COMP2011
Web Application Development

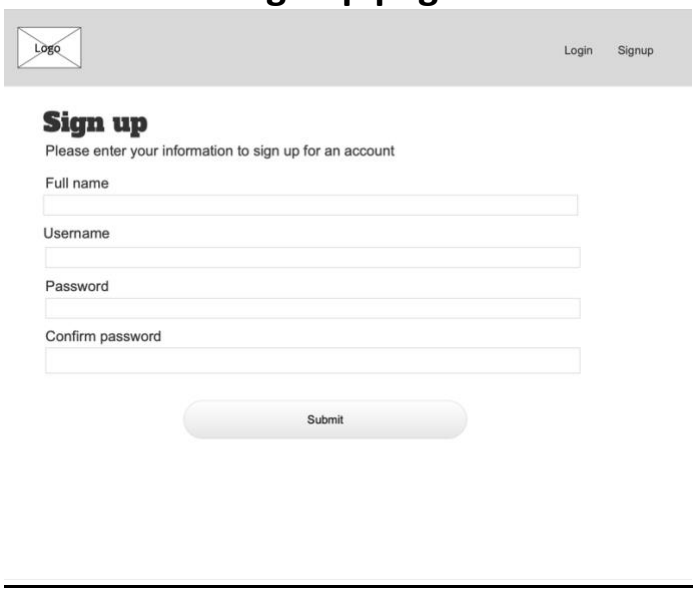
Documentation

Coursework 2 – GradeCalc

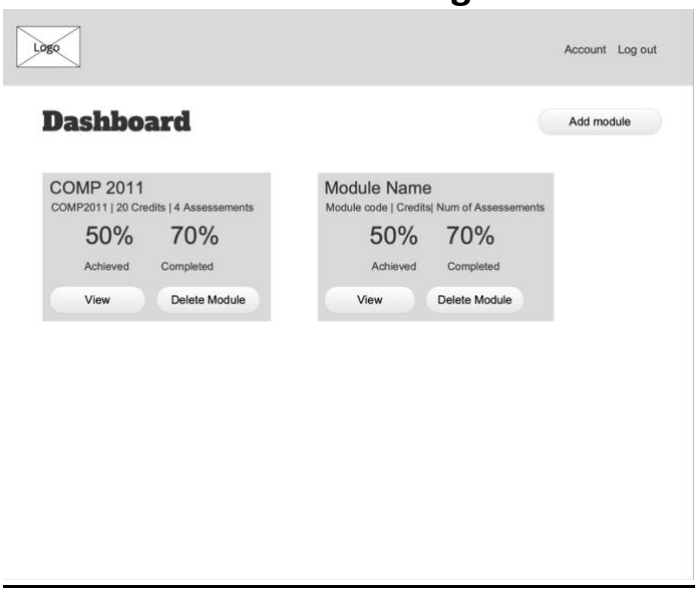
Olanrewaju Sodeinde
bn16os@leeds.ac.uk

Wireframes

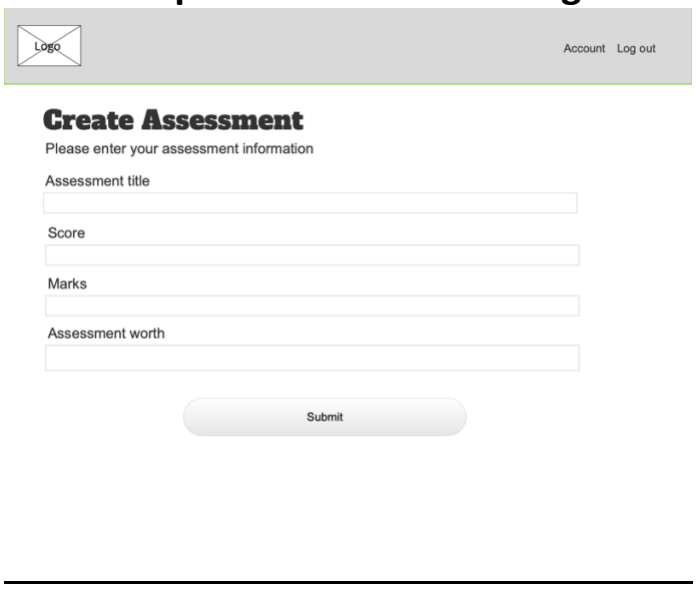
Sign up page



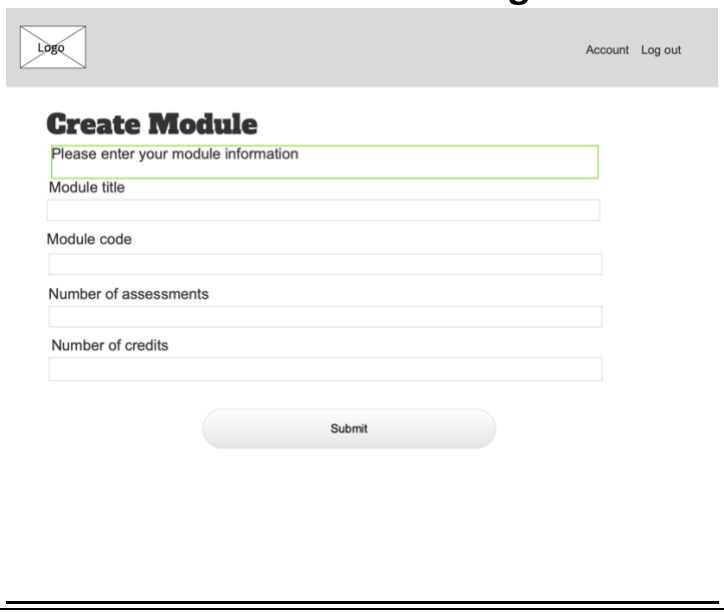
Dashboard Page



Completed Assessment Page



Create Module Page



Deployment

I have made sure to deploy my application, my application is deployed using python anywhere. Below are the website link and login details to access the application.

https://lanre.pythonanywhere.com

Username: gradecalc

Password: calculate

Introduction

Fonts & Colours

Main Colour

#0275d8 

Fonts

Logo - Font family: Bungee, monospace

Body - Font family: Arial, san serif

Statement of purpose

During this project, I built a project that implements a personal assessment to-do list. This application will enable the main users, mainly students, to keep track of their progress during their university course as they progress through a semester/year. The inspiration for this project came from my first year at the University of Leeds where I like many other students struggled to keep track of which courseworks and assignments I had completed and how much of each module I had completed as the semester progresses. This was partially due to the large number of courseworks, and assignments a problem which this application by giving students an easy way to keep track of all this information.

Since this application will mainly be used by students who will require the user interface to be easy to use and approachable, I decided to make the main colour blue. The shade of blue used in this project (hex code: #0275d8) stands for trustworthy and loyalty which conveys a message of accuracy and dependability to the user.

Since this project will be deployed somewhere on the internet, it is important that the logo looked clean, professional, and trustworthy so I used, 'Bungee', a slab font which adds to the professionalism of the application. The main font used in the rest of the application is Arial, a san serif font. Sans serif fonts are known for being modern, approachable, and clean. By choosing this type of font, the application will look modern, clutter-free, and easy to use which will encourage the younger target audience to use the application more.

List of features

The application will be split between several pages. Namely a homepage, signup page, a login page, dashboard, add module page, add assessment page, Account page and an edit password page. I have decided to split up the pages & functionality like this for simplicity and an improved user experience. Each page provides a specific function to the overall application & that and only that function is carried out on that page.

Homepage - The homepage will be the first page users arrive at when using the application for the first time, it will give users basic information about the application and direct them to sign up. This create a good user experience as it gives the user an introduction into the web application and its function before they have to provide any of their information which creates trust.

Sign up page – The sign-up page makes use of a form where a user can user will provide basic information such as their name, username, and password to create an account. The form carries out some validation before submitting, it only lets users submit the form if all the fields are filled in with valid information after which they are redirected to the login in page. A check is also made against the database of current users to make sure that the username is unique. After the form is submitted, users' passwords are hashed & saved in the database, this is done as a quick and effective and secure way to store the password in the database rather than in plain text.

Login page – On the login page, users can input their username and password to be authenticated. This is checked against the database of current users, if there is a matching username and password, the user is authenticated and taken to their dashboard. User can also choose whether or not the browser 'remembers' their login information so they are taken straight to their dashboard rather than having to authenticate themselves the next time they login.

Dashboard – On the dashboard users will be able to see all the modules they have added, as well as any feedback for the module. On this page users can click a button to add a new module where they will be redirected to the add module page, click a button to view a module where they will be redirected to the view assessment page where they can view information about the selected module, and finally click a button to delete a module which will delete the module and all its accompanying assessments. The dashboard improves the user experience of the app because it creates a familiar 'homepage/base'. In the unlikely event that a user gets lots or wants to start again from the beginning, the dashboard proves a familiar and user-friendly homepage.

Add module page - On the add module page, users will be able to input information about a module into a form and create a new module to add to their dashboard.

Add assessment page – On the add assessment page, users will be able to insert information into a form to create a new assessment to add to the selected module. Once all the information for the assessment is added the assessment is saved into the database. This also updates the module feedback to include information about the newly added assessment.

View assessment page – On the view assessment page, uses can view all the assessment they have added for a particular module.

Account page – Users can view their account information which includes their full name and username. A user can also change their password on this page which redirected then to the edit password page. The account page also improves the user experience as users are able to personalise their account by changing their password, this generate trust with the application and makes users more likely to use the application.

Edit password page – This contains a form which prompts the user to input their old password, new password and to reconfirm new password. The form queries the database to make sure that the old password is correct but also checks to confirm that the old password is not the same as the new password. If the new password and confirm password fields match, the new password is saved into the database, and the user has successfully changed their password.

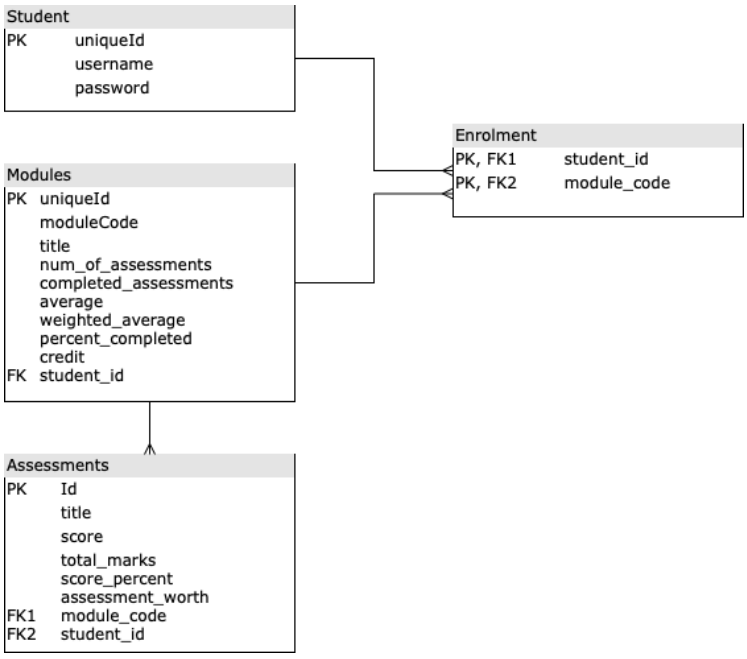
Analysis of application

Forms and Validation

My implementation makes extensive use of web forms. On the login page, signup page, add assessment page, add module page, edit password page forms are used to gather information from the user which are then sent to the server via POST requests. All forms on the application carries out some form of both client side and server-side validation. For example, on the add assessment page all the forms carry out client-side validation to ensure that fields are filled with valid data and uses the 'type' client-side validation to make sure that the information in the specified field is an integer. I also made use of server-side validation. For example, in the add module code page, I wrote a custom server-side validator which queries the data base to ensure that the module code of each module is unique.

Database

My application makes use of 3 database models to store information namely a student database, modules database and assessment database. The student database that stores all the information about the user information including the student unique ID, name, username, and password. The module database stores information about each module for each user/student. Information stored in this database includes module title, the number of assessments, number of completed assessments, module weighted average. The assessments database which contains information about each assessment completed by the users. The student and modules databases are related to each other using many to many relationships, so I have decomposed them into two one-to-many relationship and introduced a new table called enrolment.



Sessions

To pass information between webpages, my application makes use of sessions. On the dashboard, when a user wants to view the information about a about a module, I needed a way to keep track of that requested module so once I got to the view assessments so I could use that information to query the database. So, I used sessions to store the modules code keep track of the module the user is currently requesting to view. Every time a user logs out, the session information is cleared to ensure the session info from one user doesn't linger onto the next user.

Authentication

Users of the web application must create an account before they are able to login and use the web application. User will have to provide details such as their name, username, and password to create an account. The user's password is hashed using the werkzeug.security module and save in the database.

Before users can login to their account, they must be authenticated by providing their username and password. Since a properly implemented authentication method is important for the security of the web application, my application uses flask login to keep track of authentication of users.

Styling

During the implementation of the web application, I used bootstrap and custom CSS to customise my web application to my liking. Bootstrap provided easy to use classes which can be simply inserted into my code that enabled me to get the styling off application of the ground quite quickly. However, applying extensively applying custom styles to the application afforded me complete control over the look and feel of the application. I user mainly primary colours for the styling to give the application a familiar user interface. I also ensured that I chose consistent styling throughout, for example blue buttons carry out a positive action such as login and red buttons carry out a negative action deleting a module. This improves the user experience as it gives the user a familiar and easily understandable way to use the application.

Testing

Testing plan

1. To be able to create a new module with all fields populated.
2. To be able to create a new assessment.
3. To be able to view all modules.
4. To be able to view all assessments for a specific module.
5. To be able to delete a specific assessment
6. To be able to delete a specific module

Specification	Method	Description	Expected
1	Create a new module with all form fields populated		A new module is created
1	Create a new module without a title		A new module should not be created
1	Create a new module without a module code		A new module should not be created
1	Create a new module without the number of assessments		A new module should not be created
2	Create a new assessment with all form fields populated		A new assessment is created
2	Create a new assessment without an assessment title		A new assessment should not be created
2	Create a new assessment without a score		A new assessment should not be created
2	Create a new assessment without total marks		A new assessment should not be created
2	Create a new assessment without the worth of the assessment		A new assessment should not be created
3	View all modules	Create 4 modules with different names Ensure you can see all modules and their details	
4	View all assessments	Create a module with appropriate information. Click the 'view module' button. Create 3 assessments. Ensure you can see 3 assessments on the assessments page of the module.	A new module and 3 associated assessments should have been created.
5	Delete an assessment	Create a module with appropriate information. Click the 'view module' button. Create 3 assessments. Delete one assessment.	The number of assessments should go down to 2. Ensure there are only 2 assessments in the list.
5	Deleting an assessment should update the 'average field of the module.	Create a module with appropriate information. Click the 'view module' button. Create 3 assessments. Navigate 'back' by clicking the back button and take note of the average for the module. View the module again and delete one assessment. Navigate 'back' to view all the modules.	After navigating back the second time, the average for the module will have changes i.e. go down.
6	Delete a module	Create 2 modules with appropriate information. Delete one module from the list.	The module should be deleted. Ensure there is now only one module in the list.

Logging

My application makes extensive use of logging to keep track of events happening inside my application. I made use of 3 levels of severity. INFO, WARNING & ERROR to properly differentiate the type of activities taking place inside the web application.

Advanced feature - Bootstrap & jQuery

As mentioned above, my application makes use of bootstrap to style the application. My application also makes use of jQuery. In my application, I use flashed messaged to display useful warnings to users. However, these flashed messages can quickly build up as the user performs actions throughout the app and clutter up the user interface, so I used jQuery to set up a timer script to automatically dismiss flash messages after a period. This feature is directly relevant and strongly enhance the user experience because without this feature, the flashed messages will stack up, taking up the precious screen real estate meant for other elements which reduces the user experience.

Security

There are several potential security issues my application could face for example Cross site scripting. This threat occurs when an attacker submits unsanitised JavaScript to a web server that when the input is returned to the user unsanitised the browser will execute the JavaScript which can do any number of things which can leave your web application vulnerable. The flask framework I used in this project mitigates against this kind of attack. Flask by default configures the templating engine to escape all input rendered onto the page helping my application mitigate against this kind of attack.

Another threat my application could face is the Insecure Direct object reference. This attack occurs when an attacker obtains a reference to an internal object such as a file or a database record when they should not have access to the object. This could potentially allow the attacker to manipulate my web application in a way which it wasn't intended. To prevent this type of vulnerability, I use proper authorisation consistently throughout my application. For example, on the add assessment page when requesting values from the sessions object, the current user is checked to see if they are authorised.

Final comments and personal evaluation

During this project, I learnt how to properly deploy a web application using the flask framework. I have gained a greater understanding of the strength and weaknesses of the flask framework and how to make the most of it strength whilst actively working to limit the impact of its weaknesses. I have also learnt about the different common vulnerabilities that could face web applications and how one could limit / mitigate their effects.

The combination of all this information has made me much more confident in my skills of taking a project from its inception to a fulling working and tested and robust web application.

References

Ross, D.J. n.d. Bungee – Fonts for multi-color and vertical typography. *djr.com*. [Online]. [Accessed 9 December 2021]. Available from: https://djr.com/bungee.