Smart Tabs Application Window Formatter

Project Engineering

Year 4

Tony Leonard – G00372842

Bachelor of Engineering (Honours) in Software and Electronic Engineering

Atlantic Technological University

2021/2022

SMART TABS

Declaration

This project is presented in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Honours) in Software and Electronic Engineering at Galway-Mayo Institute of Technology.

This project is my own work, except where otherwise accredited. Where the work of others has been used or incorporated during this project, this is acknowledged and referenced.

Tony Leonard

____

Table of Contents

1 Summary

The purpose of the Smart Tabs Application Window Formatter is to allow the user to format their desktop the way they want it and allow them to view how long they are spending on their devices.

The user first clicks on a shortcut of a .bat file that runs a Python script. The Python script formats the user’s specified application windows that are currently open on the desktop. The mouse tracker code is also started when you click on the shortcut and continually runs while you are using your desktop or laptop. The mouse tracker code sends user data (the data consists of how long the user has used each application) to the Mongo DB database. The user can access the Smart Tabs website (hosted by Amazon Web Services or AWS) by searching for it online. The user can access the activity chart page where they can view how long they are spending on each application (the data from the database is requested by the frontend and displayed on the activity chart). The user can also access the total activity page that has a gauge that displays the total length of time they have spent of their PC.

This project will benefit its users by making their lives easier which will help boost their productivity.

2 Poster

Application Window Formatter

By Tony Leonard - G00372842 Course: BENG (H) in Software and Electronic Engineering

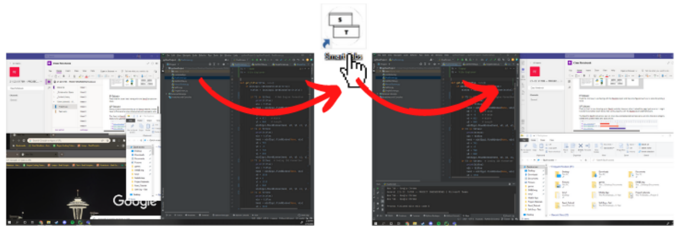
Project Summary

The purpose of the Smart Tabs Application Window Formatter is to allow the user to format their desktop the way they want it and allow them to view how long they are spending on their devices.

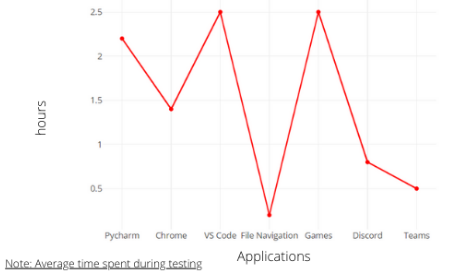
The user first clicks on a shortcut of a .bat file that runs a Python script. The Python script formats the user's specified application windows that are currently open on the desktop. The mouse tracker code is also started when you click on the shortcut and continually runs while you are using your desktop or laptop. The mouse tracker code sends user data (the data consists of how long the user has used each application) to the Mongo DB database. The user can access the Smart Tabs website hosted AWS by searching for it online. The user can access the activity chart page where they can view how long they are spending on each application. The user can also access the total activity page that has a gauge that displays the total length of time they have spent of their PC.

Technologies Used

How it Works

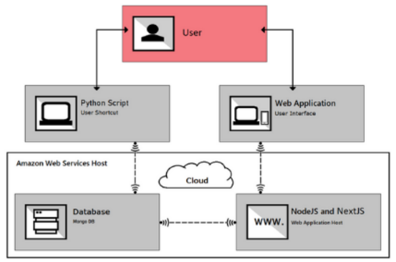


Mouse Usage Time Chart



Note: Average time spent during testing

Architecture Diagram



GitHub Repositories

Report and Misc

Python Scripts

Frontend

Backend

Conclusion

The thing I like most about this project is that I will use it again. My productivity was truly slowed down by the constant unjumbling of the many applications that were running at the same time during my work placement. This project was aimed to combat the messy desktop workspace that a lot of people encounter on a daily basis, and I think that's exactly what it does. I believe with some professional development and testing this project could be used to create a product or service that could be sold on the consumer market. This project helps boost its user's productivity and reduce the fatigue brought on by the constant popup of tabs and applications.

3 Introduction

During my work placement last year, I found myself constantly switching application windows and tabs as I could be navigating through a dozen at a time. This caused some frustration and I felt that there had to be an easier way to unjumble all the applications that were scattered around the desktop. This is when my final year project idea was born. As soon as I started my final year in college, I quickly began researching the best way to solve this predicament.

The way I create the application window formatter is by making a shortcut from a .bat file that runs a Python script. This script uses the win32gui libraries [1] to format the specified application windows and assigns them at different positions on the desktop workspace. This is done by moving the windows to the x and y co-ordinates on the desktop and then specifying what their length and width should be in pixel count.

The second part of the project is to implement another part of the Python script that tracks how much time you spend on each application. This part works by using code that tracks the cursors initial click on each application and starts a counter that increases every second and stops when a different application is clicked. Counters begin at zero and start increasing for every corresponding application that is clicked on, the counter resumes if you click on one of the previous applications that you clicked on beforehand. The data that is recorded from the script is then sent to a MongoDB [2] database which holds the values for how much time the user has spent on each application. These values are then pulled from the database and put on a chart that is displayed on my website which shows the time spent on each corresponding desktop application.

The website allows you to check how long you are spending on individual applications and the total amount of time you are spending on your desktop or laptop. By the end of this report, you should have a good idea how this project works and how you can use it to properly to organise your desktop workplace so it can be as time efficient as possible.

4 Project Tools and Coding Languages

This section describes why I chose the coding languages, IDEs, and different tools that my project is made with.

4.1 Python

Prior to my work placement at Intel, I had not touched anything with Python since the first semester of college. For the first few weeks of the internship, I was given a range of trainings and tutorials based around Python and I picked it up very quickly from there. Python was the sole coding language I worked with during my eight-month work placement. I decided I would stick with it and create my project through Python as I was now familiar with it, and I also found it quite interesting with the range of libraries that were associated with it.

4.2 PyCharm

PyCharm [3] Community Edition was the Python IDE I used while I was working at Intel. Since I was a registered student, I had access to the premium version of PyCharm. However, since it is paid software and not a free to use service, I was not allowed to use it during my work placement. PyCharm was recommended to me by my placement buddy, and it served me well during my placement. Since I was already familiar with PyCharm I decided I would use PyCharm Ultimate Edition for the Python programming part of my project.

4.3 Visual Studio Code

Visual Studio Code [4] or VS Code for short is a source code editor made by Microsoft for windows. Prior to my fourth year of college, I had not used VS code before. I was familiar with Visual Studio while I did modules with C++ but never encountered VS Code. It is safe to say I use VS Code every day at this point, it is now my go-to application for opening any file that contains code or code related files. I use VS Code for my frontend and backend parts of my final year project with the aid of other software of course.

4.4 React

React [5] or ReactJS is a frontend Library for JavaScript. I initial started using React in parallel with my Cloud Computing module as my default library for frontend development on my final year project, as well as my mini project for cloud computing. I worked with React to create my frontend website which I demonstrated for my Project Engineering Demos at Christmas. Once semester two started our Cloud Computing lecturer explained that he would prefer us to work with Next JS as it is a faster frontend library and can implement all the React libraries with little fuss.

4.5 Next JS

Next JS [6] like React is a frontend Library for JavaScript. As mentioned above Next JS is indeed a faster and more reliable frontend library than React. I initially thought it would not make much of a difference but after I did some website testing using Google's Lighthouse [7] the results are in fact night and day. The score I obtained from Lighthouse using Next JS cannot even be compared to the results I achieved from React. The project conversion from React to Next JS went very smoothly with very little hicks. It is safe to say I will continue using NextJS as my project progresses.

4.6 Mongo DB

Mongo or Mongo DB is a database software that allows a user to store or pull data to/from its database. My first interaction with MongoDB was during my Cloud Computing module in third year. At that time, I did not really understand how it worked but I'm glad to say I have a greater understanding this year and using Mongo has been a breeze so far.

4.7 Node JS

Node JS [8] is a backend environment which allows a user to run code outside of their web browser. The same goes for Node JS as it does for Mongo DB as both software tools go hand in hand. Like Mongo DB my first interaction with Node JS was also during my Cloud Computing module in third year. Unlike my experience with Mongo DB this year, it has taken me awhile to understand how Node JS works as I was encountering a lot of issues at the beginning of semester two. However, in recent weeks I can safely say that my understanding of Node JS has greatly improved, and I look forward to completing my project using Node JS.

5 Project Architecture

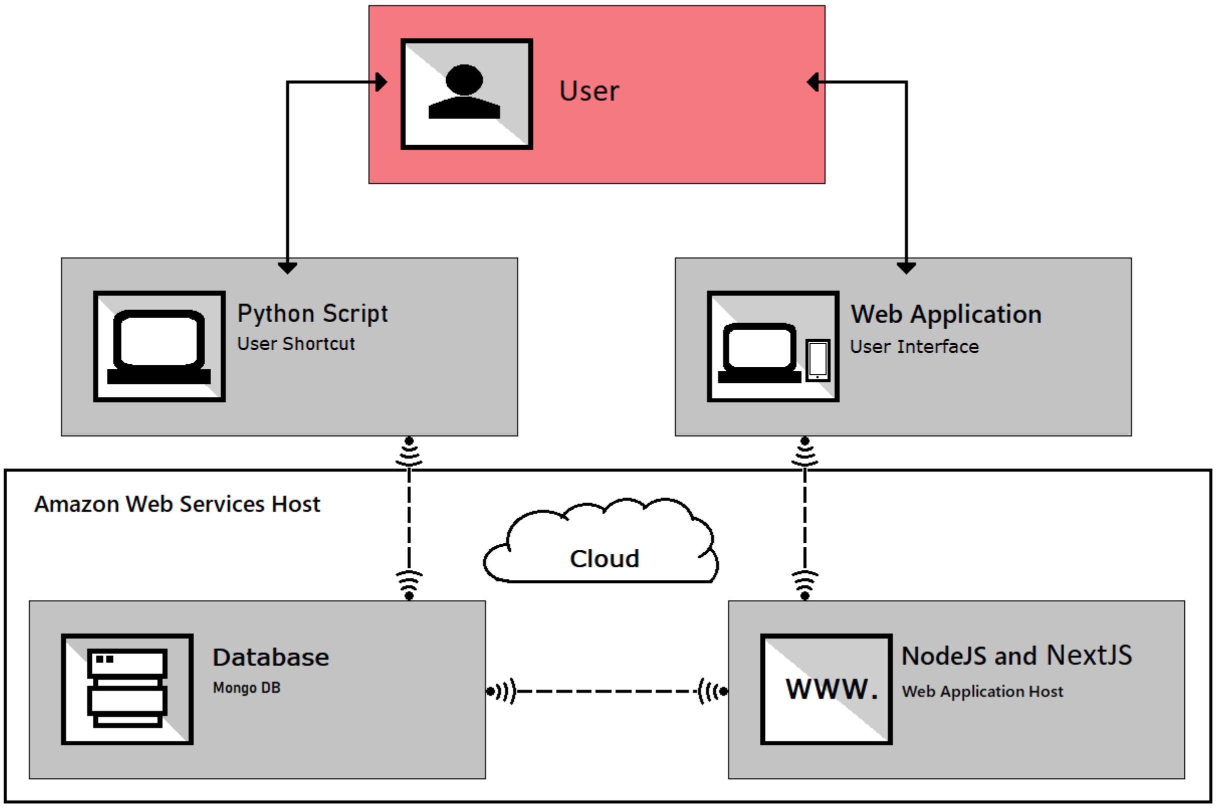


Figure 1. My Architecture Diagram

Project Architecture Overview

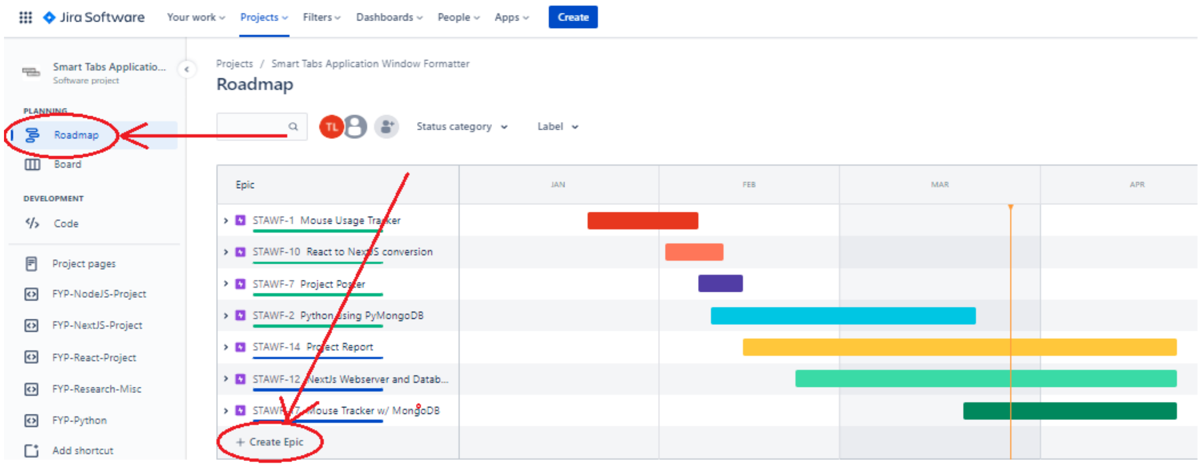
The user first clicks on a shortcut of a .bat file that runs a Python script. The Python script formats the user's specified application windows that are currently open on the desktop. The mouse tracker code is also started when you click on the shortcut and continually runs while you are using your desktop or laptop. The mouse tracker code sends user data (the data consists of how long the user has used each application) to the Mongo DB database. The user can access the Smart Tabs website (hosted by Amazon Web Services [9] or AWS) by searching for it online. The user can access the activity chart page where they can view how long they are spending on each application (the data from the database is requested by the frontend and displayed on the activity chart). The user can also access the total activity page that has a gauge that displays the total length of time they have spent of their PC.

6 Project Plan

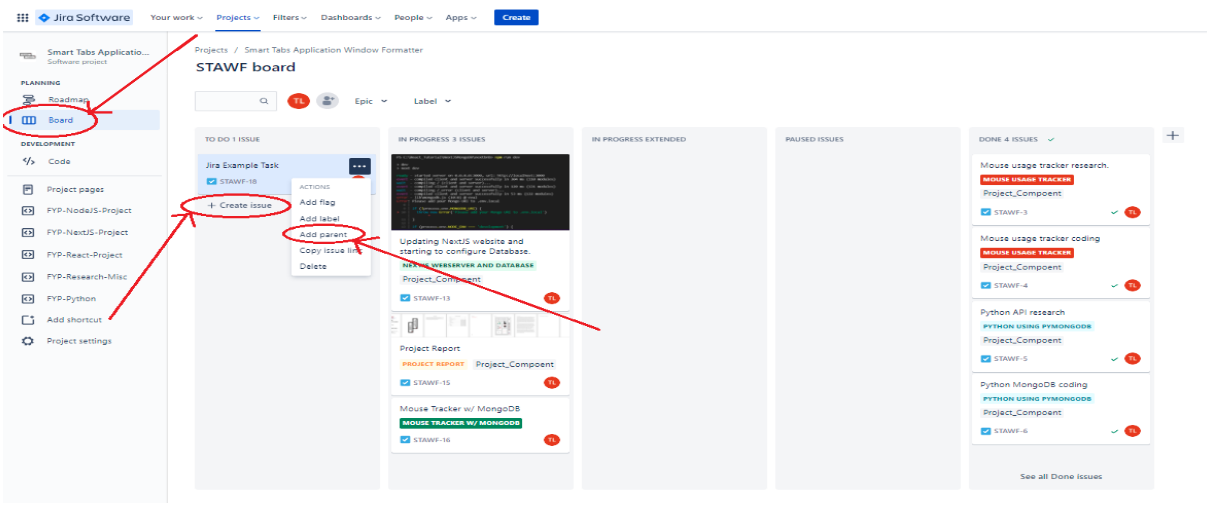
At the beginning of semester two I made an informed decision to start using a project management web tool. There were a range to choose from, but I decided I would use Jira [10] as I had used it during my work placement at Intel. During semester one I kept track of my progress by using the Project Engineering logbook on OneNote. However, in semester two I used both the Jira project management tool as well as my project logbook on OneNote.

6.1 My Experience with Jira

The Jira project management tool is very simple to use. Here is a brief summary of what I would do when I create a new epic and assign new issues to it. You first must create a parent task called an epic, you do this by going to the roadmap tab on Jira. From there you can click and name your epic. After the epic has been created it automatically generates a start date. You can change the start and end date on each epic by clicking it, which will bring up a popup on the right side of your screen.



Once the epic has been created you can then add sub tasks to each epic via the board tab on the left side of your screen. From there you can create sub-tasks for each epic by clicking on the create issue button. You can add each issue(s) to their corresponding epic by clicking on the three dots and then again on the "Add parent" button on the drop-down menu.

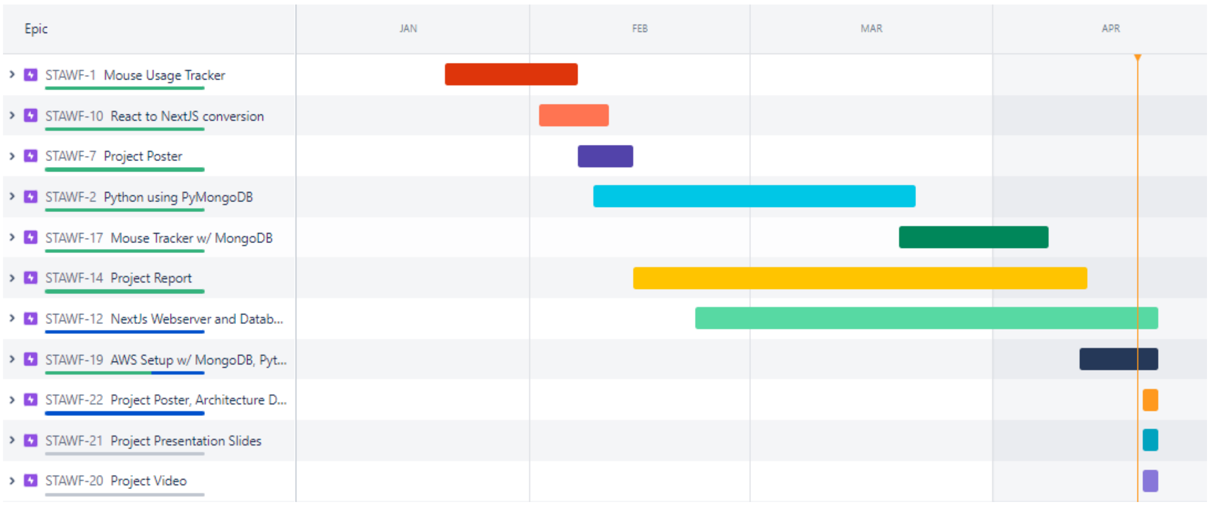


At this point you can click and drag whatever issues you want to start, by dragging them into the "IN PROGRESS" box. Once you have completed an issue you can drag it into the "DONE" box which will then update your epics and your roadmap.

6.2 Project Timeline and Updates

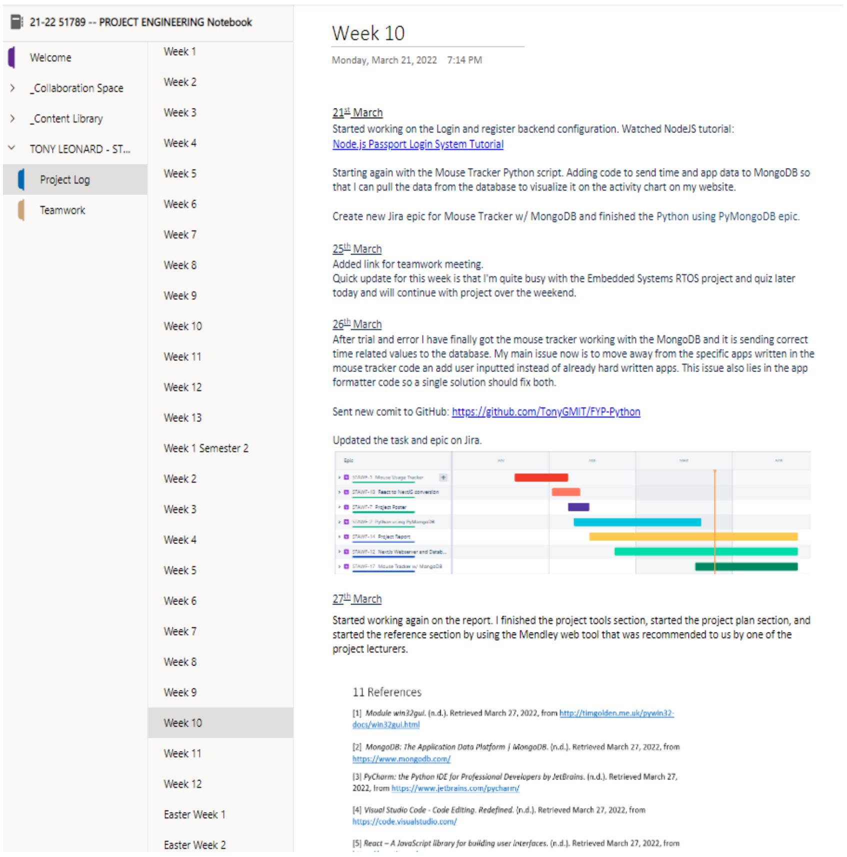
Below is a high-level screenshot of the roadmap and timeline of my final year project over the course of three months. I started planning my roadmap from the 21st of January of this year. The date for the project submission is on the 21st of April. I updated the Jira project management tool about twice a week and updated the project log with timeline updates at least once every week.

I believe the planning and timing went well overall however during the second last week before the deadline a family member passed away. This delayed some of the tasks and ultimately led to one or two of them being left out of the project.



I recorded all of my timeline updates in my project logbook on OneNote as I cannot share my Jira account details with the lecturers.

To the right is a screenshot of a typical week in the logbook. For each day that I carried out work for my project I summarized what I did and wrote it up in the logbook. I also linked all the tutorials and websites I used when I was doing my research for different parts of the project. The logbook also documents all of the blockers and issues that arose over the duration of the college year. The logbook contains numerous screenshots of project components and screenshots of the Jira roadmap as it develops over the course of semester two.



Week 10

Monday, March 21, 2022 7:14 PM

21st March

Started working on the Login and register backend configuration. Watched NodeJS tutorial:
Node.js Passport Login System Tutorial

Starting again with the Mouse Tracker Python script. Adding code to send time and app data to MongoDB so that I can pull the data from the database to visualize it on the activity chart on my website.

Create new Jira epic for Mouse Tracker w/ MongoDB and finished the Python using PyMongoDB epic.

25th March

Added link for teamwork meeting.
Quick update for this week is that I'm quite busy with the Embedded Systems RTOS project and quiz later today and will continue with project over the weekend.

26th March

After trial and error I have finally got the mouse tracker working with the MongoDB and it is sending correct time related values to the database. My main issue now is to move away from the specific apps written in the mouse tracker code an add user inputted instead of already hard written apps. This issue also lies in the app formatter code so a single solution should fix both.

Sent new comit to GitHub: https://github.com/TonyGMIT/FYP-Python

Updated the task and epic on Jira.

27th March

Started working again on the report. I finished the project tools section, started the project plan section, and started the reference section by using the Mendley web tool that was recommended to us by one of the project lecturers.

11 References

[1] *Module win32gui*. (n.d.). Retrieved March 27, 2022, from http://timgolden.me.uk/pywin32-docs/win32gui.html

[2] *MongoDB: The Application Data Platform | MongoDB*. (n.d.). Retrieved March 27, 2022, from https://www.mongodb.com/

[3] *PyCharm: the Python IDE for Professional Developers by JetBrains*. (n.d.). Retrieved March 27, 2022, from https://www.jetbrains.com/pycharm/

[4] *Visual Studio Code - Code Editing. Redefined.* (n.d.). Retrieved March 27, 2022, from https://code.visualstudio.com/

[5] *React – A JavaScript library for building user interfaces*. (n.d.). Retrieved March 27, 2022, from

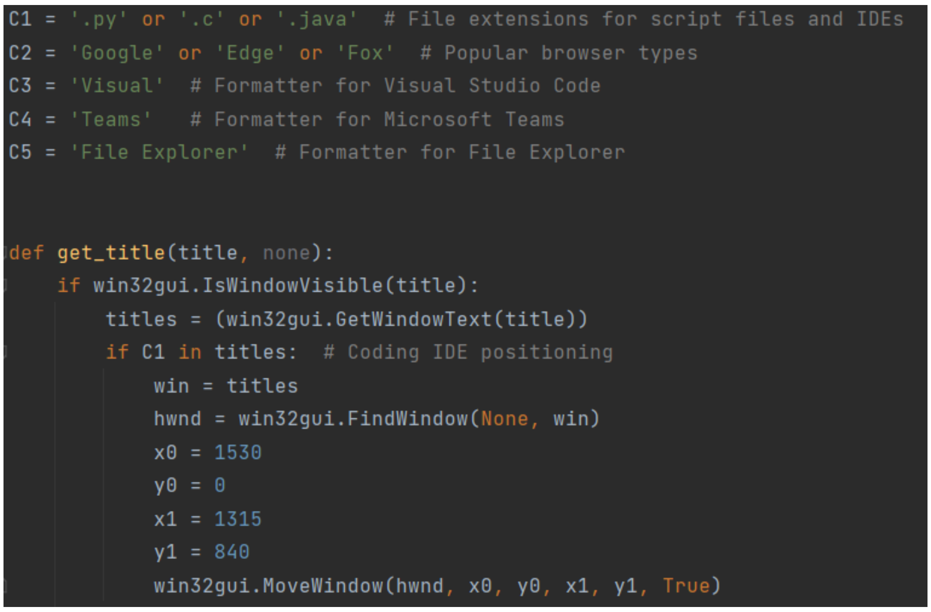
7 Project Description, Design and Code

The Smart Tabs Application Window Formatter consists of various components. These include a Python script, a fronted, a backend, a database and the cloud service AWS.

7.1 Project Python Script

The Python script in my project is perhaps the most important part and took the longest to complete. The script consists of two parts one being the application window formatter and the other being the mouse usage tracker. All of my Python scripts can be found in my Python repository on GitHub: https://github.com/TonyGMIT/FYP-Python

The formatter code uses modules from the win32gui [1] libraries that checks what applications are currently running on the PC. The script specifies which applications it wants to format and if these specific applications are running, they are formatted across the multiple monitors or screens that are in use. This is done by moving the windows to the x and y co-ordinates on the desktop and then specifying what their length and width should be in pixel count. Below is a code snippet that shows the specific applications the script formats and the position for the coding IDE on the screen.

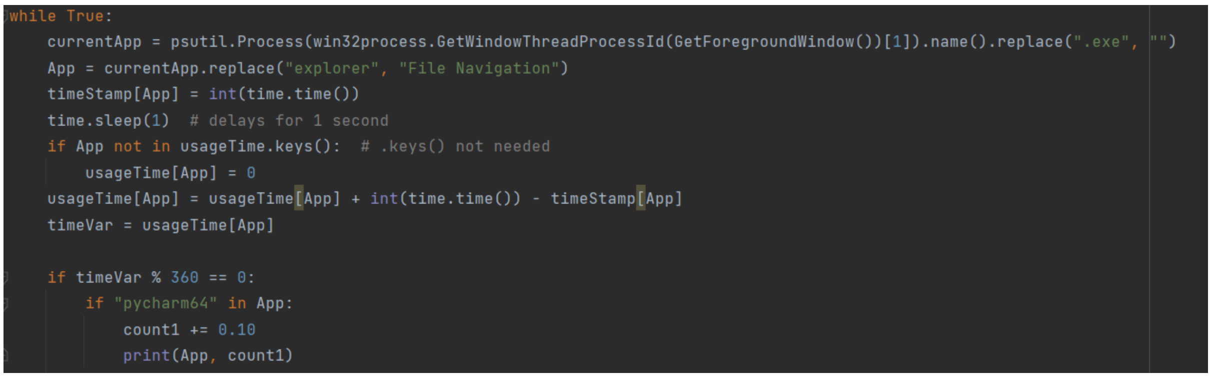


```
C1 = '.py' or '.c' or '.java'  # File extensions for script files and IDEs
C2 = 'Google' or 'Edge' or 'Fox'  # Popular browser types
C3 = 'Visual'  # Formatter for Visual Studio Code
C4 = 'Teams'   # Formatter for Microsoft Teams
C5 = 'File Explorer'  # Formatter for File Explorer


def get_title(title, none):
    if win32gui.IsWindowVisible(title):
        titles = (win32gui.GetWindowText(title))
        if C1 in titles:  # Coding IDE positioning
            win = titles
            hwnd = win32gui.FindWindow(None, win)
            x0 = 1530
            y0 = 0
            x1 = 1315
            y1 = 840
            win32gui.MoveWindow(hwnd, x0, y0, x1, y1, True)
```

The original mouse tracker code comes from a tutorial blog post [11], however I have changed and updated the code to suit my needs for my project and only use six lines of code from the tutorial. The mouse tracker uses more libraries than the formatter code. The script uses the “psutil” [12] library to receive information on running processes or applications on the pc. The “time” [13] library allows the script to manipulate the Unix clock time values to record each of the applications usage in real time. The “win32process” library like the psutil library is used to identify processes and window applications. The “pymongo” [14] library is also used in the mouse tracker code to allow the script to communicate with the MongoDB database. I will discuss pymongo and MongoDB in more detail later in this section. Lastly the like the formatter code the mouse tracker uses modules from the win32gui library.

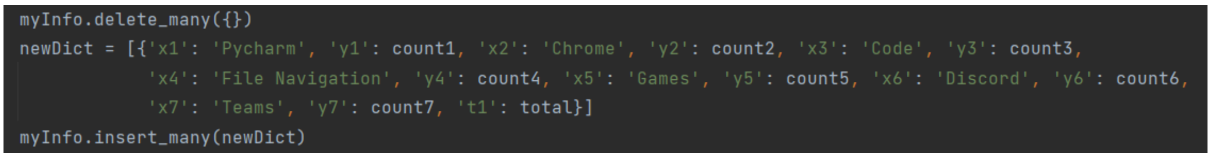
The code works by using modules from the libraries to find all the running applications on the PC and assigns them a time value that increments every second for whatever application the user is currently click on. Once the user clicks on a different application the counter stops for the previous application and start a new counter for the present application that the user has clicked on. The code I added allows the script to record specific apps that I commonly use. It increments a separate count for each application and sends the data to the database after a set amount of time has passed. The script deletes all the previous data in the database each time each time it is updated. The code snippet below shows the code that finds the applications running on the pc and assigns the time counter to each of them. When "timeVar" has reached six minutes of use for each specified application it increments a counter of 0.1 that gets sent to the database and then is displayed on a line plot on one of the webpages on my website.



```
while True:
    currentApp = psutil.Process(win32process.GetWindowThreadProcessId(GetForegroundWindow())[1]).name().replace(".exe", "")
    App = currentApp.replace("explorer", "File Navigation")
    timeStamp[App] = int(time.time())
    time.sleep(1)  # delays for 1 second
    if App not in usageTime.keys():  # .keys() not needed
        usageTime[App] = 0
    usageTime[App] = usageTime[App] + int(time.time()) - timeStamp[App]
    timeVar = usageTime[App]

    if timeVar % 360 == 0:
        if "pycharm64" in App:
            count1 += 0.10
            print(App, count1)
```

The code snippet below shows the array that is sent to the database containing the names of all the specified applications and their corresponding count variables.



```
myInfo.delete_many({})
newDict = [{'x1': 'Pycharm', 'y1': count1, 'x2': 'Chrome', 'y2': count2, 'x3': 'Code', 'y3': count3,
            'x4': 'File Navigation', 'y4': count4, 'x5': 'Games', 'y5': count5, 'x6': 'Discord', 'y6': count6,
            'x7': 'Teams', 'y7': count7, 't1': total}]
myInfo.insert_many(newDict)
```

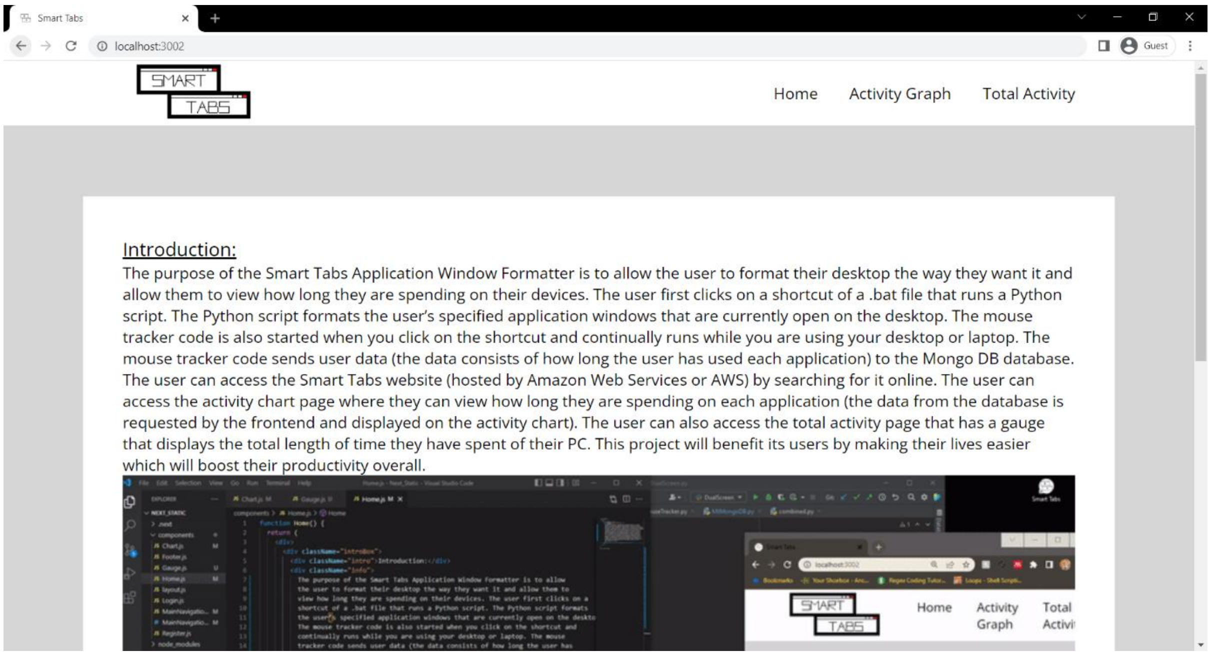
7.2 Project Frontend

Initially the frontend was done using React.JS but when I stared semester two my project supervisor told us about Next.JS. As mentioned before Next.JS is a frontend library that is much faster than the React library and achieves a far greater score on Google Lighthouse. The conversion from React to Next was very straight forward and didn't take very long. All the installed React libraries work seamlessly with Next.JS framework. The CSS style sheets, and html code all stayed the same in the conversion process. The main difference I found with converting my project from React to Next is that there is a lot more components in Next. Mainly you import each of the components into their corresponding pages repeatedly whereas in React you only have to insert the components once in the App.js file. The Next and React repositories can be found on my GitHub:
https://github.com/TonyGMIT/FYP-NextJS-Project

https://github.com/TonyGMIT/FYP-React-Project

From a design point of view, I decided I would go the minimalist route for the website. The frontends CSS style sheet colors consist of red, white, black, and a few shades of grey. The exact hex color codes can be found in the appendix.

The home page contains a summary of the project and a short gif that shows how the formatter code operates. The gif shows me clicking on a shortcut and all the unarranged applications are formatted on a single screen.

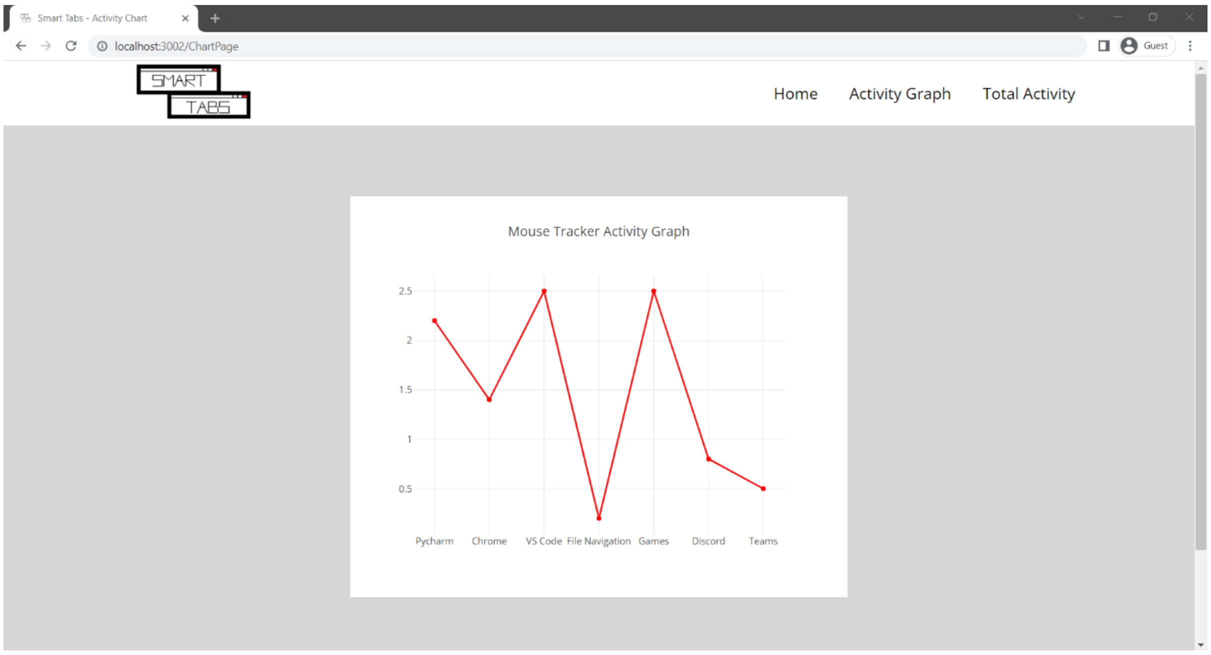


Introduction:

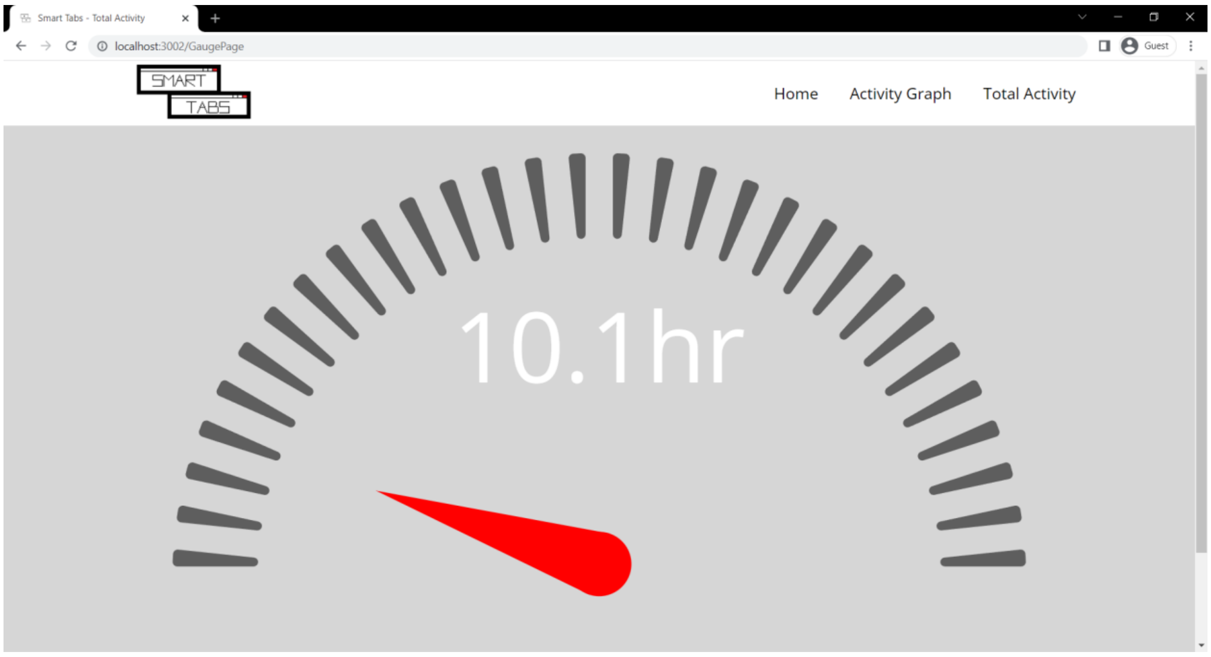
The purpose of the Smart Tabs Application Window Formatter is to allow the user to format their desktop the way they want it and allow them to view how long they are spending on their devices. The user first clicks on a shortcut of a .bat file that runs a Python script. The Python script formats the user's specified application windows that are currently open on the desktop. The mouse tracker code is also started when you click on the shortcut and continually runs while you are using your desktop or laptop. The mouse tracker code sends user data (the data consists of how long the user has used each application) to the Mongo DB database. The user can access the Smart Tabs website (hosted by Amazon Web Services or AWS) by searching for it online. The user can access the activity chart page where they can view how long they are spending on each application (the data from the database is requested by the frontend and displayed on the activity chart). The user can also access the total activity page that has a gauge that displays the total length of time they have spent of their PC. This project will benefit its users by making their lives easier which will boost their productivity overall.

```
function Home() {
  return (
    <div>
      <div className="introbox">
        <div className="intro">Introduction:</div>
        <div className="info">
          The purpose of the Smart Tabs Application Window Formatter is to allow
          the user to format their desktop the way they want it and allow them to
          view how long they are spending on their devices. The user first clicks on a
          shortcut of a .bat file that runs a Python script. The Python script formats
          the user's specified application windows that are currently open on the deskto
          The mouse tracker code is also started when you click on the shortcut and
          continually runs while you are using your desktop or laptop. The mouse
          tracker code sends user data (the data consists of how long the user has
```

The second page on the website contains a line plot chart which displays how long you have spent on certain applications. I used the Plotly.js [15] React graphing library for the line plot. The values on the plot come from the data stored on the database which has been collected by the mouse tracker code.

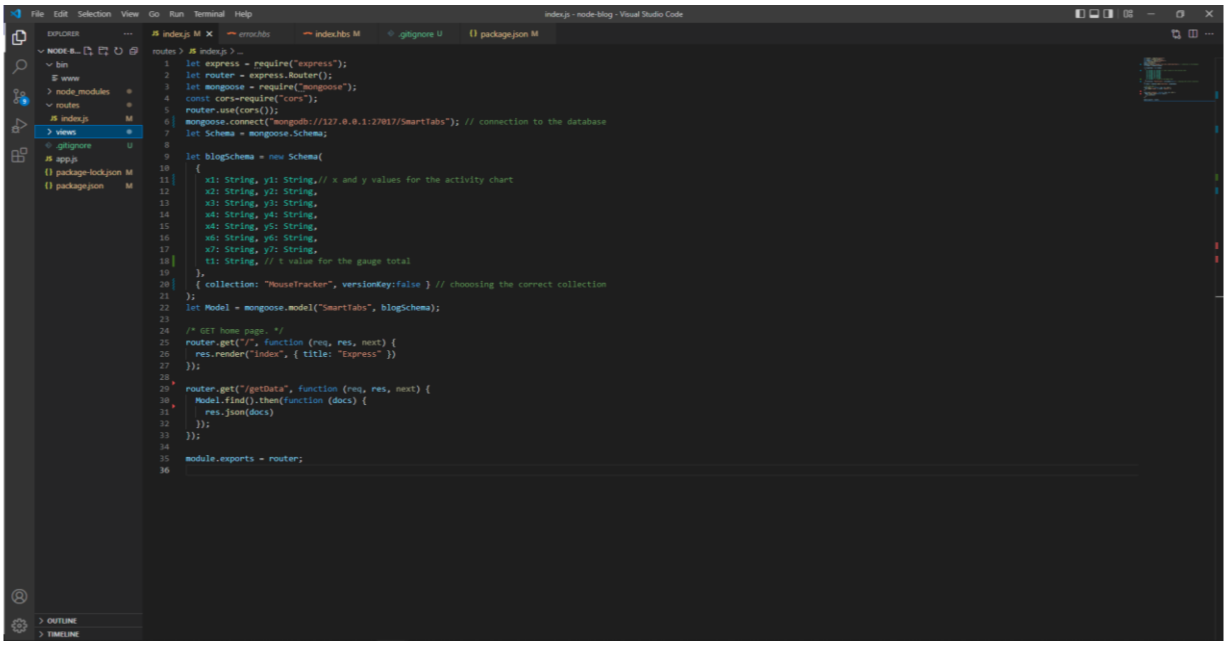


The third page on the website contains a time gauge that displays the total length of time that has been spent on all the applications combined. I used a React library called react-gauge-chart [16]. The value on the gauge comes from the data stored on the database which also has been collected by the mouse tracker code.



7.3 Project Backend

The backend is quite small in comparison to the frontend as the sole reason for it is to send and receive data from the database. We were given a sample Node.js project in our Cloud Computing module. I reconfigured the sample project so it could be implemented into my final year project. I removed all the unnecessary files and added the code for getting the data for the activity chart in the index.js file.



```
let express = require("express");
let router = express.Router();
let mongoose = require("mongoose");
const cors=require("cors");
router.use(cors());
mongoose.connect("mongodb://127.0.0.1:27017/SmartTabs"); // connection to the database
let Schema = mongoose.Schema;

let blogSchema = new Schema(
  {
    x1: String, y1: String,// x and y values for the activity chart
    x2: String, y2: String,
    x3: String, y3: String,
    x4: String, y4: String,
    x4: String, y5: String,
    x6: String, y6: String,
    x7: String, y7: String,
    t1: String, // t value for the gauge total
  },
  { collection: "MouseTracker", versionKey:false } // chooosing the correct collection
);
let Model = mongoose.model("SmartTabs", blogSchema);

/* GET home page. */
router.get("/", function (req, res, next) {
  res.render("index", { title: "Express" })
});

router.get("/getData", function (req, res, next) {
  Model.find().then(function (docs) {
    res.json(docs)
  });
});

module.exports = router;
```

7.4 Project Database

The database I used for my final year project was of course the MongoDB database. I have gotten very familiar with it over the past year and now see it as a great and easy software tool to use. Our Cloud Computing lecturer provided us with a zip folder containing the download wizard for both MongoDB and Robo3T. The setup was very simple as I started the install wizard and once it finished, ran both the executable files for MongoDB and Robo3T. From there I created a new connection on Robo3T. It was as simple as that to connect to the database on localhost.

7.5 Amazon Web Services Host

The cloud service provider I decided to use was AWS. The Cloud Computing lecturer provided us with a student credit account for AWS which had a $100 allowance on it. I didn't use the account much for my cloud computing project so I decided I would use it for my final year project. The procedure in setting up an instance was quite straightforward. First you had to create and name a new EC2 instance. From there you were prompted for what operating system you would prefer. I chose Ubuntu as I was familiar with it from third year and my work placement. From there I choose the default options for the rest of the prompted requirements. Once I finished setting up the instance, I clicked on a checkbox at the top of the screen beside the new instance I had just created. I clicked connect and straight away I was connected to the AWS server. The list of Ubuntu commands for setting up the instance, installing Git, installing MongoDB, and creating and installing both the frontend and backend can all the found in the appendix.

8 Ethics

To the best of my knowledge, I have referenced all of the tutorials, libraries, videos and websites that I used while researching, coding and testing my project. I have used the Mendeley [17] referencing tool to help me correctly reference all the links in this report. I have compiled all of the links that are in this report, and on my logbook into a PDF document that can be found here: https://github.com/TonyGMIT/FYP-SmartTabs-Information/blob/master/References%20for%20Research%20on%20GitHub.pdf

9 Conclusion

The thing I like most about this project is that I will use it again. My productivity was truly slowed down by the constant unjumbling of the many applications that were running at the same time during my work placement. This project was aimed to combat the messy desktop workspace that a lot of people encounter on a daily basis, and I think that's exactly what it does. I believe with some professional development and testing this project could be used to create a product or service that could be sold on the consumer market. This project helps boost its user's productivity and reduce the fatigue brought on by the constant popup of tabs and applications.

To conclude I really enjoyed working on this project. I am grateful to the lecturers whose modules helped me complete different aspects of this project. This project delivers the vast majority of the goals I set myself to achieve. I strongly believe that the work that has gone into this project will aid me in my future endeavours.

10 Appendix

10.1 Frontend CSS Hex Color Codes

Background: #d6d6d6

Header: #ffffff

Footer: #5e5e5e

Chart box: #ffffff

Header title overlay: #ff0000

Login and register buttons: #ff0000

10.2 AWS Setup Commands

Setting up new AWS instance with backend:

```
$ sudo apt-get update

$ sudo apt-get install git

$ mkdir projectBK

$ cd projectBK

$ git init

$ git pull https://github.com/TonyGMIT/FYP-NodeJS-Project.git

$ sudo apt install npm

$ npm install

$ npm start
```

Setting up new AWS instance with MongoDB:

```
$ wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -

$ echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
$ sudo apt update

$ sudo apt install -y mongodb-org

$ mongod –version

$ sudo systemctl start mongod

$ sudo systemctl status mongod

$ sudo systemctl stop mongod

$ sudo systemctl restart mongod
```

Setting up new AWS instance with frontend:

```
$ mkdir projectFT

$ cd projectFT

$ git init

$ git pull https://github.com/TonyGMIT/FYP-NextJS-Project.git

$ npm install

$ npm run dev
```

11 References

[1] *Module win32gui*. (n.d.). Retrieved March 27, 2022, from http://timgolden.me.uk/pywin32-docs/win32gui.html

[2] *MongoDB: The Application Data Platform | MongoDB*. (n.d.). Retrieved March 27, 2022, from https://www.mongodb.com/

[3] *PyCharm: the Python IDE for Professional Developers by JetBrains*. (n.d.). Retrieved March 27, 2022, from https://www.jetbrains.com/pycharm/

[4] *Visual Studio Code - Code Editing. Redefined*. (n.d.). Retrieved March 27, 2022, from https://code.visualstudio.com/

[5] *React – A JavaScript library for building user interfaces*. (n.d.). Retrieved March 27, 2022, from https://reactjs.org/

[6] *Next.js by Vercel - The React Framework*. (n.d.). Retrieved March 27, 2022, from https://nextjs.org/

[7] *Lighthouse | Tools for Web Developers | Google Developers*. (n.d.). Retrieved March 27, 2022, from https://developers.google.com/web/tools/lighthouse

[8] *Run JavaScript Everywhere.* (n.d.). Retrieved March 27, 2022, from https://nodejs.dev/

[9] *Cloud Computing Services - Amazon Web Services (AWS)*. (n.d.). Retrieved March 27, 2022, from https://aws.amazon.com/?nc2=h_lg

[10] *Jira | Issue & Project Tracking Software | Atlassian*. (n.d.). Retrieved March 27, 2022, from https://www.atlassian.com/software/jira

[11] *Track windows app usage time using python. - DEV Community*. (n.d.). Retrieved April 8, 2022, from https://dev.to/tkkhhaarree/track-windows-app-usage-time-using-python-h9h

[12] *psutil · PyPI*. (n.d.). Retrieved April 8, 2022, from https://pypi.org/project/psutil/

[13] *time — Time access and conversions — Python 3.10.4 documentation*. (n.d.). Retrieved April 8, 2022, from https://docs.python.org/3/library/time.html

[14] *PyMongo 4.1.0 Documentation — PyMongo 4.1.0 documentation*. (n.d.). Retrieved April 8, 2022, from https://pymongo.readthedocs.io/en/stable/

[15] *Plotly Open Source Graphing Libraries*. (n.d.). Retrieved April 12, 2022, from https://plotly.com/graphing-libraries/

[16] *react-gauge-chart - npm*. (n.d.). Retrieved April 19, 2022, from https://www.npmjs.com/package/react-gauge-chart

[17] *Search | Mendeley*. (n.d.). Retrieved April 20, 2022, from https://www.mendeley.com/search/