T-LEVELS | Institute for Apprenticeships & Technical Education

T LEVEL

Technical Qualification in Digital Production, Design and Development

Grade Standard Exemplification Materials
Summer 2022

Occupational Specialism:

Digital Production, Design and Development

Distinction

Version 1.0

Pearson

Contents

NOTE: These Grade Standard Exemplification Materials are based on the 2022 summer assessment series, during which Ofqual asked awarding organisations' awarding committees to award more generously given the context of the pandemic and because these are new qualifications.

Introduction

The assessment for the Digital Occupational Specialism is based around a scenario-based project. The scenario was to develop a digital solution for a tutoring company.

Students completed the assessment in 87 hours of supervised sessions spread over a period covering 14 weeks. The assessment is split into four tasks covering a range of topics. These are summarised below:

Task	Topic	Evidence	Time
1	Analysing the problem and designing a solution	Proposal of the designed solution and a set of design documents	20 hours
2	Developing the solution	Prototype, development documents, test log and code for prototype solution	30 hours
3a	Gathering feedback to inform future development	Plan and report on gathering feedback	15 hours
3b	Evaluating feedback to inform future development	Feedback evaluation report	2 hours

In the assessment for the Digital Production, Design and Development Occupational Specialism scenario it is expected that the student will demonstrate many of the Distinction performance characteristics detailed in the grade descriptors. However, borderline performance will demonstrate these characteristics less consistently. Some key aspects of performance include:

- comprehensive analysis of a problem showing a detailed and perceptive consideration of the factors, solving issues, and fixing obvious or less complex defects
- production of different versions of an artefact that will be primarily functional, efficient, and respond effectively to users
- effective and fluent use of technical language
- effective evaluative processes.

The portfolio selected for this report was assessed at Distinction grade. This is the assessed grade of the portfolio as a whole and not the grade of each individual piece of work submitted as part of the portfolio. Comment will be made where the student work does not fully meet the Distinction grade descriptors and will suggest the evidence that is not included and should have been included.

Task 1 Activity A (ii)

Student Evidence Review:

The student presented two pieces of work in respect of the task, each task has sub-tasks:

- proposal
- wider issues
- business context
- visual/interface
- algorithm design
- data requirements
- test strategy.

Activity A

The proposal/justification

My solution for the client would be to create a website that provides the students with an easy to use place that helps them revise the work they have been doing with tutor and create an easy way for the tutor to track the progress of their students. This website will have a student or teacher option when signing up so that once the user is logged in the website can be more tailored to what the user would be using it for.

One of the pages for this website would offer a wide variety of courses for the students and "provide access to digital content to encourage wider learning". These will help them go back over the topics they have been learning to retain and reinforce the information in their mind. This page would display all the courses on offer with a navigation bar to help find the course they want. These courses would display the information that the student has been learning about and then ask them basic questions about that information. E.g. for English asking them a multiple-choice question about what a certain word means. I've come up with this solution after looking at a website called Seneca that can been seen in my appendix or in my sources table. This is also what I am basing the look of my website on.

Another page of this website that is only accessible by a tutor would allow them to "monitor the learner progress" their students are making on each course showing things like the speed they are getting through each course, how many questions they're getting right and an estimation of how well they seem to be comprehending the task. And after seeing this information the tutor can then leave comments for the students to see. Providing "interactive teaching" that can help the student in their weaker areas. The Tutor will also be able to set deadlines so that the student knows how long they have to complete it. I got the idea of tracking the student's data in each course from go student that can been seen in my appendix.

To incorporate the features customers and tutors would want into the solution like "gamified learning" I would also make a page that allows tutors or students to create their own quizzes that students can do. These quizzes once done would show the score each student is getting on a leader board providing a competitive aspect to their learning and to try and make the quizzes seem more fun than just another bit of work to be done powerups can be added in that can provide students with different advantages during the quiz e.g. one powerup could remove some of the wrong answers so there's a better chance they get it right. I thought up this solution when looking at websites like kahoot and quizizz that can be seen in my appendix and in my sources table.

Student	Tutors
Students will be able to: • Use any of the courses available on the website • See the results on each course once they've finished it • Make their own quizzes • Use other users' quizzes • View the leader boards • View other users' profiles • Add other users as friends • Customise their profile using the rewards they get • Show off various achievements through badges • Change the text size or use text to speech	Tutors will be able to: • Use any of the courses available • Request new courses to be made • See the progress their students are making on the courses • Get more detailed data on how well their students are doing with each course • Make their own quizzes • View leader boards • Send a reward to students • Comment on a student's course • Set deadlines for courses to be done • Change the text size or use text to speech

Mitigation of potential risks

One potential risk that could occur is that the website could have glitches that weren't found or there could be issues for certain users when trying to use the website. This will be mitigated by having a customer support section where any potential problems a user has can be reported and then dealt with.

Another risk is that Users may forget their account information and so a forgot username/password option will be available for them if needed.

Functional requirements

No.	Features	Priority	Justification
1.	Users are able to sign up	HIGH	The user won't be able to use any features on the website if this doesn't work
2.	Users are able to log in	HIGH	The user won't be able to get back into the website if this doesn't work
3.	Send an email to the customer when they make an account	LOW	This is just a check that the system has the right credentials
4.	System will record user data on courses	HIGH	This will be needed in order for tutors to check how students are doing on courses and will also allow for tracking trends in what courses people are doing

Non-functional requirements

No.	Features	Priority	Justification
1.	Website has a good response time	HIGH	If the website doesn't feel responsive to users they will most likely go somewhere else
2.	Website is adaptable to any viewing device e.g. laptop, smart phone, tablet.	MEDIUM	Whilst it's important to have the website accessible to people no matter the device, a majority of users will be using a computer or laptop and so those take priority.
3.	Colour scheme is appropriate for any user	HIGH	User accessibility is important and if some users can't see the website properly they won't use the service
4.	User login credentials and any other personal data is secure	HIGH	The user needs to be able to trust the service with the credentials or they just won't use it.
5.	The system should be available at all times	HIGH	The website can lose a lot of traffic if it's not available to everyone when they want it.

Lead Examiner Commentary:

It is organised logically and an excellent attempt to identify the problems to be solved. The proposed solution considers the clients and users with complete justification and the risks associated with the development phase. The legal aspects are related to the scenario. The student, in considering broader issues related to the scenario, has included the appropriate functional and non-functional requirements, which address the client's needs and are associated with the scenario. They have also added justification for each requirement. In terms of user acceptance criteria, the student, has considered sensible criteria and rationale of how these apply to the importance of how the solution meets the project objectives.

Other considerations are legal aspects and the student has related the most appropriate legislation to the client and what impact this may have if breached. With this in mind, the student can consider what security features they may use in the development of the artefact. Each section should not be done in isolation and should set the thinking for the next section.

Task 1 Activity B Visual/Interface Design

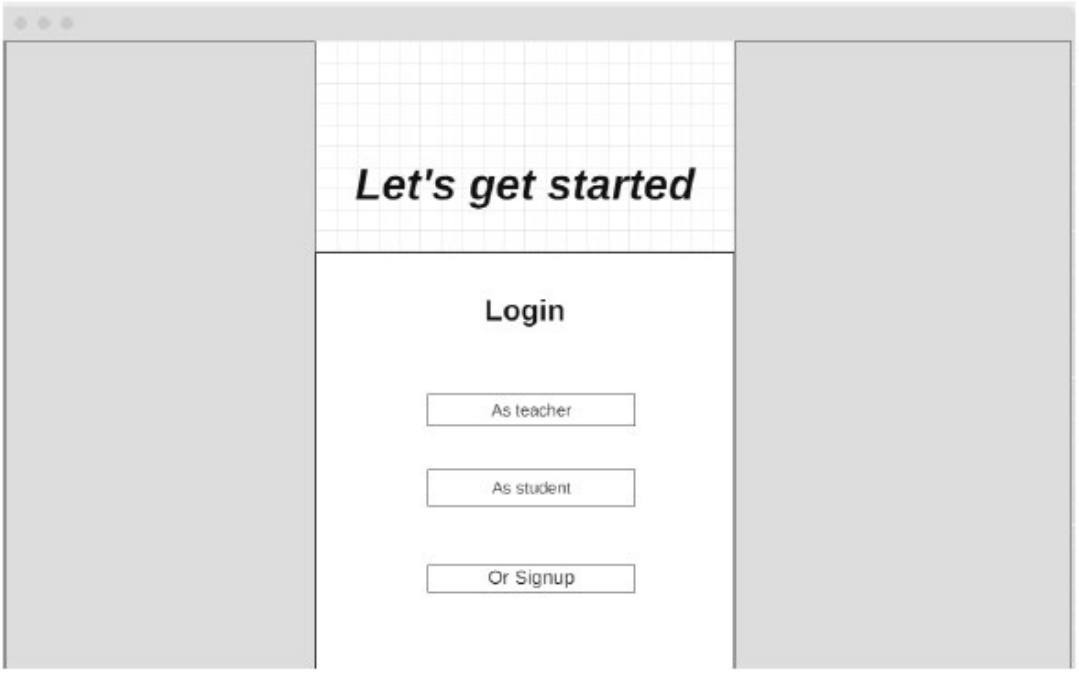
Student Evidence Review:
In this area, students need to demonstrate their understanding by providing reasoned justifications for the approaches used to create the stages of development:

- considerations such as their approach to a solution
- justifications and depth are essential
- security controls
- naming convention
- comprehensive design.

Activity B

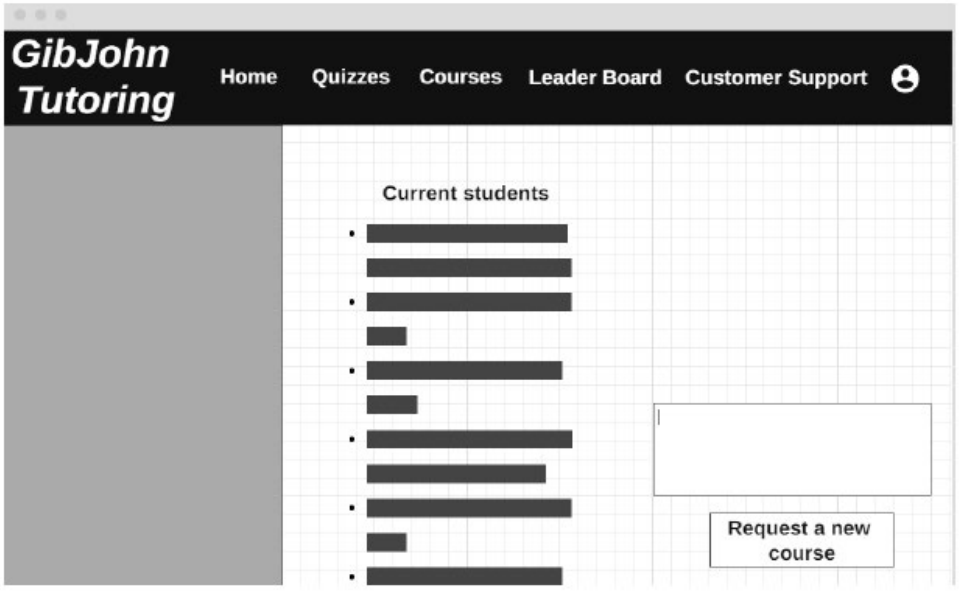
Visual designs

Login screen



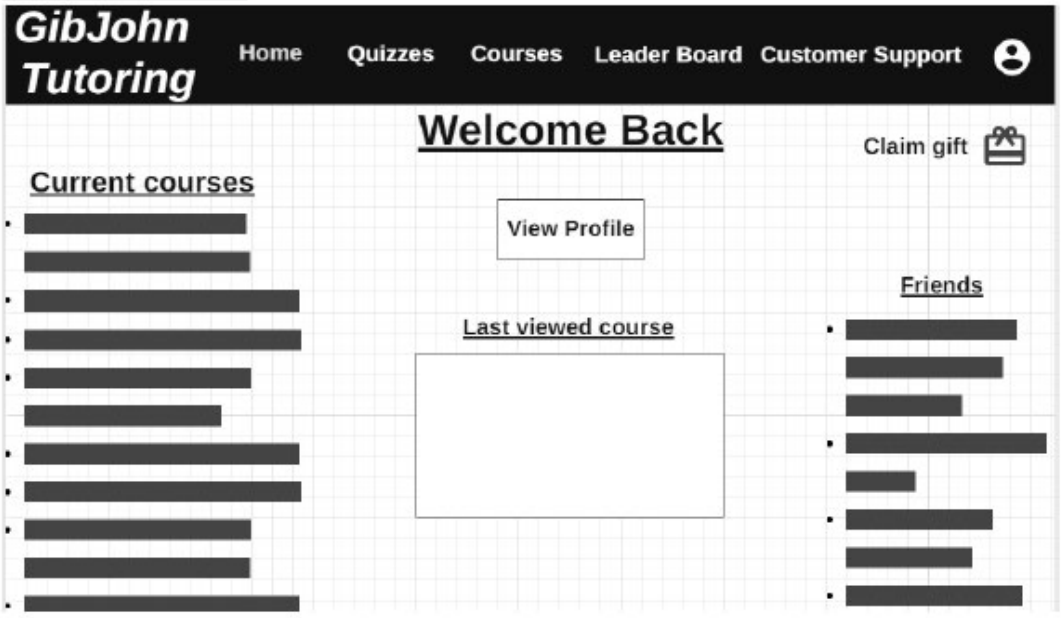
This is the layout plan for the login/signup page, I've made sure to keep it simple (as to avoid confusing any new user clicking onto the website) with clear buttons directing the user where they need to go. The grey boxes represent the areas on the page that I would want to put an image or have some form of a background due to this being the standard practice for many log-on screens and so would confuse the user less.

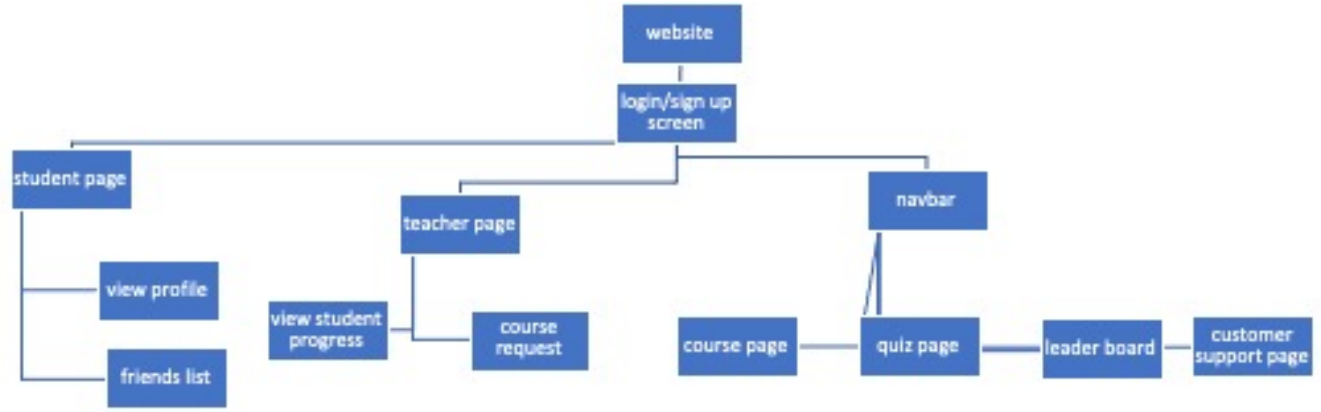
Teacher Homepage



This is the homepage screen for when a tutor logs in. I've put the navbar at the top with the company name in the top left corner and profile picture in the top right corner as it's standard practice for a majority of websites and so will help the website feel intuitive. This navbar will also be the same for every single page of the website as it allows for easy navigation. You then have in the middle of the screen a list of the students that the tutor has so that they can easily reach whichever student they want and check up on the students' progress. The option to request another course is also on this page to make it easy to find. The grey box is to show that something will be placed there in the final product, as to avoid too much white space, this will either be another function that gets thought up in later planning/testing or some sort of image.

Student homepage





Lead Examiner Commentary:
The student has provided evidence of detailed, compelling designs and shows enough for a third party to be able to create the whole artefact. The student has also considered the complex structure of the website and page specifications such as background colour, text sizes and type. You will notice that the student has provided layouts, visual hierarchy, and standard conventions.

Task 1 B Algorithm Design

In this area, students need to demonstrate their understanding by providing reasoned justifications for the approaches used to create the stages of development:

- considerations such as their approach to a solution
- justifications and depth are essential
- flowcharts or pseudocode
- precise and logical
- security controls
- naming convention
- comprehensive design.

Decomposition of problems to be solved for functional and non-functional requirements

- In order to record all the user data, we will require a large database to store it on
- Some students may not own a digital device and so therefore a set number of laptops can be provided to those students for work.
- In order to make it adaptable to multiple devices further research on how to adaptable html is required
- To make the colour scheme appropriate for any user research on what colours can be seen by everyone is required
- Use of CSRF tokens is needed to make sure user data is secure.

KPI

- How many users that click on the website make accounts.

Keeping track of this will allow you to see how many users are being lost from initial impressions of the website and if it seems like a large number of users then changes could be made to retain user attraction.

- How long on average do users spend on the website.

You need to keep track of this so you can see how well the website is doing.

- How many users use the customisation options.

It's important to track this as you'll then be able to see how effective of a reward it is and whether changes are needed.

- How many users use the quiz section of the website.

It's important to keep track of this to see how many people are liking it and whether or not changes will need to be made to improve users enjoyability of it.

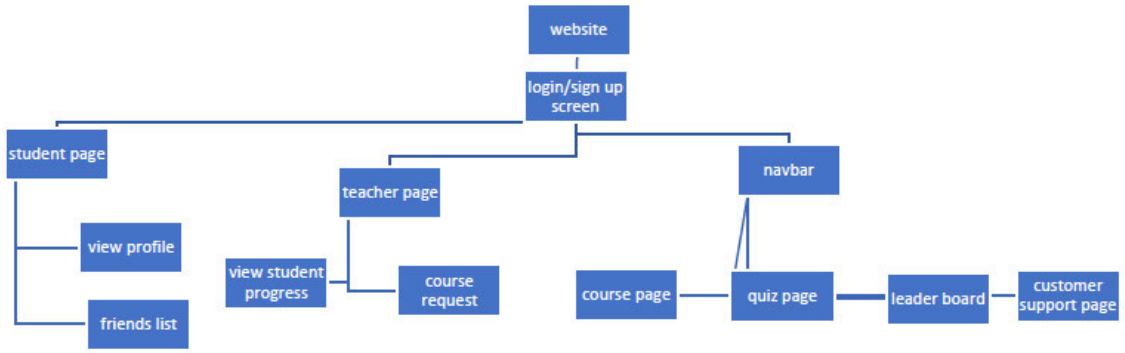
- What courses are the most popular.

It's important to keep track of this as you can then see what's in high demand and can advertise those courses more to pull in more users.

Pseudocode

```
FUNCTION SignIn()
    OPEN Account_Database
    RECEIVE Name FROM (STRING) KEYBOARD
    RECEIVE Password FROM (STRING) KEYBOARD
    RECEIVE ConfirmPassword FROM (STRING) KEYBOARD
    RECEIVE Email FROM (STRING) KEYBOARD
    IF Password == ConfirmPassword THEN
        FOR each line in Account_Database
            READ Account_Database
            IF Name AND Password AND Email == Account_Database (Name, Password, Email) THEI
                SEND "Welcome" TO DISPLAY
            ELSE
                SEND "Incorrect details please try again." TO DISPLAY
    ELSE
        SEND "Your passwords do not match please try again" TO DISPLAY
END FUNCTION

FUNCTION CreateAccount()
    OPEN Account_Database
    SET Valid TO FALSE
    WHILE valid NOT TRUE
        RECEIVE Name FROM (STRING) KEYBOARD
        RECEIVE Password FROM (STRING) KEYBOARD
        RECEIVE ConfirmPassword FROM (STRING) KEYBOARD
        RECEIVE Email FROM (STRING) KEYBOARD
        IF Password == ConfirmPassword AND contains at least 1 Uppercase character, 1 lowercas
character, 1 integer and symbol AND Password(length) >= 8 THEN
            SET Valid TO TRUE
            WRITE Name, Password, Email TO Account_Database
            CALL FUNCTION SignIn()
        ELSE
```

Hierarchy diagram



The website consists of 6 main pages these being: the student homepage, teacher homepage, quiz page, leader board, course page and customer support page. These 6 pages will be my main focus when making the website as they make up the most important features needed in order to meet my client's requirements. There are then the 4 other features involved in the student and teacher pages, these being: view profile, friends list, view student progress and course request. I will attempt to implement these provided I have enough time after making the main 6 pages with the focus being on the student progress and view profile features as these help make the website feel more involved and interactive for the user.

Lead Examiner Commentary:

The student has demonstrated their ability to break down the problem into smaller sub-systems, shown as a visual diagram; using stepwise refinement or sensible other notations is also acceptable. First, the student must show the overall breakdown of the problem. From here, they should select one sub-system and break it down in the next stage, which would be the algorithm, using a flowchart or pseudocode as demonstrated below:

Task 1 Part B The Data Requirements

These are the entities I will be including on my database. Both are simple standalone tables.

UserInfo	
Field	**Data type**
UserNameID	Int (10)
First Name	Varchar (20)
Surname	Varchar (20)
Email	Varchar (50)
Telephone	Varchar (13)

User Account	
Field	**Data type**
UserNameID	Int (10)
Password	Varchar (between 3 and 16 characters)

Data requirements

Variable name	Function	Data type	reason
$password	Assigns the users input to a variable for the password	string	To allow the system to validate the password
$passwordCon	Assigns the users input to a variable for confirm password	string	to allow the system to validate the password
$passwordError	To output an error message if the password is wrong	string	If there is an error with the password it outputs a message so the user knows
$passwordConError	To output an error message when the confirm password is checked against the password and its wrong	string	If there is an error with confirming the password it outputs a message so the user knows
$fNameCheck	Makes sure the first name field is filled in	string	To make sure there is a present value

Task 1 Activity B The Design Test Strategy

Student Evidence Review:
The student provided a test strategy including:

- white box testing
- black box testing
- others
- justification.

Testing strategy

I'm going to use black box testing where I test the inputs and outputs of the website as well as white box testing where I'll test the backend/internal structure of the website.

Test strategy	Black box testing
Purpose	the test strategy allows for a visual representation of all the testing that can be devised for each and every input and output on the program making sure everything functions as intended
who performs the test	Me
Test data set	Every action/input and output will be recorded
Test criteria	That the program outputs what's intended
When to test	Whenever a section of the website is done and the inputs and outputs need to be tested
Estimated time required	Each section of the website will take up to a maximum of 30 minutes depending on how many inputs and outputs it has
Test outcome	Outcomes will be compared with what the intended outcome was meant to be and any improvements will be made to those outcomes that don't do as intended.

Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies
	Login page: When the password doesn't meet the requirements an error message will show	Black box testing	Need access to the website Computer Input an invalid password
	Login page: When the password and confirm password don't match an error message will show	Black box testing	Need access to the website Computer Input a different password in confirm password to what's written in password
	Home page: when the various buttons are clicked the system takes the user where it's supposed to	Black box testing	Need access to the website Computer User needs to login to reach the homepage
	Course page: when a course is clicked on the correct course will then be loaded	Black box testing	Need access to the website Computer User needs to login and then click onto the courses page
	Leader board page: the leader boards update automatically every 10 minutes	Black box testing	Need access to the website Computer User needs to login and then click onto leader board page
	Quiz page: when the user clicks on a quiz that quiz will load	Black box testing	Need access to the website Computer User needs to login then go on the quiz page
	Security of the page from outside attacks	White box testing	Computer Access to the website

Lead Examiner Commentary:

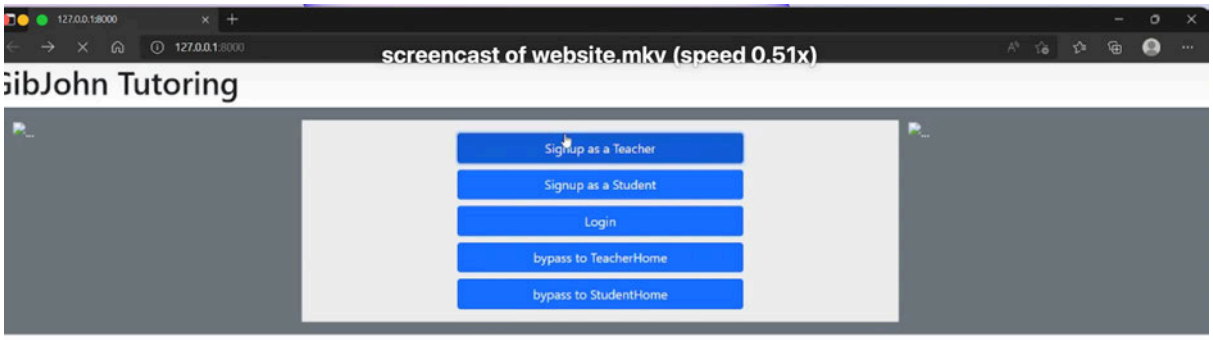
The student has explained the test strategy for the testing phase, in which they must consider a comprehensive range of tests such as Whitebox and Blackbox testing. This is not a test plan where it is clear to see that the student has demonstrated which test strategy is applied at which stage of development or which test is used at what stage of the development of the artefact. The section is not a test plan that will be followed in a later section when the test strategy is carried out.

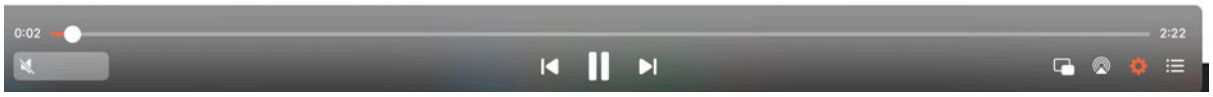
Task 2 The Solution Functionality

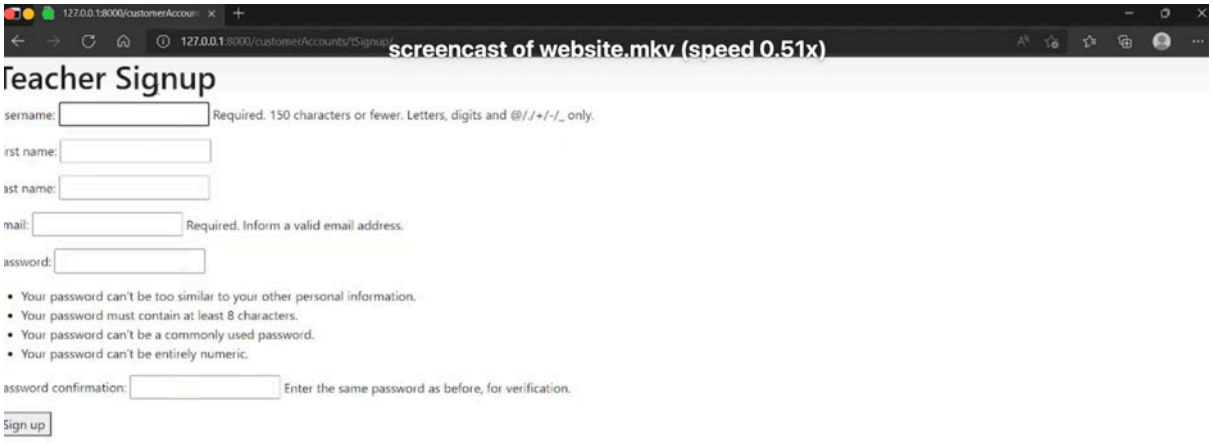
Student Evidence Review:

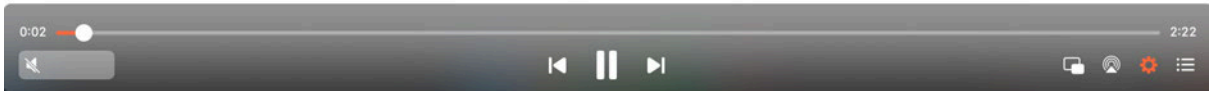
The student needs to demonstrate a working prototype:

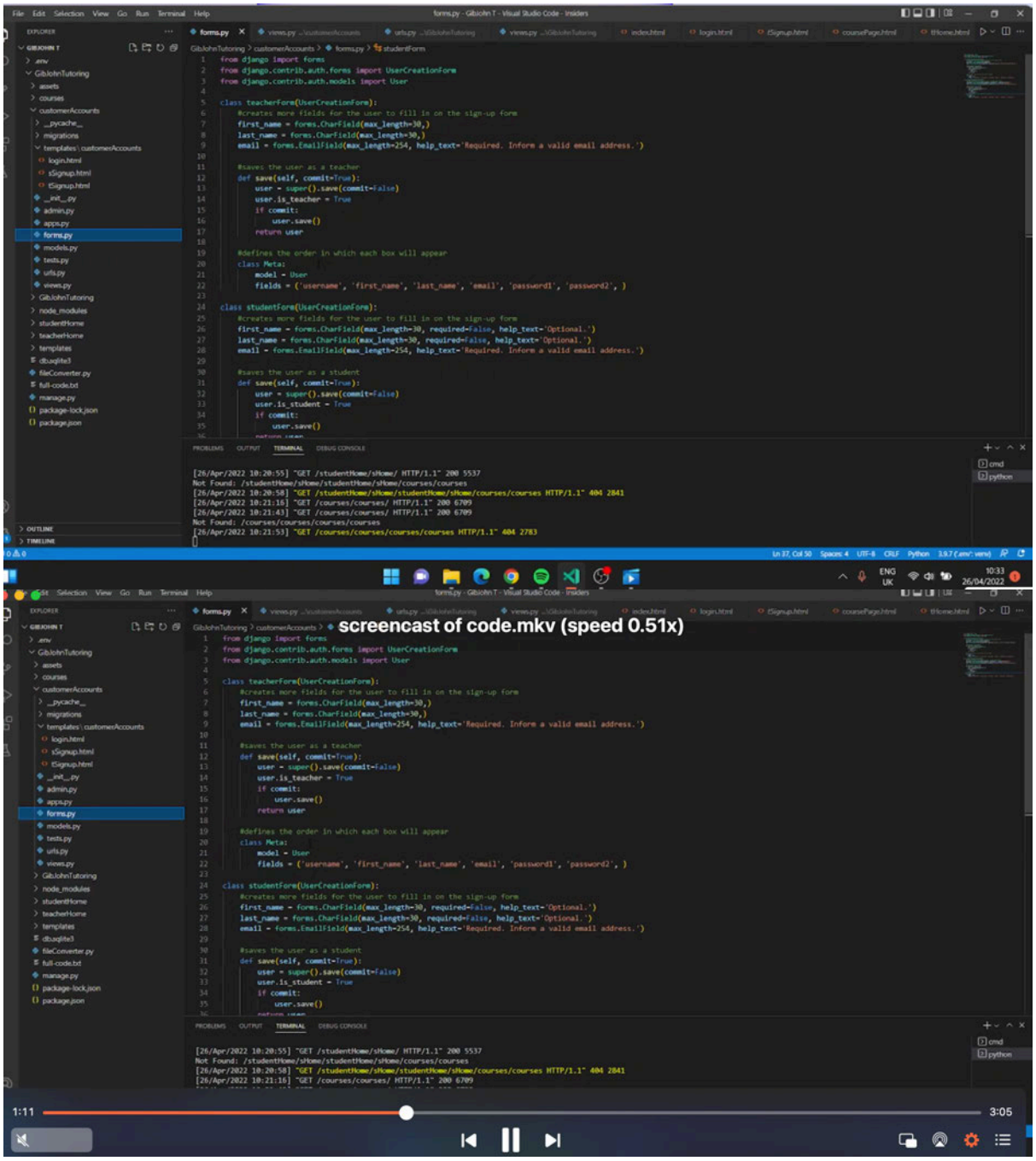
- two different paradigms
- efficient constructs
- robust structure
- user friendly
- security.


screencast of website.mkv (speed 0.51x)
ibJohn Tutoring
Signup as a Teacher
Signup as a Student
Login
bypass to TeacherHome
bypass to StudentHome


0:02
2:22


screencast of website.mkv (speed 0.51x)
eacher Signup
sername:
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
irst name:
ast name:
mail:
Required. Inform a valid email address.
assword:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.
assword confirmation:
Enter the same password as before, for verification.
ign up


0:02
2:22



Lead Examiner Commentary:

The student has demonstrated efficient functional code using two languages. We tested the solution as the student provided executable files with 'read me file'. This helped to focus user experience, and we were able to test the input handling, error messages and outputs.

In addition, the student considered standards and guidelines concerning accessibility and compatibility.

The student has demonstrated understanding by providing reasoned justifications for the approaches used. This included considerations as to their approach to a solution. They have used two different programming languages. Explanations and

depth are shown throughout, and the student has written the code on multiple parts of the project; this is where the student has justified the decisions by adding comments on the code. The student also provided a narrated video of the completed solution . The video should not only demonstrate the artefact but should also be explained by the student as it is filmed.

Task 2 The Solution-Code Organisation

Student Evidence Review:

The student needs to to demonstrate a working prototype:

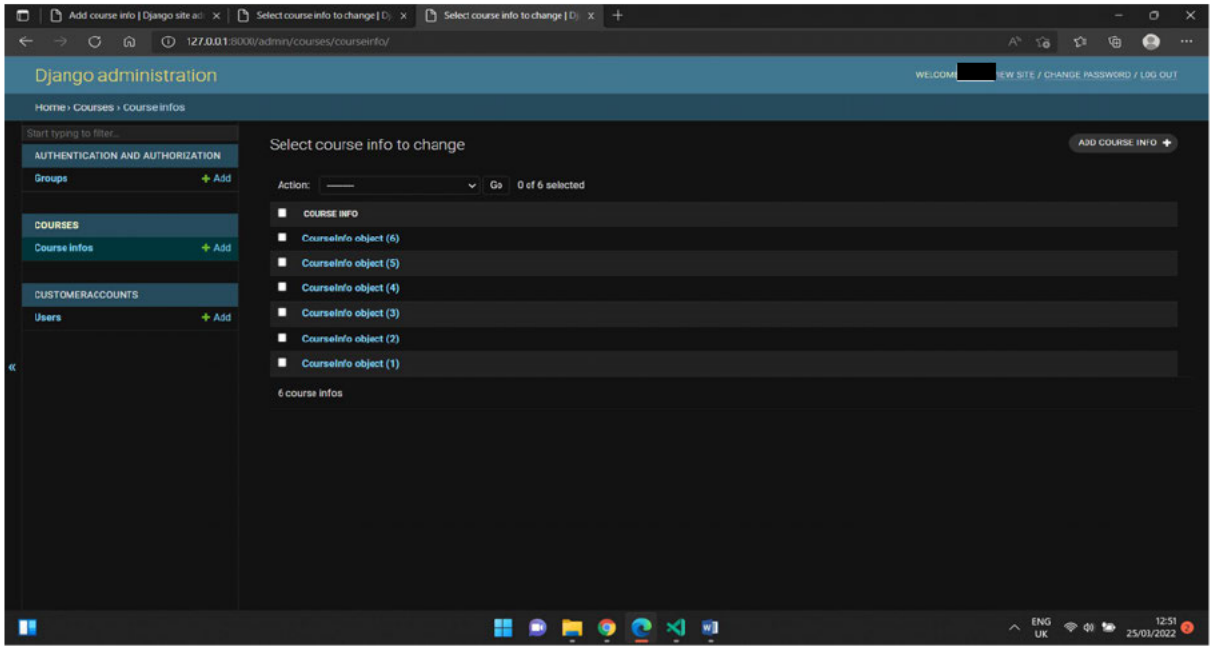
- avoid multiple pages of nested if clauses and unnecessary repeated code that could be implemented more effectively as a called function
- clear and meaningful indentation
- code should consist of pieces of logic, classes, or objects, with proper structure
- comments should be used wherever possible to help explain the logic
- good use of local variables and minimal use of global variables
- use of constants
- consistent style throughout.

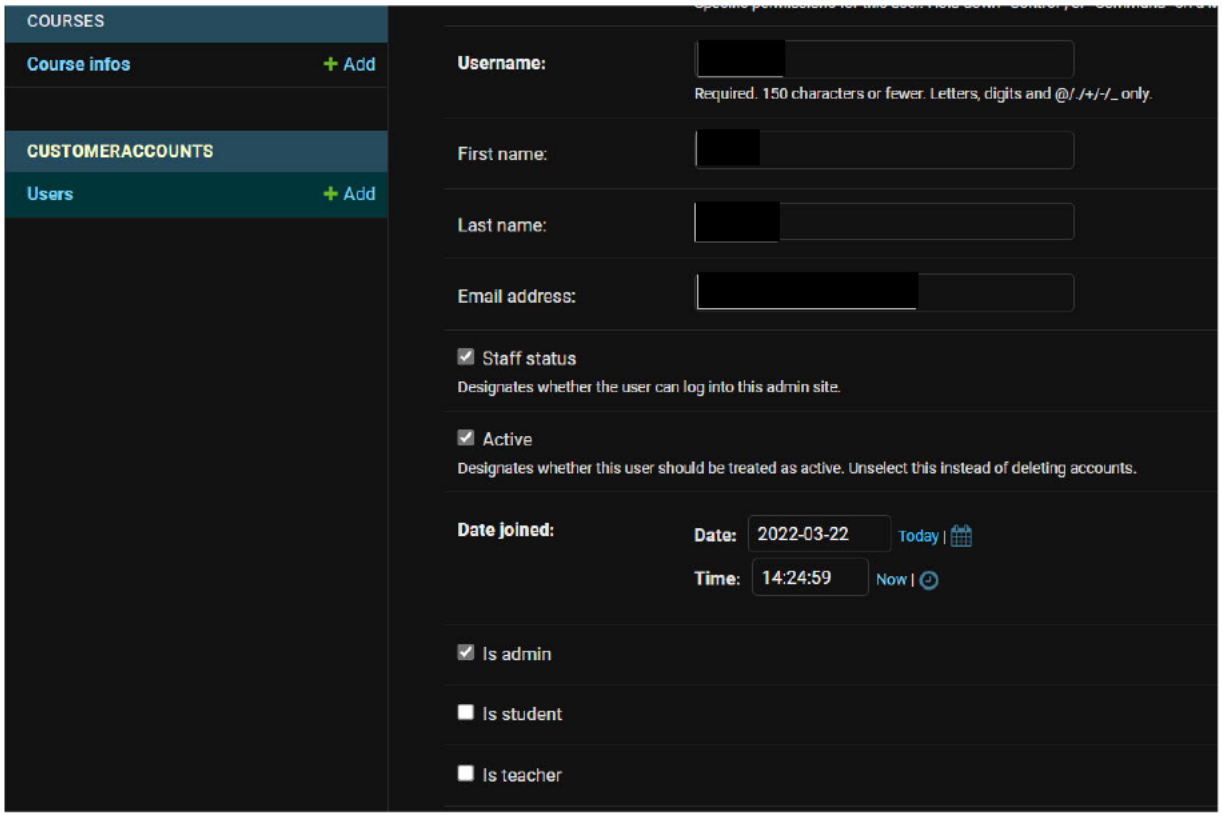
```
import shutil as s
import os

directory = os.getcwd()
folders = os.listdir(directory)
print(folders)
print("######################")
filesNeeded = ["admin.py", "forms.py", "models.py", "urls.py", "views.py", 'settings.py']
finalFile = ""

count = 0
for item in folders:
    #finalFile = f"App - {item} \n"
    try:
        nextDir = os.listdir(item)
    except:
        continue
    for nextFile in nextDir:
        if nextFile == "templates":
            directory = os.getcwd()
            folder = os.listdir(f"{directory}\{item}\{nextFile}\{item}")
            for htmlFile in folder:
                title = f"{item}/{nextFile}/{item}/{htmlFile}"
                f = open(f"{item}/{nextFile}/{item}/{htmlFile}", "r")
                finalFile += f"{title} \n\n"
                finalFile += f.read()
                finalFile += "\n\n ---------------------------------- \n\n"
        elif nextFile in filesNeeded:
            title = f"{item} - {nextFile}"
            f = open(f"{item}/{nextFile}", "r")
            finalFile += f"{item}/{nextFile} \n\n"
            finalFile += f.read()
            finalFile += "\n\n ---------------------------------- \n\n"

#print(finalFile)
f = open("full-code.txt", "w+")
f.write(finalFile)
f.close()


##s.copyfile("test.py", "test.txt")
```

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'GibJohnTutoring.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Lead Examiner Commentary:
The student demonstrated well-organised code that includes naming conventions and comments that allow third parties to pick up on what the code is doing. It is logical and precise. The student has used module and Django.

Task 2 The Solution-User Experience





Lead Examiner Commentary:

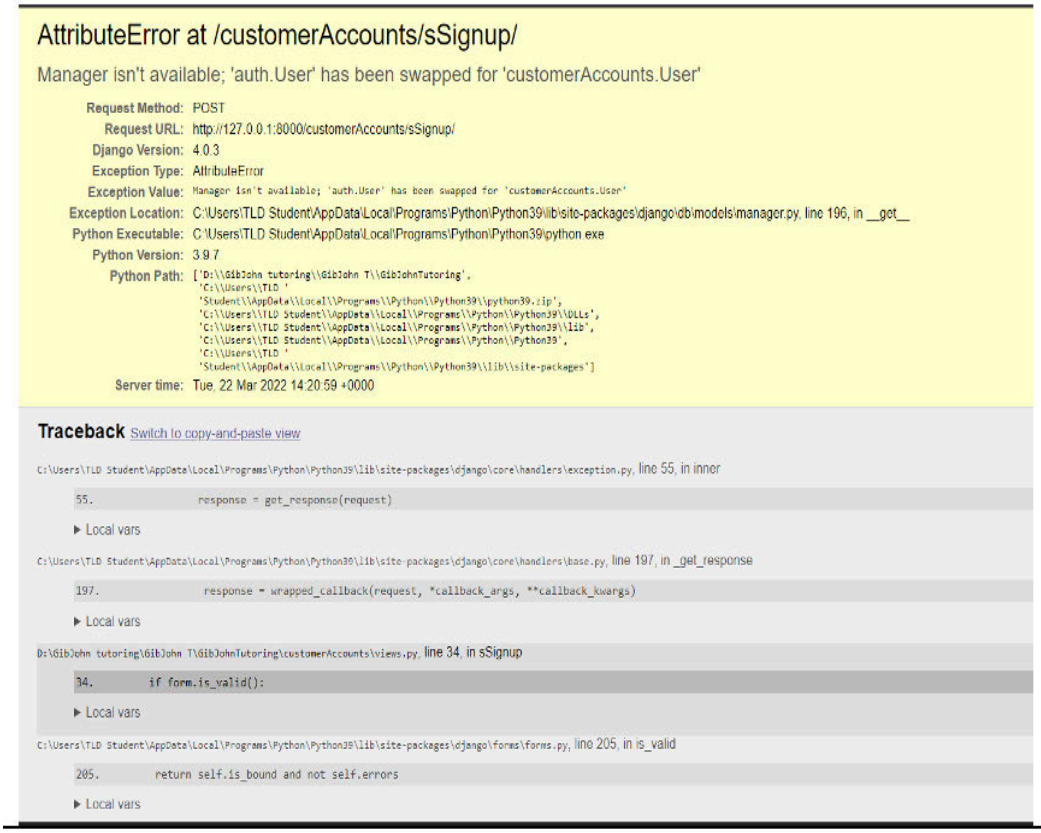
Regarding the legal and regulatory guidelines and standards, the student has demonstrated how the artefact would be viewed in different browsers. They have validated the code on the W3C website. This will let the student know what needs

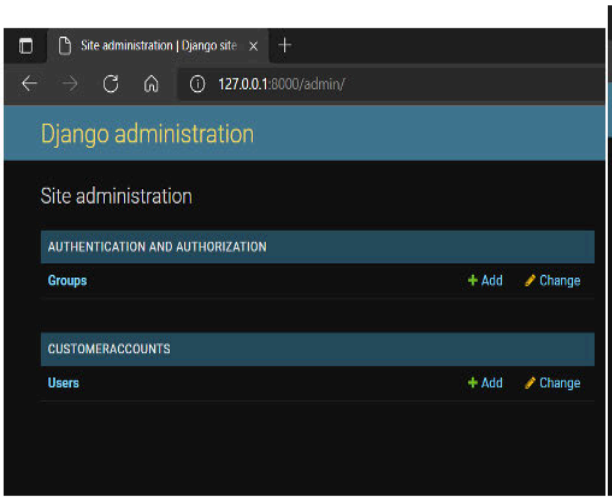
to be added for the website to be viewed as expected, no matter which browser. Security controls are good, considering various option such as using username and password, cookie notification, and even two-way authentication. This provides evidence for legal and ethical thinking by showing that the data is kept safe and secure to protect both the user and the client when using sensitive data. The feedback to the users demonstrated that the user would understand when form and login are completed as feedback messages are provided

Task 2 Testing

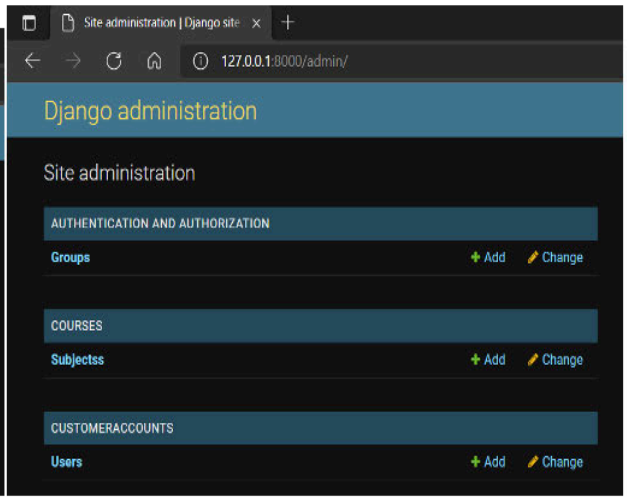
Student Evidence Review:
Suitable test data should be demonstrated (as appropriate to the student's solution), which may include:

- **test-specific data**: influences the system behaviour and reveal the case specifics under the test
- **test-reference data**: have little influence on the test performance
- **application reference data**: irrelevant to the behaviour under test, but needed to start the application
- **valid test data:** does the system function in compliance with the requirements? does the system process and store the data as intended?
- **invalid test data**: checks to see if the software correctly processes invalid values, shows the relevant messages, and notifies the user that the data are improper
- **boundary test data**: helps to reveal the defects connected with processing boundary values
- **wrong data**: entering the data in an inappropriate format, whether it shows the correct error messages thus showing the use of validation if appropriate
- **absent data**: should check that the solution handles entering a blank field.





Before change



after change

to the sign-up page				implementing it to the teacher signup

3. When any page loads the CSS will be implemented onto the webpage		The webpage will have the CSS implemented onto it	the CSS doesn't appear	I resolved this issue by going into the settings and
				`STATICFILES_DIRS = (`
				`os.path.join(BASE_DIR, "assets"),`
				including
				This allowed the system to find my assets folder that held the CSS so it could be loaded onto the page

4. When the reveal password eye is clicked the password is revealed		The password is revealed	The password gets revealed	

Forms.py

This the form for the teacher signup (the student signup is the same)

GibJohnTutoring > customerAccounts > forms.py > teacherForm > save

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class teacherForm(UserCreationForm):
    first_name = forms.CharField(max_length=30,)
    last_name = forms.CharField(max_length=30,)
    email = forms.EmailField(max_length=254, help_text='Required. Inform a valid email address.')

    def save(self, commit=True):
        user = super().save(commit=False)
        user.is_teacher = True
        if commit:
            user.save()
        return user

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', )
```

This saves the user as a teacher within the database

This defines what order the inputs on the form will appear so that it follows with common practice of registration forms improving user experience

I used the built in UserCreationForm from Django and then added in inputs for email, first name and last name. the first and last name were added as it is typical to include those on forms and the email is there so that a confirmation email can be sent to the user.

The title is in a different font than it usually is because of css

The buttons for the bootstrap are showing

Lead Examiner Commentary

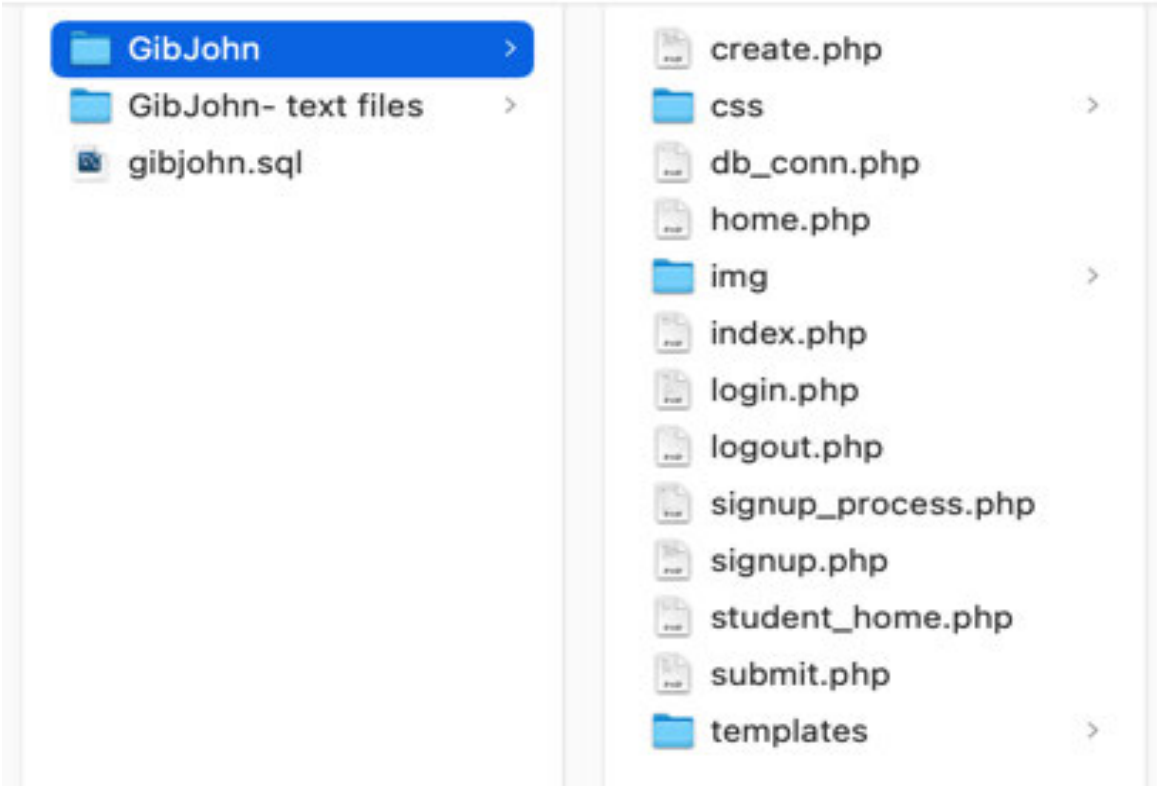
The student has demonstrated their knowledge of testing and debugging code. They have provided a test plan. The student has used various techniques for debugging, testing the code, and trying to implement a functioning solution. The student clearly shows full awareness of the function of the code (testing inputs and returns) as well as the role that validation and verification have played in the process of the code, with a clear record of actions taken to fix any issues encountered.

The testing evidence was often mirrored in the attempts to resolve the issues in order to get the code running and ensure it met user requirements. They have considered the needs of code as outlined in the task description to help inform a testing strategy beyond syntax errors, explored various validation and verification methods of inputs, and regression testing of the implemented solutions.

Task 2 Documentation

Student Evidence Review:
The student should provide evidence of:

- different versions of the prototype
- clearly documented changes
- how the version demonstrates appropriate changes
- user feedback.

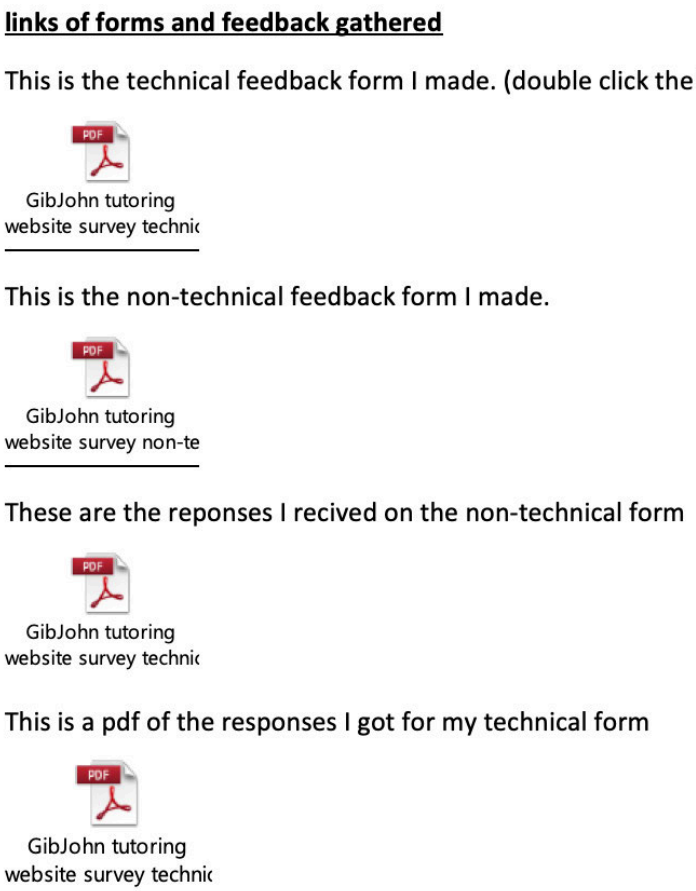


Lead Examiner Commentary:
The student has provided evidence of using the agile methodology, it is clear to see the various version that have been checked and the link between the testing phase, feedback and improvements.

Task 3 Part A Gathering Feedback to Inform Future Development

Student Evidence Review:
The student needs to demonstrate two different techniques for gathering feedback:

- e.g. survey and observations
- questions are directed at technical and non-technical testers
- visualisation of the data gathered and analysed.



Lead Examiner Commentary:
The student shows two different techniques for gathering feedback, survey and observations, which help the development of the prototype. The questions should be directed at Technical and non-technical testers. The questions should be appropriate so that data drawn from the questions can be used to benefit the further development of the artefact and hence the user. The student should provide a detailed visualisation of the data gathered and analysed. This gives us a clear idea of what the information means by giving it visual context through maps or graphs.

Task 3 Part B Evaluating Feedback to Inform Future Development

Student Evidence Review:
This task requires the student to evaluate feedback received and discuss future developments:

- evaluation of the assets and content within their prototype
- final product against measurable criteria defined in the proposal
- validity and reliability of the sources
- legal and ethical implications of using the identified assets.

Evaluation of feedback

From the feedback I gathered about my website I have been able to figure out what I could do next in order to further develop my solution. The first improvement I have recognized for the colour scheme as all testers gave the same sort of feedback saying that it was too bland and made the page feel empty. Therefore, in the future I will try using a different colour than white for the background and add a dark mode for anyone that would prefer a black background. The next improvement that I can make to the website is that there needs to be a lot more functionality as 3/4 of testers when giving their final thoughts on the website said that there needs to be more, this is also something that I recognized myself and so in the future I will get the navbar fully functioning so the user can easily navigate the website as well as have the various buttons on each of the pages fulfill their functions. Another improvement to be made is for the teacher homepage as 100% of users said no when asked if they like the page. These improvements would involve adding content like a course request button as well as a feature to track each students progress. I also realized through my user observation that there is no way for the user to reach the courses page without having to type in the exact URL and so I will develop this further by having the link to the course page in the navbar functioning next time so that users will be able to reach the page naturally. Lastly, I found that none of my testers were able to signup or login to the website and so in the future I will figure out what the error is that keeps appearing so that users will be able to successfully make an account.

From the feedback I was also able to figure out one thing that I won't have to change or improve upon too much and that is about the layout of each of the pages as many of the users when writing what they liked about the website said the layout was nice and that they liked how spaced out everything was.

The effectiveness of the assets and content used

I selected the search symbol and profile picture as I felt that they would help with the user experience as users are now so used to seeing a profile picture symbol in the top right of the screen and a search symbol next to a search bar. I chose to take these from bootstrap because I knew that it was a reliable source for improving the look of my website as I had already used it to help with the layout of the page and so felt it was the right choice to pick them over images from google or from another website. It also meant that I wouldn't have to worry about copyright as bootstrap is open source and so any legal issues could be avoided.

As for how well my solution meets with KPIs I can't check for quite a few of them as the feature that is needed hasn't been added to my website yet. However, for how long users typically spend on the website I can't know for sure as most of the testing I got users to do involved them going through the whole website so there's no way for me to know how long user retention is kept for on my website however I have figured out that it takes about 5-10 minutes for users to see through all of my website.

Finally, I can see that for the user acceptance criteria my solution was not able to meet a majority of what I had set out to be necessary. Most of the features that were required haven't even been started such as gamified learning or a way to monitor student progress whereas others have some of the front end sorted but because there's nothing for the backend none of it is functional such as a learning reward system and providing access to digital resources. However, there are a couple that I was able to meet which is that the website is adaptable as I used bootstrap so the page should naturally change to fit the screen size and because of the feedback I gathered I know that users find the website easy to navigate through.

Conclusion

In conclusion I find that I'm happy with the general look of the website and how I've spaced the content out across the page as well as how the navbar looks. It also seems users agree with that so I won't try to change that too much when making improvements however, I will put some thought into changing up the colour scheme to something that may be more agreeable. As for major improvements I would make in the future, firstly I would put a large amount of focus on making sure the user accounts are working and also making the courses page fully functioning in order to fulfil more of the user acceptance criteria as this would then make digital learning recourses available to users before then moving on to adding content to the teacher homepage so that users don't find a page with nothing on it.

Lead Examiner Commentary:

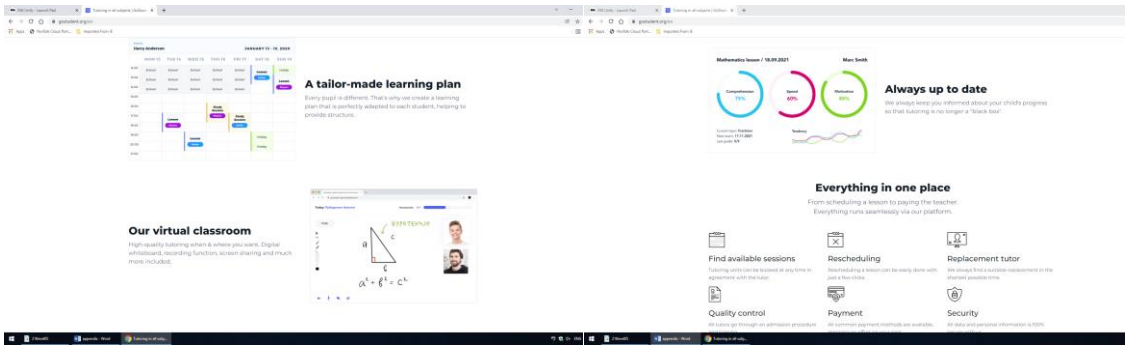
Students should provide the assets used and how valid their sources are (ensuring that no copyright laws have been broken) and the reason as to why they chose them. This should be related to the scenario and why the object was used on the website page, how it would help the user and the reason for using it. The placing of images or events must also clearly show a reason.

Evaluations were appropriate and showed an understanding of the requirements of the set task brief. While students typically at this level provide evaluative comments concerning the requirements of the task, at the borderline, user needs were effectively considered. The student has also demonstrated that future developments can be provided with later versions through using user feedback. Therefore, we would expect to see future developments that are relevant and realistic.

Appendix

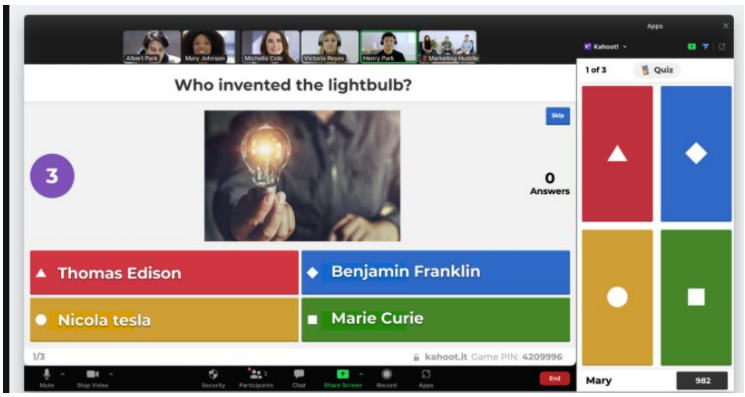
Some Software that I found and thought could be useful for the solution were things like:

- **Go student**



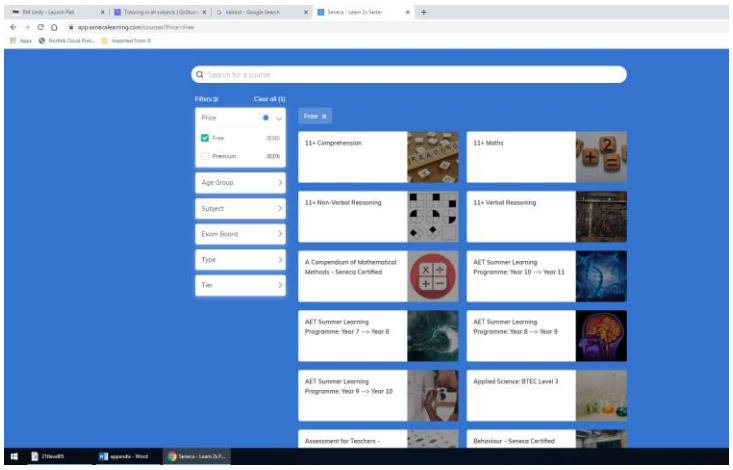
This tutoring software is a good basis for what I would go for with my solution with "support assessment and monitoring of learner progress"

- Kahoot



Kahoot will allow for potential group sessions in which they can all take part in doing the quiz and it's a solution to the want for gamified learning as I can make a quiz website like kahoot and even give it a reward system to help motivate the learners.

- Seneca



an educational platform that has a wide variety of courses/subjects and can be used to help the student outside of the sessions to retain the information they've learnt (this will help with interactive teaching and learning resources in range of subjects)

after seeing both Seneca and kahoot I've realised I can design a website that can offer a wide range of subjects but also gamified learning through quizzes with a reward system

Some hardware that I found and thought could be useful for the solution were:

- Desktops and laptops
- Tablets/handheld devices

Some emerging technologies that can potentially help the needs of users are:

- Augmented reality
- Adaptive learning
- AI (this can be used over the week by the customer for questions and to improve skills while the tutor isn't available to talk to.)

Sources table

Source name	link to source
Seneca	https://senecalearning.com/en-GB/
kahoot	https://kahoot.com/
go student	https://www.gostudent.org/en
Quizizz	https://quizizz.com/
Laws and regulations for schools	https://www.ihasco.co.uk/blog/entry/2718/regulations-that-apply-to-schools

Activity A

The proposal/justification

My solution for the client would be to create a website that provides the students with an easy to use place that helps them revise the work they have been doing with tutor and create an easy way for the tutor to track the progress of their students. This website will have a student or teacher option when signing up so that once the user is logged in the website can be more tailored to what the user would be using it for.

One of the pages for this website would offer a wide variety of courses for the students and "provide access to digital content to encourage wider learning". These will help them go back over the topics they have been learning to retain and reinforce the information in their mind. This page would display all the courses on offer with a navigation bar to help find the course they want. These courses would display the information that the student has been learning about and then ask them basic questions about that information. E.g. for English asking them a multiple-choice question about what a certain word means. I've come up with this solution after looking at a website called Seneca that can been seen in my appendix or in my sources table. This is also what I am basing the look of my website on.

Another page of this website that is only accessible by a tutor would allow them to "monitor the learner progress" their students are making on each course showing things like the speed they are getting through each course, how many questions they're getting right and an estimation of how well they seem to be comprehending the task. And after seeing this information the tutor can then leave comments for the students to see. Providing "interactive teaching" that can help the student in their weaker areas. The Tutor will also be able to set deadlines so that the student knows how long they have to complete it. I got the idea of tracking the student's data in each course from go student that can been seen in my appendix.

To incorporate the features customers and tutors would want into the solution like "gamified learning" I would also make a page that allows tutors or students to create their own quizzes that students can do. These quizzes once done would show the score each student is getting on a leader board providing a competitive aspect to their learning and to try and make the quizzes seem more fun than just another bit of work to be done powerups can be added in that can provide students with different advantages during the quiz e.g. one powerup could remove some of the wrong answers so there's a better chance they get it right. I thought up this solution when looking at websites like kahoot and quizizz that can be seen in my appendix and in my sources table.

To tie in both the learning reward system and gamified learning I want to provide students with their own profiles that will display their various achievements to others, such as, the courses they have completed, the courses they've got 100% on, the number of times they've reached number 1 on leader boards or the number of days they've been able to retain the top spot. There will also be the option for students to add each other as friends so that they can view each other's profiles easier. This can add to the competitive aspect as students will want to show off their achievements to others (gamified learning) and it will also help incentivise the student to do the courses as it'll make the course feel more worthwhile to complete (learning reward system). These achievements will be shown off in the form of badges that they obtain for completing them.

To add further detail to the “learning reward system” my solution is that as a student progresses through the course and reaches check points they will gain gifts that will reward them with all sorts of customisation options for their profile such as new profile pictures, a background for their profile and different colour schemes for the website itself. This will add to the incentive for students to work through the courses. There will also be the option for tutors to send their students gifts if they feel the student is deserving of it.

In regards to “accessibility features” my solution is to make sure the website colour scheme is something that colour-blind people would still be able to read as well as having a text to speech or font enlargement button to help those with visual impairment. I would also want the website be versatile and be able to function on various devices such as laptop, tablet and phone

Finally, for “collaborative teaching and learning tools” My solutions are already stated above with things like the comment system and allowing teachers to make quizzes that don’t have to be used for gamified learning and instead used to create a more personalised quiz for certain students.

Student	Tutors
Students will be able to: • Use any of the courses available on the website • See the results on each course once they’ve finished it • Make their own quizzes • Use other users’ quizzes • View the leader boards • View other users’ profiles • Add other users as friends • Customise their profile using the rewards they get • Show off various achievements through badges • Change the text size or use text to speech	Tutors will be able to: • Use any of the courses available • Request new courses to be made • See the progress their students are making on the courses • Get more detailed data on how well their students are doing with each course • Make their own quizzes • View leader boards • Send a reward to students • Comment on a student’s course • Set deadlines for courses to be done • Change the text size or use text to speech

<u>Mitigation of potential risks</u>

One potential risk that could occur is that the website could have glitches that weren’t found or there could be issues for certain users when trying to use the website. This will be mitigated by having a customer support section where any potential problems a user has can be reported and then dealt with.

Another risk is that Users may forget their account information and so a forgot username/password option will be available for them if needed.

Another risk is that there might not be a course the user wants. This can be mitigated by having a course request button that lets tutors suggest certain courses.

The users may not like the reward system, this can be mitigated by having a review section for users to leave their thoughts so that a better solution could be thought up.

More users than expected could load the website and cause server instability, this can be mitigated by doing stress tests on the website and making ways to redirect traffic if needed.

Regulatory guidelines and legal requirements

In order to stay in line with the health and safety (display screen equipment) regulations 1992 staff will be properly trained in how to use DSE and how to avoid any injury from long term use. They will need be given free eyesight tests in case of any need for spectacles

Training about the 2010 equality act will also be carried out for all tutors to make sure they all understand what discrimination is and what equality and diversity is all about.

Tutors would also need to carry out training on how children should be looked after in the eyes of the law to follow the 2004 children's act

In order to comply with GDPR all users will be notified any time personal data is being collected and no personal data shall be kept longer than it needs to be.

Functional requirements

No.	Features	Priority	Justification
1.	Users are able to sign up	HIGH	The user won't be able to use any features on the website if this doesn't work
2.	Users are able to log in	HIGH	The user won't be able to get back into the website if this doesn't work
3.	Send an email to the customer when they make an account	LOW	This is just a check that the system has the right credentials
4.	System will record user data on courses	HIGH	This will be needed in order for tutors to check how students are doing on courses and will also allow for tracking trends in what courses people are doing

5.	Provide gifts when users reach milestones in courses	MEDIUM	Is important for gamified learning aspects but course page and quiz page take priority
6.	Record scores on quizzes and put them on a leader board	MEDIUM	Is important for gamified learning aspects but course page and quiz page take priority
7.	Send any comment that the tutor makes to the student	HIGH	Is needed in order for tutors to provide feedback to their students

Non-functional requirements

No.	Features	Priority	Justification
1.	Website has a good response time	HIGH	If the website doesn't feel responsive to users they will most likely go somewhere else
2.	Website is adaptable to any viewing device e.g. laptop, smart phone, tablet.	MEDIUM	Whilst it's important to have the website accessible to people no matter the device, a majority of users will be using a computer or laptop and so those take priority.
3.	Colour scheme is appropriate for any user	HIGH	User accessibility is important and if some users can't see the website properly they won't use the service
4.	User login credentials and any other personal data is secure	HIGH	The user needs to be able to trust the service with the credentials or they just won't use it.
5.	The system should be available at all times	HIGH	The website can lose a lot of traffic if it's not available to everyone when they want it.

Decomposition of problems to be solved for functional and non-functional requirements

- In order to record all the user data, we will require a large database to store it on
- Some students may not own a digital device and so therefore a set number of laptops can be provided to those students for work.
- In order to make it adaptable to multiple devices further research on how to adaptable html is required
- To make the colour scheme appropriate for any user research on what colours can be seen by everyone is required
- Use of CSRF tokens is needed to make sure user data is secure.

KPI

- How many users that click on the website make accounts.

Keeping track of this will allow you to see how many users are being lost from initial impressions of the website and if it seems like a large number of users then changes could be made to retain user attraction.

- How long on average do users spend on the website.

You need to keep track of this so you can see how well the website is doing.

- How many users use the customisation options.

It's important to track this as you'll then be able to see how effective of a reward it is and whether changes are needed.

- How many users use the quiz section of the website.

It's important to keep track of this to see how many people are liking it and whether or not changes will need to be made to improve users enjoyability of it.

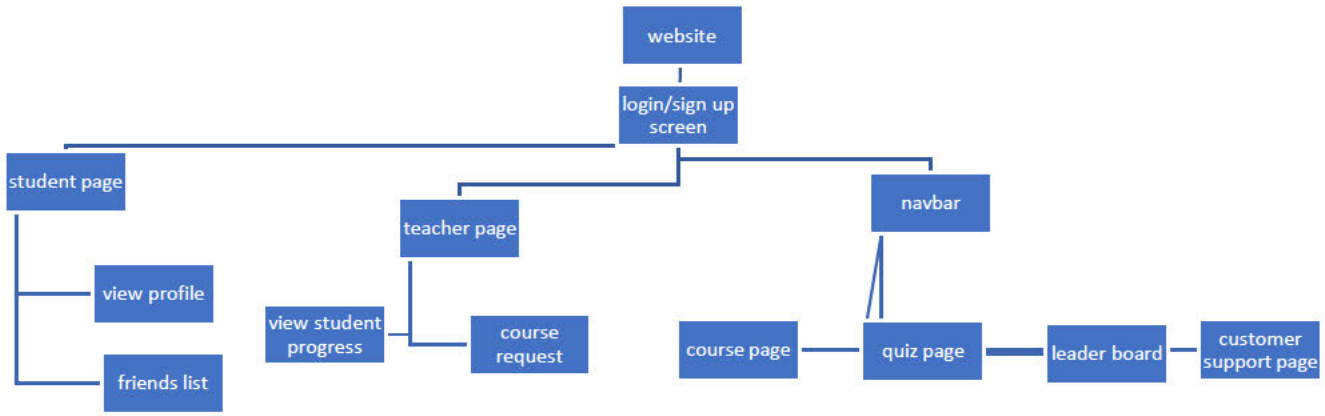
- What courses are the most popular.

It's important to keep track of this as you can then see what's in high demand and can advertise those courses more to pull in more users.

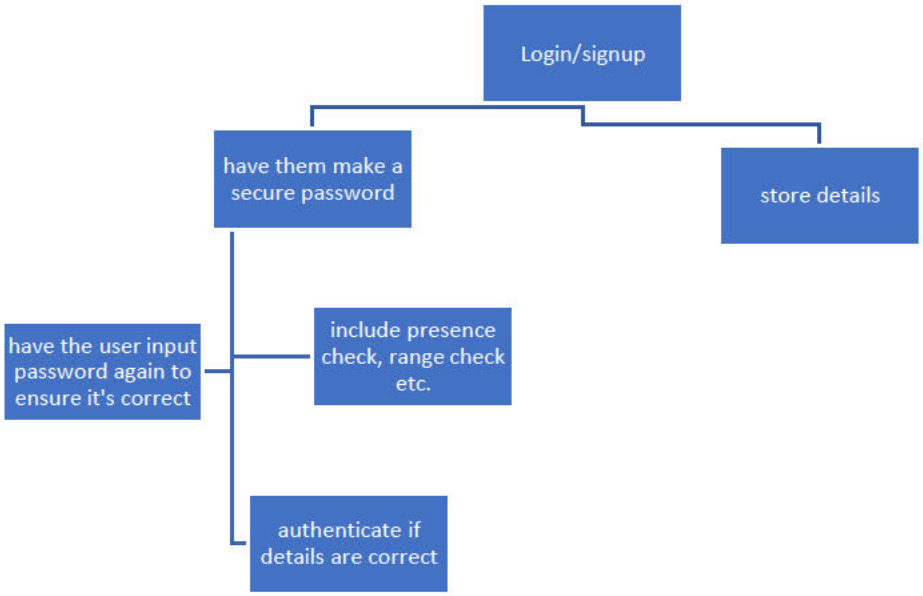
User acceptance criteria

- Must have a learning reward system that allows users to unlock customisation options on their website
- Must provide teaching and learning resources in a wide range of subjects
- Must be adaptable to various devices
- Must be able to monitor student progress
- Must provide access to digital content
- Layout of the page should be simple to use and easy to navigate
- Must have gamified learning features that allow students to have some fun.

Hierarchy diagram



The website consists of 6 main pages these being: the student homepage, teacher homepage, quiz page, leader board, course page and customer support page. These 6 pages will be my main focus when making the website as they make up the most important features needed in order to meet my client's requirements. There are then the 4 other features involved in the student and teacher pages, these being: view profile, friends list, view student progress and course request. I will attempt to implement these provided I have enough time after making the main 6 pages with the focus being on the student progress and view profile features as these help make the website feel more involved and interactive for the user.

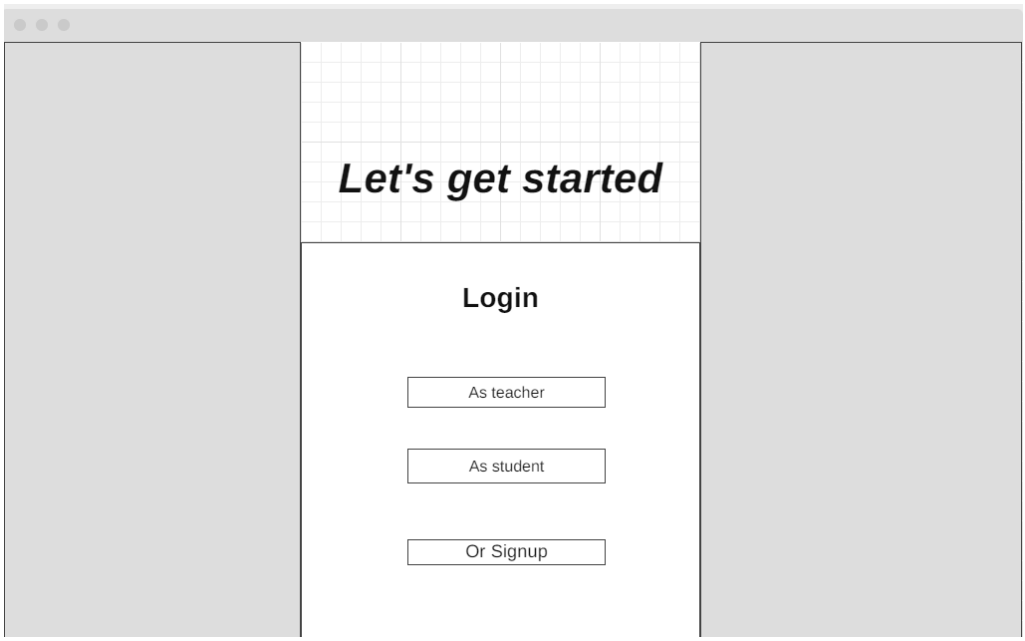


This is the breakdown of the components in signing up or logging into the website, the two parts to this component are storing the details which will store the information they input to the database- and making the secure password which has 3 main parts to. Making sure the user actually input the information, having the user input the information again and authenticating that the password is correct.

Activity B

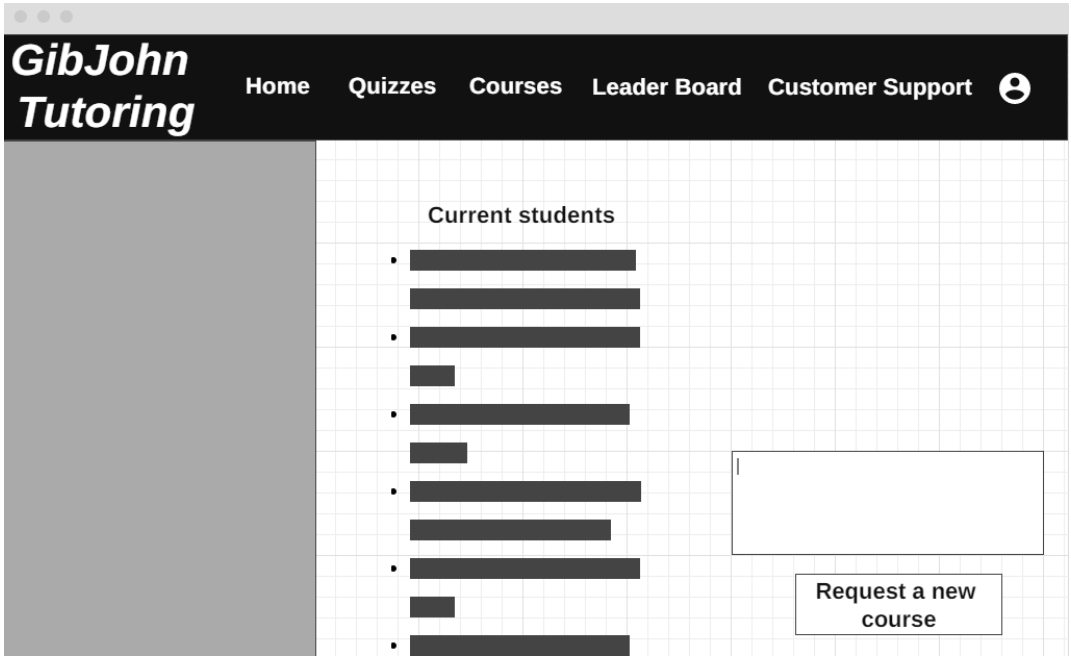
Visual designs

Login screen



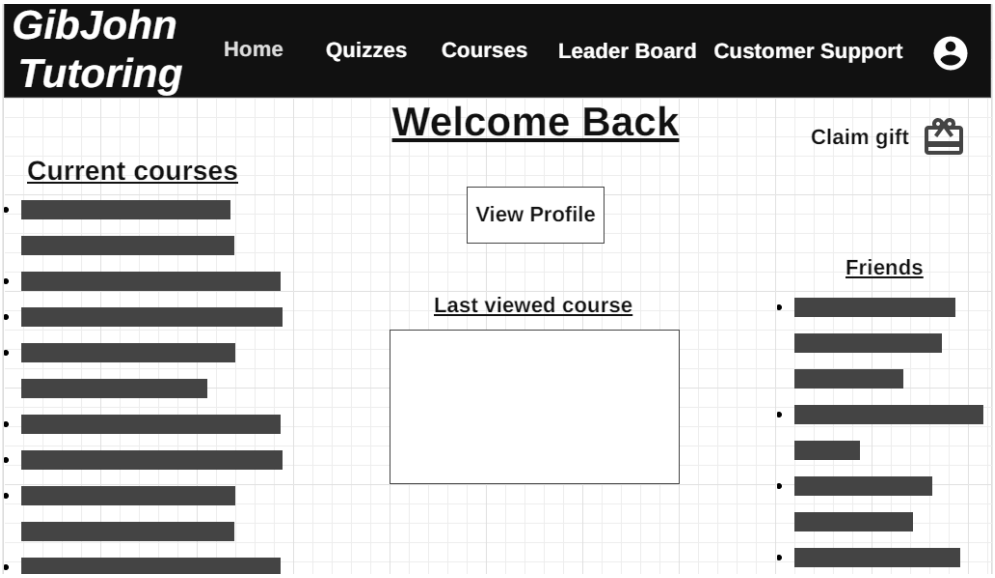
This is the layout plan for the login/signup page, I've made sure to keep it simple (as to avoid confusing any new user clicking onto the website) with clear buttons directing the user where they need to go. The grey boxes represent the areas on the page that I would want to put an image or have some form of a background due to this being the standard practice for many log-on screens and so would confuse the user less.

Teacher Homepage



This is the homepage screen for when a tutor logs in. I've put the navbar at the top with the company name in the top left corner and profile picture in the top right corner as it's standard practice for a majority of websites and so will help the website feel intuitive. This navbar will also be the same for every single page of the website as it allows for easy navigation. You then have in the middle of the screen a list of the students that the tutor has so that they can easily reach whichever student they want and check up on the students' progress. The option to request another course is also on this page to make it easy to find. The grey box is to show that something will be placed there in the final product, as to avoid too much white space, this will either be another function that gets thought up in later planning/testing or some sort of image.

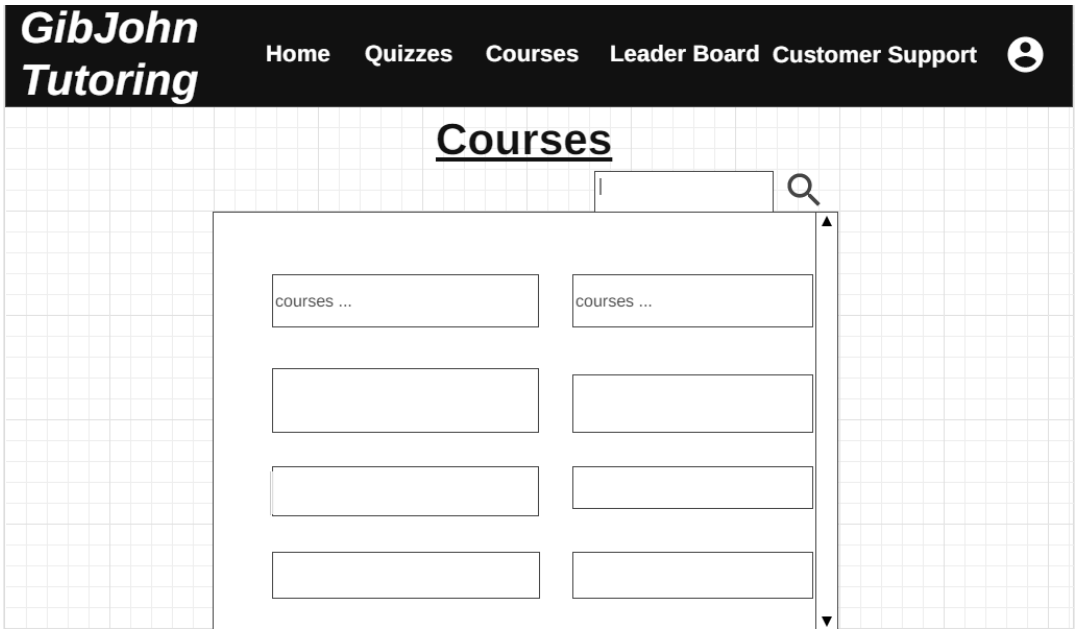
Student homepage



This is the homepage for the student, on the left of the screen there is going to be a list of the current courses that the student is doing so they can have easy navigation to whatever course it is they need with a headline of current courses as to not confuse any new user about what the list. With this same line of thought there is also going to be a last viewed course in the middle of the screen as it's more than likely the user will want to go back to the same course they were doing and this gives them a quick and clear way of reaching it. The view profile button is also in the middle of the screen and will take the user to their profile page. Finally, on the right of the screen there is the claim gift button that will allow students to claim any rewards they've been given by courses or tutor and the friends list that will let students quickly reach any of their friends' profiles. I've made sure to leave some amount of white space in between each section of the page so that it doesn't feel too clumped

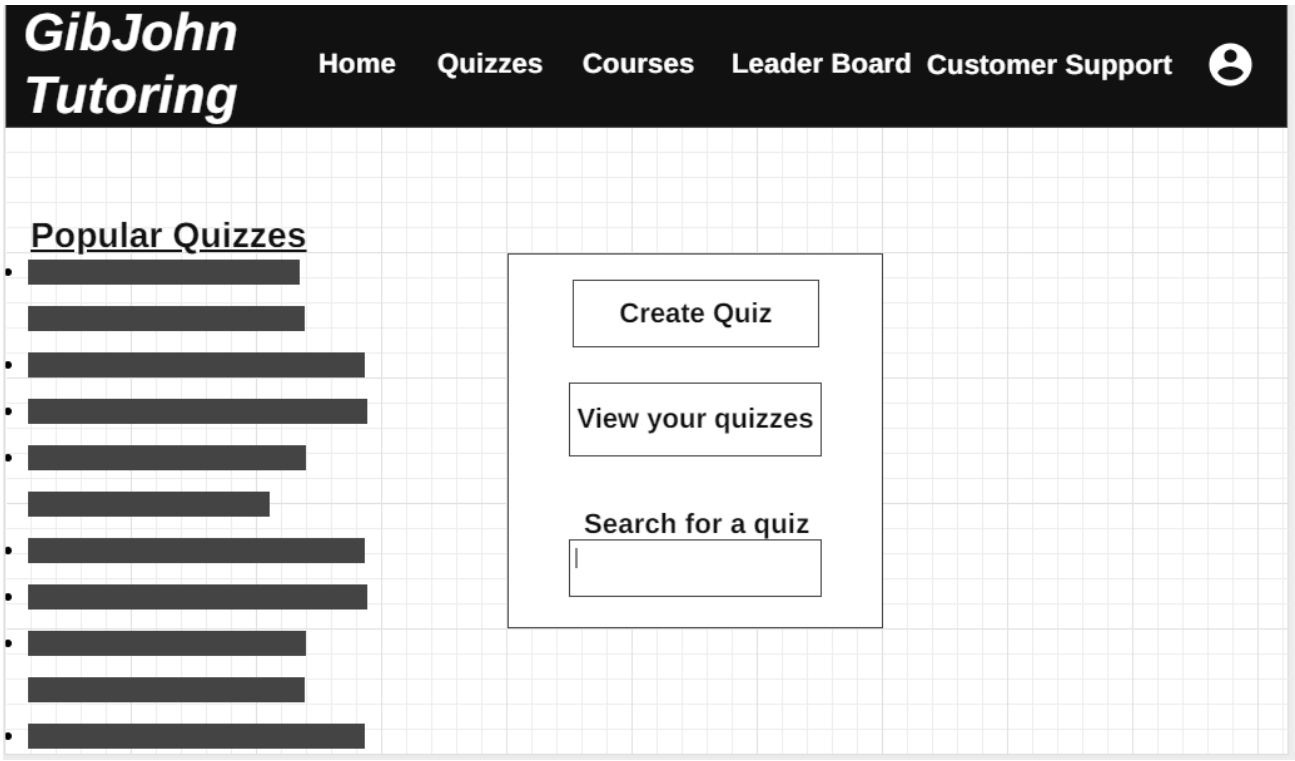
(the spacing between lines is something to do with word that I don't know how to fix)

courses page



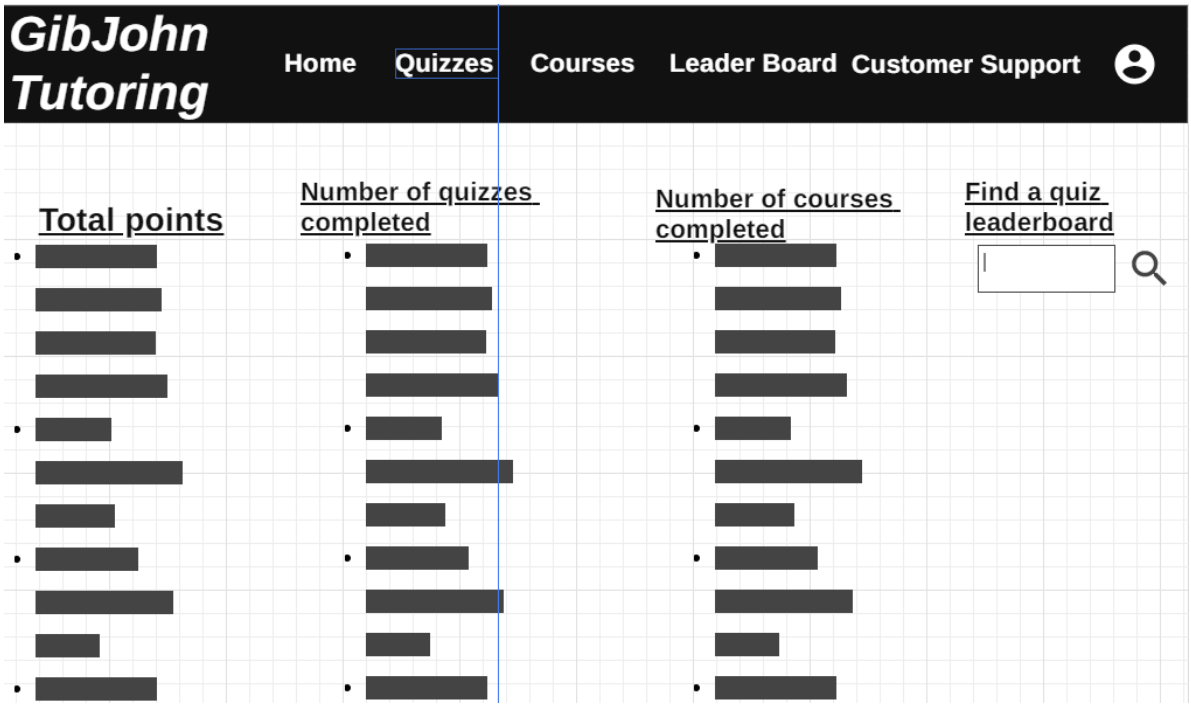
This is the Courses page, the course page itself will have a list of every course available to the user that they can scroll through until they find what they need or alliteratively they can use the search bar to see if the course is there.

Quizzes page



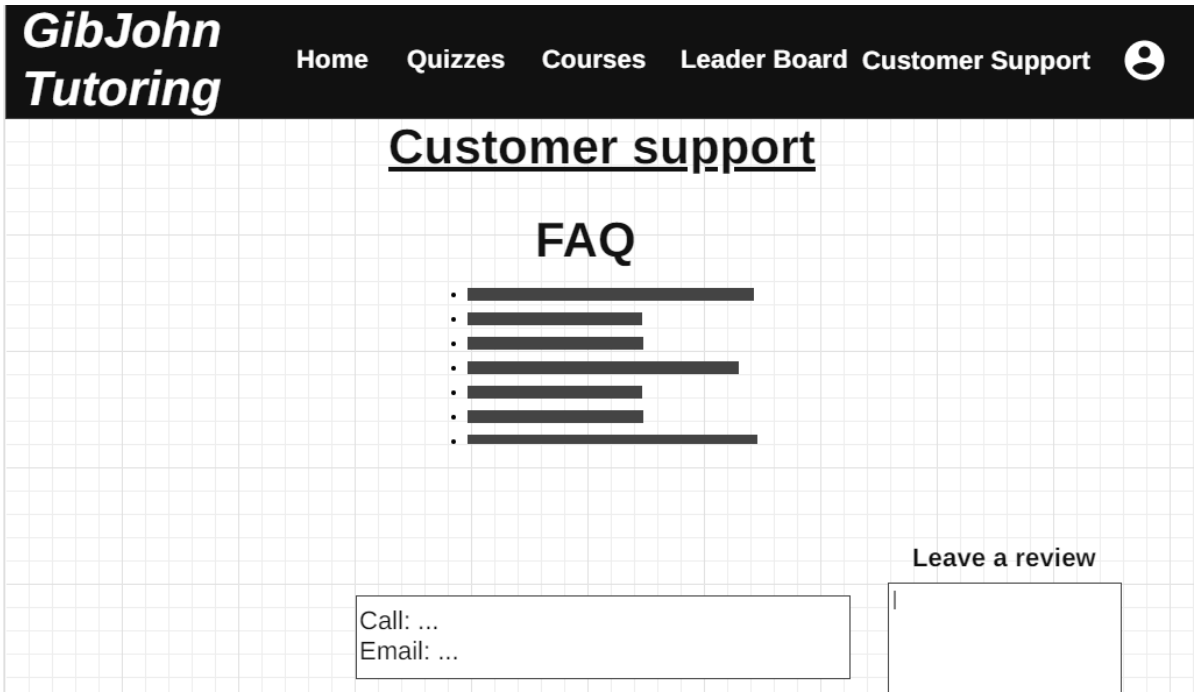
This is the quizzes page, on the left of the screen I am going to have a list of the popular quizzes as then anyone who's just looking for Any sort of quiz can immediately see well-made ones. The quizzes that show here will be decided by the amount of likes they get from users who have completed them. Then in the middle of the screen there are going to be multiple boxes that will allow the user to create their own quizzes, view any previous quizzes they've made so they can edit them or remove them and an option to search for quizzes so that anyone can find more specific quizzes they may want.

Leader board page



This is the leader board page, What I would want for this page is to have multiple general leader boards showing for anyone to look at quickly when entering the page and then have a search bar that allows users to search for a more specific leader board. For example, some users may have made their own quiz and just want to see how each other are doing for that rather than see overall figures. however, I'm currently not too happy with the layout I've made above and may make changes later on when I see fit.

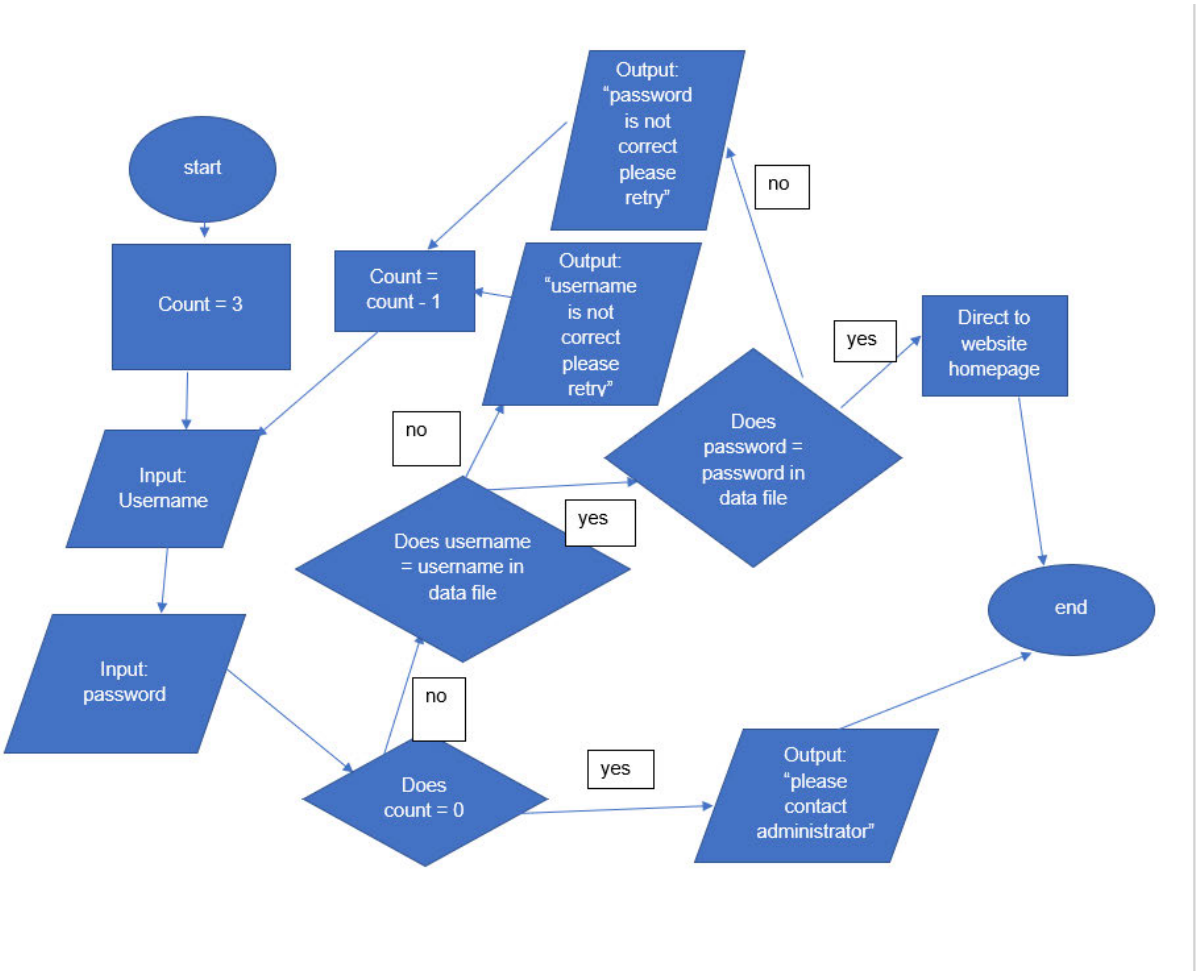
Customer support page



This is the customer support page, in the middle of the screen there will be some Frequently asked questions as this will most likely solve many users' qualms. At the bottom of the screen there will also be the contact information to the support team the user can use for any other problem they may have. Finally, there will also be a review box here for anyone that wishes to provide any feedback to the website so that further improvements can be made later down the line.

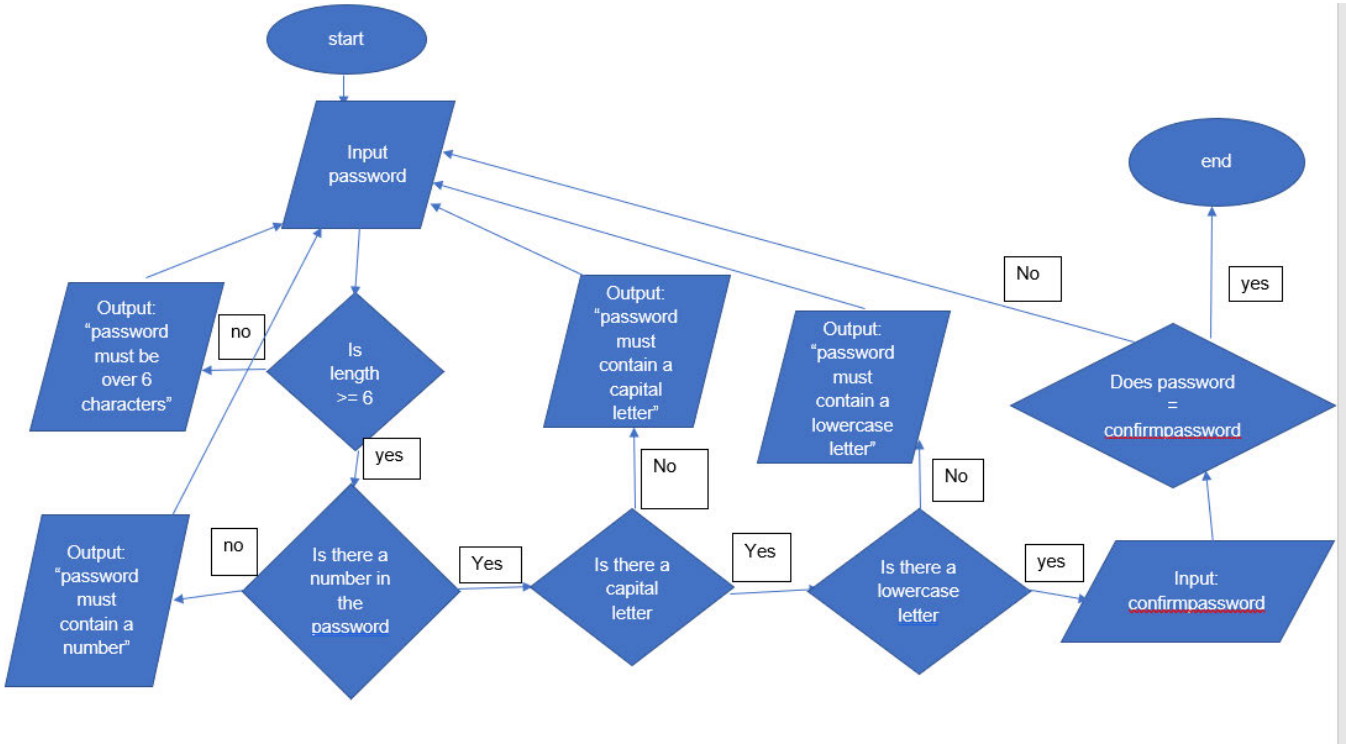
Algorithm designs

Login:



Justification: once the user puts in the correct username and password they are allowed onto the main site otherwise after three attempts the session times out due to the loop created with count that stops bots from logging into random accounts.

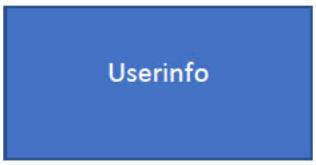
Validating password:

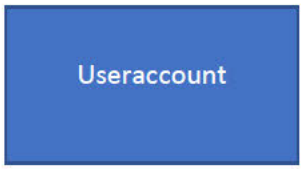


Justification: the password input and confirmpassword input would be done within a different algorithm when actually writing it but to make this easier to understand I have included them here as well. In order to make sure the password is strong the validation requires them to enter a password over 6 characters long with at least one upper and one lowercase letter. The validation will then also compare the password and confirmed password to make sure that the user's password is entered correctly.

Data requirements

Variable name	Function	Data type	reason
$password	Assigns the users input to a variable for the password	string	To allow the system to validate the password
$passwordCon	Assigns the users input to a variable for confirm password	string	to allow the system to validate the password
$passwordError	To output an error message if the password is wrong	string	If there is an error with the password it outputs a message so the user knows
$passwordConError	To output an error message when the confirm password is checked against the password and its wrong	string	If there is an error with confirming the password it outputs a message so the user knows
$fNameCheck	Makes sure the first name field is filled in	string	To make sure there is a present value
$fName	Assigns the users input to a variable for the first name	string	To assign a value to the first name
$sNameCheck	Makes sure the first name field is filled in	string	To make sure there is a present value
$sName	Assigns the users input to a variable for the surname	string	To assign a value to the surname





These are the entities I will be including on my database. Both are simple standalone tables.

UserInfo	
Field	**Data type**
UserNameID	Int (10)
First Name	Varchar (20)
Surname	Varchar (20)
Email	Varchar (50)
Telephone	Varchar (13)

User Account	
Field	**Data type**
UserNameID	Int (10)
Password	Varchar (between 3 and 16 characters)

Testing strategy

I'm going to use black box testing where I test the inputs and outputs of the website as well as white box testing where I'll test the backend/internal structure of the website.

Test strategy	Black box testing
Purpose	the test strategy allows for a visual representation of all the testing that can be devised for each and every input and output on the program making sure everything functions as intended
who performs the test	Me
Test data set	Every action/input and output will be recorded
Test criteria	That the program outputs what's intended
When to test	Whenever a section of the website is done and the inputs and outputs need to be tested
Estimated time required	Each section of the website will take up to a maximum of 30 minutes depending on how many inputs and outputs it has
Test outcome	Outcomes will be compared with what the intended outcome was meant to be and any improvements will be made to those outcomes that don't do as intended.

Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies
	Login page: When the password doesn't meet the requirements an error message will show	Black box testing	Need access to the website Computer Input an invalid password
	Login page: When the password and confirm password don't match an error message will show	Black box testing	Need access to the website Computer Input a different password in confirm password to what's written in password
	Home page: when the various buttons are clicked the system takes the user where it's supposed to	Black box testing	Need access to the website Computer User needs to login to reach the homepage
	Course page: when a course is clicked on the correct course will then be loaded	Black box testing	Need access to the website Computer User needs to login and then click onto the courses page
	Leader board page: the leader boards update automatically every 10 minutes	Black box testing	Need access to the website Computer User needs to login and then click onto leader board page
	Quiz page: when the user clicks on a quiz that quiz will load	Black box testing	Need access to the website Computer User needs to login then go on the quiz page
	Security of the page from outside attacks	White box testing	Computer Access to the website

Task 1: Test strategy

Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies

Sources table

Source name	link to source
Seneca	https://senecalearning.com/en-GB/
kahoot	https://kahoot.com/
go student	https://www.gostudent.org/en
Quizizz	https://quizizz.com/
Laws and regulations for schools	https://www.ihasco.co.uk/blog/entry/2718/regulations-that-apply-to-schools

Activity A

The proposal/justification

My solution for the client would be to create a website that provides the students with an easy to use place that helps them revise the work they have been doing with tutor and create an easy way for the tutor to track the progress of their students. This website will have a student or teacher option when signing up so that once the user is logged in the website can be more tailored to what the user would be using it for.

One of the pages for this website would offer a wide variety of courses for the students and “provide access to digital content to encourage wider learning”. These will help them go back over the topics they have been learning to retain and reinforce the information in their mind. This page would display all the courses on offer with a navigation bar to help find the course they want. These courses would display the information that the student has been learning about and then ask them basic questions about that information. E.g. for English asking them a multiple-choice question about what a certain word means. I’ve come up with this solution after looking at a website called Seneca that can been seen in my appendix or in my sources table. This is also what I am basing the look of my website on.

Another page of this website that is only accessible by a tutor would allow them to “monitor the learner progress” their students are making on each course showing things like the speed they are getting through each course, how many questions they’re getting right and an estimation of how well they seem to be comprehending the task. And after seeing this information the tutor can then leave comments for the students to see. Providing “interactive teaching” that can help the student in their weaker areas. The Tutor will also be able to set deadlines so that the student

knows how long they have to complete it. I got the idea of tracking the student's data in each course from go student that can been seen in my appendix.

To incorporate the features customers and tutors would want into the solution like "gamified learning" I would also make a page that allows tutors or students to create their own quizzes that students can do. These quizzes once done would show the score each student is getting on a leader board providing a competitive aspect to their learning and to try and make the quizzes seem more fun than just another bit of work to be done powerups can be added in that can provide students with different advantages during the quiz e.g. one powerup could remove some of the wrong answers so there's a better chance they get it right. I thought up this solution when looking at websites like kahoot and quizizz that can be seen in my appendix and in my sources table.

To tie in both the learning reward system and gamified learning I want to provide students with their own profiles that will display their various achievements to others, such as, the courses they have completed, the courses they've got 100% on, the number of times they've reached number 1 on leader boards or the number of days they've been able to retain the top spot. There will also be the option for students to add each other as friends so that they can view each other's profiles easier. This can add to the competitive aspect as students will want to show off their achievements to others (gamified learning) and it will also help incentivise the student to do the courses as it'll make the course feel more worthwhile to complete (learning reward system). These achievements will be shown off in the form of badges that they obtain for completing them.

To add further detail to the "learning reward system" my solution is that as a student progresses through the course and reaches check points they will gain gifts that will reward them with all sorts of customisation options for their profile such as new profile pictures, a background for their profile and different colour schemes for the website itself. This will add to the incentive for students to work through the courses. There will also be the option for tutors to send their students gifts if they feel the student is deserving of it.

In regards to "accessibility features" my solution is to make sure the website colour scheme is something that colour-blind people would still be able to read as well as having a text to speech or font enlargement button to help those with visual impairment. I would also want the website be versatile and be able to function on various devices such as laptop, tablet and phone

Finally, for "collaborative teaching and learning tools" My solutions are already stated above with things like the comment system and allowing teachers to make quizzes that don't have to be used for gamified learning and instead used to create a more personalised quiz for certain students.

Student	Tutors
Students will be able to: • Use any of the courses available on the website • See the results on each course once they've finished it • Make their own quizzes • Use other users' quizzes • View the leader boards • View other users' profiles • Add other users as friends • Customise their profile using the rewards they get • Show off various achievements through badges • Change the text size or use text to speech	Tutors will be able to: • Use any of the courses available • Request new courses to be made • See the progress their students are making on the courses • Get more detailed data on how well their students are doing with each course • Make their own quizzes • View leader boards • Send a reward to students • Comment on a student's course • Set deadlines for courses to be done • Change the text size or use text to speech

Mitigation of potential risks

One potential risk that could occur is that the website could have glitches that weren't found or there could be issues for certain users when trying to use the website. This will be mitigated by having a customer support section where any potential problems a user has can be reported and then dealt with.

Another risk is that Users may forget their account information and so a forgot username/password option will be available for them if needed.

Another risk is that there might not be a course the user wants. This can be mitigated by having a course request button that lets tutors suggest certain courses.

The users may not like the reward system, this can be mitigated by having a review section for users to leave their thoughts so that a better solution could be thought up.

More users than expected could load the website and cause server instability, this can be mitigated by doing stress tests on the website and making ways to redirect traffic if needed.

Regulatory guidelines and legal requirements

In order to stay in line with the health and safety (display screen equipment) regulations 1992 staff will be properly trained in how to use DSE and how to avoid any injury from long term use. They will need be given free eyesight tests in case of any need for spectacles
Training about the 2010 equality act will also be carried out for all tutors to make sure they all understand what discrimination is and what equality and diversity is all about.
Tutors would also need to carry out training on how children should be looked after in the eyes of the law to follow the 2004 children's act
In order to comply with GDPR all users will be notified any time personal data is being collected and no personal data shall be kept longer than it needs to be

Functional requirements

No.	Features	Priority	Justification
1.	Users are able to sign up	HIGH	The user won't be able to use any features on the website if this doesn't work
2.	Users are able to log in	HIGH	The user won't be able to get back into the website if this doesn't work
3.	Send an email to the customer when they make an account	LOW	This is just a check that the system has the right credentials
4.	System will record user data on courses	HIGH	This will be needed in order for tutors to check how students are doing on courses and will also allow for tracking trends in what courses people are doing
5.	Provide gifts when users reach milestones in courses	MEDIUM	Is important for gamified learning aspects but course page and quiz page take priority
6.	Record scores on quizzes and put them on a leader board	MEDIUM	Is important for gamified learning aspects but course page and quiz page take priority
7.	Send any comment that the tutor makes to the student	HIGH	Is needed in order for tutors to provide feedback to their students

<u>Non-functional requirements</u>

No.	Features	Priority	Justification
1.	Website has a good response time	HIGH	If the website doesn't feel responsive to users they will most likely go somewhere else
2.	Website is adaptable to any viewing device e.g. laptop, smart phone, tablet.	MEDIUM	Whilst its important to have the website accessible to people no matter the device, a majority of users will be using a computer or laptop and so those take priority.
3.	Colour scheme is appropriate for any user	HIGH	User accessibility is important and if some users can't see the website properly they won't use the service
4.	User login credentials and any other personal data is secure	HIGH	The user needs to be able to trust the service with the credentials or they just won't use it.
5.	The system should be available at all times	HIGH	The website can lose a lot of traffic if it's not available to everyone when they want it.

Decomposition of problems to be solved for functional and non-functional requirements

- In order to record all the user data, we will require a large database to store it on
- Some students may not own a digital device and so therefore a set number of laptops can be provided to those students for work.
- In order to make it adaptable to multiple devices further research on how to adaptable html is required
- To make the colour scheme appropriate for any user research on what colours can be seen by everyone is required
- Use of csrf tokens is needed to make sure user data is secure.

KPI

- How many users that click on the website make accounts.

Keeping track of this will allow you to see how many users are being lost from initial impressions of the website and if it seems like a large number of users then changes could be made to retain user attraction.

- How long on average do users spend on the website.

You need to keep track of this so you can see how well the website is doing.

- How many users use the customisation options.

It's important to track this as you'll then be able to see how effective of a reward it is and whether changes are needed.

- How many users use the quiz section of the website.

Its important to keep track of this to see how many people are liking it and whether or not changes will need to be made to improve users enjoyability of it.

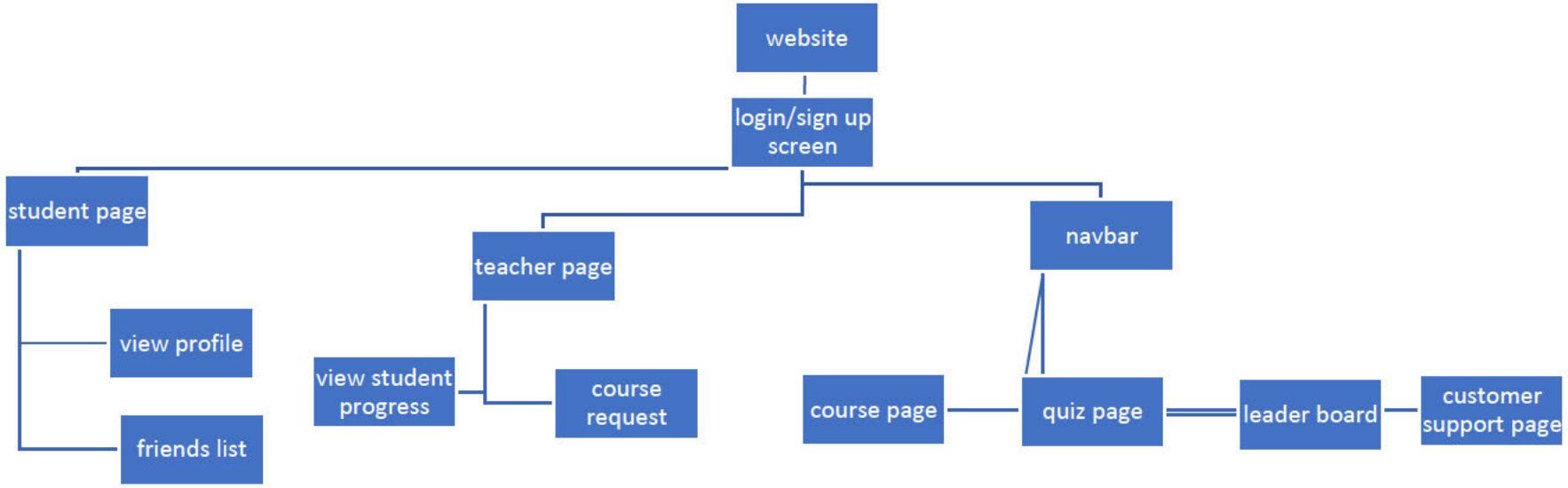
- What courses are the most popular.

It's important to keep track of this as you can then see what's in high demand and can advertise those courses more to pull in more users.

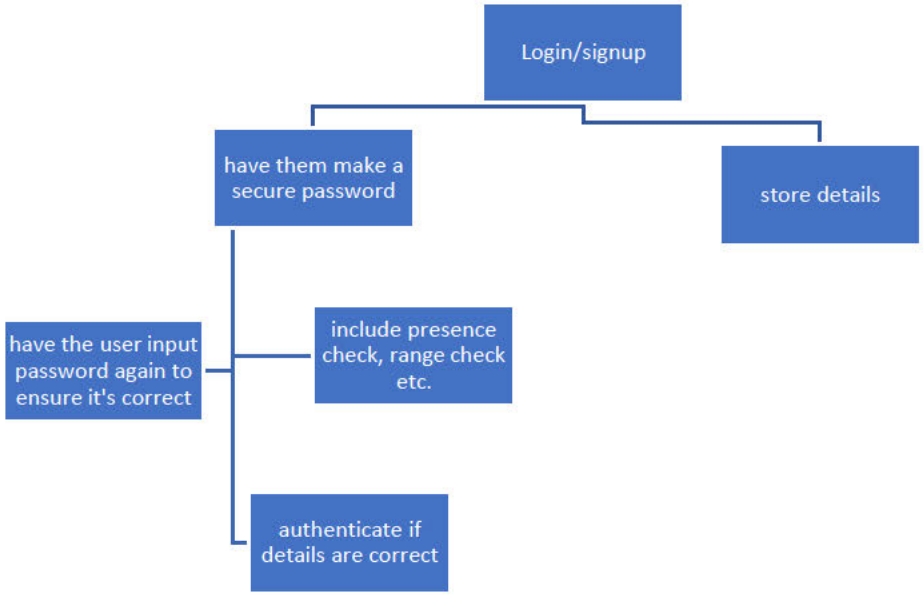
User acceptance criteria

- Must have a learning reward system that allows users to unlock customisation options on their website
- Must provide teaching and learning resources in a wide range of subjects
- Must be adaptable to various devices
- Must be able to monitor student progress
- Must provide access to digital content
- Layout of the page should be simple to use and easy to navigate
- Must have gamified learning features that allow students to have some fun.

Hierarchy diagram



The website consists of 6 main pages these being: the student homepage, teacher homepage, quiz page, leader board, course page and customer support page. These 6 pages will be my main focus when making the website as they make up the most important features needed in order to meet my client's requirements. There are then the 4 other features involved in the student and teacher pages, these being: view profile, friends list, view student progress and course request. I will attempt to implement these provided I have enough time after making the main 6 pages with the focus being on the student progress and view profile features as these help make the website feel more involved and interactive for the user.

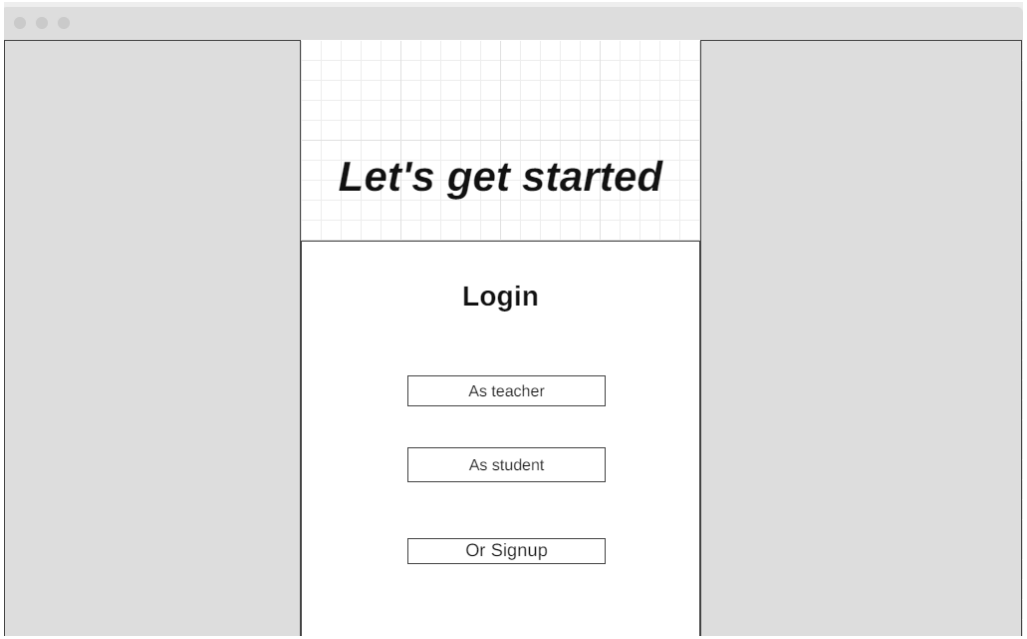


This is the breakdown of the components in signing up or logging into the website, the two parts to this component are storing the details which will store the information they input to the database- and making the secure password which has 3 main parts to. Making sure the user actually input the information, having the user input the information again and authenticating that the password is correct.

Activity B

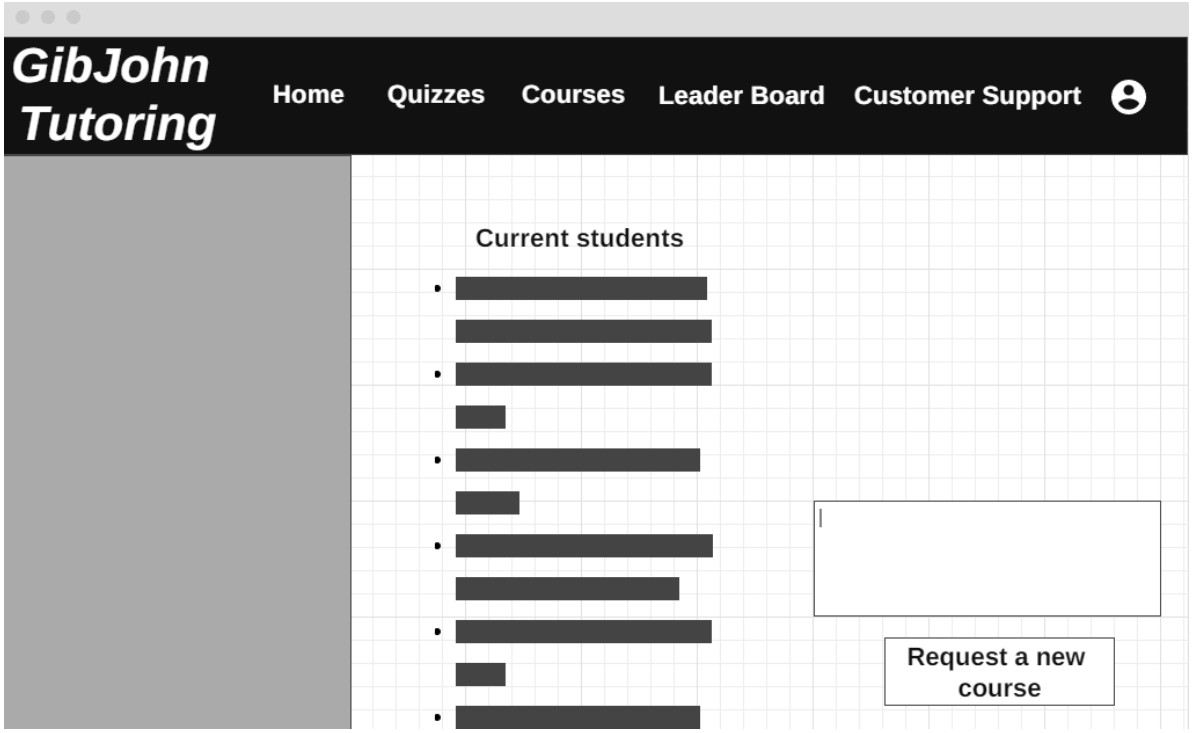
Visual designs

Login screen



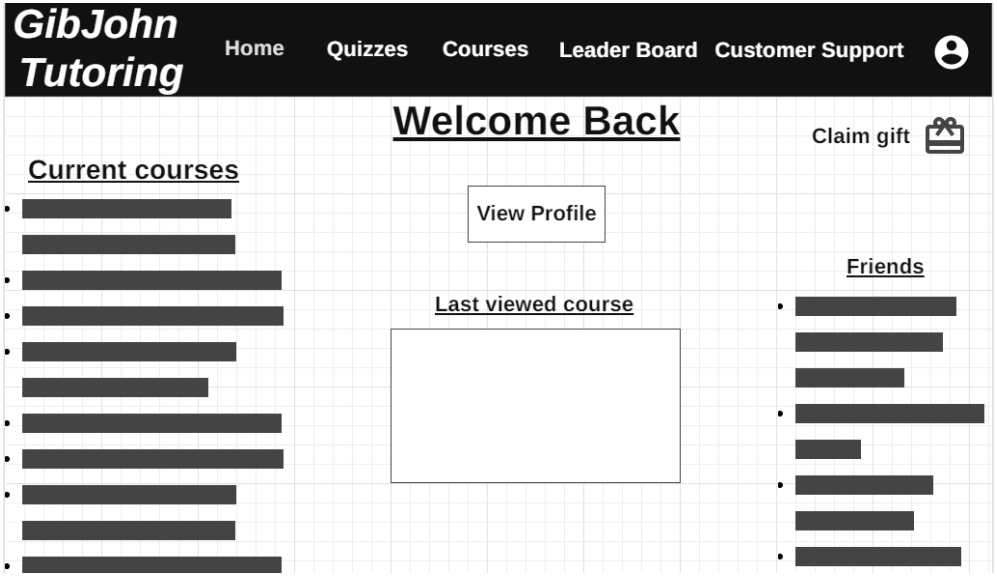
This is the layout plan for the login/signup page, I've made sure to keep it simple (as to avoid confusing any new user clicking onto the website) with clear buttons directing the user where they need to go. The grey boxes represent the areas on the page that I would want to put an image or have some form of a background due to this being the standard practice for many log on screens and so would confuse the user less.

Teacher Homepage



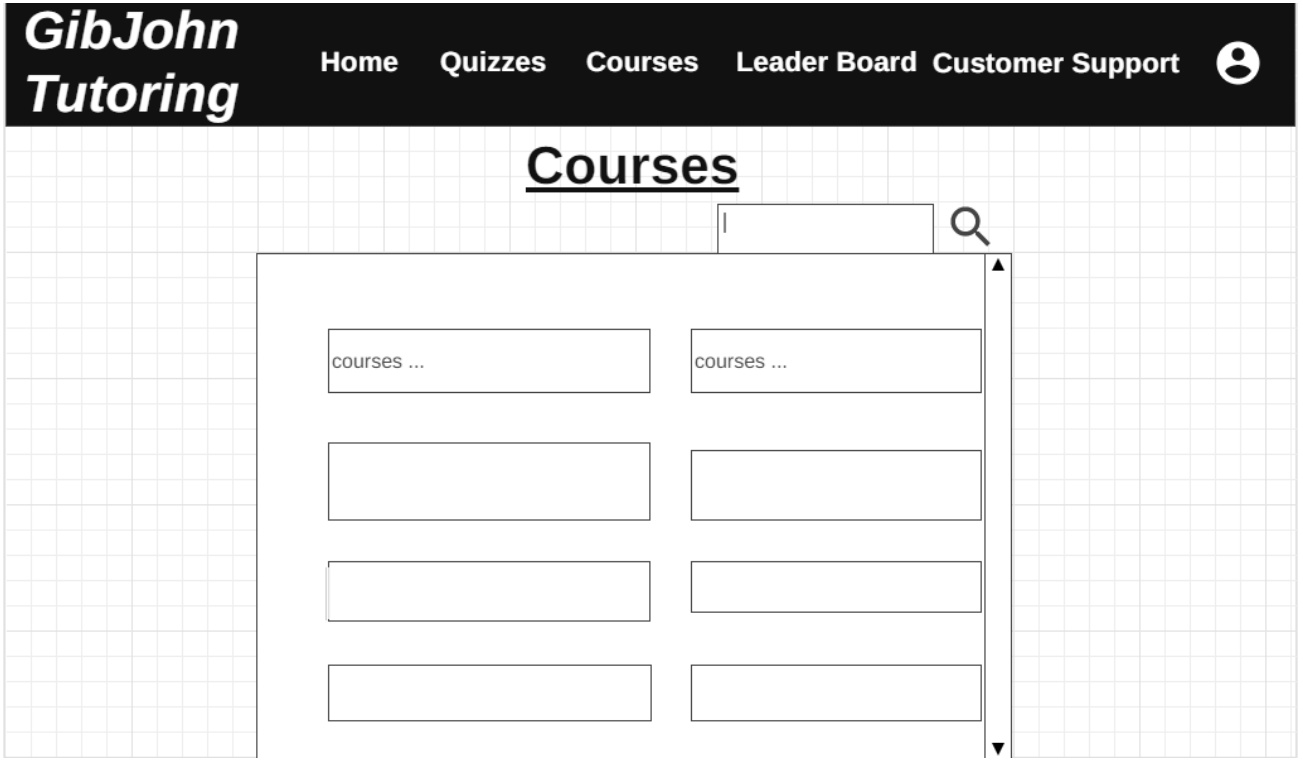
This is the homepage screen for when a tutor logs in. I've put the navbar at the top with the company name in the top left corner and profile picture in the top right corner as it's standard practice for a majority of websites and so will help the website feel intuitive. This navbar will also be the same for every single page of the website as it allows for easy navigation. You then have in the middle of the screen a list of the students that the tutor has so that they can easily reach whichever student they want and check up on the students' progress. The option to request another course is also on this page to make it easy to find. The grey box is to show that something will be placed there in the final product, as to avoid too much white space, this will either be another function that gets thought up in later planning/testing or some sort of image.

Student homepage



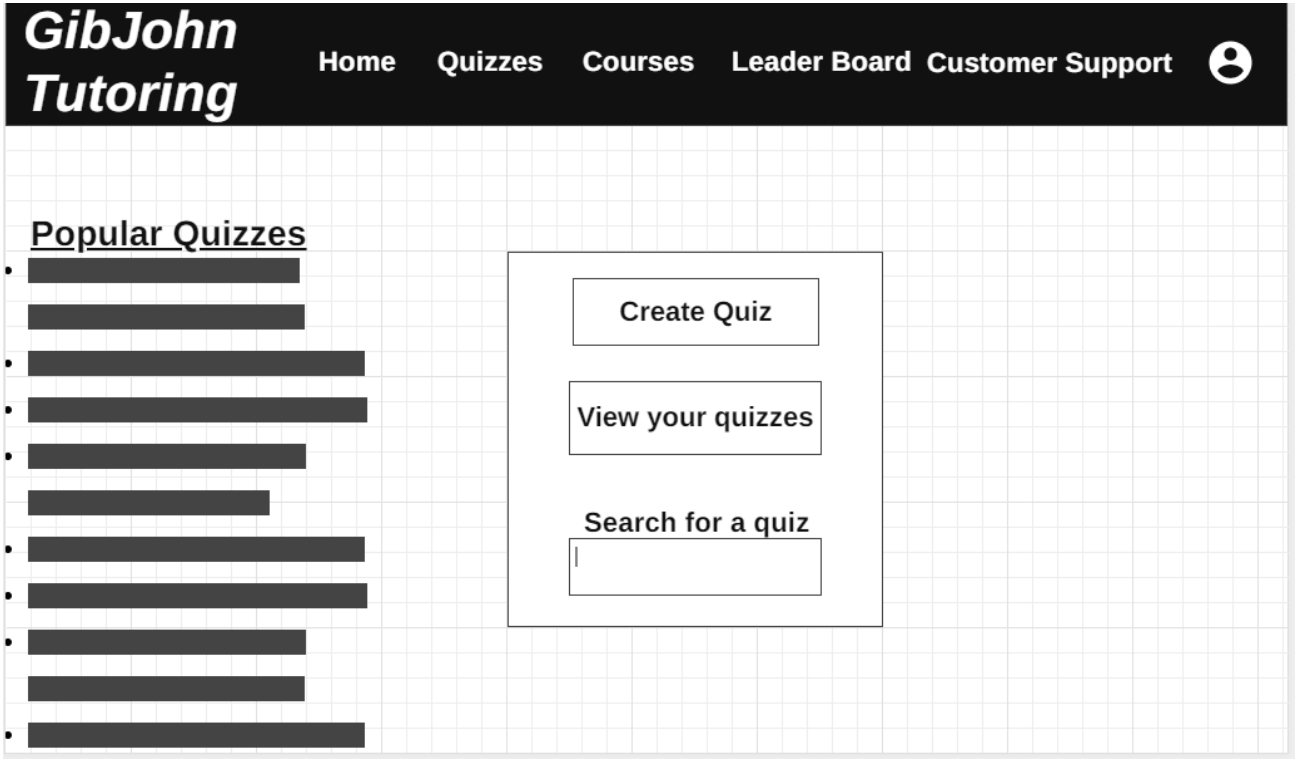
This is the homepage for the student, on the left of the screen there is going to be a list of the current courses that the student is doing so they can have easy navigation to whatever course it is they need with a headline of current courses as to not confuse any new user about what the list. With this same line of thought there is also going to be a last viewed course in the middle of the screen as it's more than likely the user will want to go back to the same course they were doing and this gives them a quick and clear way of reaching it. The view profile button is also in the middle of the screen and will take the user to their profile page. Finally, on the right of the screen there is the claim gift button that will allow students to claim any rewards they've been given by courses or tutor and the friends list that will let students quickly reach any of their friends' profiles. I've made sure to leave some amount of white space in between each section of the page so that it doesn't feel too clumped

courses page



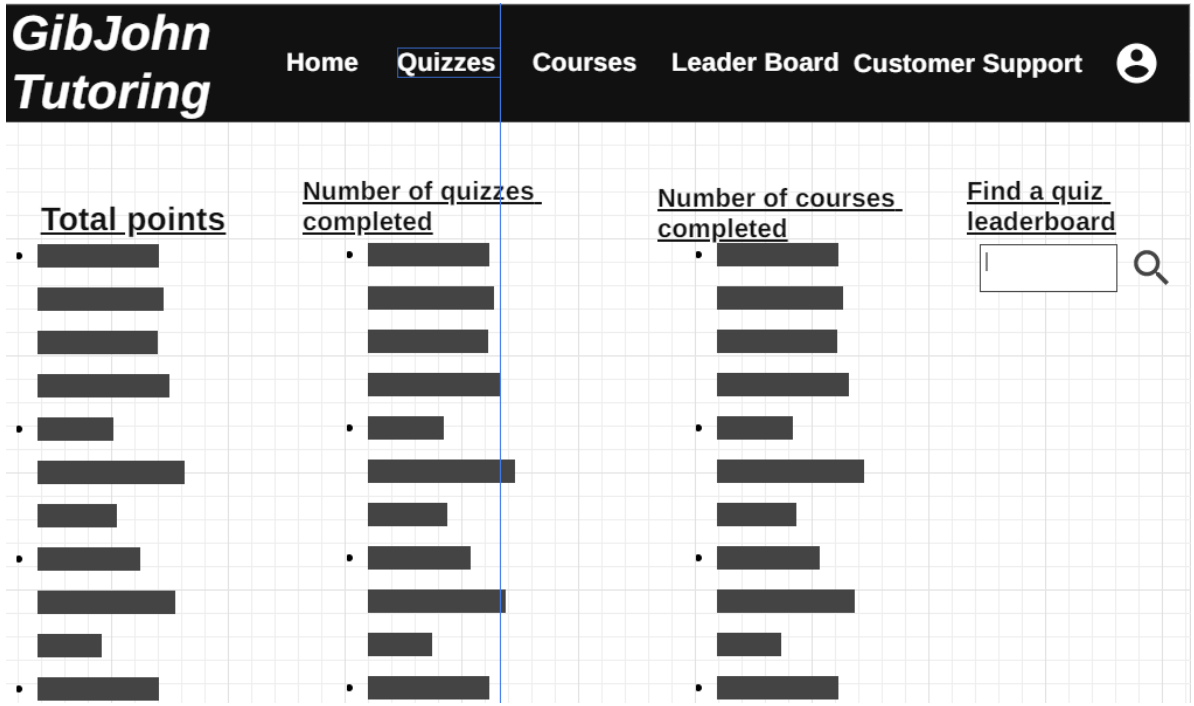
This is the Courses page, the course page itself will have a list of every course available to the user that they can scroll through until they find what they need or alliteratively they can use the search bar to see if the course is there.

Quizzes page



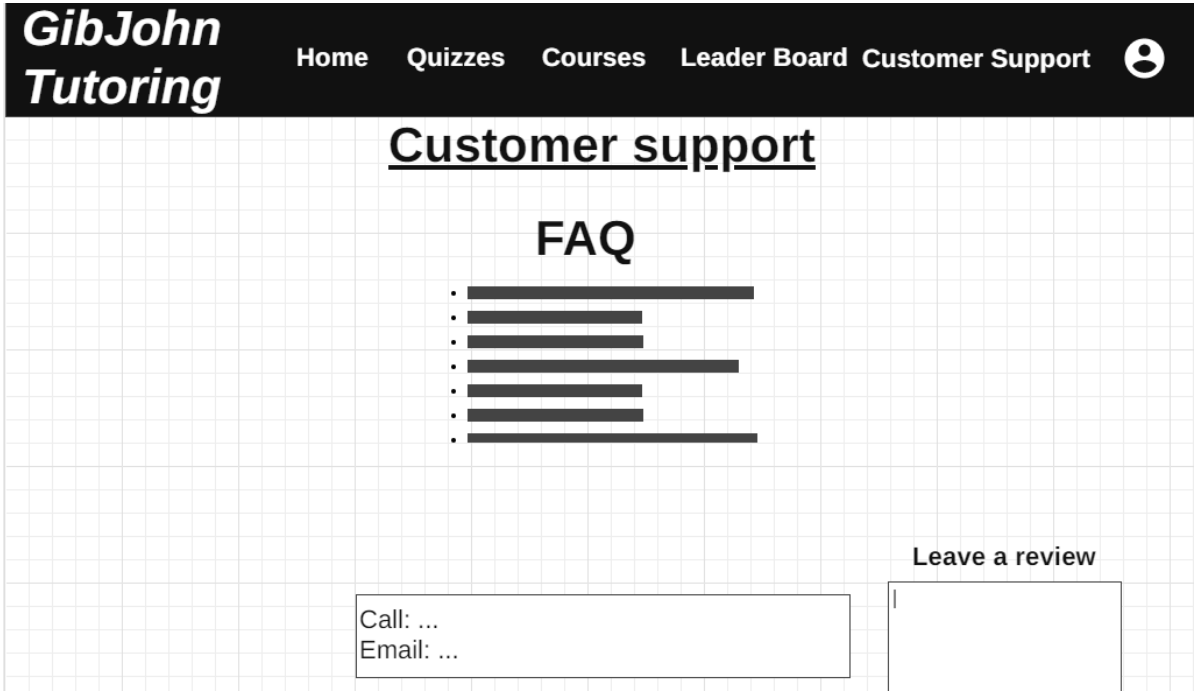
This is the quizzes page, on the left of the screen I am going to have a list of the popular quizzes as then anyone who's just looking for Any sort of quiz can immediately see well made ones. The quizzes that show here will be decided by the amount of likes they get from users who have completed them. Then in the middle of the screen there are going to be multiple boxes that will allow the user to create their own quizzes, view any previous quizzes they've made so they can edit them or remove them and an option to search for quizzes so that anyone can find more specific quizzes they may want.

Leader board page



This is the leader board page, What I would want for this page is to have multiple general leader boards showing for anyone to look at quickly when entering the page and then have a search bar that allows users to search for a more specific leader board. For example some users may have made their own quiz and just want to see how each other are doing for that rather than see overall figures. however, I'm currently not too happy with the layout I've made above and may make changes later on when I see fit.

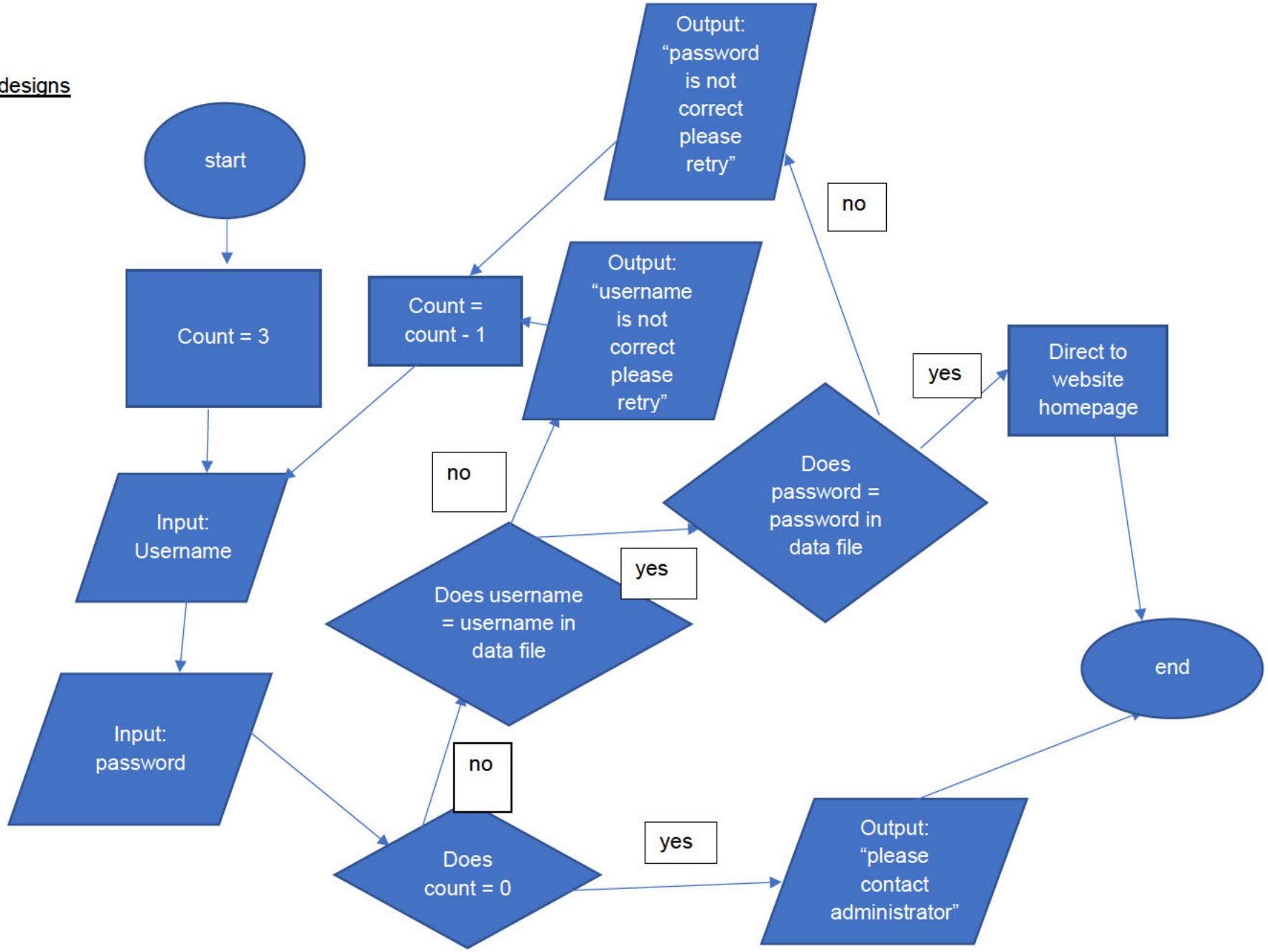
Customer support page



This is the customer support page, in the middle of the screen there will be some Frequently asked questions as this will most likely Solve many users' qualms. At the bottom of the screen there will also be the contact information to the support team the user can use for any other problem they may have. Finally, there will also be a review box here for anyone that wishes to provide any feedback to the website so that further improvements can be made later down the line.

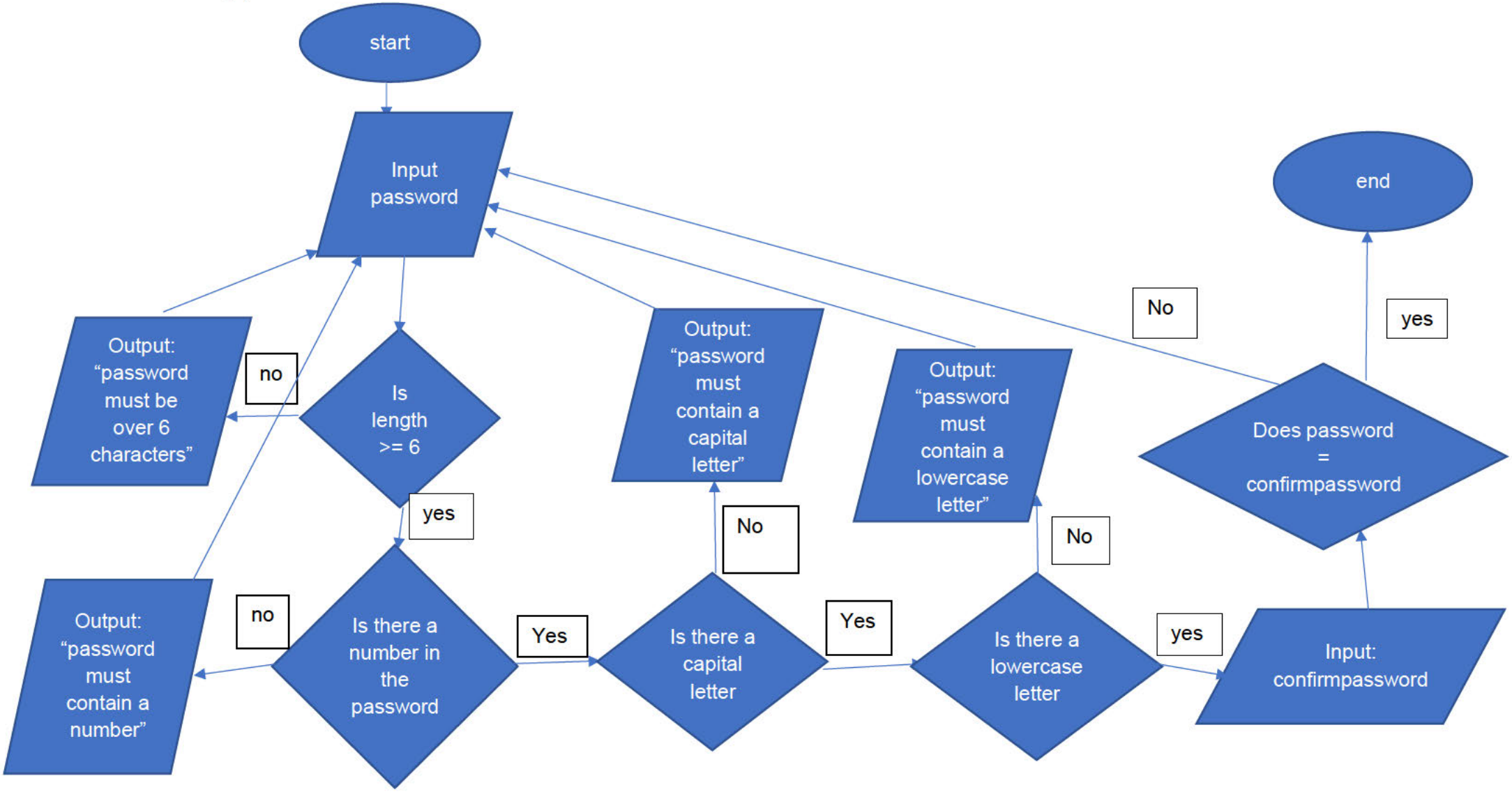
Algorithm designs

Login:



Justification: once the user puts in the correct username and password they are allowed onto the main site otherwise after three attempts the session times out due to the loop created with count that stops bots from logging into random accounts.

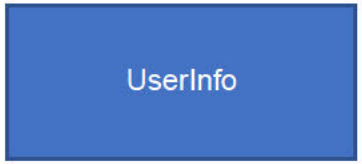
Validating password:

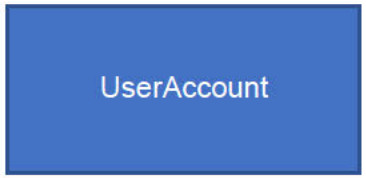


Justification: the password input and confirmpassword input would be done within a different algorithm when actually writing it but to make this easier to understand I have included them here as well. In order to make sure the password is strong the validation requires them to enter a password over 6 characters long with at least one upper and one lowercase letter. The validation will then also compare the password and confirmed password to make sure that the user's password is entered correctly.

Data requirements

Variable name	Function	Data type	reason
$password	Assigns the users input to a variable for the password	string	To allow the system to validate the password
$passwordCon	Assigns the users input to a variable for confirm password	string	to allow the system to validate the password
$passwordError	To output an error message if the password is wrong	string	If there is an error with the password it outputs a message so the user knows
$passwordConError	To output an error message when the confirm password is checked against the password and its wrong	string	If there is an error with confirming the password it outputs a message so the user knows
$fNameCheck	Makes sure the first name field is filled in	string	To make sure there is a present value
$fName	Assigns the users input to a variable for the first name	string	To assign a value to the first name
$sNameCheck	Makes sure the first name field is filled in	string	To make sure there is a present value
$sName	Assigns the users input to a variable for the surname	string	To assign a value to the surname





These are the entities I will be including on my database. Both are simple standalone tables.

UserInfo	
Field	**Data type**
UserNameID	Int (10)
First Name	Varchar (20)
Surname	Varchar (20)
Email	Varchar (50)
Telephone	Varchar (13)

User Account	
Field	**Data type**
UserNameID	Int (10)
Password	Varchar (between 3 and 16 characters)

Testing strategy

I'm going to use black box testing where I test the inputs and outputs of the website as well as white box testing where I'll test the backend/internal structure of the website.

Test strategy	Black box testing
Purpose	the test strategy allows for a visual representation of all the testing that can be devised for each and every input and output on the program making sure everything functions as intended
who performs the test	Me
Test data set	Every action/input and output will be recorded
Test criteria	That the program outputs what's intended
When to test	Whenever a section of the website is done and the inputs and outputs need to be tested
Estimated time required	Each section of the website will take up to a maximum of 30 minutes depending on how many inputs and outputs it has
Test outcome	Outcomes will be compared with what the intended outcome was meant to be and any improvements will be made to those outcomes that don't do as intended.

Date of test	Component to be tested	Type of test to be carried out	Prerequisites and dependencies
	Login page: When the password doesn't meet the requirements an error message will show	Black box testing	Need access to the website Computer Input an invalid password
	Login page: When the password and confirm password don't match an error message will show	Black box testing	Need access to the website Computer Input a different password in confirm password to what's written in password
	Home page: when the various buttons are clicked the system takes the user where it's supposed to	Black box testing	Need access to the website Computer User needs to login to reach the homepage
	Course page: when a course is clicked on the correct course will then be loaded	Black box testing	Need access to the website Computer User needs to login and then click onto the courses page
	Leader board page: the leader boards update automatically every 10 minutes	Black box testing	Need access to the website Computer User needs to login and then click onto leader board page
	Quiz page: when the user clicks on a quiz that quiz will load	Black box testing	Need access to the website Computer User needs to login then go on the quiz page
	Security of the page from outside attacks	White box testing	Computer Access to the website

Assets table

asset	Date it was retrieved	Use of asset
bootstrap	8/03/2022	I'm going to be using bootsrap throughtout my webpage to help with the frontend and improve the user experience
	18/03/2022	I am using this from bootstraps icons for the profile picture
	18/03/2022	I'm using this for the claim gift button the student homepage

Sources table

link	What the link goes to/was used for
https://docs.djangoproject.com/en/4.0/topics/forms/	Used this to understand the basic of setting up my own form.
https://www.javascripttutorial.net/javascript-dom/javascript-radio-button	I used this for help in implementing radio buttons for JavaScript

Testing throughout the project

Description of test	Test data to be used (if required)	Expected outcome	Actual outcome	Comments and intended actions
1. The website will load when I run it		The websites account registration page loads	Nothing could be found	To resolve this issue, I realised that templates wasn't added to my DIRS on the settings page
2. When signup is clicked the system will take the user		The system loads the signup page	The URL couldn't be found	I resolved this after getting rid of the signup button and realized my URL for signup was spelt wrong when

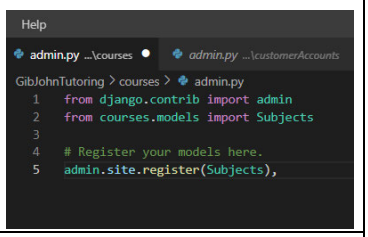
to the sign-up page				implementing it to the teacher signup

3. When any page loads the CSS will be implemented onto the webpage		The webpage will have the CSS implemented onto it	the CSS doesn't appear	I resolved this issue by going into the settings and
				`STATICFILES_DIRS = (`
				`os.path.join(BASE_DIR, "assets"),`
				including
				This allowed the system to find my assets folder that held the CSS so it could be loaded onto the page

4. When the reveal password eye is clicked the password is revealed		The password is revealed	The password gets revealed	
5. When any page loads the bootstrap will be implemented onto the webpage		The webpage will have the bootstrap implemented	The bootstrap appears and works	
6. When I click the teacher form the form will load		The system loads the form	Nothing appears	I resolved this at the same time as test number 2 as I realized that there was no {% block content %} on the html

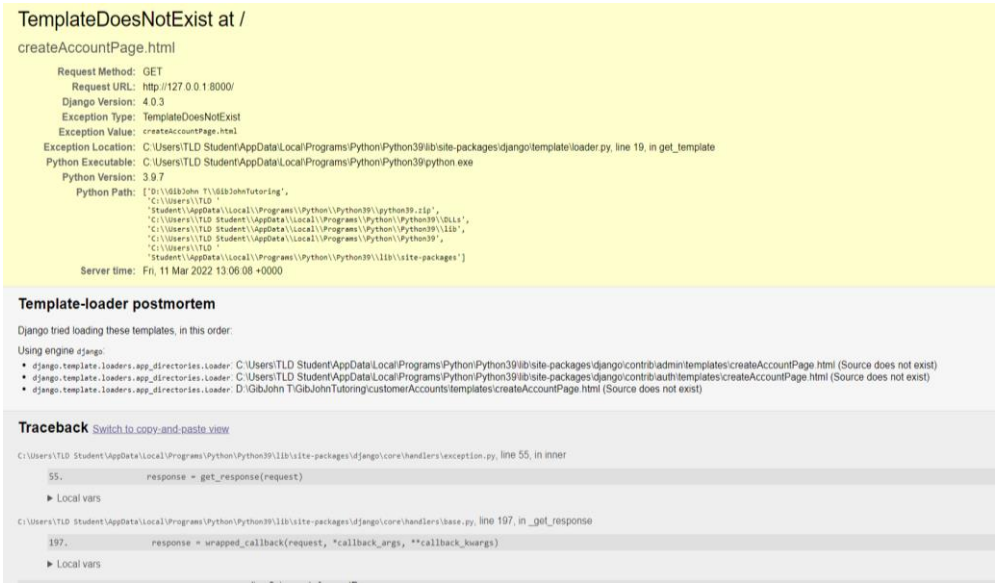
7. Making sure the teacher homepage will load later when I've set up the form properly		The webpage loads	There's no template	To resolve this, I realized that I hadn't added the teacherHome to the apps on my settings and so added that along with the other apps that had yet to be added
				 
				`INSTALLED_APPS = [`
				`'django.contrib.admin',`
				`'django.contrib.auth',`
				`'django.contrib.contenttypes',`
				`'django.contrib.sessions',`
				`'django.contrib.messages',`
				`'django.contrib.staticfiles',`
				`#my apps`
				`'customerAccounts',`
				`'courses',`
				`'studentHome',`
				`'teacherHome',`

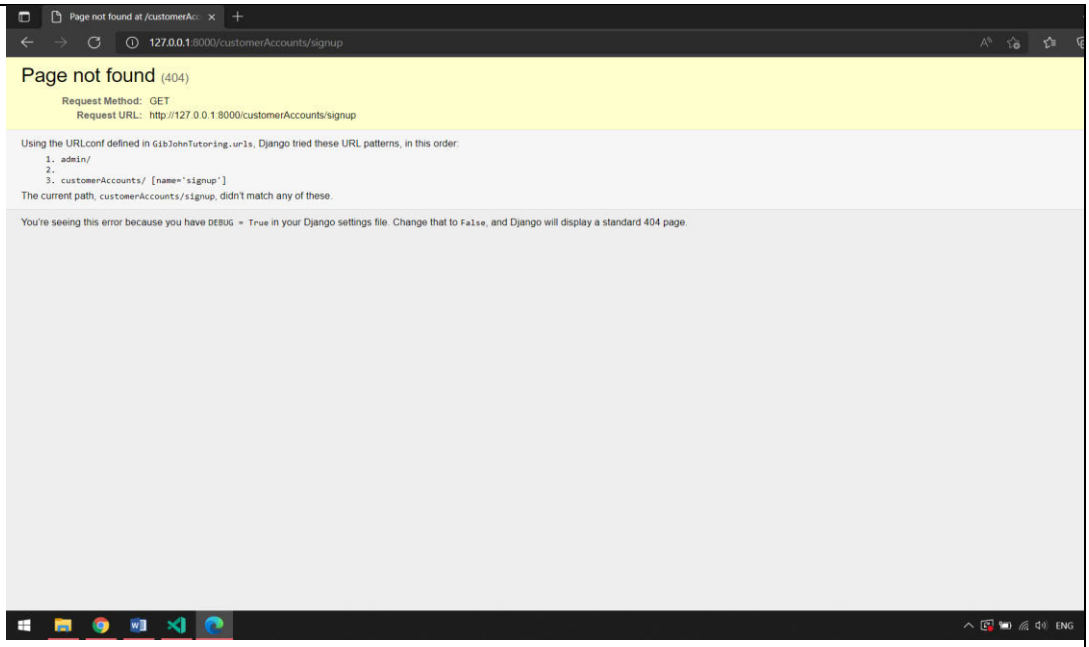
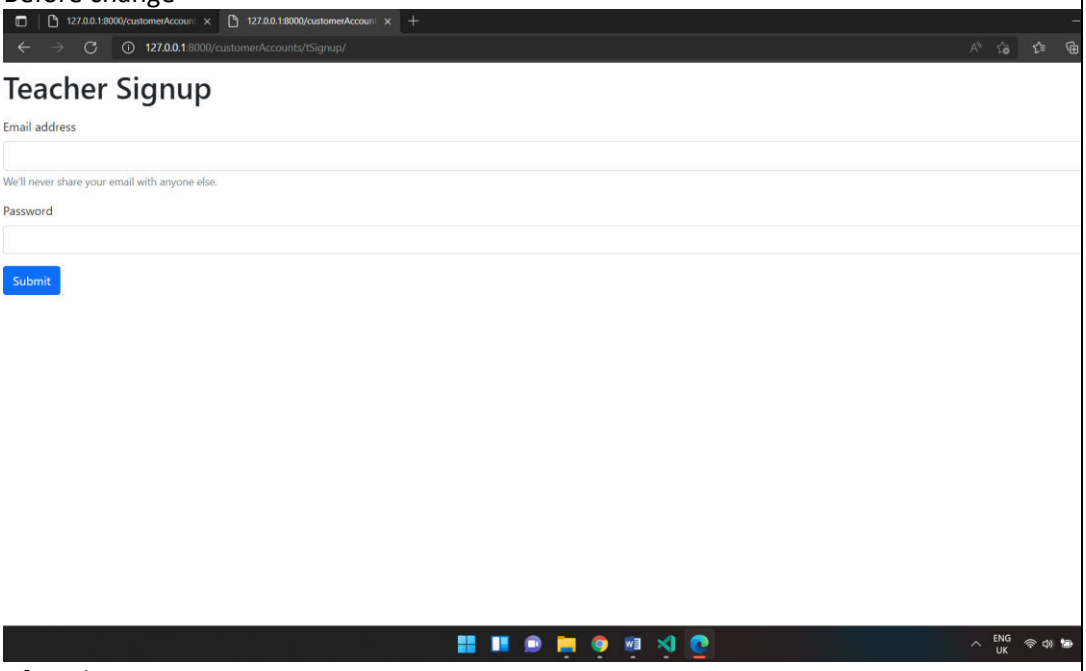
8. Making sure the student homepage will load later when I've set		The webpage loads	The templates couldn't be found	To resolve this I realized that the folder the html was in wasn't labeled as studentHome and so the system couldn't find it

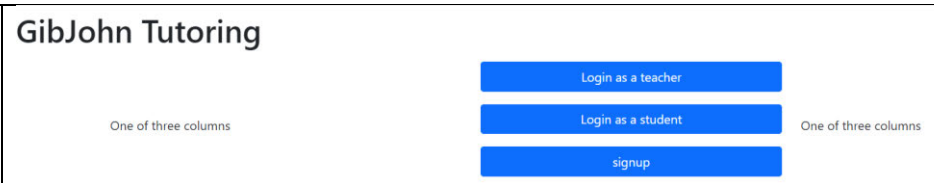
	up the form properly				
9.	When a current course is clicked on the student page that course will load		User is taken to the correct course	The link works but due to no course pages existing yet then nothing loads	
10.	When the courses page is clicked on the navbar the course page will load		The webpage loads	The page couldn't be found	(insert comments when this is resolved)
11.	When the filter button is pressed on the courses page the system acknowledges which subject was selected		Text appears saying which filter was selected	The text says which filter was selected	
12.	Does the student/teacher signup form sign the user in when valid inputs are used	Username[redacted] 21 Firstname: [redacted] Lastname:[redacted] [redacted] Email:example@email.com Password:12345678	The users account is saved to the system	An error occurs	(insert comments when this is resolved)
13.	When I load the admin page to check if the courses and user models are there they will show up		The models show up	Only the user models appeared	To resolve this I realized that I hadn't registered the courses model in the admin.py 
14.	A new course gets added to the database when its saved		The course is added to the database	An error occurs	To resolve this I realized that I forgot to migrate the models when I changed the

	on the admin site				name from subjects to courseinfo
15.	When the button on the table in the courses page is clicked it'll take the user to the appropriate course		The correct course is loaded	The link works but due to no course pages existing yet then nothing loads	(insert comments when this is resolved)

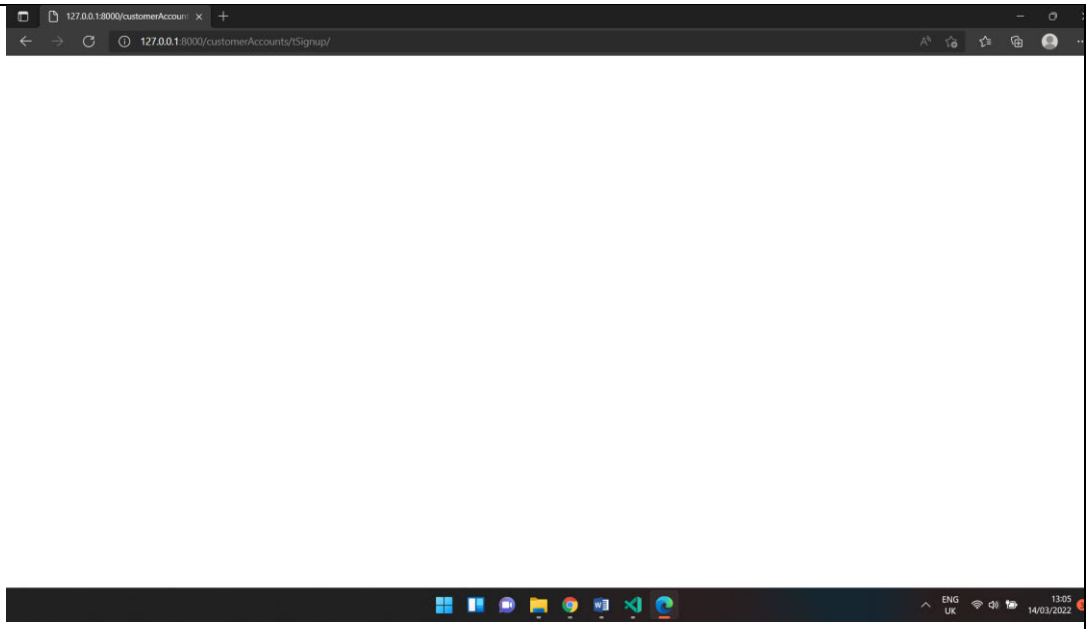
Evidence of broken code and then the result after I fixed it:

1.	 Before change  After change

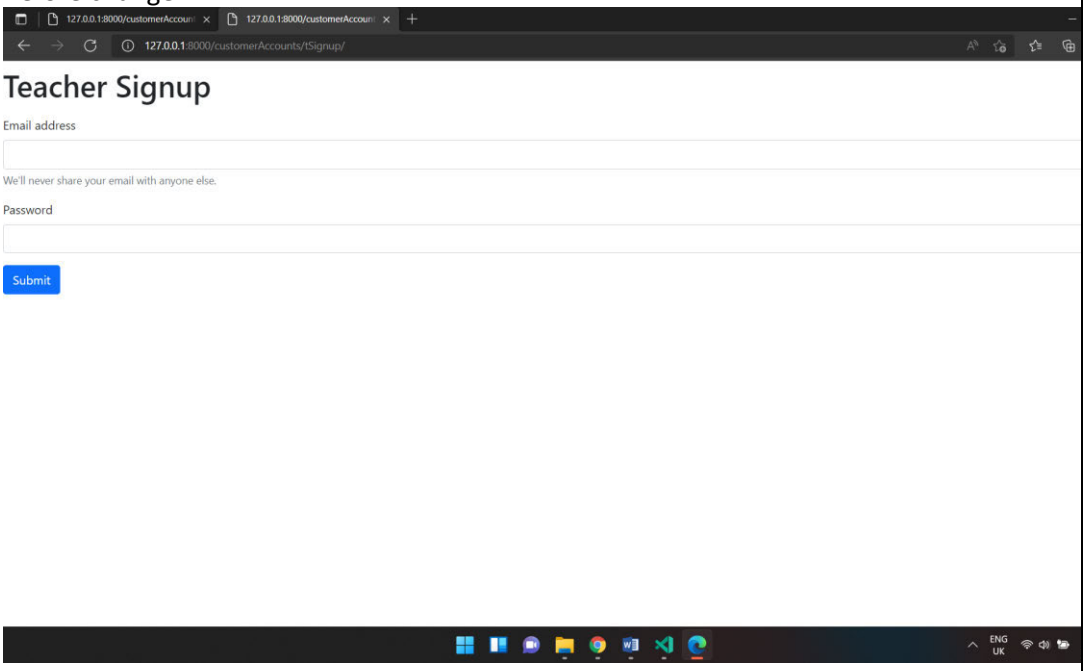
2.	 Before change  After change
3.	

	 The title is in a different font than it usually is because of css
4.	
5.	The buttons for the bootstrap are showing 

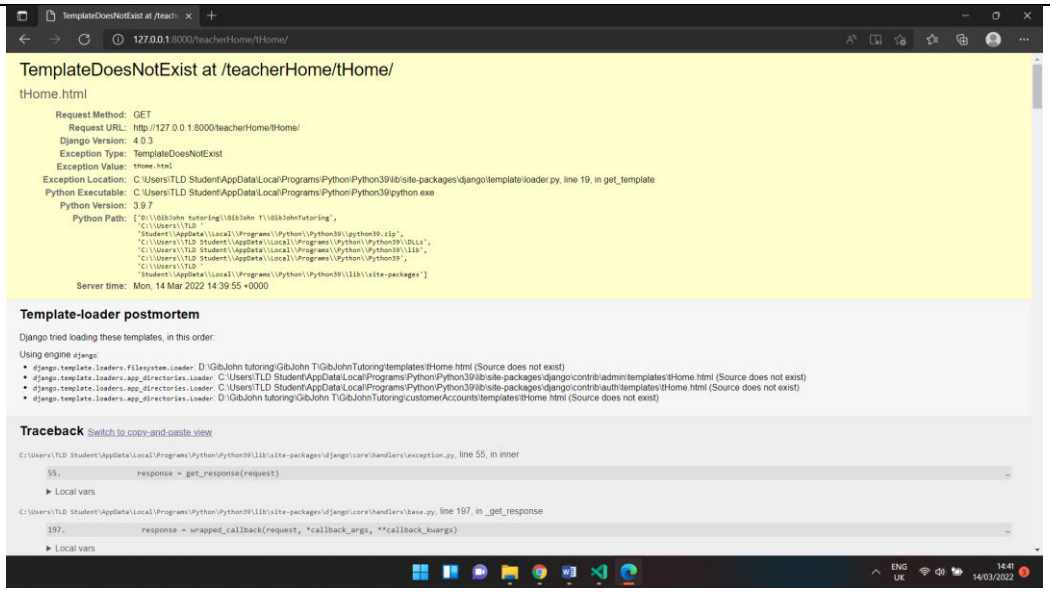
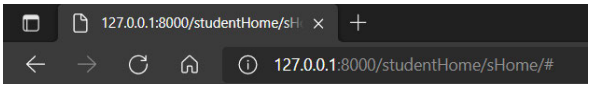
6.

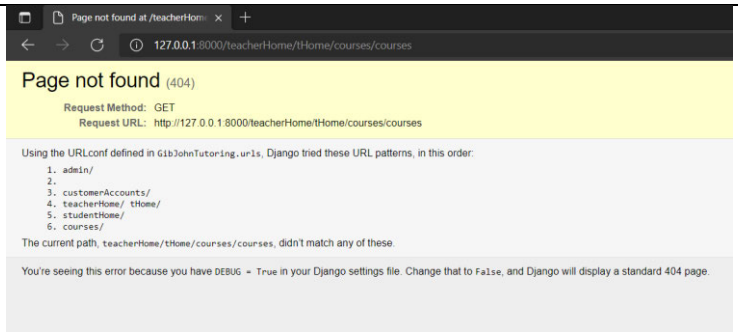


Before change



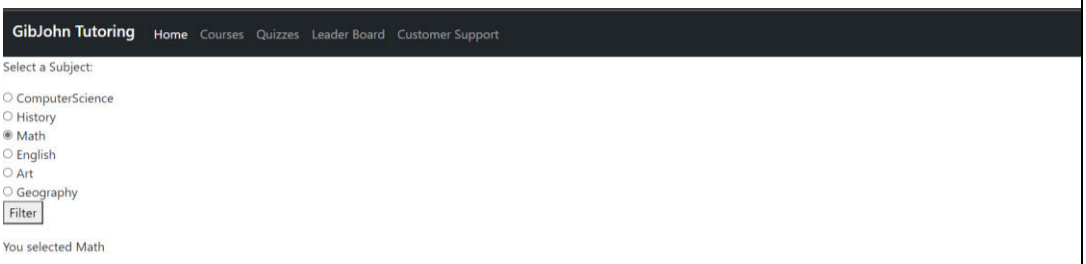
After change

7.	 Before change  After change
8.	 What I changed  After change
9.	 The hashtag got added onto the end when I clicked on a course showing that the link does work once there is one.
10.	

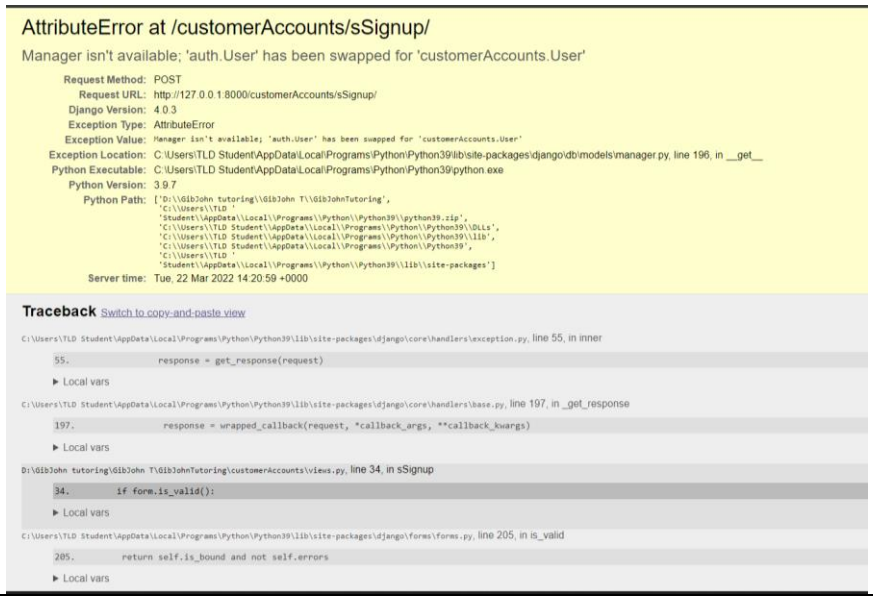


I know the problem for this is that the courses link gets appended onto the end of the url instead of restting the url but I'm currently unaware of how to fix this. To bypass this for now I'm making a different link to the courses page on the signup page

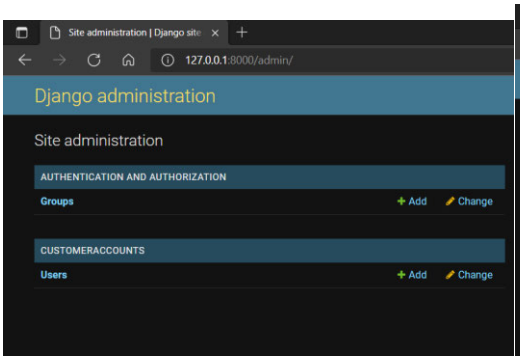
11.

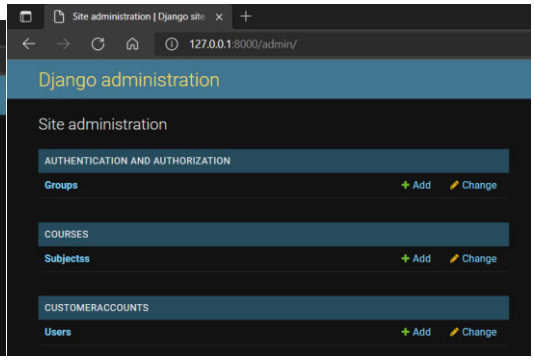


12.



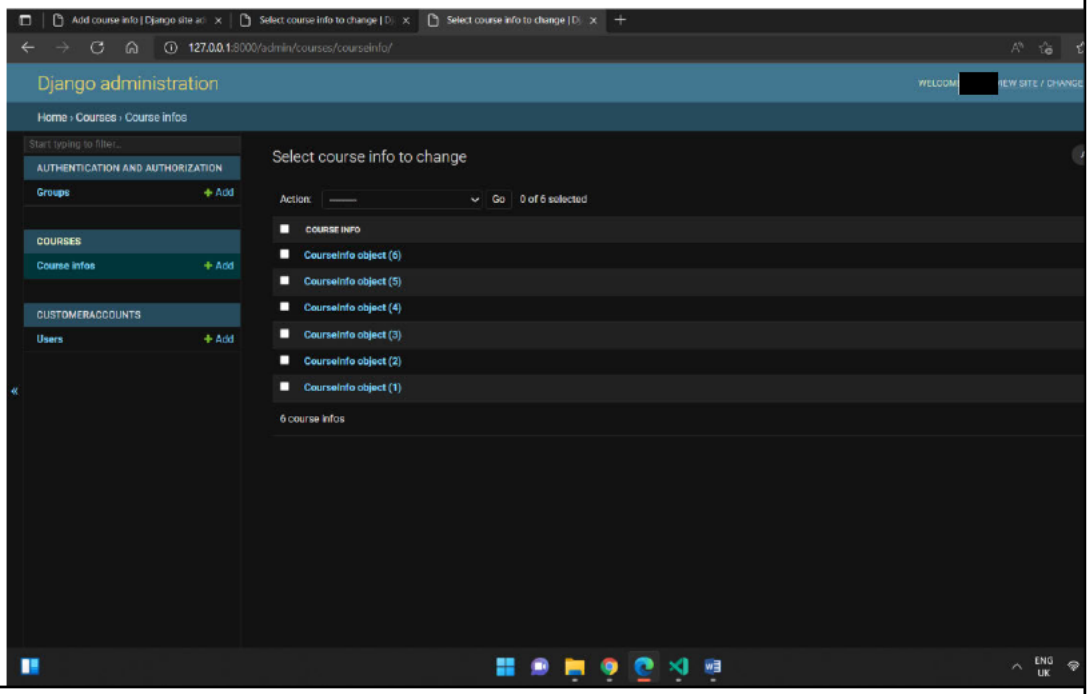
13.





Before change

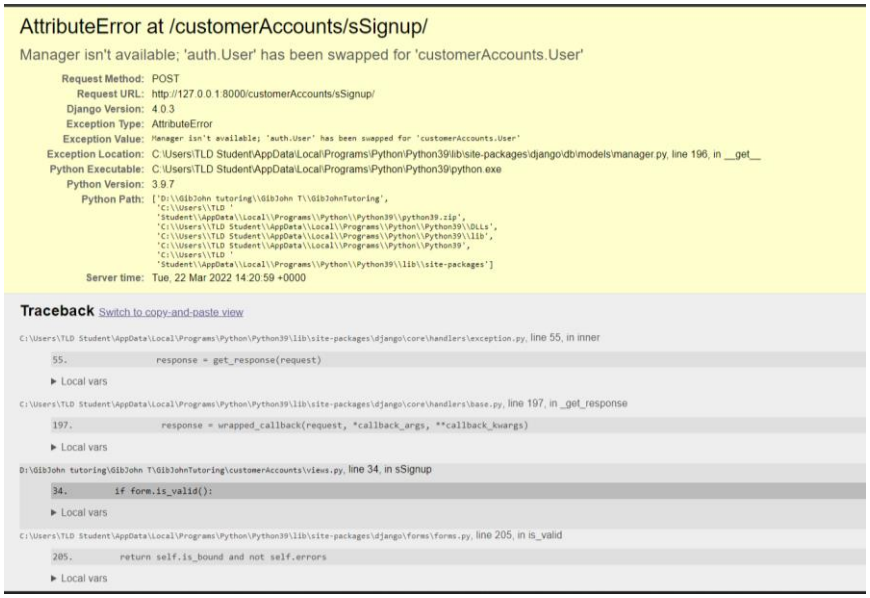
after change

14.	Didn't get a screenshot of before the migrations were made but now I can make courses within my database 
15.	

Trying to get the accounts app to work.

During my project (As you will see below in the website code section) I ended up changing my customer accounts quite a bit due to the custom accounts not working when I tried to implement them. Directly below Is the process I took when trying to get these custom accounts to work and then eventually switching back to thee built in django ones

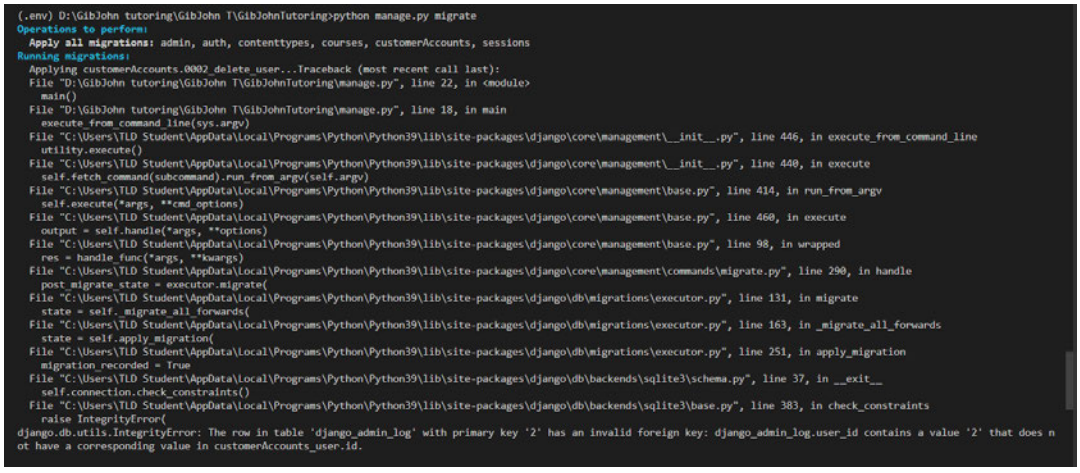
Due to my accounts app not functioning how it should have.



(I would keep getting this error)

I decided to try and simplify what was happening by getting rid of the user model and just using Django's built in user creation form so that the website could at least have the ability to log users in.

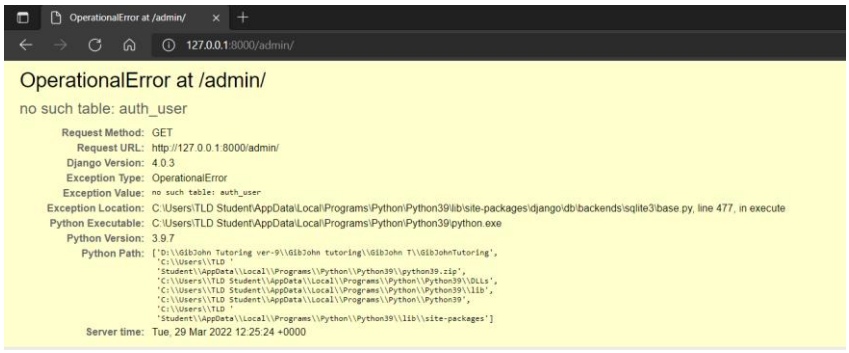
However in the process of trying to remove the model through migrations I ended up getting this error



Because of this I wasn't able to remove the model meaning that my user accounts continued to not work and I also couldn't access my admin. Since I'm unaware of how to fix this I then decided to go back to a previous copy of my website and try again.

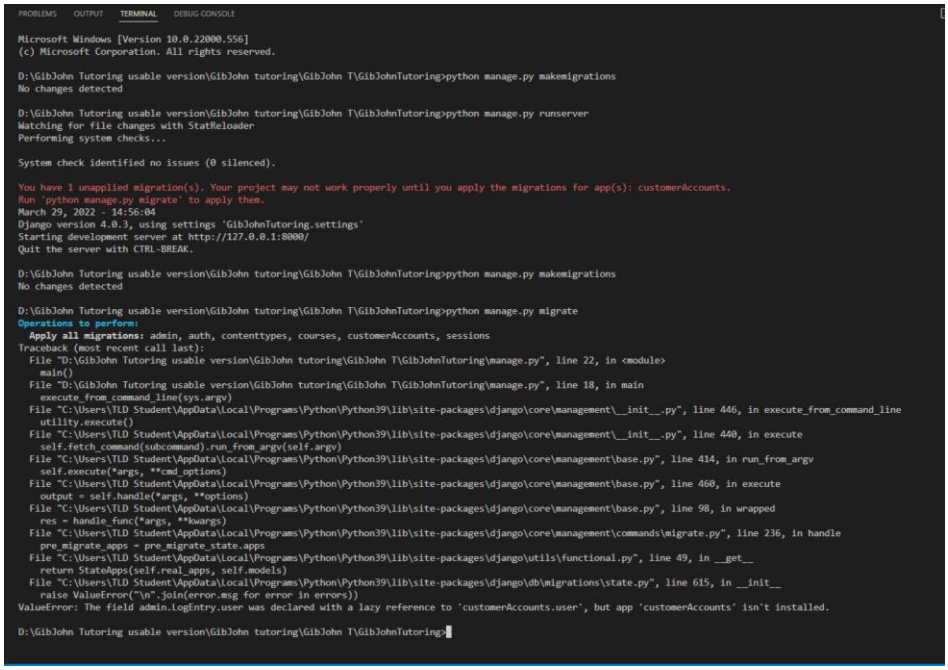
This time before changing around any of my other code like the forms.py and views.py I deleted the user that was created on the model and then removed the model with migrations which succesfully allowed me to remove the model from the database.

However this still lead to the problem of not being able to reach my admin



After an hour of trying to find the solution I have decided to go back to what I orignally had due to time constraints as that at least lets me manually set up a user.

When trying to go back to the original verision I got a new error

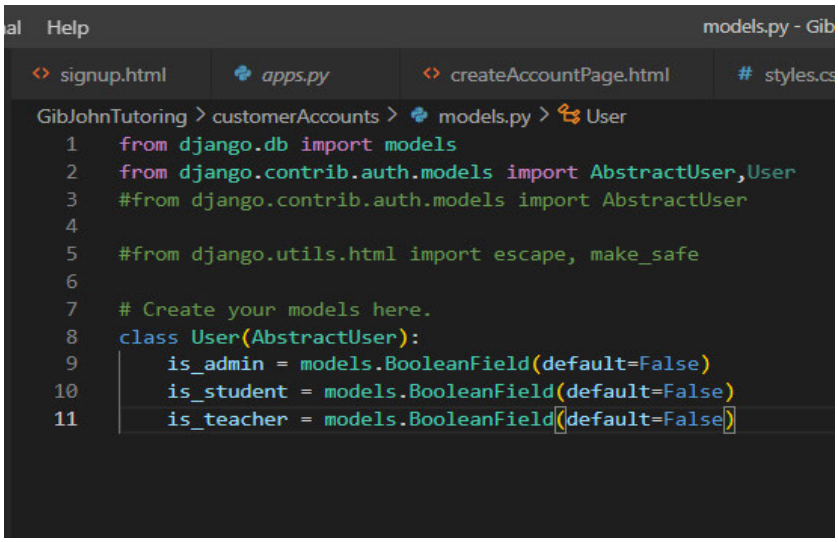


After having some time to go back over getting an acocunts app to work I tried deleting all the migrations and removing my custom user models. I tehn tried to create a super user for the new user acocunts and I once again got the error of “no such table for auth_user.”

Website code:

customerAccounts:

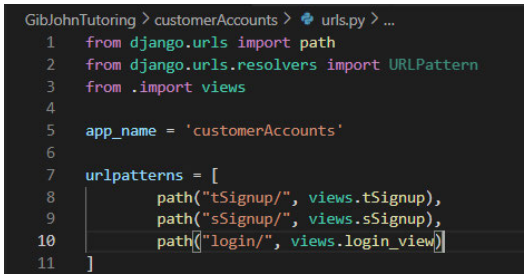
Models.py



These are the models for the customer accounts that will allow me to create multiple user types when users log in so that the system can easily differentiate what each user should and shouldn't be able to see e.g. only teachers should have the option to see details on student progress or request a new course.

(In the end these weren't used)

Urls.py



These are the URLs for the customerAccounts this will link the signup page for teachers and students to the webpage the user is viewing through the view.py

Forms.py

This the form for the teacher signup (the student signup is the same)

GibJohnTutoring > customerAccounts > forms.py > teacherForm > save

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class teacherForm(UserCreationForm):
    first_name = forms.CharField(max_length=30,)
    last_name = forms.CharField(max_length=30,)
    email = forms.EmailField(max_length=254, help_text='Required. Inform a valid email address.')

    def save(self, commit=True):
        user = super().save(commit=False)
        user.is_teacher = True
        if commit:
            user.save()
        return user

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', )
```

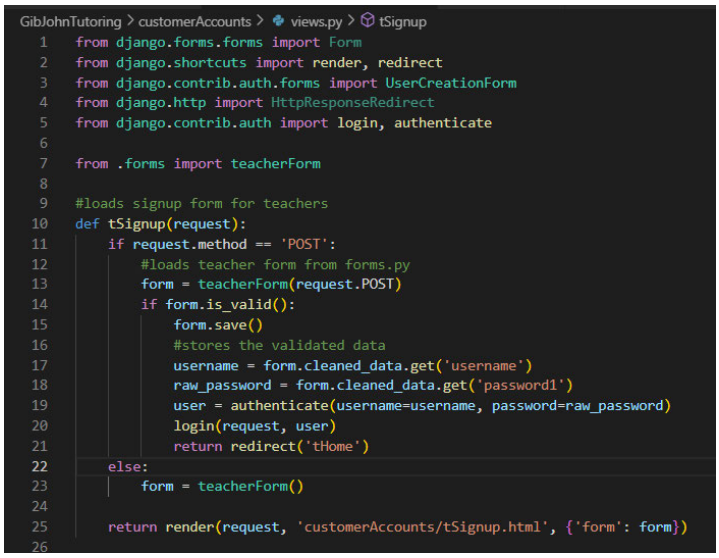
This saves the user as a teacher within the database

This defines what order the inputs on the form will appear so that it follows with common practice of registration forms improving user experience

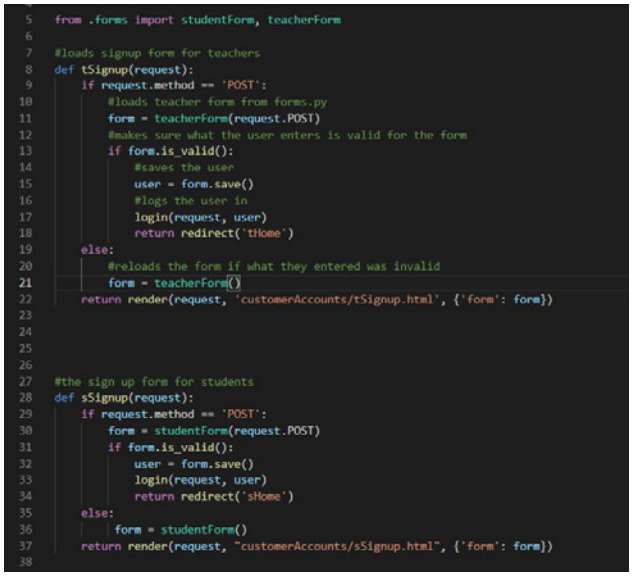
I used the built in UserCreationForm from Django and then added in inputs for email, first name and last name. the first and last name were added as it is typical to include those on forms and the email is there so that a confirmation email can be sent to the user.

Views.py

This is the views for my customer accounts. (The view that loads the students form looks the same as this)




```python
from django.forms.forms import Form
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm
from django.http import HttpResponseRedirect
from django.contrib.auth import login, authenticate

from .forms import teacherForm

#loads signup form for teachers
def tSignup(request):
    if request.method == 'POST':
        #loads teacher form from forms.py
        form = teacherForm(request.POST)
        if form.is_valid():
            form.save()
            #stores the validated data
            username = form.cleaned_data.get('username')
            raw_password = form.cleaned_data.get('password1')
            user = authenticate(username=username, password=raw_password)
            login(request, user)
            return redirect('tHome')
    else:
        form = teacherForm()

    return render(request, 'customerAccounts/tSignup.html', {'form': form})
```



```python
from .forms import studentForm, teacherForm

#loads signup form for teachers
def tSignup(request):
    if request.method == 'POST':
        #loads teacher form from forms.py
        form = teacherForm(request.POST)
        #makes sure what the user enters is valid for the form
        if form.is_valid():
            #saves the user
            user = form.save()
            #logs the user in
            login(request, user)
            return redirect('tHome')
    else:
        #reloads the form if what they entered was invalid
        form = teacherForm()
    return render(request, 'customerAccounts/tSignup.html', {'form': form})



#the sign up form for students
def sSignup(request):
    if request.method == 'POST':
        form = studentForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('sHome')
    else:
        form = studentForm()
    return render(request, "customerAccounts/sSignup.html", {'form': form})
```

This is what the student and teacher views ended up looking like when I tried to simplify the process

```python
def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            #gets the username and password
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            #authenticates the user making sure their credentials are correct
            user = authenticate(username=username, password=password)
            if user is not None:
                #logs the user in and directs them to the home page
                login(request, user)
                return redirect('homepage.html')
    form = AuthenticationForm()
    return render(request, "customerAccounts/login.html",{"login_form": form})

def logout_view(request):
    if request.method == 'POST':
        logout(request)
        return redirect('')
```

I then also ended up making a login and logout view

Courses:

```python
from re import M
from django.db import models

#Create your models here.
class CourseInfo(models.Model):
    COURSE_SUBJECT = (
        ('art','art'),
        ('computer science','computer science'),
        ('geography', 'geogrpahy'),
        ('english', 'english'),
        ('math', 'math'),
        ('history', 'history'),
    )
    GRADE_LEVEL = (
        ('1','1'),
        ('2','2'),
        ('3','3'),
        ('4','4'),
        ('5','5'),
        ('6','6'),
        ('7','7'),
        ('8','8'),
        ('9','9'),
    )
    EXAM_BOARD = (
        ('AQA','AQA'),
        ('pearson','pearson'),
    )
    CourseSubject = models.CharField(max_length=100, choices=COURSE_SUBJECT )
    ExamBoard = models.CharField(max_length=100, choices=EXAM_BOARD)
    GradeLevel = models.CharField(max_length=100, choices=GRADE_LEVEL)
```

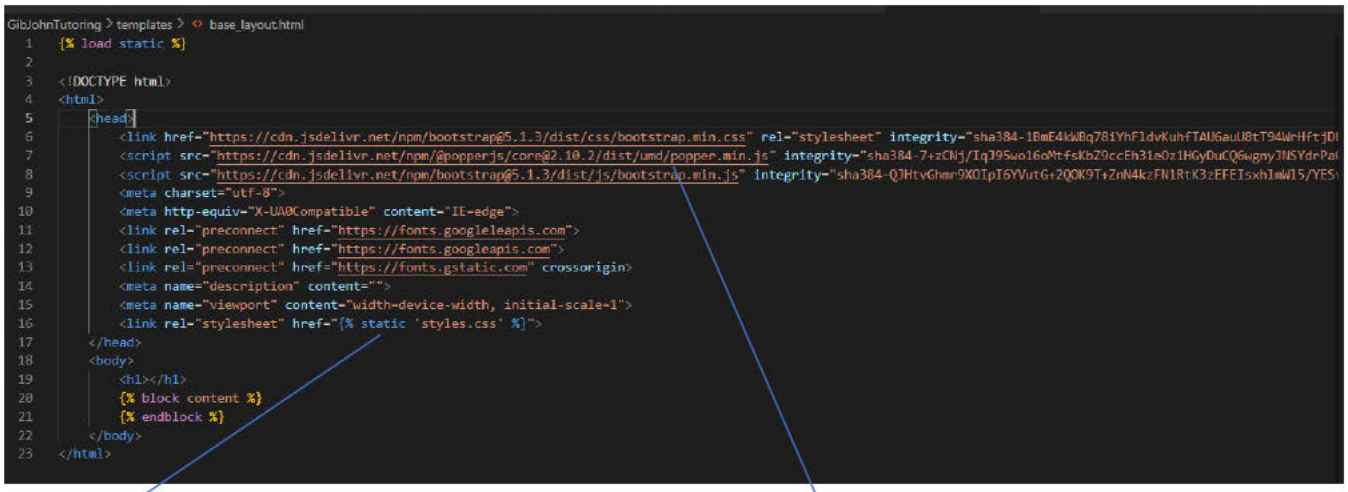
This is a list of course subject options for the system to register so courses can be labelled appropriately when they are created

Same goes for the grade level and exam board

This makes it so when a new course is being added to the database the user will select an option from each of the models

Html:

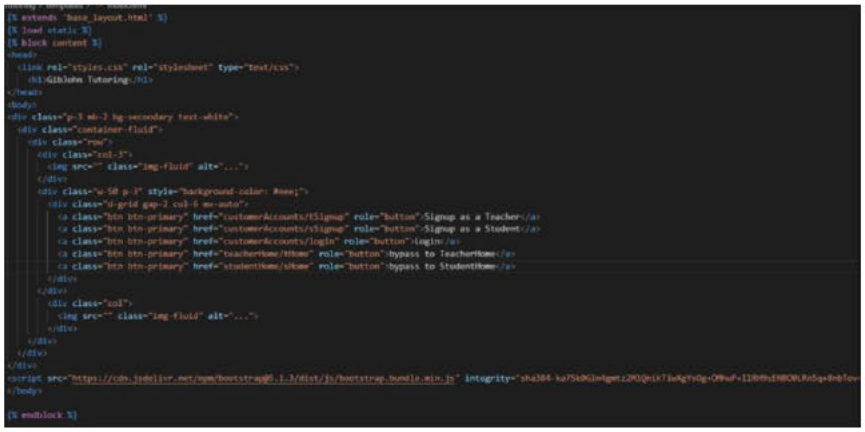
Base layout:



Links the stylesheet to the rest of the html pages so that I can implement my CSS

These 3 lines link the bootstrap to my website so that I can implement them onto any page I need to

Index.html

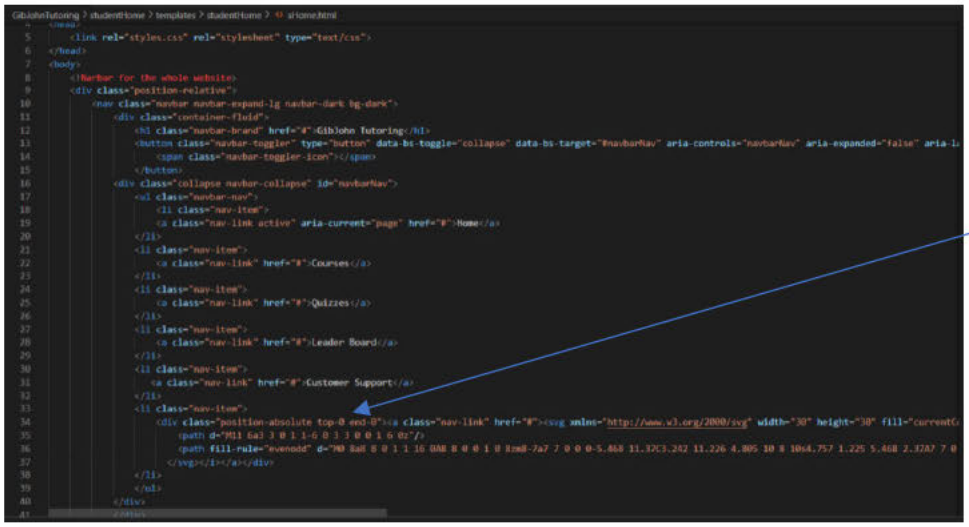


This is the html for the index page (the first page the user will see when loading the website) there's the links to each of the singup options and in the complete version there wouldn't be the links to the teacher and student pages they're just there for convenience sake.

There's also two image placeholders that would be filled with an educational looking picture that would cover each side of the page.

NavBar html

This is the html for the navbar that I'll be using throughout all of the website the links to the various pages will be added as and when I make each page



This is used to place the pfp at the end of the navbar

coursePage.html

```
<body>
    <p>Select a Subject:</p>
    <div>
        <input type="radio" name="subject" value="ComputerScience" id="ComputerScience">
        <label for="ComputerScience">ComputerScience</label>
    </div>
    <div>
        <input type="radio" name="subject" value="History" id="History">
        <label for="History">History</label>
    </div>
    <div>
        <input type="radio" name="subject" value="Math" id="Math">
        <label for="Math">Math</label>
    </div>
    <div>
        <input type="radio" name="subject" value="English" id="English">
        <label for="English">English</label>
    </div>
    <div>
        <input type="radio" name="subject" value="Art" id="Art">
        <label for="Art">Art</label>
    </div>
    <div>
        <input type="radio" name="subject" value="Geography" id="Geography">
        <label for="Geography">Geography</label>
    </div>
    <p>
        <button id="btn">Filter</button>
    </p>

    <p id="output"></p>
```

for the courses page I used JavaScript to implement radio buttons that will filter the page to show the courses the user has selected.

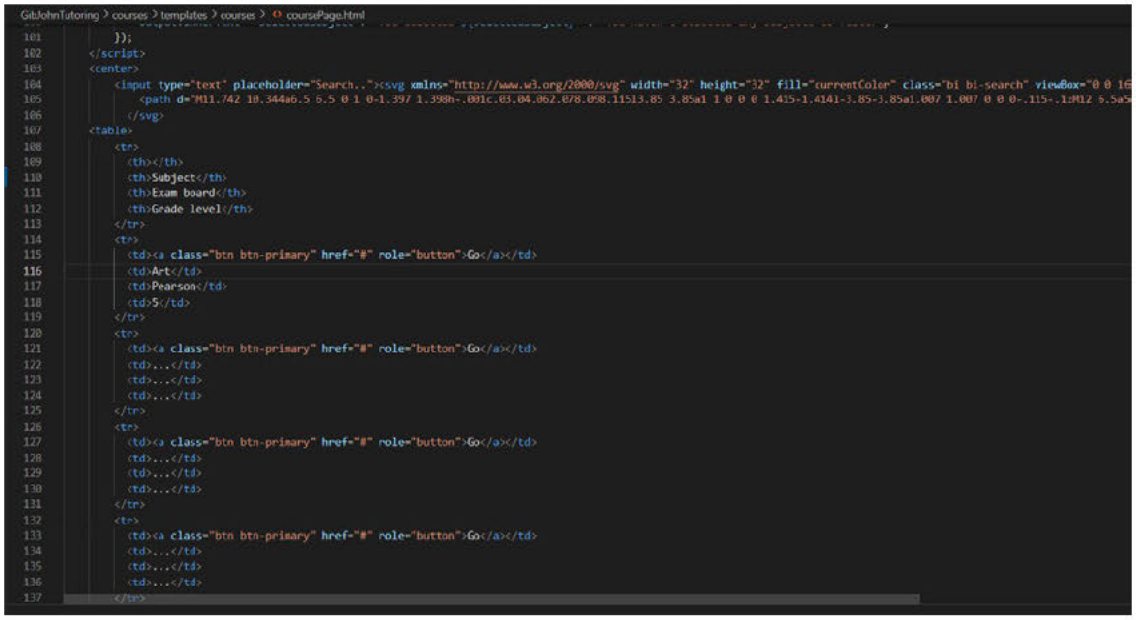
```
<script>
    const radioButtons = document.querySelectorAll('input[name="subject"]');
    btn.addEventListener("click", () => {
        let selectedSubject;
        for (const radioButton of radioButtons) {
            if (radioButton.checked) {
                selectedSubject = radioButton.value;
                break;
            }
        }
        // show the output:
        output.innerText = selectedSubject ? `You selected ${selectedSubject}` : `You haven't selected any subjects to filter`;
    });
</script>
</body>
</html>
</body>
{% endblock %}
```

This will output what is stored In Selectedsubject (which for now is text so I can see that it works) or it will tell the user nothing was selected

This will select all the buttons with the name Subject and store them to radioButtons

There is then an eventlistener that will check for when the button labelled btn is clicked and will run the code beneath it when it does

This will take the selected raidoButton and store it to the selectedSubject and as to only one button needs to be pressed, to stop the program from looping, there's an immediate break

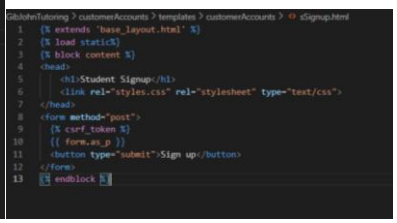


```
GibJohnTutoring > courses > templates > courses > coursePage.html
    });
</script>
<center>
    <input type="text" placeholder="Search..."><svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" fill="currentColor" class="bi bi-search" viewBox="0 0 16
        <path d="M11.742 10.344a6.5 6.5 0 1 0-1.397 1.398h-.001c.03.04.062.078.098.115l3.85a1 1 0 0 0 1.415-1.414l-3.85-3.85a1.007 1.007 0 0 0-.115-.1zM12 6.5a5
    </svg>
<table>
    <tr>
      <th></th>
      <th>Subject</th>
      <th>Exam board</th>
      <th>Grade level</th>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>Art</td>
      <td>Pearson</td>
      <td>5</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
```

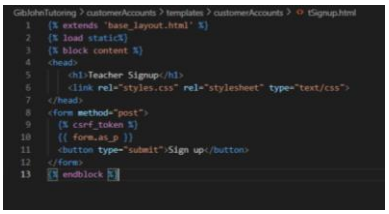
I used a temporary table in case I wasn't able to finish the interactive table using the filters as well as made a search box above the table.

Login.html/sSignup/tSignup

```
<!doctype html>
{% extends 'base_layout.html' %}
{% load static %}
{% block content %}
  <head>
      <h1>Login</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div class="accounts-form">
      <form method="POST">
        {% csrf_token %}
        {{ login_form }}
        <button class="btn btn-primary" type="submit">Login</button>
      </form>
    </div>
  </body>
</html>
{% endblock %}
```



```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
<head>
    <h1>Student Signup</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
</head>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign up</button>
</form>
{% endblock %}
```

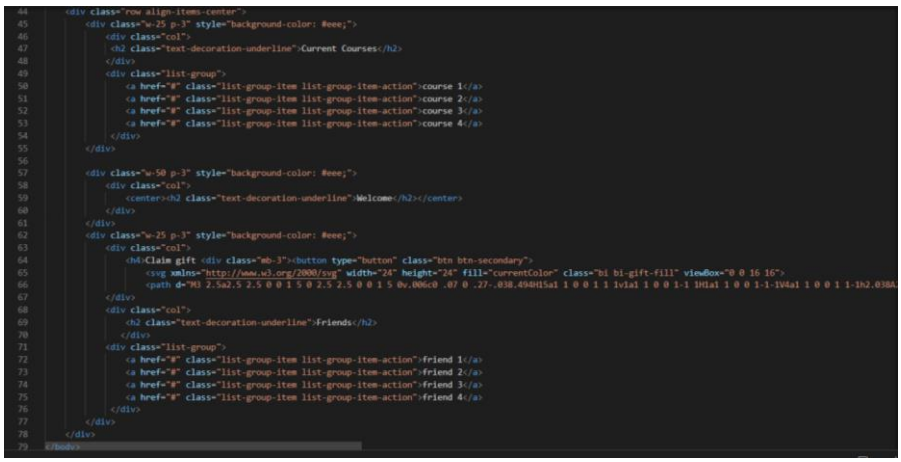


```
{% extends 'base_layout.html' %}
{% block content %}
<head>
    <h1>Teacher Signup</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
</head>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign up</button>
</form>
{% endblock %}
```

This is the html for the login and signup pages. It's very basic as I wasn't able to get around to developing the CSS/frontend of the website much.

I've made sure to use csrf token for security.

sHomepage.html



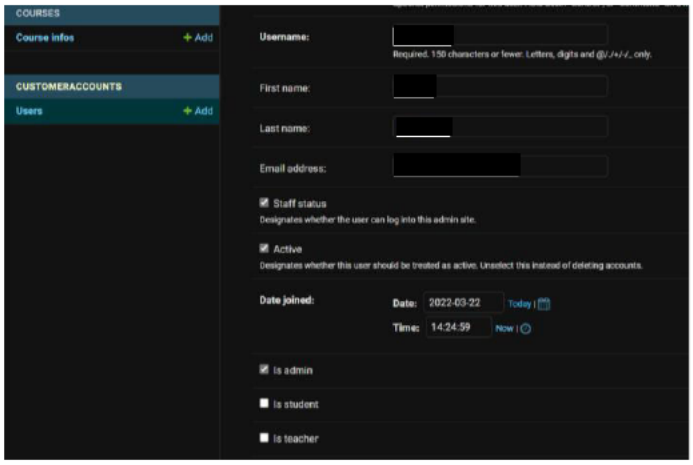
```
<div class="row align-items-center">
    <div class="w-25 p-3" style="background-color: #eee;">
        <div class="col">
          <h2 class="text-decoration-underline">Current Courses</h2>
        </div>
        <div class="list-group">
            <a href="#" class="list-group-item list-group-item-action">course 1</a>
            <a href="#" class="list-group-item list-group-item-action">course 2</a>
            <a href="#" class="list-group-item list-group-item-action">course 3</a>
            <a href="#" class="list-group-item list-group-item-action">course 4</a>
        </div>
    </div>

    <div class="w-50 p-3" style="background-color: #eee;">
        <div class="col">
            <center><h2 class="text-decoration-underline">Welcome</h2></center>
        </div>
    </div>
    <div class="w-25 p-3" style="background-color: #eee;">
        <div class="col">
            <h4>Claim gift <div class="mb-3"><button type="button" class="btn btn-secondary">
                <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="currentColor" class="bi bi-gift-fill" viewBox="0 0 16 16">
                <path d="M3 2.5a2.5 2.5 0 0 1 5 0 2.5 2.5 0 0 1 5 0v.006c0 .07 0 .27-.038.494H15a1 1 0 0 1 1 1v1a1 1 0 0 1-1 1H1a1 1 0 0 1-1-1V4a1 1 0 0 1 1-1h2.038A
        </div>
        <div class="col">
            <h2 class="text-decoration-underline">Friends</h2>
          </div>
        <div class="list-group">
            <a href="#" class="list-group-item list-group-item-action">Friend 1</a>
            <a href="#" class="list-group-item list-group-item-action">Friend 2</a>
            <a href="#" class="list-group-item list-group-item-action">Friend 3</a>
            <a href="#" class="list-group-item list-group-item-action">Friend 4</a>
        </div>
    </div>
</div>
</body>
```

including the navbar html this is html for the student homepage.

I used bootstrap to split the page into 3 sections as well as create 2 list groups for the friends and courses. There's also the claim gift button that uses an asset from bootstraps icons.

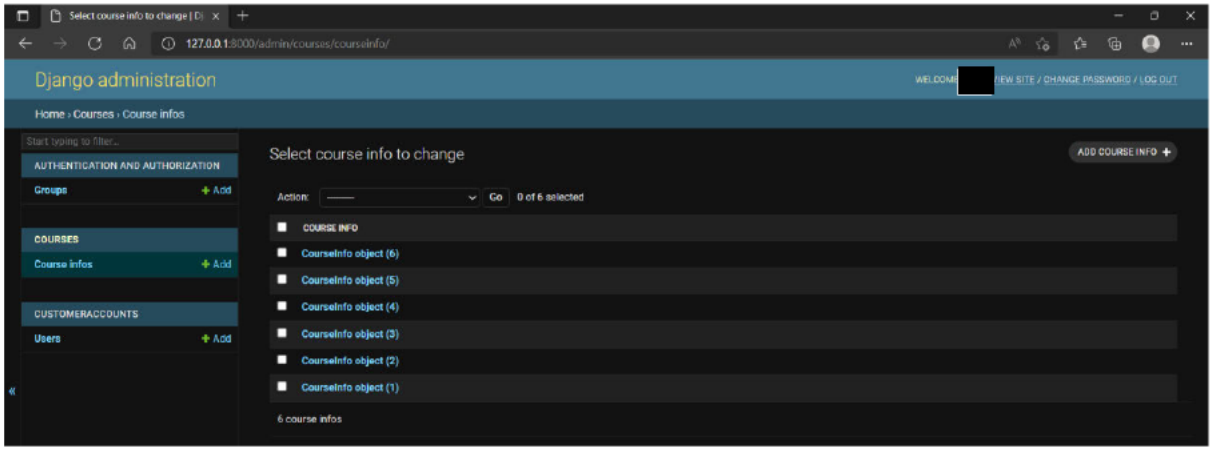
Admin progression

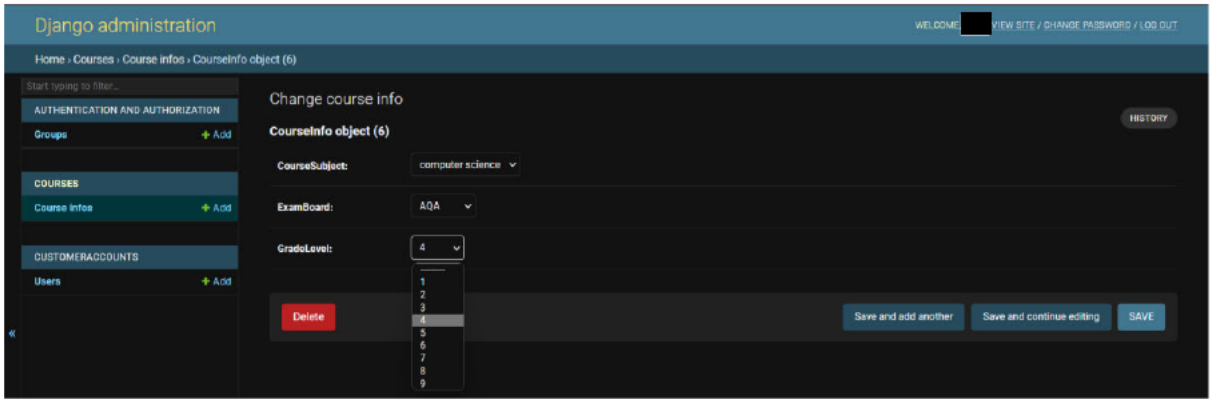
Users



In my user accounts I'm able to see the users Username, first name, last name, email and password. I can then also given them the student, teacher or admin status that will give the users access to different things and includes various features on the website depending on what their status is.

Courses

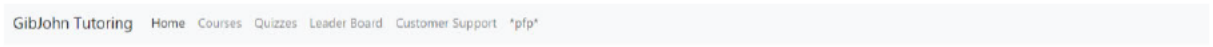




This is the courses admin, there's currently 6 courses created and when getting added the system will record the subject, exam board and level of the course.

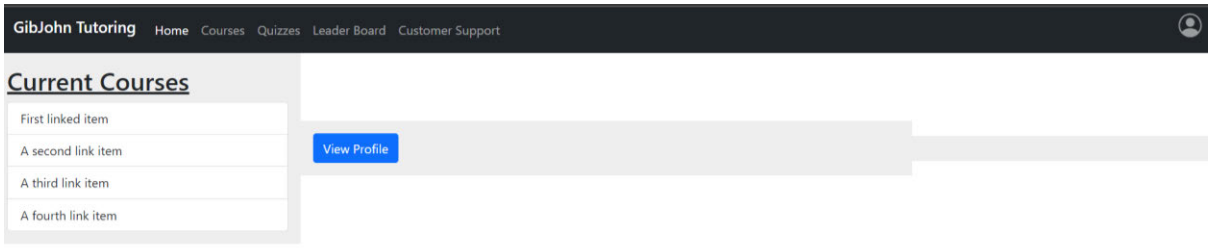
Front end progression

Student homepage

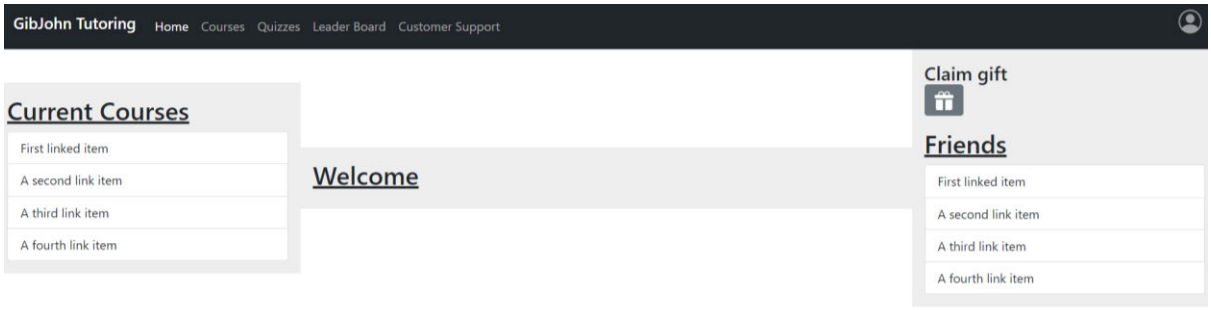


this is the first instance of my student homepage where I was setting up the different links for the navbar

I then figured out how to move what will be the profile picture to the other side of the navbar and also make the bar black so that it's more distinct



Next, I got the profile picture sorted out and split the page into 3 sections like in my plan. The current courses will later go down the whole screen and have links to the courses, I'm currently trying to find a way to make the view profile appear in the middle of the screen.



The student page now looks a little more like how I had it on the plan as there is now a friend section a claim gift button and the current courses. However, I can't figure out how to position everything into the correct spot.

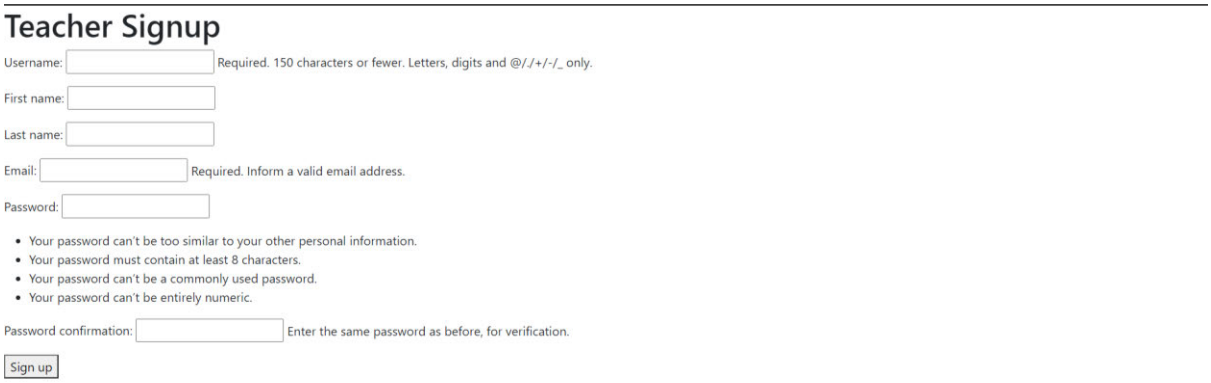
Teacher homepage

Never got around to being able to do anything on the teacher homepage

Signup pages

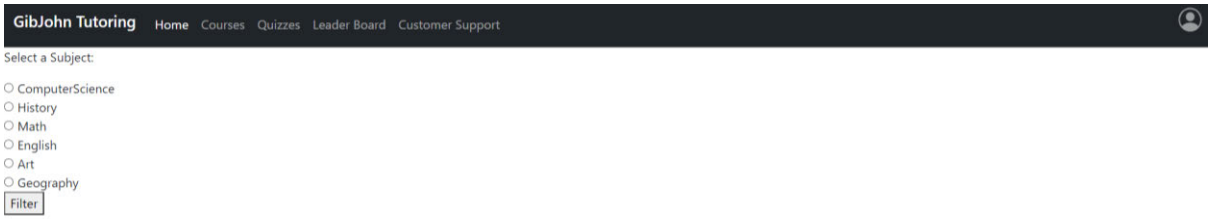
this is the starting point of the sign-up form where I was testing if the form html would work

next I want to add in an option to enter a username, birthday and a confirm password box.

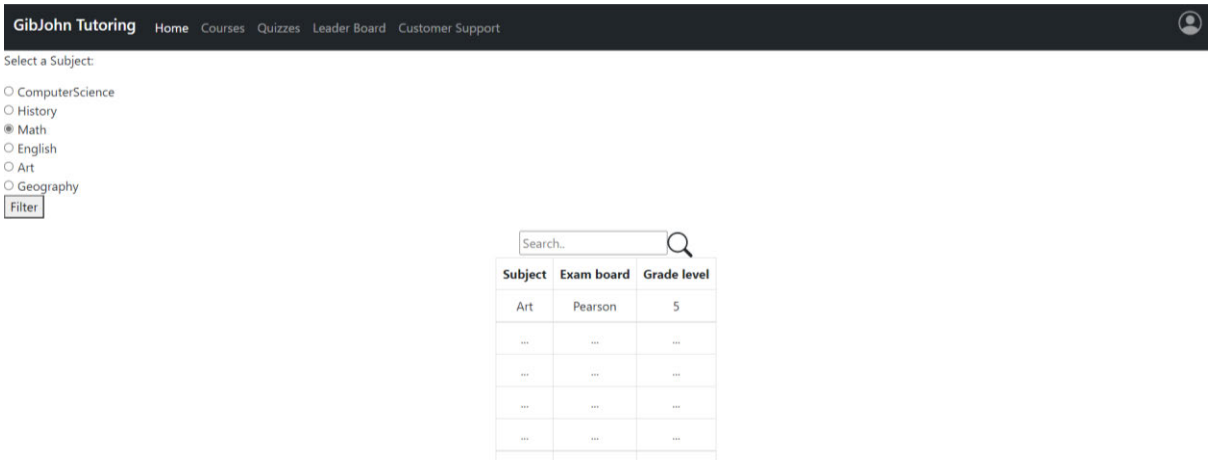


Finished sign up form

Courses page



Currently only have the filter system on the courses page but will later be adding the courses list itself with a scroll bar on it.



I then added a table that will display each course with a search bar on the top to allow users to search for something specifically

I've now made the table easier to see with the white background as well as added a button to each course that will take the user to that course.

In the complete version of this there would be many more courses that can make use of the filter system

courses/admin.py

```
from django.contrib import admin
from courses.models import CourseInfo

# Register your models here.
admin.site.register(CourseInfo),
```

---

courses/forms.py

---

courses/models.py

```
from re import M
from django.db import models

#Create your models here.
class CourseInfo(models.Model):
    COURSE_SUBJECT = (
        ('art','art'),
        ('computer science','computer science'),
        ('geography', 'geogrpahy'),
        ('english', 'english'),
        ('math', 'math'),
        ('history', 'history'),
    )
```

```
    GRADE_LEVEL = (
        ('1','1'),
        ('2','2'),
        ('3','3'),
        ('4','4'),
        ('5','5'),
        ('6','6'),
        ('7','7'),
        ('8','8'),
        ('9','9'),
    )
    EXAM_BOARD = (
        ('AQA','AQA'),
        ('pearson','pearson'),
    )
    CourseSubject = models.CharField(max_length=100, choices=COURSE_SUBJECT )
    ExamBoard = models.CharField(max_length=100, choices=EXAM_BOARD)
    GradeLevel = models.CharField(max_length=100, choices=GRADE_LEVEL)
```

---

courses/urls.py

```
import imp
from django.urls import path
from django.urls.resolvers import URLPattern
from .import views


app_name = 'courses'
```

```
urlpatterns = [
    path("courses/", views.courses),
]
```

---

courses/views.py

```
from django.shortcuts import render

# Create your views here.
def courses(request):
    return render(request, "courses/coursePage.html")
```

---

courses/templates/courses/coursePage.html

```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
{% load render_table from django_tables2 %}
<html lang="en">
<head>
    <link rel="styles.css" rel="stylesheet" type="text/css">
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
    td, th {
```

```
        border:1px solid black;
        text-align: center;
        padding: 8px;
      }
</style>
</head>

<body>
   <!Narbar for the whole website>
   <div class="position-relative">
      <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
         <div class="container-fluid">
            <h1 class="navbar-brand" href="#">GibJohn Tutoring</h1>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
               <span class="navbar-toggler-icon"></span>
            </button>
         <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
               <li class="nav-item">
               <a class="nav-link active" aria-current="page" href="#">Home</a>
            </li>
            <li class="nav-item">
               <a class="nav-link" href="courses/courses">Courses</a>
            </li>
            <li class="nav-item">
               <a class="nav-link" href="#">Quizzes</a>
            </li>
            <li class="nav-item">
               <a class="nav-link" href="#">Leader Board</a>
            </li>
```

```
            <li class="nav-item">
              <a class="nav-link" href="#">Customer Support</a>
            </li>
            <li class="nav-item">
              <div class="position-absolute top-0 end-0"><a class="nav-link" href="#"><svg xmlns="http://www.w3.org/2000/svg" width="30" height="30" fill="currentColor" class="bi bi-person-circle" viewBox="0 0 16 16">
                  <path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>
                  <path fill-rule="evenodd" d="M0 8a8 8 0 1 1 16 0A8 8 0 0 1 0 8zm8-7a7 7 0 0 0-5.468 11.37C3.242 11.226 4.805 10 8 10s4.757 1.225 5.468 2.37A7 7 0 0 0 8 1z"/>
                </svg></i></a></div>
            </li>
            </ul>
          </div>
          </div>
        </nav>
    </div>

<body>
    <p>Select a Subject:</p>
    <div>
        <input type="radio" name="subject" value="ComputerScience" id="ComputerScience">
        <label for="ComputerScience">ComputerScience</label>
    </div>
    <div>
        <input type="radio" name="subject" value="History" id="History">
        <label for="History">History</label>
    </div>
    <div>
        <input type="radio" name="subject" value="Math" id="Math">
        <label for="Math">Math</label>
    </div>
```

```
  <div>
    <input type="radio" name="subject" value="English" id="English">
    <label for="English">English</label>
  </div>
  <div>
    <input type="radio" name="subject" value="Art" id="Art">
    <label for="Art">Art</label>
  </div>
  <div>
    <input type="radio" name="subject" value="Geography" id="Geography">
    <label for="Geography">Geography</label>
  </div>
  <p>
    <button id="btn">Filter</button>
  </p>

  <p id="output"></p>

  <script>
    const radioButtons = document.querySelectorAll('input[name="subject"]');
    btn.addEventListener("click", () => {
      let selectedSubject;
      for (const radioButton of radioButtons) {
        if (radioButton.checked) {
          selectedSubject = radioButton.value;
          break;
        }
      }
      // show the output:
      output.innerText = selectedSubject ? `You selected ${selectedSubject}` : `You haven't selected any subjects to filter`;
```

```
    });
  </script>
  <center>
    <input type="text" placeholder="Search.."><svg xmlns="http://www.w3.org/2000/svg" width="32" height="32" fill="currentColor" class="bi bi-search" viewBox="0 0 16 16">
      <path d="M11.742 10.344a6.5 6.5 0 1 0-1.397 1.398h-.001c.03.04.062.078.098.115l3.85 3.85a1 1 0 0 0 1.415-1.414l-3.85-3.85a1.007 1.007 0 0 0-.115-.1zM12 6.5a5.5 5.5 0 1 1-11 0 5.5 5.5 0 0 1 11 0z"/>
    </svg>
  <table>
    <tr>
      <th></th>
      <th>Subject</th>
      <th>Exam board</th>
      <th>Grade level</th>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>Art</td>
      <td>Pearson</td>
      <td>5</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
```

```
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
    <tr>
      <td><a class="btn btn-primary" href="#" role="button">Go</a></td>
      <td>...</td>
      <td>...</td>
      <td>...</td>
    </tr>
  </table>
 </center>
</body>
</html>
</body>
{% endblock %}
```

---

customerAccounts/admin.py

```
from django.contrib import admin

# Register your models here.
```

---

customerAccounts/forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class teacherForm(UserCreationForm):
    first_name = forms.CharField(max_length=30,)
    last_name = forms.CharField(max_length=30,)
    email = forms.EmailField(max_length=254, help_text='Required. Inform a valid email address.')

    def save(self, commit=True):
        user = super().save(commit=False)
        user.is_teacher = True
        if commit:
            user.save()
        return user

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', )

class studentForm(UserCreationForm):
    first_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
```

```
    last_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
    email = forms.EmailField(max_length=254, help_text='Required. Inform a valid email address.')

    def save(self, commit=True):
        user = super().save(commit=False)
        user.is_student = True
        if commit:
            user.save()
        return user

    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1', 'password2', )
```

---

customerAccounts/models.py

```
from django.db import models

#from django.utils.html import escape, make_safe

# Create your models here.
```

---

customerAccounts/urls.py

```
from django.urls import path
from django.urls.resolvers import URLPattern
```

```
from .import views

app_name = 'customerAccounts'

urlpatterns = [
        path("tSignup/", views.tSignup),
        path("sSignup/", views.sSignup),
        path("login/", views.login_view)
]
```

---

customerAccounts/views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth import login, authenticate, logout
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm

from .forms import studentForm, teacherForm

#loads signup form for teachers
def tSignup(request):
    if request.method == 'POST':
        #loads teacher form from forms.py
        form = teacherForm(request.POST)
        #makes sure what the user enters is valid for the form
        if form.is_valid():
            #saves the user
            user = form.save()
            #logs the user in
            login(request, user)
```

```
            return redirect('tHome')
    else:
        #reloads the form if what they entered was invalid
        form = teacherForm()
    return render(request, 'customerAccounts/tSignup.html', {'form': form})




#the sign up form for students
def sSignup(request):
    if request.method == 'POST':
        form = studentForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('sHome')
    else:
         form = studentForm()
    return render(request, "customerAccounts/sSignup.html", {'form': form})




def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
```

```
            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('homepage.html')
    form = AuthenticationForm()
    return render(request, "customerAccounts/login.html",{"login_form": form})

def logout_view(request):
    if request.method == 'POST':
        logout(request)
        return redirect('')
```

---

customerAccounts/templates/customerAccounts/login.html

```
<!doctype html>
{% extends 'base_layout.html' %}
{% load static %}
{% block content %}
  <head>
     <h1>Login</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div class="accounts-form">
     <form method="POST">
       {% csrf_token %}
       {{ login_form }}
       <button class="btn btn-primary" type="submit">Login</button>
     </form>
```

```
   </div>
  </body>
</html>
{% endblock %}
```

---

customerAccounts/templates/customerAccounts/sSignup.html

```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
<head>
    <h1>Student Signup</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
</head>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign up</button>
</form>
{% endblock %}
```

---

customerAccounts/templates/customerAccounts/tSignup.html

```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
<head>
```

```
    <h1>Teacher Signup</h1>
    <link rel="styles.css" rel="stylesheet" type="text/css">
</head>
<form method="post">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Sign up</button>
</form>
{% endblock %}
```

GibJohnTutoring/settings.py

```
"""
Django settings for GibJohnTutoring project.

Generated by 'django-admin startproject' using Django 4.0.3.

For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.0/ref/settings/
"""

import os
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent
```

```
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-6r_$gc$jk^s5i%xmv1+izqenyi5(@#y+nn6_)sd!@q9(i((j5x'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    #my apps
    'customerAccounts',
    'courses',
    'studentHome',
    'teacherHome',

]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'GibJohnTutoring.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ["templates"],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'GibJohnTutoring.wsgi.application'
```

```
# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}



# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
```

```
# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, "assets"),
)

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

GibJohnTutoring/urls.py

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include
from .import views

#urls that link the whole page together
urlpatterns = [
    path('admin/', admin.site.urls),
    path("", views.index),
    path("customerAccounts/", include("customerAccounts.urls")),
    path("teacherHome/", include("teacherHome.urls")),
    path("studentHome/", include("studentHome.urls")),
    path("courses/", include("courses.urls")),

]
```

GibJohnTutoring/views.py

```
from django.shortcuts import render

from django.http import HttpResponse
from django.shortcuts import render


def index(request):
    return render(request, "index.html")
```

---

studentHome/admin.py

```
from django.contrib import admin

# Register your models here.
```

---

studentHome/models.py

```
from django.db import models

# Create your models here.
```

---

studentHome/urls.py

```
from django.urls import path
from django.urls.resolvers import URLPattern
from .import views

app_name = 'studentHome'

urlpatterns = [
    path("sHome/", views.sHome),
```

]

---

studentHome/views.py

```
from django.shortcuts import render

# Create your views here.
def sHome(request):
    return render(request, "studentHome/sHome.html")
```

---

studentHome/templates/studentHome/sHome.html

```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
<head>
    <link rel="styles.css" rel="stylesheet" type="text/css">
</head>
<body>
    <!Narbar for the whole website>
    <div class="position-relative">
        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
            <div class="container-fluid">
                <h1 class="navbar-brand" href="#">GibJohn Tutoring</h1>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
                    <span class="navbar-toggler-icon"></span>
```

```
        </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="studentHome/sHome/courses/courses">Courses</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Quizzes</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Leader Board</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Customer Support</a>
        </li>
        <li class="nav-item">
          <div class="position-absolute top-0 end-0"><a class="nav-link" href="#"><svg
xmlns="http://www.w3.org/2000/svg" width="30" height="30" fill="currentColor" class="bi bi-
person-circle" viewBox="0 0 16 16">
            <path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>
            <path fill-rule="evenodd" d="M0 8a8 8 0 1 1 16 0A8 8 0 0 1 0 8zm8-7a7 7 0 0 0-5.468
11.37C3.242 11.226 4.805 10 8 10s4.757 1.225 5.468 2.37A7 7 0 0 0 8 1z"/>
          </svg></i></a></div>
        </li>
        </ul>
      </div>
      </div>
    </nav>
  </div>
```

```
  <div class="row align-items-center">
    <div class="w-25 p-3" style="background-color: #eee;">
      <div class="col">
       <h2 class="text-decoration-underline">Current Courses</h2>
      </div>
      <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">course 1</a>
        <a href="#" class="list-group-item list-group-item-action">course 2</a>
        <a href="#" class="list-group-item list-group-item-action">course 3</a>
        <a href="#" class="list-group-item list-group-item-action">course 4</a>
       </div>
    </div>

    <div class="w-50 p-3" style="background-color: #eee;">
      <div class="col">
        <center><h2 class="text-decoration-underline">Welcome</h2></center>
      </div>
    </div>
    <div class="w-25 p-3" style="background-color: #eee;">
      <div class="col">
        <h4>Claim gift <div class="mb-3"><button type="button" class="btn btn-secondary">
          <svg xmlns="http://www.w3.org/2000/svg" width="24" height="24" fill="currentColor"
class="bi bi-gift-fill" viewBox="0 0 16 16">
          <path d="M3 2.5a2.5 2.5 0 0 1 5 0 2.5 2.5 0 0 1 5 0v.006c0 .07 0 .27-.038.494H15a1 1 0 0
1 1 1v1a1 1 0 0 1-1 1H1a1 1 0 0 1-1-1V4a1 1 0 0 1 1-1h2.038A2.968 2.968 0 0 1 3
2.506V2.5zm1.068.5H7v-.5a1.5 1.5 0 1 0-3 0c0 .085.002.274.045.43a.522.522 0 0 0 .023.07zM9
3h2.932a.56.56 0 0 0 .023-.07c.043-.156.045-.345.045-.43a1.5 1.5 0 0 0-3 0V3zm6 4v7.5a1.5 1.5 0 0
1-1.5 1.5H9V7h6zM2.5 16A1.5 1.5 0 0 1 1 14.5V7h6v9H2.5z"/></svg></button></h4>
      </div>
      <div class="col">
        <h2 class="text-decoration-underline">Friends</h2>
       </div>
      <div class="list-group">
```

```
            <a href="#" class="list-group-item list-group-item-action">friend 1</a>
            <a href="#" class="list-group-item list-group-item-action">friend 2</a>
            <a href="#" class="list-group-item list-group-item-action">friend 3</a>
            <a href="#" class="list-group-item list-group-item-action">friend 4</a>
          </div>
      </div>
   </div>
</body>
{% endblock %}
```

---

teacherHome/admin.py

```
from django.contrib import admin

# Register your models here.
```

---

teacherHome/models.py

```
from django.db import models

# Create your models here.
```

---

teacherHome/urls.py

```
from django.urls import path
from django.urls.resolvers import URLPattern
from .import views

app_name = 'teacherHome'

urlpatterns = [
        path("tHome/", views.tHome),
]
```

---

teacherHome/views.py

```
from django.shortcuts import render

# Create your views here.
def tHome(request):
    return render(request, "teacherHome/tHome.html")
```

---

teacherHome/templates/teacherHome/tHome.html

```
{% extends 'base_layout.html' %}
{% load static%}
{% block content %}
<head>
    <link rel="styles.css" rel="stylesheet" type="text/css">
```

```
</head>
<body>
  <!Narbar for the whole website>
  <div class="position-relative">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
        <h1 class="navbar-brand" href="#">GibJohn Tutoring</h1>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav">
          <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="courses/courses">Courses</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Quizzes</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Leader Board</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Customer Support</a>
        </li>
        <li class="nav-item">
```

<svg xmlns="http://www.w3.org/2000/svg" width="30" height="30" fill="currentColor" class="bi bi-person-circle" viewBox="0 0 16 16">

<path d="M11 6a3 3 0 1 1-6 0 3 3 0 0 1 6 0z"/>

<path fill-rule="evenodd" d="M0 8a8 8 0 1 1 16 0A8 8 0 0 1 0 8zm8-7a7 7 0 0 0-5.468 11.37C3.242 11.226 4.805 10 8 10s4.757 1.225 5.468 2.37A7 7 0 0 0 8 1z"/>

</svg></i>

</nav>

</body>

{% endblock %}

Tempates/base_layout.html

```
<!DOCTYPE html>
<html>
    <head>
        <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3"
crossorigin="anonymous">
        <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.10.2/dist/umd/popper.min.js
" integrity="sha384-
7+zCNj/IqJ95wo16oMtfsKbZ9ccEh31eOz1HGyDuCQ6wgnyJNSYdrPa03rtR1zdB"
crossorigin="anonymous"></script>
        <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.min.js"
integrity="sha384-
QJHtvGhmr9XOIpI6YVutG+2QOK9T+ZnN4kzFN1RtK3zEFEIsxhlmWl5/YESvpZ13"
crossorigin="anonymous"></script>
        <meta charset="utf-8">
        <meta http-equiv="X-UA0Compatible" content="IE=edge">
        <link rel="preconnect" href="https://fonts.googleleapis.com">
        <link rel="preconnect" href="https://fonts.googleapis.com">
```

```
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="{% static 'styles.css' %}">
    </head>
    <body>
        {% block content %}
        {% endblock %}
    </body>
</html>
```

Templates/index.html

```
{% extends 'base_layout.html' %}
{% load static %}
{% block content %}
<head>
  <link rel="styles.css" rel="stylesheet" type="text/css">
    <h1>GibJohn Tutoring</h1>
</head>
<body>
<div class="p-3 mb-2 bg-secondary text-white">
  <div class="container-fluid">
    <div class="row">
      <div class="col-3">
        <img src="" class="img-fluid" alt="...">
      </div>
      <div class="w-50 p-3" style="background-color: #eee;">
        <div class="d-grid gap-2 col-6 mx-auto">
          <a class="btn btn-primary" href="customerAccounts/tSignup"
role="button">Signup as a Teacher</a>
          <a class="btn btn-primary" href="customerAccounts/sSignup"
role="button">Signup as a Student</a>
          <a class="btn btn-primary" href="customerAccounts/login"
role="button">Login</a>
          <a class="btn btn-primary" href="teacherHome/tHome"
role="button">bypass to TeacherHome</a>
          <a class="btn btn-primary" href="studentHome/sHome"
role="button">bypass to StudentHome</a>
        </div>
      </div>
        <div class="col">
          <img src="" class="img-fluid" alt="...">
        </div>
    </div>
```

```
  </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js" integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
crossorigin="anonymous"></script>
</body>

{% endblock %}
```

Task 2: Test log

Description of test	Test data to be used (if required)	Expected outcome	Actual outcome	Comments and intended actions
Does the user log in when valid inputs are used	Username: [redacted] Password: 1234	The user is logged in		
When no input is entered the system won't log the user in	No login input	The user is not logged in		
When an invalid input is entered the system won't log the user in	Username: kgnr Password: j32	The user is not logged in		
When signing up an account the system recognises when all inputs are valid	Email: example@email.com Phone: 1234567890 Username: [redacted] Password: 1234 Confirm password: 1234	The account is made		
When signing up an account the system recognises when the	Password: 1234 Confirm password:12345	An Error message shows saying the passwords didn't match		

confirm password is incorrect				
When signing up an account the system will recognise an invalid email	example@email.com	An Error message shows saying its an invalid email		

Add more rows and tables as required.

Asset table

Image/website used	sources	rationale
Bootstrap	Getbootstrap.com	I used bootstrap throughout my webpage to help with developing the look of the website and to improve user experience

Sources table

link	What link goes to
Github.com/MoTechStore/Django-Mulitiple-User-Type-Model/tree/master	I used some of the source code from here to figure out how to make separate user types in my project allowing me to separate students, teachers and admins

T-LEVELS | Institute for Apprenticeships & Technical Education

T Level Technical Qualification in Digital Production, Design and Development (Level 3)

March 2022

Time 30 hours

Paper reference **19540**

Occupational Specialism

Task 2

Developing the solution

You must have:
Task2_Test_Log_Template.doc

Information

- The total mark for this task is 48.
- This booklet contains material for the completion of the set task under supervised conditions.
- This booklet is specific to each series and this material must only be issued to students who have been entered to undertake the task in the relevant series.
- This booklet should be kept securely until the start of the 4-week assessment window.

Turn over ▶

W73774A

W 7 3 7 7 4 A

Pearson

Instructions for students

You must complete ALL activities within the assessment.

This task is to be completed under supervised conditions during a window of four weeks.

The window for this task is 07 March 2022 and 01 April 2022.

Your centre will schedule **30 supervised hours during the four-week window** in which you will produce the outcomes for this task.

During this task, you are allowed:

- monitored access to the internet
- access to a copy of your proposal and design produced in Task 1.

You are **not** allowed to make changes to the evidence produced in Task 1.

Your work will be kept securely during any breaks and between scheduled sessions.

Template provided for use during this task:

- *Task2_Test_Log_Template.doc*

During this task, your tutor is permitted to provide general feedback about:

- the appropriateness of the solution you designed in Task 1
- whether the solution you are developing will function as intended.

Your tutor is **not** permitted to provide guidance on how to improve your solution.

Set Task Brief

The software development company you work for has secured a new contract to develop a digital solution for GibJohn Tutoring.

GibJohn Tutoring currently provides customers with:

- face-to-face tutoring sessions
- access to learning resources
- support to develop understanding in different subjects.

The client (owners of GibJohn Tutoring) would like to develop a digital solution that will:

- provide interactive teaching and learning resources in a range of subjects
- provide access to digital content to encourage wider learning
- support assessment and monitoring of learner progress.

The client has carried out some market research with existing customers and tutors to identify features that could be included in the digital solution. The potential features suggested by the client are:

- collaborative teaching and learning tools
- accessibility features to support a wide range of users
- a learning reward system
- gamified learning.

Activity

Developing the prototype

Develop a functional prototype of your proposed digital solution to meet the client's needs.

During development, make sure you:

- implement secure code in at least two appropriate languages to implement front-end and back-end processes
- gather and prepare appropriate assets to be used in the development of your digital solution
- make use of, and document, iterative testing
- document the iterative development process, including changes made to the solution during development
- produce code that can be maintained by a third party
- ensure a high-quality user experience
- follow legal and regulatory guidelines and standards.

When gathering assets, you need to record the sources you use in an assets log. Your log must:

- record all the sources you have used
- describe the content and its intended purpose
- log the date on which you retrieved the information.

Your testing documentation does not need to include details of every test carried out. However, it should demonstrate:

- an iterative approach to testing
- understanding of how to test inputs, calculations, validation and processes using appropriate test data.

(48)

Outcomes for submission

Prototype

Save copies of key versions of your functional prototype in your folder for submission.

Use any sensible folder structure and naming convention.

Development documentation

Save your development documents as PDF files in your folder for submission. Use this naming convention:

- Task2_[Document name]_[Registration number #]_[Surname]_[First letter of first name]

Save your content and assets log as a PDF in your folder for submission. Use this naming convention:

- Task2_Content_Assets_Log_[Registration number #]_[Surname]_[First letter of first name]

Test log

Save your test log as a PDF file in your folder for submission. Use this naming convention:

- Task2_Test_Log_[Registration number #]_[Surname]_[First letter of first name]

Code for prototype solution

Save your code as PDFs and as .txt files in your folder for submission. Use this naming convention:

- Task2_Code_[doc #]_[Registration number #]_[Surname]_[First letter of first name]

TOTAL FOR TASK = 48 MARKS

My plan for gathering feedback

Firstly, I am going to produce two surveys, one for non-technical users and another for technical users. These surveys will ask them questions about their experience with the website, checking to see if everything worked for them as well as asking what they liked about it and what they think could be improved. I will then make two screencasts of the website again for technical and non-technical users, the non-technical one will just run through the website showing all the basic functions of the website (these will both be viewable within the folder). The technical recording will show off the website but also the code for each section of the page so that I can get their thoughts on my coding and whether any improvements can be made to improve robustness/efficiency.

Once these have been made I will gather feedback from some technical users and non-technical users by having them either watch the screen recordings and then taking the survey to give their feedback, or I'll give them my laptop to try the website out in front of me where I can then observe their actions, giving me a sense of how users typically navigate through my website, as well as get them to take the survey afterwards.

Observation of user

For the first technical user I got to test my website I observed them and recorded my observations:

[redacted]

Age: 18

First, he clicked to sign in as a teacher
Then he went to sign in as the teacher but this gave back an error.

He then went to try and sign in as a student but this also gave back an error

He then decided to go to the login page and tried logging in except this also gave an error

He then took the shortcut I made to go to the teacher homepage and tried using the links on the navbar, one of which gave back an error while the others did nothing

After finishing with the teacher homepage, he went back and tried the student homepage he first tried using the links on the navbar again but this had the same result as with the teacher homepage. Then he tried to see if the links on the courses, friends' and gift sections would work but none of them did anything.

links of forms and feedback gathered

This is the technical feedback form I made. (double click the Icon)

GibJohn tutoring website survey technic

This is the non-technical feedback form I made.

GibJohn tutoring website survey non-te

These are the reponses I recived on the non-technical form

GibJohn tutoring website survey technic

This is a pdf of the responses I got for my technical form

GibJohn tutoring website survey technic

Please note these PDFs are not accessible

Evaluation of feedback

From the feedback I gathered about my website I have been able to figure out what I could do next in order to further develop my solution. The first improvement I have recognized for the colour scheme as all testers gave the same sort of feedback saying that it was too bland and made the page feel empty. Therefore, in the future I will try using a different colour than white for the background and add a dark mode for anyone that would prefer a black background. The next improvement that I can make to the website is that there needs to be a lot more functionality as 3/4 of testers when giving their final thoughts on the website said that there needs to be more, this is also something that I recognized myself and so in the future I will get the navbar fully functioning so the user can easily navigate the website as well as have the various buttons on each of the pages fulfill their functions. Another improvement to be made is for the teacher homepage as 100% of users said no when asked if they like the page. These improvements would involve adding content like a course request button as well as a feature to track each students progress. I also realized through my user observation that there is no way for the user to reach the courses page without having to type in the exact URL and so I will develop this further by having the link to the course page in the navbar functioning next time so that users will be able to reach the page naturally. Lastly, I found that none of my testers were able to signup or login to the website and so in the future I will figure out what the error is that keeps appearing so that users will be able to successfully make an account.

From the feedback I was also able to figure out one thing that I won't have to change or improve upon too much and that is about the layout of each of the pages as many of the users when writing what they liked about the website said the layout was nice and that they liked how spaced out everything was.

The effectiveness of the assets and content used

I selected the search symbol and profile picture as I felt that they would help with the user experience as users are now so used to seeing a profile picture symbol in the top right of the screen and a search symbol next to a search bar. I chose to take these from bootstrap because I knew that it was a reliable source for improving the look of my website as I had already used it to help with the layout of the page and so felt it was the right choice to pick them over images from google or from another website. It also meant that I wouldn't have to worry about copyright as bootstrap is open source and so any legal issues could be avoided.

As for the various buttons that are used throughout my website such as the gift button and the blue buttons that take you to each of the pages, I also selected these from bootstrap for the same reasons as stated above as bootstrap just felt like a trustworthy source for assets I can use to make my website look a little nicer.

How well my solution meets requirements

When comparing my solution to the functional and non-functional requirements that I set out I can see that I wasn't so successful in meeting the functional requirements as some of the high priority functions such as letting the user signup or login were not fulfilled however a majority of the non-functional requirements were able to be fulfilled such as the colour scheme being appropriate for all users to see what's on the screen as well as the website having a fast response time. The key takeaway for me here is that in the future when developing this website further is that I should put

much more focus on the functionality of the website as without that there isn't much for the user to do other than just look at each of the pages.

As for how well my solution meets with KPIs I can't check for quite a few of them as the feature that is needed hasn't been added to my website yet. However, for how long users typically spend on the website I can't know for sure as most of the testing I got users to do involved them going through the whole website so there's no way for me to know how long user retention is kept for on my website however I have figured out that it takes about 5-10 minutes for users to see through all of my website.

Finally, I can see that for the user acceptance criteria my solution was not able to meet a majority of what I had set out to be necessary. Most of the features that were required haven't even been started such as gamified learning or a way to monitor student progress whereas others have some of the front end sorted but because there's nothing for the backend none of it is functional such as a learning reward system and providing access to digital resources. However, there are a couple that I was able to meet which is that the website is adaptable as I used bootstrap so the page should naturally change to fit the screen size and because of the feedback I gathered I know that users find the website easy to navigate through.

Conclusion

In conclusion I find that I'm happy with the general look of the website and how I've spaced the content out across the page as well as how the navbar looks. It also seems users agree with that so I won't try to change that too much when making improvements however, I will put some thought into changing up the colour scheme to something that may be more agreeable. As for major improvements I would make in the future, firstly I would put a large amount of focus on making sure the user accounts are working and also making the courses page fully functioning in order to fulfil more of the user acceptance criteria as this would then make digital learning recourses available to users before then moving on to adding content to the teacher homepage so that users don't find a page with nothing on it.

Explore Pearson's
T Levels offering at
quals.pearson.com/tlevels