A-Level Revision App

NEA Project

Rameen Ather

Python

NEA

[Analysis 5](#_Toc128700216)

[The problem 5](#_Toc128700217)

[Regarding the student (main recipient): 5](#_Toc128700218)

[*Whats the issue?* 5](#_Toc128700219)

[*Why are these skills important?* 6](#_Toc128700220)

[*Whats the main problem with the issue raised?* 6](#_Toc128700221)

[*What additional factors can the affect the development of these skills?* 6](#_Toc128700222)

[Regarding the teachers/examiner (secondary recipient): 6](#_Toc128700223)

[Interview 7](#_Toc128700224)

[Research for the problem 8](#_Toc128700225)

[Similar website designs 8](#_Toc128700226)

[*Physicsandmathtutor* 9](#_Toc128700227)

[*Save my exam* 9](#_Toc128700228)

[What the questions look like: 11](#_Toc128700229)

[Solution 13](#_Toc128700230)

[Who is the intended user/s in this project? 13](#_Toc128700231)

[Objectives 14](#_Toc128700232)

[Design 15](#_Toc128700233)

[Hierarchy diagram 15](#_Toc128700234)

[Class diagrams 17](#_Toc128700235)

[Class Account 18](#_Toc128700236)

[Questions() 18](#_Toc128700237)

[Mark\_received() 18](#_Toc128700238)

[Send\_email() 19](#_Toc128700239)

[Get\_image () 19](#_Toc128700240)

[Get\_best\_topic() 19](#_Toc128700241)

[help() 20](#_Toc128700242)

[Send\_answers\_to\_mark() 20](#_Toc128700243)

[store\_logs\_database() 21](#_Toc128700244)

[Log\_teaching() 21](#_Toc128700245)

[Quick\_fire\_five() 21](#_Toc128700246)

[Add\_points() 21](#_Toc128700247)

[Add\_marks\_to\_Database(): 21](#_Toc128700248)

[Database Models 22](#_Toc128700249)

[Account 22](#_Toc128700250)

[Questions 22](#_Toc128700251)

[Topics-biology 23](#_Toc128700252)

[Account\_marks 23](#_Toc128700253)

[General entity relations between the databases’ tables 24](#_Toc128700254)

[What the website will look like 26](#_Toc128700255)

[What you see first: 26](#_Toc128700256)

[signup 27](#_Toc128700257)

[Log-in 27](#_Toc128700258)

[Home (Student) 28](#_Toc128700259)

[Home (Teacher) 28](#_Toc128700260)

[Changes made from the initial idea declared in the Analysis section. 29](#_Toc128700261)

[Technical Solution 30](#_Toc128700262)

[Python 31](#_Toc128700263)

[Auth.py 31](#_Toc128700264)

[Sign-up 31](#_Toc128700265)

[Log-in 31](#_Toc128700266)

[log-out 32](#_Toc128700267)

[\_\_init\_\_.py 33](#_Toc128700268)

[Main.py 34](#_Toc128700269)

[Creating the website and database tables 34](#_Toc128700270)

[Model.py 38](#_Toc128700271)

[Database 38](#_Toc128700272)

[Account.py 39](#_Toc128700273)

[Class account 39](#_Toc128700274)

[Views.py 56](#_Toc128700275)

[HTML pages 56](#_Toc128700276)

[Sending an email to all students 61](#_Toc128700277)

[HTML Code 63](#_Toc128700278)

[Base template for HTML pages 63](#_Toc128700279)

[Sign-up page 65](#_Toc128700280)

[Student Home page (HTML) 66](#_Toc128700281)

[Teacher Home page 69](#_Toc128700282)

[Biology logs 70](#_Toc128700283)

[Maths logs 75](#_Toc128700284)

[Biology help 79](#_Toc128700285)

[Maths help 81](#_Toc128700286)

[Displaying questions 82](#_Toc128700287)

[Giving the choice to mark themselves or to submit answers 84](#_Toc128700288)

[Displaying the answers 85](#_Toc128700289)

[Uploading answer images 86](#_Toc128700290)

[Marking other people answers 87](#_Toc128700291)

[When no answers are available to mark 88](#_Toc128700292)

[Displaying stats 89](#_Toc128700293)

[Maths stats 90](#_Toc128700294)

[Chat.html 90](#_Toc128700295)

[Testing 92](#_Toc128700296)

[Video 92](#_Toc128700297)

[Links 92](#_Toc128700298)

[Video analysis 92](#_Toc128700299)

[00:00 92](#_Toc128700300)

[00:00 – 00:39 92](#_Toc128700301)

[00:58 – 01:40 92](#_Toc128700302)

[01:44 – 02:06 92](#_Toc128700303)

[02:43 – 03:08 92](#_Toc128700304)

[03:09 – 03:40 92](#_Toc128700305)

[03:40 – 03:47 92](#_Toc128700306)

[03:47 – 04:40 92](#_Toc128700307)

[04:44 – 04:58 92](#_Toc128700308)

[05:03 – 05:06 93](#_Toc128700309)

[05:19 – 06:21 (07:58 – 08:52) 93](#_Toc128700310)

[06:29 – 06-49 93](#_Toc128700311)

[06:52 – 07-51 93](#_Toc128700312)

[09:30 – 10:05 93](#_Toc128700313)

[10:05 – end 93](#_Toc128700314)

[Testing entering values to the database. 93](#_Toc128700315)

[Questions 93](#_Toc128700316)

[Communication 94](#_Toc128700317)

[Adding Images to database 95](#_Toc128700318)

[Logs 95](#_Toc128700319)

[Evaluation: 97](#_Toc128700320)

# Analysis

## The problem

### Regarding the student (main recipient):

#### *Whats the issue?*

In a lot of cases, for students studying a level, they struggle finding the motivation, method, time to study or just not having the skills to get better grades. Which can lead to the students not getting strong enough grades to secure them a spot in the university of their choice. These skills include things such as, time management, organisation, consistency, patience or just the motivation to sit down and work in the student’s own time. A further problem with students studying for a-level or in general without the use of these skills, is overstudying, which can cause the student to be stressed and anxious, eventually leading to a burn out and therefore a further downfall to their grades.

Researchers at Stanford did a study on the effect of over-studying on students. In which they found that students that completed *“more than 3.1 hours of homework per night experienced physical health problems, depression, sleep deprivation, academic stress and a lack of balance in their lives”*

[[1]](#footnote-1)To add on, the transition from GCSE to A-Level is really tough on a lot of students because of the amount of independent study they need to do outside of lesson time. The normal independent time expected off of students is one hour per subject so 3-4 hours a day. This itself puts pressure and doubt onto the student’s mind and then the student severely lacks motivation to revise efficiently to even get good grade. In 2019[[2]](#footnote-2), The Sub released an article on the A-Level maths of the year. It only took the student getting 55% to get an A and 14.7% for them to pass A-Level maths.

#### *Why are these skills important?*

The skills mentioned before are often called ‘Soft Skills’ [[3]](#footnote-3) (or transferable skills) and are generally the skills that stand out to employers when going into work, and these skills are developed from the time the person is studying. Hence the important of these skills. Although when a person obtains these skills, they often gain a sense of self-motivation which further helps the person with jobs and confidence in themselves.

#### *Whats the main problem with the issue raised?*

These skills generally take a long period of time to obtain and especially an incredible amount of motivation and patience to stick through it. With having to spend time organising yourself to be able to do productive work and still have time to have fun in life, a person generally tends to ignore spending time on organising and managing their time leading themselves to work at unreasonable hours and rushing to finish work they were given from college, without being able to do revision of their own.

#### *What additional factors can the affect the development of these skills?*

(These factors are generally out of the students control or affect the student way too much for them to be able to have the patience and develop the skills)

* Lack motivation in general weather due to mental health issues or other reasons[[4]](#footnote-4)
* Have learning disabilities, such has ADHD, autism, Dyslexia etc.
* Students that are not able to find the proper time to organise themselves and revise properly due to personal issues.

### Regarding the teachers/examiner (secondary recipient):

When it comes to teaching and being an examiner, sometimes you need to practice questions yourself or practice marking for all the hundreds of papers that will be marked by examiners and teachers in a single year. Or just even get away from the normal teaching routine and help other students not being able to do well on their own accord.

Although the main issue isnt really to do with the teachers as they already did their a-levels, they do teach it and often have a passion to help others also do well in their a-level or at the very least understand what they are learning. Often teachers want to be able to help students with their a-levels and that can be useful to a student that struggles with lack of motivation to guide them to a right path to find a way for them to get consistent.

## Interview

An interview with a biology A-level student to get an idea of how the student studies, what they do to study and how they find revision. This gave me a further idea about the problem and its effects on the students.

*How do you normally revise?*

“Read over notes, flash cards, YouTube videos and exam questions”

*Can you access these exam questions and other resources easily?*

“I mean yeah, I just look at physicsandmathtutor and get exam questions from there. Though sometimes finding specific questions can be a struggle”

*Do you know how confident you generally are in each topic for Biology?*

“I don’t know right now; I’d have to go through each topic to see and know”

*Do you struggle finding motivation to revise?*

“YES!” (Genuine reaction) “of course”

*Are you able to manage your time well when you do study?*

“If I plan out my revision properly before studying then yes, and if I don’t use my phone… ha-ha.”

*Would you say you are organised in terms of your revision?*

“In terms of my revision, sometimes? I don’t think I’m always organised so it can be hard when revising”

*After doing this interview with this student, I developed a further understanding on the problem:*

* a further conformation on how motivation is a genuine hard struggle amongst the students.
* Questions arent hard to find, but can lack being specific in what the student wants
* Students don’t generally know how confidant they are in each subject
  + being disorganised, not knowing what to focus on and what not to focus on.
* Organisation in a student’s life is needed if they want to be able to study efficiently

## Research for the problem

### Similar website designs

Websites that are meant for revision, namely physicsandmathstutor[[5]](#footnote-5) (as mentioned by the student interviewed) and savemyexams[[6]](#footnote-6). These websites are some of the more popular and largely recommend website amongst the students

#### *Physicsandmathtutor*

Graphical user interface, application, website

Description automatically generated

General description of the website:

-*This website focuses on providing exam questions, notes, flash cards on each of the listed subjects.*

Analysis of these websites:

* Although it is a great website for students, my focus is on the A-Level biology and A-Level maths side of it.
* This website has its questions stored topic and subtopic wise, making it somewhat easy to find questions
* Although the number of questions that this website has for A-level biology and maths is little and often repeated in another subtopic.
  + Which adds on to what the student said in the interview about the struggles of finding specific questions.
* This website also has notes on each topic and a general cheat sheet for each topic in maths.

#### *Save my exam*

Graphical user interface, application, website

Description automatically generated

General description of the website:

-*This website has a variety of questions from simple to hard, notes on each topic and past papers.*

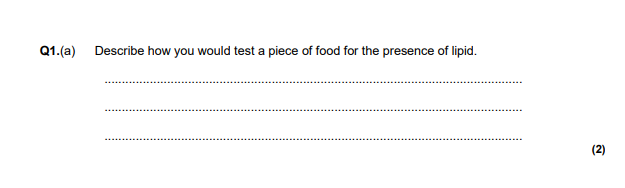
Analysis of these websites:

* Focusing on the maths and biology side of this, this a good website in terms or variety of questions which is not necessarily present in physicsandmathtutor.
* It gives you levels on difficulty on which you can choose which one to do

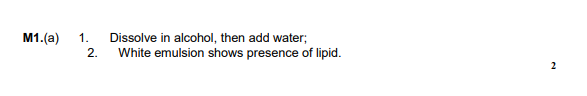
Overall, for revision this a great revision website for students.

Graphical user interface, website

Description automatically generated

What the questions look like:

They would come with their corresponding mark scheme answer, to allow the marker, either the teacher or examiner, to mark.



More examples:

There are a few instances where the questions are longer than a page, which can create problems, as it can be difficult keeping all of the question if in one image file. (Below is the corresponding answer)

Text

Description automatically generated

A close-up of a document

Description automatically generated with medium confidence

A long question would look like this after a series of screenshotting and editing have taken place.

A lot of the questions would not require editing; however, a few are longer than a page.

## Solution

My revision website will be able to help students manage their time with quick access to questions they wish and have actual teachers there that will be able to teach them. Also, will be able to manage their self-motivation with constant reminder and easy access to questions and help.

The website will be able to provide them with quick fire questions, that they are able to answer and can either mark or have a teacher/another student to mark it and be able to do them again another day to learn what you forgot.

### Who is the intended user/s in this project?

The intended end users for this website are students, specifically A-level maths, and A-level biology, due to these subjects being one of the harder subject to revise for. It is also a known fact that the jump from GCSE to A-level is harder than the jump from A-level to university, therefore a lot of students generally tend to struggle in their a-level years.

The secondary end users: Teachers will be able to use this website to teach students and mark stuff being sent to them.

## Objectives

* Registration and login system
  + Your able to pick from two different types of accounts: Student, Teacher
* 4 different types of accounts
  + **Student** 
    - on this account you can do exam questions, generated based on what topic you ask for.
    - log your confidence in each topic.
    - be able to be ‘taught’ by teacher accounts.
  + **Teacher** 
    - Can also do questions that will be generated based on what topic they ask for.
    - Teacher accounts will be able to teach students on listed topics.
      * From which they can log what they have taught on to a page to keep track.
  + Examiner
    - Can do questions and mark questions.
  + Admin
    - Moderate all the website and keep it safe.
* Database of questions
  + The questions would be stored in tables to their according label, for example questions about lipids would be stored in a table called questions and have a topic value of lipid in the database
  + The corresponding mark scheme would also be stored in the same record in the table.
* Questions generated
  + The system will be able to generate a random test made up of the certain topic the user asked for
  + The corresponding mark scheme will also be loaded.
    - When asked for, it will be shown on the screen for the student to mark, if they want to mark their own work.
  + Will be able to generate five random questions if a specific topic is not given
    - These questions will be saved and sent to the student to do again another day.
* An emailing system
  + Sends an email every day to all the student accounts with the recommended revision subject and a reminder to study.
* A confidence logging system
  + Student accounts are able to log confidence level in each topic through a series of range inputs and this will be stored until next time changed
  + Two separate ones, one for biology and one for maths
  + On another page, the mean of all the logs entered so far for each topic would be shown using a graph alongside the standard deviation.
* Teacher accounts are able to log what they have taught to the students.
  + Through a video call/chatting service
* Being able to add images
  + The user will be able to add their answers they have done in their book as images
  + Which will be sent to another user to mark.
* A file of motivational quotes
  + That will be displayed on the home page and is random each time.

# Design

## Hierarchy diagram

This is a rough design/overview of the overall job of the website, with what accounts can do what and going into slight detail on each action.

Diagram, Teams

Description automatically generated

The green at the top represents the accounts and the classes I’m going to make to each.

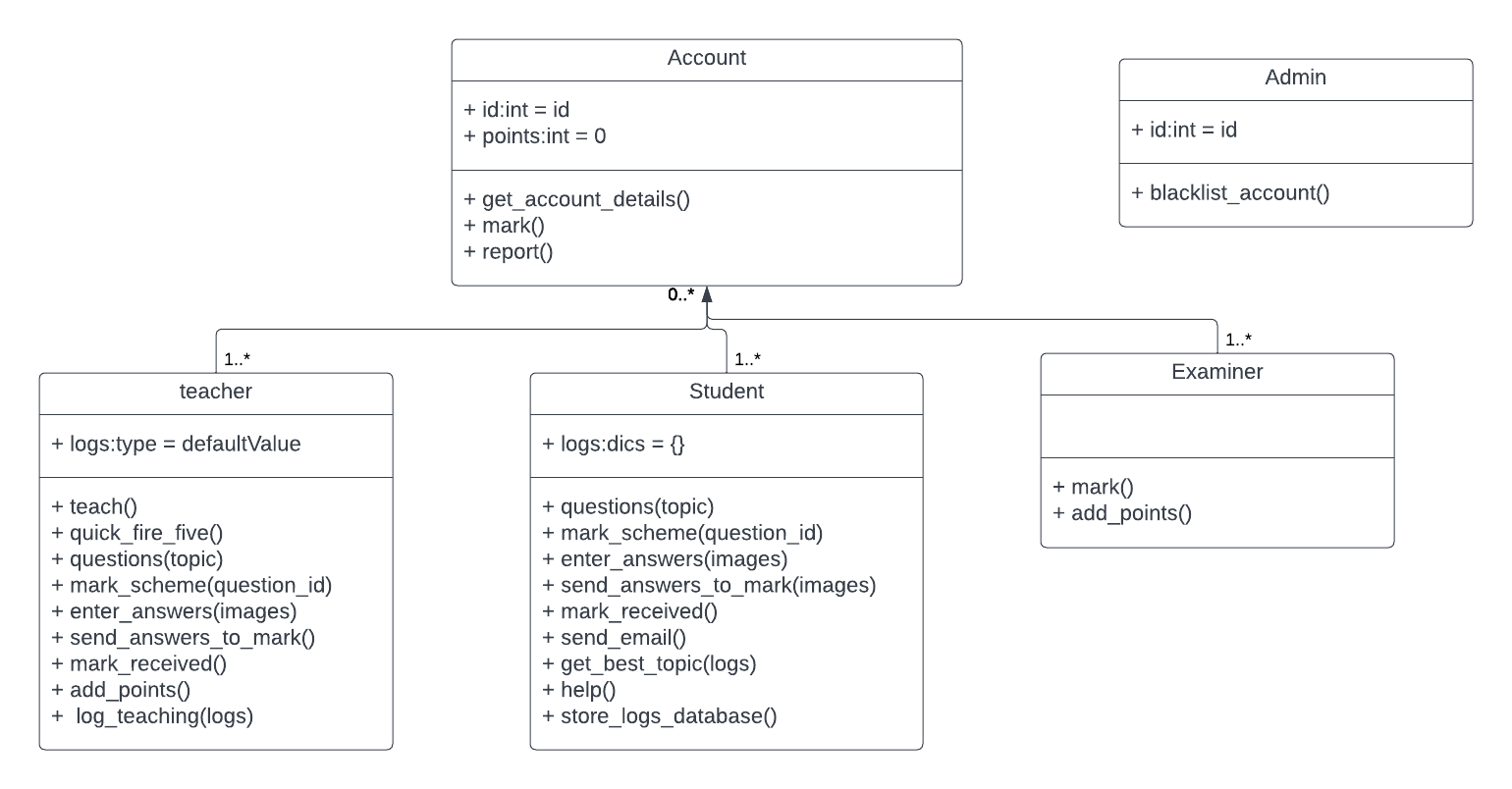
The light blue represents what main thing each account can do, the ‘methods’ of the accounts and which one links with what accounts, for example, both student and teacher are able to do a quiz.

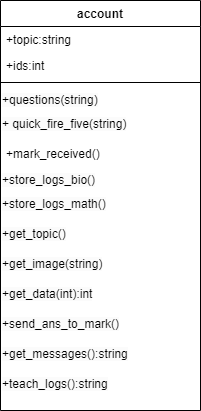
The blue represents what happens in each ‘method’. What the accounts will be able to in a general description.

The green -> Database

(I didn’t end up finishing the whole design)

## Class diagrams

This is how the general layout of the class system will be. This is what all the accounts will be able to.



Ended up changing into a single account class, from the former design. My main focus was the student accounts and there what they were supposed to do, and teacher and examiner generally had the same attributes so no need to make another class. For the admin class I didn’t have enough time to learn to make an admin system for the website.

### Class Account



#### Questions()

In this method the student will be able to give a topic and using database query, a series of questions will be picked out from the database containing the questions

SELECT id, question, answer

FROM questions

WHERE topic = topic\_selected

ORDERBY random

With randomly generated numbers 5 or so questions will be picked, these questions and answers will the converted into png images and displayed on corresponding pages for the questions and answers.

This method also calls another method which clears all instances of the pervious questions and answers from the website pages.

Then be put onto another table of the database: account\_marks, which will then record the date these were generated.

Inserting into the table with example values

INSERT INTO account\_marks (done\_id, acc\_id, QA\_id, date\_entered, topic)

VALUES (1,self.id,23, 05/10/22, question)

Marks will automatically be set to none until the questions have been marked by the user themselves or another account and that has been entered into the system then answer attribute will be filled when the someone has marked the questions and have submitted them into the system.

#### Mark\_received()

When the marking of the questions is done by another person or the person themselves, they will submit marks received in each question and this method will then update the database table: account\_marks mark section to corresponding marking.

UPDATE account\_marks

SET marks = 3

WHERE acc\_id = user\_id and QA\_id = QA\_id

There will be points where the user does the same questions more than one time on different occasions and both of them will be saved as they can show a learning curve for the teacher when seeing the results from the students attempt at the questions.

#### Send\_email()

This method will take the student accounts email and send them an email reminding them to revise

SELECT name, email, topic

FROM account

WHERE status = student

An exampere email would look like:

“Hi [name], today seems like a good day to revise [topic\_prioritity]. Get those scores up!”

#### Get\_image ()

This will allow the users’ answers to be stored in the database to be sent to another account to be marked. This happens by the account entering images of their answers and using this method they will be stored in a database table ans\_images.

Updating the record with example values

UPDATE acc\_images

SET image = answer\_image

WHERE done\_id = 2

#### Get\_best\_topic()

This method will go through an algorithm to find the most ideal topic for the student to revise for the week.

Algorithm to be decided, not sure if I want to use this one or make a better one so I’m leaving this with a simple version and if I have time when coding ill try to make it better. Right now, I’m working with a very basic algorithm that finds the mean of each documented confidence level and the finds the lowest out of all them and that’s the topic the person needs to revise.

SUBROUTINE best\_topic(dics)

Temp <- 0

Optimal <- none

TEMPLIST <- []

FOR items IN Len(dics)

FOR numbers IN dics[1]

Temp <- temp + dics[items][number]

ENDFOR

Templist[items] <- temp/LEN(dics[items])

Temp <- 0

ENDFOR

Optimal <- low(templist)

END SUBROUTINE

SUBROUTINE low(list)

Low <- 0

Index <- 0

FOR items IN list

IF items -1 < items THEN

Low <- list[item-1]

Index <- item – 1

ELSE

Low <- list[item]

Index <- item

ENDIF

END FOR

RETURN index

END SUBROUTINE

#### help()

Goes through all the teachers in the database and selects a teacher and sends the teacher an email to confirm or deny their availability to teach.

SELECT \*

FROM account

WHERE status = teacher

#### Send\_answers\_to\_mark()

This method will be user when someone asks to mark some questions and it picks a question done by someone to mark and gives them the mark scheme to mark it.

SELECT acc\_id, email, question\_done\_id

FROM account, user , account\_answer

ORDERBY RANDOM()

#### store\_logs\_database()

Once the student enters how confident they are in the logs page, this method will be called to allow all of them to be taken and be put into a database

Entering into the table with example values

INSERT INTO topic 1 (acc\_id, date, lipids, carbohydrates, proteins and enzymes, DNA , ATP, water and inorganic molecules )

VALUES (user\_id, current\_date,23,54,75,87,34,88)

These values then can be taken further and can make a graph out of them and other data analytical stuff.



#### Log\_teaching()

This method will update what the teacher taught to the student and store into a database table: teaching\_logs

INSERT INTO teaching\_logs(acc\_id, student\_acc\_id, teach\_log, date, topic)

VALUES (self.id, student\_id, 1, 05/10/22, lipids)

Once logged another method is ran called quick\_fire\_five() to generate 5 questions based on the topic the teacher taught.

#### Quick\_fire\_five()

Generates questions after the teacher has taught a student and gives it to the student to do while also storing to the account\_answer database table, with the students id.

SELECT questions.questions, questions.answers, questions.id, questions.mark

FROM teaching\_logs, questions

WHERE teaching\_logs.topic = questions.topic

ORDERBY questions.mark

#### Add\_points()

This method awards the user points as they mark questions, so the more they mark they more points will be added to the total.

#### Add\_marks\_to\_Database():

Takes the users entered values for there marks and adds them to the database table account\_marks

UPDATE marks

FROM account\_marks

WHERE question\_done\_id = 2 and user = 1

## Database Models

This is to show how the data will be stored in the databases and how the tables will be organised, what table will have what attribute and what are the primary and foreign keys.

#### Account

The database table for saving the details about an account and all the restriction placed to secure data and the account:

-The user will not be able to create an account with an email that already exists

-The password has to be more than 8 characters

-The email, name, last name all much be more than one characters

CREATE TABLE account(

ID INT NOT NULL PRIMARY KEY,

FirstName VARCHAR (50) NOT NULL,

Surname VARCHAR (50) NOT NULL,

email VARCHAR (100) NOT NULL,

password VARCHAR (50) NOT NULL,

status VARCHAR (10) NOT NULL,

points INT

)

The password will be encrypted to keep safe :D

#### Questions

CREATE TABLE questions(

Qa\_id INT NOT NULL PRIMARY KEY,

Question BLOB NOT NULL,

Answer BLOB NOT NULL,

Marks INT NOT NULL,

)

CREATE TABLE priority\_topic(

Acc\_idd int NOT NULL

Topic VARCHAR(50)

FOREIGN KEY (acc\_id) REFERENCES (account)

)

#### Topics-biology

All the tables for topic logs will be created in similar manner to the example below:

CREATE TABLE topic1(

date\_entered DATE NOT NULL PRIMARY KEY,

acc\_id INT NOT NULL

lipids INT NOT NULL,

carbohydrates INT NOT NULL,

protein and enzymes INT NOT NULL,

DNA INT NOT NULL,

ATP INT NOT NULL,

Water and inorganic ions INT NOT NULL,

FOREIGN KEY (acc\_id) REFERENCES (account)

)

#### Account\_marks

CREATE TABLE account\_marks (

Done\_id INT NOT NULL PRIMARY KEY

Date\_entered DATE NOT NULL PRIMARY KEY,

Acc\_id INT NOT NULL FOREIGN KEY,

Qa\_id INT NOT NULL FOREIGN KEY,

Marks INT

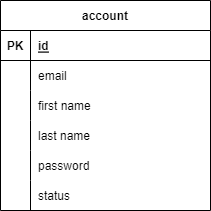
FOREIGN KEY (acc\_id) REFERENCE (account)

FOREIGN KEY (QA\_ID) REFERENCES (questions)

)

### General entity relations between the databases’ tables

(Couldn’t fit all in one page)

Graphical user interface

Description automatically generated



## What the website will look like

This won’t be 100% accurate to the final product as I’m not going in depth into colour designing and other frontend designing other than what the general website will look like overall. This was design using a website[[7]](#footnote-7) and is not coded.

### What you see first:

During the designing process I realised that there might be some users that do not wish to do one of the subject, maths, or biology, and therefore having access to questions and help from that subject would be useless, that’s why I added an option at the start to let you choose whether you want to do one or both the subjects.

This is what would be shown first, it gives you an option to choose whether you want to do math, biology, or both and then leads you to be able to sign in.

Graphical user interface, text, application, email

Description automatically generated

**Login**

### signup

Sign-up page. There is an option for the user to pick what type of account they want, student, teacher or examiner and then lead you to a home page corresponding to the account.

Graphical user interface, application

Description automatically generated

### Log-in

Log-in page. User logs in.

Graphical user interface, application

Description automatically generated

### Home (Student)

This will generally be how the home page looks and what you will be able to do with it. The logs button will lead you to a page that will allow you to pick between a range of how confident you are in each topic. The help will lead you to a page where you will get put in touch with a teacher to help you with whatever the student needs. Lastly the quiz button will lead you to a page where you can pick what kind of question you wish for.

Graphical user interface, application

Description automatically generated

### Home (Teacher)

Graphical user interface, application

Description automatically generated

#### Changes made from the initial idea declared in the Analysis section.

Text

Description automatically generated

After attempting to code a video chat system and a live text chatting system and getting this error, I ended up changing the whole thing to be database based.

Where the user can look at a dynamic list of emails of other accounts and select one and send a message and then the message is displayed on the page for the recipient.

This is where I got to using twilo.

Graphical user interface, application

Description automatically generated

# Technical Solution

(Some of the skills and their references)

|  |  |
| --- | --- |
| *Technical skills* | *Reference of it in the code (pages and the name of the heading on the page)* |
| Database structure | 33-37 -> Main.py and Model.py |
| Aggregate SQL functions | 38-54 -> account.py |
| Cross-table parameterised SQL | 38-54 - > account.py |
| Files reading | View.py and account.py |
| Simple OOP | 37-54 -> account.py |
| Server-side scripting using request and response objects | 37-54 and 54-58 -> account.py and views.py |
| Parsing JSON/XML | Pg 81 -> Account.py and javascript in html |

## Python

### Auth.py

#### Sign-up

from flask import Blueprint, render\_template, request, flash, redirect, url\_for

from models import User

from werkzeug.security import generate\_password\_hash, check\_password\_hash

from \_\_init\_\_ import db

from flask\_login import login\_user, login\_required, logout\_user, current\_user

from account import account

import re

@auth.route('/sign-up', methods=['GET', 'POST'])

def sign\_up():

if request.method == 'POST':

email = request.form.get('email')

first\_name = request.form.get('firstName')

last\_name = request.form.get('lastname')

password1 = request.form.get('password1')

password2 = request.form.get('password2')

status = request.form.get('status')

email\_pattern = "[a-zA-Z0-9]+@[gmail|outlook]+\.(com)"

user = User.query.filter\_by(email=email).first()

if user:

flash('Email already exists. ', category='error')

elif not (re.search(email\_pattern,email)):

flash('Email invalid!', category = 'error')

elif len(email) < 4:

flash('Email must be more than 3 characters.', category='error')

elif len(first\_name) < 2:

flash('First name must be greater than 1 character.', category='error')

elif len(last\_name) < 2:

flash('Last name most be greater than 1 character.', category = 'error')

elif password1 != password2:

flash("Passwords don't match.", category='error')

elif len(password1) < 8:

flash('Password must be at least 8 characters.', category='error')

elif status == None:

flash('Please pick a status label.', category = 'error')

else:

new\_user = User(email=email, first\_name=first\_name, last\_name = last\_name, status = status , password=generate\_password\_hash(

password1, method='sha256'))

db.session.add(new\_user)

db.session.commit()

login\_user(new\_user, remember=True)

if status == "teacher":

flash('Account created!', category='success')

return redirect(url\_for('views.teacher\_home'))

elif status == "student":

flash('Account created!', category='success')

return redirect(url\_for('views.student\_home'))

return render\_template("sign\_up.html", user=current\_user)

#### Log-in

@auth.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

email = request.form.get('email')

password = request.form.get('password')

user = User.query.filter\_by(email=email).first()

if user:

if check\_password\_hash(user.password, password):

#getting the status to direct the account to their correct home page

sta = str(user.status)

if sta == "teacher":

flash('Logged in successfully!', category='success')

login\_user(user, remember=True)

return redirect(url\_for('views.teacher\_home'))

elif sta == "student":

flash('Logged in successfully!', category='success')

login\_user(user, remember=True)

return redirect(url\_for('views.student\_home'))

# Send any undelivered messages to the user

connected\_user[request.sid] = user

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

cursor.execute('''SELECT \* FROM messages WHERE recipient\_id=?''', (user['id'],))

messages = cursor.fetchall()

for message in messages:

request.namespace.emit('message', message['message'])

# Delete the messages from the database

#cursor.execute('''DELETE FROM messages WHERE recipient\_id=?''', (user['id'],))

else:

flash('Incorrect password, try again.', category='error')

else:

flash('Email does not exist.', category='error')

return render\_template("login.html", user=current\_user)

#### log-out

@auth.route('/logout')

@login\_required

def logout():

logout\_user()

return redirect(url\_for('views.main\_page'))

### \_\_init\_\_.py

from flask\_socketio import SocketIO, send

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager

from flask import Flask

from os import path

db = SQLAlchemy()

DB\_NAME = "database.db"

def create\_app():

app = Flask(\_\_name\_\_)

app.config['SECRET\_KEY'] = 'mymathsapp'

app.config['SQLALCHEMY\_DATABASE\_URI'] = f'sqlite:///{DB\_NAME}'

db.init\_app(app)

from views import views

from auth import auth

app.register\_blueprint(views, url\_prefix='/')

app.register\_blueprint(auth, url\_prefix='/')

from models import User

create\_database(app)

login\_manager = LoginManager()

login\_manager.login\_view = 'auth.login'

login\_manager.init\_app(app)

@login\_manager.user\_loader

def load\_user(id):

return User.query.get(int(id))

return app

def create\_database(app):

if not path.exists('/' + DB\_NAME):

db.create\_all(app=app)

print('Created Database!')

### Main.py

#### Creating the website and database tables

from \_\_init\_\_ import create\_app, create\_database

from flask\_socketio import SocketIO, emit, send

from datetime import date, datetime, timedelta

from flask\_sqlalchemy import SQLAlchemy

from flask\_login import LoginManager

from account import account

from flask import Flask

from more\_def import \*

from os import path

import sqlite3

#creating the app :P

app = create\_app()

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

#Creating the questions table

cursor.execute("""

CREATE TABLE IF NOT EXISTS questions (id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,

topic TEXT NOT NULL,

question BLOP NOT NULL,

answer BLOP NOT NULL,

marks INTEGER NOT NULL)""")

#Create the "users" and "messages" tables if they don't exist

cursor.execute("""

CREATE TABLE IF NOT EXISTS messages

(sender\_id INTEGER NOT NULL,

recipient\_id INTEGER NOT NULL,

message text NOT NULL)""")

#storing the marks, the user got after completing questions

cursor.execute("""

CREATE TABLE IF NOT EXISTS acc\_mark (question\_done\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

acc\_id INTEGER NOT NULL,

qa\_id INTEGER NOT NULL,

marks INTEGER ,

FOREIGN KEY(acc\_id)

REFERENCES user (id),

FOREIGN KEY (qa\_id)

REFERENCES questions (id),

PRIMARY KEY (question\_done\_id,date\_entered))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS time\_taken (question\_done\_id INTEGER NOT NULL,

time INTEGER,

FOREIGN KEY(question\_done\_id)

REFERENCES account\_marks (question\_done\_id))

""")

#temporarily storing the question image

cursor.execute("""

CREATE TABLE IF NOT EXISTS account\_answer (question\_done\_id INTEGER NOT NULL,

user\_answers BLOP,

FOREIGN KEY(question\_done\_id)

REFERENCES account\_marks (question\_done\_id))""")

#creating the tables for storing logs

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic1 (t1\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

lipid INTEGER NOT NULL,

carbohydrates INTEGER NOT NULL,

protein\_and\_enzymes INTEGER NOT NULL,

DNA INTEGER NOT NULL,

ATP INTEGER NOT NULL,

Water\_and\_inorganic\_ions INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic2 (t2\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

cell\_structure INTEGER NOT NULL,

transport\_across\_membrane INTEGER NOT NULL,

cell\_cycle INTEGER NOT NULL,

immune\_system INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic3 (t3\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

gas\_exchange INTEGER NOT NULL,

digestion\_and\_absorption INTEGER NOT NULL,

mass\_transport INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic4 (t4\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

protein\_synthesis INTEGER NOT NULL,

biodiversity INTEGER NOT NULL,

genetic\_diversity INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic5 (t5\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

photosynthesis INTEGER NOT NULL,

respiration INTEGER NOT NULL,

nitrogen\_cycle INTEGER NOT NULL,

energy\_and\_ecosystem INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic6 (t6\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

response\_to\_stimuli INTEGER NOT NULL,

nervous\_coordination\_and\_muscles INTEGER NOT NULL,

homeostasis INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic7 (t7\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

inherited\_change INTEGER NOT NULL,

population\_and\_evolution INTEGER NOT NULL,

population\_in\_ecosystems INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS topic8 (t8\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

gene\_expression INTEGER NOT NULL,

recombinant\_DNA\_technology INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS pure (pure\_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

proof INTEGER NOT NULL,

algebra\_and\_functions INTEGER NOT NULL,

coordinate\_geometry INTEGER NOT NULL,

sequences\_and\_series INTEGER NOT NULL,

trigonometry INTEGER NOT NULL,

exponentials\_and\_logarithms INTEGER NOT NULL,

differentiation INTEGER NOT NULL,

integration INTEGER NOT NULL,

numerical\_methods INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS mech (mech1 INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

vectors INTEGER NOT NULL,

quantities\_and\_units\_in\_mechanics INTEGER NOT NULL,

kinematics INTEGER NOT NULL,

forces\_and\_Newton’s\_laws INTEGER NOT NULL,

moments INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

cursor.execute("""

CREATE TABLE IF NOT EXISTS stats (stats1 INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,

account\_id INTEGER NOT NULL,

date\_entered DATE NOT NULL,

statistical\_sampling INTEGER NOT NULL,

data\_presentation\_and\_interpretation INTEGER NOT NULL,

probability INTEGER NOT NULL,

statistical\_distributions INTEGER NOT NULL,

statistical\_hypothesis\_testing INTEGER NOT NULL,

FOREIGN KEY(account\_id)

REFERENCES user (id))

""")

#Sends email at 6

now = datetime.now()

if now.hour == 18 and now.minute == 00 and now.second == 00:

send\_email()

conn.commit()

cursor.close()

conn.close()

#---------------------------------------------------------------------------------------#

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

### Model.py

from flask\_login import UserMixin

from sqlalchemy.sql import func

from \_\_init\_\_ import db

class User(db.Model, UserMixin):

id = db.Column(db.Integer, primary\_key=True)

email = db.Column(db.String(150), unique=True)

password = db.Column(db.String(150))

first\_name = db.Column(db.String(150))

last\_name = db.Column(db.String(150))

status = db.Column(db.String(150))

class post(db.Model):

id = db.Column(db.Integer, primary\_key=True)

student\_email = db.Column(db.String(150), nullable = False)

student\_name = db.Column(db.String(150), nullable = False)

taught = db.Column(db.String(150), nullable = False)

date\_entered = db.Column(db.DateTime(timezone = True), default = func.now())

teacher = db.Column(db.Integer, db.ForeignKey('user.id',ondelete = "CASCADE"), nullable = False)

#### Graphical user interface, text, application Description automatically generatedDatabase

-All the tables in the database.

### Account.py

#### Class account

from flask import Blueprint, render\_template, request, flash, jsonify

from flask\_login import login\_required, current\_user

from datetime import date , datetime, timedelta

from models import User, post

from \_\_init\_\_ import db

import os

import csv

import json

import numpy

import base64

import random

import smtplib

import sqlite3

import statistics

import pandas as pd

import plotly.express as px

import matplotlib.pyplot as plt

from matplotlib.animation import FuncAnimation

class account:

#account

def \_\_init\_\_(self):

self.\_\_current\_question\_ids = []

self.\_\_last\_question\_done\_id = 0

self.\_\_current\_question\_done\_ids = []

self.marking = 0

self.\_\_topic = None

def test(self):

print("hello test :P")

user = current\_user

print(user.id)

#for question ids

def get\_current\_q\_ids(self):

return self.\_\_current\_question\_ids

def add\_current\_q\_ids(self,id):

self.\_\_current\_question\_ids.append(id)

def del\_current\_q\_ids(self):

self.\_\_current\_question\_ids = []

#for question done ids

def get\_current\_q\_done\_ids(self):

return self.\_\_current\_question\_done\_ids

def add\_current\_q\_done\_ids(self,id):

self.\_\_current\_question\_done\_ids.append(id)

def del\_current\_q\_done\_ids(self):

self.\_\_current\_question\_done\_ids = []

#for the last question done id in the database

def get\_last\_question\_id(self):

return self.\_\_last\_question\_done\_id

def set\_last\_question(self,id):

self.\_\_last\_question\_done\_id = id

def add\_last\_question(self,add):

self.\_\_last\_question\_done\_id+=add

def set\_topic(self,topic):

self.\_\_topic = topic

def \_get\_topic(self):

return self.\_\_topic

def clear\_all(self):

user = current\_user

if os.path.exists(f'static/questions/q{user.id}0.png'):

os.remove(f'static/questions/q{user.id}0.png')

os.remove(f'static/questions/q{user.id}0-ans.png')

if os.path.exists(f'static/questions/q{user.id}1.png'):

os.remove(f'static/questions/q{user.id}1.png')

os.remove(f'static/questions/q{user.id}1-ans.png')

if os.path.exists(f'static/questions/q{user.id}2.png'):

os.remove(f'static/questions/q{user.id}2.png')

os.remove(f'static/questions/q{user.id}2-ans.png')

if os.path.exists(f'static/questions/q{user.id}3.png'):

os.remove(f'static/questions/q{user.id}3.png')

os.remove(f'static/questions/q{user.id}3-ans.png')

if os.path.exists(f'static/questions/q{user.id}4.png'):

os.remove(f'static/questions/q{user.id}4.png')

os.remove(f'static/questions/q{user.id}4-ans.png')

self.del\_current\_q\_ids()

self.del\_current\_q\_done\_ids()

self.set\_last\_question(0)

def questions(self,topic):

"""Gets the question of the topic the user wants"""

#clears all instances of previous questions

self.clear\_all()

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

m = cursor.execute (f"""

SELECT id,question, answer

FROM questions

WHERE topic = ?

ORDER BY RANDOM()

""",[topic])

m = m.fetchmany(5)

user = current\_user

for i in range(5):

try:

self.add\_current\_q\_ids(m[i][0])

with open(f'static/questions/q{user.id}{i}.png','wb') as q:

q.write(m[i][1])

print("tried" )

with open(f'static/questions/q{user.id}{i}-ans.png','wb') as a:

a.write(m[i][2])

except:

i=-1

print(self.get\_current\_q\_ids())

conn.commit()

user = current\_user

current\_date = date.today()

user\_id = user.id

cursor.execute("""SELECT question\_done\_id

FROM acc\_mark

ORDER BY question\_done\_id DESC limit 1; """)

last\_id = cursor.fetchone()

print(self.get\_last\_question\_id())

if last\_id == None:

self.last\_question\_id = 1

else:

self.add\_last\_question(last\_id[0] + 1)

print(self.get\_last\_question\_id())

conn.commit()

self.add\_current\_q\_ids(self.get\_last\_question\_id())

for i in range(5):

print(self.get\_current\_q\_ids())

cursor.execute("""INSERT INTO acc\_mark (question\_done\_id, acc\_id, qa\_id , date\_entered) VALUES(?,?,?,?)""" ,

(self.get\_last\_question\_id(),user\_id,self.get\_current\_q\_ids()[i],current\_date))

self.add\_last\_question(1)

self.add\_current\_q\_ids(self.get\_last\_question\_id())

conn.commit()

cursor.close()

conn.close()

def quick\_fire\_five(self):

"""Picks 5 questions randomly for the user"""

clear\_all()

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

m = cursor.execute("""

SELECT id

FROM question

""")

data = cursor.fetchall()

conn.commit()

self.clear\_all()

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

m = cursor.execute (f"""

SELECT id,question, answer

FROM questions

ORDER BY RANDOM()

""")

m = m.fetchmany(5)

user = current\_user

for i in range(5):

try:

self.add\_current\_q\_ids(m[i][0])

with open(f'static/questions/q{user.id}{i}.png','wb') as q:

q.write(m[i][1])

with open(f'static/questions/q{user.id}{i}-ans.png','wb') as a:

a.write(m[i][2])

except:

i=-1

conn.commit()

current\_date = date.today()

user\_id = user.id

cursor.execute("""SELECT \*

FROM acc\_marks

ORDER BY question\_done\_id DESC limit 1; """)

temp = cursor.fetchone()

if temp == None:

self.set\_last\_question(1)

else:

self.set\_last\_question(temp[0])

self.add\_current\_q\_ids(self.get\_last\_question\_id())

conn.commit()

for i in range(5):

print(self.get\_current\_q\_ids())

cursor.execute("""INSERT INTO acc\_marks (question\_done\_id, acc\_id, qa\_id , date\_entered) VALUES(?,?,?,?)""" ,

(self.get\_last\_q\_id(),user\_id,self.get\_current\_q\_ids()[i],current\_date))

self.set\_last\_question(1)

self.add\_current\_q\_ids(self.get\_last\_question\_id())

conn.commit()

def mark\_received(self):

if request.method == 'POST':

q1 = request.form.get('q1')

q2 = request.form.get('q2')

q3 = request.form.get('q3')

q4 = request.form.get('q4')

q5 = request.form.get('q5')

mark\_recieved =[]

mark\_recieved.append(q5)

mark\_recieved.append(q4)

mark\_recieved.append(q3)

mark\_recieved.append(q2)

mark\_recieved.append(q1)

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

current\_date = date.today()

user\_id = user.id

cursor.execute("""SELECT question\_done\_id

FROM acc\_mark

ORDER BY question\_done\_id DESC limit 5; """)

temp = cursor.fetchall()

for i in range(0,5):

cursor.execute(f"""

UPDATE acc\_mark

SET marks = ?

WHERE acc\_id = ? and question\_done\_id = ?

""",[mark\_recieved[i],user\_id,temp[i][0]])

conn.commit()

cursor.close()

conn.close()

def store\_logs\_database(self):

"""Stores the logs entered to the database"""

if request.method == 'POST':

lipids = request.form.get('lipids')

protein = request.form.get('protein')

water = request.form.get('water')

atp = request.form.get('atp')

Carbohydrates = request.form.get('Carbohydrates')

dna = request.form.get('dna')

Cellstructure = request.form.get('cellstructure')

transportacrossmembrane = request.form.get('transportacrossmembrane')

cellcycle = request.form.get('cellcycle')

immunesystem = request.form.get('immunesystem')

gasexchange = request.form.get('gasexchange')

digestionandabsorption = request.form.get('digestionandabsorption')

masstransport = request.form.get('masstransport')

proteinsynthesis = request.form.get('proteinsynthesis')

biodiversity = request.form.get('biodiversity')

geneticdiversity = request.form.get('geneticdiversity')

photosynthesis = request.form.get('photosynthesis')

respiration = request.form.get('respiration')

nitrogencycle = request.form.get('nitrogencycle')

energyandecosystem = request.form.get('energyandecosystem')

responsetostimuli = request.form.get('responsetostimuli')

nervouscoordinationandmuscles = request.form.get('nervouscoordinationandmuscles')

homeostasis = request.form.get('homeostasis')

inheritedchange = request.form.get('inheritedchange')

populationandevolution = request.form.get('populationandevolution')

populationinecosystems = request.form.get('populationinecosystems')

geneexpression = request.form.get('geneexpression')

recombinantDNAtechnology = request.form.get('recombinantDNAtechnology')

#will be stored in database :D this is temp

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

current\_date = date.today()

user\_id = user.id

#insert into topic 1

cursor.execute("""

INSERT INTO topic1 (account\_id, date\_entered, lipid, carbohydrates, protein\_and\_enzymes, DNA, ATP, Water\_and\_inorganic\_ions) VALUES (?,?,?,?,?,?,?,?)""",

(user\_id, current\_date, lipids, Carbohydrates, protein, dna, atp, water))

#insert into topic 2

cursor.execute("""

INSERT INTO topic2 (account\_id, date\_entered, cell\_structure, transport\_across\_membrane, cell\_cycle, immune\_system) VALUES (?,?,?,?,?,?)""",

(user\_id, current\_date, Cellstructure, transportacrossmembrane, cellcycle, immunesystem))

#insert into topic 3

cursor.execute("""

INSERT INTO topic3 (account\_id, date\_entered, gas\_exchange, digestion\_and\_absorption, mass\_transport) VALUES (?,?,?,?,?)""",

(user\_id, current\_date, gasexchange, digestionandabsorption, masstransport))

#insert into topic 4

cursor.execute("""

INSERT INTO topic4 (account\_id, date\_entered, protein\_synthesis, biodiversity, genetic\_diversity) VALUES (?,?,?,?,?)""",

(user\_id, current\_date, proteinsynthesis, biodiversity, geneticdiversity))

#insert into topic 5

cursor.execute("""

INSERT INTO topic5 (account\_id, date\_entered, photosynthesis, respiration, nitrogen\_cycle, energy\_and\_ecosystem) VALUES (?,?,?,?,?,?)""",

(user\_id, current\_date, photosynthesis, respiration, nitrogencycle, energyandecosystem))

#insert into topic 6

cursor.execute("""

INSERT INTO topic6 (account\_id, date\_entered, response\_to\_stimuli, nervous\_coordination\_and\_muscles, homeostasis) VALUES (?,?,?,?,?)""",

(user\_id, current\_date, responsetostimuli, nervouscoordinationandmuscles, homeostasis))

#insert into topic 7

cursor.execute("""

INSERT INTO topic7 (account\_id, date\_entered, inherited\_change, population\_and\_evolution, population\_in\_ecosystems) VALUES (?,?,?,?,?)""",

(user\_id, current\_date, inheritedchange, populationandevolution, populationinecosystems))

#insert into topic 8

cursor.execute("""

INSERT INTO topic8 (account\_id, date\_entered, gene\_expression, recombinant\_DNA\_technology ) VALUES (?,?,?,?)""",

(user\_id, current\_date, geneexpression, recombinantDNAtechnology))

conn.commit()

cursor.close()

conn.close()

def get\_forms\_maths(self):

if request.method == 'POST':

proof = request.form.get('proof')

algebraandfunctions = request.form.get('algebraandfunctions')

coordinategeometry = request.form.get('coordinategeometry')

sequencesandseries = request.form.get('squencesandseries')

trigonometry = request.form.get('trigonometry')

exponentialsandlogarithms = request.form.get('exponentialsandlogarithms')

differentiation = request.form.get('differentiation')

integration = request.form.get('integration')

numericalmethods = request.form.get('numericalmethods')

vectors = request.form.get('vectors')

quantitiesandunitsinmechanics = request.form.get('quantitiesandunitsinmechanics')

kinematics = request.form.get('kinematics')

forcesandnewtonslaws = request.form.get('forcesandnewtonslaws')

moments = request.form.get('moments')

statisticalsampling = request.form.get('statisticalsampling')

datapresentationandinterpretation = request.form.get('datapresentationandinterpretation')

probability = request.form.get('probability')

statisticaldistributions = request.form.get('statisticaldistributions')

statisticalhypothesistesting = request.form.get('statisticalhypothesistesting')

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

current\_date = date.today()

user\_id = user.id

#insert into pure

cursor.execute("""

INSERT INTO pure (account\_id, date\_entered, proof, algebra\_and\_functions, coordinate\_geometry, sequences\_and\_series, trigonometry, exponentials\_and\_logarithms, differentiation, integration,numerical\_methods) VALUES (?,?,?,?,?,?,?,?,?,?,?)""",

(user\_id, current\_date, proof, algebraandfunctions, coordinategeometry, sequencesandseries, trigonometry, exponentialsandlogarithms,differentiation, integration,numericalmethods))

#insert into mech

cursor.execute("""

INSERT INTO mech (account\_id, date\_entered, vectors, quantities\_and\_units\_in\_mechanics, kinematics, forces\_and\_Newton’s\_laws, moments) VALUES (?,?,?,?,?,?,?)""",

(user\_id, current\_date, vectors, quantitiesandunitsinmechanics, kinematics, forcesandnewtonslaws, moments))

#insert into stats

cursor.execute("""

INSERT INTO stats (account\_id, date\_entered, statistical\_sampling, data\_presentation\_and\_interpretation, probability, statistical\_distributions, statistical\_hypothesis\_testing) VALUES (?,?,?,?,?,?,?)""",

(user\_id, current\_date, statisticalsampling, datapresentationandinterpretation, probability, statisticaldistributions, statisticalhypothesistesting))

conn.commit()

cursor.close()

conn.close()

def get\_teacher(self):

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

status = "teacher"

cursor.execute("""

SELECT id, email, first\_name, last\_name

FROM user

WHERE status = ?

ORDER BY RANDOM()""",[status])

teach = cursor.fetchone()

conn.commit()

cursor.close()

conn.close()

return teach

def get\_topic(self):

if request.method == 'POST':

self.\_\_topic = request.form.get('slct2')

if self.\_\_topic == None:

self.quick\_fire\_five()

else:

print("works")

self.questions(self.\_\_topic)

print(self.\_\_topic)

def user\_image\_db(self,filename):

"""Inserts images uploaded by the user into the database"""

with open(f'static/user\_ans/{filename}' , 'rb') as u:

image\_ans = u.read()

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

cursor.execute("""

INSERT INTO account\_answer(question\_done\_id, user\_answers) VALUES (?,?)""",

(self.last\_question\_id,image\_ans))

conn.commit()

cursor.close()

conn.close()

os.remove(f"static/user\_ans/{filename}")

def allowed\_file(filename):

return '.' in filename and \

filename.rsplit('.', 1)[1].lower() in ALLOWED\_EXTENSIONS

def get\_data(self,num):

"""Calculates the mean and standard deviation of all the logs entered by the user """

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

user\_id = user.id

mean = []

stdev = []

topics = []

if num == 1:

cursor.execute("""

SELECT AVG(lipid), AVG(carbohydrates), AVG(protein\_and\_enzymes), AVG(dna), AVG(atp), AVG(Water\_and\_inorganic\_ions)

FROM topic1

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT lipid, carbohydrates, protein\_and\_enzymes, dna, atp, Water\_and\_inorganic\_ions

FROM topic1

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

list\_2d[4].append(row[4])

list\_2d[5].append(row[5])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,None)

elif num == 2:

cursor.execute("""

SELECT AVG(cell\_structure), AVG(transport\_across\_membrane), AVG(cell\_cycle), AVG(immune\_system)

FROM topic2

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT cell\_structure, transport\_across\_membrane, cell\_cycle, immune\_system

FROM topic2

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 3:

cursor.execute("""

SELECT AVG(gas\_exchange), AVG(digestion\_and\_absorption), AVG(mass\_transport)

FROM topic3

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT gas\_exchange, digestion\_and\_absorption, mass\_transport

FROM topic3

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 4:

cursor.execute("""

SELECT AVG(protein\_synthesis), AVG(biodiversity), AVG(genetic\_diversity)

FROM topic4

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT protein\_synthesis, biodiversity, genetic\_diversity

FROM topic4

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 5:

cursor.execute("""

SELECT AVG(photosynthesis), AVG(respiration), AVG(nitrogen\_cycle), AVG(energy\_and\_ecosystem)

FROM topic5

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT photosynthesis, respiration, nitrogen\_cycle, energy\_and\_ecosystem

FROM topic5

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 6:

cursor.execute("""

SELECT AVG(response\_to\_stimuli), AVG(nervous\_coordination\_and\_muscles), AVG(homeostasis)

FROM topic6

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT response\_to\_stimuli, nervous\_coordination\_and\_muscles, homeostasis

FROM topic6

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 7:

cursor.execute("""

SELECT AVG(inherited\_change), AVG(population\_and\_evolution), AVG(population\_in\_ecosystems)

FROM topic7

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT inherited\_change, population\_and\_evolution, population\_in\_ecosystems

FROM topic7

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,topics)

elif num == 8:

cursor.execute("""

SELECT AVG(gene\_expression), AVG(recombinant\_DNA\_technology)

FROM topic8

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT gene\_expression, recombinant\_DNA\_technology

FROM topic8

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

def get\_data\_maths(self,num):

"""Calculates the mean and standard deviation of all the logs entered by the user """

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

user\_id = user.id

mean = []

stdev = []

topics = []

if num == 1:

cursor.execute("""

SELECT AVG(proof), AVG(algebra\_and\_functions), AVG(coordinate\_geometry), AVG(sequences\_and\_series), AVG(trigonometry), AVG(exponentials\_and\_logarithms) AVG(differentiation), AVG(integration), AVG(numerical\_methods)

FROM pure

WHERE account\_id =?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT proof, algebra\_and\_functions, coordinate\_geometry, sequences\_and\_series, trigonometry, exponentials\_and\_logarithms,differentiation,integration,numerical\_methods

FROM pure

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[],[],[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

list\_2d[4].append(row[4])

list\_2d[5].append(row[5])

list\_2d[6].append(row[6])

list\_2d[7].append(row[7])

list\_2d[8].append(row[8])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,None)

elif num == 2:

cursor.execute (f"""

SELECT AVG(vectors), AVG(quantities\_and\_units\_in\_mechanics),AVG(kinematics), AVG(forces\_and\_Newton’s\_laws), AVG(moment)

FROM mech

WHERE account\_id = ?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT vectors, quantities\_and\_units\_in\_mechanics, kinematics, forces\_and\_Newton’s\_laws, moments

FROM mech

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

list\_2d[4].append(row[4])

try:

for i in range(len(list\_2d)):

mean.append(statistics.mean(list\_2d[i]))

topics = [i[0] for i in cursor.description]

return (average,stdev,topics)

except:

return (0,0,None)

elif num == 3:

cursor.execute (f"""

SELECT AVG(statistical\_sampling), AVG(data\_presentation\_and\_interpretation),AVG(probability), AVG(statistical\_distributions), AVG(statistical\_hypothesis\_testing)

FROM stats

WHERE account\_id = ?

""",[user\_id])

average = cursor.fetchall()

cursor.execute (f"""

SELECT statistical\_sampling, data\_presentation\_and\_interpretation, probability, statistical\_distributions, statistical\_hypothesis\_testing

FROM stats

WHERE account\_id = ?

""",[user\_id])

topic\_1 = cursor.fetchall()

list\_2d = [[],[],[],[],[]]

for row in topic\_1:

list\_2d[0].append(row[0])

list\_2d[1].append(row[1])

list\_2d[2].append(row[2])

list\_2d[3].append(row[3])

list\_2d[4].append(row[4])

try:

for i in range(len(list\_2d)):

stdev.append(round(statistics.stdev(list\_2d[i]),2))

topics = [i[0] for i in cursor.description]

return (mean,stdev,topics)

except:

return (0,0,None)

def get\_topic\_low(self):

lowest = []

for i in range(1,9):

topic1 = self.get\_data(i)

print(topic1)

lowest\_topic = topic1[2][0]

for i in (0,len(topic1[0])):

try:

if not topic1[0][i]-topic1[1][i] <= topic1[0][i+1]+topic1[1][i+1] and topic1[0][i]+topic1[1][i] >= topic1[0][i+1]-topic1[1][i+1]:

lowest\_topic = topic1[2][i]

except:

print(lowest\_topic)

lowest.append(lowest\_topic)

def get\_time(self,time):

"""Takes the time taken by the user to finish there questions and stores it in a database table with the corresponding question\_done\_id"""

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

user\_id = user.id

for i in range(5):

cursor.execute("""

INSERT INTO time\_taken (question\_done\_id, time) VALUES (?,?)""",

(self.get\_current\_q\_ids()[i], time))

conn.commit()

conn.commit()

cursor.close()

conn.close()

def draw\_graph(self, x, y,num ):

"""Takes data input and displays it as a graph"""

fig = px.bar(x=x, y=y, title=f'Mean logs for topic {num}')

# Display the graph on a website

fig.update\_layout(xaxis\_title='Topics',

yaxis\_title='Mean logs')

plot\_div = fig.to\_html(full\_html = False)

return plot\_div

def log\_teaching(self):

"""Takes what the teacher entered for there teaching logs and stores into a database table """

if request.method == "POST":

student\_email = request.form.get("student\_email")

student\_user = request.form.get("student\_name")

taught = request.form.get("taught")

if not taught:

flash("You have not entered what you taught",category="error")

else:

post\_ = post(student\_email = student\_email,student\_name= student\_user,taught = taught, teacher = current\_user.id )

db.session.add(post\_)

db.session.commit()

flash('Teaching log added!' , category= 'success')

def send\_answer\_to\_mark(self):

"""finds an account answers to mark when a user wants to mark questions"""

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

cursor.execute("""

SELECT id, acc\_mark.question\_done\_id, user.email

FROM account\_answer, acc\_mark , user

WHERE account\_answer.question\_done\_id = acc\_mark.question\_done\_id

ORDER BY RANDOM()

""")

questions\_to\_mark = cursor.fetchone()

if questions\_to\_mark == None:

return (False)

else:

conn.commit()

m = cursor.execute("""

SELECT acc\_mark.question\_done\_id, user\_answers

FROM account\_answer, acc\_mark, user

WHERE user.id = ? and acc\_mark.question\_done\_id = ?""",

[questions\_to\_mark[0],questions\_to\_mark[1]])

ans\_images = m.fetchall()

for x in range(len(ans\_images)):

rec\_data = ans\_images[x][1]

with open(f'static/marking/mark{user.id}{x}.png','wb') as q:

q.write(rec\_data)

return (True,questions\_to\_mark[2],questions\_to\_mark[0])

conn.commit()

cursor.close()

conn.close()

def get\_mess(self):

"""when the user goes to there chatting page, this goes through the database to find every message sent to that user and who sent it """

user = current\_user

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

m = cursor.execute (f"""

SELECT message, email

FROM messages , user

WHERE recipient\_id = ? and user.id = sender\_id

""",[user.id])

m = m.fetchall()

return m

### Views.py

#### HTML pages

from flask import Blueprint, render\_template, request, flash, jsonify, redirect, url\_for

from flask\_login import login\_required, current\_user

from flask\_socketio import SocketIO , send , emit

import os

import json

import numpy

import random

import matplotlib

from more\_def import \*

from \_\_init\_\_ import \*

from models import User

from account import account

from dotenv import load\_dotenv

import matplotlib.pyplot as plt

from werkzeug.utils import secure\_filename

from matplotlib.animation import FuncAnimation

from twilio.jwt.access\_token import AccessToken

from twilio.jwt.access\_token.grants import VideoGrant

from flask import Flask, render\_template, request, abort

views = Blueprint('views', \_\_name\_\_)

acc = account()

@views.route('/', methods=['GET', 'POST'])

def main\_page():

if request.method == "POST":

button1 = request.getParameter("button1")

return render\_template("main\_page.html", user=current\_user)

#Home pages for the two account types

@views.route('/TEACHERhome', methods=['GET', 'POST'])

@login\_required

def teacher\_home():

with open("quotes.txt") as quotes:

all\_quote = quotes.readlines()

quote = random.choice(all\_quote)

return render\_template("teacher\_home.html", user = current\_user, name = current\_user.first\_name, quote = quote)

@views.route('/STUDENThome', methods=['GET', 'POST'])

@login\_required

def student\_home():

with open("quotes.txt") as quotes:

all\_quote = quotes.readlines()

quote = random.choice(all\_quote)

return render\_template("home.html", user=current\_user, name = current\_user.first\_name, quote = quote)

#Logging method for bio, maths and teacher

@views.route('/logs', methods=['GET', 'POST'])

@login\_required

def logs():

acc.store\_logs\_database()

return render\_template('logging.html', user=current\_user)

@views.route('/logs\_maths', methods=['GET', 'POST'])

@login\_required

def logs\_maths():

acc.get\_forms\_maths()

return render\_template('logging\_maths.html', user=current\_user)

@views.route('/taught', methods=['GET', 'POST'])

@login\_required

def taught():

acc.log\_teaching()

posts = post.query.all()

return render\_template('teacher\_logs.html', user=current\_user, posts = posts)

#Getting help

@views.route('/help\_bio', methods=['GET', 'POST'])

@login\_required

def help\_bio():

acc.test()

teacher = acc.get\_teacher()

t\_name = teacher[2]

t\_email = teacher[1]

return render\_template('help\_bio.html', user=current\_user, name = current\_user.first\_name, teacher\_email = t\_email,teacher\_name = t\_name)

@views.route('/help\_maths', methods=['GET', 'POST'])

@login\_required

def help\_maths():

teacher = acc.get\_teacher()

t\_name = teacher[2]

t\_email = teacher[1]

return render\_template('help\_maths.html', user=current\_user, name = current\_user.first\_name,teacher\_email = t\_email,teacher\_name = t\_name)

#Doing questions

@views.route('/quiz\_maths', methods=['GET', 'POST'])

@login\_required

def quiz\_maths():

acc.get\_topic()

return render\_template('quiz\_maths.html', user=current\_user, name = current\_user.first\_name)

@views.route('/quiz\_bio', methods=['GET', 'POST'])

@login\_required

def quiz\_biology():

acc.get\_topic()

return render\_template('quiz\_bio.html', user=current\_user, name = current\_user.first\_name)

@views.route('/questions', methods=['GET', 'POST'])

@login\_required

def questions():

return render\_template('questions.html', user=current\_user)

@views.route('/answer', methods=['GET', 'POST'])

@login\_required

def answer():

return render\_template('answer.html', user=current\_user)

@views.route('/uploadimage', methods=['GET', 'POST'])

@login\_required

def upload\_file():

if request.method == 'POST':

# check if the post request has the file part

if 'file' not in request.files:

flash('No file part')

return redirect('upload.html', user = current\_user)

file = request.files['file']

# if user does not select file, browser also

# submit an empty part without filename

if file.filename == '':

flash('No selected file')

return redirect('upload.html', user = current\_user)

if file and allowed\_file(file.filename):

filename = secure\_filename(file.filename)

print(filename)

print(file)

file.save(os.path.join('static/user\_ans', filename))

user\_image\_db(filename)

return render\_template('upload.html', user = current\_user

# Communication between accounts

@views.route('/send', methods=['POST'])

def send():

# Get the message data from the form

data = json.loads(request.data)

print(data)

recipient = data["recipient"]

message = data["message"]

print("hi")

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

user = current\_user

sender\_id = user.id

# Save the message to the database

cursor.execute("""INSERT INTO messages (sender\_id, recipient\_id, message) VALUES (?, ?, ?)""",

(sender\_id, recipient, message))

conn.commit()

cursor.close()

conn.close()

return 'Message sent!'

@views.route('/chat', methods =['GET','POST'] )

@login\_required

def chat():

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

cursor.execute('''SELECT id, email FROM user''')

users = cursor.fetchall()

user = current\_user

# Send any undelivered messages to the user

messages = acc.get\_mess()

print(messages)

#Delete the messages from the database

#cursor.execute('''DELETE FROM messages WHERE recipient\_id=?''', (user['id'],))

return render\_template('chat.html', user = current\_user, users = users, messages = messages )

@views.route('/users')

def get\_users():

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

cursor.execute('''SELECT id, email FROM user''')

users = cursor.fetchall()

conn.commit()

cursor.close()

conn.close()

return users

#Mark questions done by other people

@views.route('/marks', methods =['GET','POST'] )

@login\_required

def mark():

mark = acc.send\_answer\_to\_mark()

yesorno = mark[0]

if not mark:

return render\_template('no\_mark.html', user = current\_user)

elif mark:

uservalue = mark[1]

userid = mark[2]

return render\_template('mark.html', user = current\_user,mark\_user = uservalue,userid =userid)

# Logs shown as processed data in a graph

@views.route('/stats', methods =['GET','POST'] )

@login\_required

def update():

data1 = acc.get\_data(1)

plot\_div1 = acc.draw\_graph(data1[0],data1[2],1)

data2 = acc.get\_data(2)

plot\_div2 = acc.draw\_graph(data2[0],data2[2],2)

data3 = acc.get\_data(3)

plot\_div3 = acc.draw\_graph(data3[0],data3[2],3)

data4 = acc.get\_data(4)

plot\_div4 = acc.draw\_graph(data4[0],data4[2],4)

data5 = acc.get\_data(5)

plot\_div5 = acc.draw\_graph(data5[0],data5[2],5)

data6 = acc.get\_data(6)

plot\_div6 = acc.draw\_graph(data6[0],data6[2],6)

data7 = acc.get\_data(7)

plot\_div7 = acc.draw\_graph(data7[0],data7[2],7)

data8 = acc.get\_data(8)

plot\_div8 = acc.draw\_graph(data8[0],data8[2],8)

return render\_template('stats.html', user = current\_user, plot\_div1 = plot\_div1, plot\_div2=plot\_div2,

plot\_div3 = plot\_div3, plot\_div4=plot\_div4,

plot\_div5 = plot\_div5, plot\_div6=plot\_div6,

plot\_div7 = plot\_div7, plot\_div8=plot\_div8)

@views.route('/store', methods=['POST'])

def store():

data = json.loads(request.data)

print(data)

time\_spent = data["time\_spent"]

print(time\_spent)

acc.get\_time(time\_spent)

return jsonify({"message": "time\_spent stored in database"})

@views.route('/choice', methods=['POST'])

def choice():

return render\_template("choice.html" , user = current\_user )

#### Sending an email to all students

def send\_email():

"""Queries emails and sends an email to all the student

acc with a topic as a reminder to revise"""

conn = sqlite3.connect('database.db')

cursor = conn.cursor()

status = "student"

my\_email = "testingmycode1@outlook.com"

password = "SomethingSomething12"

now = datetime.now()

date\_week = now-timedelta(7)

with open("email.txt") as email\_points:

email\_text = email\_points.readlines()

cursor.execute("""

SELECT id, email, first\_name

FROM user

WHERE status = ?""",[status])

email\_data = cursor.fetchall()

cursor.commit()

for person in email\_date:

email\_text = random.choice(email\_text)

cursor.execute("""

SELECT email,first\_name, question\_done\_id, acc\_id, marks, topic, first\_name

FROM account\_marks, user

WHERE account\_marks.acc\_id = ? and account\_marks.date\_entered = ?""",[person[0]][date\_week])

questions\_done\_week\_ago = cursor.fetchone()

cursor.commit()

if questions\_done\_week\_ago == None:

start\_text = "Your reminder to revise"

topic = None

more\_text = " "

else:

start\_text = "You did questions a week ago on "

topic = questions\_done\_week\_ago[5]

more\_text = f"and you got {person[4]} marks on it. Want to try some more?"

self.send\_an\_email(my\_email,password,person[1],person[2],email\_text,start\_text, more\_text)

#Login email detail and sending email detail

conn.commit()

cursor.close()

conn.close()

def send\_an\_email(my\_email,password,send\_email, name,text, type, topic, more\_text):

"""Sends an email"""

with smtplib.SMTP('smtp-mail.outlook.com', port='587') as smtp:

smtp.ehlo() # send the extended hello to our server

smtp.starttls() # tell server we want to communicate with TLS encryption

smtp.login(user = my\_email, password = password) # login to our email server

smtp.sendmail(from\_addr = my\_email,

to\_addrs = send\_email,

#The email message.

msg = f"""Subject:Email from maths/Biology revision \n\n {name} Hi {type} {topic} {more\_text} \n {text}""")

print("email success")

## HTML Code

### Base template for HTML pages

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"

integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"

crossorigin="anonymous" />

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css"

crossorigin="anonymous" />

<link rel="stylesheet" href="static/styles.css " />

<link href="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/css/bootstrap.min.css" rel="stylesheet" id="bootstrap-css">

<script src="//maxcdn.bootstrapcdn.com/bootstrap/4.1.1/js/bootstrap.min.js"></script>

<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.0/src/style.css" />

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/Verly.js@v1.1.3/dist/verly.bundle.js"></script>

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.0/src/VerlyRange.js"></script>

<script src=" static/index.js"></script>

<link href="https://fonts.googleapis.com/css?family=Francois+One&display=swap" rel="stylesheet">

<title>{% block title %}Home{% endblock %}</title>

</head>

<body>

<nav class="navbar navbar-expand-lg navbar-light bg-light">

<a class="navbar-brand" href="" >Revision App</a>

<button class="navbar-toggler"

type="button"

data-toggle="collapse"

data-target="#navbar">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbar" >

<div class="navbar-nav ml-auto">

{% if user.is\_authenticated %}

{% if user.status == "teacher" %}

<a class="nav-item nav-link" id="home" href="/TEACHERhome">Home</a>

<a class="nav-item nav-link" id="logout" href="/logout">Logout</a>

{% else %}

<a class="nav-item nav-link" id="home" href="/STUDENThome">Home</a>

<a class="nav-item nav-link" id="logout" href="/logout">Logout</a>

{% endif %}

{% else %}

<a class="nav-item nav-link" id="login" href="/login">Login</a>

<a class="nav-item nav-link" id="signUp" href="/sign-up">Sign Up</a>

{% endif %}

</div>

</div>

</nav>

{% with messages = get\_flashed\_messages(with\_categories=true) %} {% if

messages %} {% for category, message in messages %} {% if category ==

'error' %}

<div class="alert alert-danger alter-dismissable fade show" role="alert">

**{{** message **}}**

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

</div>

{% else %}

<div class="alert alert-success alter-dismissable fade show" role="alert">

**{{** message **}}**

<button type="button" class="close" data-dismiss="alert">

<span aria-hidden="true">&times;</span>

</button>

</div>

{% endif %} {% endfor %} {% endif %} {% endwith %}

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

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"

integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"

crossorigin="anonymous"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js"

integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"

crossorigin="anonymous"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"

integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"

crossorigin="anonymous"></script>

<script type="text/javascript"

src="**{{** url\_for('static', filename='index.js') **}}**"></script>

</body>

</html>

### Sign-up page

{% extends "base.html" %} {% block title %}Sign Up{% endblock %} {% block

content %}

<form method="POST">

<h3 align="center">Sign Up</h3>

<div class="form-group">

<label for="email">Email Address</label>

<input type="email"

class="form-control"

id="email"

name="email"

placeholder="Enter email" />

</div>

<div class="form-group">

<label for="firstName">First Name</label>

<input type="text"

class="form-control"

id="firstName"

name="firstName"

placeholder="Enter first name" />

</div>

<div class="form-group">

<label for="lastname">Last Name</label>

<input type="text"

class="form-control"

id="lastname"

name="lastname"

placeholder="Enter last name" />

</div>

<div class="form-group">

<label for="password1">Password</label>

<input type="password"

class="form-control"

id="password1"

name="password1"

placeholder="Enter password" />

</div>

<div class="form-group">

<label for="password2">Password (Confirm)</label>

<input type="password"

class="form-control"

id="password2"

name="password2"

placeholder="Confirm password" />

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" name="status" type="checkbox" id="student" value="student">

<label class="form-check-label" for="student">student</label>

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" name="status" type="checkbox" id="teacher" value="teacher">

<label class="form-check-label" for="teacher">teacher</label>

</div>

<div class="form-check form-check-inline">

<input class="form-check-input" name="status" type="checkbox" id="admin" value="admin" disabled>

<label class="form-check-label" for="admin">Admin (disabled)</label>

</div>

<br />

<button type="submit" class="btn btn-dark">Submit</button>

</form>

{% endblock %}

### Student Home page (HTML)

{% extends "base.html" %} {% block title %}Student Home{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<div>

<h1 class="hometitle">Welcome, <span class="name">**{{**name**}}**</span></h1>

<p>**{{**quote**}}**</p>

</div>

<hr />

<div>

<h5>Keep track of how your doing in each topic </h5>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Logs - Biology</h5>

<p class="card-text">Log your confidance in each topic for Biology Alevel</p>

<a href="/logs" class="btn btn-light colourofbutton">Logs :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Logs - Maths</h5>

<p class="card-text">Log your confidance in each topic for Maths Alevel</p>

<a href="/logs\_maths" class="btn btn-light colourofbutton">Logs :)</a>

</div>

</div>

</div>

</div>

<hr />

</div>

<div>

<h5>Do questions to test your knowledge</h5>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Questions - Biology</h5>

<p class="card-text">Pick what topic questions you want and then viola</p>

<a href="/quiz\_bio" class="btn btn-light colourofbutton">questions :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Questions - Maths</h5>

<p class="card-text">Pick what topic questions you want and then viola</p>

<a href="/quiz\_maths" class="btn btn-light colourofbutton">questions :)</a>

</div>

</div>

</div>

</div>

</div>

<hr />

<div>

<h5>Mark other users questions</h5>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">mark</h5>

<p class="card-text">Mark other people biology questions</p>

<a href="/marks" class="btn btn-light colourofbutton">mark :)</a>

</div>

</div>

</div>

</div>

</div>

<hr />

<h5>Check your stats </h5>

<div>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">stats - bio </h5>

<p class="card-text">Look at the logs you have entered so far</p>

<a href="/stats" class="btn btn-light colourofbutton">stats :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Stats - Maths</h5>

<p class="card-text">Check how your logs are going </p>

<a href="/statsmaths" class="btn btn-light colourofbutton">stats :)</a>

</div>

</div>

</div>

</div>

</div>

<h5>Talk to other users</h5>

<div>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Chat</h5>

<p class="card-text">Check your messages and send messages</p>

<a href="/chat" class="btn btn-light colourofbutton">chat :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Help</h5>

<p class="card-text">Get help</p>

<a href="/help\_bio" class="btn btn-light colourofbutton">get help :)</a>

</div>

</div>

</div>

</div>

</div>

<hr />

{% endblock %}

### Teacher Home page

{% extends "base.html" %} {% block title %}Teacher Home{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<div>

<h1 class="hometitle">Welcome teacher, <span class="name">**{{**name**}}**</span></h1>

<p>**{{**quote**}}**</p>

</div>

<hr />

<div>

<h5>Log what you have taught</h5>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Teaching Logs</h5>

<p class="card-text">Log what you have taught</p>

<a href="/taught" class="btn btn-light colourofbutton">Logs :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Teach</h5>

<p class="card-text">Teach</p>

<a href="/chat" class="btn btn-light colourofbutton">Logs :)</a>

</div>

</div>

</div>

</div>

<hr />

<div>

<h5>Try doing some questions, test your knowledge</h5>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Quiz - Biology</h5>

<p class="card-text">Do questions :D</p>

<a href="/quiz\_bio" class="btn btn-light colourofbutton">Help :)</a>

</div>

</div>

</div>

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">Quiz - Maths</h5>

<p class="card-text">Do questions :D</p>

<a href="/quiz\_maths" class="btn btn-light colourofbutton">Help :)</a>

</div>

</div>

</div>

</div>

</div>

<div>

<h5>Mark other users questions</h5>

<div class="row">

<div class="col-sm-6">

<div class="card">

<div class="card-body">

<h5 class="card-title">mark</h5>

<p class="card-text">Mark other peoples questions</p>

<a href="/marks" class="btn btn-light colourofbutton">mark :)</a>

</div>

</div>

</div>

</div>

</div>

<hr />

{% endblock %}

### Biology logs

{% extends "base.html" %} {% block title %}Logs{% endblock %} {% block content

%}

<h1 class="logs">Logs Biology</h1>

<p>Enter your level of confidance with each of these topics to be recomended topics to learn and keep to track of what you have learnt and what you need to learn.</p>

<head>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.2/src/style.css" />

<link href="https://fonts.googleapis.com/css?family=Francois+One&display=swap" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/Verly.js@v1.1.4/dist/verly.bundle.js"></script>

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.2/src/VerlyRange.js"></script>

<link rel="stylesheet" href="static/styles.css " />

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0 20px;

}

body, html {

font-family: 'Francois One', sans-serif;

margin: 0;

padding: 0;

color: #f55693

}

.wrapper {

margin: 50px auto;

max-width: 980px;

}

section.all-sliders {

width: 100%;

float: left;

}

</style>

<body>

<main class="wrapper">

<form class="all-sliders" method="post">

<label class="slidecontainer third">

<p class="topictitle">Topic 1</p>

<span>Lipid</span>

<output class="numoutput" id="range">100</output>

<input class="slider" name="lipids" id="lipids" type="range" min="1" step="1" max="100" value="100" onchange=" range.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third">

<span>Protein</span>

<output id="rangeprotein">100</output>

<input class="slider" name="protein" id="protein" type="range" min="1" step="1" max="100" value="100" onchange=" rangeprotein.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third">

<span>water</span>

<output id="rangewater">100</output>

<input class="slider" name="water" id="water" type="range" min="1" step="1" max="100" value="100" onchange=" rangewater.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>atp</span>

<output id="rangeatp">100</output>

<input class="slider" name="atp" id="atp" type="range" min="1" step="1" max="100" value="100" onchange=" rangeatp.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Carbohydrates</span>

<output id="rangecarbohydrates">100</output>

<input class="slider" name="Carbohydrates" id="Carbohydrates" type="range" min="1" step="1" max="100" value="100" onchange=" rangecarbohydrates.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>dna</span>

<output id="rangedna">100</output>

<input class="slider" name="dna" id="dna" type="range" min="1" step="1" max="100" value="100" onchange=" rangedna.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 2</p>

<span>Cell structure</span>

<output id="rangecs">100</output>

<input class="slider" name="cellstructure" id="cellstructure" type="range" min="1" step="1" max="100" value="100" onchange=" rangecs.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Transport Across Cell Membrane</span>

<output id="rangetransportacrossmembrane">100</output>

<input class="slider" name="transportacrossmembrane" id="transportacrossmembrane" type="range" min="1" step="1" max="100" value="100" onchange=" rangetransportacrossmembrane.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Cell Cycle</span>

<output id="rangecellcycle">100</output>

<input class="slider" name="cellcycle" id="cellcycle" type="range" min="1" step="1" max="100" value="100" onchange=" rangecellcycle.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Immune System</span>

<output id="rangeimmunesystem">100</output>

<input class="slider" name="immunesystem" id="immunesystem" type="range" min="1" step="1" max="100" value="100" onchange=" rangeimmunesystem.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 3</p>

<span>Gas exchange</span>

<output id="rangege">100</output>

<input class="slider" name="gasexchange" id="gasexchange" type="range" min="1" step="1" max="100" value="100" onchange=" rangege.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Digestion and Absorbtion</span>

<output id="rangedigestionandabsorption">100</output>

<input class="slider" name="digestionandabsorption" id="digestionandabsorption" type="range" min="1" step="1" max="100" value="100" onchange=" rangedigestionandabsorption.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>mass transport</span>

<output id="rangemasstransport">100</output>

<input class="slider" name="masstransport" id="masstransport" type="range" min="1" step="1" max="100" value="100" onchange=" rangemasstransport.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 4</p>

<span>protein synthesis</span>

<output id="rangeps">100</output>

<input class="slider" name="proteinsynthesis" id="proteinsynthesis" type="range" min="1" step="1" max="100" value="100" onchange=" rangeps.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>biodiversity</span>

<output id="rangebiodiversity">100</output>

<input class="slider" name="biodiversity" id="biodiversity" type="range" min="1" step="1" max="100" value="100" onchange=" rangebiodiversity.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>genetic diversity</span>

<output id="rangegeneticdiversity">100</output>

<input class="slider" name="geneticdiversity" id="geneticdiversity" type="range" min="1" step="1" max="100" value="100" onchange=" rangegeneticdiversity.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 5</p>

<span>photosynthesis</span>

<output id="rangephotosynthesis">100</output>

<input class="slider" name="photosynthesis" id="photosynthesis" type="range" min="1" step="1" max="100" value="100" onchange=" rangephotosynthesis.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>respiration</span>

<output id="rangerespiration">100</output>

<input class="slider" name="respiration" id="respiration" type="range" min="1" step="1" max="100" value="100" onchange=" rangerespiration.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>nitrogen cycle</span>

<output id="rangenitrogencycle">100</output>

<input class="slider" name="nitrogencycle" id="nitrogencycle" type="range" min="1" step="1" max="100" value="100" onchange=" rangenitrogencycle.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>energy and ecosystem</span>

<output id="rangeenergyandecosystem">100</output>

<input class="slider" name="energyandecosystem" id="energyandecosystem" type="range" min="1" step="1" max="100" value="100" onchange=" rangeenergyandecosystem.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 6</p>

<span>response to stimuli</span>

<output id="rangeresponsetostimuli">100</output>

<input class="slider" name="responsetostimuli" id="responsetostimuli" type="range" min="1" step="1" max="100" value="100" onchange=" rangeresponsetostimuli.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>nervous coordination and muscles</span>

<output id="rangenervouscoordinationandmuscles">100</output>

<input class="slider" name="nervouscoordinationandmuscles" id="nervouscoordinationandmuscles" type="range" min="1" step="1" max="100" value="100" onchange=" rangenervouscoordinationandmuscles.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>homeostasis</span>

<output id="rangehomeostasis">100</output>

<input class="slider" name="homeostasis" id="homeostasis" type="range" min="1" step="1" max="100" value="100" onchange=" rangehomeostasis.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 7</p>

<span>inherited change</span>

<output id="rangeinheritedchange">100</output>

<input class="slider" name="inheritedchange" id="inheritedchange" type="range" min="1" step="1" max="100" value="100" onchange=" rangeinheritedchange.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>populationandevolution</span>

<output id="rangepopulationandevolution">100</output>

<input class="slider" name="populationandevolution" id="populationandevolution" type="range" min="1" step="1" max="100" value="100" onchange=" rangepopulationandevolution.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>population\_in\_ecosystems</span>

<output id="rangepopulationinecosystems">100</output>

<input class="slider" name="populationinecosystems" id="populationinecosystems" type="range" min="1" step="1" max="100" value="100" onchange=" rangepopulationinecosystems.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Topic 8</p>

<span>gene expression</span>

<output id="rangegeneexpression">100</output>

<input class="slider" name="geneexpression" id="geneexpression" type="range" min="1" step="1" max="100" value="100" onchange=" rangegeneexpression.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>recombinant DNA technology</span>

<output id="rangerecombinantDNAtechnology">100</output>

<input class="slider" name="recombinantDNAtechnology" id="recombinantDNAtechnology" type="range" min="1" step="1" max="100" value="100" onchange=" rangerecombinantDNAtechnology.value = value">

<div id="demo"></div>

</label>

<button type="submit" class="btn btn-outline-success">Success</button>

</form>

</main>

</body>

{% endblock %}

### Maths logs

{% extends "base.html" %} {% block title %}Logs{% endblock %} {% block content

%}

<h1 class="logs">Logs Biology</h1>

<p>Enter your level of confidance with each of these topics to be recomended topics to learn and keep to track of what you have learnt and what you need to learn.</p>

<head>

<link rel="stylesheet" href="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.2/src/style.css" />

<link href="https://fonts.googleapis.com/css?family=Francois+One&display=swap" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/Verly.js@v1.1.4/dist/verly.bundle.js"></script>

<script src="https://cdn.jsdelivr.net/gh/anuraghazra/VerlyRangeSlider@v1.0.2/src/VerlyRange.js"></script>

<link rel="stylesheet" href="static/styles.css " />

<style>

\* {

box-sizing: border-box;

}

body {

margin: 0 20px;

}

body, html {

font-family: 'Francois One', sans-serif;

margin: 0;

padding: 0;

color: #f55693

}

.wrapper {

margin: 50px auto;

max-width: 980px;

}

section.all-sliders {

width: 100%;

float: left;

}

</style>

</head>

<body>

<main class="wrapper">

<form class="all-sliders" method="post">

<label class="slidecontainer third">

<p class="topictitle">Pure</p>

<span>proof</span>

<output class="numoutput" id="rangeproof">100</output>

<input class="slider" name="proof" id="proof" type="range" min="1" step="1" max="100" value="100" onchange=" rangeproof.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third">

<span>Algebra and functions</span>

<output id="rangealgebraandfunctions">100</output>

<input class="slider" name=" algebraandfunctions" id=" algebraandfunctions" type="range" min="1" step="1" max="100" value="100" onchange=" rangealgebraandfunctions.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third">

<span>Coordinate geometry</span>

<output id="rangecoordinategeometry">100</output>

<input class="slider" name="coordinategeometry" id="coordinategeometry" type="range" min="1" step="1" max="100" value="100" onchange=" rangecoordinategeometry.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>squences and series</span>

<output id="rangesquencesandseries">100</output>

<input class="slider" name="squencesandseries" id="squencesandseries" type="range" min="1" step="1" max="100" value="100" onchange=" rangesquencesandseries.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Trigonometry</span>

<output id="rangetrigonometry">100</output>

<input class="slider" name="trigonometry" id="trigonometry" type="range" min="1" step="1" max="100" value="100" onchange=" rangetrigonometry.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Exponentials and logarithms</span>

<output id="rangeexponentialsandlogarithms">100</output>

<input class="slider" name="exponentialsandlogarithms" id="exponentialsandlogarithms" type="range" min="1" step="1" max="100" value="100" onchange=" rangeexponentialsandlogarithms.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Differentiation</span>

<output id="rangedifferentiation">100</output>

<input class="slider" name="differentiation" id="differentiation" type="range" min="1" step="1" max="100" value="100" onchange=" rangedifferentiation.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>integration</span>

<output id="rangeintegration">100</output>

<input class="slider" name="integration" id="integration" type="range" min="1" step="1" max="100" value="100" onchange=" rangeintegration.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Numerical methods</span>

<output id="rangenumericalmethods">100</output>

<input class="slider" name="numericalmethods" id="numericalmethods" type="range" min="1" step="1" max="100" value="100" onchange=" rangenumericalmethods.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Mechanics </p>

<span>Vectors</span>

<output id="rangevectors">100</output>

<input class="slider" name="vectors" id="vectors" type="range" min="1" step="1" max="100" value="100" onchange=" rangevectors.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Quantities and units in mechanics</span>

<output id="rangequantitiesandunitsinmechanics">100</output>

<input class="slider" name="quantitiesandunitsinmechanics" id="quantitiesandunitsinmechanics" type="range" min="1" step="1" max="100" value="100" onchange=" rangequantitiesandunitsinmechanics.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Kinematics</span>

<output id="rangekinematics">100</output>

<input class="slider" name="kinematics" id="kinematics" type="range" min="1" step="1" max="100" value="100" onchange=" rangekinematics.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Forces and Newton’s laws</span>

<output id="rangeforcesandnewtonslaws">100</output>

<input class="slider" name="forcesandnewtonslaws" id="forcesandnewtonslaws" type="range" min="1" step="1" max="100" value="100" onchange=" rangeforcesandnewtonslaws.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Moments</span>

<output id="rangemoments">100</output>

<input class="slider" name="moments" id="moments" type="range" min="1" step="1" max="100" value="100" onchange=" rangemoments.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<p class="topictitle">Statistics </p>

<span>Statistical sampling</span>

<output id="rangestatisticalsampling">100</output>

<input class="slider" name="statisticalsampling" id="statisticalsampling" type="range" min="1" step="1" max="100" value="100" onchange=" rangestatisticalsampling.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Data presentation and Interpretation</span>

<output id="rangedatapresentationandinterpretation">100</output>

<input class="slider" name="datapresentationandinterpretation" id="datapresentationandinterpretation" type="range" min="1" step="1" max="100" value="100" onchange=" rangedatapresentationandinterpretation.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Probability</span>

<output id="rangeprobability">100</output>

<input class="slider" name="probability" id="probability" type="range" min="1" step="1" max="100" value="100" onchange=" rangeprobability.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Statistical distributions</span>

<output id="rangestatisticaldistributions">100</output>

<input class="slider" name="statisticaldistributions" id="statisticaldistributions" type="range" min="1" step="1" max="100" value="100" onchange=" rangestatisticaldistributions.value = value">

<div id="demo"></div>

</label>

<label class="slidecontainer third ">

<span>Statistical hypothesis testing</span>

<output id="rangestatisticalhypothesistesting">100</output>

<input class="slider" name="statisticalhypothesistesting" id="statisticalhypothesistesting" type="range" min="1" step="1" max="100" value="100" onchange=" rangestatisticalhypothesistesting.value = value">

<div id="demo"></div>

</label>

<button type="submit" class="btn btn-outline-success">Success</button>

</form>

</main>

</body>

{% endblock %}

### Biology help

{% extends "base.html" %} {% block title %}bio choice{% endblock %} {% block content

%}

<head>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script type="text/javascript">

function populate(s1, s2) {

var s1 = document.getElementById(s1);

var s2 = document.getElementById(s2);

s2.innerHTML = "";

if (s1.value == "1") {

var optionArray = ['|', "lipid|lipid", "carbohydrates|Carbohydrates", "proteinandenzymes|proteins and enzymes", "dna|dna", "atp|atp", "Waterandinorganic\_ions|water"];

}

else if (s1.value == "2") {

var optionArray = ['|', "cellstructure|cell strucuture", "transportacrossmembrane|transport across membrane", "cellcycle|cell cycle", "immunesystem|immune system"];

}

else if (s1.value == "3") {

var optionArray = ['|', "gasexchange|gas exchange", "digestionandabsorption|digestion and absorption", "masstransport|mass transport"];

}

else if (s1.value == "4") {

var optionArray = ['|', "proteinsynthesis|protein synthesis", "biodiversity|biodiversity", "geneticdiversity|genetic diversity"];

}

else if (s1.value == "5") {

var optionArray = ['|', "photosynthesis|photosynthesis", "respiration|respiration", "nitrogencycle|nitrogen cycle", "energyandecosystem|energy and ecosystem"];

}

else if (s1.value == "6") {

var optionArray = ['|', "responsetostimuli|response to stimuli", "nervouscoordinationandmuscles|nervous coordination and muscles", "homeostasis|homeostasis"];

}

else if (s1.value == "7") {

var optionArray = ['|', "inheritedchange|inherited change", "populationandevolution|population and evolution", "populationinecosystems|population in ecosystems"];

}

else if (s1.value == "8") {

var optionArray = ['|', "geneexpression|gene expression", "recombinantDNAtechnology|recombinant DNA technology"];

}

for (var option in optionArray) {

var pair = optionArray[option].split("|");

var newOption = document.createElement("option");

newOption.value = pair[0];

newOption.innerHTML = pair[1];

s2.options.add(newOption);

}

}

document.getElementById("leave-button").onclick = function () {

location.href = "/questions";

};

document.getElementById("leave-button").addEventListener("click", function () {

// Send timeSpent to the server using an AJAX call

var xhr = new XMLHttpRequest();

xhr.open("POST", "/topicchoosen", true);

xhr.setRequestHeader('Content-Type', 'application/json');

xhr.send(JSON.stringify({ topic: slct2 }));

location.href = "/questions";

});

</script>

</head>

<link rel="stylesheet" href="static/styles.css" />

<body>

<div>

<h1 class="hometitle">Welcome, <span class="name">**{{**name**}}**</span></h1>

<p>Answering question time :D</p>

<p>First you have to pick what topic questions do you want or do you want them from the all specification :D</p>

</div>

<hr />

<form method="POST" >

choose topic:

<select id="slct1" name="slct1" onchange="populate(this.id,'slct2')">

<option value="">nothin</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

<option value="6">6</option>

<option value="7">7</option>

<option value="8">8</option>

</select>

<hr />

choose subtopic:

<select id="slct2" name="slct2">

<option value="">nothin</option>

</select>

<hr />

<button type="submit" class="button" id="leave-button">question time</button>

<a href="/questions">question time</a>

</form>

</body>

{% endblock %}

### Maths help

{% extends "base.html" %} {% block title %}Student Home{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<head>

<script type="text/javascript">

function populate(s1,s2) {

var s1 = document.getElementById(s1);

var s2 = document.getElementById(s2);

s2.innerHTML = "";

if (s1.value == "1") {

var optionArray = ['|', "proof|proof", "algebra\_and\_functions|algebra andfunctions", "coordinate\_geometry|coordinate geometry", "sequences\_and\_series|sequences and series", "trigonometry|trigonometry", "exponentials\_and\_logarithms|exponentials and logarithms", "trigonometry|trigonometry", "differentiation|differentiation", "integration|integration", "numerical\_methods|numerical methods"];

}

else if (s1.value == "2") {

var optionArray = ['|', "vectors|vectors", "quantities\_and\_units\_in\_mechanics|quantities\_and\_units\_in\_mechanics", "kinematics|kinematics", "forces\_and\_Newton’s\_laws|forces and Newton’s laws", "moments|moments"];

}

else if (s1.value == "3") {

var optionArray = ['|', "statistical\_sampling|statistical\_sampling", "data\_presentation\_and\_interpretation|data presentation and interpretation", "probability|probability", "statistical\_distributions|statistical distributions", "statistical\_hypothesis\_testing|statistical hypothesis testing"];

}

for (var option in optionArray) {

var pair = optionArray[option].split("|");

var newOption = document.createElement("option");

newOption.value = pair[0];

newOption.innerHTML = pair[1];

s2.options.add(newOption);

}

}

</script>

</head>

<link rel="stylesheet" href="static/styles.css" />

<body>

<div>

<h1 class="hometitle">Welcome, <span class="name">**{{**name**}}**</span></h1>

<p>Answering question time :D</p>

<p>First you have to pick what topic questions do you want or do you want them from the all specification :D</p>

</div>

<hr />

<form method="POST" action="questions">

choose topic:

<select id="slct1" name="slct1" onchange="populate(this.id,'slct2')">

<option value="">nothin</option>

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

</select>

<hr />

choose subtopic:

<select id="slct2" name="slct2">

<option value="">nothin</option>

</select>

<hr />

<button class="submit">question time</button>

</form>

</body>

{% endblock %}

### Displaying questions

{% extends "base.html" %} {% block title %}Student Home{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<div>

<p style=" margin-top: 16px; ">Answering question time :D. </p>

<p>These questions will be timed on how long you take, so make sure to not leave your device open with this page :D</p>

<br />

<p class="time" id="demo"></p>

</div>

<button type="submit" id="leave-button" onclick='run = false '>Finish</button>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

var startTime = new Date();

// Add event listener to button to get end time when clicked

document.getElementById("leave-button").addEventListener("click", function(){

var endTime = new Date();

var timeSpent = (endTime - startTime) / 1000; // time spent in seconds

// Send timeSpent to the server using an AJAX call

var xhr = new XMLHttpRequest();

xhr.open("POST", "/store", true);

xhr.setRequestHeader('Content-Type', 'application/json');

xhr.send(JSON.stringify({time\_spent: timeSpent}));

});

function imagechange() {

fetch('/testingthis')

.then(response => response.json())

.then(data => {

const id = data.id;

console.log(id);

})

var image\_1 = document.getElementById('question\_image')

this\_value = this.value

if (this\_value = 1) {

image\_1.src = 'static/questions/q${id}0.png';

}

else if (this\_value = 2) {

image\_1.src = 'static/questions/q${id}1.png';

}

else if (this\_value = 3) {

image\_1.src = 'static/questions/q${id}2.png';

}

else if (this\_value = 4) {

image\_1.src = 'static/questions/q${id}3.png';

}

else if (this\_value = 5) {

image\_1.src = 'static/questions/q${id}4.png';

}

}

document.getElementById("leave-button").onclick = function () {

location.href = "/answer";

};

var currentTime\_s = 0;

var currentTime\_m = 0;

var currentTime\_h = 0;

var totalTime = 0

var run = true;

var time = "time";

var x = setInterval(function () {

if (run == false) {

document.getElementById("test").innerHTML = "You took " + totalTime + " seconds";

document.getElementById("test").value = totaltime;

sendinfo()

return;

}

else {

currentTime\_s = currentTime\_s + 1;

totalTime = totalTime + 1;

if (currentTime\_s == 60) {

currentTime\_m = currentTime\_m + 1;

currentTime\_s = 0

if (currentTime\_m == 60) {

currentTime\_h = currentTime\_h + 1;

currentTime\_m = 0

}

}

// Display the result in the element with id="demo"

document.getElementById("demo").innerHTML = currentTime\_h + "h "

+ currentTime\_m + "m " + currentTime\_s + "s ";

};

}, 1000);

</script>

<br />

<p id="test"></p>

<br />

<p>**{{**user.id**}}**</p>

<!--Changing the image using a javascript function-->

<centre>

<div class="pagination\_section" style="margin-left:100px">

<a title="question 1" onclick="imagechange()" id="q**{{**user.id**}}**0" value="1">1</a>

<a title="question 2" id="q**{{**user.id**}}**1" onclick="imagechange()" value="2">2</a>

<a title="question 3" id="q**{{**user.id**}}**2" onclick="imagechange()" value="3">3</a>

<a title="question 4" id="q**{{**user.id**}}**3" onclick="imagechange()" value="4">4</a>

<a title="question 5" id="q**{{**user.id**}}**4" onclick="imagechange()" value="5">5</a>

</div>

</centre>

<!-- the images of the questions will go here -->

<div>

<img src="static/questions/q**{{**user.id**}}**0.png" alt="question 1" id="question\_image" style="margin-left:100px">

<img src="static/questions/q**{{**user.id**}}**1.png" alt="question 1" id="question\_image" style="margin-left:100px" >

<img src="static/questions/q**{{**user.id**}}**2.png" alt="question 1" id="question\_image" style="margin-left:100px">

<img src="static/questions/q**{{**user.id**}}**3.png" alt="question 1" id="question\_image" style="margin-left:100px">

<img src="static/questions/q**{{**user.id**}}**4.png" alt="question 1" id="question\_image" style="margin-left:100px">

</div>

{% endblock %}

### Giving the choice to mark themselves or to submit answers

{% extends "base.html" %} {% block title %}Student Home{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<div>

<h5>Now you can either mark your answers yourself or send them to someone else to mark the choice is up to you</h5>

<hr />

</div>

<div>

<a class="btn btn-primary" href="/answer" role="button">Mark your own </a>

<a class="btn btn-primary" href="upload.html" role="button">Have someone mark</a>

</div>

{% endblock %}

### Displaying the answers

{% extends "base.html" %} {% block title %}Answers{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<div>

<p style=" margin-top: 16px; ">Answers :D </p>

<p>Make sure to mark correctly and once done enter your mark :D</p>

<bh />

</div>

<div>

<form method="post">

<p>enter your mark accordingly</p>

<label>q1</label>

<input type="text" id="q1" />

<label>q2</label>

<input type="text" id="q2" />

<label>q3</label>

<input type="text" id="q3" />

<label>q4</label>

<input type="text" id="q4" />

<label>q5</label>

<input type="text" id="q5" />

<button type="submit">enter</button>

</form>

</div>

<br />

<script>

function imagechange() {

var image\_1 = document.getElementById('question\_image')

this\_value = this.value

if (this\_value = 1) {

image\_1.src = "static/questions/q{{user.id}}0-ans.png";

}

else if (this\_value = 2) {

image\_1.src = "static/questions/q{{user.id}}1-ans.png"

}

else if (this\_value = 3) {

image\_1.src = "static/questions/q{{user.id}}2-ans.png"

}

else if (this\_value = 4) {

image\_1.src = "static/questions/q{{user.id}}3-ans.png"

}

else if (this\_value = 5) {

image\_1.src = "static/questions/q{{user.id}}4-ans.png"

}

}

</script>

<centre>

<div class="pagination\_section" style="margin-left:200px">

<a title="question 1" onclick="imagechange()" id="q1" value="1">1</a>

<a title="question 2" id="q2" onclick="imagechange()" value="2">2</a>

<a title="question 3" id="q3" onclick="imagechange()" value="3">3</a>

<a title="question 4" id="q4" onclick="imagechange()" value="4">4</a>

<a title="question 5" id="q5" onclick="imagechange()" value="5">5</a>

</div>

</centre>

<div>

<img src="static/questions/q**{{**user.id**}}**0-ans.png" alt="question 1" id="question\_image" style="margin-left:200px">

<img src="static/questions/q**{{**user.id**}}**1-ans.png" alt="question 2" id="question\_image" style="margin-left:200px">

<img src="static/questions/q**{{**user.id**}}**2-ans.png" alt="question 3" id="question\_image" style="margin-left:200px">

<img src="static/questions/q**{{**user.id**}}**3-ans.png" alt="question 4" id="question\_image" style="margin-left:200px">

<img src="static/questions/q**{{**user.id**}}**4-ans.png" alt="question 5" id="question\_image" style="margin-left:200px">

</div>

<br />

{% endblock %}

### Uploading answer images

{% extends "base.html" %} {% block title %}Upload images{% endblock %} {% block content

%}

<p>Please try to upload one image per question</p>

<br />

<form method=post enctype=multipart/form-data>

<input type=file name=file>

<input type=submit value=Upload>

</form>

{% endblock %}

### Marking other people answers

{% extends "base.html" %} {% block title %}Logs{% endblock %} {% block content

%}

<link rel="stylesheet" href="static/styles.css" />

<p>Your marking **{{**mark\_user**}}** answers</p>

<h5>Make sure to mark fair!</h5>

<!--Changing the image using a javascript function-->

<centre>

<div class="pagination\_section" style="margin-left:200px">

<a title="question 1" onclick="imagechange()" id="q1" value="1">1</a>

<a title="question 2" id="q2" onclick="imagechange()" value="2">2</a>

<a title="question 3" id="q3" onclick="imagechange()" value="3">3</a>

<a title="question 4" id="q4" onclick="imagechange()" value="4">4</a>

<a title="question 5" id="q5" onclick="imagechange()" value="5">5</a>

</div>

</centre>

<div>

<p>Once marked enter the mark recived for each question</p>

<label>q1</label>

<input type="text" id="q1" />

<label>q2</label>

<input type="text" id="q2" />

<label>q3</label>

<input type="text" id="q3" />

<label>q4</label>

<input type="text" id="q4" />

<label>q5</label>

<input type="text" id="q5" />

<input type="submit" />

</div>

<br />

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

function imagechange() {

var image\_1 = document.getElementById('question\_image')

this\_value = this.value

if (this\_value = 1) {

image\_1.src = "static/marking/mark1.png";

}

else if (this\_value = 2) {

image\_1.src = "static/marking/mark2.png"

}

else if (this\_value = 3) {

image\_1.src = "static/marking/mark3.png"

}

else if (this\_value = 4) {

image\_1.src = "static/marking/mark4.png"

}

else if (this\_value = 5) {

image\_1.src = "static/marking/mark5.png"

}

}

</script>

<div>

<p>1</p>

<img src="static/marking/mark**{{**user.id**}}**0.png" alt="question 1" id="question\_image" style="margin-left:200px">

<p>2</p>

<img src="static/marking/q**{{**user.id**}}**1-ans.png" alt="question 2" id="question\_image" style="margin-left:200px">

<p>3</p>

<img src="static/marking/q**{{**user.id**}}**2-ans.png" alt="question 3" id="question\_image" style="margin-left:200px">

<p>4</p>

<img src="static/marking/q**{{**user.id**}}**3-ans.png" alt="question 4" id="question\_image" style="margin-left:200px">

<p>5</p>

<img src="static/marking/q**{{**user.id**}}**4-ans.png" alt="question 5" id="question\_image" style="margin-left:200px">

</div>

<br />

{% endblock %}

### When no answers are available to mark

{% extends "base.html" %} {% block title %}Stats :D{% endblock %} {% block content

%}

<h5>Seems like there are no questions to mark! Check back later :D</h5>

{% endblock %}

### Displaying stats

{% extends "base.html" %} {% block title %}Stats :D{% endblock %} {% block content

%}

<html>

<body>

<p>See what topics you are weak in and try to improve :D</p>

<hr />

<div>

<h2>Your logs :D</h2>

**{{**plot\_div1 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div2 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div3 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div4 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div5 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div6 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div7 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div8 | safe**}}**

</div>

<hr />

</body>

</html>

{% endblock %}

### Maths stats

{% extends "base.html" %} {% block title %}Stats for maths:D{% endblock %} {% block content

%}

<html>

<body>

<p>See what topics you are weak in and try to improve :D</p>

<hr />

<div>

<h2>Your logs :D</h2>

**{{**plot\_div1 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div2 | safe**}}**

</div>

<hr />

<div>

**{{**plot\_div3 | safe**}}**

</div>

<hr />

</body>

</html>

{% endblock %}

### Chat.html

{% extends "base.html" %} {% block title %}messages{% endblock %} {% block content

%}

<html>

<head>

<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<link rel="stylesheet" type="text/css" href="static/styles.css">

<script src="https://cdnjs"></script>

<script>

</script>

</head>

<body>

<p id="sender-input" value="**{{**user.id**}}**">**{{**user.id**}}**</p>

<form id="chat-form">

<select id="recipient-select">

{% for user in users %}

<option value="**{{** user.id **}}**">**{{** user.email **}}**</option>

{% endfor %}

</select>

<input type="text" id="message-input" placeholder="Enter your message">

<button type="submit" id="send-button">Send</button>

</form>

<script>

function updateRecipientList() {

fetch('/users')

.then(function (response) {

return response.json();

})

.then(function (users) {

var select = document.querySelector('#recipient-select');

for (var i = 0; i < users.length; i++) {

var option = document.createElement('option');

option.value = users[i][0];

option.text = users[i][1];

select.add(option);

}

});

}

window.onload = function () {

updateRecipientList();

}

document.querySelector('#send-button').addEventListener('click', function () {

var sender = document.querySelector('#sender-input').value;

var recipient = document.querySelector('#recipient-select').value;

var message = document.querySelector('#message-input').value;

var xhr = new XMLHttpRequest();

xhr.open("POST", "/send", true);

xhr.setRequestHeader('Content-Type', 'application/json');

xhr.send(JSON.stringify({ recipient: recipient, message: message }));

});

</script>

<hr />

<h5>Below will be the messages you have been sent</h5>

<div id="message">

{% for message in messages %}

<div class="card border-dark">

<div class="card-header d-flex justify-content-between align-items-center">

<a href="/messages/**{{**message.email**}}**">**{{**message.email**}}**</a>

</div>

<div class="card-body">

<div class="card-text">**{{**message**}}**</div>

</div>

</div>

{%else%}

{% endfor %}

</div>

</div>

</body>

</html>

{% endblock %}

# Testing

## Video

### Links

Qr code

Description automatically generated <https://youtu.be/OH038g1Lc2o>

### Video analysis

#### 00:00

The buttons all lead to the same sign-up page, there is no sole maths or sole biology signup/home page

(Ignore the images, I didn’t know)

#### 00:00 – 00:39

The first error sign-up is to show that only emails with gmail or outlook (not shown in the video but coded in) are allowed to be used

Creating a student account

#### 00:58 – 01:40

All the logs initial position is 0, so any topic that arent studied, the user doesn’t have to change them

All the topics have there own table in the database.

#### 01:44 – 02:06

The jiggle physics didn’t work for maths logs.

#### 02:43 – 03:08

Here the dynamic list lets you pick an option which then generates 5 questions

There are two buttons because one I couldn’t figure out how to make it go to the question page and it acts as a conformation because the user can change there topic if they want.

#### 03:09 – 03:40

The number 16 is the id of the account

There is a visual timer, which tells the user how long it’s been but the actual time that is stored has milliseconds



#### 03:40 – 03:47

The second button just leads you to a page that allows you to upload an image (not shown in the video but there are screenshot of how it works later on)

#### 03:47 – 04:40

No matter how many times the user enters values the numbers will only change the 5 as shown in the video.

As of right now there are no limitation from the stopping the user from entering something out of bound, e.g., a mark that is above the amount you can even get for that paticular question.

#### 04:44 – 04:58

As of right there are no maths questions in the database, which is why only the topic selection bit was shown for maths. Although everything is supposed to work just like how it did for biology.

#### 05:03 – 05:06

Shows this when there are no user entered questions to mark.

#### 05:19 – 06:21 (07:58 – 08:52)

The graph show the mean of each log. Later in the video I show a log of an account that has entered logs several time and how that looks compared to the one shown now.

#### 06:29 – 06-49

This is where you student accounts can see an email of a registered teacher on the website, and they can contact them and ask for help

#### 06:52 – 07-51

The dropdown list has all the accounts in the database and updates whenever a new account register.

The message is saved to the database and then shown to the recipient, as shown in the video.

The way the text is shown is weird, because I didn’t structure it better and just shows the message and who sent it.

#### 09:30 – 10:05

The users should be able to delete post, but I couldn’t figure out why it wasn’t working.

Again, there are no limitaion to what the user can enter. (just like shown in the video)

#### 10:05 – end

Everthing else is the same as the student account.

## Testing entering values to the database.

### Graphical user interface, application Description automatically generatedQuestions

Database containing the questions. (Not all of them are shown in the image)

### Communication

Graphical user interface, application

Description automatically generatedSending this message from account 1

Graphical user interface, text, application, email

Description automatically generatedWhats shown in the idktest@gmail.com account.

### Adding Images to database

Graphical user interface, text, application

Description automatically generated

Saved to database table with the corresponding question\_done\_id

Graphical user interface, text, application, email

Description automatically generated

### Logs

Adding 100 for lipids and 0 for the rest of the topics

A picture containing text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedWhat it shows on the database table: topic1

# Evaluation:

First and foremost, I did not end up coding everything I intended and definitely underestimated how much learning while coding I would have to do with adding html and JavaScript to my code. I was taking to much time coding sections of this website that left me with little to no time for some of the other sections.

A few things I wanted to add but couldn’t, were a video chatting service so the teacher accounts could actually physically teach the student. I attempt to code it and got as far as getting my camera showing on the screen with the other user not connecting. So, I ended up not adding it and instead a database focused chat service, not as cool but can communicate. This took a lot of time, trying to find a way to make it work but not fully understanding the concept, so it ended up being a waste of time.

Another thing I wished I would have added was more data analysis. With the website getting the data about the marks a student gets in each question they do, how much time they spend on the 5 questions and their logs, I could have done a lot of data analysis and present it as a dashboard of progress for the student. Such as taking the marks they got in a question and seeing their previous progress on the same topic to make a learning curve for the student.

Another thing I wished I went in detail with the limitations, currently any image can be added as an ‘answer image’, any marks can be added without there being a limit to them, if time allowed it I would have had limitation to stop someone from entering something inappropriate or largely inaccurate.

1. <https://thinkstudent.co.uk/how-hard-is-a-level-biology/> [↑](#footnote-ref-1)
2. <https://www.thesun.co.uk/news/9724339/a-level-maths-exam-grade-boundaries/> [↑](#footnote-ref-2)
3. <https://resources.workable.com/hr-terms/what-are-soft-skills> [↑](#footnote-ref-3)
4. <https://mobilehealthdata.com/effects-of-mental-health-on-academic-performance/> [↑](#footnote-ref-4)
5. <https://www.physicsandmathstutor.com/> [↑](#footnote-ref-5)
6. <https://www.savemyexams.co.uk/> [↑](#footnote-ref-6)
7. <https://webflow.com/> (how the website will look was made using this) [↑](#footnote-ref-7)