SIM UOW ISIT307

ASSIGNMENT 1

Last updated: 29 January 2024

Timothy Mah

10258663

Tutorial Group T01F

Assignment group Tim1 T01FG12

Contents

[Introduction 2](#_Toc157434264)

[Usage 2](#_Toc157434265)

[Program flow 2](#_Toc157434266)

[Screenshots 3](#_Toc157434267)

[“Database” files 3](#_Toc157434268)

[Extra features 3](#_Toc157434269)

[Server and client side input validation 3](#_Toc157434270)

[URL whitelisting 3](#_Toc157434271)

[Username whitelisting 3](#_Toc157434272)

[Mobile friendly 3](#_Toc157434273)

[Modern full stack design 3](#_Toc157434274)

[Errors are not shown to user 3](#_Toc157434275)

[Limitations 3](#_Toc157434276)

# Introduction

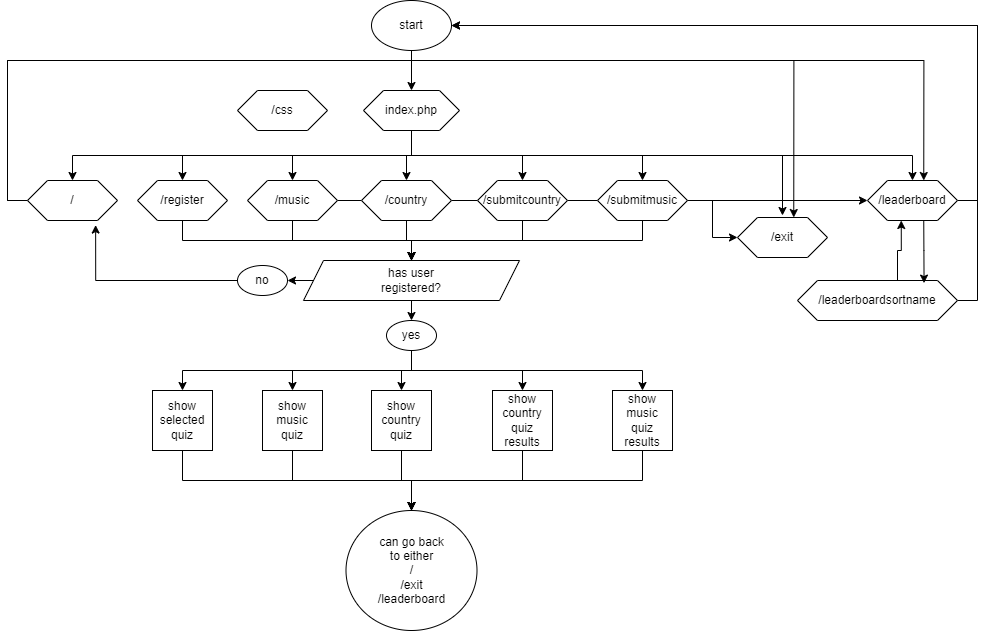
Hello this is my documentation for SIM UOW ISIT307ASSIGNMENT 1.

# Usage

Access localhost in browser

# Program flow

This is the flow of the website



When the user access the site domain – localhost, in this case, the first page that the apache server goes to is index.php. I have configured this in my .htacess file. From there, index.php takes in the request and compares it against a switch case statement essentially creating a router of sorts. The index.php is sort of like the server middleman between the frontend (HTML pages) and the backend (.txt files). In this case I have only used one “HTML” webpage, akin to React.js.

Like React Components, the different routes would send the corresponding component for each route.

The purpose of this is to

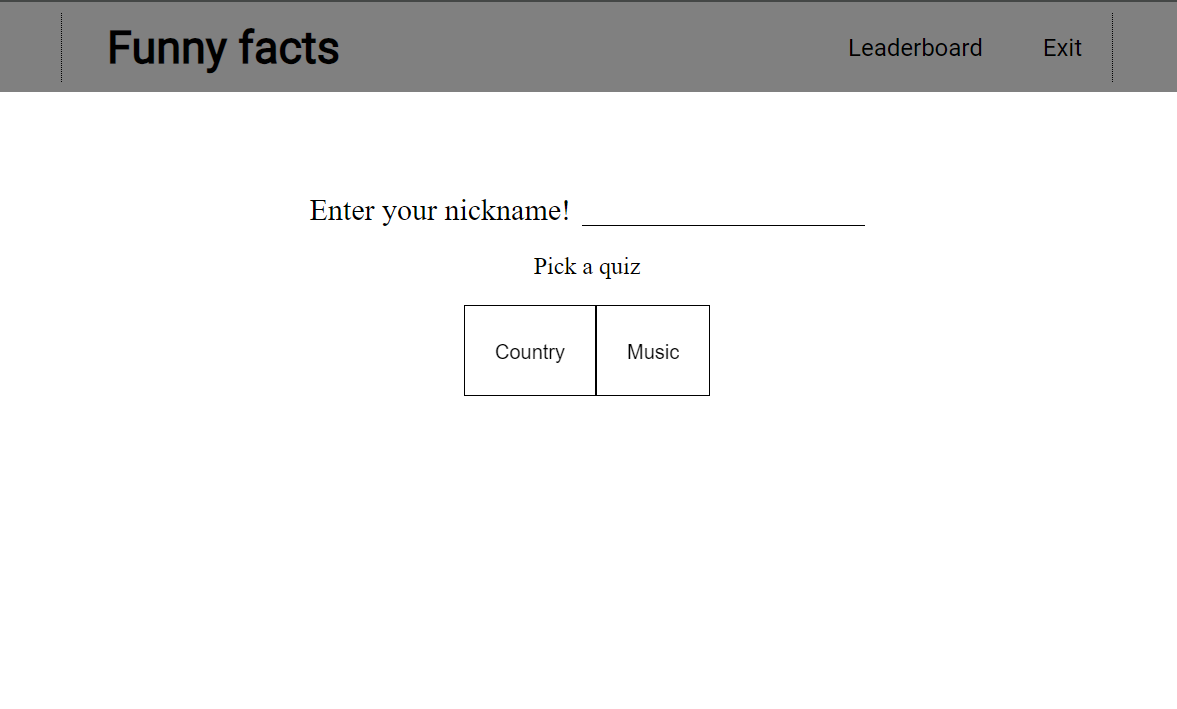
1. Make code cleaner
2. I don’t have to keep repeating the webpage header in every webpage. The header component is simply called in every instance where the webpage is shown. This applies to any segment of the webpage that is repeatedly used in different routes. Interoperability.

There are 10 routes in total

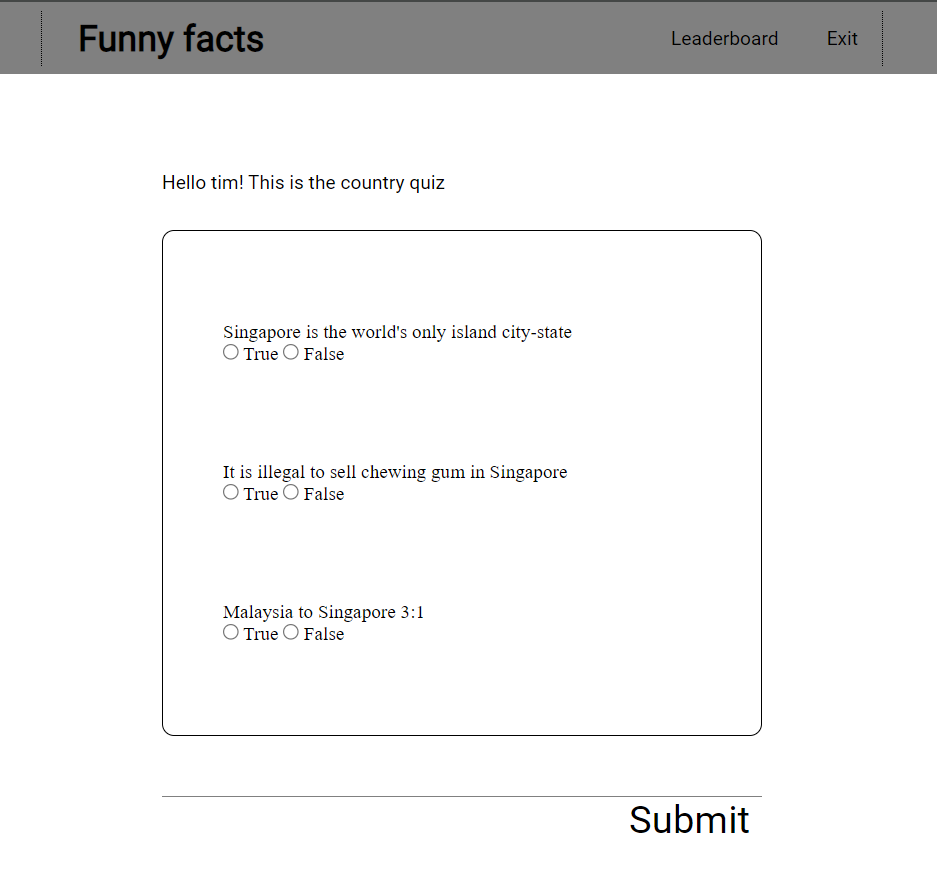
1. /
   1. This is the root route. It’s the route the user would access when accessing the website through the top level domain – localhost.
2. /css
   1. This is where the css file is stored. I use one css file for all pages. All webpages access the css file from this route
3. /register
   1. Like the other routes, the index.php server would check if there is a user logged in. If user not logged in, user is returned the “enter your nickname” components.
   2. This route does not return a webpage. It processes the user input at the / route, checks the userRecord.txt if the user exists. If not, create new user. Then finally sends the user to either the /music or /country route.
4. /music
   1. This is the music quiz page. Like the other routes, the index.php server would would check if there is a user logged in. If user not logged in, user is redirected back to /
   2. After validating for user log on, it will read the mquiz.txt to display the images of the different albums
   3. At the same time the server stores the music “questions” and “correct answers” in session
5. /country
   1. Like the other routes, the index.php server would check if there is a user logged in. If user not logged in, user is redirected back to /
   2. Similar to the /music route, after validating for user log on, it will read the mquiz.txt to display 3 different questions at random. Current datetime is used as a seed.
   3. At the same time the server stores the country “questions” and “correct answers” in session
6. /submitmusic
   1. This is where the user is directed after submitting the quiz at /music
   2. Like the other routes, the index.php server would check if there is a user logged in. If user not logged in, user is redirected back to /
   3. The server will draw the correct questions and answers, and then compare the user’s input answers. The user’s answers are stored together with the corresponding “questions”. The server then counts the number of correct and wrong questions
   4. The total score of the current sitting is calculated
   5. The total score of current sitting is accumulated with the corresponding user’s score from the userRecord.txt
7. /submitcountry
   1. Basically the same as /submitmusic but the music “questions” are not randomly shown
8. /exit
   1. Like the other routes, the index.php server would check if there is a user logged in. If user not logged in, user is redirected back to /
   2. The current user and total score of current user’s sitting is shown
9. /leaderboard
   1. Leaderboard is shown. User does not need to be logged in to view this page
   2. Can be sort by score (the default) or by name
10. /leaderboardsortname
    1. Same as /leaderboard but the leaderboard is sort by name

## Screenshots

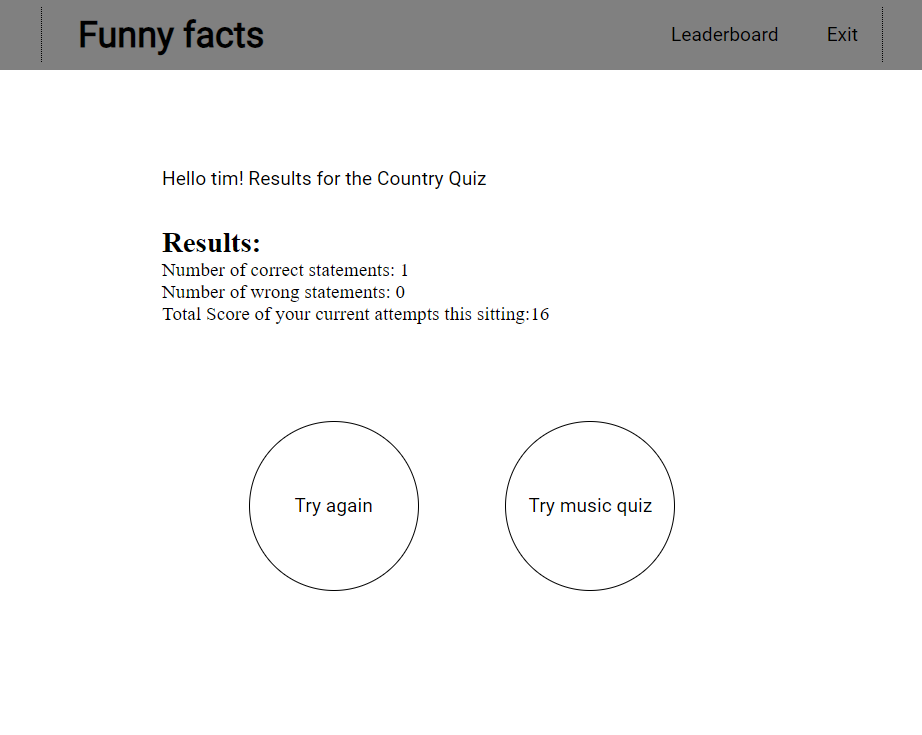
This is the first page the user sees. From here, the user may view the leaderboard. All other pages require nickname to be entered first.



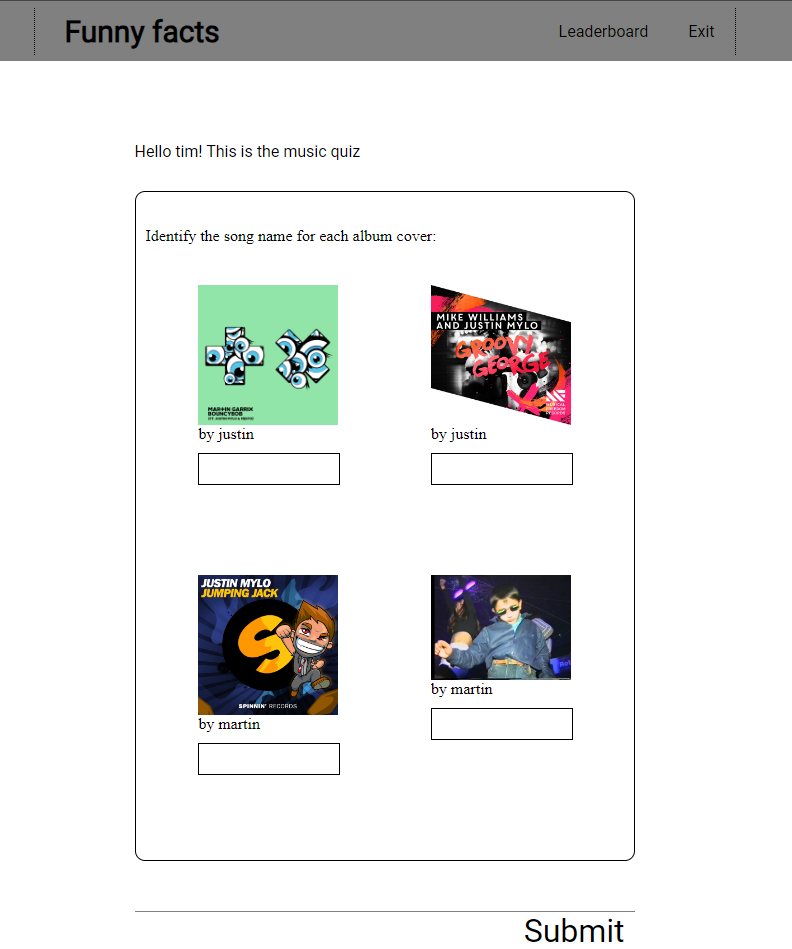
When user enters nickname and picks country, the country quiz is shown. Questions are at random as per assignment brief. From here, user can access the leaderboard page, exit page, or go back to the “home” page (the Funny facts logo) to enter different username or enter same username and attempt a different quiz



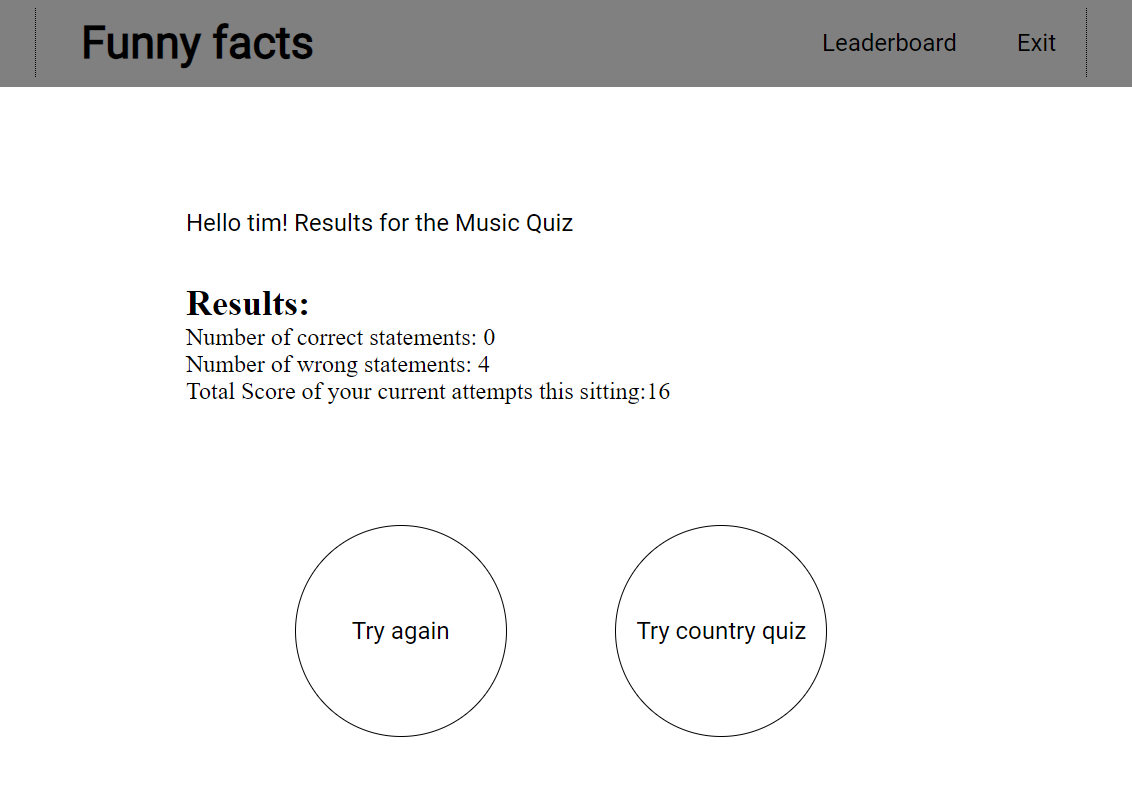
When user submits the country quiz, the results are shown. User can try the country quiz again or try the music quiz. From here, user can access the leaderboard page, exit page, or go back to the “home” page (the Funny facts logo) to enter different username or enter same username and attempt a different quiz.



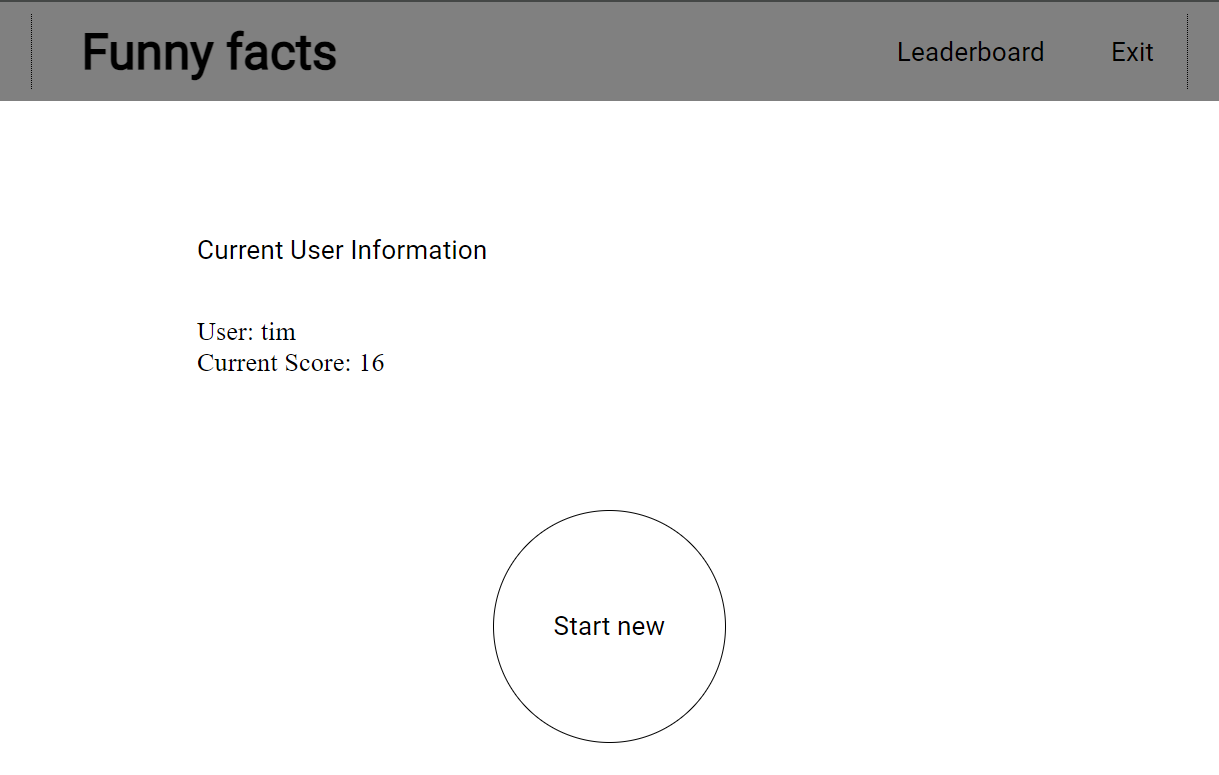
When user tries music quiz, the music quiz is shown. The album cover image followed by the composer data is shown. User may enter the album name. If left blank, the “question” is marked as wrong. From here, user can access the leaderboard page, exit page, or go back to the “home” page (the Funny facts logo) to enter different username or enter same username and attempt a different quiz.



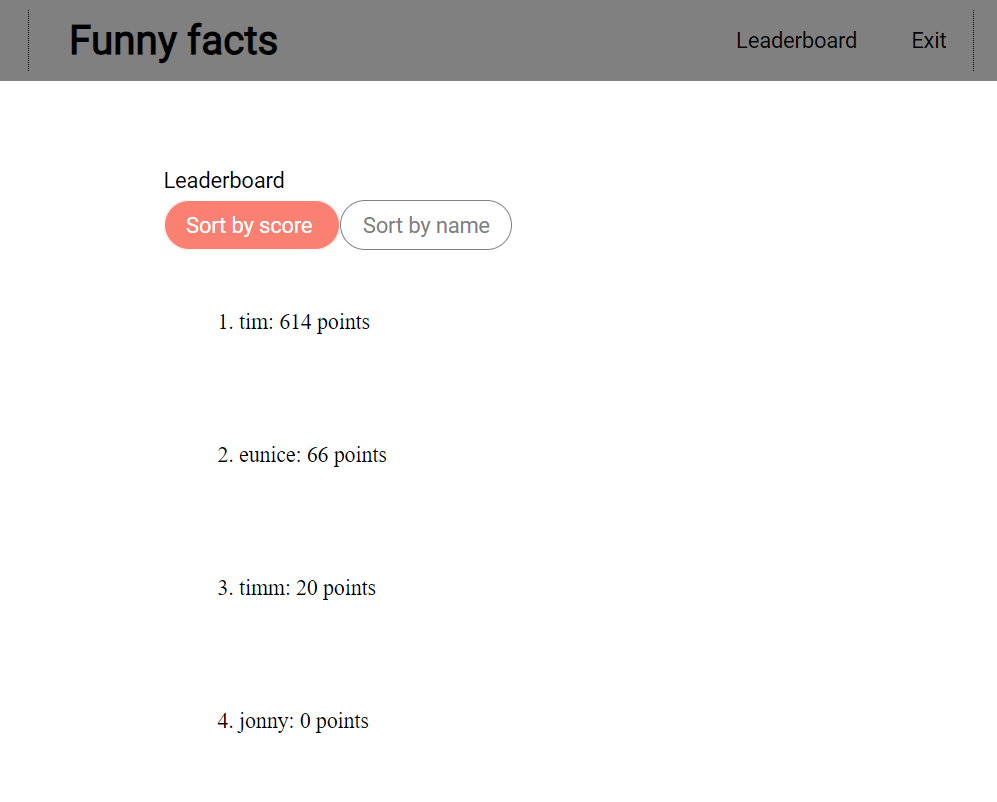
When user submits the music quiz, the results are shown. Similar to how the country quiz results are shown. From here, user can access the leaderboard page, exit page, or go back to the “home” page (the Funny facts logo) to enter different username or enter same username and attempt a different quiz.



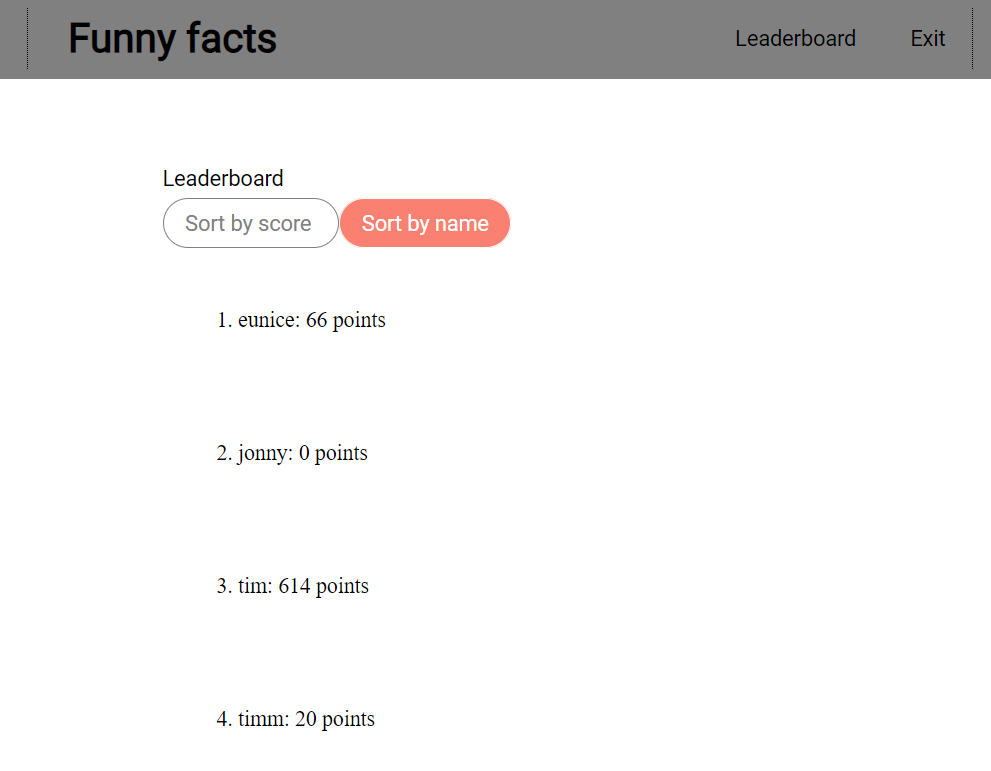
Should the user click on Exit, the exit page is shown. The user nickname and current total score of current sitting is shown. From here user can view leaderboard or start new quiz



From any page, user can view the leaderboard. By default it is sort by score. User can sort by name

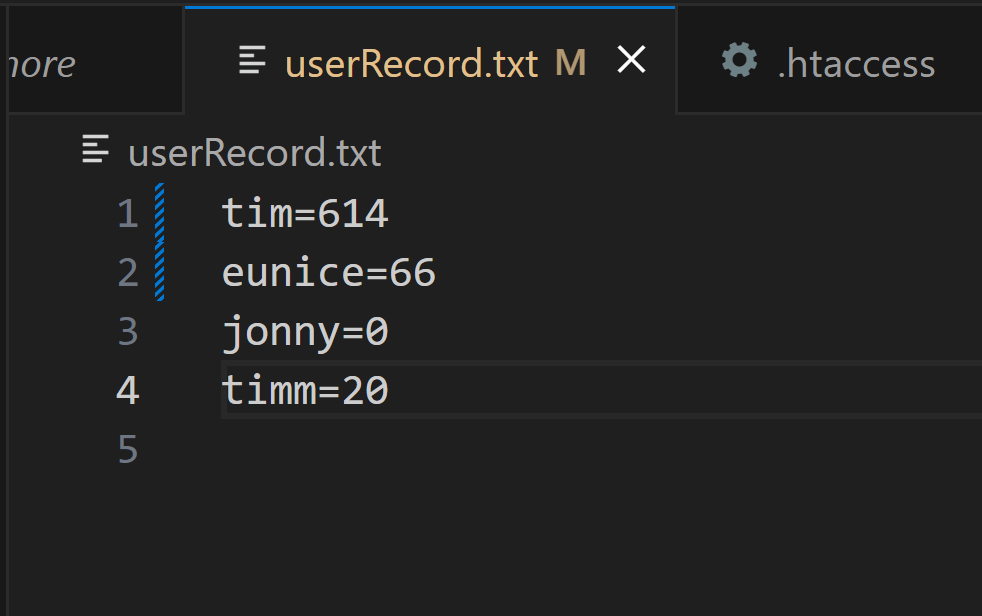
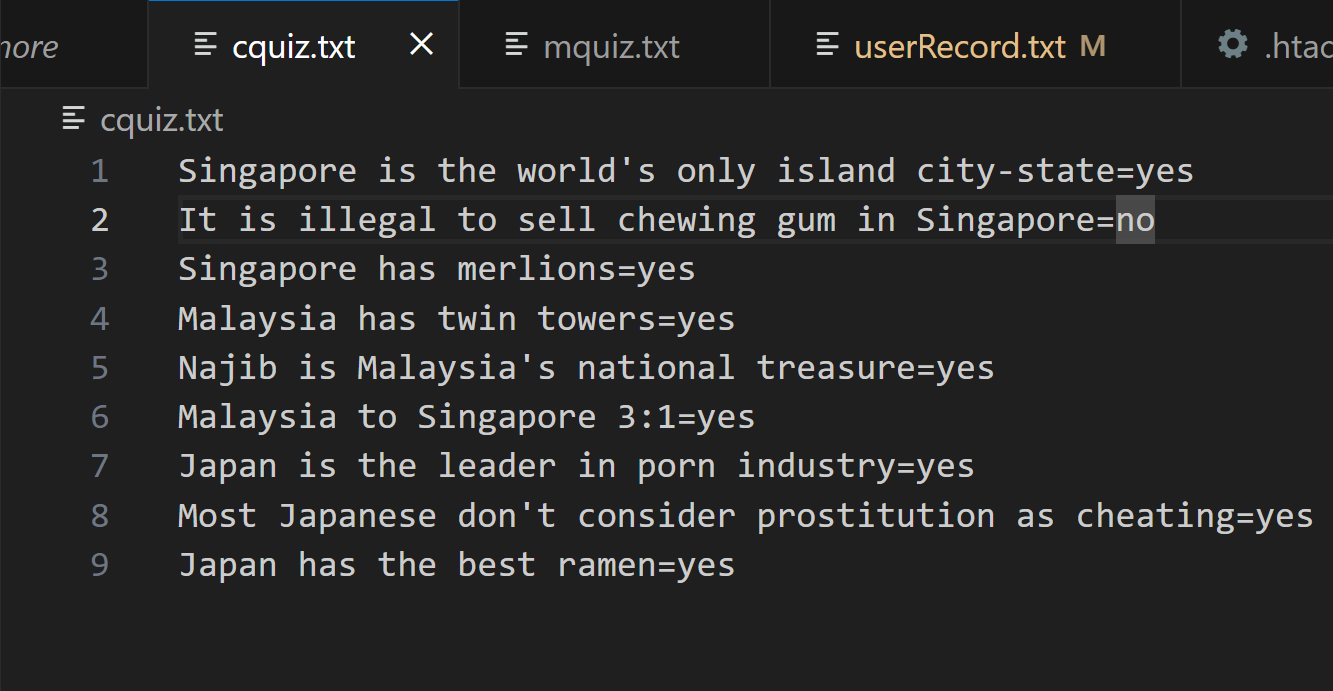
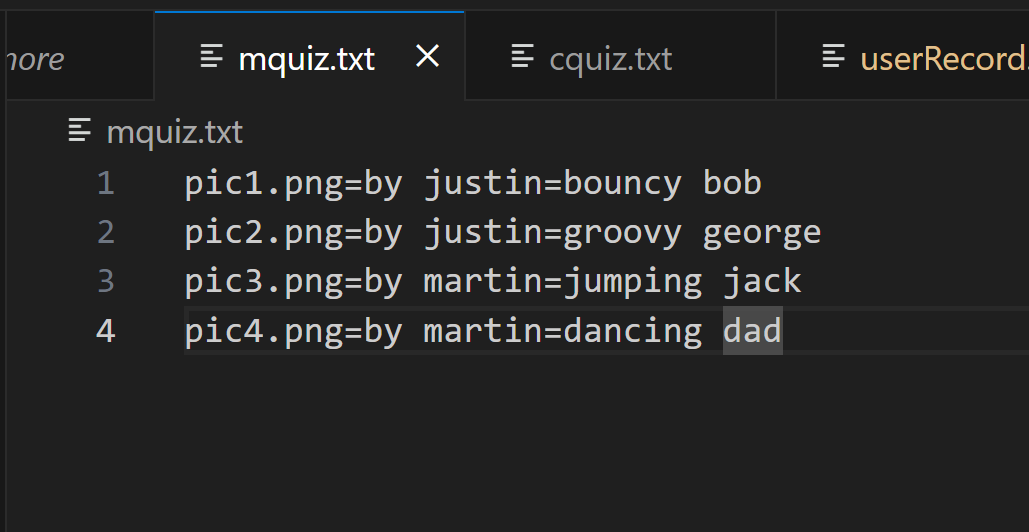


Sort by name



## “Database” files

There are three .txt files that act like database as per assignment requirements.

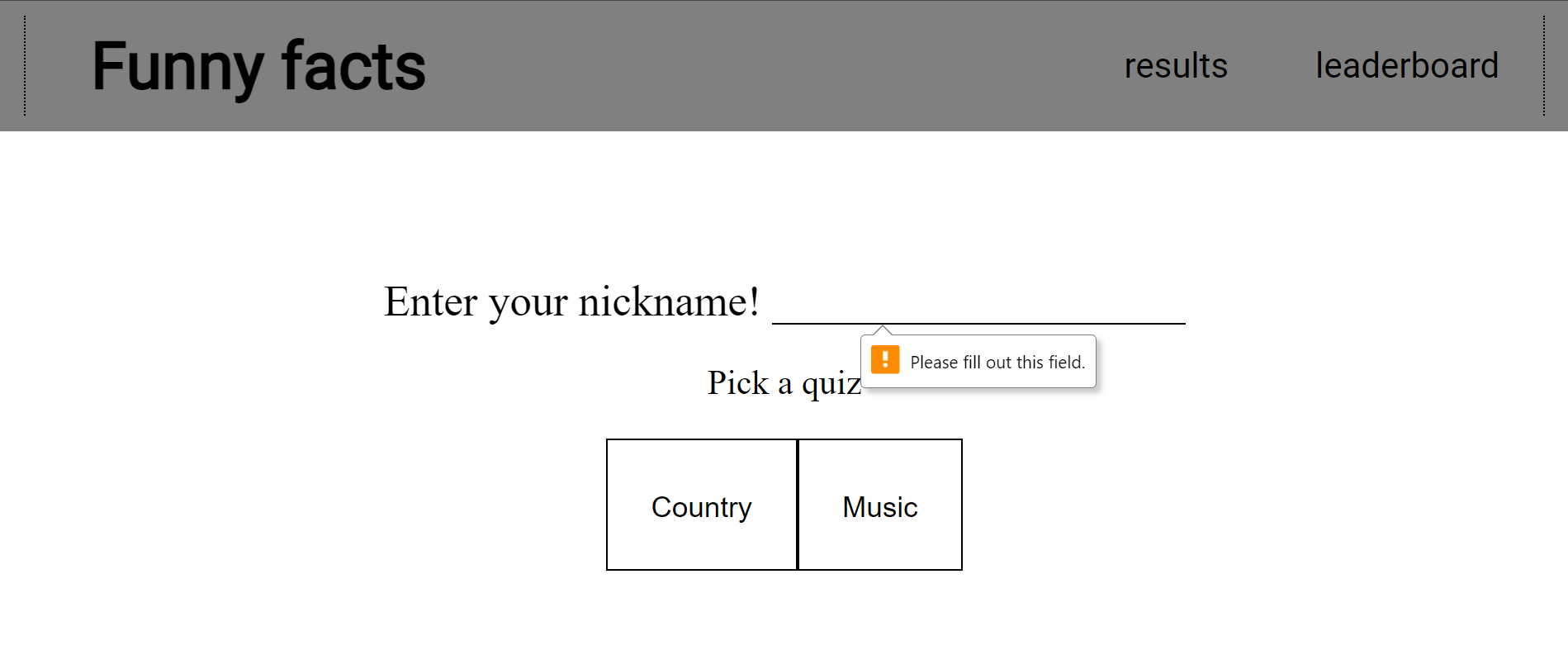
1. userRecord.txt. It holds the user nickname and their accumulated score. Separated by an “=”
   1. 
2. Cquiz.txt. It holds the country facts and true/ false. Separated by an “=”
   1. 
3. Mquiz.txt. It holds the music paths and the correct answer. Separated by an “=”
   1. 

# Extra features

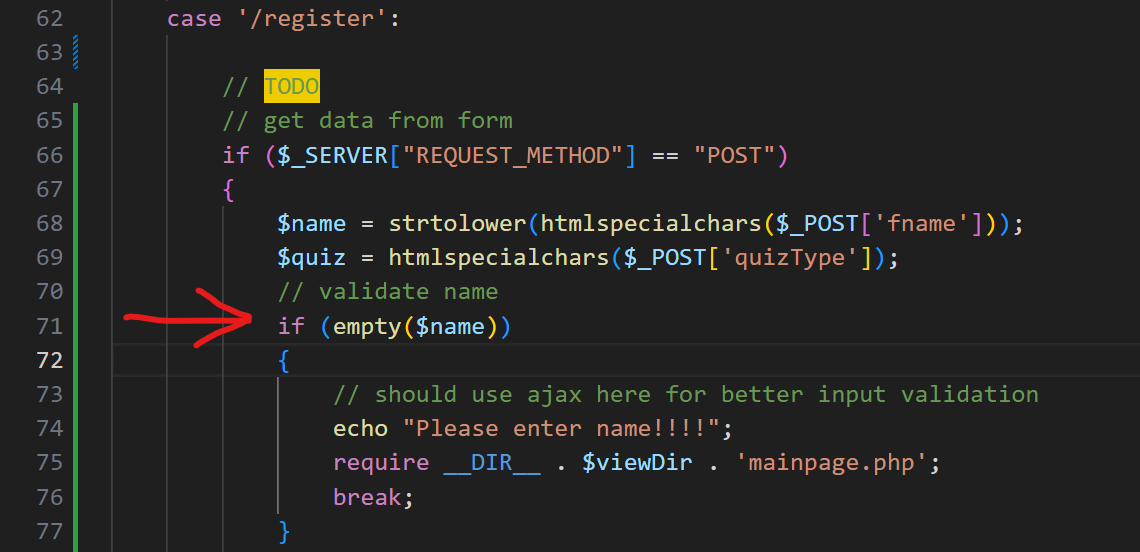
Hopefully these features give me extra marks ☺

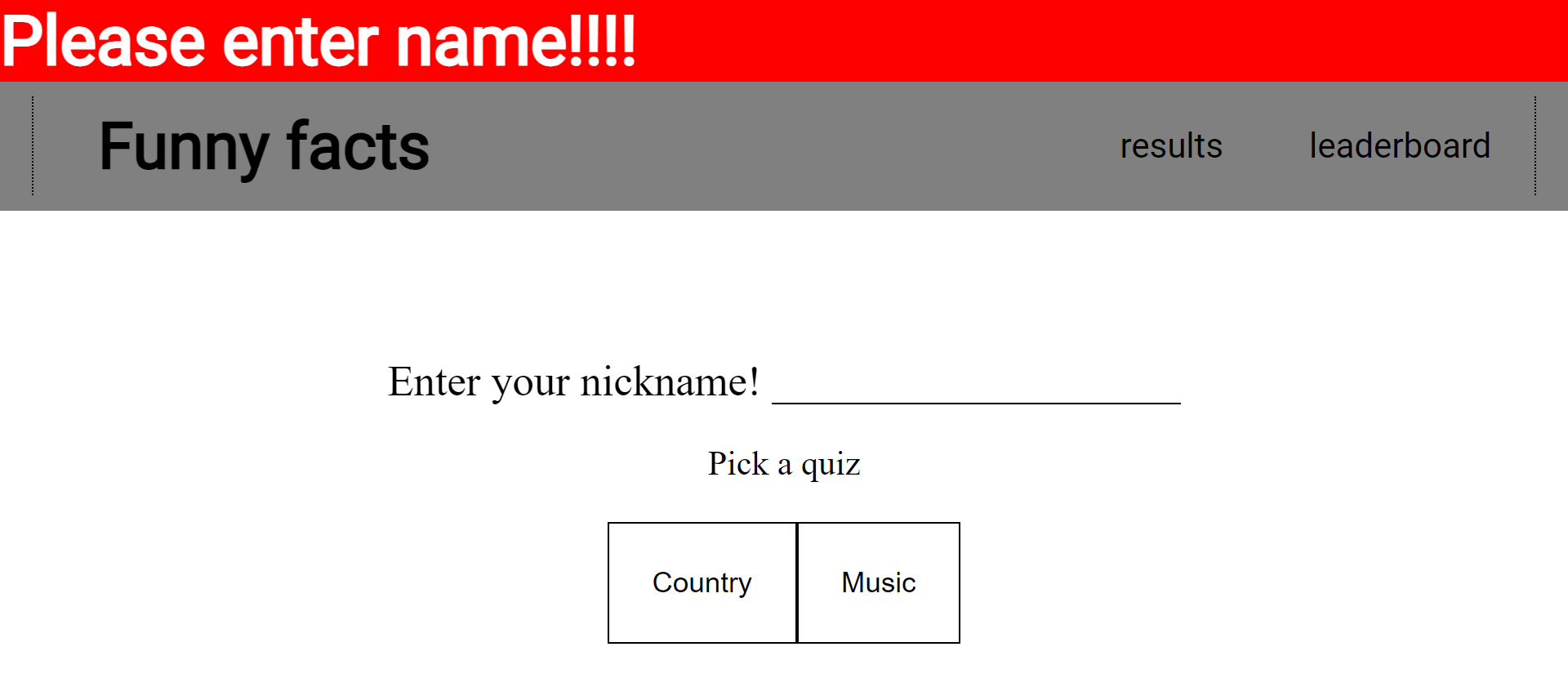
## Server and client side input validation

When user leaves nickname blank but clicks a quiz type, the form will prompt user not to leave the input field blank. This is client side validation through HTML

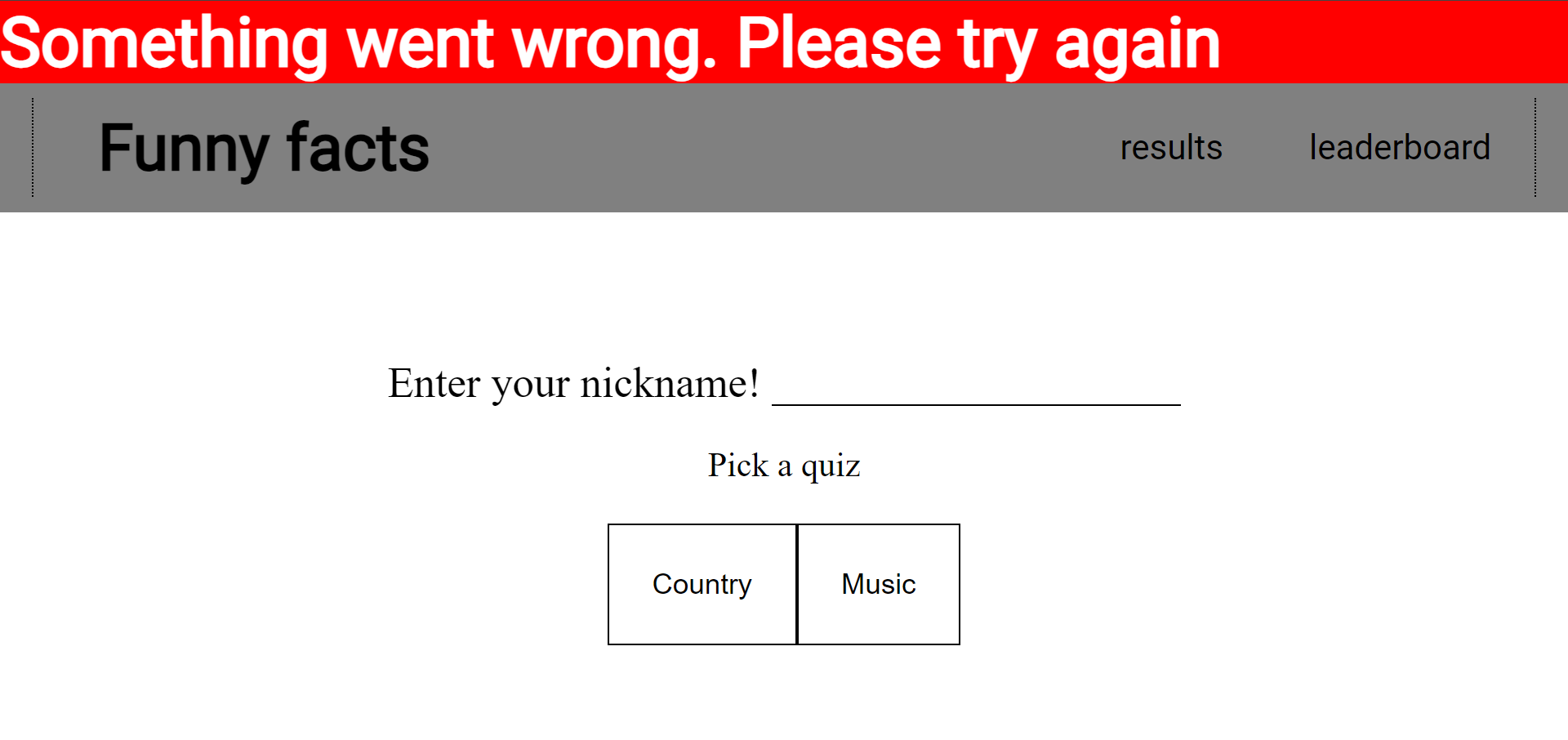


If user somehow manipulates the post request for nickname to be blank, the server will validate and return a prompt.





Similarly, if user somehow manipulates the quiz type buttons’ values, the server will validate and return a prompt.



All webpages that require user to be logged on have validation. The server checks if the user session if created. If not, means the user has not logged in, and the page redirects back to /.

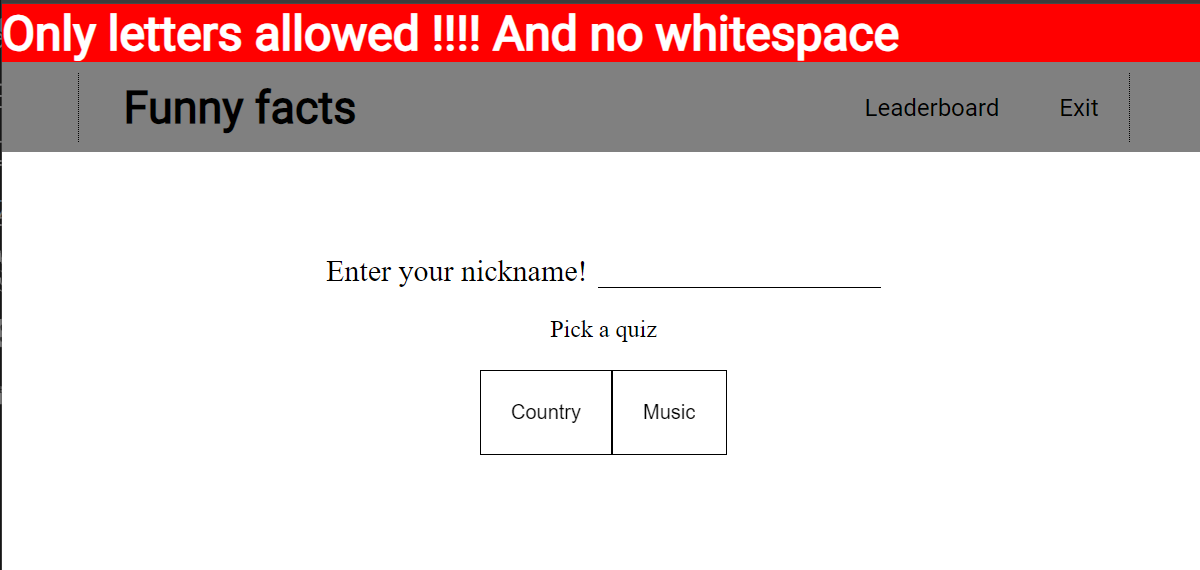
## URL whitelisting

The user can only access the 10 routes specified in the router. Each route is validated if user log in is required. This feature also means that file traversal attacks are not possible

## Username whitelisting

The user input field for nickname is restricted to letters only and no whitespace. This is to protect against file traversal attacks. This validation is performed in the server.





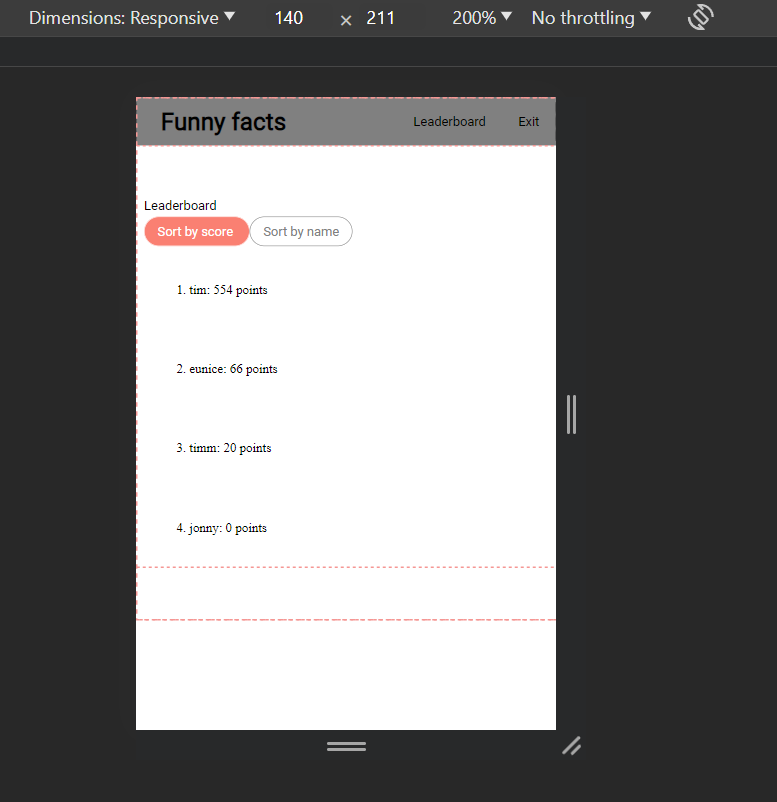
## Mobile friendly

The website can accommodate screen widths of at least 140px which should be sufficient for all mobile devices.

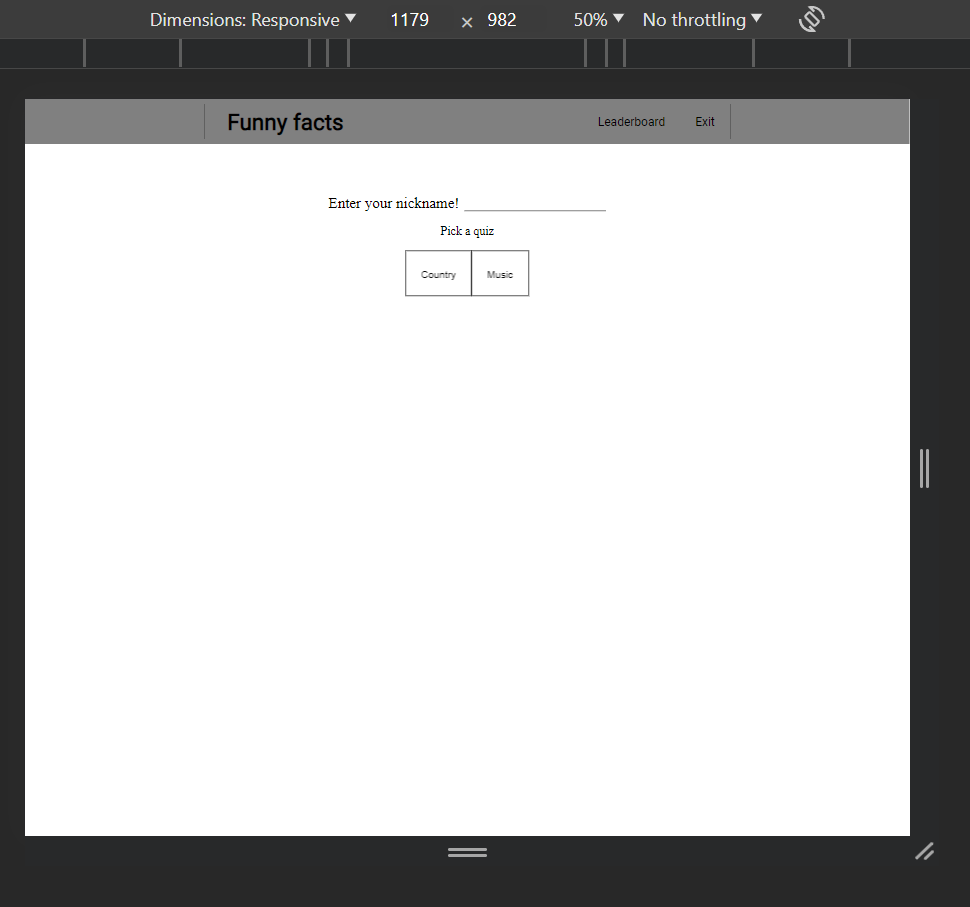
## Modern full stack design

The system design of the app is designed coming from my experience as MERN (MongoDB, Express.js, React.js, Node.js) as my preferred stack. Hence There is an API router, “react” components. The front end talks to middleman to talk to “database”. Frontend does not talk directly to “database”.

This style means that integration with a MongoDB database would not be too difficult since MongoDB strings are just JSON strings which are essentially just lines of text. Should be okay with other NoSQL or even tradition SQL databases.

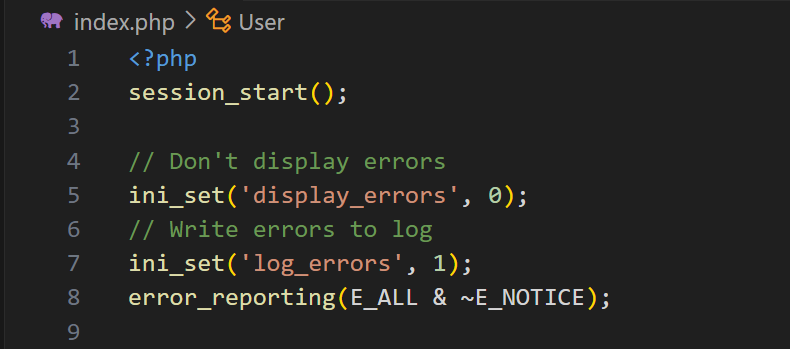


1179px is the screen width of iPhone 15



## Errors are not shown to user

Should there be any 500 errors from server side, the error will not be shown to the user but instead printed in the server side error log



# Limitations

1. Regex string may be system agnostic. But with server side processing, this can be mitigated since the server can be configured to run as a windows machine while the client can use any OS.
2. There is quite a lot of repeated code and logic between the /submitmusic and /submitcountry routes. Perhaps I could have found a way to merge them but I would have nested switch-case statement which I find not very clean. Perhaps there is a way to merge these two routes while still maintain clean, easy to read code.
3. Hover effects may look nice on desktop devices however on touchscreen devices, the hover elements do not work as intended. This is not so much of a limitation of my particular app but rather a universal limitation of css hover effect
4. Opening the web app in multiple browser windows at the same time would mess up the functionality because the server would create and overwrite the same session names