FORUM DOCUMENTATION

By Joshua Muiruri 12.13

Contents

Analysis

Problem Identification

The problem is that there is currently no site online where Biology can be discussed as a community to better peoples understanding of different topics. A site like this has the benefit of being easy to navigate and easy to find answers to common questions and even start constructive discussions about the subject. Many sites act as question-and-answer forums where questions are asked, answered, followed, and edited by users on the internet, for example, Quora and Stack Overflow do it in this style, Quora has a variety of topics while Stack Overflow focuses on coding and computer science.

Personally, as someone who is studying biology at A-Level, I know that it can become very drawn out and tedious especially when browsing through long and in-depth studies. Therefore, a website like this will be a useful source of knowledge for students. Also, with enough interaction, it can become a place of collaborative learning. I intend to create a website in this style using the Django API in the backend which can make it complex without being difficult.

The problem is computable because the use of a computer will allow users to be connected and interact with each other. Also because of how technologically developed the world is now, younger people would be more interested in an electronic solution. This is easily solved through the development of a website that has the following features:

- A system to put forward questions and enter solutions
- An upvote system that allows users to like or dislike questions and solutions
- A trending page where the most trending questions by upvote are listed
- A way to log in and save questions/ forums
- An organised toolbar with different sections like revision, discussions, example questions, etc.

These features are made simpler by Django API since it handles the server, database and other backend features without being difficult to code. Django allows you to develop a website without some of the tedious parts of web development.

Stakeholders

The primary client base for this website is made up of those who are older than 16 who are studying biology at a higher level (Starting at GCSE level).

The main group will be students aged 15-18 that are studying biology. They will use the site to discuss with others online about different questions and topics. They can choose to just look up detailed solutions to questions they find difficult while revising or they can even contribute some of their knowledge to the community of the website by answering questions asked.

The website will be used as a good source of revision for students since they will be able to view and compare many essay questions with their own. Questions and forums will be organised into separate topics so students can find what they're looking for efficiently.

One stakeholder that I have chosen is a 17-year-old student named Stephen who finds it difficult to revise biology when he has a test coming up. He specifically finds it boring to come up with example questions to answer to further his essay writing. Because of this, his test scores are lower than he wants since he normally loses marks in his essays. He wishes to push up his test grades from a C to a B and then an A if possible. The website should be a good source of revision material for Stephen, and he could even submit his essays to be peer-reviewed by the other students that use the website. This will let him identify mistakes and improvements that he could make that he never noticed. I will be asking him throughout development to put forward his ideas for any changes that he deems to be beneficial to him so that he can be happy with the end product.

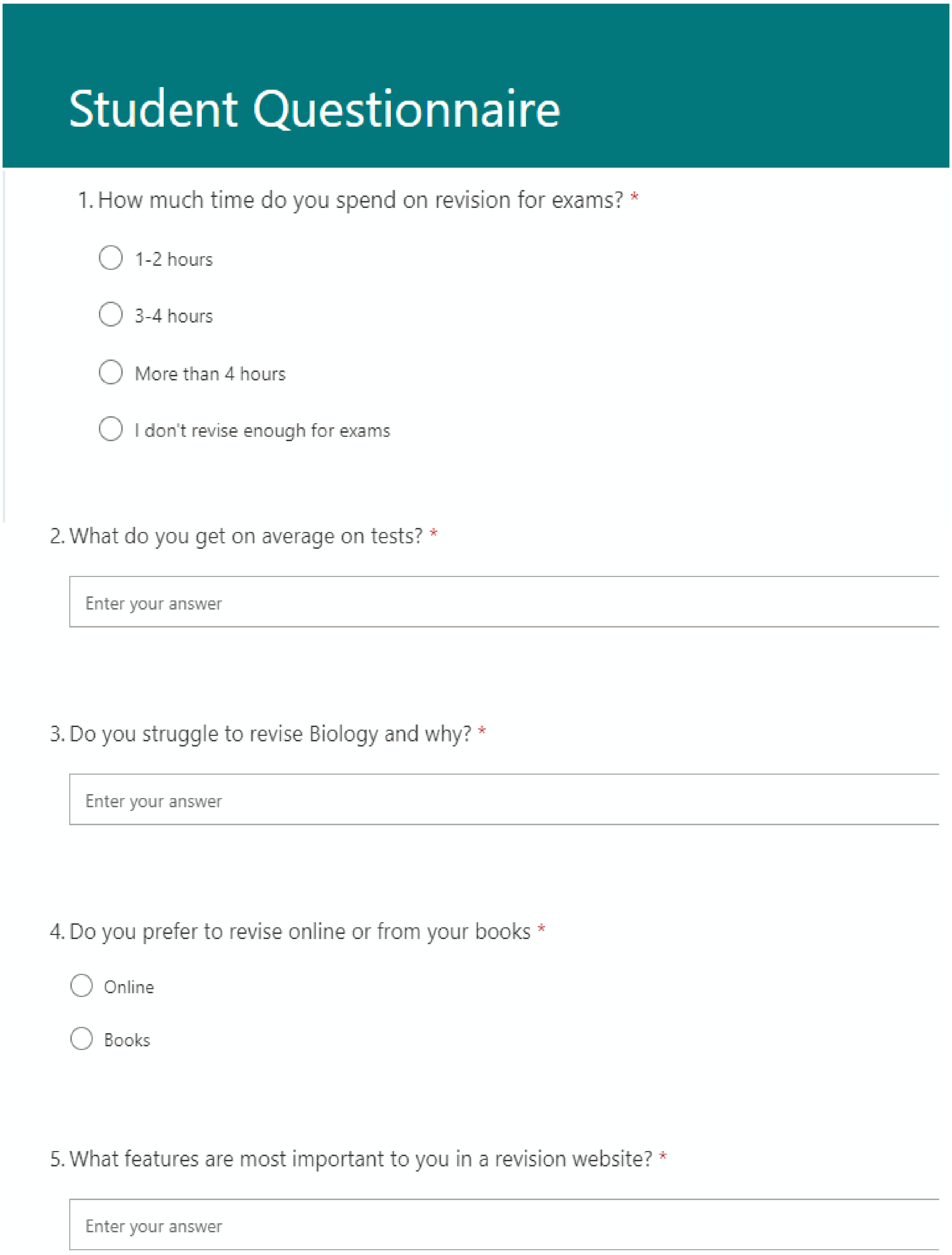
Another stakeholder chosen named Mr Smith is a biology teacher that often finds it difficult to make students more interested in biology outside of lesson, which he believes would increase their grades. He suggests a website that his classes can interact on when doing revision or coursework would be helpful so the website I intend to create as a solution should solve his problem. Including a feature where he could create a separate forum for his class where he takes control as the admin would be beneficial in regulating what content is shared by the students. I will be asking him any ideas or features that he would like developed for his use of the website.

Additionally, another stakeholder I've chosen called Kwame is a 17-year-old student studying A-level Biology who is doing pretty well compared to the rest of his class since he averages A's and B's in all his exams and coursework. Although, he aims to improve his grades to and A* but can't an efficient way of doing this without revising for hours on end every day. Having a website where he can view and take part in discussions allowing to reinforce his revision. He can also get a better understanding of topics that he isn't as good. I will be considering any improvements that he suggests being changed in the solution.

My final stakeholder that I have chosen is the head of the biology department in my school. His name is Dr Adekunle, and he believes that the majority of students have a lot of potential but often in the first year slack off but then in the second year put in more effort and get better grades. He wants to change this so that students can be more consistent and higher achieving. He suggests a website that classes can interact on when doing revision or coursework would be helpful. The website I will create should be a good solution to this. I will be asking him for any features that he wants to include or alter throughout development.

Questionnaire

I have created two questionnaires on Microsoft forms tailored to either students or staff to find out if the creation of this website is valid.



Student Questionnaire

1. How much time do you spend on revision for exams? *

◯ 1-2 hours

◯ 3-4 hours

◯ More than 4 hours

◯ I don't revise enough for exams

2. What do you get on average on tests? *

Enter your answer

3. Do you struggle to revise Biology and why? *

Enter your answer

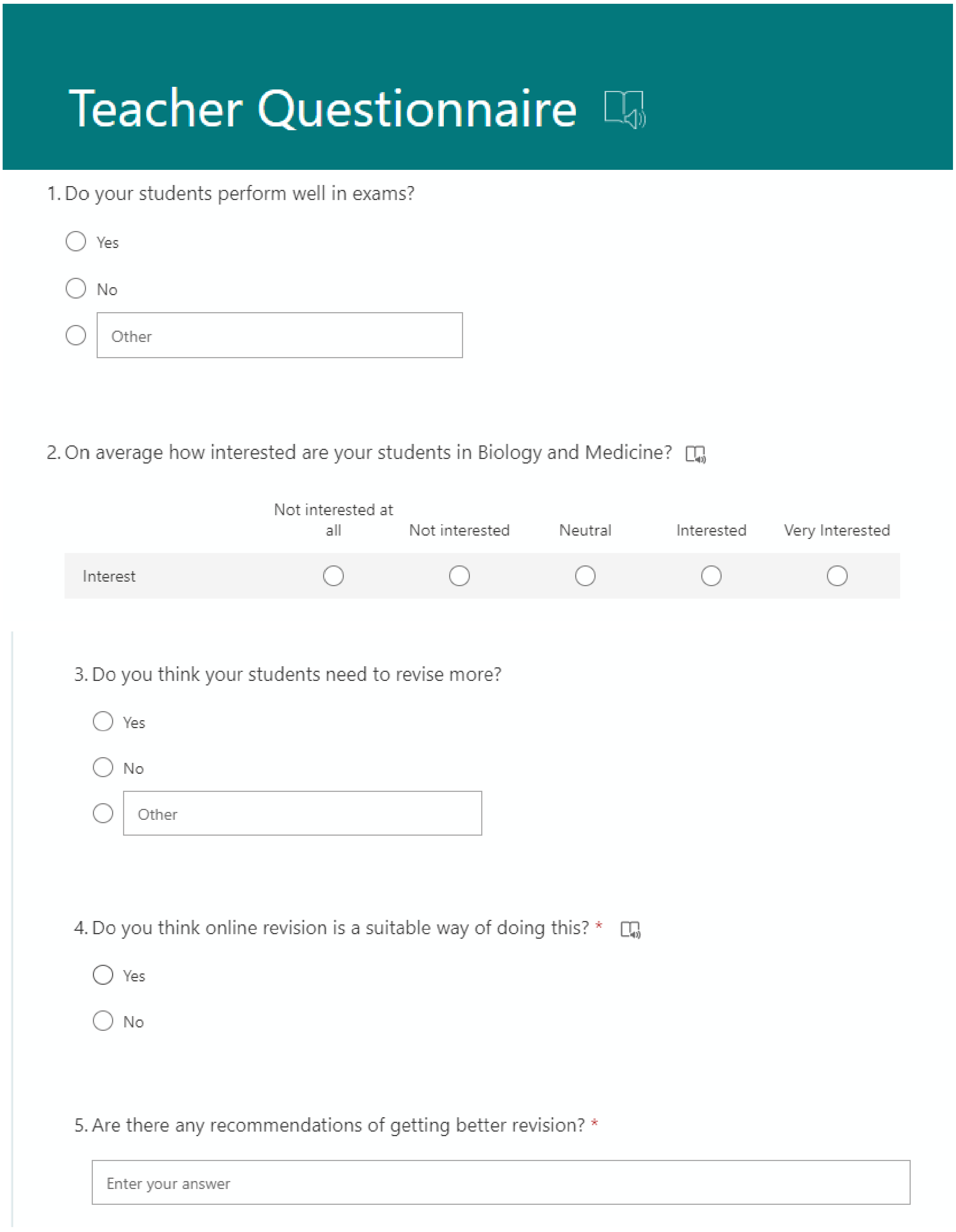
4. Do you prefer to revise online or from your books *

◯ Online

◯ Books

5. What features are most important to you in a revision website? *

Enter your answer



Teacher Questionnaire

1. Do your students perform well in exams?

- Yes
- No
- Other

2. On average how interested are your students in Biology and Medicine?

	Not interested at all	Not interested	Neutral	Interested	Very Interested
Interest	○	○	○	○	○

3. Do you think your students need to revise more?

- Yes
- No
- Other

4. Do you think online revision is a suitable way of doing this? *

- Yes
- No

5. Are there any recommendations of getting better revision? *

Enter your answer

Responses

Stephen – Student

1. I don't revise enough
2. On average I get grades B-C
3. Yes, I struggle with writing essay questions which is the main part in exams that I struggle with and lose marks on.
4. Other – I normally study using my textbook by copying out pages of notes but, more recently I've been more inclined to looking online to find more simplified and concise notes and I use a revision website called Seneca learning.
5. I would want a way that students could interact, exchange and compare essay answers and other questions so that I can improve my essay technique. It would be nice if different answers were tagged or if I was able to save good essay answers for later. Saving discussions and questions would also be useful to look back at.

Kwame - Student

1. 3-4 hours
2. I get A's and B's normally
3. I don't struggle to revise at all, but I can't seem to get better results since I'm aiming to get an A* to get into the university I want to go to.
4. Other – I use a mix of both books and online, I my book to know exactly what's required in the specification but tend to use online websites like physics maths tutor to print out past papers and questions to revise from.
5. I'd want a place to find answers to questions that I struggle on, but more importantly a way to compare my answers with people who are achieving A* grades to learn what works for them and how I can improve. Being able to have a personal account and save topics useful to me. Maybe if I discuss with them about the subject more, I can improve my interest and grades.

Mr Smith – Teacher

1. On average my students get Bs and Cs
2. Other – Half of the class is interested and intend to do biology at university, but the other half doesn't seem to care that about biology.
3. Yes, extra revision is always going to benefit them since I would like more of my class to get higher grades like A and A*s.
4. With the right websites online websites can be good at assessing your knowledge.
5. Answering questions and doing past papers can help. Comparing and looking at good essays.

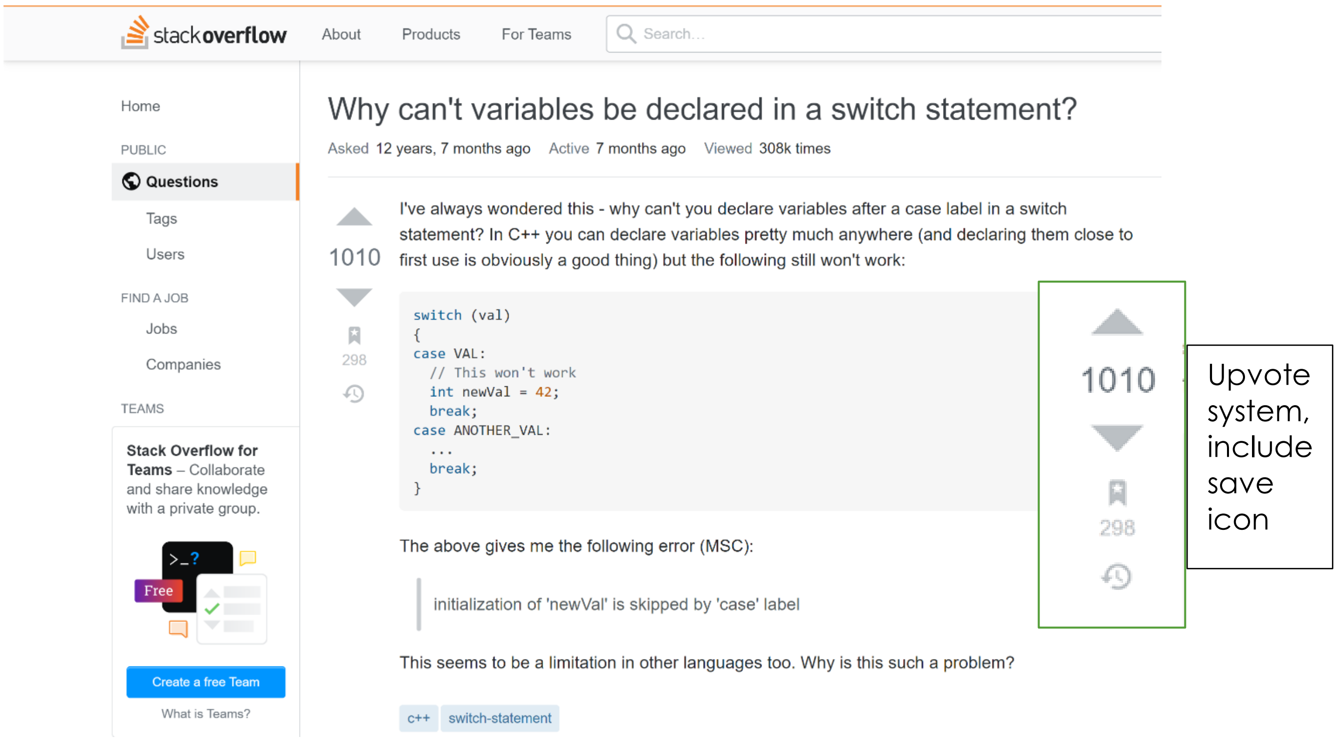
Dr Adekunle – Teacher

1. My students normally get grades ranging from A*- C
2. Interested
3. Of course, my students need to revise more since what you put in is what you get out of a subject.
4. Yes, this is a form of revision that many of my students are interested in
5. Doing past papers is by far the best form, once you know the content.

Research

Similar example – Stack Overflow

https://stackoverflow.com



Why can't variables be declared in a switch statement?

Asked 12 years, 7 months ago Active 7 months ago Viewed 308k times

I've always wondered this - why can't you declare variables after a case label in a switch statement? In C++ you can declare variables pretty much anywhere (and declaring them close to first use is obviously a good thing) but the following still won't work:

```
switch (val)
{
case VAL:
  // This won't work
  int newVal = 42;
  break;
case ANOTHER_VAL:
  ...
  break;
}
```

The above gives me the following error (MSC):

> initialization of 'newVal' is skipped by 'case' label

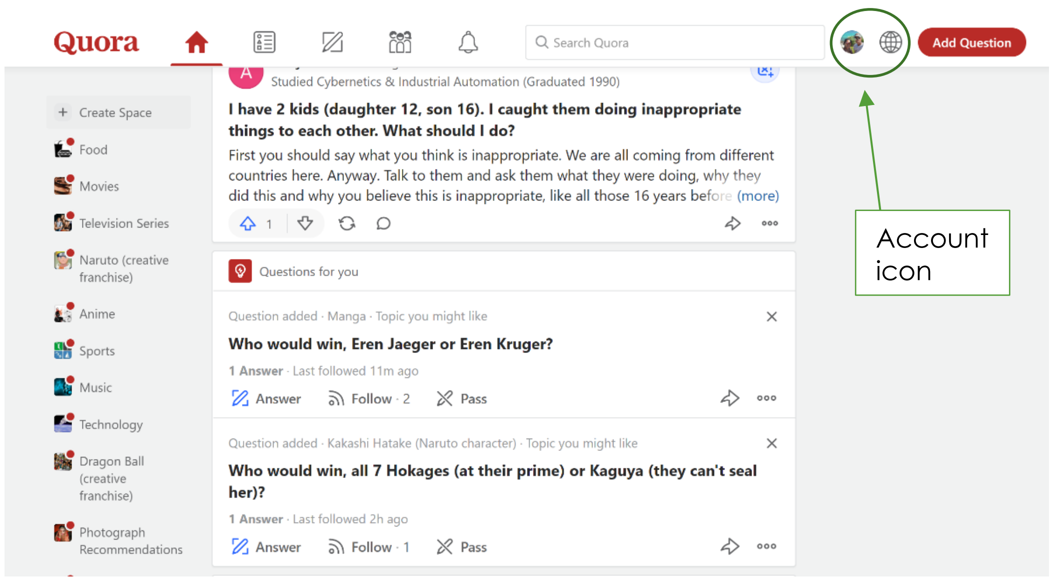
This seems to be a limitation in other languages too. Why is such a problem?

Stack Overflow is a question-and-answer style website that deals with any problem concerning computer programming. It was created on 15th September 2008 and since has become the most used website for learning programming, even professional programmers use it to solve parts of code they are stuck on. The website contains all the features that a basic site should have like a search bar, login system, toolbar linking to other pages, etc. It also effectively implements a Q&A and upvote system which are the key parts of the site. Users can submit and view questions on different topics and for each question anyone is able to answer with a solution. Users can upvote or downvote a solution to say if it worked or not, which could sort the answers provided. The website automatically displays the solutions in order of number of upvotes from highest to lowest, making it easy for users to find solutions that work for the most people. This makes the website a sort of community.

Stack overflow is a more educationally focused website in comparison to the other examples, making it the most like the website I intend to create.

Parts I wish to include in my solution:
The overall structure and concept of Stack Overflow is similar to my solution so it should be simple to apply features that will be useful. I am going to implement an upvote system into my website because it essential in sorting solutions to make the right information easier accessible for users. The idea of using an upvote system to sort solutions to questions is clearly effective and would fit well in my solution. The inclusion of this idea will be appreciated by Stephen who would want to find the best answers as quickly possible when searching through the website.

Similar example – Quora
https://www.quora.com



Studied Cybernetics & Industrial Automation (Graduated 1990)

I have 2 kids (daughter 12, son 16). I caught them doing inappropriate things to each other. What should I do?

First you should say what you think is inappropriate. We are all coming from different countries here. Anyway. Talk to them and ask them what they were doing, why they did this and why you believe this is inappropriate, like all those 16 years before (more)

Question added · Manga · Topic you might like

Who would win, Eren Jaeger or Eren Kruger?

1 Answer · Last followed 11m ago

Question added · Kakashi Hatake (Naruto character) · Topic you might like

Who would win, all 7 Hokages (at their prime) or Kaguya (they can't seal her)?

1 Answer · Last followed 2h ago

Quora, founded in June 2009, is a more of a social media website that is to discuss any question that is brought up. There is no specific focus of questions asked/ forums started, it is all down to the users. Users are able to create accounts on this website allowing a history of questions that they interacted with to be formed. Having an account also allows them to save favourite forums or topics for the next time you log in to the website. Quora then, through the use of an algorithm, recommends you questions that it believes you might like to make you keep using the site.

Quora sorts questions into different topics so that users can find questions that they want to view and interact with. This is different in the way that it allows users to post images and videos making it lean more to being a social media site, but at its core is still a question-answer website.

Parts I wish to include in my solution:
The idea of having a way to create an account and save topics for users is essential to long term usage of the website. If users are given a way to save

what they do on the website, they should have a higher chance of using the site again. This will make students more inclined to use the website for biology more often making it build its user base more. This addition should be well appreciated by Stephen who would want to save solutions to questions and topics he doesn't understand when revising or completing essays.

Essential Features

One essential feature of my solution is that there needs to be a section of my website where users can make questions, answer and discuss questions which would make it similar to a forum. All questions will be stored on the site's database through Django. This would allow users to communicate with each other and could increase how often they come back to the website. This is the main feature of my solution that makes it slightly unique to a forum.

Another essential feature is that it must have an upvote and down vote system. This will allow users to assess if a question or solution is good or not and will make it easier for users to find the most viable answers making traversing the website easy. You would be able to compare different answers quickly

Another essential feature is to have a search bar where you can search through all of the current questions and tags/topics. This is useful because it allows users to sort through and access questions that they are concerned with like on a specific topic.

The website needs to have a login system where you are able to create an account and log in to it with an email and password. This is useful because it will let users save questions, and answers that they submitted but also your account will be accessible by other users to see your answers.

A section of the website where classes/rooms can be created, and teachers can be made admin and control the content in the room/class. This would benefit teachers and school environments since students can be set to discuss work in the room which is beneficial for learning.

A toolbar where different sections of the website are linked like classrooms, discussions, questions, etc. This useful since it allows users to access different sections of website quickly and easily.

Limitations

One limitation is that the website most likely won't be compatible with mobile devices meaning it won't be useable and might be slow because phones are less powerful than computers and the website won't be optimized for phones. This is because to optimise the use of the website on the phone, a whole new user interface would need to be designed so that the site can be easily used. This would take more development time and money - and the use on a mobile phone is questionable, as the program suits a much larger screen.

Another limitation is that students might not just discuss biology related information, they may just turn the site into a chat site which is disruptive to how the site runs. This is a limitation since there is no simple way to fix this issue without employing someone to check through questions and warn users when they discuss other topics.

Finally, another limitation is that users are not able to comment on solutions. This is a limitation because, if a user makes a mistake in their solution, other users would be able to help and correct them. This increases the prominence of incorrect answers on the site which my solution will not have because of technical difficulties.

Requirements for solution

Hardware

1. A basic PC that has basic graphical capabilities that is able to run a browser like Chrome or internet explorer, so that the website can be opened on the system. It shouldn't drop frames since the website is more server orientated instead of focusing on the client.
2. A monitor so that information for the website can be displayed on a screen.
3. Accessories like a keyboard and mouse a required to interact with the website through clicking on links and asking/answering questions (filling forms).

Software

4. A basic browser like google chrome or internet explorer needs to be installed on the PC so that the website can be run.

General requirements

Requirement number	Requirements	Justification
1	The website must be able to be run in all common browsers, e.g., Microsoft Edge, Chrome, Firefox, etc.	Since it must be accessible to as many people as possible, as it is an educational website and isn't paid or limited. Half of my stakeholders are students and wouldn't want to pay to use the site.
2	The ability to create a separate, locked room where different users can be added, and an admin can be set which controls the room.	My stakeholder Mr Smith suggested that he wanted a separate forum for his students that he can control. But also, this is a useful feature to have for people that may interact with friends for a project.
3	The ability to create a personal account where you can store questions and access rooms on your account.	Both of my stakeholders who are students mentioned a way to save questions (see questionnaire for Stephen and Kwame) that they want, and the teachers would like to

		create rooms where students can interact with each other. The easiest way for this to be done is through creation of a personal account.
4	A forum that allows questions and answers to be exchanged by users.	Two of my stakeholders mentioned how this feature would be beneficial (Dr Adekunle and Stephen), allowing them to compare answers and discuss topics.
5	A toolbar which shows links to different sections of the website like trending, rooms, questions, etc.	The website needs to be easily traversed since two of the stakeholders want the website to have information that's easily accessible. This solves the problem making it easy to navigate through the site
6	A search bar where you can search through all questions/topics.	One of my stakeholders wants it to be straightforward to find information that you want which a search bar will solve.

Success criteria

The website must have a question-and-answer based system which includes:

1. A way to enter text and put forward questions for all users of the site, including those that haven't created an account so that anyone can access the site.

2. A way to put forward solutions to questions asked for all users so that you can interact with the website.

3. A way to upvote and down vote questions and answers so that they can be organised to where the answers with more upvotes are shown first.

The website must have and be:

4. Working and able to run in any browser.

5. A search bar where you are able to search and sort through all of the questions or topics in the websites database. This allows users to easily access all the content that they wish to find.

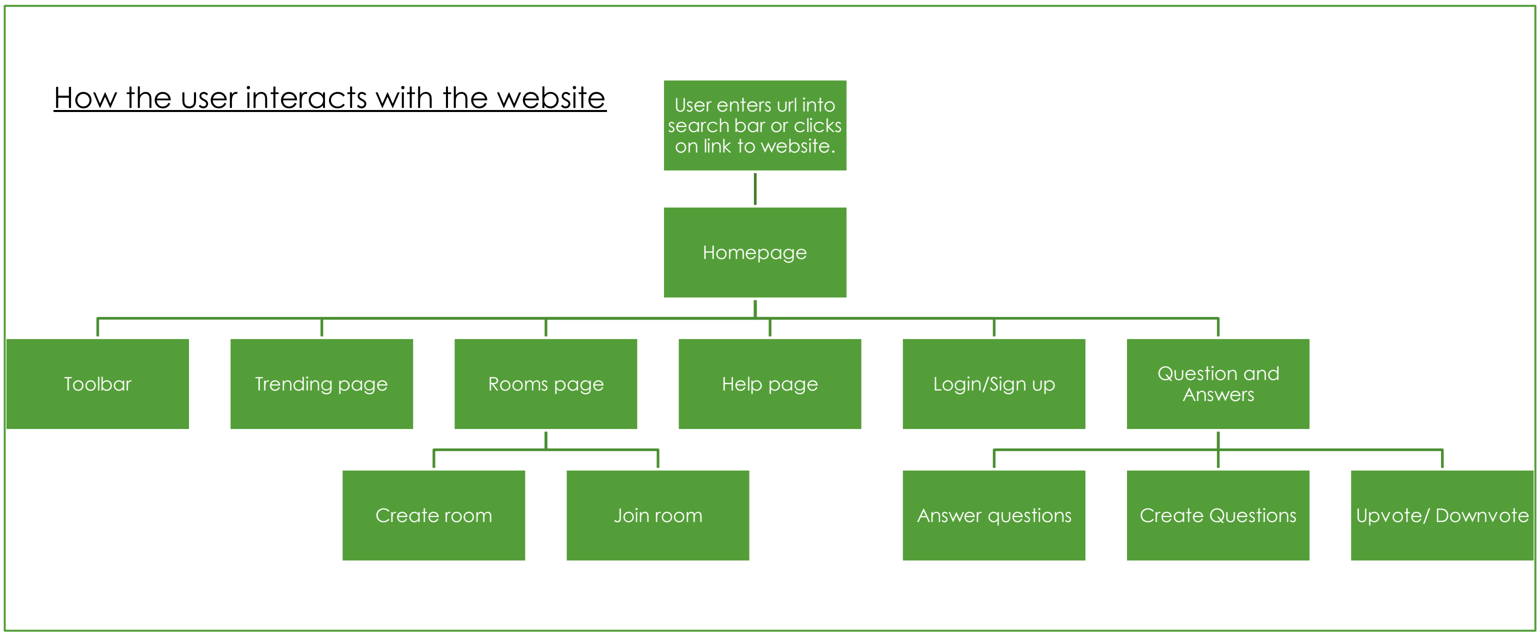
6. A toolbar at the top of the website where there are links to different parts of the website which makes traversing the website easier.

7. A way to create an account on the website allowing features like saving questions so that people will be more likely to use the website more often.

Design

Breaking down the problem

My website will be integrating Django as the main API and therefore will have to take a specific path to run and chose which exact page will be rendered. Here is that series of events broken down using a top-down design. A hierarchy of how users will interact with the website will also be shown. Since Django handles everything to do with the website including the webpage displayed and the database.



How the user interacts with a website analysis

The user, once entering the URL, will be taken to the homepage. The layout of the homepage will be separately able to be configured within Django using the template feature. Templates are created using HTML

Homepage

The homepage is necessary as the initial part of the website that the user interacts with. On the homepage there will be links to different pages of the site like:

Questions and answers

Questions and answers is the main function of the website will be able to be coded within python through Django. This will encompass everything including the creating and answering of questions. This section of the website will be individually coded using decomposition into different modules. The upvote/downvote system will be a separately coded module on the page that will also be able to rank questions asked. There will be an option to save questions if you are logged in.

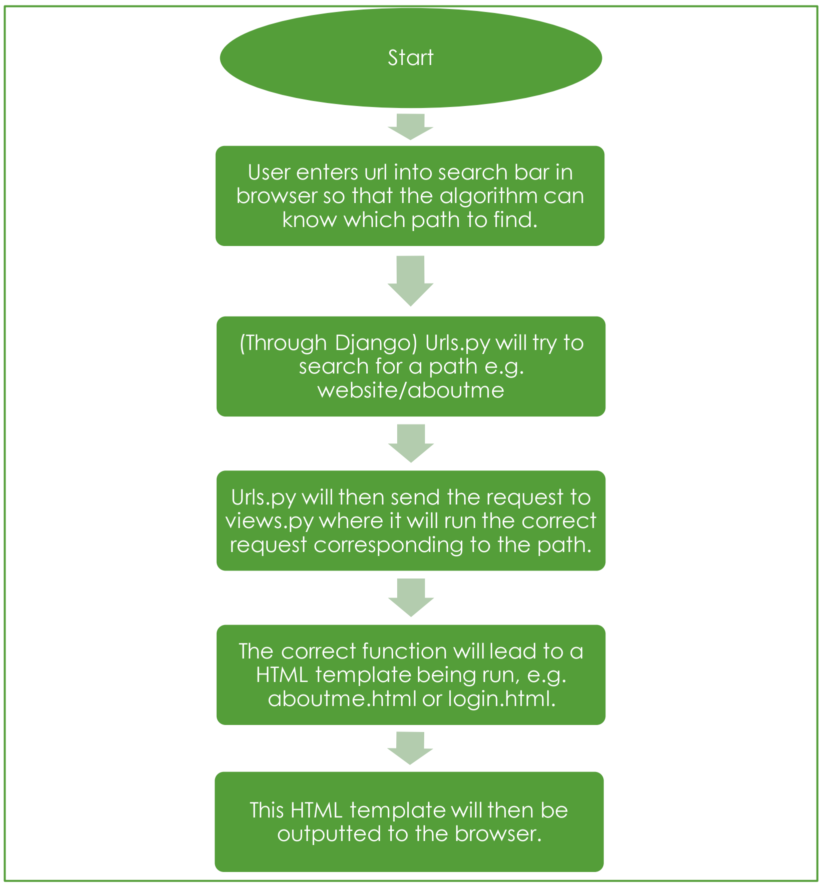
This section will be on its own individual page to make it easier

Login/signup
The login/signup page is where users will be able to log into a previously made account or create a new one to be stored in the database. This page is necessary if users wish to save questions or answer questions, so users must create an account if they want to get the full usage out of the website.

Help
The help page will be an informative page that directs users on how the website operates and what to do if they're stuck or have an issue.

Toolbar
The toolbar allows for users to quickly access different sections of the website with ease by offering links leading to important pages on the site, e.g., homepage, help, rooms page, etc. This makes the site more easily navigable for users.

How Django outputs a webpage


Start
User enters url into search bar in browser so that the algorithm can know which path to find.
(Through Django) Urls.py will try to search for a path e.g. website/aboutme
Urls.py will then send the request to views.py where it will run the correct request corresponding to the path.
The correct function will lead to a HTML template being run, e.g. aboutme.html or login.html.
This HTML template will then be outputted to the browser.

Summary
First the user will enter the URL of the website into the search on their web browser. Django will then search for the corresponding path that is entered.

Usability features

A clear layout

Buttons and all elements will be evenly spaced so that users can clearly view and interact with the website. Having elements separated and clearly labelled will make the website easier to read and less complex.

A toolbar/dropdown menu

Having links to other pages in the site compacted into a toolbar or dropdown menu will free up space on the homepage for other information and making it more readable.

A help page

If a user is confused in any way or has an issue, they can navigate to the help page which will show them how the website works and how to interact with it. This will reduce the confusion that some users experience making the website more useable.

Well formatted content

Most users don't read all of the information on a webpage and instead skim through the page. Having a well formatted page with outlined headings, subheadings, paragraphs and other elements allows users to quickly identify information that they are looking for.

Usable Forms

Forms are very important to the usability of most websites, especially this one. They are integral in creating questions, answers, registering and logging in to the page. This needs to work seamlessly with the backend of the website so that users can easily access and input content.

Algorithms of the code

Register

```
class RegisterView(FormView):

   @method_decorator(csrf_exempt)
   function dispatch(self, request, *args, **kwargs):
      return super(RegisterView, self).dispatch(request, *args, **kwargs)
   endfunction

   # Method that retrieves user data from database and renders it (GET)
function get(self, request)
      content = {}
      content['form'] = RegisterForm
      return render(request, 'register.html', content)
   endfunction

   # Method that adds new user data entered from register page to database
but also checks if it's valid (form.is.valid), e.g. password length, characters
and symbols, etc.
```

```
function post(self, request)
   content = {}
   form = RegisterForm(request.POST, request.FILES or None)
   if form.is_valid():
      save_it = form.save(commit=False)
      save_it.password = make_password(form.cleaned_data['password'])
      save_it.save()
      login(request, save_it)
      return redirect(reverse('dashboard-view'))
   content['form'] = form
   template = 'register.html'
   return render(request, template, content)
   endfunction
```

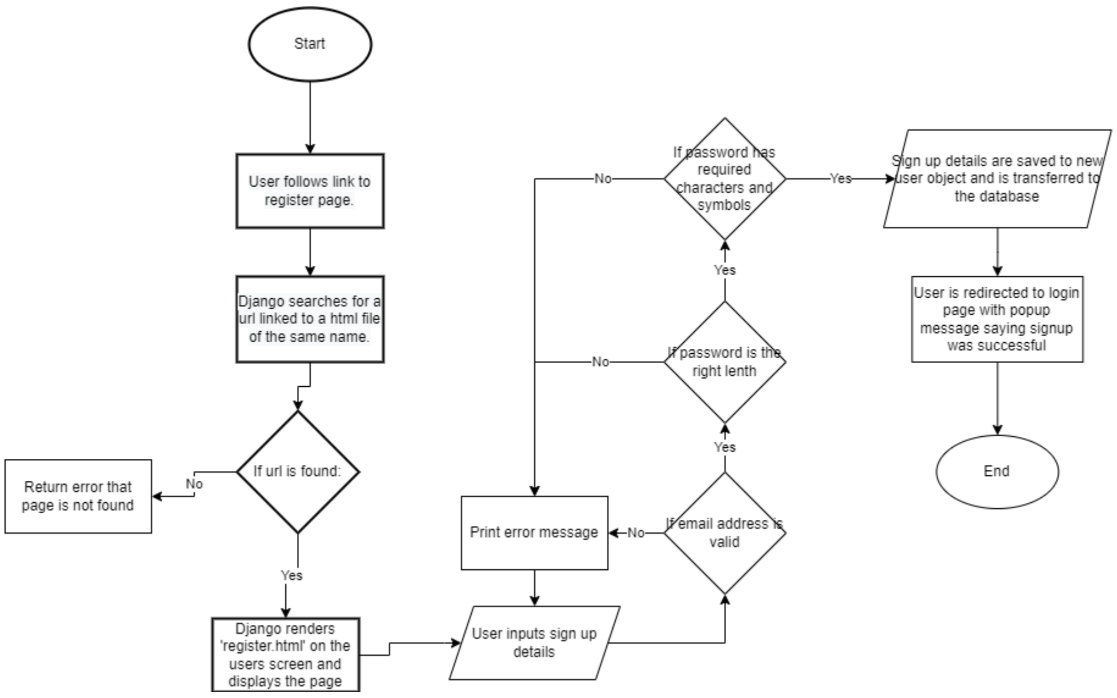

Start
User follows link to register page.
Django searches for a url linked to a html file of the same name.
If url is found:
No
Return error that page is not found
Yes
Django renders 'register.html' on the users screen and displays the page
User inputs sign up details
If email address is valid
No
Print error message
Yes
If password is the right lenth
No
Yes
If password has required characters and symbols
No
Yes
Sign up details are saved to new user object and is transferred to the database
User is redirected to login page with popup message saying signup was successful
End

Figure 1 - Register flowchart

Login

```
class LoginView(FormView):

   content = {}
   content['form'] = LoginForm

   @method_decorator(csrf_exempt)
   function dispatch(self, request, *args, **kwargs):
      return super(LoginView, self).dispatch(request, *args, **kwargs)
   endfunction

   # Method that retrieves user data from database and renders it (GET)
   function get(self, request):
      content = {}
      if request.user.is_authenticated:
         return redirect('questions_list')
      content['form'] = LoginForm
      return render(request, 'login.html', content)
   endfunction
```

# Method that sends login details to database to be checked if they match a user

```
   function post(self, request):
      content = {}
      email = request.POST['email']
      password = request.POST['password']
      try:
         users = User.objects.filter(email=email)
         user = authenticate(request, username=users.first().username,
password=password)
         login(request, user)
         return redirect('')
      except Exception as e:
         content = {}
         content['form'] = LoginForm
         content['error'] = 'Unable to login with provided credentials' + e
         return render_to_response('login.html', content)
    endfunction
```

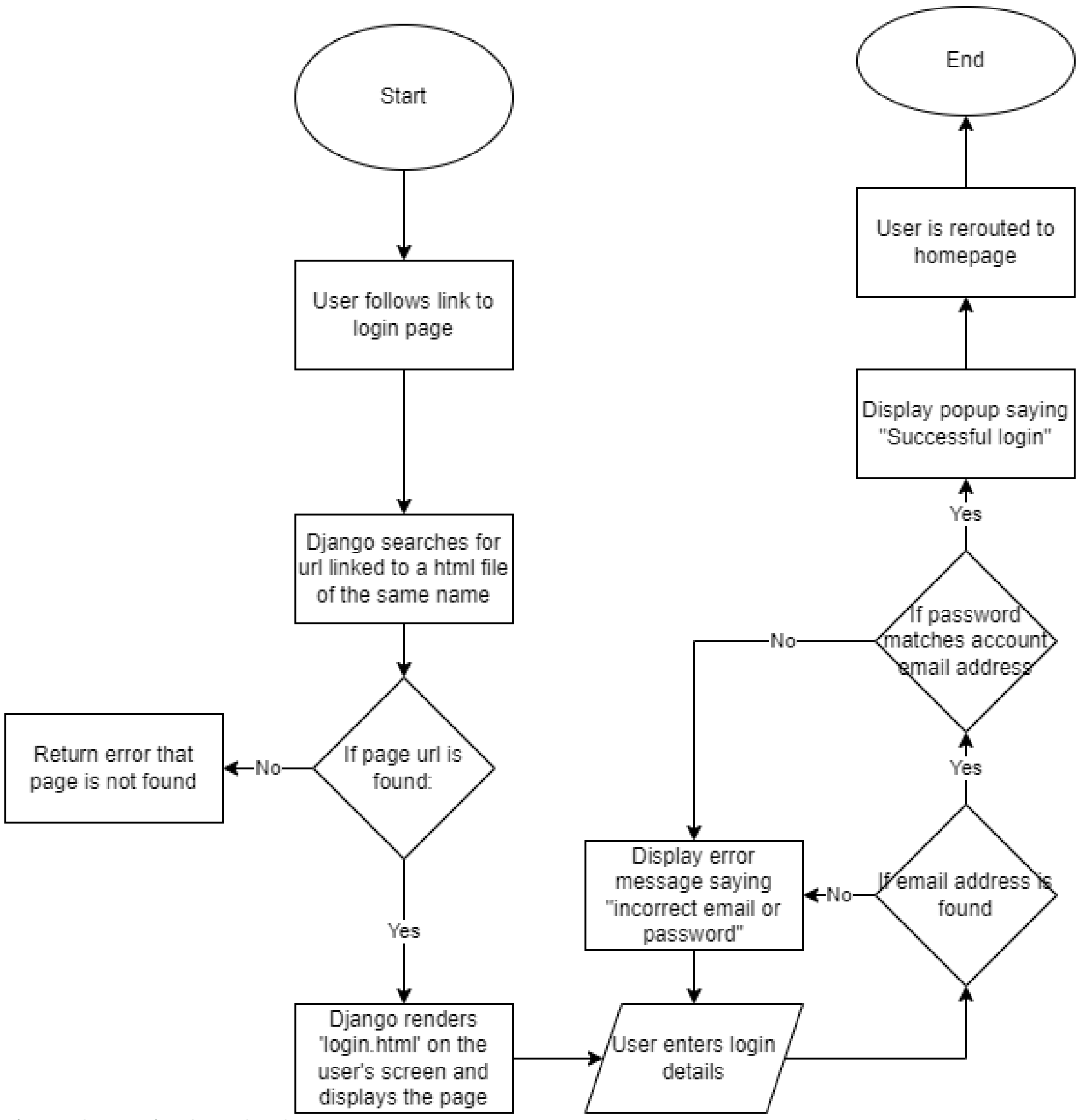

Start
User follows link to login page
Django searches for url linked to a html file of the same name
If page url is found:
No
Return error that page is not found
Yes
Django renders 'login.html' on the user's screen and displays the page
User enters login details
If email address is found
No
Display error message saying "incorrect email or password"
Yes
If password matches account email address
No
Yes
Display popup saying "Successful login"
User is rerouted to homepage
End

Figure 2 - Login Flowchart

These two algorithms come together to answer requirement 3 which is the ability to create a unique user.

Requirement 4 is then answered then answered by the following algorithms that handle question and answer creation:

Questions

```
class Questions(models.Model):
   #fields and other data
   user = models.ForeignKey(settings.AUTH_USER_MODEL,
         on_delete=models.CASCADE, null=True, blank=True)
   title = models.TextField()
   description = models.TextField(null=True, blank=True)
   points = models.IntegerField(default=0)
   group = models.ForeignKey('QuestionGroups',
         on_delete=models.CASCADE, null=True, blank=True)
```

```
    created_on = models.DateTimeField(auto_now=True)
    updated_on = models.DateTimeField(auto_now_add=True)
    slug = models.SlugField(unique=True, null=True, blank=True)

    #creates a slug of from the title of a Question
    def save(self, *args, **kwargs):
        self.slug = slugify(self.title)
        super(Questions, self).save(*args, **kwargs)

    #changes unicode within admin page to object 'title'
    def __unicode__(self):
        return self.title

    #changes str within admin page to object 'title'
    def __str__(self):
        return self.title
```

Question Form

```
class QuestionForm(forms.ModelForm):

    title = forms.CharField(max_length=90, required=True)
    description = forms.CharField(required=True, widget=forms.Textarea())

    class Meta:
        # Inherits data and fields from the 'Questions' model class
        model = Questions
        # The fields that will be displayed in the form
        fields = [ 'title', 'description', 'group']
```

Answers

```
class Answers(models.Model):
    #fields and other data
    user = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    #takes in the id of the question that is being answered
    question = models.ForeignKey(Questions, on_delete=models.CASCADE)
    #users entered answer
    answer = models.TextField()
    created_on = models.DateTimeField(auto_now=True)
    is_anonymous = models.BooleanField(default=False)


    def __unicode__(self):
        return self.id
```

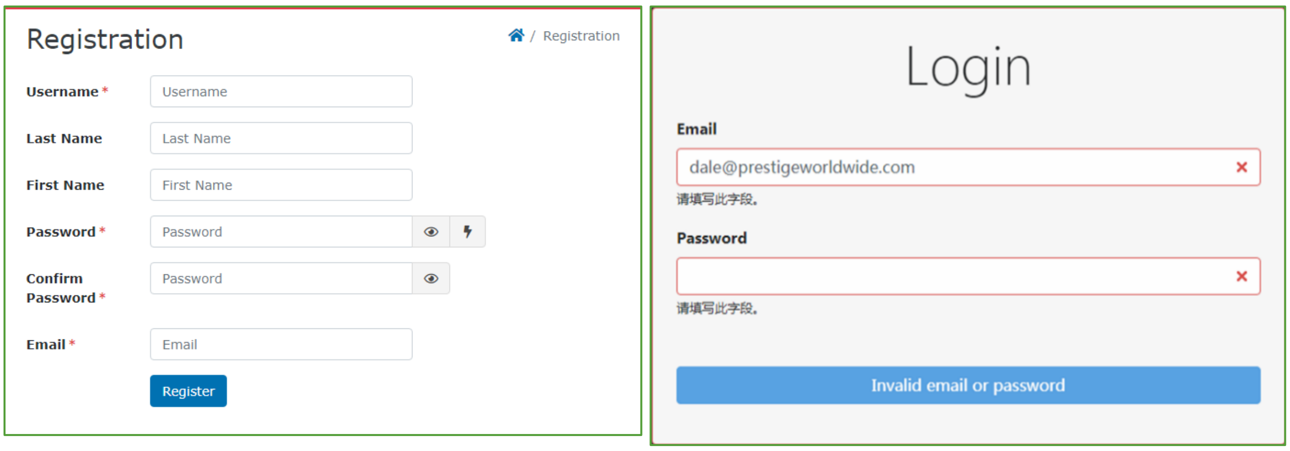
Answer Form

```
class AnswerForm(forms.ModelForm):
    class Meta:
        # Inherits data and fields from the 'Answers' model class
        model = Answers
        # The field that is displayed in the form
        fields = ['answer']
```

User interfaces

Login and Registration

The Login and Registration UI is going to be simple showing a main title followed by forms for each field in a user's account. These fields will be email and password for the login page and username, password and email for the registration page.



Registration

Registration

Username *	Username
Last Name	Last Name
First Name	First Name
Password *	Password
Confirm Password *	Password
Email *	Email

Register

Login

Email

dale@prestigeworldwide.com

请填写此字段。

Password

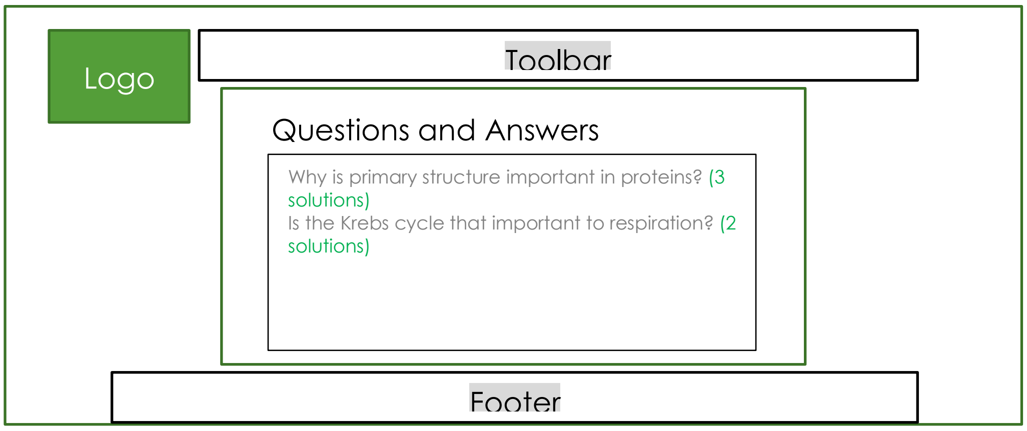
请填写此字段。

Invalid email or password

Figure 3- Login and Register page examples

Dashboard/ Homepage

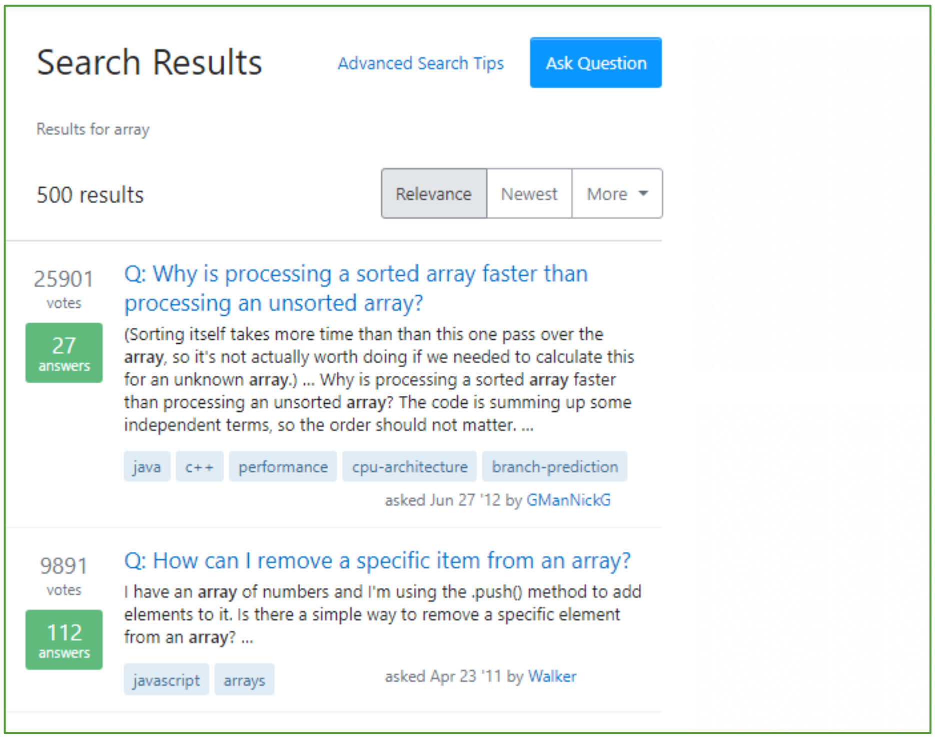
The Dashboard page will be similar to Quora and StackOverflow in the way that it will have a toolbar at the top and a footer at the bottom linking to the



Logo
Toolbar
Questions and Answers
Why is primary structure important in proteins? (3 solutions)
Is the Krebs cycle that important to respiration? (2 solutions)
Footer

Figure 4- Rough sketch of Dashboard page

different webpages. It will then have a main section in the middle where the page will display the questions and answers in an organised format showing the question title, the number of answers and the date it was created.


Search Results

Advanced Search Tips

Ask Question

Results for array

500 results

Relevance | Newest | More

25901 votes

27 answers

Q: Why is processing a sorted array faster than processing an unsorted array?

(Sorting itself takes more time than than this one pass over the array, so it's not actually worth doing if we needed to calculate this for an unknown array.) ... Why is processing a sorted array faster than processing an unsorted array? The code is summing up some independent terms, so the order should not matter. ...

java c++ performance cpu-architecture branch-prediction

asked Jun 27 '12 by GManNickG

9891 votes

112 answers

Q: How can I remove a specific item from an array?

I have an array of numbers and I'm using the .push() method to add elements to it. Is there a simple way to remove a specific element from an array? ...

javascript arrays

asked Apr 23 '11 by Walker

Figure 5- Dashboard/home page example on stack overflow

Key components

Name	Type	Justification
RegisterView()	Class	Allows the user to create unique account that when rendered in conjunction with html will display a form for account creation.
LoginView()	Class	Allows the user to login to an account that they created previously using the RegisterView class. It also redirects them to the homepage.
Questions()	Class	Manages the creation of questions that are linked to users.
Answers()	Class	Manages the creation and linking of answers to objects from the Questions class.
QuestionGroup()	Class	Manages the different topics that questions can fall into, for example respiration.

questions_list()	Function	Aids in the rendering of the homepage where all of the current questions are displayed.
questions_page()	Function	Aids in the rendering of each questions individual page.
db.sqlite3	Database file	Holds information about requests made by clients and the server to run the website.
search_view()	Function	Allows the user to search through topics and names in the database. It queries through these fields and displays links to the resulting questions.
question_form()	Function	Aids in the rendering of a form that allows users of the page to create their own questions that can be answered by other users.

Test Plan

Test No.	Test	Test Type	Test Data	Justification	Expected Output
1	Test fields on register page	Invalid	Leaving field blank	To test to see if the register form will still move data into the database without all required fields	The website, before sending data to Django will prevent the user posting the data to the database and from going forward to the dashboard. The website should display a popup box that prompts the user to enter data into the fields left open.
2	Test password on register page	Invalid	Entering a simple password with no numbers	To test to see if the website will flag an invalid password.	The website should show a popup signifying that the user has entered an invalid password and that they need to change it
3	Test login system on login page	Invalid	Entering incorrect data	To test if the website will prevent you from logging in to the site if you enter the wrong data.	The website, before sending data to Django will prevent the user posting the data to the database and from going forward to the dashboard. The website should display a popup box that prompts the user to enter data into the fields left open.
4	Test search bar	Invalid	Leaving search bar blank	This is to see if the search bar algorithm	The page should still render but instead of showing the results of the search it will print a message

				will realise that nothing has been searched and will output a suitable response for the user.	like "You didn't enter anything for this search," to tell the user what they had done.
5	Test search bar	extreme	Entering a topic that doesn't exist or isn't spelt right	This is to see if the that the algorithm can show that something was searched but didn't exist in the database	The page should render and should output text saying that "The item that that you searched isn't found or is spelt wrongly"
6	Test search bar	valid	Entering a topic that exists	This is to see that the search algorithm works and displays the correct questions to be accessed by the user.	The page should render and should output text saying, "Showing the results for search" and will show the questions that matched the topic of the search. It will also have links to each question page.
7	Test Question form	valid	All forms valid	This is to see if the form data is taken by the website and transferred to the database.	After the submit button is pressed on the form the user should be redirected and the form should appear in the admin section of the site. Also, the new question should appear on the homepage.
8	Test Question form	invalid	Description field empty on the form	This is to see if the form will reject an invalid form.	A popup should appear after the submit button is pressed saying that the empty field needs to be entered.
9	Test Question form	extreme	Title with more than 90 characters	This is to see if the form will reject an invalid form.	The form shouldn't let you input anymore characters once this threshold is reached.
10	Test Answer form	valid	All forms valid	To see if the Answer form transfers data to the database.	After submitting the form, the page should be refreshed and the answer that was entered should show up underneath the question.

Post Development Questionnaire

After the development process I am going to be using a questionnaire to assess my program, and see any improvements I will make, this is because a questionnaire is a very easy and useful way to gather my stakeholders' opinions and is in a similar style of gathering their requirements, so the stakeholders will be familiar and comfortable with this method of gathering their opinions. It can also give me a lot of qualitative data that I can take into account and can target areas of the program that I would like to improve with targeted questions, for example, asking about the graphics of the website and asking how to make them better.

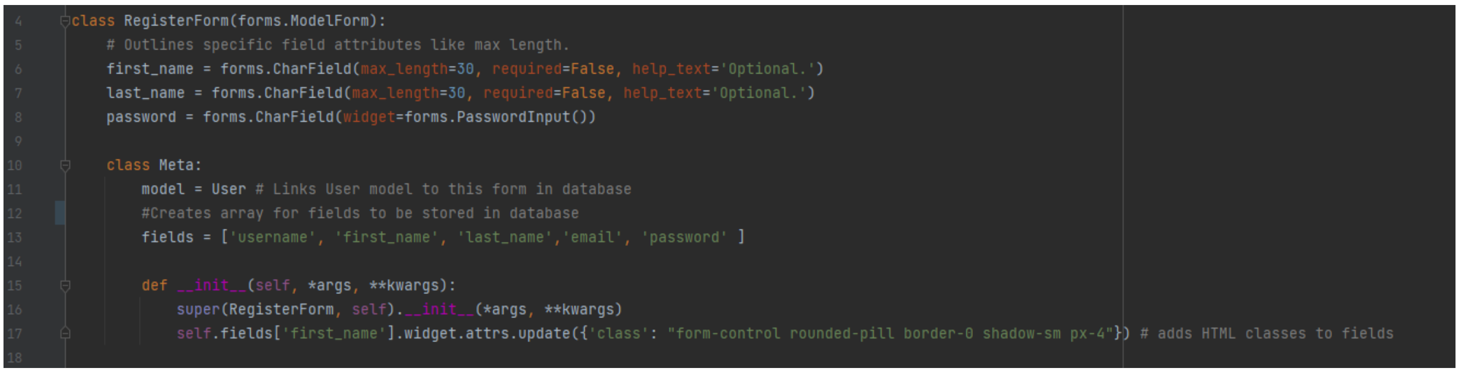
1) What do you think about the overall layout of the website?
2) How could the website function be improved?
3) How informative is the UI of the website?
4) Do you suggest any changes to the UI?

Development

First Prototype

To start, I first started developing the registration and login systems of the website.

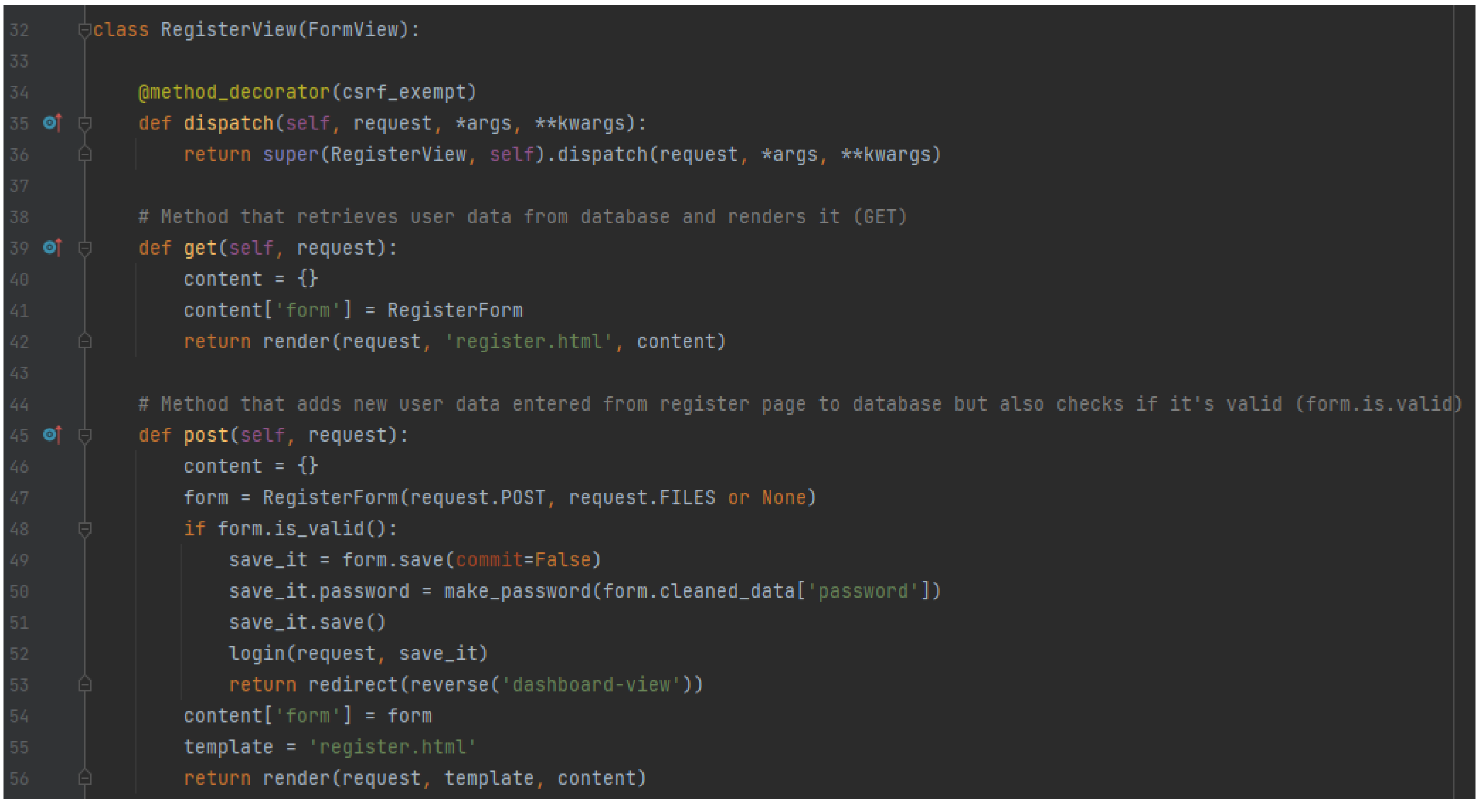
Registration page/backend



```python
class RegisterForm(forms.ModelForm):
    # Outlines specific field attributes like max length.
    first_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
    last_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
    password = forms.CharField(widget=forms.PasswordInput())

    class Meta:
        model = User # Links User model to this form in database
        #Creates array for fields to be stored in database
        fields = ['username', 'first_name', 'last_name','email', 'password' ]

        def __init__(self, *args, **kwargs):
            super(RegisterForm, self).__init__(*args, **kwargs)
            self.fields['first_name'].widget.attrs.update({'class': "form-control rounded-pill border-0 shadow-sm px-4"}) # adds HTML classes to fields

```

Figure 6- The form for the register page

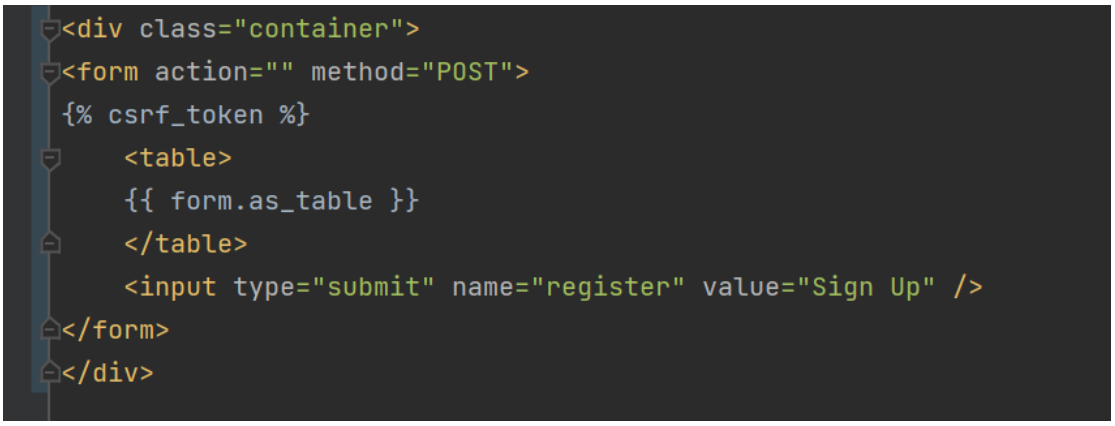
This register form class outlines the fields that each user will have like first name, last name and the password. This is then imported to the Register View where it is used.



```python
class RegisterView(FormView):

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super(RegisterView, self).dispatch(request, *args, **kwargs)

    # Method that retrieves user data from database and renders it (GET)
    def get(self, request):
        content = {}
        content['form'] = RegisterForm
        return render(request, 'register.html', content)

    # Method that adds new user data entered from register page to database but also checks if it's valid (form.is.valid)
    def post(self, request):
        content = {}
        form = RegisterForm(request.POST, request.FILES or None)
        if form.is_valid():
            save_it = form.save(commit=False)
            save_it.password = make_password(form.cleaned_data['password'])
            save_it.save()
            login(request, save_it)
            return redirect(reverse('dashboard-view'))
        content['form'] = form
        template = 'register.html'
        return render(request, template, content)
```

Figure 7- The view for the register page

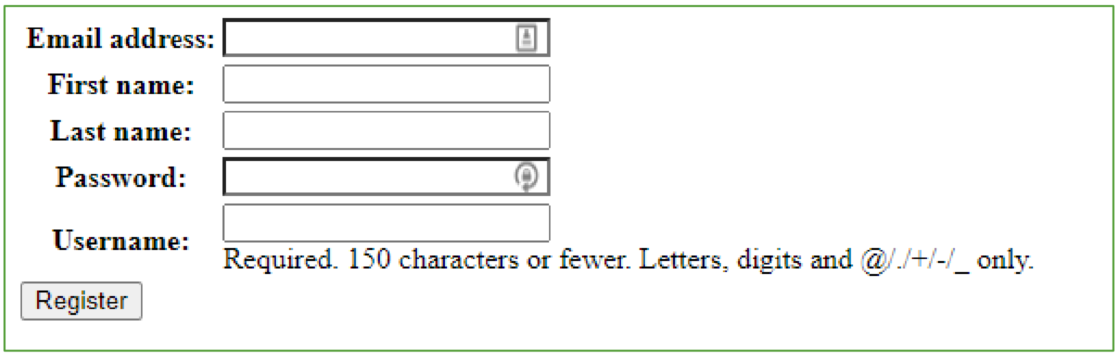
Here the view takes the form in figure 7 and renders it on the register page. This section of the code is called a GET function. Next in the POST function data that is entered by the user is processed to check if it is valid, for example a password being too short. If it is valid the user's details are stored within the database.



```html
<div class="container">
<form action="" method="POST">
{% csrf_token %}
    <table>
    {{ form.as_table }}
    </table>
    <input type="submit" name="register" value="Sign Up" />
</form>
</div>
```

Figure 8 - HTML template for register page

This HTML uses simple Django syntax called "form.as.table" to render the form as a HTML table.



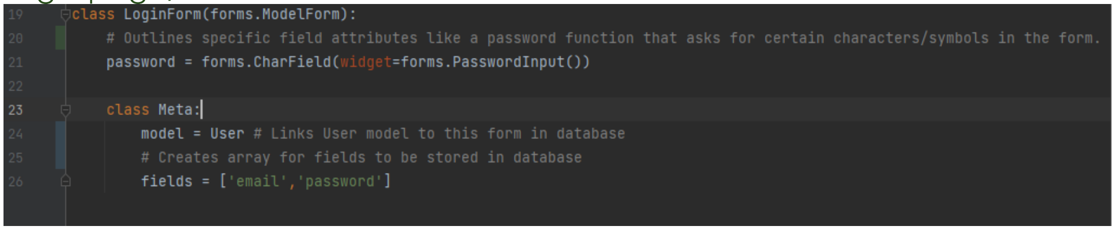
Email address:	
First name:	
Last name:	
Password:	
Username:	Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Register

Figure 9- simple register page displayed from the previous code combined

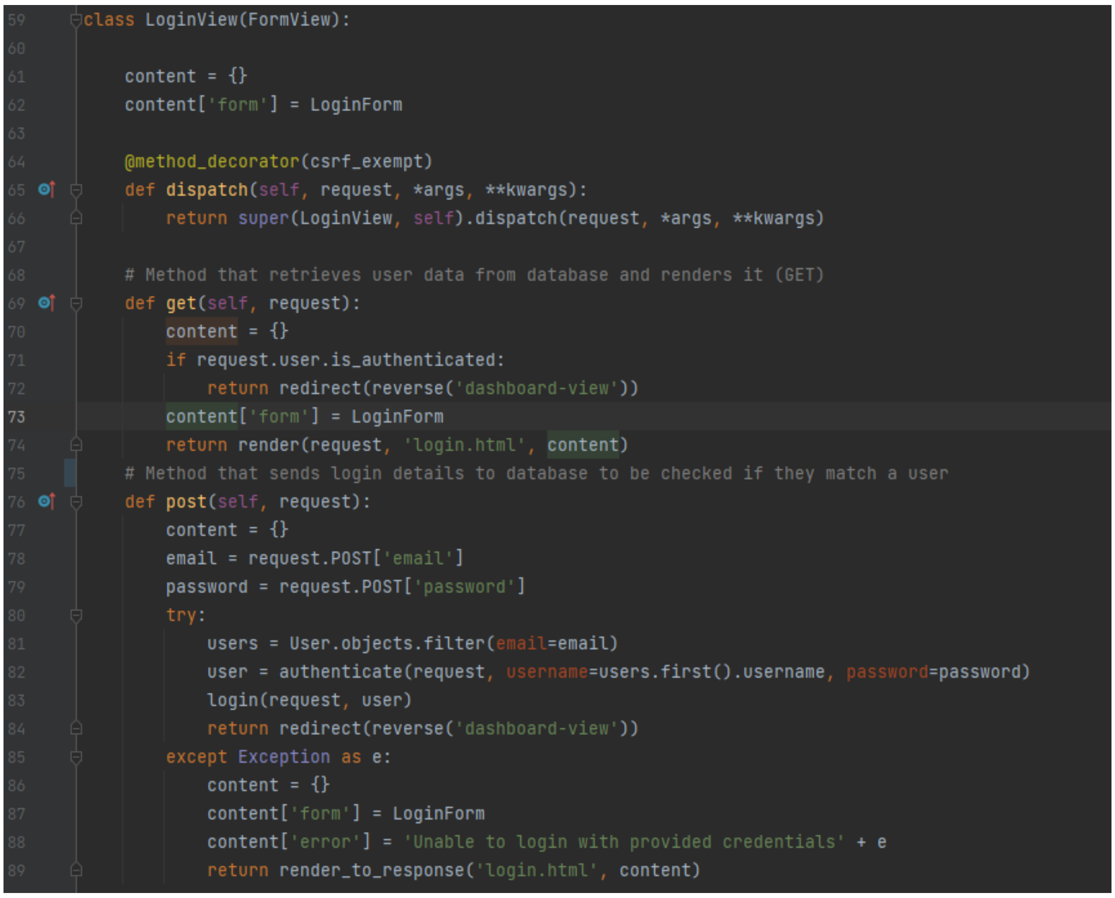
On this basic register page, you are able to enter your details like email, name and password. Through the use of the code above Django is able to POST (transfer) all of this data to the sites SQL database creating a new user object. This form is also able to prevent the data from being passed to the if certain criteria aren't met like for example if the password used invalid symbols, it would be rejected.
This register page in the first prototype doesn't visually look the best since I haven't included CSS in the html template that is rendered but also since I prioritised the function of the page first before the look of it. I can build on the look in future prototypes.

Login page/backend



```
class LoginForm(forms.ModelForm):
    # Outlines specific field attributes like a password function that asks for certain characters/symbols in the form.
    password = forms.CharField(widget=forms.PasswordInput())

    class Meta:
        model = User # Links User model to this form in database
        # Creates array for fields to be stored in database
        fields = ['email','password']
```

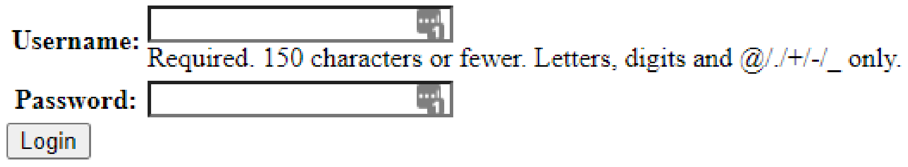
Figure 10- The form for the login page



```python
class LoginView(FormView):

    content = {}
    content['form'] = LoginForm

    @method_decorator(csrf_exempt)
    def dispatch(self, request, *args, **kwargs):
        return super(LoginView, self).dispatch(request, *args, **kwargs)

    # Method that retrieves user data from database and renders it (GET)
    def get(self, request):
        content = {}
        if request.user.is_authenticated:
            return redirect(reverse('dashboard-view'))
        content['form'] = LoginForm
        return render(request, 'login.html', content)
    # Method that sends login details to database to be checked if they match a user
    def post(self, request):
        content = {}
        email = request.POST['email']
        password = request.POST['password']
        try:
            users = User.objects.filter(email=email)
            user = authenticate(request, username=users.first().username, password=password)
            login(request, user)
            return redirect(reverse('dashboard-view'))
        except Exception as e:
            content = {}
            content['form'] = LoginForm
            content['error'] = 'Unable to login with provided credentials' + e
            return render_to_response('login.html', content)
```

Figure 11 - The view for the login page

The GET function here retrieves the form and renders it when the page is started. The POST function takes the information the user entered to login and sends it to the database to be checked.



Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

Login

Figure 12 - simple login page rendered from the previous code

On this basic login page, you are able to enter your details like email and password. Through the use of the code above Django is able to GET (retrieve) the user data for your account by scanning through the SQL database to find. This form is also able to prevent the data from being passed to the if certain criteria aren't met like for example if the password used invalid symbols, it would be rejected.
This register page in the first prototype doesn't visually look the best since I haven't included CSS in the html template that is rendered but also since I prioritised the function of the page first before the look of it. I can build on the look in future prototypes.

Also, to start the project I created a new file for my project called Coursework where I created a virtual python environment and then installed Django into the directory. From there I was able to create my Django project.

Alpha Testing for Prototype 1

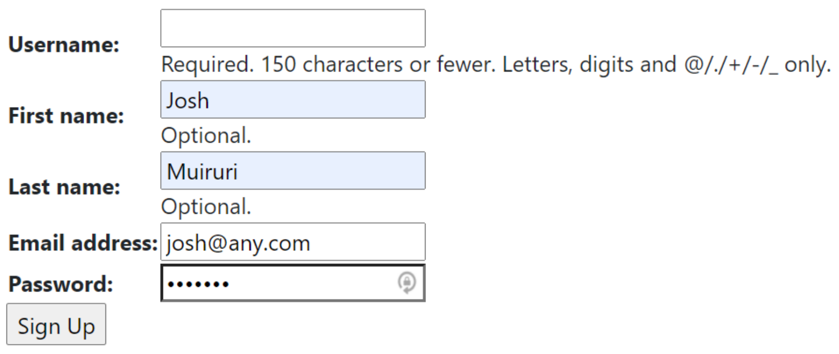
I am initially doing testing to see if the register and login pages function correctly, allowing the entering of data/information but also if they reject invalid data from being entered.

Test if register page allows multiple usernames of the same name

Here I am testing to see if the website will prevent a user being created with the same username as someone else.

Test if register page prevents invalid data

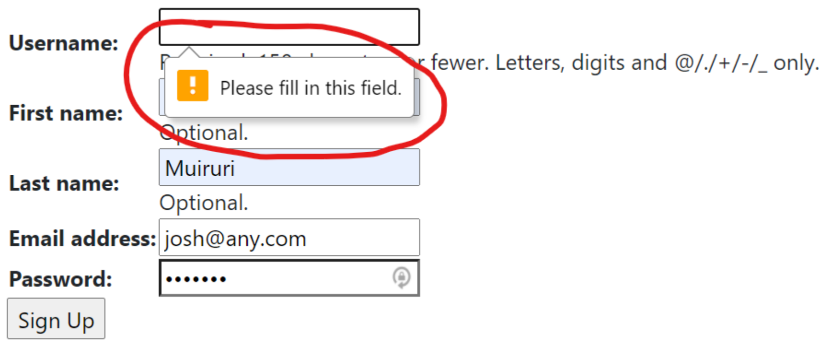
Here I am testing to see if the register page prevents invalid data, in this case blank fields from being transferred to the database. The page should show a popup box to enter the required field. Also, I slightly altered the register form (Figure 4) to be more logically ordered as shown below:



Username:	Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
First name:	Josh Optional.
Last name:	Muiruri Optional.
Email address:	josh@any.com
Password:	•••••••

Figure 13 - Invalid data test image 1

In the image above I have opened the register page and have entered information into all the fields except for the username field. I will then click the Sign-up button to attempt to send the data to the database.

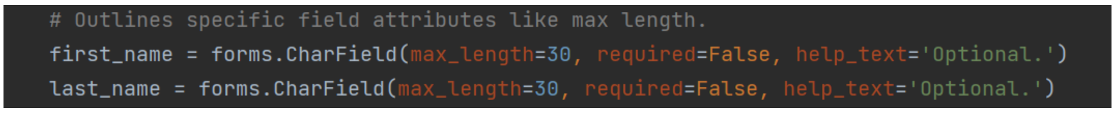


Username:	r fewer. Letters, digits and @/./+/-/_ only.
First name:	Optional.
Last name:	Muiruri Optional.
Email address:	josh@any.com
Password:	•••••••

Please fill in this field.

Figure 14 - Invalid data test image 2

After I clicked the button, the result was as follows. A popup window/warning message appearing below the empty field telling the user to fill in the field. This test applies to all basic fields except for first name and last name since in the form code I set it so that they weren't required:

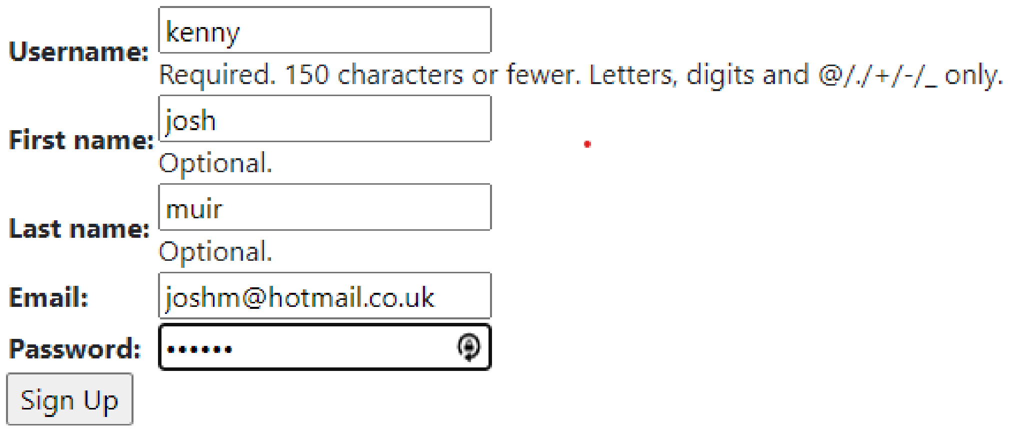


```
    # Outlines specific field attributes like max length.
    first_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
    last_name = forms.CharField(max_length=30, required=False, help_text='Optional.')
```

This test was a success.

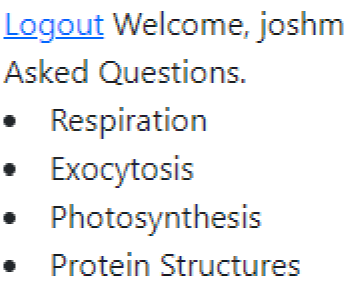
Test if register page prevents invalid password

Essentially, I'm going to carry out the same test as above except instead of leaving a field blank, I am going to enter data into all the fields, but instead the password is going to be invalid. The password is supposed to be more than 8 characters so I will enter a password without with 6 characters (joshmu). The website should print an error popup.



Username:	kenny Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
First name:	josh Optional.
Last name:	muir Optional.
Email:	joshm@hotmail.co.uk
Password:	••••••

Figure 15 - Invalid password test



Logout Welcome, joshm

Asked Questions.

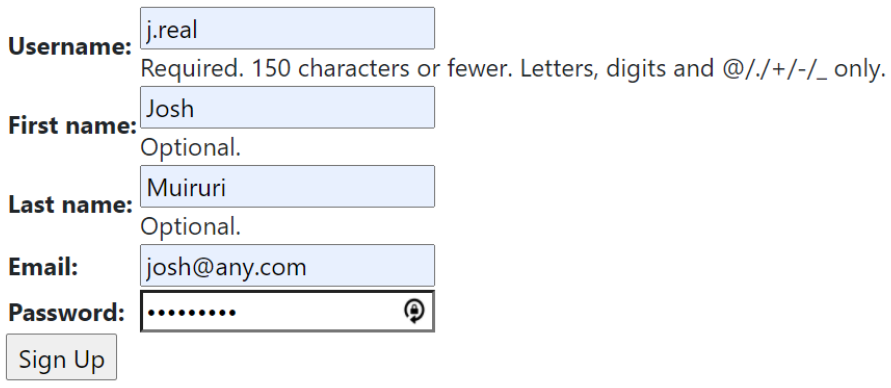
- Respiration
- Exocytosis
- Photosynthesis
- Protein Structures

Figure 16 - Invalid password test pt2

Instead of stopping the user from being registered they were added to the database and were redirected to a mock-up homepage.
This test was a failure

Test if register page posts details to database

In this test I will be testing to see if the information I passed into the register page was successfully transferred into the database. To check this, I will use the hidden admin section that comes with Django. There I will be able to view if the new user was added. Also, I will check the command-line of the server in windows PowerShell for further validation.

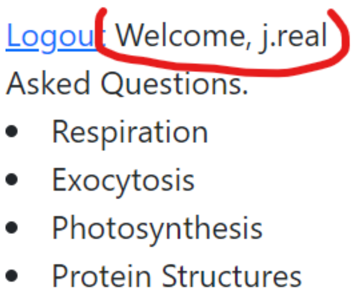


Username:	j.real Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
First name:	Josh Optional.
Last name:	Muiruri Optional.
Email:	josh@any.com
Password:	•••••••••

Sign Up

Figure 17 - Valid data entered into register page

The password entered is "passw0rd1" which is valid.



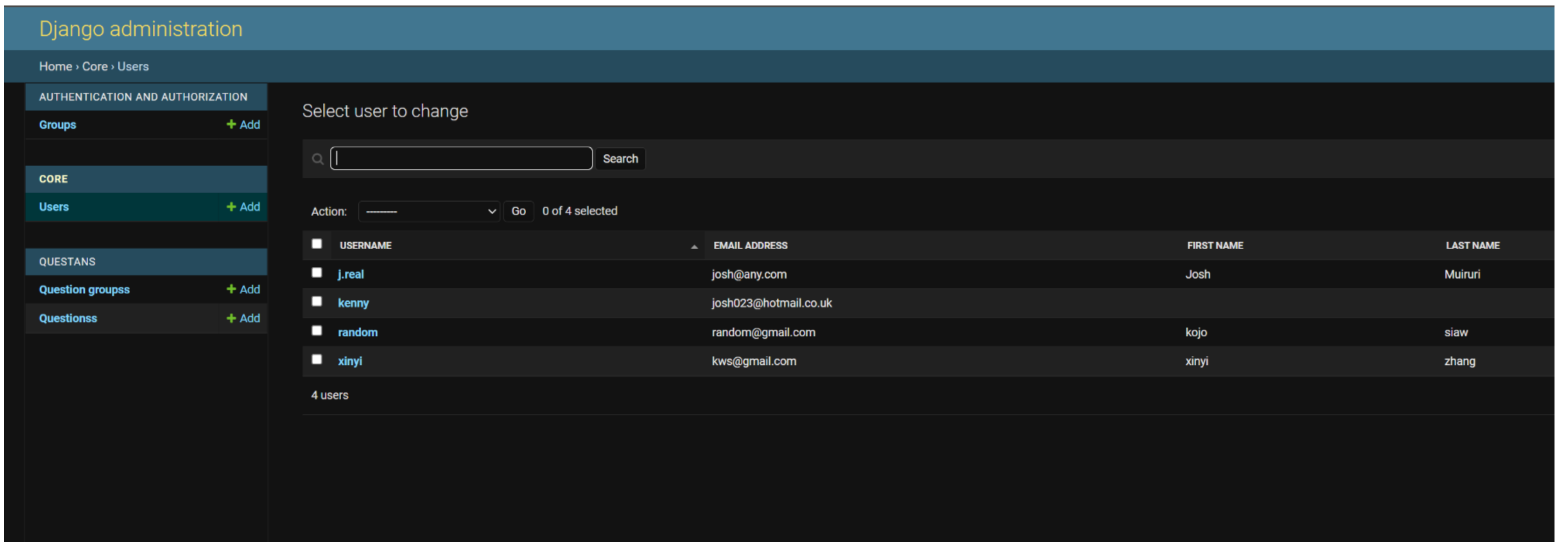
Logout Welcome, j.real

Asked Questions.

- Respiration
- Exocytosis
- Photosynthesis
- Protein Structures

Figure 18 - Result after registering

After pressing the Sign-up button the website redirected me to the homepage where it showed proof that the user was created since it output "Welcome, j.real" which is the username I inputted.



Django administration

Home › Core › Users

Select user to change

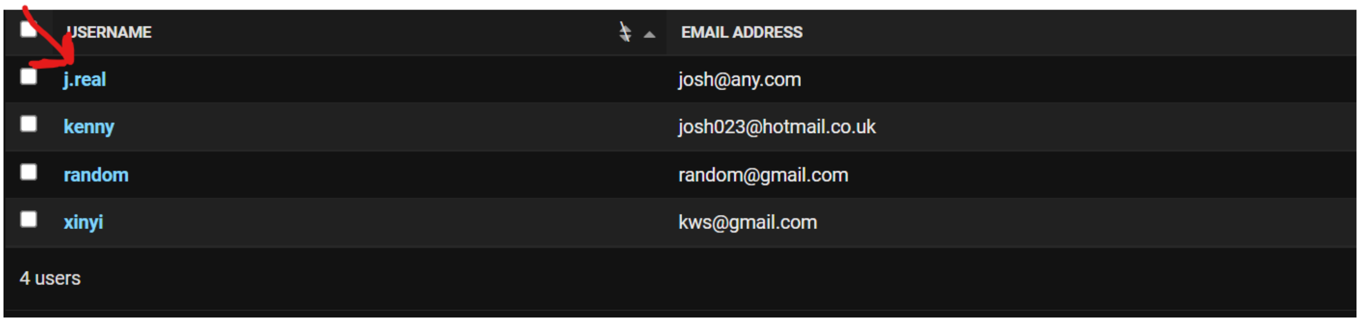
Search

Action: --------- Go 0 of 4 selected

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME
j.real	josh@any.com	Josh	Muiruri
kenny	josh023@hotmail.co.uk		
random	random@gmail.com	kojo	siaw
xinyi	kws@gmail.com	xinyi	zhang

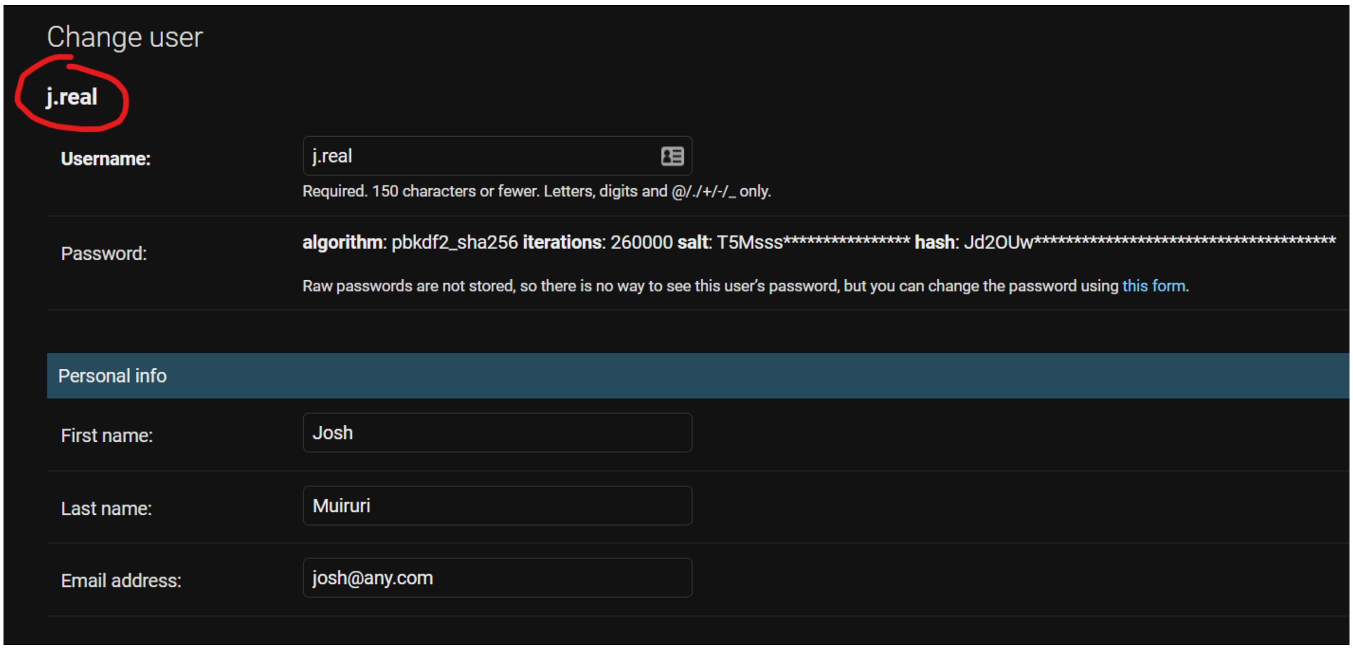
4 users

Figure 19 - Django admin page



USERNAME	EMAIL ADDRESS
j.real	josh@any.com
kenny	josh023@hotmail.co.uk
random	random@gmail.com
xinyi	kws@gmail.com

4 users



Change user

j.real

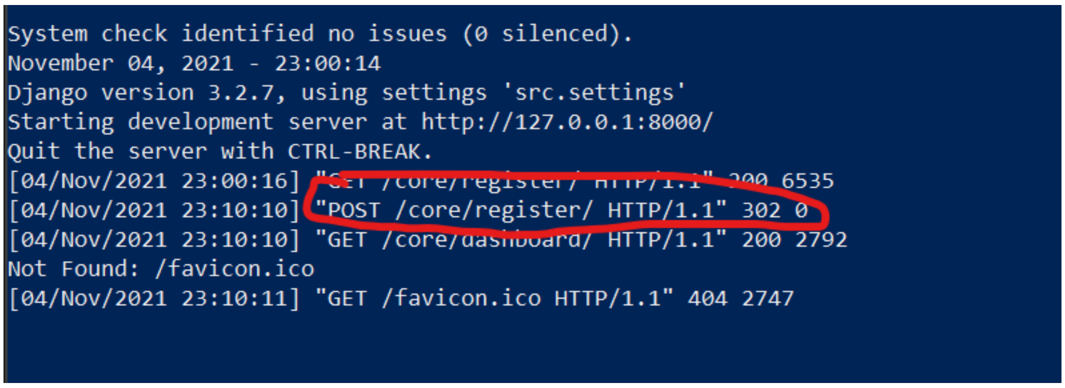
Username:	j.real Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:	**algorithm**: pbkdf2_sha256 **iterations**: 260000 **salt**: T5Msss**************** **hash**: Jd2OUw************************************** Raw passwords are not stored, so there is no way to see this user's password, but you can change the password using this form.

Personal info

First name:	Josh
Last name:	Muiruri
Email address:	josh@any.com

Figure 20 - Django admin page

In these three images the Django admin page shows proof that the data entered has been added to the database as a new user object. The user is listed with the other users in admin. You are able to login on the login page using this users' credentials now. The password is also hashed to improve security.



```
System check identified no issues (0 silenced).
November 04, 2021 - 23:00:14
Django version 3.2.7, using settings 'src.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[04/Nov/2021 23:00:16] "GET /core/register/ HTTP/1.1" 200 6535
[04/Nov/2021 23:10:10] "POST /core/register/ HTTP/1.1" 302 0
[04/Nov/2021 23:10:10] "GET /core/dashboard/ HTTP/1.1" 200 2792
Not Found: /favicon.ico
[04/Nov/2021 23:10:11] "GET /favicon.ico HTTP/1.1" 404 2747
```

Figure 21 - image of PowerShell command prompt

Here the server command prompt shows that the data was POSTED to the database.

This test was a success

What requirements have been fulfilled?

This prototype fulfils the 3rd general requirement which was allowing users of the website to create a personal user profile where they can store and answer questions. Even though I haven't completed the second part of this, I will be able to build on it in future prototypes.

Also, this prototype fulfils the 1st general requirement which is for the website to be able to work on a range of different web browsers. Since the page is rendered in HTML it can be rendered on all popular browsers like Chrome, Edge, Explorer and Firefox.

What do the Stakeholders say?

With the feedback given the majority of my stakeholders said that they like the fact that they can make a unique account since allows them to access their questions and data from anywhere. Although the stakeholders that are students (Stephen and Kwame) would like the website to visually look better because it looks very simple and difficult to navigate. Therefore, in the next prototype I aim to improve how the website looks and navigates by fulfilling my 5th and 6th general requirements.

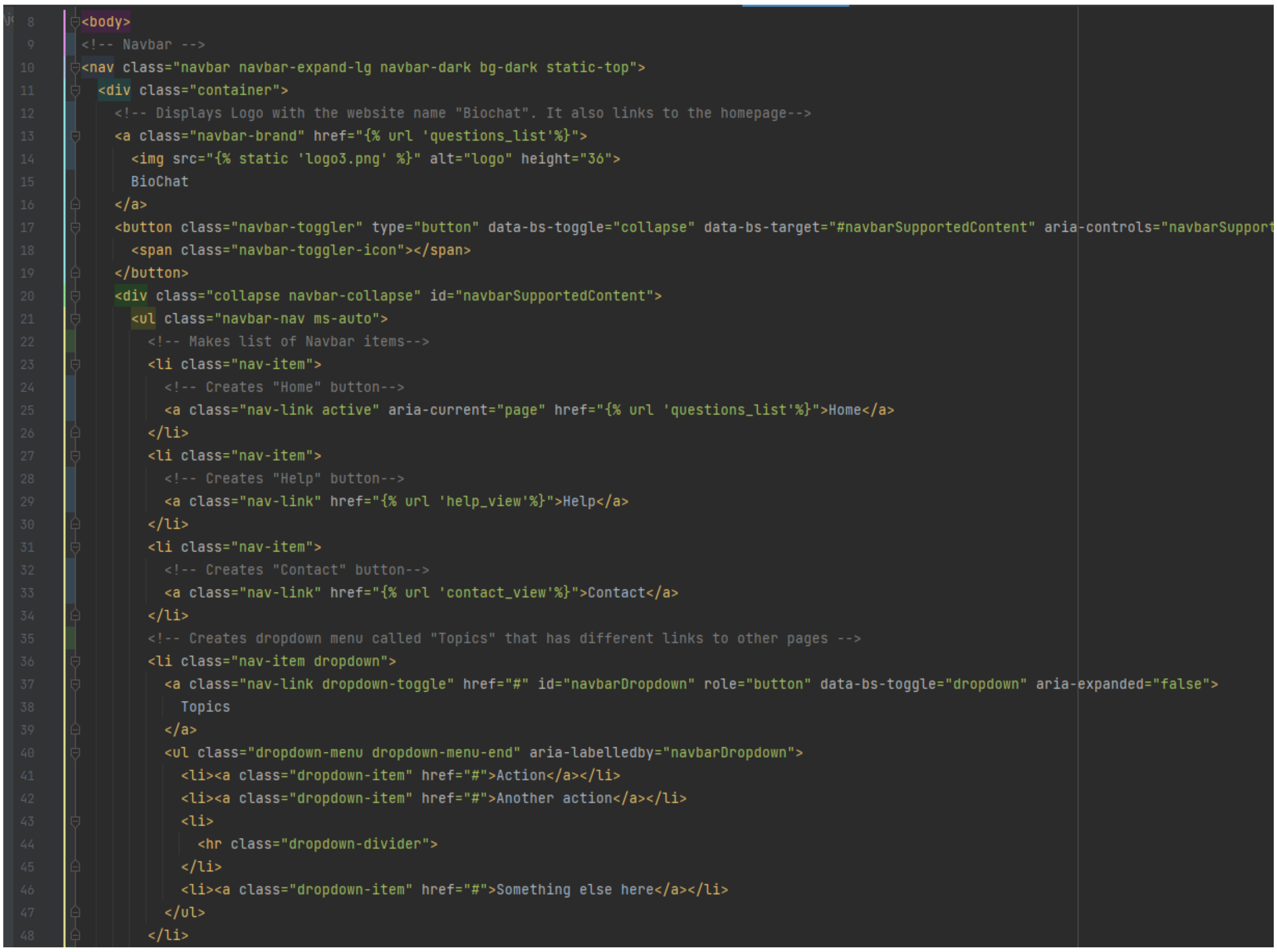
What to improve for this prototype:

8. The graphics right now are only placeholders for how the final product will look since currently I'm only focusing on the functionality of the website first. So, in a later prototype I will improve the graphics
9. Adding a toolbar since it is a key part of website allowing the user to navigate around between pages easily.
10. In developing the taskbar, I will also include a working search bar that searches through all the current questions and displays the result.
11. I need to implement the main part of the site where questions and answers are displayed but also you can create and answer them.

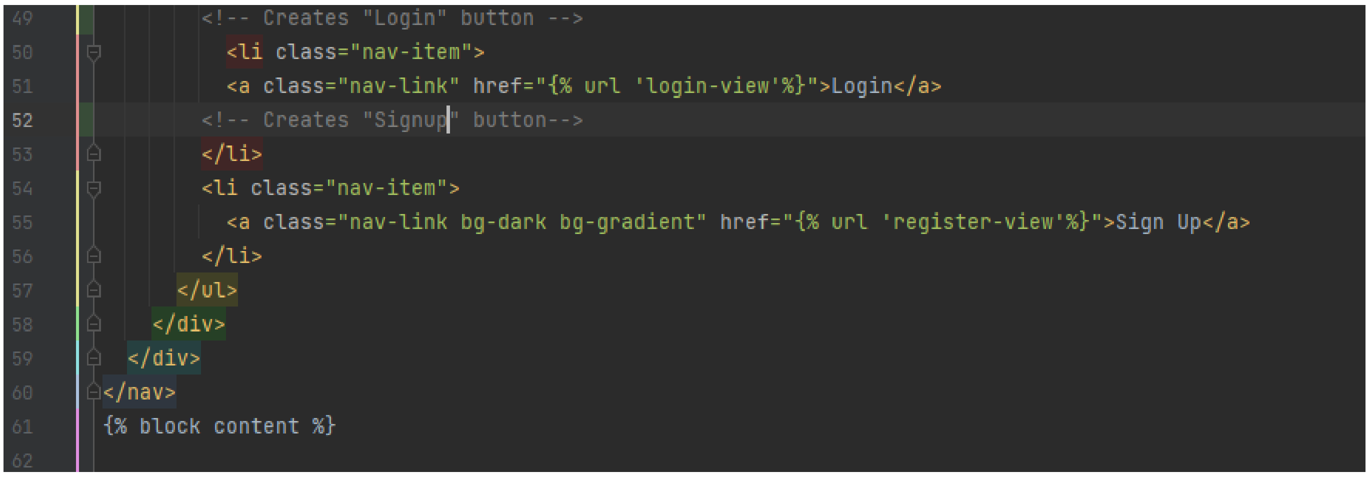
Second Prototype

In this prototype I've started developing the toolbar for the website making other pages easier navigable.

Toolbar



```
<body>
<!-- Navbar -->
<nav class="navbar navbar-expand-lg navbar-dark bg-dark static-top">
  <div class="container">
    <!-- Displays Logo with the website name "Biochat". It also links to the homepage-->
    <a class="navbar-brand" href="{% url 'questions_list'%}">
      <img src="{% static 'logo3.png' %}" alt="logo" height="36">
      BioChat
    </a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupport
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav ms-auto">
        <!-- Makes list of Navbar items-->
        <li class="nav-item">
          <!-- Creates "Home" button-->
          <a class="nav-link active" aria-current="page" href="{% url 'questions_list'%}">Home</a>
        </li>
        <li class="nav-item">
          <!-- Creates "Help" button-->
          <a class="nav-link" href="{% url 'help_view'%}">Help</a>
        </li>
        <li class="nav-item">
          <!-- Creates "Contact" button-->
          <a class="nav-link" href="{% url 'contact_view'%}">Contact</a>
        </li>
        <!-- Creates dropdown menu called "Topics" that has different links to other pages -->
        <li class="nav-item dropdown">
          <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-bs-toggle="dropdown" aria-expanded="false">
            Topics
          </a>
          <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="#">Action</a></li>
            <li><a class="dropdown-item" href="#">Another action</a></li>
            <li>
              <hr class="dropdown-divider">
            </li>
            <li><a class="dropdown-item" href="#">Something else here</a></li>
          </ul>
        </li>
```



```
        <!-- Creates "Login" button -->
          <li class="nav-item">
          <a class="nav-link" href="{% url 'login-view'%}">Login</a>
        <!-- Creates "Signup" button-->
        </li>
        <li class="nav-item">
          <a class="nav-link bg-dark bg-gradient" href="{% url 'register-view'%}">Sign Up</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
{% block content %}
```

Figure 22 - HTML for Toolbar

When rendered, this HTML code creates a navigation bar on every single page of the website allowing user to navigate throughout the website to all the pages they need. The buttons created in the navigation bar use Django syntax to link to different urls in the .urls file as shown in the previous prototype. This is shown rendered in the next image.

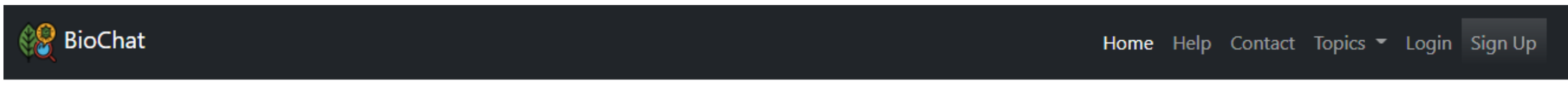


Figure 24 - Image of displayed navigation bar

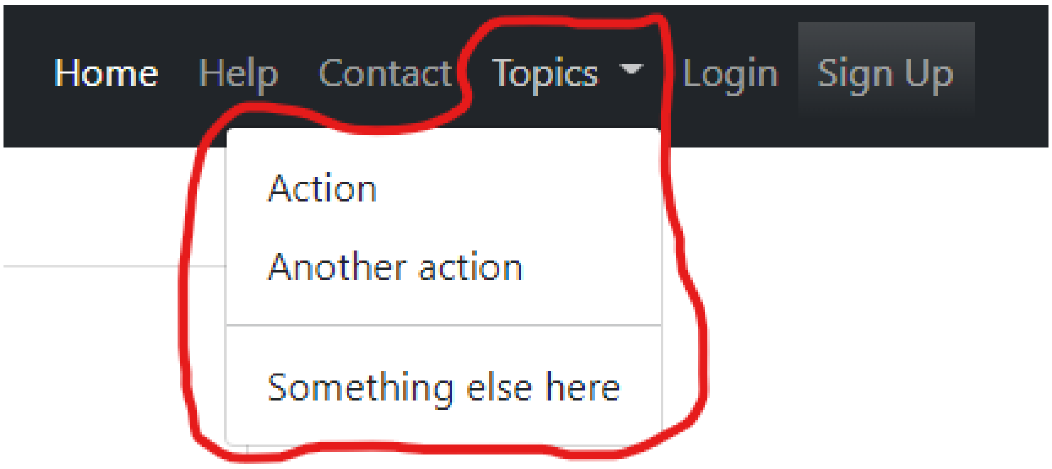
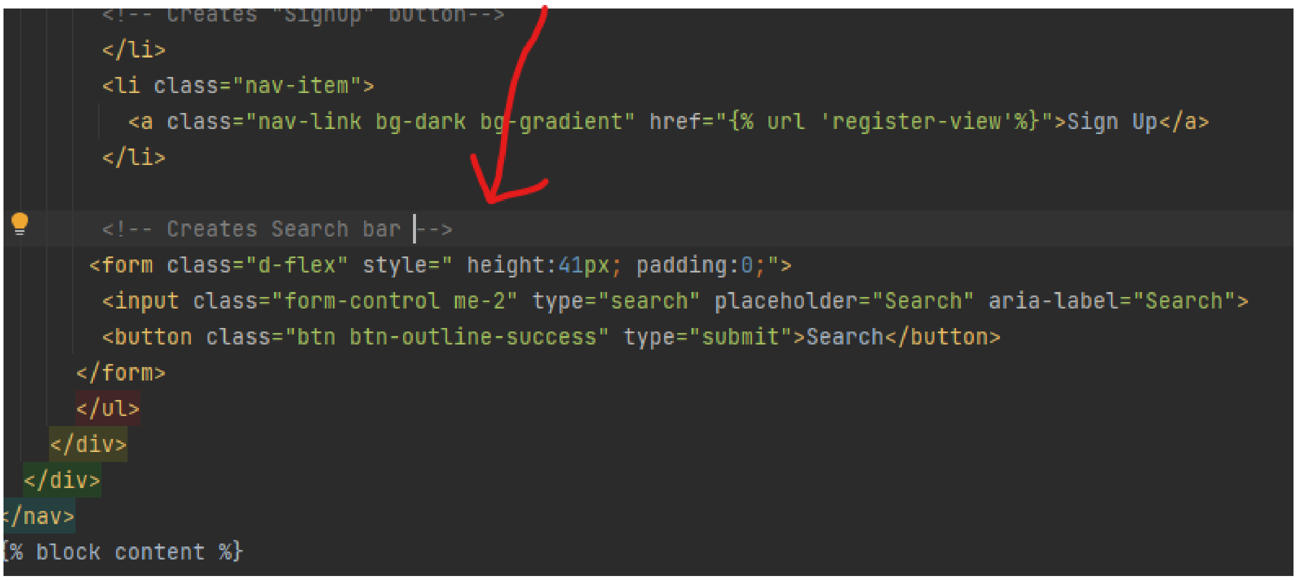


Figure 23 - Image of dropdown menu

I haven't added the search bar functionality so I will do that next.

Search bar

Next, I coded the HTML for the search bar which I placed at the end of navbar code so that it's out of the way. When fully integrated with the rest of my Django project the website the search bar will first take an input of a topic that you want to search for. Django will then go to the database of questions and search for items that match the topic inputted. When the search button is pressed, it will link to a page that displays a query for the questions.



```
        <!-- Creates "Signup" button-->
        </li>
        <li class="nav-item">
          <a class="nav-link bg-dark bg-gradient" href="{% url 'register-view'%}">Sign Up</a>
        </li>

        <!-- Creates Search bar -->
       <form class="d-flex" style=" height:41px; padding:0;">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
      </ul>
    </div>
  </div>
</nav>
{% block content %}
```

Figure 26 - HTML for search bar

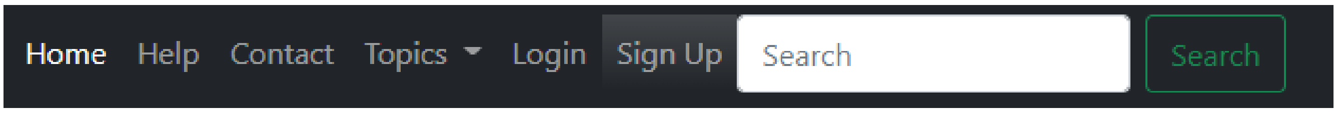
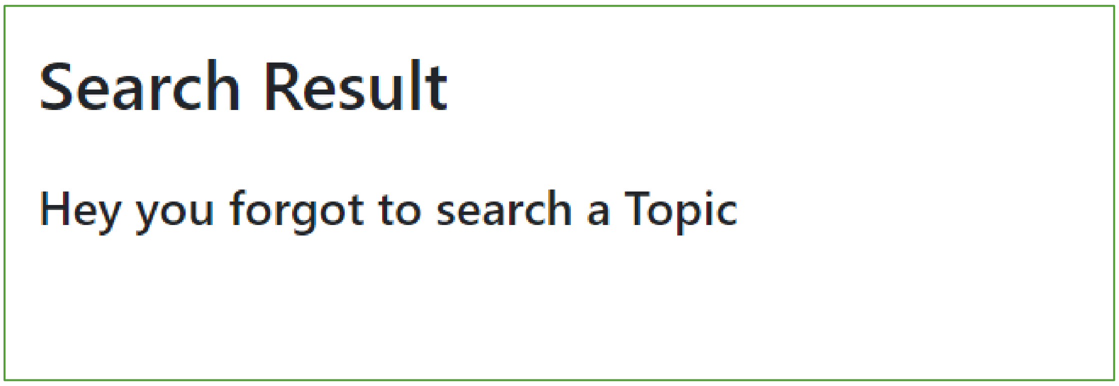


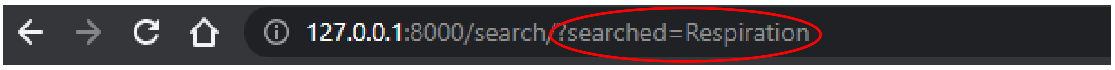
Figure 25 - Image of search bar added

Currently I have attempted to implement the backend of the search bar to make it send you to a different page and display the results of the search. Unfortunately, right now it is not working even though I entered something and gives me this error:



Search Result

Hey you forgot to search a Topic



127.0.0.1:8000/search/?searched=Respiration

Figure 27 - Proof that something was entered

What requirements were fulfilled?

This prototype fulfilled the 5th and 6th general requirements which was first to create a toolbar that links to most of the main page like home, login, sign up and others. Secondly the 6th requirement was to create a search bar that would search through all of the questions and return a filtered result. Although this is only partially met since the search bar isn't completely functional.

This prototype fulfils the 3rd general requirement which was allowing users of the website to create a personal user profile where they can store and answer questions. Even though I haven't completed the second part of this, I will be able to build on it in future prototypes.

Also, this prototype fulfils the 1st general requirement which is for the website to be able to work on a range of different web browsers. Since the page is rendered in HTML it can be rendered on all popular browsers like Chrome, Edge, Explorer and Firefox.

What do the stakeholders say?

12. Most of the stakeholders say that they are happy with the progress that is being made on the website.
13. The improvement to the graphical interface when adding the toolbar has made the page more user friendly.
14. The site is responsive when moving between pages
15. The site is still useless since the question-and-answer function hasn't been implemented yet.
16. Currently the search bar is useless.

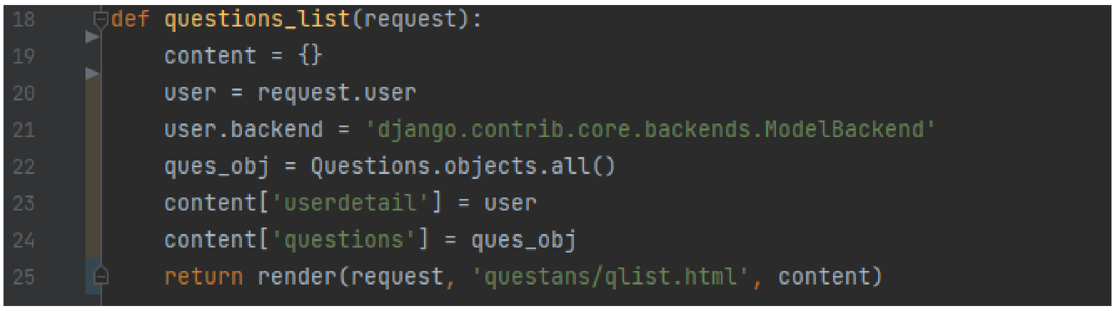
What to improve for this prototype:

The main focus is to implement and code the question-and-answer function of the website. After this is done the main section of the site will be complete and any other further improvements will be minor or graphical.

Third Prototype

Questions

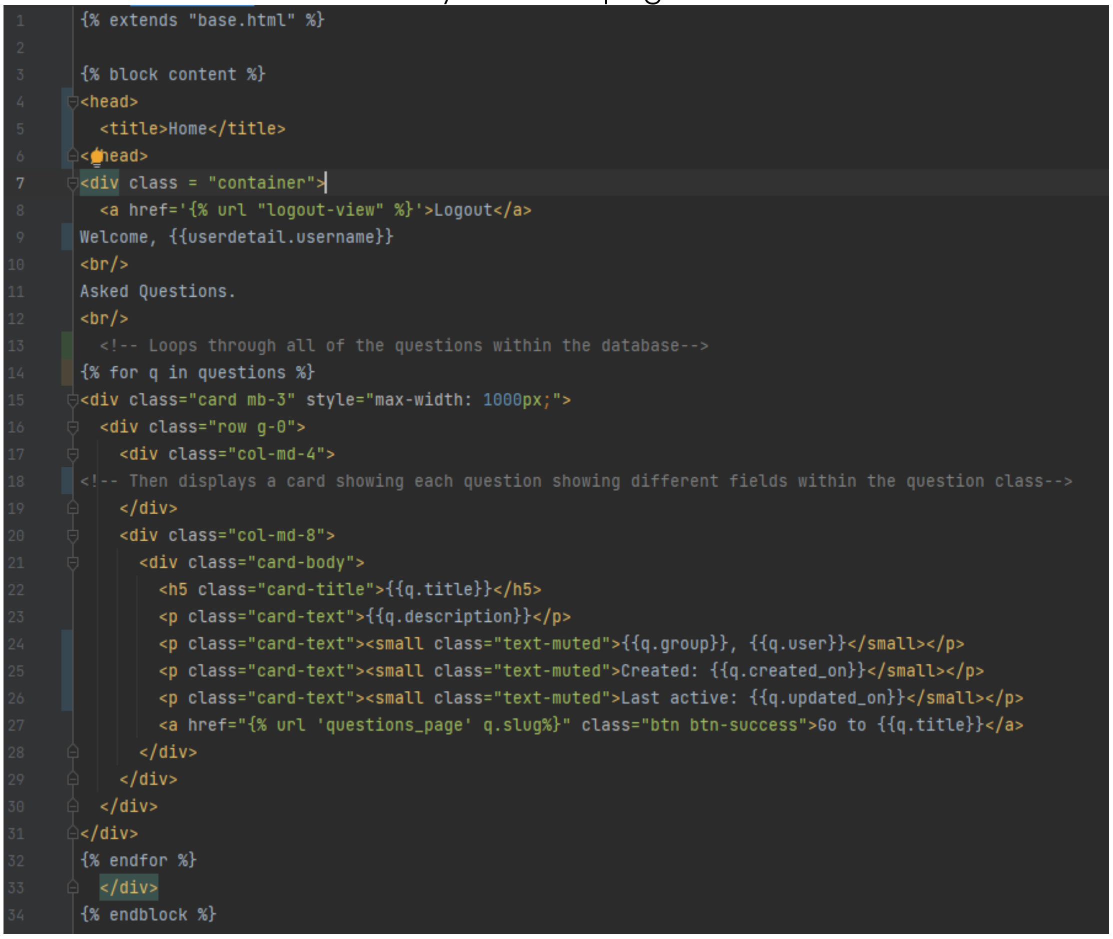
In this final prototype I am addressing the main feature of this site which is to create questions and answers. This will address the homepage which will display all of the most viewed questions and will have links to each question.



```python
def questions_list(request):
    content = {}
    user = request.user
    user.backend = 'django.contrib.core.backends.ModelBackend'
    ques_obj = Questions.objects.all()
    content['userdetail'] = user
    content['questions'] = ques_obj
    return render(request, 'questans/qlist.html', content)
```

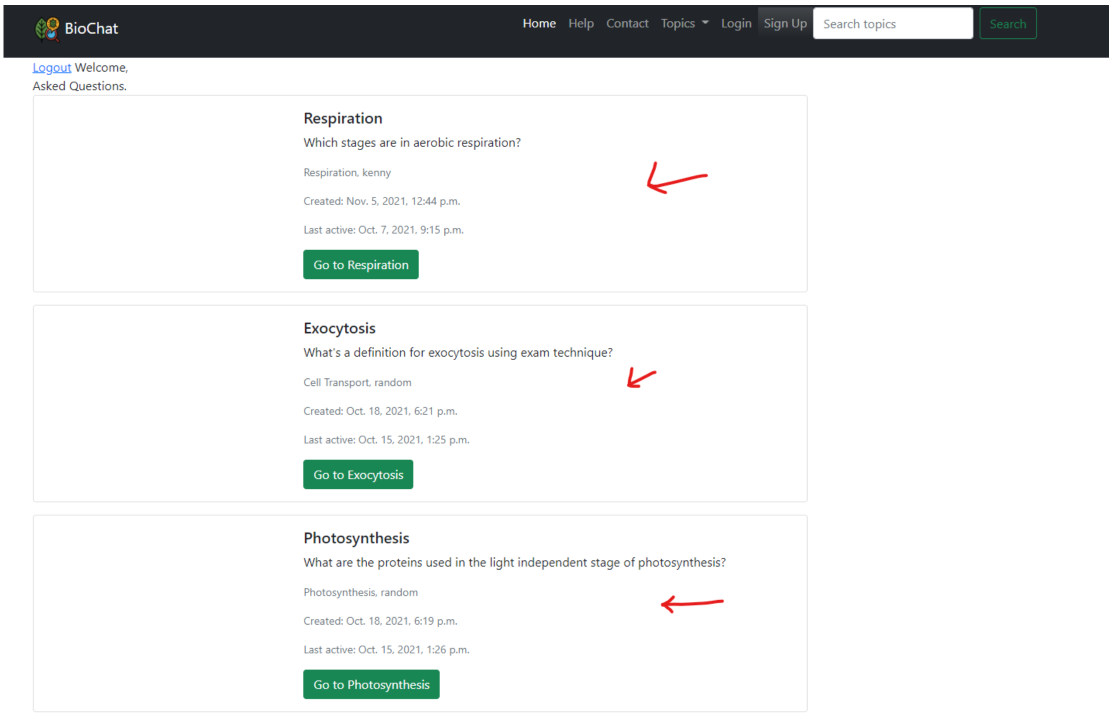
Figure 28 - view for Homepage

This function imports details from the questions class stored in the database and allows it to be accessed by the homepage.



```html
{% extends "base.html" %}

{% block content %}
<head>
  <title>Home</title>
</head>
<div class = "container">
  <a href='{% url "logout-view" %}'>Logout</a>
Welcome, {{userdetail.username}}
<br/>
Asked Questions.
<br/>
  <!-- Loops through all of the questions within the database-->
{% for q in questions %}
<div class="card mb-3" style="max-width: 1000px;">
  <div class="row g-0">
    <div class="col-md-4">
<!-- Then displays a card showing each question showing different fields within the question class-->
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">{{q.title}}</h5>
        <p class="card-text">{{q.description}}</p>
        <p class="card-text"><small class="text-muted">{{q.group}}, {{q.user}}</small></p>
        <p class="card-text"><small class="text-muted">Created: {{q.created_on}}</small></p>
        <p class="card-text"><small class="text-muted">Last active: {{q.updated_on}}</small></p>
        <a href="{% url 'questions_page' q.slug%}" class="btn btn-success">Go to {{q.title}}</a>
      </div>
    </div>
  </div>
</div>
{% endfor %}
  </div>
{% endblock %}
```

Figure 29 - HTML for Homepage



Logout Welcome,
Asked Questions.

Respiration
Which stages are in aerobic respiration?
Respiration, kenny
Created: Nov. 5, 2021, 12:44 p.m.
Last active: Oct. 7, 2021, 9:15 p.m.
Go to Respiration

Exocytosis
What's a definition for exocytosis using exam technique?
Cell Transport, random
Created: Oct. 18, 2021, 6:21 p.m.
Last active: Oct. 15, 2021, 1:25 p.m.
Go to Exocytosis

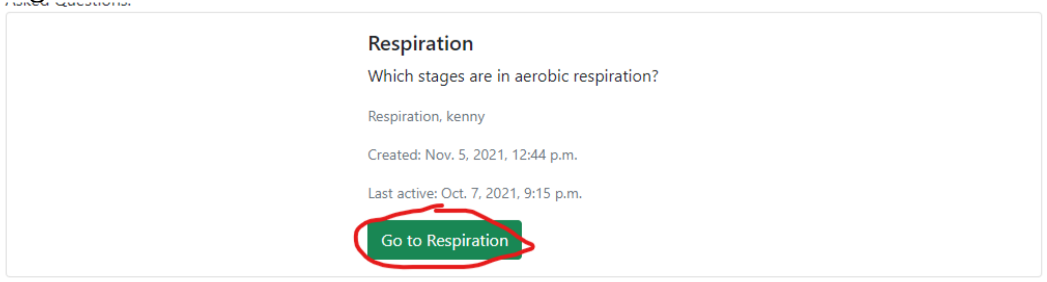
Photosynthesis
What are the proteins used in the light independent stage of photosynthesis?
Photosynthesis, random
Created: Oct. 18, 2021, 6:19 p.m.
Last active: Oct. 15, 2021, 1:26 p.m.
Go to Photosynthesis

Figure 30 - Rendered page from HTML

The combination of the view and HTML page results in a webpage being displayed. On this page is the toolbar with all its links and the question cards which display the title and description of the card defined in the database followed by a green button linking to the slug page for the given question. These questions are displayed in order of popularity in terms of clicks. Although it appears that the function that retrieves the users name failed even though I was logged in at this point.

Test if question buttons work

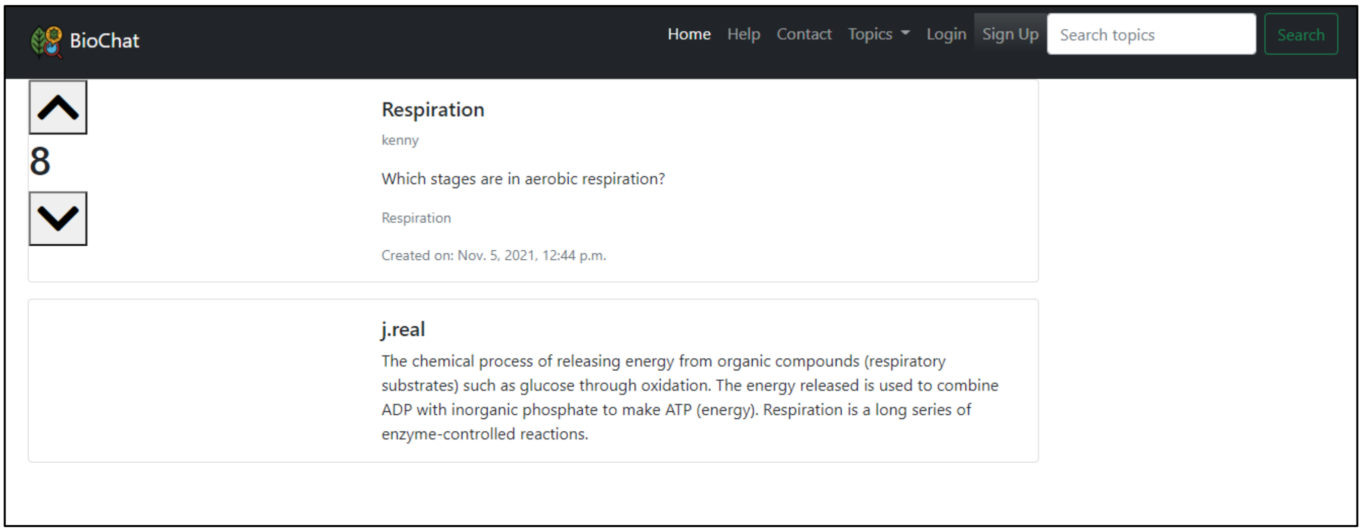
Here I am testing to see if the question buttons correctly link to the right pages.



Respiration
Which stages are in aerobic respiration?
Respiration, kenny
Created: Nov. 5, 2021, 12:44 p.m.
Last active: Oct. 7, 2021, 9:15 p.m.
Go to Respiration

Figure 31 - Button test1

When pressed this button should link to the respiration question page which I am going to show after this test.



Respiration

kenny

Which stages are in aerobic respiration?

Respiration

Created on: Nov. 5, 2021, 12:44 p.m.

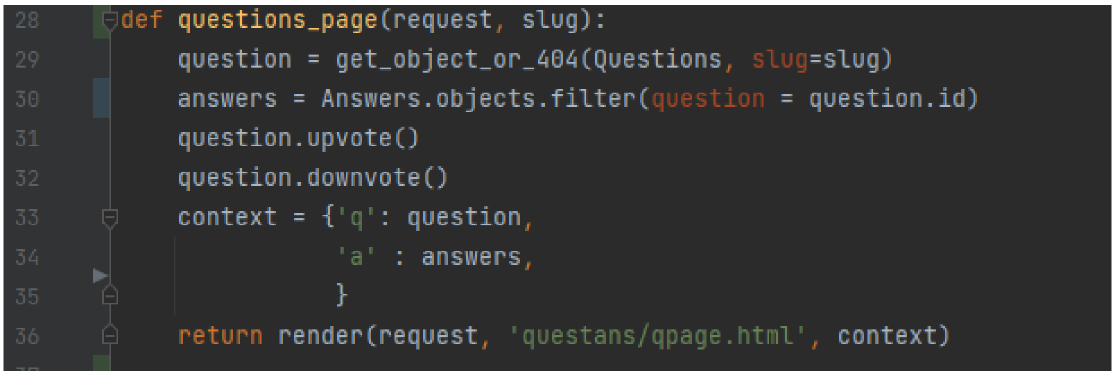
j.real

The chemical process of releasing energy from organic compounds (respiratory substrates) such as glucose through oxidation. The energy released is used to combine ADP with inorganic phosphate to make ATP (energy). Respiration is a long series of enzyme-controlled reactions.

After the button was pressed, the website was redirected to the question page for respiration.
This makes this test a success

Question page

Here I am developing the question page which will link off each individual question on the homepage. This question page will contain all details of the question like the description and user but will also include answers to the question submitted by other users.

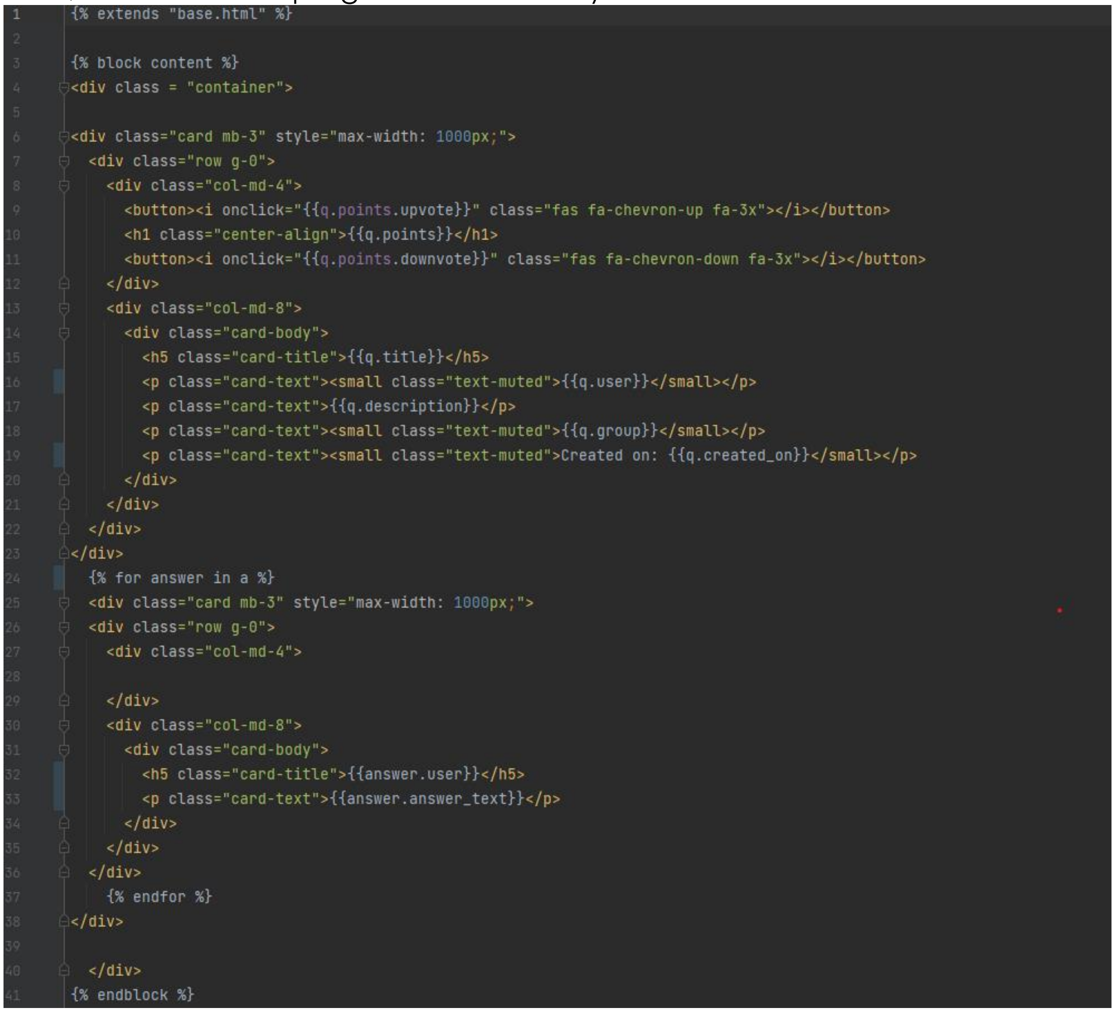


```
def questions_page(request, slug):
    question = get_object_or_404(Questions, slug=slug)
    answers = Answers.objects.filter(question = question.id)
    question.upvote()
    question.downvote()
    context = {'q': question,
               'a' : answers,
               }
    return render(request, 'questans/qpage.html', context)
```

Figure 32 - View for the Question page

This function takes in a slug as a parameter which is a variable within the question class. The slug acts a way to identify each individual question object in order to add it to the end of the web link. The function then finds the question that this slug links to and gives the web page access to all of its details. This allows the page to render out all the question's details like title, topic, description, user and start date.
The function also finds any answers that were made for this question and renders their content.

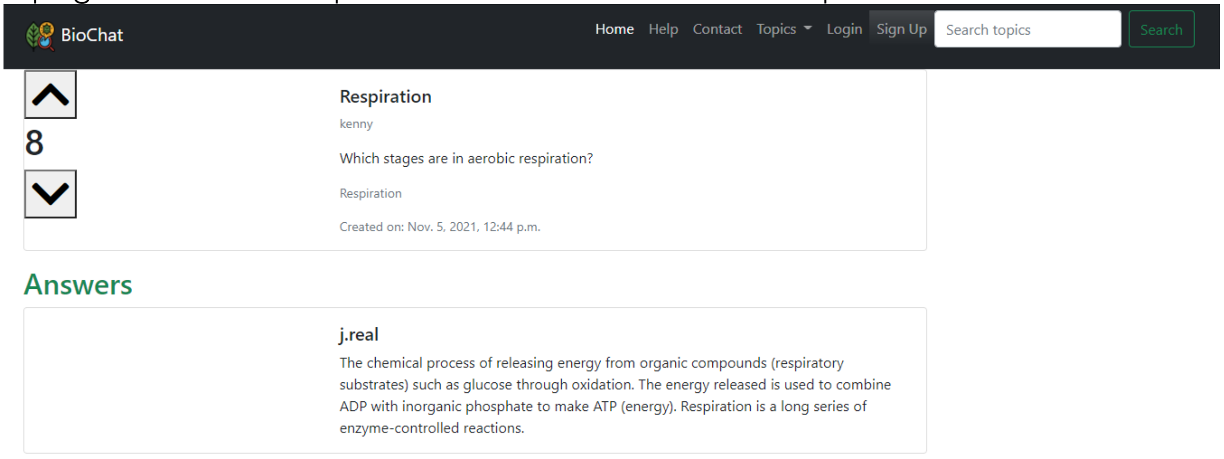
At the bottom of the page there will also be a way to submit answers in a form, but I have not programmed that in yet.



```
{% extends "base.html" %}

{% block content %}
<div class = "container">

<div class="card mb-3" style="max-width: 1000px;">
  <div class="row g-0">
    <div class="col-md-4">
      <button><i onclick="{{q.points.upvote}}" class="fas fa-chevron-up fa-3x"></i></button>
      <h1 class="center-align">{{q.points}}</h1>
      <button><i onclick="{{q.points.downvote}}" class="fas fa-chevron-down fa-3x"></i></button>
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">{{q.title}}</h5>
        <p class="card-text"><small class="text-muted">{{q.user}}</small></p>
        <p class="card-text">{{q.description}}</p>
        <p class="card-text"><small class="text-muted">{{q.group}}</small></p>
        <p class="card-text"><small class="text-muted">Created on: {{q.created_on}}</small></p>
      </div>
    </div>
  </div>
</div>
  {% for answer in a %}
  <div class="card mb-3" style="max-width: 1000px;">
  <div class="row g-0">
    <div class="col-md-4">

    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">{{answer.user}}</h5>
        <p class="card-text">{{answer.answer_text}}</p>
      </div>
    </div>
  </div>
    {% endfor %}
</div>

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

Figure 33 - HTML for Question page

This HTML code then takes the function from the above view and renders it as a page. It renders the question and answers in a card shape as shown below.



8

Respiration

kenny

Which stages are in aerobic respiration?

Respiration

Created on: Nov. 5, 2021, 12:44 p.m.

Answers

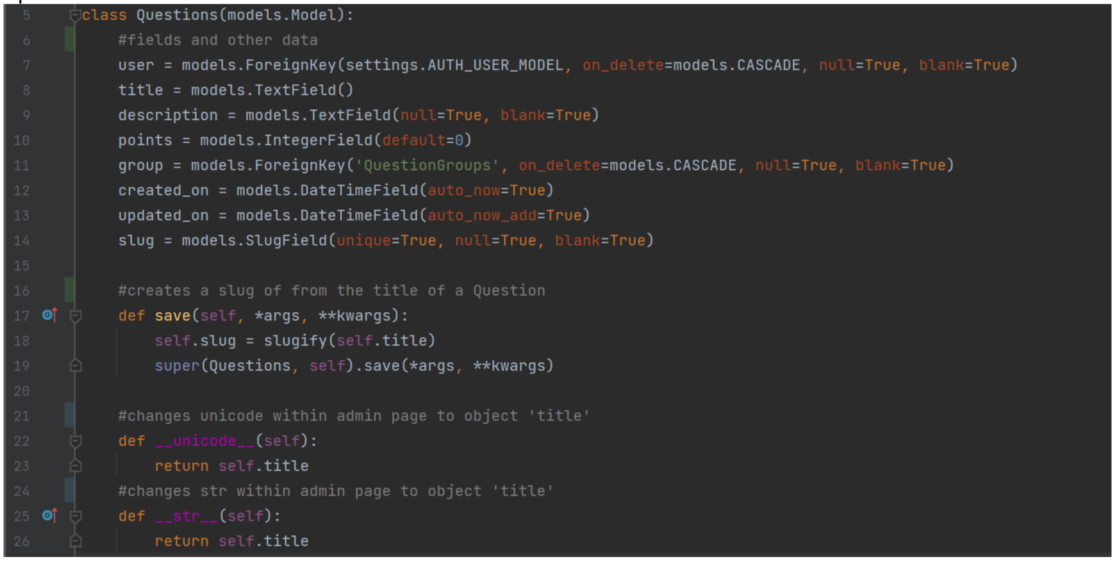
j.real

The chemical process of releasing energy from organic compounds (respiratory substrates) such as glucose through oxidation. The energy released is used to combine ADP with inorganic phosphate to make ATP (energy). Respiration is a long series of enzyme-controlled reactions.

Figure 34 - Image of mock-up Question page

Question creation

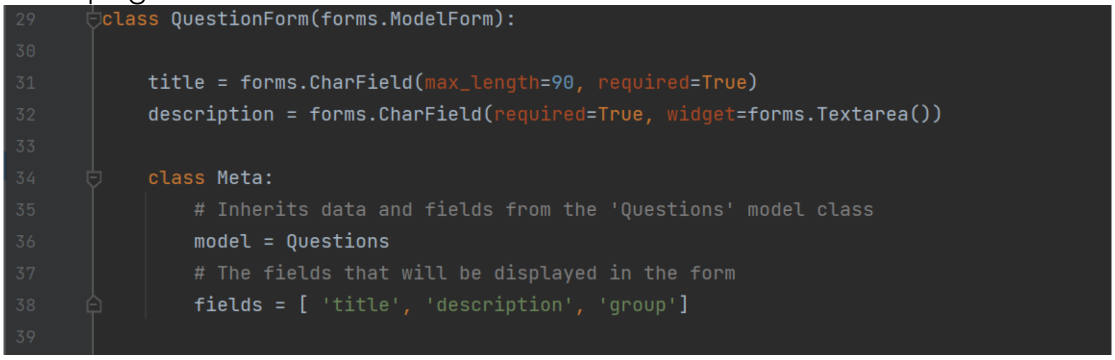
Now I am going to program in a page allowing you to create your own questions to be answered.



```
class Questions(models.Model):
    #fields and other data
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE, null=True, blank=True)
    title = models.TextField()
    description = models.TextField(null=True, blank=True)
    points = models.IntegerField(default=0)
    group = models.ForeignKey('QuestionGroups', on_delete=models.CASCADE, null=True, blank=True)
    created_on = models.DateTimeField(auto_now=True)
    updated_on = models.DateTimeField(auto_now_add=True)
    slug = models.SlugField(unique=True, null=True, blank=True)

    #creates a slug of from the title of a Question
    def save(self, *args, **kwargs):
        self.slug = slugify(self.title)
        super(Questions, self).save(*args, **kwargs)

    #changes unicode within admin page to object 'title'
    def __unicode__(self):
        return self.title
    #changes str within admin page to object 'title'
    def __str__(self):
        return self.title
```

Figure 35 - Model for Questions app

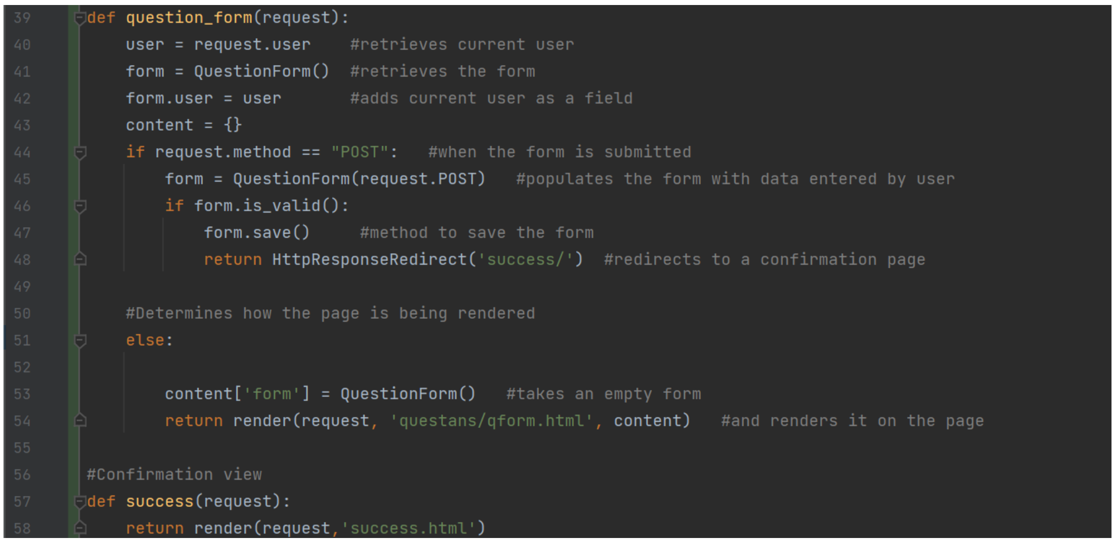
This is the model for the Questions functionality of the website. This was previously used when in the creation of each question page and the homepage.



```
class QuestionForm(forms.ModelForm):

    title = forms.CharField(max_length=90, required=True)
    description = forms.CharField(required=True, widget=forms.Textarea())

    class Meta:
        # Inherits data and fields from the 'Questions' model class
        model = Questions
        # The fields that will be displayed in the form
        fields = [ 'title', 'description', 'group']

```

Figure 36 - Question form class

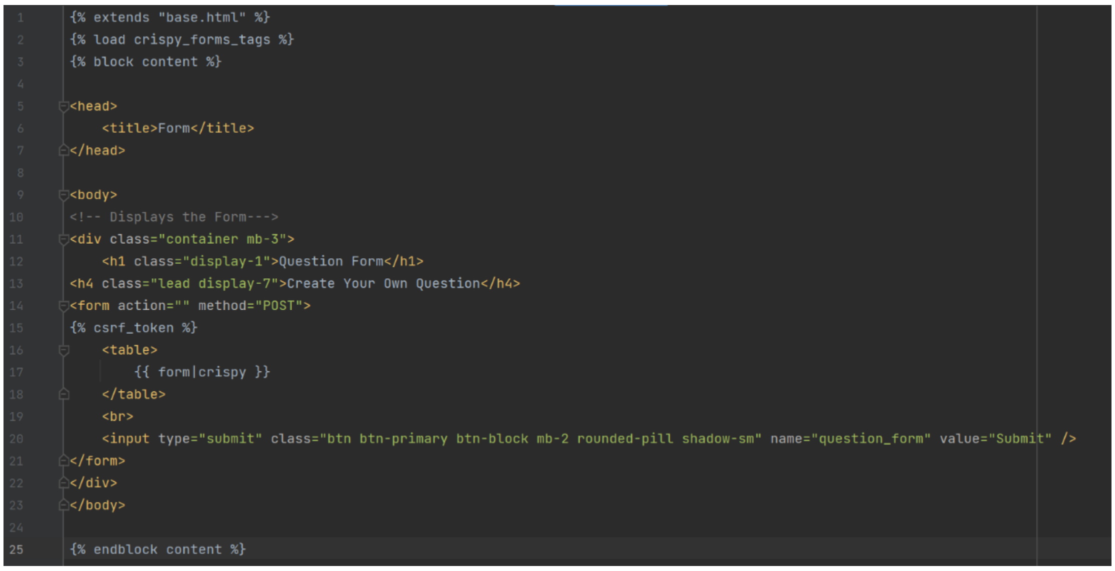
Shown above is the Question Form class that designates fields within the question creation form. This form class inherits the fields from the 'Questions' model (see Figure 35). This form is then transferred to the form creation view shown below.



```python
def question_form(request):
    user = request.user    #retrieves current user
    form = QuestionForm()  #retrieves the form
    form.user = user       #adds current user as a field
    content = {}
    if request.method == "POST":   #when the form is submitted
        form = QuestionForm(request.POST)   #populates the form with data entered by user
        if form.is_valid():
            form.save()     #method to save the form
            return HttpResponseRedirect('success/')  #redirects to a confirmation page

    #Determines how the page is being rendered
    else:

        content['form'] = QuestionForm()   #takes an empty form
    return render(request, 'questans/qform.html', content)   #and renders it on the page

#Confirmation view
def success(request):
    return render(request,'success.html')
```

Figure 37 - view to render form

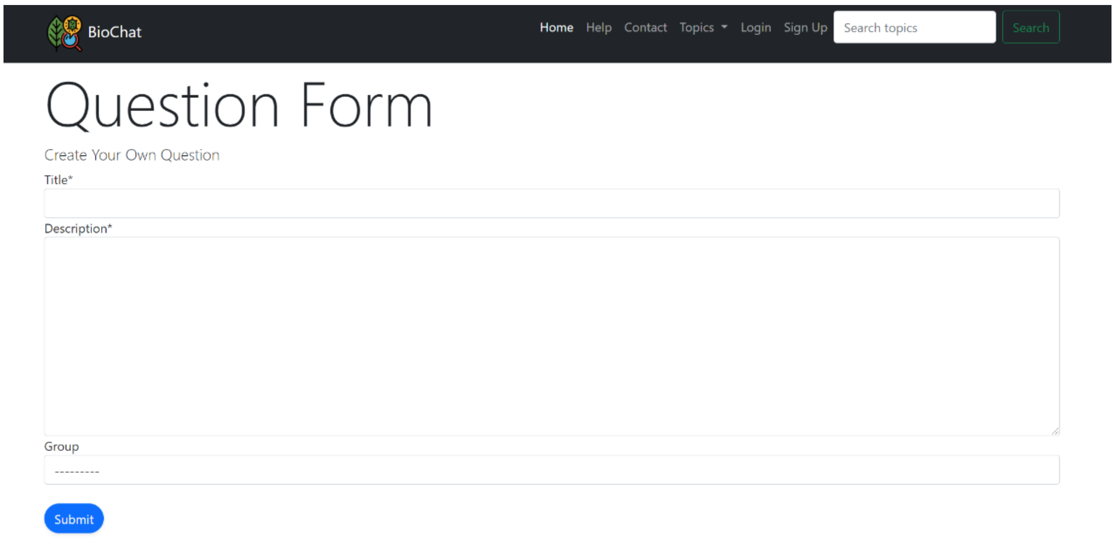
This view function not only deals with rendering the form on the page but also handles when a form is submitted. When a form is successfully submitted, it should link to a confirmation page. Before this though, the HTML for the form submission page is first rendered.



```html
{% extends "base.html" %}
{% load crispy_forms_tags %}
{% block content %}

<head>
    <title>Form</title>
</head>

<body>
<!-- Displays the Form--->
<div class="container mb-3">
    <h1 class="display-1">Question Form</h1>
<h4 class="lead display-7">Create Your Own Question</h4>
<form action="" method="POST">
{% csrf_token %}
    <table>
        {{ form|crispy }}
    </table>
    <br>
    <input type="submit" class="btn btn-primary btn-block mb-2 rounded-pill shadow-sm" name="question_form" value="Submit" />
</form>
</div>
</body>

{% endblock content %}
```

Figure 38 - HTML for Question submission form

All that code combined renders this page.



Question Form

Create Your Own Question

Title*

Description*

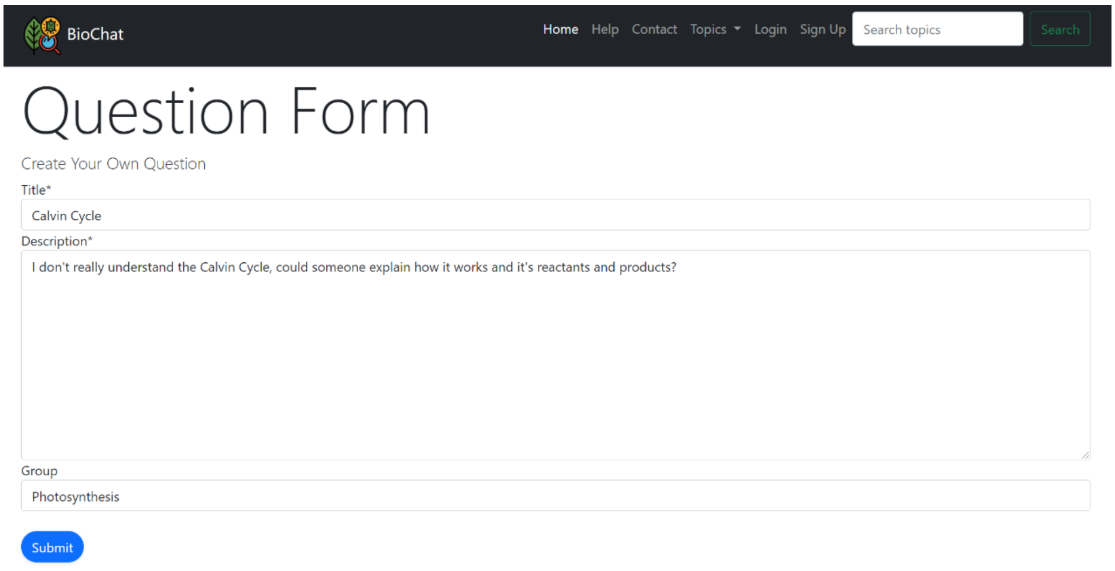
Group

Figure 39 - Rendered form submission page

On this page you can fill out the form to create a question which can then be interacted with and seen by other users.

Test to see if form page works properly

When the submit button is pressed the user should be redirected to a success page.



Question Form

Create Your Own Question

Title*

Calvin Cycle

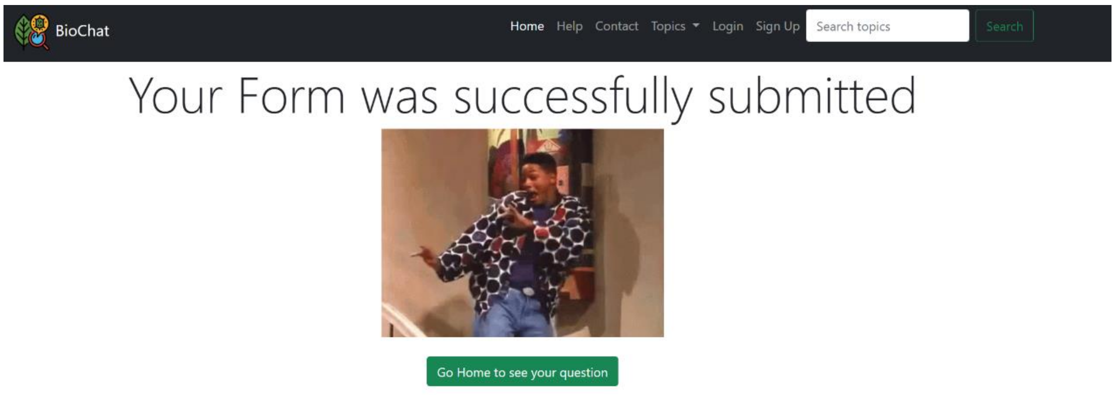
Description*

I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?

Group

Photosynthesis

Figure 40 - data entered for test



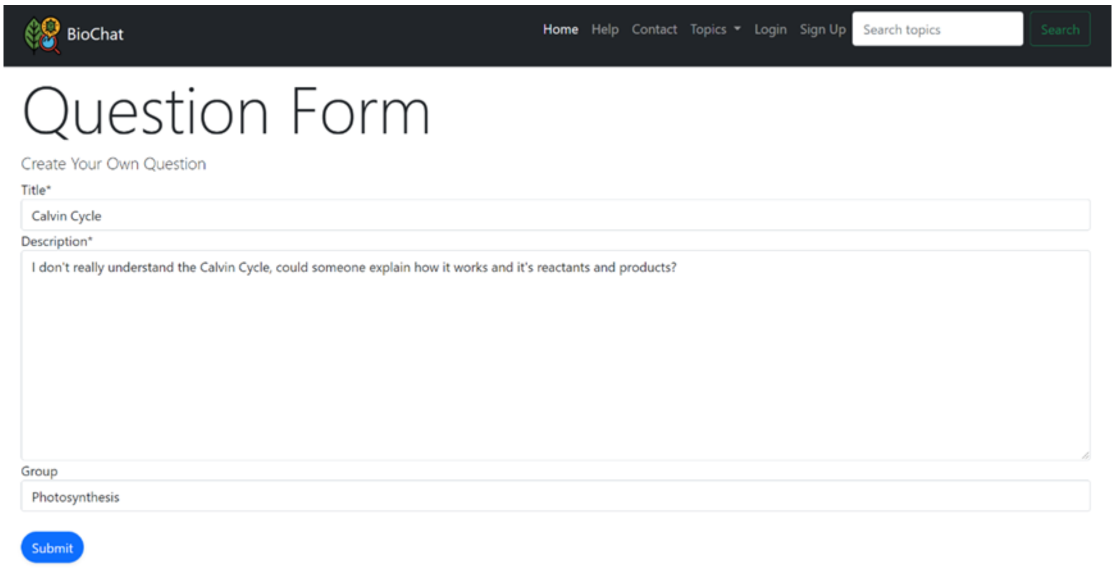
Your Form was successfully submitted

When clicked the button redirected the page to the success page showing a message and a button to go home.

This test was a success

Test to see if form is saved

I am testing to see if when filled, that the form is submitted to the database. The new question should show up on the homepage and admin.



Question Form

Create Your Own Question

Title*

Calvin Cycle

Description*

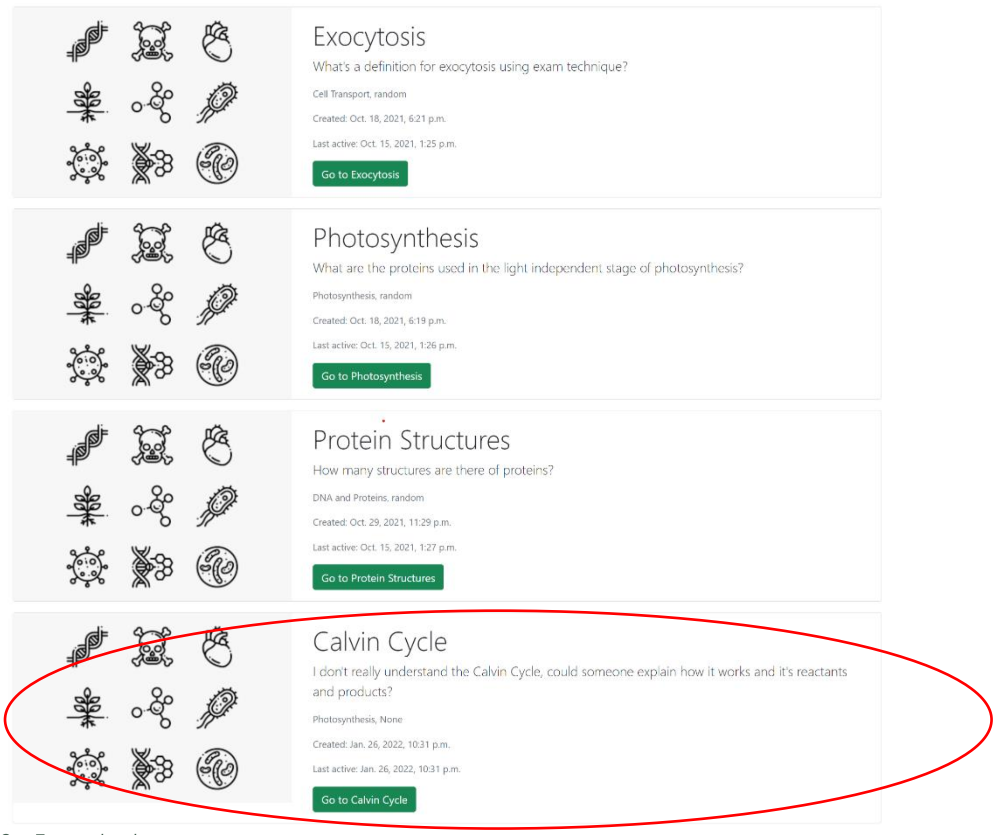
I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?

Group

Photosynthesis

Figure 41 - Data entered for test

When clicked the page was redirected to the success page. Going back to the homepage we can see if a new Question was really created.



Exocytosis
What's a definition for exocytosis using exam technique?
Cell Transport, random
Created: Oct. 18, 2021, 6:21 p.m.
Last active: Oct. 15, 2021, 1:25 p.m.
Go to Exocytosis

Photosynthesis
What are the proteins used in the light independent stage of photosynthesis?
Photosynthesis, random
Created: Oct. 18, 2021, 6:19 p.m.
Last active: Oct. 15, 2021, 1:26 p.m.
Go to Photosynthesis

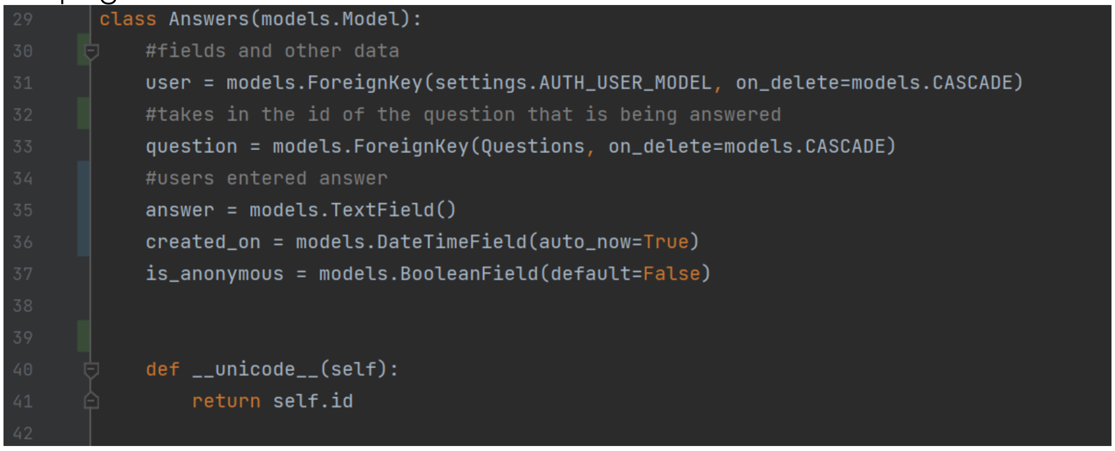
Protein Structures
How many structures are there of proteins?
DNA and Proteins, random
Created: Oct. 29, 2021, 11:29 p.m.
Last active: Oct. 15, 2021, 1:27 p.m.
Go to Protein Structures

Calvin Cycle
I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?
Photosynthesis, None
Created: Jan. 26, 2022, 10:31 p.m.
Last active: Jan. 26, 2022, 10:31 p.m.
Go to Calvin Cycle

Figure 42 - Form test

This image shows that at the bottom of the page a new question was created
This test was a success

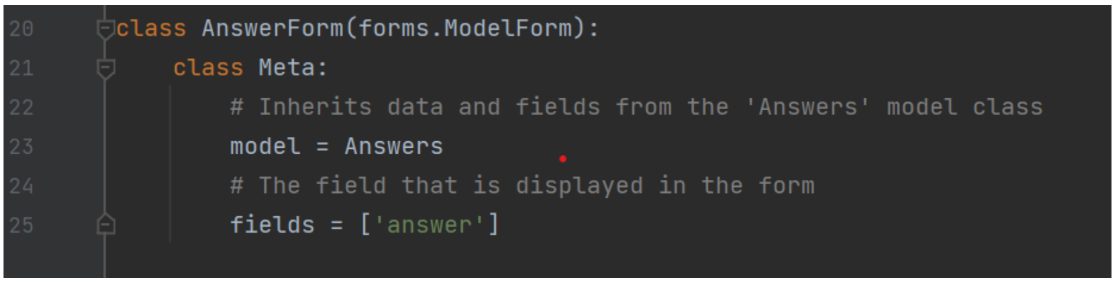
Now I am going to create an answer form that is displayed at the bottom of the page:



```
class Answers(models.Model):
    #fields and other data
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    #takes in the id of the question that is being answered
    question = models.ForeignKey(Questions, on_delete=models.CASCADE)
    #users entered answer
    answer = models.TextField()
    created_on = models.DateTimeField(auto_now=True)
    is_anonymous = models.BooleanField(default=False)


    def __unicode__(self):
        return self.id

```

Figure 43 - Model for answers

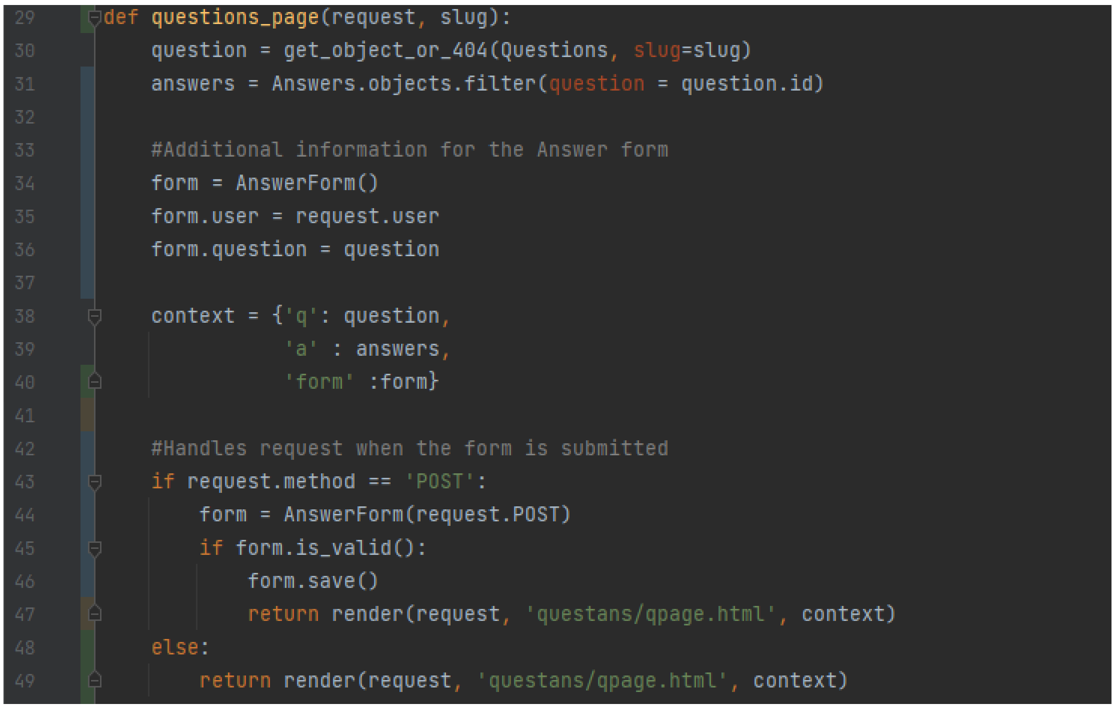
This is the model for the Answers functionality of the website.



```python
class AnswerForm(forms.ModelForm):
    class Meta:
        # Inherits data and fields from the 'Answers' model class
        model = Answers
        # The field that is displayed in the form
        fields = ['answer']
```

Figure 44 – Form for Answers

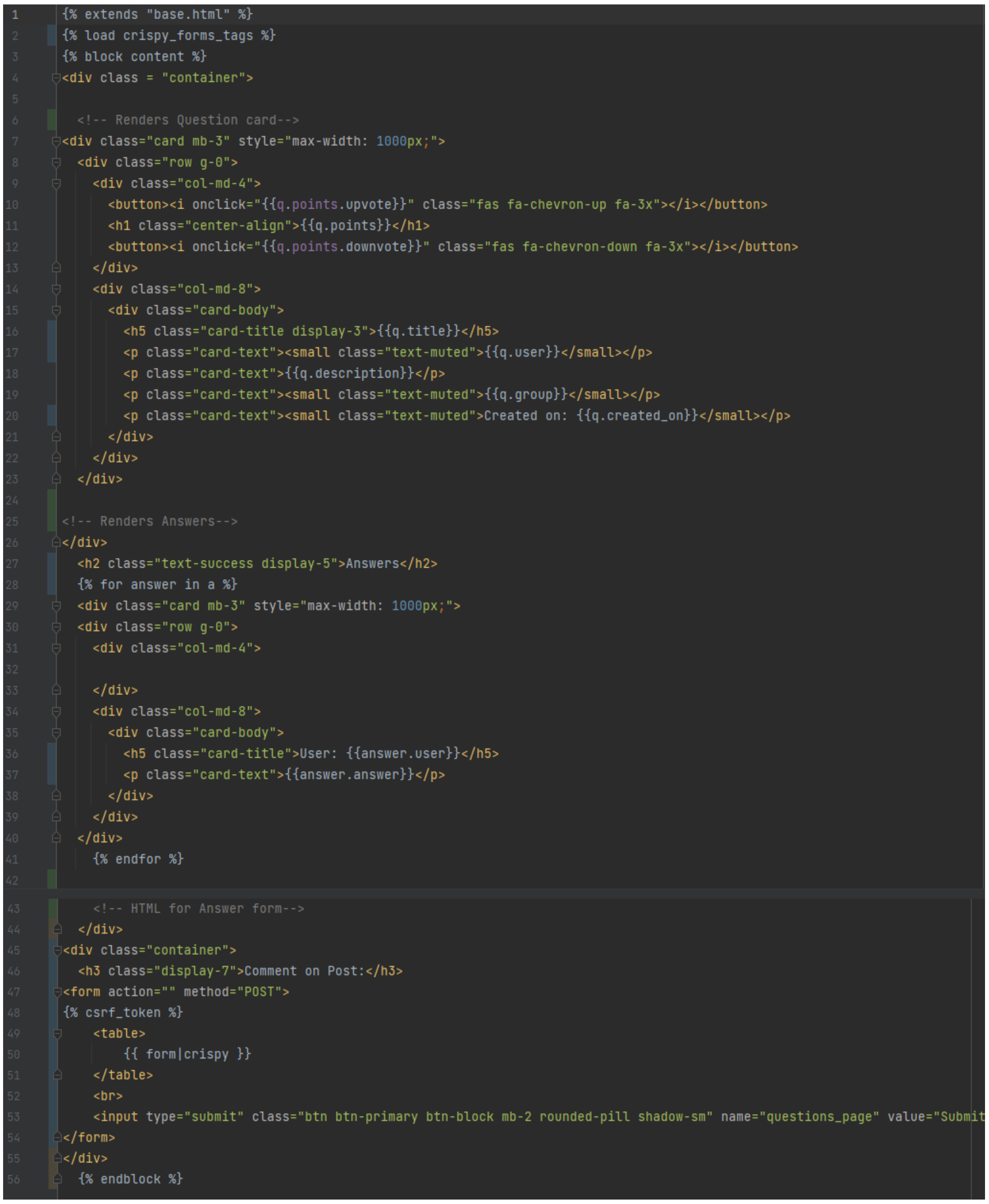
This form inherits the fields and data from the Answers model class.



```python
def questions_page(request, slug):
    question = get_object_or_404(Questions, slug=slug)
    answers = Answers.objects.filter(question = question.id)

    #Additional information for the Answer form
    form = AnswerForm()
    form.user = request.user
    form.question = question

    context = {'q': question,
               'a' : answers,
               'form' :form}

    #Handles request when the form is submitted
    if request.method == 'POST':
        form = AnswerForm(request.POST)
        if form.is_valid():
            form.save()
            return render(request, 'questans/qpage.html', context)
    else:
        return render(request, 'questans/qpage.html', context)
```

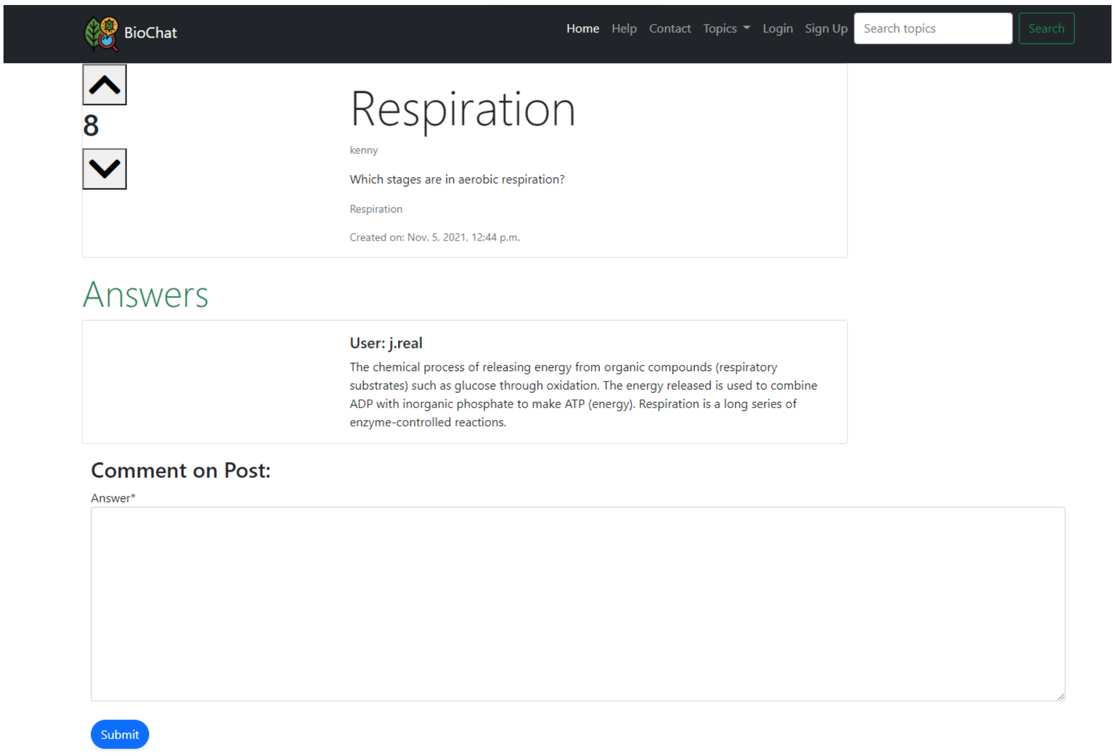
Figure 45 - Altered view for the question page

The question_page view is altered to handle the answer form and the request.



```
{% extends "base.html" %}
{% load crispy_forms_tags %}
{% block content %}
<div class = "container">

  <!-- Renders Question card-->
<div class="card mb-3" style="max-width: 1000px;">
  <div class="row g-0">
    <div class="col-md-4">
      <button><i onclick="{{q.points.upvote}}" class="fas fa-chevron-up fa-3x"></i></button>
      <h1 class="center-align">{{q.points}}</h1>
      <button><i onclick="{{q.points.downvote}}" class="fas fa-chevron-down fa-3x"></i></button>
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title display-3">{{q.title}}</h5>
        <p class="card-text"><small class="text-muted">{{q.user}}</small></p>
        <p class="card-text">{{q.description}}</p>
        <p class="card-text"><small class="text-muted">{{q.group}}</small></p>
        <p class="card-text"><small class="text-muted">Created on: {{q.created_on}}</small></p>
      </div>
    </div>
  </div>

<!-- Renders Answers-->
</div>
  <h2 class="text-success display-5">Answers</h2>
  {% for answer in a %}
  <div class="card mb-3" style="max-width: 1000px;">
  <div class="row g-0">
    <div class="col-md-4">

    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">User: {{answer.user}}</h5>
        <p class="card-text">{{answer.answer}}</p>
      </div>
    </div>
  </div>
    {% endfor %}

    <!-- HTML for Answer form-->
  </div>
<div class="container">
  <h3 class="display-7">Comment on Post:</h3>
<form action="" method="POST">
{% csrf_token %}
    <table>
        {{ form|crispy }}
    </table>
    <br>
    <input type="submit" class="btn btn-primary btn-block mb-2 rounded-pill shadow-sm" name="questions_page" value="Submit
</form>
</div>
  {% endblock %}
```

Figure 46 - HTML for new Question page


BioChat

Home Help Contact Topics Login Sign Up Search topics Search

8

Respiration

kenny

Which stages are in aerobic respiration?

Respiration

Created on: Nov. 5, 2021, 12:44 p.m.

Answers

User: j.real

The chemical process of releasing energy from organic compounds (respiratory substrates) such as glucose through oxidation. The energy released is used to combine ADP with inorganic phosphate to make ATP (energy). Respiration is a long series of enzyme-controlled reactions.

Comment on Post:

Answer*

Submit

Figure 47 - Rendered new question page with answer form

The combination of the view, form and html results in the rendering of this new question page. A comment form is rendered underneath the question and answers where users can comment on the page.

Updated Graphics

As requested by the stakeholders I updated the graphics of the Register and Login pages to make them look more modern.

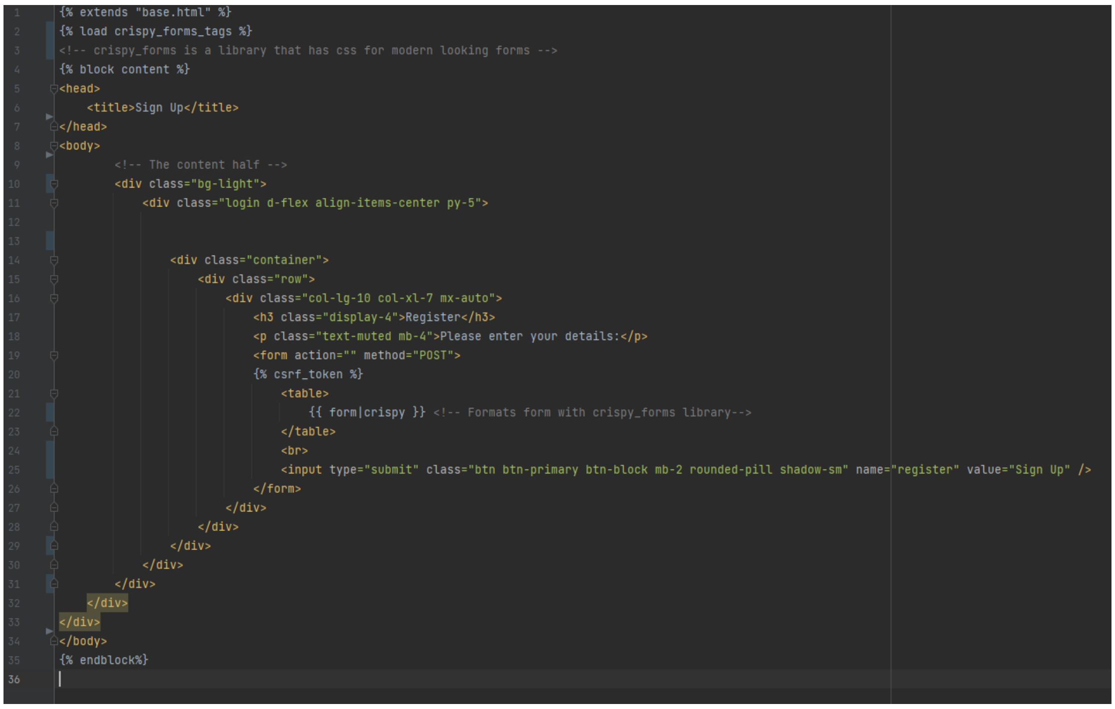
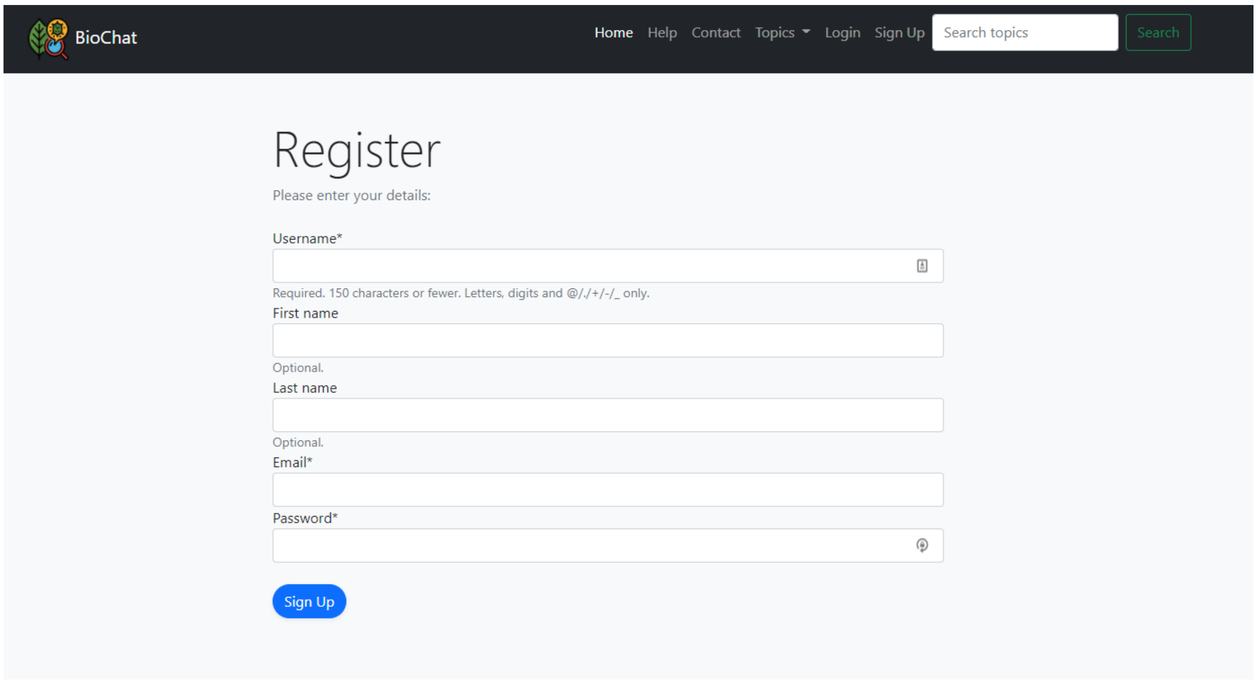

```
{% extends "base.html" %}
{% load crispy_forms_tags %}
<!-- crispy_forms is a library that has css for modern looking forms -->
{% block content %}
<head>
    <title>Sign Up</title>
</head>
<body>
        <!-- The content half -->
        <div class="bg-light">
            <div class="login d-flex align-items-center py-5">


                <div class="container">
                    <div class="row">
                        <div class="col-lg-10 col-xl-7 mx-auto">
                            <h3 class="display-4">Register</h3>
                            <p class="text-muted mb-4">Please enter your details:</p>
                            <form action="" method="POST">
                            {% csrf_token %}
                                <table>
                                    {{ form|crispy }} <!-- Formats form with crispy_forms library-->
                                </table>
                                <br>
                                <input type="submit" class="btn btn-primary btn-block mb-2 rounded-pill shadow-sm" name="register" value="Sign Up" />
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</body>
{% endblock%}
```

Figure 48 - Updated Register page HTML



Register

Please enter your details:

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

Optional.

Last name

Optional.

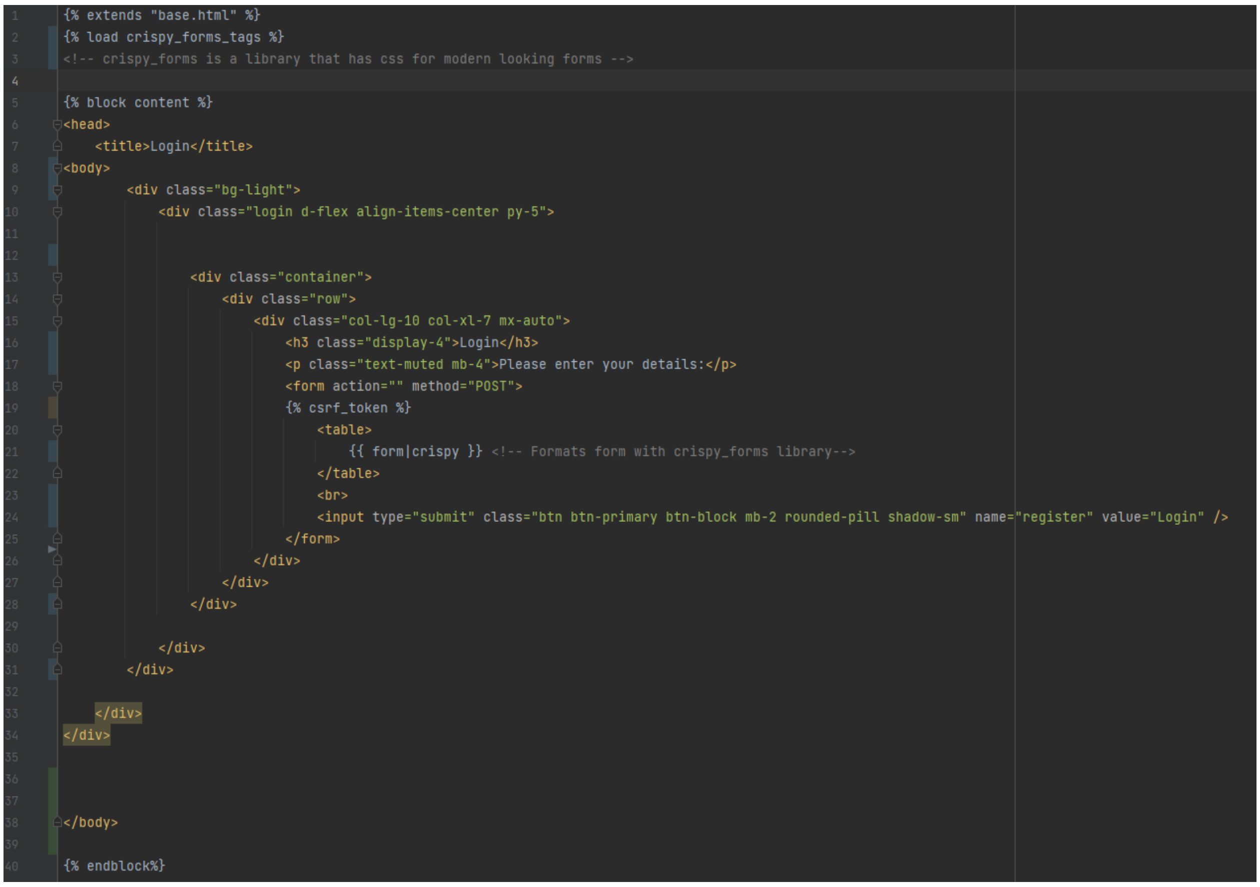
Email*

Password*

Sign Up

Figure 49 - Updated Register page

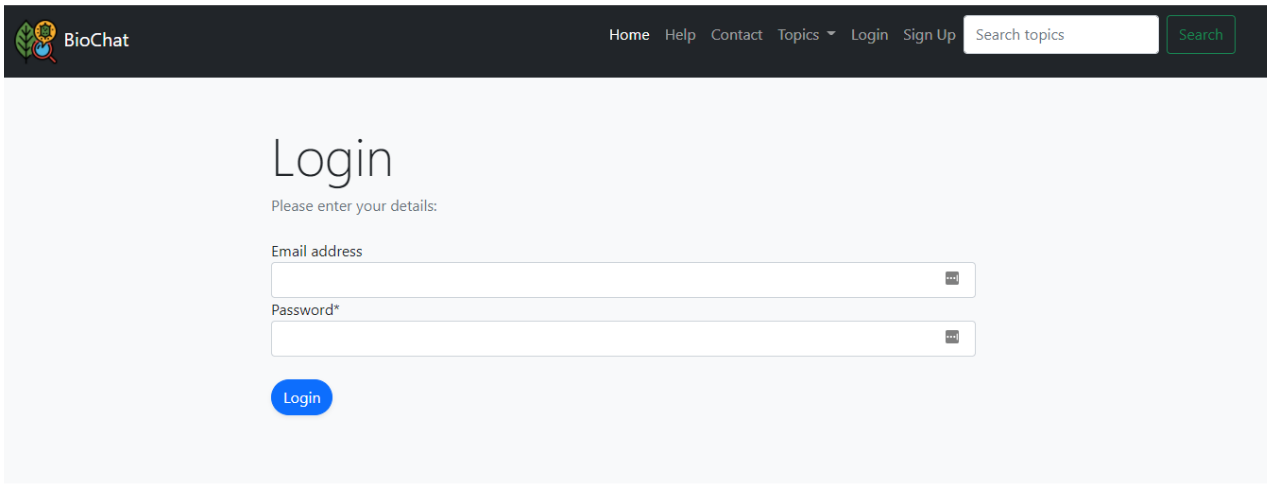
This update to the register page has a more modern appearance. Go to Figure 9 to compare the difference.



```
{% extends "base.html" %}
{% load crispy_forms_tags %}
<!-- crispy_forms is a library that has css for modern looking forms -->

{% block content %}
<head>
    <title>Login</title>
<body>
        <div class="bg-light">
            <div class="login d-flex align-items-center py-5">


                <div class="container">
                    <div class="row">
                        <div class="col-lg-10 col-xl-7 mx-auto">
                            <h3 class="display-4">Login</h3>
                            <p class="text-muted mb-4">Please enter your details:</p>
                            <form action="" method="POST">
                            {% csrf_token %}
                                <table>
                                    {{ form|crispy }} <!-- Formats form with crispy_forms library-->
                                </table>
                                <br>
                                <input type="submit" class="btn btn-primary btn-block mb-2 rounded-pill shadow-sm" name="register" value="Login" />
                            </form>
                        </div>
                    </div>
                </div>

            </div>
        </div>

    </div>
</div>



</body>

{% endblock%}
```

Figure 50 - HTML for new Login page

This altered login HTML applies a CSS library called ‘crispy forms’ to the login form making it look more modern.



Login

Please enter your details:

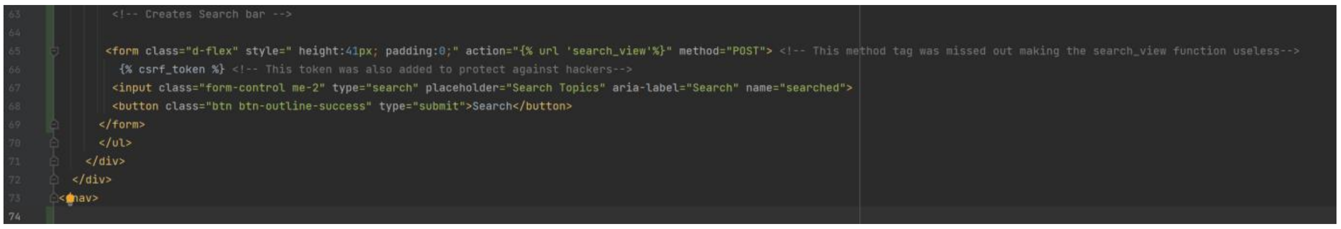
Email address

Password*

Figure 51 - Page rendered for new Login page

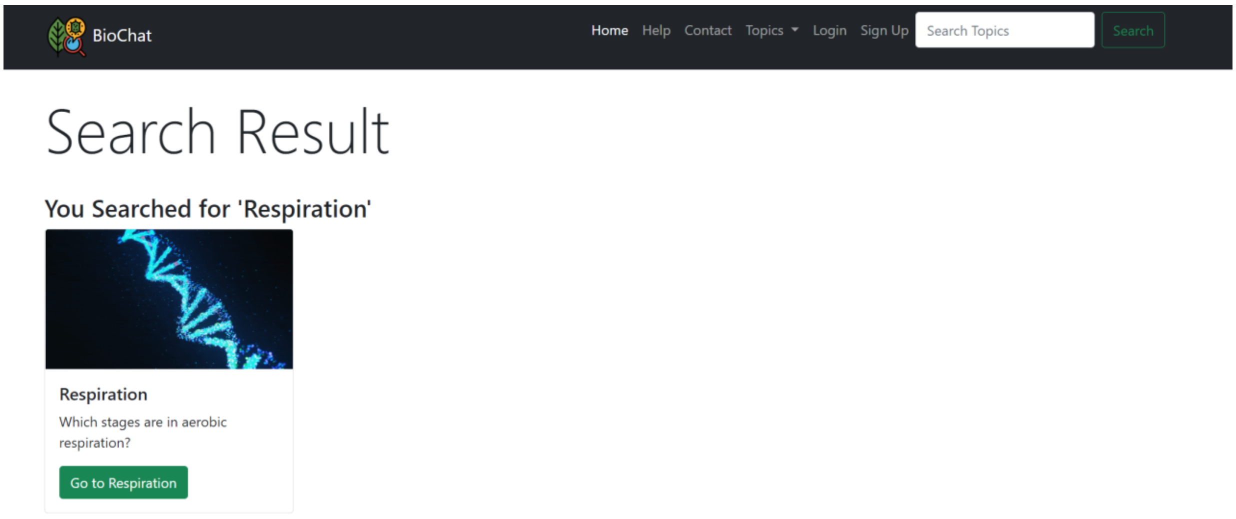
Fixed Search bar

I was also able to fix the faulty search bar by checking my code. I had realised that I had not included the method within the form tag on HTML. This resulted in the search_view function not registering the 'request' as POST. In the next few images this fix is shown:



```
            <!-- Creates Search bar -->

          <form class="d-flex" style=" height:41px; padding:0;" action="{% url 'search_view'%}" method="POST"> <!-- This method tag was missed out making the search_view function useless-->
             {% csrf_token %} <!-- This token was also added to protect against hackers-->
            <input class="form-control me-2" type="search" placeholder="Search Topics" aria-label="Search" name="searched">
            <button class="btn btn-outline-success" type="submit">Search</button>
          </form>
          </ul>
        </div>
      </div>
    </nav>
```

Figure 52 - Updated form in base HTML



Search Result

You Searched for 'Respiration'

Respiration

Which stages are in aerobic respiration?

Figure 53 - Resulting search page

Test for search bar

In this test I will be

What requirements were fulfilled?

This prototype fulfilled the 4th requirement to create question and answer forums, which is arguably the most integral requirement to the purpose of the website. This was long overdue to be implemented into the final prototype which has left the stakeholders confused and upset throughout development. Secondly the 6th requirement to create a search bar was completely fulfilled since it is working properly now unlike in prototype two.

What do the stakeholders say?

17. All the stakeholders say that they are happy with the outcome of the website since it allows them to easily create questions and answers to exchange among a large group of users, creating a community of sorts.
18. The UI and graphics of the website is adequate and has only improved as the development has continued, although both of my student stakeholders Stephen and Kwame both say that "even though the UI is modern is does seem a bit empty and dull at times"
19. The site is responsive when moving between pages
20. They are happy that the search bar is finally functional since it is the main way of accessing questions outside of the homepage.

What to improve for this prototype:

This is the final prototype – but if I was to improve upon it – I would get more feedback on how the stakeholders would like the UI to change since some of them were unhappy with how it currently is.

Also, I would implement the room functionality from my 6th requirement which would satisfy my stakeholders that are teachers.

Evaluation

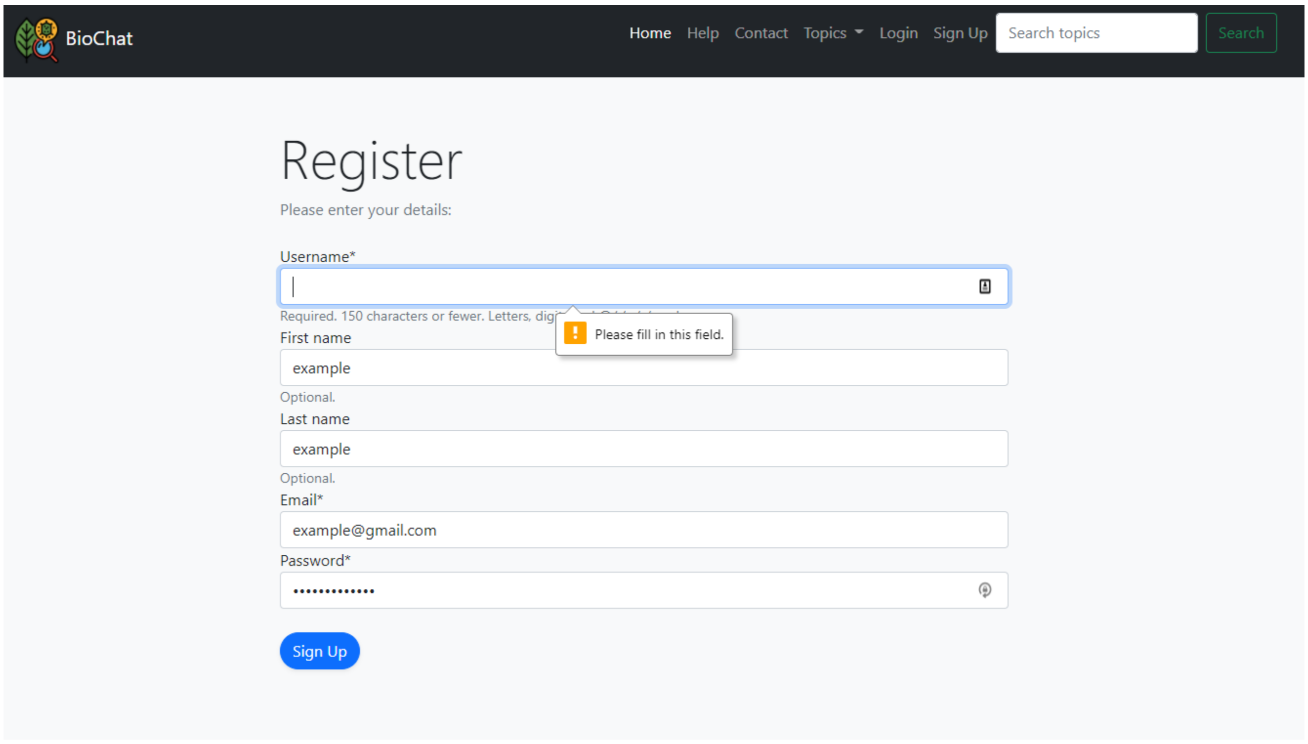
Testing

Here I will be carrying out tests that are in line with my test plan and below the table I will show all the evidence:

Test No.	Test	Explanation	Result
1	Test fields on register page	To test to see if the register form will still move data into the database without all required fields	Pass
2	Test password on register page	To test to see if the website will flag an invalid password.	Fail

3	Test login system on login page	To test if the website will prevent you from logging in to the site if you enter the wrong data.	Pass
4	Test search bar	This is to see if the search bar algorithm will realise that nothing has been searched and will output a suitable response for the user.	Pass
5	Test search bar	This is to see if the that the algorithm can show that something was searched but didn't exist in the database	Pass
6	Test search bar	This is to see that the search algorithm works and displays the correct questions to be accessed by the user.	Pass
7	Test Question form	This is to see if the form data is taken by the website and transferred to the database.	Pass
8	Test Question form	This is to see if the form will reject an invalid form.	Pass
9	Test Question form	This is to see if the form will reject an invalid form.	Pass
10	Test Answer form	To see if the Answer form transfers data to the database.	Pass

Test Evidence



Register

Please enter your details:

Username*

Required. 150 characters or fewer. Letters, dig

Please fill in this field.

First name

example

Optional.

Last name

example

Optional.

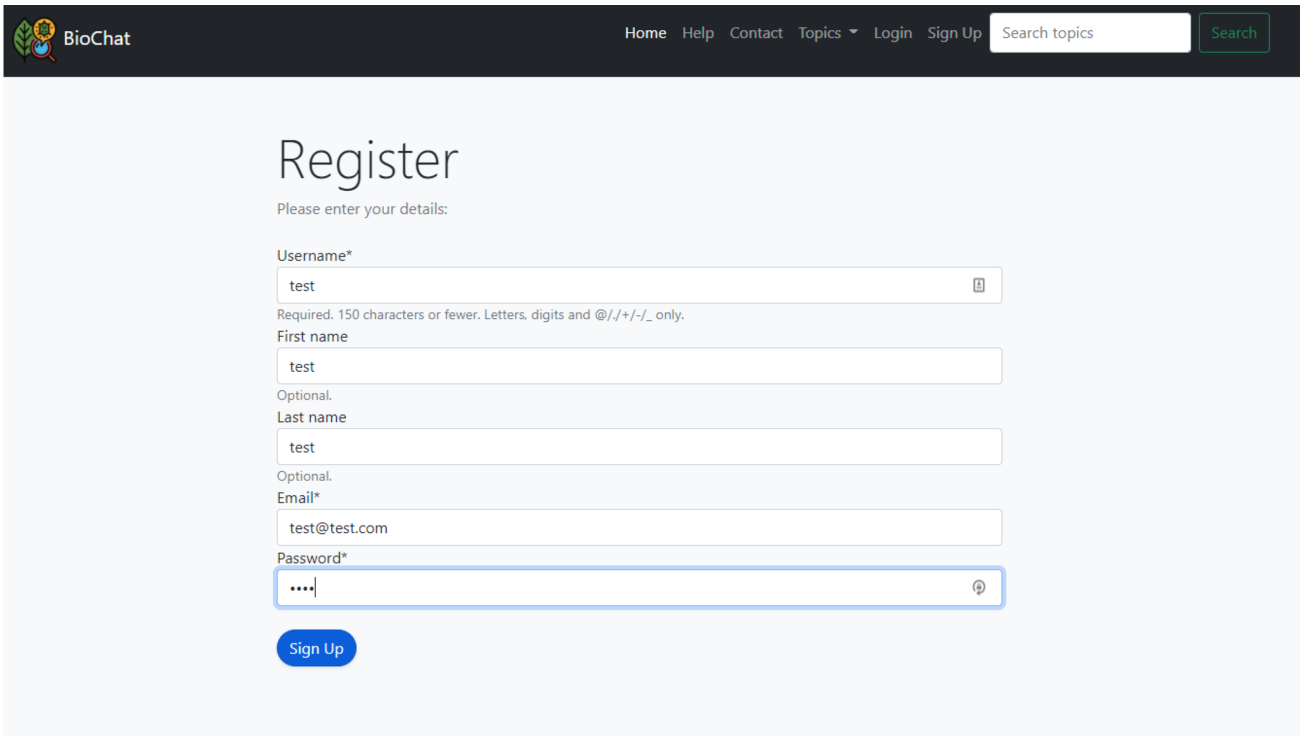
Email*

example@gmail.com

Password*

•••••••••••••

Figure 54 - Test 1 Evidence



Register

Please enter your details:

Username*

test

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

First name

test

Optional.

Last name

test

Optional.

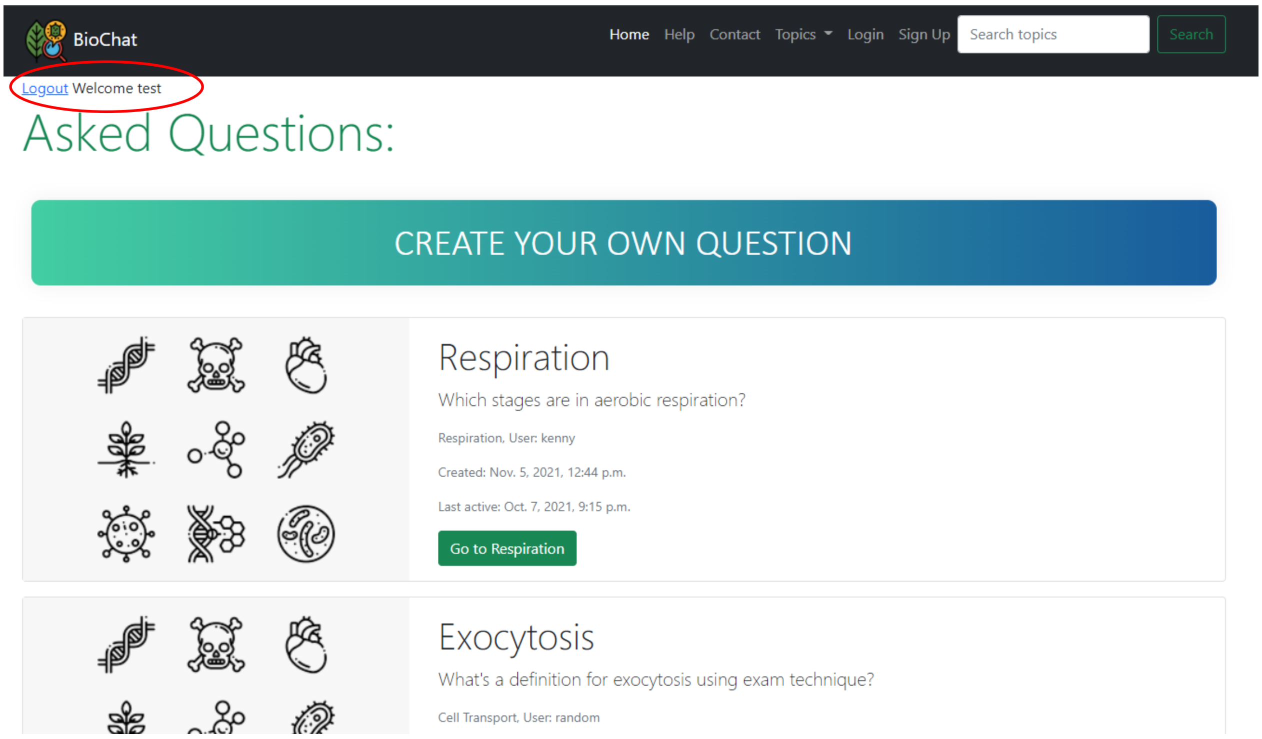
Email*

test@test.com

Password*

••••

Figure 55 - Test 2 Evidence pt1



Logout Welcome test

Asked Questions:

Respiration

Which stages are in aerobic respiration?

Respiration, User: kenny

Created: Nov. 5, 2021, 12:44 p.m.

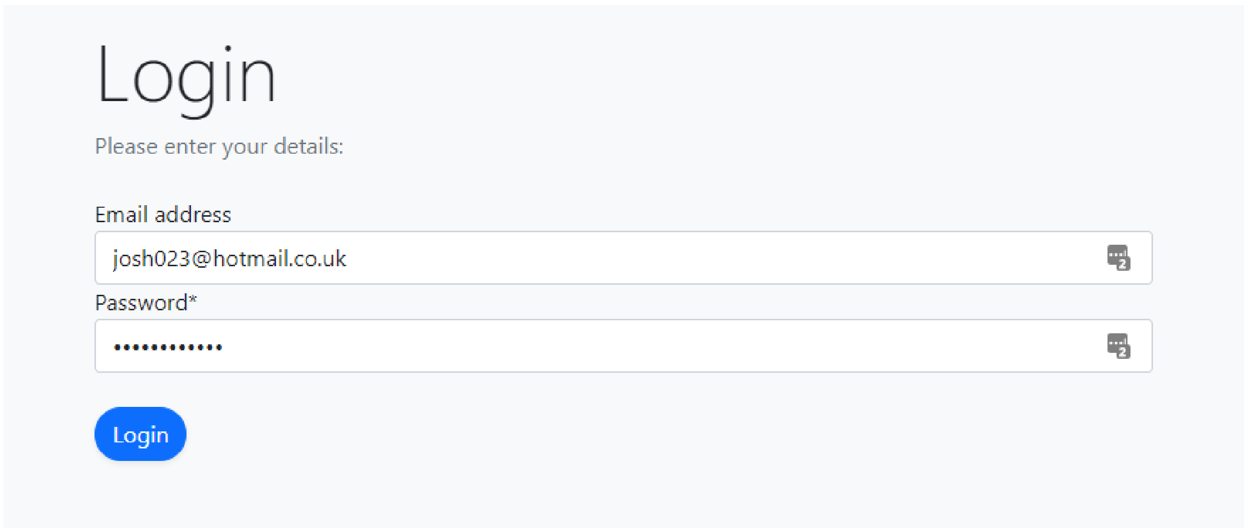
Last active: Oct. 7, 2021, 9:15 p.m.

Exocytosis

What's a definition for exocytosis using exam technique?

Cell Transport, User: random

Figure 56 - Test 2 Evidence pt2


Login

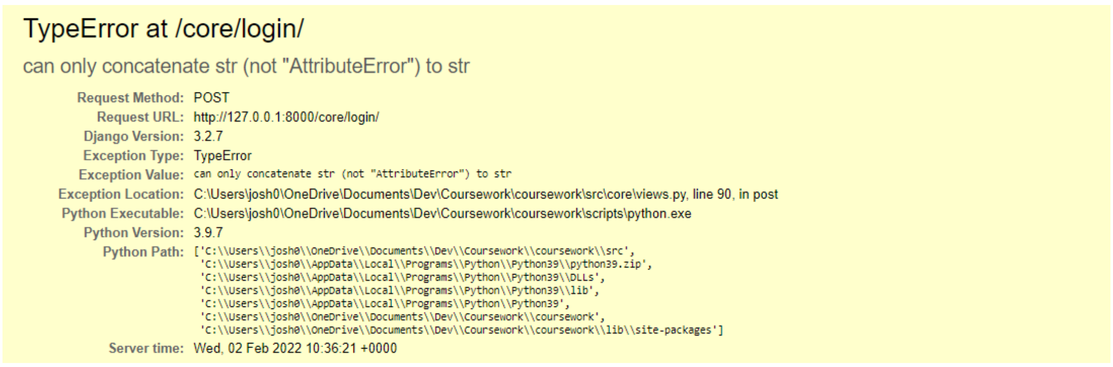
Please enter your details:

Email address: josh023@hotmail.co.uk

Password*: ••••••••••••

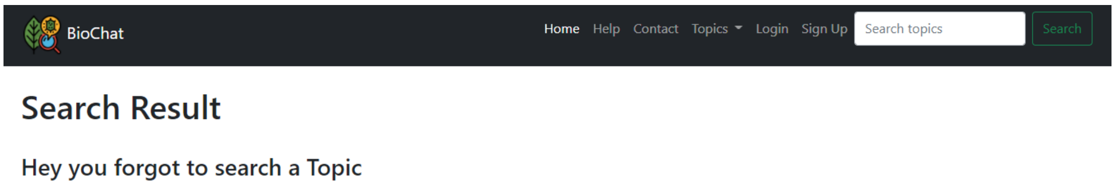
Figure 57 - Test 3 Evidence pt1

The password for this form is incorrect.



```
TypeError at /core/login/
can only concatenate str (not "AttributeError") to str

     Request Method:  POST
        Request URL:  http://127.0.0.1:8000/core/login/
     Django Version:  3.2.7
     Exception Type:  TypeError
    Exception Value:  can only concatenate str (not "AttributeError") to str
 Exception Location:  C:\Users\josh0\OneDrive\Documents\Dev\Coursework\coursework\src\core\views.py, line 90, in post
  Python Executable:  C:\Users\josh0\OneDrive\Documents\Dev\Coursework\coursework\scripts\python.exe
     Python Version:  3.9.7
        Python Path:  ['C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework\\src',
                       'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip',
                       'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\DLLs',
                       'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\lib',
                       'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39',
                       'C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework',
                       'C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework\\lib\\site-packages']
        Server time:  Wed, 02 Feb 2022 10:36:21 +0000
```

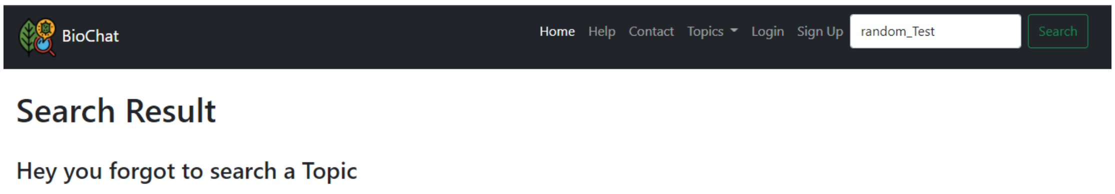
Figure 59 - Test 3 Evidence pt2



Search Result

Hey you forgot to search a Topic

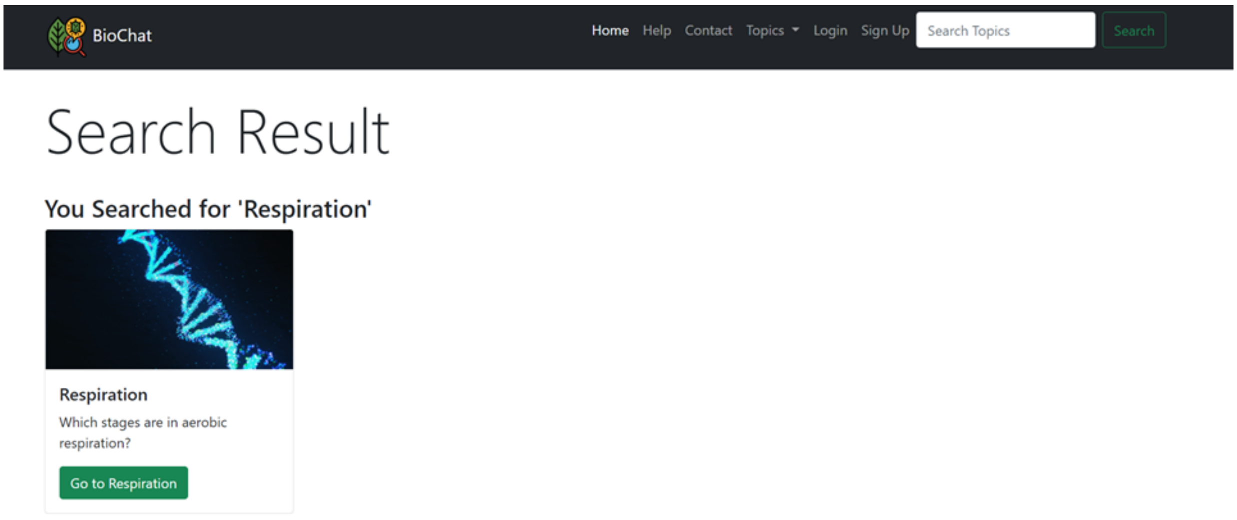
Figure 58 - Test 4 Evidence



Search Result

Hey you forgot to search a Topic

Figure 60 - Test 5 Evidence



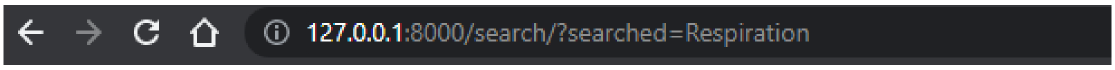
Search Result

You Searched for 'Respiration'

Respiration

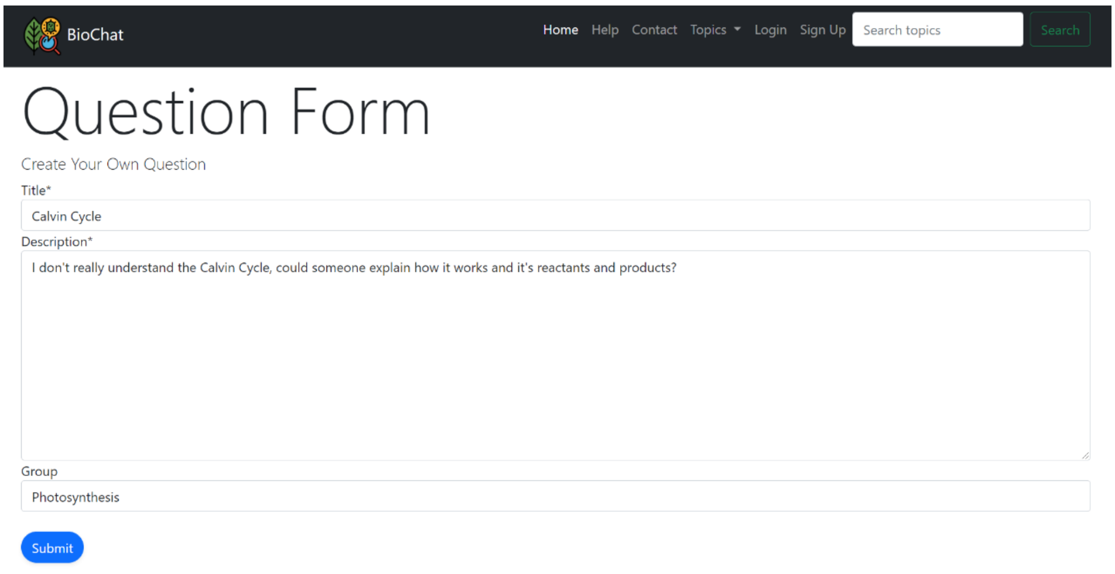
Which stages are in aerobic respiration?

Figure 61 - Test 6 Evidence pt1



127.0.0.1:8000/search/?searched=Respiration

Figure 62 - Test 6 Evidence pt2



Question Form

Create Your Own Question

Title*

Calvin Cycle

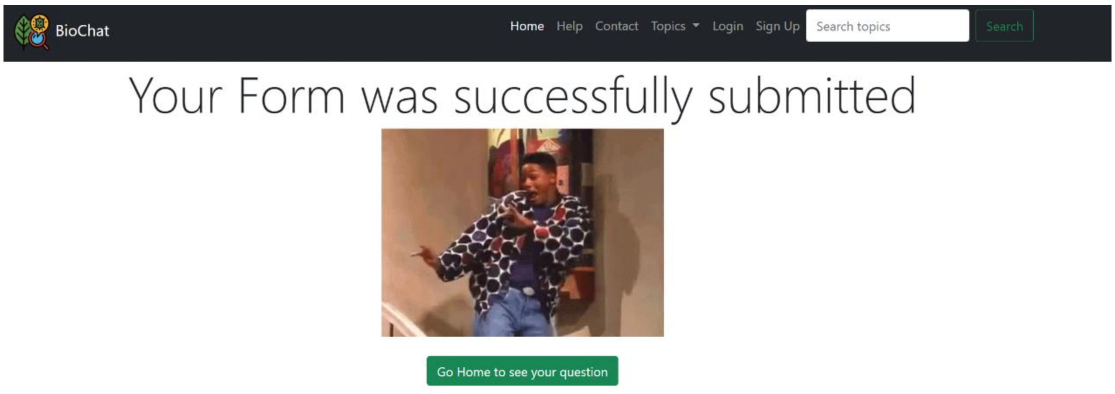
Description*

I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?

Group

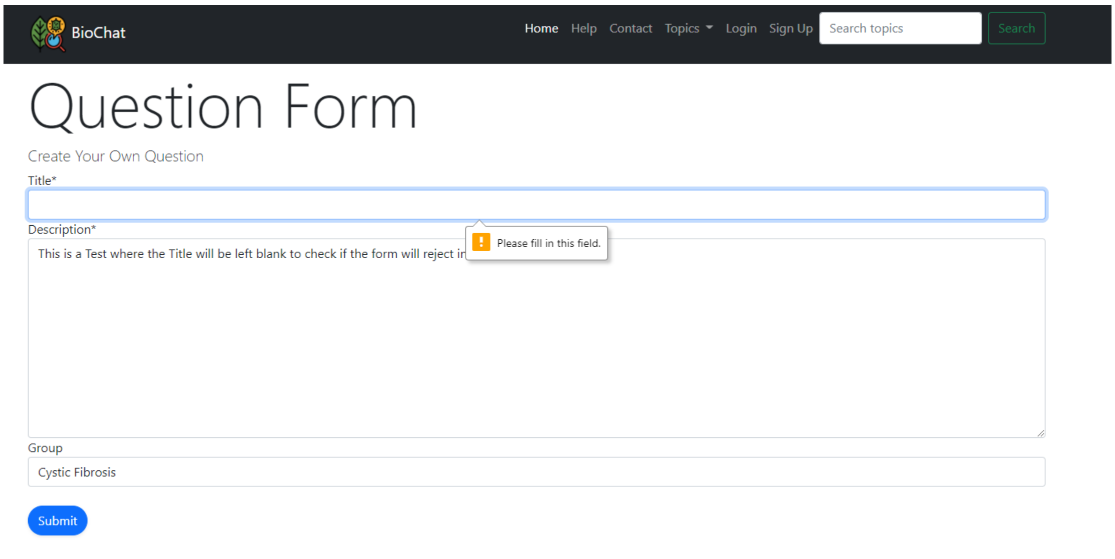
Photosynthesis

Figure 63 - Test 7 Evidence pt1



Your Form was successfully submitted

Figure 64 - Test 7 Evidence pt2



Question Form

Create Your Own Question

Title*

Description*

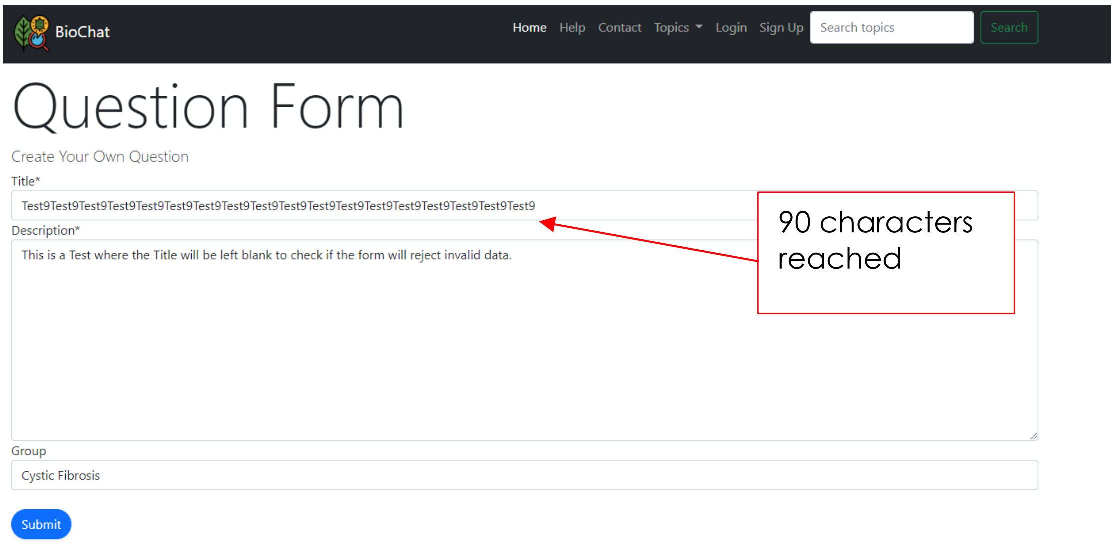
This is a Test where the Title will be left blank to check if the form will reject in

Please fill in this field.

Group

Cystic Fibrosis

Figure 65 - Test 8 Evidence



Question Form

Create Your Own Question

Title*

Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9Test9

Description*

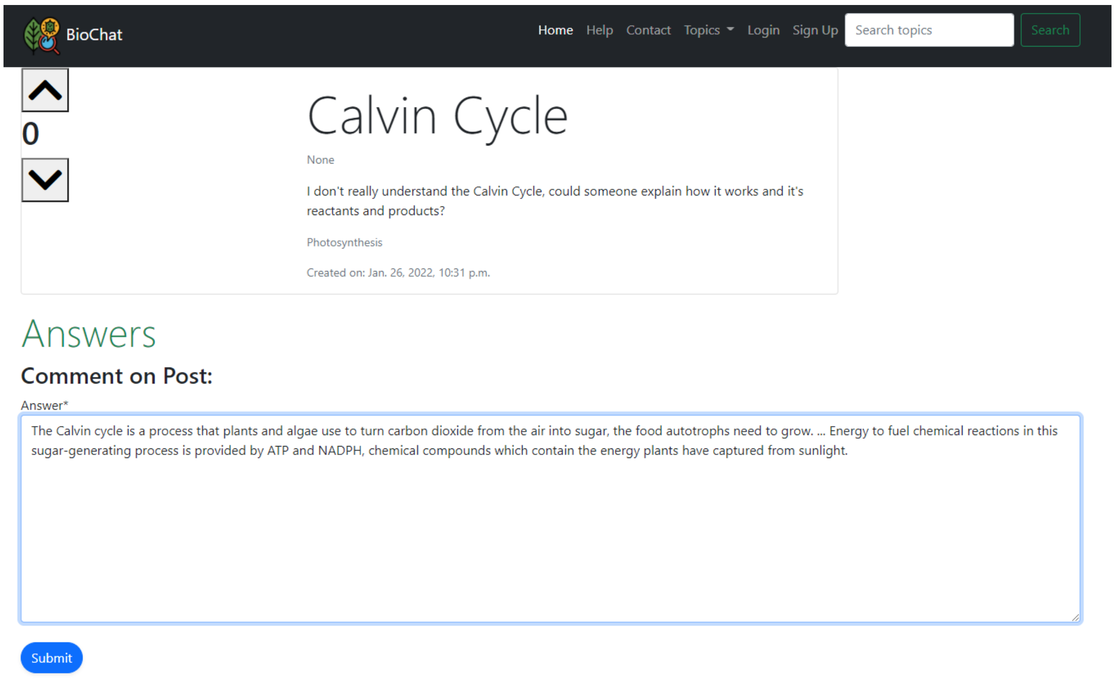
This is a Test where the Title will be left blank to check if the form will reject invalid data.

Group

Cystic Fibrosis

Figure 66 - Test 9 Evidence

The form doesn't allow you to input any more characters after the threshold of 90 is reached.



Calvin Cycle

None

I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?

Photosynthesis

Created on: Jan. 26, 2022, 10:31 p.m.

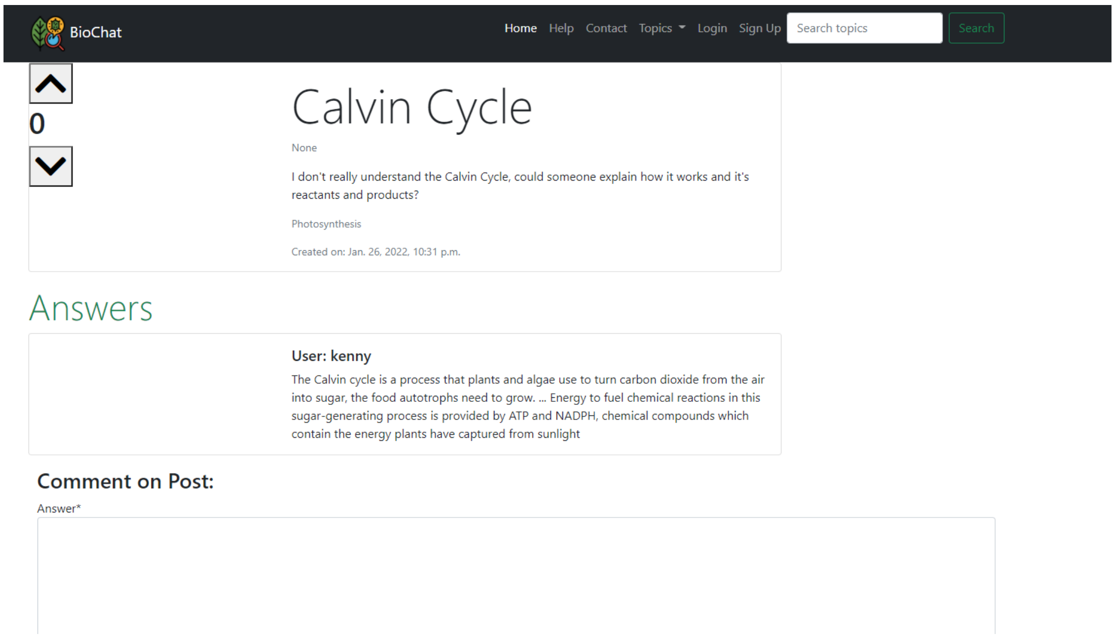
Answers

Comment on Post:

Answer*

The Calvin cycle is a process that plants and algae use to turn carbon dioxide from the air into sugar, the food autotrophs need to grow. ... Energy to fuel chemical reactions in this sugar-generating process is provided by ATP and NADPH, chemical compounds which contain the energy plants have captured from sunlight.

Figure 67 - Test 10 Evidence pt1



Calvin Cycle

None

I don't really understand the Calvin Cycle, could someone explain how it works and it's reactants and products?

Photosynthesis

Created on: Jan. 26, 2022, 10:31 p.m.

Answers

User: kenny

The Calvin cycle is a process that plants and algae use to turn carbon dioxide from the air into sugar, the food autotrophs need to grow. ... Energy to fuel chemical reactions in this sugar-generating process is provided by ATP and NADPH, chemical compounds which contain the energy plants have captured from sunlight

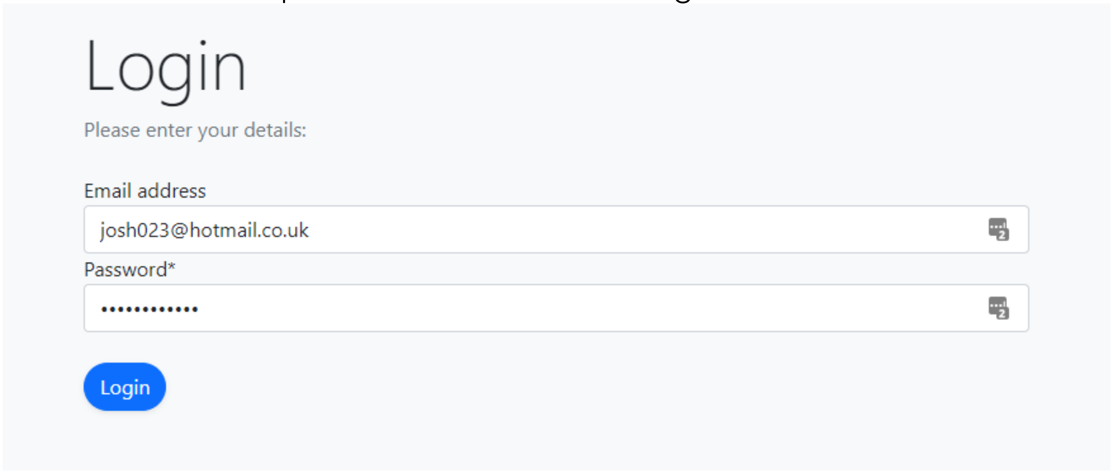
Comment on Post:

Answer*

Figure 68 - Test 10 Evidence pt2

Failed Tests

The only failed test that I encountered was with test no.3. Here a correct email but incorrect password was entered to login to the website.


Login

Please enter your details:

Email address

josh023@hotmail.co.uk

Password*

••••••••••••

Login

Figure 69 - Failed test

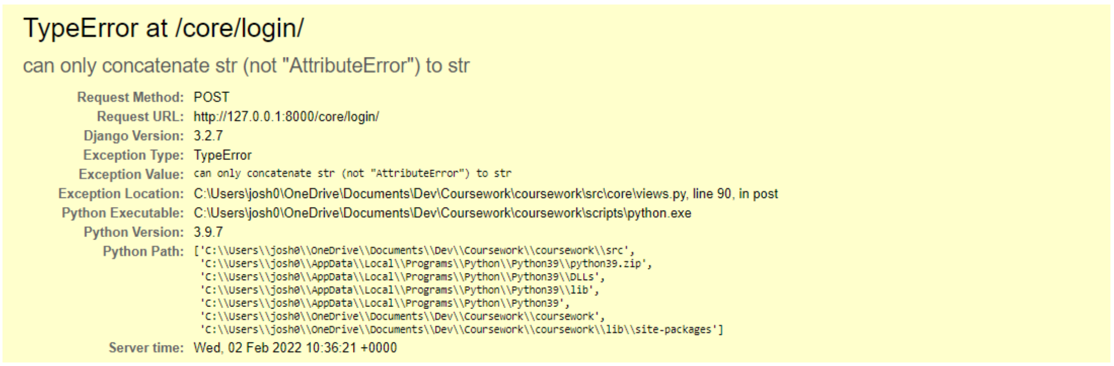

```
TypeError at /core/login/
can only concatenate str (not "AttributeError") to str

   Request Method:  POST
      Request URL:  http://127.0.0.1:8000/core/login/
   Django Version:  3.2.7
   Exception Type:  TypeError
  Exception Value:  can only concatenate str (not "AttributeError") to str
Exception Location: C:\Users\josh0\OneDrive\Documents\Dev\Coursework\coursework\src\core\views.py, line 90, in post
Python Executable:  C:\Users\josh0\OneDrive\Documents\Dev\Coursework\coursework\scripts\python.exe
   Python Version:  3.9.7
      Python Path:  ['C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework\\src',
                     'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\python39.zip',
                     'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\DLLs',
                     'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39\\lib',
                     'C:\\Users\\josh0\\AppData\\Local\\Programs\\Python\\Python39',
                     'C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework',
                     'C:\\Users\\josh0\\OneDrive\\Documents\\Dev\\Coursework\\coursework\\lib\\site-packages']
      Server time:  Wed, 02 Feb 2022 10:36:21 +0000
```

Figure 70 - Failed test

Which produced an error, crashing the page. Through analysing the code, I was able to work out that there was an error with the last part of my Login view class, particularly the code that deals with errors.

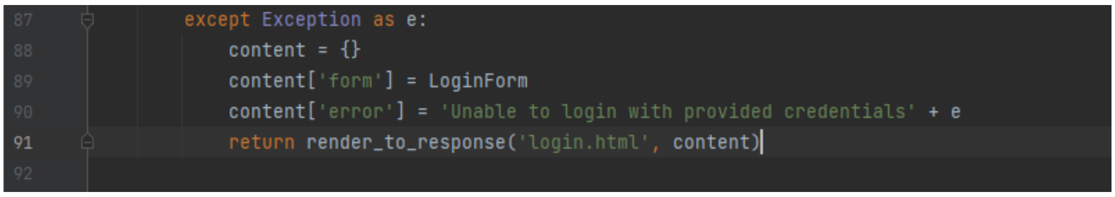

```
87          except Exception as e:
88              content = {}
89              content['form'] = LoginForm
90              content['error'] = 'Unable to login with provided credentials' + e
91              return render_to_response('login.html', content)
92
```

Figure 71 - Code causing error

As the error states the function is not able to output the code since the error message of 'Unable to login with provided credentials' clashes with the exception since it is not in string.

To solve this issue I would need to find where the exception is sent to forcibly change it into string with the ""str()" method in python.

Post development questionnaire results

After the development was completed, I let my stakeholders use the website for a week to get their opinions on the outcome of the website. After the week I gave them a questionnaire (first made in the design) to answer about how well the site turned out:

1) *What do you think about the overall layout of the website?*
2) *How could the website function be improved?*
3) *How informative is the UI of the website?*
4) *Do you suggest any changes to the UI?*

Here are their results:

Stephen

1) The website looks modern, and the layout is clear since questions and answers are arranged in cards. The Toolbar also helps this clean, modern feel.
2) There's not much that is really missing from the website I'm content with how it works
3) The UI shows all the information that you need, the toolbar helping this simplicity work.
4) Maybe changing the colour scheme could bring a new flavour to the website not that I don't like the green biology theme.

Kwame

1) The website looks better than I expected. All-important items are laid out on the homepage and the toolbar reducing the chance that you get lost.
2) The function of the website is really good. Although I wish I could access and customize my profile more for example with a profile picture.
3) The UI is very clean and informative
4) The UI looks perfect to me

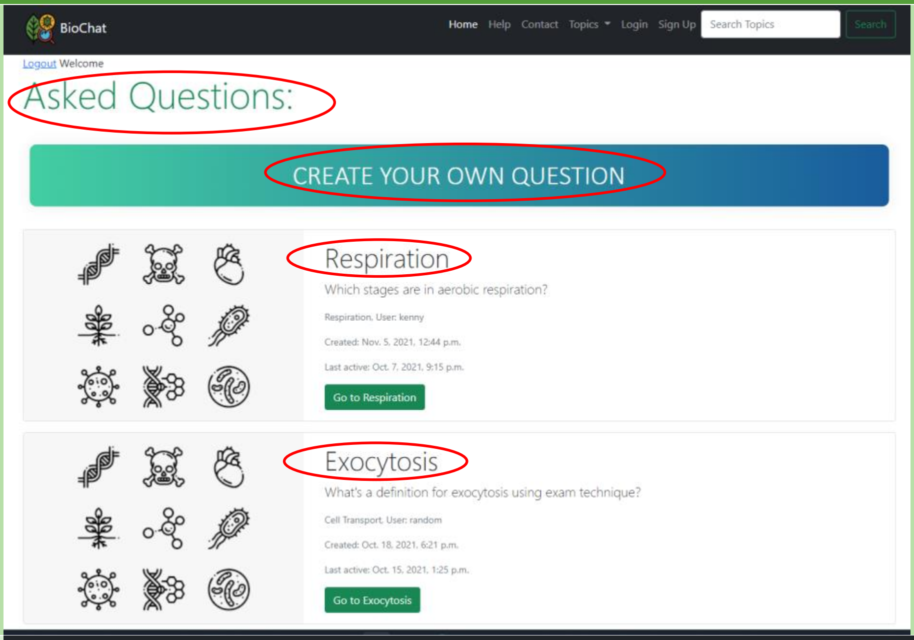
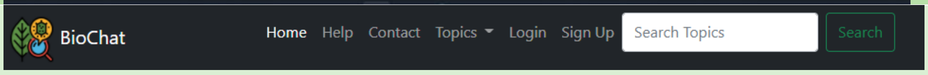
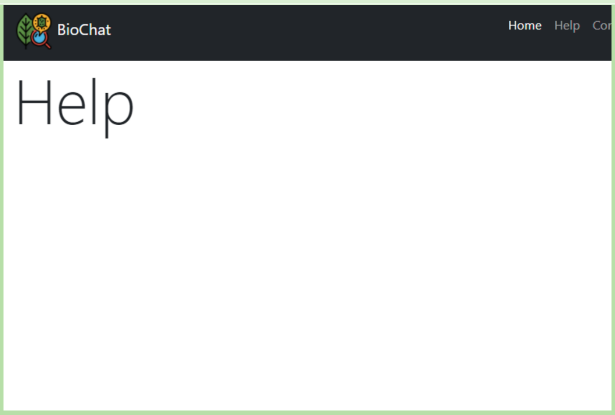
Mr Smith

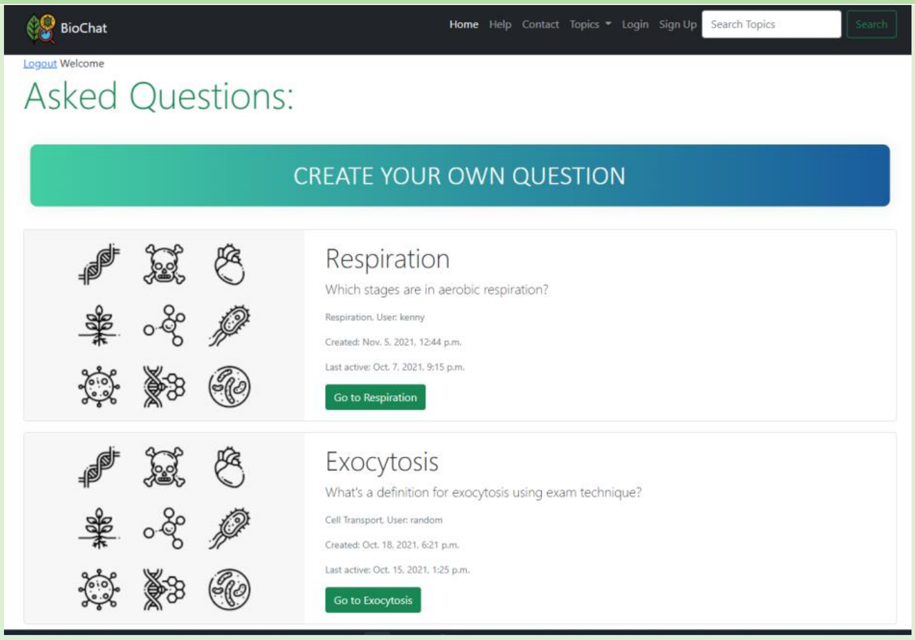
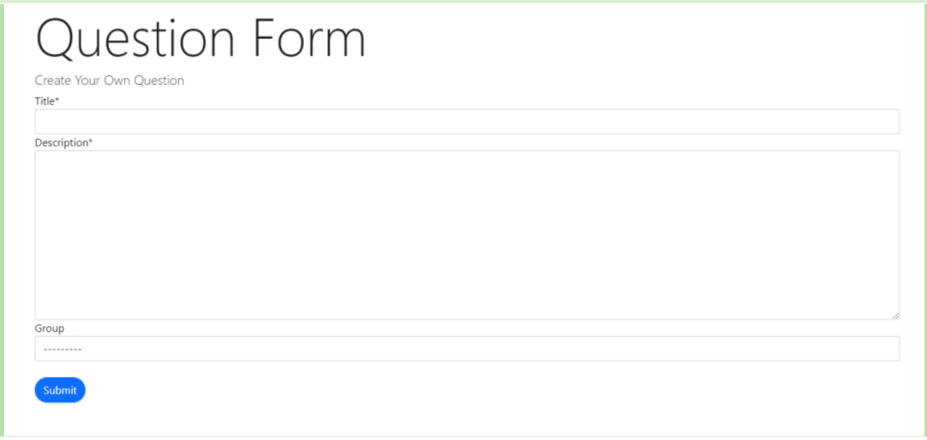
1) The layout of the website is surprisingly practical with all objects being spaced out evenly
2) The basic features currently are perfect, it's just that I wanted the feature where I could make rooms for my individual classes.
3) The UI is very informative
4) The UI is perfect as it is

Dr Adekunle

1) The overall layout is clean and is better than my expectations. Everything I want to see is spread clearly on the website.
2) The functions implemented into the website currently are perfect and well built, it's just I wanted teachers to be able to make rooms for classes that are invite-only.
3) The UI is pretty modern and is easy on the eyes
4) As a finished product the UI doesn't need any changes.

Usability Testing

Feature	Evidence	Success	Justification
A clear layout	 BioChat Home Help Contact Topics Login Sign Up Search Topics Search Logout Welcome Asked Questions: CREATE YOUR OWN QUESTION Respiration Which stages are in aerobic respiration? Respiration. User: kenny Created: Nov. 5, 2021, 12:44 p.m. Last active: Oct. 7, 2021, 9:15 p.m. Go to Respiration Exocytosis What's a definition for exocytosis using exam technique? Cell Transport. User: random Created: Oct. 18, 2021, 6:21 p.m. Last active: Oct. 15, 2021, 1:25 p.m. Go to Exocytosis	Success	Most users don't read all the information on a webpage and instead skim through the page. Therefore, I've titled and added heading to all items on the page.
Toolbar	 BioChat Home Help Contact Topics Login Sign Up Search Topics Search	Success	The user can access most of the main pages of the website through clicking buttons on the toolbar. This reduces the chance of getting lost and being confused
Help page	 BioChat Home Help Co Help	Partial success	This feature is partially met since the page is implemented. The only issue is that no content has been added to help the user yet. So, the page isn't

			populated yet but can easily be done.
Well formatted content	 BioChat Home Help Contact Topics Login Sign Up Search Topics Search Logout Welcome Asked Questions: CREATE YOUR OWN QUESTION Respiration Which stages are in aerobic respiration? Respiration, User: kenny Created: Nov. 5, 2021, 12:44 p.m. Last active: Oct. 7, 2021, 9:15 p.m. Go to Respiration Exocytosis What's a definition for exocytosis using exam technique? Cell Transport, User: random Created: Oct. 18, 2021, 6:21 p.m. Last active: Oct. 15, 2021, 1:25 p.m. Go to Exocytosis	Success	Buttons and all elements are all evenly spaced so that users can clearly view and interact with the website. Having elements separated and clearly labelled will make the website easier to read and less complex.
Usable forms	 Question Form Create Your Own Question Title* Description* Group --------- Submit	Success	Forms like the one shown are easy to access and have clear titles for fields that need to be filled.

Maintenance

Commented Code:
Having commented code means that if any future programmers want to work on my code in the future, there is already notation and code there to make it easy to see what each function does in the code and explains lines of code and what they do. This helps with maintenance as if there needs to be any updates, it is less likely that a future programmer or even myself will make a mistake and delete a vital node for the program.

Clear Variable Names:
The variable names in the code are clear to understand, and they help to make future programmers understand the code as the variables directly connote to certain points in the code, making it easier to change the

variables that are related to performance, and the code much easier to read and edit. This helps with maintenance as it means that the variables can be easily edited to improve the performance of the website and synergy between pages. It also improves the readability of the program for future updates.

Formatting Spaces:
Having spaces between functions, and the variables being loaded into the game, means that the code is much easier to read and understand. The spaces make it clear when chunks of code or functions are separated, and thus can help segment the code and make it easier to add future updates as the readability of the code is improved.

Limitations

Room's function
2 out my 4 stakeholders said in the analysis questionnaire that they would like the implementation of a rooms system where invite-only rooms could be made. This would have been for teachers and students alike since it could be used to separate the school from other users online. This would have also allowed students to be given direct feedback from their teachers easier.

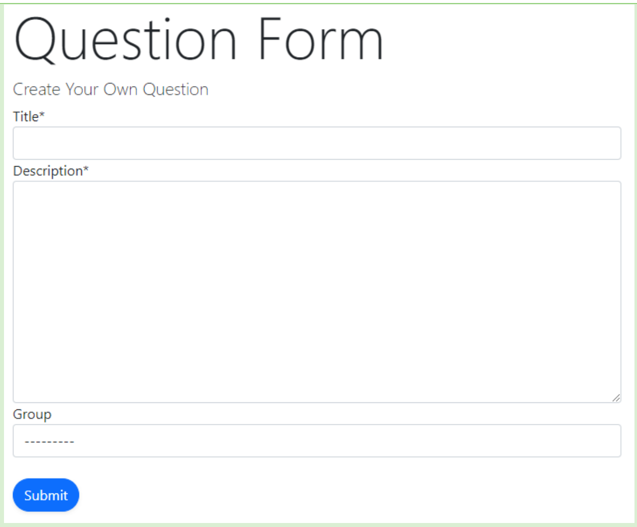
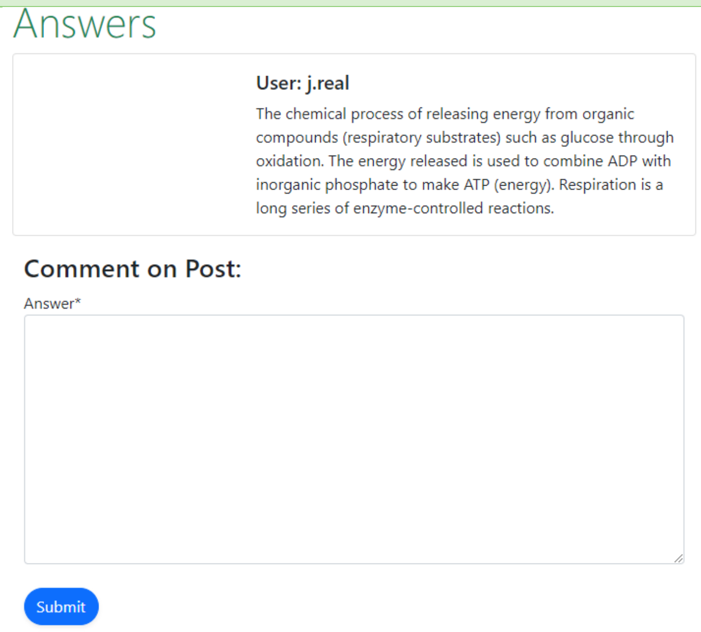
Unfortunately, this was not able to be implemented in time for the end of the development. Although for future programmers, the backend work for question and user classes is already there so development of this feature wouldn't be too difficult.

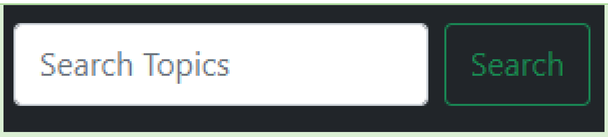
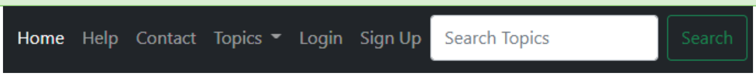
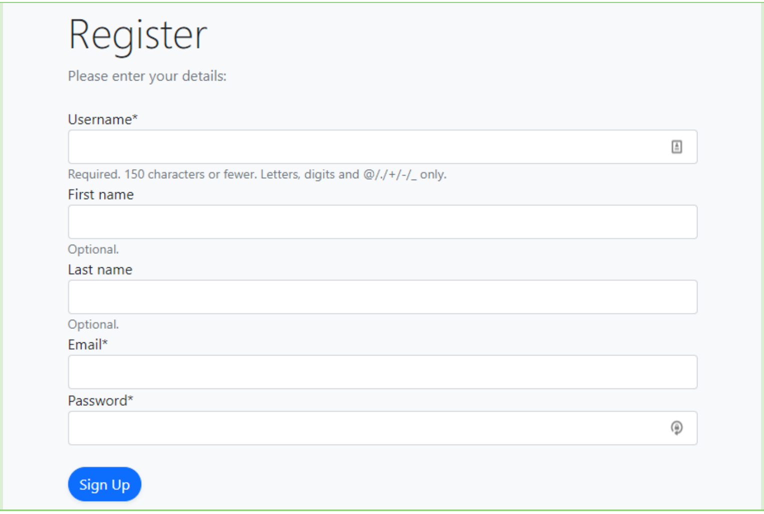
Profile customization
In the final website, users aren't given much customization over their account. For this reason, other than for making questions there isn't as much of an incentive for users to interact and answer questions. Features that could have solved this problem are allowing users to change their profile, adding bios that can be edited by the user, allowing users to change their username, etc.

Tracking function
This limitation follows on from the previous one in the fact that there are not many incentives to make users come back and interact on the site. Features like tracking the number of questions created and answered, tracking the upvotes gained from these Q&A's. Also, unique ranks could have been assigned depending on the trackers previously mentioned. All these trackers together could have made interacting on the website more enjoyable, increasing its usage and traffic.

Success Criteria Review

Criteria	Evidence	Success	Justification

1 *A way to create questions for other users to see*	 Question Form Create Your Own Question Title* Description* Group --------- Submit	Success	The main function of the website is to allow users to create and answer questions about biology which this form fulfils. This is the main function that starts the website turning into a community. The question, once made, is then displayed on the homepage and available to be found by search.
2 *A way to put forward solutions to questions asked for all users so that you can interact with the website.*	 Answers **User: j.real** The chemical process of releasing energy from organic compounds (respiratory substrates) such as glucose through oxidation. The energy released is used to combine ADP with inorganic phosphate to make ATP (energy). Respiration is a long series of enzyme-controlled reactions. **Comment on Post:** Answer* Submit	Success	This allows users to answer questions that have been asked making the site feel more active and like a web community. Users can interact with each other and work together to solve questions.
3 *A way to upvote and down vote questions and answers so that they can be organised to where the answers with more upvotes are shown first.*	**This criterion was not met.**	Failure	I was not able to implement the upvote and downvote functionality of the questions. This because I couldn't link a working button to the points field to my Question class in the time I was given to complete development.
4 *Working and able to run in any browser.*	No evidence required, check justification.	Success	The website can run on any browser as long as the server is on. This is possible because the actual rendering of the website is done using HTML which can

			be run on all common browsers like Chrome, Edge, Firefox, Opera, etc.
5 *A search bar where you can search and sort through all the questions or topics in the websites database*		Success	Gives users easier access to all the questions in the database. Quicker than scrolling through the homepage.
6 *A toolbar at the top of the website where there are links to different parts of the website which makes traversing the website easier.*	 Home Help Contact Topics Login Sign Up Search Topics Search	Success	Makes traversing the website easier for users since there are clear buttons leading to subsequent pages. Increase usability.
7 *A way to create an account on the website allowing features like saving questions so that people will be more likely to use the website more often.*	 Register Please enter your details: Username* Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only. First name Optional. Last name Optional. Email* Password* Sign Up	Success	Allows users of the website to create unique profiles that are used to save and answer questions.

Solution for Unmet criteria

Criterion 3
I was unable to create a success ranking system through upvotes and downvotes due to a strain on time when completing the development. To successfully implement this feature, I have an idea of how I would do it.

I would need to link a button on my question page to a specific function within my Question class that increases the value of the points field although it is difficult to show this on the page without refreshing. To do this I would need to integrate the use of an API called "HTMX" that deals with AJAX requests, that update the page in real time. This would allow the user to upvote and downvote questions and answers seeing the effects in real time.

Furthermore, this would allow me to implement a system where questions are displayed by descending order depending on how many points they have. This could be effectively used to create a “trending” page to show the most active and popular question at the given time.