Luca Alvisini

Book for thought

Criterion B - Solution Overview

Word count: xxx

Criterion B

When entering the website without having previously logged in or having your session data cleared you will be greeted by a login portal with the following options for logging in and signing up:

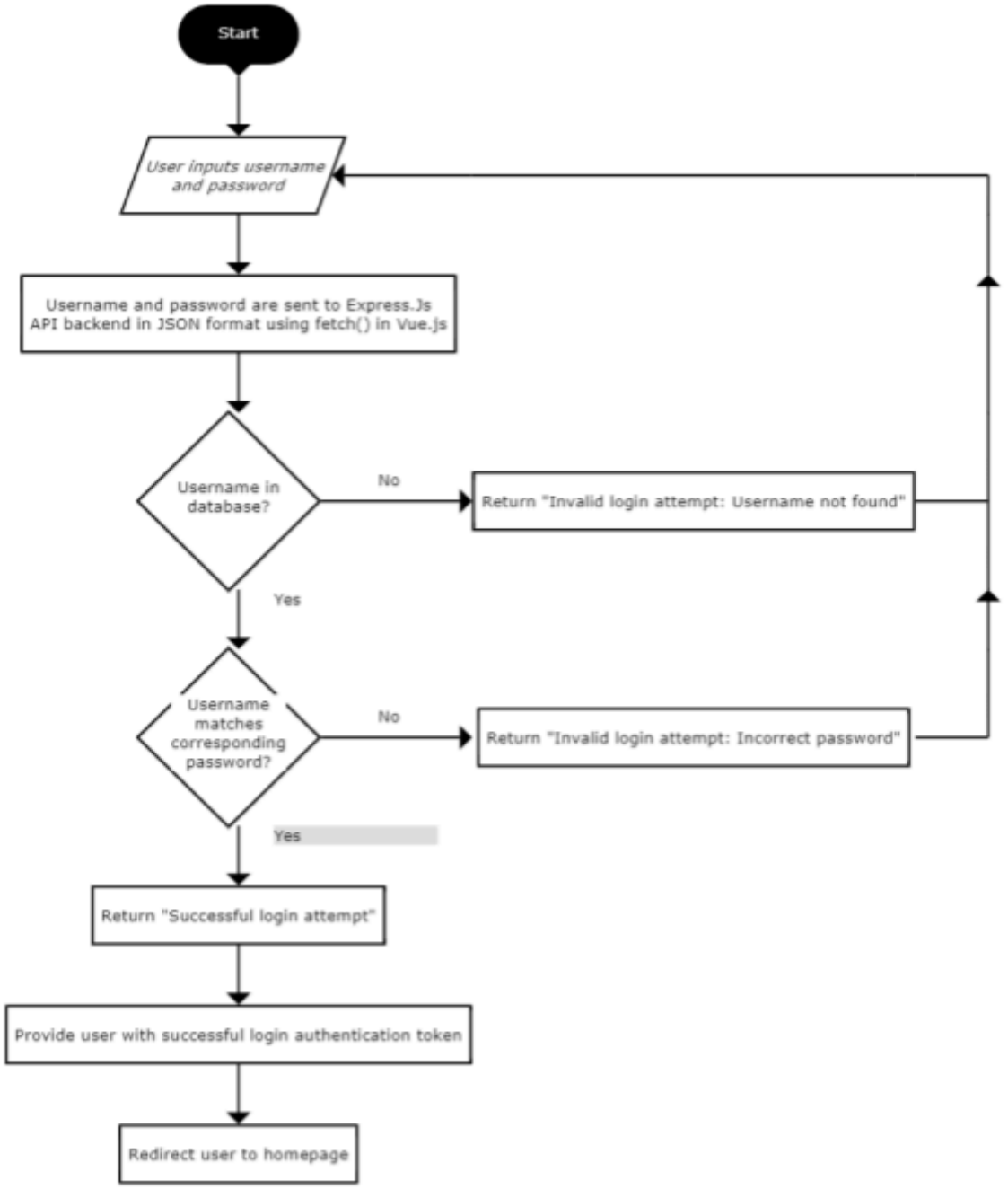


Figure 1. Flowchart outlining basic login functionality using Express.Js backend at /login endpoint

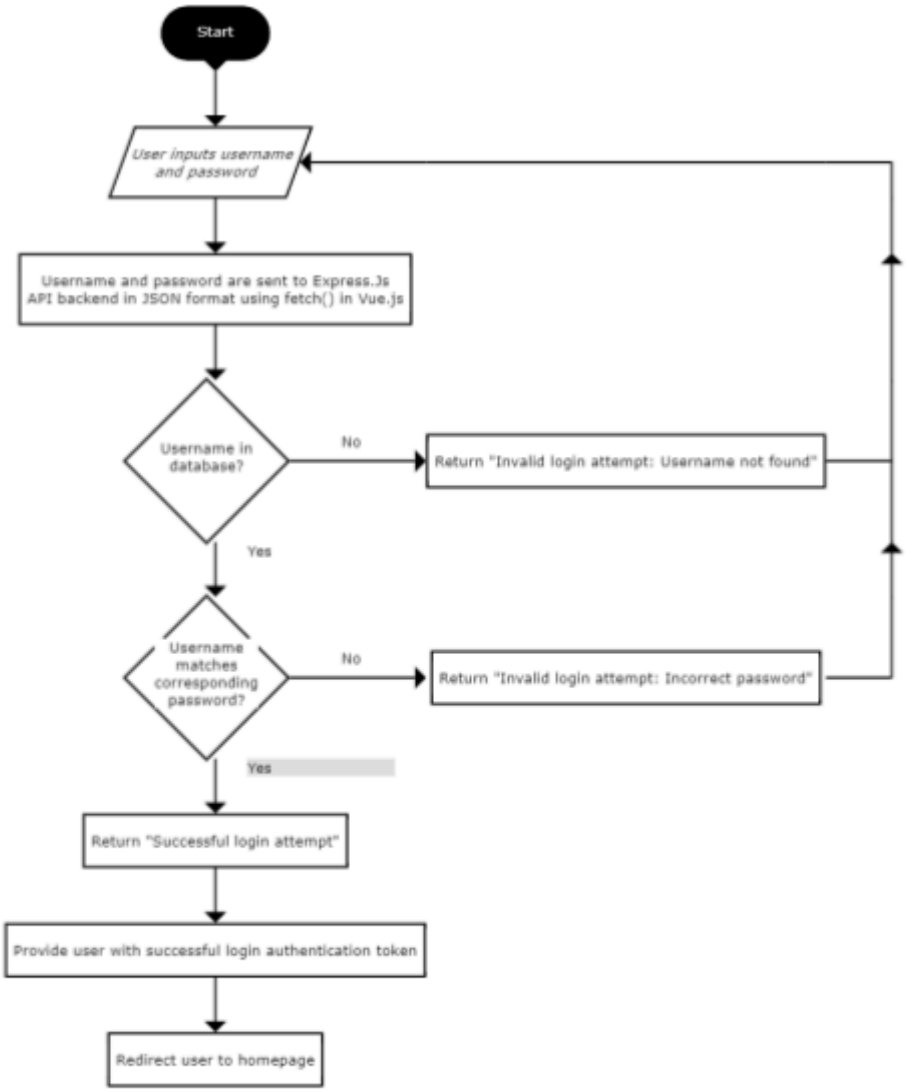


Figure 2. Flowchart outlining basic signup functionality using Express.Js backend at /signup endpoint

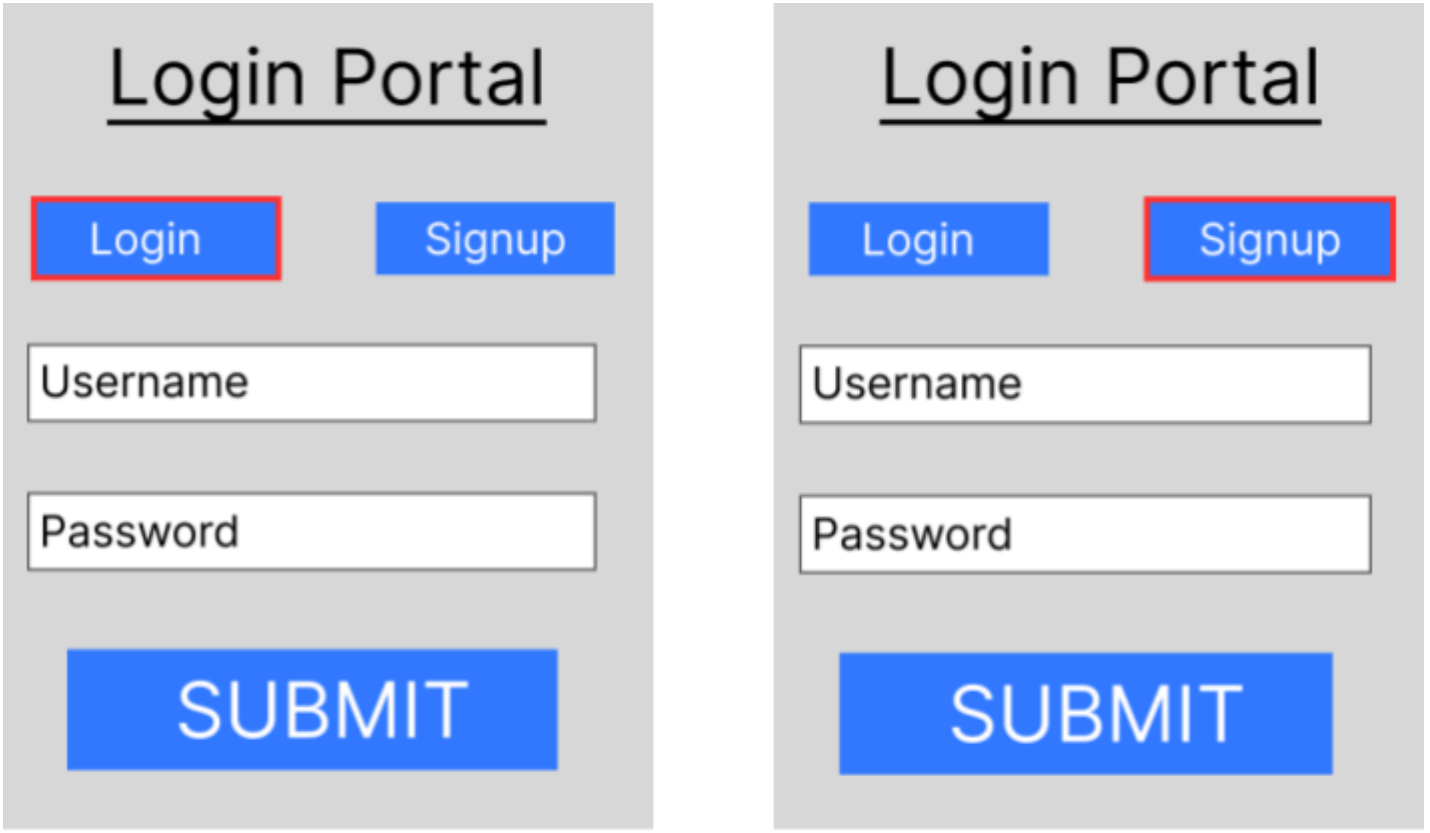


Figure 3. Prototypes for login/signup prompt

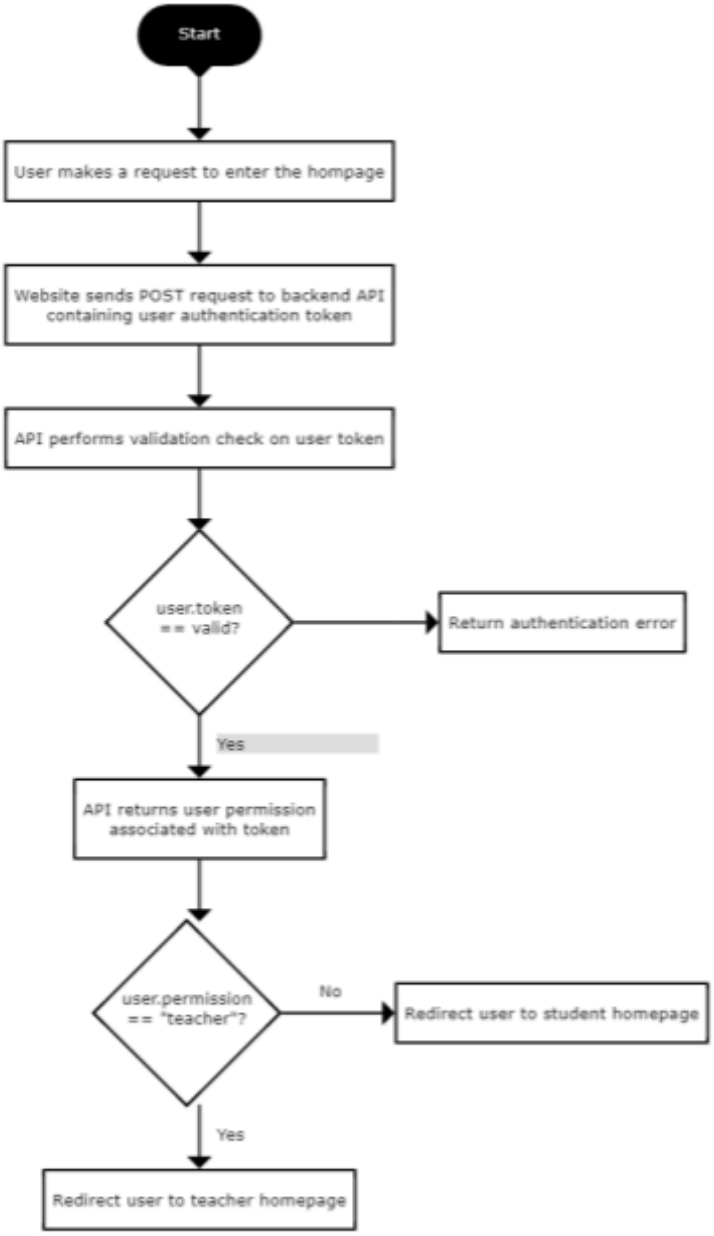


Figure 4. Flowchart detailing homepage redirection process upon signing in or opening the website when already signed in

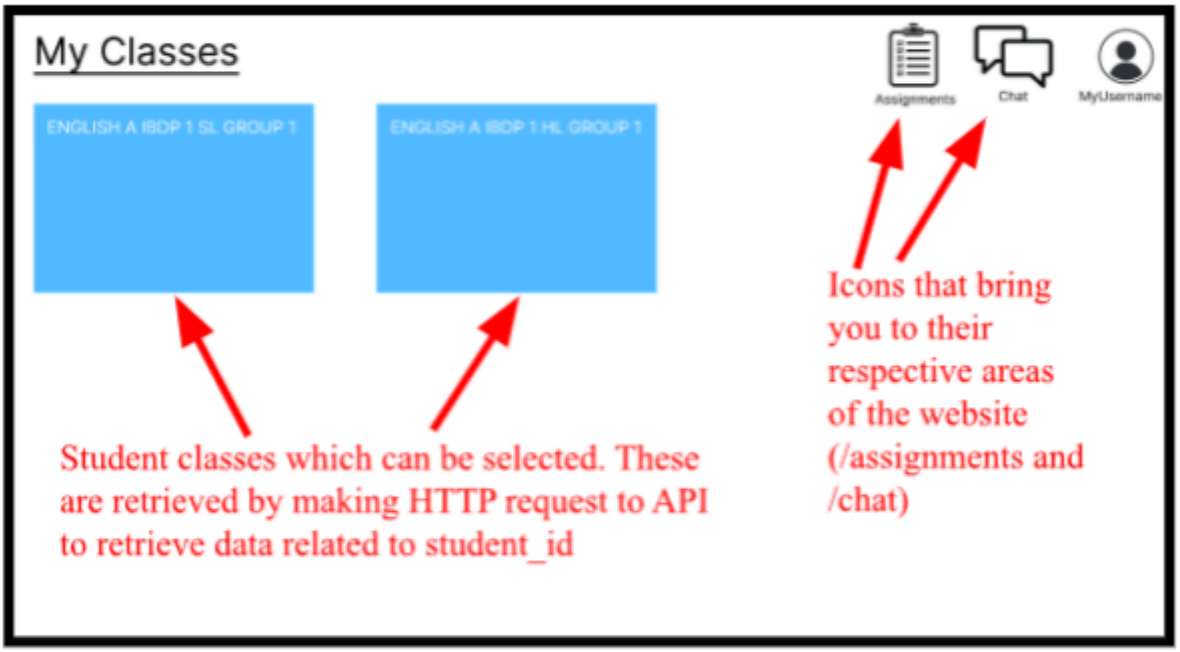


Figure 5. Prototype for student homepage UI

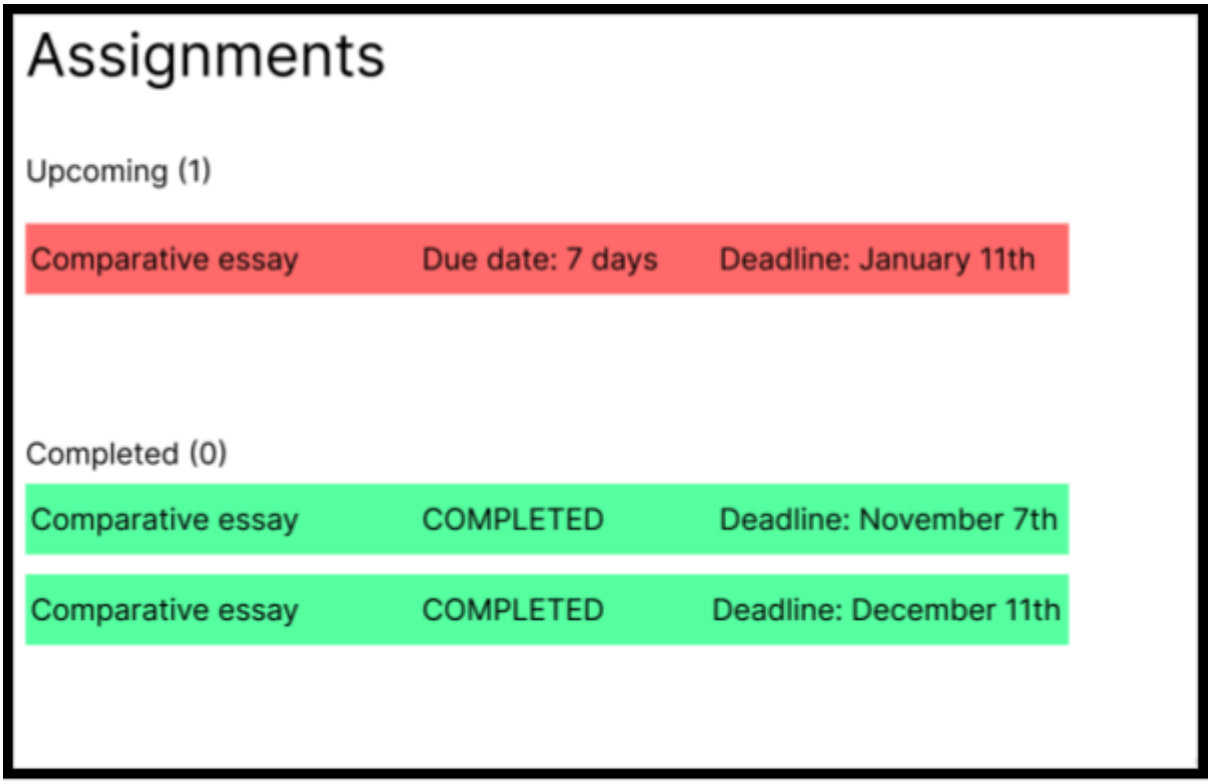


Figure 6. a

These assignments can be individually selected in order to see the description of the task and any resources that have been attached to them. It is also where students will be able to submit their work once they have completed the assignment. Below is a prototype displaying an example for an assignment.

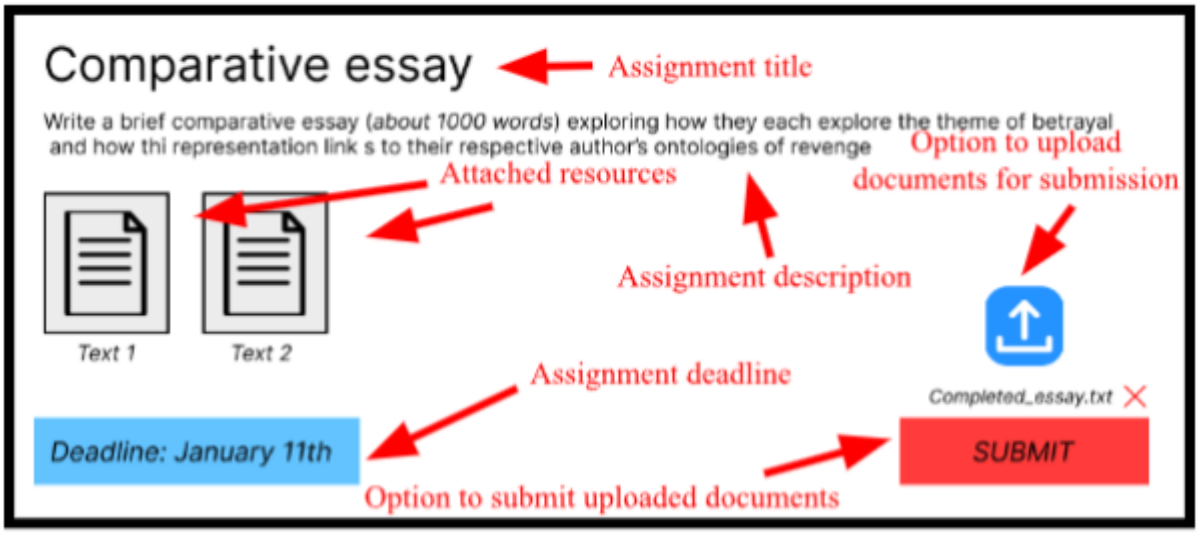


Figure 7. Prototype for the assignment page from the student view

If instead the student user selected one of the classes from the homepage they would be redirected to a page represented by the following module.

Figure 8. Prototype for classroom from student user view

Both student and teacher users have access to a chat where they can send messages to other users and upload documents through the chat which the recipient can download

Figure 9. Prototype for application chat feature

Teachers have a similar homepage to users however upon selecting the class and assignment icons from the UI they are brought to different pages

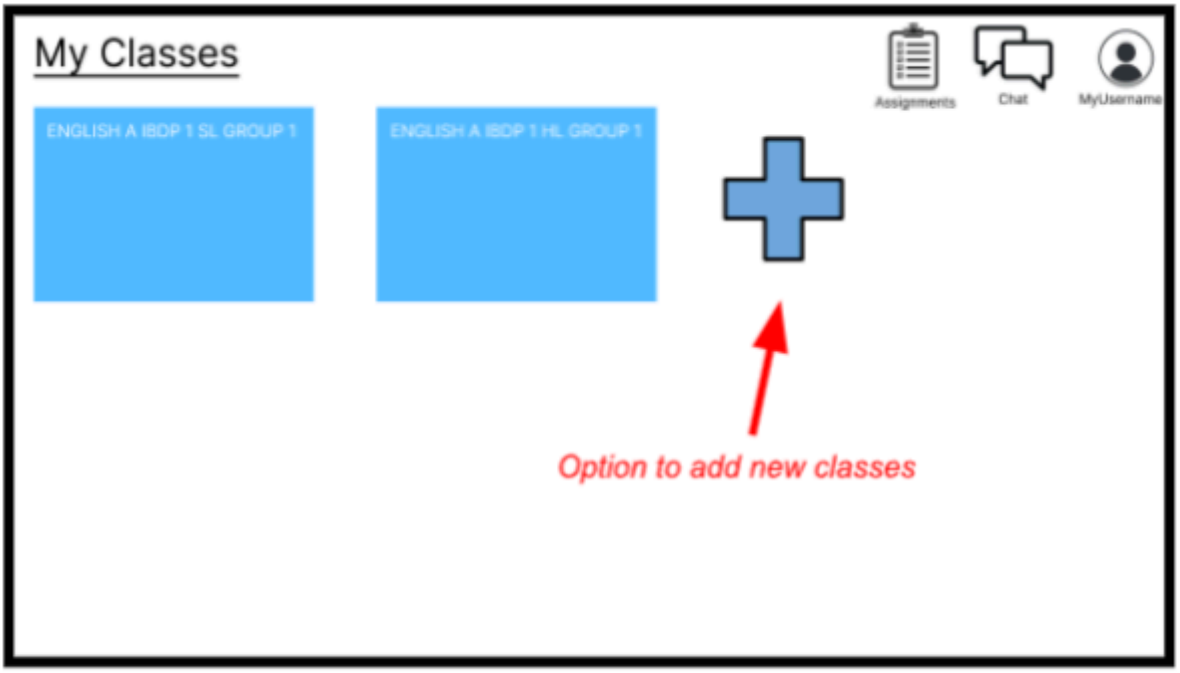


Figure 10. Prototype for UI of teacher homepage

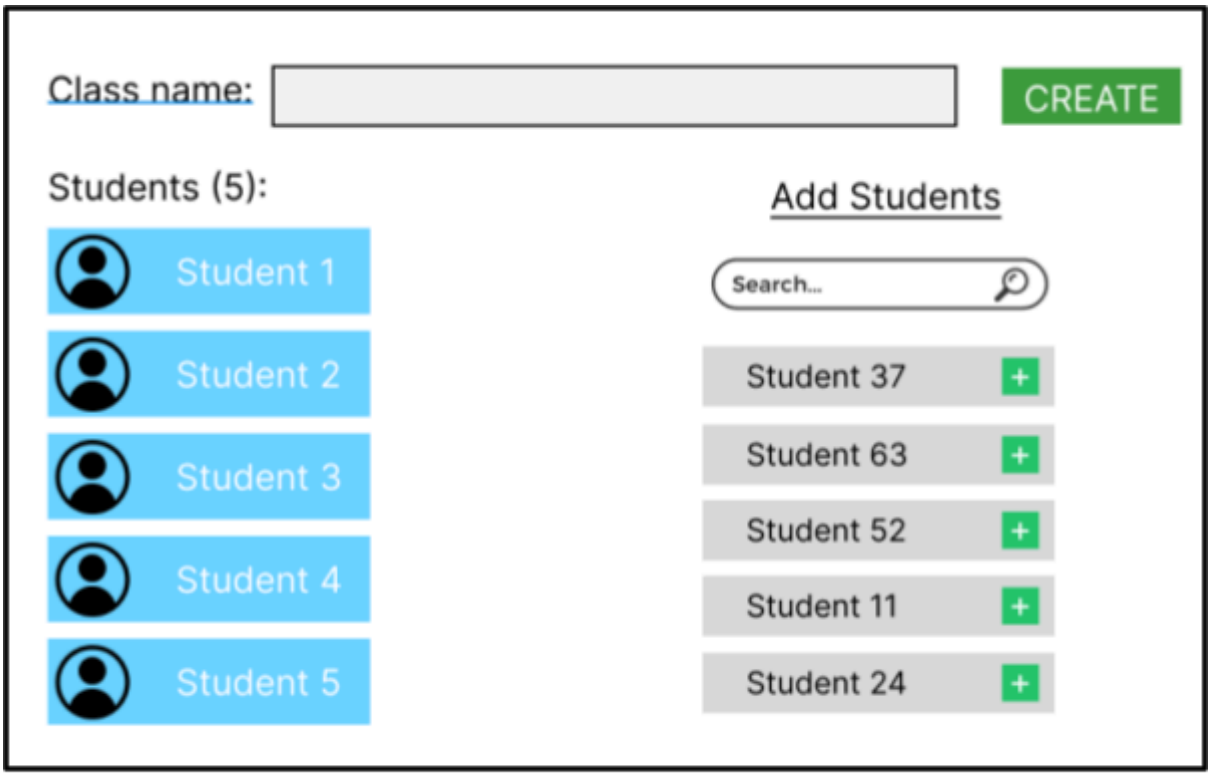


Figure 11. Prototype for class creation from teacher user side

The class view for teacher users will be similar to that of students with the only difference being the addition of several UI features that allow the teacher to add announcements and assignments that the students can see from their view. In the teacher view there is also the option to view assignments for an individual class.

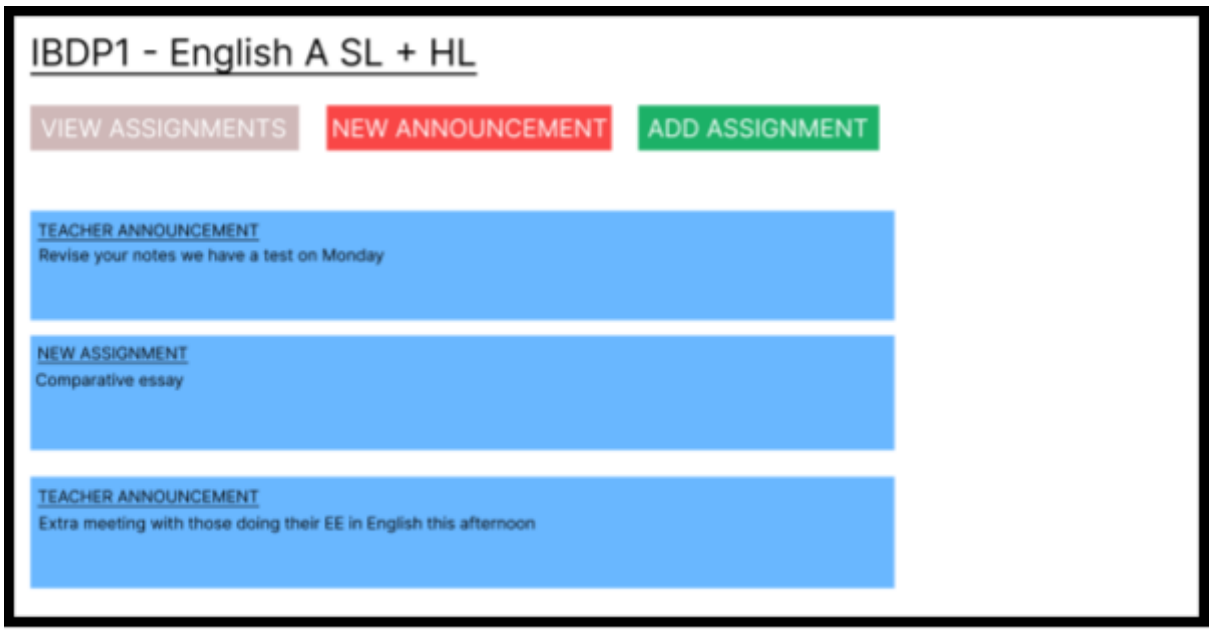


Figure 12. a

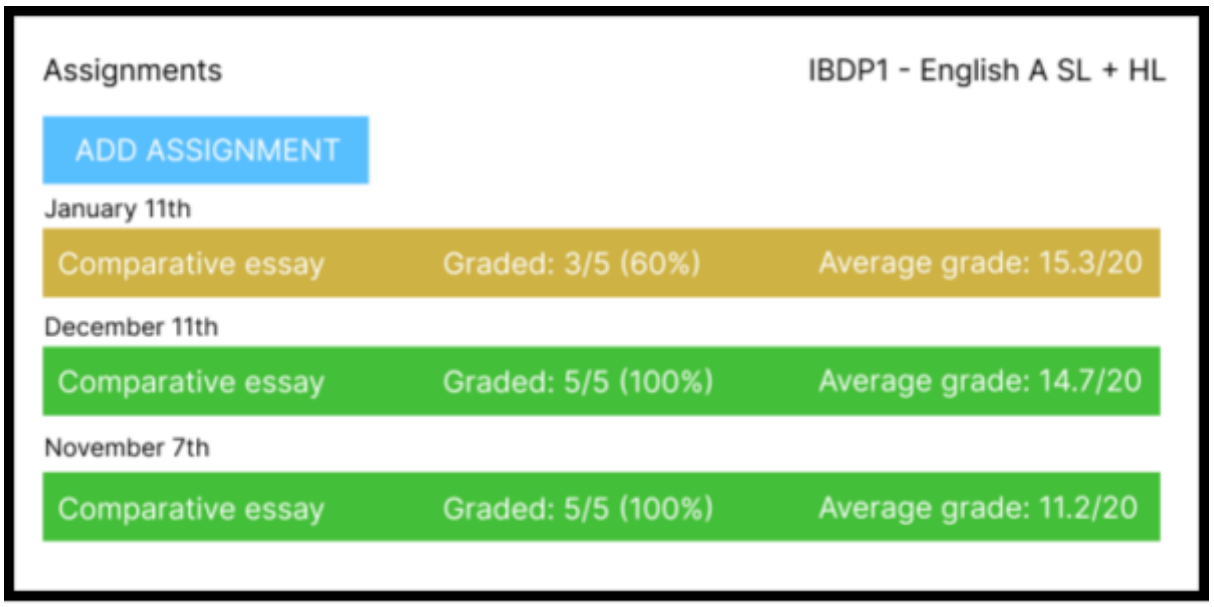


Figure 13. GUI that appears when teacher selects "VIEW ASSIGNMENTS"

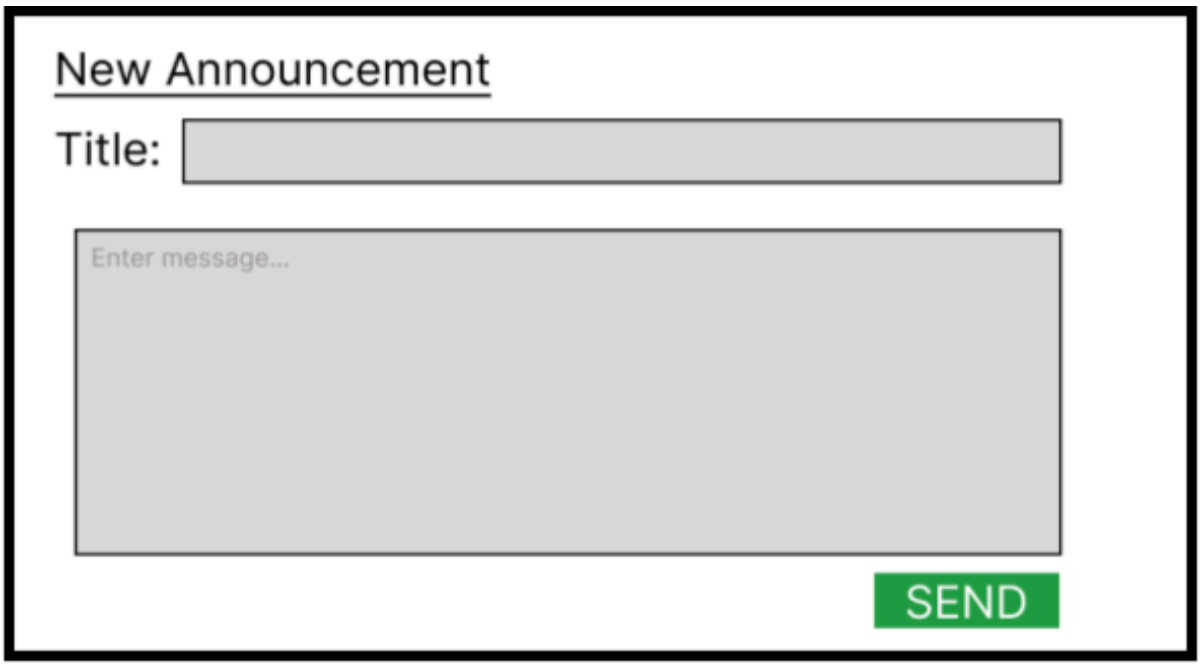


Figure 14. GUI that appears when teacher selects "NEW ANNOUNCEMENT"

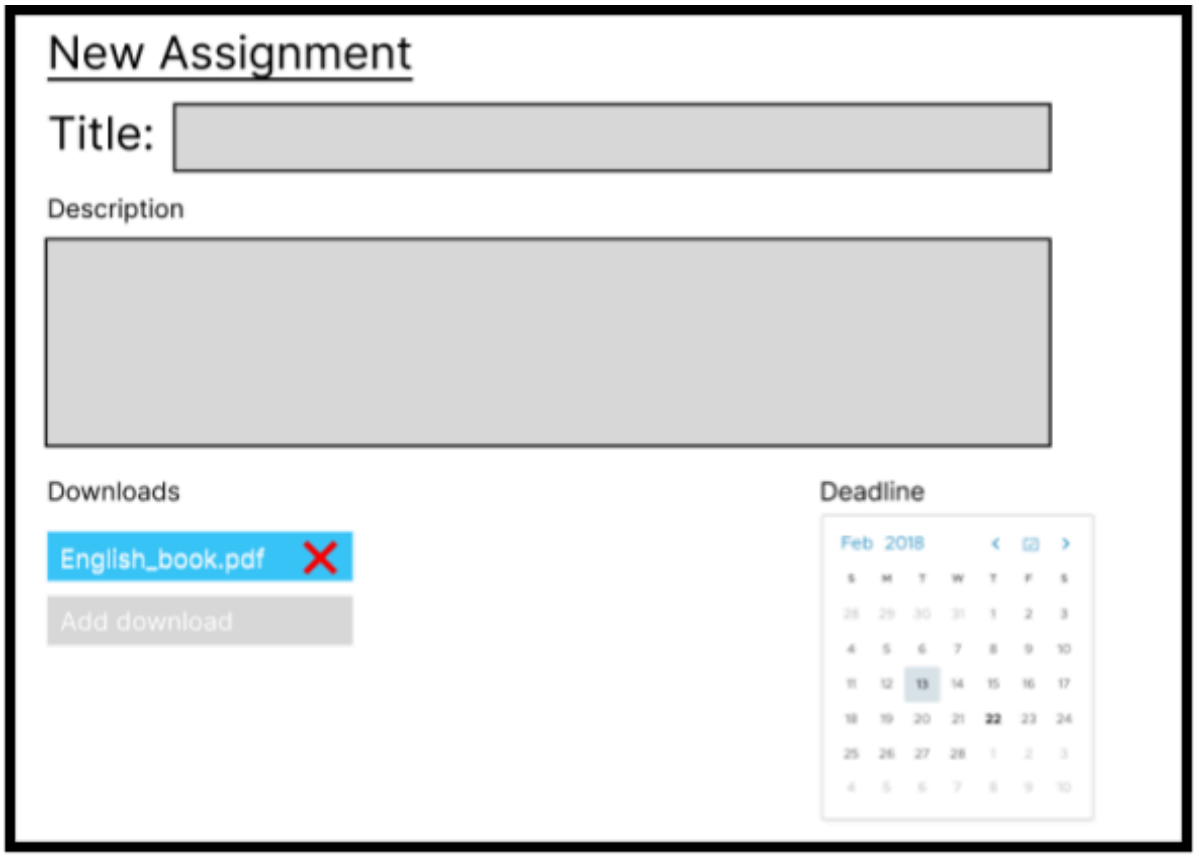


Figure 15. GUI that appears when teacher selects "NEW ASSIGNMENT"

This was the first prototype presented to the client in the form of a Google Slides presentation

Insert text talking about how this was the first prototype presented to the client but after client interview seen in Appendix … made the following adjustments

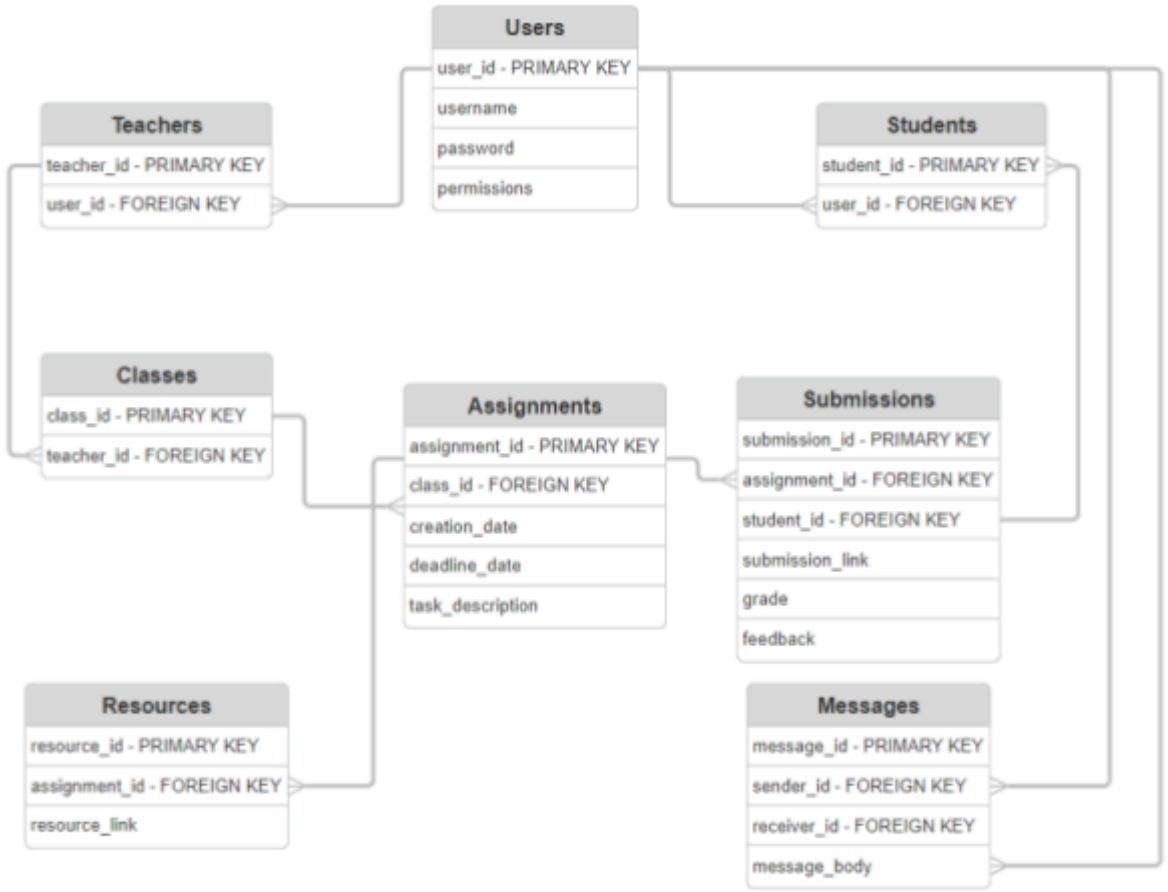


Figure x. Relationship diagram of *"database_name"* database used in project_name

"Users" table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
user_id	INT	4	**PRIMARY KEY**	0
username	VARCHAR	31	Account username	JohnSmith123
password	VARCHAR	31	Password used to sign in to account	StrXngP03wrd
permissions	ENUM	1	User permissions which dictate which pages they can see / base the UI	"teacher"

“Teachers” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
teacher_id	INT	4	**PRIMARY KEY**	0
user_id	INT	4	**FOREIGN KEY →** “Users”	0

“Students” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
student_id	INT	4	**PRIMARY KEY**	0
user_id	INT	4	**FOREIGN KEY →** “Users”	0

“Classes” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
class_id	INT	4	**PRIMARY KEY**	0
teacher_id	INT	4	**FOREIGN KEY →** “Teachers”	0
class_name				
students				
colour				

“Assignments” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
assignment_id	INT	4	**PRIMARY KEY**	0
class_id	INT	4	**FOREIGN KEY →** “Classes”	0
creation_date	DATE	3	Time	2024-01-25

			descriptor for the creation of the task	
deadline_date	DATETIME	6	Deadline for the task, includes time	2024-01-25 08:30:00
task_description	VARCHAR	511	A brief description outlining the task the teacher wants the student to perform	“Write a comparative essay on the following two bodies of literature”

“Submissions” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
submission_id	INT	4	**PRIMARY KEY**	0
assignment_id	INT	4	**FOREIGN KEY →** “Assignments”	0
student_id	INT	4	**FOREIGN KEY →** “Students”	0
submission_link	VARCHAR	255	Link that can be used to access the submission of the student	http://hereismysubmission/usfVa-uv8tf
grade	TINYINT	1	Student’s grade from 1-20	15
feedback	VARCHAR	1023	Teacher and AI generated feedback for the student	“Covers many points but does not use evidence in the text to accurately develop arguments”

“Resources” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
resource_id	INT	4	**PRIMARY KEY**	0
assignment_id	INT	4	**FOREIGN KEY →** “Assignments”	0
resource_link	VARCHAR	255	Link relating to resource attached to assignment	http://shakespearebooks/book-753

“Messages” table data dictionary

Field name	Data type	Field size (bytes)	Description	Example
message_id	INT	4	**PRIMARY KEY**	0
sender_id	INT	4	**FOREIGN KEY →** “Users”	0
receiver_id	INT	4	**FOREIGN KEY →** “Users”	0
message_body	VARCHAR	255	Content related to any specific message	“Hey, can you send me revision notes for Paper 1?”

Test plan

Success criteria	Type of test	Test	Expected outcome
2	Unit test	Test that API call is sent when user attempts to create account	Request received message from the API console
2	Integration testing	Test that account is successfully created when using signup page	Record is added to Users table and user is able to login using their credentials

3	Integration testing	Test that upon logging in user is provided with correct token outlining user data	Users log in and all their data is present within the web page (classes, chats, etc.)
4.	Unit testing	Test if webpage is able to access the priority value (student or teacher) for a given user	Webpage returns the right user interface upon login
5	Unit testing	Test that chat button opens new page containing chat interfacing	Upon clicking the icon the user is able to see all their chats and can create new ones from the page
6	Unit testing	Test if search feature allows user to select another user from the pool of users	Searching for a student account will bring up that student
6	Unit testing	Test that user sending a message send that message to the right API path	API console will return a successful message containing the body of the message sent
6	Integration testing	Test that user is able to communicate with other users through the use of the chat feature	Recipient receives messages in real time and is able to respond. All chats are stored meaning after closing they should be able to be reloaded
7	Unit testing	Test that a message sent will store it in the SQL database	Send a message from the user account and check that it is logged in the Messages table with the appropriate sender_id
9	Unit testing	Test if chat sent by user is stored in the cache	Able to successfully access the chat from the chat with the appropriate chat ID

10	Unit testing	Test that user is able to send a file	Message appears on the user end with the file which the user is able to open
10	Integration testing	Test that recipient is able to access the sent attachment	Recipient is able to download the sent file and open it on their own device
11	Unit testing	Test functionality of back button in the user UI	Pressing back button successfully returns user to homepage
12	Integration testing	Test correct classes appear on the homepage for student users	Test that users are only able to see the classes associated with them in the Classes table
13	Unit testing	Test that teacher adding a new class stores it in the database	Class and all class details are saved as a new record in the Classes table
15	Unit testing	Test functionality of teacher importing a list of students into a class	Each of the students is successfully added in the database
16	Integration testing	Test that teacher accounts are able to see all of their classes	Homepage for teacher users displays each of the classes they are associated with from the Classes table
17	Unit testing	Test that teachers are successfully able to create a new assignment	New assignment record with appropriate details is added to the Assignments table
17	Integration testing	Test that assignments work for student users	Students associated with the class where the assignment was created are able to see the assignment along with details such as deadline and supporting

			resources. Students are also able to submit their piece of work
18	Unit testing	Test that attached files are successfully stored	Attached files are stored in SQL database under appropriate record in Assignments table
19	Integration testing	Test that assignment resources are available for student users	Students are able to download resources and view them on their local machine
20	Integration testing	Test that users opening assignment will provide them with appropriate task uploaded by teacher	Description associated with assignment and other data present in the Assignment record is available to the student user
22	Unit testing	Test that users can upload their work	Uploading a submission to an assignment appropriately stores the submission in the Assignment record
23	Unit testing	Test that API can return who has completed the assignment and who has not	API returns list of students who have completed the assignment and a list of students who have yet to compelte it
23	Integration testing	Test that the user submissions is visible to the teacher	Assignment successfully shows the percentage of students who have submitted the work and highlights the students who have yet to complete it
24	Integration testing	Test that teacher can	Teacher is able to

		download student submission	open student submissions in a window on their local machine upon selecting the option on the UI
25	Unit testing	Test that point score sent to correct API path will accurately store it within the database	API request will result in update in the Assignments table for student score
25	Integration testing	Test that teacher is able to open the appropriate rubric for the task	Depending on information in Assignment record the UI associated with marking window contains the correct rubric which the teacher can reference to mark the student work
25	Integration testing	Test that teacher is able to grade the student	When teacher submits grade it is sent to appropriate API path and stored in the Assignment table of the database
27			
28	Unit testing	Test that correct feedback and document is returned to student user	Student receives their submitted document along with a point score and a page at the end of the document that contains written feedback
28	Integration testing	Test that student is able to access feedback	Grade for the task is visible from the student assignment UI and students are able to locally access corrected document

29			
30			
31			
32			
33			

Record of tasks

Task number	Planned action	Planned outcome	Time estimated	Target completion date	Criterion
1	Meet the client and try to establish what the problem they face is	I have a bullet point list detailing nuances within the client's problem and can use this to potentially develop solutions			
2	Suggest solutions to the client that potentially address the problem	Client comments on presented solutions and based on their feedback I am able to tweak details to make the solution fully geared to the client			
3	Install phpmyadmin to allow database creation and manipulation	I have a SQL database on my system that I can locally manipulate and setup without tediously using queries			
4	Outline all data that needs to be stored within the database	I understand what data has to be stored			
5	Create a prototype for the database containing all the required tables	The created database is functional but does not comply with normalisation			

		protocols			
6	Apply the normalisation rules for first normal form to the database in order to break down until the database complies with first normal form	The database ends up being the most storage efficient while maintaining high transmission speeds			
7	Create final prototype in phpmyadmin	This will be the database that I will be able to use for the final product			
8	Create SQL relationship diagram for the structure of the database	The graphical representation can then be included in Criterion B			
9	Create a data dictionaries for each of the tables in the database outlining the appropriate Primary and Foreign keys as outlined in the relationship diagram	This will be able to be included in Criterion B to give insight into the structure of the database			
10	Research potential architecture for the creation of the application	I find resources that are suited for the project and do not require too much learning time			
11	Outline the advantages and disadvantages of each software infrastructure	I will end up with an extensive pros and cons list which will aid in decision making			
12	According to the pros and cons of each framework choose what to use for the project	I end up with a selection of frameworks optimal for the project			
13	Provide the rationale for choosing each piece of software architecture in Criterion A	Using the pros and cons list I will produce a detailed explanation arguing the use of			

		each piece of architecture selected			
14	Setup a github repository in order to allow development across different machines	The repository will allow me to program whenever no matter what machine I am able to access			
15	Install Vue.js	Installing Vue will allow for the creation of the frontend			
16	Create a HTML frontend for login page	User is greeted with a login page used to access the website			
17	Include inputs for username and password and a submit button for login	Allows user to submit their login credentials			
18	Install Node.js	Installing Node.js will allow for creation of the Express.js API			
19	Create api file utilising node.js	API file will be included in github repository			
20	Install appropriate libraries (Express.js, MySQL, BodyParser)	I can begin the development of the API with the appropriate libraries			
21	Create API endpoint at port 3000	Requests can be sent to the API with an appropriate message to mark successful requests			
22	Create API path at /login that requires a username and password in the HTTP request	API will be able to handle the data submitted using the body present in the request			

23	/login path selects record with appropriate username and cross checks it with the password	Checks if user has inputted the right password			
24	/login path returns whether the login attempt was successful	Will be able to determine whether login attempt should be accepted			
25	Add HTML window allowing users to signup, with the same infrastructure as the login page (user inputs username and password)	Will allow for the creation of new users that can then access their account through logging in			
26	Create API path at /signup that requires a username and password in the HTTP request	API will handle data within the body to determine whether a new account should be created			
27	API at /signup path checks for the existence of record with inputted username	Determines whether another user with the same username already exists			
28	If no record exists with the inputted username API creates new user record in Users table	Successfully manages to create a new user			
29	Successful login/signup will redirect user to the homepage	User will then be able to access the website after logging in or signing up			
30	Successful login/signup returns authentication token to the user containing user data	User has authentication token meaning they won't have to login every time and their data is stored across windows			
31	Learn the nuances of	With the			

	Vue.js required to create dynamic webpages using javascript	newfound knowledge development of the program will be seamless as less time will be dedicated to studying documentation			
32	Use figma to create designs for UI and prototypes which can be shown to the client and included in Criterion B	Will give me an idea of the structure of the program and will have a design prototype which I can present to my client			
33	Homepage shows different UI based on user's permissions (teacher or student UI), as retrieved by their authentication token	Experience will be catered to users depending on whether they are a student or a teacher			
34	Chat icon on homepage brings users to specific chat window	Users will be able to communicate with other users from the chat window			
35	Users are able to send and receive messages using the chat	Users are able to communicate in real time with each other			
36	Users are able to send and receive files using the chat	Users will be able to view files sent to them			
37	Chats are stored in the SQL database meaning chats between users are saved	Users are able to close the application without fear of losing chat history			
38	Recent chats are stored in a Redis cache to lower loading times and store messages after crashes	Loading times on phones will be slower and the effects of unexpected crashes will be			

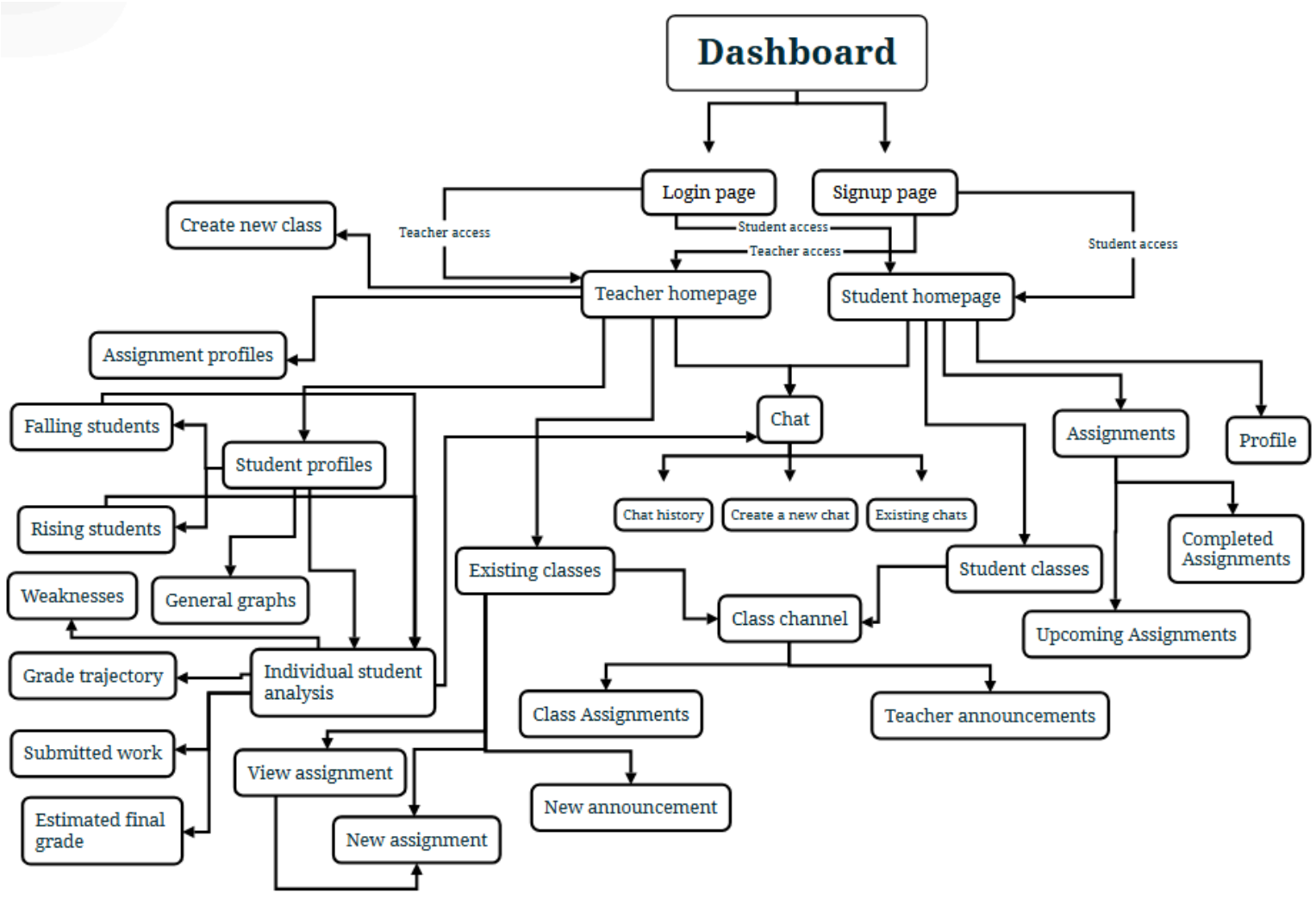
		mitigated			
39	Make students able to click on classes from the homepage and open a separate window	Students are able to access the information present within each class			
40	Create mechanism to allow teachers to create new assignments with set deadlines, descriptions, titles and resources (downloads)	Assignments created by teachers appear on the class and are accessible by students			
41	Allow students to view upcoming assignments from the class window	Students are able to view the assignments when the teacher creates them			
42	Order assignments based on deadlines	Assignments are listed based on priority leading students to complete assignments based on what is due earlier			
43	Allow students to open assignments and view content such as title, description, deadline and supporting resources	Students can access the content within the assignments			
44	Create upload system for assignments so students can submit their work	Teachers receive a file that they are able to open and grade			
45	Implement assignment creation system from teacher side	Teacher can easily create and assign new tasks			
46	Allow teacher to view student assignment submissions	Teacher can view and open student submissions			

47	Add rubric to marking window along with a section which allows the teacher to give a point score for a variety of criterias	Teacher can use the system to grade student's papers and the rubric to provide guidance for marking			
48	Implement feedback generation based on the type of paper and the point score given by the teacher	The point score for each section in the rubric is passed to an AI that generates feedback			
49	Allow teacher to add to the feedback and the submit the final feedback	Teacher can edit feedback and finalise it when they are satisfied with it			
50	Feedback is returned to student in the form of their original document with another page added on that contains feedback in cursive writing so it appears hand drawn	Students can access feedback returned to them along with the written feedback at the end			
51	Discuss common feedback given to students for different essay types	I will have a list of common feedback which I can apply and use to prompt inject the AI			
52	Create a prototype that allows teacher to input a score and an essay type and feedback is generated	Prototype will allow sample scores to be inputted and feedback to be returned			
53	Present prototype to client	Client will provide feedback based on the prototype which will guide all design decisions			
54	Repeat this process until the prototype meets the requirements of the	Will end up with a design that perfectly suits the needs of the client			

	client				
55	Develop student profile section	Teachers have a window to give an overview of student's academic performance			
56	Add student grades within the profile and a graph showcasing their progress	Student's are able to track their academic progress and easily visualise grade trajectory			
57	Add automatically generated comments based on what the student needs to work on most	Comments will be able to help students and guide them in the learning process			
58	Create system flag on the teacher side that orders students based on priority and showcases which students may need more help	Teachers receive prompts to enable to give specialised help to students that require attention			
59	Discuss with client what constitutes high priority in a student	I will have an idea of how to prompt inject the AI and it will provide a basis that I will make sure it follows			
60	Create a system architecture diagram for Criterion B	The graphical representation of the system will give the client a better idea of how the program functions and whether or not it is catered to their needs			
61	Present full prototype to the client	Client will provide me with feedback which I will use to update the prototype and create a final			

		model			
62	Adapt the prototype based on the suggestions of the client	Will end up with a final model which complies with client needs and ends up being the final product			

Structural navigation chart for program

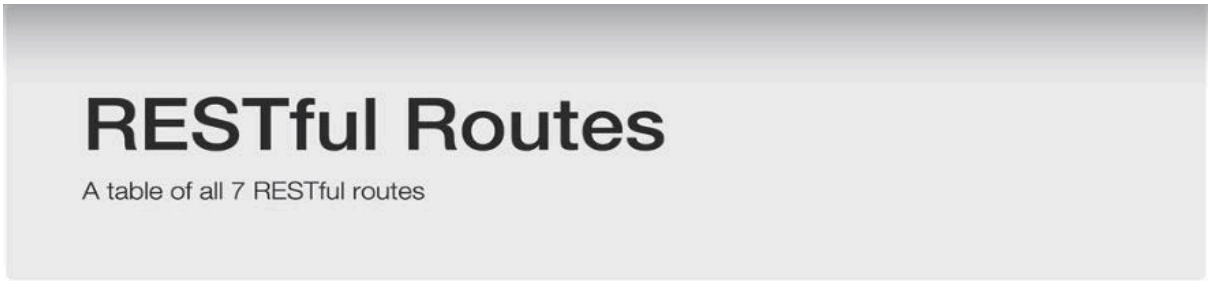


API routes

articles | (nested resource)

(Type in your table names and **click the table below** for full route info)

HTTP Request Type	URL Path	View File	Description
GET	'/articles'	/articles/index	display a list of all articles
GET	'/articles/new'	/articles/new	return an HTML form for creating a new article
POST	'/articles'	--	create a new article
GET	'/articles/:id'	/articles/show	display a specific article
GET	'/articles/:id/edit'	/articles/edit	return an HTML form for editing a article
PUT	'/articles/:id'	--	update a specific article
DELETE	'/articles/:id'	--	delete a specific article



Name	Path	HTTP Verb	Purpose
Index	/dogs	GET	List all dogs
New	/dogs/new	GET	Show new dog form
Create	/dogs	POST	Create a new dog, then redirect somewhere
Show	/dogs/:id	GET	Show info about one specific dog
Edit	/dogs/:id/edit	GET	Show edit form for one dog
Update	/dogs/:id	PUT	Update a particular dog, then redirect somewhere
Destroy	/dogs/:id	DELETE	Delete a particular dog, then redirect somewhere

Name	Path	HTTP verb	View Database Entry	Purpose
Login	/api/login	GET	/Users/username /Users/password	
Signup	/api/login	POST	/Users/username	
Access	/api/token	GET	--	
Get classes	/api/classes	GET	/	

Submission	/api/upload	POST	/Users/Assignments	
Chat History	/api/chat/history	GET		
New chat	/api/chat/new	POST		
Message	/api/chat/message	POST		

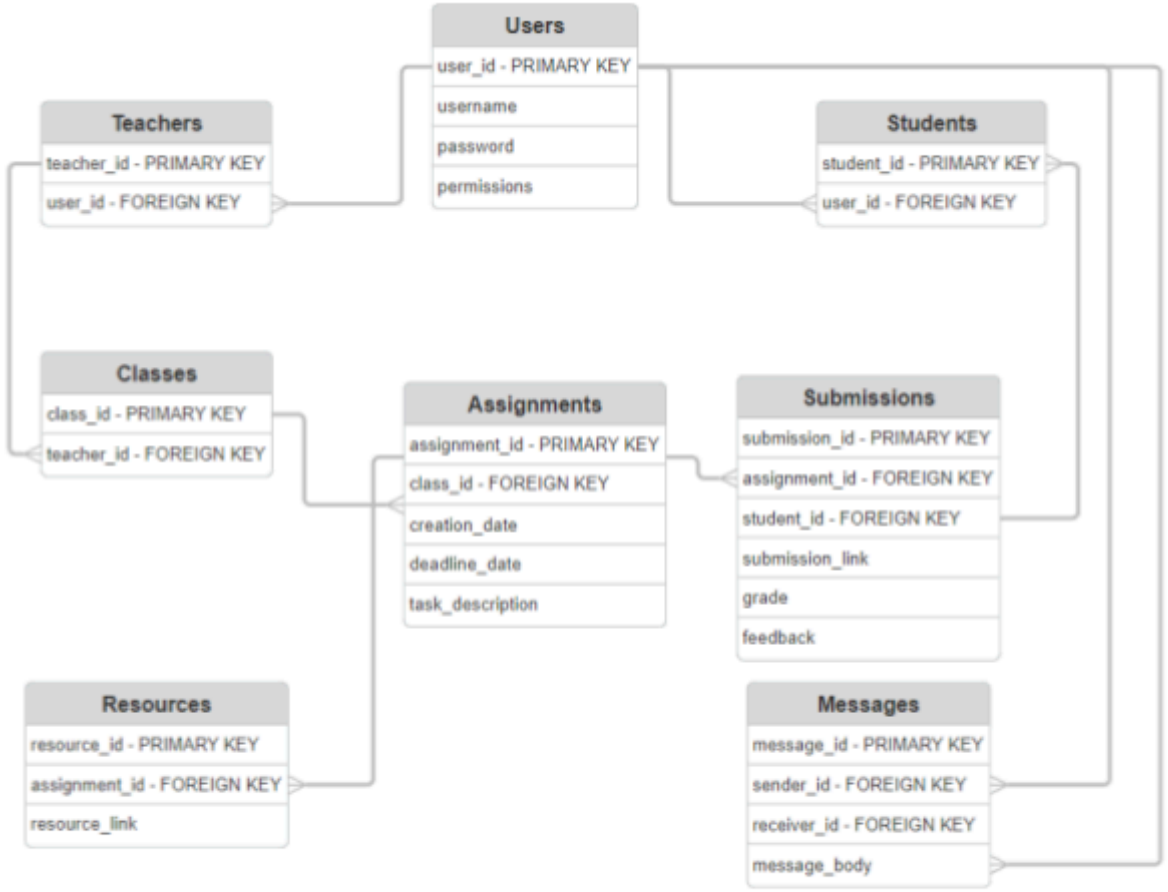
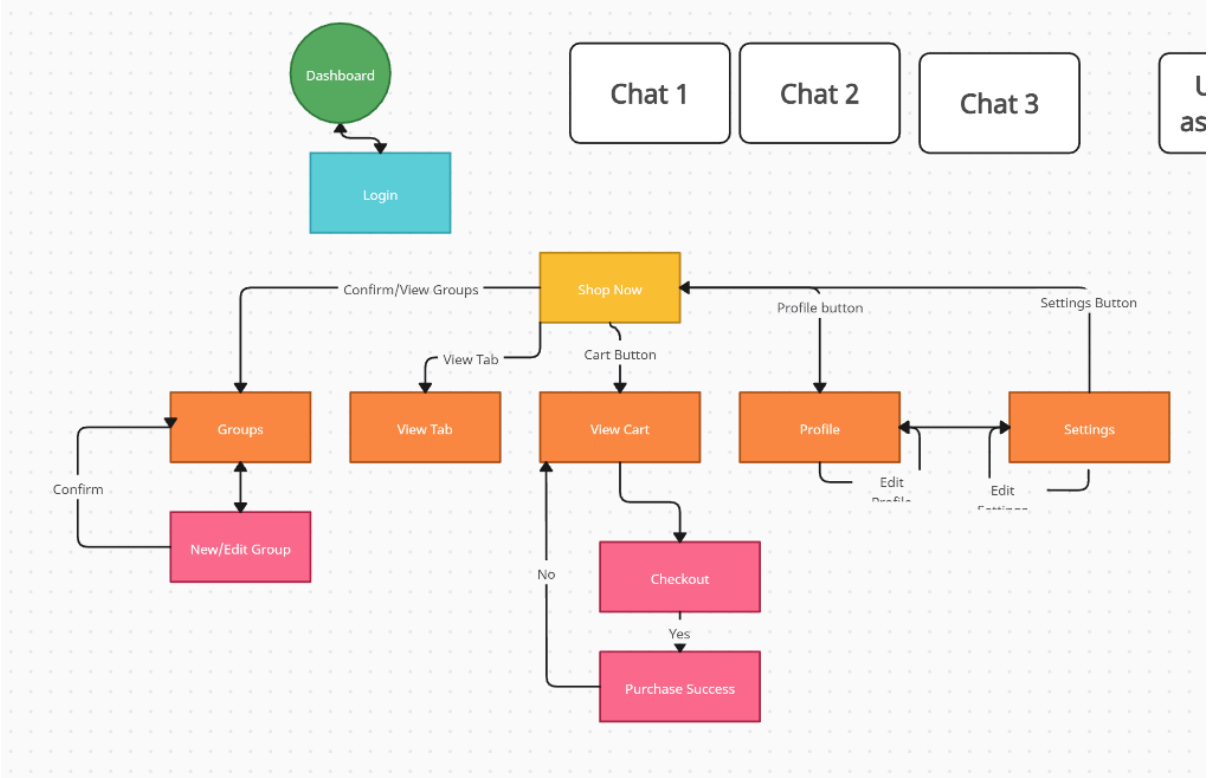
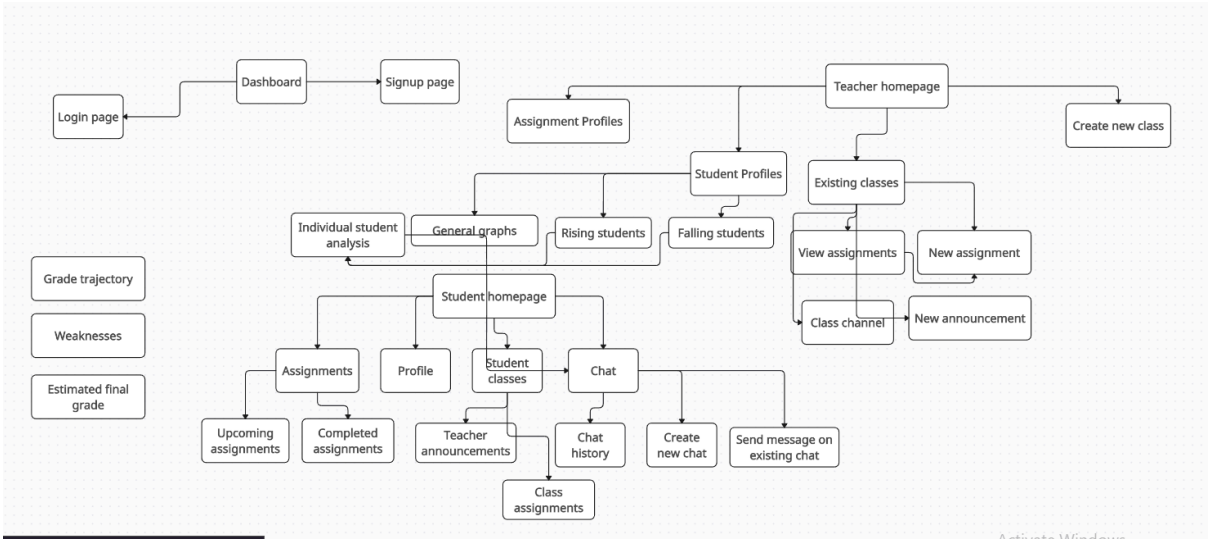


Figure x. Relationship diagram of *"database_name"* database used in project_name


Dashboard
Login
Chat 1
Chat 2
Chat 3
Shop Now
Confirm/View Groups
View Tab
Cart Button
Profile button
Settings Button
Groups
View Tab
View Cart
Profile
Settings
Confirm
New/Edit Group
No
Checkout
Yes
Purchase Success
Edit
Edit

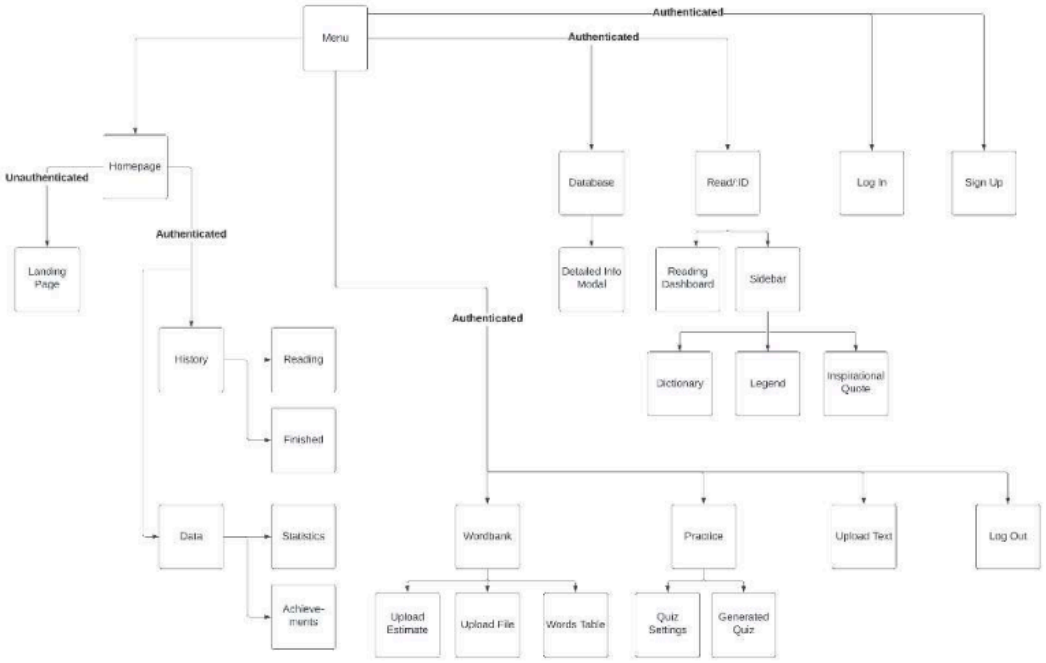

Dashboard
Signup page
Login page
Teacher homepage
Assignment Profiles
Create new class
Student Profiles
Existing classes
Individual student analysis
General graphs
Rising students
Falling students
View assignments
New assignment
Grade trajectory
Weaknesses
Estimated final grade
Student homepage
Class channel
New announcement
Assignments
Profile
Student classes
Chat
Upcoming assignments
Completed assignments
Teacher announcements
Chat history
Create new chat
Send message on existing chat
Class assignments

To do:

- API routes
- Modular design

Menu Navigation and Page Structure

Unauthenticated users will be presented with a landing page and the options to sign up or log in, whereas authenticated users will have all functionality available.



-
- Flowcharts to include:
    - Login
    - Signup
    - Access homepage using token
    - Retrieve classes
    - Submitting an assignment
    - Retrieving chat history
    - Creating a new chat
    - Sending a new message