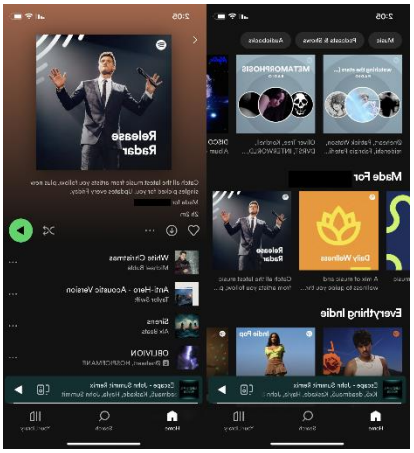
COMP 1004 – Computing practice

2023/2024

Music player application (Theme: ENTERTAINMENT)

Introduction

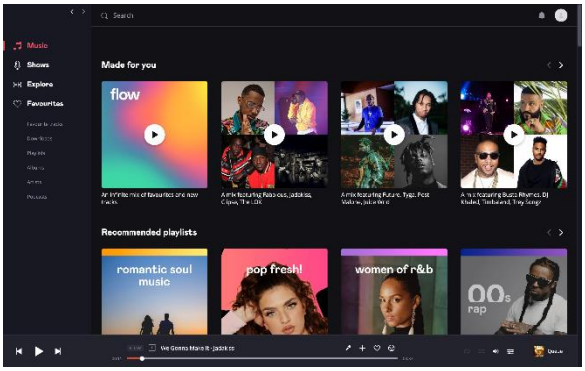
In this current day most people listen to music either online or through downloading and storing it on their devices. Most music application software allows for users to create playlists, favourite music and see recommended songs based on what users are listening too. I will be developing an online music player that if possible, will allow users to search for music online, upload their own music, create playlists and download songs for offline listening. The report I create will show the stages of the software development lifecycle along with the software requirements, including the user stories and the architecture of the system. The report will then describe my sprint plans so that I know what to do when planning my implementation. I will also be evaluating the parts of my project that I add to my report. This will allow me to filter out bits that may not be relevant to my project. This report will contain labelled graphs and diagrams to make it easier for readers to understand what my web application will involve and how it will work.



The image shows an already existing music application, known as Spotify. It allows users to stream music as well as download and upload their own songs.

Spotify uses a unique colour scheme for its branding and uses an easy on the eyes menu to allow for people of all ages to have good accessibility.

Source: Google images



Another example is Deezer, which is a similar music software that uses online music streaming and other user uploads to allow people to listen to music. It gives recommendations based off what people are listening too and I will be taking inspiration from these sorts of features. Just like Spotify it allows users to create playlists and filter through music using categories such as artists etc.

The menu also consists of bright eye-catching colours against a darker background.

Software development lifecycle (SDLC)

The software development lifecycle allows for a methodical and easy approach to developing any computer programs. There are 5 stages to this process:

- Requirements
- Design
- Implementation
- Testing
- Maintenance

There are also many types of SDLC, each one suited to a certain type of project better. They vary in how the order of the instructions are executed and the amount of time spent on each stage.

Requirements Analysis

This is the first stage of the software development lifecycle, it is about gathering all the requirements needed to start development on your project. These can be split into 3 parts which include, Functional, Non-Functional and usability. Functional requirements are requirements that include the tasks that the product performs and how it reacts to inputs. Non-functional requirements describe the restrictions and restraints of a piece of software and how well it can meet its functional requirements. The usability requirements are the last set that are needed to describe how easily a piece of software can be used. There is a fourth set called external requirements, these are a last thought and include the social, ethical and legal considerations of the project.

Design

After the requirements have been fulfilled you move onto the design stage which involves two parts. These are called detailed and style both being used for different parts of the project's development. Style is used to design the architecture and the basic layout of how the software will run and function. An object-oriented paradigm is a way of viewing the software during development and uses classes with set attributes and methods. This makes code reusable.

During the design phase you also need to create diagrams in the UML such as class diagrams, state diagrams and sequence diagrams. Class diagrams allow users to present classes and objects in software.

Implementation

The implementation stage is where you take what you developed in the design section and start putting it into code. This is usually the most important part as you are now actually writing the program. The code language for the webpage will be using HTML. During this stage it will be useful to refer back to you design ideas and class graphs.

Testing

The testing phase happens once the programmer is happy with the implementation and programming stage. The programmer then runs multiple tests on the software to weed out any bugs and errors that will need to be fixed. The testing phase can sometimes happen alongside the implementation phase as the programmer will discover bugs during development. Some errors can be worse than others and depending on the severity it might make the programmer have to revert back to the implementation or design stage to recode or rethink that entire part.

Maintenance

The final part of the lifecycle is known as the maintenance stage and is used post program launch. This stage is all about maintaining the software as the system around it updates. For example, windows operating system updating to a later version my make the current version of the software incompatible meaning it will need an update to fit with the new system. This stage might not always be about just keeping an application up to date, there will be times where the software may need some minor bug fixes are updates/patches to fix errors that will pop up throughout its lifetime.

Maintenance may be the smallest step in the SLDC when it comes to adding content but it will be the longest lasting one as it will be in effect throughout a program's lifetime. You see examples of it everyday when the majority of apps on our phones see updates regular. It's the same for computers, where the operating system is constantly seeing patches and small updates.

Types

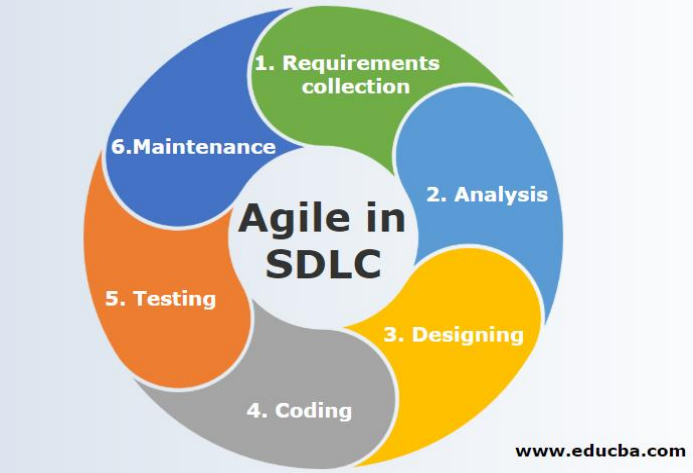
When it comes to types of the SDLC there are many to choose from. The list includes names such as the Waterfall mode, Increment model, Spiral model and the V-model. These all vary in how their stages are laid out and what order they are executed. Some of them allowing for very flexible stages where as others are more fixed in approach and are a lot stricter and more unforgiving when mistakes are made in the cycle.

For my project we have been assigned to use the AGILE software development life cycle model.

Agile

The Agile SDLC is a very requirement focussed one that relies on users seeing quick results coming from each stage. The agile approach uses periods called sprints that are normally done in two-week time slots. In this time a small amount of the project is completed, this can range from any stages of the SDLC. Then after that period, there is usually a meeting where many developers come together in an "Agile team". They discuss progress and points such as what did the developers change or what they added to there base project

idea. User stories and a backlog are a stage only present in this type of SDLC, which is seen featured later on in my report. User stories are used to describe what features a piece of software will contain in an almost first-person user perspective. These user stories may not always be fulfilled but they give a rough idea to the user of what the program should feature. The project backlog is the other unique planning stage of Agile that is a document which contains all the current features of a sprint that need to get worked on. All the features being worked on and all the completed features.



This image shows a basic graph that lays out the Agile SDLC. Agile is also a method that allows user to go back and change things in each step at any point in time. Unlike some other types that don't allow users to go back and make changes to earlier parts. Once you have done it, you must go through with it.

Design Documentation

Project description

In a modern age most people use mobile devices or computers for entertainment. Listening to music is one of the larger ways people like to spend their free time, whether it's working, studying or just relaxing. Music in the background always helps. Everyone should have access to this sort of thing. Music helps in relaxing people, getting people riled up and conveying a mixture of emotions through sound. It can allow people to cut themselves off from the outside world/reality, similar to the way video games work people can get immersed in music.

As stated in my introduction my project will be a single webpage music application. Where users can search for different songs and listen to them online. The file inputs and outputs will come from the user being able to input song names/song links and the title, artist and release date etc being outputted onto a file. The software will come with many features such as being able to create playlists out of multiple songs that a user may like. The application will have an introduction screen login that will allow a user to have an account where they can then access all their recent songs, playlists etc. This will also allow for multiple people to use the webpage application on the same device. The login page will allow for users to ask for a password reset etc.

To create this, I will be using an API and embeds taken from existing software, such as a music database from Spotify or amazon music. When it comes to the legal, social, and

ethical design aspects of my project I want to go through them all separately. They are important to address in this context.

For the Legal aspects I think the main thing to address is the fair use of all the music included in the project. Most music nowadays is free for most people to use fairly, hence why it is included in the majority of music applications. There will be certain songs and artists that may be signed off to only a few labels or companies meaning only they get the rights to use them. If these songs are used in my program, it is important to recognise the original owners and artists. When it comes to developing the webpage, I must ensure the assets that I use, such as images, text lines, names and statistics are copyright free and allow for air use. Unless I have the license for own parts they cannot be used in my own project. An example of this would be stealing a company's name or using their already created UI/Data and taking it as my own.

The social aspects of my project include having good accessibility and being able to integrate my application into a modern society. Since my project is a music streaming webpage, I don't think I have to worry too much about these impacts but it's good to take it into consideration. Accessibility may just include allowing free access to the webpage, making it easy to logon and view your saved data. The UI must be eye catching and easy to understand, this would intern raise the amount of people who would be able to use my application. (My target audience)

The final aspect to consider is the ethical impacts. This is not a large one to address either. In my development I will only be using a computer to create my application, the only impact I will be having on the environment/ethically will be the way that the power was created to power the computer. Overall, the project will not have any huge ethical impacts and I don't believe there are many aspects to keep in mind.

<u>User stories</u>

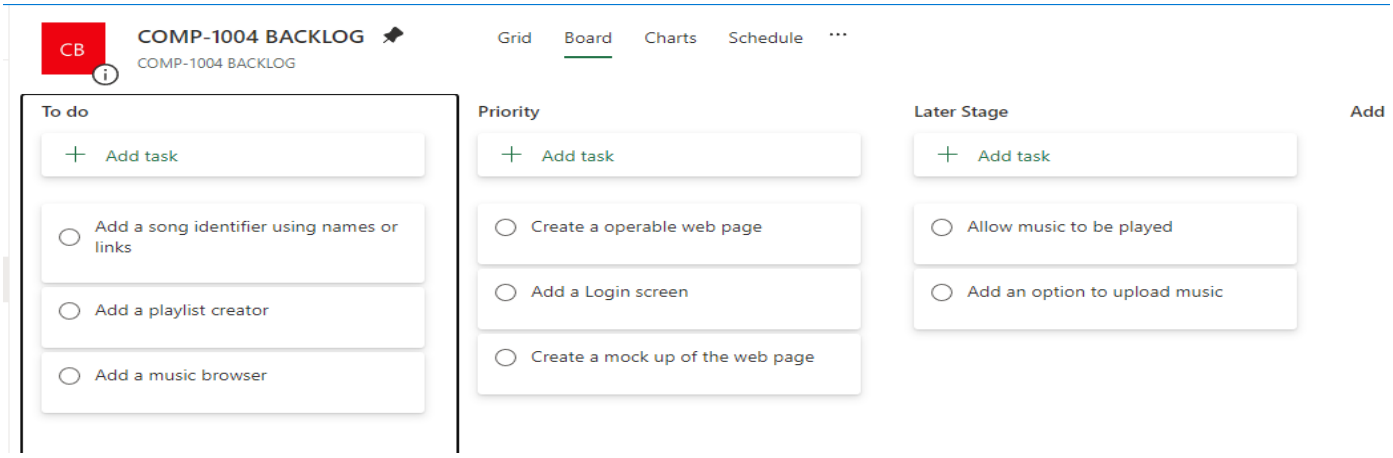
- As a User I should be able to reset my password if I forget or lose it.
- As a User I should be able to easily open and close the webpage.
- As a User I want to be able to log on to the music software, allowing me to see my saved songs/playlists. My audio settings and uploads should be saved to my account.
- As a User I want to be able to search for any released song and play it or download it for offline use.
- As a User I want to be able to create playlists with user inputted titles and descriptions.
- As a User I want to be able to upload my own music, either publicly or have it saved as private for my own use.
- As a User I want to be able to save songs as favourites and "listen for later". These categories should be easily accessible and distinguishable from each other.
- As a User I should be able to sort songs by certain criteria such as date released, artist name and alphabetical order.

- As a User I should be able to input a song link and in return receive a file that outlines the songs data, such as title, album, artist and release date.

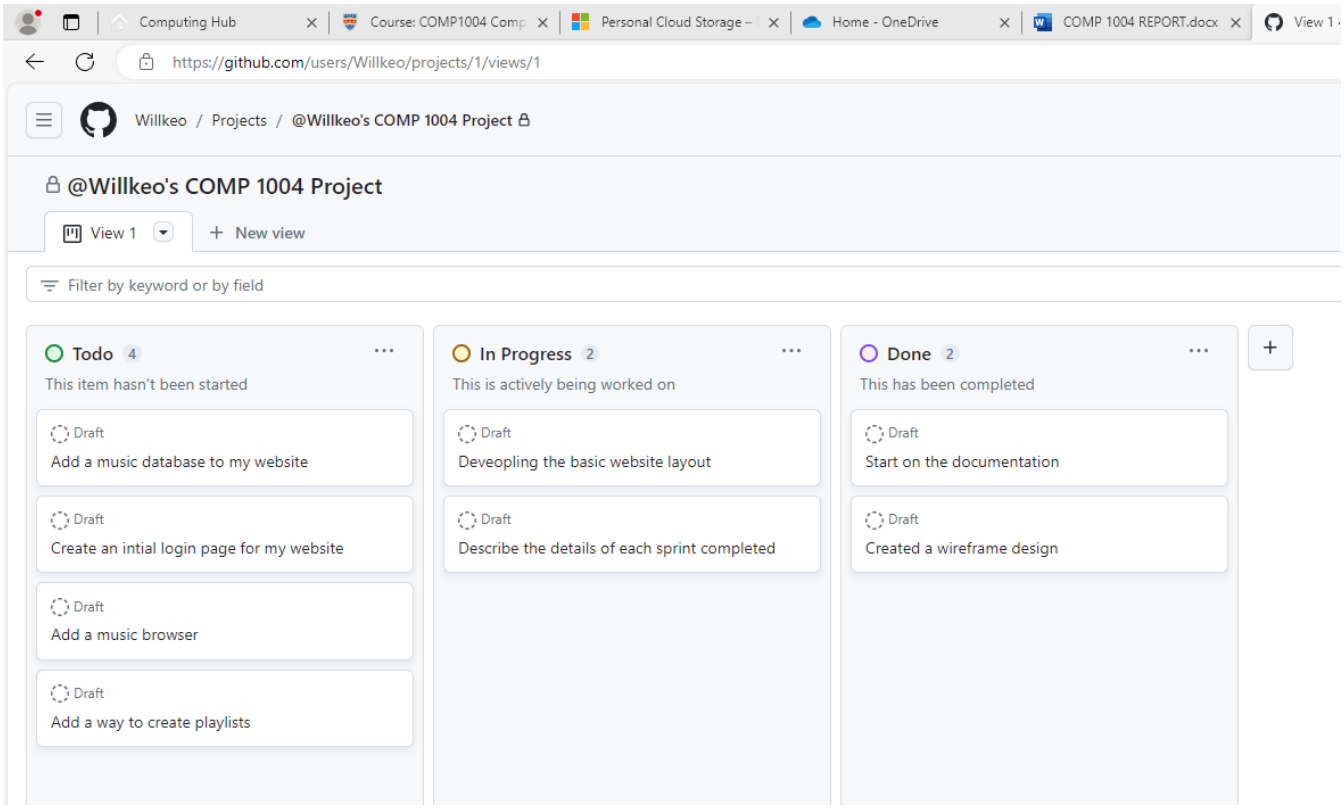
Backlog

The application backlog is part of my sprint planning phase. This will be used to store all my user stories throughout the products development. Including tasks that need to be completed to represent my user stories. There will be lower priority tasks and higher priority ones, this intern helping decide what I need to work on next. They will be a section for “NEED TO DO”, “DONE” and “CURRENTLY WORKING ON.” The backlog should be constantly updated. This backlog is going to be created in Microsoft planner.

A snapshot of what my planner looks like, it may be subject to change, and I will check stages off as I complete them during development to help me keep track.



I have also created a GitHub project that will also act as a backlog and contain the parts of my project that needs to be don’t, is ongoing and has been completed. As I move through my project's development, I will be updating this backlog to not only allow for me to keep track of what I have completed but also so I can go back and describe what I changed during each stage. I will keep track of what parts I complete and what parts still need to be done.



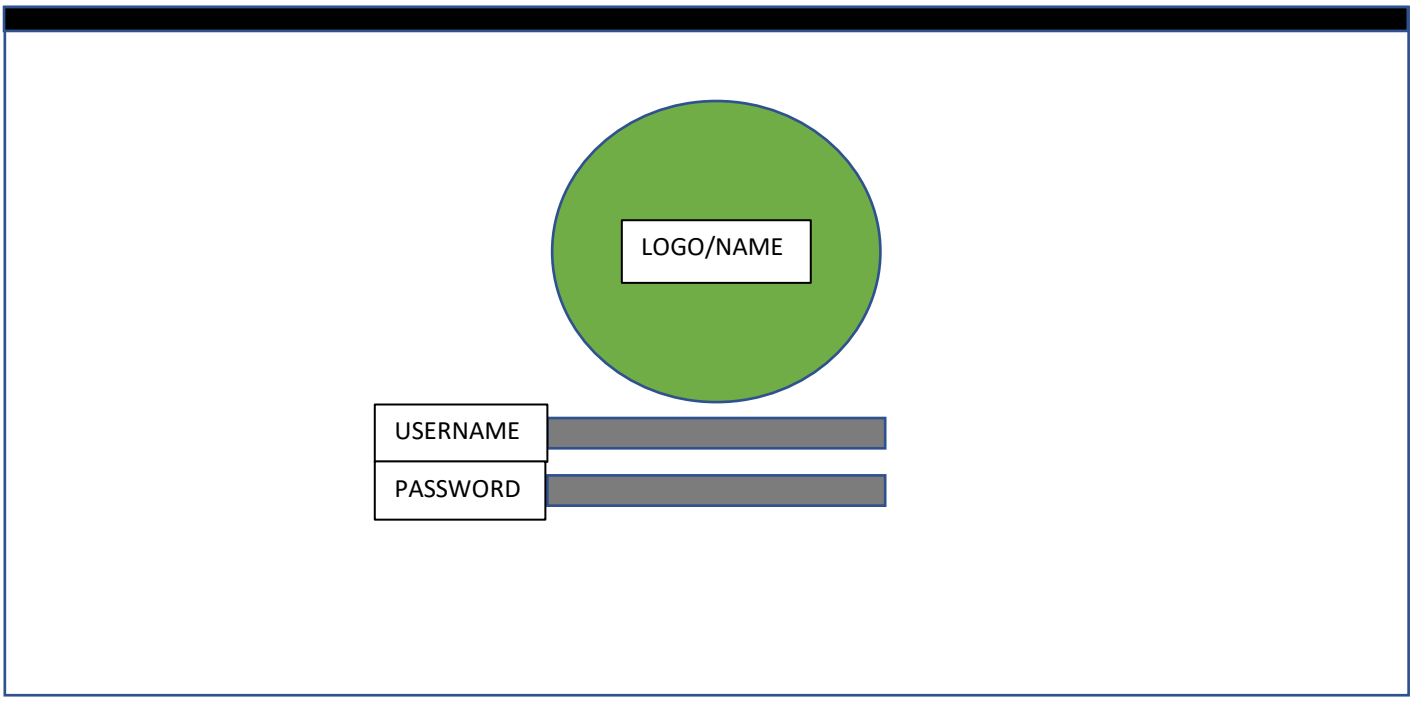
This is the current layout of the backlog I have created in GitHub projects. It has a similar layout to my Microsoft planner one but has more clearly laid out columns showing what needs to be done etc.

Having these backlogs available will also allow me to go back and see what I have completed so far. This will allow me to perhaps update or improve parts that I may not be as happy with. Having a database of what my project includes could also be useful if there was a third party viewing my project or taking over development. It may not be useful in this project alone but in future ones, if I was to give development other to a third party, they could see what has already been implemented.

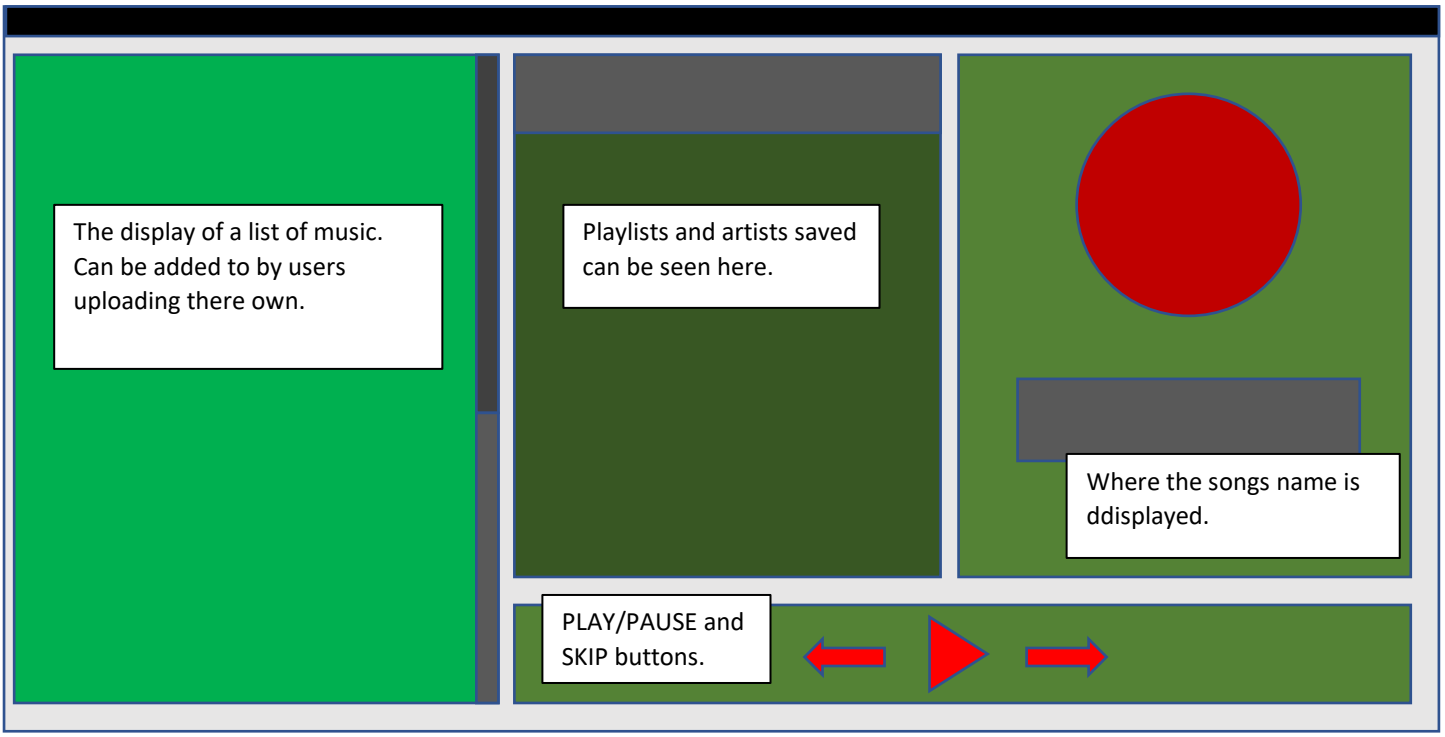
Webpage design

Wireframe

Here I will design a rough layout of what my webpage will look like. I will describe what each part is and its purpose. The design could be subject to change due to unforeseen circumstances, errors etc.



This first design shows the rough layout for how the webpage will first appear, there will be a login screen where users can then first enter their own personal details. This will then allow them to access their data store on the webpage, such as saved songs and playlists.



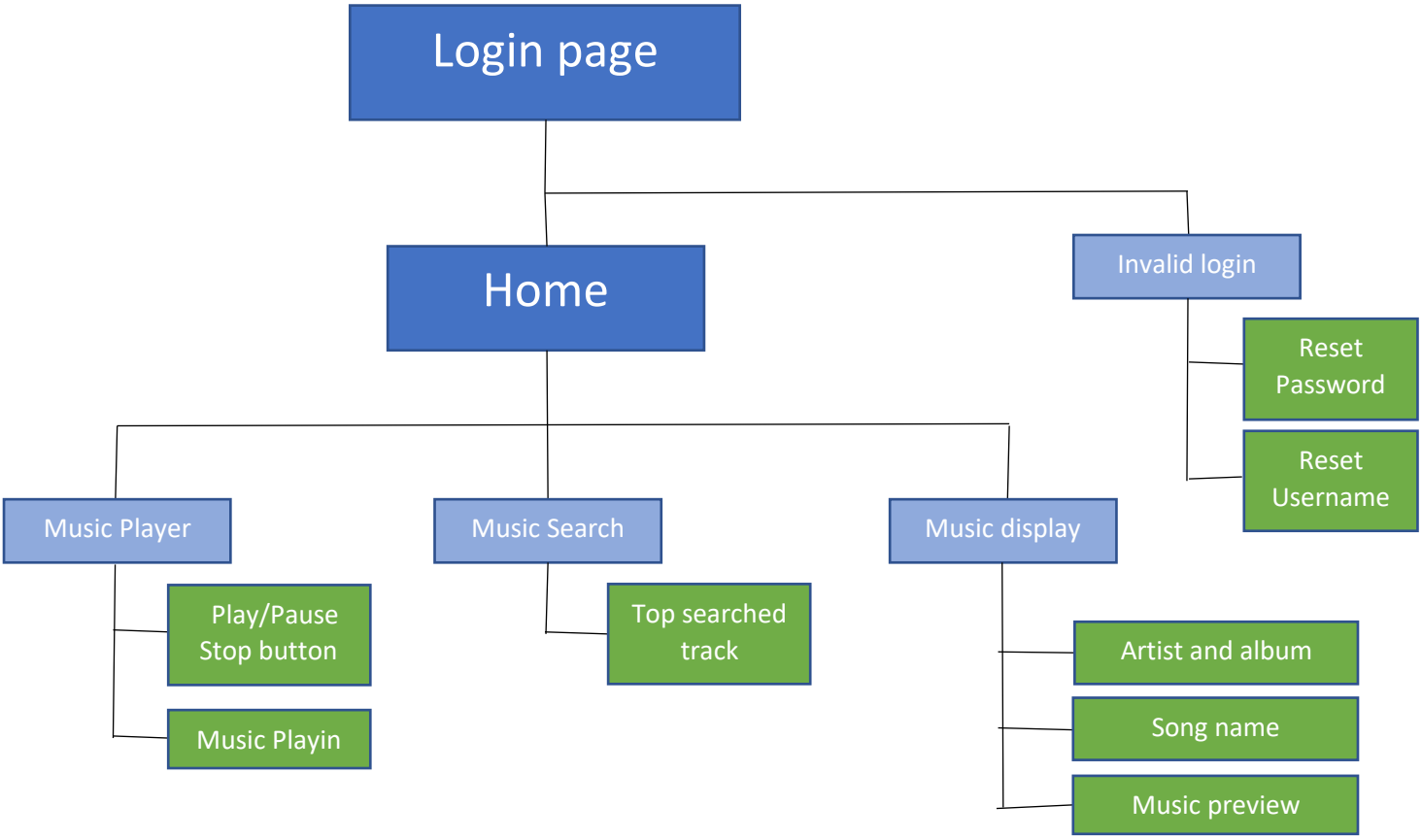
There could be a few issues that arise with this type of design. Due to its simplicity all the features that I planned out may not be able to be included or not disenable due to the lack of separate menus.

Sitemap

A sitemap is a graph designed to show the flow of an application. This sitemap will show how the user moves through my website with all the variables and buttons used. The current graph does not fully represent my website, but also takes into account planned features.

It could be useful to create another one of these when my website is complete to be able to compare them, allowing me another way of seeing what I changed from my plan to my final design

Planned sitemap



SPRINTS

My project will contain multiple sprints during its development. Sprints are two-week periods where you do a certain amount of planning or development, then you review all the content at the end. These sprints can be planned out quite early in development and if you are sure with what you want to do in each one, it can lead to a stable project design and planning stage.

In our projects we are expected to have completed two sprints before Christmas. The number of sprints that we have total varies depending on how long our website takes to develop.

Using Sprints allows for a more efficient development cycle, and it splits it down into more manageable parts. The user/developer can then focus on the things set for that two-week period. Once complete, they can then move on to the next sprint.

SPRINT 1

The first sprint that I did this year was mainly a two period of planning and laying out the details needed to start developing my website. During this period, I decided what sort of website I was going to create and lay out the design stages. I added user stories and described the sort of features I would like to see included. I described the SDLC and then stated what type I will be using for development. I then started on the wireframe design for my website which is a simple shape design that I created to quickly show where all the parts of my website will be and what it will look like. The backlog was therefore created during this period, this intern being used as a basic guide to what needs to get done in each time frame.

Suring these sprints stand up meetings, I discussed what my website would be about and how I would go about creating it. We also discussed the best way to lay out my designs and what the sprints we are doing should contain. We were also told to get our user stories done first so we had development features to work around. I am aware I will not be able to include all of them, but it is useful to think bold.

SPRINT 2

The second sprint before Christmas will include the first stages of my website development such as setting up the code and creating a basic layout. I finalised some of my design documentation and decided on the final wireframes designs after analysing. I am hoping to have made a good start on my website by the end of this two-week period.

At the start of this sprint, I began to properly develop my website by setting up the visual studio code and defining the basic outlines I would be using. I will be coding the webpage in HTML. Adding the basic elements and images into my website is a priority. Even if they are not in the right positions, as long as I can get them implemented, I can move them around and sort the final layout of my website at a later date.

UPDATE (6/12/23)

I have now created the first page of my website, that allows users to login to the music database. The username and password can be changed in the code and the login screen is in a deep green colour scheme, heavily inspired from existing music software. I am currently working on the main website and making sure the login screen works and users are redirected when entering the correct details. I am trying to stick to my original design layout, but I may have to make some changes later on.

I have also started on the main website and made it so the user can now search a database of songs from the Spotify API and then list them on a separate window. Then there is another section of my website that shows the songs details such as the artist, album and a small image of the album cover.

SPRINT 3

After Christmas, more development was started on my website. For now, it has mostly been minor tweaks and making sure my music API is working correctly. I have been using Spotify database for all my music data. This API relies on having an authentication code to allow users to access their API. I'm still planning whether to constantly update the API every time the website is use or switch to an easier music API that may not need as much attention and setup.

During this sprint i would also like to finish my layout for my website even if that doesn't include getting it all working, as long as I have all the windows and basic details in. This will include adding a currently playing tab, including the play/pause button etc. It will also include making sure the layout is correct and works in all formats of the website. My colour scheme is going to stay the same, so I want to be sure it all matches.

I would also like to add the feature to my login page where users can create their own login and passwords. At the moment the basic login is just “username” and “password”. I would like users to have unique ones that are stored in a database. These are then saved as “accounts” that can be accessed when a user enters that unique account details at the login.

Once this has been achieved, I hope to be able to have users' searched songs saved to their accounts. This was one of the things I had in my user stories and something I believe may be achievable.

SPRINT 4

Going into this sprint with not much more done. Alot of what I wanted to do in the Sprint 3 timeframe wasn't completed. The login page is the same and still uses the basic “username” and “password” for the authorised password. The Spotify API still isn't working properly so I am deliberating switching to a different one that might have an easier API system.

I have done some tweaking to the colours and layout, but it hasn't changed much in the last 2 weeks. I am currently trying to implement the extra windows containing the playlists and play/pause buttons. I hope to get these in perfect so then I can say that the main layout for my website is complete, and I can get onto getting all the Java functions working. I want the users to be able to save searched songs to a playlist that they can then name and store. I am thinking that up to 10 playlists can be created maximum to limit clutter on the website and for optimisation.

I don't know whether the functionality of the play and pause buttons will be addressed yet. If they are just there for website cosmetics that will be okay. I will attempt to allow users to play songs through the website using them. It will only be the Spotify previews but that will give users an idea of the song they have searched.

SPRINT 5

Sprint 5 will hopefully be one of my final sprints for the website development. I believe I have almost completed a fully functional website. It now allows for a user to login using a set username and password, where they can then search a song in the database. This will bring up the details of the song and save it to a list. Allowing users to see what they have previously looked up.

The API I used is still Spotify's, it might take some work to get functional, but it has the largest free to use storage of songs. I had to use a token to authorise my use of it and this may need updating in the future. Meaning is it not the most reliable code, but it will work when everything is up to date.

The website design and colour scheme has stayed the same and I am happy with the final look. The website layout may need some adjusting just so it is compatible with most bowsers. This might also help with making some aspects of the website clearer, so small things like increasing the text size or borders of some of the windows. Doing this will also improve the accessibility of the website as it will make it easier for younger people or people with visual impairments to read what's on the screen.

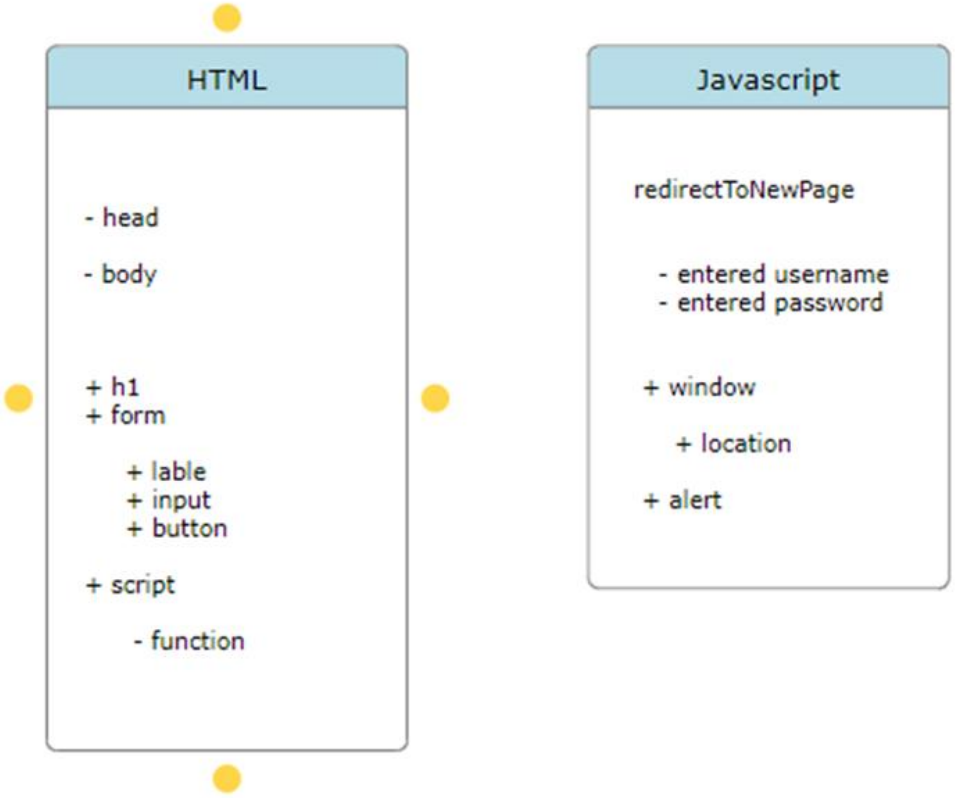
I hope this is one my final development sprints and I can wrap up my website soon.

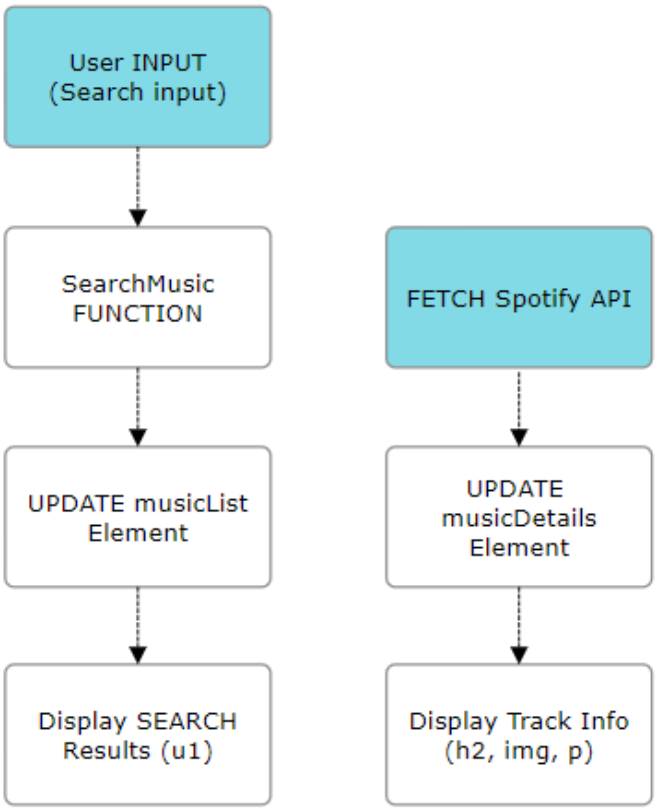
UML Diagrams

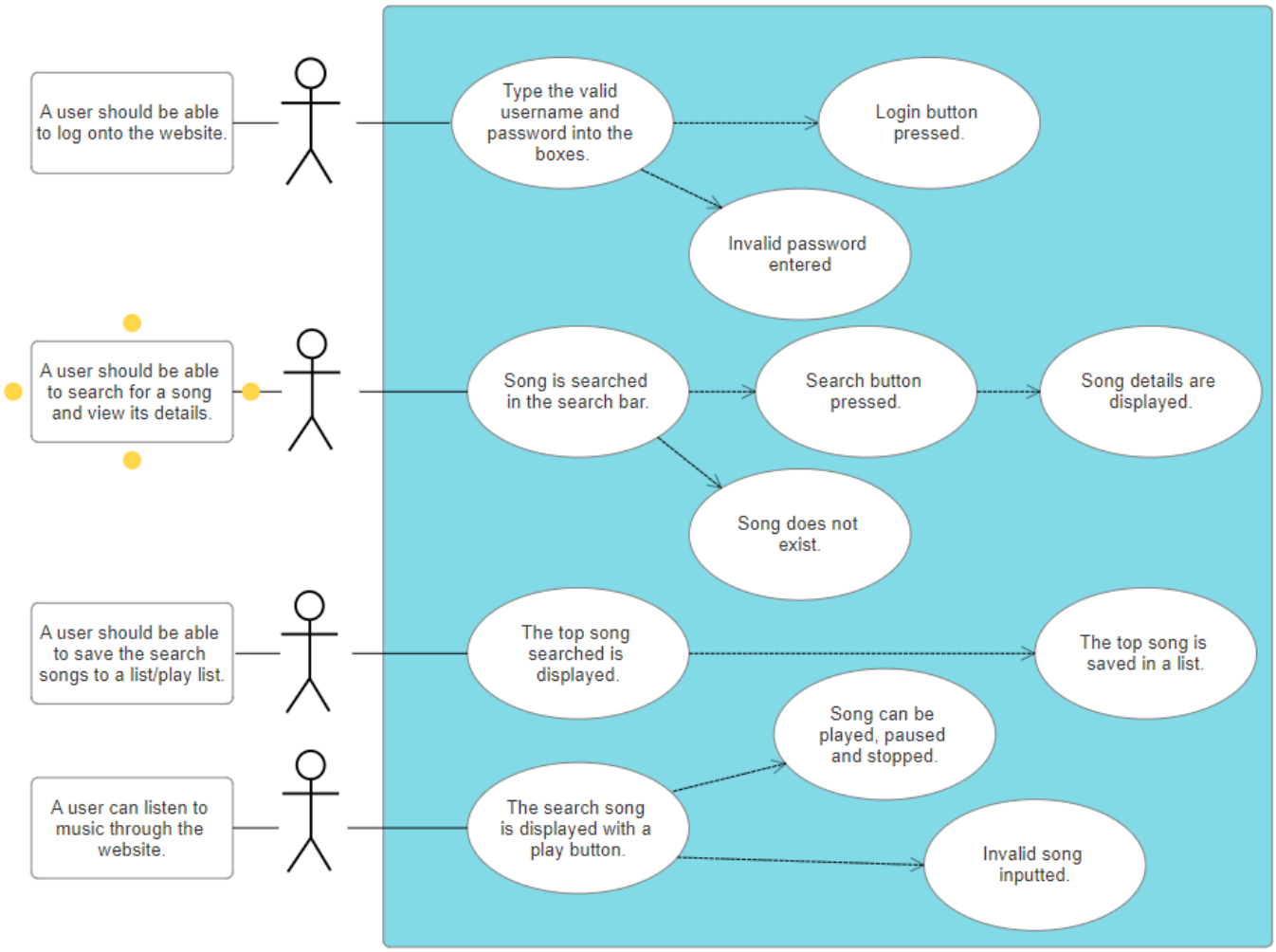
For my project I have created a few UML diagrams. Below are graphs that represent my website. The stickman design is used alongside the User stories to show how users would use the website to complete certain requirements.

Then the other graphs are UML class and activity diagrams, each one representing one of my webpages. One for the login page and one for the main database.

1. Login page
2. Main database
3. Overall website so far


HTML
- head
- body
+ h1
+ form
+ lable
+ input
+ button
+ script
- function
Javascript
redirectToNewPage
- entered username
- entered password
+ window
+ location
+ alert


User INPUT
(Search input)
SearchMusic
FUNCTION
UPDATE musicList
Element
Display SEARCH
Results (u1)
FETCH Spotify API
UPDATE
musicDetails
Element
Display Track Info
(h2, img, p)


A user should be able to log onto the website.
Type the valid username and password into the boxes.
Login button pressed.
Invalid password entered
A user should be able to search for a song and view its details.
Song is searched in the search bar.
Search button pressed.
Song details are displayed.
Song does not exist.
A user should be able to save the search songs to a list/play list.
The top song searched is displayed.
The top song is saved in a list.
Song can be played, paused and stopped.
A user can listen to music through the website.
The search song is displayed with a play button.
Invalid song inputted.