Developing the prototype Task2

Table of Contents

Overview

Toka Fitness needs a new digital solution which would help their company out to communicate with their existing and coming customers as well as helping their employees on completing the tasks. Now Toka Fitness currently provides customers with **personal training sessions, advice about fitness training and advice about healthy living**. But they are now in need of features such as:

- **provide information and advice about fitness training and healthy living**
- **provide access to digital content to support customers with their training and healthy lifestyle**
- **encourage existing customers to use more of the services provided by Toka Fitness**

Toka Fitness has also carried out market research with existing customers to identify potential features which could be included in the digital solution. Potential features included items such as:

- **free and paid-for content**
- **accessibility features for users with sight loss**
- **'social' features**
- **customisable workout and eating plans**

As a result of this, the features they required can be completed and will be done by the time they expect this solution to be completed.

For this project I have decided to create a Web based application for Toka Fitness with functionality they in need of included in it. This application will have a SQL data base system which will store customer information and the necessary data needed.

Changes to development

Payment

Due to the risks of storing users' payment information I have decided to no longer include any fields requiring a user to enter their private card number information to buy a premium account due to reasons such as security and Toka Fitness assumably not having a valid or usable e-commerce license to sell products online.

Storing information

User information will now be stored in a SQL database instead of Excel file. I have decided to make this change as SQL is fast and can handle large loads of data whereas Excel cannot. SQL is a highly secure database as it uses sophisticated encryption algorithms making it virtually impossible to break the security layers. SQL server is a commercial relational database with additional features to reduce the risks of attacks. SQL also has features allowing users to restore and recover lost or damaged data with its advanced tools.

As a result of this I no will be using Excel as a database and instead a MySQL on a virtual machine to get all users data.

Features

Workout Plan changes

Push Up page

Instead of having a workout plan page with drop downs and saving information I have decided to make it more interactive for the users and make them click a button every time they have done a push up. This will then be shown up on a counter telling them how many push ups they have done. Once user is finished, they will need to press save button to save the amount of push ups they have done to the database.

Squats Page

This page will act the same as Push ups page with the interactives of clicking a button and showcasing the amount of squats user has done which will then allow user to save it to the database.

Records Page

Data such as Push ups and Squats will be stored in a table which user will be able to see on how many they managed to achieve within the log. This will then allow users to share their information online to social media such as Facebook, Instagram and twitter. As a result of this socials page has been removed as this page will act as a social feature which Toka Fitness customers are wanting.

Socials Page

Socials page has been removed and replaced by Records page social media share buttons.

Development methodology/plan

To build this solution which Toka Fitness needs I have decided to use the agile approach to develop this product. As a result of this instead of developing everything in one go, I will deliver work in small increments such as sprints. The purpose of sprint planning is to set goals which can be delivered within a short, time boxed period. Using sprints for this project is perfect as it will allow me to give out functionality by the end of each sprint. This means the milestones set for the sprint are made and ready for the review of the client to see if its what they expect each function to work as it is. This is another advantage to using agile methodology due to always being able to have the clients themselves involved in the project and if they have any new features they want to add or remove it can be easily achieved.

When developing this product, I will be splitting this big problem into sprints with each sprint being a day to itself. This allows me to have my full attention and work to be completed for that specific sprint. This table below will be the layout which my sprints will be held in. Each sprint will have set tasks to be completed within it which will act as milestones. This will allow me to keep track of tasks which need being completed as well as giving time frames. If I don't manage to complete a certain task in (Sprint 1) it will be moved into (Sprint 2) to make sure it gets done one way or another. Each task in different Sprints will be given a number which would represent the difficulty which would make it to be completed a bit longer than the other vice versa for tasks which have lower number set.

Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6

At the end of each sprint, I will review on what I have produced and achieved while I was developing the set tasks. The review will contain information such as the tasks in the sprint I was able to achieve, the tasks didn't complete or had trouble on creating them, the tasks which need to get moved to the next sprint to be completed in and review on what I think about the sprint in general. This all then will be repeated for each end of the sprint as it will give me a clear overview of where I'm at and any more things which need testing or fixing.

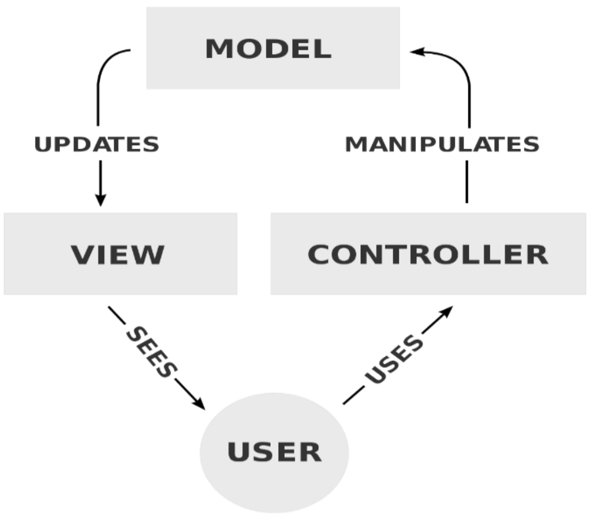
Throughout this development I will be using a wide range of testing including White box testing and Black box testing. As I'm the one building this software Black box testing will be a priority and frequently done to test out the interface and functionality. Testing such as:

- unit testing
- integration testing
- system testing
- acceptance testing
- boundary testing

All these tests will be done in the log table to make sure that no broken/non functionality is being give to the client as well as lets me know if any future development will be required on completing the product.

MVC (1)

MVC(Model-View-Controller) is pattern which I'm using for this project as it separates an application into three groups of components: Model, View and controller.



User requires are routed to a controller which is responsible for working with the model to perform user and retrieve results of queries. The controller chooses the view to display to the users and provides it with any Model data it requires.

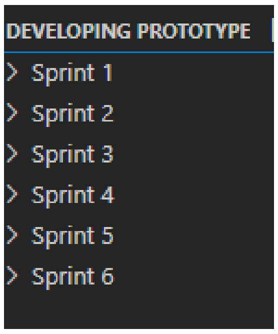
Both view and controller depend on model, however, benefit of this is that model depends on neither the view nor the controller. The benefit of this separation is that model can be built and tested independently.

Responsibilities

Model	Represents state of the application and any logical operations that should be performed by it
View	Responsible for presenting content through the user interface. There should not be any logic within view.
Controller	Handles user interaction and works with the model and selects a view to render

Version Control

As this project development will take multiple sprints I have to make sure all the files are backed up by the end of each day and the next day I copy all the files from the previous sprint on to the new one therefore if I ever need to look back on what I did in sprint 1 I will be able to.



Cookies

Will be used as part of my tracking the customer once they login using their email it will act as a cookie which could then be used to get their name so it can be displayed on the dashboard as an interactive thing.

Sprint 1

Sprint 1 will consist mostly of creating main modules like the server where it will be getting requests and sending user to specific page. Model will define what each request and what it should do then returns it to server

Goals/Requirements

Goal Set	FR#	Requirements
As a user I would like to be able to access a website	1	Create a webpage which user can access online
As a user I want to see forms to fill in on the webpage	2	Create forms for user which can be filled in
As a user I want to be able to submit forms	3	Create submit form button to submit form for user

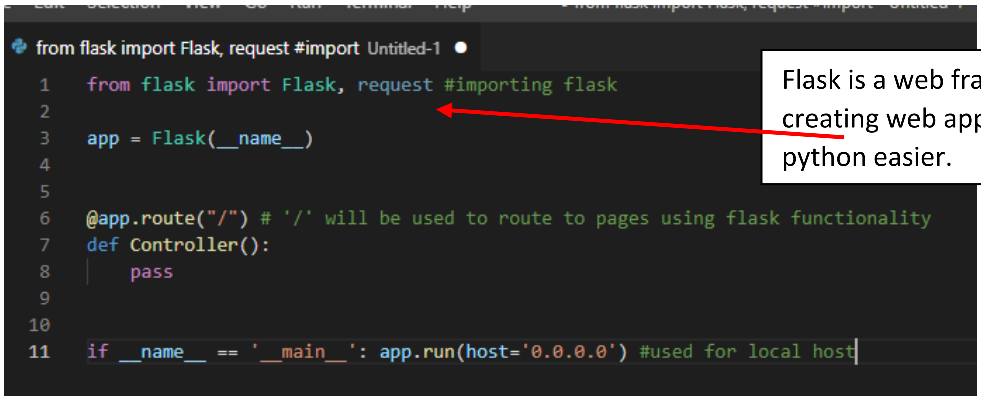
Triage/Tasks

FR#	Task #	Functional requirement	Expected outcomes
1	1.1	Create Controller module	
	1.2	Create Model module	
	1.3	Create Database module	
	1.4	Connection Controller with model	
	1.5	Create a basic homepage	User is able to get on a webpage and see 'Toka Fitness Title'
2	2.1	Create a user in database	User is made in the database
	2.2	Create a form for free account	User is able to see free account form
	2.3	Create a form for Premium account	User is able to see Premium Account Form.
	2.4	Add detail to be filled out in the forms	User can fill out any one of the forms
	2.5	Create CSS file for all CSS (designing)	

3	3.1	Create a submit form button	User is able to click on a submit forms button
	3.2	Button should lead it to another page	Once user clicks on submit button it should lead it to blank page

Log table

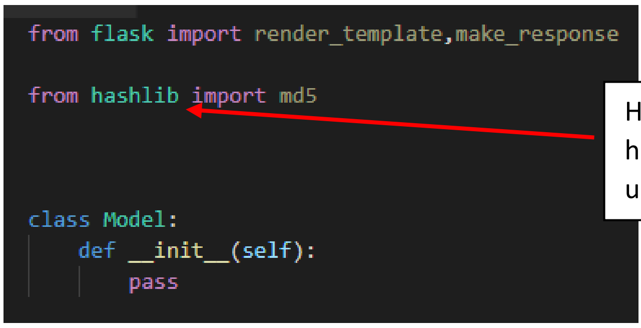
1.1 Creating Controller Module.



```
from flask import Flask, request #importing flask

app = Flask(__name__)


@app.route("/") # '/' will be used to route to pages using flask functionality
def Controller():
    pass


if __name__ == '__main__': app.run(host='0.0.0.0') #used for local host
```

Flask is a web framework for creating web applications in python easier.

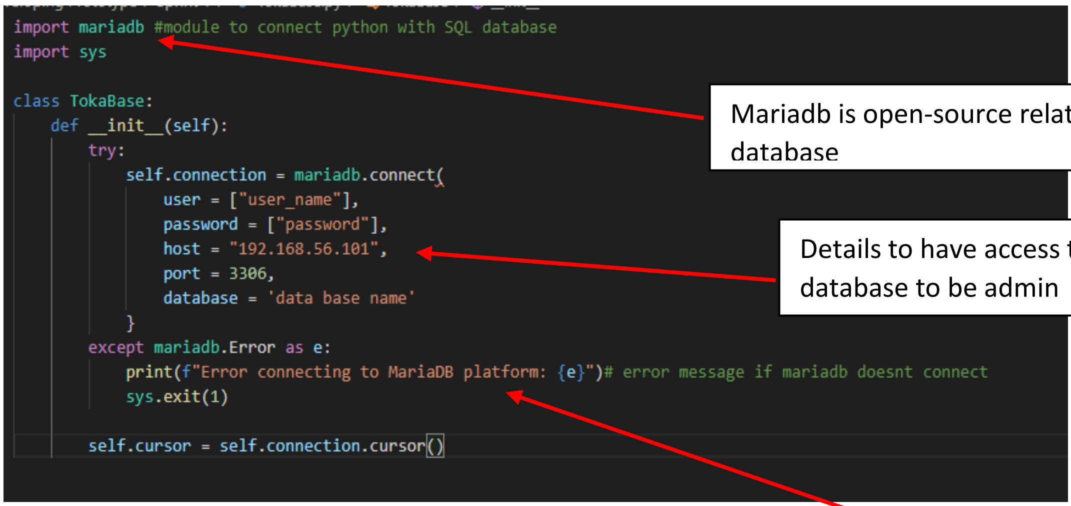
1.2 Creating Model module



```
from flask import render_template,make_response

from hashlib import md5



class Model:
    def __init__(self):
        pass
```

Hashlib module is a module for hashing messages easily (will be used for encrypting) (3)

1.3 Create Database Module



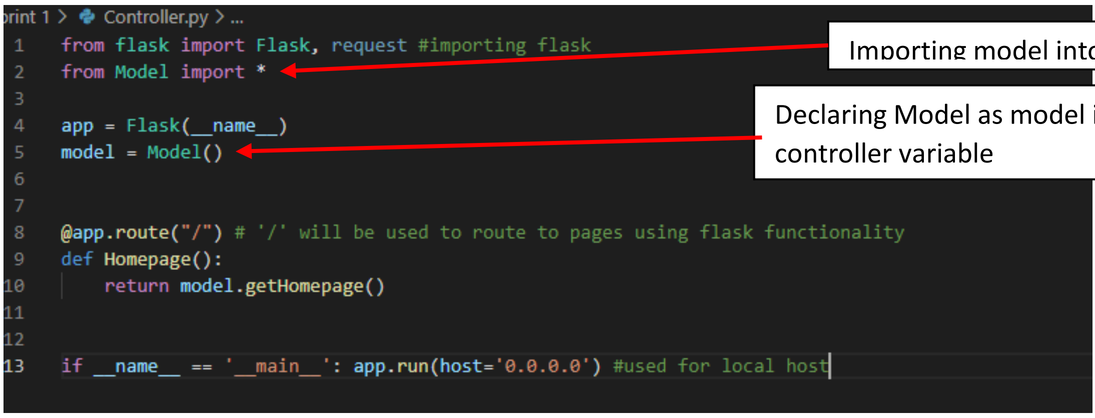
```
import mariadb #module to connect python with SQL database
import sys

class TokaBase:
    def __init__(self):
        try:
            self.connection = mariadb.connect(
                user = ["user_name"],
                password = ["password"],
                host = "192.168.56.101",
                port = 3306,
                database = 'data base name'
            )
        except mariadb.Error as e:
            print(f"Error connecting to MariaDB platform: {e}")# error message if mariadb doesnt connect
            sys.exit(1)

        self.cursor = self.connection.cursor()
```

Mariadb is open-source relational database

Details to have access to the database to be admin

If connection fails instead of getting no message, I will receive this message:

1.4 Connecting Controller with model

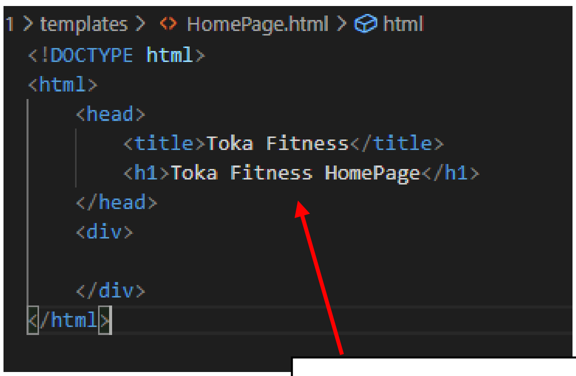


```python
from flask import Flask, request #importing flask
from Model import *

app = Flask(__name__)
model = Model()


@app.route("/") # '/' will be used to route to pages using flask functionality
def Homepage():
    return model.getHomepage()


if __name__ == '__main__': app.run(host='0.0.0.0') #used for local host
```

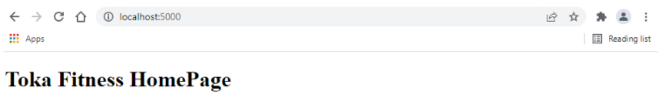
Importing model into controller

Declaring Model as model in controller variable

1.5 Creating a basic Homepage

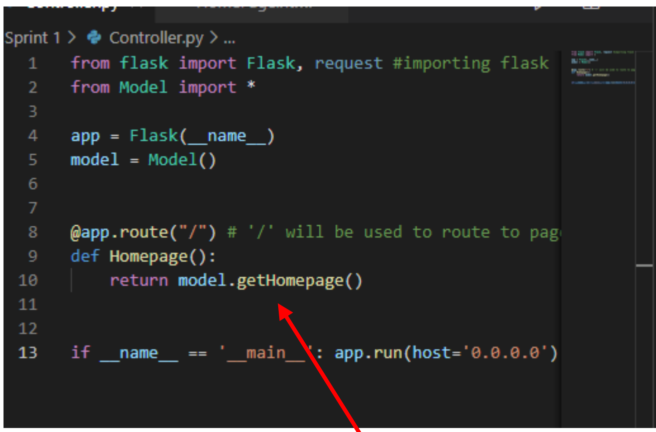


```html
<!DOCTYPE html>
<html>
    <head>
        <title>Toka Fitness</title>
        <h1>Toka Fitness HomePage</h1>
    </head>
    <div>

    </div>
</html>
```

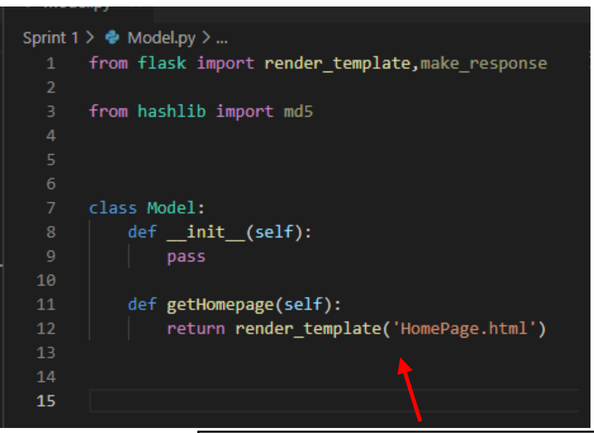


Toka Fitness HomePage

Message user will see



```python
from flask import Flask, request #importing flask
from Model import *

app = Flask(__name__)
model = Model()


@app.route("/") # '/' will be used to route to pag
def Homepage():
    return model.getHomepage()


if __name__ == '__main__': app.run(host='0.0.0.0')
```



```python
from flask import render_template,make_response

from hashlib import md5



class Model:
    def __init__(self):
        pass

    def getHomepage(self):
        return render_template('HomePage.html')


```

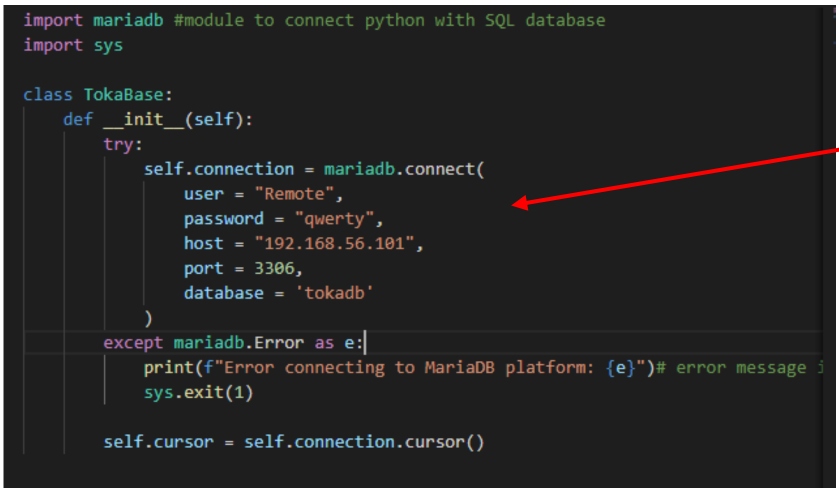
If the function in model exists it will return a template to render in this case ‘HomePage.hmtl’

Return model.getHomepage is sending this variable to model.py

2.1 To create a user in my database I done the commands:

```
create USER 'Remote'@'localhost' IDENTIFIED BY 'qwerty';
```

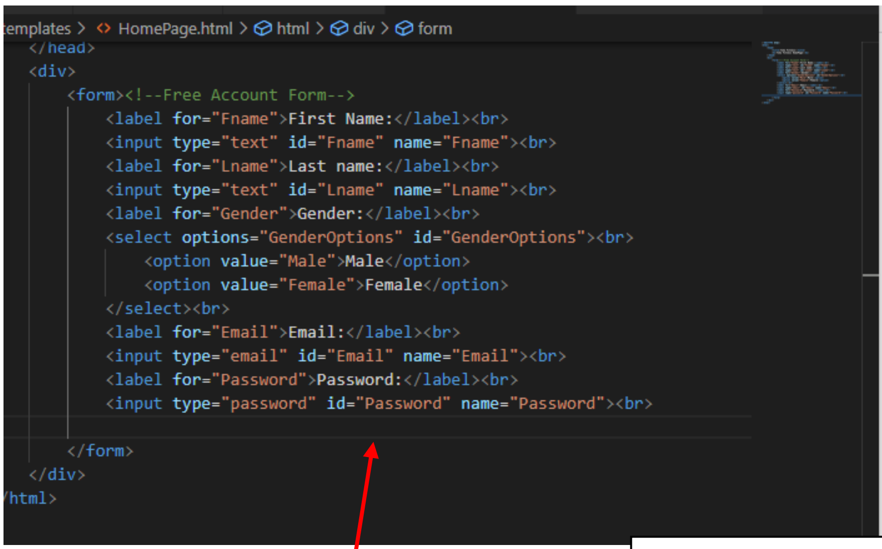
```
GRANT ALL ON tokadb.* TO 'Remote'@'localhost';
```



```
import mariadb #module to connect python with SQL database
import sys

class TokaBase:
    def __init__(self):
        try:
            self.connection = mariadb.connect(
                user = "Remote",
                password = "qwerty",
                host = "192.168.56.101",
                port = 3306,
                database = 'tokadb'
            )
        except mariadb.Error as e:
            print(f"Error connecting to MariaDB platform: {e}")# error message i
            sys.exit(1)

        self.cursor = self.connection.cursor()
```

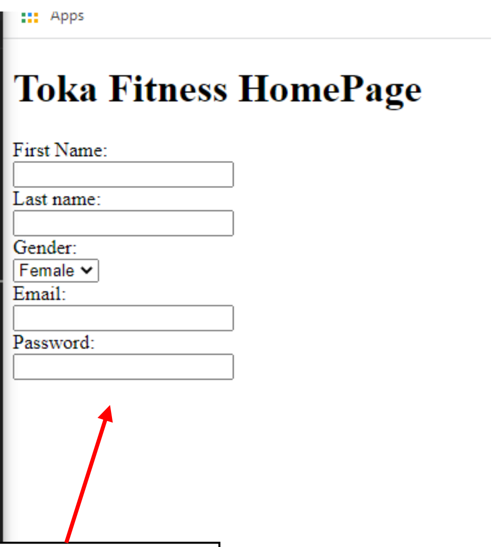
User information is now update giving access to all the commands too

2.2



emplates > HomePage.html > html > div > form

```
    </head>
    <div>
        <form><!--Free Account Form-->
            <label for="Fname">First Name:</label><br>
            <input type="text" id="Fname" name="Fname"><br>
            <label for="Lname">Last name:</label><br>
            <input type="text" id="Lname" name="Lname"><br>
            <label for="Gender">Gender:</label><br>
            <select options="GenderOptions" id="GenderOptions"><br>
                <option value="Male">Male</option>
                <option value="Female">Female</option>
            </select><br>
            <label for="Email">Email:</label><br>
            <input type="email" id="Email" name="Email"><br>
            <label for="Password">Password:</label><br>
            <input type="password" id="Password" name="Password"><br>

        </form>
    </div>
/html>
```



Apps

Toka Fitness HomePage

First Name:

Last name:

Gender: Female

Email:

Password:

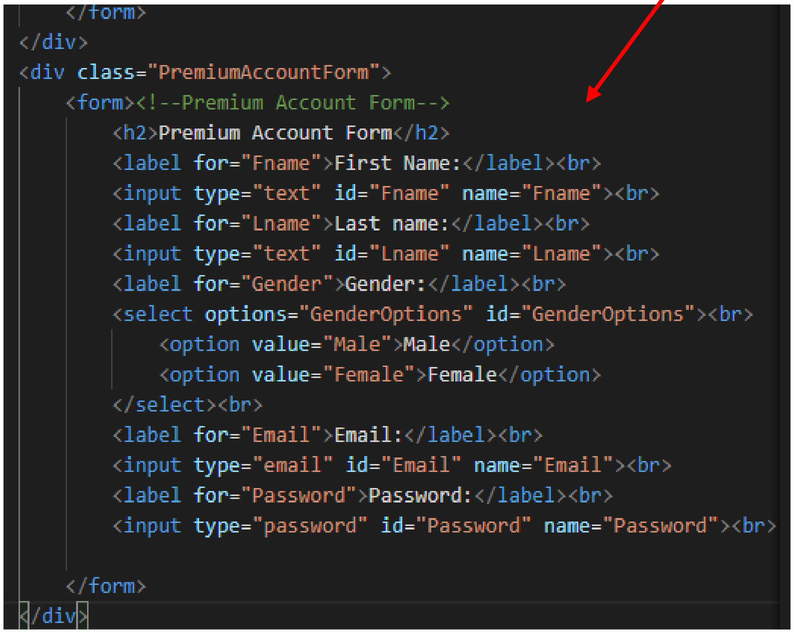
Form which user will need to fill in

All the behind the scenes of form creation such as name, password email etc

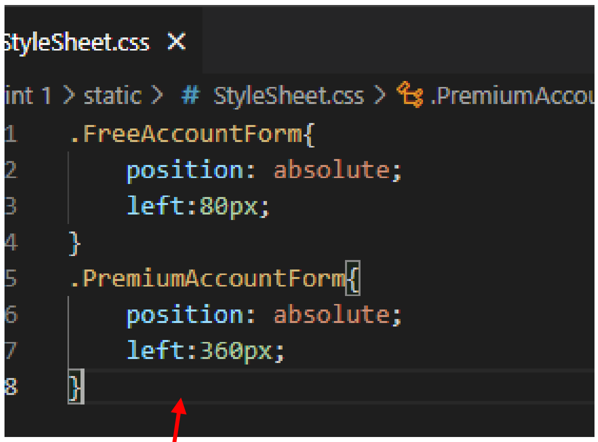
2.3 Create a form for Premium account

2.4 Add detail to be filled out in the forms

Same form Created for premium version

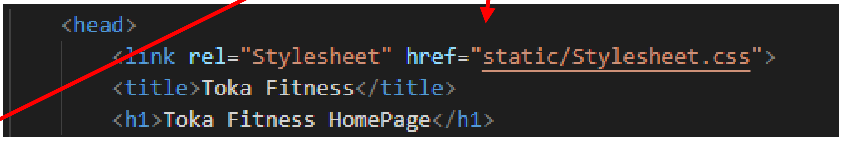


```
    </form>
</div>
<div class="PremiumAccountForm">
    <form><!--Premium Account Form-->
        <h2>Premium Account Form</h2>
        <label for="Fname">First Name:</label><br>
        <input type="text" id="Fname" name="Fname"><br>
        <label for="Lname">Last name:</label><br>
        <input type="text" id="Lname" name="Lname"><br>
        <label for="Gender">Gender:</label><br>
        <select options="GenderOptions" id="GenderOptions"><br>
            <option value="Male">Male</option>
            <option value="Female">Female</option>
        </select><br>
        <label for="Email">Email:</label><br>
        <input type="email" id="Email" name="Email"><br>
        <label for="Password">Password:</label><br>
        <input type="password" id="Password" name="Password"><br>

    </form>
</div>
```

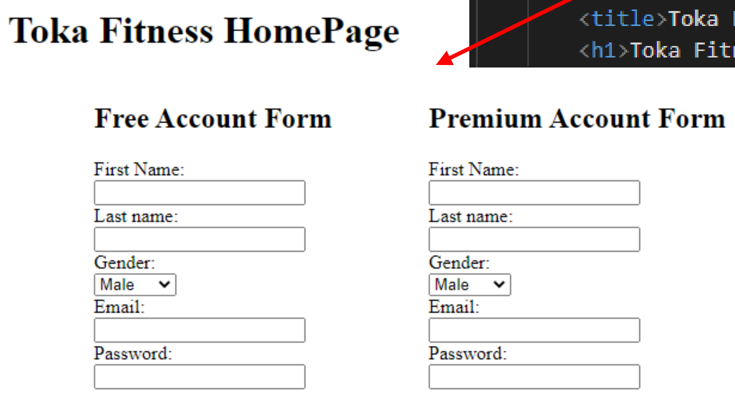


```
StyleSheet.css ×
int 1 > static > # StyleSheet.css > .PremiumAccou
.FreeAccountForm{
    position: absolute;
    left:80px;
}
.PremiumAccountForm{
    position: absolute;
    left:360px;
}
```

Creating a style sheet with CSS to get forms side by side and inserting link in Homepage (2)



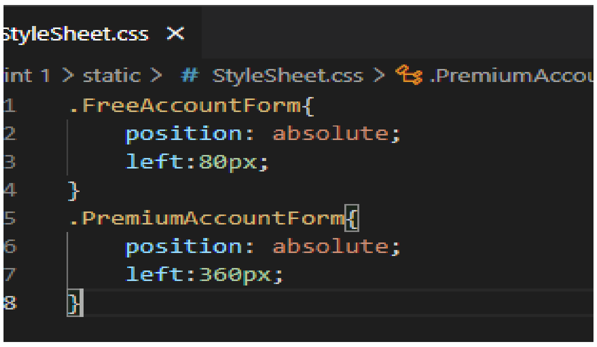
```
<head>
    <link rel="Stylesheet" href="static/Stylesheet.css">
    <title>Toka Fitness</title>
    <h1>Toka Fitness HomePage</h1>
```



Toka Fitness HomePage

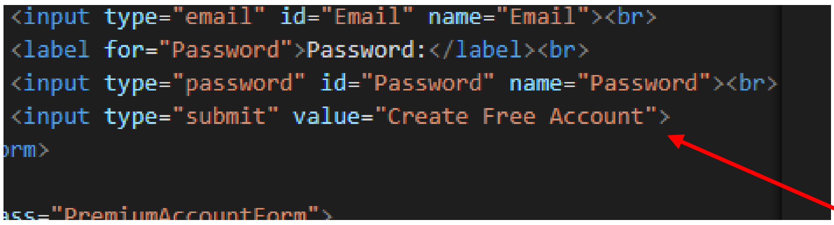
Free Account Form	Premium Account Form
First Name:	First Name:
Last name:	Last name:
Gender: Male	Gender: Male
Email:	Email:
Password:	Password:

2.5 Creating CSS file



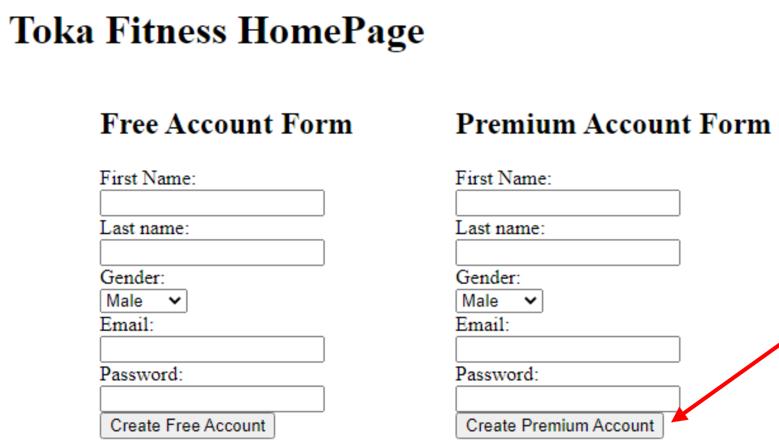
```
StyleSheet.css ×
int 1 > static > # StyleSheet.css > .PremiumAccou
.FreeAccountForm{
    position: absolute;
    left:80px;
}
.PremiumAccountForm{
    position: absolute;
    left:360px;
}
```

3.1 Create a submit form button



```
<input type="email" id="Email" name="Email"><br>
<label for="Password">Password:</label><br>
<input type="password" id="Password" name="Password"><br>
<input type="submit" value="Create Free Account">
rm>

ass="PremiumAccountForm">
```

Submit button is created and can be viewed and clicked by user

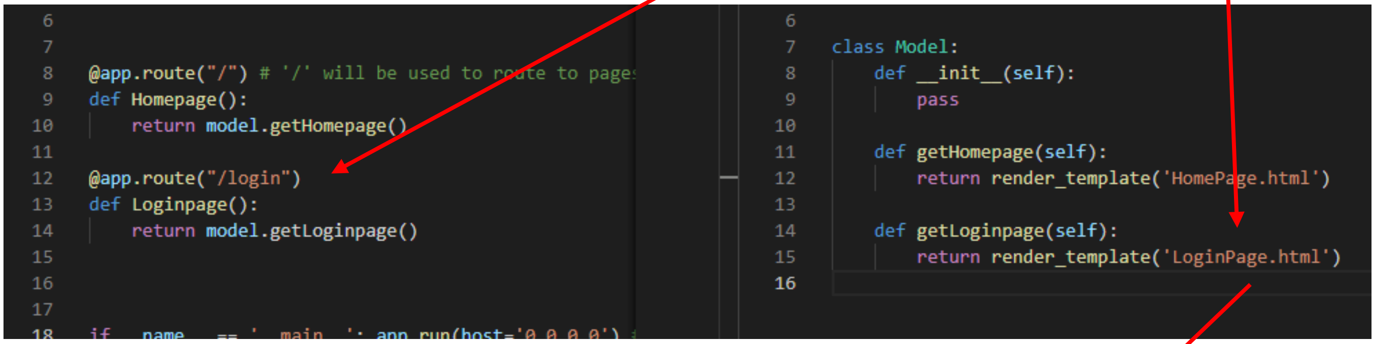


Toka Fitness HomePage

Free Account Form	Premium Account Form
First Name:	First Name:
Last name:	Last name:
Gender: Male	Gender: Male
Email:	Email:
Password:	Password:
Create Free Account	Create Premium Account

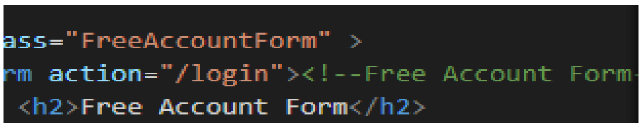
Once controller gets the root of /login sent it will send a request to model to return a template with that function which is login page

3.2 Button should lead it to another page



```
@app.route("/") # '/' will be used to route to pages
def Homepage():
    return model.getHomepage()

@app.route("/login")
def Loginpage():
    return model.getLoginpage()



if __name__ == '__main__': app.run(host='0.0.0.0')
```

```
class Model:
    def __init__(self):
        pass

    def getHomepage(self):
        return render_template('HomePage.html')

    def getLoginpage(self):
        return render_template('LoginPage.html')
```



```
ass="FreeAccountForm" >
rm action="/login"><!--Free Account Form-
 <h2>Free Account Form</h2>
```

Login

Form has an action of /login

This will work when form button is pressed

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
1	1.1	Unit testing Functional test	Testing to see if flask import is working correctly	Server should be able to run on local host	Localhost:5000 is running	Localhost:5000 Is website
	1.2	Unit testing Functional test	Just to see if model runs	Model can run by itself once clicked run	Model runs with no errors	x
	1.3	Unit testing Functional test	Checking if TokaBase (database) is running	Should work as it suppose to	Not running due to not having a user created yet	Need to create a user for database
	1.4	Integration test Functional test	Checking to see if controller and model communicate with each other	Model should receive controllers' requests	Controller and model can communicate with each other by returning template	x
	1.5	unit testing integration testing	Webpage is made and displays Toka name Controller sends request to model to show database	On localhost user should be able to see Toka Fitness Title to be displayed	Toka Fitness Homepage is getting Displayed for the user once entered in search bar for localhost:5000/	
2	2.1	Unit testing Functional test	Creating a user in MySQL database and granting it access to everything	User should be created with all the permissions	User has been created and named as 'Remote'	User has now been created for database
	2.2	Unit testing Acceptance testing	Creating a form for user to be able to fill in with all the necessary details	All field names are in the form which is required and are drop downs work as well as text	Drop down works and text can be entered in the fields	x
	2.3	Unit testing	Create a form for Premium account	All field names are in the form which is required and are drop downs work as well as text	Drop down works and text can be entered in the fields	
	2.4	Unit testing	Add detail to be filled out in the forms	All information requested in the forms getting displayed	All requested information is displayed	x
	2.5	Unit testing	Making sure CSS file is linked to home page	If linked forms should be side by side instead of on top of each other	CSS has linked with the home page and now is side by side with each other	x

3	3.1	Unit testing Functional testing	Once can see and click on submit button	User can click on submit button but should do nothing	A user can now click on a button	x
	3.2	Unit testing Functional testing	Once user clicks on submit, they should be sent to a blank page (login page)	Once submit button is clicked, they get sent to another page	Once user clicks on login button it will lead them to page with login title	x

Review

Sprint 1 was very successful as I manged to finish all the goals which I have set out and can even see some of the functionality work. The benefit of completing all these tasks is that it means that in sprint 2 there are no backlogged tasks which need completing therefore my time management is correct and should be on schedule for completing the product by the deadline.

There was a problem at the start when I tried to run controller and model together, but it didn’t work because in the controller file I haven’t imported model function which meant that there was no link with each other and therefore I kept on getting error: ‘model = Model() isn’t defined’ But this was an easy fix as I was missing a ‘from Model import*’. This line means that controller has called model function.

For Sprint 2 I will need to start adding in some designs onto the pages as sprint 1 was mostly setting everything up and not considering any design instead just functionality. Sprint 2 will also be a bit harder as it will require to me to start retrieving customers details and storing them in the database to be used for the login page. To make this easier I will split this functional requirement into smaller tasks and once again complete them one by one and test them to the fullest.

As result of completing sprint 1 early I’m able to start adding colour and designs to the homepage to not make it as plain white as it is.

Sprint 2

Sprint 2 is based of sprint 1 by developing more on the storing data in the database (TokaBase) from the forms which the user has filled in. This means there will be more database work required and more communication between the webpage and the server.

Goals/Requirements

Goal Set	FR#	Requirements
As a user I want my information to be stored in a database	1	Inserting all user information from forms into database
As a user I want to be able to log in to the dashboard I register for	2	Create login button and once user enters their information, they will be sent to the dashboard they requested

Triage/Tasks

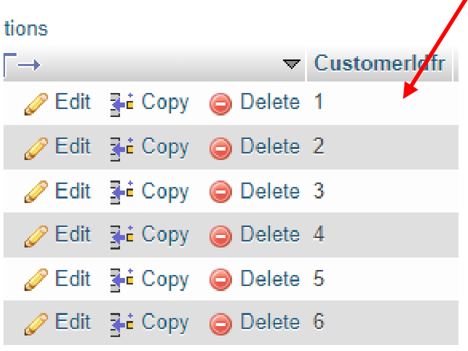
FR#	Task #	Functional requirement	Expected outcomes
1	1.1	Have Customer increment by 1	Every time user submits a form data gets sent to database and CustomerID should start incrementing
	1.2	Customer details should be placed in correct spots in database	All data should be in their correct spots such as name in name column email in email column etc

	1.3	In html forms add invisible label which will determine the account type	The hidden field will be used once user submits the form it will send it to the account type which will determine if user registered on a free account or a premium
2	2.1	Create a login Page	Login page should have a form which requests for email and password
	2.2	Create login button	User should be able to click on login button
	2.3	Login button sends them to login page	Login page asks user for information
	2.4	Verifying users information to see if it exists in database	When user fills in the requested information and it exists in database it should give user a blank page with either free of premium dashboard
	2.5	Create free dashboard	If user has a free account type, they get sent to free dashboard
	2.6	Create Premium Dashboard	If user has premium account type, they get sent to premium dashboard

1.1

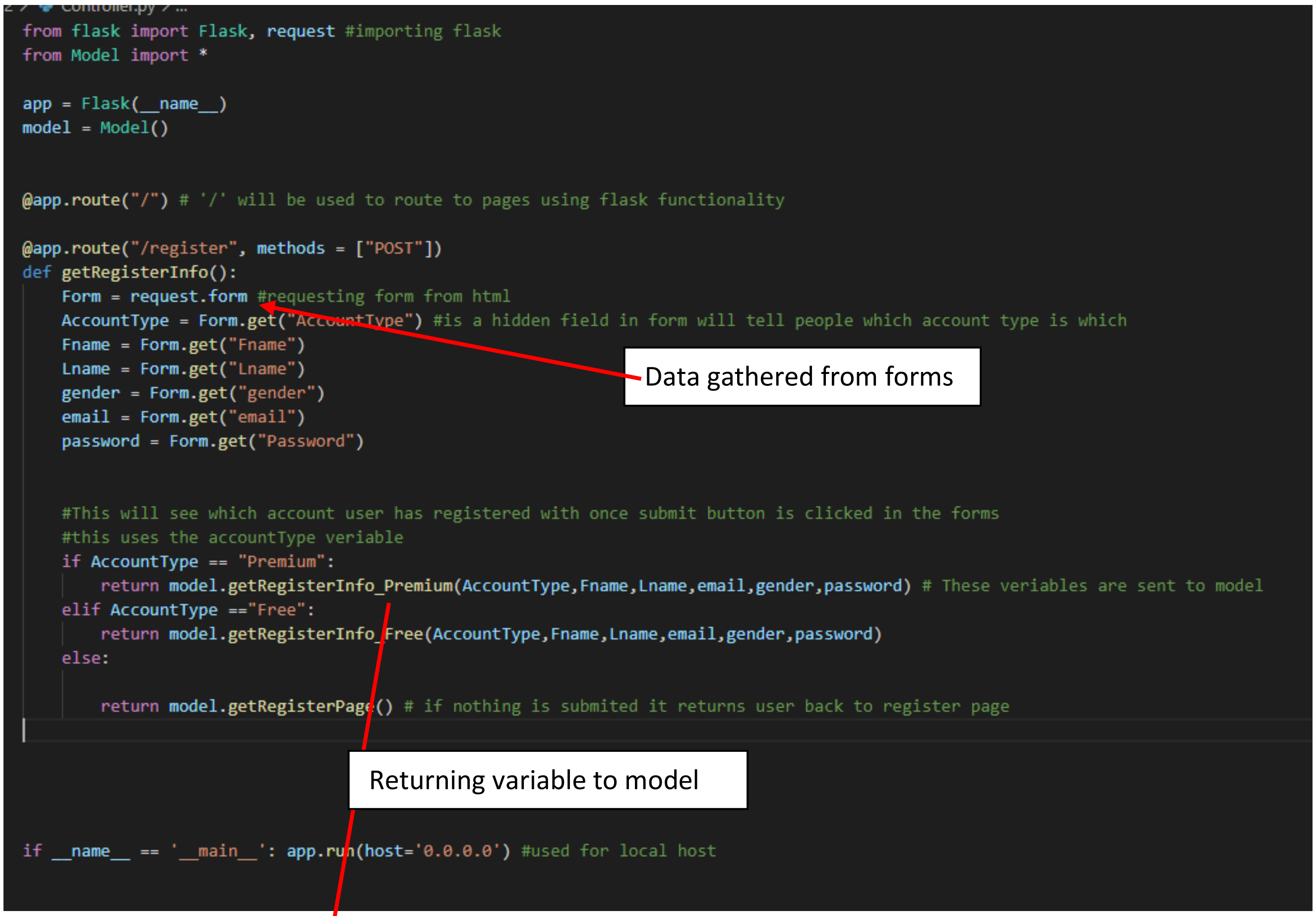
```
ALTER TABLE tblcustomerid MODIFY COLUMN CustomerIdfr INT AUTO_INCREMENT;
```

This line is telling SQL to automatically increment it once it updates meaning CustomerIdfr will be increased by 1 every time

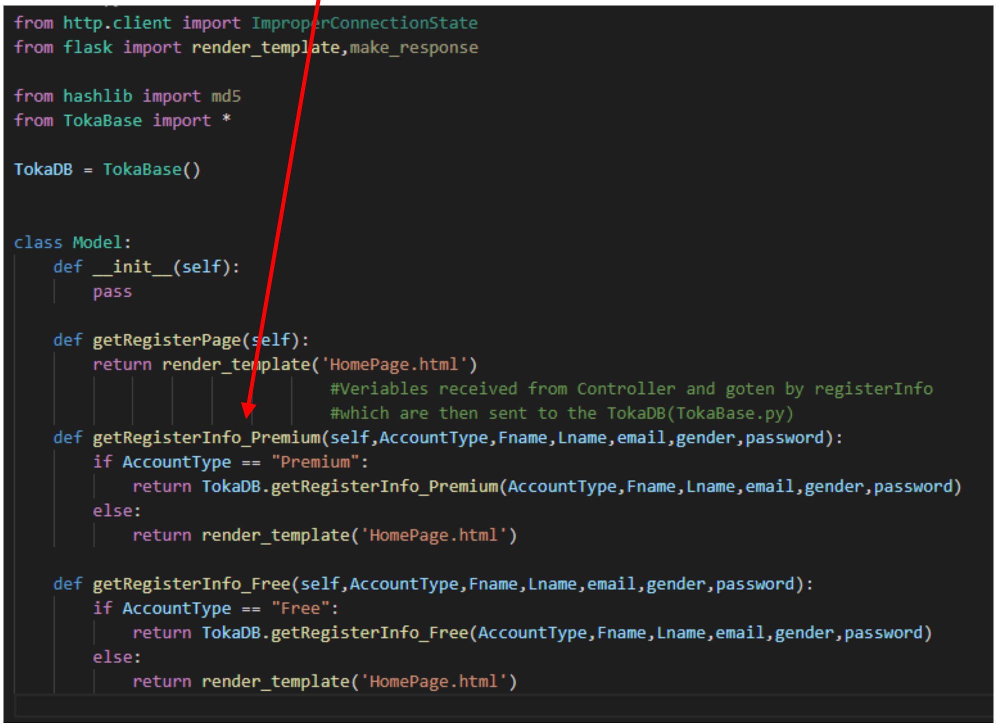


tions	CustomerIdfr
Edit Copy Delete	1
Edit Copy Delete	2
Edit Copy Delete	3
Edit Copy Delete	4
Edit Copy Delete	5
Edit Copy Delete	6

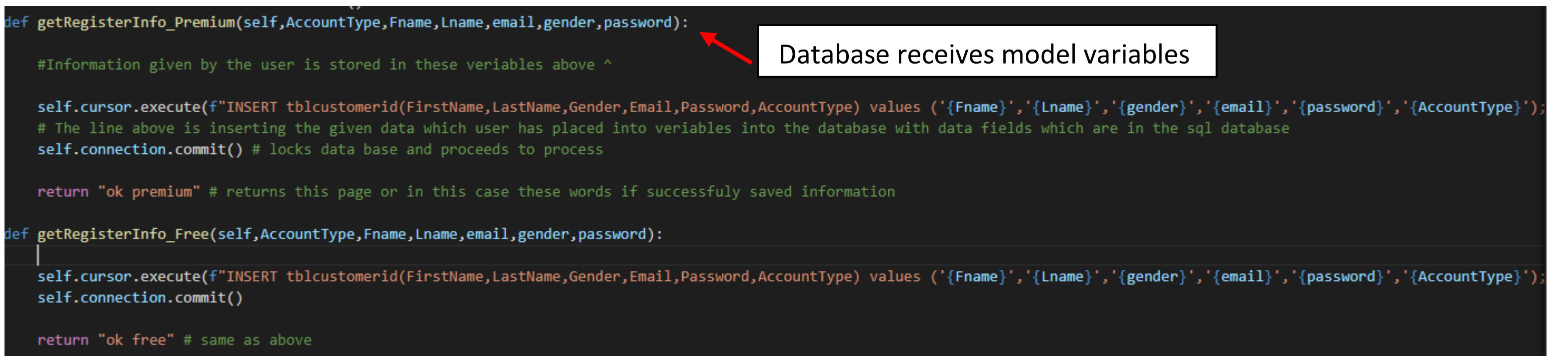
1.2



```python
from flask import Flask, request #importing flask
from Model import *

app = Flask(__name__)
model = Model()


@app.route("/") # '/' will be used to route to pages using flask functionality

@app.route("/register", methods = ["POST"])
def getRegisterInfo():
    Form = request.form #requesting form from html
    AccountType = Form.get("AccountType") #is a hidden field in form will tell people which account type is which
    Fname = Form.get("Fname")
    Lname = Form.get("Lname")
    gender = Form.get("gender")
    email = Form.get("email")
    password = Form.get("Password")


    #This will see which account user has registered with once submit button is clicked in the forms
    #this uses the accountType veriable
    if AccountType == "Premium":
        return model.getRegisterInfo_Premium(AccountType,Fname,Lname,email,gender,password) # These veriables are sent to model
    elif AccountType =="Free":
        return model.getRegisterInfo_Free(AccountType,Fname,Lname,email,gender,password)
    else:

        return model.getRegisterPage() # if nothing is submited it returns user back to register page






if __name__ == '__main__': app.run(host='0.0.0.0') #used for local host
```

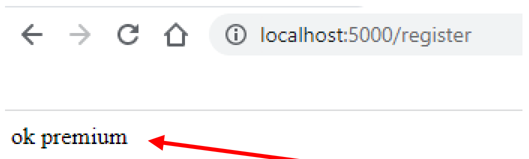


```python
from http.client import ImproperConnectionState
from flask import render_template,make_response

from hashlib import md5
from TokaBase import *

TokaDB = TokaBase()


class Model:
    def __init__(self):
        pass

    def getRegisterPage(self):
        return render_template('HomePage.html')
                                #Veriables received from Controller and goten by registerInfo
                                #which are then sent to the TokaDB(TokaBase.py)
    def getRegisterInfo_Premium(self,AccountType,Fname,Lname,email,gender,password):
        if AccountType == "Premium":
            return TokaDB.getRegisterInfo_Premium(AccountType,Fname,Lname,email,gender,password)
        else:
            return render_template('HomePage.html')

    def getRegisterInfo_Free(self,AccountType,Fname,Lname,email,gender,password):
        if AccountType == "Free":
            return TokaDB.getRegisterInfo_Free(AccountType,Fname,Lname,email,gender,password)
        else:
            return render_template('HomePage.html')
```



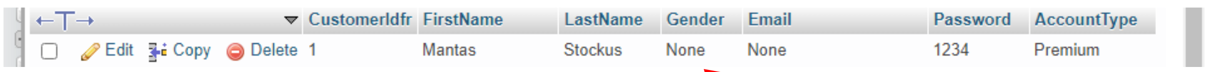
```
def getRegisterInfo_Premium(self,AccountType,Fname,Lname,email,gender,password):

    #Information given by the user is stored in these veriables above ^

    self.cursor.execute(f"INSERT tblcustomerid(FirstName,LastName,Gender,Email,Password,AccountType) values ('{Fname}','{Lname}','{gender}','{email}','{password}','{AccountType}');
    # The line above is inserting the given data which user has placed into veriables into the database with data fields which are in the sql database
    self.connection.commit() # locks data base and proceeds to process

    return "ok premium" # returns this page or in this case these words if successfuly saved information

def getRegisterInfo_Free(self,AccountType,Fname,Lname,email,gender,password):

    self.cursor.execute(f"INSERT tblcustomerid(FirstName,LastName,Gender,Email,Password,AccountType) values ('{Fname}','{Lname}','{gender}','{email}','{password}','{AccountType}');
    self.connection.commit()

    return "ok free" # same as above
```

Database receives model variables



ok premium

This means that user has entered details in the premium form, and they clicked submit therefore leading them to this page. The words ok premium act as information to development to suggest which form, they done

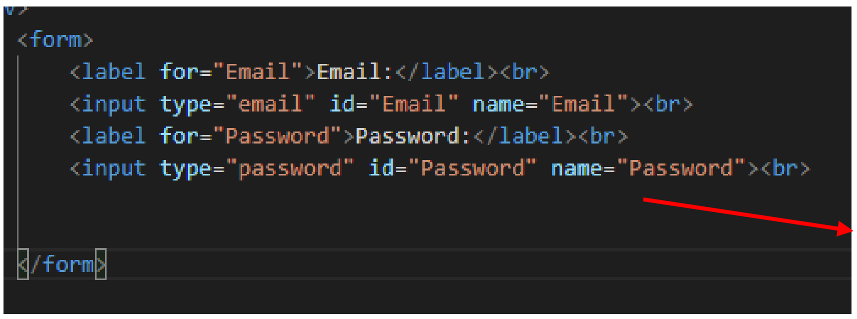


	CustomerIdfr	FirstName	LastName	Gender	Email	Password	AccountType
Edit Copy Delete	1	Mantas	Stockus	None	None	1234	Premium

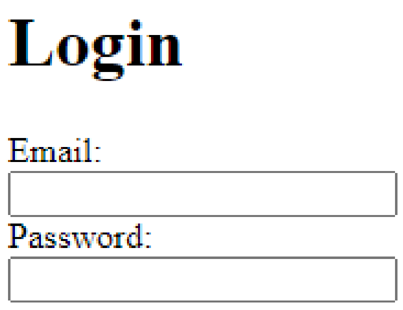
The data from the form has now been stored in database

However is missing 2 fields (variable names in html are wrongly typed). This has been fixed in test log

2.1 Create a login page

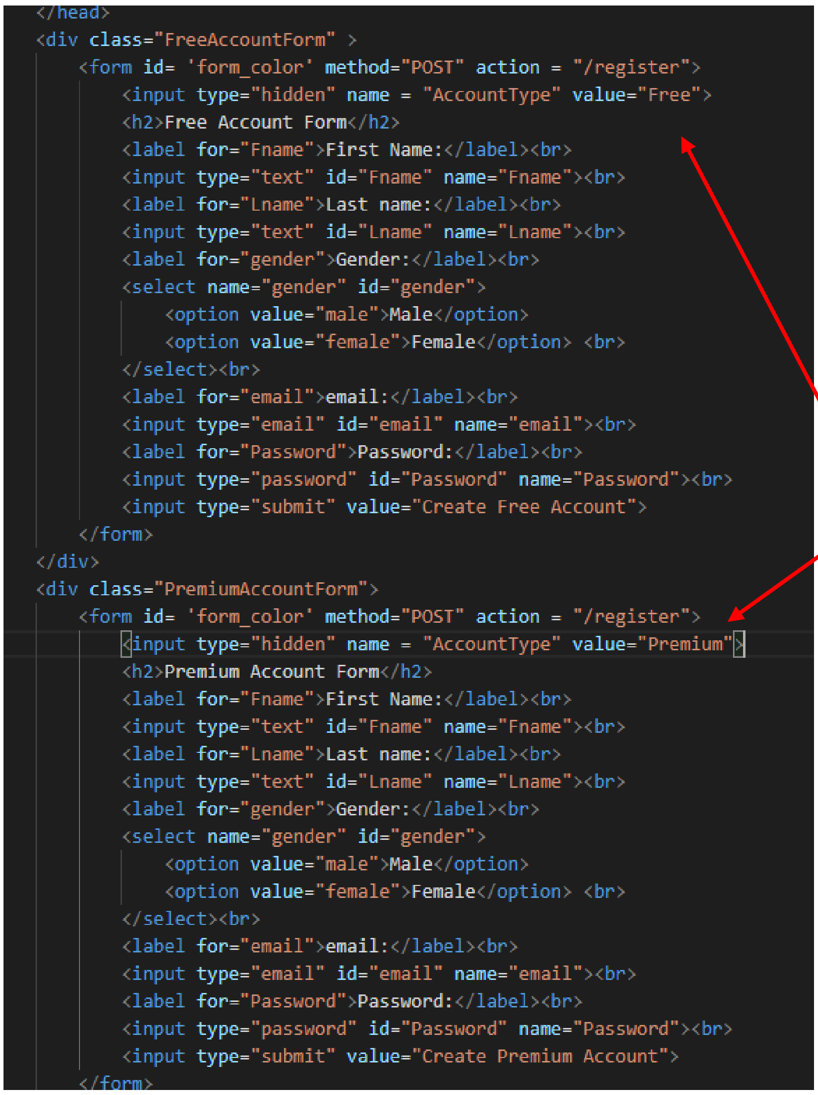


```
<form>
    <label for="Email">Email:</label><br>
    <input type="email" id="Email" name="Email"><br>
    <label for="Password">Password:</label><br>
    <input type="password" id="Password" name="Password"><br>


</form>
```



This html is the form behind the login page asking user for email and password as verification to validate

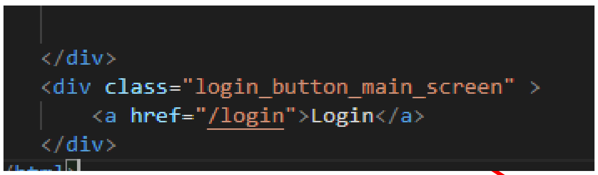
1.3 In html forms add invisible label which will determine the account type



```
    </head>
    <div class="FreeAccountForm" >
        <form id= 'form_color' method="POST" action = "/register">
            <input type="hidden" name = "AccountType" value="Free">
            <h2>Free Account Form</h2>
            <label for="Fname">First Name:</label><br>
            <input type="text" id="Fname" name="Fname"><br>
            <label for="Lname">Last name:</label><br>
            <input type="text" id="Lname" name="Lname"><br>
            <label for="gender">Gender:</label><br>
            <select name="gender" id="gender">
                <option value="male">Male</option>
                <option value="female">Female</option> <br>
            </select><br>
            <label for="email">email:</label><br>
            <input type="email" id="email" name="email"><br>
            <label for="Password">Password:</label><br>
            <input type="password" id="Password" name="Password"><br>
            <input type="submit" value="Create Free Account">
        </form>
    </div>
    <div class="PremiumAccountForm">
        <form id= 'form_color' method="POST" action = "/register">
            <input type="hidden" name = "AccountType" value="Premium">
            <h2>Premium Account Form</h2>
            <label for="Fname">First Name:</label><br>
            <input type="text" id="Fname" name="Fname"><br>
            <label for="Lname">Last name:</label><br>
            <input type="text" id="Lname" name="Lname"><br>
            <label for="gender">Gender:</label><br>
            <select name="gender" id="gender">
                <option value="male">Male</option>
                <option value="female">Female</option> <br>
            </select><br>
            <label for="email">email:</label><br>
            <input type="email" id="email" name="email"><br>
            <label for="Password">Password:</label><br>
            <input type="password" id="Password" name="Password"><br>
            <input type="submit" value="Create Premium Account">
        </form>
```

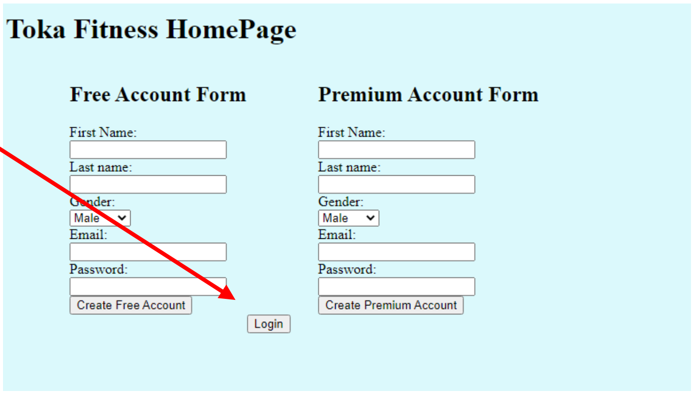
This is a hidden input which the form will be able to receive and store users account preference telling user has made premium account or free account

2.2 Create login button

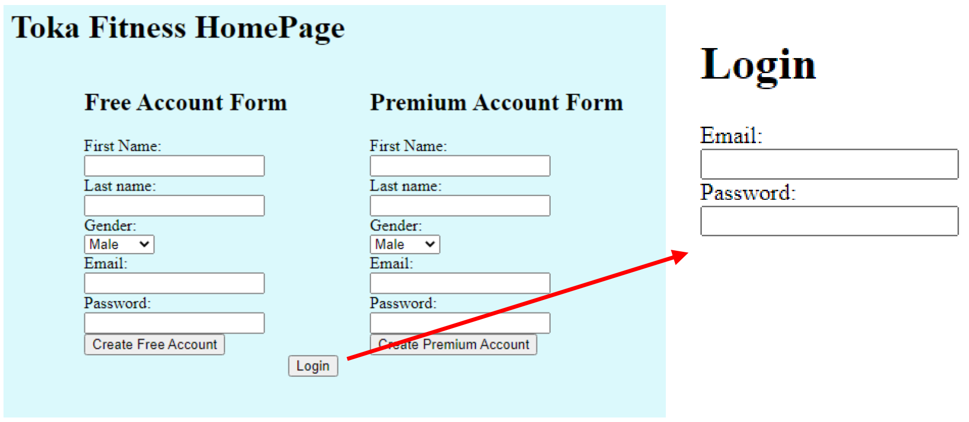


```
    </div>
    <div class="login_button_main_screen" >
        <a href="/login">Login</a>
    </div>
</html>
```

Html code for creating code which goes to login page once clicked on



2.3 Login button sends them to login page



Toka Fitness HomePage

Free Account Form

First Name:

Last name:

Gender: Male

Email:

Password:

Create Free Account

Premium Account Form

First Name:

Last name:

Gender: Male

Email:

Password:

Create Premium Account

Login

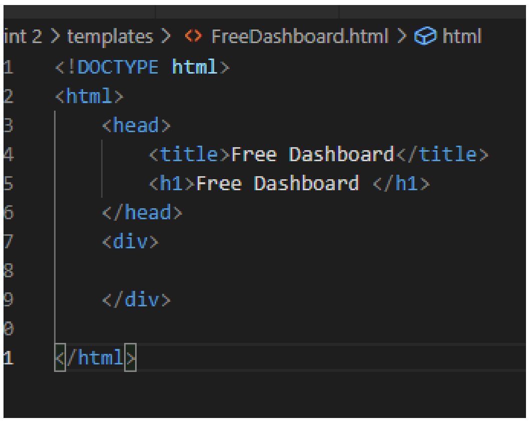
Login

Email:

Password:

Once click on login user is sent to this page with a form of login

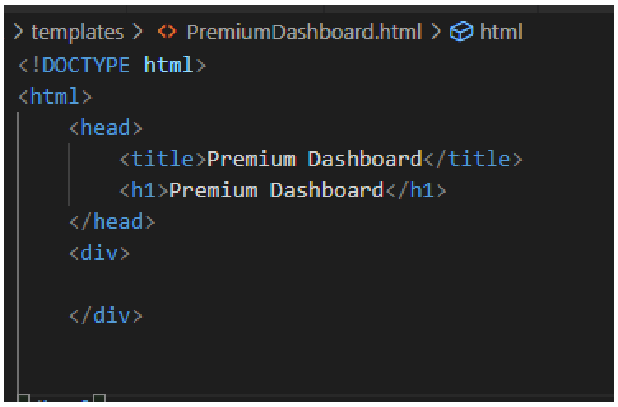
2.5 Create free dashboard



int 2 > templates > FreeDashboard.html > html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Free Dashboard</title>
        <h1>Free Dashboard </h1>
    </head>
    <div>

    </div>

</html>
```

2.6 Create Premium Dashboard



> templates > PremiumDashboard.html > html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Premium Dashboard</title>
        <h1>Premium Dashboard</h1>
    </head>
    <div>

    </div>
```

Log table

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
1	1.1	Unit testing	When there is new data added to customer table it should increase by 1. This will be used to identify which customer is which as it's a unique one number	Under customeridfr there should be numbers appear starting from 1 once there is a record added	(to be tested)	This will need to be tested in the next task as right now we are not getting any new information from user
	1.2	System testing Integration testing	There should be communication from controller received user inputs from forms submit button pressed. Controller should get form information and store it in variables which then should be sent to model. Model will receive all the information sent from controller and send all data to the database file. The database should insert those variables which contain users' data into the table in SQL and return a page with a message depending on which form has been completed	Controller receive form details from user Controller sending details to model Model receiving details Model sending data to TokaBase TokaBase using the variables in inserting the data into the database	Users' data is now stored in the database. This means all information from forms is getting correctly received and sent across different modules (controller, model, tokabase). This also means that Tokabase is storing user data.	CustomerIDfr is now getting increased by 1 (1.1) Data can now be stored in the database
	1.3	Integration testing	When retrieving forms depending on the form submitted it should store the correct account type (FREE/PREMIUM) this will be used to distinguish if user is a premium user or free user	Controller should get AccountType variable with data inside. TokaBase should send data to SQL to store it	Controller now receives AccountType TokaBase saves it in the table	AccountType is hidden field user will not see it

2	2.1	Unit testing Functional testing	Checking to see once submit button is clicked it should move user to the login page	Once submit form button Is clicked user should be sent to a login page	User is now sent to login page once clicked on submit button	
	2.2	Unit testing	When user clicks on login button, they get sent to login page	User should be able to click on button and be sent across to different page	User is now able to click on login button and get sent to login page	
	2.3	Unit testing Functional testing	There should be a transition from click on login button to get to login page	Once user clicks on login button, they should be shown a form to fill in	User can click on login button and greeted with a form to fill in	
	2.4	Integrated testing	To test the login system, it will require me to get the information user has entered in login form and check it against existing users. As a result of this I need a cookies table where each user will be uniquely stored to retrieve their information and see if they can access database, they registered for	This cannot be tested or done till sprint 3	This cannot be tested till sprint 3 therefore needs to be done	SPRINT 3 to make this work
	2.5	Unit testing Accessibility testing	Just the making of the pages themselves	There should be a free dashboard page added to files	There is a Free Dashboard page	(will be tested in sprint 3)
	2.6	Unit testing Accessibility testing	Just the making of the pages themselves	There should be a premium dashboard page added to files	There is a Premium Dashboard page	(will be tested in sprint 3)

Review

Sprint 2 was all about getting the data from the user and storing it in the database (TokaBase) from the forms filled in. As a result of this all my work was based of getting the inputs from the user and saving them in a database which was successfully achieved meaning that my goal for sprint 2 is done.

During sprint 2 I have come across a brick wall which used a lot of my time to figure out what is wrong. When I was creating a user account for the database, I was entering host name of: “192.168.56.101” in the MySQL statement which when if you run on python would give out an error saying: “Error connecting to MariaDB platform: Host 'CC-DT005-03-WCA' is not allowed to connect to”. This stumbled me as I believed I have entered all the correct details needed for the user to be able to have access to the database from python. This made me think that the host in my python file was the wrong address therefore making me change between localhost and the original address. However, the error was not in the python file it was instead in the MySQL statement where I created the user in. To get by this problem I re-created the user but this time giving it a hostname of ‘%’ which means any host can access my database. As a result of changing the hostname I have manged to create a user which has now got access to the database making my first task complete and giving me ability to complete the others

When I was inserting data into the database, I once again came across another problem of 2 data fields not getting filled in my database. The error was in html code as I have miss spelled the variables which collect the user’s information in the html code which meant that the controller who was reading them was not getting any data back and instead was just sending it blank to model and eventually tokabase sending it to the database. This however was fixed by spelling the variables correctly which meant the data is getting picked up by the controller and sending it to the other modules with variables containing the data

There is now a black log of one task to be completed in sprint 3 which is 2.4 which requires me to create a cookietbl to distinguish which account is which for the login. Cookies is a text files with small pieces of data in this case it would be email and password that will be used to identify the user who is login to the dashboard. As of right now I don’t have any cookies created meaning that login process/authenticating users’ task was unable to be completed making it go to Sprint 3 at the very top

All together I believe sprint 2 was successful as I managed to create most of the functionality which were my goals set at the start. Sadly, one of the tasks will need to be backlogged and placed in sprint 3 meaning that there will be extra work to be done in that sprint with extra time.

Sprint 3

Sprint 3 will be based of creating a cookies table and a function to login/authenticate users to determine which dashboard they are getting sent to. This also includes the task from previous Sprint 2 to be completed 2.4 for login the user in and displaying the correct dashboard for the account

Goals/Requirements

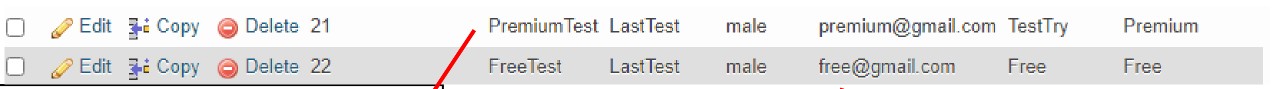
Goal Set	FR#	Requirements
As a user I want to be able to log in to the dashboard I register for (SPRINT 2)	(2 backlogged Sprint 3)	Create cookies table and allows users to use login form which would authenticate them and send them to the specific dashboard
As a user I want to be able to be sent to the correct dashboard	1	Using cookies log the user in depending on the account type they have
As a user I want to see all the options I have in the dashboard	2	Have features on dashboard which are able to be seen by the user

Triage/Tasks

FR#	Task #	Functional requirement	Expected outcomes
(2 Backlogged)	2.4	Verifying users' information to see if it exists in database	When user fills in the login form it should check their information against existing register people info and if its on there allow them to process if it isn't kept them on login page
1	1.1	Login the user in	If user details match the one's in database check to see the account type and depending on account type send them to the specific dashboard page
	1.2	If user doesn't exist	If user doesn't exist in the database for registering, they should not be able to get into any dashboard and instead stay on the login page
2	2.1	Displaying users name in the dashboard	Once user logs in they should get their name displayed on the dashboard
	2.2	Show password/hide password	User has ability to see there password and hide it if they choose to.
	2.3	Creating blocks/buttons which users can press to go to the functional pages	Create buttons that once pressed go to the page by the functional requirement

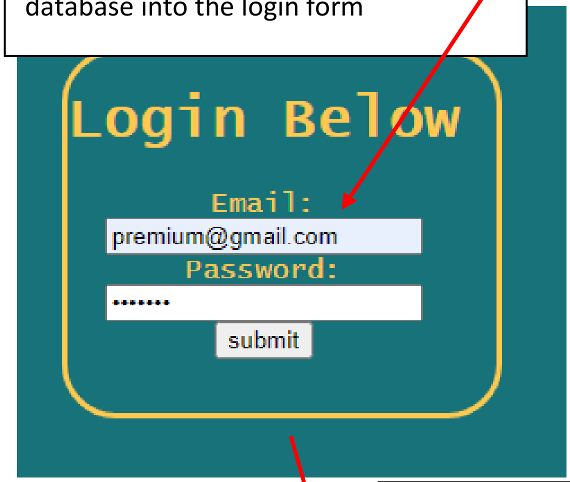
2.4(bl) Verifying users' information to see if it exists in database

1.1 is included in this task (Login the user in)



☐	Edit	Copy	Delete	21	PremiumTest	LastTest	male	premium@gmail.com	TestTry	Premium
☐	Edit	Copy	Delete	22	FreeTest	LastTest	male	free@gmail.com	Free	Free

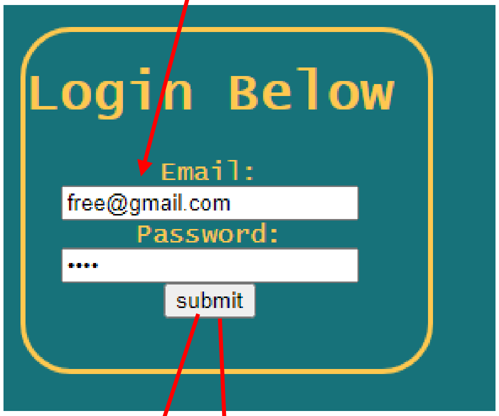
Users' credentials are entered from the database into the login form



Login Below

Email: premium@gmail.com

Password: •••••••

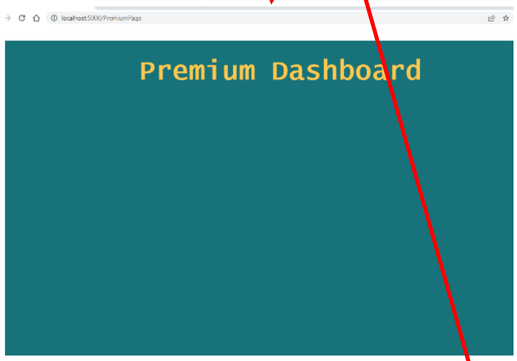


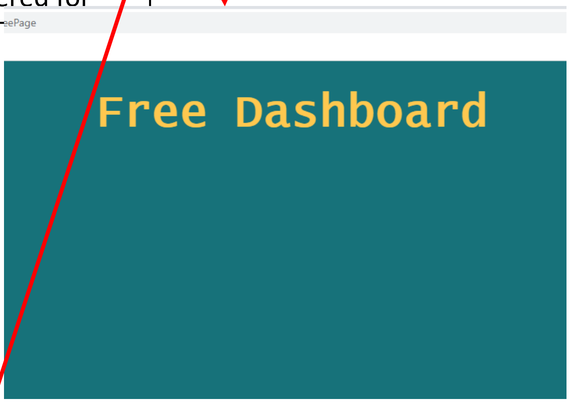
Login Below

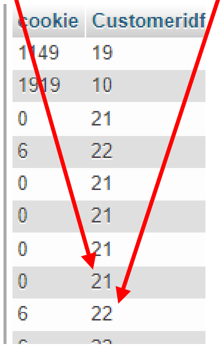
Email: free@gmail.com

Password: ••••

Once the user clicks submit their details get authenticated and sent to the dashboard they registered for

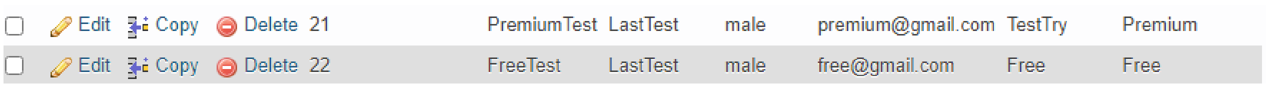






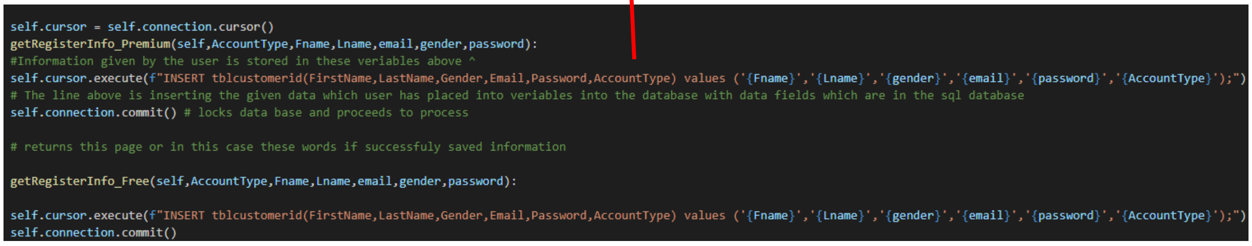
cookie	Customeridf
1149	19
1919	10
0	21
6	22
0	21
0	21
0	21
0	21
6	22

Table cookies has also been created this means once user has logged in with their account, they get a cookie set which will help me identify who is who

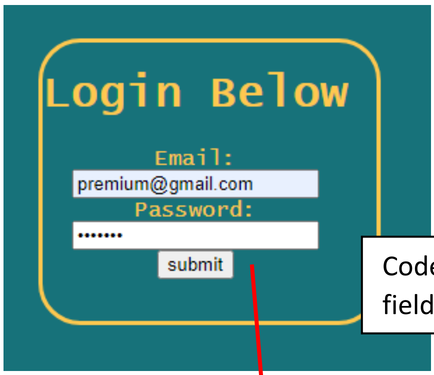


☐	Edit	Copy	Delete	21	PremiumTest	LastTest	male	premium@gmail.com	TestTry	Premium
☐	Edit	Copy	Delete	22	FreeTest	LastTest	male	free@gmail.com	Free	Free

Code for inserting users' information into the table and each column



```
self.cursor = self.connection.cursor()
getRegisterInfo_Premium(self,AccountType,Fname,Lname,email,gender,password):
#Information given by the user is stored in these veriables above ^
self.cursor.execute(f"INSERT tblcustomerid(FirstName,LastName,Gender,Email,Password,AccountType) values ('{Fname}','{Lname}','{gender}','{email}','{password}','{AccountType}');")
# The line above is inserting the given data which user has placed into veriables into the database with data fields which are in the sql database
self.connection.commit() # locks data base and proceeds to process

# returns this page or in this case these words if successfuly saved information

getRegisterInfo_Free(self,AccountType,Fname,Lname,email,gender,password):

self.cursor.execute(f"INSERT tblcustomerid(FirstName,LastName,Gender,Email,Password,AccountType) values ('{Fname}','{Lname}','{gender}','{email}','{password}','{AccountType}');")
self.connection.commit()
```

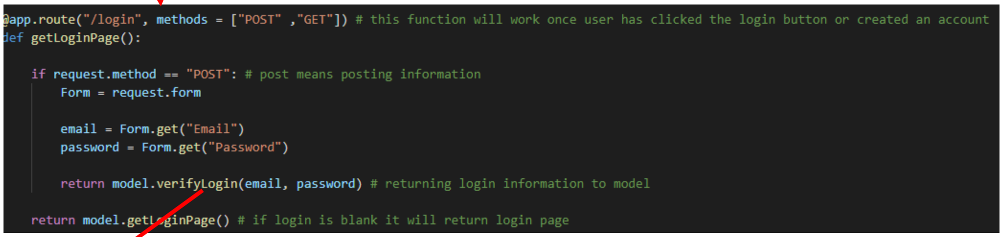


Login Below

Email: premium@gmail.com

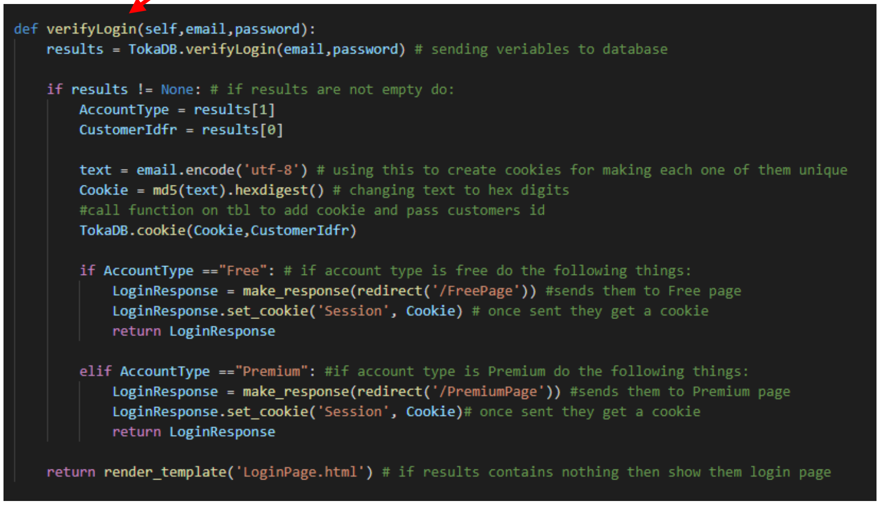
Password: •••••••

Code for retrieving the form with the fields which contain data.

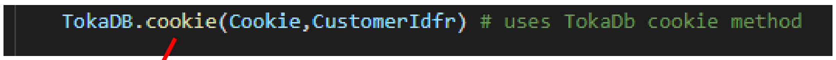


```
@app.route("/login", methods = ["POST" ,"GET"]) # this function will work once user has clicked the login button or created an account
def getLoginPage():

    if request.method == "POST": # post means posting information
        Form = request.form

        email = Form.get("Email")
        password = Form.get("Password")

        return model.verifyLogin(email, password) # returning login information to model

    return model.getLoginPage() # if login is blank it will return login page
```

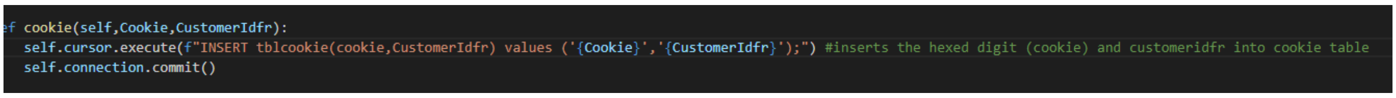
Variable retrieved by model



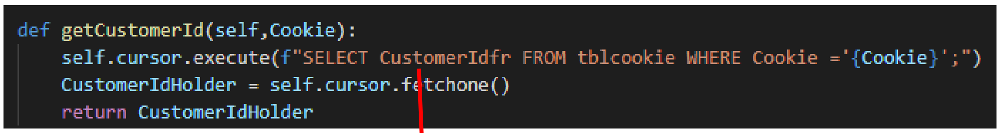
```
def verifyLogin(self,email,password):
    results = TokaDB.verifyLogin(email,password) # sending veriables to database

    if results != None: # if results are not empty do:
        AccountType = results[1]
        CustomerIdfr = results[0]

        text = email.encode('utf-8') # using this to create cookies for making each one of them unique
        Cookie = md5(text).hexdigest() # changing text to hex digits
        #call function on tbl to add cookie and pass customers id
        TokaDB.cookie(Cookie,CustomerIdfr)

        if AccountType =="Free": # if account type is free do the following things:
            LoginResponse = make_response(redirect('/FreePage')) #sends them to Free page
            LoginResponse.set_cookie('Session', Cookie) # once sent they get a cookie
            return LoginResponse

        elif AccountType =="Premium": #if account type is Premium do the following things:
            LoginResponse = make_response(redirect('/PremiumPage')) #sends them to Premium page
            LoginResponse.set_cookie('Session', Cookie)# once sent they get a cookie
            return LoginResponse

    return render_template('LoginPage.html') # if results contains nothing then show them login page
```



```python
    TokaDB.cookie(Cookie,CustomerIdfr) # uses TokaDb cookie method
```

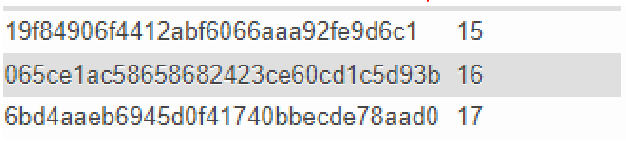


```python
ef cookie(self,Cookie,CustomerIdfr):
    self.cursor.execute(f"INSERT tblcookie(cookie,CustomerIdfr) values ('{Cookie}','{CustomerIdfr}');") #inserts the hexed digit (cookie) and customeridfr into cookie table
    self.connection.commit()
```



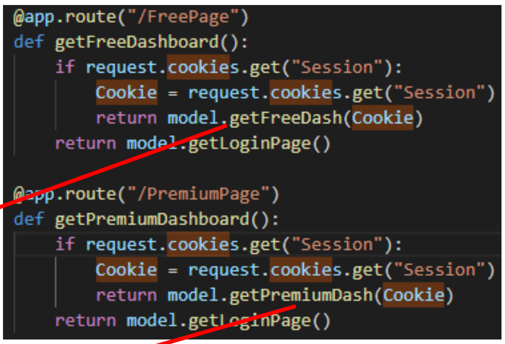
```python
def getCustomerId(self,Cookie):
    self.cursor.execute(f"SELECT CustomerIdfr FROM tblcookie WHERE Cookie ='{Cookie}';")
    CustomerIdHolder = self.cursor.fetchone()
    return CustomerIdHolder
```

Selecting customer id from cookies table



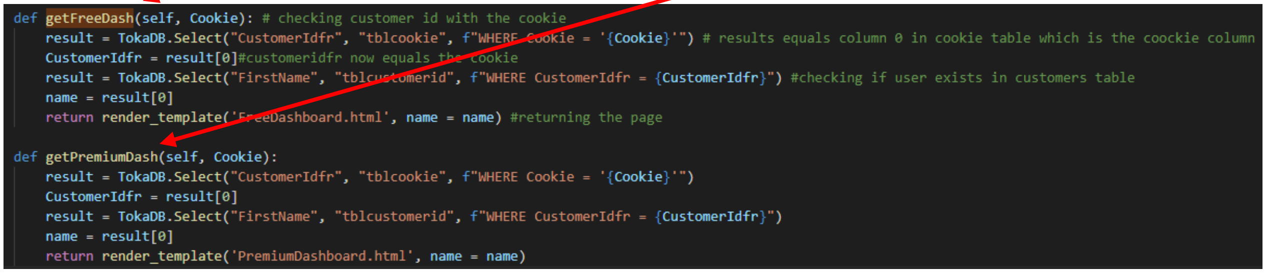
19f84906f4412abf6066aaa92fe9d6c1	15
065ce1ac58658682423ce60cd1c5d93b	16
6bd4aaeb6945d0f41740bbecde78aad0	17

Checking cookies to see which account is signed up or which dashboard



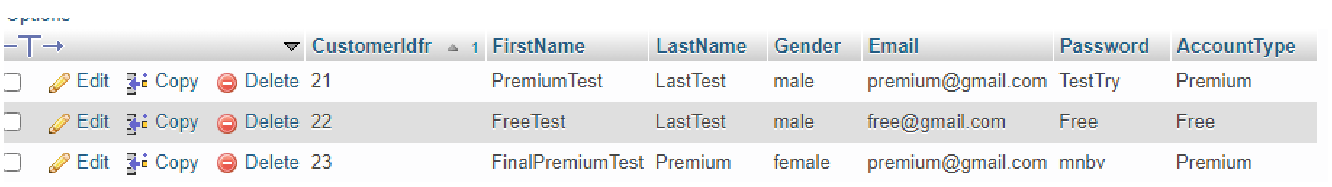
```python
@app.route("/FreePage")
def getFreeDashboard():
    if request.cookies.get("Session"):
        Cookie = request.cookies.get("Session")
        return model.getFreeDash(Cookie)
    return model.getLoginPage()

@app.route("/PremiumPage")
def getPremiumDashboard():
    if request.cookies.get("Session"):
        Cookie = request.cookies.get("Session")
        return model.getPremiumDash(Cookie)
    return model.getLoginPage()
```

Returns variable to model if cookie matches any of the details to show user the page

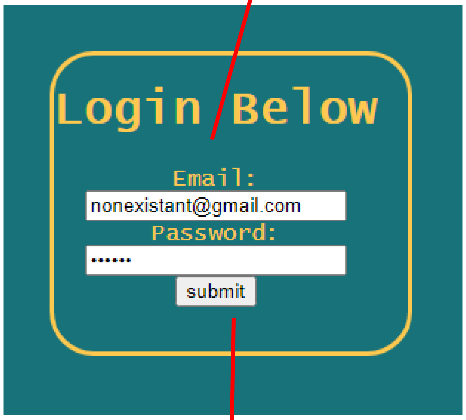


```python
def getFreeDash(self, Cookie): # checking customer id with the cookie
    result = TokaDB.Select("CustomerIdfr", "tblcookie", f"WHERE Cookie = '{Cookie}'") # results equals column 0 in cookie table which is the coockie column
    CustomerIdfr = result[0]#customeridfr now equals the cookie
    result = TokaDB.Select("FirstName", "tblcustomerid", f"WHERE CustomerIdfr = {CustomerIdfr}") #checking if user exists in customers table
    name = result[0]
    return render_template('FreeDashboard.html', name = name) #returning the page

def getPremiumDash(self, Cookie):
    result = TokaDB.Select("CustomerIdfr", "tblcookie", f"WHERE Cookie = '{Cookie}'")
    CustomerIdfr = result[0]
    result = TokaDB.Select("FirstName", "tblcustomerid", f"WHERE CustomerIdfr = {CustomerIdfr}")
    name = result[0]
    return render_template('PremiumDashboard.html', name = name)
```

1.2 If user doesn't exist

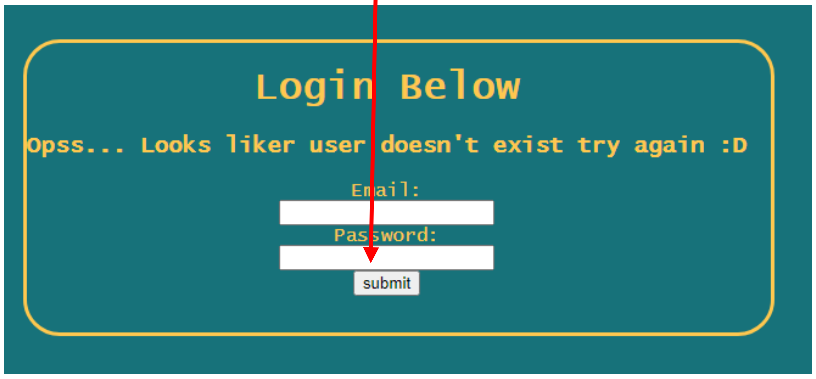


				CustomerIdfr	FirstName	LastName	Gender	Email	Password	AccountType
	Edit	Copy	Delete	21	PremiumTest	LastTest	male	premium@gmail.com	TestTry	Premium
	Edit	Copy	Delete	22	FreeTest	LastTest	male	free@gmail.com	Free	Free
	Edit	Copy	Delete	23	FinalPremiumTest	Premium	female	premium@gmail.com	mnbv	Premium



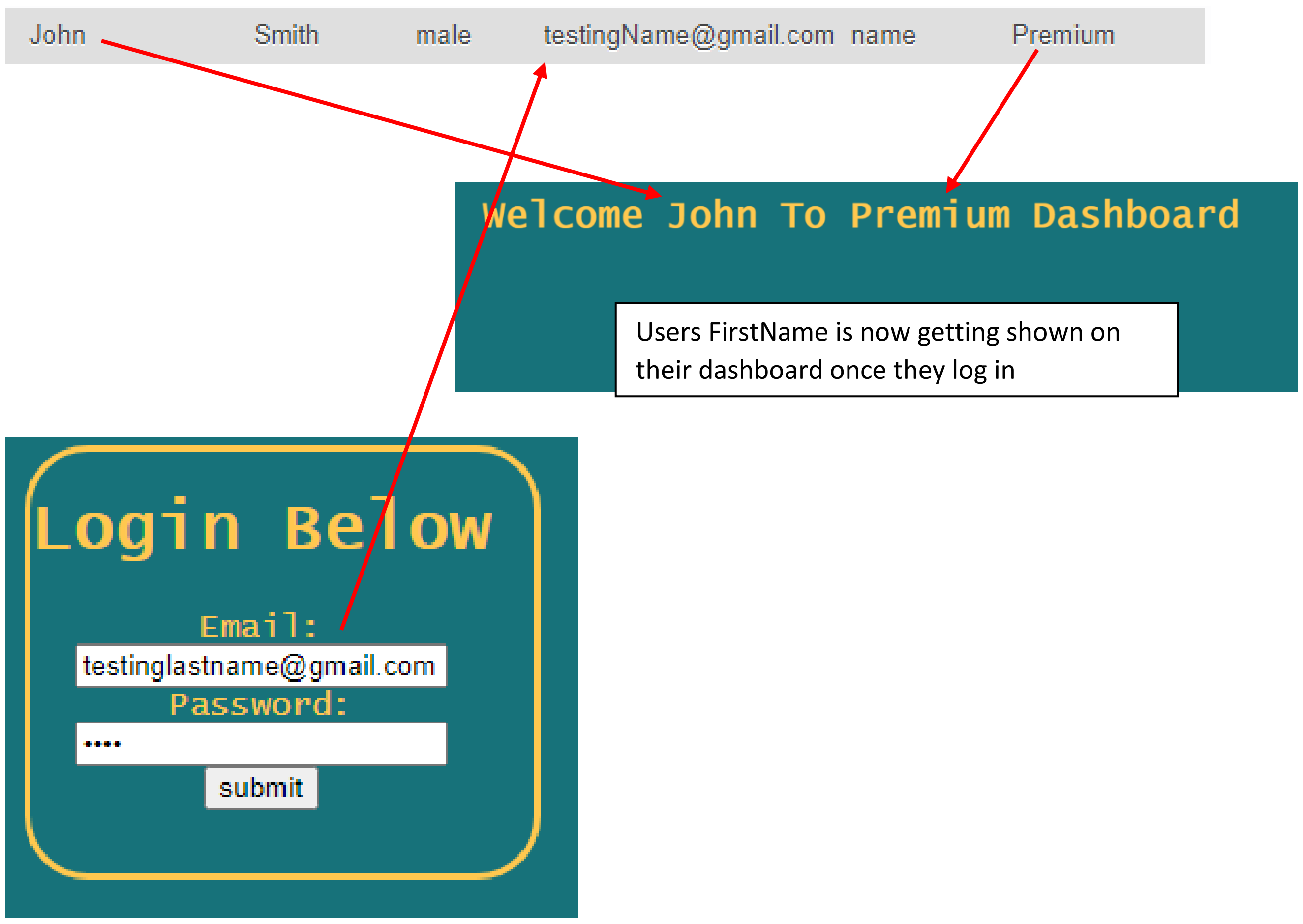
No email 'nonexistant@gmail.com' exists in the database therefore should not be found

User has not been found therefore giving them another page user does not exist



Opss... Looks liker user doesn't exist try again :D

2.1 Displaying users name in the dashboard



John	Smith	male	testingName@gmail.com	name	Premium

Welcome John To Premium Dashboard

Users FirstName is now getting shown on their dashboard once they log in

Login Below

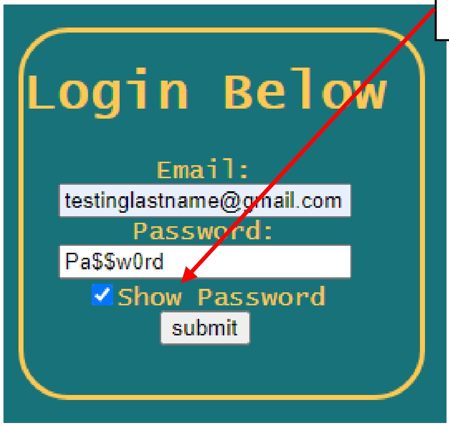
Email: testinglastname@gmail.com

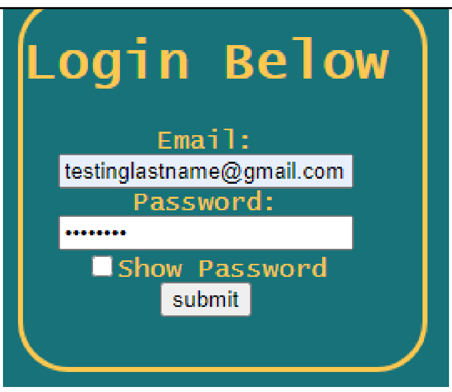
Password: ••••

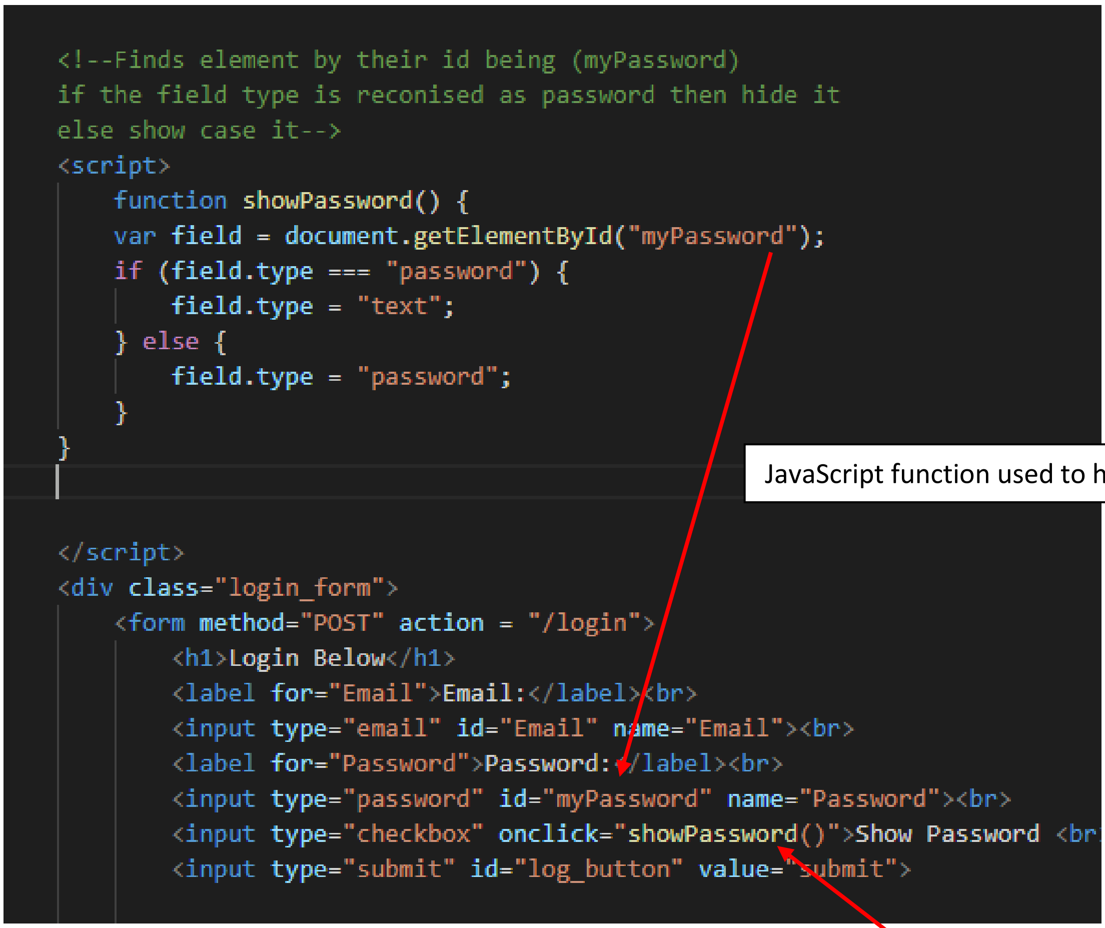
submit

2.2

Show password box working once its been clicked by user





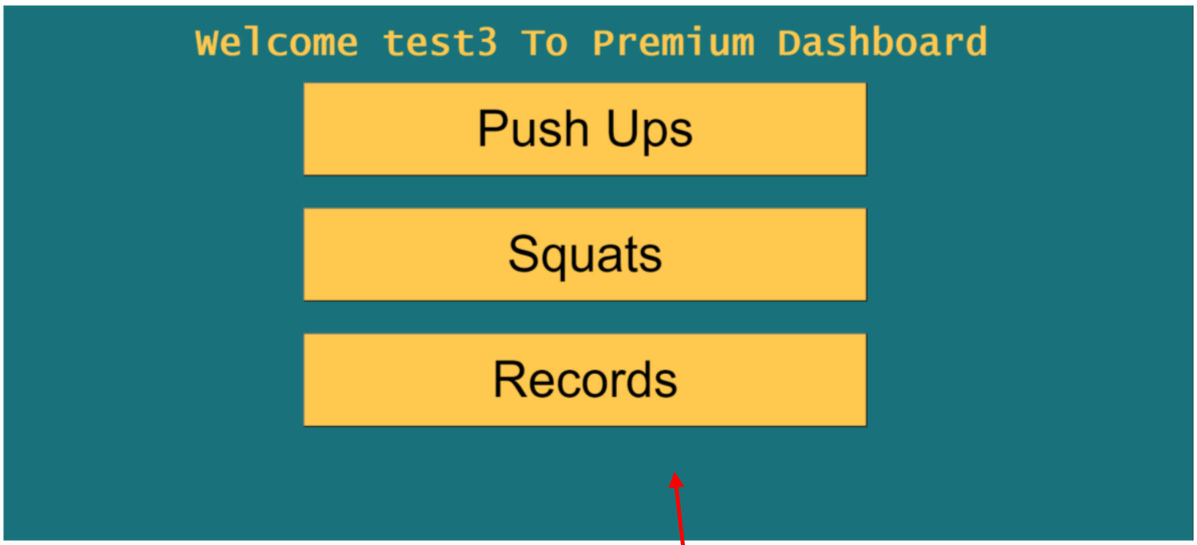


```
<!--Finds element by their id being (myPassword)
if the field type is reconised as password then hide it
else show case it-->
<script>
    function showPassword() {
    var field = document.getElementById("myPassword");
    if (field.type === "password") {
        field.type = "text";
    } else {
        field.type = "password";
    }
}


</script>
<div class="login_form">
    <form method="POST" action = "/login">
        <h1>Login Below</h1>
        <label for="Email">Email:</label><br>
        <input type="email" id="Email" name="Email"><br>
        <label for="Password">Password:</label><br>
        <input type="password" id="myPassword" name="Password"><br>
        <input type="checkbox" onclick="showPassword()">Show Password <br
        <input type="submit" id="log_button" value="submit">
```

JavaScript function used to hide password

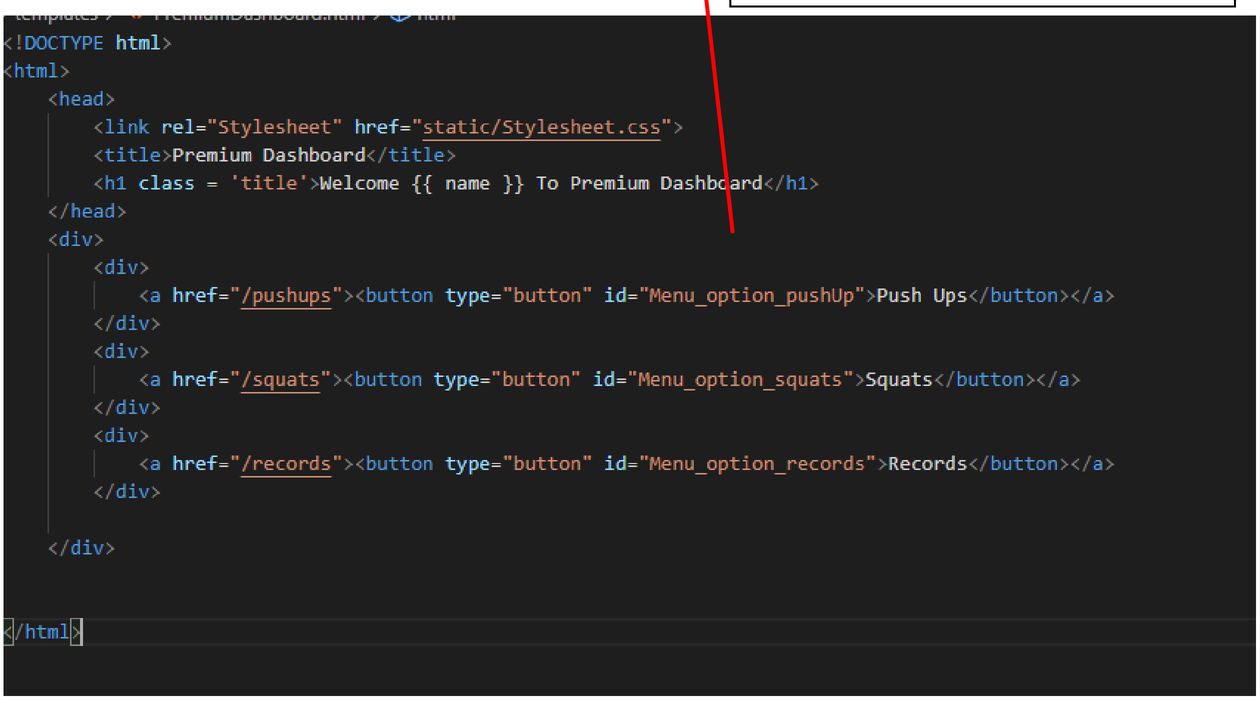
Once clicked on show Password box the function activates

2.3 Creating blocks/buttons which users can press to go to the functional pages



Links to buttons on html code

Pages still need to be made for them



```
<!DOCTYPE html>
<html>
    <head>
        <link rel="Stylesheet" href="static/Stylesheet.css">
        <title>Premium Dashboard</title>
        <h1 class = 'title'>Welcome {{ name }} To Premium Dashboard</h1>
    </head>
    <div>
        <div>
            <a href="/pushups"><button type="button" id="Menu_option_pushUp">Push Ups</button></a>
        </div>
        <div>
            <a href="/squats"><button type="button" id="Menu_option_squats">Squats</button></a>
        </div>
        <div>
            <a href="/records"><button type="button" id="Menu_option_records">Records</button></a>
        </div>

    </div>


</html>
```

Log table

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
2(BL)	2.4(bl)	Integrated testing	Testing to see if database can select the items from the database and check against user login information and if it matches procced them to their dashboard	Users' credentials should get authenticated and processed depending on if they match	The users can now enter their details in the login form and once they click submit it authenticates their details by checking the database and if exists sends them to the dashboard, they registered for	Variables were not lowercase instead of being uppercase, so form did not pick them up.
1	1.1	Unit testing Load testing (lots of details sent across and checked against)	Testing to see if user is getting sent to the correct dashboard	Users should be sent to the correct dashboard by their credentials	User is getting sent to either Free or Premium Dashboard depending on account they registered for	Only works while login in
	1.2	Unit testing User acceptance testing	Testing to see if the user doesn't exist are they still able to enter the dashboard through login	People who have not registered should not be able to log in	If a user tries to use email or password which doesn't exist in the database, they will be shown an error screen asking them to fill the details in again to confirm	If user did not enter correct details or entered blank, they will be greeted with fail page
2	2.1	Integrated testing	Testing to see if getting users name from database and trying to display it on dashboard to see if it works all well	User's name should be displayed on the dashboard title	Users FirstName is now getting displayed in their dashboard once they log in	Only works while login in
	2.2	Unit testing	Additional feature which allows users to click on show password checkbox to see their own password	Users should be able to click on a checkbox and see their password	Users can click on check box and see their password	Uses JavaScript For checkbox
	2.3	Unit testing Functional testing	Testing to see if buttons clicked go to their pages but still have no content on them/ testing to see if users can see buttons and are able to click on them	Once user logs in checking to see if they can click any of the dashboard buttons created and go to the pages	At the moment users can only see 3 buttons and once clicked on them they get internal server error (no pages made for them)	Pages will have to be made for each other content and functionality for sprint 4

Review

Sprint 3 was very challenging in many ways as I was trying to catch up on backlog task which took a very long time, however on good news I managed to complete it to a high standard which I wanted to. Sprint 3 was hard as I was mostly trying to figure out and program on how to verify users' information and giving them a cookie once they manage to log in. However, this wasn't hard to make as all I had to do was create a cookies table as well as getting users email and their customerid and encoding their email which would then be hex digested into random letters and numbers which would then act as a cookie in the cookies table. One minor inconvenience was then I was trying to retrieve user's login details I couldn't as the variables set were different to the ones, I was trying to collect meaning that my controller was picking up no data. However, this was quickly fixed by just changing the capital letters of the first variable making it now collect users' details

Displaying the name on dashboard was hard as I first had to make sure the correct user is in the correct dashboard by checking all their credential details and authenticating them with accounts made. However, to showcase the name part, wasn't as difficult as expected as all I needed to do is retrieve their name from the database and display it using cookies to get the user logged in identity. There is a small problem however as of right now to get users name to show up they have to go through login button to see their name displayed, and if they decide to go through register then login it will not showcase their name instead it will be the previous person who logged in. This however can be fixed by making the user always login even once they registered therefore this problem can be easily avoided in future sprints.

Overall, I believe sprint 3 was successful and completed to a high standard meaning that the client has now received a registering function, login function and ability to see different dashboard depending on the account. As a result of this in Sprint 4 I will start to develop the key features in dashboard to make it more entertain for users to see and interact with.

The lesson learnt in this sprint was that when naming variables, I have to make so to remember if I give them a capital letter or not as it may stop me developing and instead make me look for a simple mistake which has been made.

Sprint 4

Sprint 4 is mostly based around functionality of the dashboards such as getting push-ups,squats, blogs and records pages up and running with some of its functionality. As well as having to create another table for workouts to be stored in.

Goals/Requirements

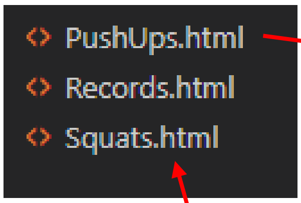
Goal Set	FR#	Requirements
As a user I want to be able to access the features on the dashboards	1	Creating different pages for the buttons on the dashboard
As a user I want to interact with the dashboard functionality	2	Allow user to click on buttons such as (squat button, push up button in those pages) Creating a table for workouts
As a user I want a way to save my workout information	3	Have a save button for each different workout done and save the amount of reps done

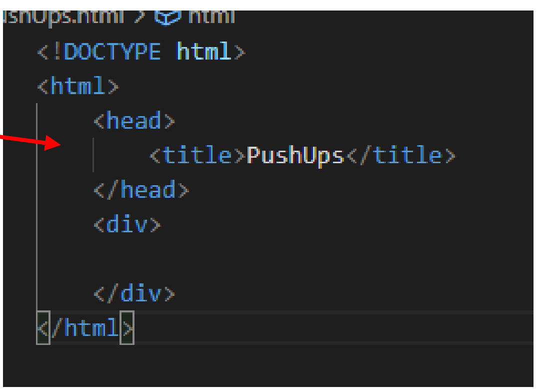
As a user I want to see my workout information	4	Have a records page display users workout progress

Triage/Tasks

FR#	Task #	Functional requirement	Expected outcomes
1	1.1	Create pages for each button	Once clicked on a button user should be sent to the page
	1.2	User can view the page	User should be able to see some colour or items on the page
2	2.1	Create buttons for squats page	User can click on squats button
	2.2	Counter go up when squat button clicked	When user clicks on squat button counter goes up by 1
	2.3	Create button for push-ups	User can click on push-up button
	2.4	Counter go up when push-up button clicked	When user clicks on push-up button counter goes up by 1
	2.5	Create a tblworkouts	Creating tblworkouts with column such as squats and pushups
3	3.1	Create a save button	User can click on a save button
	3.2	Save button saves data in a variable (JS)	Once clicked on save button all the counter information gets saved
	3.3	Tblworkout stores saved information from workout	The tblworkout should store data such as push ups and squats once clicked on save
4	4.1	Create records page with a table	Records page contains a table showing pushups and squats
	4.2	The table has users' information such as squats and pushups,	Table is now storing data from tblworkout such as number of squats and pushups

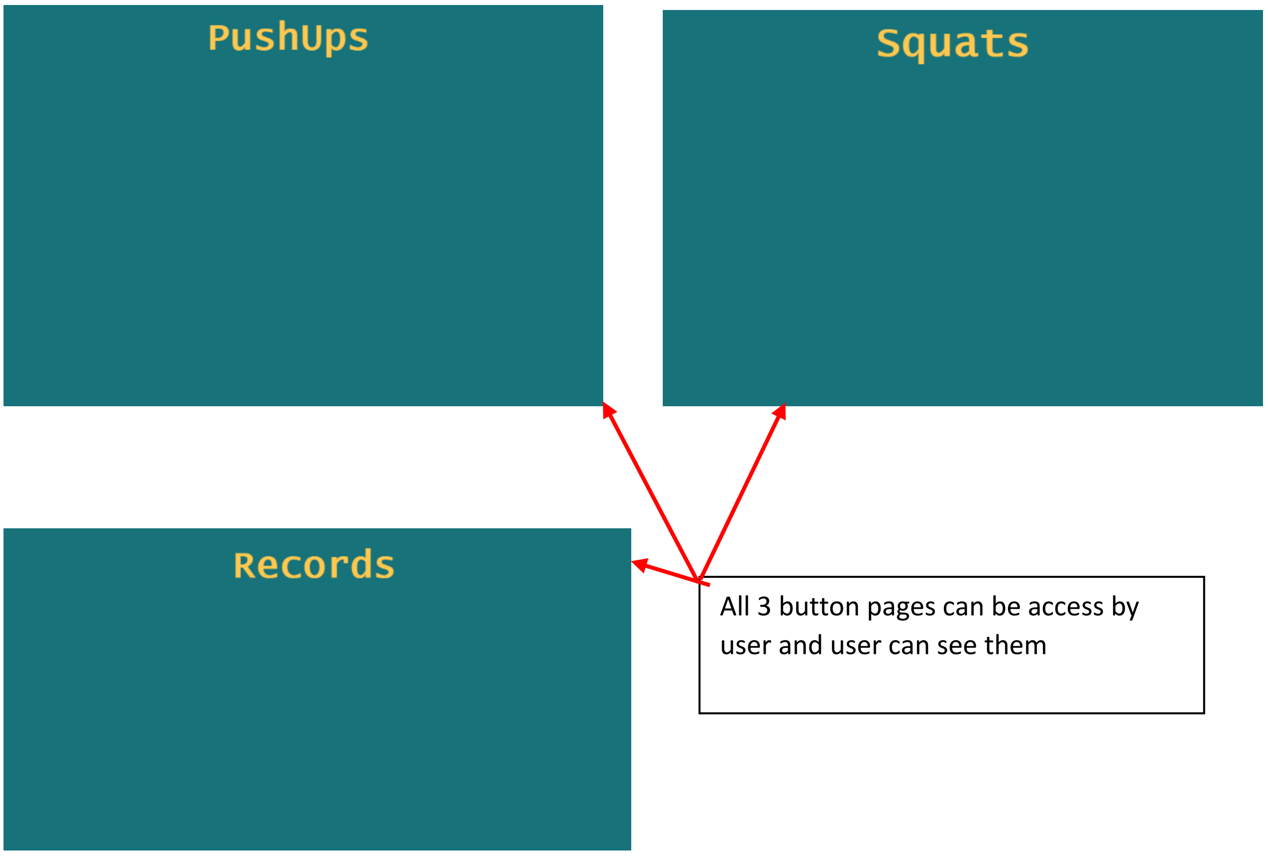
1.1 Create pages for each button




```
<!DOCTYPE html>
<html>
    <head>
        <title>PushUps</title>
    </head>
    <div>

    </div>
</html>
```

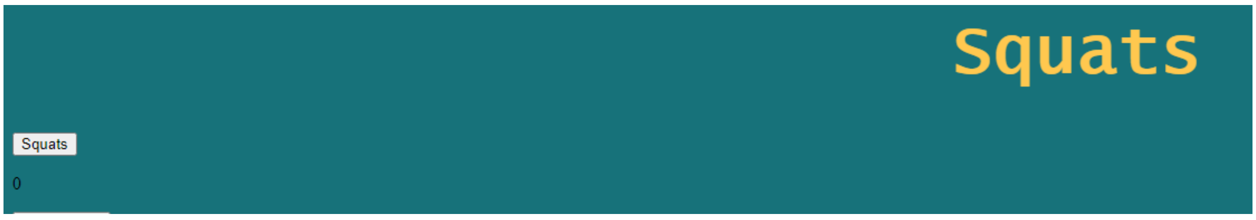
Created pages for each category in dashboard, with at the moment just containing tiles of what page is.

1.2 User can view the page

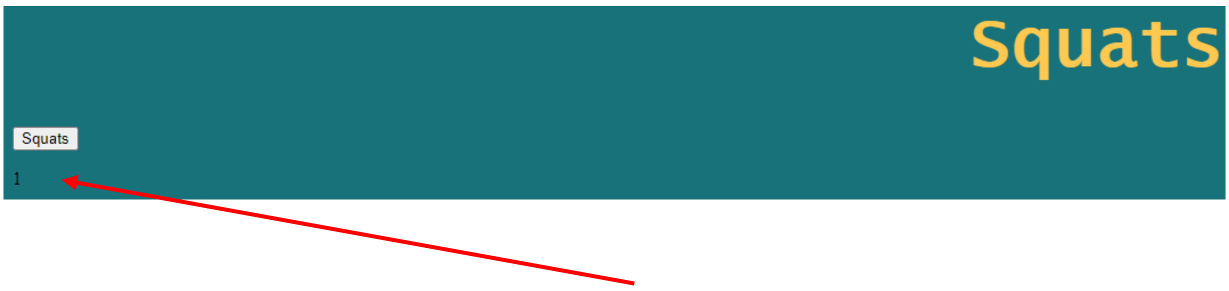


All 3 button pages can be access by user and user can see them

2.1 Create buttons for squats page

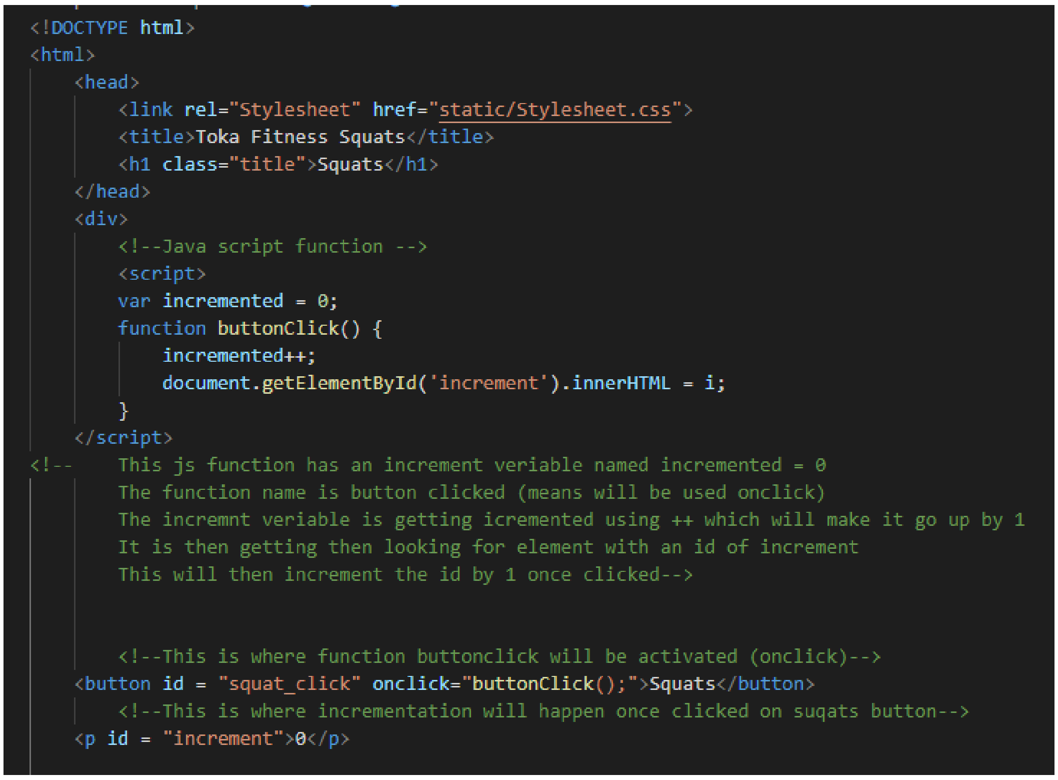


2.2 Counter go up when squat button clicked



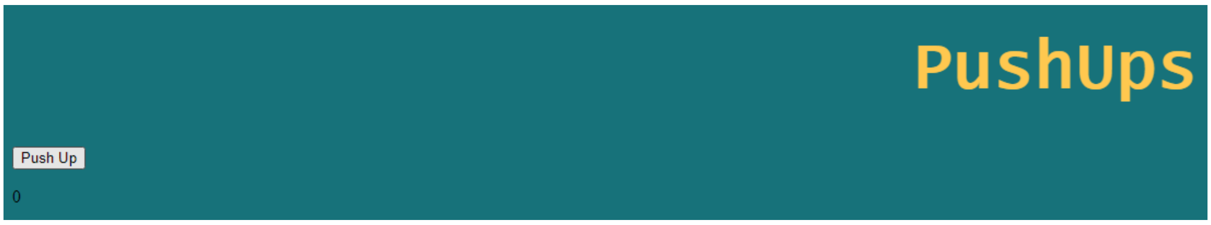
Once clicked on squat button it will increment by 1

code for 2.1 & 2.2

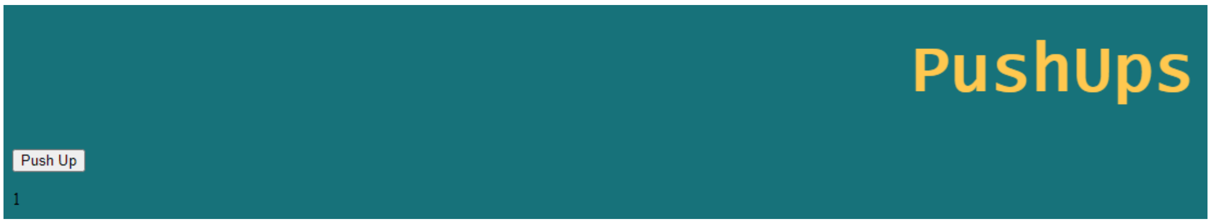


```
<!DOCTYPE html>
<html>
    <head>
        <link rel="Stylesheet" href="static/Stylesheet.css">
        <title>Toka Fitness Squats</title>
        <h1 class="title">Squats</h1>
    </head>
    <div>
        <!--Java script function -->
        <script>
        var incremented = 0;
        function buttonClick() {
            incremented++;
            document.getElementById('increment').innerHTML = i;
        }
    </script>
<!--    This js function has an increment veriable named incremented = 0
        The function name is button clicked (means will be used onclick)
        The incremnt veriable is getting icremented using ++ which will make it go up by 1
        It is then getting then looking for element with an id of increment
        This will then increment the id by 1 once clicked-->


        <!--This is where function buttonclick will be activated (onclick)-->
    <button id = "squat_click" onclick="buttonClick();">Squats</button>
        <!--This is where incrementation will happen once clicked on suqats button-->
    <p id = "increment">0</p>
```

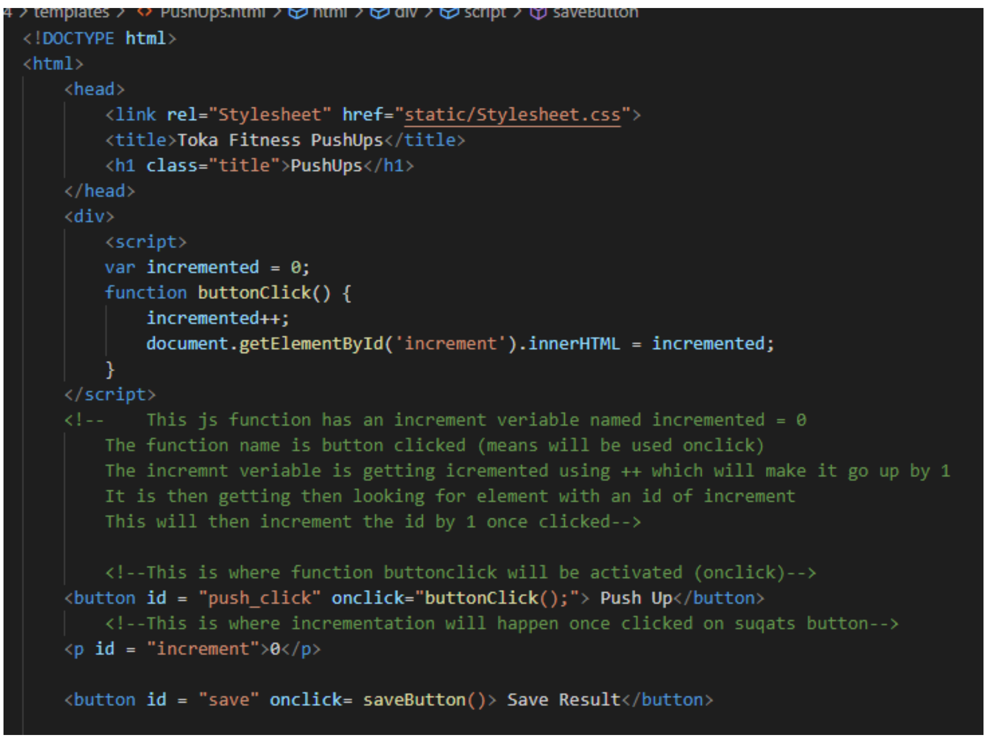
2.3 Create button for push-ups



2.4 Counter go up when push-up button clicked

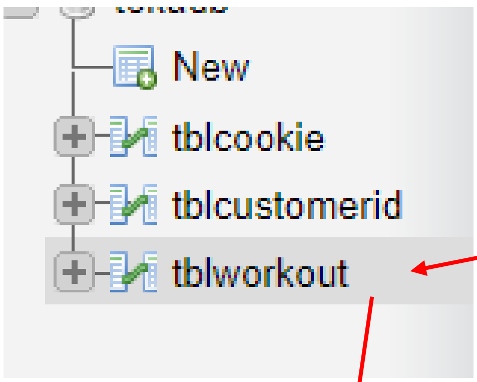


Code for 2.3 & 2.4

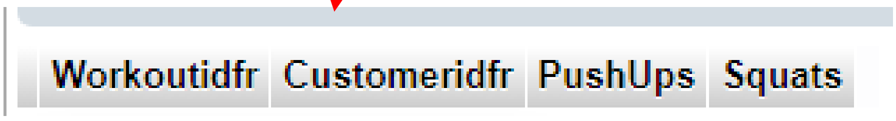


```
4 > templates > <> PushUps.html > html > div > script > saveButton
<!DOCTYPE html>
<html>
    <head>
        <link rel="Stylesheet" href="static/Stylesheet.css">
        <title>Toka Fitness PushUps</title>
        <h1 class="title">PushUps</h1>
    </head>
    <div>
        <script>
        var incremented = 0;
        function buttonClick() {
            incremented++;
            document.getElementById('increment').innerHTML = incremented;
        }
    </script>
    <!--    This js function has an increment veriable named incremented = 0
        The function name is button clicked (means will be used onclick)
        The incremnt veriable is getting icremented using ++ which will make it go up by 1
        It is then getting then looking for element with an id of increment
        This will then increment the id by 1 once clicked-->

        <!--This is where function buttonclick will be activated (onclick)-->
    <button id = "push_click" onclick="buttonClick();"> Push Up</button>
        <!--This is where incrementation will happen once clicked on suqats button-->
    <p id = "increment">0</p>

    <button id = "save" onclick= saveButton()> Save Result</button>
```

2.5 Create a tblworkouts

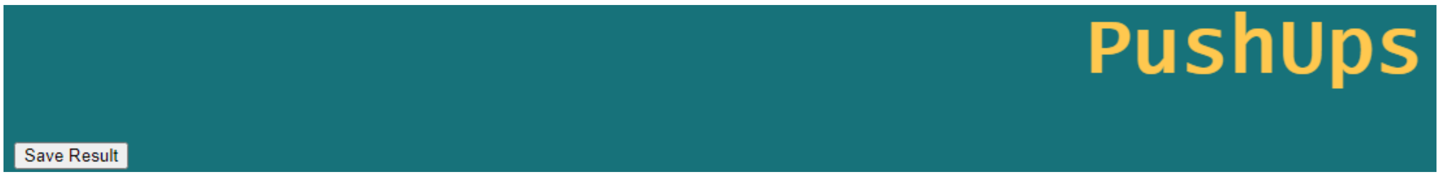


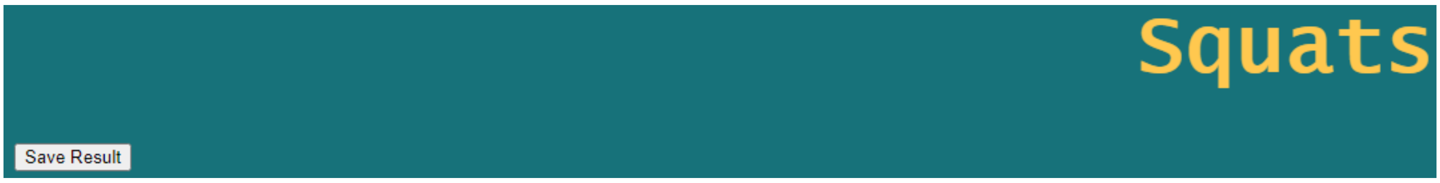
Tblworkout has now been created with columns such as: Workoutidfr, Customeridfr, PushUps, Squats



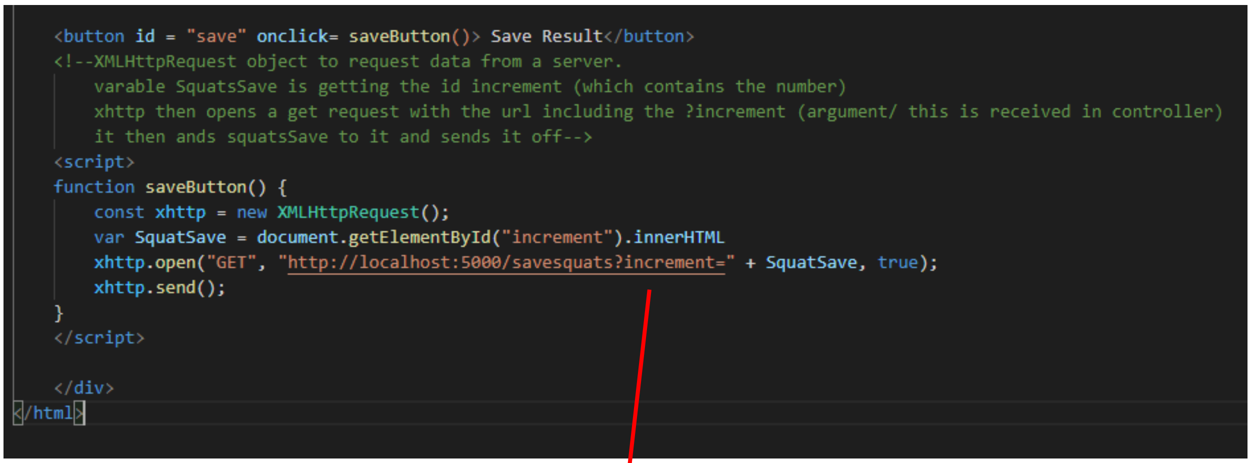
Workoutidfr	Customeridfr	PushUps	Squats

3.1 Create a save button

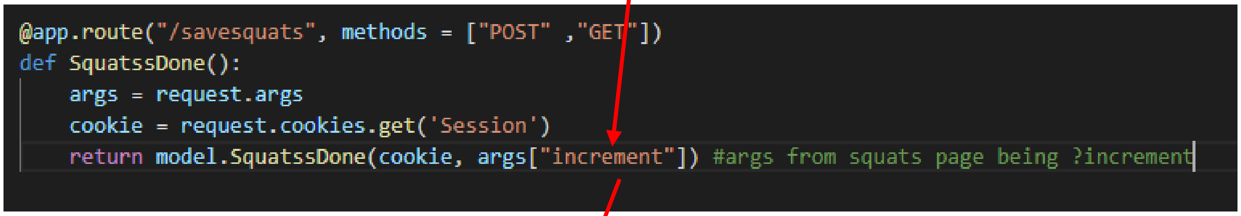




3.2 Save button saves data in a variable (JS)

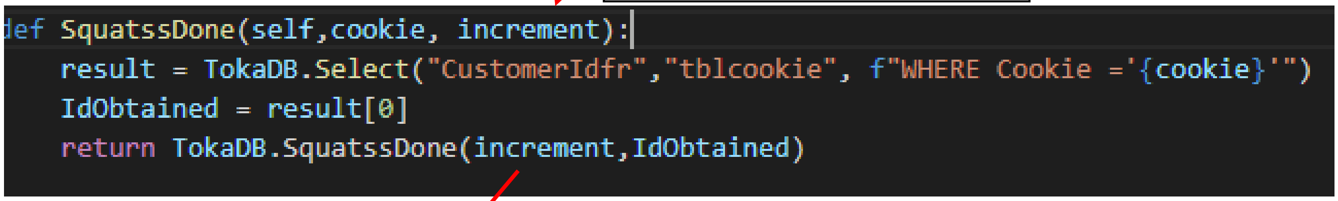


```
    <button id = "save" onclick= saveButton()> Save Result</button>
    <!--XMLHttpRequest object to request data from a server.
        varable SquatsSave is getting the id increment (which contains the number)
        xhttp then opens a get request with the url including the ?increment (argument/ this is received in controller)
        it then ands squatsSave to it and sends it off-->
    <script>
    function saveButton() {
        const xhttp = new XMLHttpRequest();
        var SquatSave = document.getElementById("increment").innerHTML
        xhttp.open("GET", "http://localhost:5000/savesquats?increment=" + SquatSave, true);
        xhttp.send();
    }
    </script>

    </div>
</html>
```



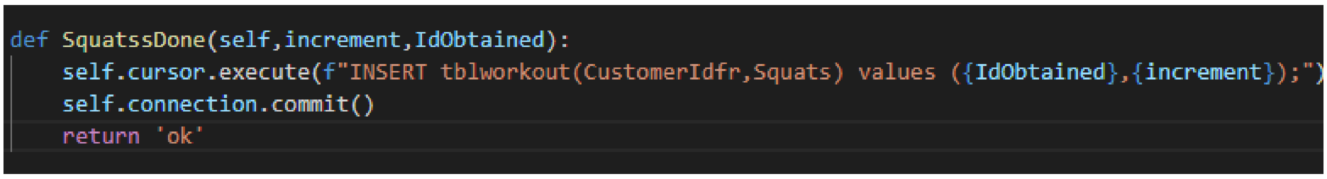
```
@app.route("/savesquats", methods = ["POST" ,"GET"])
def SquatssDone():
    args = request.args
    cookie = request.cookies.get('Session')
    return model.SquatssDone(cookie, args["increment"]) #args from squats page being ?increment
```

Increment argument is getting passed as an increment variable



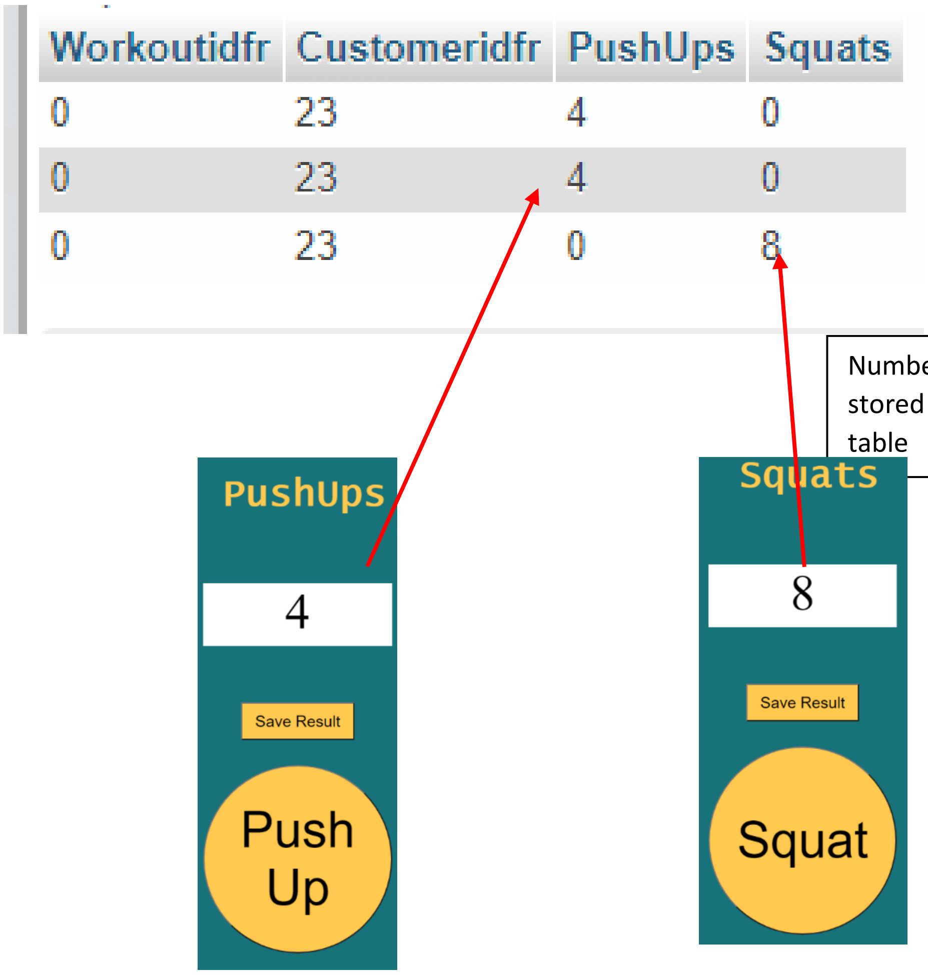
```
def SquatssDone(self,cookie, increment):
    result = TokaDB.Select("CustomerIdfr","tblcookie", f"WHERE Cookie ='{cookie}'")
    IdObtained = result[0]
    return TokaDB.SquatssDone(increment,IdObtained)
```

Inserting values into table workouts this column being the squats done



```
def SquatssDone(self,increment,IdObtained):
    self.cursor.execute(f"INSERT tblworkout(CustomerIdfr,Squats) values ({IdObtained},{increment});")
    self.connection.commit()
    return 'ok'
```

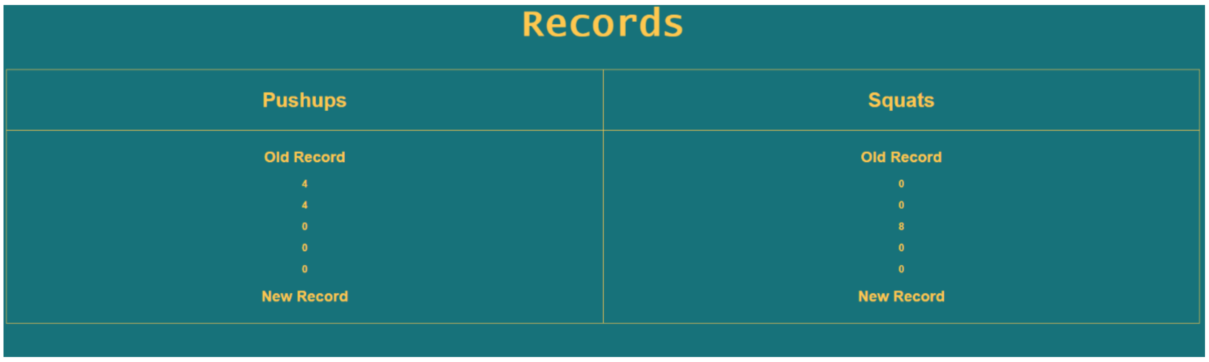
3.3 Tblworkout stores saved information from workout



Workoutidfr	Customeridfr	PushUps	Squats
0	23	4	0
0	23	4	0
0	23	0	8

Number of reps has been saved and stored in the table under its specific table

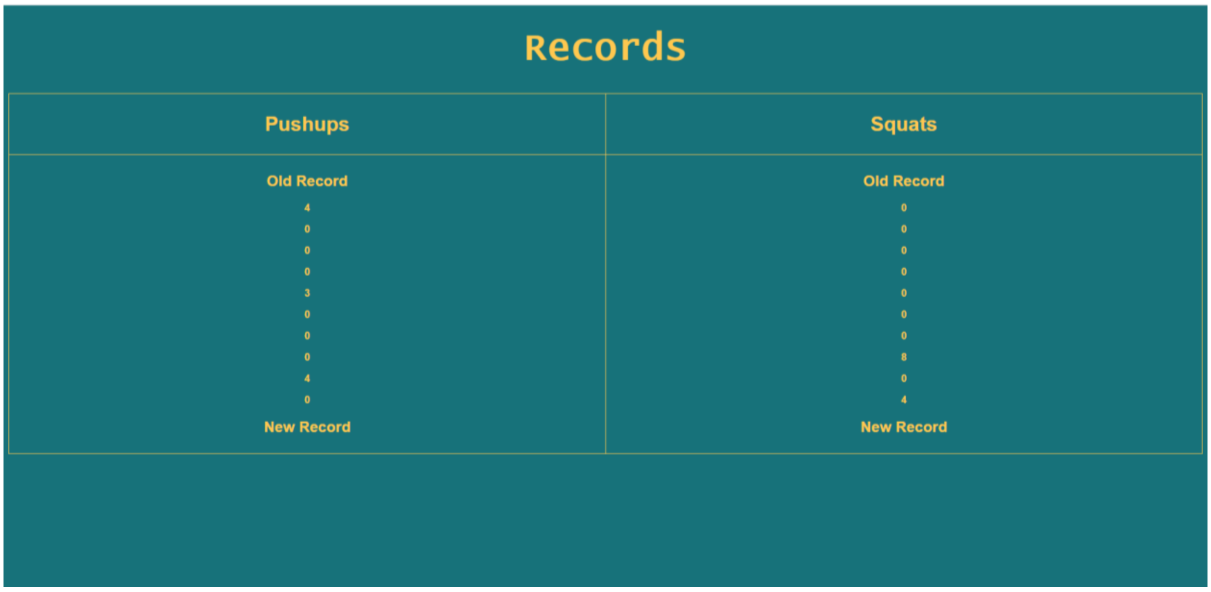
4.1 Create records page with a table



Records

Pushups	Squats
Old Record	Old Record
4	0
4	0
0	8
0	0
0	0
New Record	New Record

4.2 The table has users' information such as squats and pushups,



Records

Pushups	Squats
Old Record	Old Record
4	0
0	0
0	0
0	0
3	0
0	0
0	0
0	8
4	0
0	4
New Record	New Record

Log table

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
1	1.1	Unit testing	Checking to see if pages are successfully made	There should be pages made so next test can be done which user should be able to access those pages	Pages have now been built	Needs designing
	1.2	Unit testing Functional testing	Checking to see if user can click on a button and see the page	User should be able to click on button and view page	Pages can now be viewed by users once clicked on button	Pages have now been designed
2	2.1	Unit testing	Making sure buttons created are clickable	User should be able click on squats button	User can click on squats button	
	2.2	Integration testing	Making sure once button is clicked the counter would increase by 1 only and not anymore	Once user clicks on squats button the number 0 should increase by 1	When user clicks on squat button counter goes up by 1	
	2.3	Unit testing	Creating a button so making sure use can click and see the page	User should be able click on pushups button	User can click on push-up button	
	2.4	Integration testing	Making sure once button is clicked the counter would increase by 1 only and not anymore	Once user clicks on push ups button the number 0 should increase by 1	When user clicks on push-up button counter goes up by 1	
	2.5	Integration testing	Making sure tblcookie and tblworkout can communicate by getting customers ids and storing only their specific information	Making tblworkouts in MySQL with all fields necessary to store information in them	Creating tblworkouts with column such as squats and pushups	*tblworkout is created
3	3.1	Unit testing	Making sure save button can be pressed	User should be able to click on save	User can click on a save button	
	3.2	Integration testing	Once clicked on save button all the counter information gets saved	Once clicked on save button all the counter information should get saved	Once clicked on save button all the counter information gets saved	
	3.3	Integration testing	The tblworkout should store data such as push ups and squats once clicked on save	The tblworkout should store data such as push ups and squats once clicked on save	Tblworkout stores saved information from workouts	

4	4.1	Unit testing	Making sure that there is a table for information to be stored in	Records page contains a table showing pushups and squats	Records page now shows pushups and squats as well as results from user	
	4.2	Integration testing	Looking to see if the correct data is showing for the correct user such as is the correct number of pushups, and squats done by user correct.	Table should be storing data from tblworkout such as number of squats and pushups from other peoples too	The table now stores all users results and can be added to it if other users save their items.	

Review

Sprint 4 was easier than the other sprints as most of the things I had to do was creating pages for users to view and functionality of them which mostly included JavaScript such as getting the incrementing number when pressed on a squat button. I was also able to create tblWorkout which stores all the necessary fields such as squats and push ups. I was then able to make tblworkout and tblcustomer to communicate by getting the customer id and saving their workouts to that person.

However, I did face some challenges such as getting the table to show up and splitting the squats from push ups in a table as they all just went under each other making a massive line going down. Another difficulty I faced was getting nice to the eye colours to show up on the table as I tried keeping table in black colour, but it was just not appealing to look at and it was quite hard to read, as a result of this I changed the entire table to one colour scheme as it follows throughout my application. This however could be worked on in the future to get the best design and choice of colours which would appeal to the user more than just plain simple colours.

Another major change has occurred of deleting the show password function using JavaScript. I have done this due to issues with login and variable naming requirements. I removed this feature as I believe its not needed and when user tried to login it would sometimes mess their password up making them no long able to login in as a result this feature is no longer going to exist in my product. But could however come back in the future development stages.

Overall, I believe sprint 4 has been completed to its high standards therefore giving the client another feature which the customers will enjoy by sharing their records and showing to everybody else. Another good thing is that by completing sprint 4 we are on target to have this project finished by the client’s deadline set. If I was able to meet with the client at this point, I would be asking questions such as are the features functioning the way you have desires? does it meet your expectations? any thoughts on the colour scheme and is there any changes you would like me to make? These questions would be vital at this stage of development as it would allow me to understand what the client is thinking about the development of this product.

Lesson learn from this sprint is to get the designs and colour schemes all set out so when it comes to developing a project like this, I would know what colours go well with each over. On the other hand, Toka Fitness has not given or supplied any information about the colour scheme or colours they want to see on the website as a result of this I had to assume throughout my development process

Sprint 5

Sprint 5 will be based around creating blogs page with personal training videos included in it. This will mostly be a html and CSS work and a bit of JavaScript to get the functionality such as read more buttons and getting the YouTube videos to work. As a result of this most of the items to test will be through use of unit test. Blogs feature will only be on Premium dashboard and not free.

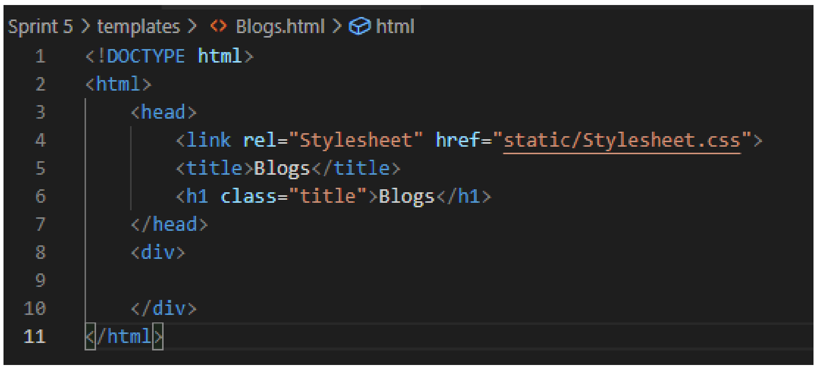
Goals/Requirements

Goal Set	FR#	Requirements
1	1	Having blogs page with readable content
2	2	Having content which has audio/video

Triage/Tasks

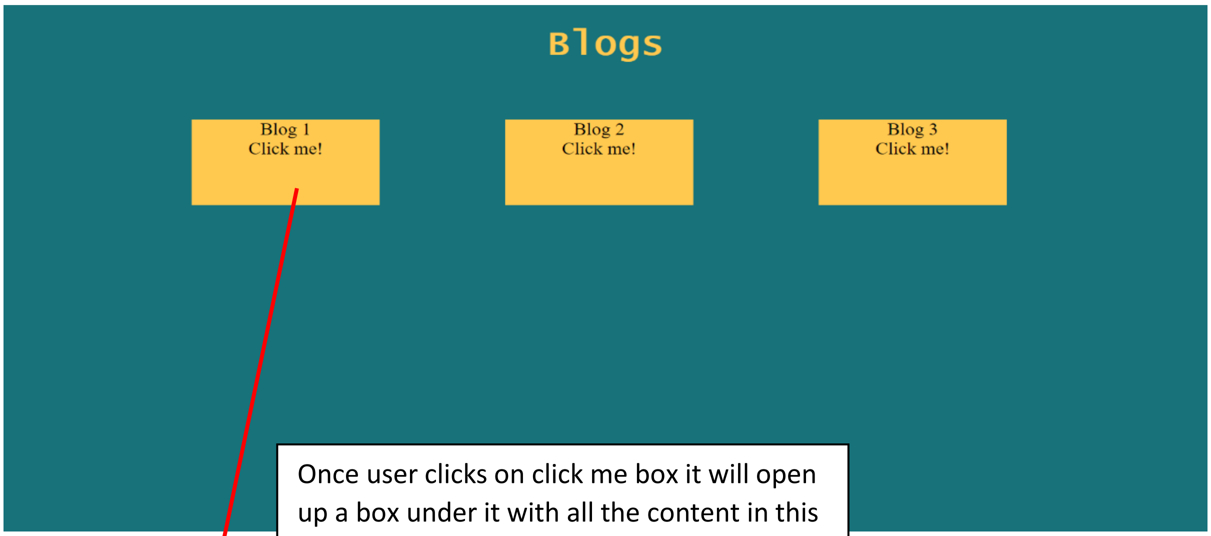
FR#	Task #	Functional requirement	Expected outcomes
1	1.1	Create a blogs page	User can access the blogs
	1.2	Add content to blogs	User can read content/text
	1.3	Add read more button	User can click on read more to see more text
	1.4	Add images to blogs	User can see images
2	2.1	Add video links to pages	User can see YouTube links
	2.2	Make videos playable and watchable	User can watch the YouTube videos

1.1 Create a blogs page

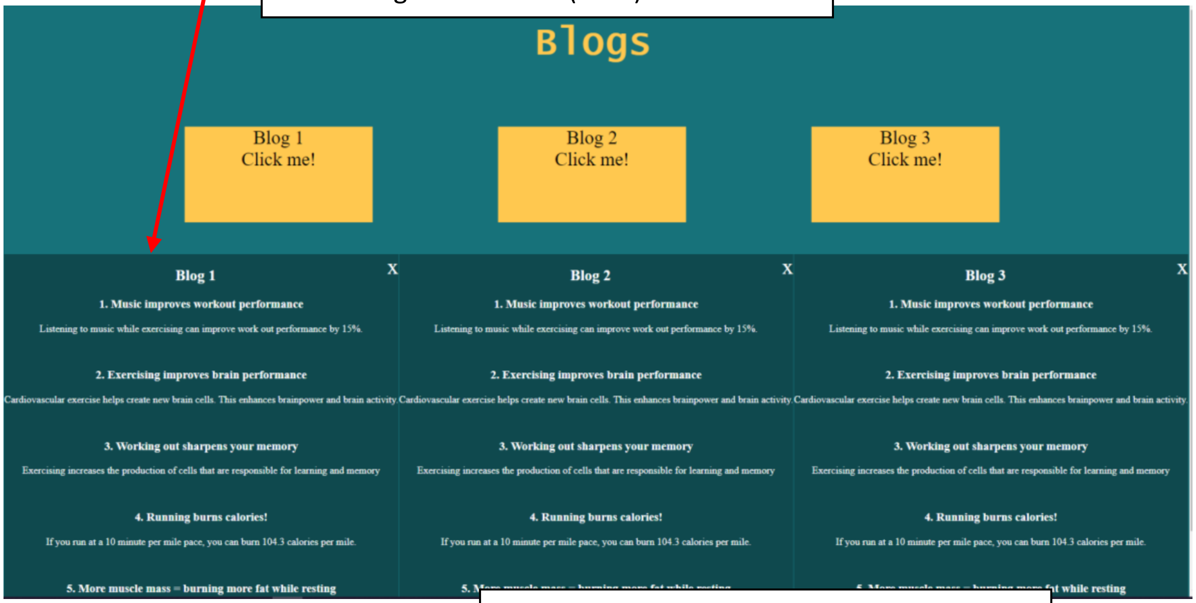

Sprint 5 > templates > Blogs.html > html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="Stylesheet" href="static/Stylesheet.css">
        <title>Blogs</title>
        <h1 class="title">Blogs</h1>
    </head>
    <div>

    </div>
</html>
```

1.2 Add content to blogs & 1.3 Add read more button



Once user clicks on click me box it will open up a box under it with all the content in this case being random text (facts)



Blog 1

1. Music improves workout performance

Listening to music while exercising can improve work out performance by 15%.

2. Exercising improves brain performance

Cardiovascular exercise helps create new brain cells. This enhances brainpower and brain activity.

3. Working out sharpens your memory

Exercising increases the production of cells that are responsible for learning and memory

4. Running burns calories!

If you run at a 10 minute per mile pace, you can burn 104.3 calories per mile.

5. More muscle mass = burning more fat while resting

Blog 2

1. Music improves workout performance

Listening to music while exercising can improve work out performance by 15%.

2. Exercising improves brain performance

Cardiovascular exercise helps create new brain cells. This enhances brainpower and brain activity.

3. Working out sharpens your memory

Exercising increases the production of cells that are responsible for learning and memory

4. Running burns calories!

If you run at a 10 minute per mile pace, you can burn 104.3 calories per mile.

5. More muscle mass = burning more fat while resting

Blog 3

1. Music improves workout performance

Listening to music while exercising can improve work out performance by 15%.

2. Exercising improves brain performance

Cardiovascular exercise helps create new brain cells. This enhances brainpower and brain activity.

3. Working out sharpens your memory

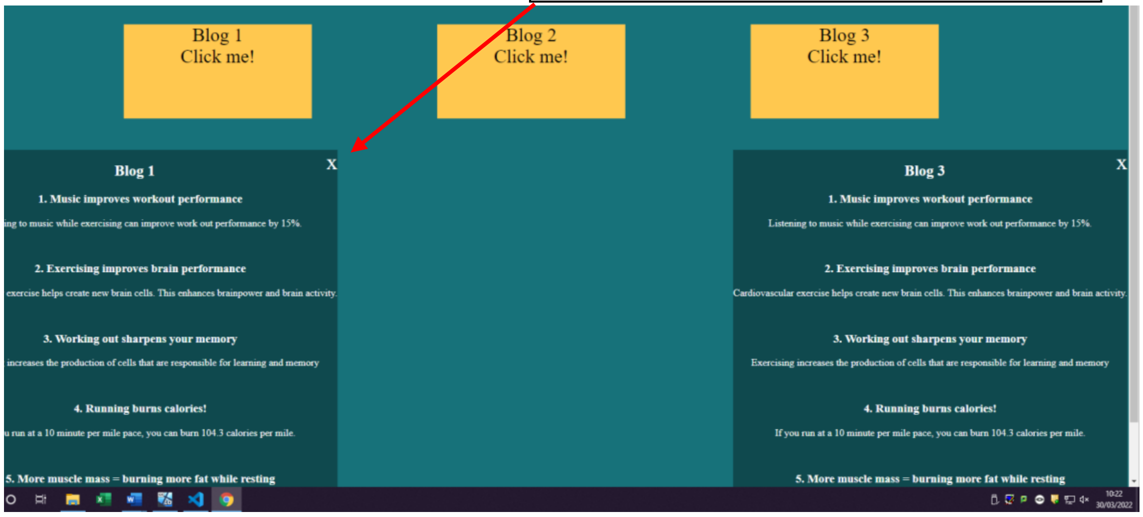
Exercising increases the production of cells that are responsible for learning and memory

4. Running burns calories!

If you run at a 10 minute per mile pace, you can burn 104.3 calories per mile.

5. More muscle mass = burning more fat while resting

User can also press on the x button to close the blogs sub part



Blog 1

1. Music improves workout performance

ing to music while exercising can improve work out performance by 15%.

2. Exercising improves brain performance

exercise helps create new brain cells. This enhances brainpower and brain activity.

3. Working out sharpens your memory

increases the production of cells that are responsible for learning and memory

4. Running burns calories!

u run at a 10 minute per mile pace, you can burn 104.3 calories per mile.

5. More muscle mass = burning more fat while resting

Blog 3

1. Music improves workout performance

Listening to music while exercising can improve work out performance by 15%.

2. Exercising improves brain performance

Cardiovascular exercise helps create new brain cells. This enhances brainpower and brain activity.

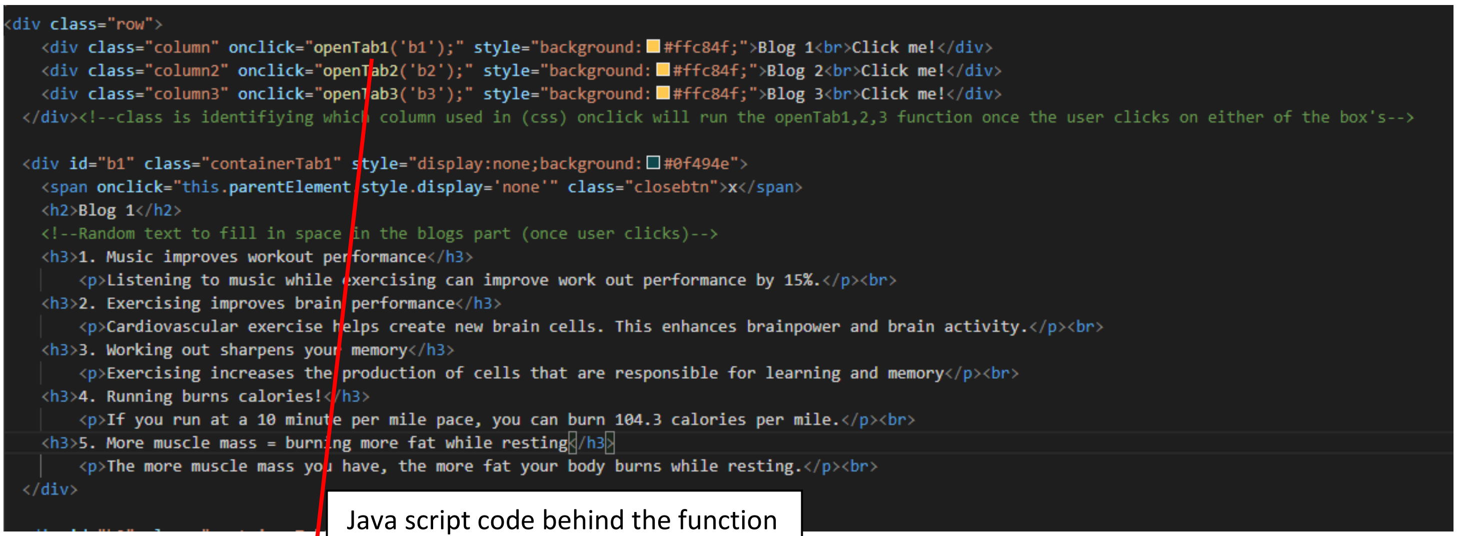
3. Working out sharpens your memory

Exercising increases the production of cells that are responsible for learning and memory

4. Running burns calories!

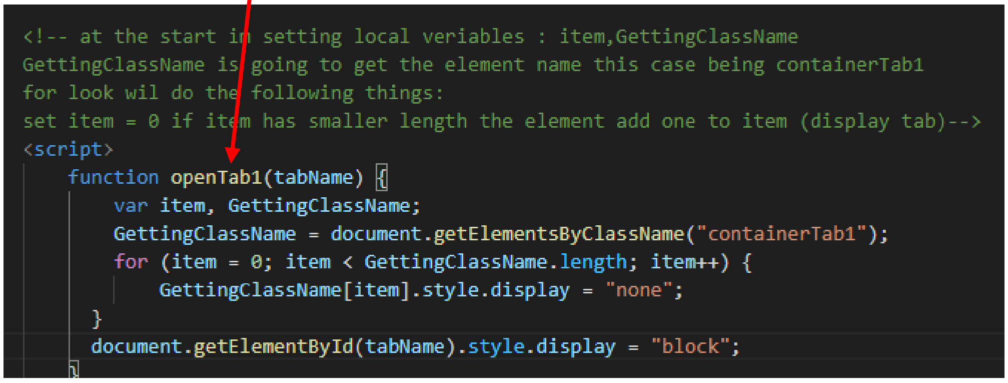
If you run at a 10 minute per mile pace, you can burn 104.3 calories per mile.

5. More muscle mass = burning more fat while resting



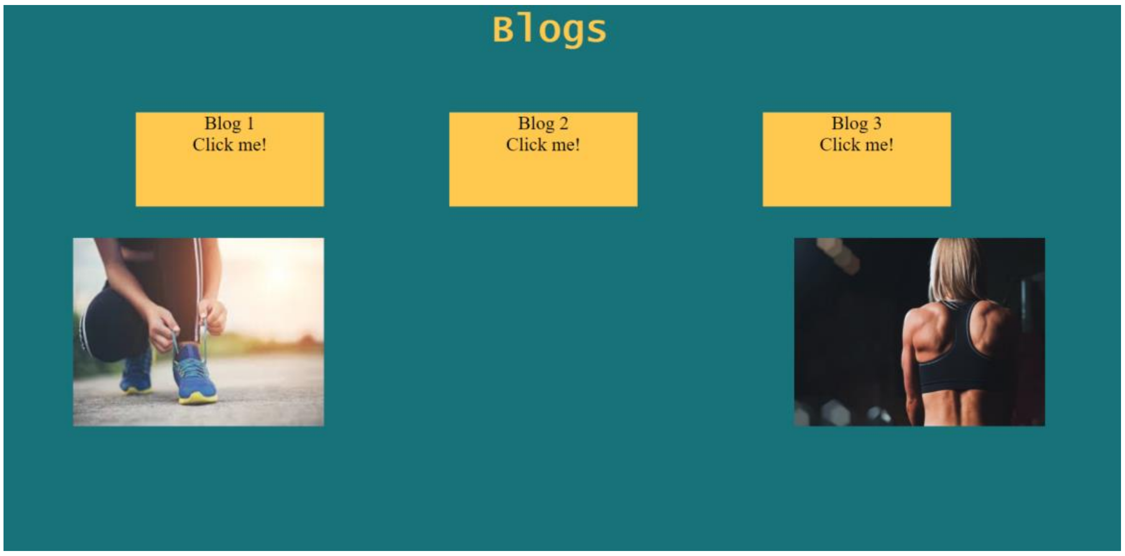
```html
<div class="row">
    <div class="column" onclick="openTab1('b1');" style="background:■#ffc84f;">Blog 1<br>Click me!</div>
    <div class="column2" onclick="openTab2('b2');" style="background:■#ffc84f;">Blog 2<br>Click me!</div>
    <div class="column3" onclick="openTab3('b3');" style="background:■#ffc84f;">Blog 3<br>Click me!</div>
  </div><!--class is identifiying which column used in (css) onclick will run the openTab1,2,3 function once the user clicks on either of the box's-->

  <div id="b1" class="containerTab1" style="display:none;background:□#0f494e">
    <span onclick="this.parentElement.style.display='none'" class="closebtn">x</span>
    <h2>Blog 1</h2>
    <!--Random text to fill in space in the blogs part (once user clicks)-->
    <h3>1. Music improves workout performance</h3>
        <p>Listening to music while exercising can improve work out performance by 15%.</p><br>
    <h3>2. Exercising improves brain performance</h3>
        <p>Cardiovascular exercise helps create new brain cells. This enhances brainpower and brain activity.</p><br>
    <h3>3. Working out sharpens your memory</h3>
        <p>Exercising increases the production of cells that are responsible for learning and memory</p><br>
    <h3>4. Running burns calories!</h3>
        <p>If you run at a 10 minute per mile pace, you can burn 104.3 calories per mile.</p><br>
    <h3>5. More muscle mass = burning more fat while resting</h3>
        <p>The more muscle mass you have, the more fat your body burns while resting.</p><br>
  </div>
```

Java script code behind the function

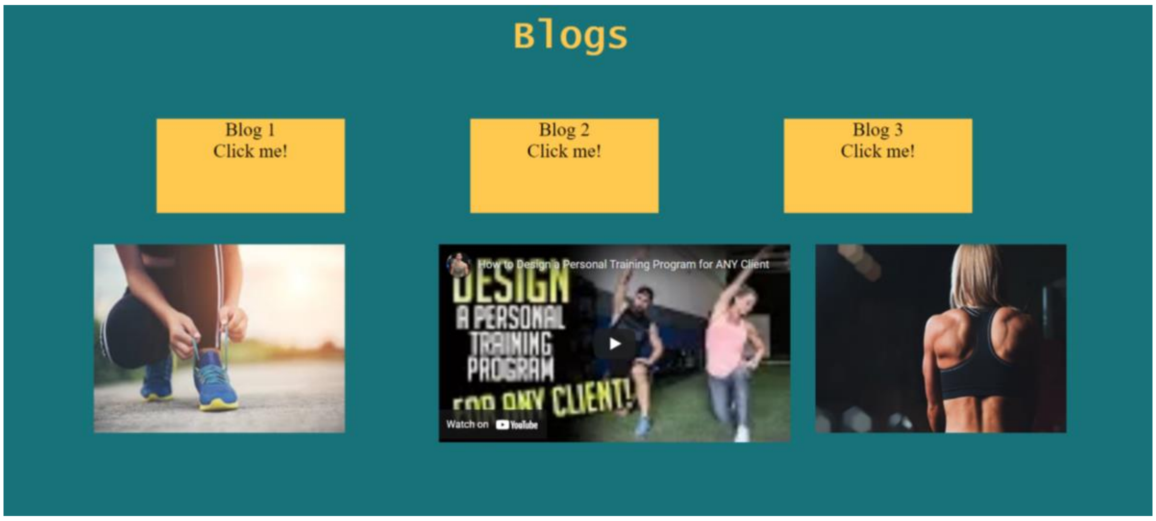


```html
<!-- at the start in setting local veriables : item,GettingClassName
GettingClassName is going to get the element name this case being containerTab1
for look wil do the following things:
set item = 0 if item has smaller length the element add one to item (display tab)-->
<script>
    function openTab1(tabName) {
        var item, GettingClassName;
        GettingClassName = document.getElementsByClassName("containerTab1");
        for (item = 0; item < GettingClassName.length; item++) {
            GettingClassName[item].style.display = "none";
      }
      document.getElementById(tabName).style.display = "block";
    }
```

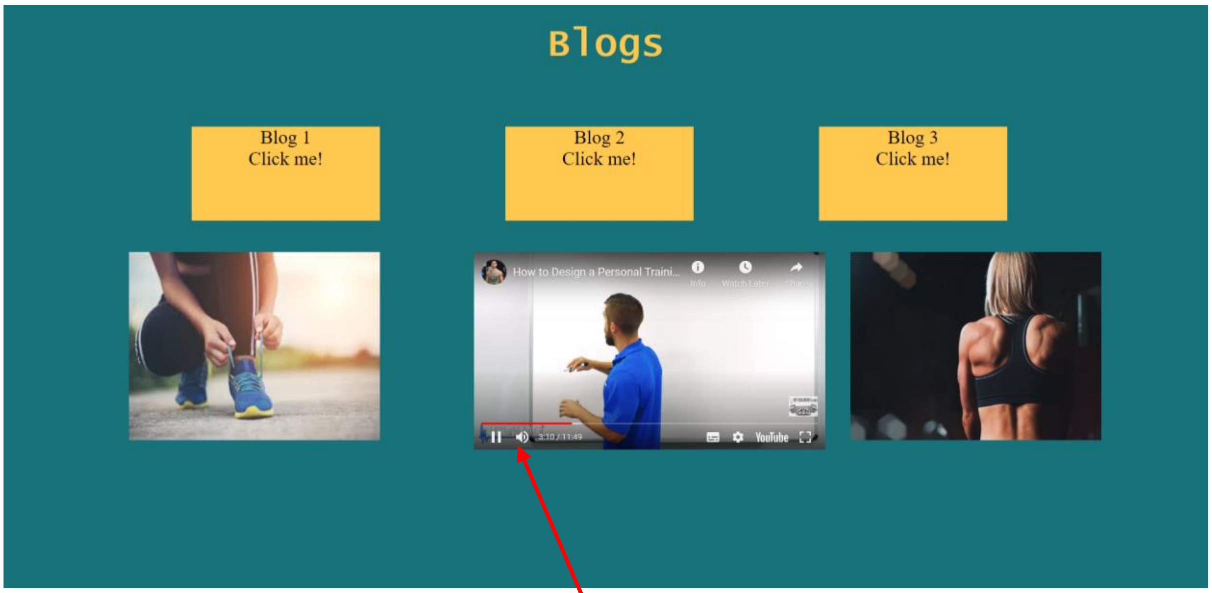
1.4 Add images to blogs



2.1 Add video links to pages


Blogs
Blog 1
Click me!
Blog 2
Click me!
Blog 3
Click me!
How to Design a Personal Training Program for ANY Client
DESIGN A PERSONAL TRAINING PROGRAM FOR ANY CLIENT!
Watch on YouTube

2.2 Make videos playable and watchable



Video is now playable, and user has controls to pause it and go full screen

Log table

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
1	1.1	Unit test	Creating blogs page	Blogs page should be created	Blogs page has now been made	Not yet displaying to user
	1.2	Unit test	Adding content to the blogs page	There should be some sentences or paragraphs of information	User can now see content on the blogs page (random content)	Blogs page is now displaying to the user
	1.3	Unit test	Testing read more button to see if extra text appears	When user clicks on read more button, they can see extra text on the screen.	Instead of button its now a box once clicked opens up sub box for more information	
	1.4	Unit test	Checking to see if images can be seen by the user and if they have good resolution	Users should be able to see the images in good quality	Images are from google with creative commons license meaning we are allowed to use them. The images are also high quality	
2	2.1	Unit test	Checking if the YouTube links selected are working	User should be able to see YouTube videos	YouTube link is just an example of how and where the video would be placed on the blogs page (but reference has been made once clicked it goes to creators channel)	
	2.2	Unit test	Testing to see if video's function when clicking play such as is it running and can you hear sound	User can watch the YouTube videos	User has all controls over the video including pause and full screen	

Review

Sprint 5 was the easiest sprint to complete as most of the work was just designing on how blogs page will look and things it will contains. Blogs page is a great addition but only for premium members as this is the feature which sets Premium account and Free differences.

In functional requirements I said I will make a read more button however as I was designing, I turned button into a box which can be clicked on. I done this because I believe its more appealing to the user as well as would look better than having a normal button under. I also changed the way my blogs will be seen, instead of blogs going straight under each box they are now separated meaning if the user decides to close 1 blog, they can through clicking the ‘x’ letter/button they can. As well as this gives the user an opportunity to have all blogs open at the same time without automatically closing the previous one. The text, video and images are all examples of how things may look like once Toka Fitness decides to put their content in.

All together I believe sprint 5 was successfully complete giving another addition to Toka Fitness web application and meaning that we are getting close to finishing this development project set by Toka.

Lessons learnt from this sprint are to have some CSS code as examples which would give me more time by not having to google on how to do a set thing instead focusing more on the design with the CSS I already have from other webpages and trying to reuse them for blogs.

Sprint 6

Sprint 6 will be finishing all the small details such as the designs and cleaning up all the code as well as to make it more maintainable in the future, adding comments on how things work and function

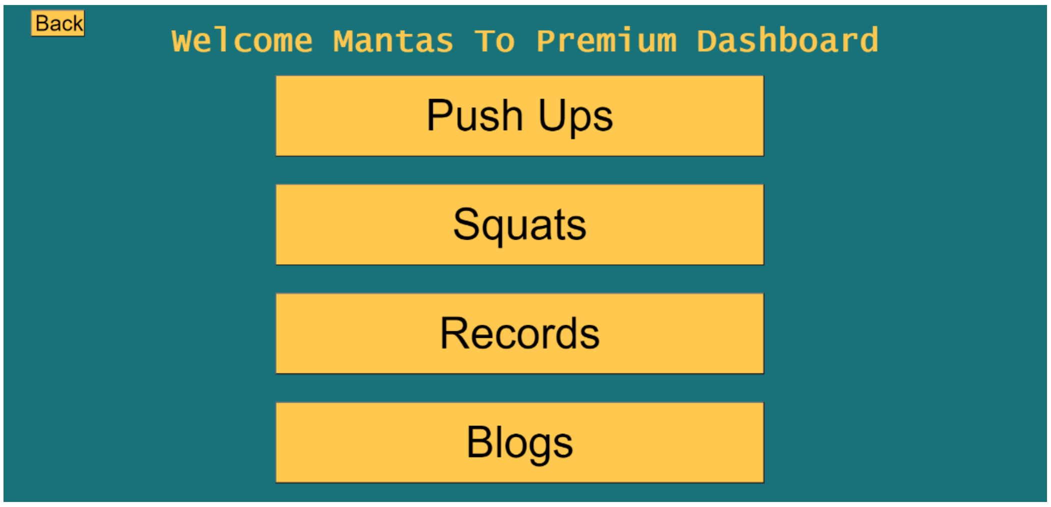
Goals/Requirements

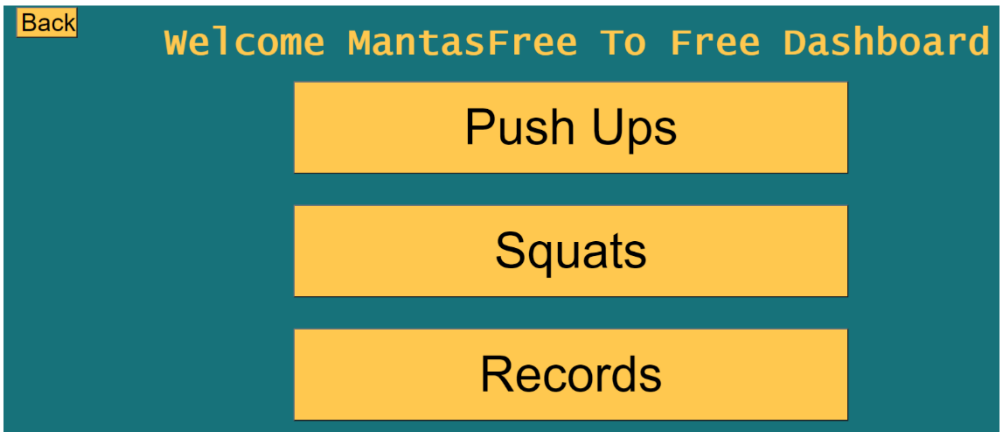
Goal Set	FR#	Requirements
1	1	Finish the designs of pages such as (back buttons)
2	2	Comment the entire code on what it does

Triage/Tasks

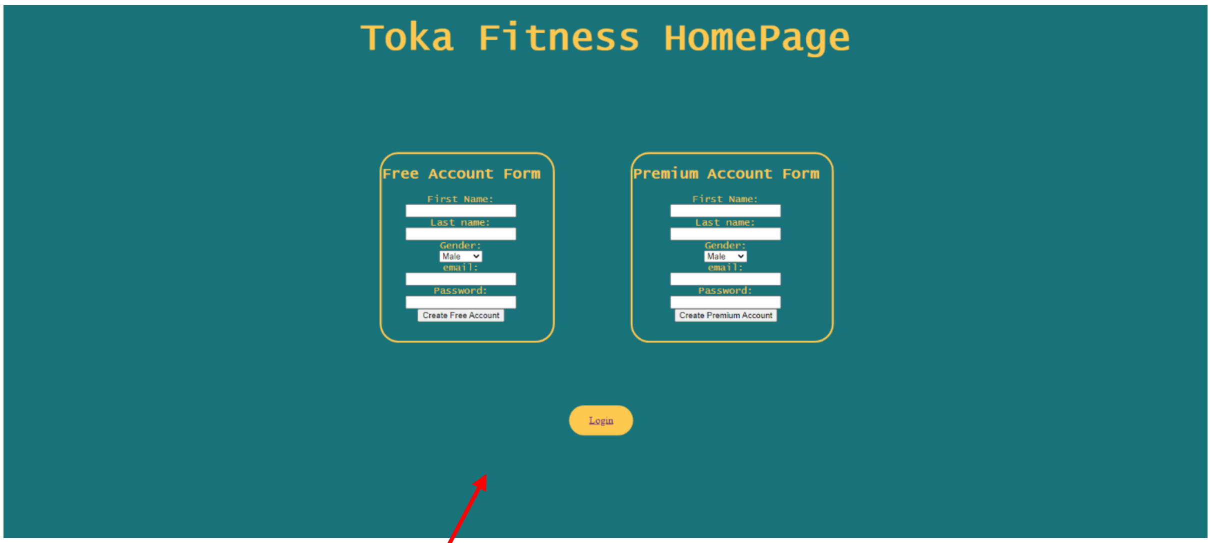
FR#	Task #	Functional requirement	Expected outcomes
1	1.1	Add back buttons to dashboard	User should be able to see and click on back buttons on each dashboard page
	1.2	Make back buttons go to homepage	Once user clicks on back button it goes to homepage
2	2.1	Comment each line of code on what it does	-

1.1 Add back buttons to dashboard





1.2 Make back buttons go to homepage



Once user has clicked on back button it sends them back to homepage

Log table

Fr#	Task#	Testing type	Description of test	Expected outcome	Actual outcome	Notes
1	1.1	Unit test	Checking to see if buttons are correctly places and can be seen by user and clicked	User should be able to see and click on back buttons on each dashboard page	Users can see and are able to click on back button but get frown internal server error as no page is yet getting returned	Throws internal error
	1.2	Integrated test	Checking to see if back buttons go back a page	Once user clicks on back button it goes to homepage	Users can now click on back button to go to homepage	
2	2.1	-	Just making sure all lines of code are commented to say what they do	-	Most of code is commented	

Review

Sprint 6 was the last sprint in this development process which means that the prototype for Toka Fitness is now in the final finished stage based of the requirements set by the client.

Sprint 6 had one user requirement which was having back buttons for dashboard pages allowing users to go from dashboard to homepage. This has been successfully complete; however, I did try making back buttons for every page but there would be a problem which would have occurred such as getting pages mixed up with premium and free dashboards. Example of this would if user is using a free account and is on Push Ups page and they click back they would have a 50/50 chance of either going to premium dashboard or free. As a result of this I have not included back buttons on functional pages. In future development of this product, I would have added back buttons which would go to the correct dashboard the user has registered from.

As this is the last sprint one of the requirements was commenting the code which I have completed throughout my development. I set that as a requirement due to the reasons such as if there is going to be another software team wanting to pursue and carry on with development, they would be able to read the comments to understand what is happening. I have also made sure to use proper variable names on what they do so even if they can't read the code, they may understand what it should do it. Each variable is using camelCase which is multiple words used to form a variable without any spaces or underscores, and each word starts with a capital letter. Example of this would be from one of my code snippets: "getRegisterInfo" this is a great way to name variable as it does what the name says gets register information(info).

Lesson learn on this final sprint was to make sure that when developing I should comment each line as I go, this is because I decided to only comment once I have finished a function and only commenting it until it works instead of commenting as I go. If I did comment as I programmed, I would have been able spot the mistakes in the code faster and it would be easier to see what each line does instead of having to read the entire code again and see what is wrong with the code. All together I believe this was a great sprint to finish on by commenting all the lines and getting the last couple features in.

Overview of Development

This part of documentation is all about seeing if I have met clients' requirements, how I done it and my thoughts of how I progressed throughout this agile development cycle

Client Requirements

Toka Fitness needs a new digital solution which would help their company out to communicate with their existing and coming customers as well as helping their employees on completing the tasks. Now Toka Fitness currently provides customers with **personal**

training sessions, advice about fitness training and advice about healthy living. But they are now in need of features such as:

- **provide information and advice about fitness training and healthy living**
- **provide access to digital content to support customers with their training and healthy lifestyle**
- **encourage existing customers to use more of the services provided by Toka Fitness**

Toka Fitness has also carried out market research with existing customers to identify potential features which could be included in the digital solution. Potential features included items such as:

- **free and paid-for content**
- **accessibility features for users with sight loss**
- **'social' features**
- **customisable workout and eating plans**

How Requirements been met

Toka Fitness has requested the solution to **provide information and advice about fitness** training and healthy living. I believe I have managed to achieve this task by creating a Blogs page which would have content such as images/videos and text about healthy living and fitness training advice. To make sure I have fulfilled this requirement I would have had a meeting the client themselves about the blogs page I made and if it contains everything they want and if its what they were thinking of. This meeting would also help in the future development of this blogs page as maybe Toka wants some additional features added to the page or any other design ideas they have.

Second feature which Toka Fitness has requested for their solution to have been a way to **provide access to digital content to support customers with their training and healthy lifestyle.** With this requested I had to think outside the box such as thinking how I should make the user interact with application and support them with the training. So, I decided to create PushUps and Squats functionality which allows user to interact by clicking on the "Push Up" button which would indicate to them that they have done 1 push up, this brings in interactivity with the user/customer and makes it more fun by having the user needing to perform the push up or squat. And once they are done they can click on save result which would send the amount of pushups/squats completed to a socials record page where everyone's results are showing in a table.

Third feature Toka Fitness has requested is a way to **encourage existing customer to use more of the services provided by Toka Fitness**. To try and complete this target I had to think about what the customer would want a Fitness website to contain and for me it was to be able to interact with objects as well as making user feel like they are important. So instead of going usual website way of having just text to read through and forms to fill out I have created dashboards for users. These dashboards display the users name which they have filled in once creating the account. By displaying the customers name in title you giving the customer a feeling that you care about them and that its made for them. Functionality such as Squats and Pushups also encourage the customers to stay and use the services provided in a competitive way with other people. This can be seen through the records page where each user is able to share their high score making other people want to try and beat the score which would keep the existing customer attracted and use more of the services such as blogs page which goes together by receiving tips on how to get in great shape to do the large amount of push ups or squats.

With these three main features Toka Fitness has requested to be complete I believe I managed to achieve this in a unique unusual way making this product stand out from any other fitness website by having more interactive parts for customer to enjoy.

Toka Fitness has also carried out some market research with existing customers to identify potential features which could be included in the digital solution. One of the features was having **free and paid-for content**. With this feature I straight away thought about having two different account types such as Premium being the one which would have to be paid and have the additional content and free which is free and wouldn't have additional content. However, this potential feature was hard to achieve as I was unable to ask users for their bank details this is because the risks of storing users' payment information, I have decided to no longer include any fields requiring a user to enter their private card number information to buy a premium account due to security and Toka Fitness assumably not having a valid or usable e-commerce license to sell products online. This does mean that users are able to create their account in premium form for free too however in future development and/or toka fitness having a valid e-commerce license this would be able to be added to premium forms such as asking for bank details and confirmation emails. As of right now though Premium(paid) version has additional content which is the blogs page allowing users to get their latest fitness information and watch personal training videos.

Another potential feature requested by existing customers was **accessibility features for users with sight loss.** I made sure to have this disability covered as many people are colour blind or have sight have interest in fitness which is a wide range of people. As a result of this I decided to make this web application use as little colours and mostly containing blue and yellow. I decided to use these colours as blue and yellow colour blindness is uncommon for people to have; I have also made sure to have large text as well as big buttons for people to click on in case they cannot read or see any small text/buttons. If Toka Fitness wanted me to change the colour scheme the meeting in Sprint 1 would have been where they would have said it as its where I determined the colour scheme should look like. However, in future development this can still be changed with ease through changing the body in the stylesheet and setting it to the colour Toka wants. However, I do believe the size of text and buttons is appropriate and users with sight loss would not have any trouble directing/moving across my web application.

Customer have also requested the prototype to have a **'social' feature**. I believe I managed to achieve this in a quite unique way which includes interactivity. This is done by users doing the workout in this case being push ups or squats and then saving results which would then be shared on records page. On records page everyone can see everybody else's high scores giving the ‘social’ feature its functionality instead of having the basic share buttons to social media platforms such as Instagram, Facebook or twitter. Instead Toka has a tailored feature just for them and customers/users only to participate in and watch how other people are doing with their high scores.

For the final requirement made by the customers was to have a **customisable workout and eating plans.** This is the requirement which I couldn’t fully fulfil I have instead created two main workout buttons being Squats and Push Ups however this isn’t customisable in a way which customers want. In future development of this project instead of having Squats and Push ups buttons there would be a workout button which acts as a drop down allowing you to select a specific exercise but function the same as others such as having to click on a button, this would then increase the counter and having a share button which would then go on records page with all the other exercises such as squats and push ups and once again giving the ability for other users to see and try beating it. For the eating plan I haven’t included anything with It due to the reasons of this application being interactive and mostly based on workouts done by the user and exercises in general. However, in my proposal I did mention ability to have a ‘calorie counter’ but I have decided to no longer include this as I believe this application doesn’t need one. On the other hand, if Toka Fitness wishes to have this calorie counter it can be added as a feature in future sprints / development.

Development progress review

At the start of Sprint 1 I was able to create all the tasks set out which where the most important ones as this would be everything, I would have been doing my work from things such as Controller and model files there created and even the first Homepage as well as login page. I Also managed to create TokaBase file haven’t tested in which backfired on me in Sprint2.

Sprint 2 was probably the hardest at the start since I was struggling on getting a connection with the TokaBase and MySQL server. This was because When I was creating a user account for the database, I was entering host name of: “192.168.56.101” in the MySQL statement which when if you run on python would give out an error saying: “Error connecting to MariaDB platform: Host 'CC-DT005-03-WCA' is not allowed to connect to”. To overcome this problem all I had to do was give user hostname of ‘%’ meaning any host can access my database now. This problem made me stuck for 2 hours of my development however once I manged to get past this, I was starting to get back up with the speed but due to the lack of spending time on trying to figure out what was wrong I had to set one of the tasks as ‘backlogged’ for Sprint 3. All of this would have been avoided if I tested my tokabase file in sprint 1, but all together sprint 2 was successful looking back on it as I manged to get most of the functional requirements, I set out to complete.

Sprint 3 was all about creating a cookies table and a function to login/authenticate users to determine which dashboard they are getting sent to. This also includes the task from previous Sprint 2 to be completed 2.4 for login the user in and displaying the correct dashboard for the account. This Sprint was the hardest by far as I was trying to figure out on how to verify users as well as giving them a unique cookie once they log in. I then had to create a way to display users name on the dashboard making sure its correct by the account they are on. Looking back on Sprint 3 I believe couple of these tasks could have been moved to sprint 4 or decomposed a bit more so I would have not struggled as much as I had to. However, I did manage to complete all the functional requirements set and have no backlogged tasks for sprint 4 which is great.

Sprint 4 was based around functionality of the dashboards such as getting push-ups, squats, blogs and records pages up and running with some of its functionality. As well as having to create another table for workouts to be stored in. There were a few challenges which I did face such as getting the table to show on records page and splitting the squats score away from push ups score. I also struggled with thinking of the eye-catching colour schemes which would go well together however I left it to my normal bright gold coloured table where its easier to see on blue background. This sprint I also had to delete one of my functionalities which was ShowPassword from the JavaScript because this would mess up my registering account password. As a result, to avoid this problem I decided to remove it, but this however can now be added in future development and not effect the register in any way or form. This Sprint had me thinking about what would of Toka Fitness liked to see on this page and in real scenario if I was able to be in meeting with them I would of asked questions on deciding features and colour schemes.

Sprint 5 was based around blogs page with personal training videos included in it. This will mostly be a html and CSS work and a bit of JavaScript to get the functionality such as read more buttons and getting the YouTube videos to work. Through development stage in Sprint 5 I decided to not include the Read More button but instead a box with click me sign which is more appealing in my opinion for the user. Sprint 5 was quick and easy sprint to start to finish on as I knew I was getting closer to finishing this development project and having easier tasks at the end is better than having hard ones.

Sprint 6 being one of the final sprints for this development of prototype only had a task of adding back button and making sure the code is maintainable for future development. Developing the back button was quick and easy for purpose of being able to go back to home screen from dashboard page. For the code maintainable task, it was all based of commenting the code and making sure all the variables are named as in the way they act/do. This was important to complete because if Toka Fitness do wish to continue and let somebody else to continue to develop this prototype and publish it the code should be maintainable by a third party and having ability to add features if necessary

Future development

There are many of things I want to add to this product in future such as getting the users calorie calculator they wanted, as well as having more than two workouts which users can pick from instead having something like a drop down and users having a wide range of choices to pick from and perform.

To make sure all this future development is possible this code had to be well commented and variables named correctly which I have done. This therefore means any third-party software developers Toka may hire would be able to understand what’s going on in the code just by reading the code and comments to see what each line of code does. I have also split the code in different functions therefore if they do wish to update and test a specific function, they are able to without affecting any of the other code.

Thoughts about the project

I believe this project was saucerful looking back on the sprints and reviews. This is because I see that I have developed skills which I haven't yet used by incorporating use of MySQL database and having a way to communicate from python to a server which would store user data. It would have been very beneficial of being able to have a meeting with Toka at the end of every sprint as well as at the start as they would be able to see if the ideas and functionality is what they want and expect to have. This would also make Toka Fitness staff feel more involved if meetings are set in place giving them ability to participate and help in the project. They would also be able to tell me anything they would have wanted to be changed or a feature to act in a certain way. However, I believe this development of a protype has been very successful testing all my skills as well as giving me ability to look up on internet and do some research on new ways of coding and completing this project.

Citations

Link	Number
https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0	(1)
https://www.w3schools.com/css/	(2)
https://docs.python.org/3/library/hashlib.html	(3)