CS3101 P2

180001693

Word count: 2097

Contents

1 Compilation, Execution & Usage Instructions

In the submission folder is the dump file (databasedump.sql) that can be used to replicate the database created. The /data folder also contains .csv files of the data in the required form to add to the database.

HTML, CSS and PHP were used to create the required GUI, and it can be found at `https://mrb21.host.cs.st-andrews.ac.uk/cs3101_p2/tennis/`. Copies of the files used to create this website are in the gui folder, and image credits go to Wikimedia Commons.

2 Overview

This submission achieves all the required functionality as specified.

2.1 Database design

The database was created according to the given schema, and the required views were created. Figure 1 illustrates the ER diagram for the implemented database.

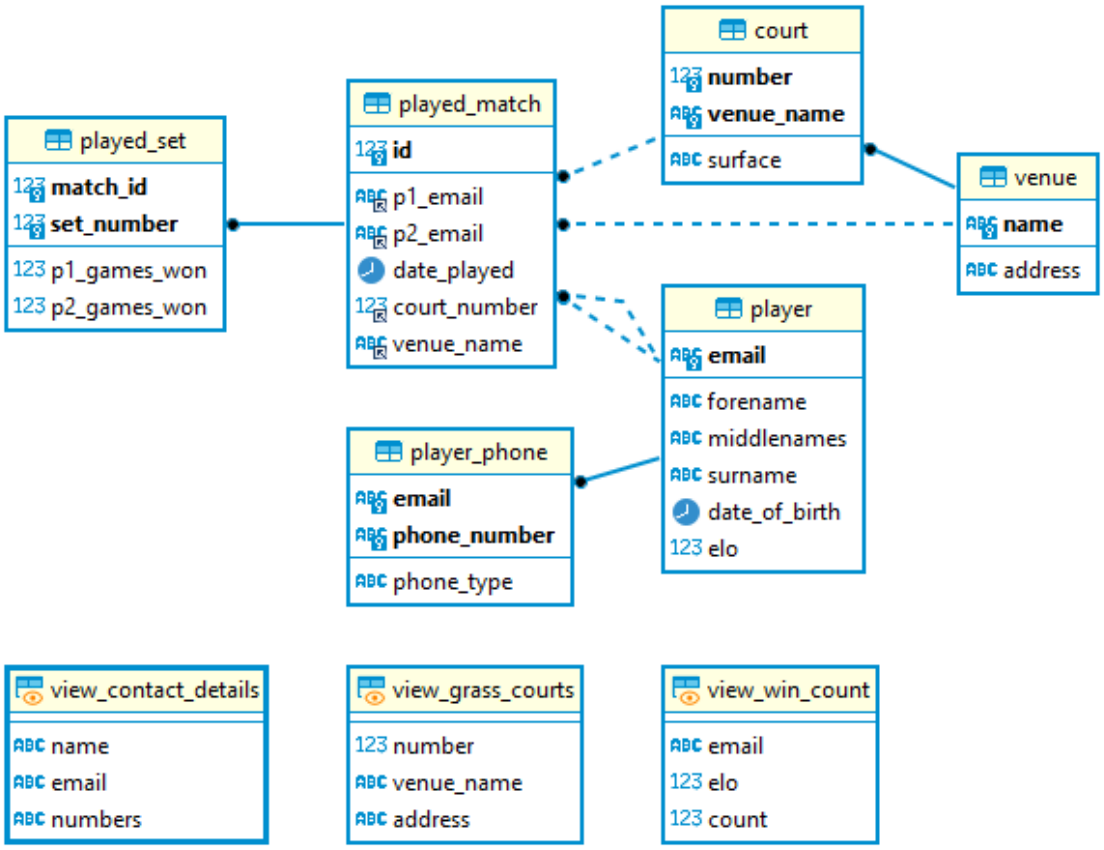

Figure 1: ER Diagram for the database corresponding to the given relational model created in DBeaver

Constraints were applied to ensure that data inputted was sensible. Details of the views and constraints will be given in Database Implementation, and procedures and functions were implemented to make it easier to perform certain queries.

All the fields were not allowed to be NULL, apart from middlenames which was optional. Elo was defaulted to 1000. Where a field has varchar as its type, sensible lengths were chosen in order to allow strings to be as large as is reasonable for what they were storing.

2.2 GUI

A GUI was created to provide an interface for the user to interact with the database without writing SQL.

The GUI has four main features.

1. Allows new members to sign up to the club (Required)
2. Allows members to view matches played at a venue (Required)
3. Allows members to add new phone numbers (Extension)
4. Allows members to see the match history of selected club members (Extension)

3 Database Implementation

3.1 Tables

Figure 2 gives an overview of the tables present in the database.

```
+-----------------------------+
| Tables_in_mrb21_cs3101_p2_db |
+-----------------------------+
| court                       |
| played_match                |
| played_set                  |
| player                      |
| player_phone                |
| venue                       |
| view_contact_details        |
| view_grass_courts           |
| view_win_count              |
+-----------------------------+
```

Figure 2: Result from executing a query to show all tables present in database

The following design decisions were made when setting up the tables.

- Phone numbers are stored as strings. This prevents the leading 0 being lost when data was inputted to the database.
- Elo was defaulted to 1000 for all players due to the fact the system is being introduced so it was assumed we would start everybody's scores off at 1000 at the start.

3.2 Views

The following views were all part of the required specification. All results are ordered by the variables that were stated in the spec. The code for each view can be found in the dump file.

3.2.1 Grass Courts

To produce the output, run : SELECT * from view_grass_courts. The output seen in Figure 3 is produced, as expected for the logic of the query.

```
MariaDB [mrb21_cs3101_p2_db]> select * from view_grass_courts;
+--------+-------------------------------+-----------------------------------+
| number | venue_name                    | address                           |
+--------+-------------------------------+-----------------------------------+
|      1 | Waterstone Crook Sports Centre | 69 Kirk Rd, Newport-on-Tay DD6 8HY |
|      2 | Waterstone Crook Sports Centre | 69 Kirk Rd, Newport-on-Tay DD6 8HY |
+--------+-------------------------------+-----------------------------------+
2 rows in set (0.001 sec)
```

Figure 3: Result from view_grass_courts

A natural join is done between court and venue, as information is required from both tables. A where clause is used to filter down to only the grass courts and to ensure that the venue name in court and venue is matching, as natural join would have created excess entries in the table being filtered.

3.2.2 Win Count

To produce the output seen in Figure 4, run : SELECT * from view_win_count

email	elo	count
butch@xyz.club	1000	6
final_fantasy_freak1993@hotmail.com	1000	3
gary_the_man@yahoo.co.uk	1000	10
jwh@hotmail.co	1000	5
leighton.buzzard@gmail.com	1000	0
louis.payne@gmail.com	1000	2
mad_maddy@gmail.com	1000	0
srrogers@yahoo.co.uk	1000	2
sylvia.hathaway@gmail.com	1000	5
tabitha.stacey@gmail.com	1000	5
tasha.marsden@gmail.com	1000	4
u_marsden@gmail.com	1000	4

12 rows in set (0.006 sec)

Figure 4: Output from the view_win_count view

Functions are used to help make the queries cleaner. The required functions for the view will be explained here.

The query gets the player's email, elo and the result of win_count when email is passed as the parameter.

The idea behind win_count is as follows. A person will have won a game if either of the conditions holds; they were player 1 and the winner of the match was player 1 or they were player 2 and the winner of the match was player 2. win_count will count how many times either of these cases occur, using is_winner_cur to return which player has won the match.

is_winner_cur also depends on some insight into how you can tell who has won a match, without needing to look at the results from 3 possible sets. The winner of the last set is the winner of the match. The difficulty now is deciding whether to look at set 2 or set 3 to find the winner. is_winner_cur queries played_set for the 3rd set, which is the last possible set. If this returns null, this means only 2 sets were played, and we just need to check which player won the second set, using winner_of_set. If the query does not return null then we follow the same steps, but look at the third set.

winner_of_set is used to avoid code duplication and just returns 1 if player 1 won the given set, and 2 otherwise.

While it seems like there is a lot of nested functions, they were used to avoid code duplication. Most of the functions are also stand alone functions that can be used in other parts of the code, if this system was expanded.

3.2.3 Contact Details

To produce the output seen in Figure 5, run : SELECT * from view_contact_details

name	email	numbers
Jamie Eugene Korey Butcher	butch@xyz.club	07969438448
Leighton Alan Buzzard	leighton.buzzard@gmail.com	01174960714,01314960962
Madeleine Daubney	mad_maddy@gmail.com	01154960961,02079460501
Sylvia Loraine Hathaway	sylvia.hathaway@gmail.com	07700900939
Jeremy Wardell Huddleston	jwh@hotmail.co	01314960470
Kirsten Aileen Louise Jackman	final_fantasy_freak1993@hotmail.com	07700900909
Gary Carl Marsden	gary_the_man@yahoo.co.uk	01514960777
Natasha Joy Bernardette Louise Marsden	tasha.marsden@gmail.com	07889344229
Ulysses Marsden	u_marsden@gmail.com	01314960745
Louis Kennard Payne	louis.payne@gmail.com	07700900654
Sue Rosemary Rogers	srrogers@yahoo.co.uk	07700900949
Tabitha Stacey	tabitha.stacey@gmail.com	07837585417

Figure 5: Output from the view_contact_details view

A natural join is taken between player and player phone, and the information about the phone numbers is retrieved, ensuring the emails match in both player phone and player tables, similar to the natural join described above.

CONCAT_WS concatenates the parts of the name together, and places a space between each part.

3.3 Constraints

Where the only constraint on the table is the primary key, this has been left out of the description, as no new insight can be found by presenting it again. All constraints are used to ensure that the data entered by the user makes sense in its context, and nobody can enter information that is not valid for that field.

E - extension feature, R - required feature

Player - information about the members of the club

1. Email contains a mixture of characters, along with one @ symbol at least (basic regex for format of email) (E)

    - This ensures that when the user is entering contact information, it is going to be usable. While full verification of an email address shouldn't be done in REGEX, this is a beginning.

Court - information about courts in a given venue

1. Court type must be one of grass, clay or hardcourt (R)

Player phone - information about phone numbers for members of the tennis club

1. Type of phone must be one of work, mobile or home (E)

    - Similar to court constraint, no need for any additional phone types.

2. The length of a phone number must be 11 (E)

    - Standard length of a phone. Allows the club to contact players if they have inputted phone, as it is going to be valid.

3. The phone number must only contain numerical characters (R)

Played set - information about the results of individual sets

1. The set number given is one of 1,2 or 3 (R)

2. In each set, one player should win exactly 6 games, and the other player should win 4 games or fewer; or alternatively, one player should win 7 games and the other player should win 5 or 6 games (R)

3.4 Triggers

1. match_date_check (E) : This is used to check that the date of a match being entered by a user is not in the future. The DATEDIFF function is used to calculate the difference between the inputted date and the current date. In this function, the difference has to be greater than 0 for a match to be allowed in DB.

2. date_check (E) : This ensures that people cannot join the tennis club too young. People are only allowed to become members when they are older than 5 years old, at which point children can begin playing sports in a club environment. The difference has to be greater than the number of days in 5 years.

3. null_middle_name (E) : This ensures that all middle names are set to null if they are passed in as an empty string. If the length of the middle name given is 0 then the middle name is set to NULL.

3.5 Functions/Procedures

1. proc_add_match (R) : This inserts new information into the DB about a recent match, while updating the elo scores of the two players according to the given formula. LAST_INSERT_ID is used to get the id of the match that has just gone into the DB, as that is auto incremented, and winner_of_set is used to decide how the C value is applied to each player.

2. name_from_email (E) : Mainly used in the GUI to allow the code to easily access the required information. Will return a concatenated version of all parts of the name given an email address.

4 GUI Implementation

The following section will detail the implementation side of the GUI developed. PHP was used due to lack of experience with web development. Figure 7 shows the front page of the website.

Figure 6: GUI as the user enters the site

4.1 Required features

The 'Join our Club' and the 'See Venue History' pages hold the required functionality of the GUI. Figures 8 and 9 illustrate the UI for these sites.

Figure 7: Form for creating new member

Figure 8: Form for retrieving venue history

Design choices for the implementation were as follows

1. Allowed one phone number for each type, which are all optional. This was the motivation for creating the additional contact information page, which would allow users to add more phone numbers. It was decided that collecting phone numbers was optional due to the ability to contact someone from their email, but if it was required, you could check the length of the entries for all three parts of the form, and only perform the query if one was non empty.

2. A free text form is given for the venue name. While this can lead to more errors, appropriate messages have been put to users. The match history is used to demonstrate ability to use results of SQL query as part of a drop down selection box instead of free text. With further time, the LIKE command could have been used to find similar venue names.

3. Links to navigate back to desired pages are placed throughout the site so that a user can easily get around the website.

Figure 9 and 10 illustrate the process of adding a new user to the database using the site.

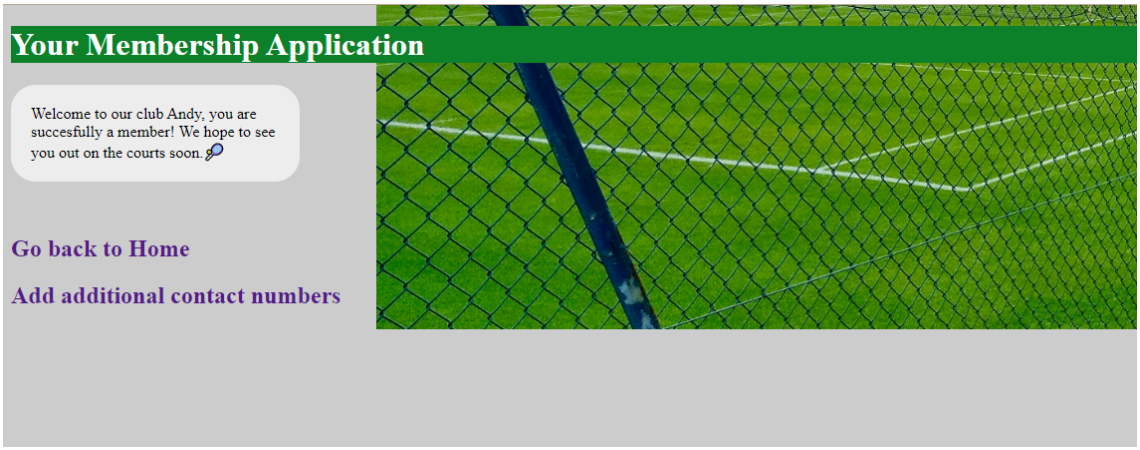

Figure 9: Site after a person is successfully entered into the database

```
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
| email                             | forename | middlenames           | surname   | date_of_birth | elo  |
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
| butch@xyz.club                    | Jamie    | Eugene Korey          | Butcher   | 1985-09-21    | 1000 |
| final_fantasy_freak1993@hotmail.com | Kirsten  | Aileen Louise         | Jackman   | 1993-10-28    | 1000 |
| gary_the_man@yahoo.co.uk          | Gary     | Carl                  | Marsden   | 1985-10-12    | 1000 |
| jwh@hotmail.co                    | Jeremy   | Wardell               | Huddleston | 1991-02-13    | 1000 |
| leighton.buzzard@gmail.com        | Leighton | Alan                  | Buzzard   | 1980-05-17    | 1000 |
| louis.payne@gmail.com             | Louis    | Kennard               | Payne     | 2000-05-31    | 1000 |
| mad_maddy@gmail.com               | Madeleine | NULL                  | Daubney   | 1991-03-08    | 1000 |
| srrogers@yahoo.co.uk              | Sue      | Rosemary              | Rogers    | 1965-07-30    | 1000 |
| sylvia.hathaway@gmail.com         | Sylvia   | Loraine               | Hathaway  | 2004-01-02    | 1000 |
| tabitha.stacey@gmail.com          | Tabitha  | NULL                  | Stacey    | 2005-09-10    | 1000 |
| tasha.marsden@gmail.com           | Natasha  | Joy Bernardette Louise | Marsden   | 1993-10-28    | 1000 |
| test@test.com                     | test     | test                  |           | 2000-12-14    | 1000 |
| u_marsden@gmail.com               | Ulysses  | NULL                  | Marsden   | 1977-05-07    | 1000 |
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
13 rows in set (0.000 sec)

MariaDB [mrb21_cs3101_p2_db]> select * from player;
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
| email                             | forename | middlenames           | surname   | date_of_birth | elo  |
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
| butch@xyz.club                    | Jamie    | Eugene Korey          | Butcher   | 1985-09-21    | 1000 |
| final_fantasy_freak1993@hotmail.com | Kirsten  | Aileen Louise         | Jackman   | 1993-10-28    | 1000 |
| gary_the_man@yahoo.co.uk          | Gary     | Carl                  | Marsden   | 1985-10-12    | 1000 |
| ilovetennis@gmail.com             | Andy     | NULL                  | Murray    | 1987-05-15    | 1000 |
| jwh@hotmail.co                    | Jeremy   | Wardell               | Huddleston | 1991-02-13    | 1000 |
| leighton.buzzard@gmail.com        | Leighton | Alan                  | Buzzard   | 1980-05-17    | 1000 |
| louis.payne@gmail.com             | Louis    | Kennard               | Payne     | 2000-05-31    | 1000 |
| mad_maddy@gmail.com               | Madeleine | NULL                  | Daubney   | 1991-03-08    | 1000 |
| srrogers@yahoo.co.uk              | Sue      | Rosemary              | Rogers    | 1965-07-30    | 1000 |
| sylvia.hathaway@gmail.com         | Sylvia   | Loraine               | Hathaway  | 2004-01-02    | 1000 |
| tabitha.stacey@gmail.com          | Tabitha  | NULL                  | Stacey    | 2005-09-10    | 1000 |
| tasha.marsden@gmail.com           | Natasha  | Joy Bernardette Louise | Marsden   | 1993-10-28    | 1000 |
| test@test.com                     | test     | test                  |           | 2000-12-14    | 1000 |
| u_marsden@gmail.com               | Ulysses  | NULL                  | Marsden   | 1977-05-07    | 1000 |
+-----------------------------------+----------+-----------------------+-----------+---------------+------+
14 rows in set (0.000 sec)
```

Figure 10: The state of the database before and after the insert, with the new member highlighted

Figure 11 illustrates the success of a query in retrieving the match history from a given venue.

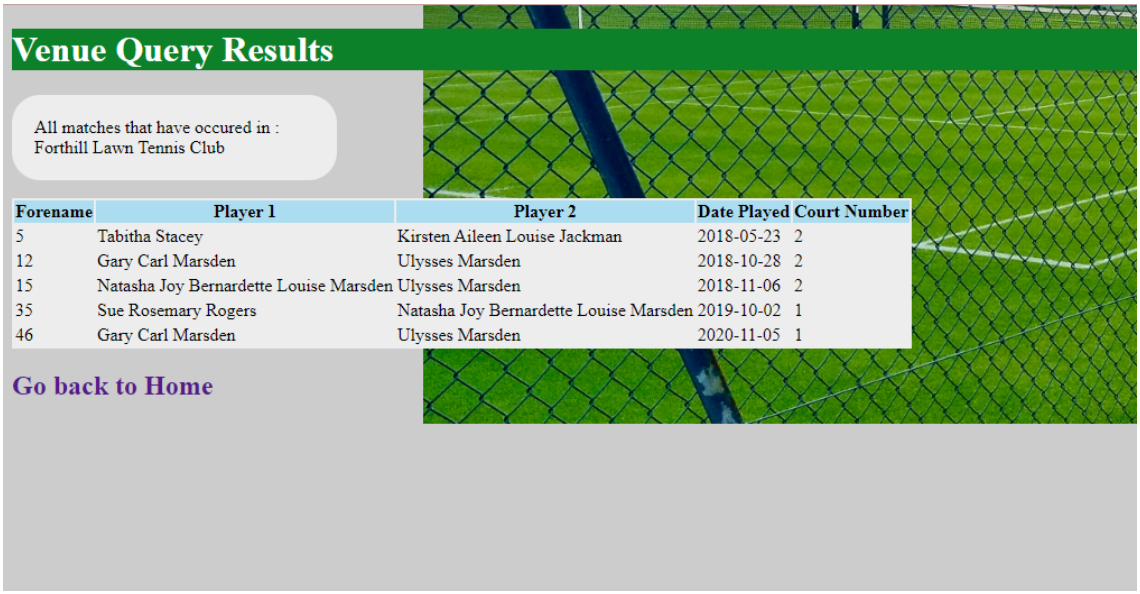

Figure 11: Retrieving the match history for Forthill Lawn Tennis Club

4.1.1 Error Messages

It is important that a user is provided by appropriate and helpful error messages when something goes wrong with the website. Mysqli error numbers are used to indicate where the error has been caused from, and error

messages are displayed accordingly. This allows the user to adjust their input, and continue to use the service.

The following error scenarios are dealt with:

1. Required information missing - happens when the user does not provide all the required information in the new membership form.
2. Error connecting to the database - unable to be demonstrated, but if something went wrong with DB connection, an error message will be displayed.
3. Constraints failed
    - Primary key (1062) - user tries to enter a variable that has already been used as a primary key (same email address on club membership etc)
    - Foreign key (1425) - user tries to enter a variable which is relying on a missing primary key in another table (contact info for email address not in records)
    - Trigger failed (1644) - user tries to enter an invalid date for match or DOB
    - Check failed (4025) - one of the constraints placed on the data is not held by what has been entered. Quite a broad category at the moment, could be benefited from added detail in error message.
4. No information returned from query (entering a venue that does not exist in the database)

Below is a select couple of screenshots of outputs from when an error is raised:

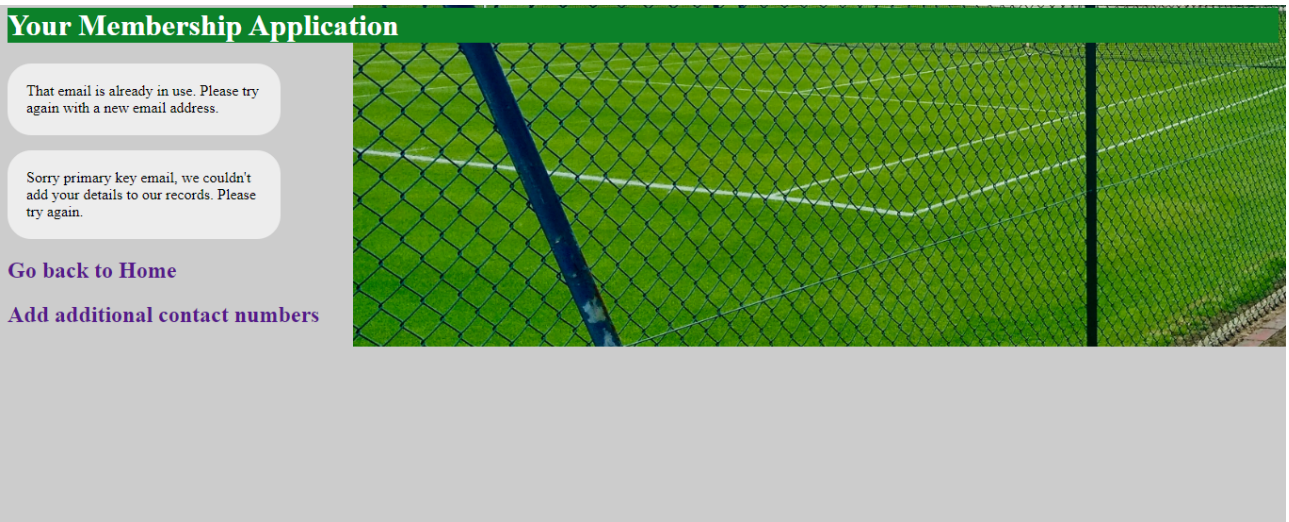

Figure 12: Error message when trying to use an already entered email for a new member

Figure 13: Error message when trying to enter contact information corresponding to a person not in the records

Figure 14: Dealing with the case where the venue name is invalid

Figure 15: Error message when the DOB of the user is within 5 years of the current date

Figure 16: Error message when the email entered for a new user does not have an @ symbol

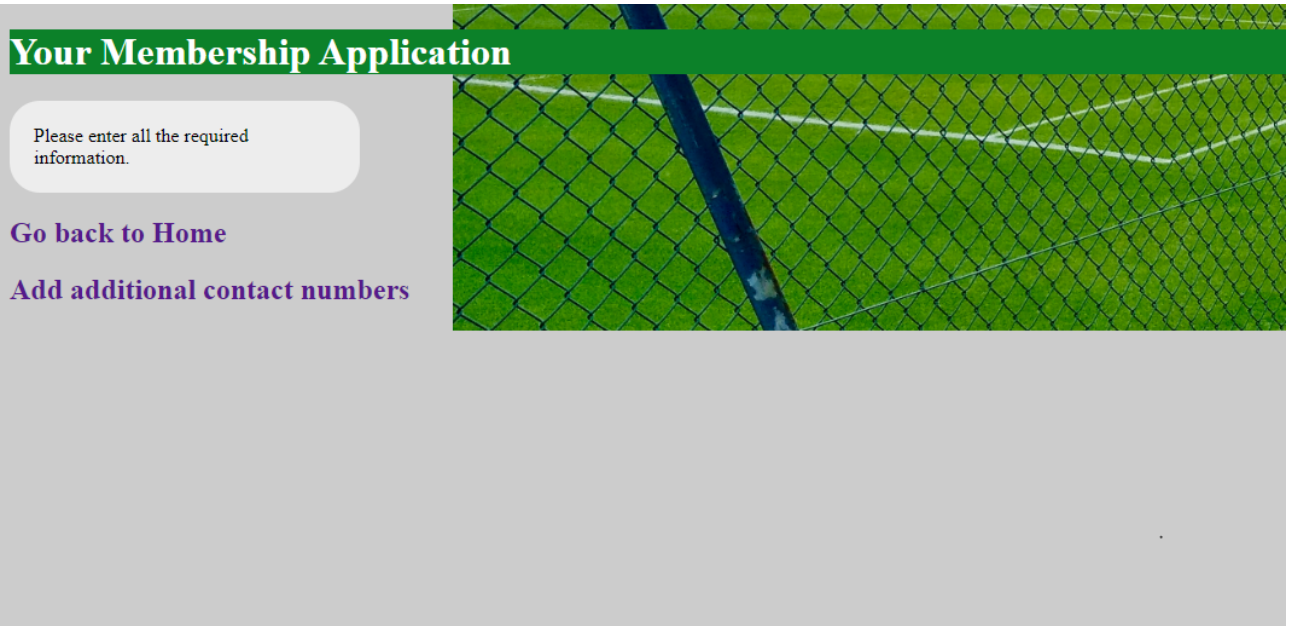

Figure 17: Trying to become a member and not filling in any of the records

4.2 Extension features

4.2.1 Contact info

Figure 18 illustrates the page which allows new contact information to be added to the records for a user already existing in the DB. The development of this page included learning how to pass information from a drop down menu back to the PHP files. This page was designed so that people could come back and fix errors with previous data entry or update records. This allows phone numbers to be added to the database.

Figure 18: GUI for entering a new phone number

Necessary error pages include if the user is not registered, and if the phone number is inputted incorrectly. Figure 19 illustrates the error messages in this instance.

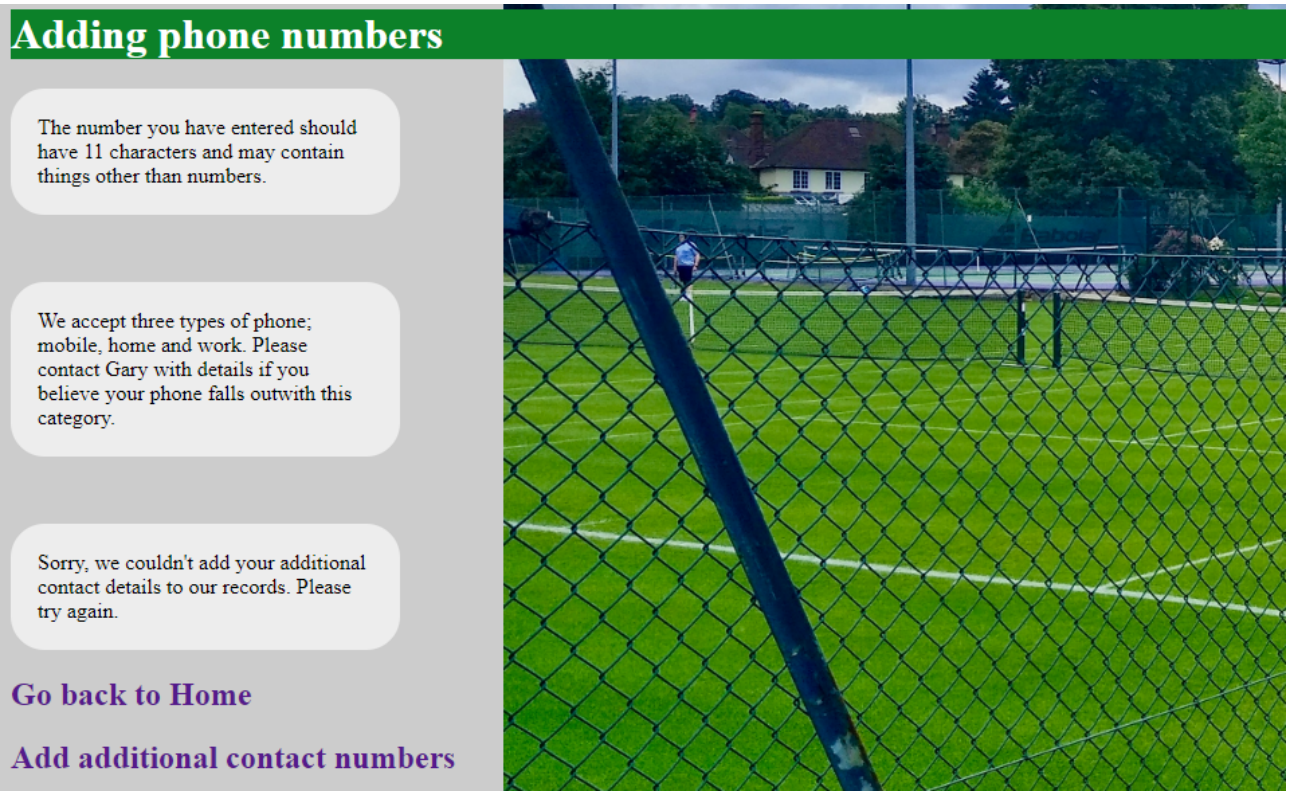

Figure 19: Error message for when the user tries to enter a phone number with letters in it

4.2.2 Match History

Figure 20 illustrates the GUI for the most complex of the website pages. The page can be used to retrieve the match history of a given player selected by their full name. It relies on a PHP file to first retrieve the names and emails of the players, displaying the names for choice and sending the corresponding email over to the next PHP, as the email allows for easier lookup of further information.

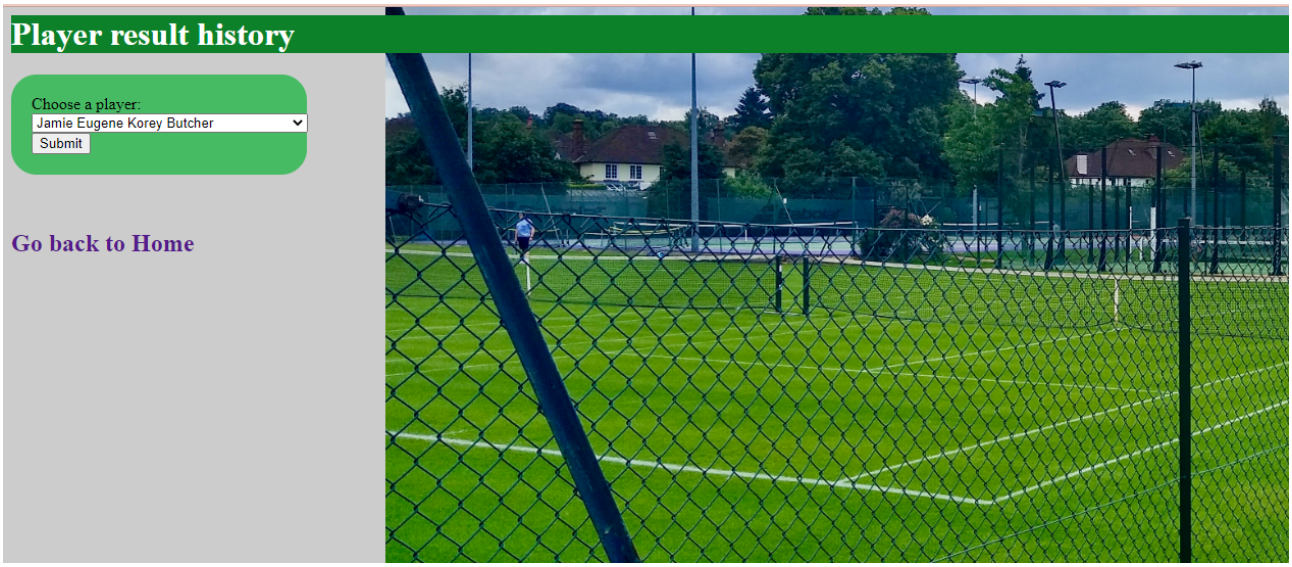

Figure 20: GUI for retrieving the match history of a player

The second PHP file retrieves all the matches a player has participated in, and outputs them onto the screen, as illustrated in Figure 21.

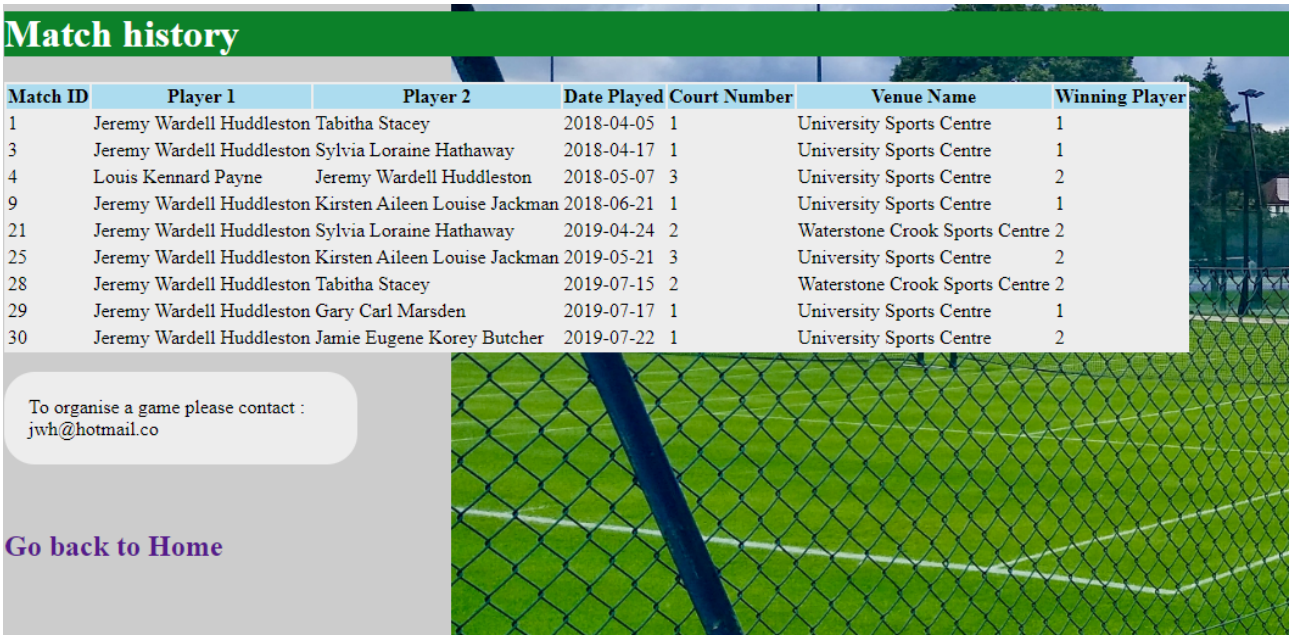

Figure 21: Result from retrieving match history of player who has participated in games

If a player has not participated in any matches, an appropriate error message is outputted, as illustrated in Figure 22.

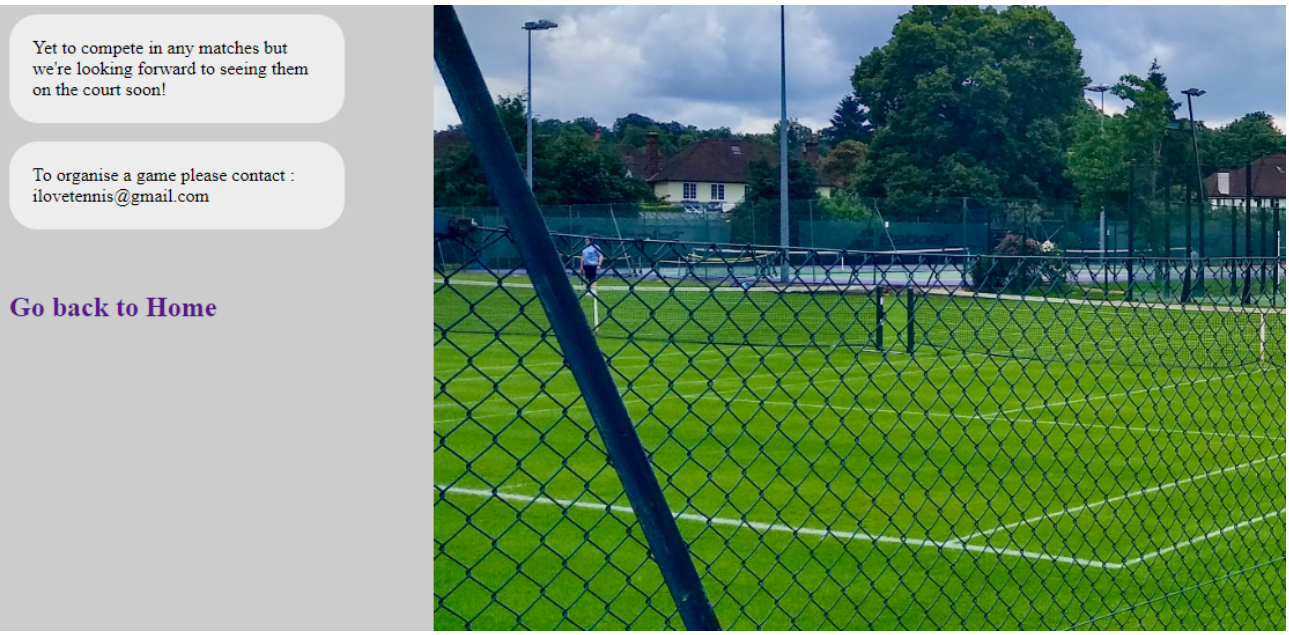

Figure 22: Result from retrieving match history of player who has not participated in games