

P2 - Implementation Report

CS3101 - Databases

Compilation, Execution & Usage Instruction

The languages I used are HTML, CSS and JavaScript alongside the node.js, express, bootstrap, and mysql frameworks. Node and express work together to run server-side logic dealing with GET and POST requests. Bootstrap was used to provide some boilerplate styling to the web-interface and mysql was used to connect to and query the MariaDB database.

I chose to use this technology stack as it fulfils the requirements of being able to query a MariaDB database and send data between the frontend and backend of the interface. It is also the stack that I am most familiar with out of the recommended sets of technologies.

To compile and run my GUI first open an ssh tunnel from localhost with port 3306 to the school host (this is the command I would run as jr263 'ssh jr263.host.cs.st-andrews.ac.uk -L 3306:localhost:3306 -N') . Once this is running navigate to the root directory in a terminal and run 'npm install' to install all the dependencies from package.json. Upon completion run the command 'node app.js' to run the server. The GUI web-interface can now be found at localhost:3005 (hopefully a free port).

The GUI is designed to be as simple to use as possible. Visiting localhost:3005 will land you on the home page which just acts as a title page. From here you can use the tabs at the top or click the 'Enter!' button. League matches can be found in the Leagues tab and new matches can be added to the database in the New Match tab.

Overview

For my submission I have created a MariaDB database on the school's MariaDB servers with tables to match each relation described in the provided relational schema (seven tables in total). To populate these tables I sorted the spreadsheet of data into clean tables with columns representing relation attributes. This numbers file will be included in the submission. I then exported the tables as csv files in an attempt to import them using the sql command 'load local data'. This didn't work for various reasons and my final solution was to write a python script that built and printed an insert query for every single record in the csv. I simply copied and pasted the output of this script into the MariaDB terminal and all the records were imported. This python script will also be included in the submission.

Once the database was populated I built out the views as required in part two of the specification, each stored under the provided names. Each of the constraints and the procedure, were also implemented as will be outlined in the next section.

My GUI implementation permits the use to choose a league and see all the matches that have been played in that league. Only matches from the selected year are shown. The web-interface also allows users to enter details of a new match into an HTML form and add a record to the database. If the resulting query results in an error, for violating a constraint for example, the user is told that there was an error but details of the error are not given. If I had more time to work on this project I would have changed the message to show details about the error in a user-friendly manner. In final, my submission fulfils all of the functional requirements missing out on extension implementation.

Database Implementation

Tables:

- court:
 - Court stores the different spots people can play at each venue. The number attribute is stored as a tinyint since no venue will realistically exceed 255 courts. The venue_name is stored as a varchar(255) which I used as a standard for all string fields for compatibility. Court uses both its attributes together to form a composite primary key.

```
MariaDB [jr263_CS3101_P2]> describe court;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| number     | tinyint(4)    | NO   | PRI | NULL    |       |
| venue_name | varchar(255)  | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.002 sec)
```

- league:
 - League stores details about the different competitions that the club offers. Here, the prize_money attribute is stored as a decimal(13,2) type which takes values with two digits after the decimal point as currency does. The winner_email attribute is of the type varchar(320) - a standard I used for all email address fields which allows 64 characters for the username, 1 character for the @ symbol, and 255 characters for the domain name [1]. League uses the title and year together as a composite primary key.

```
MariaDB [jr263_CS3101_P2]> describe league;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| name       | varchar(255)  | NO   | PRI | NULL    |       |
| year       | year(4)       | NO   | PRI | NULL    |       |
| prize_money | decimal(13,2) | NO   |     | NULL    |       |
| winner_email | varchar(320)  | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.002 sec)
```

- league_player
 - League_player stores which leagues the club members are registered to. It stores the email and string fields according to the outlined standards and the year attribute in the year type. All the attributes together make up the composite primary key.

```
MariaDB [jr263_CS3101_P2]> describe league_player;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| email      | varchar(320)  | NO   | PRI | NULL    |       |
| league_name | varchar(255)  | NO   | PRI | NULL    |       |
| league_year | year(4)       | NO   | PRI | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.002 sec)
```

- played_match

- Played_match is the largest table containing 10 attributes. Again, email and string fields follow the standards. The primary key in this table is the 'id' which I have stored as an int type to allow up to 4294967295 entries; more than enough. The id has the auto_increment property which means the id is calculated from how many entries have come before it. This removes the need for an id to be chosen and inserted by a user and ensures that all new ids will be unique. Both games_won attributes are stored as tinyint(4) since the max value is 3. This table uses a check constraint to assert that one player won three games and the other player won less than three games in every match.

```
MariaDB [jr263_CS3101_P2]> describe played_match;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
p1_email	varchar(320)	YES	MUL	NULL	
p2_email	varchar(320)	YES	MUL	NULL	
p1_games_won	tinyint(4)	YES		NULL	
p2_games_won	tinyint(4)	YES		NULL	
date_played	date	NO		NULL	
court_number	tinyint(4)	YES	MUL	NULL	
venue_name	varchar(255)	YES		NULL	
league_name	varchar(255)	YES	MUL	NULL	
league_year	year(4)	YES		NULL	

10 rows in set (0.002 sec)

- player

- Again, player uses standard types for email and string fields. The date_of_birth attribute is stored simply as a date type (ignoring the time component). The primary key of each player is their email address. The format of email addresses is enforced with a check constraint that makes the assertion email like '%_@_._%'

```
MariaDB [jr263_CS3101_P2]> describe player;
```

Field	Type	Null	Key	Default	Extra
email	varchar(320)	NO	PRI	NULL	
forename	varchar(255)	YES		NULL	
middlenames	varchar(255)	YES		NULL	
surname	varchar(255)	NO		NULL	
date_of_birth	date	NO		NULL	

5 rows in set (0.002 sec)

- player_phone

- This table stores member's phone numbers and allows for unique players to have multiple numbers with different purposes. I chose phone_number to be a varchar(15) since the international standard for phone numbers can support up to 15 digits [2]. Phone_type is stored as a varchar(6) as its largest value is 6 ("mobile"). A check constraint enforces that all entries into the phone_type column are either "home", "mobile", or "work".

```
MariaDB [jr263_CS3101_P2]> describe player_phone;
```

Field	Type	Null	Key	Default	Extra
email	varchar(320)	NO	PRI	NULL	
phone_number	varchar(15)	NO	PRI	NULL	
phone_type	varchar(6)	NO		NULL	

3 rows in set (0.002 sec)

- venue
- Venue is a very simple table with two varchar(255) attributes for the name and address of each squash venue. Nothing else to note.

```
MariaDB [jr263_CS3101_P2]> describe venue;
```

Field	Type	Null	Key	Default	Extra
name	varchar(255)	NO	PRI	NULL	
address	varchar(255)	YES		NULL	

2 rows in set (0.002 sec)

Views:

- view_contact_details:
- Query:
 - select distinct `x`.`fullname` AS `fullname`,`x`.`email` AS `email`,`y`.`numbers` AS `phone_number(s)` from (((select case when `jr263_CS3101_P2`.`player`.`middlenames` = '' then concat(`jr263_CS3101_P2`.`player`.`forename`,`jr263_CS3101_P2`.`player`.`surname`) else concat_ws('`,`jr263_CS3101_P2`.`player`.`forename`,`jr263_CS3101_P2`.`player`.`middlenames`,`jr263_CS3101_P2`.`player`.`surname`) end AS `fullname`,`jr263_CS3101_P2`.`player`.`email` AS `email`,`jr263_CS3101_P2`.`player_phone`.`phone_number` AS `phone_number` from (`jr263_CS3101_P2`.`player` left join `jr263_CS3101_P2`.`player_phone` on(`jr263_CS3101_P2`.`player`.`email` = `jr263_CS3101_P2`.`player_phone`.`email`)) order by `jr263_CS3101_P2`.`player`.`surname`,case when `jr263_CS3101_P2`.`player`.`middlenames` = '' then concat(`jr263_CS3101_P2`.`player`.`forename`,`jr263_CS3101_P2`.`player`.`surname`) else concat_ws('`,`jr263_CS3101_P2`.`player`.`forename`,`jr263_CS3101_P2`.`player`.`middlenames`,`jr263_CS3101_P2`.`player`.`surname`) end,`jr263_CS3101_P2`.`player`.`email` limit 9999999)) `x` left join (select `jr263_CS3101_P2`.`player_phone`.`email` AS `email`,group_concat(`jr263_CS3101_P2`.`player_phone`.`phone_number` separator ', ') AS `numbers` from `jr263_CS3101_P2`.`player_phone` group by `jr263_CS3101_P2`.`player_phone`.`email`) `y` on(`x`.`email` = `y`.`email`))
- Output:

```
MariaDB [jr263_CS3101_P2]> select * from view_contact_details;
```

fullname	email	phone_number(s)
Jamie Eugene Korey Butcher	butch@xyz.club	079 6943 8448
Leighton Alan Buzzard	leighton.buzzard@gmail.com	0117 496 0714, 0131 496 0962
Madeleine Daubney	mad_maddy@gmail.com	0115 496 0961, 020 7946 0501
Sylvia Loraine Hathaway	sylvia.hathaway@gmail.com	07700 900939
Jeremy Wardell Huddleston	jwh@hotmail.com	0131 496 0470
Kirsten Aileen Louise Jackman	final_fantasy_freak1993@hotmail.com	07700 900909
Gary Carl Marsden	gary_the_man@yahoo.co.uk	0151 496 0777
Natasha Joy Bernardette Louise Marsden	tasha.marsden@gmail.com	078 8934 4229
Ulysses Marsden	u_marsden@gmail.com	0131 496 0745
Louis Kennard Payne	louis.payne@gmail.com	07700 900654
Sue Rosemary Rogers	srrogers@yahoo.co.uk	07700 900949
Jordan Dylan Rowley	jr263@st-andrews.ac.uk	NULL
Tabitha Stacey	tabitha.stacey@gmail.com	07837 585417

13 rows in set (0.006 sec)

- view_never_played

- Query:

```
select `court`.`number` AS `number`,`venue`.`name` AS
`name`,`venue`.`address` AS `address` from (`court` join `venue`
on(`court`.`venue_name` = `venue`.`name`)) where !exists(select
`played_match`.`court_number`,`played_match`.`venue_name` from
`played_match` where `court`.`number` =
`played_match`.`court_number` and `venue`.`name` =
`played_match`.`venue_name` limit 1)
```

- Output:

```
[MariaDB [jr263_CS3101_P2]> select * from view_never_played;
```

number	name	address
1	The Piccadilly Health Club and Spa	21 Piccadilly, Westminster W1J 0DH
2	The Piccadilly Health Club and Spa	21 Piccadilly, Westminster W1J 0DH
3	The Piccadilly Health Club and Spa	21 Piccadilly, Westminster W1J 0DH
4	The Piccadilly Health Club and Spa	21 Piccadilly, Westminster W1J 0DH
5	The Piccadilly Health Club and Spa	21 Piccadilly, Westminster W1J 0DH
2	University Sports Centre	9 St Leonard's Rd, St Andrews KY16 9DY
3	Waterstone Crook Sports Centre	69 Kirk Rd, Newport-on-Tay DD6 8HY

```
7 rows in set (0.002 sec)
```

- view_win_count

- Query:

```
select `wincount`.`winner` AS `winner`,count(0) AS `number of
matches won` from (select case when
`jr263_CS3101_P2`.`played_match`.`p1_games_won` = 3 then
`jr263_CS3101_P2`.`played_match`.`p1_email` when
`jr263_CS3101_P2`.`played_match`.`p2_games_won` = 3 then
`jr263_CS3101_P2`.`played_match`.`p2_email` end AS `winner` from
`jr263_CS3101_P2`.`played_match`) `wincount` group by
`wincount`.`winner` order by `wincount`.`winner`
```

- Output:

```
[MariaDB [jr263_CS3101_P2]> select * from view_win_count;
```

winner	number of matches won
butch@xyz.club	3
final_fantasy_freak1993@hotmail.com	4
gary_the_man@yahoo.co.uk	6
jr263@st-andrews.ac.uk	3
jwh@hotmail.com	2
louis.payne@gmail.com	4
srrogers@yahoo.co.uk	6
sylvia.hathaway@gmail.com	7
tabitha.stacey@gmail.com	8
tasha.marsden@gmail.com	3
u_marsden@gmail.com	3

```
11 rows in set (0.002 sec)
```

Functions:

- checkLeaguePlayer
 - CREATE DEFINER=`jr263`@`%` FUNCTION
`checkLeaguePlayer`(player_email varchar(320), name varchar(255),
year year) RETURNS int(11) DETERMINISTIC
 - return exists(select * from league_player where email=player_email
and league_name=name and league_year=year)
- This function was written to run check constraints that can access fields in other tables. The plan ended up not working since you can't have a user-defined function inside a check condition.

Procedures:

- proc_add_venue
 - CREATE DEFINER=`jr263`@`%` PROCEDURE `proc_add_venue`(in
venue_name_in varchar(255), in venue_address varchar(255), in
courts tinyint)
 - begin declare counter int default 0; set counter = 1; if courts>0
then insert into venue(name, address) values (venue_name_in,
venue_address); while counter<=courts do insert into court(number,
venue_name) values (counter, venue_name_in); set counter = counter
+ 1; end while; else signal sqlstate '45000' set message_text =
'Negative number of courts specified'; end if; end
- This procedure inserts a record into both the venue and the court tables ensuring foreign key constraints hold.

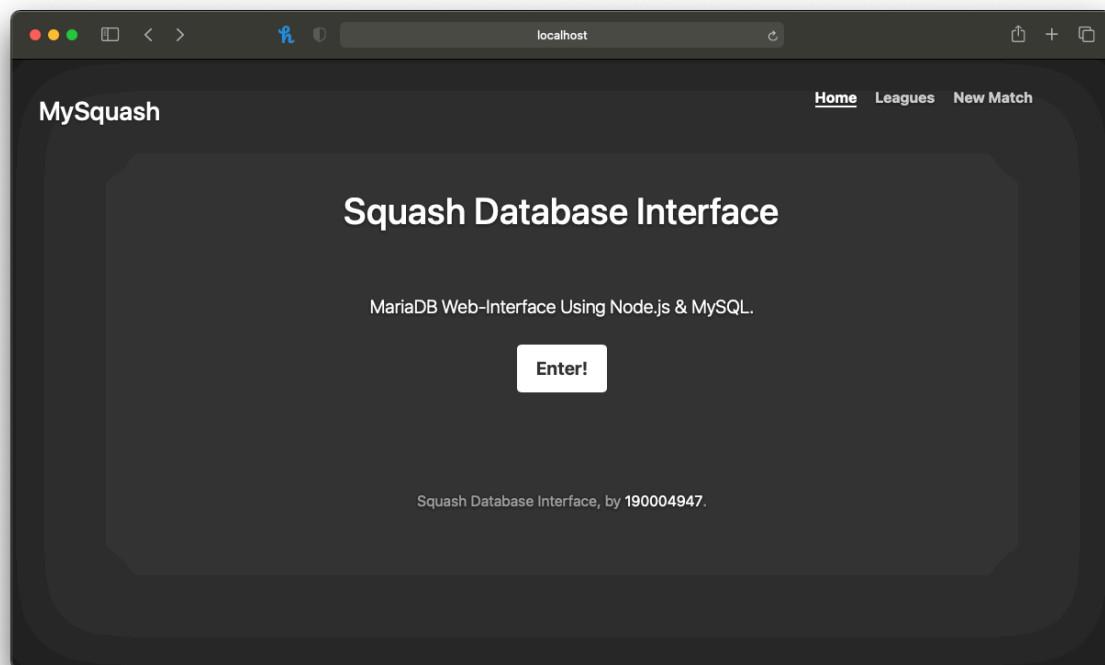
I didn't end up using triggers in my implementation although I did research the idea of using them to make cross-table check constraints possible.

GUI Implementation

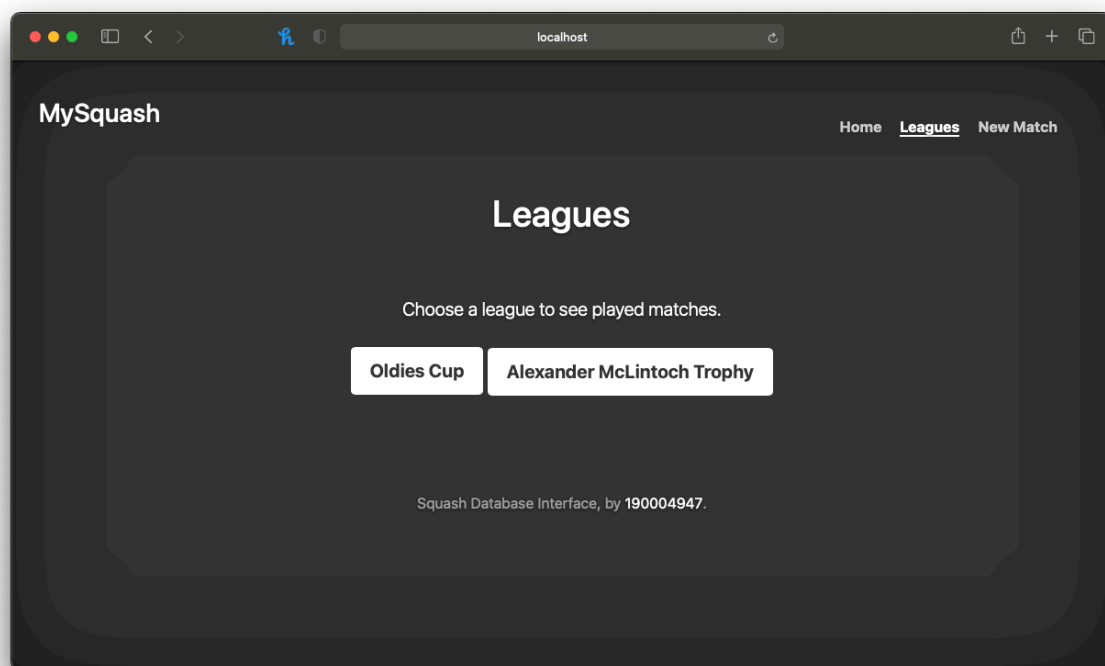
File Type	Purpose	Source
HTML	Stores a view of objects and data served to clients.	Boilerplate skeleton generated to correspond with bootstrap styling. Objects and scripts written from scratch.
CSS	Holds all details and parameters regarding the styling of every	Most styling is from the bootstrap boilerplate with additions and modifications.
JavaScript	Runs server-side logic, handles database connection and querying, facilitates GET and POST requests.	Written from scratch.
JSON	Stores information about the project and it's dependencies.	Automatically generated.

Discuss features, design, and implementation decisions of the GUI.

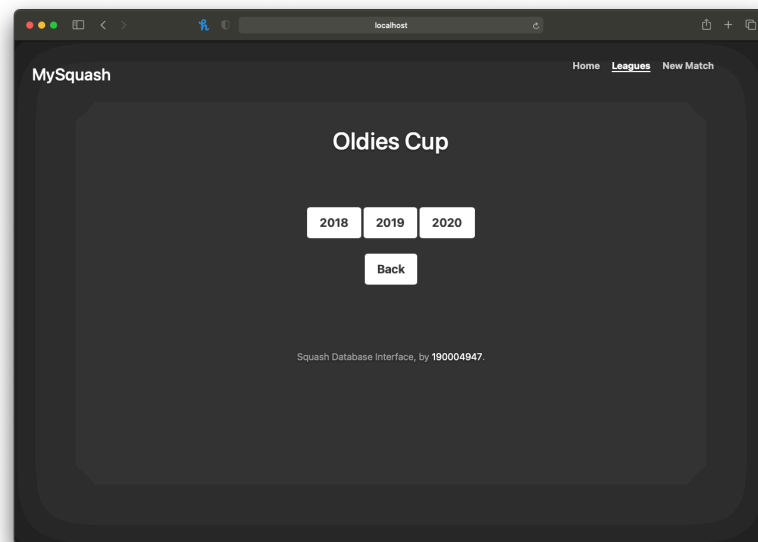
My web-interface has a unique HTML file for each page instead of using shared HTML files and deciding which objects to show based on context. The home page simply welcomes you to the interface; it has no other features.



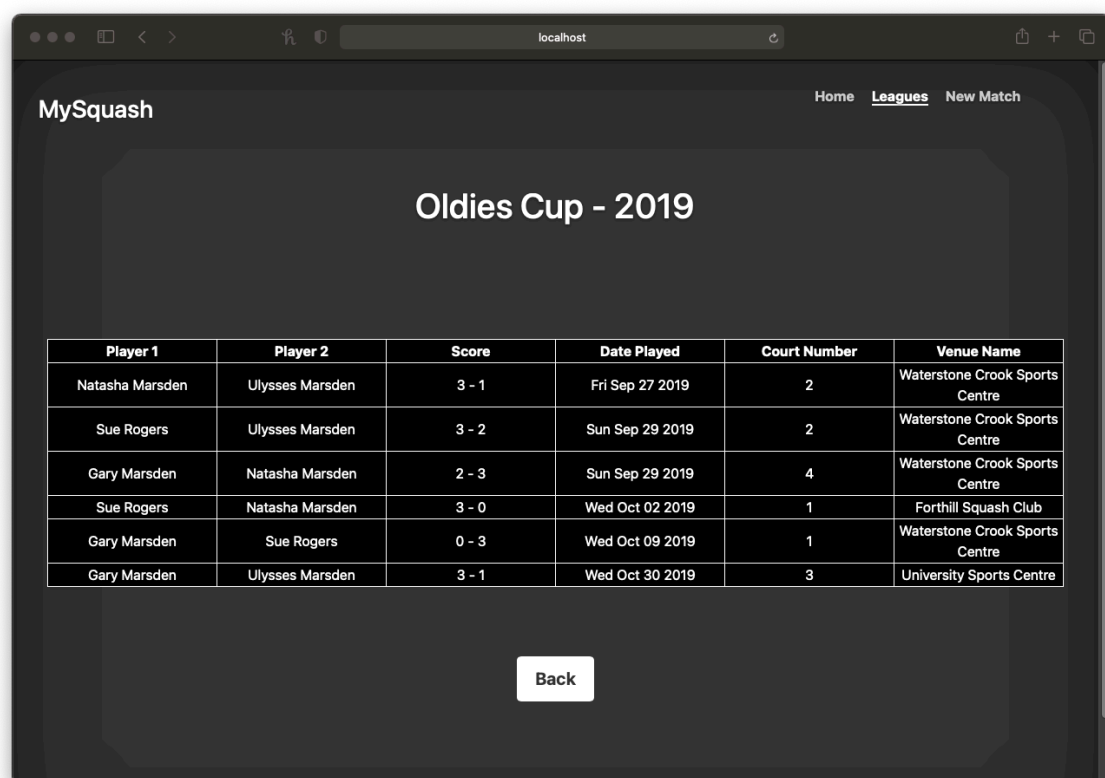
Clicking the enter button (or the Leagues tab) will bring you to the page depicted below where the user will choose which league they wish to see played matches of.



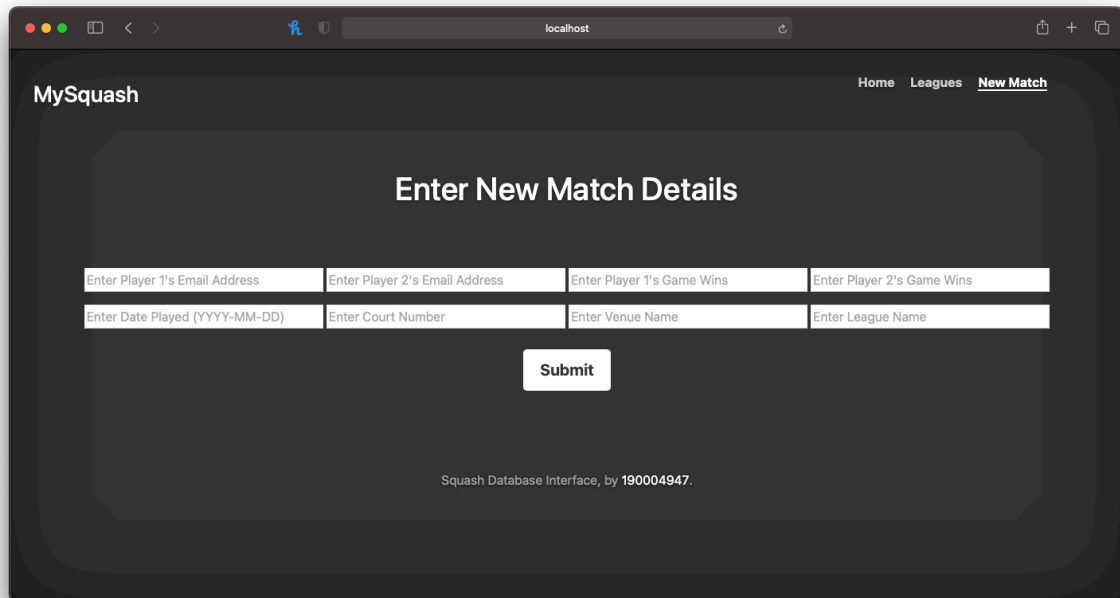
Suppose the user chooses to see matches played in the Oldies Cup, they are then brought to this page letting them specify which year they are interested in. I designed the UI in this way to keep the system as simple as possible; it is, at no point, ambiguous about where to find information a user might be searching for. It also only requires point-and-click actions.



After selecting a year, a fetch request is made to get the results of querying the database for matches played in that league during that year. The query is built using parameters passed from the webpage (for example being on the Oldies Cup 2018 page sends 'Oldies Cup' and 2018 as parameters to be used in the query). Once the backend gets a query response, the data is parsed and sent back to the frontend through a GET method. A script in the HTML file will dynamically add rows to the tables depending on how many records were returned in the query response.



The other feature of the web interface is inserting new matches into the database through an HTML form. I used input boxes and a submit button for simplicity but if I were to do this project again I would change the input boxes into dropdown menus displaying the possible options based on what records are in the database tables. However, as it stands, this page takes in parameters for an insert query through input boxes. The data from the entry fields are sent to the backend which builds the query string and inserts the new match into the played_match table if no constraints are violated.



The screenshot shows a web browser window with the URL 'localhost'. The page title is 'MySquash'. In the top right corner, there are navigation links: 'Home', 'Leagues', and 'New Match' (which is highlighted). The main content area is titled 'Enter New Match Details'. It contains a form with eight input fields arranged in two rows of four. The first row fields are: 'Enter Player 1's Email Address', 'Enter Player 2's Email Address', 'Enter Player 1's Game Wins', and 'Enter Player 2's Game Wins'. The second row fields are: 'Enter Date Played (YYYY-MM-DD)', 'Enter Court Number', 'Enter Venue Name', and 'Enter League Name'. Below the input fields is a 'Submit' button. At the bottom of the form area, there is a small text attribution: 'Squash Database Interface, by 190004947.'

Upon submission the page will either turn red or green and give the user a message saying whether or not the insert was successful. Unfortunately I didn't work out how to show the user an error message which expressed why the error happened.

References

- [1] <https://tools.ietf.org/html/rfc5321.html>
- [2] https://en.wikipedia.org/wiki/Telephone_number