

---

## **FINAL YEAR PROJECT REPORT**

---

STUDENT: FAHIM HOSSAIN  
SUPERVISOR: NICOLAS HINE  
PROGRAM OF STUDY: COMPUTER SCIENCE

GOLDSMITHS, UNIVERSITY OF LONDON

DATE: 15/08/2019

## **Abstract**

The primary role of a teacher in a classroom is to impart knowledge to their pupils. Teachers are given a curriculum that they must follow and ensure every child placed in their care understands the content to a satisfactory degree. To test this, standardised exams are taken to assess pupils' progress throughout their years in education. This, however, has fallen under scrutiny in recent years on whether this is the best approach. Standardised exams put unneeded stress onto pupils and recent developments show that teacher assessments are just as accurate at predicting future success as standardised tests anyway and is much less detrimental to pupils' mental health. This has prompted me to create a tool that helps teachers in their assessment practices, with the hope that, with advancement in teacher assessment technologies, we can move away from these standardised tests which in turn will increase pupil mental wellbeing, increase accuracy of assessment and increase quality of teaching.

## **Acknowledgements**

I would like to thank everyone involved in this project for helping it reach the point it is currently. Firstly, I would like to thank my supervisor, Nicolas Hine, for his advice and expertise helping me plan the scope and direction of the project. A special thank you to all the educators who acted as research participants and took the time out to answer my questions which allowed me to plan out the design and functionality of the project and a major thank you to those that returned for the testing phases, the feedback received was vital in tweaking development to shape the project into what it is today. Lastly, I would like to thank my friends and family for their support and encouragement throughout the development process, without them I would have never gotten as far as I did.

# Table of Contents

<b>Abstract .....</b>	<b>i</b>
<b>Acknowledgements .....</b>	<b>ii</b>
<b>1 Introduction .....</b>	<b>003</b>
1.1 Motivation .....	003
1.2 Aims and Objectives .....	003
1.3 Project Tools .....	004
<b>2 Background Research .....</b>	<b>005</b>
2.1 Educator Interviews .....	005
<b>2.2 Existing Systems .....</b>	<b>006</b>
2.2.1 Bug Club .....	007
2.2.2 Mathletics .....	008
2.2.3 MyMaths .....	009
2.2.4 InsightTracking .....	010
2.3 Children App Design .....	011
2.4 National Curriculum .....	012
<b>3 Design .....</b>	<b>012</b>
<b>    3.1 System Requirements .....</b>	<b>012</b>
3.1.1 Front End Requirements .....	012
3.1.2 Back End Requirements .....	013
3.1.3 Middle Ware Requirements .....	013
<b>    3.2 System Design .....</b>	<b>013</b>
3.2.1 Functional Overview Diagram .....	014
3.2.2 System Architecture .....	014
3.2.3 Database Design .....	018
3.2.4 Application Design .....	019
3.2.5 Activity Design .....	034
<b>4 Implementation .....</b>	<b>040</b>
<b>    4.1 HTML, CSS and JavaScript Development .....</b>	<b>040</b>
4.1.1 External Frameworks .....	040
4.1.2 General Application .....	041
4.1.3 Quiz Activity .....	043
4.1.4 Results Visualisation .....	046
<b>    4.2 PHP Development .....</b>	<b>049</b>
4.2.1 Dynamic Content .....	049
4.2.2 Registration System .....	050
4.2.3 Login and Logout Systems .....	054

4.2.4 Profile Modification System .....	056
4.2.5 Task Assignment System .....	056
4.2.6 Results Storage and Fetching Systems .....	058
<b>5 Testing .....</b>	<b>062</b>
5.1 Whitebox Testing .....	062
5.2 Blackbox Testing .....	063
5.3 User Testing .....	064
<b>6 Conclusion .....</b>	<b>064</b>
6.1 System Successes .....	064
6.2 System Failures .....	065
<b>6.3 Future Development .....</b>	<b>066</b>
6.3.1 Additional Assignments Functionality .....	066
6.3.2 Sub Categories .....	066
6.3.3 Pupil Feedback .....	066
6.3.4 App Support .....	067
<b>Bibliography .....</b>	<b>068</b>

## Appendices

<b>Appendix A: Functional Overview Diagram</b>	<b>070</b>
<b>Appendix B: System Architecture Diagrams</b>	<b>077</b>
<b>Appendix C: Database Design</b>	<b>085</b>
<b>Appendix D: Application Design Concepts</b>	<b>097</b>
<b>Appendix E: Activity Design Concepts</b>	<b>137</b>
<b>Appendix F: Code</b>	<b>157</b>
<b>Appendix G: Testing</b>	<b>279</b>
<b>Appendix H: Weekly Logs</b>	<b>289</b>

# Introduction

## 1.1 Motivation

Research published by the Journal of Child Psychology suggests that teacher assessment is just as good at predicting future pupil success as standardised exams such as SATs. [1] Findings show a strong correlation between teacher assessments and future test results across the three main subjects; Maths, English and Science, with around 90% accuracy. [1, 2] These results question the necessity of standardised exams, especially coupled with the recent YouGov poll that determined 96% of primary heads are concerned about the wellbeing of their pupils due to the pressures these exams place on them. [3] The poll indicated that 72% of headteachers felt assessing pupils under exam conditions to measure capability, were unfair and placed undue stress. [3] These findings start to raise questions; such as, if teacher assessment is just as accurate as these standardised tests, that have such detrimental repercussions, would it not be more beneficial to concentrate further on enabling educators to assess pupils with further ease and accuracy? Regular assessments over the course of a child's education would provide a much better understanding of the child's ability rather than a single exam towards the end. Consequently, this should reduce any negative effect on pupils' mental health; which according to a survey by the Key, has been on the steady decline over the past two years, with cases of stress, anxiety and panic attacks increasing in more than three-quarters of primary schools. [4] Neil Carmicheal, chair of the education committee stated, "Many of the negative effects of assessment in primary schools are caused by the use of results in the accountability system rather than the assessment system itself." [4] Changes to reflect this sentiment can already be seen in the education system with Ofsted vowing to completely overhaul how they inspect schools, relying less on exam results and more on 'quality of education'. [5] Therefore, I believe it is vital that educators are granted effective tools that enable them to achieve a higher standard of assessment.

## 1.2 Aims and Objectives

I aim to create a tool with the intention to allow educators to have comprehensive, detailed and accurate analytics, leading to much richer evaluations. A sensitivity to safeguarding pupils' wellbeing will also be considered through fun learning whilst still challenging and motivating.

Current assessment of pupils, especially at KS1, require educators to observe and make inferences from these. My application intends to make this process easier and streamlined by using technology to record pupils' analytics and display this data as clean and concise as possible. Inferences will be easier to observe and justify whilst also making it easier to compare pupils progress with their peers. This will allow educators to track a pupil's personal growth as well as the entire classes.

This project will primarily focus on assessment because content tends to vary over time and thus applications focused on these aspects become redundant and irrelevant and sometimes difficult to justify financially by educators. The method of assessing generally remains consistent so a tool that focuses on this will be cost effective thus more beneficial.

## 1.3 Project Tools

This project involves three main components, front end, back end and middleware with each component having their own specific tools that collaborate, to create the web application.

The first of these components, front end; will be the user interface. This is what the user experiences directly in the application. Hypertext Markup Language (HTML), a language that creates web content that can be displayed by a browser, Cascading Style Sheets (CSS), a language that adds styling to web content and JavaScript, a language that allows more advance functionality to these web content, will be used here to create the bulk of the user experience. Teachers and pupils can register and log into individual accounts and view specific content relevant to them. Pupils will access interactive quizzes that will be used to assess their progress on their curriculum, while teachers can use the dashboard to view these results.

The second of these components, back end, is what happens behind the scenes. It is the inner workings of the web application and the main functionality of the website. Hypertext Preprocessor (PHP), a server-side language that allows the ability to create dynamic webpages, will act as a middle-man between the browser and a database. This allows for the retrieval and storing of information and will be used throughout the application. An example could be; data received from users registering for an account from the front end, this data can be passed into a PHP file where it will be correctly handled, cleaned up and stored into a database for later retrieval.

The final main component, middle ware, will be functionality that acts as a bridge between the front and back end components. Here Asynchronous JavaScript and XML (AJAX), a set of web development techniques that is used within JavaScript, will be used for sending and receiving data from a PHP file in the background, without having to reload the entire page. This is immensely advantageous as it allows the application to send data collected from the quiz section of the application, to be stored on the database without forcibly reloading the page and disturbing the user's experience. Furthermore, the receiving of analytics from the database, for visualisation, would be a seamless non-obstructive experience.

I aim to use a PHP development environment known as ‘XAMPP’, this is to gain access to an Apache distribution; a local server to store the applications resources, (HTML documents, CSS stylesheets, media files etc) as well as handle HyperText Transfer Protocol (HTTP) requests; a protocol that allows the fetching of these resources. XAMPP also allows access to MariaDB (MySQL), a database that will be used to store data corresponding to the application. It is an open source package that can be used freely without legal ramifications.

I have decided on using MariaDB as it is a relational database. It is structured to recognise relationships between stored items, which is perfect because my application is centred around relationships. Teachers are responsible for many pupils; each pupil has different activities that they will interact with and each interacted activity holds information about every pupil.

I also plan on using Charts.js, a JavaScript library that allows you to draw different types of charts, for the visualisation aspect of the project. Data received through AJAX can be converted into information that teachers can use to assess their pupils’ capabilities.

# Background Research

Initially, I wanted to create an application that taught children their school curriculum; analysing and adapting to the child's learning style in the process. The thought process behind this was that if a child was taught in their learning style, perhaps this could improve upon their understanding of the content. However, this was scrapped upon further research. An article by Cedar Riener and Daniel Willingham [6] states there is no real proof of learning styles existing. Other factors such as a child's interest, social class and upbringing have more influence on how fast or slow a child learns.

A study by Ikseon Choi, Sang Joon Lee and Jeongwan Kang [7] supports this claim by concluding that learning style didn't have an impact on learning, but that pupils simply just adapted to the learning style they were being taught in. The study's concluding line, "Students may have preferences about how to learn, but no evidence suggests that catering to those preferences will lead to better learning," allowed me to evaluate that perhaps the different learning styles aspect of the project is unnecessary and evidently unfounded upon. The next focus was to contact educators, the target stakeholders, to conduct market research. This would allow me to not only ensure the tool is useful, but also provides feedback directly from experts in this field, ensuring design and functionality have a solid foundation rather than using assumption as initially done.

## 2.1 Educator Interviews

Initial market research consisted of Interviews. Questions asked within these were aspects related to how assessment are already conducted, whether similar technology is already integrated and finally, an evaluative response to how successful the current systems within schools are. Responses demonstrated the following;

Although technology is currently used within the classroom, these tools are mainly used for teaching rather than assessing. Therefore, educators were generally positive to this project because they were already successfully using technology to teach and were happy to utilise it to assess.

Pupils are assessed through a range of assessment methods, typically using colour coding to make things easier. Pink is frequently used to signify a pupil that has not understood the learning objectives and needs assistance, while green is used to show they have understood. Pupils are also asked to fill out a form before any new learning, to discuss what they felt confident in, providing teachers with information to adapt plans to include or possibly omit certain topics based on pupil response. Upon completion of a topic, pupils are often assessed through written tests; these tests are then observed by a teacher throughout the year, to measure progress. Duration to complete tasks becomes more important in KS2, than KS1, as speed and stamina become a higher priority. Whilst time isn't explicitly measured during KS1, it is still an important factor allowing teachers to know if there is a trend in pupils taking too long on a specific topic and investigate what could be hindering them.

In terms of what features would be expected from an assessment tool, the most popular response received was having detailed individual data as well as a general overview of the whole classes' performance on individual areas. Teachers regularly attend termly assessment meetings where pupil progress are discussed at length. Often teachers are questioned why pupils aren't meeting their targets. It seems apparent that having access to comprehensive information of pupil's abilities would allow teachers to identify key reasons as to what could be an obstructing their learning and making it easier to identify how they can improve and adapt their teaching to support these pupils. A class overview would also allow teachers to analyse whether certain topics require reaffirmation for the entire class or if problems are at an individual level.

Another common response was having the ability to create sub groups within their class. Often teachers split and group pupils into sub categories to make teaching easier, this, however, can have adverse repercussion to the pupils. If a child is categorised into the low attaining set, it can have lasting impact on their self-confidence. By having the assessment tool handle the sub grouping, this would allow teachers a way to segment their class into groups resulting in better planning and specific teachings for each group, without the ramifications discussed. Furthermore, the added benefit would mean teachers can check up on each group at their own time, i.e. if a teacher wanted to focus on their low attaining set, they could bring up the data for only those pupils, allowing for a more focused experience. Additionally, it would allow teachers to adapt assessment to each child's ability level, those struggling could be brought up to their peers' level slowly while allowing those excelling to continue being challenged.

Another common requested feature was the ability to have the assessments be customisable. Existing education tools tend to come with their own data, whilst useful when teaching, often, these data aren't entirely compatible with what is taught within lessons. There could be different methods used; changes in topics or specific learning criteria not being within the tool at all. Allowing a tool where teachers can input their own data and create tests specific to the lessons learnt, would provide assessments that are tailored specifically to that class. They can also edit and adapt content for future pupils with different learning needs, leading to much more accurate data as well as keeping the tool relevant at all times.

## 2.2 Existing Systems

A vital piece of information that was acquired was the existing systems being used currently within the classroom for the purpose of education. Market research around these tools would allow understanding of what features are expected of an education tool, common design sensibilities could be reflected in the project to allow easy transfer of skills and missing features could be identified to allow greater functionality to the project. For instance, most of these tools seem to focus on the teaching aspect with little to no assessment features present and those that do, could do more with the concept. Furthermore, it seems education tools are very specialised, focusing on a single topic like English or Maths. This would mean educators are required to invest in multiple tools to cover the breadth of subjects taught. This could be eliminated by having a single tool that is able to assess in a large assortment of subjects rather than many small tools. This also allows teachers to have all their data in a single place rather than thinned out across multiple applications. Pupils would no longer need to remember a multitude of accounts and constantly logging in to each to see if they were assigned any work as a single account is only necessary.

## 2.2.1 Bug Club

Bug Club [8] is an English centric education platform centred around helping pupils improve upon their reading and vocabulary skills. It mainly consists of having pupils read extracts from stories and provided with multiple-choice questions where they are required to select the correct answer. This is to assess their reading comprehension and to see if they can internalise and retain information. Design wise, Bug Club caters towards the younger age group by having multiple cartoon characters throughout the website. When answering a multiple-choice question, one of these characters lets the pupil know whether the answer was correct.

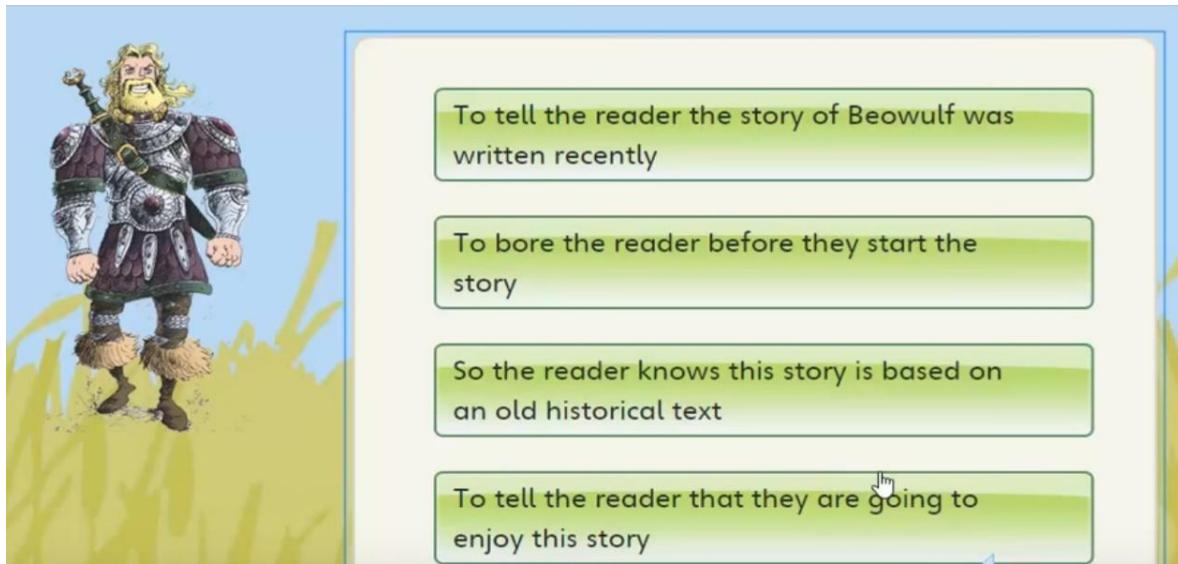


Figure 2.1: BugClub Activity Page

A screenshot of a BugClub character, a blue and yellow insect-like creature with wings, standing next to a text box. The text box contains the message: "Go back and read the question carefully." Below the text box is a yellow circular button with the word "OK" and a hand cursor icon pointing at it. In the background, there is a dark grey area with some text and a vertical bar with the letters "ophe" and "p".

Figure 2.2: BugClub Character

These design sensibilities were a great inspiration to my activity designs; having a cartoon character during activities will help children engage and make the assessment seem less like an exam, minimising stress levels. Sound effects will aid in this effort too, making the activity fun and dynamic. Bug Club does fall short though, aside from the few sound effects, there is no music. This can make things feel boring and uneventful, especially with how static the visuals are. The visuals are stationary images, making the activity feel less dynamic. The colour choices, while consistent, aren't particularly vibrant. These missing features could be incorporated into my applications to allow it to stand out from the competition and help make the application far more appealing in the process.

## 2.2.2 Mathletics

Mathletics [9] is a mathematics centred online learning space where children can have healthy competitions on challenges to allow pupils to strive to achieve their potential. The main premise is that pupils can select anywhere on a world map and compete with pupils worldwide. Participants have sixty seconds to answer as many maths related questions in order to win. Pupils are then shown a screen with their results and are ranked based upon their performance.

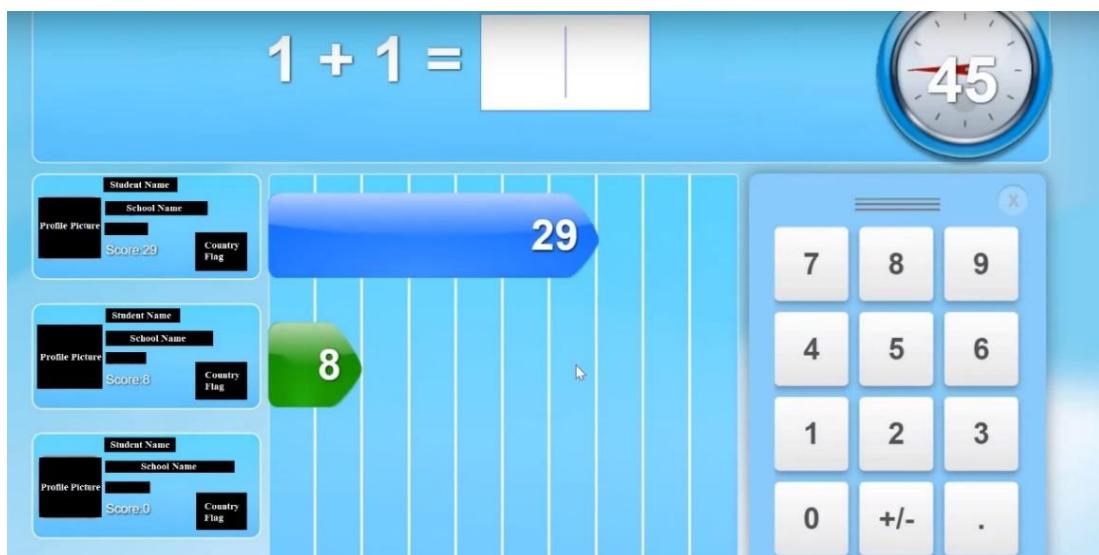


Figure 2.3: Mathletics Activity Page

Your Answers	Results	Corrections
$1 + 5 = 6$	✓	
$1 + 1 = 2$	✓	
$3 + 3 = 6$	✓	
$1 + 4 = 5$	✓	
$1 + 1 = 2$	✓	
$3 + 3 = 6$	✓	
$1 + 1 = 2$	✓	

Figure 2.4: Mathletics Results Page

I decided to adopt Mathletics results page approach, signalling to pupil exactly what they got right and wrong. This provides pupils an efficient visual representation of their performance on the activity without deprecating their efforts and a simple way to allow pupils to reflect on their time during the activity. They can decide whether they are happy with their results or would like to attempt again to improve. However, I felt the competitive aspect of Mathletics would not be a good fit for this project. This is due to the fact the tool is primarily an assessment application with the aim to make assessments fun and gain meaningful analytics and by children competing, it could add a level of unnecessary stress and may cause inaccurate data.

### 2.2.3 MyMaths

MyMaths [10] is a learning platform that offers resources to aid in teaching and developing pupils' math skills. It offers fun and interactive teaching both on an individual and classroom level to improve pupils' confidence and understanding. Teachers can set lessons to pupils, which will not only teach content but assess their abilities. Teachers can filter out which pupils they assign certain tasks to and assign deadlines for the tasks if they wish. They can then later view the results of these activities from a dashboard.

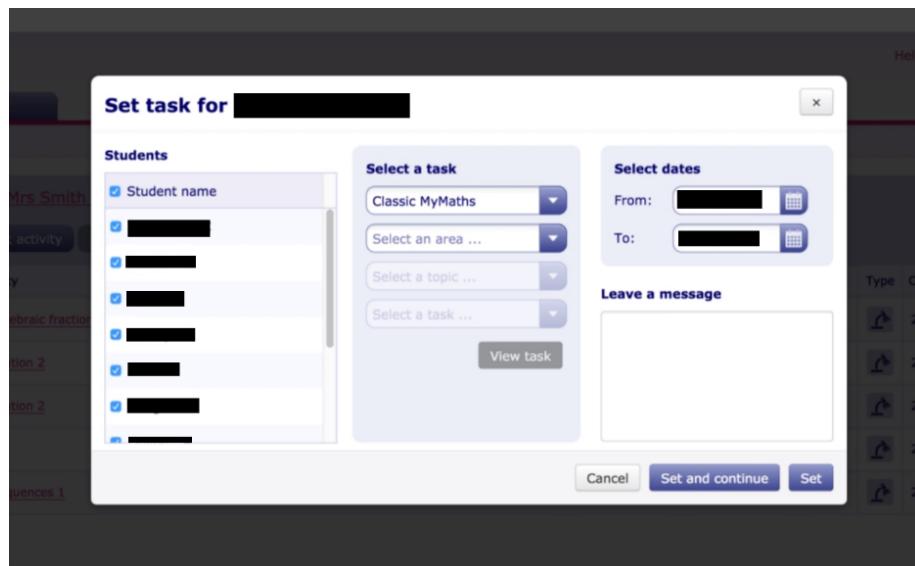


Figure 2.5: MyMaths Student Assignment Page

Allocation		Results		Admin	
		Individual Tasks		Summary	
<b>Narrow by...</b>		All Classes > [REDACTED]			
<input checked="" type="checkbox"/> Selected Dates		<a href="#">Download results</a> • <a href="#">Download reports</a>		<a href="#">Delete these results</a>	<a href="#">Message students</a> <a href="#">Toggle Fullscreen</a>
<input checked="" type="checkbox"/> Level					
<input checked="" type="checkbox"/> Type of Task					
<input checked="" type="checkbox"/> Library					
<input checked="" type="checkbox"/> Number		Topic	Task name	[REDACTED]	[REDACTED]
<input type="checkbox"/> Algebra		1	Number facts and doubles 1	25%	
<input type="checkbox"/> Shape		3	Adding in columns	89%	81%
<input type="checkbox"/> Data		4	D2C Multiples		89%
<input checked="" type="checkbox"/> Boosters		5	Shape Rotation symmetry	43%	
<input checked="" type="checkbox"/> A level		6	Data Converting measures		
<input type="checkbox"/> IGCSE		7	C2B Ratio dividing 2	60%	
<input type="checkbox"/> Show percentages		7	Algebra Brackets	48%	54%
<input type="checkbox"/> Show marks		7	Data Questionnaires	50%	40%
<input checked="" type="checkbox"/> Include work not done yet		8	Shape Circle theorem proof	100%	71%
<input checked="" type="checkbox"/> Only show allocated work		8	IGCSE Depreciation		
		8	IGCSE Multiplying algebraic fract...		

Figure 2.6: MyMaths Results Page

I decided to adopt the dashboard approach from My Maths and have all the functionality in a single space. Features such as, the ability to add and remove pupils are vital as classrooms are ever shifting. Assigning certain pupils' specific tasks is imperative as discussed during the educator interviews. Different pupils' progress at a different pace and this feature would allow the ability to have slower pupils time to catch up whilst ensuring the more gifted pupils are properly challenged. The listing of the results, however, seem quite limited. It only tells you the percentage a pupil received for an assignment, this is only a single factor in assessing pupils and my application would display much more data than seen here. This does mean a simple design of listing a pupil name and their data on a single page wouldn't be the best approach. I would need to look deeper into design choices and discuss with teachers on how best to proceed, while ensuring functionality is kept simple and intuitive.

## 2.2.4 InsightTracking

InsightTracking [11] is a spreadsheet assessment tracking system that accumulates assessment data to be displayed in a spreadsheet format. It allows teachers to fill in individual pupil progress through different learning objectives, compare between pupils and lighten teachers' workflow.

	Overview	Barrentine, Angla	Barrera, Freeman	Beaston, Harley	Behringer, Katie	Bellman, Roxann	Benge, Esteban	Bolden, Latoria
Y5 Objectives								
□ Discuss understanding of texts, including exploring meaning of words in context								
□ Ask questions to improve understanding of texts								
□ Summarise ideas drawn from more than one paragraphs, identifying key details								
□ Predict future events from details stated and implied								

Figure 2.7: InsightTracking Spreadsheet

The simplicity of InsightTracking user-interface (UI) design is something I aim to take inspiration from. Ensuring teachers can readily use the application without having to learn is crucial, the project aims to make assessment easier on teachers so having them invest time and money into learning a new system would be nonsensical. Moreover, InsightTracking's use of colour to differentiate between different data allows better comparability. I aim to adopt this approach in my project so that, once users are accustomed to the colour scheme, they can easily find the data they are looking for simply by observing the colours on display. Where InsightTracking fails, however, is in its presentation. InsightTracking displays data in a spreadsheet-like format, whilst this is useful for reading data, it isn't easy for teachers to interpret the data in a meaningful way. My project aims to improve this by turning data into information through visualization. This would further improve a teacher's ability to assess pupils, while making the application easier to use. Using charts and graphs to visualize large amount of data makes it easier to derive meaning than filtering through a large spreadsheet manually.

## 2.3 Children App Design

Designing for children can be a difficult endeavour as children think differently to adults, they have different cognitive abilities, logic, motor skills and many other factors that need to be thoroughly considered in the design process. [12] The situation is a complex procedure and research into designing for children is critical at this early stage, it would allow any design mock-ups to be justified through facts and ensure designs that impede on functionality aren't considered, ensuring resource wasting is kept to a minimum. Through my research I had discovered three main criteria that need to be considered during any design process.

The first criteria, reduced learning, allow users to navigate through the application without putting much cognitive strain on the end user. This can be achieved through consistency within the application and following well established UI norms that users have become accustomed too. [13] Children, by the age of three, become well acquainted with devices like computers and iPad and thus become familiar with the different navigation possibilities such as scrolling and swiping. [12] This means the application should follow many navigation norms when it comes to web pages, such as ensuring nav bars are at the top of each page. This has the added benefit of teachers also being familiar with these design sensibilities and thus designing the application around children wouldn't necessarily hinder the user experience for the teachers. During the activity portion of the application I intend to use symbols to signify to the user the functionality an interaction would have. I would need to ensure symbols are unique and once the user has internalised a symbol with a specific action, the application doesn't betray this. It is imperative that every action the user can do is dependable; if a child expects a certain action to have a certain output and that output isn't performed then they may feel betrayed and lose confidence. [12]

The second criteria, functional feedback, states that children often expect visual and auditory feedback whenever they do something. [14] Children love praise and reward [15], by ensuring the application constantly encourages and rewards, it not only keeps the child engaged but also acts as a functional feedback, letting a child know an interaction was successful. I intend to use sound for the auditory feedback, letting the user know if their interaction was successful e.g. if they got an answer correct or are doing well. This would also act as a reward system for the user, eliciting happy music when they are doing well or encouraging music if they need a confidence boost.

The final criteria, engaging design, is the most crucial aspect. Engaging design gives users a motive to use the application; if a user isn't engaged within the app, they will quickly lose interest. As previously discussed, sound can be a factor in keeping users engaged but there are plenty of other aspects to consider. Animation can also play a significant role, children enjoy apps that have animations, a study demonstrated further that children even expect animations and are willing to spend several minutes trying to get the page to play an animation before they start. [16] This, along with research into Bug Club [8], an existing education tool, has prompted me to create activities with small characters that encourage pupils' as they work through the activity. The character animates positively when pupils get answers correct or worried to encourage pupils when they struggle. This would help keep engagement high and make the activities feel less like exams.

During the design process of the application, I plan on using experts, educators, to further improve design and functionality. Through dialogue discussing what they expect the application to look like, if an activity is appropriate, appealing to their pupils and if they would include any additions to designs to improve the overall quality of the project.

## **2.4 National Curriculum**

Research into the KS1 national curriculum was conducted to ensure the application conformed to the standards and practises set by the government. Teachers are required to prepare annual reports for parents before the end of the summer term, consisting of the pupil's personal attainment as well as the pupil's comparative attainment between the class. [17] This would be another facet that the project could fulfil. The project already aims to store data pertaining to pupil's assessment however most educators are also interested in accumulating classroom data that they could use to interpret the entire classes' ability. This would make end of summer reports easier to produce as teachers now have a visual representation of each pupil's progress all year round. Furthermore, teachers will have an easier time comparing between the pupil's and the classroom's progress fulfilling the two main report criteria set by the government, drastically reducing teacher workload.

In order to create the activities, I studied example papers [18], looking for patterns and common practices within them and determining the level at which the government expects these pupils to be at. This allowed me to create a set of questions that could be added to the activities, aligning themselves to those taught by the national curricula. For further credibility, during the testing phase of the activity, I gained teacher feedback to tweak questions, where appropriate, to better conform to existing assessments.

# **Design**

## **3.1 System Requirements**

To ensure the aims and objectives of the project are met, a set of minimum requirements for each component of the project will be clearly stated. This is to distinctly outline when a component is completed and at later stages will aid with testing. Should a component not pass all the minimum requirements, the project cannot be deemed to have met its objectives, allowing for a tangible end goal for the project.

### **3.1.1 Front End Requirements**

The front-end component must successfully pass the following tasks:

- Allow users ability to register an account with the site, ensuring the requested user name is available and that any input from the user is valid
- Allow users ability to modify their account credentials, ensuring the changes are reflected in their current session, where appropriate, and safeguarded against overlapping of existing usernames should the user's modification contain any clashes
- Allow users ability to securely log into and log out from these accounts at any point during their time on the application, changing the content of the web application's current session accordingly
- Prevents access to users to pages they are not permitted to, ensuring the site security and functionality remains intact
- Teacher accounts given the ability to view and modify student accounts under them

- Teachers account capable of assigning student accounts assignments and subsequently view results once complete
- Student accounts capable of viewing and completing these assignments
- All users able to visit, granted they have the permissions to, all pages available within the application through navigation alone
- Be responsive to the end users window size, adapting content accordingly

### **3.1.2 Back End Requirements**

The back-end component must successfully pass the following tasks:

- Sanitise and validate any user input that requires processing to minimise potentiality of SQL Injection attacks
- Ensure data that involves inserting to any database is valid and that all data that is required is present before inserting to maintain integrity.
- When data handling is determined by user accounts, the correct accounts involved should be fetched
- User account credentials secure and no sensitive information be available within the database. In the case of passwords, only their hash outputs should be deposited
- The retrieval and inserting of data through the application must be quick and efficient

### **3.1.3 Middle Ware Requirements**

The middle ware component must successfully pass the following tasks:

- Must quietly in the background pass data collected from the pupils' time during the activity to a separate PHP file for storing and inform the user whether the action was successful
- Receive pupil activity data from a separate PHP file, analyse and transform the metrics to visual information that can then be displayed on the web page

## **3.2 System Design**

System diagrams are essential to the development process, they act as blueprints on how the final application gets built. By stripping a large project down to its core components, it prevents ambiguity, allows for better planning and ensures resource wasting is kept to a minimum. These diagrams can be scrutinized by stakeholders allowing the development to be fine tuned to their specifications. Should inspection of current plans reveal that major overhauls are required, these problems can be more easily corrected at these earlier stages.

### 3.2.1 Functional Overview Diagram

A functional overview diagram was created [Appendix A: Functional Overview Diagram] to act as guidance for the projects outlines. Functional diagrams allow, at a glance, to see clear boundaries of a system and how each component is intended to collaborate. It also allows easy visualisation of the flow of data and how this is manipulated from input through the application to receive the intended output. Moreover, it will also aid in explaining the application to the stakeholders comprehensively, receive accurate feedback minimising the need for the stakeholder to have much technical understanding.

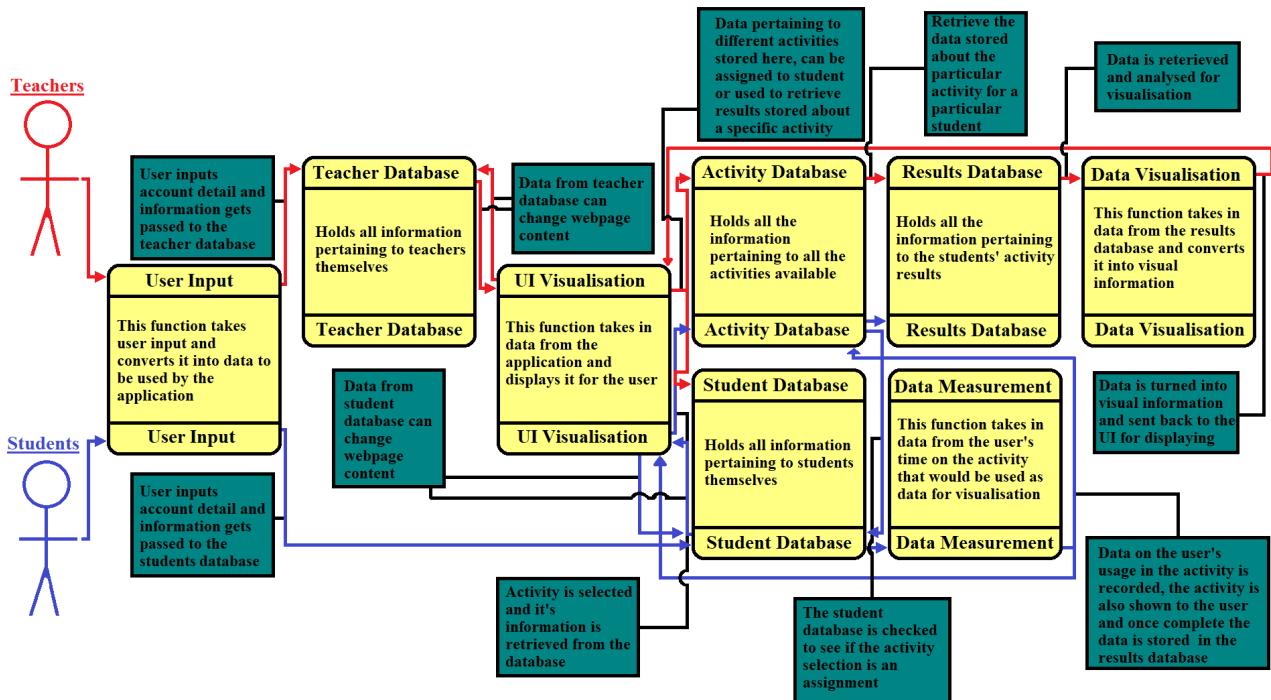


Figure 3.1: Functional Overview Diagram

### 3.2.2 System Architecture

The following segments aim to outline the architecture of each aspect of the application, the first being the login and registration system. The login system is planned to be a multi-level user system, redirecting one of three account types to their respective landing pages. The three planned account types are Admin, Teacher and Student. Admin accounts will have the ability to create new teacher accounts while teachers can create new student accounts automatically associated with them.

Figures 3.2 and 3.3 are snippets from the login and registration system architecture diagram [Appendix B: System Architecture Diagrams – Figure 1] and provides a general overview of the system, demonstrating the main functionality of each component of the system, the purpose of each components and how these components interact. The login system takes user input and validates the data in preparation for comparing existing data within the database, should the input data pass then the user is logged into the application.

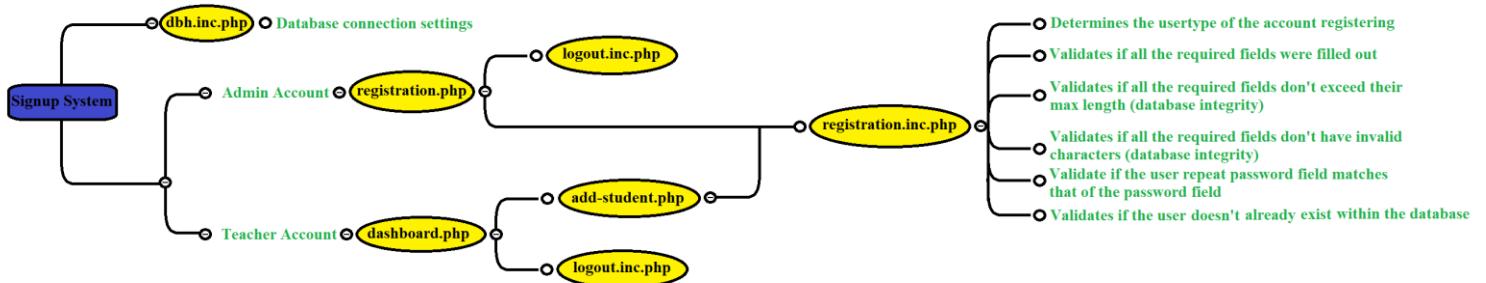


Figure 3.2: Registration System Architecture

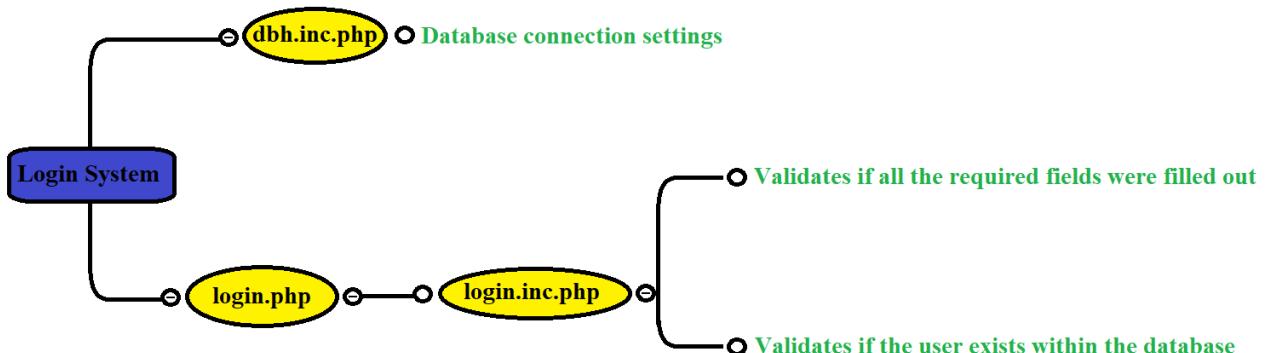
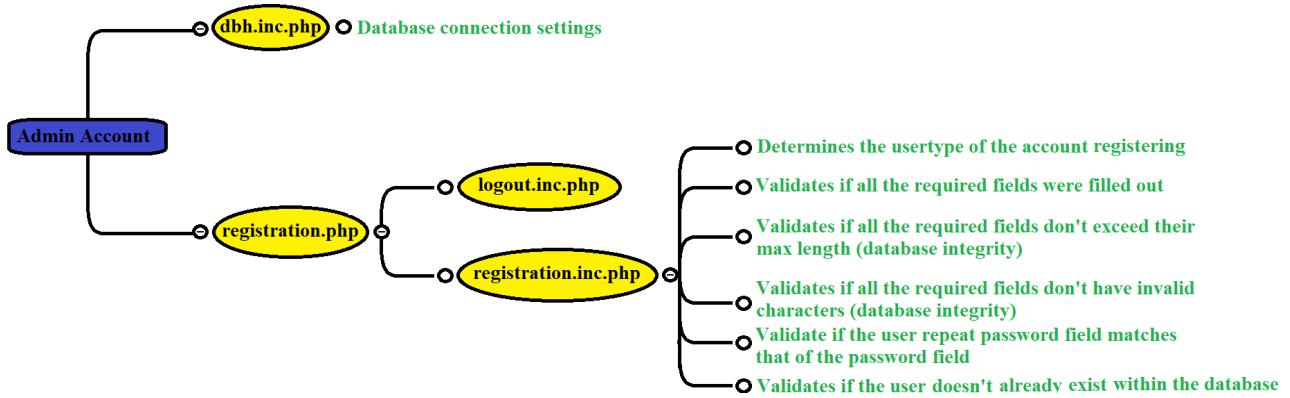


Figure 3.3: Login System Architecture

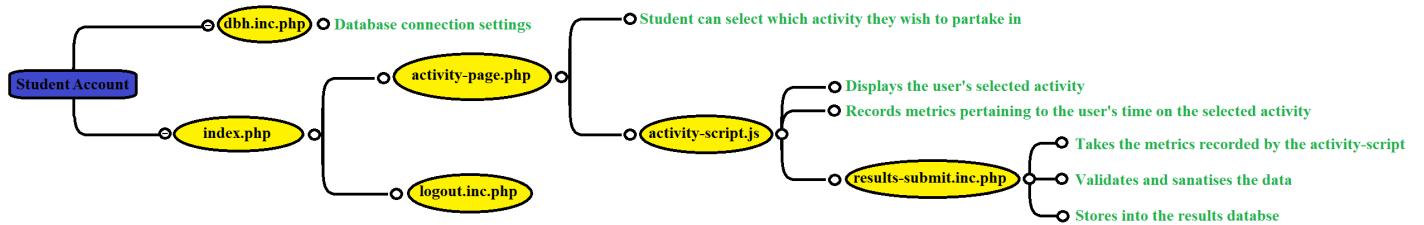
Similarly, for the registration process, the user is asked for a set of data that is used as credentials when logging in. Only admin accounts can create teacher accounts; this is to give a school full control over their system. In order to create an account, the following details are needed; the user's full name; a unique username; a password for the user's account and a repeat of the password to act as safeguarding against mistyping. Once this information is provided, it is processed and validated before being inserted. This is to maintain database integrity and protect against attacks such as SQL Injections. The system has no way of knowing what data its being given so it would be too dangerous to insert the data directly into the database without review. Simple mistakes akin to missing fields on required data could result in system failure or more malevolent attacks like SQL Injections, whereby a user purposely inserts malicious code into input fields in the hopes of destroying a system, could occur and thus these checks are imperative for system protection.

For registering student accounts, the process is functionally identical, however, only a teacher account can create a student account. Students are required by the system to be under a teacher for the application to be functional, if students had the ability to create their own account there is no guarantee that account would ever be registered to a teacher. Furthermore, especially for the younger students, it is tedious to have them register their own accounts when teachers know exactly which student they are teaching. By giving teachers the ability to add and remove students as appropriate, this allows the system to be prepared before term starts, so that students can immediately begin using the application rather than wasting time.

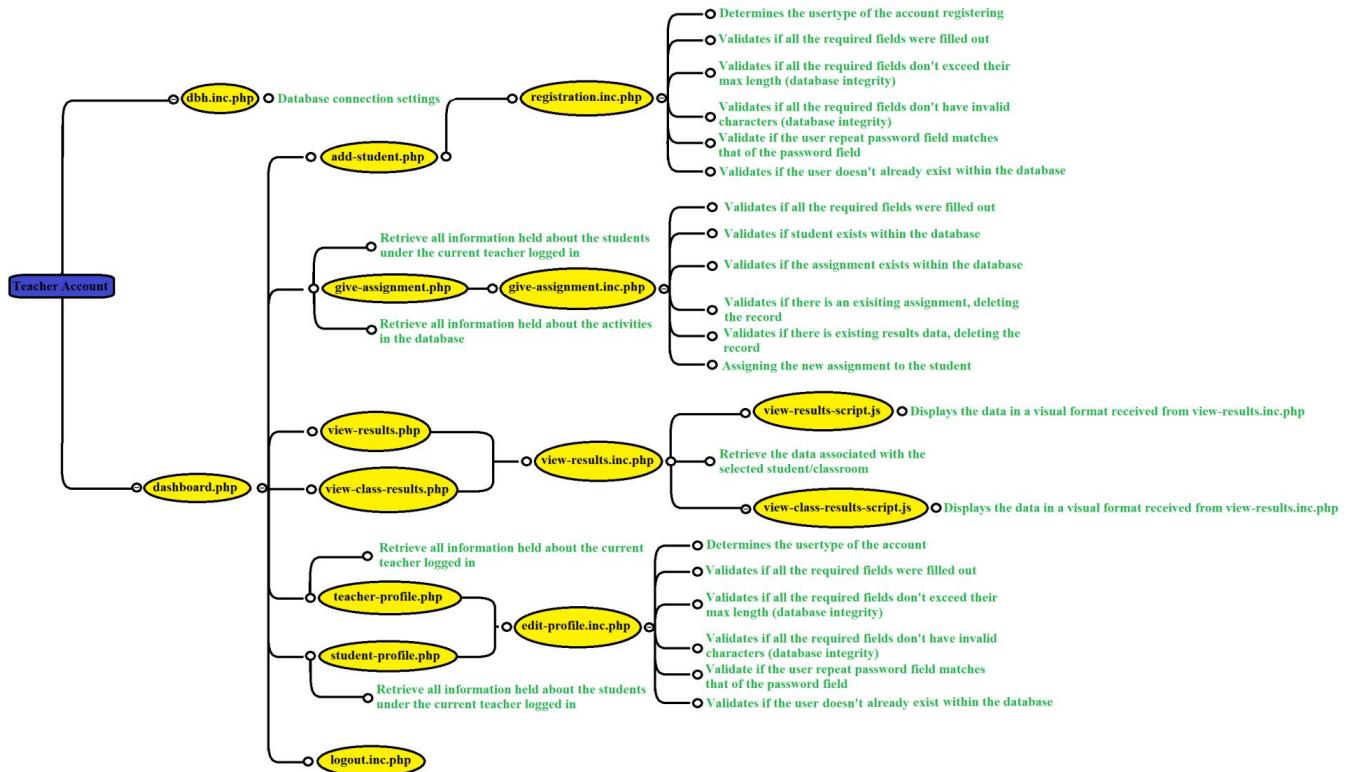
As stated, the application will have three account types all with their own specific functionality.



**Figure 3.4: Admin Account Architecture**



**Figure 3.5: Student Account Architecture**



**Figure 3.6: Teacher Account Architecture**

The diagrams above [Appendix B: System Architecture Diagrams – Figure 4, 5 and 6] represent the admin, student and teacher systems respectively.

Admin accounts will have the ability to create teacher accounts, the handling of which would be executed through a PHP file titled “registration.inc.php”, data is validated and inserted into the “teachers account” database.

Students will gain access to an activity page and select one to partake in. Once it is selected the JavaScript file “activity-script.js” will begin displaying the activity and quietly in the background record metrics for later assessment. Upon completion, the results are passed ‘results-submit.inc.php” to be handled and stored.

Teacher accounts will be the bulk of the application and can perform four distinct actions. The first action is registering student accounts, this follows an identical process to registration of teacher accounts. “Registration.inc.php” processes data into a student database. The second action, appointing assignments to students, involves retrieving all information held about students associated with the logged in teacher as well as retrieving activities held in the database. Once the user is presented with this information, they can allocate students to relevant activities and pass the data to “give-assignment.inc.php”. Here, several actions take place, data is validated, and student’s active assignment are updated accordingly, and any existing results stored for previous assignments are deleted, awaiting results related to the newly given assignment. Once students complete their assignments and there is data stored within the database, teachers can view results, either individually or as a class summary. Both actions use “view-results-inc.php” to retrieve the data within the database and depending on if they chose individual or class summery, the data is sent to view-results-script.js and view-class-results-scripts.js respectively. These JavaScript files aim to convert the metric data into visual information and then display the information to the user. In the case of individual summaries, the user can seamlessly change between users, without resorting to constantly refreshing the page. The final actions teacher accounts are capable of is modifying the user details of their own accounts as well as student’s. The user submits the modified data to “edit-profile.inc.php” where the data is validated and updated to their corresponding database.

### 3.2.3 Database Design

The database chosen to be used within the application is relational, meaning it is structured to recognise relationships between stored items i.e. teachers have many pupils under them, each pupil has different activities they can interact with and each interacted activity holds certain information about every pupil. The Entity Relationship (ER) diagram [Appendix C: Database – Figure 1] below demonstrates that the database will consist of 6 tables, admins, students, teachers, activities, activity assignments and result storage.

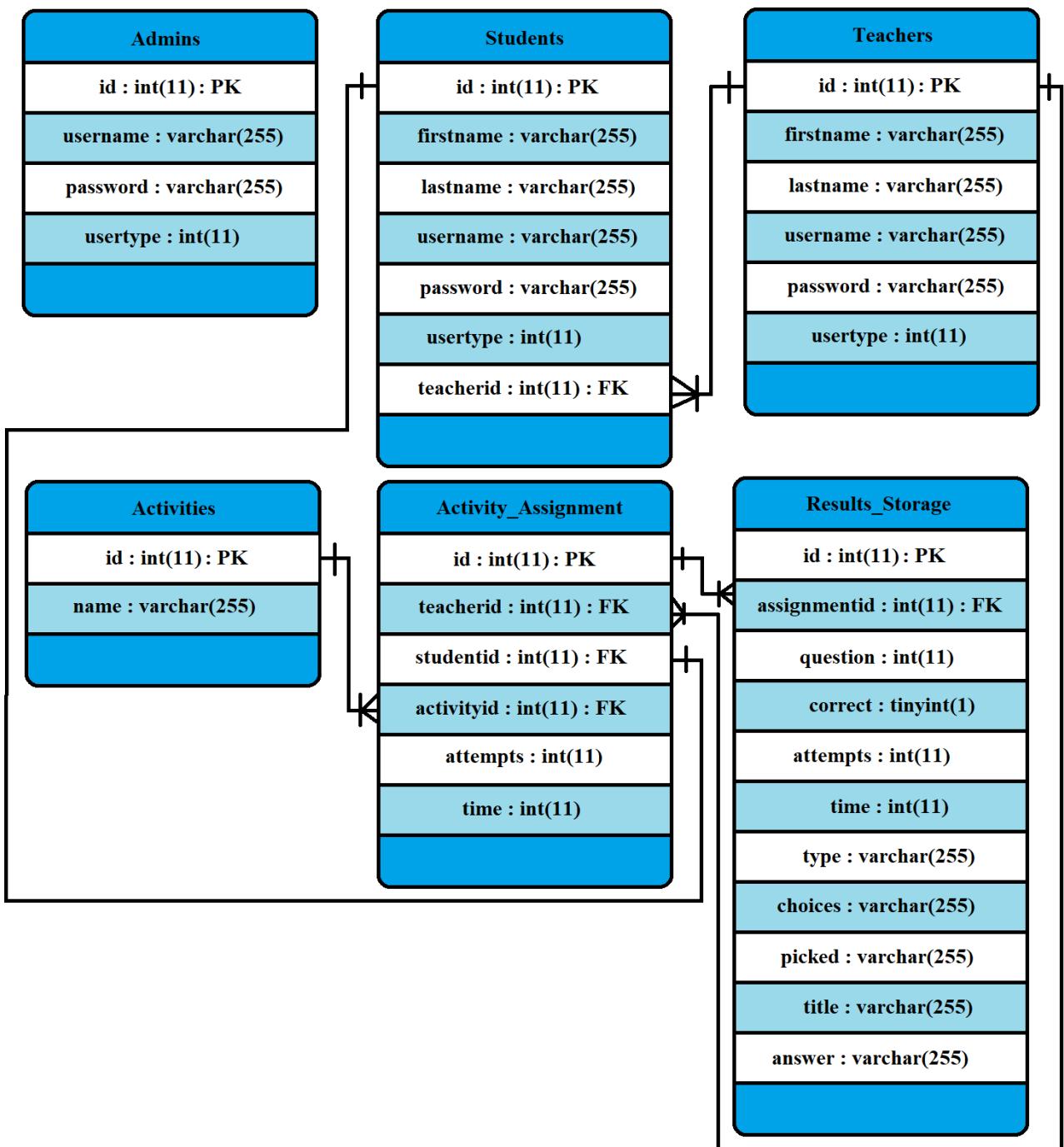


Figure 3.7: Database ER Diagram

Each table holds attributes which are properties associated with that table, for instance the activities table has a name attribute that is used to identify an activity's title. Certain attributes can be considered a primary or foreign key. Primary keys are used to uniquely identify every record within a database while foreign keys reference external primary keys. In the case of results\_storage, the "studentid" attribute is a reference to the primary key within the student database, this allows the system to know exactly which student the result is being stored for. The links between the tables represent their cardinality [Appendix C: Database – Figure 8] i.e. how the tables relate to one another. Students are allocated a single teacher, while teachers have multiple students.

ER diagrams allow a visual framework of the logical structure of a database that could be used as blueprints for building the system. Teachers will have access to multiple students, to achieve this, each student account will have a foreign key attached to it referring to their corresponding teacher. When a teacher assigns a student an activity, the activity database is referenced within the assignment to inform the system which activity requires monitoring. The teacher who assigned the task and the student who received it will also be stored as foreign keys so that the correct teacher can view the results and the correct student metrics are recorded. Once the student has completed the task, the results are stored with a foreign key referencing the assignment that the results belonged too. As the assignment table already holds information on the teacher and student identities, the results tables also gain access to this information. This would allow the system the ability to assign students tasks and relay the correct information back to the intended teacher.

### 3.2.4 Application Design

For the application design, I created multiple mock up design templates [Appendix D: Application Design Concepts] and tweaked them until the feedback I received from educators were positive. This segment aims to explain the design choices that prevailed and continued to the final product.

The three main devices that teachers and pupils could use the application on are desktops/laptops, tablets or smaller devices like mobile phones. I needed to ensure my application is appealing and functional regardless of screen size the end user is working with. I needed the application to work responsively, adapting itself to the current screen size and shifting content to better fit it's medium.

For the home page design, I decided to create it bright and vibrant to elicit excitement within the pupils. I also decided to preview the possible activities front and centre within the application so that pupils can delve right into their learning. I wanted these activities to be the focus, to achieve this I used minimal colouring on the surrounding aspects of the page naturally drawing the user's attention to the more distinct activity thumbnails. The navigation bar is initially grey, aside from the page the user is currently on, for similar intentions. However, once the user intends to navigate the website the nav bar changes each distinct link to a unique colour. This is to convey to the user that each link will navigate the user to a different page. The user can hover over each link which would then increase in size acting as a visual feedback to convey, without ambiguity, exactly which link the user is selecting. To accommodate younger children, I used universal icons within each link to also convey their functions, such as using a house icon to communicate a home button.



Figure 3.8: Desktop Homepage Layout

Figure 3.8 is the final design for the desktop homepage. Desktops tend to have large screen real-estate, to take advantage of this I created a well-spaced layout. The navigation bar is completely visible at the top of the page, allowing users immediate access to quick and easy exploration of the site. The large and bold banner welcomes the user to the site, highlights the site's function, learning, and acts as a barrier between the site's navigation bar and content. Similarly, the bottom panel acts as a barrier between the footer and provides more information regarding the application. This is least important to pupils and having it situated under the activities, shows the hierarchy of the different components. The asynchronous grid layout of the activities is not only appealing but also allows as much activities on the screen as possible.

This leads to a better user experience as they would require less scrolling to see all the contents within a page. An added benefit is that users could view all possible activities at once to consider all their choices before selecting.



Figure 3.9: Tablet Homepage Layout

Designing the tablet format of the homepages required adaptations to ensure user experience isn't diminished. The first modification is that the navigation bar is tucked into a hamburger menu. Tablet screens are smaller than desktops, by redesigning the navigation bar into a hamburger menu, minimal screen real-estate is used, allowing the screen to stay clean and less cluttered. Likewise, the banner has been shrunk to allow more immediate access to the web pages content. The activity grid maintains its asynchronous style; however, its proportions have been lessened to better comply with the size decrease. The information panel has shifted its content to be vertical to each other, should the original horizontal layout have been kept, the writing would have needed to be shrunk significantly causing it to become illegible. The footer also needed its contents to be shifted, moving the social links towards the bottom of the page rather than being inline with the rest of the footer contents. The social links are the largest element within the footer, choosing to shift them to their own line allows the rest of the content within the footer to stay consistent; consistency being a critical design principle and a concept at the forefront when designing these differences.



Figure 3.10: Mobile Homepage Layout

The mobile format continues many of the design changes introduced above, with a few minimal changes. Mobiles are one of the smallest screen sizes a user can use the application with, to maintain clarity, a font size increase was needed. Subsequently, the mobile banner needed to increase to accommodate the increase in font size. The grid lost its asynchronous style, opting to a simple vertical list. Should the grid have kept its original layout, the thumbnails would be too small to be usable and lose visual coherence. The list-like style allows users to both clearly differentiate between each individual item and provides them with enough space to select an item precisely, especially for younger children where dexterity can be an issue.

The design of the log-in system followed the rule of simplicity. Overwhelming users with sensory overload, causes frustration and leads to a poor user-experience. Login pages should be coherent requiring no cognitive strain. The page should be appealing to encourage users to want to access the deeper functionality of the website and the process should be quick and simple. A background of the night sky was chosen to evoke the feelings of calm and serenity within the user, this is especially vital when juxtaposed with the bright red error messages when the user makes a mistake. The background keeps the user feeling calm while the error messages lets the user know that they have made a mistake. To keep inline with simplicity, the login page is initially quite minimal, only showing the input fields required from the user and the submit button. Once, and only once, the user has made a mistake are error messages and suggestions given to the user. This keeps the screen less cluttered and only providing extra information when needed. The blue submit button contrasted with the white background allows it to strike out informing the user of its presence.

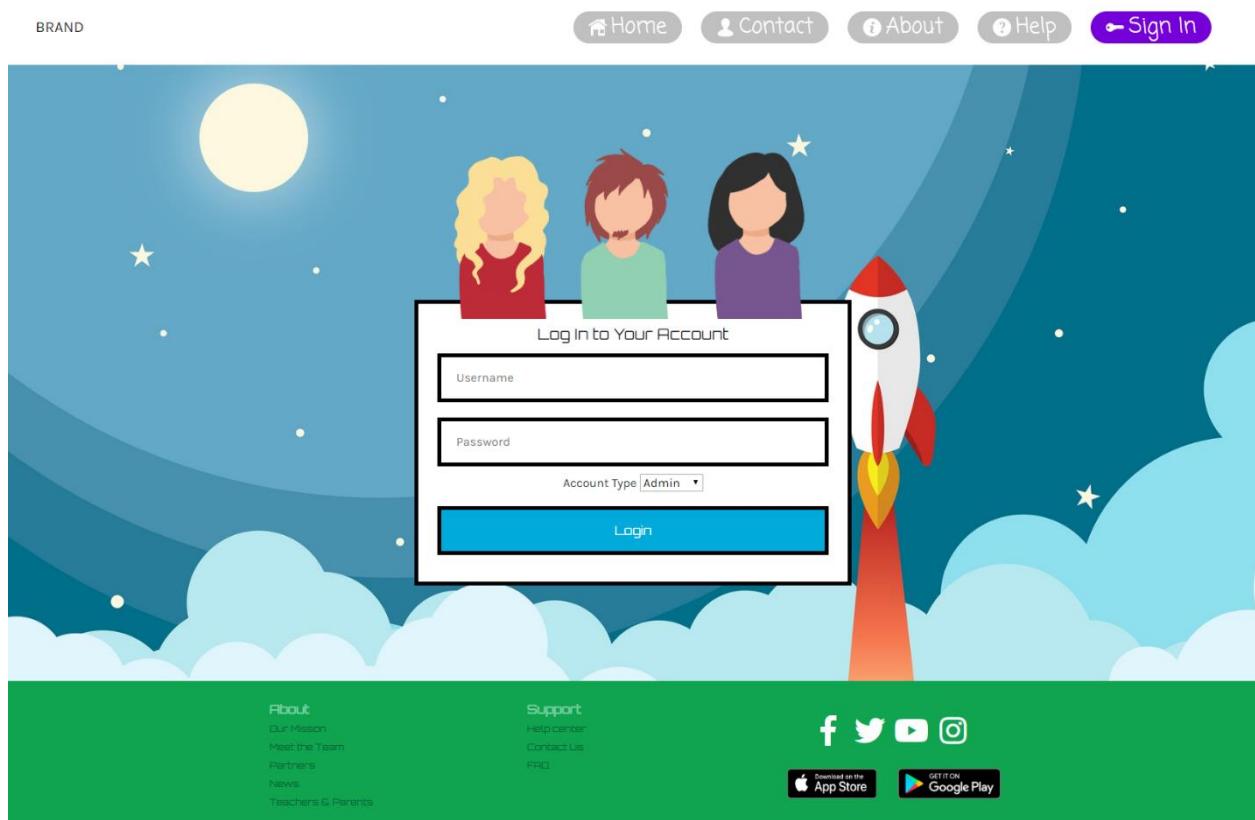


Figure 3.11: Desktop Login Page Layout

They layout for the desktop design is spacious and allows the form to breathe. The login fields are front and centre with a white background, the lack of colour contrasted with the background draws the user's eyes to it. The characters above the form visually represent that this is how the user can log into their unique account. The fields are clearly labelled so the user know what information is being asked of them. The placeholders are grey, whereas the rest of the writing black, subtly clues the user that the writing inside the fields aren't final. When the user does input into the fields, the writing becomes black, signifying that this will be used to check their credentials.

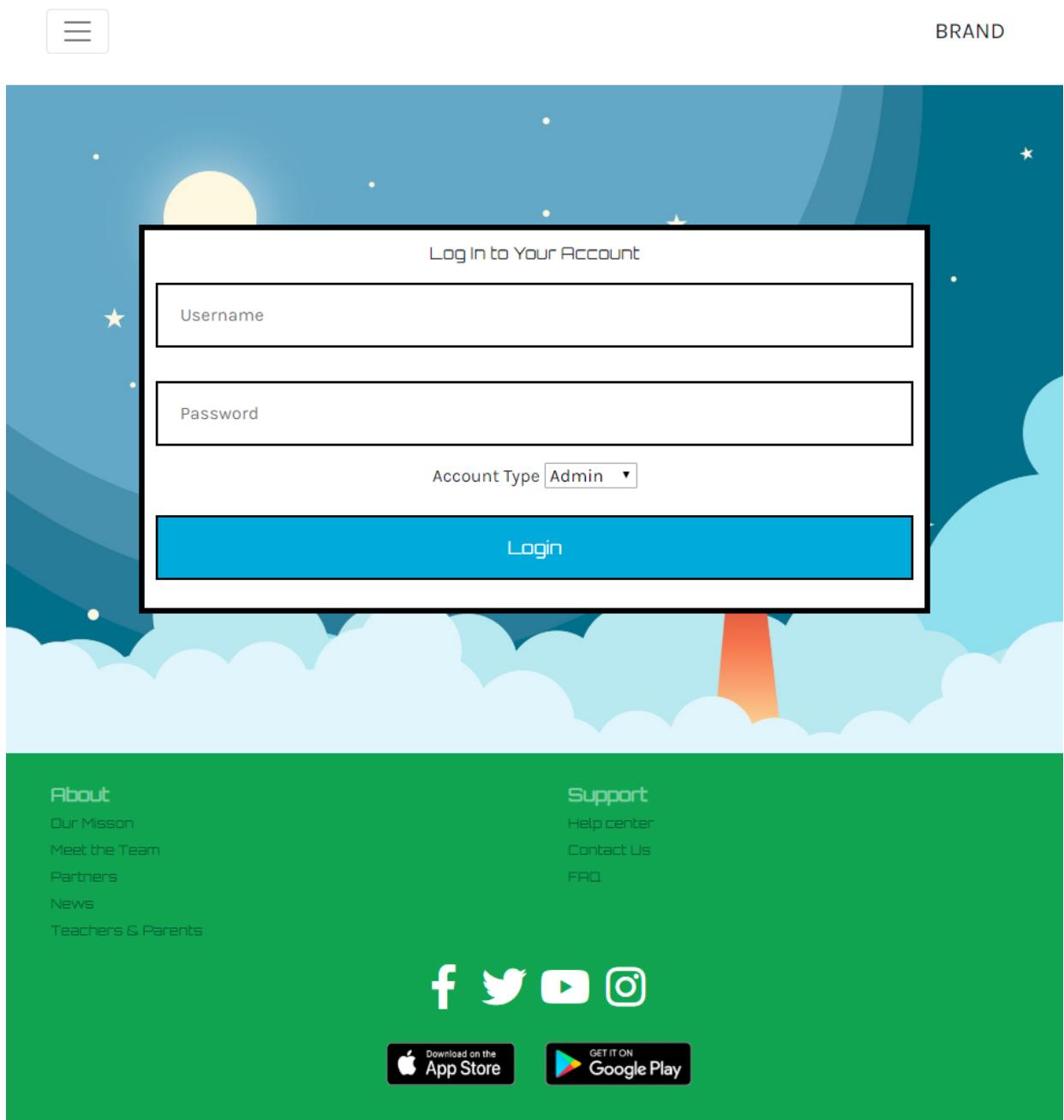
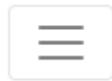


Figure 3.12: Tablet Login Page Layout

Continuing with the minimalistic design, the tablet design removes the characters from the screen to further declutter the page and keep things simple. The fields are more stretched out and larger to allow easier selecting on the smaller device.



BRAND

## Log In to Your Account

Account Type Admin ▾

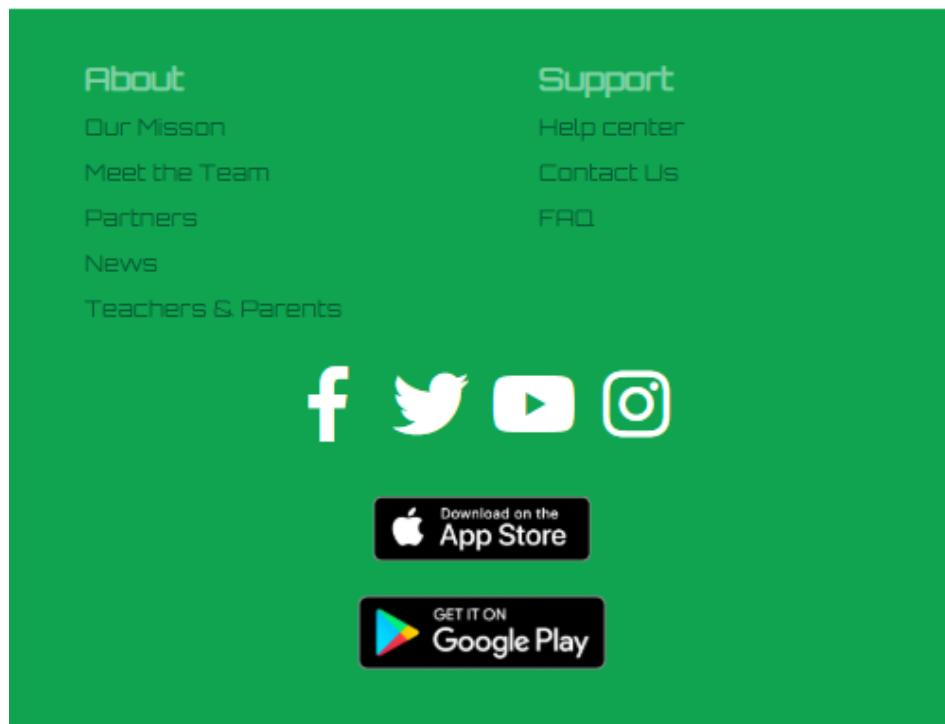


Figure 3.13: Mobile Login Page Layout

The mobile design completely cuts the background and border for the form, this cuts all distractions and makes the user input fields the first thing the user sees. The text has also been enlarged for better readability and the user input boxes have increased to accommodate.

Teachers have access to a dashboard where all their functionality is presented. Here functionality is given priority, design should not compromise the user's ability to fulfil their intended action. Teachers can modify their or their student's profiles, add new students to their class, assign assignments and view individual or classroom results. These functions are clearly displayed with unique colours to eliminate ambiguity. For further clarity, each function is given an icon to convey its intended use.

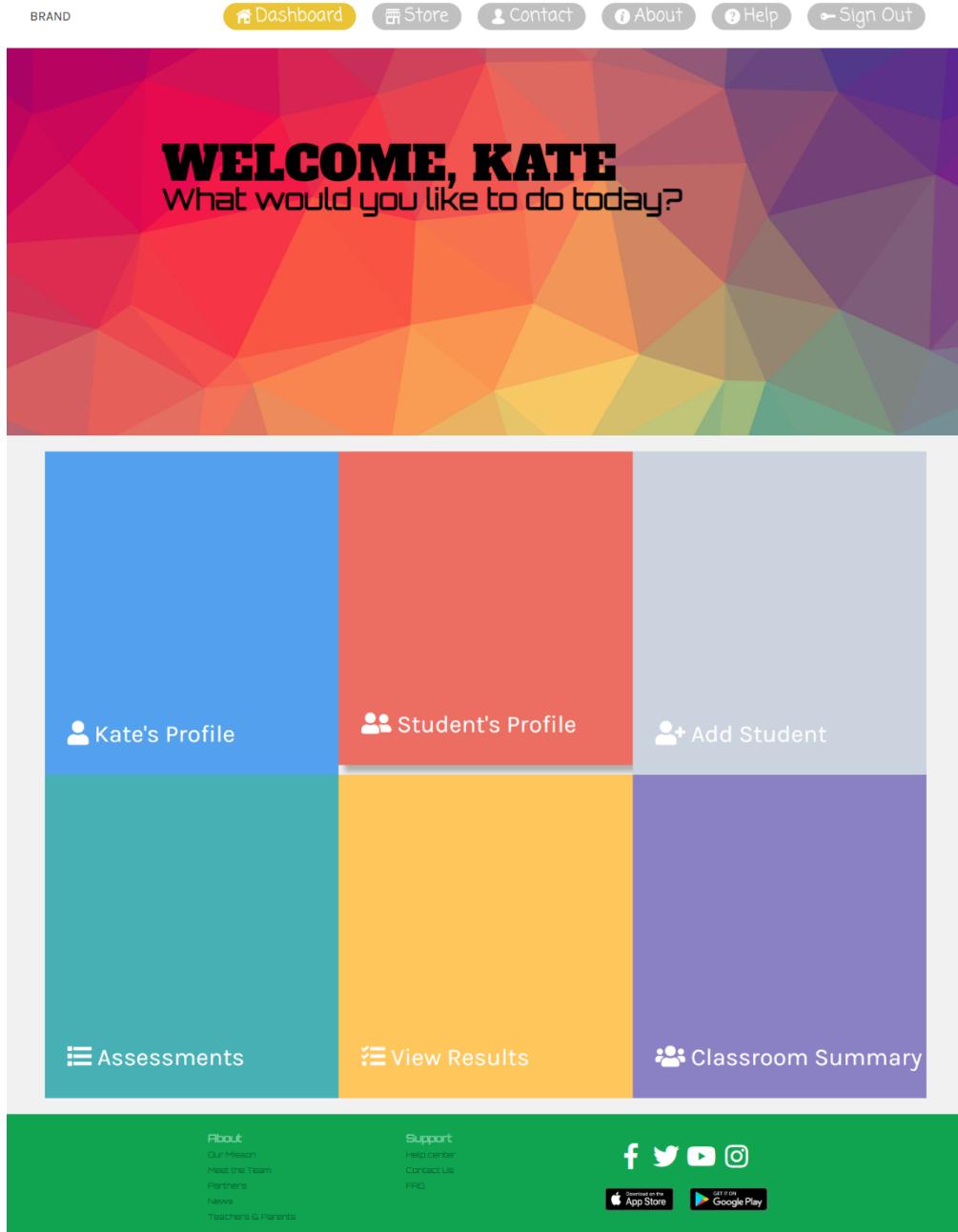


Figure 3.14: Desktop Dashboard Page Layout

The layout for the desktop dashboard adopts a clean grid structure, functionality that has to do with accounts above and functionality pertaining to assignments below. Each function is bulky to make use of the larger screen size, reducing as much white space as possible. Smaller icons would lead to the page looking emptier and imply less functionality so the larger icon size was decided.

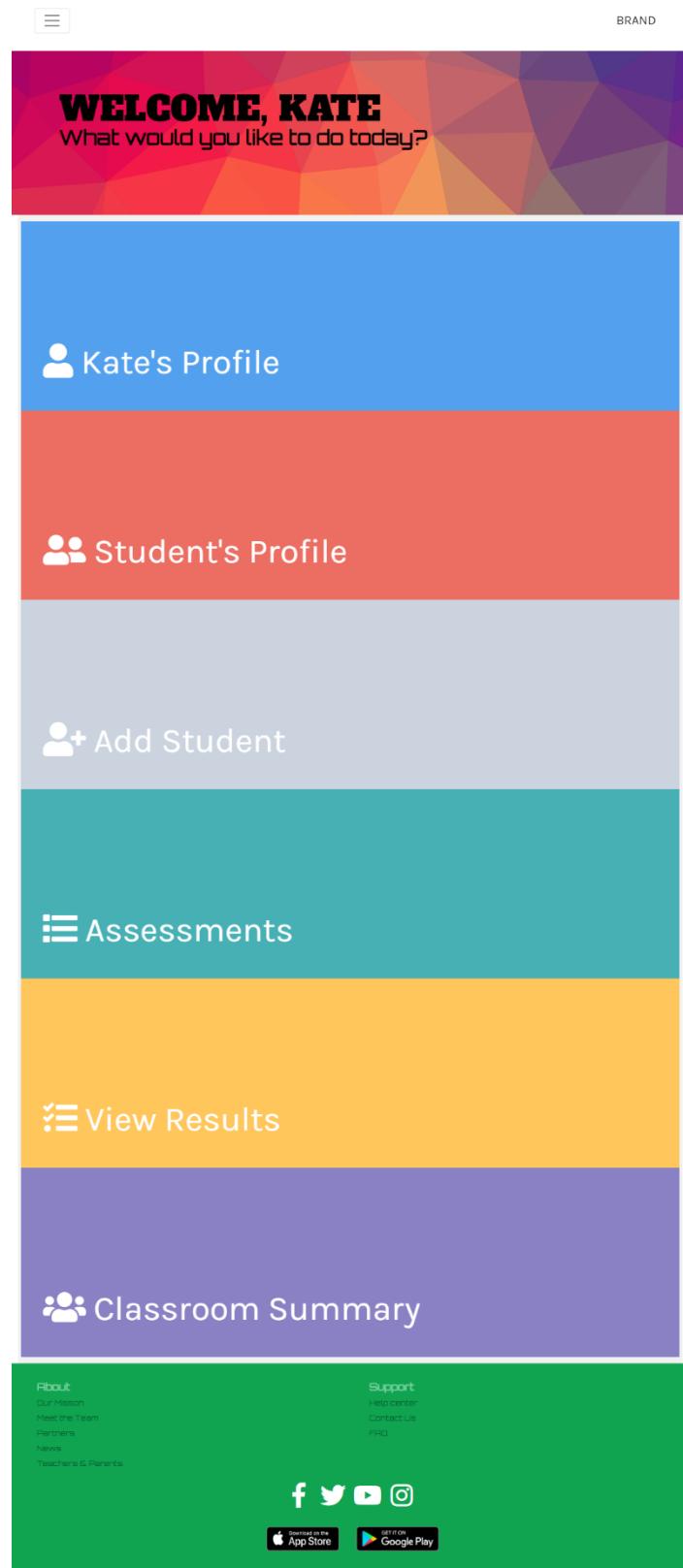
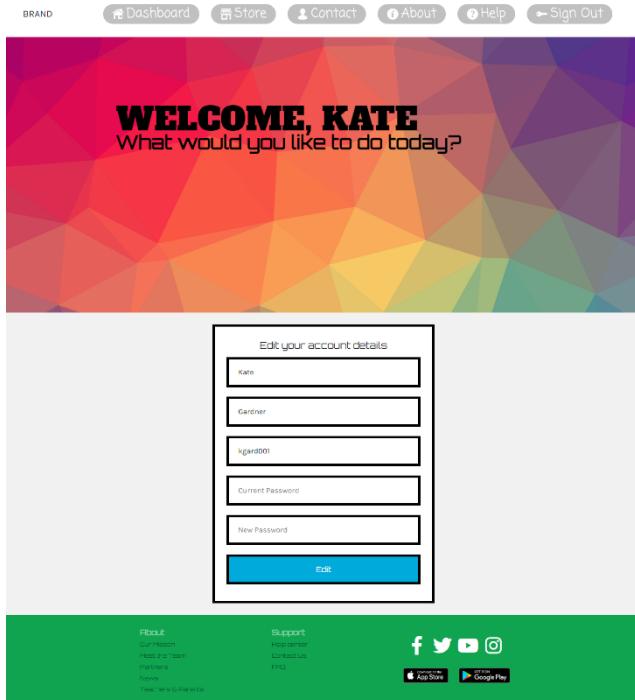


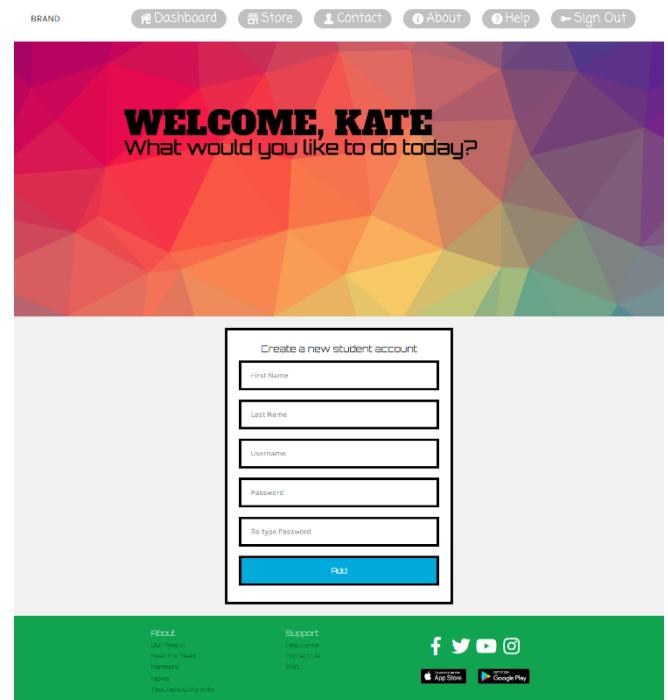
Figure 3.15: Tablet/Mobile Dashboard Page Layout

The tablet and mobile designs display the functions in a list like structure, a format better suited to smaller screens. Additionally, the icons and font sizes increased sizes. The general layout is kept, having the top three functions related to account usage and the bottom three for assignments and the colours are all kept consistent between the screen sizes.

The design sensibilities used for the login page are used for the teacher profile and add student functions. Keeping the page minimalistic and as uncluttered as possible, removing unneeded visual information as the screen reduces in size.



**Figure 3.16:** Desktop Teacher Profile Page Layout



**Figure 3.17:** Desktop Add Student Profile Page Layout

Both functions take in five inputs which is then used to either modify or insert into a database. In the case of teacher profile, existing data held is already filled into their corresponding field, the user is simply required to adjust the contents and click “edit” to save their new changes. For security purpose, the system requires the user to input their current password so that the application could verify it is indeed the user who intends to change their details.

The student profile page is similar because it also takes in input for the purpose of modification, however, the process is slightly different to improve user experience. Firstly, the user isn't required to input a password. Teachers have many pupils and it would be a difficult endeavour, if the system required teachers to input every student account password for modification and lead to much unneeded frustration. Furthermore, listing all students on a single page can, especially with larger classes, overwhelm the user and cause the page to feel cumbersome and unreadable. The process of pagination was decided to divide the list of students into discrete pages. Teacher can choose from a drop-down menu, the number of rows they wish to see on a single page resulting them to freely decide how they use the application. Further functionality allows them to decide if they want the list to be in ascending or descending order and which field they wish to order by. Once the fields are chosen, the page dynamically adapts to their criteria and the user can navigate through their students by page number.

Aida	Duarte	aduar001	New Password	Edit
Alisa	Juarez	ajuar001	New Password	Edit
Araceli	Pena	apena001	New Password	Edit
Booker	Phillips	bphil001	New Password	Edit
Brittney	Baxter	bbaxt001	New Password	Edit

1    2    3    4    5    6

Order by     Sort by     Number of rows

**About**

- [Our Mission](#)
- [Meet the Team](#)
- [Partners](#)
- [News](#)
- [Teachers & Parents](#)

**Support**

- [Help center](#)
- [Contact Us](#)
- [FAQ](#)

Figure 3.18: Desktop Student Profile Page Layout

The desktop design lists the student records in a row-like structure, the large screen allowing each field to be large enough to be legible and spaced out enough to keep things distinct. The padding between each row is kept minimal to accommodate to fit larger classes onto a single screen.

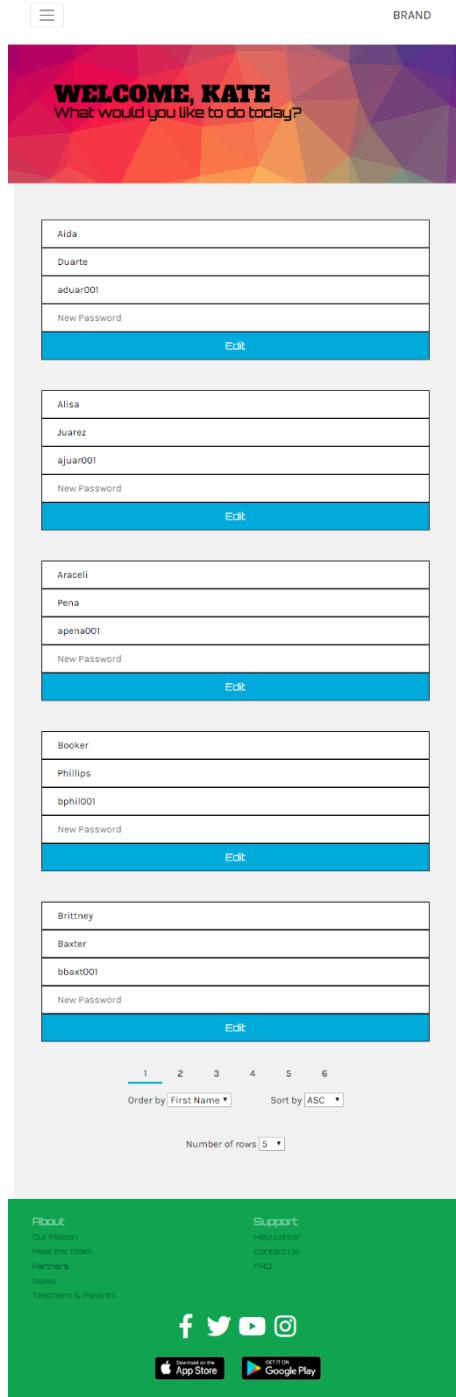


Figure 3.19: Tablet/Mobile Student Profile Page Layout

For the tablet and mobile designs, it was significantly adjusted to maintain user-experience. The row-like structure would no longer be adequate, and a list-like structure was adopted instead. To adequately differentiate between the records, further padding between records was required. The submit button was also adjusted to the size of other input fields to maintain design consistency and further act as a method of differentiating between the different records.

Similar design sensibilities were also used for the assignment page, with tweaks made for functionality purposes. Instead of a password field it is replaced with a drop-down; allowing the educator a way to assign pupils an activity.

Aida	Duarte	aduar001	None	<a href="#">Edit</a>
Alisa	Juarez	ajuar001	None	<a href="#">Edit</a>
Araceli	Pena	apena001	None	<a href="#">Edit</a>
Booker	Phillips	bphil001	None	<a href="#">Edit</a>
Brittney	Baxter	bbaxt001	None	<a href="#">Edit</a>

1    2    3    4    5    6

Order by     Sort by     Number of rows

**About**

- [Our Mission](#)
- [Meet the Team](#)
- [Partners](#)
- [News](#)
- [Teachers & Parents](#)

**Support**

- [Help center](#)
- [Contact Us](#)
- [FAQ](#)

Download on the App Store
 GET IT ON Google Play

Figure 3.20: Desktop Visualisation Page Layout, Student Tab

The visualisation aspect is split into two sections, a student and results tab. The student tab consists of listing all the pupils under the current logged in teacher and whether they have an assignment attached to them. The teacher can then view, if any results are on record, the visualised form of the metrics collected by the application within the results tab. Correctly designing the results tabs was vital, so large volumes of data are represented in easy to digest visual information. The application had to ensure the information it produces, is easy to understand, user-friendly and succinct. Bloating the user with needless information would

make extracting meaning difficult whilst on the other hand not showing enough would make results trivial and surface level.

The screenshot shows a desktop application interface for a learning platform. At the top, there is a navigation bar with links: Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a colorful, geometric background with the text "WELCOME, KATE" and "What would you like to do today?".

The main area is divided into two tabs: "Students" (selected) and "Results". The "Results" tab displays a student's performance details:

- Student ID:** wdunn001
- Attempts:** 13
- Time taken:** 02:56 to complete
- Score:** 2/10 correct

The results are presented in a grid format:

Question	Options	First Choice	Second Choice	Third Choice	Time Taken
1) Remember to _____ off the light	Switch, Sweetch, Swich, Swithce	Switch	Sweetch	Swich	01:02
2) Why did Michael drop his tray?	The floor was wet, The food was too hot, He didn't like the food, He tripped on a sign	The floor was wet	The food was too hot	He didn't like the food	00:49
3) Why was Jamila tired when she got to school?	She missed her bus, She decided to run to school, Her school was very far away, School is so boring	Her school was very far away	She decided to run to school	She missed her bus	00:01
4) I love my older _____	Brover, Brother, Brudduh, Bruvver	Brover	Bruduh	Brucker	00:01
5) A ___ LE	P, N, B, Y	P	N	B	00:01

Below the questions, there are two donut charts showing the percentage difference of question types answered correctly and incorrectly. At the bottom, there is a bar chart showing the average time taken for each question type:

Question Type	Average Time (approx.)
Spelling	15 seconds
Punctuation	2 seconds
Grammar	2 seconds
Understanding	24 seconds

At the bottom of the page, there are links for "About", "Contact", "Help", "FAQ", "Support", "Feedback", and "Resources & Parents". There are also social media icons for Facebook, Twitter, YouTube, and Instagram. Finally, there are download links for the App Store and Google Play.

Figure 3.21: Desktop Visualisation Page Layout, Results Tab

The final design showed the pupils attempt at each question during the activity. The page holds information such as; the pupils name, number of attempts, duration and percentage correct. Each question is listed in rows, with two dividing segments. The left holding information about the question, whilst the right holds metrics. The question segment holds; a title, choices and question-type. The metric segment holds; the first, second and third choices selected, and time taken on that question. The teacher can gauge at a glance whether the pupil got the question correct, through its border colour; green for correct and red for incorrect.

Displaying the choices picked by the pupils allows the teacher to delve deeper; for example, if a spelling section is incorrect, the educator can spot whether spelling patterns are not being remembered and instead phonetics are being incorrectly applied and thus they can support individual pupils with this specific issue. Likewise, duration to complete, allows educators to identify which section is taking too long and make conclusions to adapt subsequent teaching.

The two graph types chosen were pie (doughnut) chart and bar graph because these are most universally well-known and easy to understand. By presenting the data in these formats it makes interpreting meaning and spotting trends easy to accomplish. Doughnut charts show the contribution that different nominal (categorical) data contribute to an overall total and thus was chosen to demonstrate the percentage difference between the question types.

There will be Two doughnut charts side-by-side. The chart on the left will visually represent percentage difference between the question type the pupil got correct. Similarly, the chart on the right will visualise the same information for those they got incorrect. This information is useful for educators on many levels such as, supporting pupils, future planning, pupil progress meetings, etc, adhering to most of the requirements educators requested. The colour scheme of the doughnut-charts is consistent ensuring that each slice is distinct to avoid confusion and to facilitate comparison between the two. Below the chart, there is a legend to inform educators which colour corresponds to which slice. Educators can omit slices that may not be relevant, and the application will readjust, providing further freedom of use.

Bar graphs are used to compare numerical units between different categorical data, making it ideal for showing the time difference between the different question types. Bar charts are laid out so that the length of each bar can easily portray the differences between the different categories. In this case the larger the bar, the longer the pupil took on average with that question type.

Teachers can then use all the information presented to them by the application to make educated inferences. Regular assessments only inform the teacher the percentage of questions the pupil got correct, however this is inefficient because the data would not already be extrapolated making this tool a better choice for assessment.

Each aspect of the application supports educators to assess and make inferences. Educators can easily see, for example, questions that a pupil got incorrect followed a pattern of being spelling questions, they can then further delve into aspects such as punctuation questions were all answered correctly, however, the pupil took far longer than their peers. The teacher could then ascertain that to improve the pupil understand, they require work on spelling and further practice on punctuation to improve confidence and stamina.

### 3.2.5 Activity Design

Like the application design process, the activity design went through various iterations [[Appendix E: Activity Design Concepts](#)]. This segment aims to explain the design choices that prevailed and continued to the final product.

Through researching children app design and dialogue with educators, the overall message was, in order to keep pupils motivated, the activity needed to be visually appealing, encouraging and interactive.

A bright purple colour scheme was chosen to elicit a happy and positive mood and reduce stress that may arise from answering difficult questions. The activity contains essentially five main screens, a landing page, a title screen, a question screen, a feedback screen and a results page.

A silent landing page was created to comply with chrome's development policy; stating that for a webpage to be granted access to sound, some form of user action is required. This is to avoid web pages exploiting users. The user is required to click the start button, granting the activity permission to play sound and leading into the activity's real starting page where sound is played throughout.



Figure 3.22: Activity Launch Page

Research indicates children often expect apps to contain interaction. To show pupils that this is an interactive application, the landing page starts immediately with a large star that spins and pulsating “start” message enticing the user to click. Introducing motion during the landing page and sound immediately after the user clicks, signals that the application does have these features and that pupils have some control over these factors.

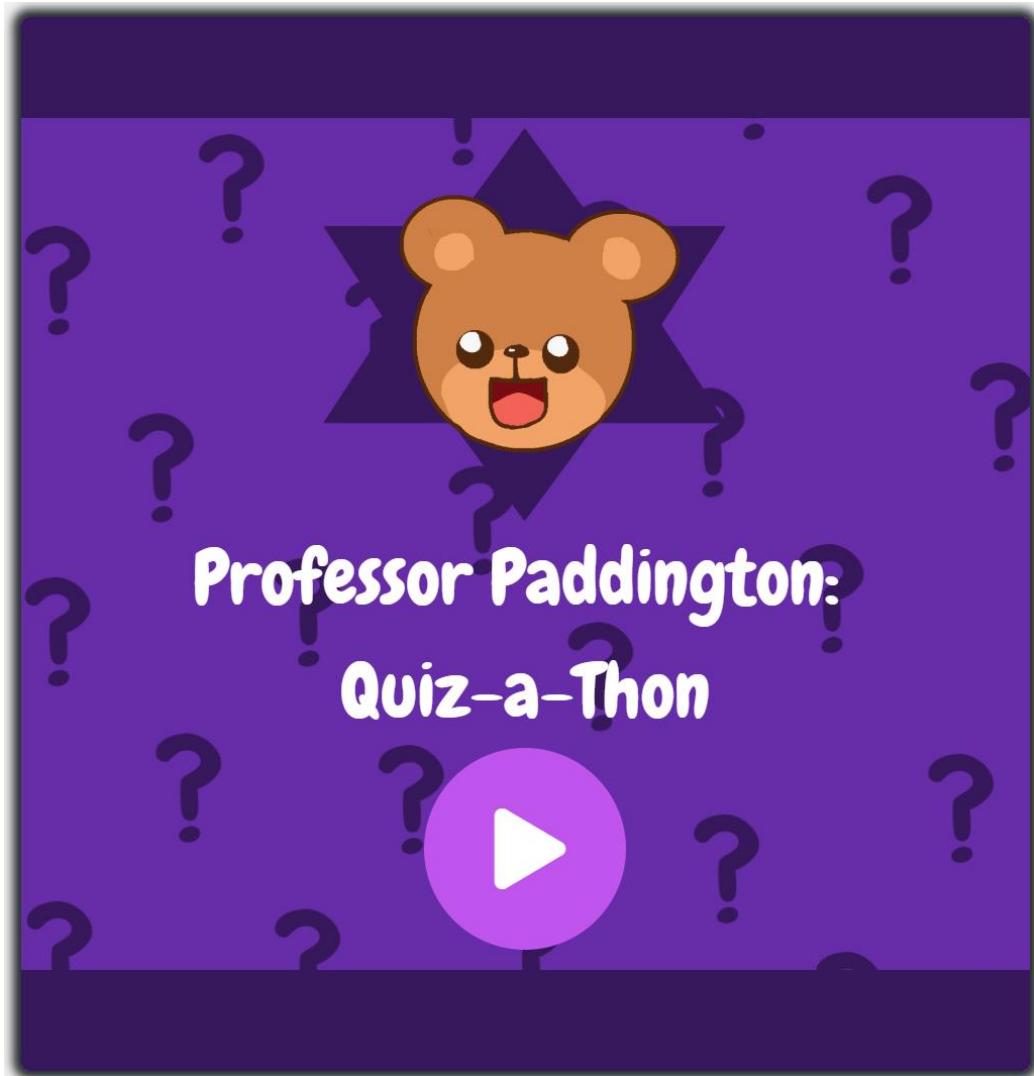


Figure 3.23: Activity Title Page

Once the user is sent to the title page, the screen’s main segments slide into place and the start button shakes to entice clicking. Positive music is played to provoke an atmosphere to encourage the user to know that, while this may be an assessment it should not be stressful. The play button has the universal icon for “playing” to let the user know its function without text. The button, once hovered, becomes a darker shade and increases in size as further evidence that it is interactable.

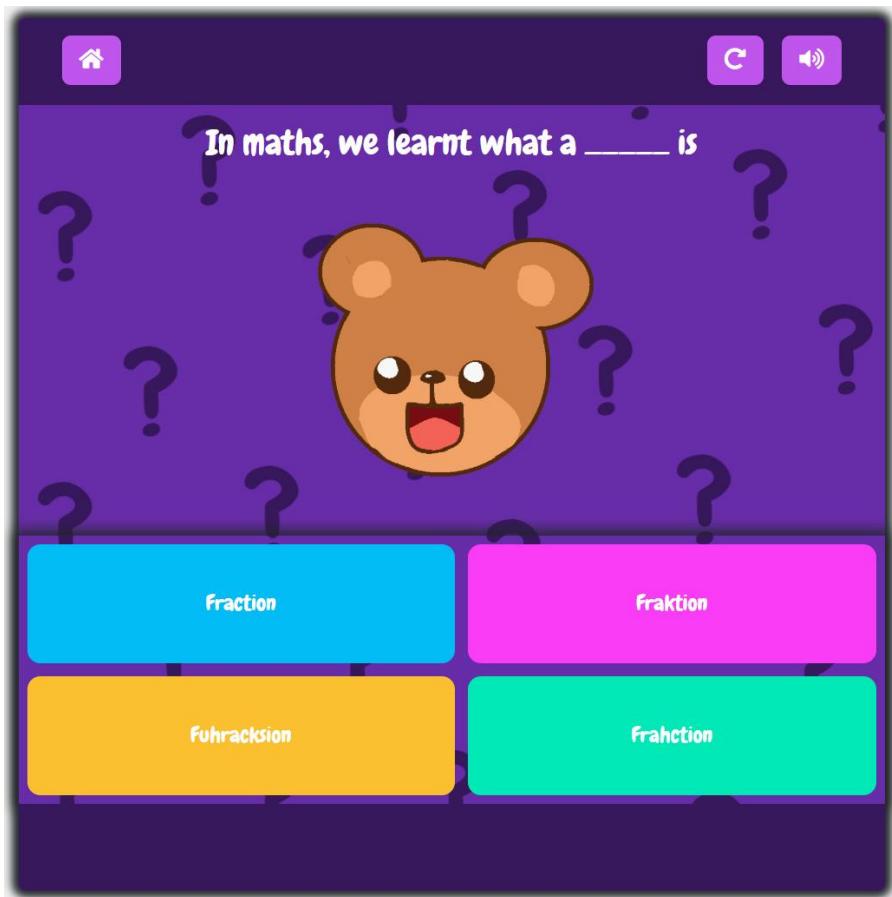


Figure 3.24: Activity Quiz Page, Happy Bear

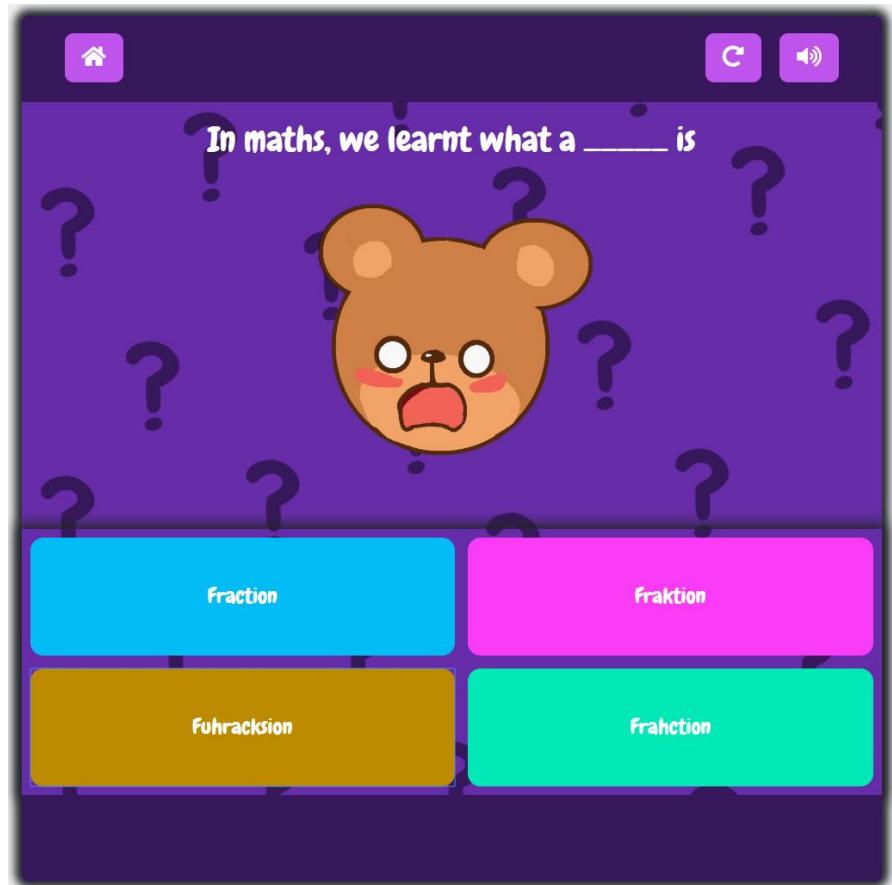


Figure 3.25: Activity Quiz Page, Sad Bear

Once the user enters the application, they are greeted with a set of questions, the application begins recording metrics here in order to be used for visualisation. In terms of what the user sees, they are greeted with a question and four multiple choices. A text-to-speech bot reads out the question and the pupil can select an answer. A shorthand visual question is shown on screen to remind the user what they are required to answer, however the speech-to-text voice is where most of the explanation is situated. The same bear cartoon found within the title screen is also displayed, allowing pupils to attribute the voice to this character. Should the pupil select an incorrect answer, the bear's cheerful expression changes to a confused face and the bear encourages them to try again pushing the pupil to work at their best. Additionally, it acts as feedback that the interaction was successful, to further facilitate this, a short sound-bite is also played.

The multiple-choices are not only bright to be appealing, but the colours contrasts well with the purple background. This allows the options to be distinct from one another and not blend in to the background allowing users to know they are interactable. Further measures were taken by darkening each option when hovered. At the top of the activity, the user is given three options to improve user experience. The user can go back to the title page of the activity, repeat the current question or mute all sound. Each option is given the universal icons for their function to convey their meaning. In the case of sound, the icon changes depending on its current state, when sound is currently playing then the icon is a speaker emanating sound, however, once the icon is activated and sound is no longer playing the icon is changed to a speaker with a cross. Throughout the activity the bear icon sways not only keeping pupils engaged, but also act as a visual indicator that the activity is still running as pupils tend to equate lack of movement with lack of function. The background music chosen for this section is an upbeat yet suspenseful tune, so that pupils understand that careful considering is required during these segments. The pupil is transported to a feedback page through either one of two ways, selecting correct answers or selecting incorrect answers thrice in a row. This feedback page is slightly modified depending on which method was used to arrive there.

If the pupil correctly answers, a positive screen is displayed, exclaiming a job well done to reward them for their efforts. The music and bear are joyful to encourage the child to continue working hard.



Figure 3.26: Activity Transition Page, Well Done

Contrarily, pupils who get three incorrect answers are encouraged to try harder. Sombre melancholic music is played with the bear's confused expression, allowing the pupil time to reflect why they may have gotten the answer wrong and allow them time to overcome and continue with their assessment.

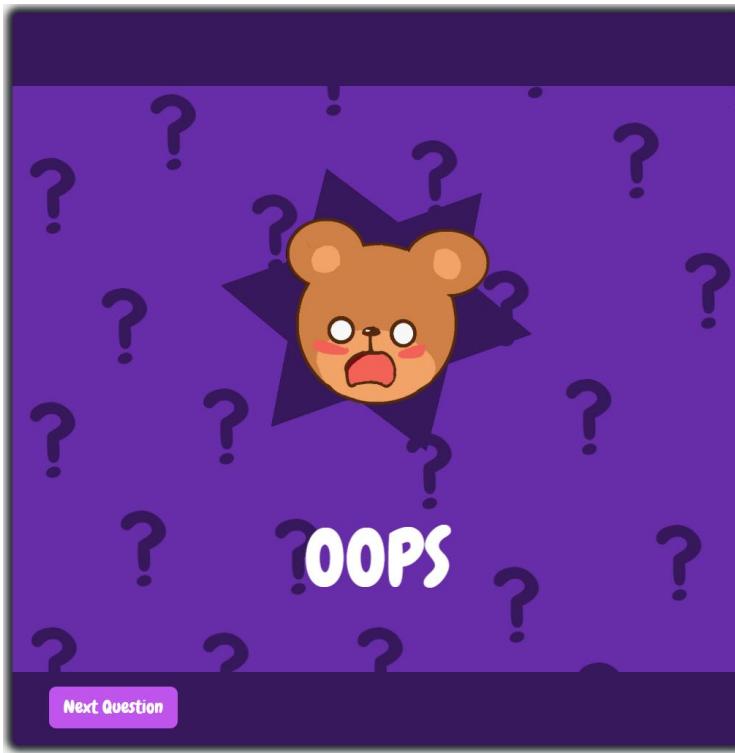


Figure 3.27: Activity Transition Page, Oops

Once the assessment is complete, users can see a results screen with a summary of their assessment, a red cross, representing a failed question and a green tick, representing a passed one. The colours were selected as they are universally known for having negative and positive connotations. Pupils can see, from a glance, how well they did by determining whether the colour green or red appears more. This page also allows the application, in the background, to collect the pupil's metrics for storing. An alert box is used to notify the user if their results storage was successful or if an error had occurred. Depending on the user's results, a well-done message is played to reward them for their efforts, or a reassuring message encouraging them to try and improve.

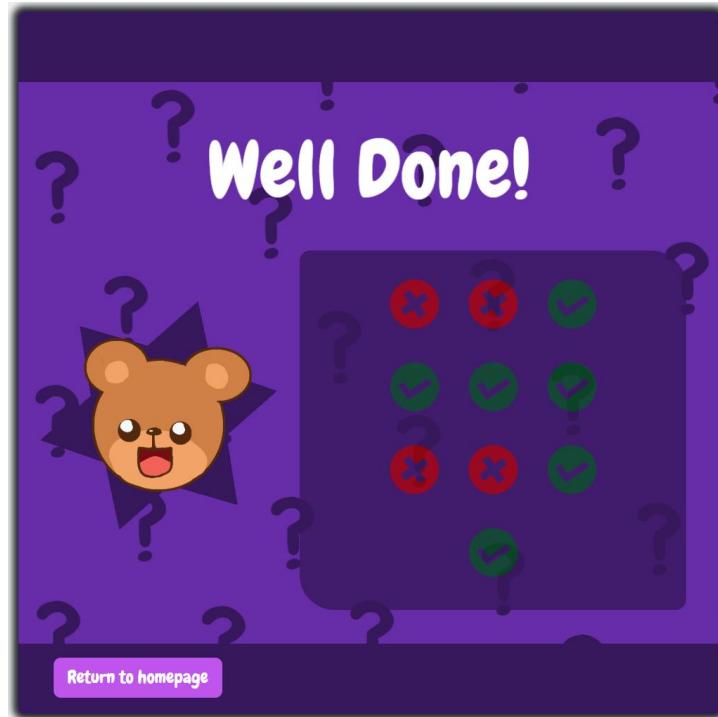


Figure 3.28: Activity Results Page, Well Done

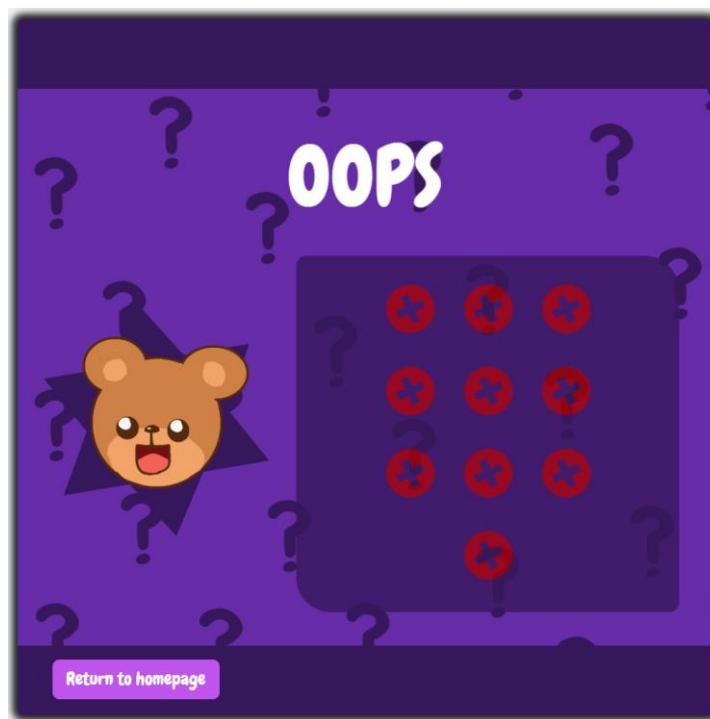


Figure 3.29: Activity Results Page, Oops

# Implementation

This segment explains the implementation of the project; relevant code snippets are shown throughout to supplement explanation. The full code will be provided in the appendix of the report [Appendix F: Code].

## 4.1 HTML, CSS and JavaScript Development

### 4.1.1 External Framework

The external frame work, Bootstrap, was used to assist with development and enhance the user experience. The project heavily utilised Bootstrap's grid layout system, a flexible twelve column system that automatically re-arranges content depending on the current screen size. During the design process three layouts were designed for each page, one for desktop, tablet and mobile, Bootstrap allowed for the application to seamlessly shift into these layouts without requiring any input from the user.

```
<div class="row" id="english">
    <div class="col-sm-6 thumbnail">
        <a href="activity-page.php?activityID=1"></a>
    </div>
    <div class="col-sm-6 thumbnail">
        <a href="activity-page.php?activityID=1"></a>
    </div>
</div>
<div class="row">
    <div class="col-sm-8" id="maths">
        <div class="row">
            <div class="col-sm-6 thumbnail">
                <a href="activity-page.php?activityID=1"></a>
            </div>
            <div class="col-sm-6 thumbnail">
                <a href="activity-page.php?activityID=1"></a>
            </div>
        </div>
        <div class="row">
            <div class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1"></a>
            </div>
            <div class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1"></a>
            </div>
            <div class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1"></a>
            </div>
        </div>
    </div>
    <div class="col-sm-4 thumbnail" id="science">
        <a href="activity-page.php?activityID=1"></a>
    </div>
</div>
```

Code 4.1: Index Page Activity Grid

Code 4.1 demonstrates how the grid system is used. Each “row” consists of 12 columns, which can be grouped together depending on the situation, and given rules that determine their behaviour. An example of this is using the classes “col-sm-12” and “col-lg-6” to direct the system to allocate 6 columns for an item when the screen is large but shift the contents to 12 columns when the screen is smaller. Other aspects of bootstraps were also used such as their carousel system, HTML content and CSS code.

#### 4.1.2 General Application

Generally, the application's main framework was written in HTML, providing the structure of the web pages and CSS, determining their visual layout.

```
<div class="dashboard-container">
  <div class="row">
    <a class="col-x1-4 thumbnail dashboard-button" href="teacher-profile.php">
      <p class="dashboard-content">
        <i class="fas fa-user dashboard-icon"></i> Teacher Profile
      </p></a> <a class="col-x1-4 thumbnail dashboard-button" href="student-profile.php">
      <p class="dashboard-content">
        <i class="fas fa-user-friends"></i> Student's Profile
      </p></a> <a class="col-x1-4 thumbnail dashboard-button" href="add-student.php">
      <p class="dashboard-content">
        <i class="fas fa-user-plus"></i> Add Student
      </p></a>
    </div>
    <div class="row">
      <a class="col-x1-4 thumbnail dashboard-button" href="give-assignment.php">
        <p class="dashboard-content">
          <i class="fas fa-list"></i> Assessments
        </p></a> <a class="col-x1-4 thumbnail dashboard-button" href="view-results.php">
        <p class="dashboard-content">
          <i class="fas fa-tasks"></i> View Results
        </p></a> <a class="col-x1-4 thumbnail dashboard-button" href="view-class-results.php">
        <p class="dashboard-content">
          <i class="fas fa-users"></i> Classroom Summary
        </p></a>
      </div>
    </div>
</div>
```

Code 4.2: Dashboard Page

Code 4.2 demonstrates how the content is structured. There are six links, leading to their own specific page, each link having a name and icon.

```
.thumbnail {
  top: 0;
  transition: top ease .5s;
  padding: 3px;
}
.thumbnail:hover {
  top: -15px;
  box-shadow: 10px 10px 8px #b0b0b0;
  cursor: pointer;
  padding: 0px;
}
.dashboard-button {
  height: 500px;
}
.dashboard-content {
  font-size: 2.5vw;
  position: absolute;
  bottom: 20px;
  left: 35px;
  color: #fff;
}
```

Code 4.3: Dashboard Page CSS

Each HTML component can be assigned classes or ids that corresponds to stylings found in a style sheet. Code 4.3 corresponds to the styling for Code 4.2, the example demonstrates, an assigned fixed height to the dashboard buttons of 500px, positioning of each buttons label to the bottom left, the font is white with a size of 2.5 view width. View width takes the current width of the user's screen and scales the text to 2.5 units. Additionally, the buttons are set so that when the user hovers over them, the button is eased 15 pixels higher.

Media queries denoted by @media allows further utility in responsiveness by applying different CSS traits depending on the user's screen size.

```
@media (max-width:1300px) {
    .dashboard-button {
        height: 400px;
    }
}
@media (max-width:1200px) {
    .dashboard-button {
        height: 300px;
    }
    .dashboard-content {
        font-size: 5vw;
    }
}
@media (max-width:575.5px) {
    .dashboard-button {
        height: 150px;
    }
    .dashboard-content {
        font-size: 6vw;
        bottom: 5px;
        left: 15px;
    }
}
```

**Code 4.4: Media Queries**

Code 4.4 changes the CSS applied to dashboard.php depending if the view width is 13,000 px, 12,000px or 575.5px, through reducing the size of the dashboard buttons to better fit in line with the reduced screen real-estate.

Another aspect of CSS is animations, allowing simple animations to inject personality to a webpage.

```
@keyframes spin {
    0% {transform: rotate(0deg);}
    100% {transform: rotate(359deg);}
}
@keyframes slide-in {
    0% {left: 0px; top: -500px;}
    100% {left: 0px; top: 0px;}
}
.transition-star {
    position: relative;
    top: 100px;
    left: 0;
    animation: spin 5s infinite linear;
}
.titlescreen-star {
    position: relative;
    top: 0;
    left: 0;
    animation-name: slide-in;
    animation-duration: 1s;
}
```

**Code 4.5: Keyframe Animation**

Code 4.5, taken from the activity page, shows two keyframe animations title “spin” and “slide-in”. Spin rotates content 360 degrees while slide-in translates content downwards. Each animation is then attached to a component, transition and title screen star respectively. Each animation is given properties at the time of assignment, allowing components that share animations to be unique.

### 4.1.3 Quiz Activity

The quiz activity was created using HTML and CSS to determine visual characteristics and JavaScript to dictate behaviour. The code referenced at [20] was used as a foundation and built upon.

```
<div class="hide" id="title-page">
    <div class="top-bar"></div>
    <div class="titlescreen-page">
        
        <div class="bear-icon-title"></div>
        <div class="titlescreen-title">
            Professor Paddington:<br> Quiz-a-Thon
        </div>
    </div>
    <button id="title-btn"><i class="fas fa-play"></i></button>
    <div class="bottom-bar"></div>
</div>
<div class="hide" id="activity">
    <div class="top-bar">
        <button id="home-btn"><i class="fas fa-home"></i></button>
        <button id="repeat-btn"><i class="fas fa-redo-alt"></i></button>
        <button id="mute-btn"><i class="fas fa-volume-up" id="mute-btn-icon"></i></button>
    </div>
    <div class="h1" id="question"></div>
    <div id="bear-icon-activity"></div>
    <div class="choices" id="choices">
        <button class="choice1"></button>
        <button class="choice2"></button>
        <button class="choice3"></button>
        <button class="choice4"></button>
    </div>
    <div class="bottom-bar"></div>
</div>
```

Code 4.6: Activity Page HTML

HTML laid groundwork for the structure of the quiz, the code snippet demonstrates the skeleton for the title and activity screens. Each section only contains the essential frame for their respective screens, these frames would then later be expanded upon through JavaScript. Each section is then styled, adding background images, colour and animation. Each screen is hidden using CSS and called when necessary. This is achieved through giving each component a unique id, thereby allowing direct referral back to the specific component through that unique id.

```
const startContainer = document.getElementById('title-page');
const activatePage = document.getElementById('start-page');
const quizContainer = document.getElementById('activity');
const endContainer = document.getElementById('end-page');
const startBtn = document.getElementById('title-btn');

function titleScreen(e) {
    startContainer.classList.remove('hide');
    activatePage.classList.add('hide');
    quizContainer.classList.add('hide');
    endContainer.classList.add('hide');
    startBtn.addEventListener('click', startQuiz);
}

function startQuiz() {
    startContainer.classList.add('hide');
    quizContainer.classList.remove('hide');
}
```

Code 4.7: Activity Pages Initialisation

Within the JavaScript file, it is possible to store each component into a variable to be manipulated later. Code 4.7 shows the script storing the title page, start page, activity screen, results page and start button into their own variables. Within each screen, all unnecessary pages are kept hidden until needed. This process is used throughout the quiz to create a seamless experience for the user and simulate a single page activity.

The questions for the quiz are stored within an array and each question is its own object. Each question has four properties; a visual question that will be displayed on screen, an audible question that will be fed to a text-to-speech API, four multiple choices and the question type itself.

```
const quiz = [
    {
        visualQuestion: 'B   N   N   ',
        audableQuestion: 'What letter fits to complete the word, Banana',
        options: [
            {choice: 'A', correct: true},
            {choice: 'N', correct: false},
            {choice: 'B', correct: false},
            {choice: 'Y', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'A _ _ L E',
        audableQuestion: 'What letter fits to complete the word, Apple',
        options: [
            {choice: 'P', correct: true},
            {choice: 'N', correct: false},
            {choice: 'B', correct: false},
            {choice: 'Y', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'I love my older ____',
        audableQuestion: 'Choose the correct spelling, I love my older brother',
        options: [
            {choice: 'Brother', correct: true},
            {choice: 'Brudduh', correct: false},
            {choice: 'Bruvver', correct: false},
            {choice: 'Brovor', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'Remember to _____ off the light',
        audableQuestion: 'Choose the correct spelling, Remember to switch off the light',
        options: [
            {choice: 'Switch', correct: true},
            {choice: 'Sweetch', correct: false},
            {choice: 'Swich', correct: false},
            {choice: 'Switche', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'In maths, we learnt what a ____ is',
        audableQuestion: 'Choose the correct spelling, we learnt what a fraction is',
        options: [
            {choice: 'Fraction', correct: true},
            {choice: 'Fraktion', correct: false},
            {choice: 'Fuhracksion', correct: false},
            {choice: 'Frahction', correct: false}
        ],
        type: 'Spelling'
    }
]
```

**Code 4.8: Activity Quiz Array**

To display the questions, a display quiz function was created, that first strips the quiz screen of all its existing content, removing the previous question shown. It then takes the pseudo randomly shuffled quiz array and displays the next available question.

```
function displayQuiz (e) {
    while (choicesContainer.firstChild) {
        choicesContainer.removeChild(choicesContainer.firstChild);
    }
    questionContainer.innerText = e.visualQuestion;
    shuffledAnswers = e.options.sort(function() {return 0.5 - Math.random()});
    textToSpeech(e.audableQuestion);
    let i = 1;
    shuffledAnswers.forEach(function(option) {
        const button = document.createElement('button');
        button.innerText = option.choice;
        questions[currentQuestion].choicesList.push(option.choice);
        let btnClass = 'choice' + i;
        button.classList.add(btnClass);
        if(option.correct){
            button.dataset.correct = option.correct;
            questions[currentQuestion].correctChoice = option.choice;
        }
        choicesContainer.appendChild(button)
        i++;
    })
}
```

**Code 4.9: Activity DisplayQuiz Function**

The SpeechSynthesis API [21] reads the question out to the user. The API takes a single string and outputs it through speech. This can communicate further instructions without overcomplicating the user interface and displaying large walls of text

```
function textToSpeech(e) {
    if ('speechSynthesis' in window && !mute) {
        synth.cancel();
        speech.text = e;
        synth.speak(speech);
    }
}
```

**Code 4.10: Activity TextToSpeech Function**

Once the user clicks a choice, the selectAnswer function is called to validate. This checks whether the choice had the dataset “correct” attached to it, if so, then the user is taken to the positive transition screen, however, if the user is incorrect, they are allowed three incorrect attempts before transferring to a negative transition screen.

```
function selectAnswer(e) {
    const selected = e.target;
    if (selected.dataset.correct) {
        transitionScreen();
    } else {
        setTimeout(resetBear, 1500);
        if (currentAttempts == maxAttempts) {
            transitionScreen();
        }
    }
}
```

**Code 4.11: Activity SelectAnswer Function**

Throughout, metrics relating to the usage of the application, are recorded. These metrics are the amount of attempts the pupil took on a question, what those attempts were, the time taken, the question’s type, title and choices as well as whether they got the question correct.

Once the user reached the end, an ajax method is called to collect and store these metrics into a database.

```
$.ajax({
    type: 'POST',
    url: 'inc/results-submit.inc.php',
    data: {
        data: questions,
        time: overallTimeTaken,
        activityid: activityid,
        submit: 'submit-results'
    },
    success: function(data) {
        alert(data);
    }
});
```

**Code 4.12: Activity Ajax Function, Data Submit**

Ajax makes a post request to submit data, meaning information is kept hidden from the URL, to the results submit php file. Upon success, the method waits for a response back and alerts the user. The response is either a success message, or an error message informing them that a problem was raised with potential solutions to rectify it.

#### 4.1.4 Results Visualisation

The results visualisation page follows a similar structure to the quiz activity, using HTML and CSS to determine visual characteristics and JavaScript to dictate behaviour. The pupil and visualisation tab are both hidden by default and shown when appropriate. Once the teacher has selected to view a pupil's results, a visualisation function is called which attempts to retrieve the data associated with the pupil and display them on screen.

To retrieve the data, an ajax method is called, that sends over the username of the pupil, if applicable, and the type of data requested.

```
function visualisation(username, typeOfData) {
    $.ajax({
        type: 'POST',
        url: 'inc/view-results.inc.php',
        data: {username: username, typeOfData: typeOfData, submit : 'submit'},
        beforeSend: function(){
            displayVisualisation();
        }
    });
}
```

**Code 4.13: Visualisation Ajax Function, Data Retrieval**

Once data is received, the method begins readying the data to be displayed. Data stored within the database first need to be handled and converted to useable information, an example being converting time to an easy to understand format. Code 4.14 shows pupil time being converted to a more interpretable format, percentage correct being calculated and the choices picked during each question neatly packaged.

```

let minutes = Math.floor(storedData[(storedData.length - 1)] / 60);
let seconds = storedData[(storedData.length - 1)] - minutes * 60;
let value = '';
if (minutes < 10) {
    value += '0';
}
value += minutes + ':';
if (seconds < 10) {
    value += '0';
}
value += seconds;
studentTime.innerHTML = value + '<br> to complete';

let percentage = 0;
for (let i = 0; i < (storedData.length - 2); i++) {
    if (storedData[i].correct == 1) {
        percentage++;
    }
}
studentScore.innerHTML = percentage + '/10 <br> correct';

let picked = storedData[i].picked.split(',');
let choicesLabel = ['First Choice:', 'Second Choice:', 'Third Choice:'];
let pickedCount = 0;
for (let j = 0; j < picked.length; j++) {
    dataContainer[i].getElementsByClassName('picked')[j].innerHTML = choicesLabel[j] + ' ' +
picked[j];
    pickedCount = j + 1;
}
for (let j = pickedCount; j < 3; j++) {
    dataContainer[i].getElementsByClassName('picked')[j].innerHTML = '';
}

```

#### **Code 4.14: Visualisation Data Handling**

The data are then fed into an API known as charts.js which then interpolate and output, in the case of the application, a bar or pie chart.

```

piechartContainer.classList.add('pie-graph-container');
let labels = ['Spelling', 'Punctuation', 'Grammar', 'Understanding'];
let values = [0, 0, 0, 0];
for (let i = 0; i < (storedData.length - 2); i++) {
    if (storedData[i].correct == 1) {
        for (let j = 0; j < labels.length; j++) {
            if (storedData[i].type == labels[j]) {
                values[j] += 1;
            }
        }
    }
}

```

#### **Code 4.15: Pie Chart, Correct Answers**

Code 4.16 takes the data about each question the pupil answered and isolates those that were incorrect. These incorrect questions are then analysed, and a count of each question type is stored. The data is then sent to the charts.js API to create a doughnut chart displaying the percentage differences between each question type answered incorrect. This chart is to help teachers understand what type of questions the user/class struggled with the most.

```

let colourHex = ['#02BCF580', '#FA3CF780', '#FAC02F80', '#00E9B780'];
let goodPieChart = piechartGood.getContext('2d');
positivePieChart = new Chart(goodPieChart, {
    type: 'doughnut',
    data: {
        labels: labels,
        datasets: [{
            label: 'Question Types',
            backgroundColor: colourHex,
            borderWidth: 5,
            data: values
        }]
    },
    options: {
        responsive: true,
        legend: {
            position: 'bottom'
        },
        title: {
            display: true,
            text: 'Percentage difference of the question type that were answered correctly',
            fontSize: 25,
            fontColor: '#000',
            padding: 15
        }
    }
});

```

**Code 4.16: Pie Chart, Incorrect Answers**

A similar process is used for the bar charts, where data is handled, extracted and sent to the API for interpolating and producing the final chart.

```

let barChartVis = barchart.getContext('2d');
timeBarChart = new Chart(barChartVis, {
    type: 'bar',
    data: {
        labels: labels,
        datasets: [{
            label: 'Average time in seconds',
            backgroundColor: colourHex,
            data: values
        }]
    },
    options: {
        responsive: true,
        legend: {
            display: false
        },
        title: {
            display: true,
            text: 'Bar chart showing average time taken for each question type',
            fontSize: 25,
            fontColor: '#000',
            padding: 15
        },
        layout: {
            padding: 50
        }
    }
});

```

**Code 4.17: Bar Chart**

## 4.2 PHP Development

### 4.2.1 Dynamic Content

PHP, within the application, is largely used for validation and data handling, however, it also contributes to creating dynamic content. Certain criteria can be set up so that, should these criteria be fulfilled, the contents within a HTML document modifies.

```
$query = "SELECT * FROM students WHERE teacherid=? ORDER BY ".$order.' '.$sort;
$stmt = mysqli_stmt_init($connection);
mysqli_stmt_bind_param($stmt, 'i', $_SESSION['id']);
mysqli_stmt_execute($stmt);
$results = mysqli_stmt_get_result($stmt);
$numResults = mysqli_num_rows($results);
$numPages = ceil($numResults / $maxNumRows);
$startingRow = ($currentPage - 1) * $maxNumRows;
mysqli_data_seek($results, $startingRow);
for($i = 0; $i < $maxNumRows; $i++){
    <div class="pagination">
        <?php
            for($i = 1; $i < $numPages + 1; $i++){
                $pagination = '<a href="student-profile.php?page=' . $i;
                if(isset($_GET['orderby'])){
                    $pagination .= '&orderby=' . $_GET['orderby'];
                }
                if(isset($_GET['sort'])){
                    $pagination .= '&sort=' . $_GET['sort'];
                }
                if(isset($_GET['limit'])){
                    $pagination .= '&limit=' . $_GET['limit'];
                }
                if(isset($_GET['page'])){
                    if($_GET['page'] == $i){
                        $pagination .= '" class="current-page"';
                    }
                } else if($i == 1){
                    $pagination .= '" class="current-page"';
                }
                $pagination .= '>' . $i . '</a>';
                echo $pagination;
            }
        ?>
    </div>
}
}
```

Code 4.18: Pagination

Once such example is the pagination system created, using the code referenced at [19] as foundation. It splits large pupil records into pages which can then be navigated by the user. Initially, all the pupils registered under the logged in teacher is fetched and stored within a variable, results and the number of pages required to display all the pupils with the desired limit is calculated. To determine which records require displaying, Code 4.18, the user's current page is determined, initially being one. The starting row is then calculated by negating the current page number by one and multiplying the value by the max limit. This value is then iterated by the value of the limit, displaying the intended records. Figure 4.1 illustrates this process.

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

↓  
**Current Page**

**MaxNumRows = 5**

**StartingRow = ((1 -1) \* 5) = 0**

**Showing rows 0 - 4 (5 total)**

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

↓  
**Current Page**

**MaxNumRows = 5**

**StartingRow = ((2 - 1) \* 5) = 5**

**Showing rows 5 - 9 (5 total)**

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

↓  
**Current Page**

**MaxNumRows = 5**

**StartingRow = ((3 - 1) \* 5) = 10**

**Showing rows 10 - 14 (5 total)**

**Figure 4.1: Pagination Explanation**

#### 4.2.2 Registration System

As already stated, PHP is largely used for form validation and data handling, which is why it is featured extensively within the backend systems. One such system is the ability for the user to register an account with the site, created and expanded upon from [23]. A HTML form is used to acquire information that would be used within the system, once this form is submitted, the data is sent through to a PHP file, registration.inc.php, where it is checked and handled. Here several factors are validated before creating the user's account.

First the system connects to a database, the credentials of which are stored in dbh.inc.php which can be called when needed.

```
<?php

$server = 'localhost';
$username = 'root';
$password = '';
$database = 'educationalappsystem';

$connection = mysqli_connect($server, $username, $password, $database);

if (!$connection) {
    die("Unfortunatly something went wrong. ".mysqli_connect_error()." Please try again in a
little while...");
}
```

**Code 4.19: Database Connection**

The user inputs are stored in variables and validated throughout the script.

```
$userFirstname = $_POST['firstname'];
$userLastname = $_POST['lastname'];
$userID = $_POST['username'];
$userPass = $_POST['password'];
$userRepeatPass = $_POST['repeat-password'];

$errorMessage = '?errors=found';
$errorMessagePass = '&password=';
$userFieldsValues = array($userFirstname, $userLastname, $userID);
$userFieldsName = array('firstname', 'lastname', 'username');
$userFieldsChanges = False;
$userFieldsPassChanges = False;
```

**Code 4.20: Registration System Initialisation**

The first of these data handling is determining what user type is being registered, this allows the account to be inserted into the correct database.

```
if($_SESSION['usertype'] == 1){
    $usertype = 2;
    $databaseTable = 'teachers';
} else if($_SESSION['usertype'] == 2) {
    $usertype = 3;
    $teacherid = $_SESSION['id'];
    $databaseTable = 'students';
}
```

**Code 4.21: Registration System Database Determining**

Should the account type registering the account be allocated as ‘1’ meaning an admin is creating a teacher account, then the “teachers” database is selected, however, should the account be allocated “2”, a teacher is creating a pupil account, then the “student” database is selected and the current teacher’s account is catalogued to store the student account under them.

Next are validation checks, these checks ensure the data follows strict guidelines that are accepted by the database. Databases are specific, small mistakes can cause failure and thus these checks are vital in ensuring the application runs smoothly.

The first validation checks to see if all the required data were received, documenting which fields weren’t and flagging validation as “failed”. Later the user is sent back to the registration page with the empty field highlighted. This not only feedbacks to the user that an error occurred but also explicitly informs them what the error entails and how they could rectify the situation. All other validation also sends the user back with unique error messages.

```
//EMPTY FIELD CHECK
for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
    if(empty($userFieldsValues[$i])){
        $userFieldsValues[$i] = 'empty';
        $userFieldsChanges = True;
    }
}
if(empty($userPass)){
    $errorMessagePass .= 'empty';
    $userFieldsPassChanges = True;
}
```

**Code 4.22: Registration System Empty Field Check**

Another critical validation is string length. Each field within a database has a set max string length that can be stored. Code 4.23 confirms whether the string that was given exceed that specific field's max length, later highlighting the fields that require shortening.

```
//STRING LENGTH CHECK
for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
    if(strlen($userFieldsValues[$i]) > 255){
        $userFieldsValues[$i] = 'toolong';
        $userFieldsChanges = True;
    }
}
if(strlen($userPass) > 50){
    $errorMessagePass .= '&toolong';
    $userFieldsPassChanges = True;
}
```

**Code 4.23: Registration System String Length Check**

Certain fields have specific characters that they can accept, username filed accepting letters, numbers and underscores, password fields allowing letters, numbers and pre-selected special characters while name fields only allowing letters.

```
//STRING CHARACTER CHECK
for ($i = 0; $i < sizeof($userFieldsValues) - 1; $i++) {
    if(!preg_match("/^[\w_0-9]*$/", $userFieldsValues[$i])){
        $userFieldsValues[$i] = 'invalidchar';
        $userFieldsChanges = True;
    }
}
if(!preg_match("/^[\w]*$/", $userFieldsValues[2])){
    $userFieldsValues[2] = 'invalidchar';
    $userFieldsChanges = True;
}
if(!preg_match("/^(?=.*\d)(?=.*[^w_0-9])[w@%+!#$^?:.(){}[\]\~]*$/", $userPass) &&
 !$userFieldsPassChanges){
    $errorMessagePass .= 'invalidchar';
    $userFieldsPassChanges = True;
}
```

**Code 4.24: Registration System String Character Check**

Should the user input a character not accepted by it's corresponding field, the user is sent an error message informing them of what is explicitly acceptable.

The user is required to input their password twice, mitigating typos and avoiding storing a password the user didn't intend. By confirming that both fields match, the system could be confident that the password the user gave is the intended one.

```
//REPEAT PASSWORD CHECK
if(($userPass != $userRepeatPass) && (!$userFieldsPassChanges)){
    $errorMessagePass .= 'mismatch';
    $userFieldsPassChanges = True;
}
```

**Code 4.25: Registration System Repeat Password Check**

The final and most vital validation is to check if the username the user chose already exists within the userbase. The login system uses each username as a unique id to fetch the intended account and therefore, when creating new accounts, avoiding username clashes is pivotal to avoid system errors.

```

//EXISTING USER CHECK
$query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
$statement = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 's', $userID);
    mysqli_stmt_execute($statement);
    $results = mysqli_stmt_get_result($statement);
    if($row = mysqli_fetch_assoc($results)){
        $userFieldsValues[2] = 'taken';
        $userFieldsChanges = True;
    }
} else {
    $userFieldsChanges = True;
}

```

**Code 4.26: Registration System Existing User Check**

A query that fetches all the accounts inside the database where the username is equal to that which the user intends is called, if any results are returned, an account with the username already exists and the user is informed. Special precautions, when a database is accessed, are taken to ensure malicious attacks like SQL injections are avoided. Queries are prepared and bound with the external input before being executed, this allows the system to understand that any input from the user should never be executed, avoiding the system from fulfilling externally scripts.

Once each validation is complete, the script determines if the data can be used to create an account. If not, the system would send back the information that passed their validation auto completed within their respective field and only show error messages pertaining to those fields that failed and their reasons why. This increases the user experience as they now only require inputting and fixing fields that had problems rather than filling the entire form all over again. If all validations, however, do passed, then values are stored into the database, creating the account.

```

//VALIDATION PASSED, INSERT DETAILS INTO DATABASE
$query = "INSERT INTO ". $databaseTable ." (firstname, lastname, username, password,
usertype";
if($databaseTable == 'teachers'){
$query.= ') VALUES (?, ?, ?, ?, ?)';
} else {
$query.= ', teacherid) VALUES (?, ?, ?, ?, ?, ?, ?)';
}

if(mysqli_stmt_prepare($statement, $query)){
    $hash = password_hash($userPass, PASSWORD_DEFAULT);
    if($databaseTable == 'teachers'){
        mysqli_stmt_bind_param($statement, 'ssssi', $userFirstname, $userLastname, $userID, $hash,
$userType);
    } else {
        mysqli_stmt_bind_param($statement, 'ssssii', $userFirstname, $userLastname, $userID,
$hash, $usertype, $teacherid);
    }
    mysqli_stmt_execute($statement);
} else {
    $errorMessage .= $errorMessagePass;
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
        $errorMessage .= '&.'. $userFieldsName[$i]. '='. $userFieldsValues[$i];
    }
    if($_SESSION['usertype'] == 1){
        header('Location: ../registration.php' . $errorMessage);
        exit();
    } else {
        header('Location: ../add-student.php' . $errorMessage);
        exit();
    }
}

```

**Code 4.27: Registration System Inserting into Database**

A crucial step taken in Code 4.27 is the hashing of passwords before storage. Storing passwords directly into a database through plain text causes security issues. If the database becomes compromised, anyone would be able to simply read each account's password. Hashing allows converting passwords into a unique string of arbitrary characters through a very specific algorithm. This "hash" is stored instead, then when the user logs in, their input is put through the same algorithm and compared. If the hashes match, then the system can be confident that the user inputted the correct password and allow the user into their account.

Once the user account has been created, the database is closed, and the user is redirected to the page they came from with a success message.

```
mysqli_stmt_close($statement);
mysqli_close($connection);

if($_SESSION['usertype'] == 1){
    header('Location: ../registration.php?success');
    exit();
} else {
    header('Location: ../add-student.php?success');
    exit();
}
```

Code 4.28: Registration System Redirect

### 4.2.3 Login and Logout Systems

The login system follows a similar structure, using a HTML form to receive user input, validating the data to ensure they are within the correct formats and sending the user back to the login page with appropriate error messages.

```
//EXISTING USER CHECK
if(!$userFieldsChanges){
    $query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query)){
        mysqli_stmt_bind_param($statement, 's', $userID);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if(($row = mysqli_fetch_assoc($results))){
            //PASSWORD CHECK
            if(!password_verify($userPass, $row['password'])){
                $errorMessagePass .= 'incorrect';
                $userFieldsChanges = True;
            }
        } else {
            $userID = 'invaliduser';
            $userFieldsChanges = True;
        }
    } else {
        $userFieldsChanges = True;
    }
}
```

Code 4.29: Login System Existing User Check

Code 4.29 demonstrates how the system validates account credentials. A query that checks to see if the database holds the username given by the user is prepared and bound with the user's input, if no results are returned then the user is informed that the account doesn't exist, however, should the query pass then the following steps are taken.

The user's password is hashed and compared with the hash stored, redirecting back to the login page upon failure.

Should the username and password both validate, however, the system is able to log the user into their account through sessions. By default, HTTP protocols are stateless, meaning information is not kept between pages. Sessions allows the system to simulate a sense of statefulness by allowing information to be passed on, letting the application keep track of the user currently logged in

```
session_start();
if($usertype != 1){
    $_SESSION['firstname'] = $row['firstname'];
    $_SESSION['lastname'] = $row['lastname'];
}
$_SESSION['id'] = $row['id'];
$_SESSION['username'] = $row['username'];
$_SESSION['usertype'] = $row['usertype'];

mysqli_stmt_close($statement);
mysqli_close($connection);

if($_SESSION['usertype'] == 1){
    header('Location: ../registration.php');
    exit();
} else if($_SESSION['usertype'] == 2){
    header('Location: ../dashboard.php');
    exit();
} else if($_SESSION['usertype'] == 3){
    header('Location: ../index.php');
    exit();
}
```

**Code 4.30: Login System Initialisation**

Once the session is started, specific information is stored into session variables that would be used to change content within the application and the user is redirected to their corresponding starting page.

To log the user out, the application simply destroys the current session and sends the user back to the index page

```
<?php

session_start();
session_unset();
session_destroy();
header('Location: ../index.php?success');
exit();
```

**Code 4.31: Logout System**

#### 4.2.4 Profile Modification System

The profile modification system follows a nearly identical structure to the registration system, validating user input before updating the current stored data. Format check such as empty, length and character string are performed. Existing user checks are also performed to ensure if the user requires a username change, no clashes with existing usernames are created. Security checks such as password validation are made to ensure only the owner of the account is facilitating the changes.

Once the modifications are complete, the user's session variables are updated to conform to the changes and the user is redirected back to their profile page with a success message.

```
session_start();
$_SESSION['firstname'] = $userFirstname;
$_SESSION['lastname'] = $userLastname;
$_SESSION['username'] = $newuserID;
```

Code 4.32: Sessions

#### 4.2.5 Task Assignment System

Three main data are required by the task assignment system, the teacher who assigned the task, the task itself and the pupil who the task is being assigned too. The system locates the pupil and activity within the database and assigns the desired activity under the assigning teacher. Any existing assignments on record, along with any results attached to them, are deleted in the process.

```
//USER INPUT VALIDATION
if(empty($userID) || empty($assignmentID) || empty($studentID)){
    header($urlError);
    exit();
}
//USER ID VALIDATION
$query = "SELECT * FROM students WHERE username=? && id=?";
$stmt = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, 'si', $userID, $studentID);
    mysqli_stmt_execute($stmt);
    $results = mysqli_stmt_get_result($stmt);
    if(!$row = mysqli_fetch_assoc($results)){
        header($urlError);
        exit();
    }
} else {
    header($urlError);
    exit();
}
//EXISTING ACTIVITY VALIDATION (Ignore if none since that means no assignments given)
if($assignmentID != 'none') {
    $query = "SELECT * FROM activities WHERE id = ?";
    if(mysqli_stmt_prepare($stmt, $query)) {
        mysqli_stmt_bind_param($stmt, 'i', $assignmentID);
        mysqli_stmt_execute($stmt);
        $results = mysqli_stmt_get_result($stmt);
        if(!$row = mysqli_fetch_assoc($results)){
            header($urlError);
            exit();
        }
    } else {
        header($urlError);
        exit();
    }
}
```

```

//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment WHERE studentid = ?";
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 'i', $studentID);
    mysqli_stmt_execute($statement);
    $results = mysqli_stmt_get_result($statement);
    if (($row = mysqli_fetch_assoc($results))) {
        //DELETING EXISTING STORED RESULTS
        $query = "DELETE FROM results_storage WHERE assignmentid = ?";
        if(mysqli_stmt_prepare($statement, $query)) {
            mysqli_stmt_bind_param($statement, 'i', $row['id']);
            mysqli_stmt_execute($statement);
            //DELETING ASSIGNMENT ITSELF (second because it is a foreign id to
            results_storage)
            $query = "DELETE FROM activity_assignment WHERE id = ?";
            echo '.$row['id'].' and '.$row['time'];
            if(mysqli_stmt_prepare($statement, $query)) {
                mysqli_stmt_bind_param($statement, 'i', $row['id']);
                mysqli_stmt_execute($statement);
            } else {
                header($urlError);
                exit();
            }
        } else {
            header($urlError);
            exit();
        }
    }
} else {
    header($urlError);
    exit();
}
//ASSIGNING NEW ASSIGNMENT TO STUDENT
if($assignmentID != 'none'){
    $query = "INSERT INTO activity_assignment (teacherid, studentid, activityid) VALUES
    (?, ?, ?)";
    if(mysqli_stmt_prepare($statement, $query)) {
        mysqli_stmt_bind_param($statement, 'iii', $_SESSION['id'], $studentID, $assignmentID);
        mysqli_stmt_execute($statement);
    } else {
        header($urlError);
        exit();
    }
}
header($urlSuccess);
exit();

```

Code 4.33: Task Assignment System

## 4.2.6 Results Storage and Fetching Systems

An AJAX method is used to validate and store results into a database.

```
//EMPTY VALIDATION
if( empty($data) || empty($overallTime) || empty($activityID)) {
    echo $errorMessage;
    exit();
}
```

Code 4.34: Results Storage Empty Field

The validations performed in Code 3.34 checks to see that any data was given to the system, informing the user if no data was found.

```
//FORMAT VALIDATION
for($i = 0; $i < sizeof($data); $i++) {
    for($j = 0; $j < sizeof($data[$i]['choicesList']); $j++) {
        if( !preg_match("/^a-zA-Z0-9!?,* ]*$/", $data[$i]['choicesList'][$j]) ){
            echo $errorMessage;
            exit();
        }
    }
    for($j = 0; $j < sizeof($data[$i]['choicesPicked']); $j++) {
        if( !preg_match("/^a-zA-Z0-9!?,* ]*$/", $data[$i]['choicesPicked'][$j]) ){
            echo $errorMessage;
            exit();
        }
    }
    if( !preg_match("/^a-zA-Z ,? ]*$/", $data[$i]['questionTitle']) || !preg_match("/^a-zA-
Z0-9! ]*$/", $data[$i]['correctChoice']) ) {
        echo $errorMessage;
        exit();
    }
    if( !in_array($data[$i]['questionType'],
array('Spelling','Understanding','Punctuation','Grammar')) ){
        echo $errorMessage;
        exit();
    }
    if( !is_numeric($data[$i]['attempts']) || !is_numeric($data[$i]['timeTaken']) || 
!is_numeric($data[$i]['correct']) || !is_numeric($overallTime) || !is_numeric($activityID) ){
        echo $errorMessage;
        exit();
    }
}
```

Code 4.35: Results Storage Format Validation

The data is then assessed to ensure it wasn't tampered with for system protection.

Once all validations pass, the system increments the user's attempts by one, updates their newest time and deletes existing date to be replaced with their new results.

```
//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment WHERE studentid = ?";
$stmt = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($stmt, $query)) {
    mysqli_stmt_bind_param($stmt, 'i', $_SESSION['id']);
    mysqli_stmt_execute($stmt);
    $assignmentResults = mysqli_stmt_get_result($stmt);
    while ( ($assignmentRow = mysqli_fetch_assoc($assignmentResults)) ) {
        if( $assignmentRow['activityid'] == $activityID ) {
            $overallAttempts = $assignmentRow['attempts'] + 1;
            $query = "DELETE FROM results_storage WHERE assignmentid = ?";
            if(mysqli_stmt_prepare($stmt, $query)) {
                mysqli_stmt_bind_param($stmt, 'i', $assignmentRow['id']);
                mysqli_stmt_execute($stmt);
                $query = "UPDATE activity_assignment SET attempts=?, time=? WHERE id=?";
                if(mysqli_stmt_prepare($stmt, $query)) {
                    mysqli_stmt_bind_param($stmt, 'iii', $overallAttempts, $overallTime,
$assignmentRow['id']);
                    mysqli_stmt_execute($stmt);
                    $query = "INSERT INTO results_storage (assignmentid, question, correct,
attempts, time, type, choices, picked, title, answer) VALUES (?,?,?,?,?, ?,?, ?,?)";
                    for($i = 0; $i < sizeof($data); $i++) {
                        $choicePicked = implode(' ', $data[$i]['choicesPicked']);
                        $choiceList = implode(' ', $data[$i]['choicesList']);
                        if(mysqli_stmt_prepare($stmt, $query)) {
                            mysqli_stmt_bind_param($stmt, 'iiiiisssss',
$assignmentRow['id'], $i, $data[$i]['correct'], $data[$i]['attempts'], $data[$i]['timeTaken'],
$data[$i]['questionType'], $choiceList, $choicePicked, $data[$i]['questionTitle'],
$data[$i]['correctChoice']);
                            mysqli_stmt_execute($stmt);
                        } else {
                            echo $errorMessage;
                            exit();
                        }
                    }
                } else {
                    echo $errorMessage;
                    exit();
                }
            } else {
                echo $errorMessage;
                exit();
            }
        }
    }
} else {
    echo $errorMessage;
    exit();
}
}
```

**Code 4.36: Results Storage**

Fetching results storage for visualisation also follows a similar structure, receiving a fetch request from an AJAX method, handling the request and outputting the correct results.

The script first checks to see if the teacher requested an individual's result or a class summery, depending on which, the script has slight variation in execution.

For individual results, the script validates if the pupil requested exists, if the pupil had an assignment attached to them and if there are any data stored to do with said pupil. If, at any point, the script fails then an error message is sent back to inform the teacher that a problem occurred and with possible solutions on how to rectify it, otherwise the results are collected and sent through to the results script to be turned into visual information for displaying.

```

//USER VALIDATION
$query = "SELECT * FROM students WHERE username=?";
$stmt = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, 's', $userID);
    mysqli_stmt_execute($stmt);
    $studentResults = mysqli_stmt_get_result($stmt);
    if(!$studentRow = mysqli_fetch_assoc($studentResults)){
        echo 'Unfortunatly something might have gone wrong, the user appears to not exist.';
        exit();
    } else {
        mysqli_data_seek($studentResults,0);
    }
} else {
    echo $errorMessage;
    exit();
}

//USER ASSIGNMENT VALIDATION
$query = "SELECT * FROM activity_assignment WHERE studentid = ?";
if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, 'i', $studentRow['id']);
    mysqli_stmt_execute($stmt);
    $assignmentResults = mysqli_stmt_get_result($stmt);
    if(!$assignmentRow = mysqli_fetch_assoc($assignmentResults)){
        echo 'Unfortunatly something might have gone wrong, the user appears to not have been
tasked any assignments.';
        exit();
    } else {
        if($assignmentRow['attempts'] == 0){
            echo 'The user has yet to attempt the assignment.<br>Please try again in a little
while...';
            exit();
        }
        mysqli_data_seek($assignmentResults,0);
    }
} else {
    echo $errorMessage;
    exit();
}

//RESULTS EXISTING CHECK
$query = "SELECT * FROM results_storage WHERE assignmentid = ?";
if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, 'i', $assignmentRow['id']);
    mysqli_stmt_execute($stmt);
    $results = mysqli_stmt_get_result($stmt);
    if(!$row = mysqli_fetch_assoc($results)){
        echo 'Unfortunatly something went wrong and we couldn\'t store the user\'s re-
sults<br>Please advise user to try the test again.';
        exit();
    } else {
        mysqli_data_seek($results,0);
        while($row = mysqli_fetch_assoc($results)){
            array_push($storedResults, $row);
        }
        array_push($storedResults, $assignmentRow['attempts']);
        array_push($storedResults, $assignmentRow['time']);
    }
} else {
    echo $errorMessage;
    exit();
}

```

Code 4.37: Results Fetching Individual

For class summaries, the script checks to see if the teacher had given out any assignments and if at least a single pupil had completed it. Then all the pupil's data are tallied together and sent over to the results script for displaying.

```

//CLASS ASSIGNMENT VALIDATION
$query = "SELECT * FROM activity_assignment WHERE teacherid=?";
$stmt = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($stmt, $query)){
    mysqli_stmt_bind_param($stmt, 'i', $_SESSION['id']);
    mysqli_stmt_execute($stmt);
    $assignmentResults = mysqli_stmt_get_result($stmt);
    if(!($assignmentRow = mysqli_fetch_assoc($assignmentResults))){
        echo 'There appears to be no active assignments to display.';
        exit();
    }
} else {
    echo $errorMessage;
    exit();
}

//CLASS RESULTS EXISTING CHECK
mysqli_data_seek($assignmentResults, 0);
$classAttempts = 0;
$classTime = 0;
$count = 0;
while($assignmentRow = mysqli_fetch_assoc($assignmentResults)){
    $query = "SELECT * FROM results_storage WHERE assignmentid = ?";
    if(mysqli_stmt_prepare($stmt, $query)){
        mysqli_stmt_bind_param($stmt, 'i', $assignmentRow['id']);
        mysqli_stmt_execute($stmt);
        $results = mysqli_stmt_get_result($stmt);
        $count++;
        $classAttempts += intval($assignmentRow['attempts']);
        $classTime += intval($assignmentRow['time']);
        if(!$row = mysqli_fetch_assoc($results)) {
            echo 'There appears to be no results on record to display.<br>No student has yet
to attempt the assigned task.';
            exit();
        } else {
            mysqli_data_seek($results, 0);
        }
        while($row = mysqli_fetch_assoc($results)){
            array_push($storedResults, $row);
        }
    } else {
        echo $errorMessage;
        exit();
    }
}
array_push($storedResults, $count);
array_push($storedResults, $classAttempts);
array_push($storedResults, $classTime);
echo json_encode($storedResults);

```

**Code 4.38: Results Fetching Class**

# Testing

Testing of the system's various components were carried out both during and after the development process. During implementation, design examples were shown and critiqued by educators until a minimum satisfactory level was achieved, the system was then created and checked periodically to ensure the different components functioned correctly, executing and terminating when expected and confirming that any and all data output were expected and valid. Upon completion, final user testing was conducted in addition to testing the component against it's corresponding system requirement. Once all tests were concluded positive, the component was considered complete and finalised.

## 5.1 Whitebox Testing

Whitebox testing is a testing method where a system is tested at a structural level, the inner workings of a system is scrutinised resulting in improvement to security and ensuring data are consistent throughout the application. Below is a snippet of the testing conducted, for full tables see [Appendix H: Testing]

Action	Expected Outcome	Actual Outcome	Status
User selects one of three account types in the login page	The system takes the user type selected and uses its corresponding database	The system takes the user type selected and uses its corresponding database	Pass
User leaves fields empty	System sends user back to the login page with the empty fields highlighted and informs the user that the field is required.	System sends user back to the login page with the empty fields highlighted and informs the user that the field is required.	Pass
User inputs a username that doesn't exist in the database	System sends user back to the login page with the username field highlighted and informs the user that the username doesn't exist	System sends user back to the login page with the username field highlighted and informs the user that the username doesn't exist	Pass
User inputs a username that does exist in the database but incorrect password	System sends user back to the login page with the username field prefilled with its previous data and the password field highlighted and informs the user that the password was incorrect	System sends user back to the login page with the username field prefilled with its previous data and the password field highlighted and informs the user that the password was incorrect	Pass

User inputs a username that does exist in the database with correct password	System creates a session and stores user info into session variables	System creates a session and stores user info into session variables	Pass
--	--	--	------

## 5.2 Blackbox Testing

Blackbox testing is a testing method where a system is tested at a user level, the outer shell of a system is tested to confirm if the functions of an application is usable by the intended users, resulting in better user experience and design. Below is a snippet of the testing conducted, for full table see [Appendix H: Testing]

Action	Expected Outcome	Actual Outcome	Status
Admin account logs in	User is taken to the teacher registration page	User is taken to the teacher registration page	Pass
Admin inputs details into the registration page	A teacher account is created, and a success message is shown	A teacher account is created, and a success message is shown	Pass
Teacher account logs in	User is taken to their dashboard page	User is taken to their dashboard page	Pass
Teacher navigates to teacher profile page	The accounts details are shown to the user with the ability to input changes	The accounts details are shown to the user with the ability to input changes	Pass
Teacher updates details in teacher profile page	A success message and new details are shown	A success message and new details are shown	Pass
Teacher navigates to student profile page	User is taken to the student profile page with details about each student that is registered to them shown with the ability to input changes	User is taken to the student profile page with details about each student that is registered to them shown with the ability to input changes	Pass
Teacher updates details for a chosen student	A success message and new details are shown	A success message and new details are shown	Pass

## 5.3 User Testing

During the development process, educators were asked to provide feedback on design and functionality of the application which were then fed back into the implementation to make changes that adhered more to the stakeholder's requirements. These feedbacks modified aspects such as layout, colour schemes, additional functionalities, etc. Once development was complete, users were asked one last time to provide their thoughts on the application which was used for evaluation purposes.

# Conclusion

This section will evaluate the application's validity in terms of its system requirements and user testing. It also provides the future prospect of the application by providing further possible developments and enhancements that could be implemented to increase functionality and usability.

## 6.1 System Successes

General feedback from user testing was quite positive, teachers enjoyed the simplicity of the application and were appreciative of its direction. They admired that all the functionality that teachers can perform within the application were partitioned off into individual pages. Furthermore, they felt if all the functionality was together the page would look cumbersome and may overwhelm some, making it difficult to use. The children's activity was brightly coloured with many moving images, which teachers said would be immensely helpful in keeping pupils engaged.

Another aspect which had a positive impact and was highly praised was the page adapting to each pupil e.g. mentioning their name. Educators felt it made the assignment feel more personable. They liked that the activity encouraged pupils as they were being assessed, praising them when they did well while motivating them when they struggled.

One of the most highly admired function was the ability to view a classroom summary, teachers said this was advantageous in school meetings where teachers often had to communicate their classes performance, or during parents evening where teachers are required to report each pupil's ability. They felt it would dramatically reduce the work required whilst also providing far more accurate results.

The ability to assign pupil individual assignments, rather than the entire class, was helpful since pupils tend to work at different rates. Some pupils might require harder assignments while those that need more help can be given an easier time.

Overall, evaluation of the system suggests it to be successful, meeting all system requirements, passing both white and black box testing and largely accepted by the intended stakeholders. The system performs its core functionality; assigning pupil activities, monitoring their usage and feedbacking the results to the teacher.

## 6.2 System Failures

Users did, however, feel there were many aspects of the application that could improve. While appreciative of the current information shown, they felt additional analytics would allow for richer assessments, further improving the tool's effectiveness.

Another major criticism was the inability to see past assessments. Currently, once teachers reassign new assessments, any existing results get destroyed. Teachers felt this drastically reduced the potential of the tool as without the ability to compare pupils' assessment termly, the application can only really tell how pupils are currently doing. If the system kept records of pupils' entire year's progress, it would be a better indicator of pupil progress.

A quality of life improvement suggested was the ability to execute functions in batches. Every function within the application can only be done one at a time. If teachers require creating multiple student accounts or assigning multiple assignments, they had to do so individually. This was tiring for educators and put some off on using the tool.

The lack of notifications within the app was also a point of contention. The application doesn't inform when pupils complete assignments, instead, teachers are required to sought out the information themselves. This, like the previous statement, can be burdensome on teachers, a notification system would completely elevate this struggle.

Furthermore, it was brought up that pupils also aren't notified of their assignments, pupils have no current way of knowing what tasks were given them. As there was only a single task in this prototype, this was a major oversight in design. Once the application is populated with a multitude of activities, informing pupils which activity they will be assessed on would be a critical functionality that the application would need to adhere.

Another system failure revealed during testing was that the application doesn't discriminate between assignments. A functionality of the application is a classroom summery, however, as the prototype only contains a single activity, the system simply retrieves every assignment given by the educator. This would be a critical failure in future, with multiple activities. The system would retrieve results from unrelated assessments, accumulate it and display as if it were a single assessment, resulting in highly inaccurate and unusable data. Tweaking is required so that only applicable data is retrieved during this process.

Admin accounts only functionality is to create teacher accounts, it was felt that broadening their role would improve the application's versatility. If admins were granted master control over all user accounts, allowing not only teacher account creation but student accounts as well, it would alleviate a lot of busy work for teachers. Currently, it was up to the teachers to create accounts for all their pupils. Teachers felt this task was more suitable for the school to perform. Furthermore, giving admin accounts the ability to modify account details would mean that pupils could modify their details, without having to burden the teacher, through a simple request to the school's IT system.

Where pagination is involved, should users manually edit the URL to view a smaller amount of accounts per page then allowed, then the pagination, especially for smaller devices, tends to break. If the system was forced to display one student record per page, it resulted in far too many pages within the pagination, as the system isn't equipped to handle this issue, the pagination was sent off screen. While functionality wasn't entirely broken, as the issue was easily rectified through changing the limit to a more acceptable value, it did make the page look unappealing. This issue would need to be fixed by either removing the ability to manually edit the limit or by designing the pagination to dynamically adapt itself to better accommodate.

## 6.3 Future Development

### 6.3.1 Additional Assignment Functionality

The application currently only allows single tasks to be assigned to a pupil at any one time, while usable, often educators teach multiple content during a week, having the ability to test everything the child has learnt through multiple assignments would greatly beneficial. Furthermore, it would also allow teachers the ability to decide which assignment they require data on, giving them more freedom when using the tool. Another improvement would be allowing for the system to be customisable. Educators often notice that education tools come with pre-set data which, at times, clash with that taught in class. The ability for the application be adaptable to each specific class was highly requested. It would make assessment much more accurate if the system was testing specifics taught in class. This feature would not only benefit teachers but would also allow stronger justification to educators in investment if they knew that the tool wouldn't be outdated within a few years due to its adaptability.

### 6.3.2 Sub Categories

Sub categories were an equally well sought out feature. The ability to split classes into certain groups, assigning each different assignment, depending on their ability, would assist teachers in planning lessons and further improve the application's utility. Furthermore, it would give teachers the freedom to decide what they require visualising, if a teacher wanted a summary of just their best pupils, this feature would facilitate that. The feature would allow teachers to keep track of pupils of interest without requiring them to constantly search them out.

### 6.3.3 Pupil Feedback

Existing assessment methods include receiving pupil feedback by asking them to fill out a form before every new learning to discuss what they felt confident on, providing teachers with information that could be used for planning future lessons. This could be replicated digitally by the system; upon completion of an assessment the pupil could be asked to provide feedback. This would need to be simple enough so that it doesn't feel like a chore for pupils, however, complex enough to provide practical uses for educators. This system would require rigorous testing with pupils to ensure the feature is not only well received but functionally suffice.

#### **6.3.4 App Support**

The system's current form is a web application, while adequate, pupils, especially during the younger years, tend to prefer smaller devices such as iPads. The application attempts to accommodate this through adapting itself for smaller screen, however, a more permanent solution would be porting the system into an app format. This would be beneficial as it could introduce further utility such as offline support. A tablet app could hold information regarding the pupil's interaction and upload them to the database once internet access is granted, allowing pupils to complete activities regardless of where they are. This would increase the user experience as users no longer need to be tethered to the internet. Furthermore, the application would be redesigned with tablets in mind, catering to the unique experience tablets grant and would be on a more familiar device for pupils.

# Bibliography

1. ‘Teacher Assessment “as Good as Tests” at Predicting Success’. *Tes*, <https://www.tes.com/news/teacher-assessment-good-tests-predicting-success>.
2. Rimfeld, Kaili, et al. ‘Teacher Assessments during Compulsory Education Are as Reliable, Stable and Heritable as Standardized Test Scores’. *Journal of Child Psychology and Psychiatry*, vol. 0, no. 0. *Wiley Online Library*, doi:10.1111/jcpp.13070.
3. ‘Sats Create Needless Pressure for Teachers and Pupils, Heads Warn’. *Tes*, <https://www.tes.com/news/sats-create-needless-pressure-teachers-and-pupils-heads-warn>.
4. ‘More Primary School Children Suffering Stress from Sats, Survey Finds’. *The Guardian*, 30 Apr. 2017. [www.theguardian.com](http://www.theguardian.com), <https://www.theguardian.com/education/2017/may/01/sats-primary-school-children-suffering-stress-exam-time>.
5. ‘Ofsted Inspectors to Stop Using Exam Results as Key Mark of Success’. *The Guardian*, 11 Oct. 2018. [www.theguardian.com](http://www.theguardian.com), <https://www.theguardian.com/education/2018/oct/11/ofsted-to-ditch-using-exam-results-as-mark-of-success-amanda-spielman>.
6. Riener, Cedar, and Daniel Willingham. ‘The Myth of Learning Styles’. *Change: The Magazine of Higher Learning*, vol. 42, no. 5, Aug. 2010, pp. 32–35. *Taylor and Francis+NEJM*, doi:10.1080/00091383.2010.503139
7. Choi, Ikseon, et al. ‘Implementing a Case-Based e-Learning Environment in a Lecture-Oriented Anaesthesiology Class: Do Learning Styles Matter in Complex Problem Solving over Time?\*’. *British Journal of Educational Technology*, vol. 40, no. 5, 2009, pp. 933–47. *Wiley Online Library*, doi:10.1111/j.1467-8535.2008.00884.x
8. Bug Club - Capture Your Children’s Imagination and Nurture Lifelong Readers. <https://www.pearsonschoolsandfecolleges.co.uk/Primary/Literacy/AllLiteracyresources/BugClub/bug-club-overview.aspx>.
9. Mathletics - ‘Mathletics: Powering Maths Learning across the UK and Europe’. Mathletics UK, <http://uk.mathletics.com/>.
10. MyMaths - Bringing Maths Alive - Home. <https://www.mymaths.co.uk/>.
11. Insight | Online Pupil Tracking for Primary Schools. <https://www.insighttracking.com/>.
12. ‘A UX Guide To The Child’s Mind’. *UX Studio*, 31 July 2018, <https://uxstudioteam.com/ux-blog/design-for-kids/>.
13. Wong, Euphemia. ‘Principle of Consistency and Standards in User Interface Design’. *The Interaction Design Foundation*, <https://www.interaction-design.org/literature/article/principle-of-consistency-and-standards-in-user-interface-design>.
14. ‘The Definitive Guide to Building Apps For Children’. *Toptal Design Blog*, <https://www.toptal.com/designers/interactive/guide-to-apps-for-children>.
15. *Designing Apps for Kids - Best Practices* /. 21 Mar. 2017, <https://www.savahapp.com/blog/designing-kids-apps/>.
16. ‘Children’s UX: Usability Issues in Designing for Young People’. *Nielsen Norman Group*, <https://www.nngroup.com/articles/childrens-websites-usability-issues/>.
17. ‘Reporting to Parents at the End of Key Stages 1 and 2’. *GOV.UK*, <https://www.gov.uk/guidance/reporting-to-parents-at-the-end-of-key-stages-1-and-2>.
18. ‘National Curriculum Assessments: Practice Materials’. *GOV.UK*, <https://www.gov.uk/government/collections/national-curriculum-assessments-practice-materials>.
19. Automating Everything. *Create Pagination in PHP and MySQL*. *YouTube*, <https://www.youtube.com/watch?v=gdEpUPMh63s>.
20. Web Dev Simplified. *Build A Quiz App With JavaScript*. *YouTube*, <https://www.youtube.com/watch?v=riDzcEQbX6k>.
21. ‘SpeechSynthesis’. *MDN Web Docs*, <https://developer.mozilla.org/en-US/docs/Web/API/SpeechSynthesis>.
22. *Chart.js / Open Source HTML5 Charts for Your Website*. <https://www.chartjs.org/>.

23. mmtuts. *How To Create A Complete Login System In PHP / Procedural MySQLi / 2018 PHP Tutorial* / Mmtuts. YouTube, <https://www.youtube.com/watch?v=LC9GaXkdxF8>.
24. *PHP: Hypertext Preprocessor*. <https://www.php.net/>.

**All icons used within the application were sourced through these royalty free repositories**

25. *Font Awesome*. <https://fontawesome.com/>.
26. *Icons DB - Free Custom Icons*. <https://www.iconsdb.com/>.

**All images used within the application were either sourced through these royalty free repositories or drawn by me.**

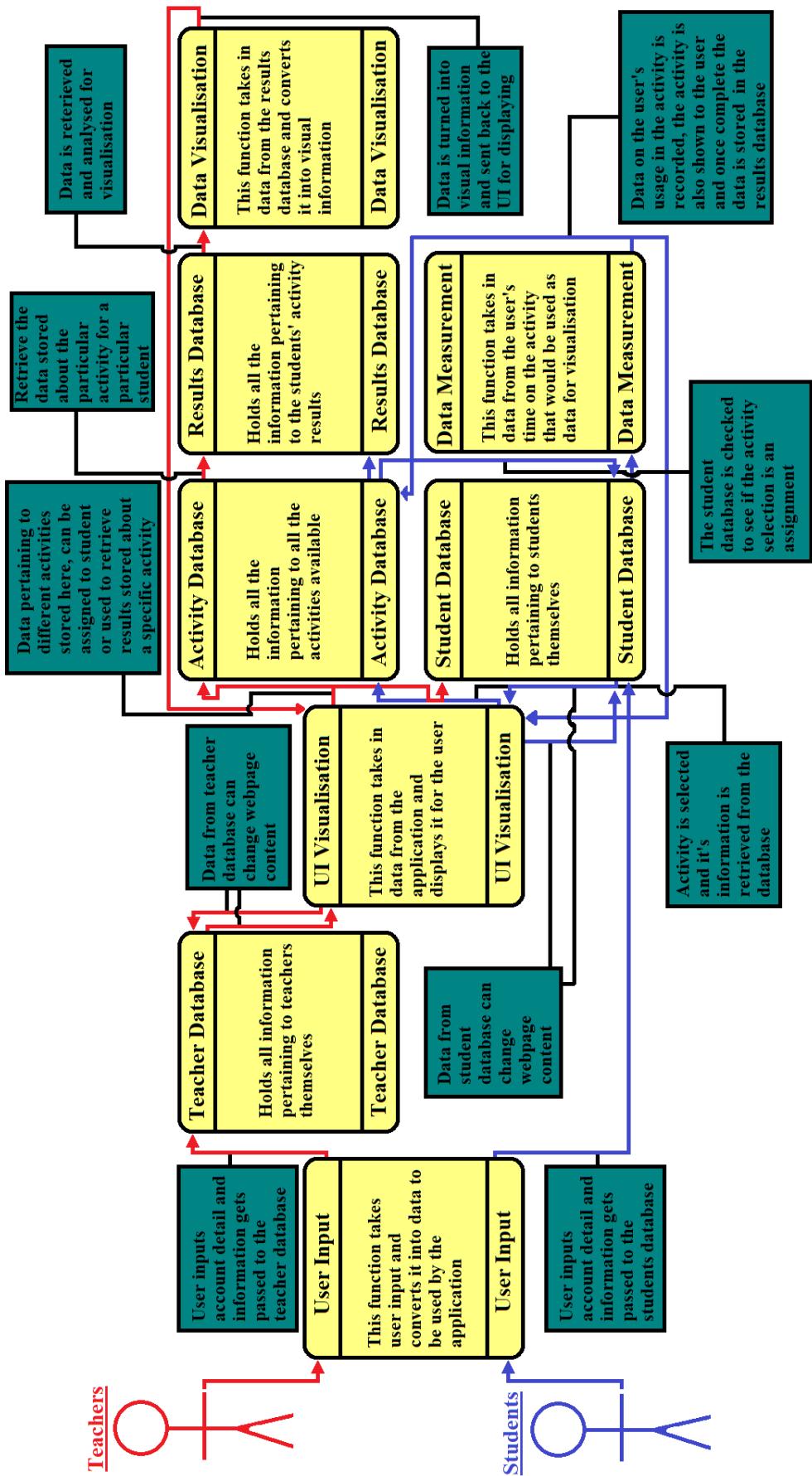
27. *1 Million+ Stunning Free Images to Use Anywhere - Pixabay*. <https://pixabay.com/>.
28. *Unsplash. Beautiful Free Images & Pictures / Unsplash*. <https://unsplash.com/>.

**All music used within the application were sourced through these royalty free repositories**

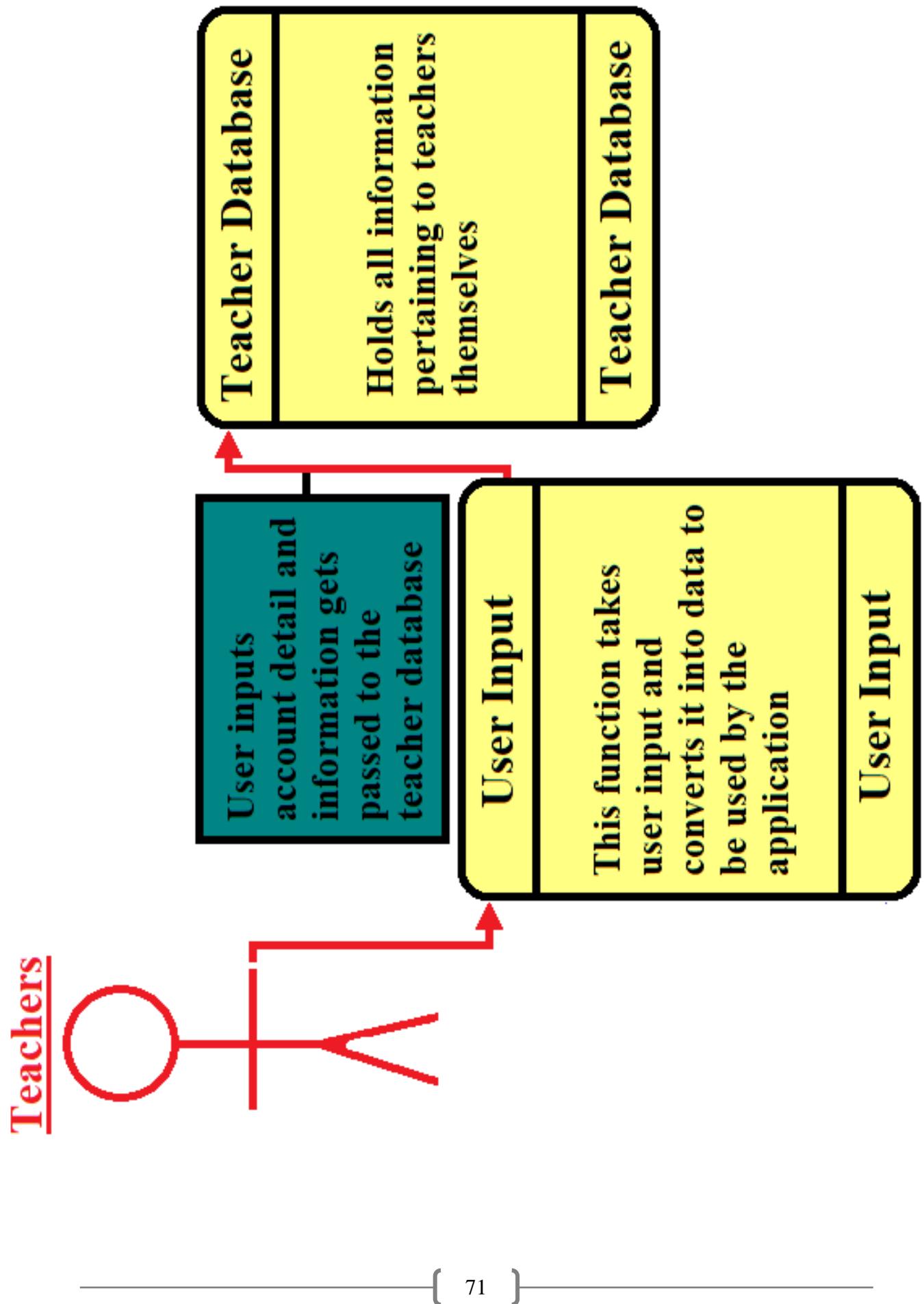
29. *Royalty Free Music by Bensound / Stock Music*. <https://www.bensound.com/>.
30. *Freesound - Freesound*. <https://freesound.org/>.

# Appendix A: Functional Overview Diagram

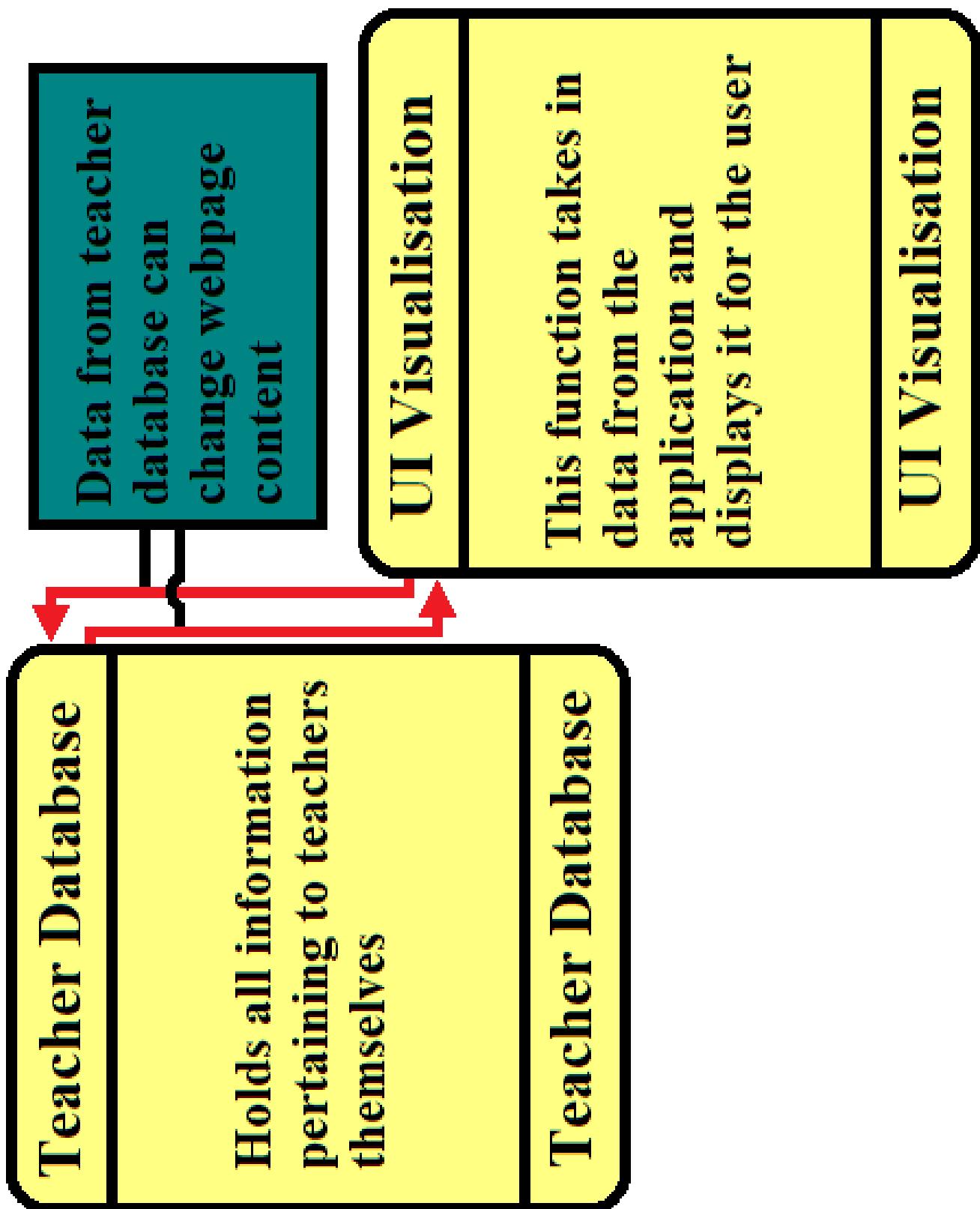
[Figure 1]



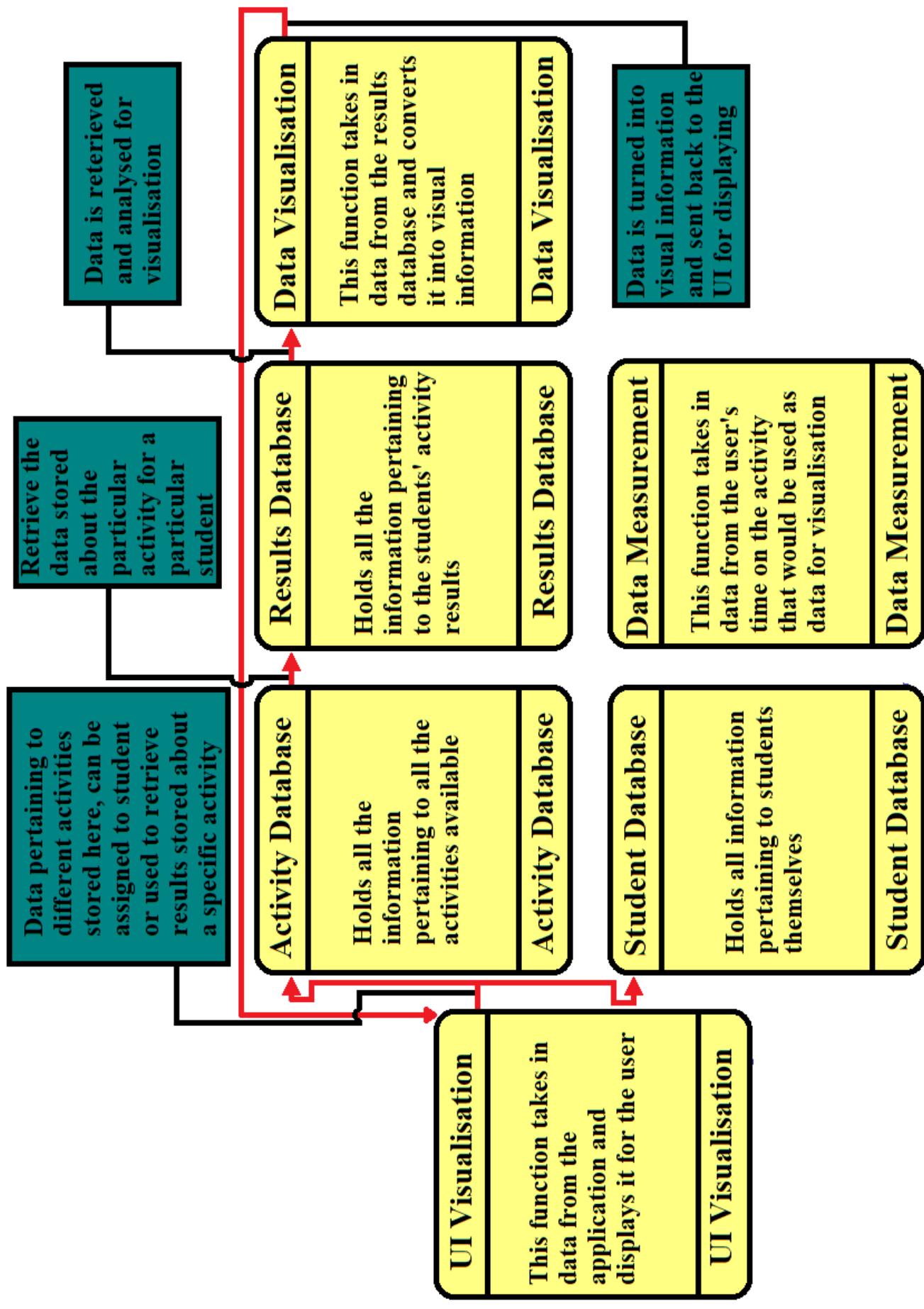
[Figure 2]



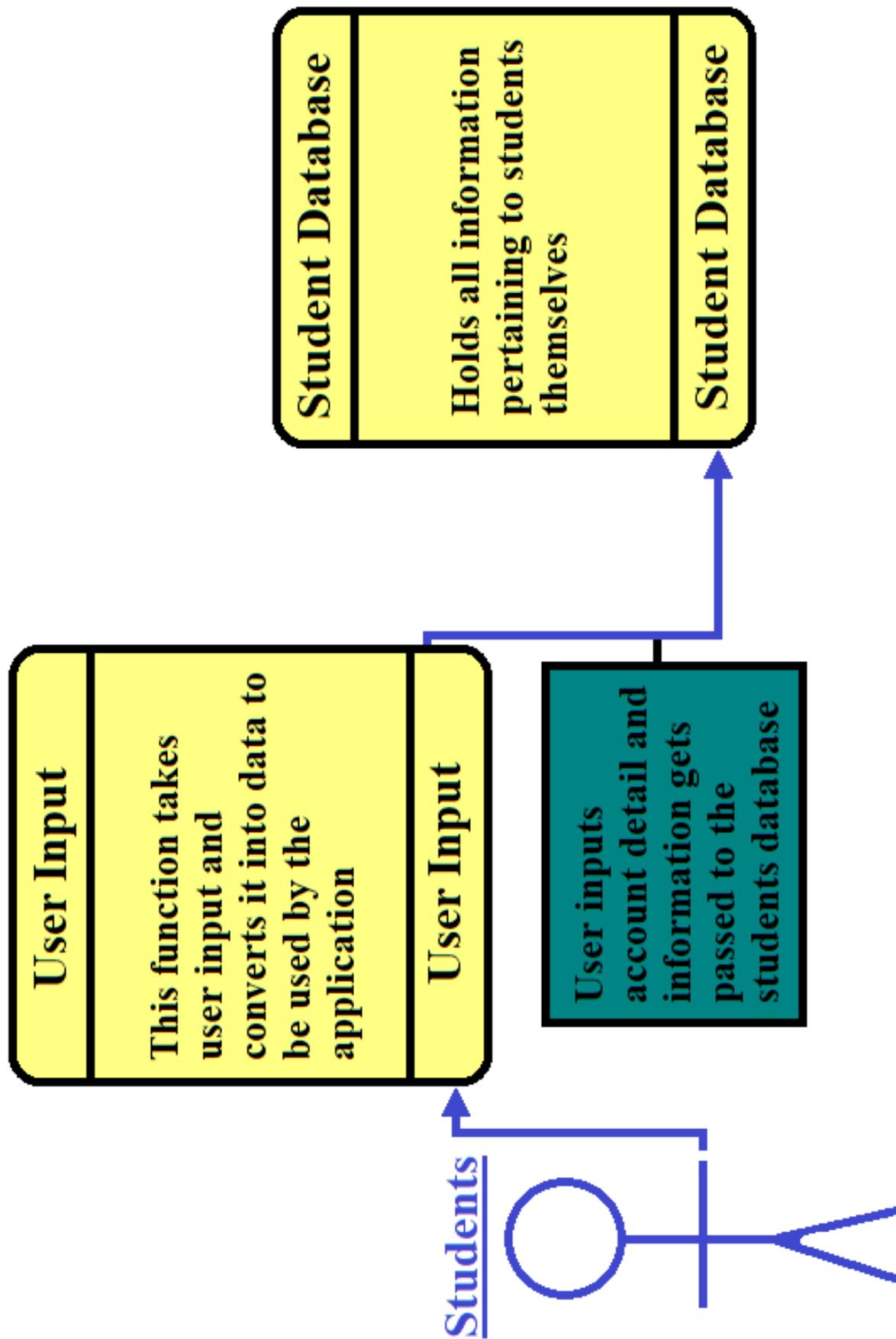
[Figure 3]



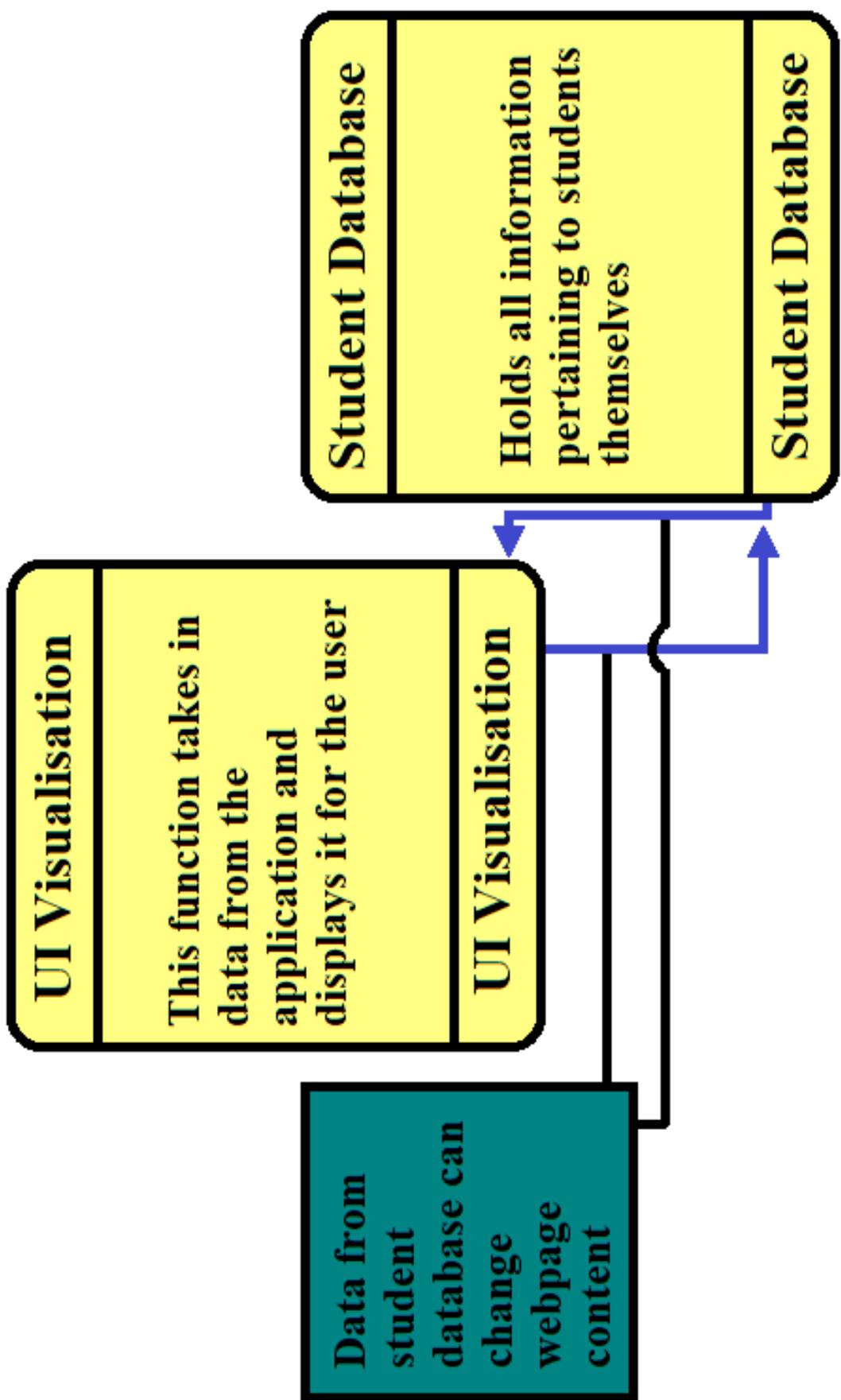
[Figure 4]



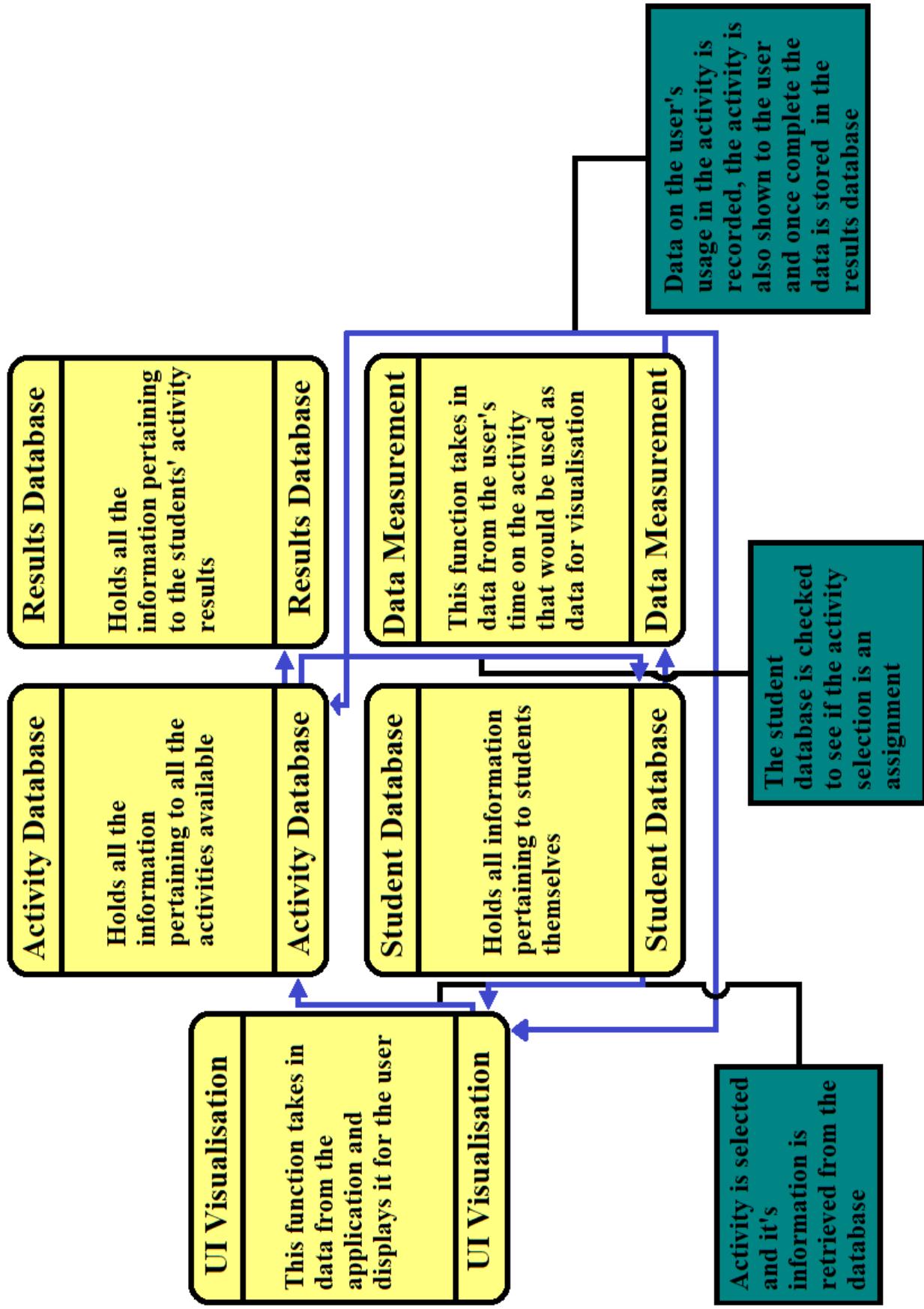
[Figure 5]



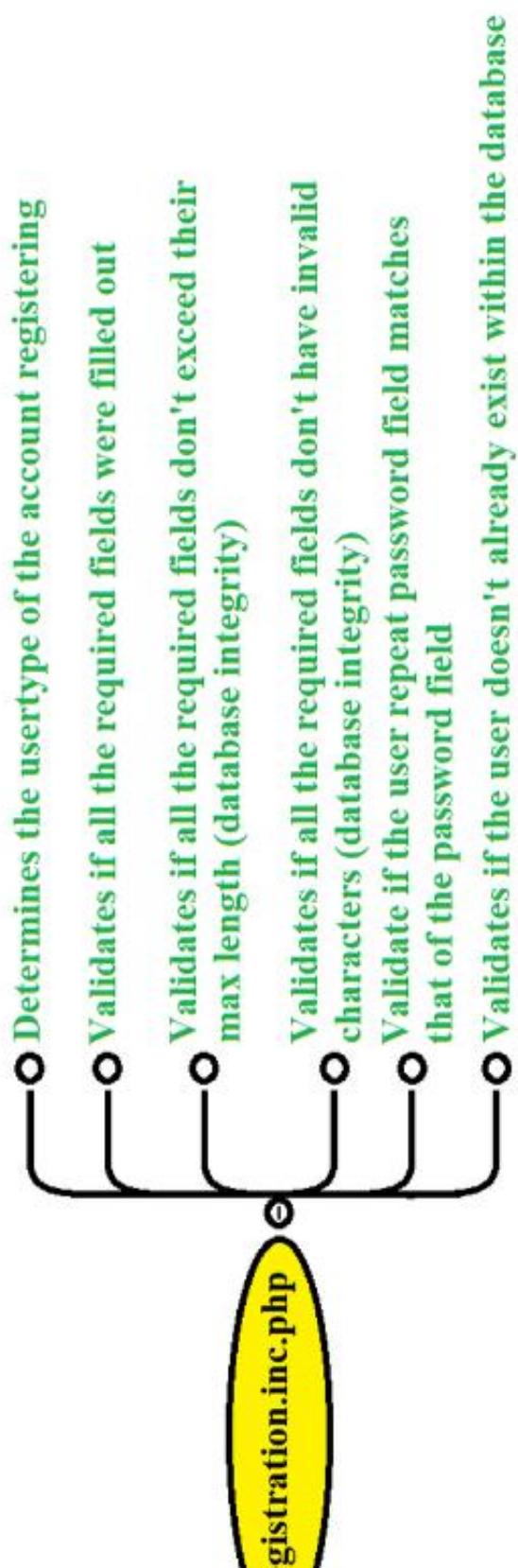
[Figure 6]

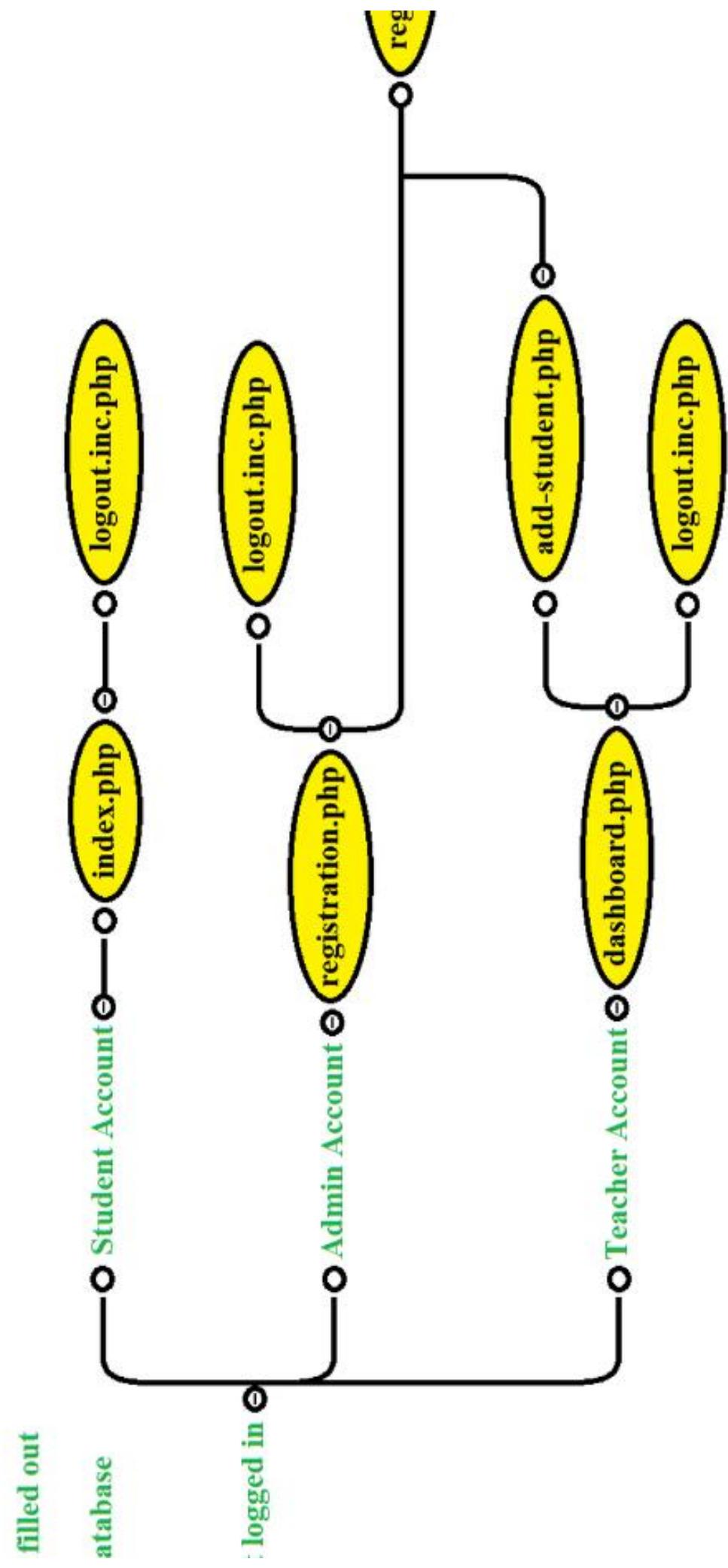


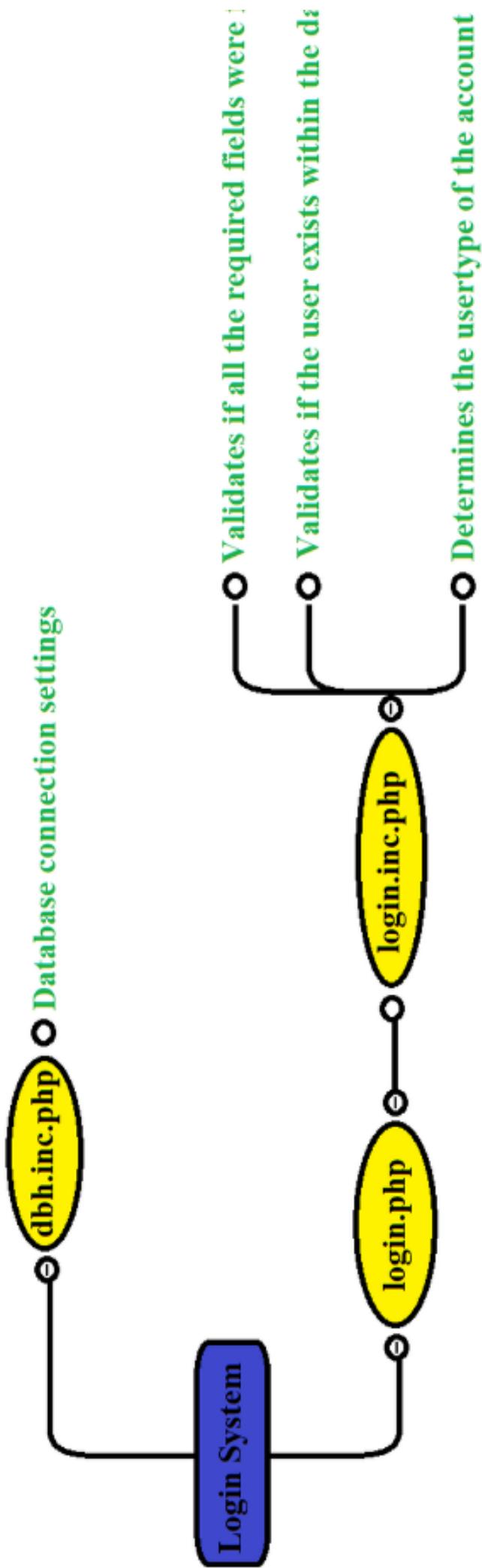
[Figure 7]



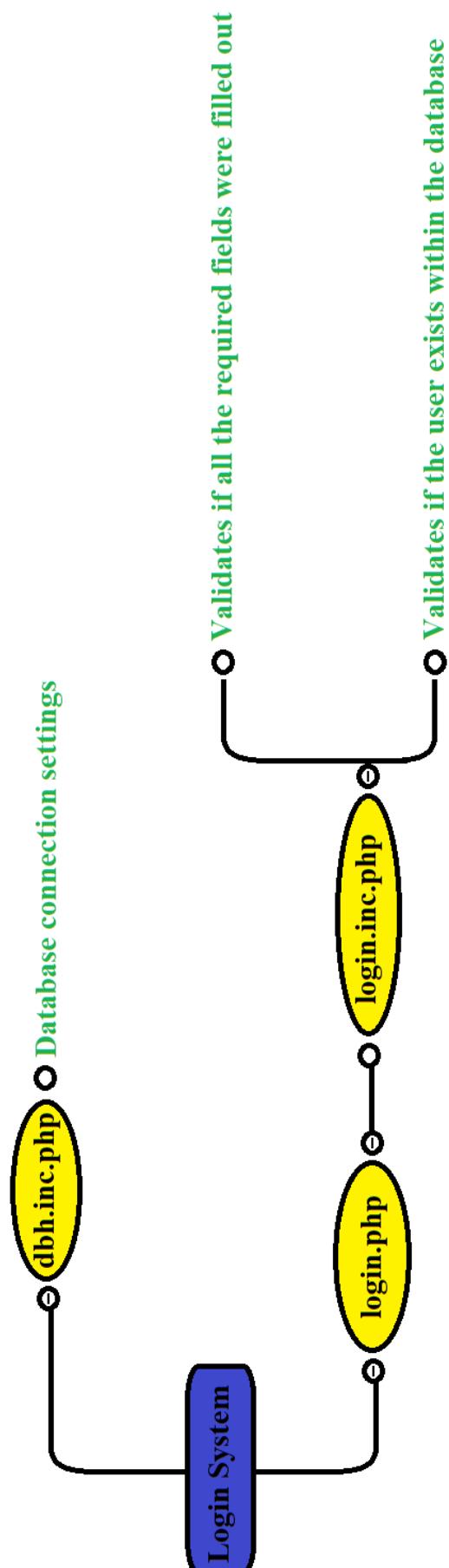
## Appendix B: System Architecture Diagrams - [Figure 1]



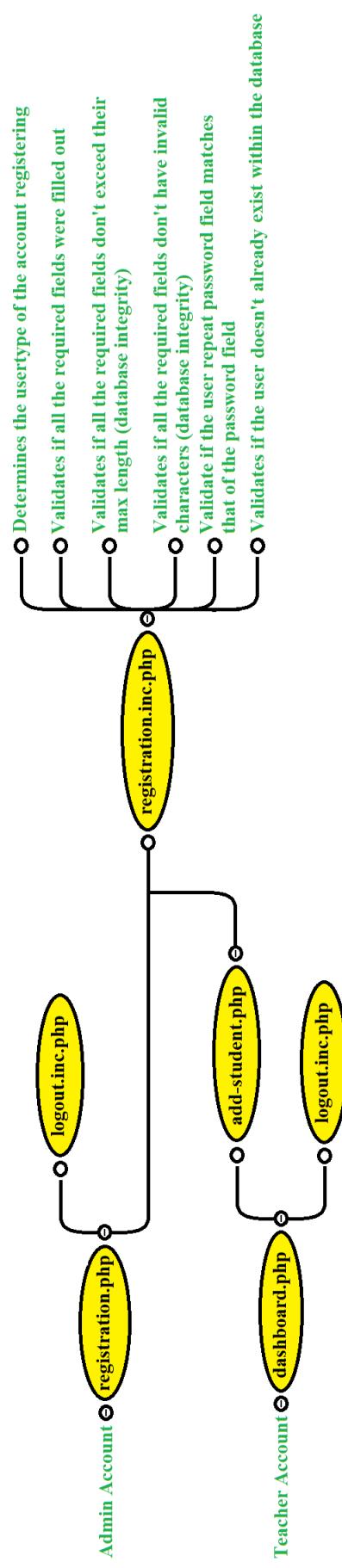




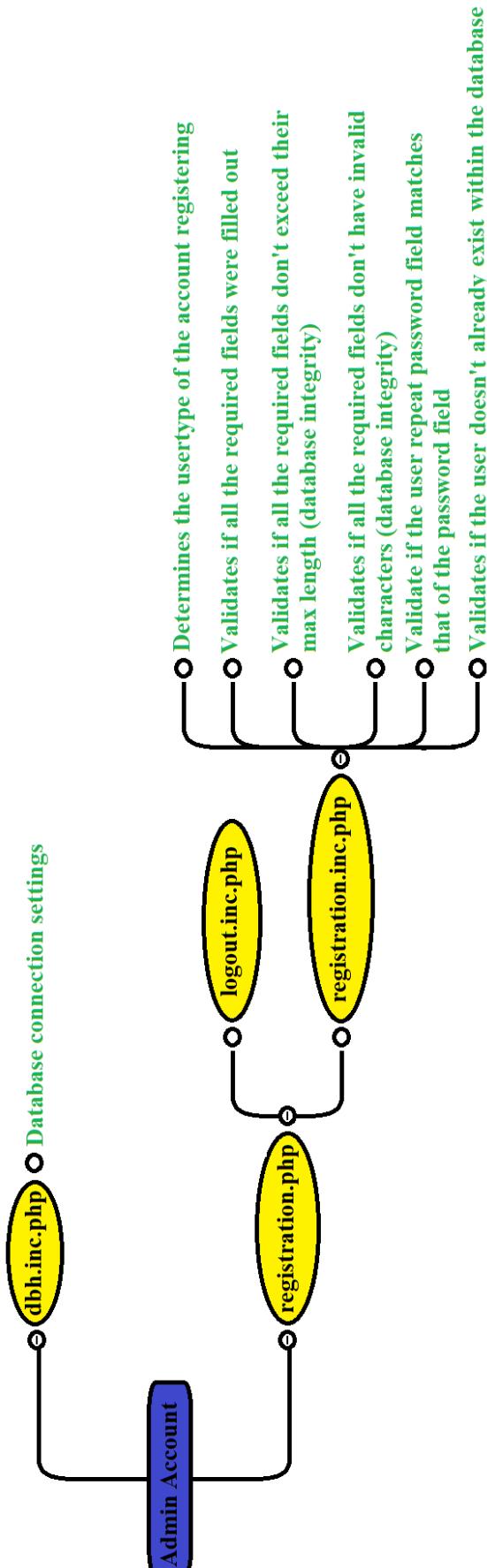
[Figure 2]



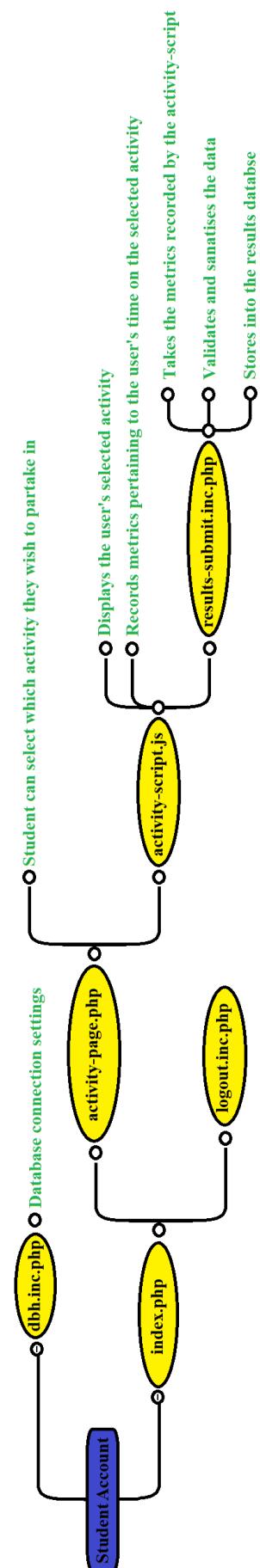
[Figure 3]



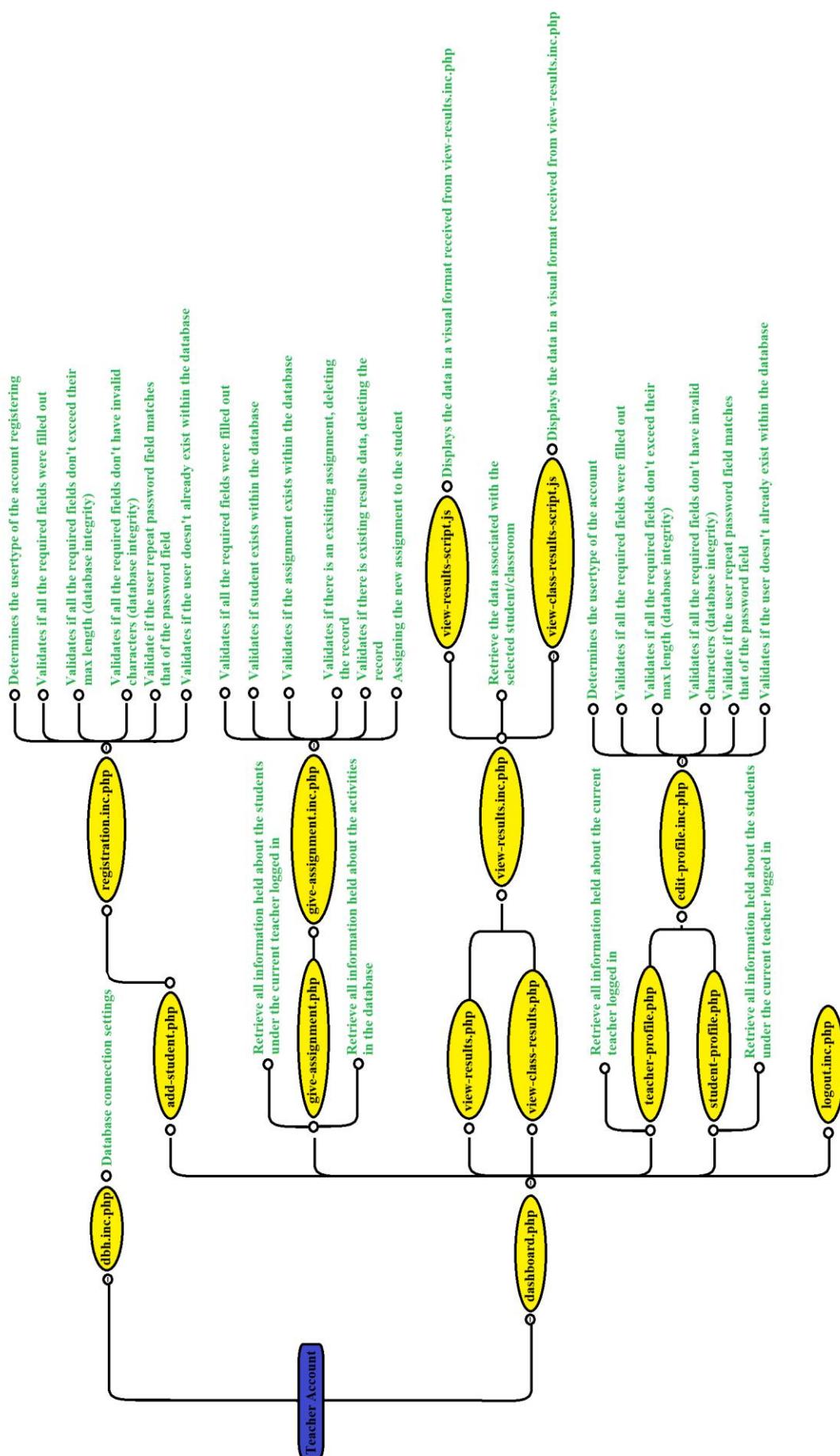
[Figure 4]



[Figure 5]

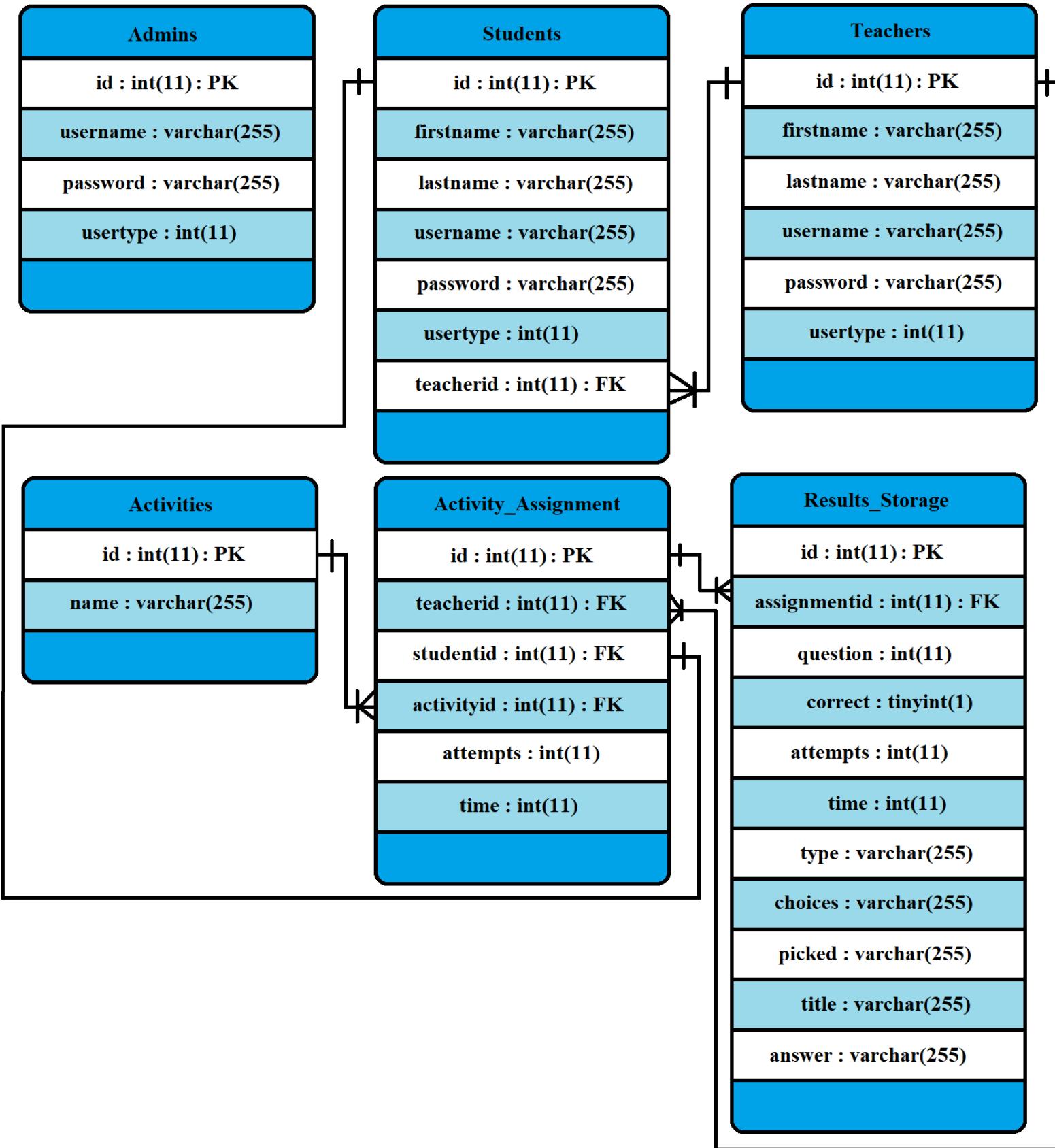


[Figure 6]

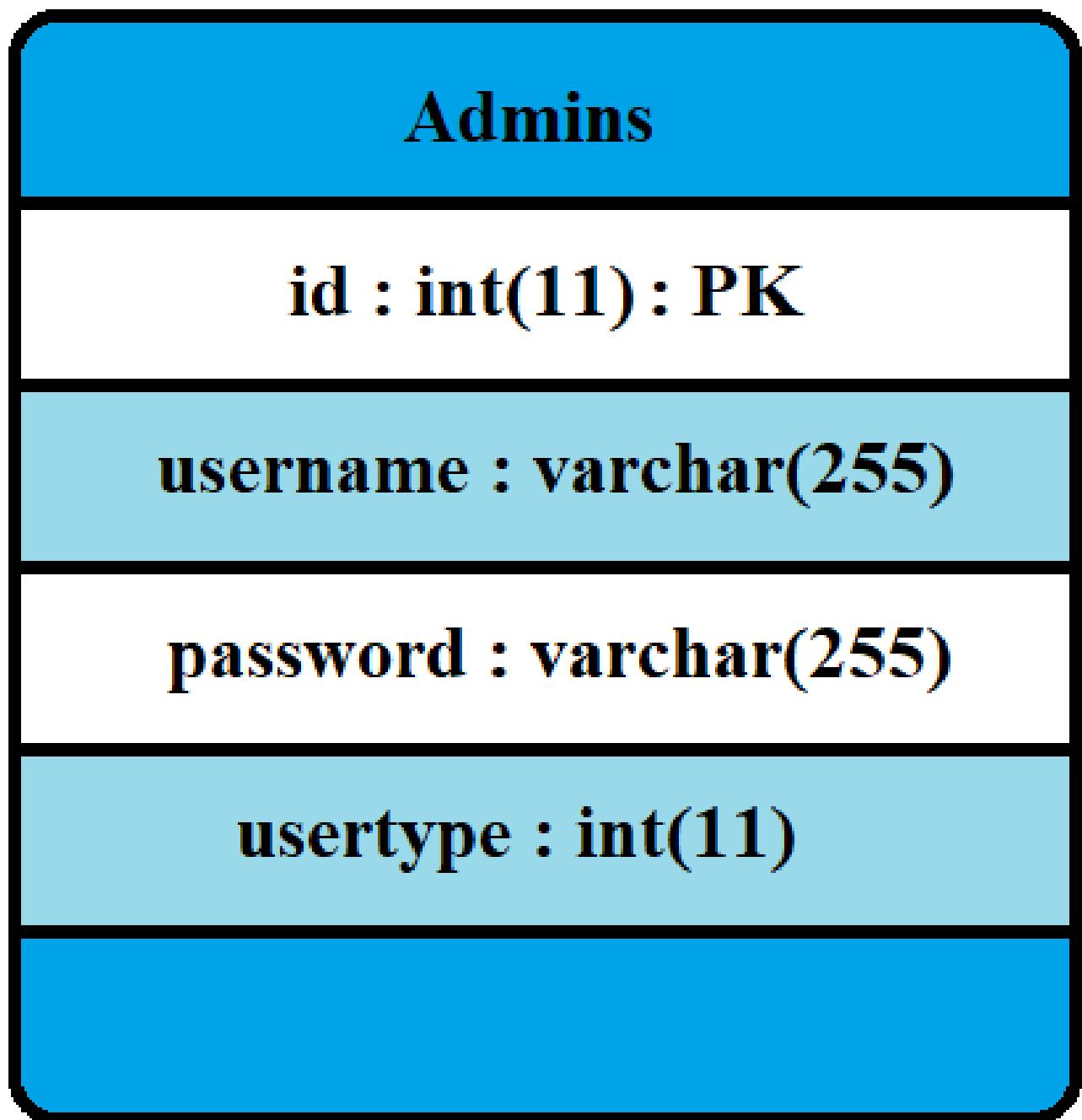


## Appendix C: Database Design

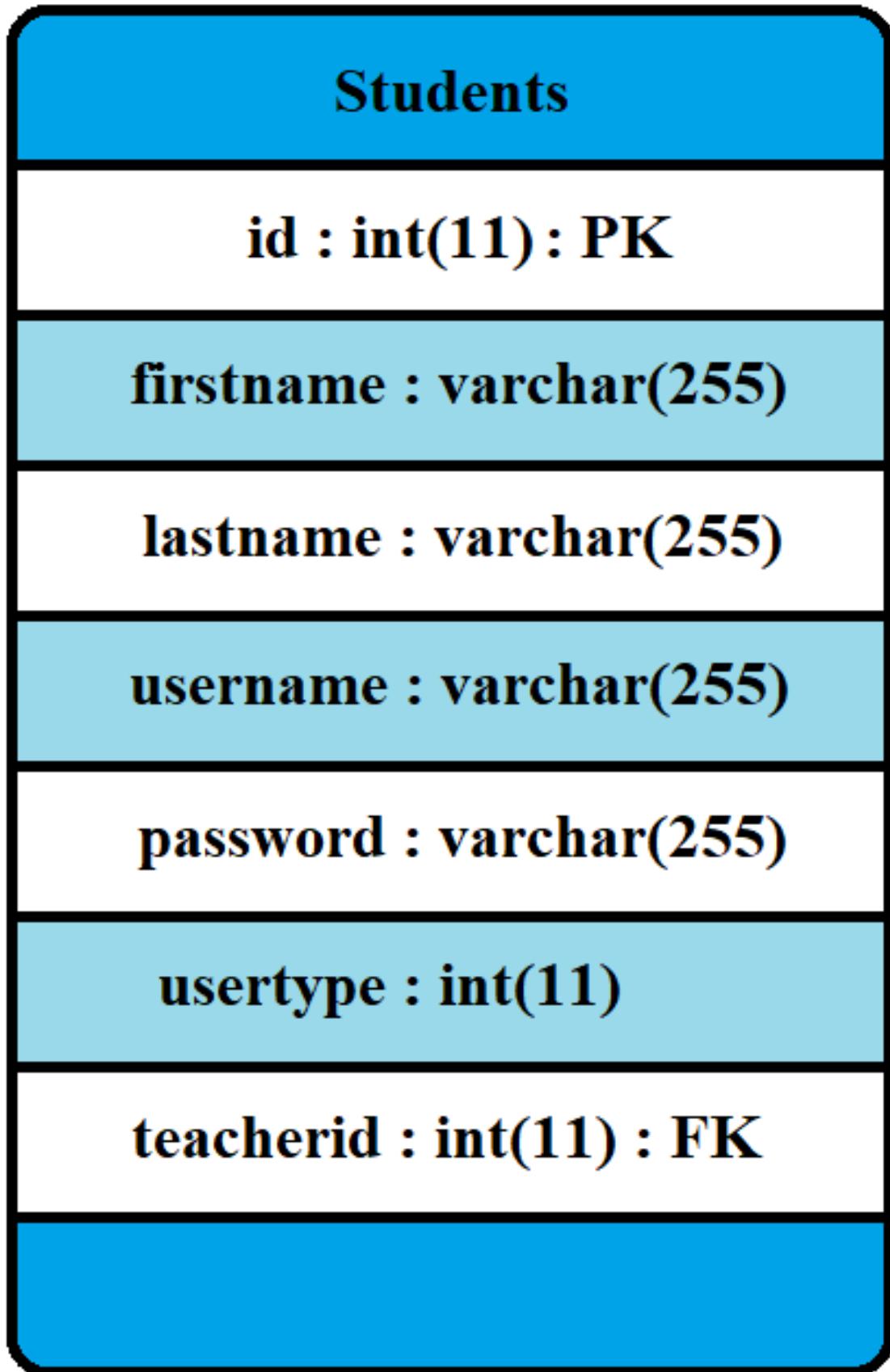
[Figure 1]



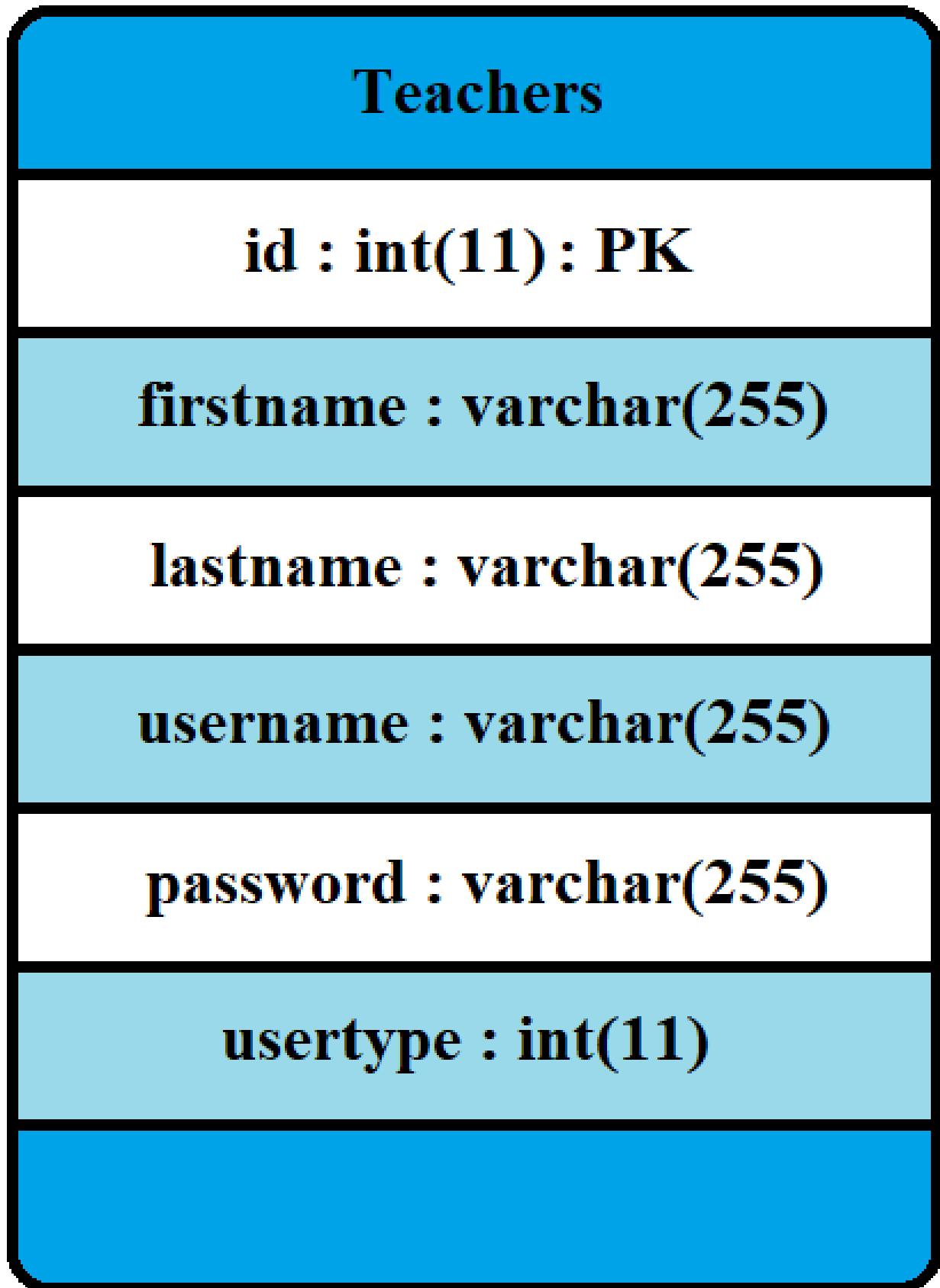
[Figure 2]



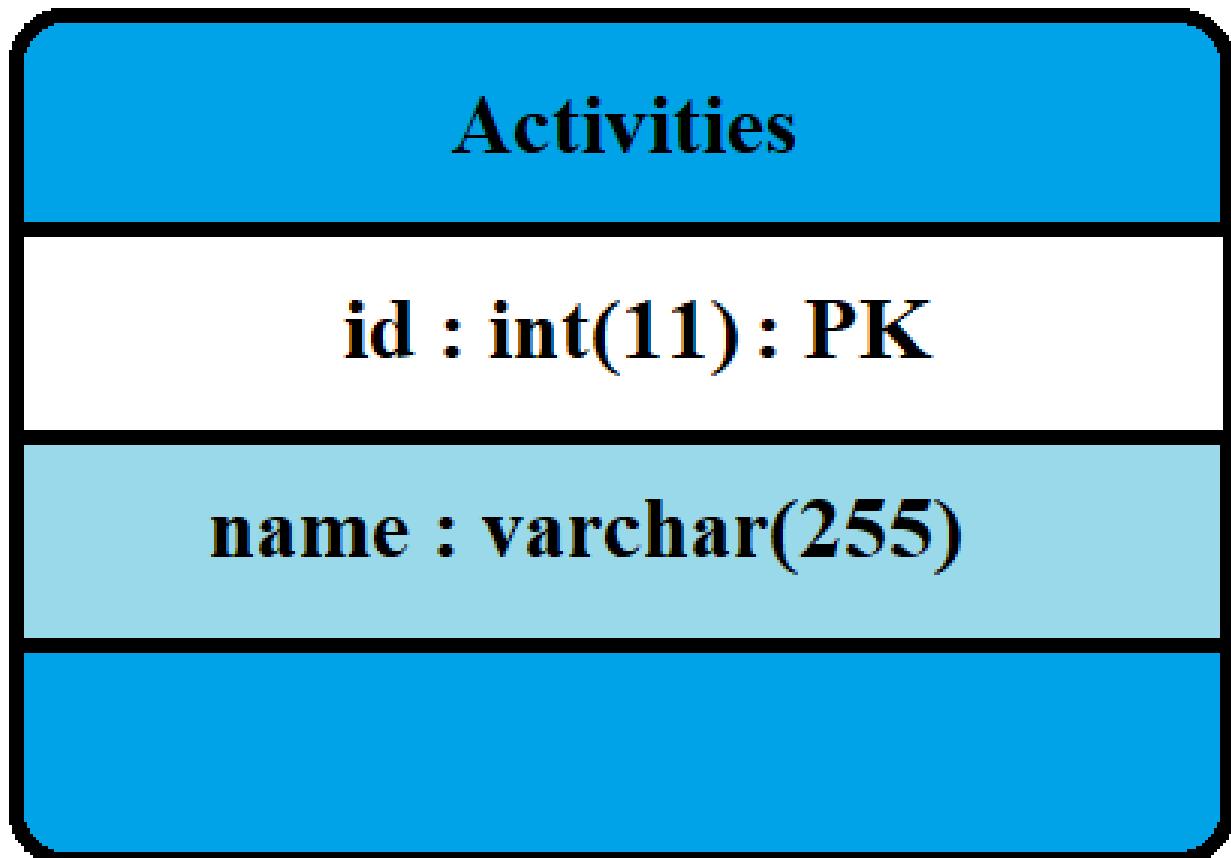
[Figure 3]



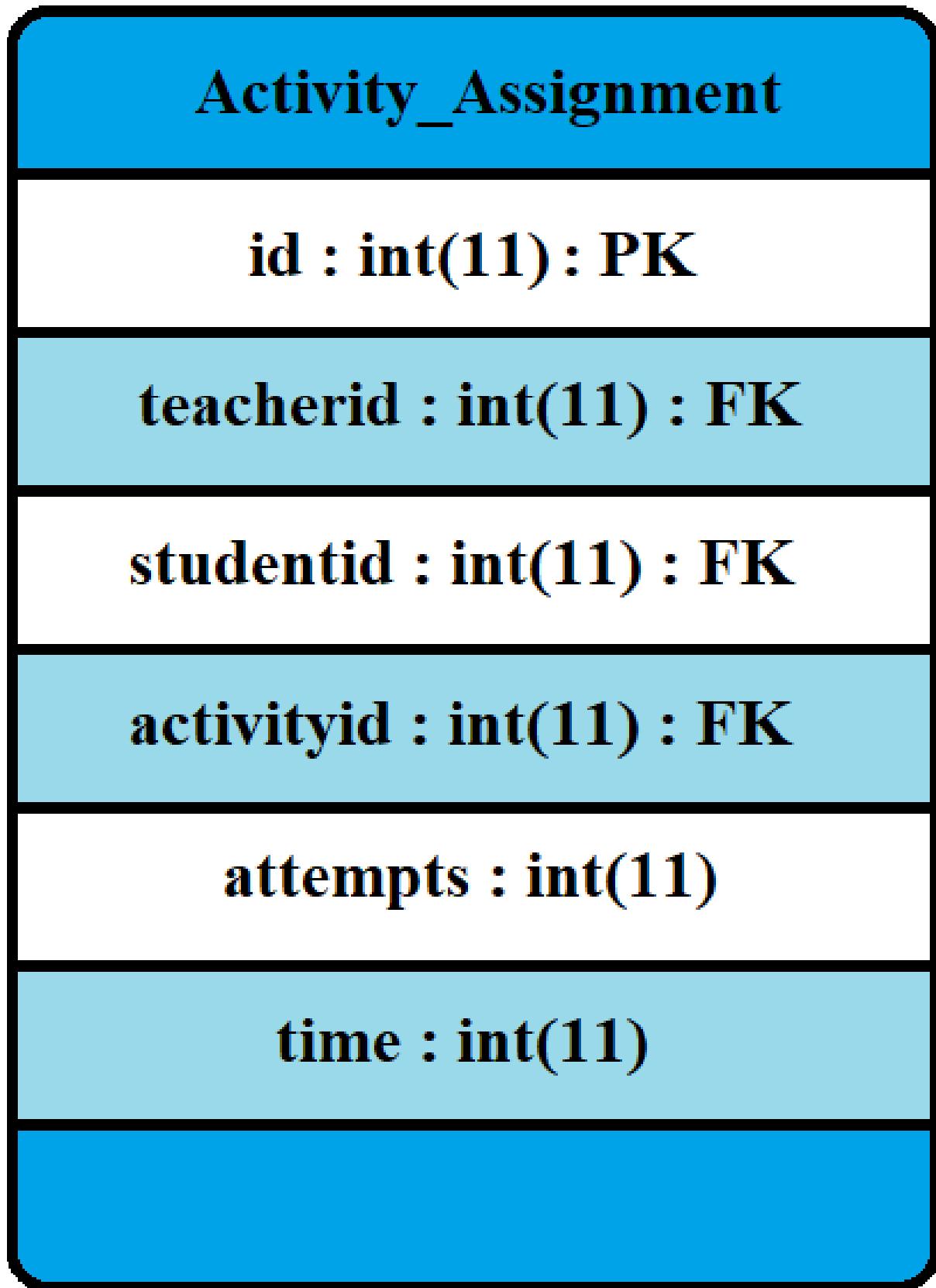
[Figure 4]



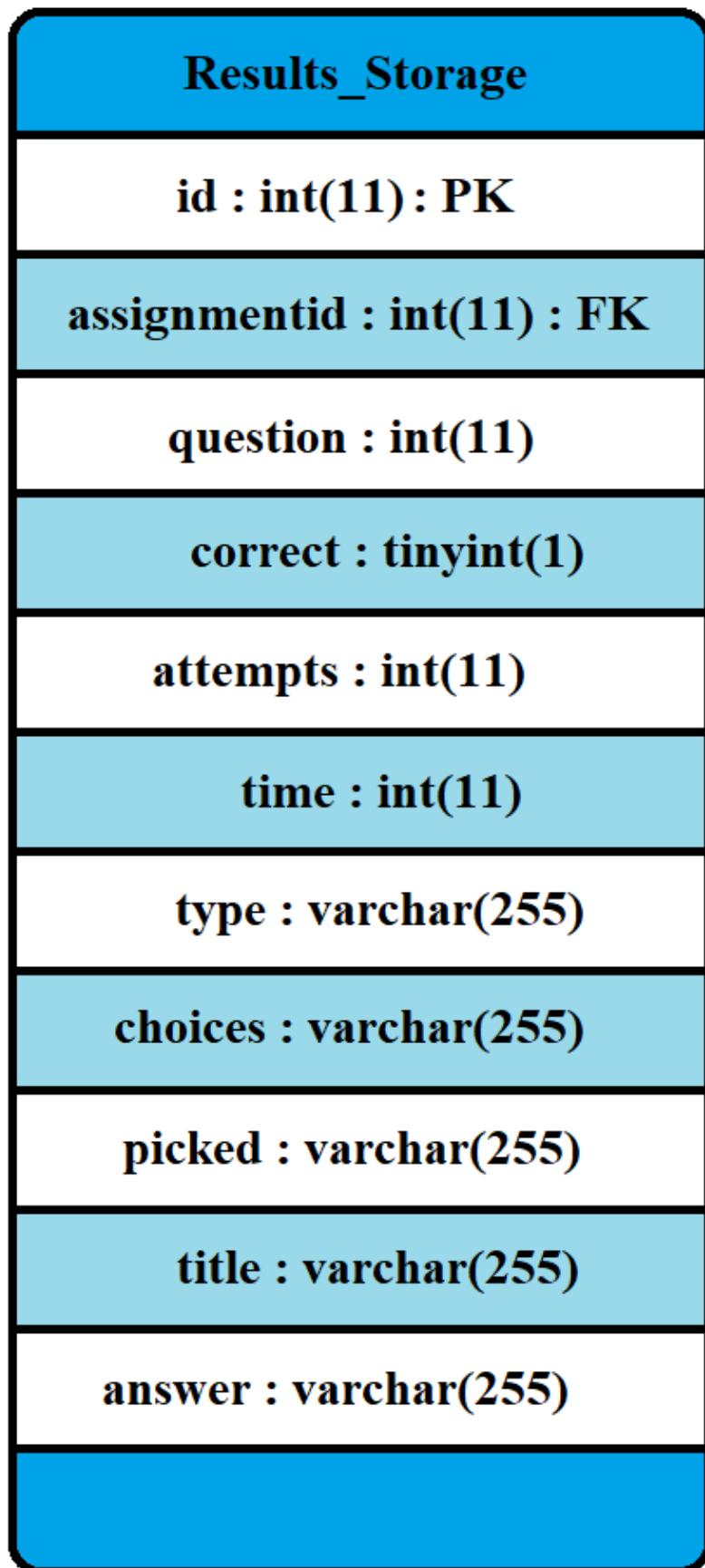
[Figure 5]



[Figure 6]

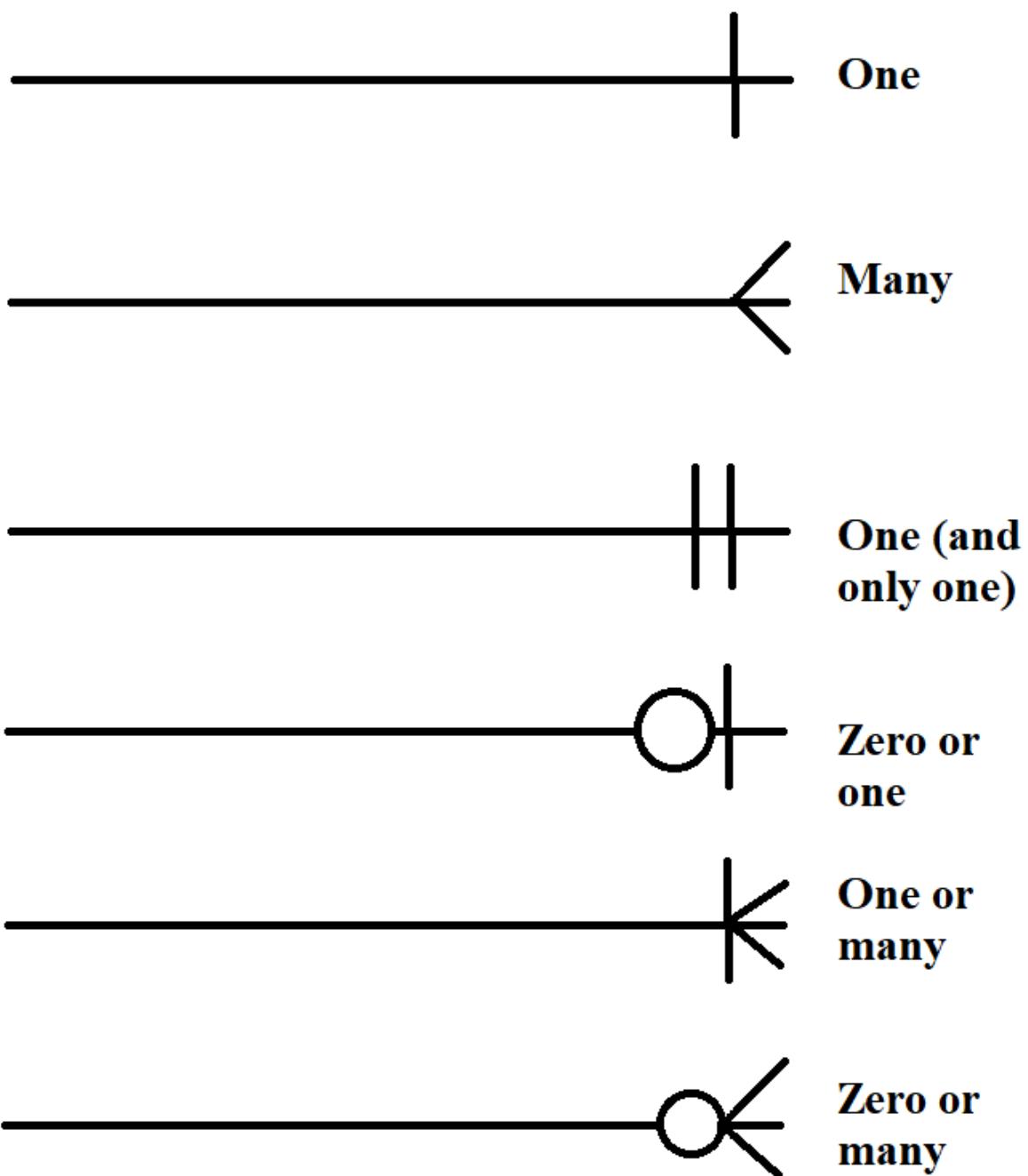


[Figure 7]



[Figure 8]

### ER Diagram Cardinality



[Figure 9]

<b>id</b>	<b>username</b>	<b>password</b>	<b>usertype</b>
1	admin	\$2y\$10\$flK7Tus8YZoXdVOvb3C5p.uD7QrvJ1t2ZxTI1yP4Ykfb...	1

## [Figure 10]

<b>id</b>	<b>firstname</b>	<b>lastname</b>	<b>username</b>	<b>password</b>	<b>usertype</b>	<b>teacherid</b>
1	Wilfredo	Dunn	wdunn001	\$2y\$10\$nU3pThkBtmbB7EeVrCbl.so/bq7Eb1iTj3Pwj6VJC...	3	1
2	Micheal	Miles	mmile001	\$2y\$10\$rAWgsc77RGL2Xmapdx7uOlzmv6X61ulwg6nyf1.is...	3	1
3	Lou	Fletcher	lflet001	\$2y\$10\$G2JwWKEkFxtvTddYQo4FO/RWbXqpXS/dGAVy1HY3u...	3	1
4	Erwin	Dixon	edixo001	\$2y\$10\$jsLe2rr4E2QaoOZpmUmrvDveyjkemKMRZxztOPWho...	3	1
5	Booker	Phillips	bphill001	\$2y\$10\$e0m3OVbOMe2WKsxqAahGq.bXQVA087qrnTavyNcNmP...	3	1
6	Tuan	Blackburn	tblac001	\$2y\$10\$q4e64DsjnEBs6595DZb8uA9mMudt634ymGyrT1hdWs...	3	1
7	Logan	Mason	lmaso001	\$2y\$10\$1zn0BAx.YKCXikNagiEBVernalE7dhBgG3Gdzwx54p8y...	3	1
8	Carey	Duarte	cduar001	\$2y\$10\$jK9Myai1gEL0FvbqrKqv5eacdKkqH8aumiGxt4AosU...	3	1
9	Kim	Lutz	klutz001	\$2y\$10\$L.bxNTPr5VfkreOpbjlgvOyMQLvQ5YDG3lcAeMsks5KO...	3	1
10	Seth	Shannon	sshann001	\$2y\$10\$cJieaFYiszUr1CX9RMHgv.2VeSEKkpgseMv5lBet4Fc...	3	1
11	Kraig	Orozco	koroz001	\$2y\$10\$S4srYqieHiCHVrkIw0/TROW81bjiumgoq5pcu1etTzc...	3	1
12	Rodrigo	Holder	rholder001	\$2y\$10\$hVaYVQvjkVjdzYteTLCxde5V7tfjvrzTFZ/vWoSw6AZ...	3	1
13	Lowell	Webster	lwwebs001	\$2y\$10\$zCgEXf9LgUBcMtOzVM9BnOkouUQJ17LXdco9XDTVsa...	3	1
14	Kerry	Strong	kstro001	\$2y\$10\$3F2okTQnwrfU5GhkLaLfc.6Z6B2nfUKw2cCa6U696MH...	3	1
15	Brittney	Baxter	bbaxt001	\$2y\$10\$pnp4Zb17S10qYBTQ1A0Mye82UvjCtw44X53xBIRWILD...	3	1
16	Lindsay	Marshall	lmars001	\$2y\$10\$H7m3s94v0Ahi01.33abccOSH9jA6h77ULRdJvwvlQq...	3	1
17	Araceli	Pena	apena001	\$2y\$10\$IHHXH0zDV5ckdT2QiYcx.zQyYT5EG2qLB9DgSmLWL...	3	1
18	Judith	Trujillo	jtruj001	\$2y\$10\$Wn4Lp5ihPaoc89rcEpgMexZmGow3OOw5n.Vs.qD8Qu...	3	1
19	Jenna	Bass	jbass001	\$2y\$10\$9wT1gPnwzhGDW1HcncE0rV7NKVNJ3TuLGoJnqDDYc...	3	1
20	Simone	Glenn	sglenn001	\$2y\$10\$GQdv2SCTLonfrQrzilK1e5mnSLRNrqeFH9DxXHgJbt...	3	1
21	Chrystal	Whitney	cwhit001	\$2y\$10\$0pmQnstE9fVJ90IJclNB.Naj.7H/pUSG7hzdbFpx...	3	1
22	Nicole	Mcknight	nmckn001	\$2y\$10\$TSYg38my22sdFnT72J7vq..3Mfh3QMFLBTN.qrl8Koc...	3	1
23	Erna	Khan	ekhan001	\$2y\$10\$A3BR9iwZqzlgKWjhMeEspeyESzLjh/CyrC3jl4BgK...	3	1
24	Aida	Duarte	aduar001	\$2y\$10\$LuAX2lqz33N132LMUwrB0.JgfzY5hg7JNhMI8kBfzhh...	3	1
25	Leigh	Nunez	lnune001	\$2y\$10\$HrFH5T0QQq5nae2CAzdrLuoPybfQx7JEV/ax9kIjrN...	3	1
26	Alisa	Juarez	ajuar001	\$2y\$10\$iDcQwA9oL6.rtUL.pceKGu53n5pX/ztWfvYxbG9DFL...	3	1
27	Leonor	Oliver	loliv001	\$2y\$10\$qls6ynvQHng6CKGNZ8uk.CwMoX6NqfNBG/LPDLXQV...	3	1
28	Deirdre	Joyce	djoyc001	\$2y\$10\$VD9wkiqHPWrOZ.9.dCJOReTYmDnlb4WGI.g8ylGyZx3...	3	1
29	Mae	Wiggins	mwigg001	\$2y\$10\$CkByh1w7Jrn5TXWhqKQtTO9Fzbk7A7sPTldsqjw/CL...	3	1
30	Diego	Hamilton	dhami001	\$2y\$10\$gb3zj93bc1i6X2yykOK9SOBH.6dE9zEuohGZRyjkAML...	3	2
31	Benito	Pena	bpena001	\$2y\$10\$0m/Zhr9XvEv6xrnavFqU.o/dRDICZA/r8HyF0aa0n...	3	2
32	Brooks	Lyons	blyon001	\$2y\$10\$6P0Aa/9ipkAyUoowCkxR.B74MxhCeq2on3gPzpLPdz...	3	2
33	Lorenzo	Meza	lmeza001	\$2y\$10\$DrUhW2MxttMjMceTJ/Y3eiC7Wz7GvugXZ.mJpg7GI...	3	2
34	Werner	Gilbert	wgilb001	\$2y\$10\$QFafEmntCxazFDR9JnOvfurYtknnPSyv5aZuc5C4...	3	2
35	Adrian	Huang	ahuan001	\$2y\$10\$YaqGqHxSO6AxPBQWmlf.70jT92oNBHCqBepFEzSKYoV...	3	2
36	Jorge	Cameron	jcame001	\$2y\$10\$TOgNr1cyWRb9L1Bb6wSJ6OYjis9zHEBC8DRIE0qA9...	3	2
37	Minh	Bush	mbush001	\$2y\$10\$mC6pMia/qB6ZQGKYb2f09OClapn14rHqr5Kr059GrU...	3	2
38	Gale	Salinas	gsali001	\$2y\$10\$oomKbSMhltw6quSVgsRAv.i.lxS5Hx/Gf7wnEodlBDP...	3	2
39	Faustino	Carroll	fcarr001	\$2y\$10\$Y72ptGXc8dqoqN9sTG4L7IOEZRTdWJQya9s.8anwrcvO...	3	2
40	Warren	Hansen	whans001	\$2y\$10\$h904UShdHg0u0TQdf4YXnOONoIFNZQMPDj3eKYQM8k...	3	2
41	Kim	Compton	kcomp001	\$2y\$10\$rrUuuG0bEkmdn3ZyCCg8uXdcLcv1/yqSM11tmqnf...	3	2
42	Cole	George	cgeor001	\$2y\$10\$eW0KWyNCeCeeGR0rhQNNeojRWqvV.XHLNGALpJkEHKG...	3	2
43	Granville	Cuevas	gcuev001	\$2y\$10\$6sH23k0nlx6k3s65HWC12.8lnK/SkrPBwn.M.Lva5Vr...	3	2
44	Gilberto	Carpenter	gcarp001	\$2y\$10\$W2B0L44yue5s4DNaVmlufesEaa371Zv2RpnD19RfbB...	3	2
45	Matilda	Serrano	mserr001	\$2y\$10\$WFSxEn8kK8CnQKXlkPc3m.r.e2y4miGpzFWXQHOozp...	3	2
46	Barbra	Russell	bruss001	\$2y\$10\$scszX.4zTSj7TRh2LLgtY0.RFVshsYO4T2vbApRvg5v...	3	2
47	Kelsey	Kemp	kkemp001	\$2y\$10\$2ArhUWft8181TyGYrRSqCOFZdh5HVTCmlbmhRykork...	3	2
48	Fern	Sims	fsims001	\$2y\$10\$PiLnJFieEphy9hbNQDk53OFwD.ujT40alk6NceThW7...	3	2
49	Dorthy	Beltran	dbelt001	\$2y\$10\$OV9lHomhJ7CRCr0E0TQogRGITZwyG3uZinoAUrkEy...	3	2
50	Madeleine	Green	mgree001	\$2y\$10\$Q.9je9vVoxCoDhJaaHtDeiO5CDF5Jsv5qvn7s3Uov...	3	2
51	Jordan	Choi	jchoi001	\$2y\$10\$Syibf2SqpjZ8Y6c0jh6udlfYnPy1Zm34WZ34Oh2en...	3	2
52	Margo	Rocha	mroch001	\$2y\$10\$plxWnlmFerlUqpZ3vzeL.etznXPfbwEPIQPv9bPj.u...	3	2
53	Angela	Larson	alars001	\$2y\$10\$/b08qntv0O0UlHv2RzgbGeAeUTUmCGket3g1dPovK...	3	2
54	Lynne	Zamora	lzamo001	\$2y\$10\$6xR1CK.cBkn.374dX8b4xeSjbnEc2VyARTbmlmhT8r...	3	2
55	Debora	Hood	dhood001	\$2y\$10\$K7RwpTwxRV2zrqVTRS7ZKuiOENACIUH2iXrMhSVoZx...	3	2
56	Alicia	Mitchell	amitc001	\$2y\$10\$6b38.FABgkMmSv189FzGUOHjfk8Jbjp7p9DX4uB4We...	3	2
57	Rhonda	Wilkinson	rwilk001	\$2y\$10\$ic7jw3ZhrZYo0LMyPXiM/uq8cpGmg2EkPqaHupy1RG2...	3	2
58	Melody	Velez	mvele001	\$2y\$10\$Ga3h2dVtvdXC6m9Dqk0eOmiWkZbjgwRCUhnMeli8n.U...	3	2
59	Brandie	Glenn	bglen001	\$2y\$10\$M5JkZid5HEa4estJ68rXi.ksM7W5Us6FOCinu4K/SVK...	3	2

**[Figure 11]**

<b>id</b>	<b>firstname</b>	<b>lastname</b>	<b>username</b>	<b>password</b>	<b>usertype</b>
1	Kate	Gardner	kgard001	\$2y\$10\$rEgYjesHLDDIQ09zMTLIfJR6YAXRJ2tT2rb2.maOc...	2
2	Bradley	Morgon	bmorg001	\$2y\$10\$KB6WHsY5WJtN82qQSS7hwuvs5xQVcenPZy13IEABajS...	2

**[Figure 12]**

<b>id</b>	<b>name</b>
1	Mr Paddington's Quiz-a-thon

**[Figure 13]**

<b>id</b>	<b>teacherid</b>	<b>studentid</b>	<b>activityid</b>	<b>attempts</b>	<b>time</b>
8	1	1	1	13	176
10	1	6	1	1	156

[Figure 14]

id	assignmentid	& 1	question	correct	attempts	time	type	choices	picked	title	answer
14		8	0	0	3	62	Spelling	Switch, Sneath, Swiche	Switch, Sneath, Swiche	Remember to _____ off the light	Switch
15		8	1	1	49	Understanding	The floor was wet. The food was too hot. He didn't... She missed her bus and had to walk			Why did Micheal drop his tray?	The floor was wet
16		8	2	0	3	1	Understanding	She missed her bus and had to walk. She decided to...		Her school was very far away. Her school was very ... Why was Jamila tired when she got to school?	She missed her bus and had to walk
17		8	3	0	3	1	Spelling	Briov, Brother, Brudduh, Bruvver	Bruduh, Bruduh, Bruduh	I love my older _____	Brother
18		8	4	0	3	1	Spelling	P, N, B, Y	B, B, B	A, _ L E	P
19		8	5	0	3	1	Spelling	Fraction, Fraktion, Fraktun, Fuhraktion	Fraction, Fraktion, Fraktun	In maths, we learnt what a _____ is	Fraction
20		8	6	0	3	1	Punctuation	This puzzle is hard. Where is my puzzle. That puzz...		Which sentence is a question?	Where is my puzzle
21		8	7	0	3	1	Grammar	cool, girl, Mayesha, friend	Majesha, Majesha, Majesha	My friend, Mayesha, is a cool girl	cool
22		8	8	0	3	1	Grammar	so, when, where, if	where, where, where	We will make it on time _____ we leave right now if	
23		8	9	1	1	1	Grammar	Dirty, Dinted, Dirty, Dirt	Dirty	Jason's clothes were _____ after football practice. Dirty	
24		10	0	1	2	12	Grammar	if, when, where, so	so, if	We will make it on time _____ we leave right now if	
25		10	1	0	3	13	Understanding	She missed her bus and had to walk. School is so b...	School is so boring. She decided to run to school...	Why was Jamila tired when she got to school?	She missed her bus and had to walk
26		10	2	1	2	11	Spelling	Bruduh, Brother, Bruvver, Brower	Bruduh, Brother	I love my older _____	Brother
27		10	3	1	1	33	Understanding	The floor was wet. The food was too hot. He tripe...	The floor was wet	Why did Micheal drop his tray?	The floor was wet
28		10	4	1	3	14	Spelling	Switch, Switche, Sneath, Swich	Switch, Switche, Switch	Remember to _____ off the light	Switch
29		10	5	1	1	7	Punctuation	I hate puzzles. That puzzle is mine. Where is my p...		Which sentence is a question?	Where is my puzzle
30		10	6	1	1	7	Spelling	P, N, B, Y	P	A, _ – L E	P
31		10	7	0	3	18	Understanding	8 years, 500 years, 5 years, 10 years	8 years, 5 years, 10 years	How long have ducks been living as pets for?	500 years
32		10	8	1	3	27	Spelling	Fuharkzion, Fraktion, Fraction, Fraktion	Fuharkzion, Fraktion, Fraction, Fraktion	In maths, we learnt what a _____ is	Fraction
33		10	9	1	1	9	Punctuation	!, ?, ..	!	Simon's cat is so cute	!

## Appendix D: Application Design Concepts

[Figure 1]



[Figure 2]

BRAND

**WELCOME, WILFREDO**

What would you like to learn today?

Professor Paddington:  
Quiz-a-Thon

Place Holder

Place Holder

Place Holder

Place Holder

Place Holder

Place Holder

Our Mission

The primary role of a teacher in a classroom is to impart knowledge to their students. Teachers are given a curriculum that they must follow and ensure every child placed in their care understands the content to a satisfactory degree. This, however, does present a problem; with ever-increasing classroom sizes, teachers are having greater trouble catering to each child's needs. Furthermore, testing and evaluating students is becoming a more difficult endeavour. This application attempts to act as a possible solution to these problems, making the assessment of children within the classroom easier and more streamlined for teachers while keeping it fun and enjoyable for the children.

**About**

[Our Mission](#)  
[Meet the Team](#)  
[Partners](#)  
[News](#)  
[Teachers & Parents](#)

**Support**

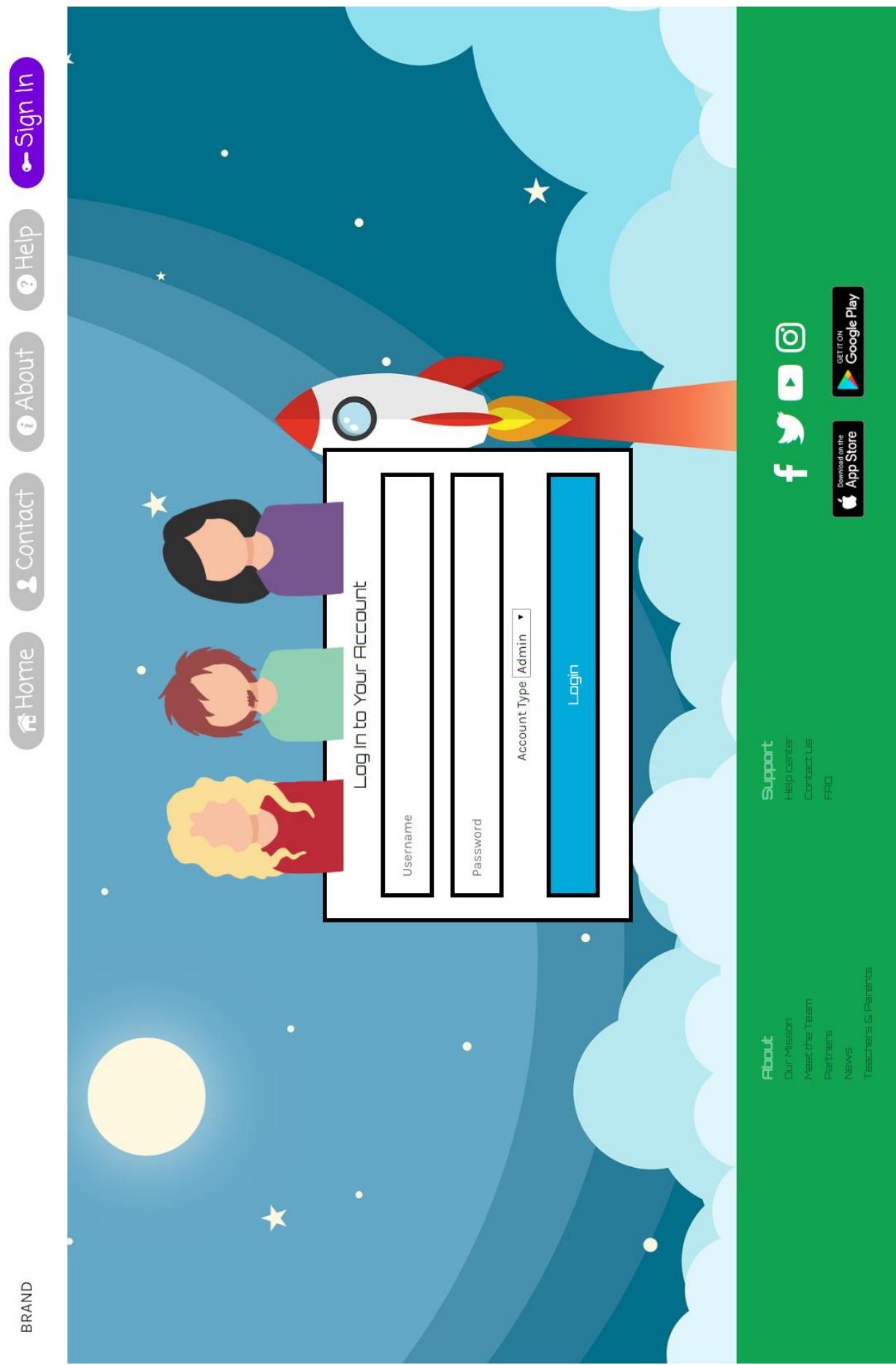
[Help center](#)  
[Contact Us](#)  
[FAQ](#)

Download on the App Store Get it on Google Play

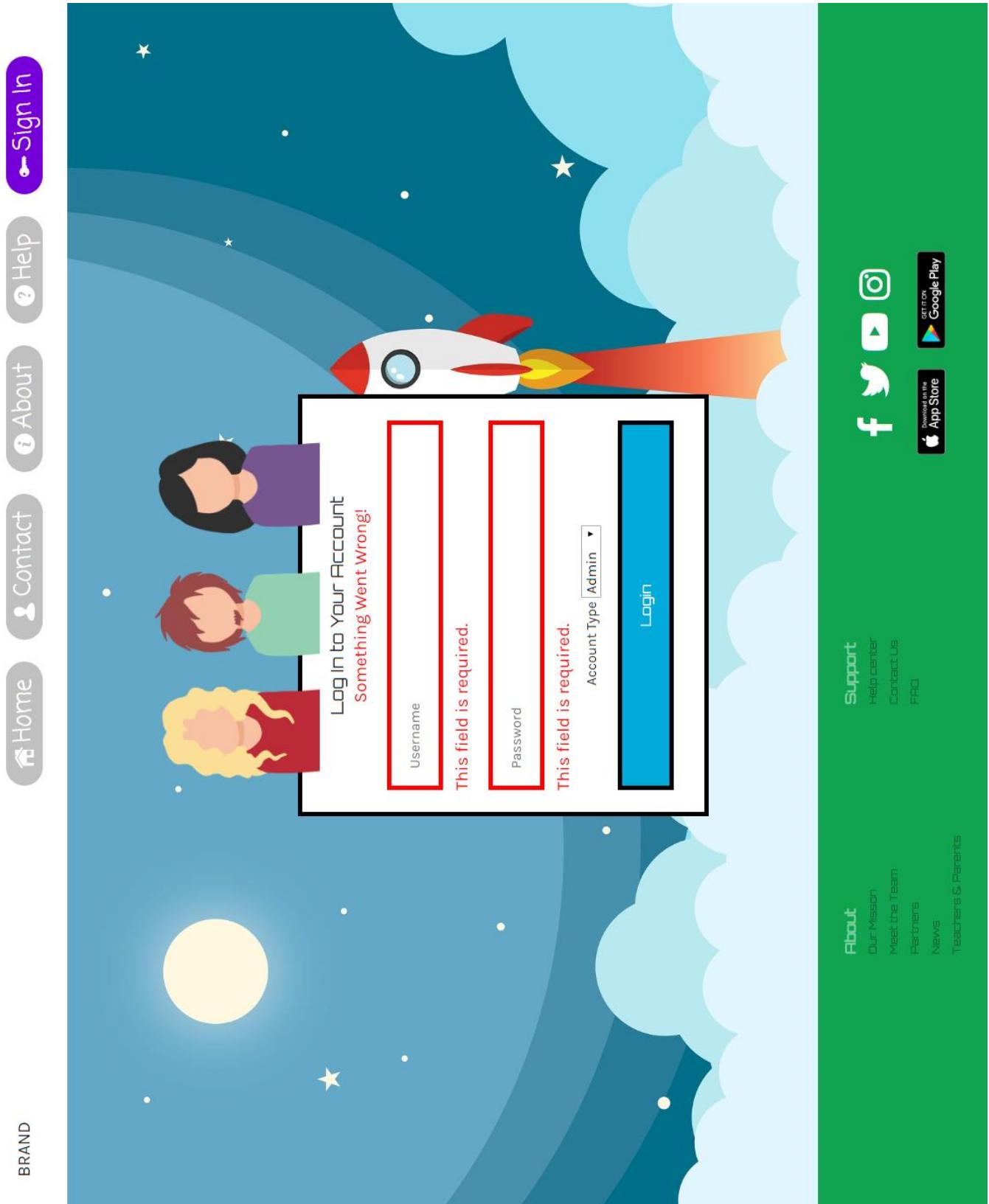
[Figure 3]



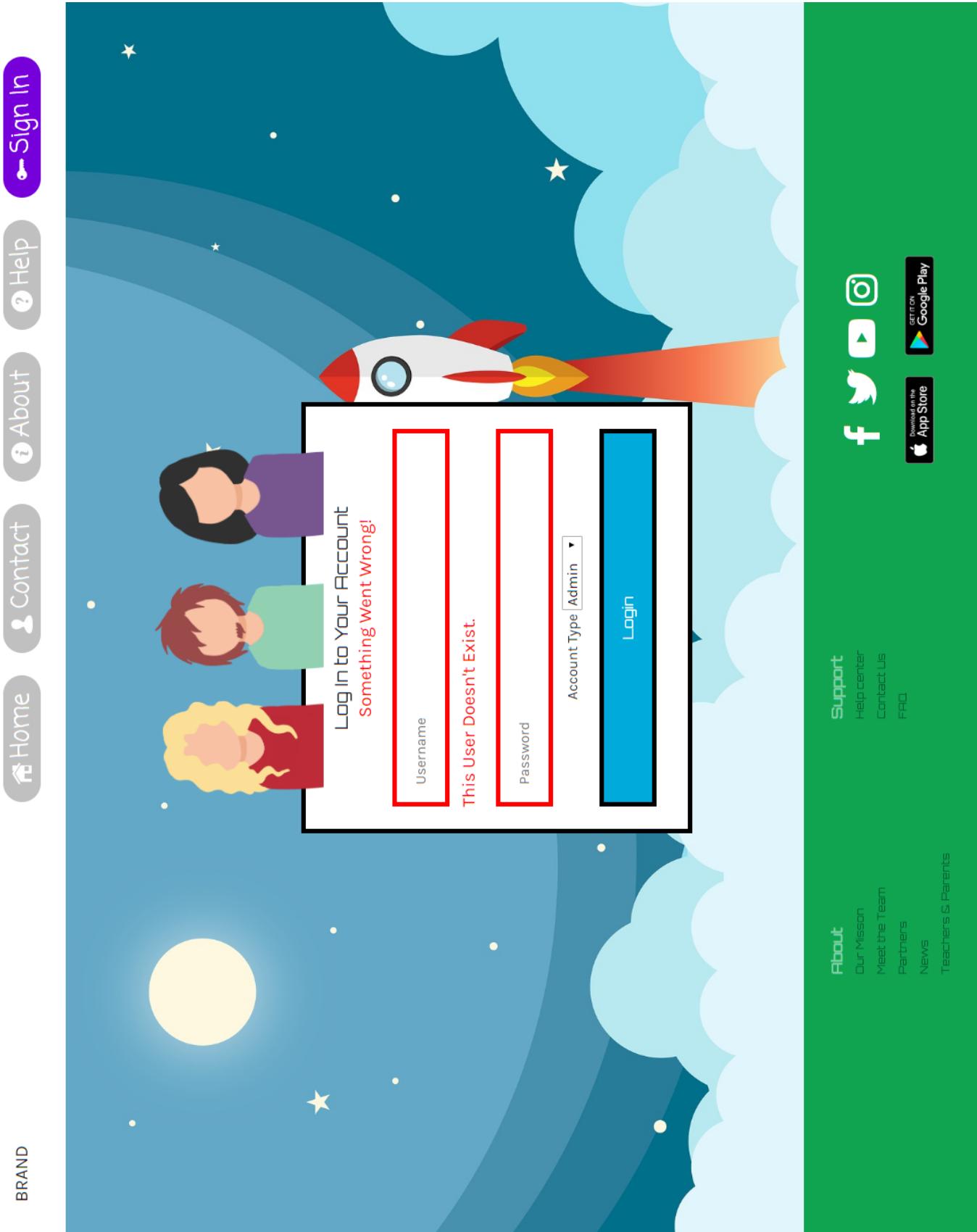
[Figure 4]



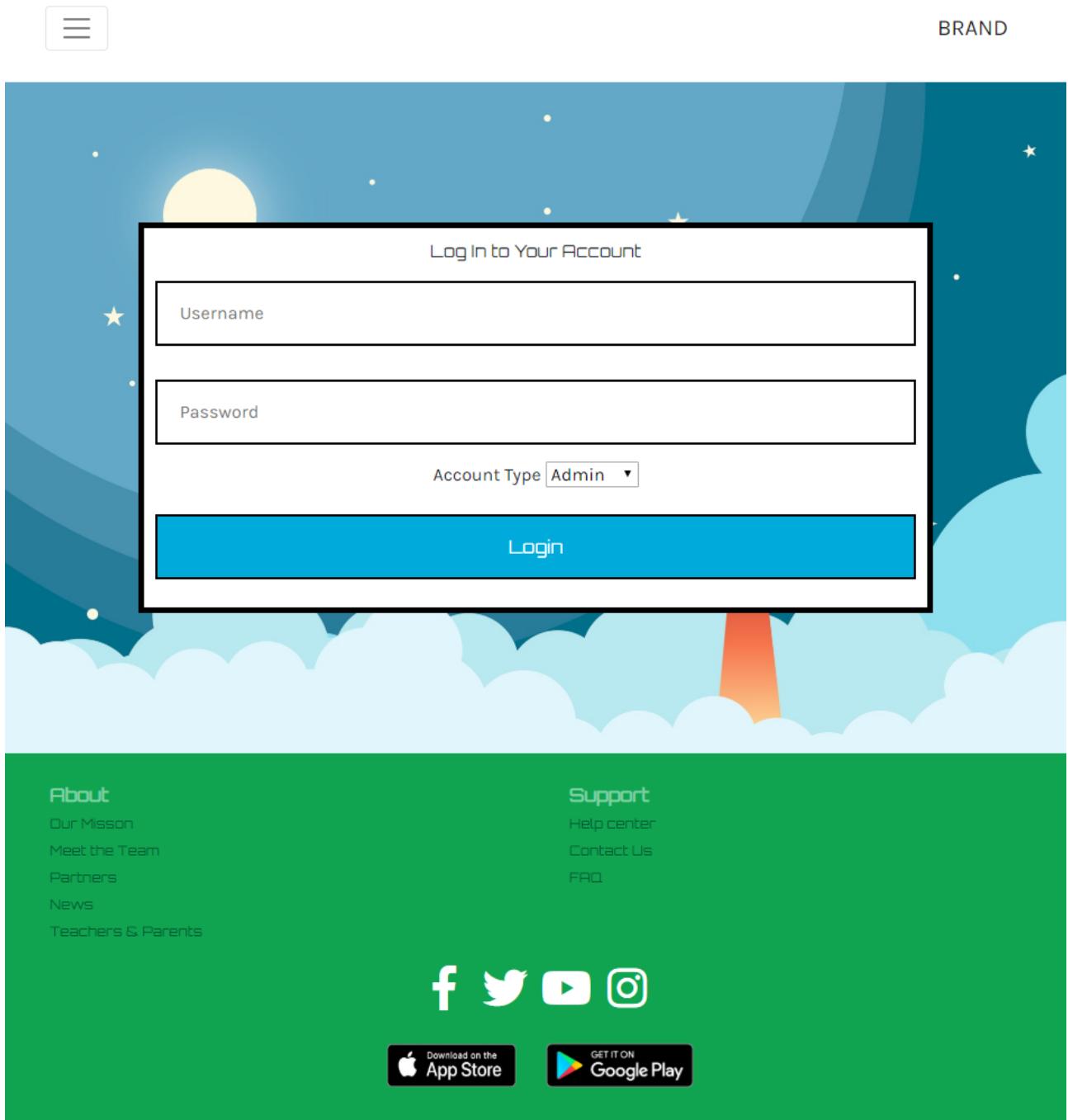
[Figure 5]



[Figure 6]



[Figure 7]



[Figure 8]

BRAND

## Log In to Your Account

Username

Password

Account Type Admin ▾

Login

About

- Our Mission
- Meet the Team
- Partners
- News
- Teachers & Parents

Support

- Help center
- Contact Us
- FAQ

f

Download on the App Store

GET IT ON Google Play

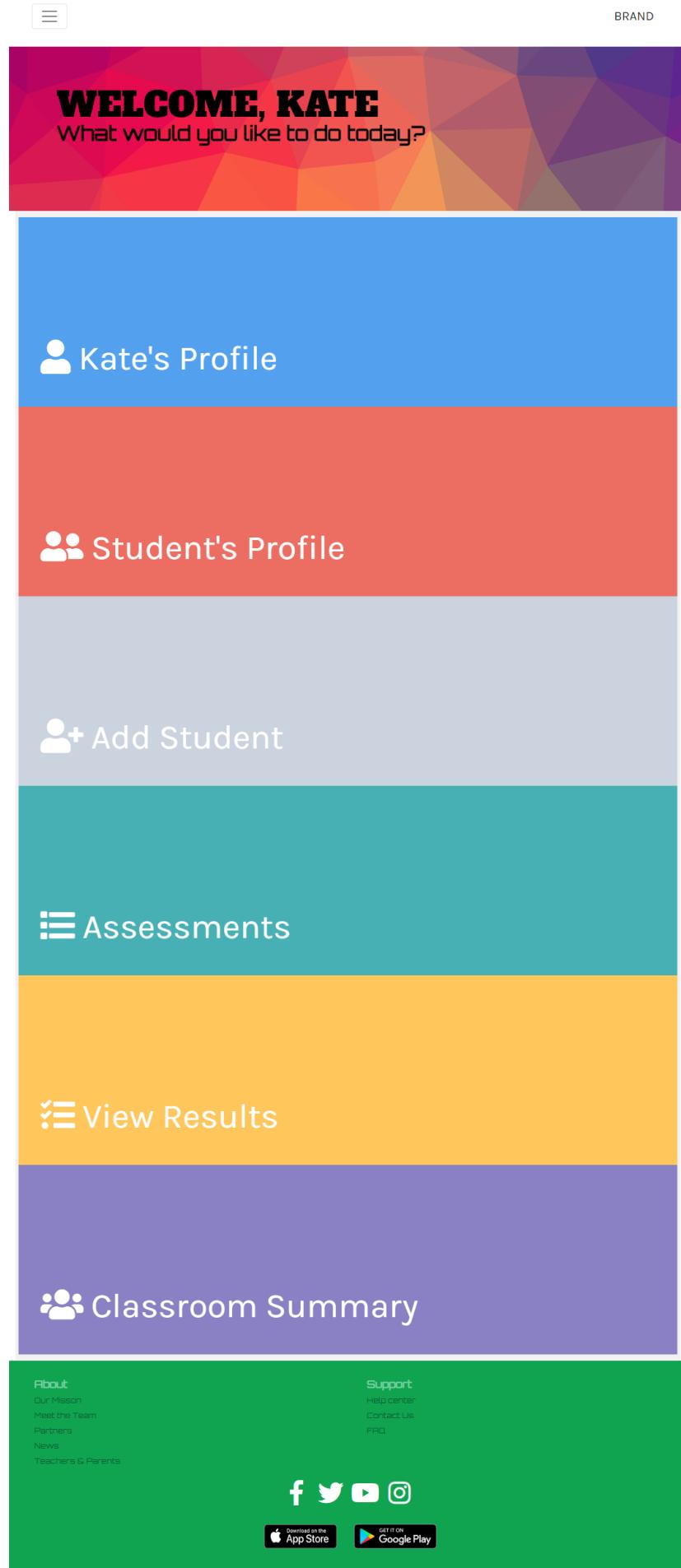
**[Figure 9]**

The screenshot displays a vibrant, multi-colored geometric background in shades of red, orange, yellow, green, blue, and purple. At the top, a navigation bar features links for Dashboard (highlighted in yellow), Store, Contact, About, Help, and Sign Out. Below the header, a large, bold "WELCOME, KATE" message is centered, followed by the question "What would you like to do today?". The main content area is organized into six colored boxes arranged in a 2x3 grid:

- Kate's Profile** (Blue box)
- Student's Profile** (Red box)
- Add Student** (Grey box)
- Assessments** (Teal box)
- View Results** (Yellow box)
- Classroom Summary** (Purple box)

At the bottom, a green footer bar contains links for About (Our Mission, Meet the Team, Partners, News, Teachers & Parents), Support (Help center, Contact Us, FAQ), social media icons for Facebook, Twitter, YouTube, and Instagram, and download links for the App Store and Google Play.

[Figure 10]



[Figure 11]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background composed of triangles in shades of red, orange, yellow, green, and purple. In the center of this background, the text "WELCOME, KATE" is displayed in a large, bold, black font. Below it, the question "What would you like to do today?" is shown in a smaller, regular black font.

Below the background, there is a form titled "Create a new student account". The form consists of six input fields:

- First Name
- Last Name
- Username
- Password
- Re-type Password
- Add (a blue button)

At the bottom of the page is a green footer bar containing links for "About", "Support", and "Teachers & Parents", along with social media icons for Facebook, Twitter, YouTube, and Instagram. It also includes download links for the App Store and Google Play.

[Figure 12]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background with shades of red, orange, yellow, green, and purple.

In the center, the text "WELCOME, KATE" is displayed in a large, bold, black font. Below it, the question "What would you like to do today?" is shown in a smaller, regular black font.

The main content area contains a form for creating a new student account. The form fields are as follows:

- First Name: Fahim
- Last Name: (Field highlighted with a red border)
- Username: (Field highlighted with a red border)
- Password: (Field highlighted with a red border)
- Re-type Password: (Field highlighted with a red border)

Below the form, error messages are displayed in red text:

- "Something Went Wrong!" (above the Last Name field)
- "This field is required." (above the Last Name field)
- "Please Match the requested format.  
This field only accepts letters, numbers and underscores." (above the Username field)
- "This field is required." (above the Password field)
- "This field is required." (above the Re-type Password field)

At the bottom of the form is a blue "Add" button.

At the very bottom of the page is a green footer bar containing links for About, Support, and social media icons (Facebook, Twitter, YouTube, Instagram). It also includes download links for the App Store and Google Play.

[Figure 13]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background with a red-to-purple gradient. In the center, the text "WELCOME, KATE" is displayed in a large, bold, black font, followed by the question "What would you like to do today?". Below this, there is a form for creating a new student account. The form fields are as follows:

- Name: Fahim
- Surname: Hossain
- Username: fhoss001
- Password: (Field highlighted with a red border)
- Re-type Password: (Field highlighted with a red border)

Below the password fields, an error message says "Passwords didn't match, please try again". At the bottom of the form is a blue "Add" button.

At the bottom of the page, there is a green footer bar containing links for About, Support, and social media icons (Facebook, Twitter, YouTube, Instagram). It also includes download links for the App Store and Google Play.

[Figure 14]

The screenshot displays a web application interface. At the top, there is a navigation bar with the following items: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background pattern composed of triangles in shades of red, orange, yellow, green, and purple.

In the center of the page, the text "WELCOME, KATE" is displayed in large, bold, black capital letters. Below it, the question "What would you like to do today?" is shown in a smaller, regular black font.

Below the welcome message is a form titled "Edit your account details". The form contains five input fields:

- A text input field containing "Kate".
- A text input field containing "Gardner".
- A text input field containing "kgard001".
- A text input field labeled "Current Password".
- A text input field labeled "New Password".

At the bottom of the form is a blue rectangular button with the word "Edit" in white text.

At the very bottom of the page is a green footer bar. On the left side of the footer, under the heading "About", are links to "Our Mission", "Meet the Team", "Partners", "News", and "Teachers & Parents". On the right side, under the heading "Support", are links to "Help center", "Contact Us", and "FAQ". To the right of these links are icons for social media platforms: Facebook, Twitter, YouTube, and Instagram. Below these icons are download links for the App Store and Google Play.

[Figure 15]

The screenshot shows a mobile application interface. At the top, there is a navigation bar with five items: 'BRAND' (selected), 'Dashboard', 'Store', 'Contact', 'About', 'Help', and 'Sign Out'. Below the navigation bar is a large, colorful polygonal background image. Overlaid on this background is a welcome message: 'WELCOME, KATE' in large bold black letters, followed by 'What would you like to do today?' in smaller black letters.

The main content area contains a form titled 'Edit your account details'. Inside the form, there is an error message 'Something Went Wrong!'. The form fields are as follows:

- Name: 'Kate' (red border, error message: 'This field is required.')
- Surname: 'Gardner' (red border, error message: 'This field is required.')
- Username: 'kgard001' (red border, error message: 'This field is required.')
- Current Password: (red border, error message: 'This field is required.')
- New Password: (red border)
- Edit** button (blue background)

At the bottom of the screen is a green footer bar containing links for 'About', 'Support', and social media icons for Facebook, Twitter, YouTube, and Instagram. It also includes download links for the App Store and Google Play.

[Figure 16]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background with a red-to-purple gradient. In the center, the text "WELCOME, KATE" is displayed in large, bold, black capital letters. Below it, the question "What would you like to do today?" is shown in a smaller, regular black font.

The main content area contains a form titled "Edit your account details". Inside the form, there is an error message "Something Went Wrong!" above four input fields, each with a red border:

- Kate
- Gardner
- kgard001
- Current Password

Below the password field, an error message "Incorrect Password." is displayed. There is another input field for "New Password" and a blue "Edit" button at the bottom of the form.

At the bottom of the page, there is a green footer bar containing links for "About", "Support", and social media icons for Facebook, Twitter, YouTube, and Instagram. It also includes download links for the App Store and Google Play.

[Figure 17]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background with a central welcome message.

**WELCOME, KATE**  
What would you like to do today?

Below the welcome message is a table displaying a list of users:

Aida	Duarte	aduar001	New Password	Edit
Alisa	Juarez	ajuar001	New Password	Edit
Araceli	Pena	apena001	New Password	Edit
Booker	Phillips	bphil001	New Password	Edit
Brittney	Baxter	bbaxt001	New Password	Edit

Below the table are pagination controls (1, 2, 3, 4, 5, 6), sorting options (Order by First Name ASC), and a row limit selector (Number of rows 5). The bottom of the page features a green footer with links for About, Support, and social media icons (Facebook, Twitter, YouTube, Instagram). It also includes download links for the App Store and Google Play.

[Figure 18]

The screenshot displays a mobile application interface with a vertical list of user profiles. Each profile card contains the following information:

- Aida
- Duarte
- aduar001
- New Password
- Edit

Below this is another set of profile cards:

- Alisa
- Juarez
- ajuar001
- New Password
- Edit

Then another set:

- Araceli
- Pena
- apena001
- New Password
- Edit

Finally, another set:

- Booker
- Phillips
- bphil001
- New Password
- Edit

At the bottom of the list, there are navigation controls:

- Order by: First Name ▾
- Sort by: ASC ▾
- Number of rows: 5 ▾

The footer of the application is a solid green bar containing the following links and icons:

- About
- Our Mission
- Meet the Team
- Partners
- News
- Teachers & Parents
- Support
- Help center
- Contact Us
- FAQ
- Social media icons: Facebook, Twitter, YouTube, Instagram
- Download links: App Store and Google Play

[Figure 19]

The screenshot shows a mobile application interface. At the top, there is a colorful, geometric background banner with the text "WELCOME, KATE" in large, bold, black letters. Below the banner, a question "What would you like to do today?" is displayed in a smaller, white font.

The main content area displays a list of users, each with their first name, last name, and a placeholder for a new password. Each user entry is followed by a blue "Edit" button.

- Aida
- Duarte
- aduar001
- New Password
- Edit
- Alisa
- Juarez
- ajuar001
- New Password
- Edit
- Araceli
- Pena
- apena001
- New Password
- Edit
- Booker
- Phillips
- bphil001
- New Password
- Edit
- Brittney
- Baxter
- bbaxt001
- New Password
- Edit

Below the list, there are navigation controls: a page number indicator (1, 2, 3, 4, 5, 6) with the current page highlighted in blue, a dropdown menu for sorting ("Order by First Name ▾"), a sort order dropdown ("Sort by ASC ▾"), and a row count dropdown ("Number of rows 5 ▾").

The bottom section of the screen has a dark green background and contains links for "About" (Our Mission, Meet the Team, Partners, News, Teachers & Parents) and "Support" (Help center, Contact Us, FAQ). It also features social media icons for Facebook, Twitter, YouTube, and Instagram, and download links for the App Store and Google Play.

[Figure 20]

The screenshot shows a web application interface. At the top, there is a navigation bar with links: Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a large, colorful geometric background graphic. Overlaid on this is a welcome message: "WELCOME, KATE" in large bold letters, followed by "What would you like to do today?" in a smaller font.

The main content area displays a table of user data. The columns are labeled: First Name, Last Name, Username, and Action (labeled "New Password"). Each row contains a set of data and an "Edit" button. The users listed are:

First Name	Last Name	Username	Action
Wilfredo	Dunn	wdunn001	New Password
Tuan	Blackburn	tblac001	New Password
Seth	Shannon	sshann001	New Password
Simone	Glenn	sglenn001	New Password
Rodrigo	Holder	rhold001	New Password
Nicole	Mcknight	nmckn001	New Password
Mae	Wiggins	mwigg001	New Password
Micheal	Miles	mmile001	New Password
Lowell	Webster	lwebs001	New Password
Leonor	Oliver	loliv001	New Password
Leigh	Nunez	lnune001	New Password
Logan	Mason	lmaso001	New Password
Lindsay	Marshall	lmars001	New Password
Lou	Fletcher	lflet001	New Password
Kerry	Strong	kstro001	New Password
Kraig	Orozco	koroz001	New Password
Kim	Lutz	klutz001	New Password
Judith	Trujillo	jtruj001	New Password
Jenna	Bass	jbass001	New Password
Erna	Khan	ekhan001	New Password

Below the table, there are pagination controls showing page 1 of 2, and sorting/filtering options: Order by Username (sorted DESC), Number of rows (set to 20).

The bottom of the page has a green footer bar with links for About, Support, and various social media icons (Facebook, Twitter, YouTube, Instagram). It also includes download links for the App Store and Google Play.

[Figure 21]

The screenshot displays a web application interface with the following components:

- Header:** A navigation bar with links for Dashboard, Store, Contact, About, Help, and Sign Out.
- Welcome Message:** A large, bold "WELCOME, KATE" followed by the question "What would you like to do today?"
- User List:** A table showing five users with columns for First Name, Last Name, User ID, and Role. Each row includes an "Edit" button.

Aida	Duarte	aduar001	None	Edit
Alisa	Juarez	ajuar001	None	Edit
Araceli	Pena	apena001	None	Edit
Booker	Phillips	bphili001	None	Edit
Brittney	Baxter	bbaxt001	None	Edit

- Pagination and Filters:** A page number indicator at 1, a "Order by" dropdown set to "First Name", a "Sort by" dropdown set to "ASC", and a "Number of rows" dropdown set to 5.
- Footer:** A green footer section with "About" and "Support" links, social media icons for Facebook, Twitter, YouTube, and Instagram, and download links for the App Store and Google Play.

[Figure 22]

The screenshot shows a web application interface with the following components:

- Header:** A navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out.
- Welcome Message:** "WELCOME, KATE" followed by "What would you like to do today?"
- User List:** A table displaying five users with columns: First Name, Last Name, Username, and Group. Each row includes an "Edit" button.

Seth	Shannon	sshan001	None	Edit
Simone	Glenn	sglenn001	None	Edit
Tuan	Blackburn	tblac001	Mr Paddington's Quiz-a-thon	Edit
Wilfredo	Dunn	wdunn001	Mr Paddington's Quiz-a-thon	Edit
- Pagination:** A page number indicator showing 1, 2, 3, 4, 5, and 6, with 6 highlighted in blue.
- Sorting Options:** Order by First Name (ASC), Sort by ASC, and Number of rows (5).
- Footer:** A green footer with sections for About (Our Mission, Meet the Team, Partners, News, Teachers & Parents) and Support (Help center, Contact Us, FAQ). It also features social media icons for Facebook, Twitter, YouTube, and Instagram, and download links for the App Store and Google Play.

[Figure 23]

The screenshot shows a web-based application interface. At the top, there is a navigation bar with links: BRAND, Dashboard, Store, Contact, About, Help, and Sign Out. The main header features a large, stylized, multi-colored geometric background with the text "WELCOME, KATE" in bold black letters, followed by "What would you like to do today?". Below this, there are two tabs: "Students" and "Results". The "Students" tab is active, displaying a table with five rows of student data:

Seth	Shannon	sshan001	No Assignments	<a href="#">View</a>
Simone	Glenn	sglenn001	No Assignments	<a href="#">View</a>
Tuan	Blackburn	tblac001	Mr Paddington	<a href="#">View</a>
Wilfredo	Dunn	wdunn001	Mr Paddington	<a href="#">View</a>

Below the table are pagination controls showing numbers 1 through 6, with 6 being underlined. There are also dropdown menus for sorting and filtering: "Order by First Name", "Sort by ASC", and "Number of rows 5".

The bottom section of the page has a green footer bar containing links for "About", "Our Mission", "Meet the Team", "Partners", "News", and "Teachers & Parents". It also contains "Support" links for "Help center", "Contact Us", and "FAQ". On the right side of the footer, there are social media icons for Facebook, Twitter, YouTube, and Instagram. Additionally, there are download links for the "App Store" and "Google Play".

[Figure 24]

The screenshot shows a digital learning platform interface. At the top, there is a navigation bar with links for Dashboard, Store, Contact, About, Help, and Sign Out. Below the navigation bar is a colorful, geometric background with a large, bold "WELCOME, KATE" message and a sub-question "What would you like to do today?".

Below the welcome message, there are two tabs: "Students" and "Results". The "Results" tab is selected, displaying a summary for a user named "wdunn001".

**wdunn001**

13	02:56	2/10
attempt(s)	to complete	correct

The results section displays three questions:

**1) Remember to \_\_\_\_\_ off the light**

<input type="checkbox"/> Switch	<input type="checkbox"/> Sweetch
<input type="checkbox"/> Swich	<input type="checkbox"/> Switche

Question Type: Spelling  
Correct Answer: Switch

First Choice: Switche  
Second Choice: Sweetch  
Third Choice: Swich  
Time Taken: 01:02

**2) Why did Micheal drop his tray?**

<input type="checkbox"/> The floor was wet	<input type="checkbox"/> The food was too hot
<input type="checkbox"/> He didn't like the food	<input type="checkbox"/> He tripped on a sign

Question Type: Understanding  
Correct Answer: The floor was wet

First Choice: The floor was wet  
Time Taken: 00:49

**3) Why was Jamila tired when she got to school?**

<input type="checkbox"/> She missed her bus	<input type="checkbox"/> She decided to run to school
<input type="checkbox"/> and had to walk	
<input type="checkbox"/> Her school was very far away	<input type="checkbox"/> School is so boring
	far away

Question Type: Understanding  
Correct Answer: She missed her bus and had to walk

First Choice: Her school was very far away  
Second Choice: Her school was very far away  
Third Choice: Her school was very far away  
Time Taken: 00:01

4) I love my older \_\_\_\_\_

- |  |                               |
|--|-------------------------------|
| <input type="radio"/> Brovor             | <input type="radio"/> Brother |
| <input checked="" type="radio"/> Brudduh | <input type="radio"/> Bruvver |

Question Type: Spelling

Correct Answer: Brother

First Choice: Brudduh

Second Choice: Brudduh

Third Choice: Brudduh

Time Taken: 00:01

5) A \_ \_ L E

- |                                    |                         |
|------------------------------------|-------------------------|
| <input type="radio"/> P            | <input type="radio"/> N |
| <input checked="" type="radio"/> B | <input type="radio"/> Y |

Question Type: Spelling

Correct Answer: P

First Choice: B

Second Choice: B

Third Choice: B

Time Taken: 00:01

6) In maths, we learnt what a \_\_\_\_\_ is

- |   |                                   |
|---|-----------------------------------|
| <input type="radio"/> Fraction            | <input type="radio"/> Frahction   |
| <input checked="" type="radio"/> Fraktion | <input type="radio"/> FuhrackSION |

Question Type: Spelling

Correct Answer: Fraction

First Choice: Fraktion

Second Choice: Fraktion

Third Choice: Fraktion

Time Taken: 00:01

7) Which sentence is a question?

- |  |  |
|--|--|
| <input type="radio"/> This puzzle is hard            | <input type="radio"/> Where is my puzzle |
| <input checked="" type="radio"/> That puzzle is mine | <input type="radio"/> I hate puzzles     |

Question Type: Punctuation

Correct Answer: Where is my puzzle

First Choice: That puzzle is mine

Second Choice: That puzzle is mine

Third Choice: That puzzle is mine

Time Taken: 00:01

8) My friend, Mayesha, is a cool girl

- |  |                              |
|--|------------------------------|
| <input type="radio"/> cool               | <input type="radio"/> girl   |
| <input checked="" type="radio"/> Mayesha | <input type="radio"/> friend |

Question Type: Grammar

Correct Answer: cool

First Choice: Mayesha

Second Choice: Mayesha

Third Choice: Mayesha

Time Taken: 00:01

9) We will make it on time \_\_\_\_\_ we leave right now

- |  |                            |
|--|----------------------------|
| <input type="radio"/> so               | <input type="radio"/> when |
| <input checked="" type="radio"/> where | <input type="radio"/> if   |

Question Type: Grammar

Correct Answer: if

First Choice: where

Second Choice: where

Third Choice: where

Time Taken: 00:01

10) Jason's clothes were \_\_\_\_\_ after football practice

- |  |                               |
|--|-------------------------------|
| <input type="radio"/> Dirtyly          | <input type="radio"/> Dиритед |
| <input checked="" type="radio"/> Dirty | <input type="radio"/> Dirt    |

Question Type: Grammar

Correct Answer: Dirty

First Choice: Dirty

Time Taken: 00:01

Correct Answer: Dirty

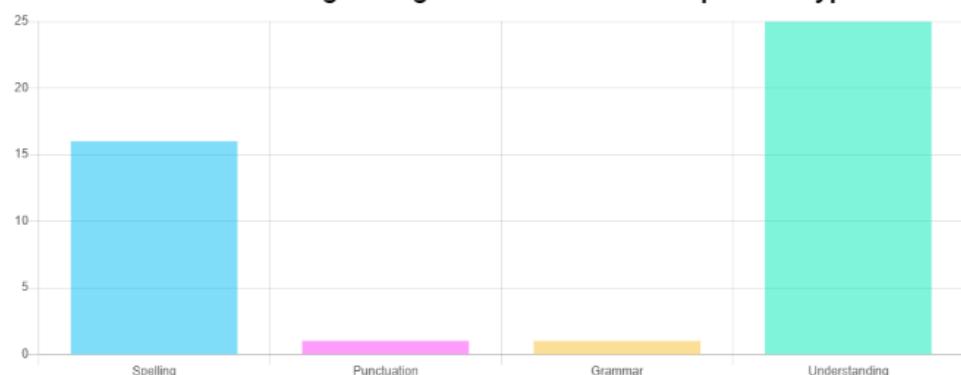
Percentage difference of the question type that were answered correctly



Percentage difference of the question type that were answered incorrectly



Bar chart showing average time taken for each question type



#### About

- [Our Mission](#)
- [Meet the Team](#)
- [Partners](#)
- [News](#)
- [Teachers & Parents](#)

#### Support

- [Help center](#)
- [Contact Us](#)
- [FAQ](#)



[Figure 25]

The screenshot shows a user interface for a school management system. At the top, there is a navigation bar with several buttons: Dashboard, Store, Contact, About, Help, Sign Out, and a BRAND logo. The main content area features a large, colorful, geometric background image. Overlaid on this image is a large, bold text "WELCOME, KATE" and a smaller question "What would you like to do today?". To the right of the main content, there are two tabs: "Students" and "Results". The "Results" tab is currently selected, displaying the message "No student selected, please select a student from the 'student' tab". On the far right, there is a green sidebar containing social media icons for Facebook, Twitter, YouTube, and Instagram, along with links to the app's help center, contact us page, and F-ROD information. There are also download links for the App Store and Google Play, and a Teachers & Parents section.

BRAND

Dashboard Store Contact About Help Sign Out

WELCOME, KATE  
What would you like to do today?

Students Results

No student selected, please select a student from the 'student' tab

Support

Help center Contact Us F-ROD

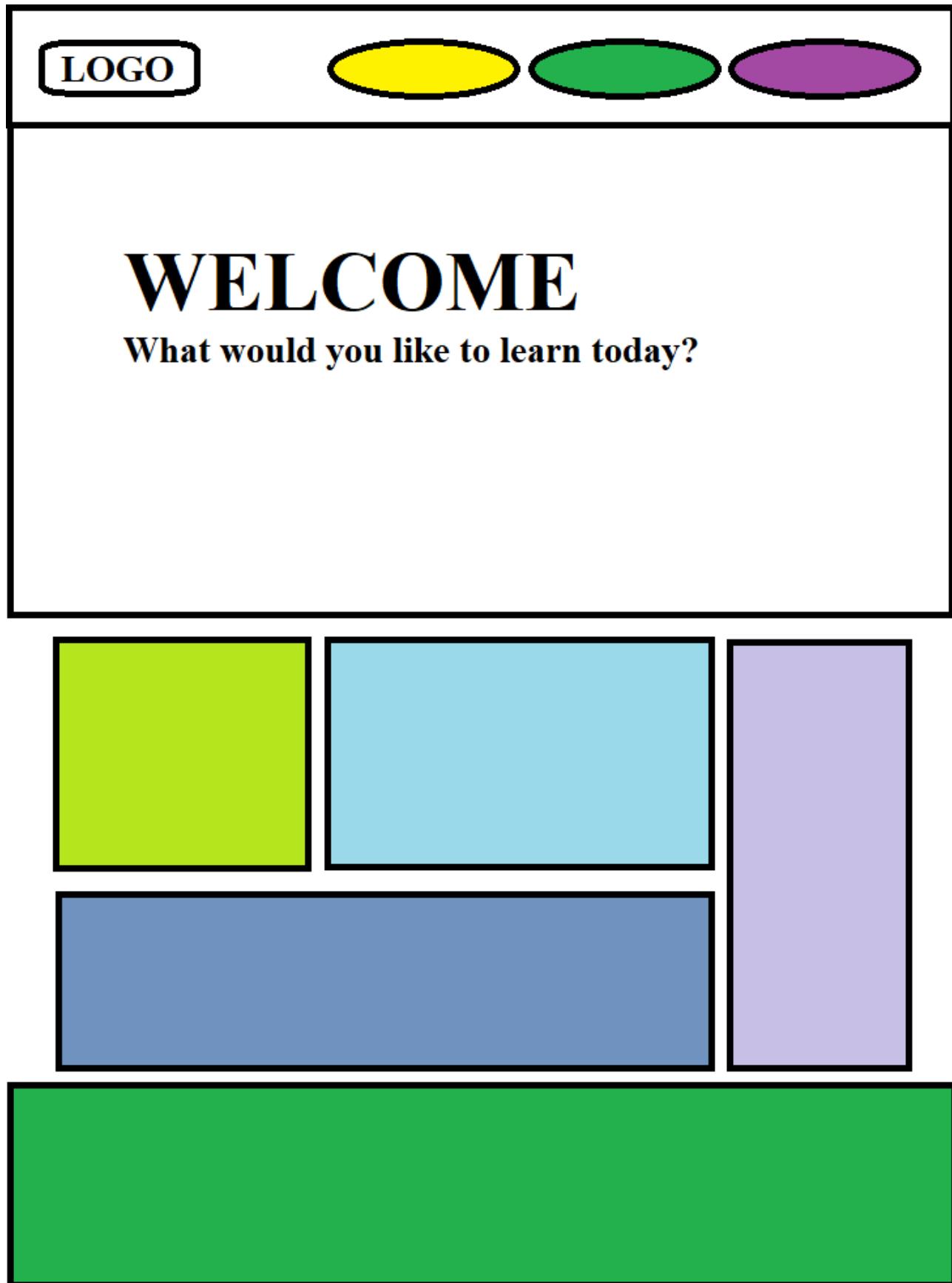
GET IT ON

App Store Google Play

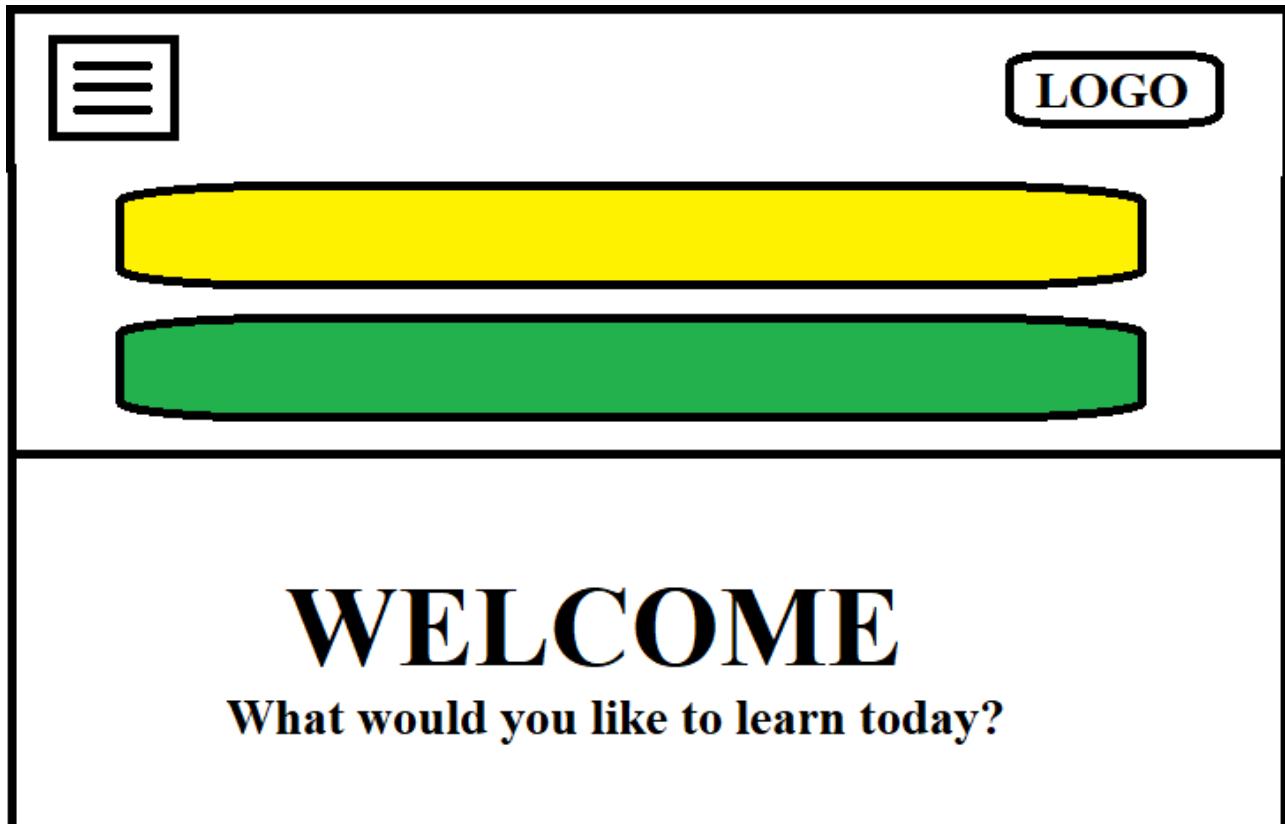
About

Our Mission Meet the Team Partners News Teachers & Parents

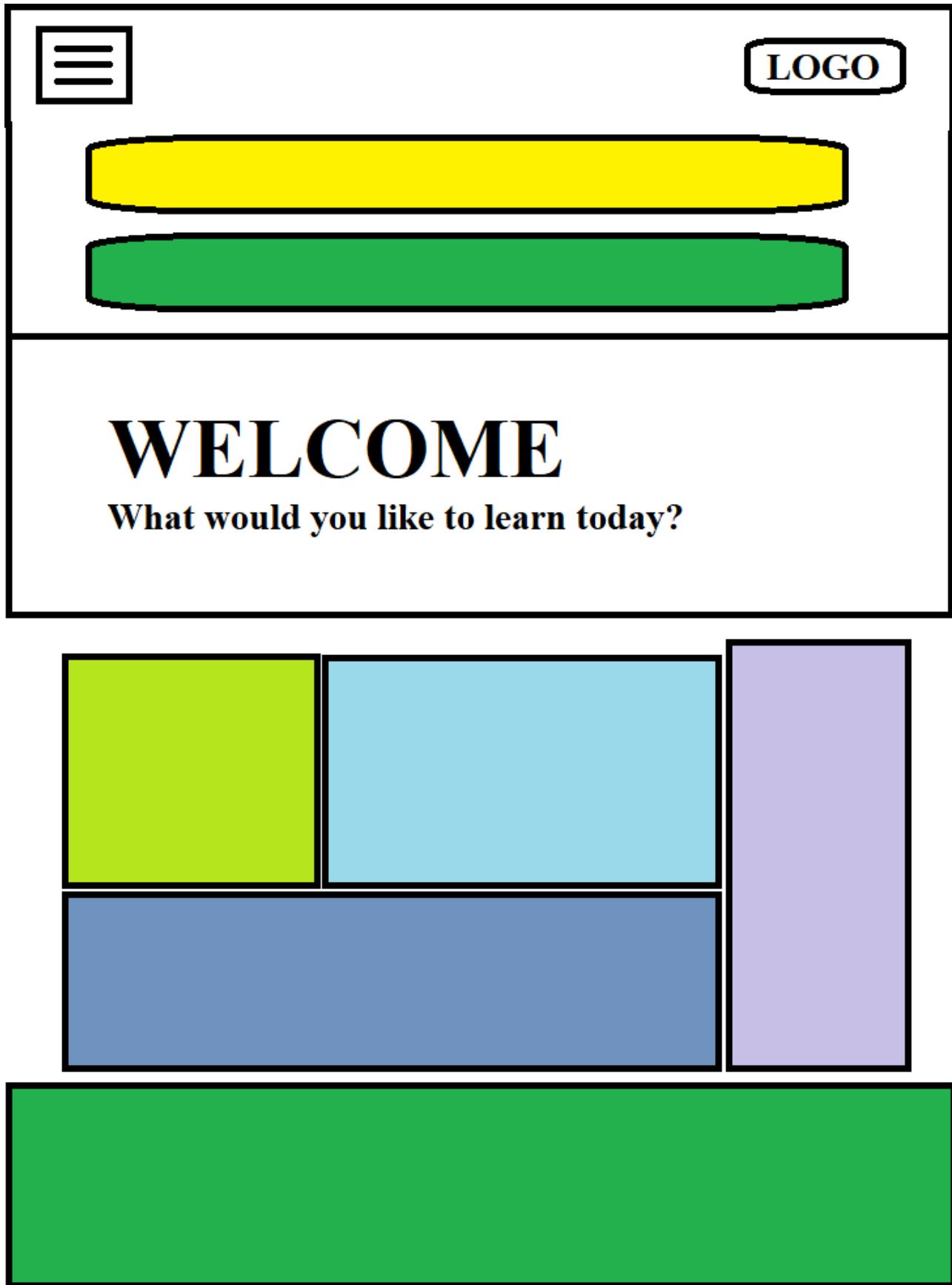
[Figure 26]



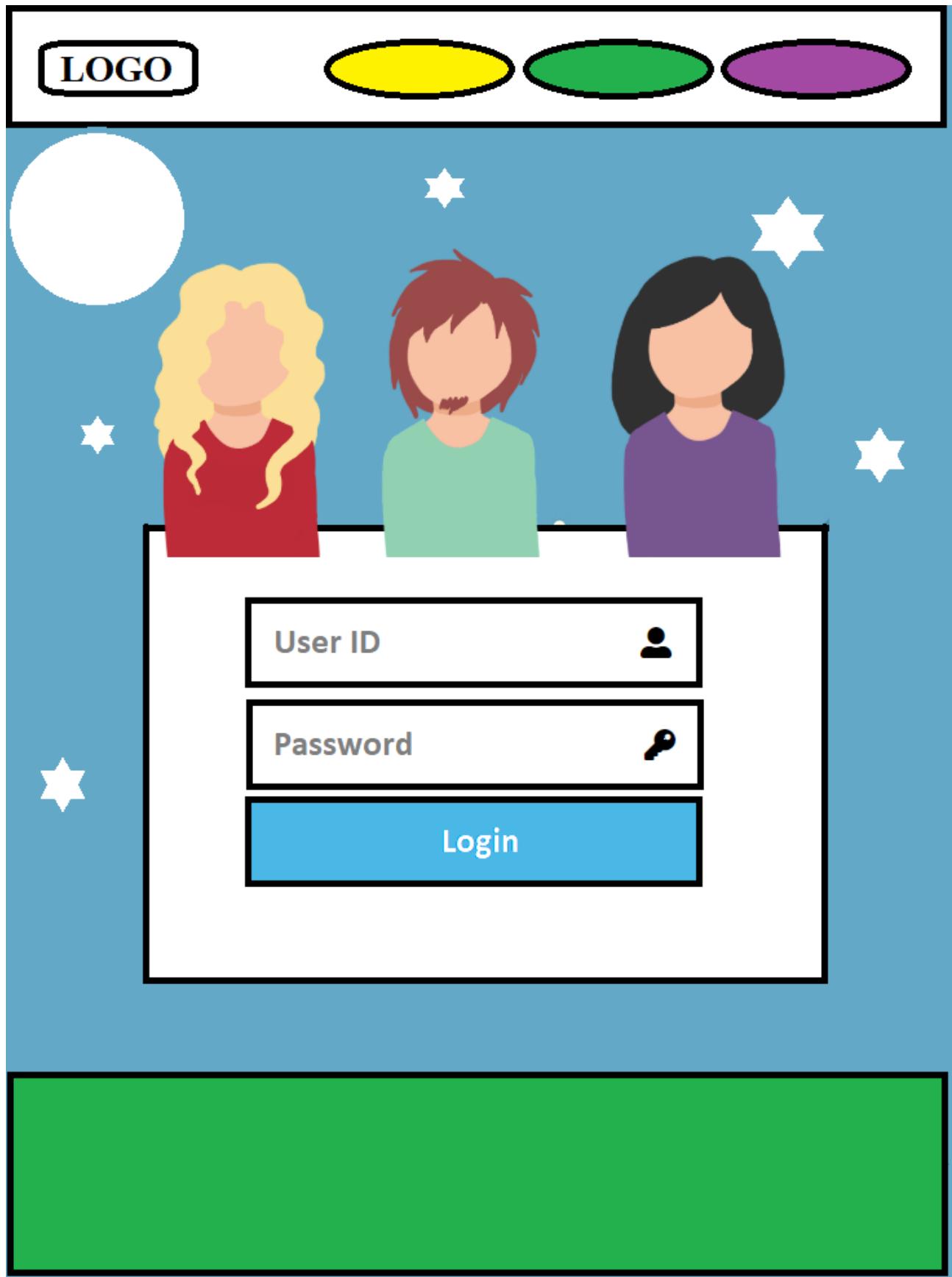
[Figure 27]



[Figure 28]



[Figure 29]



[Figure 30]

Welcome

Log in/Sign out

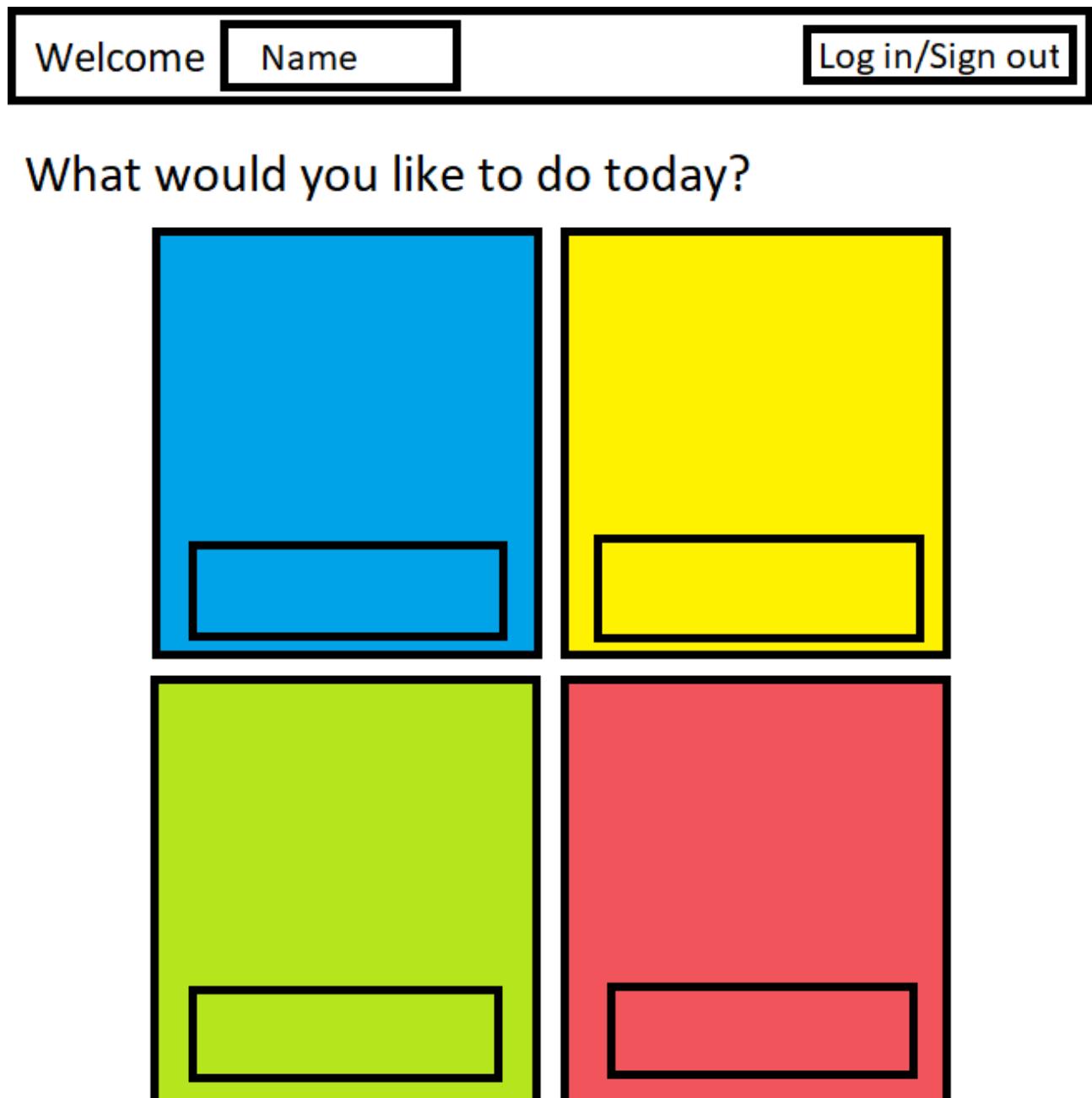
← Log-in

Username

Password

Log-in

[Figure 31]

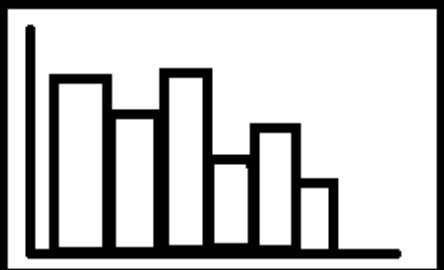


[Figure 32]

Welcome    Name    Log in/Sign out

Showing Results For [[[Name]]]    

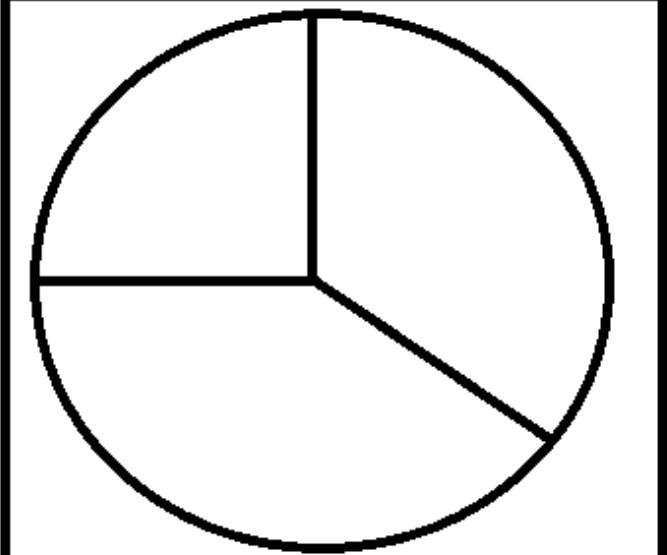




~~~~~

~~~~~





~~~~~

~~~~~

~~~~~

~~~~~

[Figure 33]

The image shows a mobile application interface. At the top, there is a navigation bar with five items: "Home" (selected), "Store", "Contact", "About", and "Help". To the left of the "Home" item is a "BRAND" placeholder. On the right is a "Sign In" button. Below the navigation bar is a large, central modal window titled "Log In to Your Account". The modal contains three input fields: "User ID" (with a person icon), "Password" (with a key icon), and a blue "Login" button. Above the modal, there are three small, circular profile pictures of people. The footer section is green and contains links for "About", "Our Mission", "Meet the Team", "Partners", "News", and "Teachers & Parents" under the "About" heading. It also contains links for "Support", "Help center", "Contact Us", and "FAQ". On the right side of the footer are social media icons for Facebook, Twitter, YouTube, and Instagram. Below these are download links for the "App Store" and "Google Play".

BRAND

Home Store Contact About Help Sign In

Log In to Your Account

User ID

Password

Login

About

Our Mission

Meet the Team

Partners

News

Teachers & Parents

Support

Help center

Contact Us

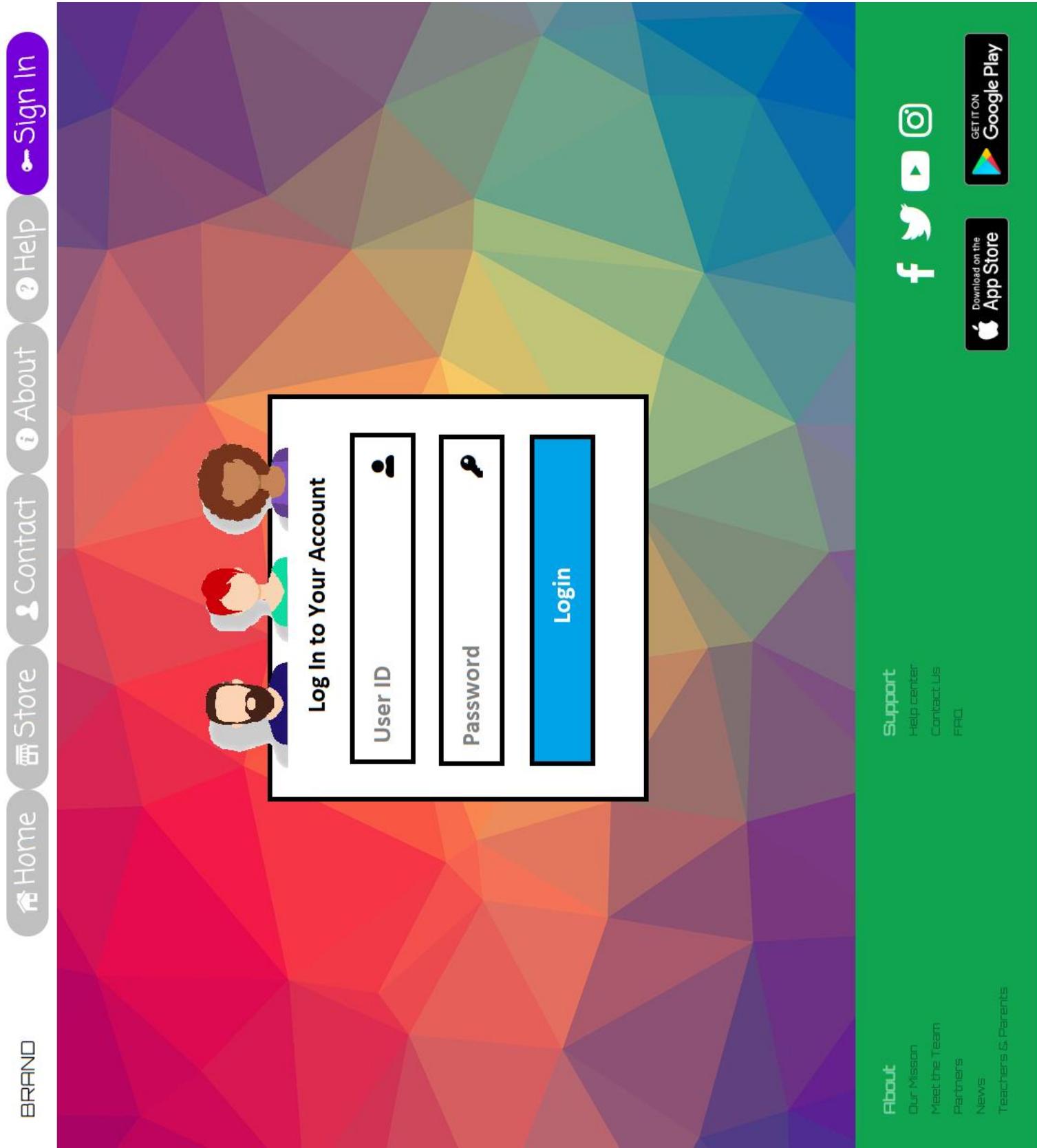
FAQ

f t y i

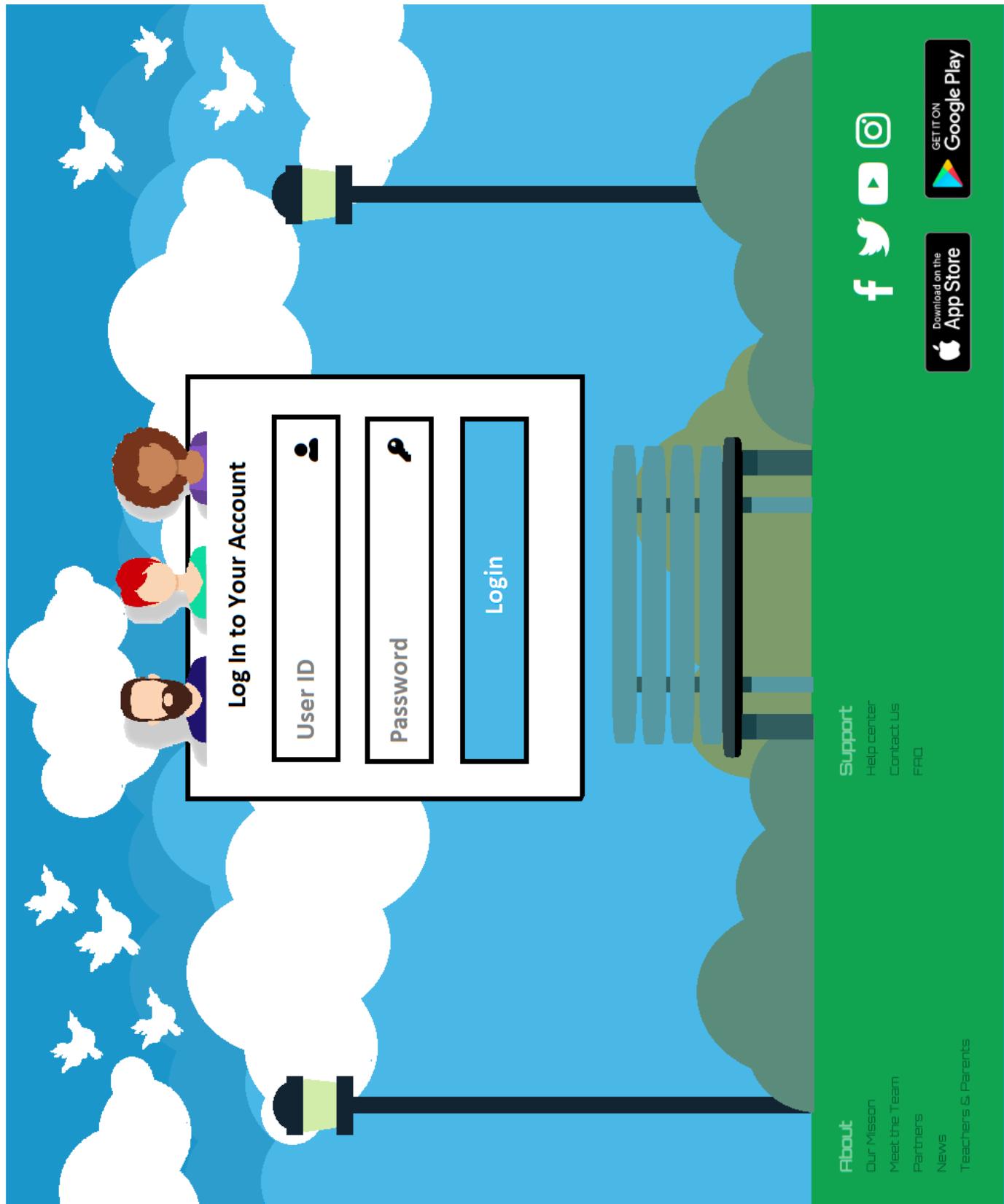
Download on the App Store

GET IT ON Google Play

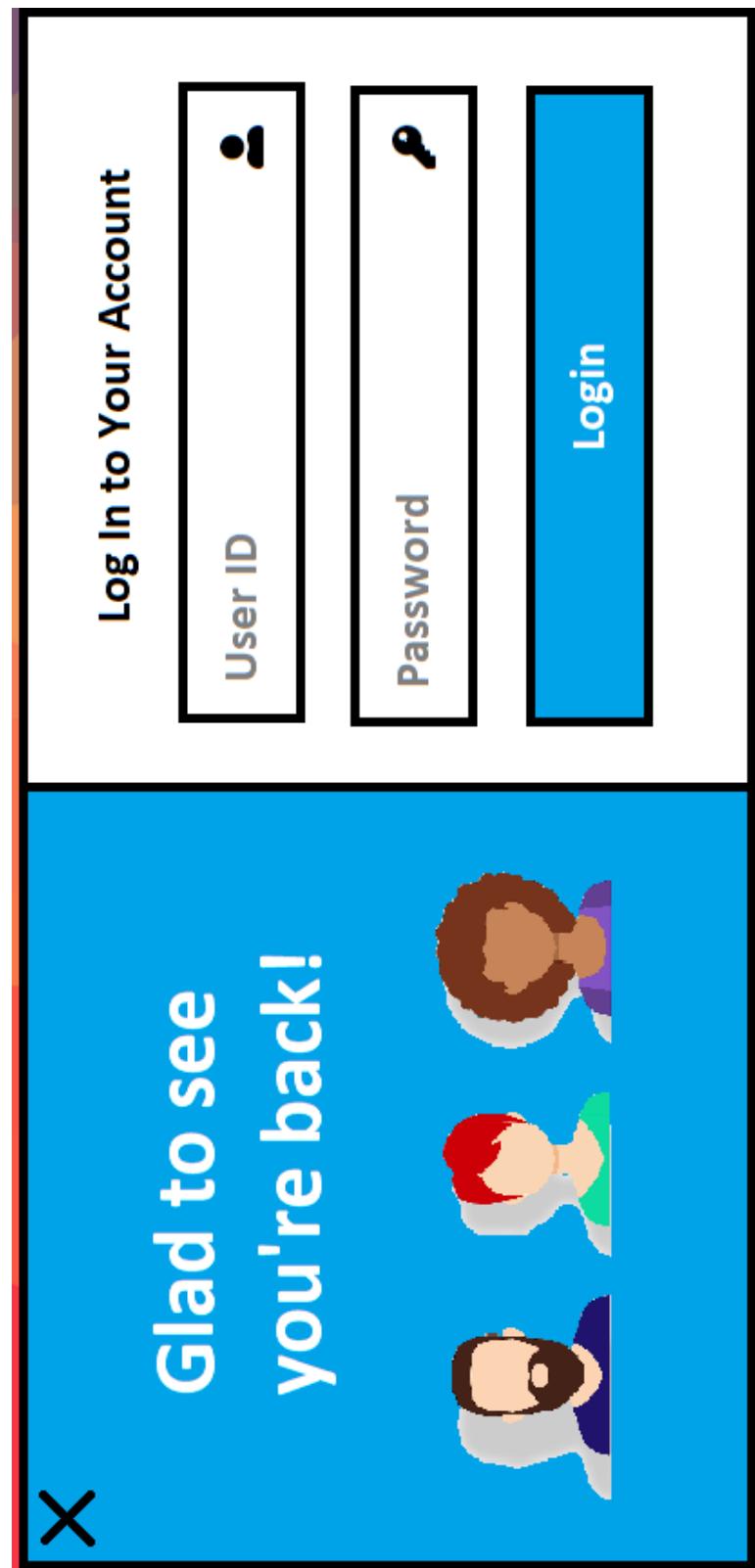
[Figure 34]



[Figure 35]



[Figure 36]



[Figure 37]

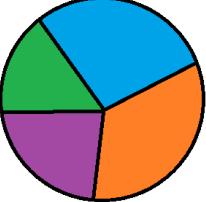
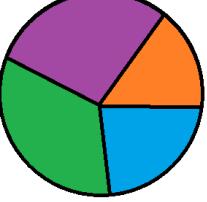
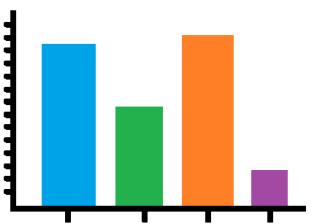
The screenshot shows a website layout. At the top, there is a black navigation bar with links: Home, Store, Contact, About, Help, and Sign In. Below the navigation bar is a decorative header featuring a row of colorful pencils of various colors (red, pink, purple, blue, green, yellow, orange, brown) against a light background. The word "WELCOME" is written in large, bold, black capital letters above the text "What would you like to do today?". To the left of the main content area is a vertical red sidebar containing icons and labels for different subjects: MATHS (with a calculator icon), ENGLISH (with a keyboard icon), SCIENCE (with a molecular structure icon), ART & DESIGN (with a paint palette icon), and HISTORY (with a circular icon). The main content area is a grid of ten boxes labeled Activity A through Activity K. The grid is organized as follows: Row 1: Activity A (wide), Activity B (wide). Row 2: Activity C (wide), Activity D (wide). Row 3: Activity E (wide), Activity F (wide), Activity G (wide). Row 4: Activity I (wide), Activity J (wide), Activity K (wide).

<-- Information Here -->

<-- Footer Links Here -->

<-- Social Links Here -->

[Figure 38]

Student Name		
5 attempts	15:20 to complete	6/10 Correct
<p>1)</p> <p>What letter is missing A _ _ L E?</p> <p><input type="radio"/> A   <input type="radio"/> P <input type="radio"/> B   <input type="radio"/> Q</p> <p>Question Type: Spelling Correct Answer: P</p>	<p>First Choice: A Second Choice: B Third Choice: P Time Taken: 00:30</p>	
<p>2)</p> <p>What letter is missing A _ _ L E?</p> <p><input type="radio"/> A   <input type="radio"/> P <input type="radio"/> B   <input type="radio"/> Q</p> <p>Question Type: Spelling Correct Answer: P</p>	<p>First Choice: A Second Choice: B Third Choice: B Time Taken: 01:15</p>	
<p>Pie Chart showing type of questions student got right</p> 	<p>Pie Chart showing type of questions student got wrong</p> 	
<p>Bar chart showing average time taken for each question type</p> 		

## Appendix E: Activity Design Concepts

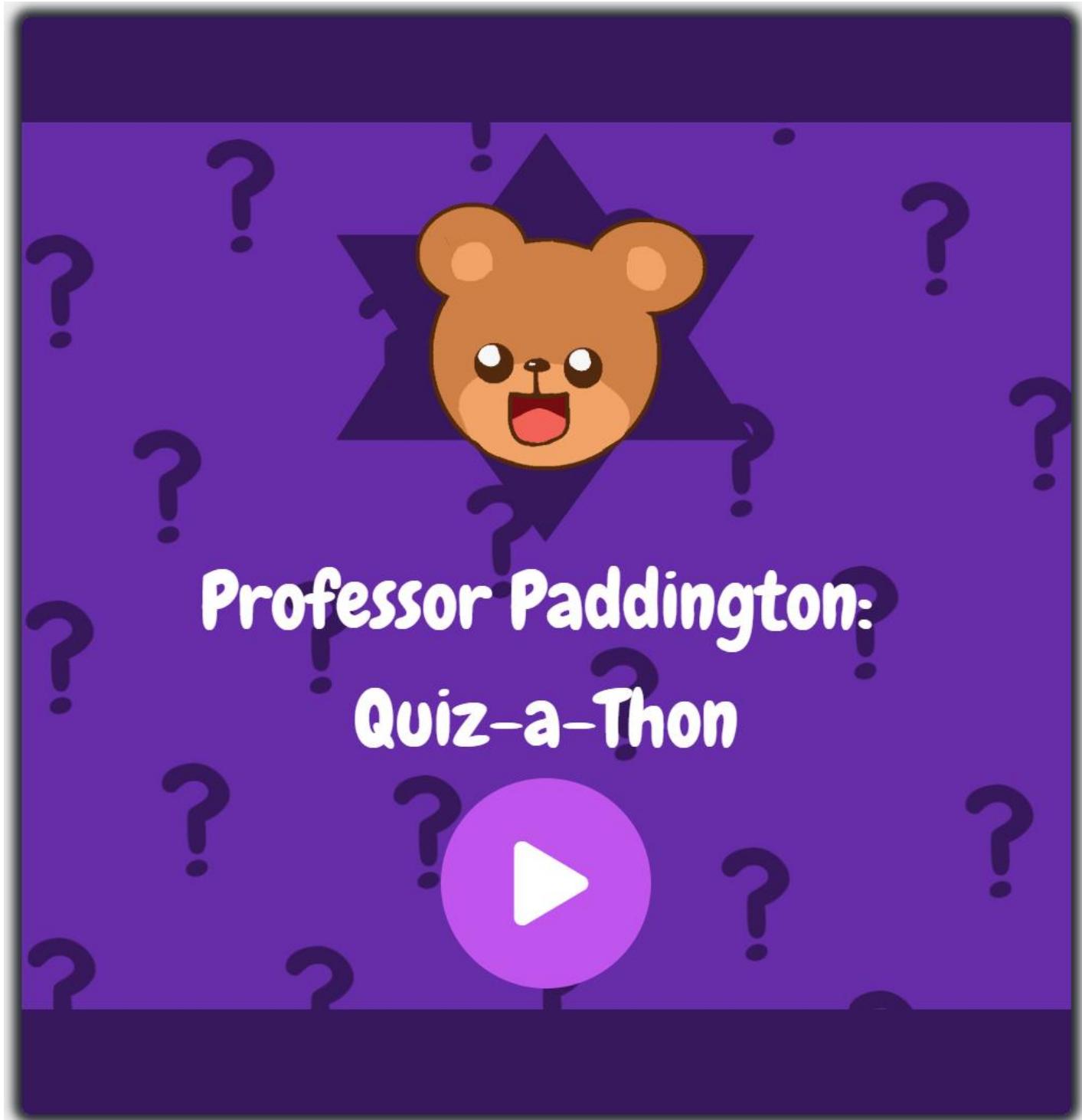
[Figure 1]

The image shows a mobile application screen. At the top, there is a navigation bar with five items: "Home", "Contact", "About", "Help", and "Sign Out". Below the navigation bar is a large, colorful polygonal background in shades of red, orange, yellow, green, and blue. Overlaid on this background is the text "WELCOME, WILFREDO" in bold black letters, followed by "Let's see how well you do on, Mr Paddington's Quiz-a-thon". The main content area features a purple background with a cartoon bear wearing a graduation cap in the center. The bear is surrounded by question marks. Below the bear, the text "Professor Paddington: Quiz-a-Thon" is displayed in white. A large purple play button with a white play icon is centered at the bottom. At the very bottom of the screen is a green footer bar containing links for "About", "Support", "Social media icons (Facebook, Twitter, YouTube, Instagram)", and download links for "App Store" and "Google Play".

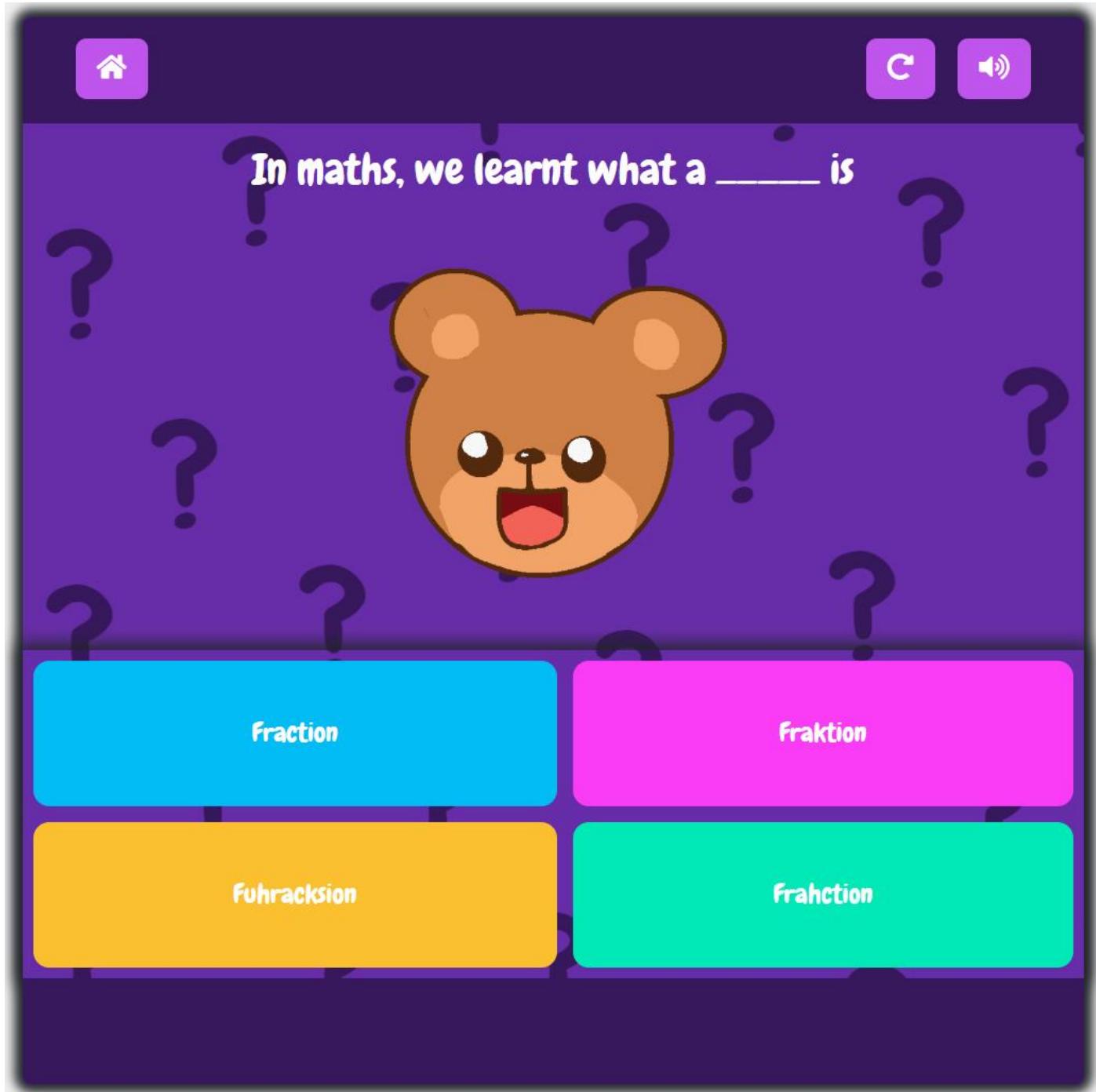
[Figure 2]



[Figure 3]



[Figure 4]



[Figure 5]

In maths, we learnt what a \_\_\_\_\_ is

Fraction

Fraktion

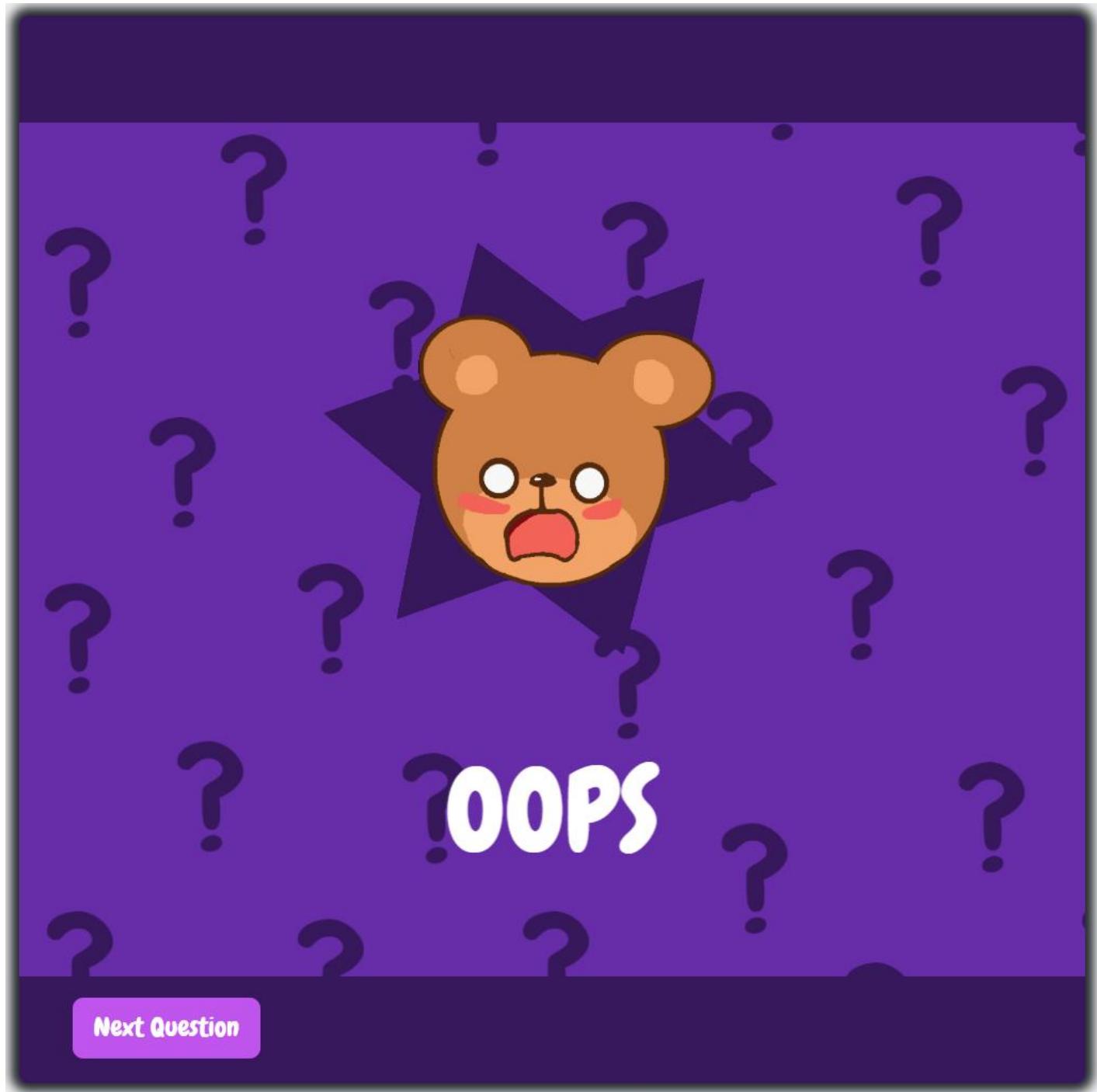
Fuhracksion

Frahction

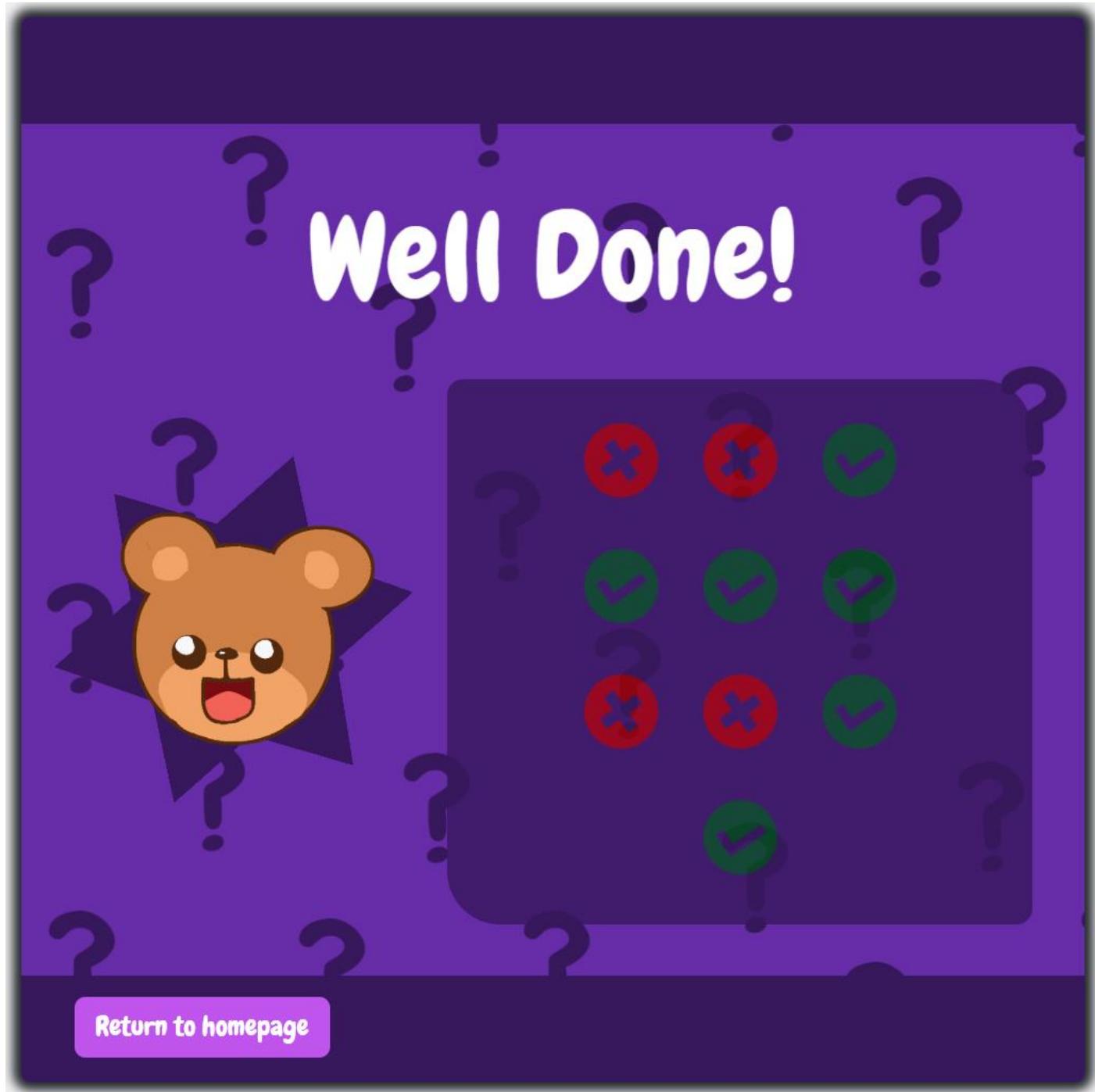
[Figure 6]



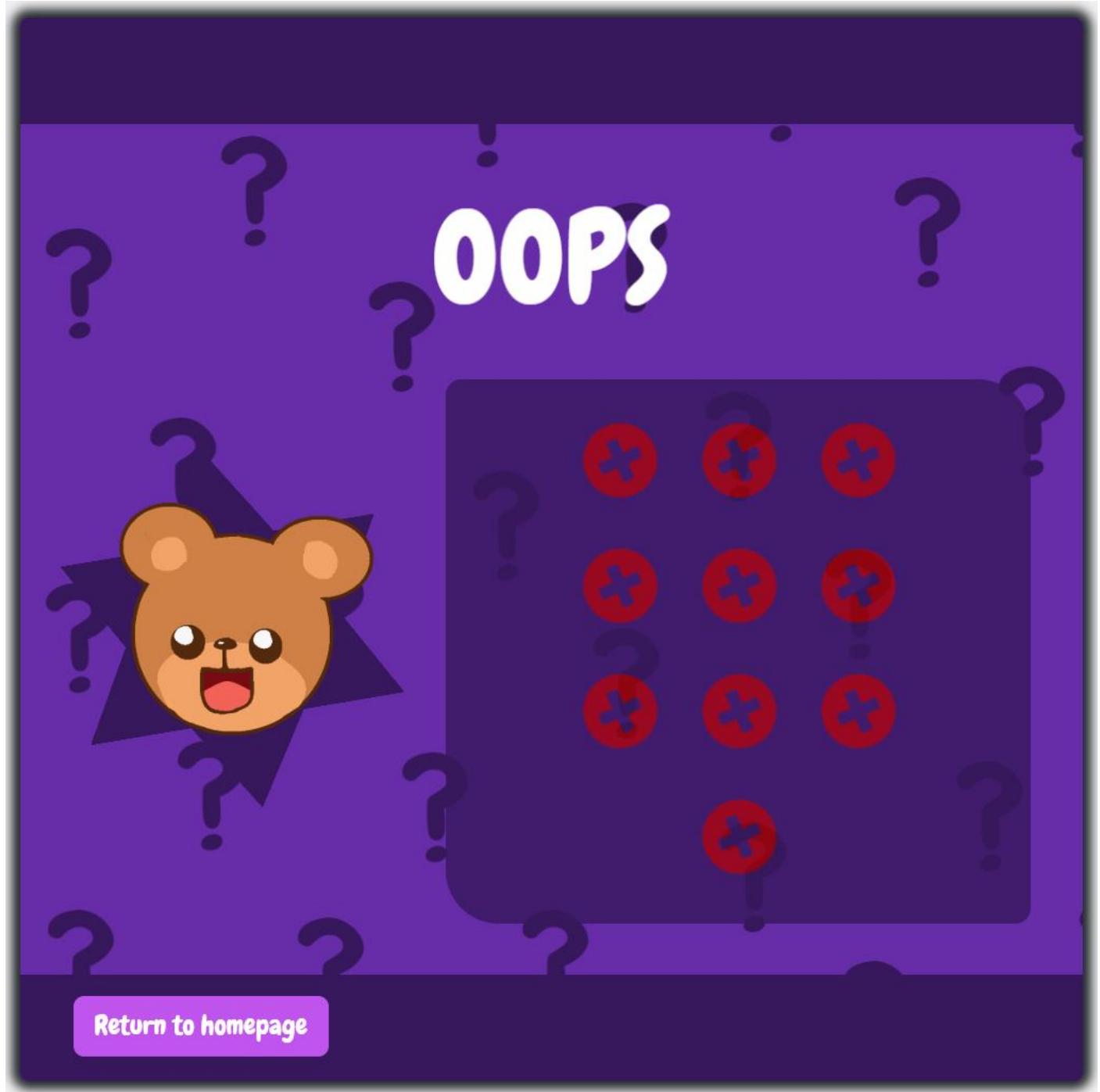
[Figure 7]



[Figure 8]



[Figure 9]



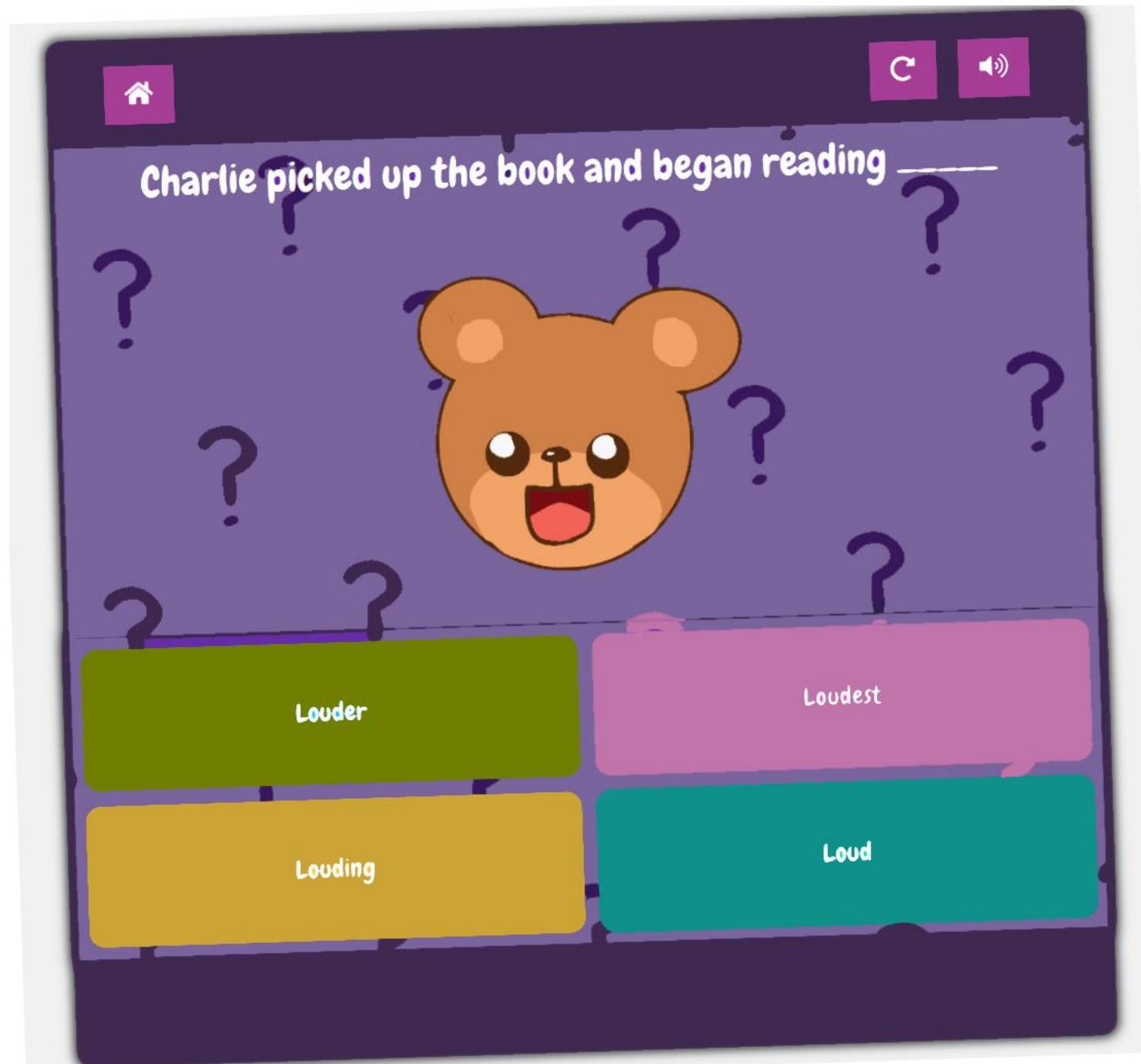
[Figure 10]



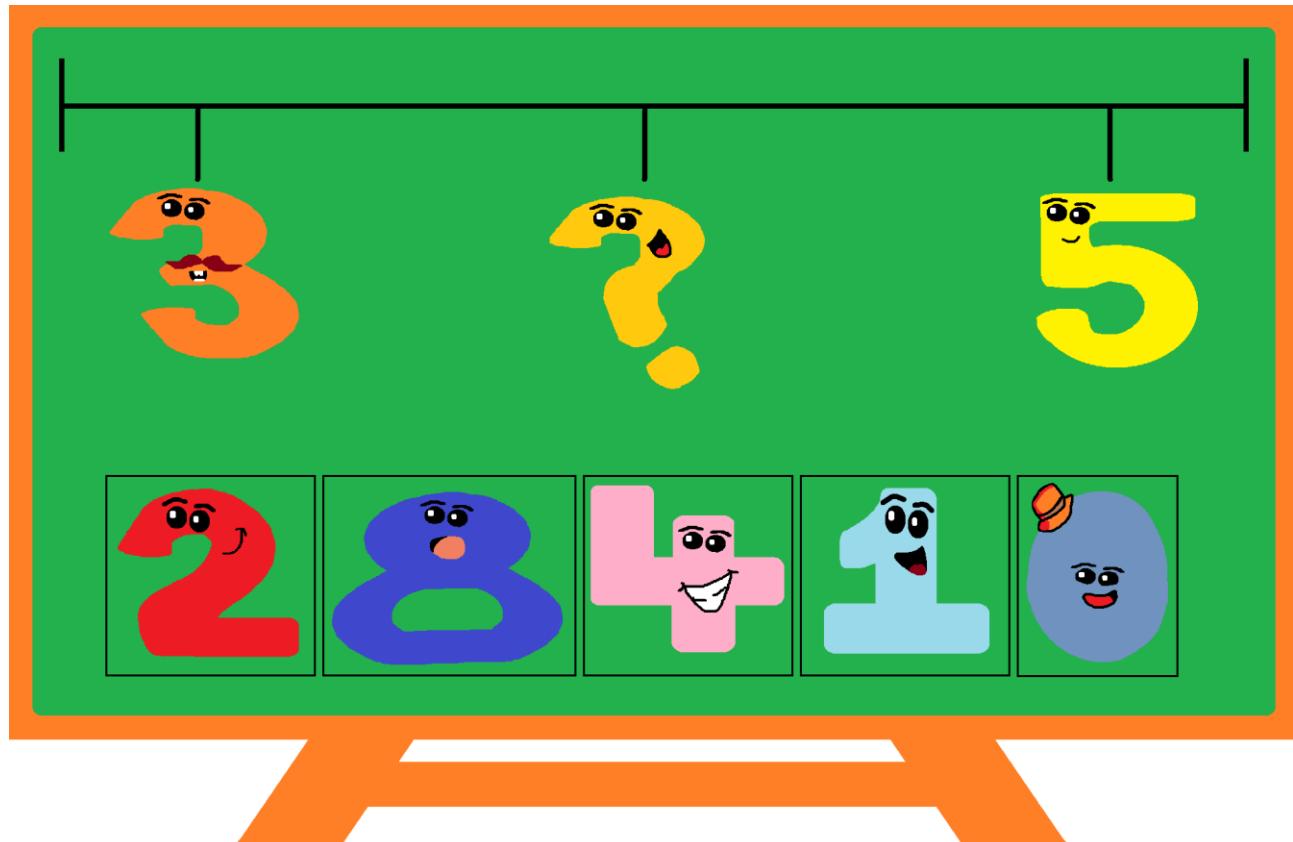
[Figure 11]



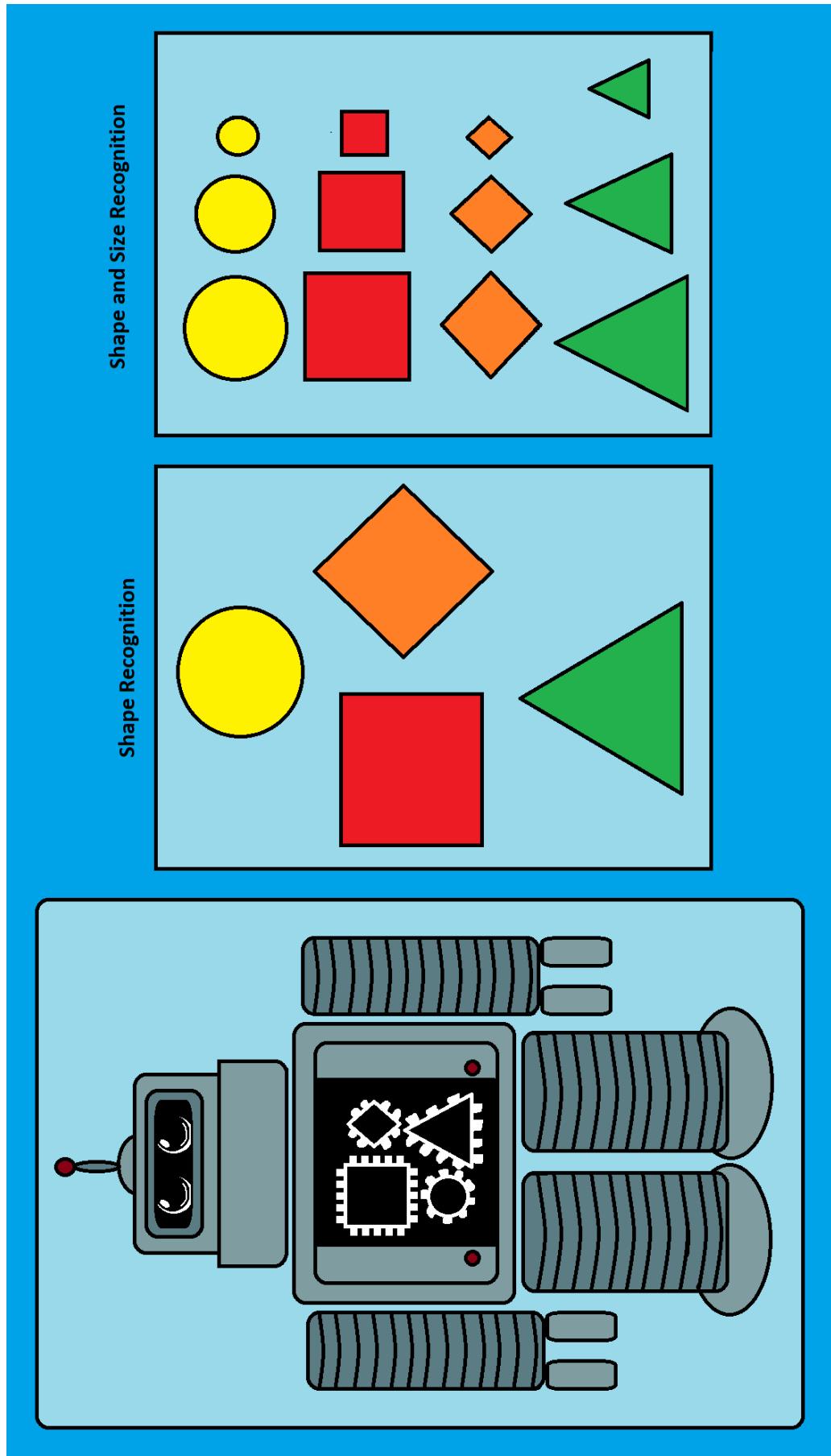
[Figure 12]



[Figure 13]



[Figure 14]



[Figure 15]



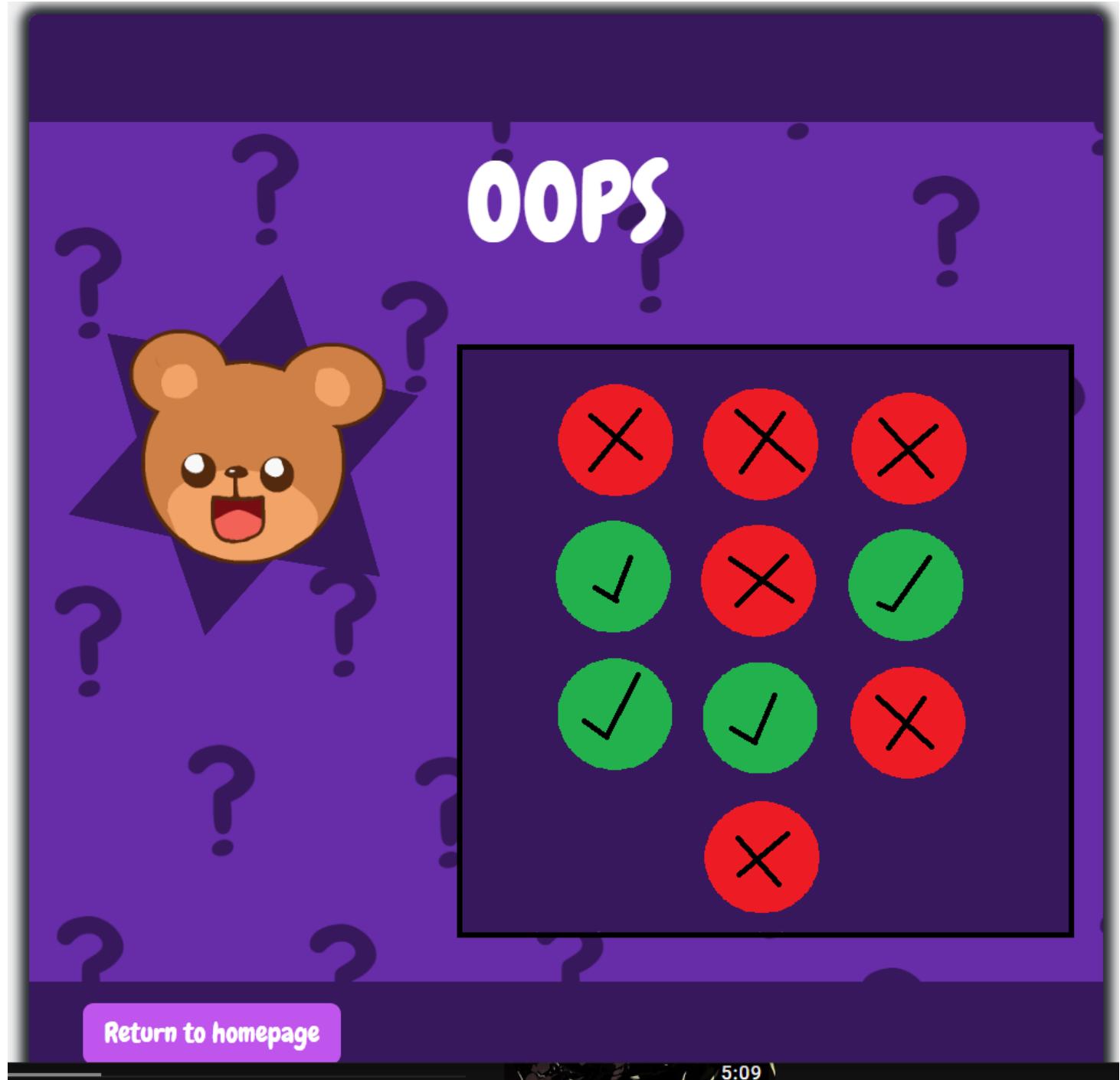
[Figure 16]



[Figure 17]



[Figure 18]



[Return to homepage](#)

5:09

[Figure 19]



[Figure 20]



## Appendix F: Code

[Figure 1]

### dbh.inc.php

```
<?php

$server = 'localhost';
$username = 'root';
$password = '';
$database = 'educationalappsystem';

$connection = mysqli_connect($server, $username, $password, $database);

if (!$connection) {
    die("Unfortunatly something went wrong. ".mysqli_connect_error()." Please try again in a little while...");
}
```

[Figure 2]

header.php

```
<?php
    session_start(); #header is in all webpages, having the session start
here ensure the session is accessable in all pages
    $current_page = basename($_SERVER['SCRIPT_NAME']); #determine what the
current file is
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta content="width=device-width, initial-scale=1.0, shrink-to-fit=no"
name="viewport">
        <meta content=" Fahim Hossain" name="author"><!-- Place your icon here
-->
        <!-- <link rel="icon" href="home-favicon.ico"> -->
        <title><?php #Depending on what the current file is, the title on the
browser changes
            if($current_page == 'view-results.php'){ echo 'Results
Page';}
            else if($current_page == 'view-class-results.php'){ echo
'Class Results Page';}
            else if($current_page == 'teacher-profile.php'){ echo 'Your
Profile';}
            else if($current_page == 'student-profile.php'){ echo 'Stu-
dent Profile';}
            else if($current_page == 'registration.php'){ echo 'Regis-
tration Page';}
            else if($current_page == 'login.php'){ echo 'Login Page';}
            else if($current_page == 'give-assignment.php'){ echo 'As-
signment Page';}
            else if($current_page == 'dashboard.php'){ echo 'Dash-
board';}
            else if($current_page == 'add-student.php'){ echo 'Student
Registration Page';}
            else if($current_page == 'activity-page.php'){ echo 'Activ-
ity Page';}
            else if($current_page == 'index.php'){ echo 'Home';}
?></title><!--fonts, stylesheet, bootstrap and awesomefont-->
    <link href="https://fonts.googleapis.com/css?family=Com-
ing+Soon|Chewy|Karla|Alfa+Slab+One|Orbitron&display=swap" rel="stylesheet">
    <link href="https://use.fontawesome.com/releases/v5.6.3/css/all.css"
rel="stylesheet">
    <link href="https://stackpath.bootstrapcdn.com/boot-
strap/4.3.1/css/bootstrap.min.css" rel="stylesheet">
    <link href="css/styles.css" rel="stylesheet" type="text/css">
</head>
<body>
</body>
</html>
```

**[Figure 3]**

**footer.php**

```
<div class="my-footer-container">
    <footer class="footer-content">
        <div class="row">
            <div class="col-xl-8">
                <div class="row">
                    <div class="col-xs-6 footer-links">
                        <div class="footer-heading">
                            About
                        </div>
                        <ul>
                            <li>
                                <a href="#">
                                    Our Mission
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    Meet the Team
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    Partners
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    News
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    Teachers & Parents
                                </a>
                            </li>
                        </ul>
                    </div>
                    <div class="col-xs-6 footer-links">
                        <div class="footer-heading">
                            Support
                        </div>
                        <ul>
                            <li>
                                <a href="#">
                                    Help center
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    Contact Us
                                </a>
                            </li>
                            <li>
                                <a href="#">
                                    FAQ
                                </a>
                            </li>
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </footer>
</div>
```

```

        </div>
    </div>
    <div class="col-xl-4">
        <div class="row social-links">
            <ul>
                <li>
                    <a href="https://www.facebook.com/" target="_blank">
                        <i class="fab fa-facebook-f"></i>
                    </a>
                </li>
                <li>
                    <a href="https://twitter.com/" target="_blank">
                        <i class="fab fa-twitter"></i>
                    </a>
                </li>
                <li>
                    <a href="https://www.youtube.com/" target="_blank">
                        <i class="fab fa-youtube"></i>
                    </a>
                </li>
                <li>
                    <a href="https://www.instagram.com/" target="_blank">
                        <i class="fab fa-instagram"></i>
                    </a>
                </li>
            </ul>
        </div>
        <div class="row app-links">
            <div class="col-sm-6">
                <a href="https://www.apple.com/uk/ios/app-store/" target="_blank">
                    
                </a>
            </div>
            <div class="col-sm-6">
                <a href="https://play.google.com/store?hl=en" target="_blank">
                    
                </a>
            </div>
        </div>
    </div>
</footer>
</div>

<!--jquery, bootstrap and charts.js--&gt;
&lt;script src="https://code.jquery.com/jquery-3.4.1.min.js" integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo=" crossorigin="anonymous"&gt;&lt;/script&gt;
&lt;script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-7/aI8w0QdNrFqtJ/ZPvzWDxFKZLu/NWjndRo391aKfC01+8TCqE7i8tWABvGKa2lq" crossorigin="anonymous"&gt;&lt;/script&gt;
&lt;script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0/dist/Chart.min.js"&gt;&lt;/script&gt;
</pre>

```

```
<?php #add script to the page they are used within
    $current_page = basename($_SERVER['SCRIPT_NAME']);
    if($current_page == 'activity-page.php'){ echo '<script
src="./js/activity-script.js"></script>' ;}
    if($current_page == 'view-results.php'){ echo '<script src="./js/view-
results-script.js"></script>' ;}
    if($current_page == 'view-class-results.php'){ echo '<script
src="./js/view-class-results-script.js"></script>' ;}
?>

</body>

</html>
```

**[Figure 4]**

**banner.php**

```
<div class="banner-container">

    <div class="banner">

        <h1 class="pb-3 display-4">
            <?php
                $message = 'WELCOME';
                if (isset($_SESSION['firstname'])){
                    $message .= ', '.$_SESSION['firstname'];
                }
                echo $message; #if logged in, welcome the user by name
            ?>
        </h1>
        <?php
            $current_page = basename($_SERVER['SCRIPT_NAME']);
            if($current_page == 'activity-page.php') { #if activity page,
display activity name
                echo '<h2 class="pb-3 display-5"> <strong> Lets see how
well you do on, '.$row['name'].' </strong> </h2>';
            } else if($current_page == 'index.php') { #if index page, dis-
play following
                echo '<h2 class="pb-3 display-5"> <strong> What would you
like to learn today? </strong> </h2>';
            } else { #all other pages, display following
                echo '<h2 class="pb-3 display-5"> <strong> What would you
like to do today? </strong> </h2>';
            }
        ?>
    </div>

</div>
```

[Figure 5]

navbar-default.php

```
<?php $current_page = basename($_SERVER['SCRIPT_NAME']); #determine the  
current file name to change what page is active in the navbar?>  
  
<div class="navigation-container nav-bg">  
    <nav class="navbar navbar-expand-xl navbar-light">  
        <button class="navbar-toggler" data-toggle="collapse" data-target="#my-nav">  
            <span class="navbar-toggler-icon"></span>  
        </button>  
        <a class="navbar-brand" href="#"> BRAND </a>  
        <div class="collapse navbar-collapse" id="my-nav">  
            <ul class="navbar-nav ml-auto">  
                <li class="nav-item <?php if($current_page == 'index.php') {  
echo 'activated'; }?>">  
                    <a class="nav-item nav-link home-tab" href="index.php">  
                        <span class="home-icon"></span> Home  
                    </a>  
                </li>  
  
                <li class="nav-item <?php if($current_page == 'contact.php') { echo 'activated'; }?>">  
                    <a class="nav-item nav-link contact-tab" href="#">  
                        <span class="contact-icon"></span> Contact  
                    </a>  
                </li>  
  
                <li class="nav-item <?php if($current_page == 'about.php') {  
echo 'activated'; }?>">  
                    <a class="nav-item nav-link about-tab" href="#">  
                        <span class="about-icon"></span> About  
                    </a>  
                </li>  
  
                <li class="nav-item <?php if($current_page == 'help.php') {  
echo 'activated'; }?>">  
                    <a class="nav-item nav-link help-tab" href="#">  
                        <span class="help-icon"></span> Help  
                    </a>  
                </li>  
  
                <li class="nav-item <?php if($current_page == 'login.php') {  
echo 'activated'; }?>">  
                    <a class="nav-item nav-link signin-tab"  
                        <?php  
                            if (isset($_SESSION['usertype'])) {  
                                echo 'href="inc/logout.inc.php"'; #if al-  
ready logged in, then log user out  
                            } else {  
                                echo 'href="login.php"'; #if not logged in,  
take user to login page  
                            }  
                        ?> >  
                        <span class="signin-icon"></span>  
                    <?php  
                        if (isset($_SESSION['usertype'])) {  
                            echo 'Sign Out'; #change labeling depending  
on function above  
                        } else {  
                            echo 'Sign In';  
                        }  
                    </a>  
                </li>  
            </ul>  
        </div>  
    </nav>  
</div>
```

```
        }
    ?>
    </a>
</li>
</ul>
</div>
</nav>
</div>
```

**[Figure 6]**

navbar-admin.php

```
<?php $current_page = basename($_SERVER['SCRIPT_NAME']);?>

<div class="navigation-container nav-bg">
    <nav class="navbar navbar-expand-xl navbar-light">
        <button class="navbar-toggler" data-toggle="collapse" data-target="#my-nav">
            <span class="navbar-toggler-icon"></span>
        </button>
        <a class="navbar-brand" href="#"> BRAND </a>
        <div class="collapse navbar-collapse" id="my-nav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item <?php if($current_page == 'registration.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link home-tab" href="registration.php" >
                        <span class="home-icon"></span> Registration
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'login.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link signin-tab"
                        <?php
                            if (isset($_SESSION['usertype'])){echo 'href="inc/logout.inc.php"';} else {echo 'href="login.php"';}
                        ?> >
                        <span class="signin-icon"></span>
                    <?php
                        if (isset($_SESSION['usertype'])){echo 'Sign Out';} else {echo 'Sign In';}
                    ?>
                    </a>
                </li>
            </ul>
        </div>
    </nav>
</div>
```

[Figure 7]

navbar-teacher.php

```
<?php $current_page = basename($_SERVER['SCRIPT_NAME']);?>

<div class="navigation-container nav-bg">
    <nav class="navbar navbar-expand-xl navbar-light">
        <button class="navbar-toggler" data-toggle="collapse" data-target="#my-nav">
            <span class="navbar-toggler-icon"></span>
        </button>
        <a class="navbar-brand" href="#"> BRAND </a>
        <div class="collapse navbar-collapse" id="my-nav">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item <?php if($current_page == 'dashboard.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link home-tab" href="dashboard.php">
                        <span class="home-icon"></span> Dashboard
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'store.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link store-tab" href="#">
                        <span class="store-icon"></span> Store
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'contact.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link contact-tab" href="#">
                        <span class="contact-icon"></span> Contact
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'about.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link about-tab" href="#">
                        <span class="about-icon"></span> About
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'help.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link help-tab" href="#">
                        <span class="help-icon"></span> Help
                    </a>
                </li>

                <li class="nav-item <?php if($current_page == 'login.php'){ echo 'activated';}?>">
                    <a class="nav-item nav-link signin-tab" >
                        <?php
                            if (isset($_SESSION['usertype'])) {
                                echo 'href="inc/logout.inc.php"';
                            } else {
                                echo 'href="login.php"';
                            }
                        ?> >
                        <span class="signin-icon"></span>
                    <?php
                
```

```
        if (isset($_SESSION['usertype'])) {
            echo 'Sign Out';
        } else {
            echo 'Sign In';
        }
    ?>
    </a>
</li>
</ul>
</div>
</nav>
</div>
```

**[Figure 8]**

registration.inc.php

```
<?php

session_start();

if (isset($_POST['signup-submit'])) { #if the user tried directly accessing
this file then send them to the index page

    require 'dbh.inc.php';

    $userFirstname = $_POST['firstname']; #Store firstname variable
    $userLastname = $_POST['lastname']; #Store lastname variable
    $userID = $_POST['username']; #Store username variable
    $userPass = $_POST['password']; #Store password variable
    $userRepeatPass = $_POST['repeat-password']; #Store repeat-password
variable

    $errorMessage = '?errors=found'; #error message for non password fields
    $errorMessagePass = '&password='; #error message for password fields
    $userFieldsValues = array($userFirstname, $userLastname, $userID);
#variables in an array to make iteration easier
    $userFieldsName = array('firstname', 'lastname', 'username'); #variables
in an array to make iteration easier
    $userFieldsChanges = False; #if validation failed, this will be turned
true
    $userFieldsPassChanges = False; #if validation failed, this will be
turned true

    if($_SESSION['usertype'] == 1){ #if a admin is creating the account
        $usertype = 2; #User is a teacher account
        $databaseTable = 'teachers'; #and will be inserted into the teach-
ers database
    } else if($_SESSION['usertype'] == 2) { #if a teacher is creating the
account
        $usertype = 3; #User is a student account
        $teacherid = $_SESSION['id']; #under this teacher
        $databaseTable = 'students'; #and will be inserted into the student
database
    }

    //EMPTY FIELD CHECK
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) { #check all fields
to see if any are empty
        if(empty($userFieldsValues[$i])){
            $userFieldsValues[$i] = 'empty'; #if so, change that field
value to empty so that the user can be informed
            $userFieldsChanges = True;
        }
    }
    if(empty($userPass)){
        $errorMessagePass .= 'empty';
        $userFieldsPassChanges = True;
    }

    //STRING LENGTH CHECK
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) { #check all fields
to see if any strings exceed their allowed length
        if(strlen($userFieldsValues[$i]) > 255){
            $userFieldsValues[$i] = 'toolong'; #if so, change that field
value to toolong so that the user can be informed
    }
}
```

```

        $userFieldsChanges = True;
    }
}

if(strlen($userPass) > 50){
    $errorMessagePass .= '&toolong';
    $userFieldsPassChanges = True;
}

//STRING CHARACTER CHECK
for ($i = 0; $i < sizeof($userFieldsValues) - 1; $i++) { #check all
fields to see if any strings invalid characters
    if(!preg_match("/^[\^W_0-9]*$/", $userFieldsValues[$i])){ #name
fields can only contain letters
        $userFieldsValues[$i] = 'invalidchar'; #if so, change that
field value to invalidchar so that the user can be informed
        $userFieldsChanges = True;
    }
}
if(!preg_match("/^[\w]*$/", $userFieldsValues[2])){ #username field can
contain letters, numbers and underscores
    $userFieldsValues[2] = 'invalidchar';
    $userFieldsChanges = True;
}
if(!preg_match("/^(?=.*\d)(?=.*[^W_0-9])[\w@%+!#$^?:.(){}[\]^~]*$/",
$userPass) && !$userFieldsPassChanges){ #password fields must have a let-
ter, a number and can contain these specified special characters
    $errorMessagePass .= 'invalidchar';
    $userFieldsPassChanges = True;
}

//REPEAT PASSWORD CHECK
if(($userPass != $userRepeatPass) && (!$userFieldsPassChanges)){ #check
to see if the two passwords match
    $errorMessagePass .= 'mismatch';
    $userFieldsPassChanges = True;
}

//EXISTING USER CHECK
$query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
$statement = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($statement, $query)){ #preparing query to avoid
SQL injections
    mysqli_stmt_bind_param($statement, 's', $userID); #binding the us-
er's input to the query
    mysqli_stmt_execute($statement); #Executing the query
    $results = mysqli_stmt_get_result($statement); #storing the results
in a variable
    if($row = mysqli_fetch_assoc($results)){ #if result found, the user
exists
        $userFieldsValues[2] = 'taken'; #change that field value to
taken so that the user can be informed
        $userFieldsChanges = True;
    }
} else {
    $userFieldsChanges = True; #if any problem occurred and the state-
ment couldn't be executed, send an error message
}

//Validation Check
if($userFieldsChanges || $userFieldsPassChanges){ #if any of these are
true, meaning validations failed, send the user back to their registration

```

```

page with the variables, if validation passed then that field will be au-
tofilled
    $errorMessage .= $errorMessagePass;
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
        $errorMessage .= '&' . $userFieldsName[$i] . '=' . $userFieldsVal-
ues[$i];
    }
    if($_SESSION['usertype'] == 1){
        header('Location: ../registration.php' . $errorMessage);
        exit();
    } else {
        header('Location: ../add-student.php' . $errorMessage);
        exit();
    }
}

//VALIDATION PASSED, INSERT DETAILS INTO DATABASE
$query = "INSERT INTO ". $databaseTable ." (firstname, lastname,
username, password, usertype";
if($databaseTable == 'teachers'){
    $query.= ') VALUES (?, ?, ?, ?, ?)';
} else {
    $query.= ', teacherid) VALUES (?, ?, ?, ?, ?, ?, ?)';
}

if(mysqli_stmt_prepare($statement, $query)){
    $hash = password_hash($userPass, PASSWORD_DEFAULT); #hash the pass-
word as we can't store plain text for security purposes
    if($databaseTable == 'teachers'){ #Student accounts have one more
field to store, the teacher id
        mysqli_stmt_bind_param($statement, 'ssssi', $userFirstname, $us-
erLastname, $userID, $hash, $usertype);
    } else {
        mysqli_stmt_bind_param($statement, 'ssssii', $userFirstname, $us-
erLastname, $userID, $hash, $usertype, $teacherid);
    }
    mysqli_stmt_execute($statement);
} else { #if an error occurred, then send an error message back
    $errorMessage .= $errorMessagePass;
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
        $errorMessage .= '&' . $userFieldsName[$i] . '=' . $userFieldsVal-
ues[$i];
    }
    if($_SESSION['usertype'] == 1){
        header('Location: ../registration.php' . $errorMessage);
        exit();
    } else {
        header('Location: ../add-student.php' . $errorMessage);
        exit();
    }
}

mysqli_stmt_close($statement);
mysqli_close($connection);

#Once successfully registers, send the user back to their registration
page with the success messages
if($_SESSION['usertype'] == 1){
    header('Location: ../registration.php?success');
    exit();
} else {

```

```
    header('Location: ../add-student.php?success') ;
    exit();
}

else {
    header('Location: ../index.php') ;
    exit();
}
```

## [Figure 9]

login.inc.php

```
<?php

if (isset($_POST['login-submit'])) { #if the user tried directly accessing
this file then send them to the index page or login page
    require 'dbh.inc.php';

    $userID = $_POST['username']; #Store username variable
    $userPass = $_POST['password']; #Store password variable
    $usertype = intval($_POST['usertype']); #Store usertype variable as an
int value

    if($usertype == 1){ #determine what database to look for the account in
        $databaseTable = 'admins';
    } else if($usertype == 2) {
        $databaseTable = 'teachers';
    } else if($usertype == 3) {
        $databaseTable = 'students';
    }

    $errorMessage = '?errors=found'; #error message for non password fields
    $errorMessagePass = '&password='; #error message for password fields
    $userFieldsChanges = False; #if validation failed, this will be turned
true

    //EMPTY FIELD CHECK
    if(empty($userID)){ #check all fields to see if any are empty
        $userID = 'empty'; #if so, change that field value to empty so that
the user can be informed
        $userFieldsChanges = True;
    }
    if(empty($userPass)){
        $errorMessagePass .= 'empty';
        $userFieldsChanges = True;
    }

    //EXISTING USER CHECK
    if(!$userFieldsChanges){
        $query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
        $statement = mysqli_stmt_init($connection);
        if(mysqli_stmt_prepare($statement, $query)){ #preparing query to
avoid SQL injections
            mysqli_stmt_bind_param($statement, 's', $userID); #binding the
user's input to the query
            mysqli_stmt_execute($statement); #Executing the query
            $results = mysqli_stmt_get_result($statement); #storing the re-
sults in a variable
            if(($row = mysqli_fetch_assoc($results))){ #if result found,
the user exists
                //PASSWORD CHECK
                if(!(password_verify($userPass, $row['password']))){ #com-
pare the hash of the password the user inputted with that stored in the da-
tabase
                    $errorMessagePass .= 'incorrect'; #if the hashes dont
match then the password is wrong
                    $userFieldsChanges = True;
                }
            } else {
                $userID = 'invaliduser'; #if no result found, the user
doesnt exists
            }
        }
    }
}
```

```

        $userFieldsChanges = True;
    }
} else {
    $userFieldsChanges = True; #if any problem occurred and the
statement couldn't be executed, send an error message
}
}

//Validation Check
if($userFieldsChanges){ #if any of the validations failed
    $errorMessage .= $errorMessagePass.'&username='.$userID.'&us-
ertype='.$usertype; #send the information back to the login page
    header('Location: ../login.php'.$errorMessage);
    exit();
} else{ #if all validations passed and the user was found
    session_start(); #start a session to log the user in and store the
account details into session variables
    if($usertype != 1){ #admin accounts dont have name details
        $_SESSION['firstname'] = $row['firstname'];
        $_SESSION['lastname'] = $row['lastname'];
    }
    $_SESSION['id'] = $row['id'];
    $_SESSION['username'] = $row['username'];
    $_SESSION['usertype'] = $row['usertype'];

    mysqli_stmt_close($statement);
    mysqli_close($connection);

    if($_SESSION['usertype'] == 1){ #Send the user to their respective
starting page
        header('Location: ../registration.php');
        exit();
    } else if($_SESSION['usertype'] == 2){
        header('Location: ../dashboard.php');
        exit();
    } else if($_SESSION['usertype'] == 3){
        header('Location: ../index.php');
        exit();
    }
}
} else {
    if (isset($_SESSION['usertype'])){
        header('Location: ../index.php');
        exit();
    } else {
        header('Location: ../login.php');
        exit();
    }
}

```

**[Figure 10]**

**logout.inc.php**

```
<?php
#when logging out, delete all session variables, close the session and send
user to the home page
session_start();
session_unset();
session_destroy();
header('Location: ../index.php?success');
exit();
```

## [Figure 11]

### edit-profile.inc.php

```
<?php
if (isset($_POST['edit-submit'])) { #if the user tried directly accessing
this file then send them to the index page

    require 'dbh.inc.php';

    $userFirstname = $_POST['firstname']; #Store firstname variable
    $userLastname = $_POST['lastname']; #Store lastname variable
    $newuserID = $_POST['new-username']; #Store new-username variable
    $olduserID = $_POST['old-username']; #Store old-username variable
    $newUserPass = $_POST['new-password']; #Store new-password variable
    $usertype = $_POST['usertype']; #Store usertype variable
    $id = $_POST['id']; #Store id variable

    $errorMessage = '?errors=found'; #error message for non password fields
    $errorMessagePass = '&password='; #error message for password fields
    $errorMessageNewPass = '&newpassword='; #error message for new password
fields
    $userFieldsValues = array($userFirstname, $userLastname, $newuserID);
#variables in an array to make iteration easier
    $userFieldsName = array('firstname', 'lastname', 'username'); #variables
in an array to make iteration easier
    $userFieldsChanges = False; #if validation failed, this will be turned
true
    $userFieldsPassChanges = False; #if validation failed, this will be
turned true
    $userFieldsNewPassChanges = False; #if validation failed, this will be
turned true
    $urlAdditions = $_POST['url']; #stores url for when user is taken back
to the profile page, their order is kept

    if($usertype == 2){ #if a teacher account is being edited
        $databaseTable = 'teachers'; #use the teacher database
        $currentUserPass = $_POST['current-password']; #store the password
(student accounts don't need passwords)
    } else if($usertype == 3) { #if a student account is being edited
        $databaseTable = 'students'; #use the student database
    } else if(empty($olduserID) || empty($id) || empty($usertype)){ #if any
of the important fields are empty, send user back with an error message
        header('Location: ../dashboard.php?error');
        exit();
    }

//USER ID VALIDATION
$query = "SELECT * FROM ". $databaseTable ." WHERE username=? && id=?";
#check to make sure the username and given id exists (incase user changes
this through the form)
$statement = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 'si', $olduserID, $id);
    mysqli_stmt_execute($statement);
    $results = mysqli_stmt_get_result($statement);
    if(!$row = mysqli_fetch_assoc($results)){ #if no user found, mean-
ing they don't match
        header('Location: ../dashboard.php?error'); #send user back
with an error message
        exit();
    }
} else {
```

```

        header('Location: ../dashboard.php?error'); #if any problem occurred
and the statement couldn't be executed, send an error message
        exit();
    }

    if(empty($newUserPass)){ #if new password field is empty
        $changePassword = False; #user doesn't wanna change their password
    } else {
        $changePassword = True; #if not, then they do
    }

    //EMPTY FIELD CHECK
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) { #check all fields
to see if any are empty
        if(empty($userFieldsValues[$i])){
            $userFieldsValues[$i] = 'empty'; #if so, change that field
value to empty so that the user can be informed
            $userFieldsChanges = True;
        }
    }
    if(empty($currentUserPass) && $usertype != 3){
        $errorMessagePass .= 'empty';
        $userFieldsPassChanges = True;
    }

    //STRING LENGTH CHECK
    for ($i = 0; $i < sizeof($userFieldsValues); $i++) { #check all fields
to see if any strings exceed their allowed length
        if(strlen($userFieldsValues[$i]) > 255){
            $userFieldsValues[$i] = 'toolong'; #if so, change that field
value to toolong so that the user can be informed
            $userFieldsChanges = True;
        }
    }
    if(strlen($newUserPass) > 50){
        $errorMessageNewPass .= 'toolong';
        $userFieldsNewPassChanges = True;
    }

    //STRING CHARACTER CHECK
    for ($i = 0; $i < sizeof($userFieldsValues) - 1; $i++) { #check all
fields to see if any strings invalid characters
        if(!preg_match("/^[\w_0-9]*$/", $userFieldsValues[$i])){ #name
fields can only contain letters
            $userFieldsValues[$i] = 'invalidchar'; #if so, change that
field value to invalidchar so that the user can be informed
            $userFieldsChanges = True;
        }
    }
    if(!preg_match("/^[\w]*$/", $userFieldsValues[2])){ #username field can
contain letters, numbers and underscores
        $userFieldsValues[2] = 'invalidchar';
        $userFieldsChanges = True;
    }
    if(!preg_match("/^(?=.*\d)(?=.*[^W_0-9])[\w@%+!#$^?:.(){}[\]\~]*$/",
$newUserPass) && !$userFieldsNewPassChanges && $changePassword){ #password
fields must have a letter, a number and can contain these specified special
characters
        $errorMessageNewPass .= 'invalidchar';
        $userFieldsNewPassChanges = True;
    }
}

```

```

//EXISTING USER NAME
if($newuserID != $olduserID){
    $query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
    if(mysqli_stmt_prepare($statement, $query)){
        mysqli_stmt_bind_param($statement, 's', $newuserID);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if($row = mysqli_fetch_assoc($results)){ #if result found, the
user already exists
            $userFieldsValues[2] = 'taken'; #change that field value to
taken so that the user can be informed
            $userFieldsChanges = True;
        }
    } else {
        header('Location: ../dashboard.php?error'); #if any problem oc-
cured and the statement couldn't be executed, send an error message
        exit();
    }
}

//PASSWORD CHECK
if(!$userFieldsPassChanges && $usertype != 3){ #as long as the password
field wasn't changed (since if the system changed it for error messages
then it wouldn't make sense to validate) and the account isn't a student
account
    $query = "SELECT * FROM ". $databaseTable ." WHERE username=?";
    if(mysqli_stmt_prepare($statement, $query)){
        mysqli_stmt_bind_param($statement, 's', $olduserID);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if($row = mysqli_fetch_assoc($results)){
            if(!password_verify($currentUserPass, $row['password'])){ #compare
the hash of the password the user inputted with that stored in the
database
                $errorMessagePass .= 'incorrect'; #if the hashes dont
match then the password is wrong
                $userFieldsChanges = True;
            }
        } else {
            header('Location: ../teacher-profile.php?error'); #if any
problem occurred, send an error message
            exit();
        }
    } else {
        header('Location: ../teacher-profile.php?error'); #if any prob-
lem occurred and the statement couldn't be executed, send an error message
        exit();
    }
}

//Validation Check
if($userFieldsChanges || $userFieldsPassChanges || $userFieldsNewPass-
Changes) { #if any of the validations failed
    if($usertype != 3){ #password fields for non student accounts
        $errorMessage .= $errorMessagePass.$errorMessageNewPass;
        for ($i = 0; $i < sizeof($userFieldsValues); $i++) {
            $errorMessage .= '&'.$userFieldsName[$i].'='.$userFields-
Values[$i];
        }
    }
}

```

```

        if($usertype == 2){ #teacher account gets taken to teacher profile
            header('Location: ../teacher-profile.php' . $errorMessage);
            exit();
        } else if($usertype == 3) { #student account gets taken to student
profile
            header('Location: ../student-profile.php' . $errorMessage . $ur-
lAdditions);
            exit();
        }

    //VALIDATION PASSED, UPDATE DETAILS IN DATABASE
    $query = "UPDATE ". $databaseTable ." SET firstname=? , lastname=? ,
username=?";
    if($changePassword){
        $query .= ', password=?';
    }
    $query .= ' WHERE id=?';
    if(mysqli_stmt_prepare($statement, $query)){
        if($changePassword){
            $hash = password_hash($newUserPass, PASSWORD_DEFAULT);
            mysqli_stmt_bind_param($statement, 'sssss', $userFirstname,
$userLastname, $newuserID, $hash, $id);
        } else {
            mysqli_stmt_bind_param($statement, 'ssss', $userFirstname, $us-
erLastname, $newuserID, $id);
        }
        mysqli_stmt_execute($statement);
    } else {
        header('Location: ../dashboard.php?error');
        exit();
    }

    mysqli_stmt_close($statement);
    mysqli_close($connection);

    if($usertype == 2){ #teacher account need to update sessions since val-
ues changed
        session_start();
        $_SESSION['firstname'] = $userFirstname;
        $_SESSION['lastname'] = $userLastname;
        $_SESSION['username'] = $newuserID;
        header('Location: ../teacher-profile.php?success'); #teacher ac-
count gets taken to teacher profile
        exit();
    } else if($usertype == 3){
        header('Location: ../student-profile.php?success' . $urlAdditions);
#student account gets taken to student profile
        exit();
    }
}
else {
    header('Location: ../index.php');
    exit();
}

```

## [Figure 12]

### give-assignment.inc.php

```
<?php
session_start();

if (isset($_POST['assign-submit'])) { #if the user tried directly accessing
this file then send them to the index page
    require 'dbh.inc.php';

    $userID = $_POST['username']; #Store username variable
    $assignmentID = $_POST['assignment']; #Store assignment variable
    $studentID = $_POST['studentId']; #Store studentId variable
    $urlAdditions = $_POST['url']; #Store url variable
    $urlError = 'Location: ../give-assignment.php?error' . $urlAdditions;
#error message
    $urlSuccess = 'Location: ../give-assignment.php?success' . $urlAdditions;
#success message

    //USER INPUT VALIDATION
    if(empty($userID) || empty($assignmentID) || empty($studentID)){ #if
any of the variables are empty, send error message
        header($urlError);
        exit();
    }

    //USER ID VALIDATION
    $query = "SELECT * FROM students WHERE username=? && id=?"; #find the
student requested
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query)){
        mysqli_stmt_bind_param($statement, 'si', $userID, $studentID);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if(!$row = mysqli_fetch_assoc($results)){ #if student not found
send error message
            header($urlError);
            exit();
        }
    } else {
        header($urlError); #if error occured during retrieval send error
message
        exit();
    }

    //EXISTING ACTIVITY VALIDATION (Ignore if none since that means no as-
signments given)
    if($assignmentID != 'none') {
        $query = "SELECT * FROM activities WHERE id = ?"; #find the activ-
ity requested
        if(mysqli_stmt_prepare($statement, $query)) {
            mysqli_stmt_bind_param($statement, 'i', $assignmentID);
            mysqli_stmt_execute($statement);
            $results = mysqli_stmt_get_result($statement);
            if(!($row = mysqli_fetch_assoc($results))){ #if activity not
found send error message
                header($urlError);
                exit();
            }
        } else {
            header($urlError); #if error occured during retrieval send error
message
        }
    }
}
```

```

        exit();
    }

//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment WHERE studentid = ?"; #find
id the student has existing assignment
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 'i', $studentID);
    mysqli_stmt_execute($statement);
    $results = mysqli_stmt_get_result($statement);
    if (($row = mysqli_fetch_assoc($results))) { #if the student does
        //DELETING EXISTING STORED RESULTS
        $query = "DELETE FROM results_storage WHERE assignmentid = ?";
#get all the results for this assignment and delete the
        if(mysqli_stmt_prepare($statement, $query)) {
            mysqli_stmt_bind_param($statement, 'i', $row['id']);
            mysqli_stmt_execute($statement);
            //DELETING ASSIGNMENT ITSELF (second because it is a for-
eign id to results_storage)
            $query = "DELETE FROM activity_assignment WHERE id = ?";
            echo ''.$row['id'].' and '.$row['time'];
            if(mysqli_stmt_prepare($statement, $query)) {
                mysqli_stmt_bind_param($statement, 'i', $row['id']);
                mysqli_stmt_execute($statement);
            } else {
                header($urlError);#if error occured during retrieval
send error message
                exit();
            }
        } else {
            header($urlError);#if error occured during retrieval send
error message
            exit();
        }
    } else {
        header($urlError);#if error occured during retrieval send error mes-
sage
        exit();
    }
//ASSIGNING NEW ASSIGNMENT TO STUDENT
if($assignmentID != 'none'){ #if assignment is to be made
    $query = "INSERT INTO activity_assignment (teacherid, studentid,
activityid) VALUES (?,?,?)"; #create the assignment
    if(mysqli_stmt_prepare($statement, $query)) {
        mysqli_stmt_bind_param($statement, 'iii', $_SESSION['id'],
$studentID, $assignmentID);
        mysqli_stmt_execute($statement);
    } else {
        header($urlError);#if error occured during retrieval send error
message
        exit();
    }
}

header($urlSuccess); #send user back with success message
exit();

}
else {

```

```
header('Location: ../index.php');
exit();
}
```

[Figure 13]

results-submit.inc.php

```
<?php
session_start();

if (isset($_POST['submit'])) { #if the user tried directly accessing this
file then send them to the index page
    require 'dbh.inc.php';

    $data = $_POST['data']; #Store data variable
    $overallTime = $_POST['time']; #Store time variable
    $activityID = $_POST['activityid']; #Store activityid variable
    $overallAttempts; #Store overall attempts
    $errorMessage = "Unfortunatly something went wrong, unable to submit
your results.\nPlease try again in a little while..."; #error message
    $successMessage = "Results Stored!"; #success message

    //EMPTY VALIDATION
    if( empty($data) || empty($overallTime) || empty($activityID)){ #if any
of the important fields are empty, send user back with an error message
        echo $errorMessage;
        exit();
    }

    //FORMAT VALIDATION
    for($i = 0; $i < sizeof($data); $i++) { #loop through the data to en-
sure no tampering was made
        for($j = 0; $j < sizeof($data[$i]['choicesList']); $j++) { #make
sure the choices only have letters, numbers and these specific special
characters
            if( !preg_match("/^a-zA-Z0-9!?,* ]*$/", $data[$i]['choic-
esList'][$j]) ){
                echo $errorMessage; #if fail, send user back with error
message
                exit();
            }
        }
        for($j = 0; $j < sizeof($data[$i]['choicesPicked']); $j++) { #make
sure the choices picked only have letters, numbers and these specific spe-
cial characters
            if( !preg_match("/^a-zA-Z0-9!?,* ]*$/", $data[$i]['choic-
esPicked'][$j]) ){
                echo $errorMessage; #if fail, send user back with error
message
                exit();
            }
        }
        if( !preg_match("/^a-zA-Z_,?'. ]*$/", $data[$i]['questionTitle'])
|| !preg_match("/^a-zA-Z0-9! ]*$/", $data[$i]['correctChoice']) ) { #make
sure the question titles only have letters and these specific special char-
acters
            echo $errorMessage; #if fail, send user back with error message
            exit();
        }
        if( !in_array($data[$i]['questionType'], array('Spelling','Under-
standing','Punctuation','Grammar')) ){ #make sure the question types are
one of these strings
            echo $errorMessage; #if fail, send user back with error message
            exit();
    }
}
```

```

        if( !is_numeric($data[$i]['attempts']) || !is_numeric($data[$i]['timeTaken']) || !is_numeric($data[$i]['correct']) || !is_numeric($overallTime) || !is_numeric($activityID) ){ #make sure these values are only numbers
            echo $errorMessage; #if fail, send user back with error message
            exit();
        }
    }

//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment WHERE studentid = ?"; #find the students assignment
$statement = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($statement, $query)) {
    mysqli_stmt_bind_param($statement, 'i', $_SESSION['id']);
    mysqli_stmt_execute($statement);
    $assignmentResults = mysqli_stmt_get_result($statement);
    while ( ($assignmentRow = mysqli_fetch_assoc($assignmentResults)) ) { #if no assignment, this all gets skipped
        if( $assignmentRow['activityid'] == $activityID ) { #however, if this is an assignment then...
            $overallAttempts = $assignmentRow['attempts'] + 1; #increment attempts by 1
            $query = "DELETE FROM results_storage WHERE assignmentid = ?"; #delete any existing results
            if(mysqli_stmt_prepare($statement, $query)) {
                mysqli_stmt_bind_param($statement, 'i', $assignmentRow['id']);
                mysqli_stmt_execute($statement);
                $query = "UPDATE activity_assignment SET attempts=?, time=? WHERE id=?"; #update with the new attempt count and time
                if(mysqli_stmt_prepare($statement, $query)) {
                    mysqli_stmt_bind_param($statement, 'iii', $overallAttempts, $overallTime, $assignmentRow['id']);
                    mysqli_stmt_execute($statement);
                    $query = "INSERT INTO results_storage (assignmentid, question, correct, attempts, time, type, choices, picked, title, answer) VALUES (?,?,?,?,?,?,?,?,?,?)"; #update with the new results
                    for($i = 0; $i < sizeof($data); $i++){ #loop through the data to insert everything
                        $choicePicked = implode(' ', $data[$i]['choicesPicked']);
                        $choiceList = implode(' ', $data[$i]['choiceList']);
                        if(mysqli_stmt_prepare($statement, $query)) {
                            mysqli_stmt_bind_param($statement, 'iiiiisssss', $assignmentRow['id'], $i, $data[$i]['correct'], $data[$i]['attempts'], $data[$i]['timeTaken'], $data[$i]['questionType'], $choiceList, $choicePicked, $data[$i]['questionTitle'], $data[$i]['correctChoice']);
                            mysqli_stmt_execute($statement);
                        } else {
                            echo $errorMessage; #if an error occurred, send error message
                            exit();
                        }
                    }
                } else {
                    echo $errorMessage; #if an error occurred, send error message
                    exit();
                }
            }
        }
    }
}

```

```
        }
    } else {
        echo $errorMessage; #if an error occurred, send error
message
        exit();
    }
}
} else {
    echo $errorMessage; #if an error occurred, send error message
    exit();
}

mysqli_stmt_close($statement);
mysqli_close($connection);
echo $successMessage; #echo sucess message if data was stored
}
else {
    header('Location: ../index.php');
    exit();
}
```

## [Figure 14]

### view-results.inc.php

```
<?php
session_start();

if (isset($_POST['submit'])) { #if the user tried directly accessing this
file then send them to the index page
    require 'dbh.inc.php';

    $userID = $_POST['username']; #Store username variable
    $type = $_POST['typeOfData']; #Store typeOfData variable
    $errorMessage = "Unfortunatly something might have gone wrong, unable
to show your results.<br>Please try again in a little while..."; #error
message to be shown to the user
    $storedResults = []; #array to store results

    //USER INPUT VALIDATION
    if(empty($userID) && $type != 'class'){ #if an individual result is re-
quested and the user id is empty, tell the user something went wrong
        echo $errorMessage;
        exit();
    }

    //TYPE OF DATA VALIDATION
    if(!in_array($type, array('individual','class'))){ #if the user edit
the type of data varaiable to be something other than individual or class,
don't do anything and just send an error message
        echo $errorMessage;
        exit();
    }

    if($type == 'individual') { #if an individual result is requested
        //USER VALIDATION
        $query = "SELECT * FROM students WHERE username=?"; #get the stu-
dents account details
        $statement = mysqli_stmt_init($connection);
        if(mysqli_stmt_prepare($statement, $query)){
            mysqli_stmt_bind_param($statement, 's', $userID);
            mysqli_stmt_execute($statement);
            $studentResults = mysqli_stmt_get_result($statement);
            if(!$studentRow = mysqli_fetch_assoc($studentResults)){ #if
user isn't found, send error message letting the user know
                echo 'Unfortunatly something might have gone wrong, the
user appears to not exist.';
                exit();
            } else {
                mysqli_data_seek($studentResults,0); #if user is found, re-
set pointer
            }
        } else {
            echo $errorMessage; #if an error occured during reterival, send
error message
            exit();
        }

        //USER ASSIGNMENT VALIDATION
        $query = "SELECT * FROM activity_assignment WHERE studentid = ?";
#get the student's assignment details
        if(mysqli_stmt_prepare($statement, $query)){
            mysqli_stmt_bind_param($statement, 'i', $studentRow['id']);
            mysqli_stmt_execute($statement);
        }
    }
}
```

```

$assignmentResults = mysqli_stmt_get_result($statement);
if(!$assignmentRow = mysqli_fetch_assoc($assignmentResults)){
#if assignment isn't found, send error message letting the user know
    echo 'Unfortunately something might have gone wrong, the
user appears to not have been tasked any assignments.';
    exit();
} else {
    if($assignmentRow['attempts'] == 0){ #if assignment isn't
attempted yet, send error message letting the user know
        echo 'The user has yet to attempt the assign-
ment.<br>Please try again in a little while...';
        exit();
    }
    mysqli_data_seek($assignmentResults,0); #if assignment is
found, reset pointer
}
} else {
    echo $errorMessage; #if an error occurred during retrieval, send
error message
    exit();
}

//RESULTS EXISTING CHECK
$query = "SELECT * FROM results_storage WHERE assignmentid = ?";
#get the student's assignment results
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 'i', $assignmentRow['id']);
    mysqli_stmt_execute($statement);
    $results = mysqli_stmt_get_result($statement);
    if(!$row = mysqli_fetch_assoc($results)){ #if results isn't
found, send error message letting the user know
        echo 'Unfortunately something went wrong and we couldn\'t
store the user\'s results<br>Please advise user to try the test again.';
        exit();
    } else {
        mysqli_data_seek($results,0); #if results is found, reset
pointer
        while($row = mysqli_fetch_assoc($results)){ #loop through
all the results and push them to the results array
            array_push($storedResults, $row);
        }
        array_push($storedResults, $assignmentRow['attempts']); #at
end, push attempts and time in seconds
        array_push($storedResults, $assignmentRow['time']);
    }
} else {
    echo $errorMessage; #if an error occurred during retrieval, send
error message
    exit();
}
} else { #if a class result is requested
//CLASS ASSIGNMENT VALIDATION
$query = "SELECT * FROM activity_assignment WHERE teacherid=?";
#get all the assignments, the teacher gave
$statement = mysqli_stmt_init($connection);
if(mysqli_stmt_prepare($statement, $query)){
    mysqli_stmt_bind_param($statement, 'i', $_SESSION['id']);
    mysqli_stmt_execute($statement);
    $assignmentResults = mysqli_stmt_get_result($statement);
    if(!$assignmentRow = mysqli_fetch_assoc($assignmentResults)){
#if assignments isn't found, send error message letting the user know

```

```

        echo 'There appears to be no active assignments to dis-
play.';
        exit();
    }
} else {
    echo $errorMessage; #if an error occurred during reterival, send
error message
    exit();
}

//CLASS RESULTS EXISTING CHECK
mysqli_data_seek($assignmentResults,0); #reset pointer
$classAttempts = 0; #initially start at 0
$classTime = 0; #initially start at 0
$count = 0; #initially start at 0
while($assignmentRow = mysqli_fetch_assoc($assignmentResults)){
#loop through all the assignments
    $query = "SELECT * FROM results_storage WHERE assignmentid =
?"; #get the result of each assignment
    if(mysqli_stmt_prepare($statement, $query)){
        mysqli_stmt_bind_param($statement, 'i', $assignmen-
tRow['id']);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        $count++;
        $classAttempts += intval($assignmentRow['attempts']); #add
this student's attempts to the tally
        $classTime += intval($assignmentRow['time']); #add this
student's time to the tally
        if(!$row = mysqli_fetch_assoc($results)) { #if no results
found, send error message letting the user know
            echo 'There appears to be no results on record to dis-
play.<br>No student has yet to attempt the assigned task.';
            exit();
        } else {
            mysqli_data_seek($results,0); #if results found, reset
pointer
        }
        while($row = mysqli_fetch_assoc($results)){ #loop through
all the results and push them to the results array
            array_push($storedResults, $row);
        }
    } else {
        echo $errorMessage; #if an error occurred during reterival,
send error message
        exit();
    }
}
array_push($storedResults, $count); #at end, push count, attempts
and time in seconds
array_push($storedResults, $classAttempts);
array_push($storedResults, $classTime);
}

echo json_encode($storedResults); #send the data back in a json format
}
else {
    header('Location: ../index.php');
    exit();
}

```

## [Figure 15]

### index.php

```
<?php

    require './inc/header.php';
    require './inc/navbar-default.php';
    include './inc/banner.php';

    if (isset($_SESSION['usertype'])){
        if($_SESSION['usertype'] == 1){ #if logged in teacher or admin accounts tried accessing the page they are redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 2){
            header('Location: dashboard.php');
            exit();
        }
    }
?>

<div class="main-container">

    <div class="content">
        <div class="activity-container">
            <div class="row">
                <div class="col-sm-12 activities">
                    <div id="english" class="row">
                        <div class="col-sm-6 thumbnail">
                            <a href="activity-page.php?activityID=1">
                                
                            </a>
                        </div>
                        <div class="col-sm-6 thumbnail">
                            <a href="activity-page.php?activityID=1">
                                
                            </a>
                        </div>
                    </div>
                    <div class="row">
                        <div id="maths" class="col-sm-8">
                            <div class="row">
                                <div class="col-sm-6 thumbnail">
                                    <a href="activity-page.php?activityID=1">
                                        
                                    </a>
                                </div>
                                <div class="col-sm-6 thumbnail">
                                    <a href="activity-page.php?activityID=1">
                                        
                                    </a>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```

                <a href="activity-page.php?activityID=1">
                    
                </a>
            </div>
            <div class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1">
                    
                </a>
            </div>
            <div class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1">
                    
                </a>
            </div>
            <div id="science" class="col-sm-4 thumbnail">
                <a href="activity-page.php?activityID=1">
                    
                </a>
            </div>
            <div id="other" class="row">
                <div class="col-sm-4 thumbnail">
                    <a href="activity-page.php?activityID=1">
                        
                    </a>
                </div>
                <div class="col-sm-4 thumbnail">
                    <a href="activity-page.php?activityID=1">
                        
                    </a>
                </div>
                <div class="col-sm-4 thumbnail">
                    <a href="activity-page.php?activityID=1">
                        
                    </a>
                </div>
            </div>
        </div>
    <div class="panel">
        <div class="panel-content">
            <div class="row">
                <div class="col-x1-6">
                    <div class="panel-text-container">
                        <div class="panel-text">
                            <h1>Our Mission</h1>

```

```
<p>
    The primary role of a teacher in a class-
room is to impart knowledge to their students. Teachers are given a curric-
ulum that they must follow and ensure every child placed in their care un-
derstands the content to a satisfactory degree. To test this, standardised
exams are taken to assess students' progress throughout their years in edu-
cation. This, however, has fallen under scrutiny in recent years on whether
this is the best approach. Standardised exams put unneeded stress onto stu-
dents and recent developments show that teacher assessments are just as ac-
curate at predicting future success as standardised test anyway and is much
less detrimental to students' mental health. This has prompted me to create
a tool that helps teachers in their assessment practises, with the hope
that, with advancement in teacher assessment technologies, we can move away
from these standardised tests which in turn will increase student mental
wellbeing, increase accuracy of assessment and increase quality of teach-
ing.
```

```
</p>
</div>
</div>
</div>
<div class="col-x1-6">
    <div class="carousel-container">
        <div class="my-carousel">
            <div id="myCarousel" class="carousel slide"
data-ride="carousel" data-interval="5000">
                <div class="carousel-inner">
                    <div class="carousel-item active">
                        
                    </div>
                    <div class="carousel-item">
                        
                    </div>
                </div>
                <a class="carousel-control-prev" href="#my-
Carousel" role="button" data-slide="prev">
                    <span class="carousel-control-prev-
icon" aria-hidden="true"></span>
                    <span class="sr-only">Previous</span>
                </a>
                <a class="carousel-control-next" href="#my-
Carousel" role="button" data-slide="next">
                    <span class="carousel-control-next-
icon" aria-hidden="true"></span>
                    <span class="sr-only">Next</span>
                </a>
            </div>
        </div>
    </div>
</div>
</div>

</div>

<?php require './inc/footer.php'; ?>
```

## [Figure 16]

### login.php

```
<?php

    require './inc/header.php';
    require './inc/navbar-default.php';

?>

<div class="login-container">
    <div class="login-content">
        <div class="form-img-container">
            
        </div>
        <div class="form-input-container">
            <form action="inc/login.inc.php" method="post">
                <div class="login-title">
                    Log In to Your Account
                </div>
                <?php
                    if(isset($_GET['errors'])){ #if error occurred, then
display an error message
                        echo '<div class="h5" style="color:red;">Something
Went Wrong!</div>';
                    }
                ?>
                <input type="text" name="username" <?php
                    $field = 'placeholder="Username"';
                    $style = 'class= "default"';
                    if(isset($_GET['errors'])){
                        if ( !in_array($_GET['username'], array('empty','invaliduser')) ) {#if not an error message
                            $field = 'value='.$_GET['username']; #auto
fill the field with previous inputed value
                        } else {
                            $style = 'class= "error"';
                        }
                    }
                    echo $field.' '.$style; #if error, turn field border
to red, else keep the field as its default
                ?> maxlength="255">
                <?php
                    if(isset($_GET['errors'])){
                        if( $_GET['username'] == 'empty' ) {
                            echo '<div class="h5" style="color:red; text-
align: left;">This field is required.</div>';
                        } else if( $_GET['username'] == 'invaliduser' ) {
                            echo '<div class="h5" style="color:red; text-
align: left;">This User Doesn\'t Exist.</div>';
                        }
                    } #if error, inform user of the type of error
                ?>
                <input type="password" name="password" placeholder="Pass-
word" <?php
                    $style = 'class = "default"';
                    if(isset($_GET['errors'])){
                        if ( in_array($_GET['password'], array('emp-
ty','incorrect',''))) ) {
                            $style = 'class= "error"';
                        }
                    }
                ?>
```

```

        echo $style;
    ?> maxlength="50">
<?php
    if(isset($_GET['errors'])){
        if( $_GET['password'] == 'empty' ){
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if( $_GET['password'] == 'incorrect' ){
            echo '<div class="h5" style="color:red; text-align: left;">Incorrect Password.</div>';
        }
    }
?>
<label>Account Type</label>
<select name="usertype">
    <option value="1" <?php
        if(isset($_GET['errors'])){
            if ( $_GET['usertype'] == 1 ) {
                echo 'selected'; #if error, have
this selection be what user previously selected
            }
        }
        ?>>Admin</option>
    <option value="2" <?php
        if(isset($_GET['errors'])){
            if ( $_GET['usertype'] == 2 ) {
                echo 'selected';
            }
        }
        ?>>Teacher</option>
    <option value="3" <?php
        if(isset($_GET['errors'])){
            if ( $_GET['usertype'] == 3 ) {
                echo 'selected';
            }
        }
        ?>>Student</option>
    </select>
    <button type="submit" name="login-submit">Login</button>
</form>
</div>
</div>
</div>

<?php require './inc/footer.php'; ?>
```

[Figure 17]

registration.php

```
<?php

    require './inc/header.php';
    require './inc/navbar-admin.php';
    if (!isset($_SESSION['usertype'])) { #if the user isn't logged in, take
them to the login page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 2) { #if a teacher tried accessing the
page, redirect them to the dashboard
            header('Location: dashboard.php');
            exit();
        } else if($_SESSION['usertype'] == 3) { #if a student tried access-
ing the page, redirect them to the index
            header('Location: index.php');
            exit();
        }
    }

?>

<div class="login-container">
    <div class="login-content">
        <div class="form-img-container">
            
        </div>
        <div class="form-input-container">
            <form action="inc/registration.inc.php" method="post">
                <div class="login-title">
                    Create a new teacher account
                </div>
                <?php
                    if(isset($_GET['errors'])){ #if error code received
                        echo '<div class="h5" style="color:red;">Something
Went Wrong!</div>'; #inform the user
                    } else if(isset($_GET['success'])){ #if sucess code re-
ceived
                        echo '<div style="color:green;">Registration Suc-
cessful!</div>'; #inform the user
                    }
                >
                <input type="text" name="firstname" <?php
                    $field = 'placeholder="First Name"';
                    $style = 'class= "default"';
                    if(isset($_GET['errors'])){
                        if ( ! in_array($_GET['firstname'], array('emp-
ty','toolong','invalidchar')) ) { #if not an error message
                            $field = 'value='.$_GET['firstname']; #auto
fill the field with previous inputed value
                        } else {
                            $style = 'class= "error"';
                        }
                    }
                    echo $field.' '.$style; #if error, turn field border
to red, else keep the field as its default
                ?> maxlength="255">
                <?php
                    if(isset($_GET['errors'])){


```

```

        if($_GET['firstname'] == 'empty'){
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if($_GET['firstname'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['firstname'] == 'invalidchar'){
            echo '<div style="color:red; text-align: left;">
                    <p class="h5"> Please Match the requested format. </p>
                    <p> This field only accepts letters.
                </div>';
        }
    } #if error, inform user of the type of error
?>
<input type="text" name="lastname" <?php
    $field = 'placeholder="Last Name"';
    $style = 'class= "default"';
    if(isset($_GET['errors'])){
        if( !in_array($_GET['lastname'], array('empty','toolong','invalidchar')) ) {
            $field = 'value='.$_GET['lastname'];
        } else {
            $style = 'class= "error"';
        }
    }
    echo $field.' '.$style;
?> maxlength="255">
<?php
    if(isset($_GET['errors'])){
        if($_GET['lastname'] == 'empty'){
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if($_GET['lastname'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['lastname'] == 'invalidchar'){
            echo '<div style="color:red; text-align: left;">
                    <p class="h5"> Please Match the requested format. </p>
                    <p> This field only accepts letters.
                </div>';
        }
    }
?>
<input type="text" name="username" <?php
    $field = 'placeholder="Username"';
    $style = 'class= "default"';
    if(isset($_GET['errors'])){
        if( !in_array($_GET['username'], array('empty','toolong','invalidchar','taken')) ) {
            $field = 'value='.$_GET['username'];
        } else {
            $style = 'class= "error"';
        }
    }
    echo $field.' '.$style;

```

```

    ?> maxlength="255">
<?php
    if(isset($_GET['errors'])){
        if($_GET['username'] == 'empty'){
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if($_GET['username'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['username'] == 'invalidchar'){
            echo '<div style="color:red; text-align: left;">

```

```

        }
    ?>
    <input type="password" name="repeat-password" place-
holder="Re-type Password" <?php
        $style = 'class = "default"';
        if(isset($_GET['errors'])){
            if ( in_array($_GET['password'], array('emp-
ty','toolong','mismatch','invalidchar', '')) ) {
                $style = 'class= "error"';
            }
        }
        echo $style;
    ?> maxlength="50">
<?php
    if(isset($_GET['errors'])){
        if($_GET['password'] == 'mismatch'){
            echo '<div class="h5" style="color:red; text-
align: left;"> Passwords didn\'t match, please try again </div>';
        } else {
            echo '<div class="h5" style="color:red; text-
align: left;">This field is required.</div>';
        }
    }
    ?>
    <button type="submit" name="signup-submit">Signup</button>
</form>
</div>
</div>
</div>

<?php require './inc/footer.php'; ?>

```

[Figure 18]

dashboard.php

```
<?php
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])) { #if the user isn't logged in
        header('Location: login.php'); #take them to loggin page
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
            tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }
?>

<div class="main-container">
    <div class="content">
        <div class="dashboard-container">
            <?php
                if(isset($_GET['error'])){#if error occured, user is noti-
fied
                    echo '<div class="h5" style="color:red; text-align:cen-
ter; ">Something Went Wrong!</div>';
                }
            ?>
            <div class="row">
                <a href="teacher-profile.php" class="col-xl-4 thumbnail
dashboard-button" style="background-color: #53A1EE;">
                    <p class="dashboard-content"><i class="fas fa-
user"></i><?php if (isset($_SESSION['firstname'])){echo ' '. $_SES-
SION['firstname']. "'s";} #personalise by refrencing user's name?> Pro-
file</p>
                </a>
                <a href="student-profile.php" class="col-xl-4 thumbnail
dashboard-button" style="background-color: #EC6E62;">
                    <p class="dashboard-content"><i class="fas fa-user-
friends"></i> Student's Profile</p>
                </a>
                <a href="add-student.php" class="col-xl-4 thumbnail dash-
board-button" style="background-color: #CBD3DF;">
                    <p class="dashboard-content"><i class="fas fa-user-
plus"></i> Add Student</p>
                </a>
            </div>
            <div class="row">
                <a href="give-assignment.php" class="col-xl-4 thumbnail
dashboard-button" style="background-color: #47B0B4;">
                    <p class="dashboard-content"><i class="fas fa-
list"></i> Assessments</p>
                </a>
                <a href="view-results.php" class="col-xl-4 thumbnail dash-
board-button" style="background-color: #FFC65B;">
                    <p class="dashboard-content"><i class="fas fa-
tasks"></i> View Results</p>
```

```
        </a>
        <a href="view-class-results.php" class="col-xl-4 thumbnail
dashboard-button" style="background-color: #8A81C4;">
            <p class="dashboard-content"><i class="fas fa-us-
ers"></i> Classroom Summary</p>
        </a>
    </div>
</div>
</div>

<?php require './inc/footer.php'; ?>
```

[Figure 19]

teacher-profile.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])){ #if the user isn't logged in then
they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }

    //GETTING USER'S INFO
    $query = "SELECT * FROM teachers WHERE username=?"; #get teachers ac-
count details
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query)) {
        mysqli_stmt_bind_param($statement, 's', $_SESSION['username']);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if(!$row = mysqli_fetch_assoc($results)){ #if no results found,
send user back to dashboard with error message
            header('Location: dashboard.php?error');
            exit();
        }
    } else { #if error occurred, send user back to dashboard with error mes-
sage
        header('Location: dashboard.php?error');
        exit();
    }
?>

<div class="main-container">
    <div class="login-container">
        <div class="register-content form-input-container">
            <form action="inc/edit-profile.inc.php" method="post">
                <div class="login-title">
                    Edit your account details
                </div>
                <?php
                    if(isset($_GET['errors']) || isset($_GET['error'])){

#If errors, inform user message
                        echo '<div class="h5" style="color:red;"> Something
Went Wrong! </div>';
                    } else if(isset($_GET['success'])){ #if success, inform
user
                        echo '<div style="color:green;">Edits Success-
ful!</div>';
                    }
                ?>
```

```

<input type="text" name="firstname" <?php
    $field = 'value='.$row['firstname']; #fill field
with user account name
    $style = 'class= "default"'; #system assumes by de-
fault no problems
        if(isset($_GET['errors'])){ #in teacher attempted to
change this field before and an error occured turn the field red
            $style = 'class= "error"';
        }
        echo $field.' '.$style;
    ?> maxlength="255">
<?php
    if(isset($_GET['errors'])){ #if error did occur, deter-
mine what the error was and give the correct error message and solutions
        if($_GET['firstname'] == 'empty'){
            echo '<div class="h5" style="color:red; text-
align: left;">This field is required.</div>';
        } else if($_GET['firstname'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-
align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['firstname'] == 'invalidchar'){
            echo '<div style="color:red; text-align:
left;">
                <p class="h5"> Please Match the re-
quested format. </p>
                <p> This field only accepts letters.
            </div>';
        }
    }
?>
<input type="text" name="lastname" <?php
    $field = 'value='.$row['lastname'];
    $style = 'class= "default"';
    if(isset($_GET['errors'])){
        $style = 'class= "error"';
    }
    echo $field.' '.$style;
?> maxlength="255">
<?php
    if(isset($_GET['errors'])){
        if($_GET['lastname'] == 'empty'){
            echo '<div class="h5" style="color:red; text-
align: left;">This field is required.</div>';
        } else if($_GET['lastname'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-
align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['lastname'] == 'invalidchar'){
            echo '<div style="color:red; text-align:
left;">
                <p class="h5"> Please Match the re-
quested format. </p>
                <p> This field only accepts letters.
            </div>';
        }
    }
?>
<input type="text" name="new-username" <?php
    $field = 'value='.$row['username'];
    $style = 'class= "default"';

```

```

        if(isset($_GET['errors'])){
            $style = 'class= "error"';
        }
        echo $field.' '.$style;
    ?> maxlength="255">
<?php
    if(isset($_GET['errors'])){
        if($_GET['username'] == 'empty'){
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if($_GET['username'] == 'toolong'){
            echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['username'] == 'invalidchar'){
            echo '<div style="color:red; text-align: left;">


Please Match the requested format. </p>
<p> This field only accepts letters, numbers and underscores. </p>
</div>';
        } else if($_GET['username'] == 'taken'){
            echo '<div style="color:red; text-align: left;">


Username Taken </p>
<p> Please try something else. </p>
</div>';
        }
    }
?>
<input type="password" name="current-password" placeholder="Current Password" <?php
    $style = 'class = "default"';
    if(isset($_GET['errors'])){
        $style = 'class= "error"';
    }
    echo $style;
?> maxlength="50">
<?php
    if(isset($_GET['errors'])){
        if( $_GET['password'] == 'empty' ) {
            echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
        } else if(in_array($_GET['password'], array('incorrect', ''))){
            echo '<div class="h5" style="color:red; text-align: left;">Incorrect Password.</div>';
        }
    }
?>
<input type="password" name="new-password" placeholder="New Password" <?php
    $style = 'class = "default"';
    if(isset($_GET['errors']) && $_GET['newpassword'] != ''){
        $style = 'class= "error"';
    }
    echo $style;
?> maxlength="50">
<?php
    if(isset($_GET['errors'])){


```

```

        if( $_GET['newpassword'] == 'toolong' ) {
            echo '<div class="h5" style="color:red; text-
align: left;">This field cannot exceed 255 characters.</div>';
        } else if($_GET['newpassword'] == 'invalidchar'){
            echo '<div style="color:red; text-align:
left;">
                <p class="h5"> Please Match the re-
quested format. </p>
                <p> This field accepts letters, num-
bers, underscores and the following special characters: <br /> @ % + \` ! #
$ ^ ? : . ( ) { } [ ] ~ </p>
                <p> Please include at least 1 letter
and 1 number. </p>
            </div>';
        }
    }
?>
<input type="hidden" name="usertype" value=<?php echo
$row['usertype'] ?>>
<input type="hidden" name="old-username" value=<?php echo
$row['username'] ?>>
<input type="hidden" name="id" value=<?php echo $row['id']
?>>
<button type="submit" name="edit-submit">Edit</button>
</form>
</div>
</div>
</div>

<?php require './inc/footer.php'; ?>

```

## [Figure 20]

### student-profile.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])){ #if the user isn't logged in then
they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }

    //GETTING STUDENTS' INFO
    if(!empty($_GET['orderby'])){ #if the orderby variable isn't empty in
the url
        $order = $_GET['orderby']; #order is this variables value
    } else {
        $order = 'firstname'; #otherwise by default it is firstname
    }
    if(!empty($_GET['sort'])){ #if the sort variable isn't empty in the url
        $sort = $_GET['sort']; #sort is this variables value
    } else {
        $sort = 'asc'; #otherwise by default it is asc
    }
    if(!empty($_GET['limit'])){ #if the limit variable isn't empty in the
url
        $maxNumRows = $_GET['limit']; #limit is this variables value
    } else {
        $maxNumRows = 5; #otherwise by default it is 5
    }
    if(!empty($_GET['page'])){ #if the page variable isn't empty in the url
        $currentPage = $_GET['page']; #page is this variables value
    } else {
        $currentPage = 1; #otherwise by default it is 1
    }
    $startingRow = ($currentPage - 1) * $maxNumRows; #determine which record
to begin from, determined by what page the user is currently on
    $query = "SELECT * FROM students WHERE teacherid=? ORDER BY ".$order.''.
$.sort; #query that gets all the student records in the correct order
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query)) {
        mysqli_stmt_bind_param($statement, 'i', $_SESSION['id']);
        mysqli_stmt_execute($statement);
        $results = mysqli_stmt_get_result($statement);
        if(!$row = mysqli_fetch_assoc($results)){ #if no results found,
then send user back to dashboard with error message
            header('Location: dashboard.php?error');
            exit();
    }
}
```

```

        #if results, find the number of records to determine the number of
        pages for pagination
        $numResults = mysqli_num_rows($results);
        $numPages = ceil($numResults / $maxNumRows);
    } else {
        header('Location: dashboard.php?error'); #if an error occurred, send
        user back to dashboard with error message
        exit();
    }
?>

<div class="main-container">
    <div class="content">
        <div class="student-records-container">
            <?php
                if(isset($_GET['errors']) || isset($_GET['error'])){ #if
                errors, inform user
                    echo '<div class="h5" style="color:red;">Something Went
                Wrong!</div>';
                } else if(isset($_GET['success'])){ #if success, inform
                user
                    echo '<div style="color:green;">Edits Success-
                ful!</div>';
                }
            ?>
            <div class="student-records-content">
                <?php
                    if(mysqli_data_seek($results,$startingRow)){ #if the
                    student records can start in the stored starting row (if not send user back
                    to dashboard with error message)
                        for($i = 0; $i < $maxNumRows; $i++){ #loop throught
                        from the starting position to the number of rows to show
                            if(!$row = mysqli_fetch_assoc($results)){ #if
                            loop not over and results finish, break out the loop
                                break;
                            }
                        ?>
                        <!--form that shows each student's record-->
                        <form action="inc/edit-profile.inc.php" method="post">
                            <input type="text" name="firstname" <?php echo
                            "value='".$row["firstname"]."' ?> maxlength="255">
                            <input type="text" name="lastname" <?php echo
                            "value='".$row["lastname"]."' ?> maxlength="255">
                            <input type="text" name="new-username" <?php echo
                            "value='".$row["username"]."' ?> maxlength="255">
                            <input type="password" name="new-password" place-
                            holder="New Password" maxlength="50">
                            <input type="hidden" name="usertype" value=<?php echo
                            $row['usertype'] ?>>
                            <input type="hidden" name="old-username" value=<?php
                            echo $row['username'] ?>>
                            <input type="hidden" name="id" value=<?php echo
                            $row['id'] ?>>
                            <?php #determine the url so when student account de-
                            tails are changed, the system goes back to the users chosen order
                            if(!empty($_GET['page']) && empty($_GET['order-
                            by'])) && empty($_GET['sort']) && empty($_GET['limit'])) {
                                $value = '';
                                if(!empty($_GET['page'])){
                                    $value .= '&page=' . $_GET['page'];
                                }
                            }
                        ?>
                    
```

```

        if(!empty($_GET['orderby'])){
            $value .= '&orderby='.$_GET['orderby'];
        }
        if(!empty($_GET['sort'])){
            $value .= '&sort='.$_GET['sort'];
        }
        if(!empty($_GET['limit'])){
            $value .= '&limit='.$_GET['limit'];
        }
        echo '<input type="hidden" name="url"
value="'.$value.''; #send it as a hidden field
    }
    ?>
    <button type="submit" name="edit-submit">Edit</button>
</form>
<?php
}
} else {
    header('Location: dashboard.php?error');
    exit();
}
?>
</div>
<div class="student-records-nav">
    <div class="pagination">
        <?php
            for($i = 1; $i < $numPages + 1; $i++){ #starting from
one, to the number of pages required
                $pagination = '<a href="student-pro-
file.php?page='.$i; #build the pagination, making sure each link has the
chosen order details as well
                if(isset($_GET['orderby'])){
                    $pagination .= '&orderby='.$_GET['orderby'];
                }
                if(isset($_GET['sort'])){
                    $pagination .= '&sort='.$_GET['sort'];
                }
                if(isset($_GET['limit'])){
                    $pagination .= '&limit='.$_GET['limit'];
                }
                if(isset($_GET['page'])){
                    if($_GET['page'] == $i){
                        $pagination .= '" class = "current-page"';
#make the page number the user is on the current page
                    }
                } else if($i == 1){ #defualt when user starts
they're on page 1
                    $pagination .= '" class = "current-page"';
                }
                $pagination .= "'>' . $i . '</a>';
                echo $pagination;
            }
        ?>
    </div>
    <form action="student-profile.php" method="get">
        <fieldset>
            <label for="orderby">Order by</label>
            <select name="orderby" onchange="this.form.sub-
mit()">
                <option value="firstname">?php #label the se-
lected option as what the user is currently seeing

```

```

        if(isset($_GET['orderby'])) {
            if ( $_GET['orderby'] == 'firstname' ) {
                echo 'selected';
            }
        }
    ?>>First Name</option>
<option value="lastname"><?php
    if(isset($_GET['orderby'])) {
        if ( $_GET['orderby'] == 'lastname' ) {
            echo 'selected';
        }
    }
?>>Last Name</option>
<option value="username"><?php
    if(isset($_GET['orderby'])) {
        if ( $_GET['orderby'] == 'username' ) {
            echo 'selected';
        }
    }
?>>Username</option>
</select>
</fieldset>
<fieldset>
    <label for="sort">Sort by</label>
    <select name="sort" onchange="this.form.submit()">
        <option value="asc"><?php #label the selected
option as what the user is currently seeing
            if(isset($_GET['sort'])) {
                if ( $_GET['sort'] == 'asc' ) {
                    echo 'selected';
                }
            }
        ?>>ASC</option>
        <option value="desc"><?php
            if(isset($_GET['sort'])) {
                if ( $_GET['sort'] == 'desc' ) {
                    echo 'selected';
                }
            }
        ?>>DESC</option>
    </select>
</fieldset>
<fieldset>
    <label for="limit">Number of rows</label>
    <select name="limit" onchange="this.form.submit()">
        <option value="5"><?php #label the selected op-
tion as what the user is currently seeing
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 5 ) {
                    echo 'selected';
                }
            }
        ?>>5</option>
        <option value="10"><?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 10 ) {
                    echo 'selected';
                }
            }
        ?>>10</option>

```

```

<option value="15"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 15 ) {
            echo 'selected';
        }
    }
?>>15</option>
<option value="20"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 20 ) {
            echo 'selected';
        }
    }
?>>20</option>
<option value="25"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 25 ) {
            echo 'selected';
        }
    }
?>>25</option>
<option value="30"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 30 ) {
            echo 'selected';
        }
    }
?>>30</option>
<option value="35"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 35 ) {
            echo 'selected';
        }
    }
?>>35</option>
<option value="40"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 40 ) {
            echo 'selected';
        }
    }
?>>40</option>
<option value="45"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 45 ) {
            echo 'selected';
        }
    }
?>>45</option>
<option value="50"><?php
    if(isset($_GET['limit'])) {
        if ( $_GET['limit'] == 50 ) {
            echo 'selected';
        }
    }
?>>50</option>
</select>
</fieldset>
<noscript>

```

```
        <!--normally once the user changes the field, the  
page is auto updated but is javascript isn't on then the user can use this  
button-->  
                <button type="submit" name="order-btn">Edit  
Filters</button>  
                </noscript>  
            </form>  
        </div>  
    </div>  
</div>  
<?php require './inc/footer.php'; ?>
```

[Figure 21]

add-student.php

```
<?php
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])) { #if the user isn't logged in then
        they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
            tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }
?>

<div class="main-container">
    <div class="login-container">
        <div class="register-content form-input-container">
            <form action="inc/registration.inc.php" method="post">
                <div class="login-title">
                    Create a new student account
                </div>
                <?php
                    if(isset($_GET['errors'])){ #if errors, inform user
                        message
                        echo '<div class="h5" style="color:red;"> Something
Went Wrong! </div>';
                    } else if(isset($_GET['success'])){ #if success, inform
                        user
                        echo '<div style="color:green;">Registration Suc-
cessful!</div>';
                    }
                ?>
                <input type="text" name="firstname" <?php
                    $field = 'placeholder="First Name"'; #fill field
                    with a placeholder as default
                    $style = 'class= "default"'; #system assumes by de-
                    fault no problems
                    if(isset($_GET['errors'])){ #if teachers attempted
                        to change this field before and an error occured
                        if ( !in_array($_GET['firstname'], array('emp-
                            ty','toolong','invalidchar')) ) {
                            $field = 'value='.$_GET['firstname']; #if
                            this field didn't have the error, auto fill it with previous input
                        } else {
                            $style = 'class= "error"'; #otherwise turn
                            the field red so user know this field had a mistake
                        }
                    }
                    echo $field.' '.$style;
                ?> maxlength="255">
            <?php
```

```

        if(isset($_GET['errors'])) { #if error did occur, determine what the error was and give the correct error message and solutions
            if($_GET['firstname'] == 'empty'){
                echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
            } else if($_GET['firstname'] == 'toolong'){
                echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
            } else if($_GET['firstname'] == 'invalidchar'){
                echo '<div style="color:red; text-align:left;">

```

```

        }
        echo $field.' '.$style;
        ?> maxlength="255">
    <?php
        if(isset($_GET['errors'])){
            if($_GET['username'] == 'empty'){
                echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
            } else if($_GET['username'] == 'toolong'){
                echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 255 characters.</div>';
            } else if($_GET['username'] == 'invalidchar'){
                echo '<div style="color:red; text-align:left;">
                    <p class="h5"> Please Match the requested format. </p>
                    <p> This field only accepts letters, numbers and underscores. </p>
                    </div>';
            } else if($_GET['username'] == 'taken'){
                echo '<div style="color:red; text-align:left;">
                    <p class="h5"> Username Taken </p>
                    <p> Please try something else. </p>
                    </div>';
            }
        }
    ?>
    <input type="password" name="password" placeholder="Password" <?php
        $style = 'class = "default"';
        if(isset($_GET['errors'])){
            if( in_array($_GET['password'], array('empty','toolong','mismatch','invalidchar', '')) ) {
                $style = 'class= "error"';
            }
        }
        echo $style;
        ?> maxlength="50">
    <?php
        if(isset($_GET['errors'])){
            if( in_array($_GET['password'], array('empty', '')) ){
                echo '<div class="h5" style="color:red; text-align: left;">This field is required.</div>';
            } else if($_GET['password'] == 'toolong'){
                echo '<div class="h5" style="color:red; text-align: left;">This field cannot exceed 50 characters.</div>';
            } else if($_GET['password'] == 'invalidchar'){
                echo '<div style="color:red; text-align:left;">
                    <p class="h5"> Please Match the requested format. </p>
                    <p> This field accepts letters, numbers, underscores and the following special characters: <br /> @ % + \` ! # $ ^ ? : . ( ) { } [ ] ~ </p>
                    <p> Please include at least 1 letter and 1 number. </p>
                    </div>';
            } else if($_GET['password'] == 'mismatch'){

```

```

        echo '<div class="h5" style="color:red; text-
align: left;"> Passwords didn\'t match, please try again </div>';
    }
}
?>
<input type="password" name="repeat-password" place-
holder="Re-type Password" <?php
    $style = 'class = "default"';
    if(isset($_GET['errors'])){
        if ( in_array($_GET['password'], array('emp-
ty','toolong','mismatch','invalidchar', '')) ) {
            $style = 'class= "error"';
        }
    }
    echo $style;
?> maxlength="50">
<?php
    if(isset($_GET['errors'])){
        if($_GET['password'] == 'mismatch'){
            echo '<div class="h5" style="color:red; text-
align: left;"> Passwords didn\'t match, please try again </div>';
        } else {
            echo '<div class="h5" style="color:red; text-
align: left;">This field is required.</div>';
        }
    }
?>
<button type="submit" name="signup-submit">Add</button>
</form>
</div>
</div>
</div>

<?php require './inc/footer.php'; ?>
```

[Figure 22]

give-assignment.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])){ #if the user isn't logged in then
they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }

    //GETTING STUDENTS' INFO
    if(!empty($_GET['orderby'])){ #if the orderby variable isn't empty in
the url
        $order = $_GET['orderby']; #order is this variables value
    } else {
        $order = 'firstname'; #otherwise by default it is firstname
    }
    if(!empty($_GET['sort'])){ #if the sort variable isn't empty in the url
        $sort = $_GET['sort']; #sort is this variables value
    } else {
        $sort = 'asc'; #otherwise by default it is asc
    }
    if(!empty($_GET['limit'])){ #if the limit variable isn't empty in the
url
        $maxNumRows = $_GET['limit']; #limit is this variables value
    } else {
        $maxNumRows = 5; #otherwise by default it is 5
    }
    if(!empty($_GET['page'])){ #if the page variable isn't empty in the url
        $currentPage = $_GET['page']; #page is this variables value
    } else {
        $currentPage = 1; #otherwise by default it is 1
    }
    $startingRow = ($currentPage - 1) * $maxNumRows; #determine which record
to begin from, determined by what page the user is currently on
    $query = "SELECT * FROM students WHERE teacherid=? ORDER BY ".$order.''.
$.sort; #query that gets all the student records in the correct order
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query) ) {
        mysqli_stmt_bind_param($statement, 'i', $_SESSION['id']);
        mysqli_stmt_execute($statement);
        $studentResults = mysqli_stmt_get_result($statement);
        if(!$studentRow = mysqli_fetch_assoc($studentResults)){ #if no
results found, then send user back to dashboard with error message
            header('Location: dashboard.php?error');
            exit();
    }
}
```

```

        #if results, find the number of records to determine the number of
pages for pagination
        $numResults = mysqli_num_rows($studentResults);
        $numPages = ceil($numResults / $maxNumRows);
    } else {
        header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
        exit();
    }

//GETTING ACTIVITY INFO
$query = "SELECT * FROM activities"; #get all activity information
if(mysqli_stmt_prepare($statement, $query) ) {
    mysqli_stmt_execute($statement);
    $activityResults = mysqli_stmt_get_result($statement);
    if(!$activityRow = mysqli_fetch_assoc($activityResults)){ #if no
results found, send user back to dashboard with error message
        header('Location: index.php');
        exit();
    }
} else {
    header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
    exit();
}

//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment"; #get all assignment in-
formation
$areAssignments = true; //system assumes there are assignments in the
database
if(mysqli_stmt_prepare($statement, $query) ) {
    mysqli_stmt_execute($statement);
    $assignmentResults = mysqli_stmt_get_result($statement);
    if(!$assignmentRow = mysqli_fetch_assoc($assignmentResults)){ //if no
assignments found, then set are assignments to false
        $areAssignments = false;
    }
} else {
    header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
    exit();
}
?>

<div class="main-container">
    <div class="content">
        <div class="student-records-container">
            <?php
                if(isset($_GET['errors']) || isset($_GET['error'])){ #if
errors, inform user
                    echo '<div class="h5" style="color:red;">Something Went
Wrong!</div>';
                } else if(isset($_GET['success'])){ #if success, inform
user
                    echo '<div style="color:green;">Edits Success-
ful!</div>';
                }
            ?>
            <div class="student-records-content">
                <?php

```

```

        if(mysqli_data_seek($studentResults,$startingRow)){ #if
the student records can start in the stored starting row (if not send user
back to dashboard with error message)
            for($i = 0; $i < $maxNumRows; $i++){ #loop through
from the starting position to the number of rows to show
                if(!$studentRow = mysqli_fetch_assoc($studen-
tResults)){ #if loop not over and results finish, break out the loop
                    break;
                }
            ?>
            <!--form that shows each student's record-->
            <form action=".inc/give-assignment.inc.php" method="post">
                <input type="text" name="firstname" <?php echo
"value='". $studentRow["firstname"] ."'> maxlength="255" readonly>
                <input type="text" name="lastname" <?php echo
"value='". $studentRow["lastname"] ."'> maxlength="255" readonly>
                <input type="text" name="username" <?php echo
"value='". $studentRow["username"] ."'> maxlength="255" readonly>
                <select name="assignment">
                    <option value="none" <?php
                        $isAssigned = false; #system assumes students
have no assignments
                    if($areAssignments){
                        mysqli_data_seek($assignmentResults,0);
                    }
                    #reset pointer
                    while($assignmentRow = mysqli_fetch_as-
soc($assignmentResults)){ #loop through all assignments and see if the stu-
dent has one
                        if($assignmentRow['studentid'] == $stu-
dentRow["id"]){
                            $isAssigned = true;
                        }
                    }
                    if(!$isAssigned){ #if studnet has no assign-
ment, then none is selected
                        echo 'selected';
                    }
                    ?>> None </option>
                <?php
                    $assignment = '';
                    if($areAssignments){
                        mysqli_data_seek($assignmentResults,0);
                    }
                    #reset pointer
                    while($assignmentRow = mysqli_fetch_as-
soc($assignmentResults)){ #loop through all assignments
                        if($assignmentRow['studentid'] == $stu-
dentRow["id"]){
                            #store the assignment id of this student's
                            $assignment = $assignmentRow['ac-
tivityid'];
                        }
                    }
                }
                mysqli_data_seek($activityResults,0); #loop all
assignments name
                while($activityRow = mysqli_fetch_assoc($activi-
tyResults)) {
                    if($assignment != $activityRow['id']){
                        echo '<option value="'. $activi-
tyRow['id'] .'">' . $activityRow['name'] . '</option>';
                    }
                }
            
```

```

        } else { #make the student's active assignment the default
            echo '<option value="'. $activityRow['name'] . '</option>';
        }
    }
?>
</select>
<input type="hidden" name="studentId" value=<?php echo
$studentRow['id'] ?>
<?php #determine the url so when assignments are
changed, the system goes back to the users chosen order
if(!empty($_GET['page']) && empty($_GET['order-
by']) && empty($_GET['sort']) && empty($_GET['limit'])) {
    $value = '';
    if(!empty($_GET['page'])){
        $value .= '&page=' . $_GET['page'];
    }
    if(!empty($_GET['orderby'])){
        $value .= '&orderby=' . $_GET['orderby'];
    }
    if(!empty($_GET['sort'])){
        $value .= '&sort=' . $_GET['sort'];
    }
    if(!empty($_GET['limit'])){
        $value .= '&limit=' . $_GET['limit'];
    }
    echo '<input type="hidden" name="url"
value=' . $value . '>'; #send it as a hidden field
}
?>
<button type="submit" name="assign-submit"> Edit </but-
ton>
</form>
<?php
}
} else {
    header('Location: dashboard.php?error');
    exit();
}
?>
</div>
<div class="student-records-nav">
    <div class="pagination">
        <?php
        for($i = 1; $i < $numPages + 1; $i++){ #starting from
one, to the number of pages required
            $pagination = '<a href="give-assign-
ment.php?page=' . $i; #build the pagination, making sure each link has the
chosen order details as well
            if(isset($_GET['orderby'])){
                $pagination .= '&orderby=' . $_GET['orderby'];
            }
            if(isset($_GET['sort'])){
                $pagination .= '&sort=' . $_GET['sort'];
            }
            if(isset($_GET['limit'])){
                $pagination .= '&limit=' . $_GET['limit'];
            }
            if(isset($_GET['page'])){ #make the page number the
user is on the current page

```

```

        if($_GET['page'] == $i){
            $pagination .= '" class = "current-page"';
        }
    } else if($i == 1){ #defualt when user starts
they're on page 1
        $pagination .= '" class = "current-page"';
    }
    $pagination .= '>'.$i.'</a>';
    echo $pagination;
}
?>
</div>
<form action="give-assignment.php" method="get">
    <fieldset>
        <label for="orderby">Order by</label>
        <select name="orderby" onchange="this.form.submit()">
            <option value="firstname" <?php #label the selected option as what the user is currently seeing
                if(isset($_GET['orderby'])){
                    if ( $_GET['orderby'] == 'firstname' )
{
                        echo 'selected';
                    }
                }
            ?>>>First Name</option>
            <option value="lastname" <?php
                if(isset($_GET['orderby'])){
                    if ( $_GET['orderby'] == 'lastname' ) {
                        echo 'selected';
                    }
                }
            ?>>>Last Name</option>
            <option value="username" <?php
                if(isset($_GET['orderby'])){
                    if ( $_GET['orderby'] == 'username' ) {
                        echo 'selected';
                    }
                }
            ?>>>Username</option>
        </select>
    </fieldset>
    <fieldset>
        <label for="sort">Sort by</label>
        <select name="sort" onchange="this.form.submit()">
            <option value="asc" <?php #label the selected option as what the user is currently seeing
                if(isset($_GET['sort'])){
                    if ( $_GET['sort'] == 'asc' ) {
                        echo 'selected';
                    }
                }
            ?>>>ASC</option>
            <option value="desc" <?php
                if(isset($_GET['sort'])){
                    if ( $_GET['sort'] == 'desc' ) {
                        echo 'selected';
                    }
                }
            ?>>>DESC</option>
        </select>

```

```

</fieldset>
<fieldset>
    <label for="limit">Number of rows</label>
    <select name="limit" onchange="this.form.submit()">
        <option value="5" <?php #label the selected op-
tion as what the user is currently seeing
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 5 ) {
                    echo 'selected';
                }
            }
        ?>>5</option>
        <option value="10" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 10 ) {
                    echo 'selected';
                }
            }
        ?>>10</option>
        <option value="15" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 15 ) {
                    echo 'selected';
                }
            }
        ?>>15</option>
        <option value="20" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 20 ) {
                    echo 'selected';
                }
            }
        ?>>20</option>
        <option value="25" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 25 ) {
                    echo 'selected';
                }
            }
        ?>>25</option>
        <option value="30" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 30 ) {
                    echo 'selected';
                }
            }
        ?>>30</option>
        <option value="35" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 35 ) {
                    echo 'selected';
                }
            }
        ?>>35</option>
        <option value="40" <?php
            if(isset($_GET['limit'])) {
                if ( $_GET['limit'] == 40 ) {
                    echo 'selected';
                }
            }
        ?>>40</option>

```

```

<option value="45" <?php
    if(isset($_GET['limit'])){
        if ( $_GET['limit'] == 45 ) {
            echo 'selected';
        }
    }
?>>45</option>
<option value="50" <?php
    if(isset($_GET['limit'])){
        if ( $_GET['limit'] == 50 ) {
            echo 'selected';
        }
    }
?>>50</option>
</select>
</fieldset>
<noscript>
    <!--normally once the user changes the field, the
page is auto updated but is javascript isn't on then the user can use this
button-->
        <button type="submit" name="order-btn">Edit
Filters</button>
    </noscript>
</form>
</div>
</div>
</div>
<?php require './inc/footer.php'; ?>

```

[Figure 23]

view-results.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])){ #if the user isn't logged in then
they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }

    //GETTING STUDENTS' INFO
    if(!empty($_GET['orderby'])){ #if the orderby variable isn't empty in
the url
        $order = $_GET['orderby']; #order is this variables value
    } else {
        $order = 'firstname'; #otherwise by default it is firstname
    }
    if(!empty($_GET['sort'])){ #if the sort variable isn't empty in the url
        $sort = $_GET['sort']; #sort is this variables value
    } else {
        $sort = 'asc'; #otherwise by default it is asc
    }
    if(!empty($_GET['limit'])){ #if the limit variable isn't empty in the
url
        $maxNumRows = $_GET['limit']; #limit is this variables value
    } else {
        $maxNumRows = 5; #otherwise by default it is 5
    }
    if(!empty($_GET['page'])){ #if the page variable isn't empty in the url
        $currentPage = $_GET['page']; #page is this variables value
    } else {
        $currentPage = 1; #otherwise by default it is 1
    }
    $startingRow = ($currentPage - 1) * $maxNumRows; #determine which record
to begin from, determined by what page the user is currently on
    $query = "SELECT * FROM students WHERE teacherid=? ORDER BY ".$order.''.
$.sort; #query that gets all the student records in the correct order
    $statement = mysqli_stmt_init($connection);
    if(mysqli_stmt_prepare($statement, $query)) {
        mysqli_stmt_bind_param($statement, 'i', $_SESSION['id']);
        mysqli_stmt_execute($statement);
        $studentResults = mysqli_stmt_get_result($statement);
        if(!$studentRow = mysqli_fetch_assoc($studentResults)){ #if no
results found, then send user back to dashboard with error message
            header('Location: dashboard.php?error');
            exit();
    }
}
```

```

        #if results, find the number of records to determine the number of
pages for pagination
        $numResults = mysqli_num_rows($studentResults);
        $numPages = ceil($numResults / $maxNumRows);
    } else {
        header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
        exit();
    }

//GETTING ACTIVITY INFO
$query = "SELECT * FROM activities"; #get all activity information
if(mysqli_stmt_prepare($statement, $query) ) {
    mysqli_stmt_execute($statement);
    $activityResults = mysqli_stmt_get_result($statement);
    if(!$activityRow = mysqli_fetch_assoc($activityResults)){ #if no
results found, send user back to dashboard with error message
        header('Location: dashboard.php?error');
        exit();
    }
} else {
    header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
    exit();
}

//GETTING ASSIGNMENT INFO
$query = "SELECT * FROM activity_assignment"; #get all assignment in-
formation
if(mysqli_stmt_prepare($statement, $query) ) {
    mysqli_stmt_execute($statement);
    $assignmentResults = mysqli_stmt_get_result($statement);
} else {
    header('Location: dashboard.php?error'); #if an error occurred, send
user back to dashboard with error message
    exit();
}
?>

<div class="main-container">
    <div class="view-results-page">
        <div class="tabs">
            <button id="students" class="tab-selected"> Students </button>
            <button id="results" class="tab-unselected"> Results </button>
        </div>
        <div class="content">
            <div class="student-records-container">
                <div id="student-records">
                    <div class="student-records-content">
                        <?php
                            if(mysqli_data_seek($studentResults,$start-
ingRow)){ #if the student records can start in the stored starting row (if
not send user back to dashboard with error message)
                                for($i = 0; $i < $maxNumRows; $i++){ #loop
through from the starting position to the number of rows to show
                                    if(!$studentRow = mysqli_fetch_as-
soc($studentResults)){ #if loop not over and results finish, break out the
loop
                                        break;
                                }
                            ?>

```

```

        <!--form that shows each student's record-->
        <form action="javascript:void(0)" method="post">
            <input type="text" name="firstname" <?php echo
"value='".\$studentRow["firstname"]."'" ?> maxlength="255" readonly>
            <input type="text" name="lastname" <?php echo
"value='".\$studentRow["lastname"]."'" ?> maxlength="255" readonly>
            <input type="text" name="username" <?php echo
"value='".\$studentRow["username"]."'" ?> maxlength="255" readonly>
            <input type="text" name="assignment" <?php
                $assignment = 'No Assignments'; #by de-
                fualt system assumes no assignment
                mysqli_data_seek($assignmentResults,0);
#start from the beginning
                while($assignmentRow = mysqli_fetch_as-
soc($assignmentResults)){ #loop through all the assignments
                    if($assignmentRow['studentid'] ==
$studentRow['id'])){ #if this student has that assignment
                        mysqli_data_seek($activi-
tyResults,0); #start from the beginning
                        while($activityRow =
mysqli_fetch_assoc($activityResults)) { #find what the assignment activity
                            name is
                                if($assignmentRow['activi-
tyid'] == $activityRow['id']){
                                    $assignment = $activi-
tyRow['name'];
                                }
                            }
                        }
                    }
                echo "value='".\$assignment."'" #tell
teacher what this student's assignment is
                ?> maxlength="255" readonly>
                <button type="submit" name="view-results"<?php
if($assignment == 'No Assignments'){echo 'disabled';} ?> onclick='visuali-
sation("<?php echo \$studentRow["username"]; ?>", "individual");'> View
</button>
        </form>
        <?php
    }
} else {
    header('Location: dashboard.php?error');
    exit();
}
?>
</div>
<div class="student-records-nav">
    <div class="pagination">
        <?php
            for($i = 1; $i < $numPages + 1; $i++){ #start-
ing from one, to the number of pages required
                $pagination = '<a href="view-re-
sults.php?page='.$i; #build the pagination, making sure each link has the
chosen order details as well
                if(isset($_GET['orderby'])){
                    $pagination .= '&orderby='.$_GET['or-
derby'];
                }
                if(isset($_GET['sort'])){
                    $pagination .= '&sort='.$_GET['sort'];
                }
}

```

```

        if(isset($_GET['limit'])){
            $pagination .= '&limit=' . $_GET['lim-
it'];
        }
        if(isset($_GET['page'])){ #make the page
number the user is on the current page
            if($_GET['page'] == $i){
                $pagination .= '' class = "current-
page"';
            }
            } else if($i == 1){ #default when user
starts they're on page 1
                $pagination .= '' class = "current-
page"';
            }
            $pagination .= '>' . $i . '</a>';
            echo $pagination;
        }
    ?>
</div>
<form action="view-results.php" method="get">
    <fieldset>
        <label for="orderby">Order by</label>
        <select name="orderby" on-
change="this.form.submit()">
            <option value="firstname"><?php
                if(isset($_GET['orderby'])) { #label
the selected option as what the user is currently seeing
                    if ( $_GET['orderby'] ==
'firstname' ) {
                        echo 'selected';
                    }
                }
            ?>>First Name</option>
            <option value="lastname"><?php
                if(isset($_GET['orderby'])) {
                    if ( $_GET['orderby'] == 'last-
name' ) {
                        echo 'selected';
                    }
                }
            ?>>Last Name</option>
            <option value="username"><?php
                if(isset($_GET['orderby'])) {
                    if ( $_GET['orderby'] ==
'username' ) {
                        echo 'selected';
                    }
                }
            ?>>Username</option>
        </select>
    </fieldset>
    <fieldset>
        <label for="sort">Sort by</label>
        <select name="sort" on-
change="this.form.submit()">
            <option value="asc"><?php #label the se-
lected option as what the user is currently seeing
                if(isset($_GET['sort'])) {
                    if ( $_GET['sort'] == 'asc' ) {
                        echo 'selected';
                    }
                }
            ?>
```

```

        }
    }
?>>ASC</option>
<option value="desc"><?php
    if(isset($_GET['sort'])){
        if ( $_GET['sort'] == 'desc' )
{
            echo 'selected';
        }
    }
?>>DESC</option>
</select>
</fieldset>
<fieldset>
    <label for="limit">Number of rows</label>
    <select name="limit" on-
change="this.form.submit()">
        <option value="5"><?php #label the se-
lected option as what the user is currently seeing
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 5 ) {
                echo 'selected';
            }
        }
?>>5</option>
<option value="10"><?php
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 10 ) {
                echo 'selected';
            }
        }
?>>10</option>
<option value="15"><?php
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 15 ) {
                echo 'selected';
            }
        }
?>>15</option>
<option value="20"><?php
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 20 ) {
                echo 'selected';
            }
        }
?>>20</option>
<option value="25"><?php
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 25 ) {
                echo 'selected';
            }
        }
?>>25</option>
<option value="30"><?php
        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 30 ) {
                echo 'selected';
            }
        }
?>>30</option>
<option value="35"><?php

```

```

        if(isset($_GET['limit'])){
            if ( $_GET['limit'] == 35 ) {
                echo 'selected';
            }
        }
    ?>>35</option>
<option value="40"<?php
    if(isset($_GET['limit'])){
        if ( $_GET['limit'] == 40 ) {
            echo 'selected';
        }
    }
?>>40</option>
<option value="45"<?php
    if(isset($_GET['limit'])){
        if ( $_GET['limit'] == 45 ) {
            echo 'selected';
        }
    }
?>>45</option>
<option value="50"<?php
    if(isset($_GET['limit'])){
        if ( $_GET['limit'] == 50 ) {
            echo 'selected';
        }
    }
?>>50</option>
</select>
</fieldset>
<noscript>
    <!--normally once the user changes the
field, the page is auto updated but is javascript isn't on then the user
can use this button-->
        <button type="submit" name="order-btn">Edit
Filters</button>
    </noscript>
</form>
</div>
</div>
<!--initial message-->
<div id="message" class="hide">
    No student selected, please select a student from the
'student' tab
</div>
<div id="visualisation" class="hide">
    <div id="student-name"></div>
    <div class="results-overview">
        <div class="row">
            <div class="col-md-4">
                <div id="attempts"></div>
            </div>
            <div class="col-md-4">
                <div id="average-time-taken"></div>
            </div>
            <div class="col-md-4">
                <div id="percentage-correct"></div>
            </div>
        </div>
    </div>
    <?php for($i = 0; $i < 10; $i++) { #for all 10 ques-
tions answered in quiz, show each result?>

```

```

<div class="quiz-information">
    <div class="row">
        <div class="col-lg-6">
            <div class="question-container">
                <div class="question-title"></div>
                <div class="question-choices">
                    <div class="quiz-choice"></div>
                    <div class="quiz-choice"></div>
                    <div class="quiz-choice"></div>
                    <div class="quiz-choice"></div>
                </div>
                <div class="question-type"></div>
                <div class="question-correct"></div>
            </div>
        </div>
        <div class="col-lg-6">
            <div class="data-container">
                <div class="picked"></div>
                <div class="picked"></div>
                <div class="picked"></div>
                <div class="data-time-taken"></div>
            </div>
        </div>
    </div>
    <?php } ?>
    <div id="pie-chart-container" class="quiz-information">
        <div class="row">
            <div class="col-lg-6">
                <div id="pie-chart-correct">
                    <canvas id="piechart-good"></canvas>
                </div>
            </div>
            <div class="col-lg-6">
                <div id="pie-chart-incorrect">
                    <canvas id="piechart-bad"></canvas>
                </div>
            </div>
        </div>
    </div>
    <div id="bar-chart-container" class="quiz-information">
        <div id="bar-chart">
            <canvas id="barchart"></canvas>
        </div>
    </div>
    <!--smaller screens can't view charts well, so they get
shown this message instead-->
    <div id="chart-message" class="hide">
        Please move to a larger device to see more data on
this student.
    </div>
</div>
</div>
</div>
</div>
<?php require './inc/footer.php'; ?>

```

[Figure 24]

view-class-results.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-teacher.php';
    include './inc/banner.php';

    if (!isset($_SESSION['usertype'])){ #if the user isn't logged in then
they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 3){
            header('Location: index.php');
            exit();
        }
    }
?>

<div class="main-container">
    <div class="view-results-page">
        <div class="content">
            <!--If no results found, user is told. Message changes
depending on type of error-->
            <div id="class-message">
                No Records Stored
            </div>
            <!--visualisation container-->
            <div id="class-visualisation" class="hide">
                <div class="results-overview">
                    <div class="row">
                        <div class="col-md-6">
                            <div id="class-attempts"></div>
                        </div>
                        <div class="col-md-6">
                            <div id="class-average-time-taken"></div>
                        </div>
                    </div>
                </div>
                <div id="class-pie-chart-container" class="quiz-
information">
                    <div class="row">
                        <div class="col-lg-6">
                            <div id="class-pie-chart-correct">
                                <canvas id="class-piechart-good"></canvas>
                            </div>
                        </div>
                        <div class="col-lg-6">
                            <div id="class-pie-chart-incorrect">
                                <canvas id="class-piechart-bad"></canvas>
                            </div>
                        </div>
                    </div>
                    <div id="class-bar-chart-container" class="quiz-
information">
```

```
<div id="class-bar-chart">
    <canvas id="class-barchart"></canvas>
</div>
</div>
<!--smaller screens can't view charts well, so they get
shown this message instead--&gt;
&lt;div id="class-chart-message" class="hide"&gt;
    Please move to a larger device to see more data on this
student.
&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;/div&gt;
&lt;?php require './inc/footer.php'; ?&gt;</pre>
```

[Figure 25]

activity-page.php

```
<?php
    require './inc/dbh.inc.php';
    require './inc/header.php';
    require './inc/navbar-default.php';

    if (!isset($_SESSION['usertype'])) { #if the user isn't logged in then
        they are redirected to the loggin page
        header('Location: login.php');
        exit();
    } else {
        if($_SESSION['usertype'] == 1){ #if a admin or student account
            tried accessing then they get redirected
            header('Location: registration.php');
            exit();
        } else if($_SESSION['usertype'] == 2){
            header('Location: dashboard.php');
            exit();
        }
    }

    if(!empty($_GET['activityID'])){ #as long as the id is in the url. if
        not send user back to index
        $query = "SELECT * FROM activities WHERE id=?"; #find that activty
        corresponding to the id in the url
        $statement = mysqli_stmt_init($connection);
        if(mysqli_stmt_prepare($statement, $query)) {
            mysqli_stmt_bind_param($statement, 'i', $_GET['activityID']);
            mysqli_stmt_execute($statement);
            $results = mysqli_stmt_get_result($statement);
            if(!$row = mysqli_fetch_assoc($results)){ #if it isn't found,
                send user back to index page
                header('Location: index.php');
                exit();
            }
        } else {
            header('Location: index.php'); #if error occured, send user
            back to index page
            exit();
        }
    } else {
        header('Location: index.php');
        exit();
    }

    include './inc/banner.php';
?>

<div class="container">
    <div class="content">
        <div class="activity-container">
            <!-- tell user that they need javascript for this part of the
            application -->
            <noscript>
                <p> You need javascript enabled to run this app. </p>
            </noscript>
            <div id="start-page">
                <div class="top-bar"></div>
                <div class="startpage">
```

```

        
            <!-- send this activity id to the activity script to use
to store results to database -->
            <button id="startpage-btn" onclick="titleScreen(<?php
echo $_GET['activityID'] ?>)"> Start </button>
        </div>
        <div class="bottom-bar"></div>
    </div>
    <div id="title-page" class="hide">
        <div class="top-bar"></div>
        <div class="titlescreen-page">
            
            <div class="bear-icon-title"></div>
            <div class="titlescreen-title">Professor Paddington:
<br /> Quiz-a-Thon</div>
            </div>
            <button onmouseenter="synth.cancel(); textToSpeech('Play')"
id="title-btn"> <i class="fas fa-play"></i> </button>
            <div class="bottom-bar"></div>
        </div>
        <div id="activity" class="hide">
            <div class="top-bar">
                <button onmouseenter="synth.cancel(); textToSpeech('Ex-
it')" id="home-btn"> <i class="fas fa-home"></i> </button>
                <button onmouseenter="synth.cancel(); textToSpeech('Re-
peat')" id="repeat-btn"> <i class="fas fa-redo-alt"></i> </button>
                <button onmouseenter="synth.cancel();
textToSpeech('Sound Off')" id="mute-btn"> <i id="mute-btn-icon" class="fas
fa-volume-up"></i> </button>
            </div>
            <div id="question" class="h1"></div>
            <div id="bear-icon-activity"></div>
            <div id="choices" class="choices">
                <button class="choice1"></button>
                <button class="choice2"></button>
                <button class="choice3"></button>
                <button class="choice4"></button>
            </div>
            <div class="bottom-bar"></div>
        </div>
        <div id="transition" class="hide">
            <div class="top-bar"></div>
            <div class="transition-page">
                
                <div id="bear-icon-transition"></div>
                <div id="transition-message"></div>
            </div>
            <div class="bottom-bar">
                <button id="nextBtn"> Next Question </button>
            </div>
        </div>
        <div id="end-page" class="hide">
            <div class="top-bar"></div>
            <div class="end-page">
                <div id="end-message"></div>
                
                <div class="bear-icon-end"></div>

```

```

        <div class="results">
            <div id="results-container">
                <div class="row">
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-check-circle result-
good"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-check-circle result-
good"></i>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-check-circle result-
good"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-check-circle result-
good"></i>
                    </div>
                    <div class="col-xs-4 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                </div>
                <div class="row">
                    <div class="col-xs-12 result-icon">
                        <i class="fas fa-times-circle result-
bad"></i>
                    </div>
                </div>
            </div>
            <div class="bottom-bar">
                <button id="endBtn"> Return to homepage </button>
            </div>
        </div>
    </div>
</div>

```

```
<?php require './inc/footer.php'; ?>
```

## [Figure 26]

### activity-script.js

```
//upon page load, initialise important variables
window.addEventListener('load', init);

//question that will be used within the quiz (researched existing KS1 exams to get the same style and content)
const quiz = [
    {
        visualQuestion: 'B _ N _ N _', //question shown on screen
        audibleQuestion: 'What letter fits to complete the word, Banana',
        //question spoken to the user
        options: [ //option they can pick
            {choice: 'A', correct: true}, //this is the correct answer
            {choice: 'N', correct: false},
            {choice: 'B', correct: false},
            {choice: 'Y', correct: false}
        ],
        type: 'Spelling' //this is the type of question this is
    },
    {
        visualQuestion: 'A _ _ L E',
        audibleQuestion: 'What letter fits to complete the word, Apple',
        options: [
            {choice: 'P', correct: true},
            {choice: 'N', correct: false},
            {choice: 'B', correct: false},
            {choice: 'Y', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'I love my older ____',
        audibleQuestion: 'Choose the correct spelling, I love my older brother',
        options: [
            {choice: 'Brother', correct: true},
            {choice: 'Brudduh', correct: false},
            {choice: 'Bruvver', correct: false},
            {choice: 'Brovor', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'Remember to _____ off the light',
        audibleQuestion: 'Choose the correct spelling, Remember to switch off the light',
        options: [
            {choice: 'Switch', correct: true},
            {choice: 'Sweetch', correct: false},
            {choice: 'Swich', correct: false},
            {choice: 'Switche', correct: false}
        ],
        type: 'Spelling'
    },
    {
        visualQuestion: 'In maths, we learnt what a _____ is',
        audibleQuestion: 'Choose the correct spelling, In maths, we learnt what a fraction is',
        options: [
            {choice: 'Fraction', correct: true},

```

```

        {choice: 'Fraktion', correct: false},
        {choice: 'Fuhracksion', correct: false},
        {choice: 'Frahction', correct: false}
    ],
    type: 'Spelling'
},
{
    visualQuestion: 'Why did Jasmine paint the top of the shed?',
    audibleQuestion: 'Jasmine and her younger brother wanted to paint the shed. As Jasmine fetched the ladder she said "These steps are a bit wobbly, I\\'ll do the roof". She placed the ladder down and began painting the roof purple, her favourite colour.',
    options: [
        {choice: 'The steps were dangerous', correct: true},
        {choice: 'She was the tallest', correct: false},
        {choice: 'She was the oldest', correct: false},
        {choice: 'The shed roof was her favourite colour', correct: false}
    ],
    type: 'Understanding'
},
{
    visualQuestion: 'Why was Jamila tired when she got to school?',
    audibleQuestion: 'Jamila was on her way to school but missed her bus! Oh no, now she has to walk all the way there, what a pitty. When she arrived at school she was all tired out!',
    options: [
        {choice: 'She missed her bus and had to walk', correct: true},
        {choice: 'She decided to run to school', correct: false},
        {choice: 'Her school was very far away', correct: false},
        {choice: 'School is so boring', correct: false}
    ],
    type: 'Understanding'
},
{
    visualQuestion: 'How long have ducks been living as pets for?',
    audibleQuestion: 'Ducks are friendly creatures and have been living as pets and farm animals for more than 500 years! Ducks can live between 5 to 10 years in the wild and 8+ years in captivity.',
    options: [
        {choice: '500 years', correct: true},
        {choice: '5 years', correct: false},
        {choice: '10 years', correct: false},
        {choice: '8 years', correct: false}
    ],
    type: 'Understanding'
},
{
    visualQuestion: 'How do Kangaroos move?',
    audibleQuestion: 'Kangaroos use their small but powerful legs to hop around everywhere and use their strong tails for balance.',
    options: [
        {choice: 'Use their legs to jump', correct: true},
        {choice: 'Use their tails to jump', correct: false},
        {choice: 'They can\\'t their legs are too small', correct: false},
        {choice: 'They walk just like us', correct: false}
    ],
    type: 'Understanding'
},
{

```

```

        visualQuestion: 'Why did Micheal drop his tray?',
        audibleQuestion: 'Micheal got his lunch from the dinner line and
was trying to decide on which table to eat at. He made his way to the table
but he didn\'t see the wet floor sign and slipped dropping his tray and all
his food on the ground',
        options: [
            {choice: 'The floor was wet', correct: true},
            {choice: 'The food was too hot', correct: false},
            {choice: 'He tripped on a sign', correct: false},
            {choice: 'He didn\'t like the food', correct: false}
        ],
        type: 'Understanding'
    },
    {
        visualQuestion: 'the boys raced to the park.',
        audibleQuestion: 'What is needed in this sentence, the boys raced
to the park',
        options: [
            {choice: 'a capital letter', correct: true},
            {choice: 'a question mark', correct: false},
            {choice: 'a comma', correct: false},
            {choice: 'an apostrophe', correct: false}
        ],
        type: 'Punctuation'
    },
    {
        visualQuestion: 'Are we there yet',
        audibleQuestion: 'What is needed in this sentence, Are we there
yet',
        options: [
            {choice: 'a question mark', correct: true},
            {choice: 'a capital letter', correct: false},
            {choice: 'a comma', correct: false},
            {choice: 'an apostrophe', correct: false}
        ],
        type: 'Punctuation'
    },
    {
        visualQuestion: 'my friend _____ just loves to dance',
        audibleQuestion: 'Where does the capital letter go? my friend James
just loves to dance',
        options: [
            {choice: 'James', correct: true},
            {choice: 'JAMES', correct: false},
            {choice: 'jaMes', correct: false},
            {choice: 'jAmes', correct: false}
        ],
        type: 'Punctuation'
    },
    {
        visualQuestion: 'Which sentence is a question?',
        audibleQuestion: 'Which sentence below is a question?',
        options: [
            {choice: 'Where is my puzzle', correct: true},
            {choice: 'This puzzle is hard', correct: false},
            {choice: 'I hate puzzles', correct: false},
            {choice: 'That puzzle is mine', correct: false}
        ],
        type: 'Punctuation'
    },
    {

```

```

        visualQuestion: 'Simon\'s cat is so cute',
        audableQuestion: 'Choose the correct punctuation, Simon\'s cat is
so cute!',,
        options: [
            {choice: '!', correct: true},
            {choice: '?', correct: false},
            {choice: ',', correct: false},
            {choice: '*', correct: false}
        ],
        type: 'Punctuation'
    },
    {
        visualQuestion: 'We will make it on time ____ we leave right now',
        audableQuestion: 'Choose the correct word to complete the sen-
tence',,
        options: [
            {choice: 'if', correct: true},
            {choice: 'when', correct: false},
            {choice: 'where', correct: false},
            {choice: 'so', correct: false}
        ],
        type: 'Grammar'
    },
    {
        visualQuestion: 'Jason\'s clothes were _____ after football prac-
tice',,
        audableQuestion: 'Choose the correct word to complete the sen-
tence',,
        options: [
            {choice: 'Dirty', correct: true},
            {choice: 'Dirted', correct: false},
            {choice: 'Dirt', correct: false},
            {choice: 'Dirty', correct: false}
        ],
        type: 'Grammar'
    },
    {
        visualQuestion: 'Zahir \'run\' to his house because it was rain-
ing',,
        audableQuestion: 'Choose the correct word to change \'run\' to past
tense',,
        options: [
            {choice: 'Ran', correct: true},
            {choice: 'Runned', correct: false},
            {choice: 'Running', correct: false},
            {choice: 'Runner', correct: false}
        ],
        type: 'Grammar'
    },
    {
        visualQuestion: 'My friend, Mayesha, is a cool girl',
        audableQuestion: 'Which word from this sentence is the adjective,
My friend, Mayesha, is a cool girl',,
        options: [
            {choice: 'cool', correct: true},
            {choice: 'Mayesha', correct: false},
            {choice: 'friend', correct: false},
            {choice: 'girl', correct: false}
        ],
        type: 'Grammar'
    },

```

```

    {
        visualQuestion: 'Milo lightly tapped me on the shoulder',
        audableQuestion: 'What type of word is \'lightly\' in the sentence
Milo lightly tapped me on the shoulder',
        options: [
            {choice: 'Adverb', correct: true},
            {choice: 'Adjective', correct: false},
            {choice: 'Noun', correct: false},
            {choice: 'Verb', correct: false}
        ],
        type: 'Grammar'
    }
};

let questions = []; //store results that would be sent to database
let maxNumQuestions = 10; //only show 10 questions to the student
let maxAttempts = 3; //at max, each student is given 3 goes per question
let overallTimeTaken = 0; //store overall time taken to complete the quiz,
in seconds
let numQuestionCorrect = 0; //store how many questions the student got correct
let currentAttempts; //used to determine how many attempts the student is currently on
let startTime, endTime, overallStartTime, overallEndTime, activityid;
//more statistics to store in database

//setting the question array up so that it's ready to take in metrics
for(let i = 0; i < maxNumQuestions; i++){
    let input = {
        correct: undefined, //if student got the question wrong
        attempts: undefined, //how many attempts at the question they took
        timeTaken: undefined, //how long they took on this question
        questionType: undefined, //what the question's type is
        questionTitle: undefined, //question's title
        correctChoice: undefined, //correct answer
        choicesList: [], //all options given to the student
        choicesPicked: [] //what choices they picked
    }
    questions.push(input);
}

let currentQuestion, shuffledQuestions; //used to determine which questions get shown to the student
let speech = new SpeechSynthesisUtterance(); //initialising Speech Synthesis

//containers and button used within quiz page
const startContainer = document.getElementById('title-page');
const resultsContainer = document.getElementById('results-container');
const startBtn = document.getElementById('title-btn');
const quizContainer = document.getElementById('activity');
const questionContainer = document.getElementById('question');
const choicesContainer = document.getElementById('choices');
const endContainer = document.getElementById('end-page');
const bearIcon = document.getElementById('bear-icon-activity');
const bearIconTrans = document.getElementById('bear-icon-transition');
const synth = window.speechSynthesis;
const homeBtn = document.getElementById('home-btn');
const endBtn = document.getElementById('endBtn');
const muteBtn = document.getElementById('mute-btn');
const muteBtnIcon = document.getElementById('mute-btn-icon');

```

```

const transitionContainer = document.getElementById('transition');
const repeatBtn = document.getElementById('repeat-btn');
const nextBtn = document.getElementById('nextBtn');
const activatePage = document.getElementById('start-page');
const transMessage = document.getElementById('transition-message');
const endMessage = document.getElementById('end-message');

//sound files
const titleScreenSound = new Audio("./audio/titlescreen.mp3");
const quizScreenSound = new Audio("./audio/quiz.mp3");
const transitionScreenSoundGood = new Audio("./audio/transition-correct.mp3");
const transitionScreenSoundBad = new Audio("./audio/transition-wrong.mp3");
const sucessSound = new Audio("./audio/success.wav");
const failSound = new Audio("./audio/fail.wav");
const endScreenGood = new Audio("./audio/endscreen-welldone.mp3");
const endScreenBad = new Audio("./audio/endscreen-tryagain.mp3");

let voices = synth.getVoices();
let mute = false; //used to mute application, by default sound is playing
window.addEventListener("beforeunload", synth.cancel()); //if the user
leaves the page and sound is playing, cancel it so the sound doesn't bleed
over

//takes in a string and speaks it as long as the sound isn't muted
function textToSpeech(e) {
    if('speechSynthesis' in window && !mute){
        synth.cancel();
        speech.text = e;
        synth.speak(speech);
    }
}

//loading voices, it is asynchronous so the function is called over and over
until voice that is needed is loaded
function loadVoices() {
    voices = synth.getVoices();
    if (voices.length !== 0) {
        if(voices[5] !== undefined) { //this is the choosen voice for the
application
            speech.voice = voices[5];
        } else {
            speech.voice = voices[0]; //non chrome browser sometimes don't
support it so chose a voice they do support
        }
    } else {
        loadVoices();
    }
}

//when page is first loaded, load the voices and set their critera up
//set the music critera up as well and set all the music to loop
function init () {
    if('speechSynthesis' in window){
        loadVoices();
    }
    speech.rate = 0.9;
    speech.pitch = 1;
    endScreenGood.volume = 0.2;
    endScreenBad.volume = 0.2;
    titleScreenSound.volume = 0.4;
}

```

```

quizScreenSound.volume = 0.2;
transitionScreenSoundGood.volume = 0.2;
transitionScreenSoundBad.volume = 0.2;
titleScreenSound.loop = true;
quizScreenSound.loop = true;
transitionScreenSoundGood.loop = true;
transitionScreenSoundBad.loop = true;
endScreenBad.loop = true;
endScreenGood.loop = true;
}

//title screen for the application
function titleScreen (e) {
    activityid = e; //grap the id for this activity, will be used to store
into database
    if(!mute){ //if not muted
        titleScreenSound.currentTime = 0; //start music from the beginning
        titleScreenSound.play(); //play the music
    }
    transitionScreenSoundGood.pause(); //pause any exisitng music that
shouldn't play on this page
    transitionScreenSoundBad.pause(); //pause any exisitng music that
shouldn't play on this page
    quizScreenSound.pause(); //pause any exisitng music that shouldn't play
on this page
    endScreenBad.pause(); //pause any exisitng music that shouldn't play on
this page
    endScreenGood.pause(); //pause any exisitng music that shouldn't play
on this page
    startContainer.classList.remove('hide'); //show the start container
    activatePage.classList.add('hide'); //hide all others
    quizContainer.classList.add('hide'); //hide all others
    endContainer.classList.add('hide'); //hide all others
    currentAttempts = 0; //attempts start at 0
    startBtn.addEventListener('click',startQuiz); //when user clicks the
start button, get ready to start the quiz
}

//starting the quiz
function startQuiz () {
    startContainer.classList.add('hide'); //hide containers not needed
    endContainer.classList.add('hide'); //hide containers not needed
    quizContainer.classList.remove('hide'); //show the quiz container
    shuffledQuestions = quiz.sort(function(){return 0.5 - Math.random()});
//pseudo randomly shuffle the questions in the quiz array
    currentQuestion = 0; //questions start from index 0
    overallStartTime = new Date(); //start recording overal time from now
    displayQuiz(shuffledQuestions[currentQuestion]); //show the first ques-
tion to the student
}

function displayQuiz (e) {
    questions[currentQuestion].questionTitle = shuffledQuestions[cur-
rentQuestion].visualQuestion; //record the questions title for storage
    questions[currentQuestion].questionType = shuffledQuestions[cur-
rentQuestion].type; //record the questions type for storage
    startTime = new Date(); //start recording the student's time on this
particular question
    titleScreenSound.pause(); //pause the music not needed on this screen
    transitionScreenSoundGood.pause(); //pause the music not needed on this
screen
}

```

```

transitionScreenSoundBad.pause(); //pause the music not needed on this
screen
if(!mute){ //if not muted
    quizScreenSound.currentTime = 0; //start music from the beginning
    quizScreenSound.play(); //play the music
}
repeatBtn.addEventListener('click',repeat); //if the repeat button is
clicked, repeat the question for the user
muteBtn.addEventListener('click',toggleMute); //if the mute button is
clicked, toggle sound between on and off
homeBtn.addEventListener('click', function(){location.reload()}); //if
the home button is clicked, refresh page
while (choicesContainer.firstChild) { //remove all existing choices to
make room for this new question's one
    choicesContainer.removeChild(choicesContainer.firstChild);
}
questionContainer.innerText = e.visualQuestion; //display the visual
question on screen
repeat(); //speak the question to the user
shuffledAnswers = e.options.sort(function(){return 0.5 - Math.ran-
dom()}); //pseudo randomly shuffle the choices
let i = 1; //each choice class follows the naming convention of
"choice" followed by a number, this keeps track of it
shuffledAnswers.forEach(function(option) { //for every option this
question has
    const button = document.createElement('button'); //create a button
    button.innerText = option.choice; //display a choice for that but-
ton
    questions[currentQuestion].choicesList.push(option.choice); //store
    the choice into the question array to store in database
    let btnClass = 'choice' + i; //add a button class to it (i gets in-
cremented so first button gets choice1, second choice2 etc)
    button.classList.add(btnClass);
    if(option.correct){ //if this button is correct, add the correct
    dataset to it so the system knows
        button.dataset.correct = option.correct;
        questions[currentQuestion].correctChoice = option.choice;
    //store that this choice was the correct one in the question array to store
    in database later
    }
    button.addEventListener('click', selectAnswer); //if this button
    get's clicked, do some checks
    choicesContainer.appendChild(button)
    i++;
})
}

//if an option get's selected
function selectAnswer (e) {
    currentAttempts++; //increment attempt by one as student used an at-
    tempt up
    const selected = e.target;
    questions[currentQuestion].choicesPicked.push(selected.innerHTML);
    //store this choice as a choice the student clicked to be stored in the da-
    tabase later
    if(selected.dataset.correct){ //if this option was the correct answer
        if(!mute){ //if not muted
            sucessSound.currentTime = 0; //reset the success sound
            sucessSound.play(); //play the success soundbite
        }
    }
}

```

```

        questions[currentQuestion].correct = 1; //store in the qustions ar-
ray that the student got this question right
        questions[currentQuestion].attempts = currentAttempts; //store the
amount of attempts the student took on this question
        endTime = new Date(); //end recording the time as the studnent com-
pleted the question
        questions[currentQuestion].timeTaken = Math.ceil((endTime - start-
Time) / 1000.0); //store time taken in seconds
        transitionScreen(); //move students to the tranision screen
    } else { //if this option was incorrect
        if(!mute){ //if not muted
            failSound.currentTime = 0; //reset the fail sound
            failSound.play(); //play it to the user
        }
        bearIcon.style.backgroundImage = "url(img/bear-sad.png)"; //turn
the bear icon sad
        textToSpeech('Almost had it, Try again!'); //reassure the student
they could try again
        setTimeout(resetBear, 1500); //resut the bear to happy after 1.5
seconds
        if(currentAttempts == maxAttempts) { //if user used all their allo-
cated attempts
            questions[currentQuestion].correct = 0; //store that the user
couldn't get the question right
            questions[currentQuestion].attempts = currentAttempts; //store
how many attempts they too (here will always be 3 since that's the current
max allowed)
            endTime = new Date(); //end recording the time as the question
is complete
            questions[currentQuestion].timeTaken = Math.ceil((endTime -
startTime) / 1000.0); //store time taken in seconds
            transitionScreen(); //move students to the tranision screen
        }
    }
}

//tranisition screen before questions
function transitionScreen() {
    currentAttempts = 0; //reset the user's available attempts
    quizScreenSound.pause(); //pause the music not needed on this screen
    if(!mute){ //if not muted
        if(questions[currentQuestion].correct == 1){ //if user got the
question correct
            transitionScreenSoundGood.currentTime = 0; //reset the success
music
            transitionScreenSoundGood.play(); //play the success music
        } else { //if user got the question wrong
            transitionScreenSoundBad.currentTime = 0; //reset the failure
music
            transitionScreenSoundBad.play(); //play the failure music
        }
    }
    quizContainer.classList.add('hide'); //hide containers not needed
    transitionContainer.classList.remove('hide'); //show the transition
screen
    resetBear(); //reset the bear to default state
    if(questions[currentQuestion].correct == 1) { //if user got the ques-
tion correct
        bearIconTrans.style.backgroundImage = "url(img/bear-happy.png)";
//show he bear happy
        transMessage.innerHTML = 'Well Done!' //congratulate the student
    }
}

```

```

        textToSpeech('Well Done, that was the correct answer'); //congratulate the student
    } else { //if user got the question wrong
        bearIconTrans.style.backgroundImage = "url(img/bear-sad.png)";
//show he bear sad
        transMessage.innerHTML = 'OOPS' //consolidate the student
        textToSpeech('OOPS, that wasn\'t right'); //consolidate the student
    }
    nextBtn.addEventListener('click',nextQuestion); //when user clicked
next question button, move on
}

function nextQuestion () {
    currentQuestion++; //move to the next question
    if(currentQuestion >= maxNumQuestions){ //ensure the value isn't the
end
        overallEndTime = new Date(); //if end, stop recording overal time
        end(); //end the quiz
    } else { //if more questions left
        transitionContainer.classList.add('hide'); //hide the transition
screen
        quizContainer.classList.remove('hide'); //show quiz
        displayQuiz(shuffledQuestions[currentQuestion]); //display the next
question
    }
}

function resetBear() {
    bearIcon.style.backgroundImage = "url(img/bear-happy.png)"; //resets
the beat back to happy
}

//end of the quiz
function end () {
    overallTimeTaken = Math.ceil((overallEndTime - overallStartTime) /
1000.0); //store time taken in
    for(let i = 0; i < questions.length; i++){ //determine how many ques-
tions the student got corrcet
        if(questions[i].correct){
            numQuestionCorrect++;
        }
    }
    transitionScreenSoundGood.pause(); //pause the music not needed on this
screen
    transitionScreenSoundBad.pause(); //pause the music not needed on this
screen
    endContainer.classList.remove('hide'); //show the end container screen
    transitionContainer.classList.add('hide'); //hide all others
    quizContainer.classList.add('hide'); //hide all others
    if(numQuestionCorrect >= 5){ //if user got above half right, they did
well
        if(!mute){ //if not muted
            endScreenGood.currentTime = 0; //reset the success music
            endScreenGood.play(); //play the success music
        }
        endMessage.innerHTML = 'Well Done!'; //congratulate the student
        textToSpeech('Well Done! That was amazing!'); //congratulate the
student
    } else {
        if(!mute){ //if not muted
            endScreenBad.currentTime = 0; //reset the failure music
        }
    }
}

```

```

        endScreenBad.play(); //play the failure music
    }
    endMessage.innerHTML = 'OOPS'; //consolidate the student
    textToSpeech('Good Attempt, maybe you should try it again?');
//consolidate the student
}
for(let i = 0; i < questions.length; i++) { //loop through all questions, showing a tick for those correct and a cross for those incorrect
    let currentIcon = resultsContainer.getElementsByClassName("result-icon")[i];
    if(questions[i].correct){
        currentIcon.innerHTML = '<i class="fas fa-check-circle result-good"></i>';
    } else {
        currentIcon.innerHTML = '<i class="fas fa-times-circle result-bad"></i>';
    }
    //send data to a php file for storing
    $.ajax({ //post request to store data to database
        type: 'POST',
        url: 'inc/results-submit.inc.php',
        data: {data: questions, time : overallTimeTaken, activityid: activityid, submit : 'submit-results'},
        success: function(data) {
            alert(data); //upon success alert to the user the given success or error message
        }
    });
    endBtn.addEventListener('click', function(){location.reload()}); //when the end button is clicked, reload the page to refresh everything
}

//if mute button is clicked
function toggleMute() {
    mute = !mute; //toggle between on and off
    if(mute){ //if sound off
        quizScreenSound.pause(); //stop playing music
    } else { //if sound on
        quizScreenSound.play(); //play the music
    }
    //swap icons depending on function
    muteBtnIcon.classList.toggle("fa-volume-up");
    muteBtnIcon.classList.toggle("fa-volume-mute");
}

//if repeat question button is clicked
function repeat() {
    let question = shuffledQuestions[currentQuestion]; //find what the current question is
    textToSpeech(question.audableQuestion); //speak it's audio string
}

```

## [Figure 27]

### view-results-script.js

```
//containers and button used within results page
const studentContainer = document.getElementById('student-records');
const visualisationContainer = document.getElementById('visualisation');
const messageContainer = document.getElementById('message');
const resultContainer = document.getElementById('results-text');
const studentbtn = document.getElementById('students');
const visualisationbtn = document.getElementById('results');

const quizQuestionContainers = document.getElementsByClassName('quiz-information');
const questionTitle = document.getElementsByClassName('question-title');
const questionChoices = document.getElementsByClassName('question-choices');
const questionType = document.getElementsByClassName('question-type');
const questionCorrectAnswer = document.getElementsByClassName('question-correct');
const studentName = document.getElementById('student-name');
const studentAttempts = document.getElementById('attempts');
const studentTime = document.getElementById('average-time-taken');
const studentScore = document.getElementById('percentage-correct');
const dataContainer = document.getElementsByClassName('data-container');
const questionTime = document.getElementsByClassName('data-time-taken');

//variables used for chart creation
const piechartContainer = document.getElementById('pie-chart-container');
const barchartContainer = document.getElementById('bar-chart-container');
const piechartGood = document.getElementById('piechart-good');
const piechartBad = document.getElementById('piechart-bad');
const barchart = document.getElementById('barchart');
const chartMessage = document.getElementById('chart-message');

let dataLoaded = false;
let newStudent = false;
let positivePieChart, negativePieChart, timeBarChart;

//event listeners for when buttons are clicked
studentbtn.addEventListener('click', displayStudent);
visualisationbtn.addEventListener('click', displayVisualisation);
window.addEventListener('resize', windowSizeCheck);

//chart looks unappealing on small screens, this checks to see if the
screen is too small, if so a warning message is revealed
function windowSizeCheck() {
    let w = window.innerWidth;
    if(w < 700){
        piechartContainer.classList.add('hide');
        barchartContainer.classList.add('hide');
        chartMessage.classList.remove('hide');
    } else {
        piechartContainer.classList.remove('hide');
        barchartContainer.classList.remove('hide');
        chartMessage.classList.add('hide');
    }
}

//When student tab clicked, this function sets the tab as active
function displayStudent () {
    messageContainer.classList.add('hide');
    visualisationContainer.classList.add('hide');
```

```

visualisationbtn.classList.add('tab-unselected');
visualisationbtn.classList.remove('tab-selected');
studentContainer.classList.remove('hide');
studentbtn.classList.remove('tab-unselected');
studentbtn.classList.add('tab-selected');

}

//When results tab clicked, this function sets the tab as active
function displayVisualisation () {
    studentContainer.classList.add('hide');
    studentbtn.classList.add('tab-unselected');
    studentbtn.classList.remove('tab-selected');
    if(dataLoaded) {
        visualisationContainer.classList.remove('hide');
    } else {
        messageContainer.classList.remove('hide');
    }
    visualisationbtn.classList.remove('tab-unselected');
    visualisationbtn.classList.add('tab-selected');
}

//takes metrics from database and turns them into visual information
function visualisation(username, typeOfData) {
    $.ajax({ //post request to get data from database
        type: 'POST',
        url: 'inc/view-results.inc.php',
        data: {username: username, typeOfData: typeOfData, submit : 'submit'},
        beforeSend: function(){ //before requesting data, the visualisation
page is shown
            dataLoaded = false;
            windowSizeCheck();
            displayVisualisation();
        },
        success: function(data) { //once data is received
            visualisationContainer.classList.add('hide');
            messageContainer.classList.remove('hide');
            if(!isJsonString(data)){ //checks to see if the data gotten is
json string or not, if not then an error message is shown
                messageContainer.innerHTML = data;
            } else { //otherwise the data is handled and converted to in-
formation
                dataLoaded = true;
                visualisationContainer.classList.remove('hide');
                messageContainer.classList.add('hide');
                storedData = JSON.parse(data);
                studentName.innerHTML = username; //student name is shown
                studentAttempts.innerHTML = storedData[(storedData.length -
2)] + '<br> attempt(s)'; //students attempts shown
                let minutes = Math.floor(storedData[(storedData.length -
1)] / 60);
                let seconds = storedData[(storedData.length - 1)] - minutes
* 60;
                let value = '';
                if(minutes < 10) {
                    value += '0';
                }
                value += minutes + ':';
                if(seconds < 10) {
                    value += '0';
                }
            }
        }
    });
}

```

```

        value += seconds;
        studentTime.innerHTML = value + '<br> to complete'; //time
is converted to minuets and seconds before being shown
        let percentage = 0;
        for(let i = 0; i < (storedData.length - 2); i++){
            if(storedData[i].correct == 1){
                percentage++;
            }
        }
        studentScore.innerHTML = percentage + '/10 <br> correct';
//student results out of 10 is calculated and shown
        for(let i = 0; i < (storedData.length - 2); i++){ //loop
through all questions in the database (since last two items in array are
time and attempts)
            quizQuestionContainers[i].classList.remove('correct-
container');
            quizQuestionContainers[i].classList.remove('incorrect-
container');
            if(storedData[i].correct == 1){ //if question was cor-
rect, the green container is used
                quizQuestionContainers[i].classList.add('correct-
container');
            } else { //if question was incorrect, the red container
is used
                quizQuestionContainers[i].classList.add('incorrect-
container');
            }
            questionTitle[i].innerHTML = (i+1) + ' ' + stored-
Data[i].title; //title of question is shown
            let choices = storedData[i].choices.split(','); //each
question choices are shown
            questionChoices[i].getElementsByClassName('quiz-
choice')[0].innerHTML = '<i class="fas fa-square questions-choice1"></i> ' +
+ choices[0];
            questionChoices[i].getElementsByClassName('quiz-
choice')[1].innerHTML = '<i class="fas fa-square questions-choice2"></i> ' +
+ choices[1];
            questionChoices[i].getElementsByClassName('quiz-
choice')[2].innerHTML = '<i class="fas fa-square questions-choice3"></i> ' +
+ choices[2];
            questionChoices[i].getElementsByClassName('quiz-
choice')[3].innerHTML = '<i class="fas fa-square questions-choice4"></i> ' +
+ choices[3];
            questionType[i].innerHTML = 'Question Type: ' + stored-
Data[i].type; //question type is shown
            questionCorrectAnswer[i].innerHTML = 'Correct Answer: ' +
+ storedData[i].answer; //the correct answer is shown
            let picked = storedData[i].picked.split(',');
            let choicesLabel = ['First Choice:', 'Second
Choice:', 'Third Choice:'];
            let pickedCount = 0;
            for(let j = 0; j < picked.length; j++){ //each choices
picked for the question is shown
                dataContainer[i].getElementsByClass-
Name('picked')[j].innerHTML = choicesLabel[j] + ' ' + picked[j];
                pickedCount = j + 1;
            }
            for(let j = pickedCount; j < 3; j++){ //if not all
three choices picked needed, refresh the unused slots
                dataContainer[i].getElementsByClass-
Name('picked')[j].innerHTML = '';

```

```

        }
        minutes = Math.floor(storedData[i].time / 60);
        seconds = storedData[i].time - minutes * 60;
        value = 'Time Taken: ';
        if(minutes < 10) {
            value += '0';
        }
        value += minutes + ':';
        if(seconds < 10) {
            value += '0';
        }
        value += seconds;
        questionTime[i].innerHTML = value; //time is converted
to minuets and seconds before being shown
    }
    piechartContainer.classList.add('pie-graph-container');
    //preparing pie chart to show question type of all ques-
tions answered correct
    let labels = ['Spelling', 'Punctuation', 'Grammar', 'Under-
standing'];
    let values = [0,0,0,0]; //will be used to store values to
use for the chart
    let colourHex =
['#02BCF580','#FA3CF780','#FAC02F80','#00E9B780']; //will be used for the
colours of each slice in the pie chart
    for(let i = 0; i < (storedData.length - 2); i++){ //loop
through all the questions
        if(storedData[i].correct == 1){ //if the student got
the question correct
            for(let j =0; j < labels.length; j++){ //determine
what question type it was and increment the count by 1
                if(storedData[i].type == labels[j]){
                    values[j] += 1;
                }
            }
        }
    }
    if(newStudent) { //if a pie chart was created before,
destory it before making a new one
        positivePieChart.destroy();
    }
    //creating the pie chart
    let goodPieChart = piechartGood.getContext('2d');
    positivePieChart = new Chart(goodPieChart, {
        type: 'doughnut',
        data: {
            labels: labels,
            datasets: [
                {
                    label: 'Question Types',
                    backgroundColor: colourHex,
                    borderWidth: 5,
                    data: values
                }
            ],
            options: {
                responsive: true,
                legend: {
                    position: 'bottom'
                },
                title: {
                    display: true,

```

```

        text: 'Percentage difference of the question
type that were answered correctly',
        fontSize: 25,
        fontColor: '#000',
        padding: 15
    }
}
});
//preparing pie chart to show question type of all ques-
tions answered incorrect
values = [0,0,0,0]; //will be used to store values to use
for the chart
for(let i = 0; i < (storedData.length - 2); i++){ //loop
through all the questions
    if(storedData[i].correct == 0){ //if the student got
the question incorrect
        for(let j =0; j < labels.length; j++){ //determine
what question type it was and increment the count by 1
            if(storedData[i].type == labels[j]){
                values[j] += 1;
            }
        }
    }
}
if(newStudent) { //if a pie chart was created before,
destory it before making a new one
    negativePieChart.destroy();
}
//creating the pie chart
let badPieChart = piechartBad.getContext('2d');
negativePieChart = new Chart(piechartBad, {
    type: 'doughnut',
    data: {
        labels: labels,
        datasets: [{
            label: 'Question Types',
            backgroundColor: colourHex,
            borderWidth: 5,
            data: values
        }]
    },
    options: {
        responsive: true,
        legend: {
            position: 'bottom'
        },
        title: {
            display: true,
            text: 'Percentage difference of the question
type that were answered incorrectly',
            fontSize: 25,
            fontColor: '#000',
            padding: 15
        }
    }
});
barchartContainer.classList.add('bar-graph-container');
//preparing bar chart to show average time taken for each
question type
values = [0,0,0,0]; //will be used to store values to use
for the chart

```

```

        let valuesCount = [0,0,0,0]; //will be used to store values
        to use for the chart
            for(let i = 0; i < (storedData.length - 2); i++){ //loop
            through all the questions
                for(let j =0; j < labels.length; j++){ //loop through
                all question types
                    if(storedData[i].type == labels[j]){ //if the ques-
                    tion type is the current type being checked
                        values[j] += storedData[i].time; //increment
                    the time taken
                        valuesCount[j] += 1; //that question type count
                    is incremented by 1
                }
            }
        }
        for(let i = 0; i < values.length; i++){ //loop through all
        the times divided by their count to get average time
            values[i] = Math.floor(values[i] / valuesCount[i]);
        }
        if(newStudent) { //if a bar chart was created before,
        destroy it before making a new one
            timeBarChart.destroy();
        }
        //creating the bar chart
        let barChartVis = barchart.getContext('2d');
        timeBarChart = new Chart(barChartVis, {
            type: 'bar',
            data: {
                labels: labels,
                datasets: [
                    {
                        label: 'Average time in seconds',
                        backgroundColor: colourHex,
                        data: values
                    }
                ],
                options: {
                    responsive: true,
                    legend: {
                        display: false
                    },
                    title: {
                        display: true,
                        text: 'Bar chart showing average time taken for
each question type',
                        fontSize: 25,
                        fontColor: '#000',
                        padding: 15
                    },
                    layout: {
                        padding: 50
                    }
                }
            });
        newStudent = true; //once these charts have been made once,
        set to true so subsequent charts get destroyed before new charts are made
    }
},
error: function(errorMessage) { //if an error occurred
    messageContainer.innerHTML = errorMessage; //let the user know
}
);

```

```
}

function isJsonString(str) { //checks if given string is in a json format
    try {
        let data = JSON.parse(str); //if this fails
    } catch (e) {
        return false; //it isn't
    }
    return true; //otherwise it is
}
```

## [Figure 28]

### view-class-results-script.js

```
//containers and button used within results page
const classAttempts = document.getElementById('class-attempts');
const classTime = document.getElementById('class-average-time-taken');
const classMessageContainer = document.getElementById('class-message');
const classVisualisationContainer = document.getElementById('class-visualisation');

//variables used for chart creation
const classPiechartContainer = document.getElementById('class-pie-chart-container');
const classBarchartContainer = document.getElementById('class-bar-chart-container');
const classPiechartGood = document.getElementById('class-piechart-good');
const classPiechartBad = document.getElementById('class-piechart-bad');
const classBarchart = document.getElementById('class-barchart');
const classChartMessage = document.getElementById('class-chart-message');

//event listeners for when buttons are clicked
window.addEventListener('resize', windowSizeCheck);
window.addEventListener('load', visualisation);

//chart looks unappealing on small screens, this checks to see if the screen is too small, if so a warning message is revealed
function windowSizeCheck() {
    let w = window.innerWidth;
    if(w < 700){
        classPiechartContainer.classList.add('hide');
        classBarchartContainer.classList.add('hide');
        classChartMessage.classList.remove('hide');
    } else {
        classPiechartContainer.classList.remove('hide');
        classBarchartContainer.classList.remove('hide');
        classChartMessage.classList.add('hide');
    }
}

//takes metrics from database and turns them into visual information
function visualisation() {
    $.ajax({ //post request to get data from database
        type: 'POST',
        url: 'inc/view-results.inc.php',
        data: {username: '', typeOfData: 'class', submit : 'submit'},
        beforeSend: function(){ //before requesting data, the visualisation page prepared
            windowSizeCheck();
        },
        success: function(data) { //once data is received
            classMessageContainer.classList.remove('hide');
            if(!isJSONString(data)){ //checks to see if the data gotten is json string or not, if not then an error message is shown
                classMessageContainer.innerHTML = data;
            } else { //otherwise the data is handled and converted to information
                classVisualisationContainer.classList.remove('hide');
                storedData = JSON.parse(data);
                classMessageContainer.classList.add('hide');
                classAttempts.innerHTML = storedData[(storedData.length - 2)] + '<br> total attempt(s)'; //students attempts shown
            }
        }
    });
}
```

```

        let averageTime = Math.floor(storedData[(storedData.length
- 1)] / storedData[(storedData.length - 3)]);
        let minutes = Math.floor(averageTime / 60);
        let seconds = averageTime - minutes * 60;
        let value = '';
        if(minutes < 10) {
            value += '0';
        }
        value += minutes + ':';
        if(seconds < 10) {
            value += '0';
        }
        value += seconds;
        classTime.innerHTML = value + '<br> on average to
complete'; //time is converted to minuets and seconds before being shown
        //preparing pie chart to show question type of all
questions answered correct
        classPiechartContainer.classList.add('pie-graph-
container');
        let labels = ['Spelling', 'Punctuation', 'Grammar',
'Understanding'];
        let values = [0,0,0,0]; //will be used to store values to
use for the chart
        let colourHex =
['#02BCF580','#FA3CF780','#FAC02F80','#00E9B780']; //will be used for the
colours of each slice in the pie chart
        for(let i = 0; i < (storedData.length - 3); i++){ //loop
through all the questions (since last three items in array are time,
attempts and count)
            if(storedData[i].correct == 1){ //if the student got
the question correct
                for(let j =0; j < labels.length; j++){ //determine
what question type it was and increment the count by 1
                    if(storedData[i].type == labels[j]){
                        values[j] += 1;
                    }
                }
            }
        }
        //creating the pie chart
        let goodPieChart = classPiechartGood.getContext('2d');
        new Chart(goodPieChart, {
            type: 'doughnut',
            data: {
                labels: labels,
                datasets: [{

                    label: 'Question Types',
                    backgroundColor: colourHex,
                    borderWidth: 5,
                    data: values
                }]
            },
            options: {
                responsive: true,
                legend: {
                    position: 'bottom'
                },
                title: {
                    display: true,
                    text: 'Percentage difference of the question
type that were answered correctly',
                }
            }
        });
    }
}

```

```

        fontSize: 25,
        fontColor: '#000',
        padding: 15
    }
}
});

//preparing pie chart to show question type of all
questions answered incorrect
values = [0,0,0,0]; //will be used to store values to use
for the chart
for(let i = 0; i < (storedData.length - 3); i++){//loop
through all the questions
    if(storedData[i].correct == 0){ //if the student got
the question incorrect
        for(let j = 0; j < labels.length; j++){ //determine
what question type it was and increment the count by 1
            if(storedData[i].type == labels[j]){
                values[j] += 1;
            }
        }
    }
}

//creating the pie chart
let badPieChart = classPiechartBad.getContext('2d');
new Chart(badPieChart, {
    type: 'doughnut',
    data: {
        labels: labels,
        datasets: [{
            label: 'Question Types',
            backgroundColor: colourHex,
            borderWidth: 5,
            data: values
        }]
    },
    options: {
        responsive: true,
        legend: {
            position: 'bottom'
        },
        title: {
            display: true,
            text: 'Percentage difference of the question
type that were answered incorrectly',
            fontSize: 25,
            fontColor: '#000',
            padding: 15
        }
    }
});
//preparing bar chart to show average time taken for each
question type
classBarchartContainer.classList.add('bar-graph-
container');

let barChartVis = classBarchart.getContext('2d');
values = [0,0,0,0]; //will be used to store values to use
for the chart
let valuesCount = [0,0,0,0]; //will be used to store values
to use for the chart
for(let i = 0; i < (storedData.length - 3); i++){ //loop
through all the questions

```

```

        for(let j = 0; j < labels.length; j++){ //loop through
all question types
            if(storedData[i].type == labels[j]){ //if the
question type is the current type being checked
                values[j] += storedData[i].time; //increment
the time taken
                valuesCount[j] += 1; //that question type count
is incremented by 1
            }
        }
    }
    for(let i = 0; i < values.length; i++){ //loop through all
the times divided by their count to get average time
    values[i] = Math.floor(values[i] / valuesCount[i]);
}
//creating the bar chart
new Chart(barChartVis, {
    type: 'bar',
    data: {
        xAxisID: 'HELLO',
        labels: labels,
        datasets: [
            {
                label: 'Average time in seconds',
                backgroundColor: colourHex,
                data: values
            }
        ],
        options: {
            responsive: true,
            legend: {
                display: false
            },
            title: {
                display: true,
                text: 'Bar chart showing average time taken for
each question type',
                fontSize: 25,
                fontColor: '#000',
                padding: 15
            },
            layout: {
                padding: 50
            }
        }
    });
},
error: function(errorMessage) { //if an error occurred
    messageContainer.innerHTML = errorMessage; //let the user know
}
});
}

function isJsonString(str) { //checks if given string is in a json format
try {
    let data = JSON.parse(str); //if this fails
} catch (e) {
    return false; //it isn't
}
return true; //otherwise it is
}

```

## [Figure 29]

### styles.css

```
body {  
    font-family: 'Karla', sans-serif;  
    padding: 0;  
    margin: 0;  
    background-color: #fff  
}  
  
.banner-container h1 {  
    color: #000;  
    font-size: 5vw; /* scale font 5 view width */  
    text-transform: uppercase;  
    font-family: 'Alfa Slab One', cursive;  
    position: relative;  
    bottom: 70px  
}  
  
.banner-container h2 {  
    font-family: Orbitron, sans-serif;  
    color: #000;  
    font-size: 3vw;  
    position: relative;  
    bottom: 110px  
}  
  
.nav-bg {  
    background-color: #fff  
}  
  
#my-nav {  
    text-align: center;  
    font-family: 'Coming Soon', cursive;  
}  
  
#my-nav a {  
    font-size: 28px;  
    padding: 0 20px;  
    color: #fff;  
    transition: all .3s ease-in-out;  
}  
  
#my-nav ul li {  
    padding: 0px 25px 0px 0px;  
}  
  
.home-tab {  
    border-radius: 25px;  
    background-color: #C0C0C0  
}  
  
.nav-bg:hover .home-tab,  
.activated .home-tab {  
    background-color: #EAC435;  
    transition: background .6s ease-in-out  
}  
  
.store-tab {  
    border-radius: 25px;  
    background-color: #C0C0C0;  
}
```

```
.nav-bg:hover .store-tab,
.activated .store-tab {
    background-color: #345995;
    transition: background .6s ease-in-out
}

.contact-tab {
    border-radius: 25px;
    background-color: #C0C0C0
}

.nav-bg:hover .contact-tab,
.activated .contact-tab {
    background-color: #03CEA4;
    transition: background .6s ease-in-out
}

.about-tab {
    border-radius: 25px;
    background-color: #C0C0C0
}

.nav-bg:hover .about-tab,
.activated .about-tab {
    background-color: #CA1551;
    transition: background .6s ease-in-out
}

.help-tab {
    border-radius: 25px;
    background-color: #C0C0C0
}

.nav-bg:hover .help-tab,
.activated .help-tab {
    background-color: #FB4D3D;
    transition: background .6s ease-in-out
}

.signin-tab {
    border-radius: 25px;
    background-color: #C0C0C0
}

.nav-bg:hover .signin-tab,
.activated .signin-tab {
    background-color: #7600da;
    transition: background .6s ease-in-out
}

#my-nav ul li:hover a {
    transform: scale(1.1);
}

.navigation-container {
    padding: 20px
}

.home-icon {
    background: url(..../img/home.png) no-repeat left
```

```

}

.store-icon {
    background: url(..../img/store.png) no-repeat left
}

.contact-icon {
    background: url(..../img/contact.png) no-repeat left
}

.about-icon {
    background: url(..../img/about.png) no-repeat left
}

.help-icon {
    background: url(..../img/help.png) no-repeat left
}

.signin-icon {
    background: url(..../img/signin.png) no-repeat left
}

#my-nav span {
    padding: 15px
}

.banner-container {
    padding: 25px 4% 25px 4%;
    background: url(..../img/background.png) no-repeat;
    background-size: cover;
}

.banner {
    padding: 180px 180px;
}

.content {
    padding: 25px 5%;
    background-color: #f1f1f1;
}

.col-sm-2,
.col-sm-4,
.col-sm-6 {
    padding: 0
}

img {
    width: 100%;
    height: 100%
}

.thumbnail {
    top: 0;
    transition: top ease .5s;
    padding: 3px;
}

.thumbnail:hover {
    top: -15px;
    box-shadow: 10px 10px 8px #b0b0b0;
}

```

```
    cursor: pointer;
    padding: 0px;
}

.activities {
    background-color: transparent;
}

.my-footer-container {
    font-family: Orbitron, sans-serif;
    width: 100%;
    height: auto;
    margin: auto;
    background-color: #10A450;
}

.footer-content {
    margin: 0px 20%;
    padding: 25px 0px;
}

.footer-links {
    width: 50%;
    padding: 0px 20px;
}

.footer-heading {
    font-weight: 600;
    padding: 0px 10px;
    color: #83D0A4;
}

.footer-links ul {
    margin: 0px;
    padding: 0px 10px;
    list-style: none;
}

.footer-links ul li {
    text-decoration: none;
}

.footer-links ul li a {
    font-size: 12px;
    text-decoration: none;
    color: #035B31;
}

.footer-links ul li a:hover,
.social-links ul li a:hover {
    color: #10DC96;
}

.social-links {
    width: 100%;
    margin: auto;
}

.social-links ul {
    margin: auto;
    padding: 0;
```

```
    text-align: center;
}

.social-links ul li {
    display: inline-block;
    width: 50px;
    margin: 10px 0px;
}

.social-links ul li a {
    font-size: 40px;
    color: #fff
}

.app-links {
    margin: auto;
}

.panel {
    font-family: Orbitron, sans-serif;
    background-color: #00aada;
}

.panel-content {
    padding: 35px
}

.carousel-container {
    width: 100%;
    height: 100%;
    display: flex;
    align-items: center
}

.my-carousel {
    margin: 0 auto;
    width: 520px;
    max-width: 100%
}

.panel-text-container {
    display: flex;
    align-items: center;
    height: 100%;
    text-align: center
}

.panel-text {
    margin: auto;
    padding-left: 150px
}

.panel-text h1 {
    color: #fff
}

.carousel-control-next:hover,
.carousel-control-prev:hover {
    background-color: #006e8e;
    transition: background .9s ease-in-out
}
```

```
.login-container {  
    margin: auto;  
    text-align: center;  
}  
  
.login-content {  
    background: url(..../img/login-background.jpg) no-repeat;  
    background-size: cover;  
    background-position: center center;  
    padding-bottom: 100px;  
}  
  
.register-container {  
    margin: auto;  
    text-align: center;  
    padding: 25px;  
}  
  
.register-content {  
    background-color: #f1f1f1;  
}  
  
.login-content form,  
.register-content form {  
    margin: auto;  
    width: 35%;  
    padding: 25px;  
    border: 5px solid #000;  
    background-color: #fff;  
}  
  
.form-img-container {  
    width: 32%;  
    margin: auto;  
    height: auto;  
}  
  
.avatar {  
    position: relative;  
    top: 50px;  
}  
  
.form-input-container {  
    margin: auto;  
    padding: 25px 0px;  
}  
  
.form-input-container button {  
    font-family: Orbitron, sans-serif;  
    background-color: #00aada;  
    border: 5px solid #000;  
    width: 100%;  
    padding: 15px 20px;  
    margin: 10px 0;  
    color: #fff  
}  
  
.form-input-container button:hover {  
    background-color: #0082a7;  
}
```

```
.login-title {  
    font-family: Orbitron, sans-serif;  
    font-weight: 600px;  
    font-size: 20px;  
}  
  
.error {  
    border: 5px solid red;  
    width: 100%;  
    padding: 15px 20px;  
    margin: 10px 0;  
}  
  
.default {  
    border: 5px solid #000;  
    width: 100%;  
    padding: 15px 20px;  
    margin: 10px 0;  
}  
  
.dashboard-content {  
    font-size: 2.5vw;  
    position: absolute;  
    bottom: 20px;  
    left: 35px;  
    color: #fff;  
}  
  
.dashboard-button{  
    height: 500px;  
}  
  
.student-records-container {  
    text-align: center;  
}  
  
.student-records-content form {  
    padding: 5px 0px;  
}  
  
.student-records-content input,  
.student-records-content button,  
.student-records-content select {  
    padding: 10px 25px;  
    border: 3px solid #000;  
    background-color: #fff;  
    color: #000;  
}  
  
.student-records-content button {  
    font-family: Orbitron, sans-serif;  
    background-color: #00aada;  
    color: #fff;  
}  
  
.student-records-content button:hover {  
    background-color: #0082a7;  
}
```

```
.student-records-nav {  
    padding: 15px;  
    text-align: center;  
}  
  
.student-records-nav fieldset {  
    display: inline-block;  
    padding: 20px 30px;  
}  
  
.pagination {  
    display: inline-block;  
}  
  
.pagination a {  
    color: black;  
    padding: 5px 25px;  
    text-decoration: none;  
}  
  
.pagination a:hover {  
    border-bottom: 3px solid #0082a7;  
}  
  
.current-page {  
    border-bottom: 3px solid #00aada;  
}  
  
.top-bar,  
.bottom-bar {  
    padding: 50px 0px;  
    background-color: #38185C;  
}  
  
.top-bar button,  
.bottom-bar button {  
    margin: -30px 50px;  
    background-color: #BF55EC;  
    border: none;  
}  
  
.top-bar {  
    border-top-left-radius: 8px;  
    border-top-right-radius: 8px;  
}  
  
.bottom-bar {  
    border-bottom-left-radius: 8px;  
    border-bottom-right-radius: 8px;  
}  
  
#title-btn{  
    border-radius: 50%;  
    width: 200px;  
    height: 200px;  
    position: relative;  
    top: -10px;  
    margin: 10px;  
    background-color: #BF55EC;  
    border: none;  
    color: #fff;
```

```

        animation-name: shake-in; /* Assigning the shake-in animation to what
ever component has this id */
        animation-duration: 2.5s; /* The animation should last for 2.5 seconds
*/
        animation-delay: 2.5s; /* Wait 2.5 seconds before beginning the anima-
tion */
    }

#title-btn i {
    position: relative;
    left: 5px;
    margin: 10px;
    font-size: 80px;
}

#title-page {
    background-image: url(..../img/quiz-background.png);
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center center;
    border-radius: 8px;
    box-shadow: 0px 0px 10px 8px;
    width: 100%;
    font-size: 25px;
    text-align: center;
    font-family: 'Chewy', cursive;
}

.result-good {
    color: #004F19;
}

.result-bad {
    color: #A60000;
}

.result-icon {
    transform: rotate(-25deg);
    font-size: 72px;
    padding: 5px 20px;
    animation: shake-in-result 3s infinite linear,
    /* Assigning the shake-in-result animation to what ever component has
this class */
    /* Animation last for 3 seconds and loops indefinitely */
}

#results-container {
    padding: 20px 50px;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}

.results {
    width: 550px;
    background-color: #38185C;
    opacity: 0.8;
    position: relative;
    top: -550px;
    left: 400px;
}

```

```

        border-radius: 15px 50px;
    }

#end-message{
    color: #fff;
    font-size: 100px;
    position: relative;
    top: 50px;
    animation: enlarge-in-start 5s infinite linear;
}

.bear-icon-end {
    height: 300px;
    position: relative;
    top: -180px;
    left: -290px;
    animation: up-n-down-end 1.5s infinite linear;
}

.end-star {
    width:300px;
    height:340px;
    position: relative;
    top: 155px;
    left: -300px;
    animation: spin 5s infinite linear;
}

.startpage-star {
    width:480px;
    height:520px;
    position: relative;
    top: 120px;
    left: 150px;
    animation: spin 20s infinite linear;
}

#startpage-btn {
    position: relative;
    font-size: 140px;
    top: 150px;
    left: -240px;
    background-color: transparent;
    border: none;
    color: #fff;
    animation: enlarge-in-start 1s infinite linear;
}

#home-btn, #mute-btn, #repeat-btn, #howtoplay-btn, #nextBtn, #endBtn {
    padding: 10px 20px;
    color: #fff;
    border-radius: 10px;
}

#repeat-btn {
    position: relative;
    right: 90px;
}

#home-btn:hover, #mute-btn:hover, #repeat-btn:hover, #howtoplay-btn:hover,
#nextBtn:hover, #endBtn:hover {

```

```

        background-color: #8516B3;
        transform: scale(1.2);
    }

#title-btn:hover {
    background-color: #8516B3;
    transform: scale(1.1);
}

#home-btn, #nextBtn, #endBtn {
    float: left;
}

#mute-btn, #repeat-btn, #howtoplay-btn {
    float: right;
}

#choices{
    padding: 10px;
}

#question {
    padding: 20px;
    color: #fff;
}

.activity-container {
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: 'Chewy', cursive;
}

.hide {
    display: none;
}

.startpage {
    height: 800px;
}

#activity, #transition, #start-page, #end-page {
    background-image: url(..../img/quiz-background.png);
    background-color: #000;
    background-repeat: no-repeat;
    background-size: cover;
    background-position: center center;
    border-radius: 8px;
    box-shadow: 0px 0px 10px 8px;
    width: 100%;
    font-size: 25px;
    text-align: center;
}

.choices {
    display: grid;
    grid-template-columns: repeat(2, auto);
    gap: 15px;
    box-shadow: 0px 0px 15px 5px;
}

```

```
.choice1 {
    background-color: #02BCF5;
    padding: 50px 0px;
    border: none;
    color: #fff;
    border-radius: 15px;
}

.choice2 {
    background-color: #FA3CF7;
    padding: 50px 0px;
    border: none;
    color: #fff;
    border-radius: 15px;
}

.choice3 {
    background-color: #FAC02F;
    padding: 50px 0px;
    border: none;
    color: #fff;
    border-radius: 15px;
}

.choice4 {
    background-color: #00E9B7;
    padding: 50px 0px;
    border: none;
    color: #fff;
    border-radius: 15px;
}

.choice1:hover {
    background-color: #0087BC;
}

.choice2:hover {
    background-color: #BD00BE;
}

.choice3:hover {
    background-color: #BD8B00;
}

.choice4:hover {
    background-color: #00B082;
}

.transition-star {
    padding-top: 10px;
    width: 380px;
    height: 420px;
    position: relative;
    top: 100px;
    left: 0;
    animation: spin 5s infinite linear;
}

.transition-page, .end-page {
    height: 800px;
}
```

```

.titlescreen-page {
    position: relative;
    top: 0;
    left: 0;
    overflow: hidden;
}

.titlescreen-star {
    padding-top: 10px;
    width: 400px;
    height: 400px;
    position: relative;
    top: 0;
    left: 0;
    animation-name: slide-in;
    animation-duration: 1s;
}

@keyframes spin { /* spin animation used within the activity */
    0% {transform: rotate(0deg);}
    100% {transform: rotate(359deg);}
}

@keyframes slide-in { /* slide-in animation used within the activity title
page background star */
    0% {left: 0px; top: -500px;}
    100% {left: 0px; top: 0px;}
}

@keyframes slide-in-bear { /* slide-in animation used within the activity
title page bear image */
    0% {left: 10px; top: -500px;}
    100% {left: 10px; top: 25px;}
}

@keyframes up-n-down { /* up-n-down animation used within the activity ti-
tle page bear image */
    0% {top: 25px;}
    50% {top: 50px;}
    100% {top: 25px;}
}

@keyframes up-n-down-activity { /* up-n-down animation used within the ac-
tivity quiz page bear image */
    0% {top: -50px;}
    50% {top: 0px;}
    100% {top: -50px;}
}

@keyframes up-n-down-end { /* up-n-down animation used within the activity
results page bear image */
    0% {top: -180px;}
    50% {top: -160px;}
    100% {top: -180px;}
}

@keyframes enlarge-in { /* enlarge-in animation used within the activity's
title page name */
    0% {transform: scale(0.0);}
}

```

```

    50% {transform: scale(1.2);}
    100% {transform: scale(1.0);}
}

@keyframes enlarge-in-start { /* enlarge-in animation used within the ac-
tivity's launch page start button */
    0% {transform: scale(1.0);}
    50% {transform: scale(1.2);}
    100% {transform: scale(1.0);}
}

@keyframes shake-in { /* shake-in animation used within the activity's ti-
tle page start button */
    0% {left: 0px; transform: rotate(0deg);}
    25% {left: 20px; transform: rotate(25deg);}
    50% {left: -20px; transform: rotate(-25deg);}
    100% {left: 0px; transform: rotate(0deg);}
}

@keyframes shake-in-result { /* shake-in animation used within the activi-
ty's result page result icons */
    0% {transform: rotate(-25deg);}
    50% {transform: rotate(25deg);}
    100% {transform: rotate(-25deg);}
}

.bear-icon-title, #bear-icon-activity, #bear-icon-transition, .bear-icon-
end {
    background-image: url(..../img/bear-happy.png);
    width:100%;
    background-repeat:no-repeat;
    background-size:contain;
    background-position: center center;
}

#transition-message {
    color: #fff;
    font-size: 100px;
    position: relative;
    top: -200px;
    animation: enlarge-in 2s, enlarge-in-start 1.5s infinite linear 2s;
}

.titlescreen-title{
    color: #fff;
    font-size: 75px;
    transform: scale(0.0);
    animation-name: enlarge-in;
    animation-duration: 2s;
    animation-delay: 1s;
    animation-fill-mode: forwards;
}

#bear-icon-transition {
    height:350px;
    position: relative;
    top: -290px;
    left: 10px;
}

#bear-icon-activity {

```

```

height:400px;
position: relative;
top: -50px;
animation: up-n-down-activity 2.5s infinite linear;
}

.bear-icon-title {
  position: absolute;
  top: 25px;
  left: 10px;
  height: 350px;
  animation: slide-in-bear 2s, up-n-down 2s infinite linear 2s;
}

.tabs {
  display: flex;
  justify-content: center;
  font-family: Orbitron, sans-serif;
}

.tabs button {
  height: 50px;
  border: none;
  color: #000;
  width: 500px;
  border-top-left-radius: 8px;
  border-top-right-radius: 8px;
}

.tabs button:focus {
  outline:0;
}

.tab-selected {
  background-color: #f1f1f1;
}

.tab-unselected {
  background-color: #fff;
}

.view-results-page {
  padding: 10px 0px;
}

.results-overview {
  font-size: 42px;
  background-color: #E0E0E0;
  padding: 25px 0px;
  width: 100%;
}

#student-name {
  font-size: 32px;
  padding: 10px 0px;
  background-color: #fff;
  font-family: 'Alfa Slab One', cursive;
}

.quiz-information {

```

```

padding: 25px 0px;
font-size: 20px;
}

.question-container, #pie-chart-correct, #class-pie-chart-correct {
    padding: 0px 20px;
    margin-left: 25px;
    background-color: #fff;
}

#pie-chart-incorrect, #class-pie-chart-incorrect {
    padding: 0px 20px;
    margin-right: 25px;
    background-color: #fff;
}

.question-container div{
    padding: 5px 0px;
}

.data-container {
    padding: 0px 20px;
    background-color: #fff;
    height: 100%;
    margin-right: 25px;
    display:flex;
    display: grid;
    grid-template-columns: repeat(1, auto);
}

.data-container div {
    margin:auto;
}

.correct-container {
    background-color: #50C878;
    margin: 25px 25px;
}

.incorrect-container {
    background-color: #ff0033;
    margin: 25px 25px;
}

.pie-graph-container {
    background: linear-gradient(to right, #50C878 50%, #ff0033 50%);
    margin: 25px 25px;
}

.bar-graph-container {
    background: linear-gradient(to right, #ff0033 50%, #50C878 50%);
    margin: 25px 25px;
}

.question-choices {
    display: grid;
    grid-template-columns: repeat(2, auto);
}

.questions-choice1{
    color: #02BCF580;
}

```

```

}

.questions-choice2{
    color: #FA3CF780;
}

.questions-choice3{
    color: #FAC02F80;
}

.questions-choice4{
    color: #00E9B780;
}

#bar-chart, #class-bar-chart {
    background-color: #fff;
    margin: 0 25px;
}

#message, #class-message {
    font-family: 'Alfa Slab One', cursive;
}

#chart-message {
    padding: 0 25px;
    padding-bottom: 25px;
    font-family: 'Alfa Slab One', cursive;
}

#class-attempts, #class-average-time-taken {
    text-align: center;
}

#visualisation, #class-visualisation {
    border: 5px solid #000;
}

/* When window width is less than 1300px */
@media (max-width:1300px) {
    .panel-text {
        padding-left: 0
    }

    #my-nav ul li {
        padding: 0px 0px 0px 0px;
    }

    .footer-content {
        margin: 0px 25px;
    }

    .dashboard-button{
        height: 400px;
    }
}

/* When window width is less than 1200px */
@media (max-width:1200px) {
    .student-records-content input,
    .student-records-content button,

```

```
.student-records-content select {
    width: 100%;
    margin: 5px;
}

.student-records-content form {
    padding: 25px 0px;
}

.banner {
    padding: 50px 60px;
}

.banner-container {
    padding: 10px 15px 0 15px
}

.main-container {
    padding: 0 10px 0 10px
}

.content {
    padding: 10px 20px
}

.banner-container h1 {
    position: relative;
    bottom: 0
}

.banner-container h2 {
    position: relative;
    bottom: 30px
}

#my-nav {
    padding: 25px;
}

#my-nav a {
    padding: 0 20px;
}

.home-tab {
    background-color: #EAC435
}

.store-tab {
    background-color: #345995;
}

.contact-tab {
    background-color: #03CEA4
}

.about-tab {
    background-color: #CA1551
}

.help-tab {
    background-color: #FB4D3D
```

```
}

.signin-tab {
    background-color: #7600da
}

.app-links {
    width: 300px;
    margin: auto;
}

.login-content form,
.register-content form {
    width: 50%;
    padding: 50px;
}

.form-img-container {
    width: 28%;
}

.avatar {
    position: relative;
    top: 60px;
}

.form-input-container {
    padding: 25px 0px;
}

.form-input-container button {
    padding: 10px 20px;
    margin: 5px 0;
}

.error {
    padding: 10px 20px;
    margin: 5px 0;
}

.default {
    padding: 10px 20px;
    margin: 5px 0;
}

.login-title {
    font-size: 25px;
}

.dashboard-content {
    font-size: 5vw;
}

.dashboard-button{
    height: 300px;
}

.titlescreen-title{
    font-size: 62px;
}
```

```

.bear-icon-end {
    display: none;
}

.end-star {
    display: none;
}

.results {
    position: relative;
    top: 80px;
    left: 180px;
}
}

/* When window width is less than 991px */
@media (max-width:991px) {
    .startpage-star {
        width:380px;
        height:420px;
        top: 150px;
        left: 120px;
    }

    #startpage-btn {
        font-size: 120px;
        top: 180px;
        left: -200px;
    }

    .results {
        width: 100%;
        position: relative;
        top: 80px;
        left: 0px;
    }

    .question-container, #pie-chart-correct, #class-pie-chart-correct {
        margin-right: 25px;
    }

    .data-container, #pie-chart-incorrect, #class-pie-chart-incorrect {
        margin-left: 25px;
    }
}

/* When window width is less than 950px */
@media (max-width:950px) {
    .login-content,
    .register-content {
        padding: 100px;
    }

    .login-content form,
    .register-content form {
        width: 95%;
        padding: 10px;
    }
}

.avatar {

```

```

        display: none;
    }

.form-input-container button {
    padding: 15px 20px;
    border: 2px solid #000;
    margin: 15px 0;
}

.error {
    padding: 15px 20px;
    border: 2px solid red;
    margin: 15px 0;
}

.default {
    padding: 15px 20px;
    border: 2px solid #000;
    margin: 15px 0;
}

.login-title {
    font-size: 15px;
}

.student-records-container {
    padding: 25px;
}

.student-records-content input,
.student-records-content button,
.student-records-content select {
    margin: 0px;
    border: 1px solid #000;
}

.results-overview {
    font-size: 32px;
    padding: 0px;
}

.results-overview div {
    padding: 10px 0px;
}

.question-container div{
    padding: 10px 0px;
}

.data-container div {
    padding: 10px 0px;
}
}

/* When window width is less than 767px */
@media (max-width:767px) {
    .titlescreen-title{
        font-size: 45px;
    }
    .bear-icon-title {
        top: 5px;
    }
}

```

```

        left: 5px;
        height: 400px;
    }
@keyframes slide-in-bear {
    0%   {left:5px; top:-500px;}
    100% {left:5px; top:5px;}
}
.titlescreen-star {
    width:400px;
    height:450px;
}
.startpage-star {
    width:280px;
    height:320px;
    top: 180px;
    left: 10px;
}
#startpage-btn {
    font-size: 100px;
    top: -50px;
    left: 10px;
}

.choices {
    grid-template-columns: repeat(1, auto);
    gap: 10px;
}

#choices{
    padding: 5px;
}

.choice1 {
    padding: 35px 0px;
}

.choice2 {
    padding: 35px 0px;
}

.choice3 {
    padding: 35px 0px;
}

.choice4 {
    padding: 35px 0px;
}

#bear-icon-activity {
    height:250px;
}

#transition-message {
    font-size: 70px;
    top: -50px;
}

.transition-page {
    height: 900px;
}

```

```
.transition-star {
    width:280px;
    height:320px;
    top: 200px;
}

#bear-icon-transition {
    height:250px;
    top: -90px;
    left: 5px;
}
#end-message{
    font-size: 70px;
}
}

/* When window width is less than 575.5px */
@media (max-width:575.5px) {
    .content {
        padding: 0 15px
    }

    .banner-container {
        padding: 10px 10px 0 10px
    }

    .banner-container h1 {
        font-size: 10vw;
        text-align: center;
        position: relative;
        top: 20px
    }

    .banner-container h2 {
        font-size: 6vw;
        text-align: center;
        position: relative;
        top: 10px
    }

    .banner {
        padding: 10px 10px
    }

    .app-links {
        width: 150px;
    }

    .login-content,
    .register-content {
        padding: 0px;
        height: auto;
        background: none;
        background-color: #fff;
    }

    .register-content {
        padding: 50px 0px;
    }

    .login-content form,
```

```
.register-content form {
  padding: 0px 10px;
  margin: auto;
  width: 100%;
  border: none;
}

.form-input-container button {
  padding: 10px 20px;
  border: 1px solid #000;
  margin: 5px 0;
}

.error {
  padding: 10px 20px;
  border: 1px solid red;
  margin: 5px 0;
}

.default {
  padding: 10px 20px;
  border: 1px solid #000;
  margin: 5px 0;
}

.login-title {
  font-size: 25px;
}

.dashboard-content {
  font-size: 6vw;
  bottom: 5px;
  left: 15px;
}

.dashboard-button{
  height: 150px;
}

.student-records-content form {
  padding: 5px 0px;
}

.student-records-container {
  padding: 10px;
}

.student-records-content input,
.student-records-content button,
.student-records-content select {
  border: none;
}

.pagination a {
  padding: 5px 10px;
}
}
```

# Appendix G: Testing

## [Figure 1]

### Whitebox Testing

#### Registering a User Account

Action	Expected Outcome	Actual Outcome	Status
User leaves fields empty	System sends user back to their registration page with the empty fields highlighted and informs the user that they must fill the field out	System sends user back to their registration page with the empty fields highlighted and informs the user that they must fill the field out	Pass
User inputs a string longer than allowed	System sends user back to their registration page with the field highlighted and informs the user that the string can't exceed its allocated length	System sends user back to their registration page with the field highlighted and informs the user that the string can't exceed its allocated length	Pass
User inputs a character into a field that doesn't accept it	System sends user back to their registration page with the field highlighted and informs the user then what the specific field can accept	System sends user back to their registration page with the field highlighted and informs the user then what the specific field can accept	Pass
User inputs two different strings into password and repeat password fields	System sends user back to their registration page with the field highlighted informs the user that the passwords don't match	System sends user back to their registration page with the field highlighted informs the user that the passwords don't match	Pass
User inputs an existing username	System sends user back to the registration page with the field highlighted informs the user that the username is already taken	System sends user back to the registration page with the field highlighted informs the user that the username is already taken	Pass
User inputs all fields correctly	Correct database to input the account details is identified through user	Correct database to input the account details is identified through user	Pass

	account type and the data are stored	account type and the data are stored	
User inputs some fields correctly	System sends user back to the login page with the correctly input fields pre filled with their previous data	System sends user back to the login page with the correctly input fields pre filled with their previous data	Pass

**[Figure 2]**

**Logging into an Account**

Action	Expected Outcome	Actual Outcome	Status
User selects one of the three account types in the login page	The system takes the user type selected and uses its corresponding database	The system takes the user type selected and uses its corresponding database	Pass
User leaves fields empty	System sends user back to the login page with the empty fields highlighted and informs the user that they must fill the field out	System sends user back to the login page with the empty fields highlighted and informs the user that they must fill the field out	Pass
User inputs a username that doesn't exist in the database	System sends user back to the login page with the username field highlighted and informs the user that the username doesn't exist	System sends user back to the login page with the username field highlighted and informs the user that the username doesn't exist	Pass
User inputs a username that does exist in the database but incorrect password	System sends user back to the login page with the username field prefilled with its previous data and the password field highlighted and informs the user that the password was incorrect	System sends user back to the login page with the username field prefilled with its previous data and the password field highlighted and informs the user that the password was incorrect	Pass
User inputs a username that does exist in the database with correct password	System creates a session and stores user info into session variables	System creates a session and stores user info into session variables	Pass

**[Figure 3]**

### Teacher Dashboard Main Functions

Action	Expected Outcome	Actual Outcome	Status
User selects an order to arrange the students in the assignment page	Order is shown in the URL which the system pulls from to order the students in the requested format	Order is shown in the URL which the system pulls from to order the students in the requested format	Pass
User selects a page number that doesn't exist	System sends user back to the dashboard page with an error message	System sends user back to the dashboard page with an error message	Pass
User selects a page number that does exist	The pagination is updated to reflect the new page number, the students that are required to be shown are calculated and shown on screen	The pagination is updated to reflect the new page number, the students that are required to be shown are calculated and shown on screen	Pass
User manages to send empty data to the system for assignments	System sends user back to the dashboard page with an error message	System sends user back to the dashboard page with an error message	Pass
User manages to send data pertaining to a non-existing student for assignments	System sends user back to the dashboard page with an error message	System sends user back to the dashboard page with an error message	Pass
User manages to send data pertaining to a non-existing activity for assignments	System sends user back to the dashboard page with an error message	System sends user back to the dashboard page with an error message	Pass
User assigns a student "none"	System deletes any existing assignments in the database and deletes any corresponding results stored. The user is then sent back to the assignment page with their order parameters intact	System deletes any existing assignments in the database and deletes any corresponding results stored. The user is then sent back to the assignment page with their order parameters intact	Pass
User assigns a student a task	System deletes any existing	System deletes any existing	Pass

	assignments in the database and deletes any corresponding results stored before inserting the new assignment. The user is then sent back to the assignment page with their order parameters intact	assignments in the database and deletes any corresponding results stored before inserting the new assignment. The user is then sent back to the assignment page with their order parameters intact	
User selects the student tab in view results page	Result visualisation tab is given the “hide” class and it removed from the student tab	Result visualisation tab is given the “hide” class and it removed from the student tab	Pass
User selects the results tab in view results page	Student tab is given the “hide” class and it removed from the visualisation tab	Student tab is given the “hide” class and it removed from the visualisation tab	Pass
A non-existing student result is requested	System hides the student tab, shows the results tab with a message informing the user doesn’t exist	System hides the student tab, shows the results tab with a message informing the user doesn’t exist	Pass
A student result is requested where no assignment was given	System hides the student tab, shows the results tab with a message informing the user didn’t have an assignment	System hides the student tab, shows the results tab with a message informing the user didn’t have an assignment	Pass
A student result is requested where no attempts was made	System hides the student tab, shows the results tab with a message informing the user has yet to attempt the assignment	System hides the student tab, shows the results tab with a message informing the user has yet to attempt the assignment	Pass
A student result is correctly requested	System retrieves the correct student’s details and outputs their corresponding data	System retrieves the correct student’s details and outputs their corresponding data	Pass

**[Figure 4]****Student Activity**

Action	Expected Outcome	Actual Outcome	Status
Student clicks landing page's start button	Browser is given permission to play sound due to user interaction, title screen sound is played, and CSS animations are shown	Browser is given permission to play sound due to user interaction, title screen sound is played, and CSS animations are shown	Pass
Student clicks title page's start button	Quiz script shuffles the questions inside the question array, displays the first of these questions to the user, plays the quiz music and begins recording user metrics	Quiz script shuffles the questions inside the question array, displays the first of these questions to the user, plays the quiz music and begins recording user metrics	Pass
Home button clicked	System sends the user back to the title screen of the activity, destroying existing stored metrics and resetting to be collected again when play is clicked	System sends the user back to the title screen of the activity, destroying existing stored metrics and resetting to be collected again when play is clicked	Pass
Repeat question clicked	System uses the Speech Synthesis API to talk the question out loud for the user	System uses the Speech Synthesis API to talk the question out loud for the user	Pass
Sound button clicked	The sound is either muted or unmuted depending on its current state (Boolean)	The sound is either muted or unmuted depending on its current state (Boolean)	Pass
Wrong answer clicked	Wrong answer soundbite is played, the metrics of the attempts the user had on the question is increased by one and if all three attempts were used then the user is sent to the transition page	Wrong answer soundbite is played, the metrics of the attempts the user had on the question is increased by one and if all three attempts were used then the user is sent to the transition page	Pass
Correct answer clicked	Correct answer soundbite is played,	Correct answer soundbite is played,	Pass

	the metrics of the attempts the user had on the question is increased by one and if all three attempts were used then the user is sent to the transition page	the metrics of the attempts the user had on the question is increased by one and if all three attempts were used then the user is sent to the transition page	
Next question button clicked	The question array is incremented by one, time-based metrics are reset, and the user is shown the new question	The question array is incremented by one, time-based metrics are reset, and the user is shown the new question	Pass
Reached the end of the quiz	If any complications arise, the user is alerted with an error message informing them with the reason for failure and possible solutions on how to rectify otherwise the results are stored into the database, shown to the student and music is played depending on if the student did well or not	If any complications arise, the user is alerted with an error message informing them with the reason for failure and possible solutions on how to rectify otherwise the results are stored into the database, shown to the student and music is played depending on if the student did well or not	Pass
Student completes a quiz they weren't assigned	Student results aren't stored in the database as their teacher didn't request it	Student results aren't stored in the database as their teacher didn't request it	Pass

**[Figure 5]**

**Miscellaneous Testing**

Action	Expected Outcome	Actual Outcome	Status
User attempts to access a page they don't have permission to view	User is sent back to the earliest page they can access	User is sent back to the earliest page they can access	Pass
A logged in user attempts to view their navigation bar	The navigation bar is changed depending on which user type is logged in to reflect the new pages they are given permission to visit	The navigation bar is changed depending on which user type is logged in to reflect the new pages they are given permission to visit	Pass
User shifts their screen size	Contents within the application adapt to the end user window size	Contents within the application adapt to the end user window size	Pass
User edits values in URL which is used to change contents within the page	The page adapts to the changes or sends the user back to an earlier safe page with an error message	The page adapts to the changes or sends the user back to an earlier safe page with an error message	Pass

**[Figure 6]**

**Blackbox Testing**

Action	Expected Outcome	Actual Outcome	Status
Admin account logs in	User is taken to the teacher registration page	User is taken to the teacher registration page	Pass
Admin inputs details into the registration page	A teacher account is created, and a success message is shown	A teacher account is created, and a success message is shown	Pass
Teacher account logs in	User is taken to their dashboard page	User is taken to their dashboard page	Pass
Teacher navigates to teacher profile page	The accounts details are shown to the user with the ability to input changes	The accounts details are shown to the user with the ability to input changes	Pass
Teacher updates details in teacher profile page	A success message and new details are shown	A success message and new details are shown	Pass
Teacher navigates to student profile page	User is taken to the student profile page with details about each student that is registered to them shown with the ability to input changes	User is taken to the student profile page with details about each student that is registered to them shown with the ability to input changes	Pass
Teacher updates details for a chosen student	A success message and new details are shown	A success message and new details are shown	Pass
Teacher navigates to add student page	User is taken to the student registration page	User is taken to the student registration page	Pass
Teacher inputs details into the registration page	A student account is created, and a success message is shown	A student account is created, and a success message is shown	Pass
Teacher navigates to assignment page	User is taken to the assignment page with details about assignments for each student that is registered to them shown with the ability to input changes	User is taken to the assignment page with details about assignments for each student that is registered to them shown with the ability to input changes	Pass
Teacher updates details to student's assignment status	A success message and new details are shown	A success message and new details are shown	Pass

Teacher navigates to view results page	User is taken to the results page with details about each student that is registered to them shown stating whether they have an assignment that can be viewed or not	User is taken to the results page with details about each student that is registered to them shown stating whether they have an assignment that can be viewed or not	Pass
Teacher clicks to view the results for a student	That student's individual results are shown to the user or an error messages if any problems occur	That student's individual results are shown to the user or an error messages if any problems occur	Pass
Teacher navigates to view class results page	User is taken to the class summery page with the class's results shown or an error messages if any problems occur	User is taken to the class summery page with the class's results shown or an error messages if any problems occur	Pass
Student account logs in	User is taken to the index page where all activities are shown to them	User is taken to the index page where all activities are shown to them	Pass
Student navigates to an activity page	The activity is displayed to the user waiting for them to interact	The activity is displayed to the user waiting for them to interact	Pass
Student interacts with the activity	The activity works smoothly from start to finish, in this case each question is shown to the user and the ability to fully complete the quiz is given	The activity works smoothly from start to finish, in this case each question is shown to the user and the ability to fully complete the quiz is given	Pass
Student reaches the end of the activity	The user is notified that their results were stored or in the case of an error, how they could fix the problem	The user is notified that their results were stored or in the case of an error, how they could fix the problem	Pass
An account logs out the application	The user is sent back to the homepage with their account logged out	The user is sent back to the homepage with their account logged out	Pass

# Appendix H: Weekly Logs

## [Figure 1]

Below are the bullet points of my project outlines that I have decided upon from my research.

### Stakeholders:

- **KS1 Students**
- **Teacher**
- **Parents**

### Type of Application:

- **Web App**

### Technologies:

- **React.js?**
- **JSON?**
- **Databases?**

### Topics Covered:

- **Maths**
  - Numbers and Shape
  - <https://www.bbc.com/bitesize/topics/z9pnbgq>
  - <https://www.bbc.com/bitesize/topics/zwyv4wx>
- **English**
  - Spelling and Grammar
  - <https://www.bbc.com/bitesize/topics/zcgv39q>
  - <https://www.bbc.com/bitesize/topics/zrqqtfr>
- **Science**
  - Animal Classification
  - <https://www.bbc.com/bitesize/topics/z6882hv>

### Learning Styles:

- **Visual**
  - Show the student visual representation of the content
  - Have them visualise it within their head

- **Verbal**

- Use sounds to represent the content
- Have the student repeat phrases to help them learn

- **Physical**

- Let the student use physical objects or their body for learning
- If it's counting, let the student use their fingers etc

Nick's Feedback:

Nick stated that my current design and choices were based on my own experience of education rather than justified through academic curriculum and expert options. He brought up how it was no use building an app if the users of the app won't find any use for it since I built it using my own assumptions rather than checking with students and teachers on what they would want. We discussed on options such as getting in touch with teachers who I could then walk through my project with and see what aspects they liked and didn't and use that feedback to guide my project. We also discussed on ethics of the project, as my project revolves around children it was imperative that ethical considerations were to take place, would I need to get a certification to work with kids or if I had the teacher's supervision was that enough? These are aspects we would need to consider as the project goes forward. Ultimately, he suggested I go and do as much research I can to help decide the best direction to take the project in.

To Do Next:

Research and justify design on the project

Rough plans on how to continue forward with the project

Find experts in the field related to the project

Set up Zotero

## [Figure 2]

Set-up Zotero and the browser extension to save research I do during the project process for future references. I then used it for research on my concept to justify (or debunk) the design:

An article by CEDAR RIENER AND DANIEL WILLINGHAM states there are no real proof of learning styles existing. Other factors such as a child's interest, social class and upbringing have more influence on how fast or slow a child learns. A study by Ikseon Choi, Sang Joon Lee and Jeongwan Kang back this claim up concluding that learning style didn't have an impact on learning and that students simply just adapted to the learning style they were being taught in.

"Students may have preferences about how to learn, but no evidence suggests that catering to those preferences will lead to better learning"

This concluding line has made me assess that perhaps the different learning styles of my project is unneeded and evidently unfounded upon. Perhaps I should keep my web application sole on assessing students. This is where teachers can create quizzes and interactive games based on what they've been teaching in class and the app simply relays which students are struggling and which aren't based on factors such as time to complete each question, time to complete as a whole, number of questions gotten wrong etc and thus the teacher can easily see which student's need more help with the content and which students can move on with the curriculum. According to an article on TES.com, Oxford scores for infant schools are extremely high while JR schools are comparatively low. This is a problem as it shows a decrease in performance of education as children grow. This, however, according to the article, may be due to inflated test results. My suggestion on using the app to assess the student's understanding on the content and thus helping those who are struggling may combat this. Teachers can help struggling students to improve rather than just moving on with the curriculum and assuming they know it due to high results. Thus, when moving on to JR class they will understand the content better and hopefully raise the results.

Need to discuss with Nick?

Furthermore, according to the UK government website, at the end of KS1 teachers are expected to feedback to parents the student's results of their time at school and the comparison of others in the same age group. As my application will assess students throughout the year, this would make these requirements much easier as they can see easily see how a student improved throughout the year and compare to their classmates

### Nick's Feedback:

We discussed the small revision to my application and Nick felt dropping the educational aspect of the app and making it solely for assessment may not be the correct choice. He suggested tweaking the idea so that it both educates and assesses a child, i.e. the app has an activity (or activities) that can both teach a child something as well as assess the child's understanding of it. We also discussed how to proceed with the project.

### To Do Next:

- Sketch out a reasonable architecture for what I believe my end application should be.
  - These include:
    - ER Diagrams for any databases needed
    - How the user interface should look (ensuring it is usable by a child)
    - Technologies I may need use
    - Can combine different databases? Is this possible? Is this useful?
    - ETC
- Discuss with Lahcen what my options are with aspects to do with my RASA
- Plan out the milestones needed from now till viva

### [Figure 3]

Completed the compliance statement talking about what my intentions are and how I plan on protecting and keeping all aspects of my project pertaining to outside individuals private and confidential. I aimed to ensure all participants will feel safe, I gain consent for every piece of data, if any, I collect and that they have the right to drop out and have their data destroyed at any time during the project process

I sketched up a template user interface design for my application, using UI conventions and other applications designed for children as inspiration to help guide the design and hopefully make it usable for children. Some key aspects include, having the navbar always situated at the top. This is a ui norm that children will grow up with and learn while adults are already accustomed too. All the technical aspects of the app are always in the navbar, meaning the log in page and other important aspect. Most children at this stage will struggle with logging in and thus naturally will require a teacher or parent to help them with this. The navbar will be a clean simple black design, this juxtaposed with the colourful design of the interactable for children will let the child know that anything dark and black isn't for them. This should stop the child from focusing and interacting with things they shouldn't. This allows me to have a black bar at the top of every interactable game displaying things such as score, this isn't important for the child to know but helps the parent or teacher to see the progress of the child at a glance. Since this is similar design to the navbars that the child associates with something for grownups, the child once again will generally leave this alone. Each activity will have a large icon as the buttons, I can't expect every child that uses the application can read thus I will use buttons and icons to convey meaning rather than words exclusively. Each activity will be colour coded and the design of the activity will follow that colour convention. If the icon is red, the activity will have a lot of red colours. This way a child associates an activity with a colour, so they can click on the correct buttons without know what the icons name is. Within the app, I will have 3 difficulty levels, easy, medium and hard. This, once again, needs to be conveyed visually rather than with words. I plan on doing this through size, hierarchical and colour theory. It is natural to click on things from top down, thus the icons will be placed from easy to hard from top down. Each icon will have harsher colours as they go on, easy will be a light and inviting green, while hard would be a dark orange. This naturally makes the child want to click the more inviting colour before the less inviting one. The buttons will also be shaped to convey the order, easy will be larger than hard, this way the child naturally would want to click on the bigger button first. I will place icons with the home button and back button in the same place for every page, this way once the child sees what the action of a button is they know if they see the same icon in the same position it will have the same effect. Other design considerations have been placed too...

Created a possible ER Diagram for my project, will discuss with Nick on the logistics

Created a plan to follow for the remainder of the project.

### Nick's Feedback:

Nick suggested I get in contact with teachers at this point to justify the assessment aspect of the app. I should ask them if they already have access to similar technologies and if there's anything missing. This ensures my product is justified in that it is different from other existing products and thus has a reason to exist. Nick also suggested it is time for me to start planning the functionality of the technology, this is an abstract way to visualising how all the possible different components connect to create the bigger picture and make it easier for me to start development when I do start. This will also help me see which aspects of the application I can create in the time provided and how I can extend the application in the future. Nick also suggested I do market research on other similar existing application, so I can see where the market currently is and if there is anything missing that my application can fill in and provide.

### To Do Next:

- Create a functional overview of the project (planning)
- Looking at other similar products (market research)
- Ask Teachers how they assess their students (market research)
- Look at google scholar for government websites that talk about the specifications on assessing KS1 students (market research)

## [Figure 4]

Contacted a few teachers regarding the five following questions:

- 1) Do you already use technology within the classroom for either teaching or assessing children, if so could you provide examples?
- 2) Is having a way to teach and assess students through technology an idea you believe is worthwhile and would you see yourself adopting it if proven to work?
- 3) Currently, how are students assessed on their understanding of concepts?
- 4) Are there aspects of assessing you believe would be valuable but aren't possible with current techniques?
- 5) What types of features would you expect to see from an application with a concept like this?

I have received feedback on these and am expecting to receive more replies soon. To summarise the data I have collected,

- 1) Yes, technology is used within the classroom already, but it is mainly used as a tool for teaching students rather than a tool for assessing them.
- 2) Generally positive to the idea, as they already use technology in teaching as is, they see it as a natural progression to move on to using technology to assess students as well.
- 3) Students are assessed through a range of assessment methods, usually using colour coding to make things easier. I can use this existing colour coding within my visualisation to make the use of the program easier to learn and use. For example, pink is used to signify the student has not understood the learning objectives where as green is used to show they have etc. Students also are asked to fill out a form before new learning to discuss what they already know of the topic and what they feel they need help on. This gave me an idea ton perhaps have a primitive rating system after each assessment for the student to convey how they felt about the topic. This won't give detailed feedback to the teacher on the student's current understanding but coupled with the assessment scores will be a good starting point for teachers to help further develop each child's understanding of the topics. Student's are often assessed through tests to see their level of understanding. Time taken isn't a huge factor when assessing KS1, however, it does play a more important role when moving on from KS1 to KS2 and onwards.
- 4) It would be useful to see how long students answer each specific question more from KS2 onwards, it appears. This would help inform teachers if there is a trend of students taking too long on a specific topic and if the teacher would need to go over this again with the whole class.

5) The two common responses received was that a general overview of the entire class would be helpful as it would provide teachers with an Insite into the whole class performance and they can adapt their teaching styles to maintain or improve on it. The other common response was allowing the application to tailor itself to suit individual pupils at differing levels would be helpful as students who may be struggling can be brought up with their peer's level at their own pace while those excelling can continue to challenge themselves.

I also did further research in the market of my application for inspiration and justification of design.

I created a functional diagram of the application I plan on creating to help me visualise the key components and how each would interact with each other

Nick's Feedback:

Nick said that the feedback I received was great, but he thought more specific answers would be more useful. Such as what specific measurements do teachers look for when assessing students etc. We also discussed how my questions can lead people and receiving wrong feedback, this is a good lesson when it comes to designing the application. We discussed my functional diagram and he suggested improvements to it I can make to make the look of it much clearer and more understandable. We discussed development styles in that I develop and test my application in tandem so that my design is always justified and going in the correct direction.

To Do Next:

- Get more in-depth feedback from teachers
- Look for teachers who are willing to test the application
- Improve the functional diagram so it is cleaner
- Begin developing/designing the project up

## **[Figure 5]**

Continued with development of frontend, sent prototype designs to teachers and asked for further feedback and any extra information that may be helpful to me. Began designing the backend as well.

### Nick's Feedback:

Discussed with Nick that my development has slowed down due to personal issues, Nick said he understands however I should, if I can, avoid slowdowns at this stage. He told me to continue development and making progress with the project. I told him that the teachers I have been in contact with stated they weren't KS1 teachers and he said while their feedback is helpful, receiving feedback from actual KS1 students would be more beneficial and I should ask these teachers to pass on my contact details on to their colleagues.

### To Do Next:

- Continue development
- Ask existing teachers emailed for KS1 contacts for a more accurate feedback

## **[Figure 6]**

No progress due to a medical issue

### Nick's Feedback:

Nick suggests I apply for extenuating circumstances

### To Do Next:

- See what my options for extenuating circumstances are and decide what to do

## **[Figure 7]**

No progress due to continued medical issue

### Nick's Feedback:

Haven't been able to contact Lahcen about extenuating circumstances, emailed Nick asking for possible solutions before term is over

### To Do Next:

- Awaiting Nick and Lahcen replies.

## **[Figure 8]**

Medical issue is starting to subside, will begin working on the project again soon.

### Nick's Feedback:

I have discussed with Lahcen and Nick about extenuating circumstances. I have been told to fill out a form and provide evidence, which I have done and sent off to [computing@gold.ac.uk](mailto:computing@gold.ac.uk). Should receive a reply back confirming soon.

### To Do Next:

- Begin development back up again