Pokemon Team Builder

Gregor Bell
40200288@live.napier.ac.uk
Edinburgh Napier University - Advanced Web Technologies (SET09103)

Abstract

The aim of this coursework was to create a web-app that demonstrates our understanding of the Python Flask micro-framework. We were given total control over the subject of the web-app. This report details the core features of the web-app as well as the URL hierarchy of the site. The report also identifies enhancements that could be made to improve the web-app and evaluates elements of the web-app that work well and elements that do not.

Keywords – SET09103, Advanced, Web, Technologies, Edinburgh, Napier, University, Pokemon, Team, Builder, Report

1 Introduction

In the Pokemon series of video games, trainers can capture and battle with creatures called Pokemon. Trainers can have up to 6 Pokemon in their party at a given time. The purpose of this web-app is to provide the user with the ability to create teams of 6 Pokemon and save them for later viewing.

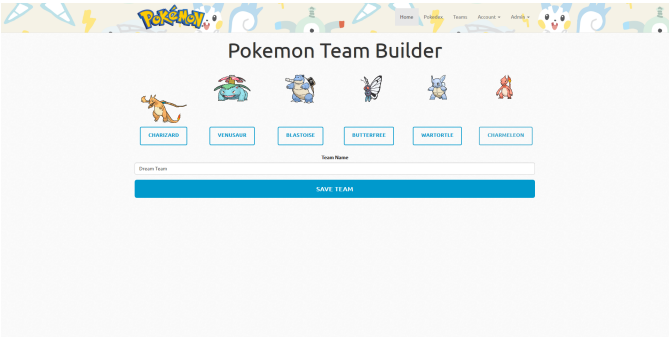

Figure 1: **Main Screen** - The main team builder screen of the web-app

A comments section has also been implemented for each team generated by the web-app, allowing discussion surrounding the created teams to thrive.

A Pokedex feature has been integrated into the web-app to allow users to view information about each Pokemon.

The ongoing goal of this project is to create a one-stop shop for avid Pokemon fans to strategise and prepare so that they can equip themselves with the best team possible, as well as discuss their creations with like-minded individuals.

2 Design

I have created the following figure to illustrate the URL hierarchy that has been utilised in this web-app:

Figure 2: **URL Hierarchy** - A horizontal hierarchy showing the map of URLs used in the web-app

2.1 /

The sole purpose of this page is to redirect the user to /home/, the main page of the web-app.

2.2 /home/

The primary page of the web-app, and the page that user's will see when they first navigate to the website. This page contains the team building function of the app. Users can click on any of the Pokemon buttons to launch a Bootstrap modal. This modal contains a list of all Pokemon currently available in the database, as well as a search bar to aid the user's traversal of the list. Users are not required to enter the full name of their desired Pokemon, as the search function can handle fragments of the name. There are also no restrictions on case, which makes the search experience more intuitive for the user.

After deciding which Pokemon they would like to incorporate into their team, users can click the name of the respective Pokemon. The Modal then closes and the relevant button changes to display the name of their Pokemon. An animated gif of the Pokemon also appears above the button.

The button functionality has been implemented using a mix of Flask and jQuery so that these steps do not require the user's browser to refresh. Users can then enter a name for their team and click the "Save Team" button to add their team to the database.

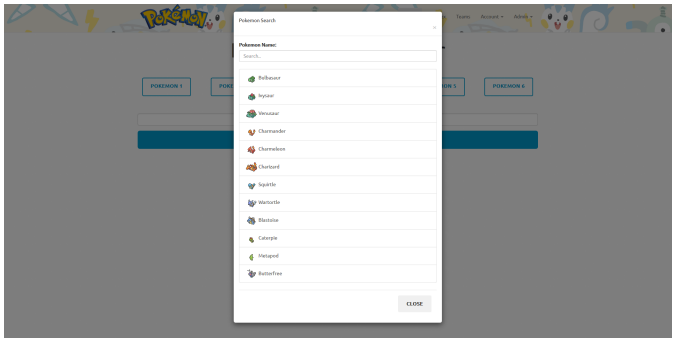

Figure 3: **Pokemon Search Modal** - Bootstrap modal used to select which Pokemon to add to the team

2.3 /user/user_name/

This page displays a list of the teams created by the user who is currently logged in. The user can then click on each team name and be redirected to the relevant /teams/team_name/ page.

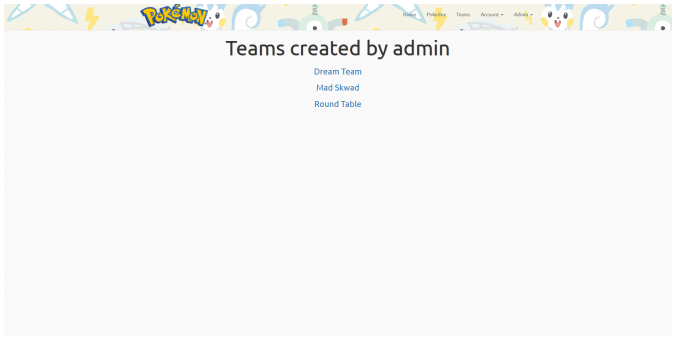

Figure 4: **List of User's Teams** - A list of the teams created by the currently logged in user

2.4 /teams/

/teams/ provides a list of all the teams that have been created and stored in the database. Each team name acts as a link to the respective team. This page serves as a simple method for users to browse various Pokemon teams. It has been added to the navbar to make access to the page quick and easy to find.

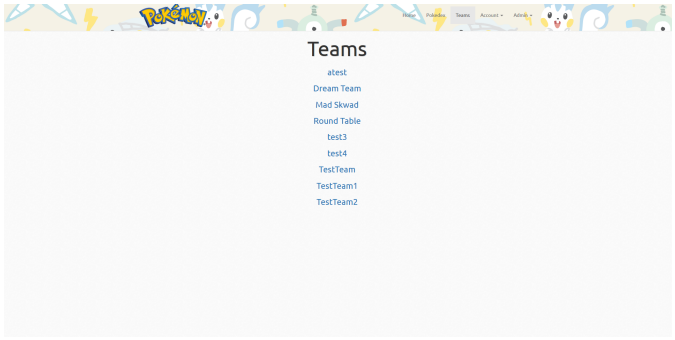

Figure 5: **List of Teams** - A list of all the teams stored in the database

2.5 /teams/team_name/

This section of the web-app displays the stored team, where team_name is equal to the name given to the team by the original author. Any user can view a created team. There is also a unique comments section for each team, enabling users to critique and divulge their opinion on the team in question.

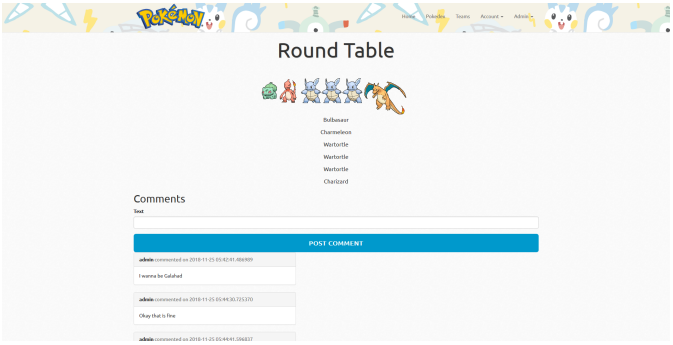

Figure 6: **Team Details** - Details of the selected team, includes a comments section

2.6 /pokedex/

In the Pokemon video games, the Pokedex utility allows trainers to view a list of Pokemon they have discovered. Similarly, the /pokedex/ route of the web-app presents the user with a list of all the Pokemon stored in the database. Users can then click on the Pokemon name to be redirected to the relevant /pokedex/pokemon_name/ page.

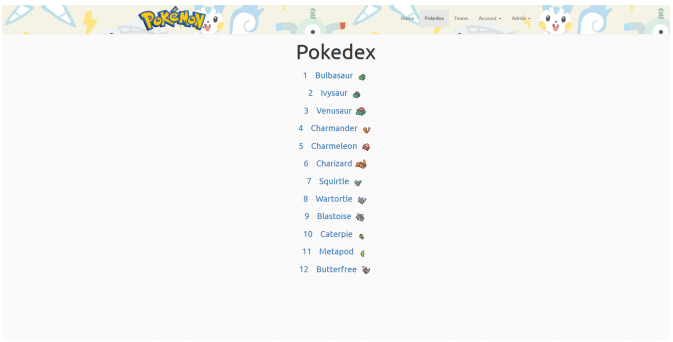

Figure 7: **Pokedex** - List of all the Pokemon in the database

2.7 /pokedex/pokemon_name/

This page displays a .gif of the respective Pokemon as well as information about it's typing, abilities, evolution and in-game Pokedex entry.

2.8 /register/

The /register/ route serves the user with a registration form. Users can use this form to create an account on the web-app and store their details in the database. Creating an account is advisable as, although users can create teams and view other teams while not signed in, they are unable to save a new team or post a comment until they register an account.

The only details required from the user are a username, password and email address. Any other details would not be used

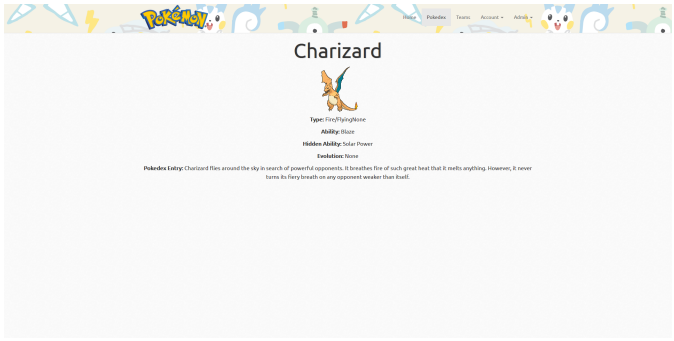

Figure 8: **Pokemon Details** - Details of the selected Pokemon

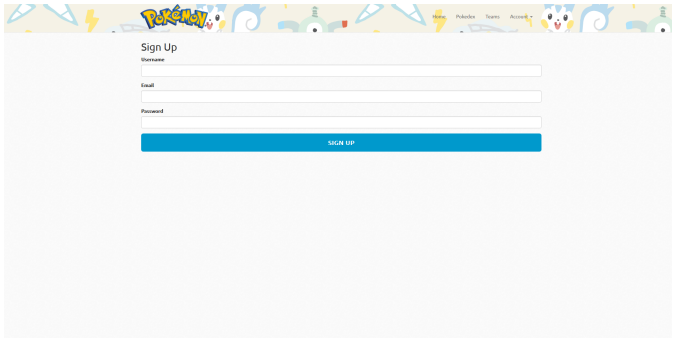

Figure 9: **Registration Form** - Form for registering an account

and storing unnecessary data is not advisable under the General Data Protection Regulation (GDPR) as we are "unlikely to have a lawful basis for retention" [1].

Passwords are encrypted using the SHA 256 hashing algorithm, imported from the werkzeug.security Flask package. This algorithm randomly generates an 80 character hash. The generated hash is then stored in the database, rather than storing the plain text password. This means that if the web-app's database was compromised, the infiltrator would only have access to the password hash, making it a lot harder to compromise the account itself.

2.9 /login/ and /logout/

After a user has registered an account, they can login to the account at the /login/ route. If the user provides an incorrect username or password, they will be served an error template. Once the user has succesfully logged in, the current user session is saved. This is achieved using the Flask-Login extension. (see Appendix A)

Users can then navigate to the /logout/ route, either manually or by clicking the Logout button in the navbar dropdown menu. The Flask_Login LoginManager will then logout the current user.

2.10 /admin/register/ and /admin/upload/

The /register/ route is a valid option for creating normal user accounts. To create a new administrator account, however, a current administrator will have to navigate to /admin/register/. The admin registration form is very similar to the user form, with the exception of the "Is Administrator?" checkbox. When checked, clicking the "Sign Up" button will add a new administrator account to the database.

I created a form for adding new Pokemon to the database. The form is located at /admin/upload/. This form makes it much simpler and quicker to add a new Pokemon to the web-app, rather than inserting SQL commands straight into the database.

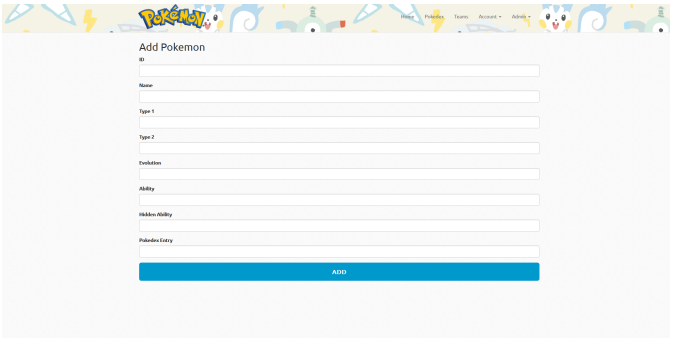

Figure 10: **Upload Form** - Form for uploading a new Pokemon

As both /admin/register/ and admin/upload/ handle sensitive operations, they have been restricted to administrator access only. If a normal user account tries to access either of these pages, they will be served with an "Access Denied" error template.

2.11 /background_process/ and /background_process_2/

/background_process/ and /background_process_2/ are not routes that should be called manually by the user. They exist purely to assist the jQuery functions used in the team builder on the home page. They take in data once the relevant button is clicked and feed back the correct ids for the .gifs as well as the path to where the .gifs are located in the file directory.

3 Enhancements

Although I am pleased with how the web-app turned out, there are a number of features that I would like to add as well as current features that I feel could be polished further.

3.1 Dynamic Type Chart

The main feature that I would like to add to the web-app is a Pokemon Type Chart.

My vision for this type chart is that as each Pokemon is added to the team, the type chart would dynamically update to reflect the current strengths and weaknesses of the team. This would make the team builder much more appealing to Pokemon fans, as it would allow them to form balanced teams much more effectively.

This type chart was part of my original vision for this coursework, however I ran out of time before I was able to implement it. I do not think it would be terribly difficult to program and it would take the team builder to the next level.

3.2 Comments Section Improvements

The current implementation of the comments section is functional but very basic. An improvement that I would like to make would be to enable a reply feature, so that users could reply to specific comments. Their reply would then be nested underneath the original comment. This would result in a tidier comments section and would enable the users to engage with each other to a greater degree.

I would also like to add a points system, so that users could upvote and downvote comments that they like or dislike. This system could also be applied to the generated teams. The /teams/ list could then be sorted by points to display the most popular teams. This would be an interesting way to determine what Pokemon people enjoy using the most, or what team proves most effective.

Another feature that will likely become a necessity if the user base of the web-app was to grow would be the ability to remove/moderate comments. This ability could be given to the administrator accounts.

3.3 General Code Improvements

Although I have managed to resolve the majority of errors that the web-app throws, there are some that I have not found a solution for yet. I am using the Flask-SQLAlchemy (see Appendix B) extension to assist with implementing the database in the application and I have declared a number of fields in the database to be unique. This is necessary as, for instance, we cannot have two teams with the same name as that would lead to issues. If another team is created with the same name as a previously generated team, the web-app does not create the team and throws an error. I have not implemented an error template for these errors yet so the errors given are the plain Flask errors.

I am also using the Flask-WTF extension (see Appendix C) to handle forms in the web-app. I am currently creating all the database models and form classes in the same file as my routes. The program file would be less convoluted and easier to understand if I created a models.py file for the database models and a forms.py file for the form classes.

4 Critical Evaluation

I am happy with the standard to which most of the web-app's features have been implemented, however there are some features that could do with some improvement.

4.1 Flask With jQuery

I was pleased to successfully integrate Flask with jQuery to produce the team builder on the home page. I do not have much experience with jQuery so I had to utilise a tutorial to get this feature working [2]. I was delighted that the page does not have to reload. Single page web design has always been of interest to me and it was interesting to put some aspects of it into practice in this coursework.

4.2 Accounts System

Being able to implement the account system was another development during this coursework that I was pleased with. This was one of the enhancements that I had noted in my previous coursework and I feel that it provides a great benefit to the functionality of the application.

Implementing both standard user accounts as well as administrator accounts enabled me to restrict certain aspects of the system to administrator accounts only. This is a boon for the security of the system and means that the people who wield the power to interact with the database itself are trustworthy.

4.3 Design

An area where improvements can be made is the design of the web-app. The main issue I have with the design of the app is the that the gifs on the home page are all different sizes. This is not an issue if the user selects Pokemon who are all of similar stature, however, if the user picks both a large and small Pokemon, the smaller monster will appear to be floating.

This could be resolved by setting a maximum size for the gif container and scaling the gif to fit. An issue with this method, however, is that the gifs for smaller Pokemon will have to be scaled up to fit the container and will appear fuzzy as a result.

Another issue with the design of the system is the design of the /pokedex/pokemon_name/ screen. The layout is bland and visually unappealing. The page could benefit from a larger image of the Pokemon and to have the data sorted into columns. I would also like to include additional information about each Pokemon so that this section of the website can become a truly helpful resource to users.

5 Personal Evaluation

This coursework was enjoyable to work on due to the total creative control we were given over the nature of our web-app. I was able to select a topic that I am passion about and as a result, I found myself having fun while working. Putting together each piece of the project was immensely satisfying and I found myself thinking about what I wanted to work on next even when I wasn't at my computer.

After researching other team builders, I feel that, with a bit of polish, my web-app could be a valuable utility in the competitive Pokemon scene and over Christmas I will look into fully deploying the app.

My proficiency with Python and Flask has gone from zero to fairly confident while working on this coursework and now feel comfortable taking on other projects with these tools. Implementing jQuery has made it clear, however, that I am not as well-versed in JavaScript as I should be if I wish to pursue a career in the field of Web Development.

For future projects I will start working as soon as possible, rather than once the deadline begins to loom. I feel that this extra time would be the deciding factor in levelling a passable project into an impressive one.

6 Conclusion

I am satisfied with how my web-app turned out, although I do feel that improvements could be made. My Python and Flask skills have improved tremendously and I look forward to utilising these in my future career.

References

[1] Information Commissioner's Office, *Guide to the General Data Protection Regulation (GDPR)*. 2018.

[2] sentdex, "jquery with flask tutorial," 2015. [Online; accessed 24-November-2018].

A Flask-Login Extension

This extension is used for managing login states. It enables users to log in and traverse the web-app while the web-app remembers the current user.

You can install Flask-Login using the below command:

pip install –user flask-login

B Flask-SQLAlchemy Extension

This extension is used to enable the web-app to manage a database using classes, objects and methods rather than running SQL queries.

You can install Flask-SQLAlchemy using the below command:

pip install –user flask-sqlalchemy

C Flask-WTF Extension

This extension is used to let me create forms inside the python app, rather than creating them individually in the HTML.

You can install Flask-SQLAlchemy using the below command:

pip install –user flask-wtf

D Flask-Bootstrap

This extension is used to make use of the Bootstrap classes, in order to make the web-app more aesthetically pleasing.

You can install Flask-Bootstrap using the below command:

pip install –user flask-bootstrap