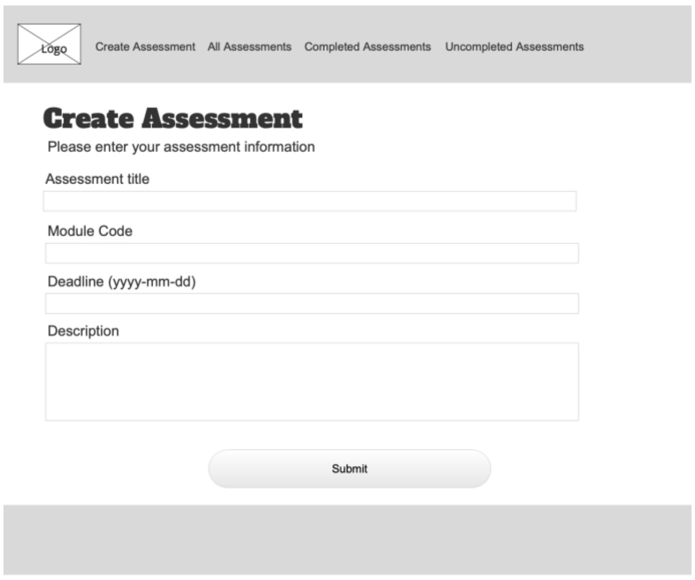
COMP2011 Web Application Development

Documentation

Coursework 1

Olanrewaju Sodeinde
bn16os@leeds.ac.uk

Create Assessment Page



Create Assessment

Please enter your assessment information

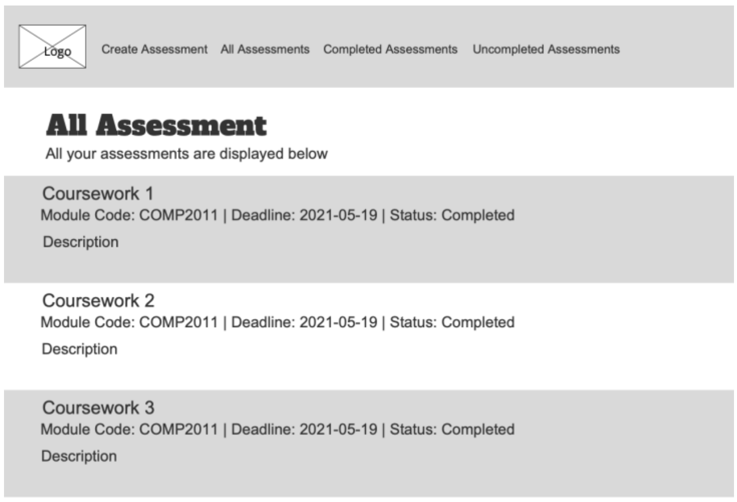
Assessment title

Module Code

Deadline (yyyy-mm-dd)

Description

All Assessments Page



All Assessment

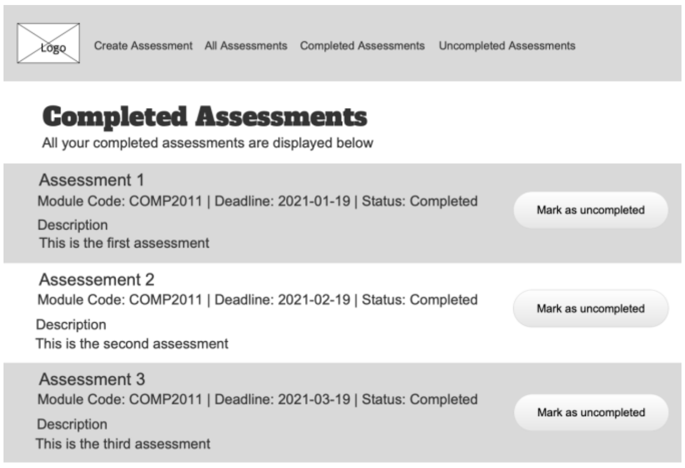
All your assessments are displayed below

Coursework 1
Module Code: COMP2011 | Deadline: 2021-05-19 | Status: Completed
Description

Coursework 2
Module Code: COMP2011 | Deadline: 2021-05-19 | Status: Completed
Description

Coursework 3
Module Code: COMP2011 | Deadline: 2021-05-19 | Status: Completed
Description

Completed Assessment Page



Completed Assessments

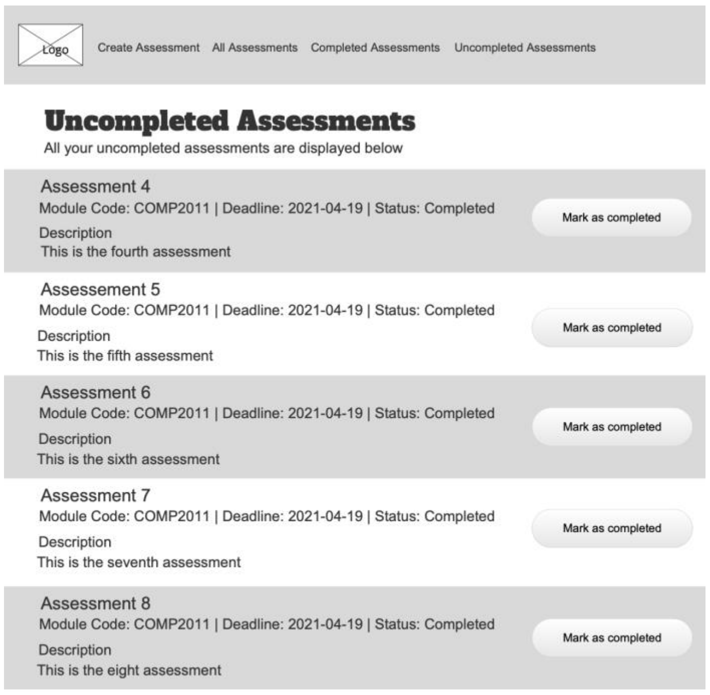
All your completed assessments are displayed below

Assessment 1
Module Code: COMP2011 | Deadline: 2021-01-19 | Status: Completed
Description
This is the first assessment

Assessment 2
Module Code: COMP2011 | Deadline: 2021-02-19 | Status: Completed
Description
This is the second assessment

Assessment 3
Module Code: COMP2011 | Deadline: 2021-03-19 | Status: Completed
Description
This is the third assessment

Uncompleted Assessment Page



Uncompleted Assessments

All your uncompleted assessments are displayed below

Assessment 4
Module Code: COMP2011 | Deadline: 2021-04-19 | Status: Completed
Description
This is the fourth assessment

Assessment 5
Module Code: COMP2011 | Deadline: 2021-04-19 | Status: Completed
Description
This is the fifth assessment

Assessment 6
Module Code: COMP2011 | Deadline: 2021-04-19 | Status: Completed
Description
This is the sixth assessment

Assessment 7
Module Code: COMP2011 | Deadline: 2021-04-19 | Status: Completed
Description
This is the seventh assessment

Assessment 8
Module Code: COMP2011 | Deadline: 2021-04-19 | Status: Completed
Description
This is the eight assessment

Wireframes

Introduction

Fonts & Colours

Main Colour
#0275d8 🟦

Fonts
Logo - Font family: Monospace, sans serif
Body - Font family: Arial, san serif

Introduction & Evaluation of Intended Audience

During this project, I built a project that implements a personal assessment to-do list. This application will enable users, mainly students, to submit their assessment information to the web application where it will be processed and presented in a friendly manner. 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.

The main font used in the project 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.

Planning & Layout

The application will be split into 4 different pages. A create assessment page where users will be able to create assessment, an 'all assessments page' where users will be able to view all their assessments, a 'completed assessments' page where users will be able to view their completed assessments and finally and 'incomplete assessments' page where users can view their incomplete assessments. 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.

Design & Architecture

My implementation makes extensive use of HTTP requests. On the create assessment page, my web app contains a form where users can insert enter their assessment information, when the user clicks the submit button a HTTP POST request is made to the server containing all the information entered in the form, which is then processed, validated, and saved into the database. Similarly on the completed assessment & uncompleted assessments page, a button is provided next to each assessment which also makes HTTP POST requests.

In my routes provided by my views.py file, I defined a complete assessment & uncompleted assessment routes that both make use of HTML requests. When a request is made, each route checks whether a POST request was made, if this condition is fulfilled then assessments are marked as 'completed' or 'uncompleted' depending on the context. However, if the request made is not a POST request (e.g., a GET request) then no action should be taken.

During the implementation of this web application, I also made use of a three-tier architecture. The three tiers include the presentation layer, the business logic & data access.

In the presentation layer, I made extensive use of Flask's templating & template inheritance. My application is to be split between four pages. All four pages make use of templates extensively by inheriting from a base template called 'base.html'. The 'base.html' file contains the template that runs in all the pages and the remainder of the template will be substituted in instead (% block content %) template. Whilst all the pages inherit from the same base code, by utilising the power of templating & template inheritance, I can present dynamic content to the user on each page.

Using Flask.route & the Jinga 2 engine, I was able to implement business logic of the web application. The render_template() function provided by the Flask framework takes the filename of the template to be rendered as well as a set of arguments which should be substituted into the template. Then within the render_template() function, the function invokes the Jinga 2 engine which substitutes the brackets with the corresponding argument passed to the function call.

To implement the data access layer, I made use of models provided by SQLAlchemy. My model made use of 6 fields named id, title, module_code, status, deadline & description which allows me to store the information submitted by the user into the database.

Here is a table that contains each field & its arguments, and the justification for the data type I have chosen to use:

Field	**Justification for data type**
id = db.Column(db.Integer, primary_key = True)	The data type of the id field in an db.Integer. This enables a unique number to be generated for each tuple added to the database that uniquely identifies each entry.
title = db.Column(db.String(1000), index = True)	The data type for the title field is a db.String of 100 characters that enables the title to be stored in the database as a string of characters to be processed.
module_code = db.Column(db.String(1000))	The data type for the module code field is a db.String data type because the module codes will be inserted as a string of characters to be processed.
status =db.Column(db.String(1000), default='Uncompleted')	I decided to use a db.String rather than a db.Boolean, or other data type because this will enable me to use the filter_by method to get the records that are either 'uncompleted' or 'completed'.
deadline = db.Column(db.Date)	I decided to use a db.Date rather than db.DateTime data type because I needed to store only the date, whereas the db.DateTime datatype stores both the date and time in the database.
description = db.Column(db.String(10000)	The data type of the description column is a string of 10000 characters because this field could potentially contain a lot of text information, so I needed to provide a space large enough to hold the description.

Table 1: Fields in the database model and the justification of their datatype

Implementation

There were no major changes between my original design and my implementation. The only minor stylistic change made to the website were on the pages displaying the assessments, such as the completed assessments page.

I realised that to provide an even better user experience than I originally designed, rather than having each assessment alternate their background colour between grey and white, it will be better if a colour is displayed behind the assessment, when its box is hovered on. This provided an even greater level of interactivity that improves the user experience. I also implemented a feature where when a user indicates that they want to print a list of assessments for any of the page, the webpage is automatically formatted to fit into the width of an a4 page and all interactive elements such as the navigation bar and the buttons are hidden to enable proper printing.

Security – Preventing Cross Site Request Forgery (CSRF)

CSRF is a type of attack on a website or web application that involves a third party forging a request to the application's server. The 'wtf forms' import I used in my implementation make it possible to prevent CSRF by generating a unique token when rendering each form. That token is meant to be passed back to the server along with the form data in the POST request and must be validated before the form is accepted. Every time a request is made a unique identification is generated to prevent CSRF. This feature improves that security of the web application and makes it harder for to make malicious requests.

Testing Methodology

To ensure that every additional feature has been tested properly, I used a manual testing approach to deliver robustness. I manually tested every feature to ensure that it works as intended across multiple web browsers such as Google Chrome, Apple Safari & Microsoft Edge. Since users must submit requests to be processed, I manually tested each feature to ensure that the forms are submitted correctly, the information is added to the database correctly and the information can be retrieved from the database. Where there were errors, I made sure to correct them and I tested the website and its functionality thoroughly in other browsers such as Google Chrome, Apple Safari & Microsoft Edge to ensure no errors were present. I fully validated the HTML and CSS files to make sure there are no errors in the code, and they are fully compliant with HMTL5 and CSS3 standards. All these steps ensure that the final product is of high quality and works fully as intended.

Accessibility

The web is used by billions of people every day, with different levels of ability and different disabilities so it is important that the web application is inclusive & accessible to all users and takes them into consideration in its design.

WCAG is a standard used to provide web accessibility web content accessibility that meets the needs of individuals, organizations, and governments internationally. To make my website WCAG compliant, I have taken several steps to ensure all the webpages meet several its principles e.g., Principle 3, Understandable. For example, to meet guideline **3.1. Readable**, I have defined the language on each of the webpages as English("en") in the head tag of base.html template. Since my website makes use of the flask framework, when each page is rendered, it inherits from the base.html page which ensures that all the pages also have "en" in the head tag, fulfilling guideline for **3.1 Readable.**

I also made sure all webpages meet guideline **3.2 Predictable**, by giving all webpages a navigation bar in the top right and a logo in the top left. This ensured that there is consistent navigation on each page across the web application unless a change is initiated by the user. In **3.3. Input Assistance**, **Guideline 3.3.2** states that labels or instructions are provided when content requires user input. To meet this guideline, all input fields in the create assessment form, on the create assessment page all have an associated label describing the purpose of each input field.

To make the web application accessible to visually impaired users, I have made sure that the website is easy to read in a text only mode. I tested this feature by opening the webpages in several text-only readers to make sure the webpages maintain readability . I also made sure to provide alternative fonts for the webpages, so when there is no support for the default website font, users of non-standard web browsers can still view the website as intended.

Since 8% of the general population has some form of colour blindness, it is important to make sure the website is accessible to them. I have also checked the text and background colour combinations to ensure that the contrast is sufficient. As mentioned above, this also allows my web app to implement a 'printing' feature for the website. HTML heading elements have also fully been used to represent page structure, supporting assistive technologies recognise and understand content of the page.

Challenges

During the implementation of the web application, due to my inexperience with the flask framework and my familiarity with plain HMTL & CSS, I found it quite difficult to customise the website to my liking. Whilst bootstrap provided easy to use classes which can be simply inserted into my code, customising these classes proved quite challenging at first until I created a separate CSS file where I applied custom styles to make my application look exactly how I needed it to.

I also experienced some difficulty with connecting the database with the website. Whilst it was easy to create the database model and add data into the database, I found it difficult to display that information on the webpage and how to create the different view based on the context (e.g., completed assessments and uncompleted assessments). To solve this problem, I did a lot of additional research to consolidate my knowledge on databases from the module site and previous module. This led me to the realisation that I will need to figure out a method of making POST request from the website to retrieve information from database to be displayed.

Final comments and personal evaluation

During this project, I learnt the importance of the 3-tier architecture and how it uses stacks of tiers to fulfil its purpose, namely the presentation layer, the logic layer, and the data access layer.

During previous modules and projects, I learnt how to use advanced features of HTML & CSS & JavaScript and how to create simple interactive web applications. In this project, I learnt how to implement the Flask Web framework and the process of connecting a database to a web application. I learnt about HTTP request and how you can leverage their power to make request to a server / database to access stored data.

I believe the combination of all this information has made me much more confident in my skills of using multiple technologies to create accurate & robust web applications & I look forward to using the skills and information I have learnt in future projects.