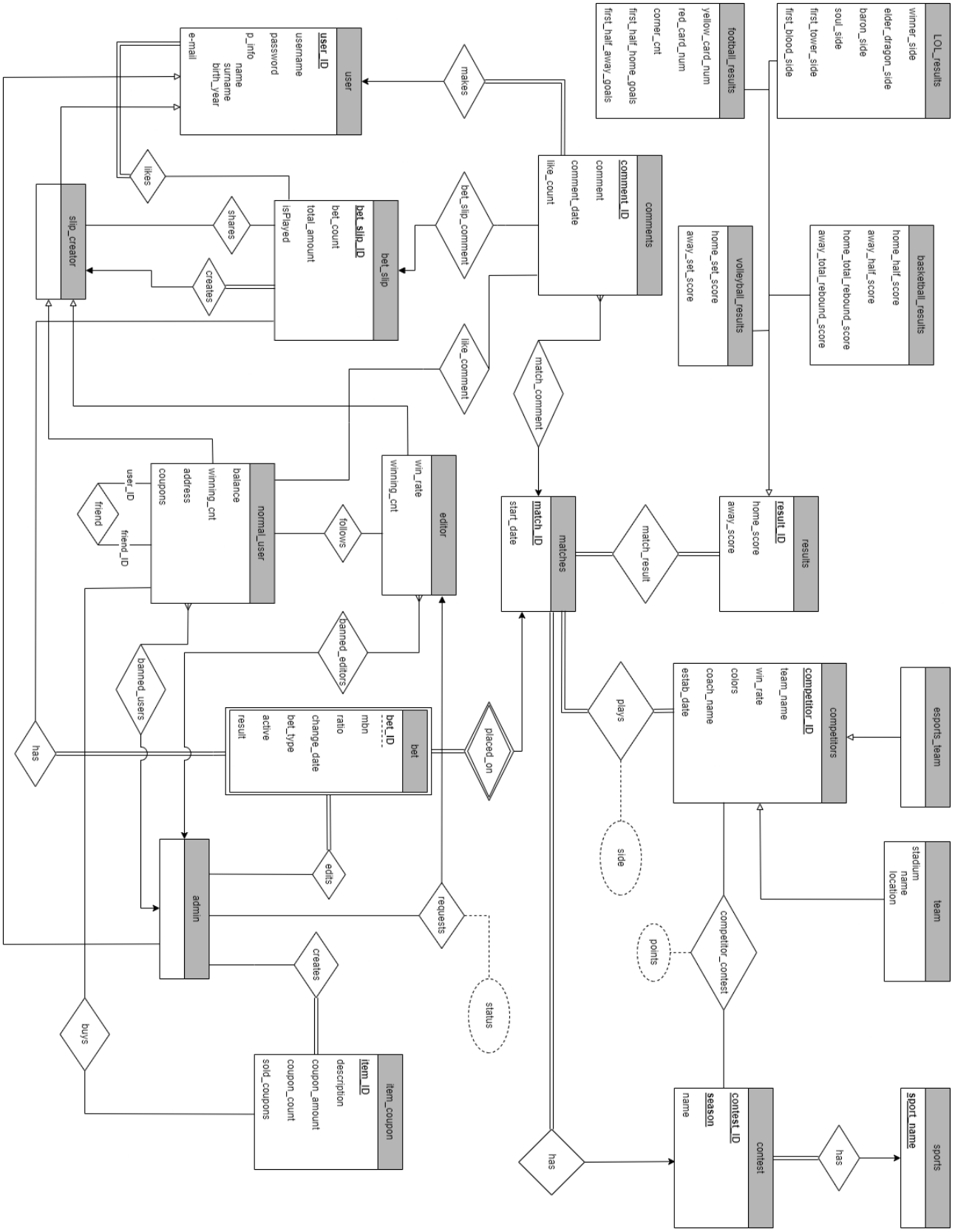
In the application, there are bets, matches and users. All matches have different types of bets which have their own ratio and minimum betting number (MBN). Users are separated into three types: normal users, editors and the admin. All users can see the match schedule of all kinds of sports in their timeline. Before the match time, the admin can edit the ratio of the bets belonging to the match while normal users and editors can see the background information about the teams such as previous matches and statistical scores and they can create bet slips according to this information.

In the social betting platform, normal users and editors cannot be able to change the bet but only admin can. In addition, normal users can make bets in bet slips and share these slips with other users. Also, they can comment on matches, comment and like the contents of other users. Besides, users can add other users as friends, see their activities on timeline and follow editors. At the same time, users can buy the tickets presented to them for a certain fee and use them as coupons.

From the editor's point of view, they can make bet slips, comment on matches and followers of the editors can see these activities on their timeline.

From the admin's points of view, the admin can edit the ratio of the bets and remove the bet. Also, the admin can ban the users according to their inappropriate behavior.

2. Final E/R Model



3. Final List of Tables

3.1 User

Relational Model: user(user_ID, username, password, name, surname, birth_year, e-mail)

PRIMARY_KEY(user_ID)

3.2 Normal User

Relational Model: normal_user(n_user_ID, balance, winning_cnt, address, coupons)

PRIMARY KEY(n_user_ID)
FOREIGN KEY (n_user_ID) REFERENCES slip_creator(creator_ID)

3.3 Normal User Friend

Relational Model: normal_user_friend(user_ID, friend_ID)

PRIMARY KEY (user_ID, friend_ID),
FOREIGN KEY (user_ID) REFERENCES normal_user(n_user_ID)
FOREIGN KEY (friend_ID) REFERENCES normal_user(n_user_ID)

3.4 Normal User Follows

Relational Model: normal_user_follows(editor_ID, user_ID)

PRIMARY KEY (editor_ID, user_ID),
FOREIGN KEY (editor_ID) REFERENCES editor(editor_ID)
FOREIGN KEY (user_ID) REFERENCES normal_user(n_user_ID)

3.5 Editor

Relational Model: editor(editor_ID, win_rate, winning_cnt)

PRIMARY KEY (editor_ID),
FOREIGN KEY (editor_ID) REFERENCES slip_creator(creator_ID)

3.6 Editor Request

Relational Model: editor_request(editor_ID, admin_ID, status)

PRIMARY KEY (editor_ID),
FOREIGN KEY (admin_ID) REFERENCES admin(admin_ID)
FOREIGN KEY (editor_ID) REFERENCES editor(editor_ID)

3.7 Slip Creator

Relational Model: slip_creator(creator_ID)

PRIMARY KEY (creator_ID),
FOREIGN KEY (creator_ID) REFERENCES user(user_ID)

3.8 Admin

Relational Model: admin(admin_ID)

PRIMARY KEY (admin_ID),
FOREIGN KEY (admin_ID) REFERENCES user(user_ID)

3.9 Bet

Relational Model: bet(bet_ID, match_ID, mbn, ratio, change_date, bet_type, active, result)

PRIMARY KEY (bet_ID, match_ID),
FOREIGN KEY (match_ID) REFERENCES matches(match_ID)

3.10 Bet Slip

Relational Model: bet_slip(bet_slip_ID, creator_ID, bet_count, total_amount, isPlayed)

PRIMARY KEY (bet_slip_ID)

3.11 Comments

Relational Model: comments(comment_ID, user_ID, comment, comment_date, like_count)
PRIMARY KEY (comment_ID),
FOREIGN KEY(user_ID) REFERENCES user(user_ID)

3.12 Item Coupon

Relational Model: item_coupon(item_ID, description, coupon_amount, coupon_count, sold_coupons)
PRIMARY KEY (item_ID)

3.13 Matches

Relational Model: matches(match_ID, start_date, sport_name)
PRIMARY KEY(match_ID),
FOREIGN KEY(sport_name) REFERENCES contest(sport_name)
FOREIGN KEY(sport_name) REFERENCES sports(sport_name)

3.14 Results

Relational Model: results(result_ID, home_score, away_score)
PRIMARY KEY(result_ID)

3.15 Voleyball Results

Relational Model: volleyball_results(v_result_ID, home_set_score, away_set_score**)**
PRIMARY KEY(v_result_ID),
FOREIGN KEY (v_result_ID) REFERENCES results(result_ID)

3.16 Basketball Results

Relational Model: basketball_results(b_result_ID, home_half_score, away_half_score, home_total_rebound_score, away_total_rebound_score**)**

PRIMARY KEY(b_result_ID),
FOREIGN KEY (b_result_ID) REFERENCES results(result_ID)

3.17 Football Results

Relational Model: football_results(f_result_ID, yellow_card_num, red_card_num, corner_cnt, first_half_home_goals, first_half_away_goals**)**
PRIMARY KEY(f_result_ID),
FOREIGN KEY (f_result_ID) REFERENCES results(result_ID)

3.18 LOL Results

Relational Model: LOL_results(l_result_ID, winner_side, elder_dragon_side, baron_side, soul_side, first_tower_side, first_blood_side**)**
PRIMARY KEY(l_result_ID),
FOREIGN KEY (l_result_ID) REFERENCES results(result_ID)

3.19 Esports Team

Relational Model: esports_team(competitor_ID**)**
PRIMARY KEY(competitor_ID),
FOREIGN KEY (competitor_ID) REFERENCES competitors(competitor_ID)

3.20 Team

Relational Model: team(competitor_ID, name, location)
PRIMARY KEY(competitor_ID),
FOREIGN KEY (competitor_ID) REFERENCES competitors(competitor_ID)

3.21 Sports

Relational Model: sports(sport_name)
PRIMARY KEY(sport_name)

3.22 Contest

Relational Model: contest(contest_ID, season,sport_name, name)
PRIMARY KEY (contest_ID, season),
FOREIGN KEY (sport_name) REFERENCES sports(sport_name)

3.23 Competitors

Relational Model: competitors(competitor_ID, team_name, win_rate, colors, estab_date, coach_name)
PRIMARY KEY(competitor_ID)

3.24 Match Comment

Relational Model: match_comment(comment_ID,match_ID)
PRIMARY KEY(comment_ID),
FOREIGN KEY(match_ID) REFERENCES matches(match_ID)

3.25 Bet Slip Comment

Relational Model:bet_slip_comment(bet_slip_ID, comment_ID)
PRIMARY KEY(comment_ID, bet_slip_ID),
FOREIGN KEY(comment_ID) REFERENCES comments(comment_ID)
FOREIGN KEY(bet_slip_ID) REFERENCES bet_slip(bet_slip_ID)

3.26 Competitor Contest

Relational Model: competitor_contest(competitor_ID, contest_ID, season, points)
PRIMARY KEY(competitor_ID, contest_ID, season),
FOREIGN KEY (competitor_ID) REFERENCES competitors(competitor_ID)
FOREIGN KEY(contest_ID, season) REFERENCES contest(contest_ID, season)

3.27 Banned Users

Relational Model: banned_users(n_user_ID, admin_ID)
PRIMARY KEY(n_user_ID),

FOREIGN KEY(admin_ID) REFERENCES admin(admin_ID)

3.28 Banned Editors

Relational Model: banned_editors(editor_ID,admin_ID)
PRIMARY KEY(editor_ID),
FOREIGN KEY(admin_ID) REFERENCES admin(admin_ID)

3.29 Plays

Relational Model: plays(match_ID, competitor_ID, side)
PRIMARY KEY(match_ID, competitor_ID),
FOREIGN KEY(match_ID) REFERENCES matches(match_ID)
FOREIGN KEY (competitor_ID) REFERENCES competitors(competitor_ID)

3.30 Like Comment

Relational Model: like_comment(comment_ID, n_user_ID)
PRIMARY KEY(comment_ID, n_user_ID),
FOREIGN KEY(n_user_ID) REFERENCES normal_user(n_user_ID)
FOREIGN KEY (comment_ID) REFERENCES comments(comment_ID)

3.31 Share Bet Slip

Relational Model: shared_slip(bet_slip_ID, sharer_ID)
PRIMARY KEY(bet_slip_ID, sharer_ID),
FOREIGN KEY(bet_slip_ID) REFERENCES bet_slip(bet_slip_ID)
FOREIGN KEY (sharer_ID) REFERENCES slip_creator(creator_ID)

3.32 Edit Bet

Relational Model: edits(admin_ID, bet_ID, match_ID)
PRIMARY KEY(admin_ID, bet_ID, match_ID),
FOREIGN KEY (admin_ID) REFERENCES admin(admin_ID)

FOREIGN KEY(bet_ID, match_ID) REFERENCES bet(bet_ID, match_ID)

3.33 Buys

Relational Model: buys(<u>item_ID, n_user_ID</u>)
PRIMARY KEY(item_ID, n_user_ID),
FOREIGN KEY(item_ID) REFERENCES item_coupon(item_ID)
FOREIGN KEY(n_user_ID) REFERENCES normal_user(n_user_ID)

3.34 Has

Relational Model: has(<u>bet_ID,match_ID</u>,bet_slip_ID)
PRIMARY KEY(bet_ID, match_ID),
FOREIGN KEY(bet_slip_ID) REFERENCES bet_slip(bet_slip_ID)

3.35 Creates

Relational Model: item_coupon(<u>item_ID</u>, admin_ID)
PRIMARY KEY(item_ID),
FOREIGN KEY(admin_ID) REFERENCES admin(admin_ID)

4. Implementation Details

We used MySQL for the database and Visual Studio Code to develop the application. In order to automate database creation and populate database tables, we wrote create statement queries for each table in the database and sent those queries to the server using python. On each modification in the database, we updated our database tables and its entries in our python file.

For the user interface, we used JavaScript and ReactJS framework. To communicate with the backend service through HTTP requests, we used Axios library which is a JavaScript library that provides automatic JSON data transformation.

We used the Flask web application framework in Python to implement system operations on the backend side. Connection with the front end is established via the endpoints provided by Flask. Cursors were used to fetch data from the database into the application and also used to execute insert, update and delete operations on the database.

In the backend, we wrote the HTTP requests using the python requests library at first. However, we encountered some synchronization problems when we ran the program. In order to solve this, we used the Axios HTTP client in the componentDidMount() function. In this way, we were able to send HTTP requests in an easy and synchronized way.

Turgut Alp Edis and Utku Gökçen were responsible for the implementation of the backend side. Turgut implemented the register, login, feed, home, and admin operations and wrote the queries of listing bets, filtering them according to MBN, sport, contest, and keyword. Utku implemented the editor, profile, and raffle operations on the backend side. He wrote the queries of buying raffle tickets, editor bet modifying, and profile edit. İdil Yılmaz implemented the login and registration pages for different types of users (user, editor, admin). She also helped with the implementation of the home page. Doğukan Ertunga Kurnaz was responsible for the implementation of social, profile, and raffle and home pages.

5. Advanced Database Features

5.1 Reports

5.1.1 Find Ended Bet Slips

```
cursor.execute("WITH user_bet_slips AS (SELECT bet_slip_ID FROM bet_slip WHERE creator_id = {0}), ended_slip AS"
               " (SELECT DISTINCT u.bet_slip_ID FROM user_bet_slips u WHERE NOT EXISTS (SELECT bet_ID FROM "
               " user_bet_slips NATURAL JOIN placed_on NATURAL JOIN bet WHERE bet_slip_ID = u.bet_slip_ID AND"
               " result = 'PENDING')), all_bet_data AS (SELECT * FROM ended_slip NATURAL JOIN placed_on"
               " NATURAL JOIN bet), match_data AS (SELECT * FROM all_bet_data NATURAL JOIN plays),"
               " all_competitors AS (SELECT team_name, competitor_ID FROM competitors) SELECT * FROM match_data \
                  NATURAL JOIN all_competitors".format(info["user_ID"]))
```

This query finds all the bet slips of the user which has ended since the creation of the user account.

5.1.2 Check MBN Condition

```
check_mbn_query = "WITH user_bet_slip AS (SELECT bet_slip_ID FROM bet_slip WHERE creator_ID = %s AND isPlayed = FALSE, " \
                "current_bets AS (SELECT * FROM user_bet_slip NATURAL JOIN bet), " \
                "current_cnt_bet AS (SELECT COUNT(bet_slip_ID) AS bet_cnt FROM current_bets), " \
                "max_mbn_cnt AS (SELECT MAX(mbn) AS max_mbn FROM current_bets) " \
                "SELECT CASE WHEN current_cnt_bet.bet_cnt < max_mbn_cnt.max_mbn THEN 'MBN_NOT_OK' " \
                "WHEN current_cnt_bet.bet_cnt >= max_mbn_cnt.max_mbn THEN 'MBN_OK' " \
                "END AS response FROM current_bet_cnt, max_mbn_cnt"
execute_check = cursor.execute(check_mbn_query, value)
```

This query checks the mbn condition. If the user bet count is less than mbn, the response is "MBN_NOT_OK", if it is larger, the response is "MBN_OK".

5.1.3 Filter Bets

```
filter_query = "WITH s_filter AS ( SELECT match_ID from matches WHERE sport_name = %s), " \
               "b_filter AS (SELECT match_ID FROM bet WHERE active = TRUE AND mbn <= %s), " \
               "c_filter AS (SELECT match_ID FROM matches NATURAL JOIN contest WHERE contest.name IN %s), " \
               "esport_filter AS (SELECT match_ID FROM plays NATURAL JOIN competitors NATURAL JOIN esports_team WHERE team_name LIKE %s), " \
               "team_filter AS (SELECT match_ID FROM plays NATURAL JOIN competitors NATURAL JOIN team WHERE team_name LIKE %s), " \
               "competitor_union AS (SELECT match_ID FROM esport_filter UNION TABLE team_filter), "\
               "final_filter AS (SELECT DISTINCT match_ID FROM s_filter INNER JOIN b_filter USING(match_ID) INNER JOIN c_filter USING(match_ID) INNER JOIN
               "INNER JOIN competitor_union USING(match_ID))"
```

This query filters the bets by category, mbn value and keywords using the LIKE operator.

5.1.4 Find Editor Win Count

```
if cursor.execute("WITH editor_slips AS (SELECT bet_slip_ID, creator_ID as editor_ID FROM bet_slip WHERE "
                "creator_ID = {0}), editor_bet_ID AS (SELECT * FROM placed_on NATURAL JOIN editor_slips)"
                " SELECT COUNT(bet_slip_ID) AS won_bet_slip_count FROM editor_slips WHERE NOT EXISTS (SELECT"
                " bet_ID, match_ID FROM editor_bet_ID NATURAL JOIN bet WHERE bet_slip_ID = bet_slip_ID "
                "AND (result = 'LOST' OR result = 'PENDING')) GROUP BY editor_ID"
                        .format(editor_id["editor_ID"])) > 0:
```

This query finds the number of all bets the editor has won since the account was created, using the editor ids group by.

5.2 Constraints

We have some constraints in the system which checks if the side value of a tuple in a match is “HOME” or “AWAY”. Also, we have another check constraint which checks if the result of a bet is one of “WON”, “LOST” or “PENDING”.

5.3 Stored Procedures

We have used some stored procedures in our system. One of them returns all bet slip id’s of the ended bet slips of the user. These id’s were later used to gather information of the user’s bet slips in order to display them in the user interface. Also, we have another stored procedure which keeps the id’s of the items bought.

6. User's Manual

6.1 Register Page

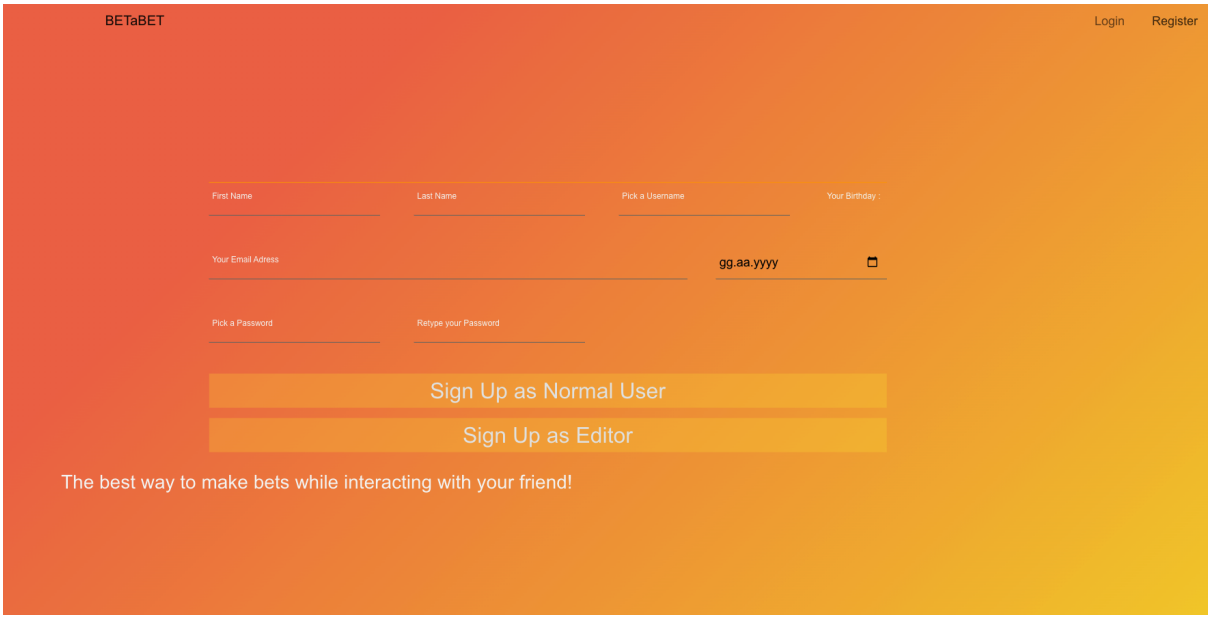


Figure 1. Register Page

When the users landed on our site, they can quickly register by clicking the register button in the navigation bar. On this page, our site asks for First Name, Last Name, Username, Birthday, Email Address, and Password from the user. After that, users can choose the registration type from the buttons below. If they choose to Sign up as Editor, their Editor application will be sent to the Admin to further checks.

6.2 Login Page

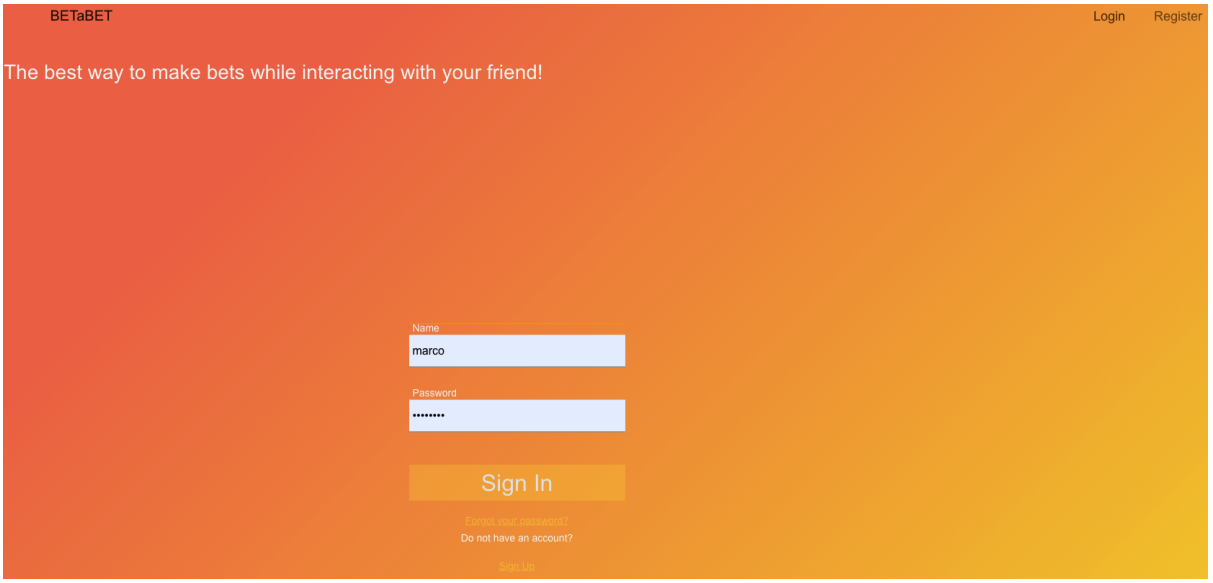


Figure 2. Login Page

We decided to use our login page as a landing page as well. Because the flow always starts from this page, this was the decision made by our team. To reach this page, users can click the Login button in the navigation bar as well. After that, users will be asked to fill in username and password fields with their credentials in order to log in successfully. If the credentials are incorrect a proper message will be shown to the user as well.

6.3 Home Page

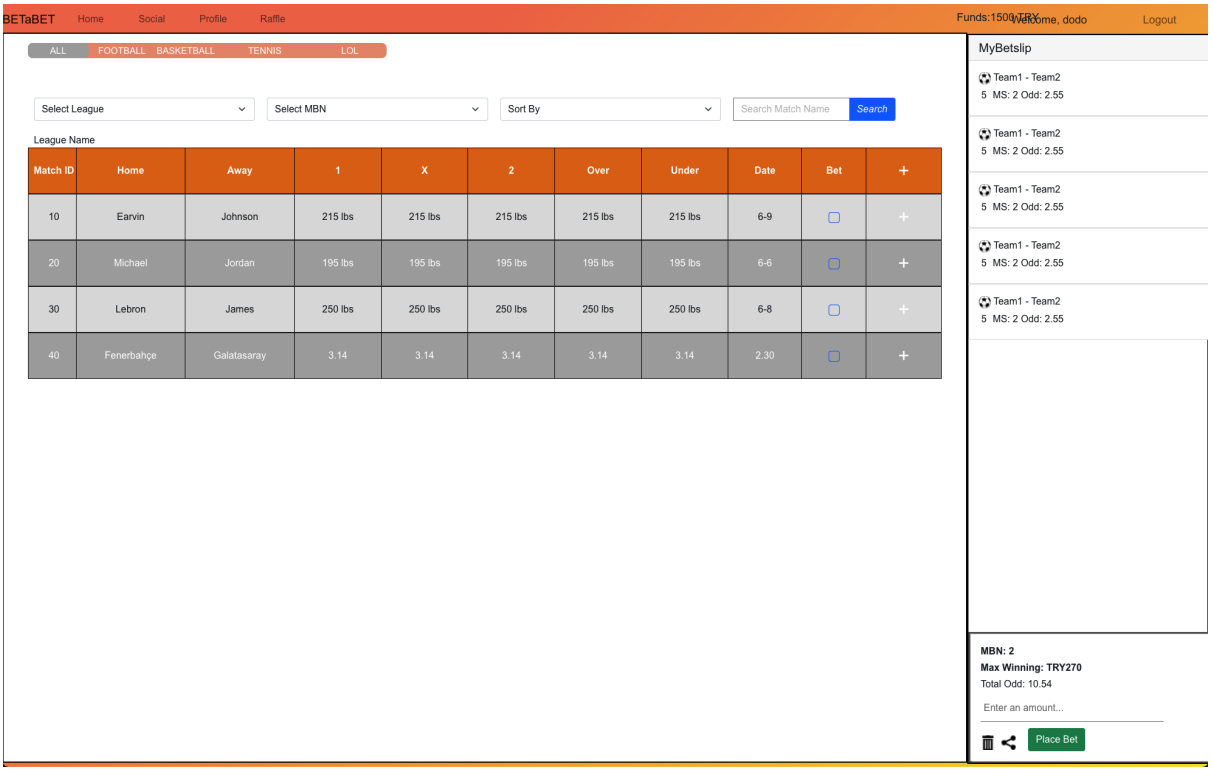


Figure 3. Home Page

In the Homepage, users can select their interested sports type from the slider. So that, they can filtrate all matches according to its type and get only the related results. In the Figure 3, users can see that they can navigate to different parts of the website using the navigation bar. Also, they can see their remaining funds from the top right of the navigation bar. To log out, they can simply click the Logout button in order to drop their session. Matches are shown in a table view, users can choose Minimum Bet Number, or League, and they can sort the matches by their name alphabetically, or in ascending, descending order by their match ids. Each row contains the bet rate for each related situation. Users can select the ratios from the nonexpanded row easily and fast. After they are done with selection, they can simply check the bet checkbox, after the check was made the bet will be added to the betslip in the right panel. If they need more betting options they can expand each row by clicking the “+” button on the right column.

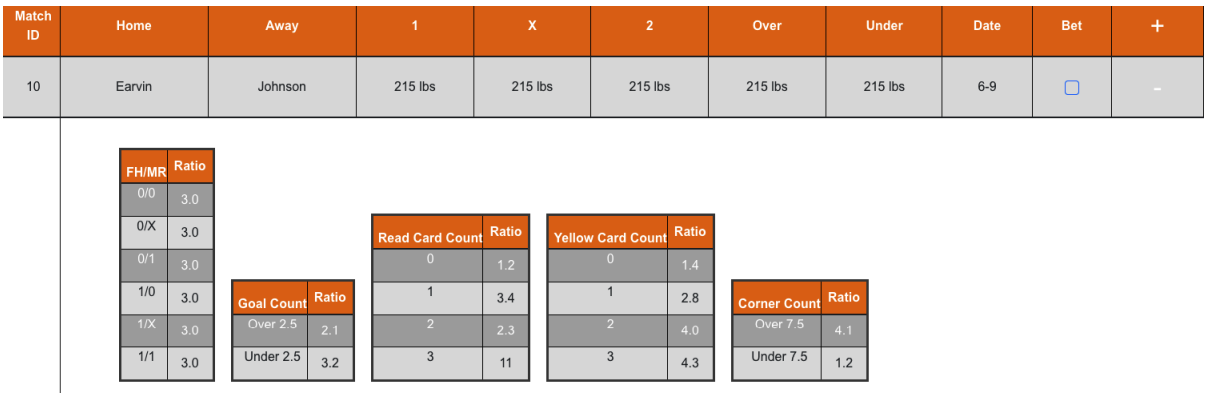


Figure 4. Expanded Row

Figure 4 shows that users can see more detailed betting types and their ratios in the expanded view of rows. They can choose ratios to bet on. If they satisfied the MBN constraint, they can place their bets with the desired amount. Also, in the bottom panel of the betslip users can see their max winning money from their betslip and their total odds. They have an option to delete their betslip by clicking the "trash bin" icon in the same place or they can share their prepared betslips with their friends on the social page of our website.

6.4 Social Page

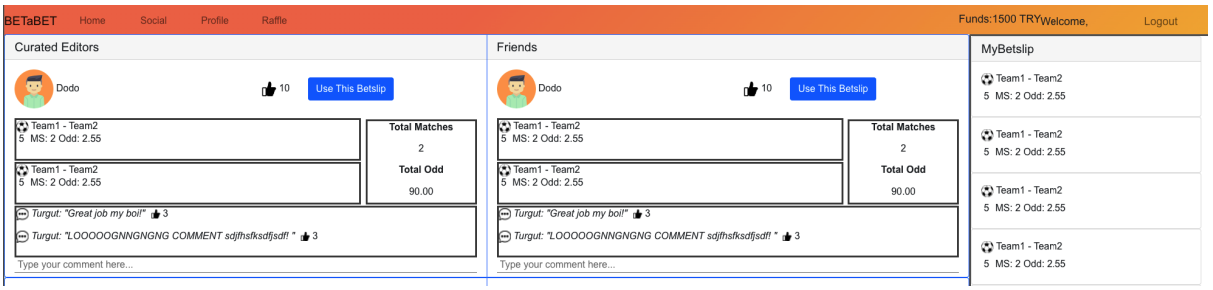


Figure 5. Social Page

In Figure 5, users can see the Social page of our website. Users can make comments on the shared betslips or like the betslip or even copy the shared betslip to their betslip panel to play with it. Also, in the shared betslip users can see the Total Match number in the betslip and Total Odd number as well as individual matches. They can also see the previous comments made by others to the shared betslips. This page contains betslip feed both from their friends and our curated editors.

6.5 Raffle Page

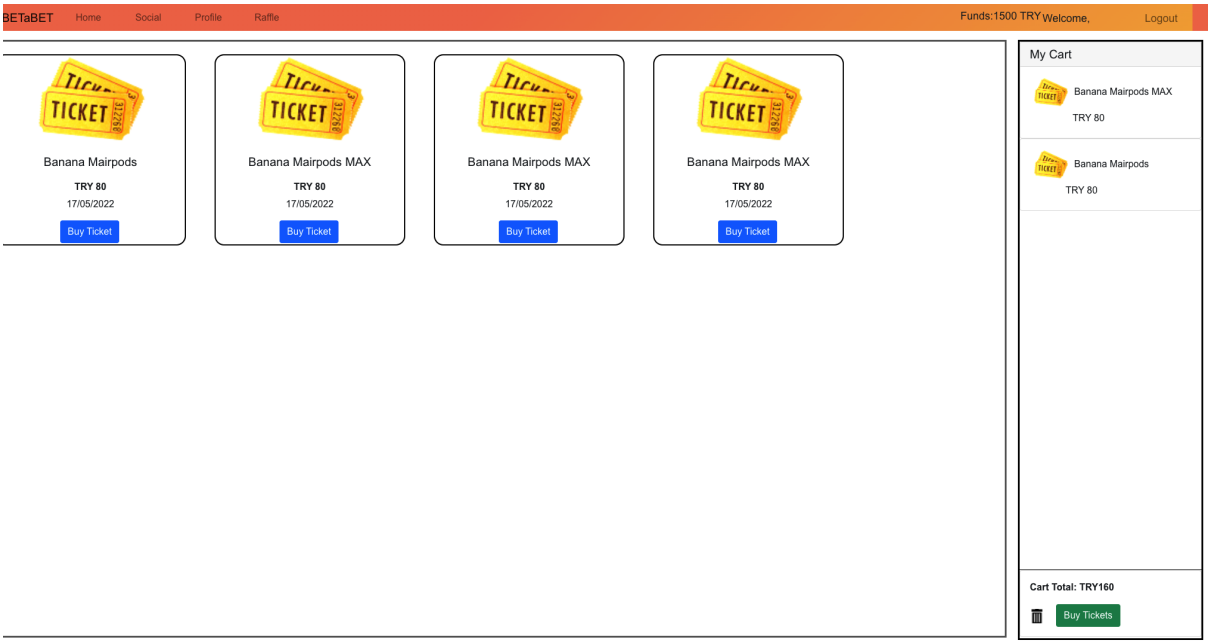


Figure 6. Raffle Page

In Figure 6, users can see the Raffle page of the betting website. They can buy raffle tickets from this page by using their deposited funds. Users can add multiple raffle tickets to their cart. When the user clicks the “Buy Tickets” button, the system will check the Cart total, if the user has enough funds, the operation will be successful.

Project Web Page

https://turgut-edis.github.io/SocialBettingProject/