Advanced Web Technologies CourseWork 2

Sonas MacRae
40277542@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET09103)

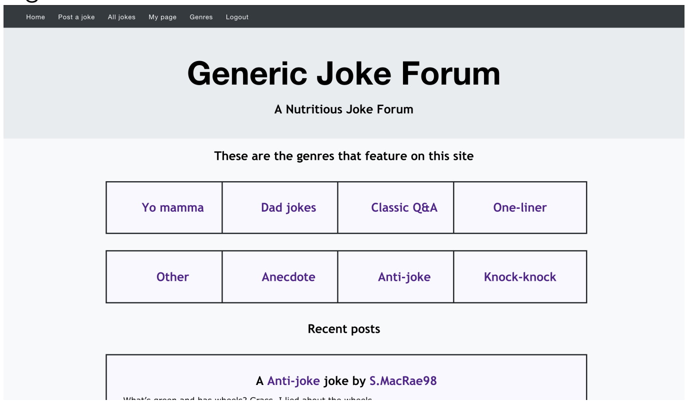
1 Title

Generic Joke Forum

2 Introduction

This web app is a forum, the purpose of this forum is to allow users to share jokes with one another. It is a fairly basic concept, a user creates an account, which allows access to the content of the website, where they will be able to make new posts, in this case the post will be in the form of a joke where the user is able to specify the type of joke in which they are posting. In addition to this, users will be able to update their posts as well as delete them altogether. Each post hosts its own comments page, where any user can post comments to and users will have the option to update or delete their own comments. There is more functionality with filtering jokes, users can filter jokes by genre and by users, each user has their own profile page which displays all of the posts in which they have made. Figure 1 and Figure 2 below shows the layout of the home page, the design is simple, the green buttons on the posts stand out. These buttons are used to update, delete and comment on posts. The navigation bar at the top of the page (shown at the top of Figure 1) allows users to post a joke, view all of the jokes on the web app, navigate to their profile page, navigate to the page which displays all of the genres as well as log out.

Figure 1:



Generic Joke Forum

A Nutritious Joke Forum

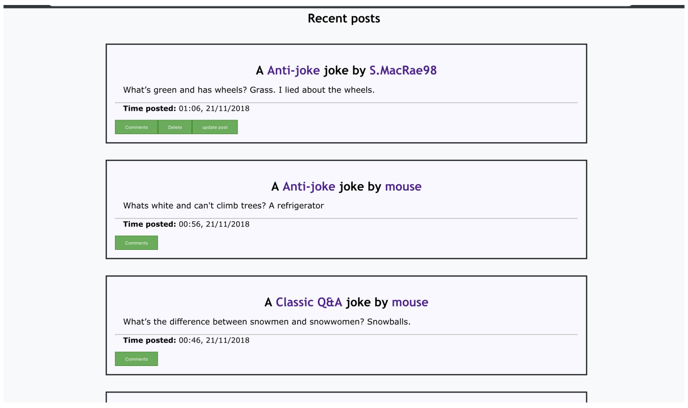
These are the genres that feature on this site

Yo mamma	Dad jokes	Classic Q&A	One-liner
Other	Anecdote	Anti-joke	Knock-knock

Recent posts

A Anti-joke joke by S.MacRae98

Figure 2:


Recent posts

A Anti-joke joke by S.MacRae98
What's green and has wheels? Grass. I lied about the wheels.
Time posted: 01:06, 21/11/2018

A Anti-joke joke by mouse
Whats white and can't climb trees? A refrigerator
Time posted: 00:56, 21/11/2018

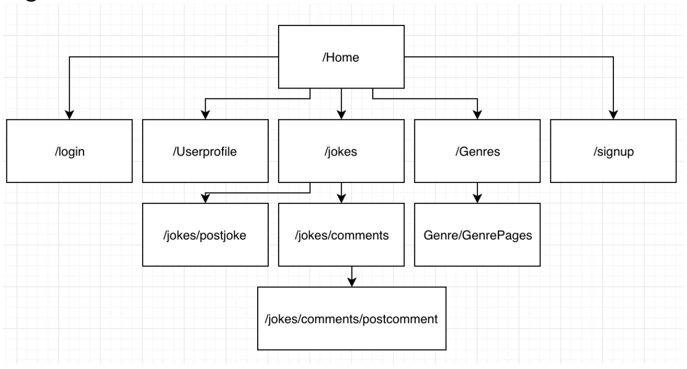
A Classic Q&A joke by mouse
What's the difference between snowmen and snowwomen? Snowballs.
Time posted: 00:46, 21/11/2018

3 Design

3.1 Visual design

The web app is structured in such a way that the user will be able to seamlessly navigate through the pages and will never find themselves at a dead-end. The content is spaced out, this was designed so the user isn't overwhelmed with condensed content, buttons stand out, the green buttons are much more colourful than the rest of the site making them easy to find. colours don't blend together, text is large enough to read without difficulty and the font is standard so that no characters look obscured. The navigation bar, which is situated at the top of the page allows the user to navigate to various landmarks throughout the website, this is easy to see as the colour of the navigation bar is different to the design of the rest of the pages, the same navigation bar appears on every page, this helps to keep the navigation consistent. The URL Hierarchy is shown in Figure 3 below.

Figure 3:

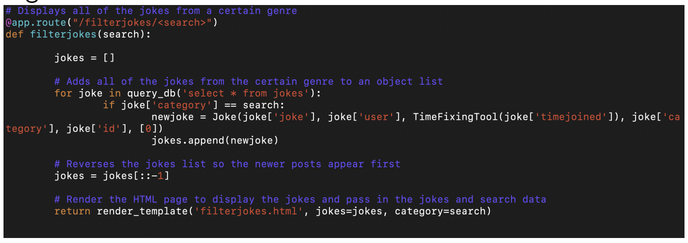


3.2 Code design

The first goal of this project was the build a web app that is capable of scaling well and being able to cope with change without too much difficulty, allowing future updates easy to implement. A database was used to store user data, which hosted three tables, a user table, comments table and a jokes table. This allowed querying, adding, updating and deleting data very easy to use. Setting up this technology took time and research, although it paid off as features were added very quickly one after the other, for example all of the query functions are very similar therefore little to no bugs came up when adding more.

To add to this, functions were carefully planned so that fewer of them had to be written, an example of this is the filtering of jokes function, which can display all of the jokes from any of the genres, this saves having to write a function to display the jokes from each genre. This efficient, resource saving mindset was implemented throughout the project, cutting down on the duplicate code. Figure 3 shows how this is done, this function takes in a search variable from the URL and passes this variable into the function as a parameter, which is then used to query the jokes table of the database. Each post from the jokes table has an ID, the query compares the parameter to each joke's ID in the jokes table, if there is a match then the joke is added to a list. After the table has been searched through the list of jokes is reversed, so that the newer posts would appear at the top of the web apps feed, before the jokes list is passed into a rendered HTML page that displays all of the jokes.

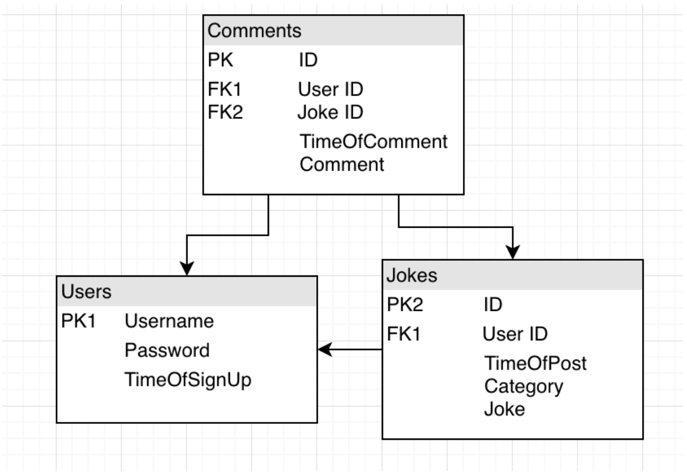
Figure 4:



```python
# Displays all of the jokes from a certain genre
@app.route("/filterjokes/<search>")
def filterjokes(search):

        jokes = []

        # Adds all of the jokes from the certain genre to an object list
        for joke in query_db('select * from jokes'):
                if joke['category'] == search:
                        newjoke = Joke(joke['joke'], joke['user'], TimeFixingTool(joke['timejoined']), joke['ca
tegory'], joke['id'], [0])
                        jokes.append(newjoke)

        # Reverses the jokes list so the newer posts appear first
        jokes = jokes[::-1]

        # Render the HTML page to display the jokes and pass in the jokes and search data
        return render_template('filterjokes.html', jokes=jokes, category=search)
```

This strategy is used throughout the project, for example when displaying comments, each comment in the comments table of the database has its own joke ID, which matches a joke from the jokes table allowing querying to return all of the comments that belong to a certain post. Whereas to display all of the jokes that a user has posted, each joke in the jokes table has a user ID, which matches up to a user name in the user table, this is to keep track of who posted what, a query can return all of the posts that a user has made and this data can be passed into a rendered HTML page to display the jokes on the users profile page. Figure 4 shows how the tables in the database are connected

Figure 5:



4 Enhancements

4.1 Missing features

Features that could have been added include admin privileges, for example an admin would have the authority to delete users accounts, edit and delete any post as well as having access to data such viewing all of the users who are registered on the web app. Another feature which could be added to improve the user experience would be to have user profile pictures, this would make the profile pages more unique and help them to stand out from one another, on top of this users could have their own bio's as well as their profile pages hosting some more information about the user, this would make the profile pages much more personal.

The ability to up-vote or down-vote on each post was initially planned to be implemented, although an efficient and clean method to implement this was not found in time, this would allow jokes to have scores. This concept can be improved further as the highest scoring (worked out when the total number of down-votes are taken away from the total number of up-votes) jokes can be displayed, further even categories could have been created to display the highest scores per day, week, year and even of all time could be shown across all the categories. Each user has the option to interact with the web app, although there isn't an easy way to track all of their contributions, allowing users to see the interactions of one another through the implementation of histories would solve this problem. Although some users may not be happy with the thought of anyone watching what they do, so privacy options would be the next enhancement. Gifting users with the option to choose who sees their liked posts and their profile pages. To make the process of privacy easier, allowing accounts to be linked via a request sent from one account to another would be an opportunity to connect users more easily, a connected accounts list, which would be named as a 'friends list' would allow easy access to the content that the accounts on this list would post. Posting images would greatly improve the content of the web-app, jokes can be in the form of images and some jokes may even require illustration.

A footer could have been added which could contain information about the project, for example the author, name of the project and the purpose of it.

4.2 Features to be improved

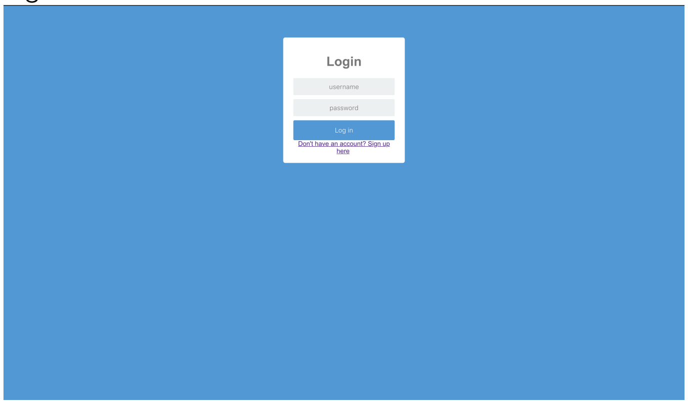
The drop down menu which contains all of the categories, used when posting to the website, could contain more categories, or give the user an option to create their own category. All of the pages could contain more content, this would make the pages look less bare. When a user updates a post or comment, the time of update could be displayed along with a label that states that the post or comment has been updated. Across time zones the times of posts, comments and creation of profiles could be changed to match the time zone.

5 Critical evaluation

5.1 Features that work well

The process of signing up and signing in is easy to use, There was use of session variables, which allowed storing information such as the user in the form of a string, being their user name as well as a Boolean which kept track of whether the user was signed in or not. This was very easy to implement and allowed me to stop people from accessing the web-app without signing in. Logging out is done by clicking the log out button on the navigation bar which is located at the top of the page, which will trigger the 'logged in' Boolean to switch to false and will prevent access back into the forum without signing in again. The sign-in form is shown in the figure below

Figure 6:

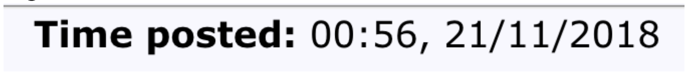


A feature that works excellently is the filtering of the data which is stored, all of the data which is posted to the web app is stored in a database, the data is split between three tables, a table to store the comments, another table which stores all of the user information and the third table stores all of the jokes. Functions were written to filter the data. (image shown below of the code) illustrates this point, the function queries the database and returns all of the posts from the genre specified in the URL which is passed in, "jokes = jokes[::-1]" simply reverses the list of posts so that the newer posts appear at the top of the web apps feed. This works for all of the genres, and in the case that more genres were to be added to the database, the function can return all of the posts within the new genres thanks to the query, this makes this feature very scale-able and as a result makes it a very good feature.

When attempting to redirect to a URL that doesn't exist on the website, a user will be redirected to a 404 error page, where they will be presented with an error message and a button which will redirect them to the home page of the site. On top of this, the navigation bar will be present so the user will have the option to travel to other pages too. This is a good feature to have as it limits the number of bugs someone may run into, and helps to keep the user experience smooth and pleasant.

A small yet useful feature is the use of time throughout the forum, when a new account is created, the time of creation is saved and presented on the accounts profile page. When a new joke is posted, the time of post is saved and displayed, furthermore the time at which a new comment is posted is also saved and displayed on the comment. The format that time is displayed is easy to read, a string manipulating function was written to take care of this.

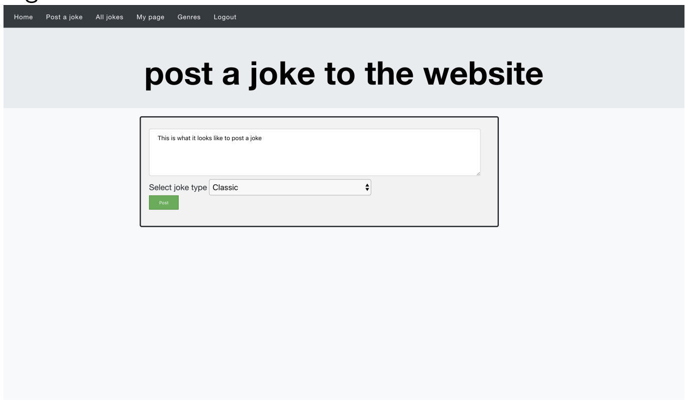
Figure 7:



Time posted: 00:56, 21/11/2018

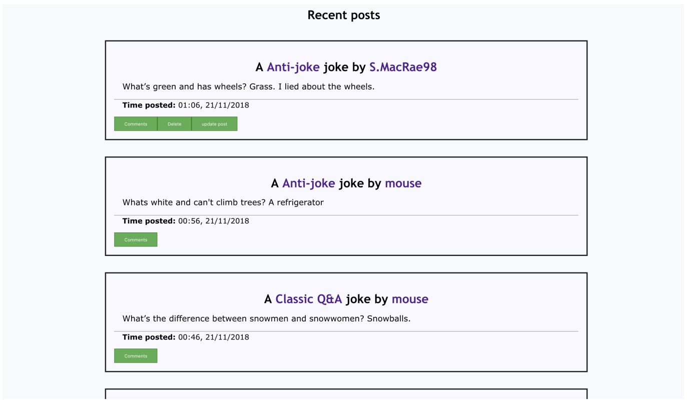
The process of posting to the website is very easy to use, a user inputs their post as text, then they must choose a joke category from a supplied drop down menu which is loaded with categories, before pressing the 'post' button, the user will then be redirected to the home page where they will see their post. The post is saved in the jokes table of the database so it is persistent. The form used to post a joke is shown in Figure 8 below

Figure 8:



Updating and deleting posts is satisfying and easy to use, this was implemented very well. Users will only be able to delete their posts and not others, as shown in Figure 9 it can be seen that there is a button used to delete posts and another button to update the post.

Figure 9:


Recent posts

A Anti-joke joke by S.MacRae98

What's green and has wheels? Grass. I lied about the wheels.

Time posted: 01:06, 21/11/2018

A Anti-joke joke by mouse

Whats white and can't climb trees? A refrigerator

Time posted: 00:56, 21/11/2018

A Classic Q&A joke by mouse

What's the difference between snowmen and snowwomen? Snowballs.

Time posted: 00:46, 21/11/2018

Each post has its own comments page, this comments page displays the post, a form for posting new comments for the post, as well as a section that hosts all of the comments for the post. This feature works well as it isn't confusing, it is clear what the comments page is for and is very easy to navigate to. The comments are neatly laid out with space between them which makes them clear to read. The comments are displayed from old to new in terms of time posted, the reason behind this is because this is the standard way to display comments across social media sites and thus is what a user would expect. On top of this, comments can be updated and deleted by the user who made the comment, it works the same way as updating and deleting posts.

The web app has a very smooth user experience, content is large and stands out, buttons are large enough to see without any problems and their colour stands out from the background. Each post holds information that doubles as links to new pages, which are the user who made the post, that redirects to that users profile page and the category of the joke that links to the respective genre page.

6 Personal evaluation

6.1 What was learned

From undertaking this coursework, my confidence with Python has grown massively, the more I worked on this project the less scared I was with facing new challenges and attempting harder problems. An example of this would be how I approached dealing with the database, I wrote functions which took care of querying the database, which shortened the amount of code required when dealing with the database. The organization of data is clean and works without bugs or crashes, the project was set up very well and greatly improved my skill with databases, this skill was relatively poor before this coursework.

Because of errors faced, researching how to fix the errors developed my skills with using the internet to expand my knowledge of Flask and programming languages in general. Reading documentation directly from the Flask website ensured that the code I was learning was up to date and correct, I relied on Stack Overflow a lot less for this coursework and trusted my capabilities and experience, taking time to examine errors and attempting to understand them without getting stressed and running for help when tasks get difficult. Learning to have trust in myself was the most important aspect I took from this project.

6.2 Challenges faced

Setting up the database, at first I attempted using MongoDB in the form of Mongoalchemy, this was fairly straight forward to set up, although I found that there was a lack of documentation and examples online to study. I had many problems when trying to implement the knowledge that I had gained from this studying, therefor I gave up as this was wasting too much time, the solution to this problem was to switch to SQL, using SQLite3, the workbook for this module was a great starting point, I had enough knowledge to implement an SQL database into the web app and test it. With a poor background in databases I had to research how to query, update and delete records from tables in the database, once I had become more confident with using this technology I was able to undertake all of the challenges required to finish the web app to a standard that I was happy with.

Expanding the issues I had with SQL in more detail, I had written a function that allowed easier querying of the database, the function took in a string and produced an SQL command format which could be executed to interact with the database. The problem was that when building a string with placeholders the function wouldn't format the string properly and this brought up logic errors. I never found out why this was broken, although a solution was found, instead of using placeholders I simply built the whole command using concatenation.

A few errors came about as a result of the syntax of Python being different to the languages I am more fluent in such as C# and Java, since Python has a lot of higher level syntax, I often over complicated concepts and attempted to write unnecessary code or misuse the syntax, a perfect example is would be the 'and' and 'or' statements in Python, which are exactly that in syntax, the table below illustrates this:

Statement	C#	Java	Python
and	&&	&&	and
or	||	||	or
print	console.write	System.out.print	print
declare int	int x = 5	int x = 5	x = 5
End statement	;	;	

Table 1: Comparison of language syntax

An issue that was present for weeks was where the buttons are displayed on the posts, ideally they were to be displayed side by side, getting the buttons to appear beside one another caused some problems, the height of the posts on the website were expanded since the buttons were stacked one on top of the other, this was unattractive and wasted space. I had to research CSS button groups, after adding a CSS class that allowed content to be displayed horizontally, the buttons of each post were placed inside this class and as a result the buttons were displayed in the desired format.

When designing the input forms, I had troubles with allowing text to be written on multiple lines, when making the input form taller, the text would write in the middle, if text filled up the width of the text box the text would carry on horizontally, this was impractical. The solution to this problem was the switch to a text area where the number of rows and columns could be specified.

Choosing a colour scheme wasn't easy, this was left to last, after trying out many colour schemes, my conclusion was to not over complicate the design, keeping a simple design that still looked attractive. Without having lots of experience with user interface design, designing a beautiful interface didn't seem possible without excessive research, I also opted to design the site myself without the use of bootstrap.

6.3 Performance

In terms of performance, I feel that I did very well, the way that I approached this coursework was carefully planned out, before I had written any code I researched how to use databases with Python, as well as planning out how I would move data around. With a solid plan I started writing code, which took around three weeks to complete, the finished product is polished, looks very clean and is extremely straight forward to use. Content is carefully positioned across all of the pages with a consistent colour scheme and layout throughout the website. My time management for this coursework has been better than ever before, working on the project daily as well as starting the coursework early. Because I started the coursework at a reasonable time, I had weeks to study the material needed, read documentation, test out functionality with ample time to complete the writing of code. In comparison to the last social media web app I had attempted, this coursework outshines it, in fact the difference is incredible, I am very happy with what I have produced. In spite of how I feel, I still feel as if there is plenty room for improvement, there were ideas that were never implemented.

This may not be my best project to date, but it has been one of my favourites, the freedom relieved a lot of pressure and with a calmer attitude I feel as if I was a lot more comfortable, as a result I have learned a lot from this project, nothing was rushed. Even after writing all of the code, I have the urge to go back and carry on adding features.

7 References

Some of the code for this project was extracted or based upon the tutorials from these references:

Navigation bar - Navigation bar
User login - Navigation bar
Easy database querying - Navigation bar