Website
Reflection Report

Advanced Web Technologies

SET09103

https://webtech-2324-15.napier.ac.uk/

https://github.com/Andy-Dickinson/advanced_web_tech_server

Andrew Taison
Matriculation: 40538519
Word | Page count: 1538 | 5
(excl. references and appendices)
Submission date: 29rd November 2023

Table of Contents

Appendices

Introduction and Description

Building on the conceptual work laid out in part one, this report aims to reflect upon the implementation and practical side of things. The project is centred around creating a server-side webapp using Python Flask microframework. With a passion for golf, the design centred around building a social website, bringing golfers together to organise games and chat about the finer details.

The website can be found at:

https://webtech-2324-15.napier.ac.uk/

Implementation Design Differences

Overall, of the features which were implemented, most are very close to the initial design plan (with the exception of those discussed below.). Of the features which were planned, but not implemented, these are discussed in the 'potential site improvements' section below.

The full list of implemented features in can be found in appendix A

Implementation has taken longer than the initial expected timeframe. As a result, certain features couldn't be integrated as planned. To address this, some routes have been included for demonstration and testing purposes:

1. '/add_multi_clubs' - adds 10 clubs
2. '/add_one_club' – adds 1 club
3. '/delete_clubs' – deletes all clubs
4. '/add_events' – adds 6 events, must be called after adding multi clubs
5. '/delete_events' – deletes events
6. '/print_tables' – prints all database tables to terminal
7. '/clear_session'
8. '/clear_request_data'

It's important to note these should be removed for a production environment. Due to time constraints, the creation of templates necessary to add and approve clubs and events was not feasible, so these routes have been used as a temporary solution.

Login manager [1] is used to keep track of user's sessions and authentication status. This also redirects users when trying to access a 'login required' route back to the base route.

User input from modals (sign up / log in / handicap change) are verified client side for a more restful approach, but also verified server side for security.

The colour scheme planned was modified as the original was found to clash a little and look messy. The planned 60-30-10 split has still been applied to allow for a cleaner appearance and better user experience.

SQLAlchemy [2] only accepts Python datetime objects, therefore the planned usage of ISO formatting was not feasible. Currently timestamps are stored as local time. This could be converted to UTC to be universal, if the website was used across different countries and time zones. This was attempted, however it proved to be problematic, so the idea was abandoned and local time was used. The decision to use SQLAlchemy over SQLite as outlined in the design report, is worth reemphasising for clarity:
Whilst SQLAlchemy may introduce some performance overhead, it significantly makes life easier for development, especially with shorter development timeframes. SQLAlchemy provides ORMs (Object-relational-mappers) which enables work with the database using Python objects and classes.

The database tables are slightly different to design, messages and their timestamps have been moved to a new table 'Message'. The chat table is used as a junction table between users, events and messages. An updated ER diagram can be seen in appendix B

The clubs listed in the 'manage subscriptions' modal are listed alphabetically and events listed on 'find a game' template are listed by nearest date first. This allows for items to be located more intuitively.
Allowing events to have an optional handicap min and / or max access was planned as an additional feature. However this was found to be relatively straightforward to implement during development of the 'find a game' template and '/add_user_chat' route and so was included. If the user attempts to join a

game which has a handicap requirement and they either do not have one or it is outside of the range, then they will be denied access, and a message will be flashed to the user.

Potential Site Improvements

The suggested improvements below have been separated into planned and suggested lists for clarity. The planned items were regrettably not implemented due to time restrictions.

Planned, but not implemented:

- User ability to create event / game – button setup, currently placeholder.
- Allow user to add new club – link setup, currently placeholder.
- Admin – method to add admin user, and admin related pages to allow them to approve new events and clubs. Admin Boolean setup in user table.
- Option to leave chat – if creator leaves, a new event leader should be designated.
- Option to delete chat (if creator / leader.).
- Display link to club along with other event details on ‘my game chats’ page at top of message window when chat loaded, club URL is stored in database.
- Allow user to filter events listed on ‘my game chats’ page similar to ‘find a game’ page (by date / club / subscribed.).

Suggested:

- Option to delete account.
- Option to change password.
- ‘My game chats’ page would require a completely different setup for mobile. Event chat list should be on one page and chat window on another, or events in a dropdown menu. The site has been setup to be responsive, which works well for the majority of the site (the exception being ‘my game chats’.). Currently the chat list is put at the top of the page (which could be a fairly long list) and the chat window below the list when viewed on mobile.

- Check box on ‘find a game’ page to hide events which user has already joined.
- Sitemap.
- 404 route - most likely leading to sitemap.
- Timeout after password wrongly input multiple times.
- Allow user to sign in with email as well as username (the current option).
- Verify new account with email verification.
- Event creator to have ability to close chat / event to new participants along with edit event settings (name / description etc.).
- Notification badges for chats with unread messages.
- End-to-end encryption for chat messages.

Reflection

Many aspects have been successfully implemented, closely resembling the initial design. The full list of implemented features can be found in appendix A. Overall the project has been a success, mainly due to a lot of planning during the design process. Like with most projects, the to do list can seem never ending, and this project has been no different. It was the intention to have completed more, however time was a major issue.

The biggest issue that presented itself on multiple occasions was the difficulties of learning whilst doing, along with bug fixing adding additional time. It is believed that this is largely due to this only being the second web project ever designed and implemented, and also the first server side (the learning curve is steep.). The naivety of such tools did present problems, for example, it wasn’t initially obvious that Jinja parameters are for use server side only, and an alternative solution was required for using parameters in client side JavaScript. Some solutions included storing data in hidden elements or class names for a more restful approach. When data security was a concern, async functions have been used to obtain the information from the server. The fact that flask forms has built in verification was only discovered far too late to implement, and due to the lack of this knowledge, a lot of the verification is done with RegEx (although this is an achievement in itself.).

The site presents a nice user experience and seems to be intuitive to use and navigate (based on feedback from those asked to test the site.). To be fully production ready, it would be sensible to complete at least the planned implementation list, and probably the first few items of the site improvement suggestions. It should also be mentioned that the images have been sourced from copyright free websites [4,5].

A few notable successful features are as follows:

- On the ‘find a game’ page, the filtration of events by dates works very well, and displays by nearest first. If the user has subscriptions, they will be presented with events related to those on page load. If they have none, then all clubs will be the initial display. The date selection boxes affect each other and make unavailable dates greyed out (i.e. dates prior to today. Also if a start date is selected, then the earliest date available on end date will be that selected and vice versa.).
- The ‘my game chats’ page uses real time chat utilising SocketIO [3], and positions each message based on if the user is the owner. Usernames are only displayed if the user is not the owner. On initial load of a conversation, the input bar is brought into focus, avoiding the need for the user to click on the bar to enter a message. After a message is sent, the input bar is cleared ready for the next message. If the user visits the page via joining a new event, the relevant event conversation will be loaded automatically.
- The modals present a nice way for the users to log in, signup as well as update their handicap and manage subscriptions (which contains some nice toggle switches.). The use of models provides a better user experience than taking the user to a new page for these common tasks. To ensure update handicap and manage subscription modals cannot be accessed when not authenticated, the scripts are not present on the base template. This is achieved by swapping the scripts in the (logged in) home template.

References

[1] P. Gedam, "Flask-Login," *Read the Docs*. Online. Available: https://flask-login.readthedocs.io/en/latest/

[2] “Flask SQL Alchemy,” *Pallets Projects.* Online. Available: https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/

[3] M. Grinberg, “Flask-SocketIO,” *Read the Docs.* Online. Available: https://flask-socketio.readthedocs.io/en/latest/

[4] “Unsplash,” Online. Available: https://unsplash.com/

[5] “Pixabay,” Online. Available: https://pixabay.com/

Appendix

Appendix A – Full List of Implemented Features

Below is a list of the successfully implemented features along with a short description and relevant information.

- Readable text – Allows users to easily scan and read content.
- Logo – For brand recognition and for visitors to be able to click from any page and return to the home page for easy navigation. This is also used in the favicon. On the home pages, the logo takes the user to the top of the page when clicked.
- Strong calls-to-action – To help guide visitors what to do next.
- Simple, but contrasting colour scheme – To assist readability of website content, make elements stand out, and provide a good user experience.
- Clearly labelled links – To aid navigation and help direct users.
- Feedback provided to the user – Feedback is presented to the user by use of and flash messages combined with bootstrap alerts. The function used to produce these has two possible heights for the alert bar (a smaller one used for when modals are displayed, and a taller one otherwise.).
- User accounts – Users are only able to access the public base page containing log in and sign up modals unless authenticated. If an unauthorised user attempts to access a page requiring authentication, they are redirected to the base page.
- Use of templates – For reusability and ease of future development and maintenance.
- Navigation bar – To provide easy access to common pages. The navigation bar changes depending on the users authenticated status.
- Footer – Contains copyright information and contact information.
- Bootstrap modal pop up windows – Called using JS functions from account dropdown menu and used for common forms (log in, sign up, manage subscriptions, update handicap.).

- Validation client side and server side – client side for a more restful approach and server side for security. This is mostly carried out using RegEx and enforces the following:
    - Username: between 4 and 25 characters, only letters and or numbers.
    - Password: minimum 8 characters, at least 1 uppercase letter, lowercase letter, digit and special character. Password must match confirmation password.
    - First name, surname and handicap are optional.
    - When first name and or surname provided: must be 100 characters or less, and only contain letters, hyphens and spaces.
    - When handicap is provided: handicap must be between +10 and 54. Can be provided as an integer or a float to 1 decimal place. For non golfers reading this, handicaps beginning with '+' are better than a 0 handicap or handicap with no leading '+', and shots are added to their final score (as opposed to taken off for regular handicaps without a leading '+').
    - Email: must be in a format resembling an email address and no longer than 255 characters.
- Username and email checked if exists – When signing up a new user, the username and email are checked to ensure there is not an account with those details already.
- Use of a database – To store information regarding each user along with golf club information, events listed, and chat messages.
- Hashed passwords – Using Bcrypt. Passwords should never be stored as a raw string. This adds additional security.
- Dropdown account menu – Located in the navigation bar. Displays the username at the top of the dropdown along with options to manage subscriptions, update handicap and sign out.
- Option to change subscriptions – Clubs in the database are listed in alphabetical order, each is shown with its own toggle switch. There is also a select all button at the top for ease. The window is given a scroll bar when it overflows. The modal closes and the nav bar (if expanded for a smaller screen) collapses when the options are saved.

- Option to change handicap – A users handicap will change regularly, they should be able to update it easily. This can be used as an optional parameter when creating events (in so allowing a minimum or maximum handicap for admittance to the event.). The current and updated handicap is displayed to the user in the modal.
- Multiple ways to access pages – Users can access pages either via the nav bar options or by clicking on the images related to the page.
- Appropriate events listed – Events listed on the 'Find a game' page are appropriate to the users selections (date choices, subscribed clubs or selected club.). Events displayed are only those which are 'open' – i.e. the event time and date have not passed and they are not at full capacity. When the user first visits the page, the display option is set to 'subscribed clubs' if the user has any subscriptions, otherwise it is set to 'all clubs'. This also changes if the user is on the page and changes their subscriptions.
- Event access checks – When the user clicks to join an event, the event is first checked to ensure its status hasn't changed since page load (deleted, full capacity, closed.). Following this, the users handicap is checked against any restriction on the event. The user is also checked to see if they are already a member. The user is denied access if any of these fails.
- Chat load upon join – When the user joins a new event, the user is redirected to the 'my game chats' page where the chat for that new event is automatically loaded. If the user visits the page from another route, then no chat is loaded until one is selected.
- Current chat feedback – The current chat displayed (when one is loaded) is feedback to the user by means of adding a background colour to the relevant event and making the font weight bold.
- Real time chat – SocketIO is utilised and allows messages to be sent to all users in the relevant event. When the message is sent to the server and successfully added to the database, it is then emitted to all the users in the 'room'.
- User personalised display of messages – Messages in the message window on the 'my game chats' page are positioned based on if the user is the owner (left for another user, right for user owned.). The username is only displayed if not the users own message.

- Focused input bar – On initial load of a conversation on ‘my game chats’ page, the input bar is brought into focus, avoiding the need for the user to click on the bar to enter a message. After a message is sent, the input bar is cleared ready for the next.
- Chat history – Past messages are able to be viewed in the message window on the ‘my game chats’ page.
- Overflow handling – both the event list and message window are set to have a scroll bar added if either of them overflow. The event list remains at the top, whereas the message window will scroll to the bottom on load. If the user scrolls up to see the past messages, the input bar will remain in view.
- Ability to sign out – Users can sign out from their account and in doing so are redirected to the base home page.

Appendix B – Updated ER Diagram

Figure 1 below shows the updated ER Diagram as it has been implemented.

Figure 1. Updated ER Diagram as Implemented