COMP2011 Coursework 1

Design & Evaluation Document

Callum Andrew Harrison

201392132

sc20cah

28 October 2021

Layout

Wireframes

Table

Description automatically generated Table

Description automatically generated

Table

Description automatically generated Graphical user interface, application, table

Description automatically generatedTable

Description automatically generated

Table

Description automatically generatedGraphical user interface, application

Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Popup:

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generatedResponsive WireframesA picture containing chart

Description automatically generated

Diagram

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Justification

This website has 4 pages each of which I have provided the wireframes above for each scenario listed below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Page Title** | **Descriptions** | **Scenario 1** | **Scenario 2** |
| All Tasks (Index) | Initial page on load.  This lists all tasks both complete and incomplete if there is any | Tasks available, I have shown the styles for each type of task. The main difference being the completed tasks don’t have a checkmark to mark them as complete | No tasks available so an alert is shown instead telling the user there aren’t any tasks |
| Incomplete Tasks | Lists all the incomplete tasks | Each task is listed with its title, a drop-down button, a check button to mark as complete and a delete button | No tasks available so an alert is shown instead |
| Complete Tasks | Lists all the completed tasks | Each task is listed with its title, a drop-down button and a delete button | No tasks available so an alert is shown instead |
| Create A Task | Page used to create each task. Each task is created and initially marked as incomplete | A form is shown which has prompts for the user to fill in each field and a submit button to send the data to be validated. | Task has been created so a success alert is displayed alongside the form. |

For the following visual components, I have opted to use Bootstrap 5.0 as Bootstrap is a widely used framework, so it is most likely the case that the website posses a familiar look to many other websites making the user more comfortable and allows the website to be more navigable.

* Navigation Menu – Traditional and simple styling which also provides a responsive menu button for smaller screens.
* Alert Messages – Simple and colour coded alerts which direct the users view to important information.
* Popup Modal – Common and familiar styling and intuitive to use. Also darkens the webpage in the background so it is easily visible.
* Form Fields – Common and familiar styling with a coloured submit button to direct the user’s attention.

On the pages which have the tasks listed, each task has a drop-down button. This button opens a bootstrap modal which darkens the entire webpage for contrast and shows a dialog in the centre of the screen. These modal shows all the details for the task selected. In contrast to the addition of the navigation menu, for this website I have opted for no page footer. My main reason for this is due to the lack of content on some pages, this means that if there are little to no tasks on a task list page then the footer would appear higher up in the browsers view. In my opinion this makes the website look more cluttered and less appealing to users.

Architecture

Tier 1 - Presentational

This tier of the website architecture includes aspects of the web application that manages the presentation of the application such as HTML responses and handling form data for each request. In this website this is managed by the Flask framework. For this project there are 2 webpage templates. One of which is for the 3 pages which list the tasks (All, Incomplete, Complete) and the other is for the create a task page. To return the templated HTML & CSS code flask uses the Jinja2 templating engine and for form data and form modelling it uses FlaskWTF. For the create a task page form it is modelled as such:

|  |  |  |  |
| --- | --- | --- | --- |
| Title | Text Field | Required | Max 500 Characters |
| Module | Text Field | Required | Max 500 Characters |
| Due Date | Date Field | Required | DD-MM-YYYY |
| Description | Text Field | Required | Max 500 Characters |

Tier 2 – Business Logic

The business logic for this website covers the views of the website. This is covered by Flask.routes. This component of Flask takes the request and compares the URL (e.g. /) and type (e.g. POST or GET) to a given list. From this it returns a given html template along with any required data fetched from the third tier of this architecture.

Tier 3 – Data Access

This tier of the architecture is focused on communicating with the database. The database is stored as an SQLite database which can be managed using the SQLalchemy framework. The structure of the database can be changed and updated using database migrations. The only table in the database for the website is called Tasks, of which each entry follows the following structure:

|  |  |  |
| --- | --- | --- |
| Id | Integer – ensures every entry is unique | Primary Key |
| Title | String – Custom specified title without formatting restrictions | Max 500 Characters |
| Module | String – Stores a module code without restricting to any format | Max 500 Characters |
| Deadline | DateTime – The deadline date |  |
| Description | String – Custom user description | Max 500 Characters |
| Complete | Boolean – Marks the task complete |  |

Evaluation

The Website has been implemented directly according to the wireframes. The only difference is the addition of the colours. In some aspect ratios and screen sizes the website has the same sizes as the wireframe models. I have implemented scalability and responsiveness such that the website looks suitable on most if not all devices and aspect ratios an example of this is the responsive navigation bar which slightly changes styling to fit different screen sizes, another is the movement of delete and complete buttons on tasks to avoid screen overflow and task titles which are not readable before they are cut short.

One of the biggest challenges I faced during development was implementing the creation of each task. I decided upon the form sending a HTTP POST request to the same URL. From this, the code for the views detects the different HTTP request type and creates a table entry in the database then returns a success message and finally outputs the create page template. In addition to this the complete and delete buttons on the task send a HTTP GET request to the URLs ‘/delete’ and ‘/mark\_complete’ and then after submission it redirected to the previous page where the user clicked the relevant button.

Testing & Debugging

Testing and debugging code is one of the most vital steps of software development. It helps prevent against malicious attacks in addition to ensuring that the product meets the quality standards of the customer and keeps their data safe. I have outlined below my test data for all inputs:

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Data Input** | **Expected Outcome** | **Actual Outcome** |
| Title | Empty | Data not submitted and  User prompted to fill in field | Data not submitted and  User prompted to fill in field |
| Any String  (e.g. Coursework 1) | Data submitted and a task created | Data submitted and a task created |
| Module Code | Empty | Data not submitted and  User prompted to fill in field | Data not submitted and  User prompted to fill in field |
| Any String  (e.g. COMP2011) | Data submitted and a task created | Data submitted and a task created |
| Date | Empty | Data not submitted and  User prompted to fill in field | Data not submitted and  User prompted to fill in field |
| String of Invalid Format  (e.g. ADNJSDH) | User warned that the data is invalid and the form is not submitted | User warned that the data is invalid and the form is not submitted |
| String of Correct Format  (e.g. 25-10-2021) | Data submitted and a task created | Data submitted and a task created |
| Description | Empty | Data not submitted and  User prompted to fill in field | Data not submitted and  User prompted to fill in field |
| Any String  (e.g. To-Do Web Application) | Data submitted and a task created | Data submitted and a task created |

To test the website presentation, during initial development Chromium based browsers such as Chrome, Microsoft Edge and Brave were used in addition to Mozilla Firefox which is based on Quantum Browser not Chromium. These browsers were selected as they covered the most commonly used web browsers. These browsers were used to test different screen sizes and CSS effects. After initial development improvements were made using older browsers such as Internet Explorer to ensure the website is suitable on browsers which don’t support some CSS styling and HTML elements. Unfortunately, on some older browsers which don’t support the latest CSS3 and HTML5 the website isn’t properly displayed and/or styled; however, this could be a future improvement for the website. This could be achieved by using CSS media queries which detect browser versions to change the styling of content to a different but similarly appropriate style.

In the effort to organise my code I have separated files relating to each tier of the website’s architecture into separate files, from which anything required from other files are imported. An example of such is the linking of the forms from tier 1 and the database from tier 3 into the views code for tier 2. This separation of code came in use when it came to debugging as it allowed me to test each tier independently and find any errors.

Security

As this website has been implemented as a single user application the data, which could possibly be sensitive, is available for anyone on a local network who has the local IP Address or if made available on the internet, anyone who has the URL. Perhaps in future revisions a user system could be implemented where user details are encrypted and stored. Additionally, as the database is stored locally with a ‘.db’ file extension the data should be further encrypted to keep it secure if the file is stolen or copied. Alternatively, a cloud-based database solution could be used as it is sufficiently secured and the database file is inaccessible to everyone except the database host.

Accessibility

Accessibility is an important aspect of any website or program. To test accessibility, I have tried the following.

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Outcome** | **Actual Outcome** |
| Disabling The Mouse | The website is still navigable using the Tab key on the keyboard to move through all the buttons and fields. | The website is still navigable using the Tab key on the keyboard to move through all the buttons and fields. |
| Disabling Images | Images are replaced by relevant substitutes to keep webpage structure | There are no images, so the website remains the same |
| High Contrast Mode | Still Navigable, Legible and Structured | The website structure remains the same however, the background blends into other widgets. This could be changed in future revisions |
| Disabling CSS | The website will still be navigable and legible | All items remain however the popup function is not enabled instead the data is always shows in-line |

I believe the only aspect of accessibility that could improve for the accessibility of this website is the styling of high contrast mode, this could be solved in future revisions using additional CSS styling.

References

Bootstrap 5 is under the MIT open sources license and is referenced in the code.

Flask is provided under the BSD 3-Clause license