Project Specification:

Problem Domain:

* Current solutions that synchronise music with lighting are inbuilt and often expensive.
* They can be impossible to expand on if the user wishes to add more lights.
* They may also tie a user to a specific brands ecosystem, meaning that they must buy their products regardless of the price.
* Audiophiles looking to enhance the experience of listening to music would love the chance to synchronise their lighting with their music.

How the problem will be addressed:

* Currently there are few solutions that allow a user to easily synchronise music with their wireless smart bulbs.
* This Web Application will initially allow users who use the Wiz brand of bulbs to synchronise multiple bulbs with their music.
* The Web Application will be a user-friendly user interface that allows users to control the synchronisation of their music, they will be able to choose colours, brightness, and intensity.

Proposed system features:

* Authentication – Many bulb brands require authentication, to make the user experience more fluid handling this authentication in the Web Application will be the goal
* Non-Music Control – The Web Application will have the ability to control the lights without music, meaning that the user can set a static lighting.
* Music Control – The Web Application will have the ability to synchronise the music with music being played in the room that the server running the application is in.

Proposed Interface Elements:

* The Web Application is being designed with Mobile and PC views in mind
* The user will be greeted with an interface showing them default settings
* The user will be able to turn on or turn off the music synchronisation setting
* The user will be able to change the lights (brightness, colour, etc.) from the application
* The user will be able to edit the default settings, which would be stored in a database
* The user will be able to create ‘scenes’ which contains a pre-set version of the settings for easy access

Technology Investigation:

Bulb brands:

* + Wiz
  + TP-Link Kasa
  + TP-Link Tapo
  + YeeLight
  + Shelly
  + Philips Hue
  + Athom.tech
  + SwitchBot

From these brands Wiz was selected for these reasons:

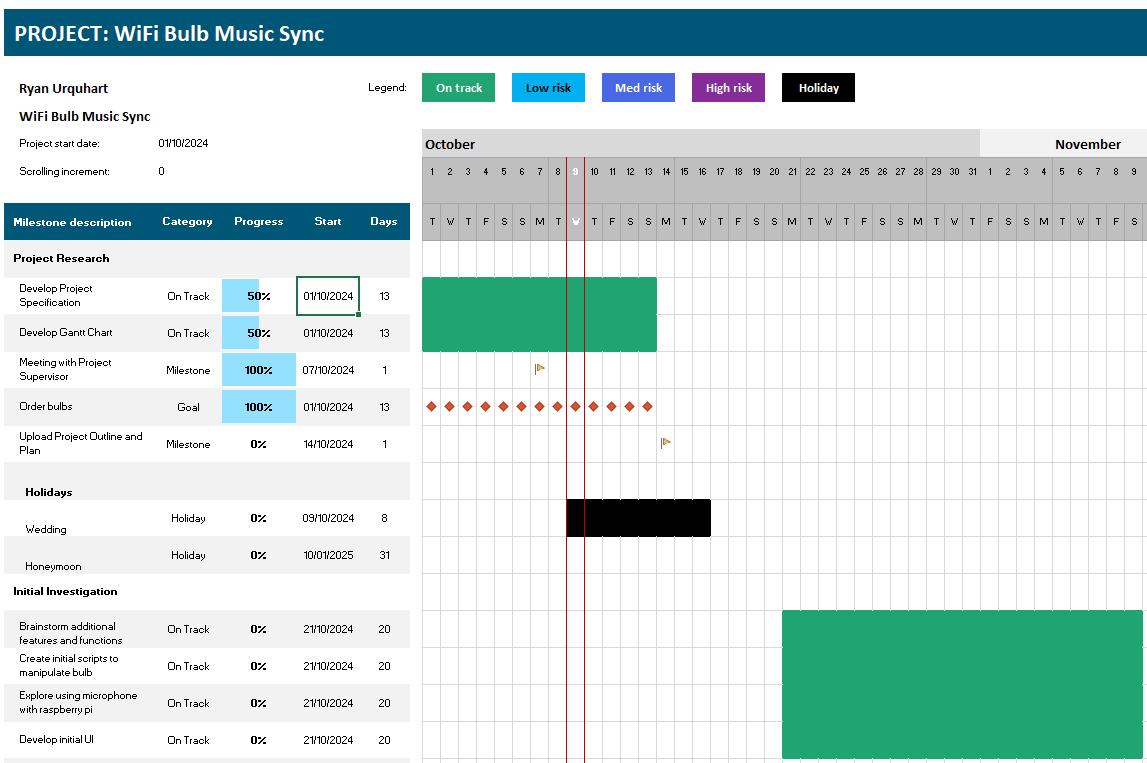
* + Wiz is cheaper than many of the other brands
  + Wiz has an open API unlike some other brands (TP-Link Tapo)
  + Wiz bulbs do not require a hub to work unlike some other brands (Philips Hue)
  + Wiz was also highly recommended by other users for similar use-cases

Frontend Frameworks:

* As I will be working with a raspberry pi I will be working with Linux, Apache, PHP, and SQL
* This is a popular framework, so there is a lot of advice available online

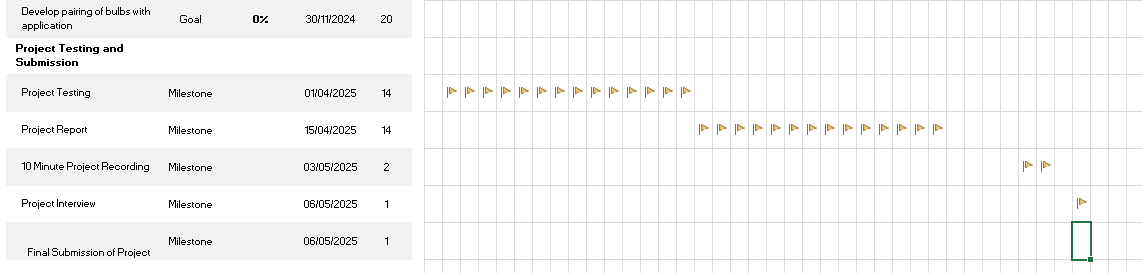
Backend Framework:

* In the backend of the application I will be using Java to run the API calls to the bulbs



A green rectangles and a white background

Description automatically generated



This project will be aiming to complete the following:

Create a solution to allow the synchronization of music with the brightness and colour of Wi-Fi controlled smart bulbs. Beyond this, the synchronization should be customisable, through customising the selected colours, the maximum and minimum brightness, and other possible configu rations.

The imagined users of this project are people interested in using Wi-Fi smart bulbs to compliment the music that they listen to. Given the colour and brightness of the bulbs will be customisable via the developed application then the application would be applicable to various music listeners regardless of preferred genre.

These kinds of users are used to setting up smart home technology, so

This can be achieved with (in order of attempts) a web application, or a software and hardware-based solution (raspberry pi and a microphone), or finally an android application:

1. In the case of a web application, the aim would be to utilise the hardware of the user’s device, this would be the microphone of the user’s phone/pc, then to run the application in browser that would control the local bulbs.
2. In the case of a software and hardware-based solution, this would involve a USB microphone and a raspberry pi, with a web user interface that would allow the user to configure the application.
3. In the case of an android application, it would be an application that would have a user interface for configuration, then would control the Wi-Fi bulbs in the background.

In consideration of these three potential approaches there are some difficulties:

* Option one will be run in-browser, this will likely lead to performance issues, there is also the issue of hosting the site. The hope with it being a web application is that even accessing it remotely it would be able to access lights on the local users’ network. Though a proof of concept is needed before this can be acted upon.
* Option one may also only work as long as the user has the browser open, so may not be suitable for such an application.
* Option two may not be as user friendly as would be ideal, it would require knowledge of setting up a raspberry pi etc.
* Option three would work well, however, it is something that I would find hard as I have never developed an android application before.

Initial Research:

To begin I picked the hardware I was going to be working with, this meant looking through forums to decide which bulbs had been used for similar projects in the past. From these forums I extracted a list of bulb brands from which I could choose:

* Wiz
* TP-Link Kasa
* TP-Link Tapo
* YeeLight
* Shelly
* Philips Hue
* Athom.tech
* SwitchBot

Of these manufacturers I have decided to try using the Wiz Wi-Fi smart bulbs. The reasons for this was, they were highly recommended on the forums I visited, they have a significant amount of documentation online, and they are cheap enough that should I need to order more than one I can. Another reason that I chose these bulbs over others was that these bulbs do not need a hub to function, unlike other bulbs such as Philips Hue. While the hub is useful in decreasing latency, and in linking together multiple bulbs without much interaction from the user. They do add another charge for the project and remove some control from the application and the end-user.

Notes

Create an application/web app/or software solution to allow Wi-Fi bulbs to be synced up with music playing. Either locally or remotely.

Primary goal, web app, uses microphone from device accessing site, then syncs Wi-Fi lightbulbs with the input from that microphone. (Confident in this, worry about access of the devices microphone, as I am not sure how to do that, also worried about performance. Running almost completely in the browser would not be ideal.. probably)

Second thought is to create a more hardware based solution, either using microcontroller with a microphone, or using a raspberry pi with a microphone for the same effect. (I have the most confidence in this)

Finally, if the above fails then perhaps an android app, though I am unused to android studio and have struggled to upgrade android apps in the past.

List of products looked at:

Wiz (Seems good) - <https://docs.pro.wizconnected.com/#authorisation>

Decided to go with WiZ, cheap, and solid API docs, with js examples, which suits the primary attempt of this project

TP-Link Kasa (Discontinued)

TP-Link Tapo (Closed API)

YeeLight (Seems good) - <https://gitlab.com/stavros/python-yeelight>

Shelly (Wrong type of bulb for me, otherwise good)

Philips Hue (Needs a hub to be used, bulbs seem to be zigbee connected to the hub, expensive)

<https://www.athom.tech/> (Tasmota flashed smart bulbs, API needs some figuring) - <https://tasmota.github.io/docs/API/>

SwitchBot (Seems good, though experience with them makes me think long delivery time) - <https://github.com/OpenWonderLabs/SwitchBotAPI?tab=readme-ov-file#color-bulb>

Notes on JS audio capture: <https://stackoverflow.com/questions/27846392/access-microphone-from-a-browser-javascript>