Defining the problem

My client is a senior student nearing the end of their A-Levels diploma. Upon the interview with them[1], they explained their enjoyment in creating concise notes to help them better prepare for any upcoming exams. They have started to make flashcards due to their simplicity and efficiency in memory training. The client has considered transitioning their methods of making flashcards from paper-based cards to a website or application that allows them to easily study for their economics classes.

During the conversation, they addressed the current problems of using an older computer and their worry of not being able to save the cards after they have been created. Although there are other applications for the client to create their flashcards offline, they state that the vast majority are very complex and do not implement any scheduling tools to ensure that they are developing their knowledge on a frequent basis. With the understanding that they would also like to review their study sets on the go when they do not have their computer, they requested that the proposed solution involves the program running on any smart device they choose to use.

Rationale for the Proposed Solution

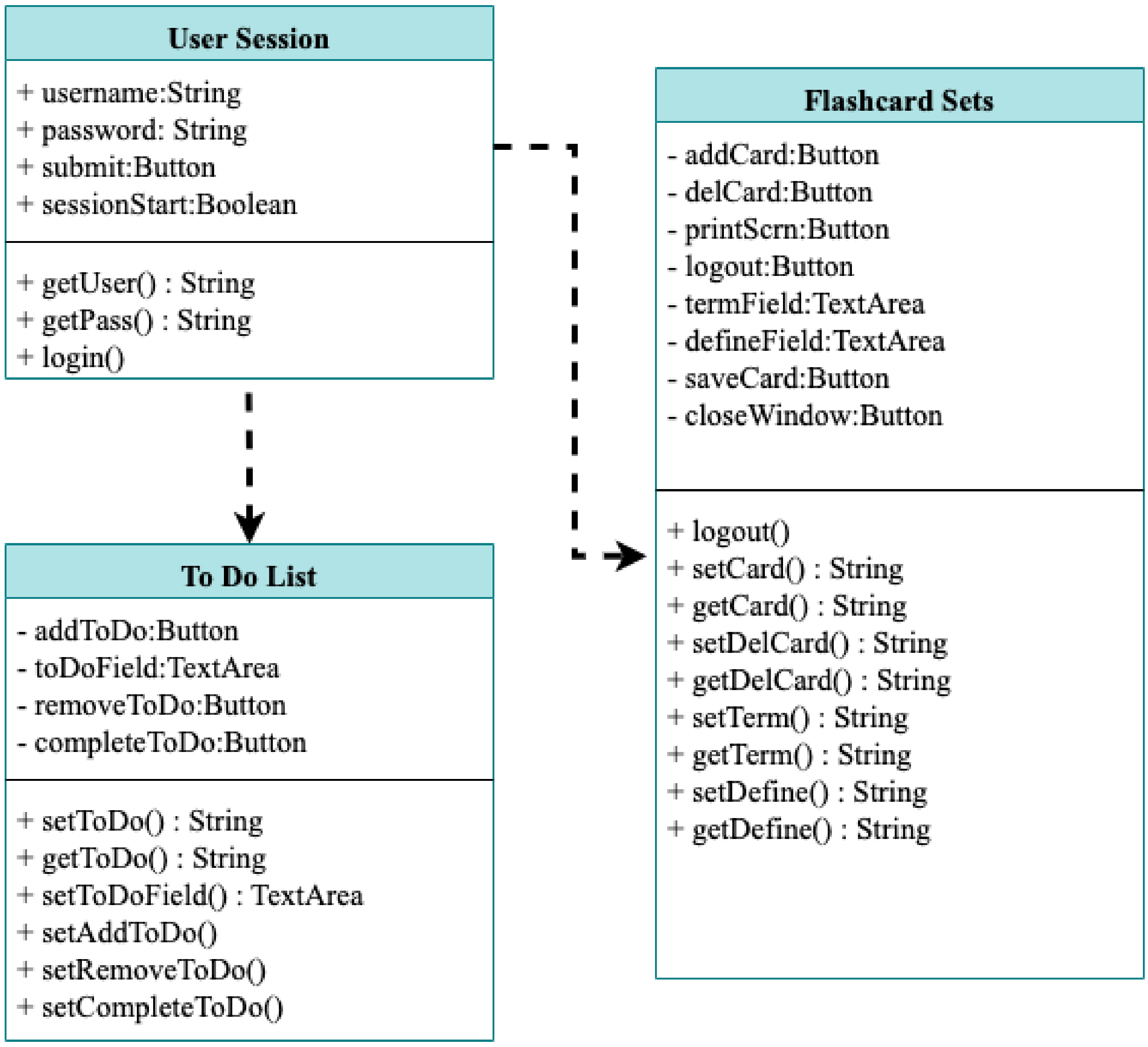
Based on the requirements for both functionality and workability, I will be using Visual Studio Code to create the website-based flashcard tool with regards to my familiarity with website development. By considering the capabilities of the client's devices, there will not be any additional requirements or hardware needed to enter and interact with the website. The only essential is a stable internet connection, which they already have. With previously failed experiences of linking a third-party database to manage user inputs, I decided that a more suitable approach for the client's flashcard management website would be to store their cards through a local directory, without the need of opening the cards every instance they access the site. This workaround would be identical to a database, in which case the information will be pulled and collected from the devices used. The strategies of considering colour theory will also be approached to create an aesthetically pleasing website, desired by the client himself. With regards to portability, the website does not require any additional hardware to be run on the specified laptop. Final considerations of usability were evaluated, and recommendations of also being able to print the flashcard sets were also advised.

[1] *Appendix A, p.22*

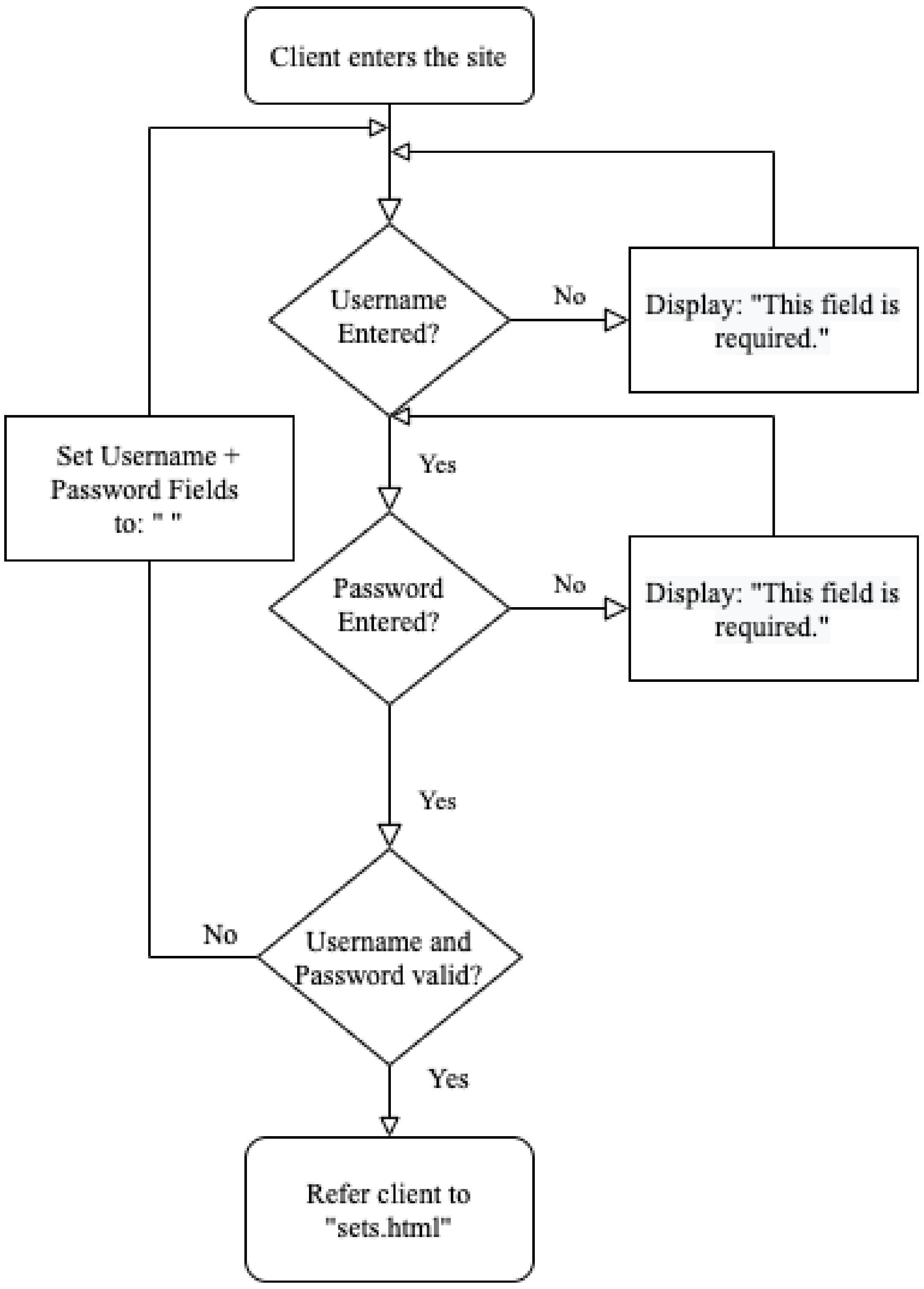
<u>Success Criteria</u>

- Ability to authenticate the client into the web application, preferably through a login screen.
- Function that creates a session once the client's credentials have been entered correctly into an authentication page.
- Ability to redirect the client to a following page once the session is created.
- Clicking on options of being redirected to sets or planning page works.
- Ability to refer my client to different economics flashcards sets of their choosing.
- Ability to redirect my client to previous pages visited in the web application.
- Function to add a flashcard to a chosen set.
- Function to delete all cards in a chosen set.
- Function that allows the client to print all cards.
- The client has the ability to enter to-do list items into a form which creates a to-do item.
- Created to-do list items may be filtered as completed, incompleted or even removed.

UML Summary of Classes

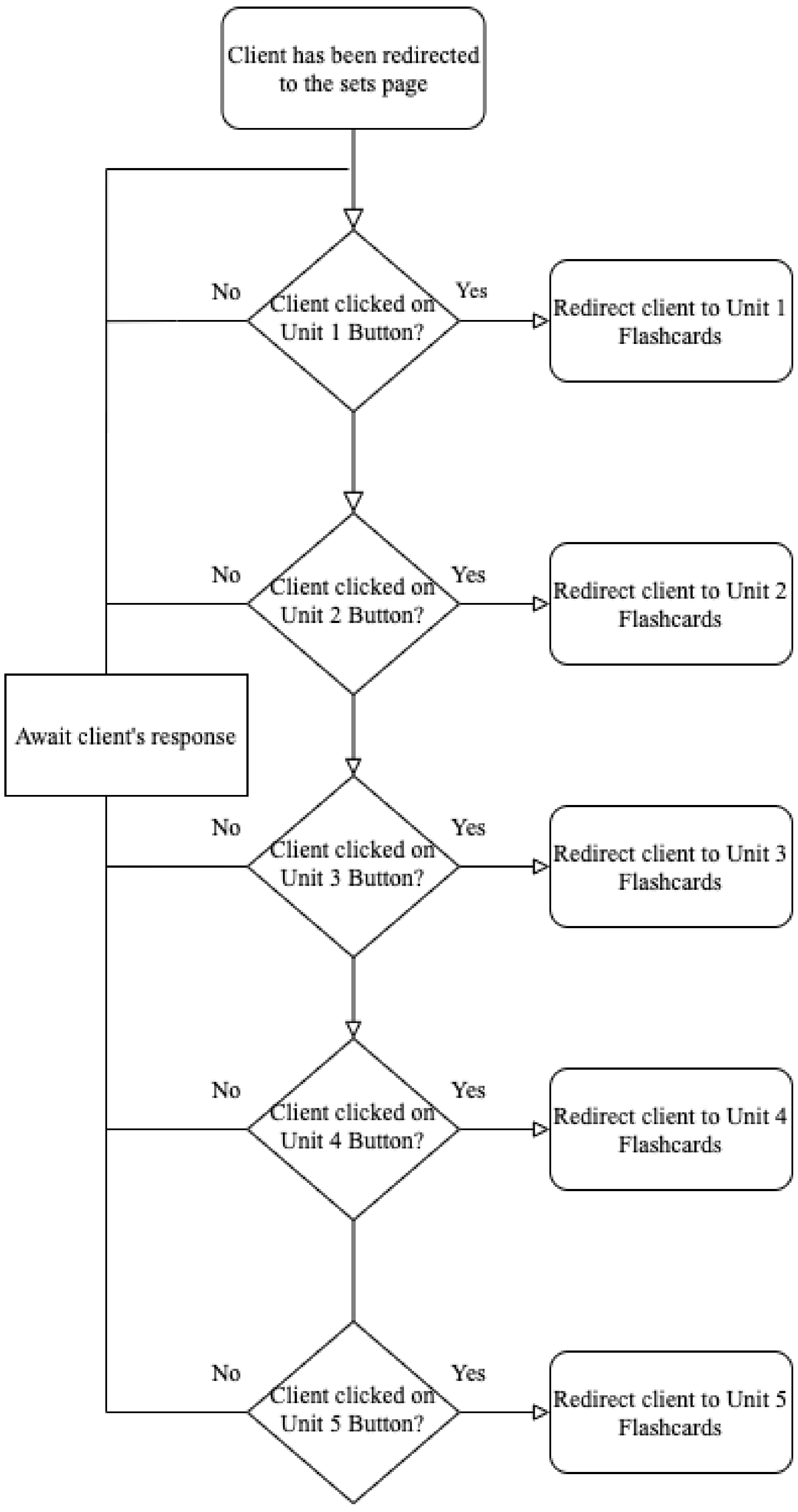


Flowchart 1: Validation of Login Page

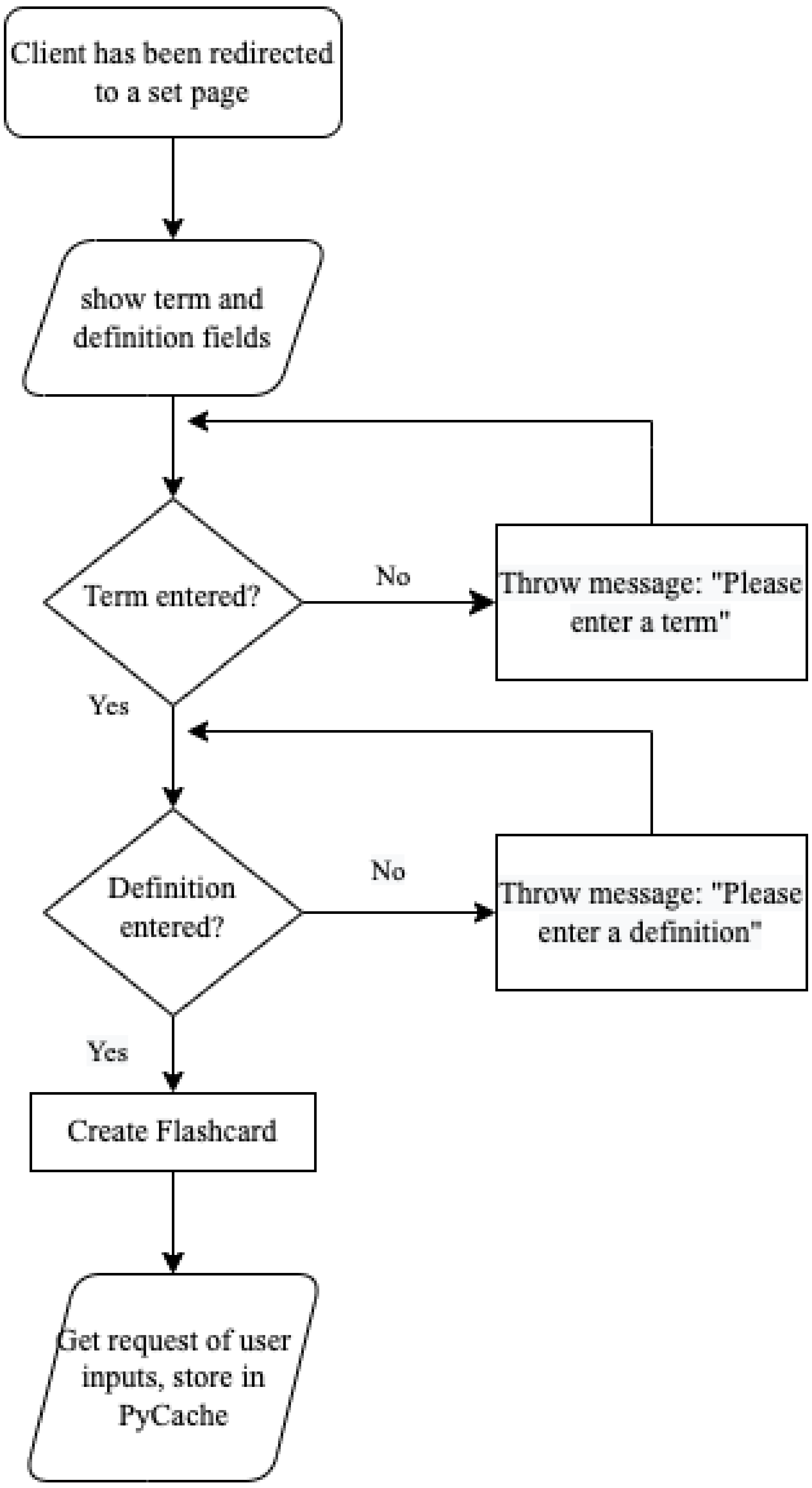


Client enters the site
Username Entered?
No
Display: "This field is required."
Yes
Password Entered?
No
Display: "This field is required."
Set Username + Password Fields to: " "
Yes
No
Username and Password valid?
Yes
Refer client to "sets.html"

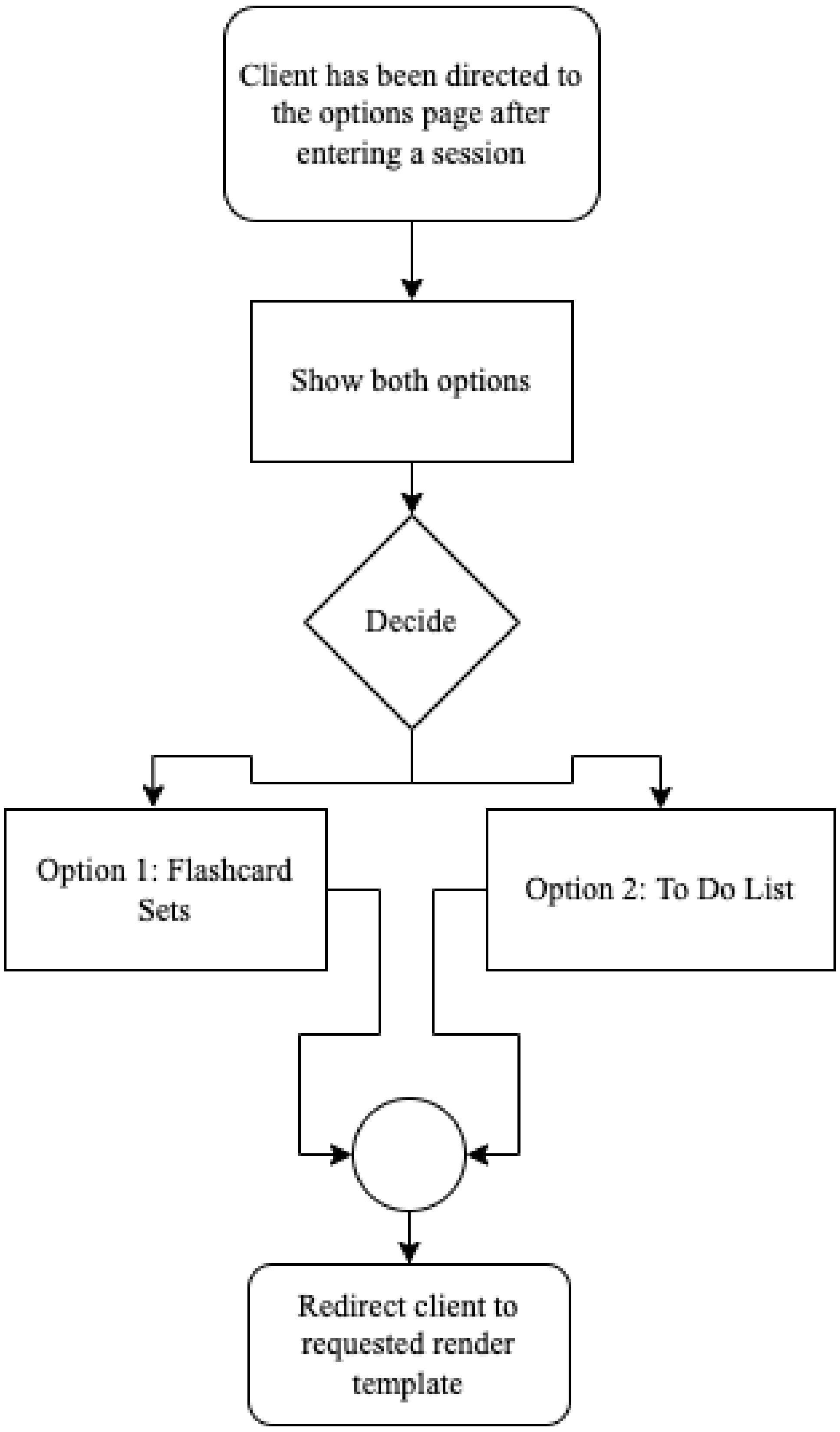
Flowchart 2: Interaction between the client and buttons which redirect to the various sets.


Client has been redirected to the sets page
Client clicked on Unit 1 Button?
No
Yes
Redirect client to Unit 1 Flashcards
Client clicked on Unit 2 Button?
No
Yes
Redirect client to Unit 2 Flashcards
Await client's response
Client clicked on Unit 3 Button?
No
Yes
Redirect client to Unit 3 Flashcards
Client clicked on Unit 4 Button?
No
Yes
Redirect client to Unit 4 Flashcards
Client clicked on Unit 5 Button?
No
Yes
Redirect client to Unit 5 Flashcards

Flowchart 3: How the client would insert a new flashcard to a specified set.

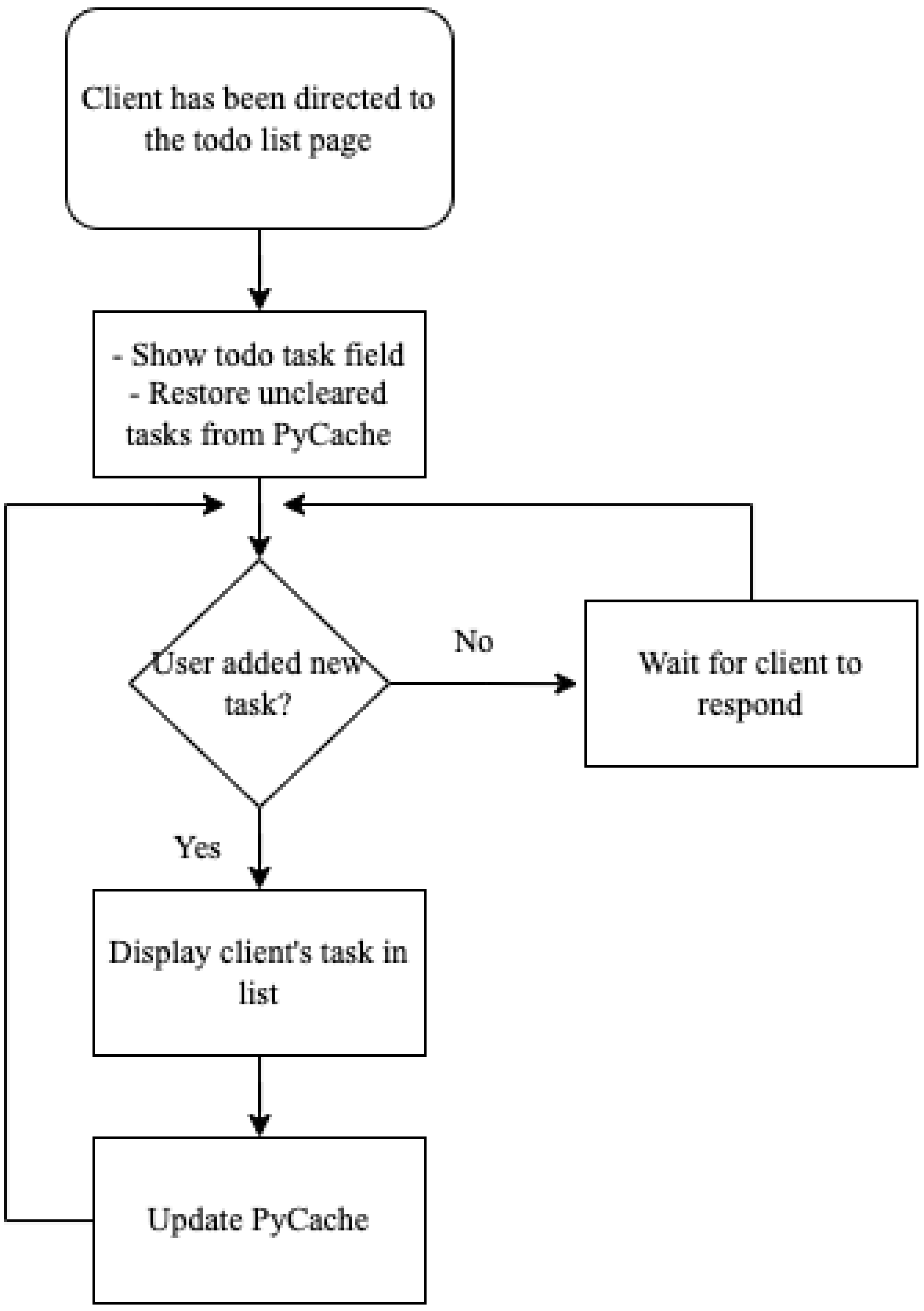

Client has been redirected to a set page
show term and definition fields
Term entered?
No
Throw message: "Please enter a term"
Yes
Definition entered?
No
Throw message: "Please enter a definition"
Yes
Create Flashcard
Get request of user inputs, store in PyCache

<u>Flowchart 4: Selection of options for the client (Flashcard sets or to-do list).</u>



Client has been directed to the options page after entering a session
Show both options
Decide
Option 1: Flashcard Sets
Option 2: To Do List
Redirect client to requested render template

Flowchart 5: Creating a new task for the user through the to-do list


Client has been directed to the todo list page
- Show todo task field
- Restore uncleared tasks from PyCache
User added new task?
No
Wait for client to respond
Yes
Display client's task in list
Update PyCache

Test Plan

Test Type	*Nature of Testing*	*Expected Outcome*
Ability to authenticate the client into the web application, preferably through a login screen.	Check that each element is visible on the webpage.	This would be the opening screen for the client once they enters the website.
Function that creates a session once the client's credentials have been entered correctly into an authentication page.	Check that inputs are submitted correctly and redirected to the following page.	Once details are entered, a new session is created.
Ability to redirect the client to a following page once the session is created.	Click on the 'login' button which is on the login page.	the client will be redirected to the homepage after details are validated.
Ability to refer my client to different economics flashcards sets of thier choosing.	The client would click on both buttons to be redirected.	The buttons are centred on the screen which would send the client to either render template of choice.
Ability to redirect my client to previous pages visited in the web application.	Buttons on these pages redirect to the allocated render templates to make the flashcards.	the client is forwarded to the correct set of links after clicking the labelled buttons.
Function to add a flashcard to a chosen set.	Clicking the button 'Add Card' shows the popup to enter in terms and definitions.	This pop-up is visible on the webpage and may be closed when needed.
Function to delete all cards in a chosen set.	Clicking this button will clear all visible flashcards in the set.	All cards with terms and definitions are removed from the screen.
Function that allows the client to print all cards.	Check that the 'Print' button is visible in the flashcard sets' menu bars.	When interacting with these buttons, a print page preview is shown to the client.
The client has the ability to enter to-do list items into a form which creates a to-do item.	The text field is visible within the webpage and this is usable by the client.	To-do list items are presented in a container on the screen.
Created to-do list items may be filtered as completed, incompleted, or even removed.	Check that both buttons of the green tick and red bin work.	Clicking the green tick should present a strikethrough the to-do item and the red bin should fade the task away from the screen.

Graphical Visualisation

Drafted Designs

Illustration 1: Flashcard Creator Tool Login Page

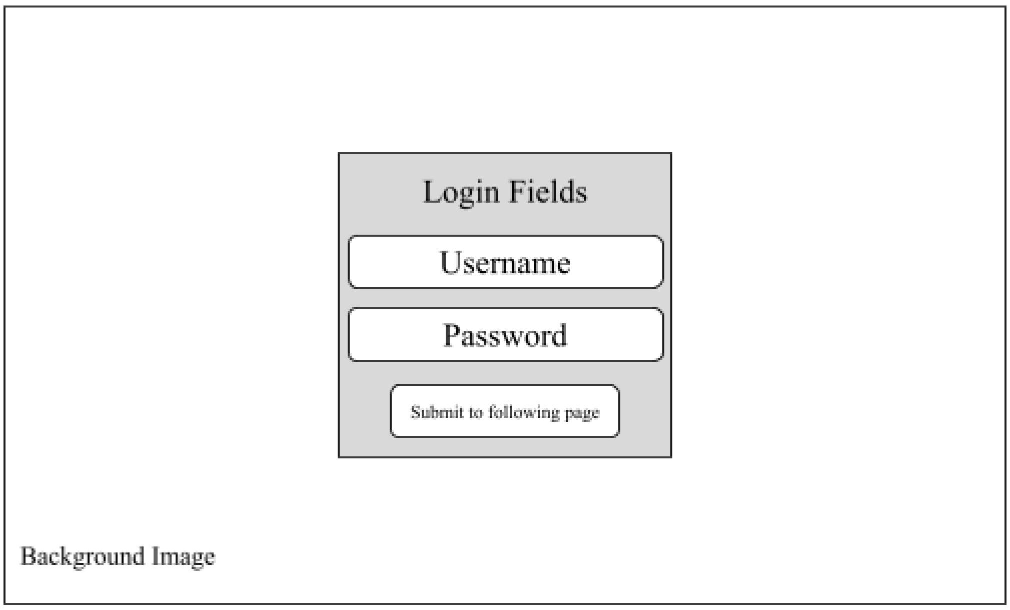


Illustration 2: Flashcard Sets Page

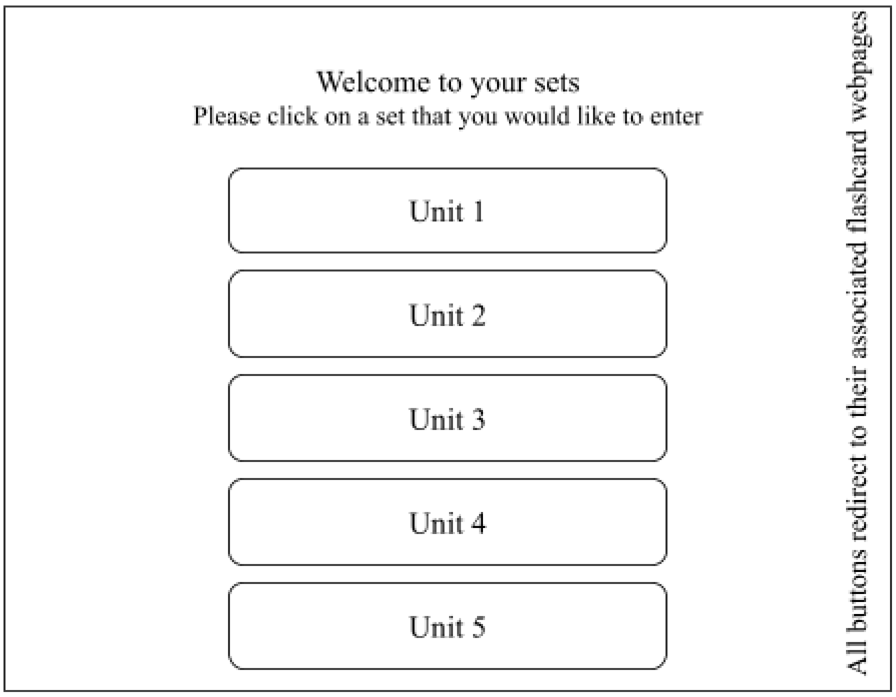
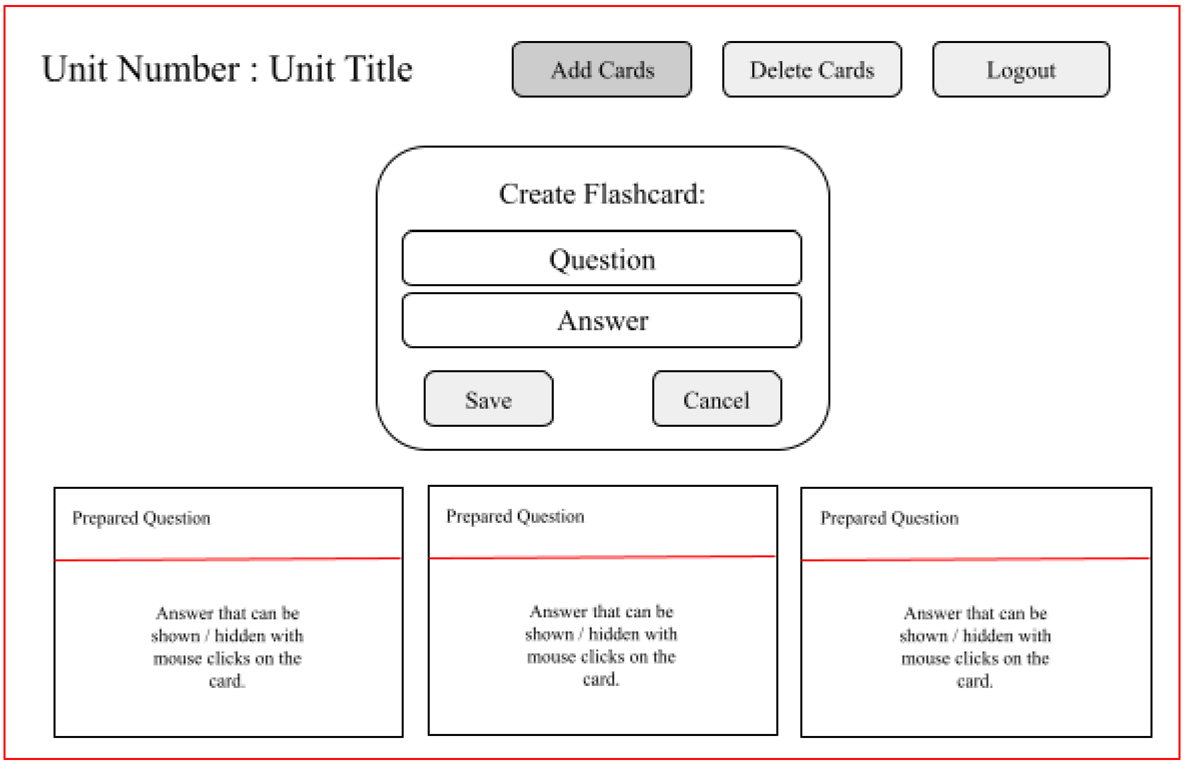

Welcome to your sets
Please click on a set that you would like to enter
Unit 1
Unit 2
Unit 3
Unit 4
Unit 5
All buttons redirect to their associated flashcard webpages

Illustration 3: Flashcard Pages:


Unit Number : Unit Title
Add Cards
Delete Cards
Logout
Create Flashcard:
Question
Answer
Save
Cancel
Prepared Question
Answer that can be shown / hidden with mouse clicks on the card.
Prepared Question
Answer that can be shown / hidden with mouse clicks on the card.
Prepared Question
Answer that can be shown / hidden with mouse clicks on the card.

Second Draft of Designs

The following designs were constructed through Google Slides.

Illustration 4: Login Page

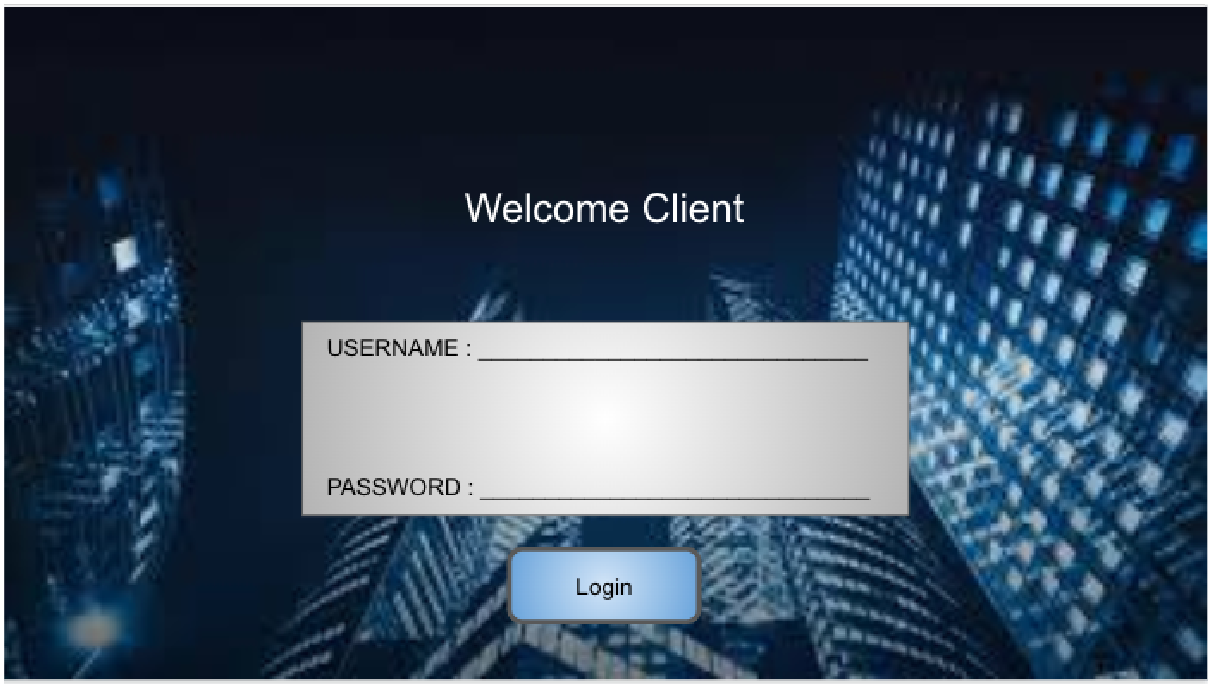


Illustration 5: Sets Page

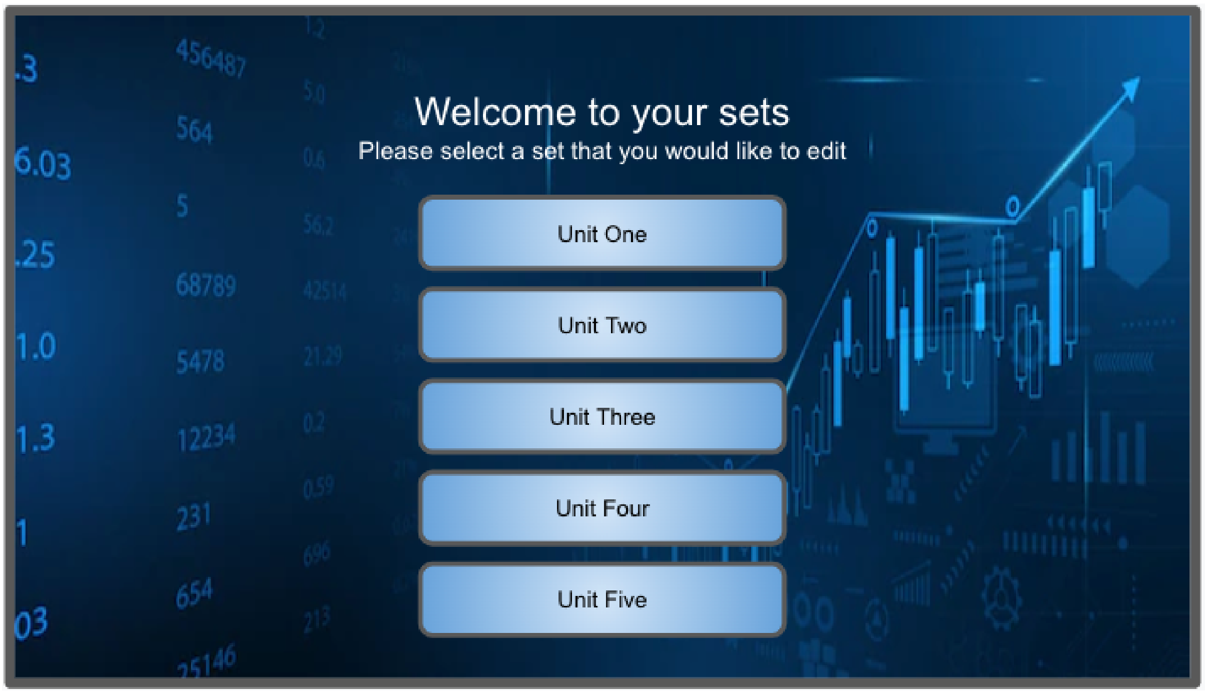
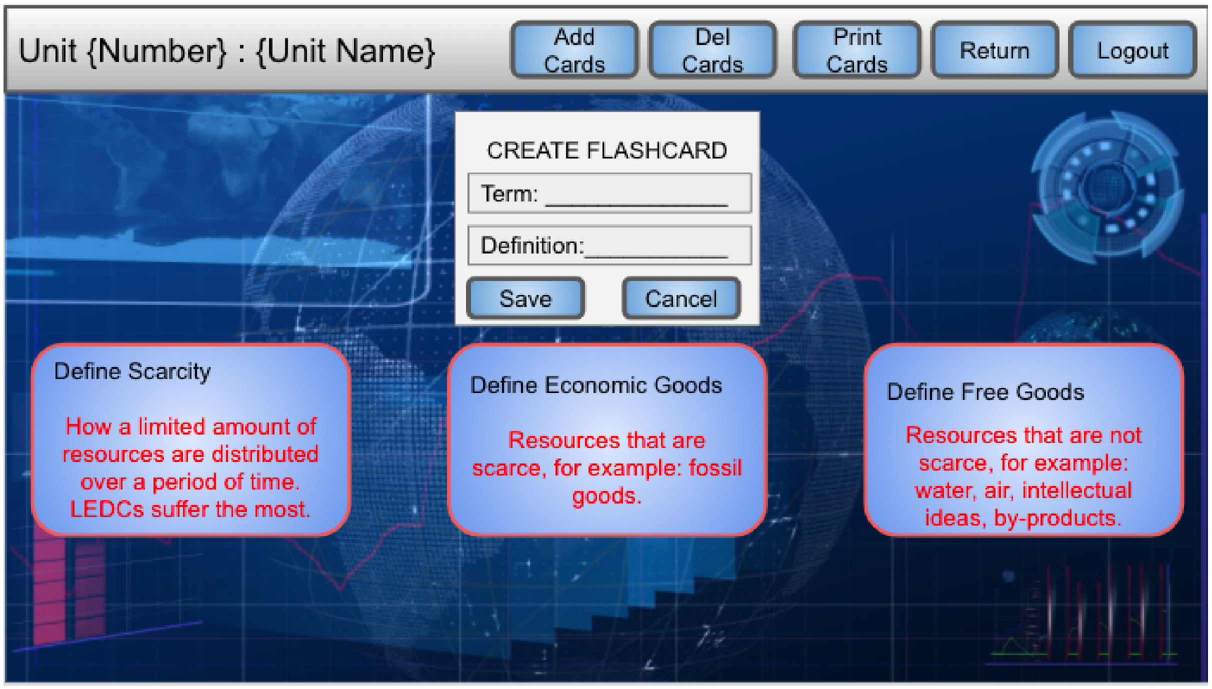


Illustration 6: Page that holds the cards



Define Scarcity

How a limited amount of resources are distributed over a period of time. LEDCs suffer the most.

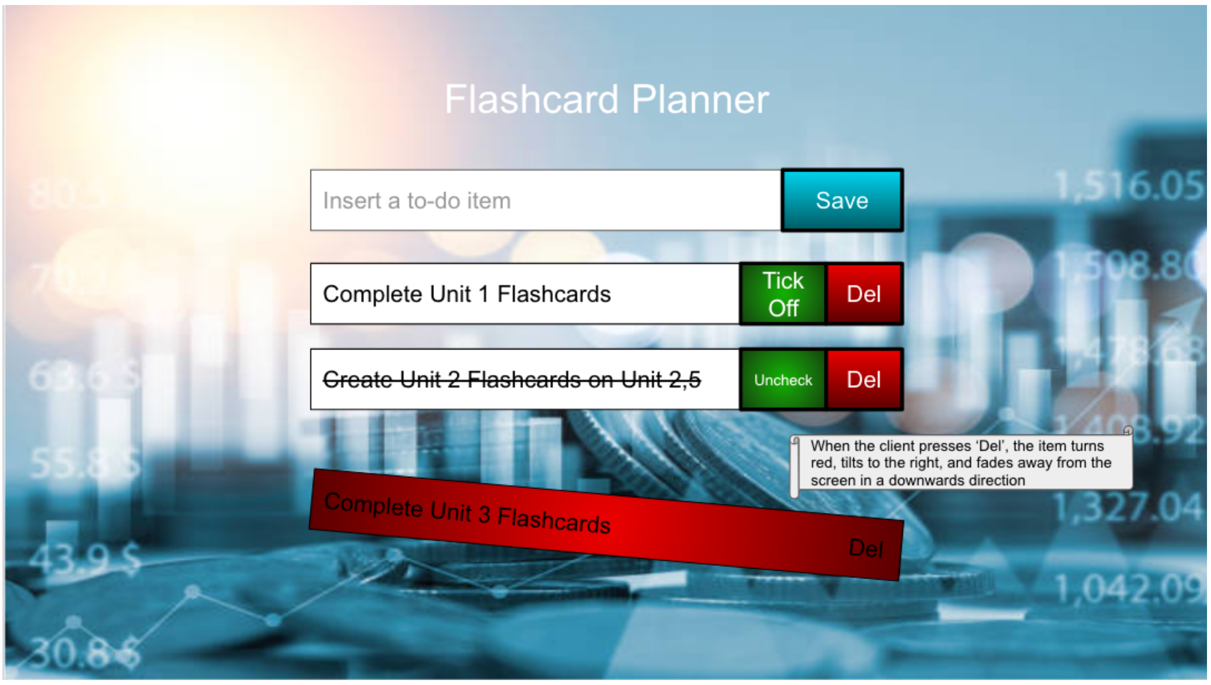
Define Economic Goods

Resources that are scarce, for example: fossil goods.

Define Free Goods

Resources that are not scarce, for example: water, air, intellectual ideas, by-products.

Illustration 7: To-Do List Page



Complete Unit 1 Flashcards

~~Create Unit 2 Flashcards on Unit 2,5~~

Complete Unit 3 Flashcards

When the client presses 'Del', the item turns red, tilts to the right, and fades away from the screen in a downwards direction

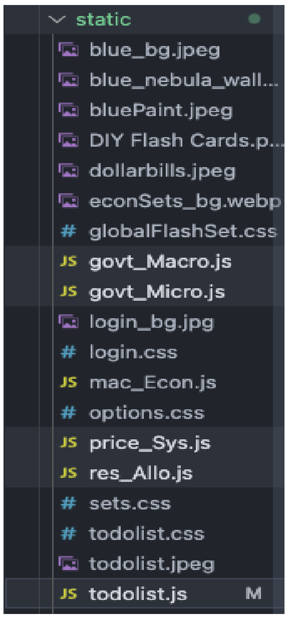
Record of Tasks

Task Number	Planned Action	Planned Outcome	Time Estimated	Target Completion Date	Criterion
1	Contact the client	A success criteria is listed	2 days	21/6/21	A
2	Initial Interview with the client	Walkthrough the success criteria and their expectations of the application's functionality. Feasibility of constructing this in Java was also mentioned	1 day	22/6/21	A
3	Interview with Computer Science teacher	Explain all current findings and expectations of the client	1 week	29/6/21	A
4	Refine success criteria and confirm with the client	Concise goals that are achievable when coding the application.	1 week	5/7/21	A
5	Begin writing Criterion A documentation	Finished copy of Criterion A for client to read through	1 week	19/7/21	A
6	Sketch initial designs	Confirm with the client whether the ideas are satisfactory and can be worked on further	2 weeks	26/7/21	B
7	Create flowcharts to break down the steps for coding	Will give the client an understanding of the requirements that the application must contain	2 week	8/8/21	B
8	Implementing the designs into formats through HTML	Will give the client a better representation of the flashcard creator	1 week	15/8/21	C
9	Send sample images of website designs to the client	Receive initial fixes / adjustments that should be considered	3 days	18/8/21	C
10	Ensure that web pages can interact with one another.	Buttons and all other attributes displayed in the frame and work effectively.	1 week	25/8/21	C
11	Begin writing	Display all flowcharts,	5 days	30/8/21	B

	Criterion B documentation	illustrations, UML Diagram, Record of Tasks.			
12	Create remaining designs of Study, Edit and Create Panels in vscode	All css implementations are presentable in the expected looks.	1 week	6/9/21	C
13	Start creating main algorithms of all panels in NetBeans	Polish flowcharts and other design implementations	2 weeks	20/9/21	C
14	Inform client about progress and ask for any additional changes that should be considered	Clear and concise points are provided by the client and edits to designs or flowcharts will be made accordingly	2 weeks	4/10/21	C
15	Bug fixing any errors that needed rainchecking	Solutions to fixing the unexpected bugs will be approached and applied	1 week	11/10/21	C
16	Alpha testing the workings of the flashcard creator	Showcase application to client	1 week	18/10/21	C
17	Reflecting on client's comments and implementing ideas	Changes may primarily consist of UI flaws. Any colour schemes, fonts, positions that the client wants changed will be modified	1 week	25/10/21	C
18	Alpha testing Flashcard Creator application with new implementations	Ensure that the new version is stable and minimum fixes are required	1 day	26/10/21	C
19	Upload code to github to run the website off of their server.	Confirmation from the client that they have their MacBook up-to-date with any system requirements.	1 day	27/10/21	C, E
20	Display current version of the Flashcard Creator Tool to Computer Science teacher and receive recommendations for improvement	Obtain opinions on the functionality and overall design of the application. Expecting mentions on additional fixes that should be considered for the final version of the application.	1 week	3/11/21	C

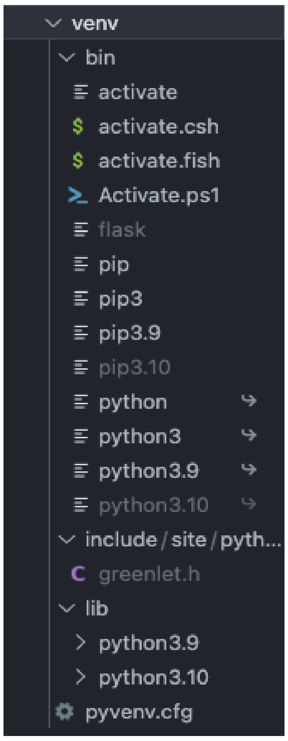
21	Ask the client for feedback on the completed application, and if there are any rooms for improvement to benefit their experience in using the website.	Obtain final feedback and make changes to both UI and back-end development.	1 week	10/11/21	E
22	Finalise and complete all Criterion C discussions.	Ensure that I am concise with my extended writing, and that I am constantly stating the benefits of using the approaches I used instead of simply explaining what I did.	2 weeks	24/11/21	C
23	Installation of web application to client's computer.	All files are exported successfully to the client's computer so that they are able to run the program.	1 day	25/11/21	E
24	Training my client how to use the web application.	The client is able to work their way across the web application with ease and understands each function associated to the program that they will be using for their flashcards	1 day	25/11/21	E
23	Final interview with client.	Receive information regarding the advantages and disadvantages of the application.	1 day	30/1/22	E
24	Complete the write up for Criterion E	The section contains all the necessary aspects to complete Criterion E	1 week	2/2/22	E

Figure 1: List of static classes used for both front and backend development


```
static
  blue_bg.jpeg
  blue_nebula_wall...
  bluePaint.jpeg
  DIY Flash Cards.p...
  dollarbills.jpeg
  econSets_bg.webp
  globalFlashSet.css
  govt_Macro.js
  govt_Micro.js
  login_bg.jpg
  login.css
  mac_Econ.js
  options.css
  price_Sys.js
  res_Allo.js
  sets.css
  todolist.css
  todolist.jpeg
  todolist.js          M
```

<u>Storing Dependencies</u>

Figure 2: Dependencies within the virtual environment


```
venv
  bin
    activate
    activate.csh
    activate.fish
    Activate.ps1
    flask
    pip
    pip3
    pip3.9
    pip3.10
    python
    python3
    python3.9
    python3.10
  include / site / pyth...
    greenlet.h
  lib
    python3.9
    python3.10
  pyvenv.cfg
```

The involvement of a virtual environment was important for the creation of the web application as it would install the required dependencies (such as Flask, OS, SQLAlchemy, etc.) into the local environment for this particular project. This helped to solve the success criteria to avoid installing the essential components into the system libraries. This was beneficial for prototyping the web application on my client's device as all dependencies would be installed on their virtual environment and would not alter or reconfigure any system libraries.

Installing Flask directly into the computer system libraries also installs additional libraries such as 'itsdangerous' and 'Werkzeug'[2]; these would be installed globally when they are potentially not needed for other projects. Therefore, isolation of python dependencies for different libraries or frameworks is appropriate to avoid using a "site-packages" repository that contains other versions of the same frameworks.

Thus, a fail-safe solution to avoid the incompatibility of file formats and/or essential libraries.

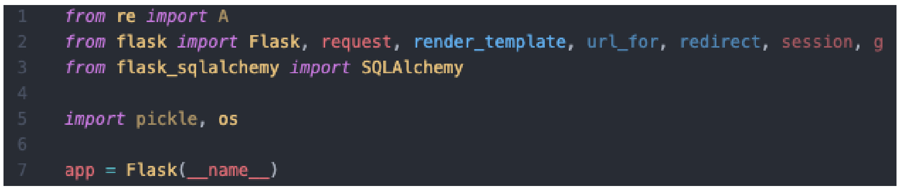
[2] *What are the advantages and disadvantages of using Flask as a web framework?*

<u>Implementation of a REST API</u>

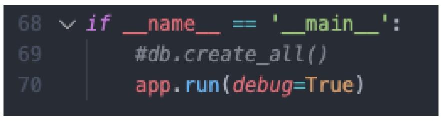
A Rest API is important to the development of my client's web application as it is abstract. This ensures easier syntax for constructing the service for my client. In addition to this, REST APIs are lightweight[3], provoking that I could easily incorporate HTML and JSON constructs. This works towards the success criteria of both sessions and adding cards to the sets onto my client's web application.

The specific API that has been involved in this project is Flask. I have chosen this API as it encourages modularity, which provides the ability to create live servers that can be used to test prototypes before deployment. Ultimately, it allows me to use these live servers to seek greater performance in meeting the success criteria, this is due to the involvement of asynchronous communication.

Source Code Extract 1: Flask REST API Integration



```python
from re import A
from flask import Flask, request, render_template, url_for, redirect, session, g
from flask_sqlalchemy import SQLAlchemy

import pickle, os

app = Flask(__name__)
```



```python
if __name__ == '__main__':
    #db.create_all()
    app.run(debug=True)
```

The above extract displays partial boilerplate code as well as my implementation for the development of this project.

Flask and Django were the two considerations in mind as the primary web framework for development. Django would not suit my client's success criteria as the approach follows the concept of 'convention over configuration'[4]. In essence, template modification would be rather difficult to satisfy my client's requirements of functionality (UX) and appearance (UI).

However, it is worthy to understand that Django does support object-relational mapping (ORM), leading to more iterations in developing such a web application like my client's. This alone has its own drawback whereby the risk of maintenance to the application is significantly lower than with Flask. The choice of both frameworks solely depends on either delivery of complex features (through Django) or performance (through Flask).

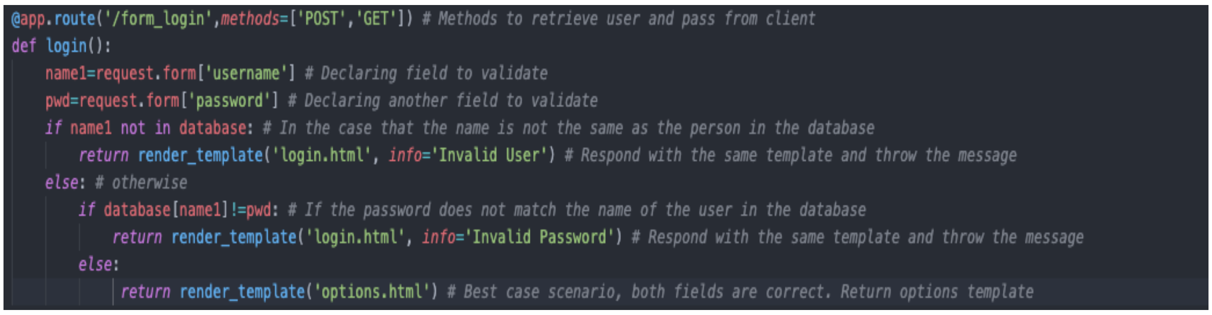
---

[3] *Pros and cons of client-side rendering, Pluralsight.com.*
[4] *Top 3 benefits of REST APIs, MuleSoft.*

Overall, ensuring performance to this particular application is more significant to meeting the success criteria and encourages technical experimentation whilst facing a more problematic technical stack.

<u>Routing with a templating language</u>

Source Code Extract 2: Use of Render Templates

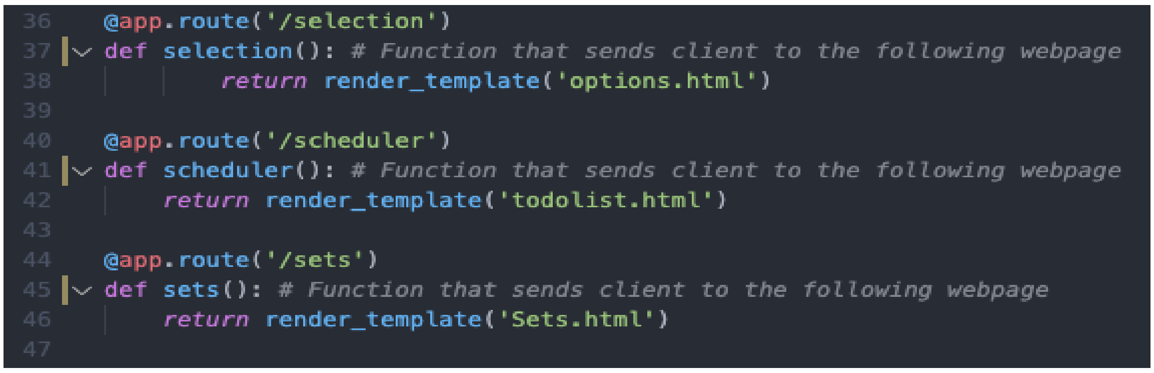


```python
@app.route('/form_login',methods=['POST','GET']) # Methods to retrieve user and pass from client
def login():
    name1=request.form['username'] # Declaring field to validate
    pwd=request.form['password'] # Declaring another field to validate
    if name1 not in database: # In the case that the name is not the same as the person in the database
        return render_template('login.html', info='Invalid User') # Respond with the same template and throw the message
    else: # otherwise
        if database[name1]!=pwd: # If the password does not match the name of the user in the database
            return render_template('login.html', info='Invalid Password') # Respond with the same template and throw the message
        else:
            return render_template('options.html') # Best case scenario, both fields are correct. Return options template
```

Routing was another significant factor in the development of my client's web application. The close association between efficiency and speed of rendering to a different webpage was essential to this project. Jinja2 templates played a huge role within the validation of usernames and passwords entered into the login screen due to asynchronous execution. Thus, the success criteria of a working login page were achieved.

It is also crucial to highlight the meaningful variables that have been incorporated into the login form. Variables of names "username", "password" suggest easy readability. The form titled "form_login" exclaims precisely what the algorithm aims to do.

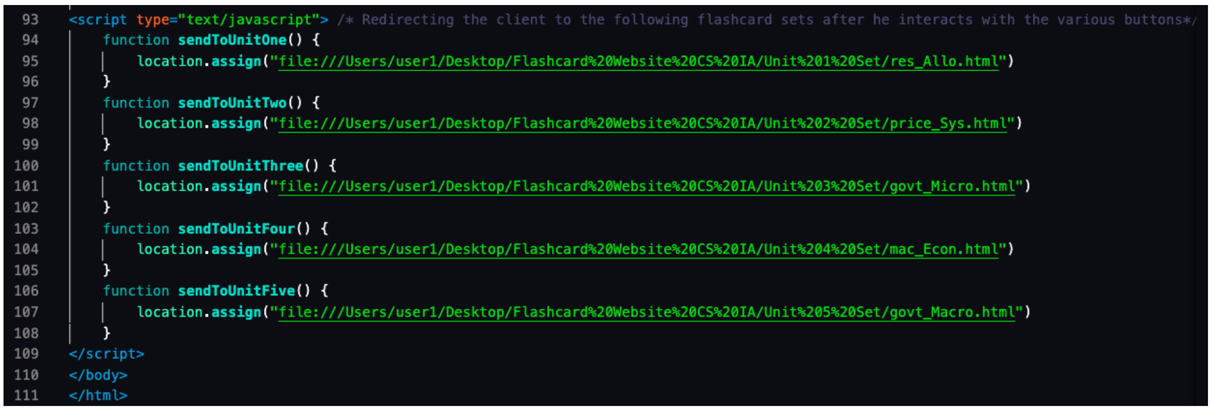
Source Code Extract 3: Declaration of location assignments with Jinja2



```python
@app.route('/selection')
def selection(): # Function that sends client to the following webpage
        return render_template('options.html')

@app.route('/scheduler')
def scheduler(): # Function that sends client to the following webpage
    return render_template('todolist.html')

@app.route('/sets')
def sets(): # Function that sends client to the following webpage
    return render_template('Sets.html')
```

In client-side routing, templates use functions that cause delays in execution (being synchronous). This correlates to routes and components having to be fetched and executed on the initial request.

Source Code Extract 4: Routing with JavaScript before involving Flask REST API



```
<script type="text/javascript"> /* Redirecting the client to the following flashcard sets after he interacts with the various buttons*/
    function sendToUnitOne() {
        location.assign("file:///Users/user1/Desktop/Flashcard%20Website%20CS%20IA/Unit%201%20Set/res_Allo.html")
    }
    function sendToUnitTwo() {
        location.assign("file:///Users/user1/Desktop/Flashcard%20Website%20CS%20IA/Unit%202%20Set/price_Sys.html")
    }
    function sendToUnitThree() {
        location.assign("file:///Users/user1/Desktop/Flashcard%20Website%20CS%20IA/Unit%203%20Set/govt_Micro.html")
    }
    function sendToUnitFour() {
        location.assign("file:///Users/user1/Desktop/Flashcard%20Website%20CS%20IA/Unit%204%20Set/mac_Econ.html")
    }
    function sendToUnitFive() {
        location.assign("file:///Users/user1/Desktop/Flashcard%20Website%20CS%20IA/Unit%205%20Set/govt_Macro.html")
    }
</script>
</body>
</html>
```

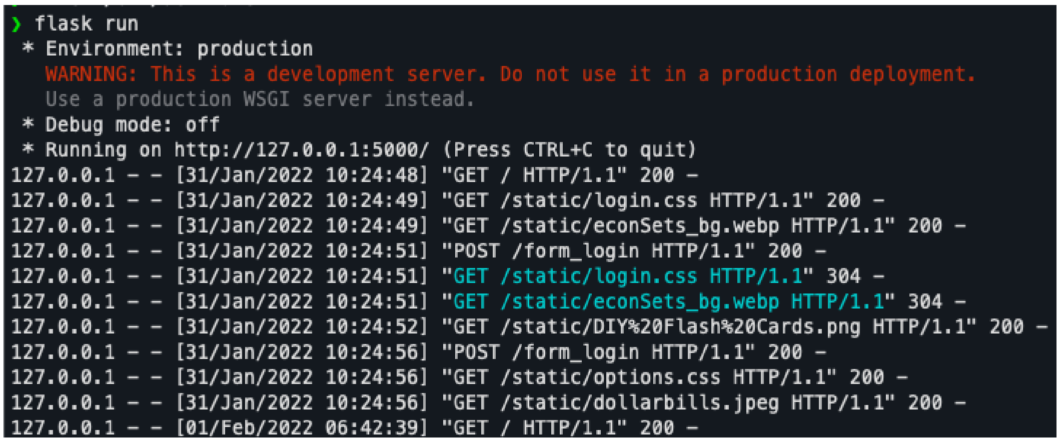
By formulating render templates through JavaScript (JS) as displayed above, browsers are unable to cache the structure of the pages. Though these functions could be considered as the modular tasks that make up the workings of the web page, caching the JS would prove costly as this can take up significant amounts of space within my client's browser memory[5].

Therefore, server-side rendering through Flask and Jinja templates are used alongside express codes for HTTP requests (GET, POST) to validate my client's inputs. The switch from JS to Jinja templates contributes greatly to the session aspect of my client's success criteria because the client may now traverse across each webpage on a local network rather than running each individual HTML file.

<u>The extent of using Python for backend development</u>

To justify this remark, the overarching goal is to utilise a development server that would run continuously and this was achieved through the Flask REST API. To be specific, this particular server responds to end-user requests with the association of buttons on each webpage that redirect to the following web pages. Thus, achieving the vast majority of the success criteria that involve buttons.

Figure 2: Command Line Interface of a development server responding to end-user requests



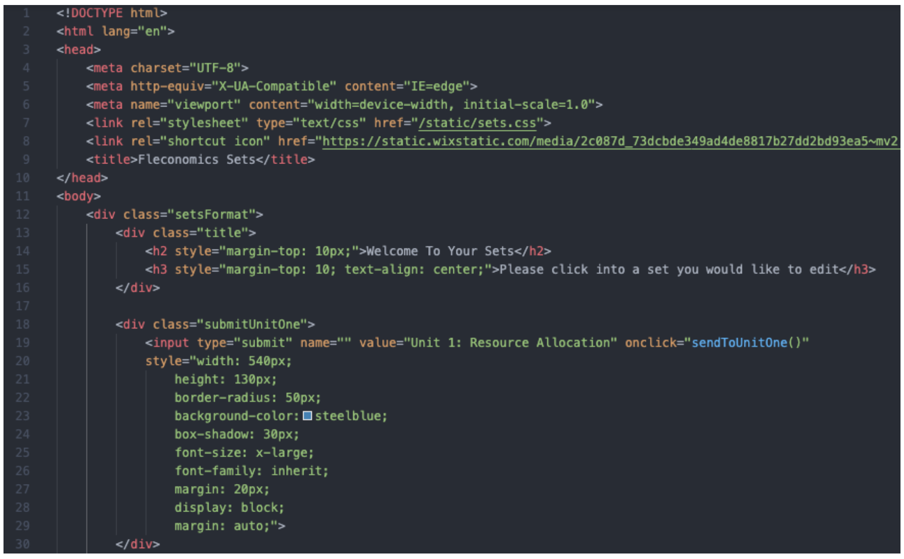
```
❯ flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [31/Jan/2022 10:24:48] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:49] "GET /static/login.css HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:49] "GET /static/econSets_bg.webp HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:51] "POST /form_login HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:51] "GET /static/login.css HTTP/1.1" 304 -
127.0.0.1 - - [31/Jan/2022 10:24:51] "GET /static/econSets_bg.webp HTTP/1.1" 304 -
127.0.0.1 - - [31/Jan/2022 10:24:52] "GET /static/DIY%20Flash%20Cards.png HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:56] "POST /form_login HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:56] "GET /static/options.css HTTP/1.1" 200 -
127.0.0.1 - - [31/Jan/2022 10:24:56] "GET /static/dollarbills.jpeg HTTP/1.1" 200 -
127.0.0.1 - - [01/Feb/2022 06:42:39] "GET / HTTP/1.1" 200 -
```

[5] *What are the downsides of Python's Flask?, Quora*

<u>Remarks on Ingenuity with Style Sheet and Markup Language</u>

The following code outlines the styling schemes that have been associated with the buttons on the Sets Page which would eventually redirect the client to the various flashcard sets.

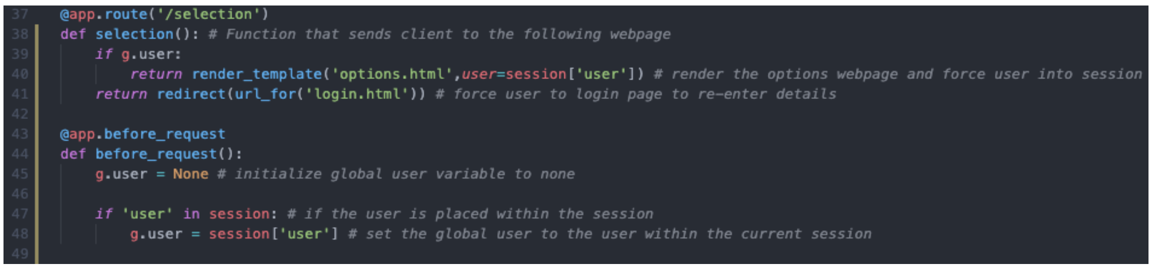
Source Code Extract 5: CSS Implementations to 'Sets Page' Buttons



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="/static/sets.css">
    <link rel="shortcut icon" href="https://static.wixstatic.com/media/2c087d_73dcbde349ad4de8817b27dd2bd93ea5~mv2
    <title>Fleconomics Sets</title>
</head>
<body>
    <div class="setsFormat">
        <div class="title">
            <h2 style="margin-top: 10px;">Welcome To Your Sets</h2>
            <h3 style="margin-top: 10; text-align: center;">Please click into a set you would like to edit</h3>
        </div>

        <div class="submitUnitOne">
            <input type="submit" name="" value="Unit 1: Resource Allocation" onclick="sendToUnitOne()"
            style="width: 540px;
                height: 130px;
                border-radius: 50px;
                background-color: steelblue;
                box-shadow: 30px;
                font-size: x-large;
                font-family: inherit;
                margin: 20px;
                display: block;
                margin: auto;">
        </div>
```

As shown in *Source Code Extract 5*, the code has been well-formatted. This is displayed through the indentation of containers that hold together the styling of the webpage. The use of a container brings efficiency to my code as the division tags of the various buttons that are displayed along the sets page are formatted with minimal lines of code. This helps to ensure that programmers are not confused with the attributes of styling that I have made with the page itself and the buttons. Therefore, the design of background, font families, and extra colour pallets have been expanded on a separate CSS file.

Source Code Extract 6: Implementation of sessions through Jinja templates



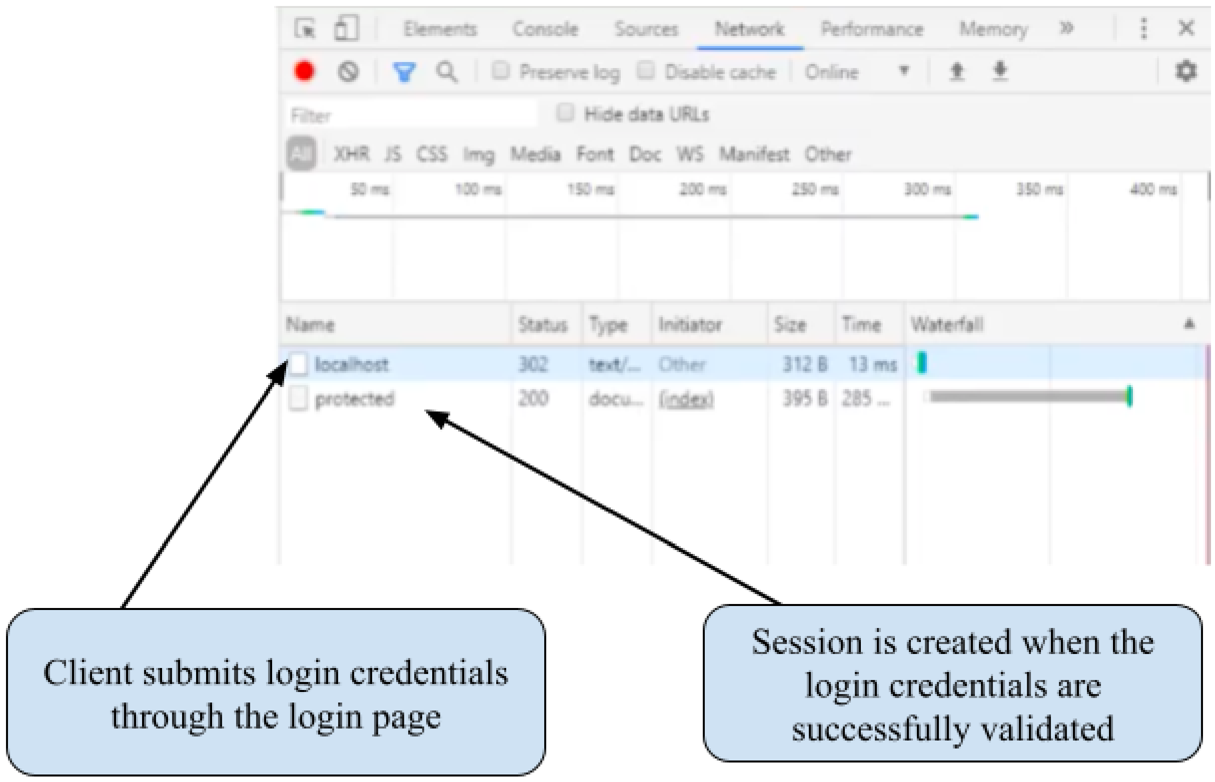
```
@app.route('/selection')
def selection(): # Function that sends client to the following webpage
    if g.user:
        return render_template('options.html',user=session['user']) # render the options webpage and force user into session
    return redirect(url_for('login.html')) # force user to login page to re-enter details

@app.before_request
def before_request():
    g.user = None # initialize global user variable to none

    if 'user' in session: # if the user is placed within the session
        g.user = session['user'] # set the global user to the user within the current session

```

The use of middleware by implementing the 'session' class has been involved to protect my client from privacy violation[6]. This is justifiable for use in my client's web application and success criteria as the chaining of routes allows for compartmentalisation to the code as well as encourages

[6] *Python Flask Login System With Sessions. Youtube*

interoperability[7], which has been useful to redirect users not in sessions back to the login screen to prevent accessing my client's data.

Figure 3: Inspecting the network of the flask server when the session is created



Name	Status	Type	Initiator	Size	Time
localhost	302	text/...	Other	312 B	13 ms
protected	200	docu...	(index)	395 B	285 ...

Further examples of this justification can be found in the appendix[8].

[7] *Security middleware approaches and issues for ubiquitous applications, Camwa*
[8] *Appendix, source code location: app.py*

Evaluation of the Product

Success Criterion	Result
Ability to authenticate the client into the web application, preferably through a login screen.	Completed - Client is able to login to the web application through an authentication page. According to the client, the interface was functional and followed the UI designs that had been developed during the designing stage[9].
Function that creates a session once the client's credentials have been entered correctly into an authentication page.	Completed - Once the client submits their login credentials a session is created on their device.
Ability to redirect the client to a following page once the session is created.	Completed - Once the session is created on the client's device, they are redirected to the landing page.
Clicking on options of being redirected to sets or planning page works.	Completed - The client is successfully able to select the options for either editing their flashcard sets or their to do list.
Ability to refer my client to different economics flashcards sets of their choosing.	Completed - A designated page has been created for my client for this success criterion. This webpage contains various buttons that redirect to all of the flashcard sets that they may edit.
Ability to redirect my client to previous pages visited in the web application.	Completed - There are various return buttons placed on each webpage of the web application. This has helped to interconnect all of the different web pages that make up the program for my client.
Function to add a flashcard to a chosen set.	Completed - A popup window has been made for the set pages so that my client can add in their flashcards.
Function to delete all cards in a chosen set.	Completed - Delete buttons have been added to each set.
Function that allows the client to print all cards.	Completed - Print buttons have been embedded to sets so that my client may print out their cards.
The client has the ability to enter to-do list items into a form which creates a to-do item.	Completed - To-do list terms may be entered into the text field and this will create the item.
Created to-do list items may be filtered as completed, incompleted or even removed.	Completed - The items that my client makes have the option of being flagged as completed, incompleted, or even removed from the list of all the other to-do items.

[9] Appendix C: Final interaction with the client

Recommendations for Further Improvements

1) Method to delete various flashcards:

The first change which the client had recommended to be altered was the 'delete' button on each of the sets. This is because all of the elements within the set would be removed from the PyCache files once pressed. The client stated that they would much prefer an implementation where the flashcard elements could be removed individually in the form of a LinkedList rather than an ArrayList.

2) Ability to create new sets:

The client had suggested the further implementation of a function that would allow them to delete various flashcards instead of the entire set being wiped. The functionality of my client being able to create new sets would make the cancellation process of faulty cards way easier for him.

3) Token Based Authentication:

Through my use of session-based authentication, I have realised that cookies of my client have the potential of being exposed from man-in-the-middle attacks, therefore allowing these attackers to intercept session ID that my client is associated with. This is important towards the security of the web application as these attackers can submit DDoS requests to the server as it could significantly slow down the connectivity/transmission speed of the server, making it unable for my client to access thier flashcards.

The other method for consideration towards further development is token-based authentication. This is relevant towards security as with the involvement of a secret key, the server can validate the signature and further authenticate any of the client's requests. Moreover, the use of tokens eliminates that number of times that my client would need to log into the web application, ensuring greater convenience for my client when they use the tool that I have prepared for him.

4) Heroku Platform as a Service (PaaS):

The use of Heroku for the web application would make the accessibility of my client's program a lot easier. Heroku would allow me to deploy the web program on a cloud service, and this would not require my client to boot up the flask server before accessing the flashcard web application. In addition, launching the web-based application through Heroku provides free SSL encryption. Therefore, implying greater security to the flask application as the likelihood of attackers modifying the contents of my client's information is low.

Al-Jaroodi, J. et al. (2010) "Security middleware approaches and issues for ubiquitous applications," Computers & mathematics with applications (Oxford, England: 1987), 60(2), pp. 187–197. doi: 10.1016/j.camwa.2010.01.009.

css position guide (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/6625282/css-position-guide (Accessed: October 30, 2021).

Error in JSON Serialisation with HTML Content (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/29763979/error-in-json-serialisation-with-html-content (Accessed: October 30, 2021).

How can I use a HTML form to replace presets in a PHP script to submit a ticket to osTicket? (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/27932036/how-can-i-use-a-html-form-to-replace-presets-in-a-php-script-to-submit-a-ticket (Accessed: October 30, 2021).

Is there a secure way to store passwords for a website? (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/64549247/is-there-a-secure-way-to-store-passwords-for-a-website (Accessed: October 30, 2021).*Jinja advantages and disadvantages (no date) Webforefront.com. Available at: https://www.webforefront.com/django/usejinjatemplatesindjango.html (Accessed: February 1, 2022).*

Return information using a submit button in HTML (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/35813561/return-information-using-a-submit-button-in-html (Accessed: October 30, 2021).

Return JSON response from Flask view (no date) *Stackoverflow.com*. Available at: https://stackoverflow.com/questions/13081532/return-json-response-from-flask-view/13089975 (Accessed: October 2021).

Shiksha, C. (2019) Python Flask Login System With Sessions. Youtube. Available at: https://www.youtube.com/watch?v=PrsmxWdthg0 (Accessed: February 1, 2022).

Singhal, G. (no date) Pros and cons of client-side rendering, Pluralsight.com. Available at: https://www.pluralsight.com/guides/pros-and-cons-of-client-side-rendering (Accessed: February 1, 2022).

Top 3 benefits of REST APIs (no date) MuleSoft. Available at: https://www.mulesoft.com/resources/api/top-3-benefits-of-rest-apis (Accessed: February 1, 2022).

What are the advantages and disadvantages of using Flask as a web framework? (no date) Quora. Available at: https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-Flask-as-a-web-framework (Accessed: February 1, 2022).

What are the downsides of Python's Flask? (no date) Quora. Available at: https://www.quora.com/What-are-the-downsides-of-Pythons-Flask (Accessed: February 1, 2022).

Appendix A: First Interaction with the client

Me: May you please state your background?

Cient: I am currently studying Economics in my A-Levels course. Over the past two years, I have been invested in making notes that are well prepared as well as easy to follow. I recently decided to switch my learning methods to written flashcards as they serve the purpose of quick action, on-the-go revision material. But it has come to my attention that I tend to lose my cards on a frequent basis.

Me: What would you like the program to involve?

Client: The application or website, designed whichever preference you would like, must allow me to create the various flashcards that I need for the 5 different units of my Economics syllabus. I would also like to reassure you that I am away from my computer the majority of the weekends as I am out with my family. During this time I also like to review my cards to keep up to date with my knowledge on the topics. As I do not have my computer on me, it would be great that this program is functional across a multitude of devices, such as my phone and tablet. Both are fairly old devices, running on Android OS.

Me: Any additional requirements that the program should involve?

Client: The webpage should involve a simple scheduling tool so that I can mark off any notices or reminders for revision on specific dates. Please ensure that the program is well secured throughout the login page as well as a scheduling tool.

Me: I will take note of all of these things, thank you very much for your information.

Client: Thank you.

Appendix B: Second Interaction with the client

Me : Thank you so much for meeting with me today. I would like to display the website to you in its current state and inform me of your thoughts.

Client : Sure thing.

I provided the client with a thorough understanding of the website that I have created for him

Me: After displaying the updated details, may you please provide me insight as to what you would like to see updated to the website, or any minor changes which you would like me to make before final polishments are made?

Client: Overall, I am very pleased with the current design of the website. The only concern that I have at the moment is the security of the login page. I fear that people still may bypass the login system regardless of knowing the password or not. Asides from this, the password is easily inspectable through viewing the source code. I would assume that this is a measure which you have already taken into consideration and I hope that there are plans for improving the security of the website. I would advise that the same regulations should apply to the other web pages, and if there is a new device detected it sends them to the login page, preventing anonymous and untrustworthy users from overwriting my data.

Me: I completely follow your comments. And yes, I have already taken into consideration the security aspect of the login page. Plans are already in place by using two third-party APIs in order to store the correct credentials in an online server.

Client: Perfect; would like to be updated on the progress of the website.

Me: Sure thing, absolutely no worries.

Client: Thank you so much for this update.

Appendix C: Final interaction with the client

Me: Hi the client! Hope the web application has been working really well for you with the use of the virtual environment that we set up on your computer so that you can use the software.

Client: Yeah it has been fun to use.

Me: Could you please let me know about any successful points that I managed to complete according to the success criteria that both of us discussed in our original meeting?

Client: Yeah sure. Well, it's right off the bat that you've clearly involved a login page which is great, at least it develops some form of security so that I can keep my flashcards safe without the consideration of other people gaining access to my work. One thing I did notice though was that the sessions were not starting after my username and password were validated by the program. I think that this is something which you could really look into fixing as it would drastically reduce the chances of potential threats or people wanting to bypass the login system. You said that you were using render templates if I'm not mistaken? I have a slight feeling that someone could simply redirect themselves through the web application with the use of these templates and it would just diminish the use of the login system. So I'd say the top priority right now would be to debug that session implementation. Asides that the rest of the application looked just fine. Asides from following quite a cool-looking colour scheme which in fact really does help me want to use it more, the buttons and flow of moving between every other webpage are pretty sleek.

Me: And what is your opinion on both the flashcard sets of making your cards as well as the to-do list planner?

Client: Creating my cards is really easy, I mean I never really clear out the cache files of my computer anyway so it doesn't bug me at all. Though I would feel a little more comfortable if my cards were stored in a database. Though I do understand that you did already come across a major issue of trying to integrate SQLAlchemy as the database management system. Nonetheless, the options you've given me work really well. If I could suggest one more thing if you do not mind, I would like to see an option to delete individual cards, as this would make the cancellation process of faulty cards way easier for me.

You were also asking about the to-do list? Yeah, I really enjoyed using that. Really aesthetic and gets the exact job done of what I really needed; just a list where I can manage the cards which I would need to make in the future. Functionality-wise, all works really well. The sorting between completed and incomplete tasks is really great. All in all, I’m thoroughly satisfied with the web application that you have made for me. Over the past month, it has significantly helped me revise for my Econ classes.

Me: Makes complete sense. I’ve taken into consideration your points and I have a strong feeling that I may be able to make these final changes before uploading all files to Heroku, that way you can just access the web application just as a normal link on the internet. And that’s so great to hear that it has been working well for you. I’ll keep you updated once I make the final changes you had recommended.

Client: Thank you so much!

Me: No worries.