OTTER: Online TMS Training Experience Reboot Final Report

Prepared By Team 3 - Ghost Code

Bailey Alexander
Back-end & Deployment

Ayesha Essop
Co- Leader & Front-end

Lucas Tavares
Leader & Front-end

Sahil Kumar
Back-end & Front-end

Angus Wright
Back-end & Deployment

Emily Zhao
Front-end

Executive Summary

"Early disease detection is where digital health could make a difference (1)."

Aiden Petrie

VERIFY is an innovative study that started in New Zealand and has spread worldwide with time. Multiple health organisations became a part of the study, and slowly, new clinicians became a part of it, too. Clinicians in this field utilise Transcranial Magnetic Stimulation (TMS) to identify stroke status in enrolled patients and determine recovery outcomes. Strokes are a significant health concern globally and in New Zealand, affecting millions and causing substantial disability (2,3). The VERIFY study addresses this by utilising TMS to provide valuable prognostic information, thereby aiding in formulating effective rehabilitation strategies.

Proper staff training is crucial to support the implementation of TMS in the clinical setting. The original method for assessing the learning progress of clinicians involved in the VERIFY study was through Google Forms. However, this approach was deemed inefficient, offering no organisational structure and failing to provide immediate feedback on quiz responses. This lack of feedback restricted the learning process and made it challenging to track clinicians' progress.

We developed a streamlined quiz platform tailored to clinicians' busy schedules in response to these challenges. Our platform is designed to be minimalistic and user-friendly, allowing clinicians to sign up effortlessly and access all necessary quizzes in one place. This eliminates the need to manage multiple forms and reduces email clutter. The platform also enables clients to monitor clinicians' progress and control the quiz content displayed, ensuring a more organised and efficient learning environment.

Our investigation revealed several outcomes. The new platform significantly improves the organisation and efficiency of the learning process for clinicians. Clinicians receive immediate feedback on quiz answers, enhancing their learning experience. Clients can easily track and manage clinicians' progress, providing a comprehensive overview in one centralised location.

In conclusion, our minimalistic and straightforward quiz platform addresses the inefficiencies of the old system, making it easier for clinicians to engage with the training material and for clients to manage and assess the learning progress. This innovation supports the overarching goals of the VERIFY study by ensuring that clinicians are well-equipped to utilise TMS effectively, ultimately improving the prognosis and rehabilitation of stroke patients.

Table of Contents

Introduction

Our project aimed to rework and improve the online training and certification system that a clinician working in a hospital or other medical office would use to become certified in TMS. The certification process involves passing 6 modules, each with a practice and final quiz of multiple choice questions. Upon completing all 6 modules, a clinician can also complete a recertification quiz to keep their certification valid, as a certification in TMS is only valid for 1 year after issue. There is also a practical test a clinician must pass; however, our project was focused only on the theory and related quizzes.

We approached this project by building a website that would be used to access the practice and final quizzes for each module, as well as the recertification quiz. It will also serve as the main place where a clinician is awarded their certification and as a data collecting system for admins to view and review statistics about quiz completion, clinician progress, etc.

We proposed the following as the primary outcomes of our project:

- Our website should have a functioning quiz system which allows a clinician to progress through their training
- Our website should be easy to use to keep stress levels for clinicians down to a minimum
- Our website should be easy to service and maintain for admins
- Our website should be scalable to be used on a worldwide stage at multiple medical institutes in numerous countries

If our website satisfied these points at a minimum, then our project could be called a success.

As previously mentioned, our website is being used worldwide at hospitals, medical centers and other health organizations. Our primary users are clinicians looking to be certified to use TMS, whether they are doctors, nurses, therapists or specialists. These clinicians can lead busy and stressful lives, so our website should be quick and easy to use to minimize the extra mental load. We also provide services to allow an administrator to create, edit and delete questions, monitor a clinician's progress or adjust a clinician's information, e.g. certification status. These processes should not be complicated, allowing for quick processing of tasks.

Background

The Verify Study is a study set up by medical researchers at the University of Auckland. Its goal is to understand better how a stroke affects a patient and how to improve their recovery process. Part of this study is conducted using a process called Transcranial Magnetic Stimulation (TMS), and a critical step in the study is being certified to use TMS. This is where OTTER comes in. OTTER stands for the Online TMS Training Experience Reboot, and it forms the primary goal of our project.

Our client's existing system involved a series of Google Forms, which acted as the quiz and certification software and used a paid plugin. A user would be sent a link to a Google Form, acting as a practice quiz for a training module. Upon completing this quiz, another email will be sent with a link to a Google Form that will act as the final quiz for the same module. Upon completing this quiz, another email will be sent with the certificate of achievement for that module. This process would repeat another 5 times until all 6 training modules were passed. Similarly, the recertification quiz was another link sent via another email.

Our clients were particularly having issues with giving appropriate feedback to a correct or incorrect answer, producing a quiz of randomly selected questions to discourage cheating, and using a paid plugin to mark and award certification. Google Forms did not provide our clients the desired level of freedom to customise the quizzes to their liking, and this is where our project will help out the most. Our website will allow our clients to provide well-structured, randomly generated quizzes that provide good feedback and discourage cheating while encouraging learning and maintaining clinician engagement by keeping the user experience simple and easy.

The value of our work is how it will improve the training and certification experience for clinicians new to TMS. This will help increase the engagement with the study, furthering our knowledge and understanding of strokes and how to help recover stroke victims better.

Tools

TOOL	AREA OF USE	DESCRIPTION
.NET 8	Backend	.NET is an open-source, cross-platform development platform supported by Microsoft. It is used to create web applications, mobile apps, desktop apps and game development. It is written in the C# language
Microsoft SQL Server*	Backend	SQL (Structured Query Language) is a programming language that allows data storage in tables. It is of a relational database form where the rows and columns of a table represent the relationships between data.
SQLite*	Backend	SQLite is a small and fast SQL database that offers the full functionality and stability of a standard SQL database while taking up less disk space.
React	Frontend	React is a user interface development platform built using a syntax extension of JavaScript. It implements the HTML markup language, allowing complete freedom and flexibility when designing UIs.
Figma	Frontend	Figma is an online user interface design tool. It allows users to design the various pages of a tool and then implement a basic navigation scheme, allowing basic UX testing.
AWS	Deployment	AWS (Amazon Web Services) is a cloud computing service run by Amazon that allows users to deploy and maintain an application quickly. AWS offers storage, networking, load balancing, security, server hosting, and many more services.
Git/ GitHub	Version Control & Code Sharing	Git is a version control tool that allows users to control and distribute various stages of application development to other developers. GitHub is the online Git repository for a project, providing an easy-to-use interface for code sharing.
Jira	Project Management	Jira is a powerful project management tool, allowing charts and timelines to be produced and maintained as development continues.

*SQLite was used as the initial database service in the initial construction of the backend. It was subsequently replaced in the final build by Microsoft SQL Server.

Specifications

Our client initially used Google Forms for clinician certification quizzes. They found this assessment method inefficient because admins had no organisation, and clinicians taking the quiz had no feedback on incorrect questions. After considering their needs, they decided to ditch Google Forms, which clogged emails and lacked organisation. Google Forms doesn't give users immediate feedback on incorrect questions, so they wanted a simpler solution. They also wanted admins to collect detailed question attempt data and identify frequently incorrect questions. This would help analyse and improve quiz content, improving clinician training and certification.

In creating the quiz platform, we ensured the implementation of appropriate user requirements and functionality to accommodate working clinicians' busy lives, including users and admins. However, there were some features we did not implement but provided alternative features to accommodate this change:

- Incorporated a range of quiz questions that not only include multiple choice but also randomising questions with every attempt to maximise the learning experience of clinicians. We originally wanted to implement drag-and-drop questions; however, due to technical difficulties, we allowed the option to upload images to quiz questions instead.
- The platform includes intentional roadblocks, such as unlocking quizzes once they score a particular grade in a practice quiz within the same module, to encourage learning without skipping through content.
- The platform gives clinicians feedback on incorrectly answered questions to help them understand their mistakes. Detailed feedback is given for practice quizzes to assist their learning; however, final quizzes provide limited feedback to measure their knowledge without assistance.
- The admin dashboard includes a settings page that allows users to change their details, make new admins, access a list of all registered admins and their contact information, and add new disciplines and sites to the database for user registration.
- The Admin dashboard allows admins to create, edit and delete questions from each module. It also allows admins to search for clinicians and access their details while enabling them to change the user's details and make them certified or not certified.
- Admins can see data related to the pass accuracy of each question within each module. Admins can also access clinicians' profiles; they can access information on their individual progress on quizzes attempted by the user.
- The platform is accessible via a browser on any device.

Use Cases

To accommodate the project specifications, provided below are use cases to demonstrate how the applications will be used by clinicians and admins while also providing alternative flows to demonstrate invalid requests.

Use Cases - Users (Clinicians)

Use Case: Registering User

Actor: Unregistered User

Goal: To register for the quiz platform and access the quiz

Preconditions: The user must have a registered account.

Main Flow:

1. The user navigates to the registration page "Start Quiz."
2. The user enters their full name and email and chooses the site and discipline they are affiliated with.
3. The system will add their details to the database
4. The system grants access and redirects users to the quiz dashboard.

Alternate Flows:

- If the user's email is in use, they'll be redirected to the login page
- if the user's email is invalid, they'll be prompted to re-enter a valid email

Postconditions: The user is logged into the system and can access the quiz dashboard.

Use Case: Completing Quiz as a User

Actor: Logged in User

Goal: To attempt quizzes on the user dashboard

Preconditions: The user must have a registered account.

Main Flow:

- The user clicks on the module and may only click on the practice quiz.
- Once the practice quiz is complete with 80%, they can attempt the final quiz
- They will do this for all six modules.
- Once done, they'll be notified that they can now do the practical.

Alternate Flows:

- If the user does not complete the practice quiz with 80%, they cannot proceed to the final quiz until they re-attempt the practice with 80%
- if the user does not complete the final quiz with 80%, they will be prompted to re-attempt before receiving a notification

Postconditions: The user has completed the theory quiz and can move on to the practical quiz on their site.

Use Cases - Admins

Use Case: Accessing Admin Settings

Actor: Logged in Admin

Goal: To attempt to use the functions in the settings tab

Preconditions: The user must have a registered admin account.

Main Flow:

1. The Admin will select the settings tab and may navigate to the four main functions.
2. The admin can change their details.
3. They can register new admins.
4. Add new sites and disciplines for users to select from registration
5. See a list of all admins with their contact details

Alternate Flows:

- if the email an admin is trying to use for their profile is already in use, they will be notified that it is in use. Same for making a new admin

Postconditions: The admin will have their details changed, add a new admin, and add new sites and disciplines.

Use Case: Searching for a Clinician

Actor: Logged in Admin

Goal: Access Clinician profiles

Preconditions: The user must have a registered admin account

Main Flow:

1. Admin clicks on the clinician tab and can dynamically search for clinicians
2. Admin clicks on relevant clinicians and can see their details
3. They can change their details where needed
4. They can change their certified status to certified or uncertified
5. Access statistics on modules a clinician has passed or failed
6. Access certificates for each module passed

Alternate Flows:

- If the clinician doesn't exist, nothing will show

Postconditions: The admin has successfully changed the user's details and can certify them if they've completed the practical component.

Use Case: Adding, Editing and Deleting Questions in Each Module

Actor: Logged in Admin

Goal: Attempt to add a question to a module

Preconditions: The user must have a registered admin account

Main Flow:

1. The admin clicks on questions
2. The admin clicks on the module, and they want to add a new question to it.
3. They can search for questions and click on them to delete or edit them
4. They can use the filter option to filter questions by topic
5. They can click on the add button to make a new question
6. When adding a new question, they can select as many answers as they want, select the correct ones and add feedback.

Alternate Flows:

- If the admin attempts to make a question with one answer option, they will be prompted to add another one

Postconditions: The admin's changes will be reflected immediately in the quizzes.

Design

Figure 1: *Colour Pallet*

We have used a very bland colour palette to make the VERIFY colours stand out. We originally had an orange to contract with the teal, but after stakeholder feedback, we decided to keep it simple. We did not want to make this website too complicated as our users are mainly busy clinicians. Not having too many distracting colours would benefit the users in completing their tasks. We have a consistent colour scheme on our website. We wanted a dark theme option, but due to time restraints, we could not implement this.

For consistency, most of our pages use the same elements and styling. Text elements are usually styled to the left to ensure eye flow, while important text is centred.

Law of proximity, similarity, symmetry, fate, region, continuity:

Elements that belong together are styled with minimal spacing and separated by a clear div colour. These elements in the same group have the same flow, alignment, and style. Most elements on this page have rounded edges, but the dashboard (Figure 6) has a sharp edge to stand out. Figure 3 shows the quiz feedback's different colours and straight edges to 'attach' it to the question.

We designed this website for simplicity and consistency. Clinicians are busy, so we want them to navigate quickly and clearly. Search, return, and dropdown icons help users navigate faster without guidance. Text size and bold text show hierarchy.

This draws attention to the main element to save users time looking for them through all the text.

When taking the quiz, users can clearly indicate what stage they are in along with previous and next questions (Figure 4.) to allow for user error and flexibility to review their answers before submitting. There are back buttons throughout the webpage to allow the users to return with ease without having to search for it. For important buttons that finalise any changes, such as delete questions, a popup (Figure 2.) comes up to confirm the admin's decision in case of any misclicks.

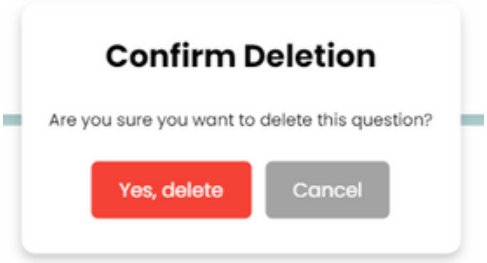


Figure 2: *Confirmation Pop Up*

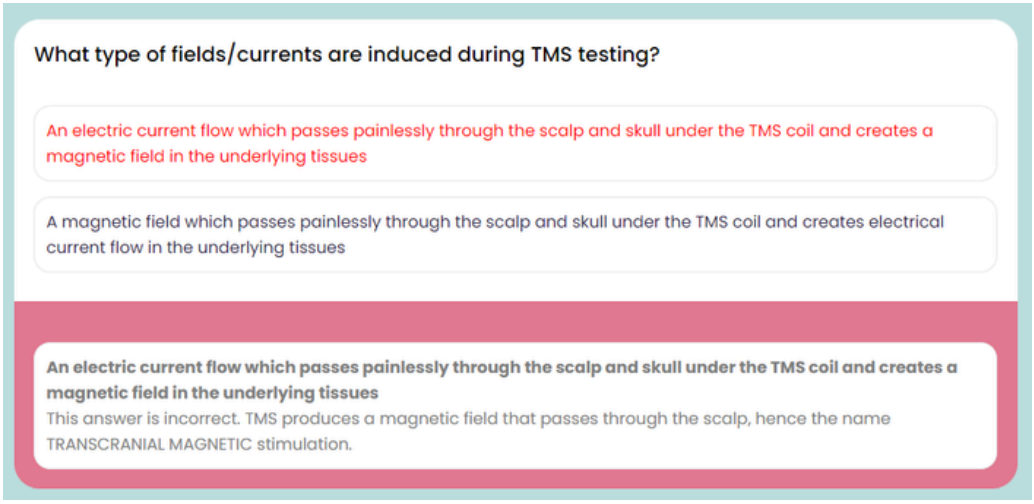


Figure 3: *Quiz Feedback Display*

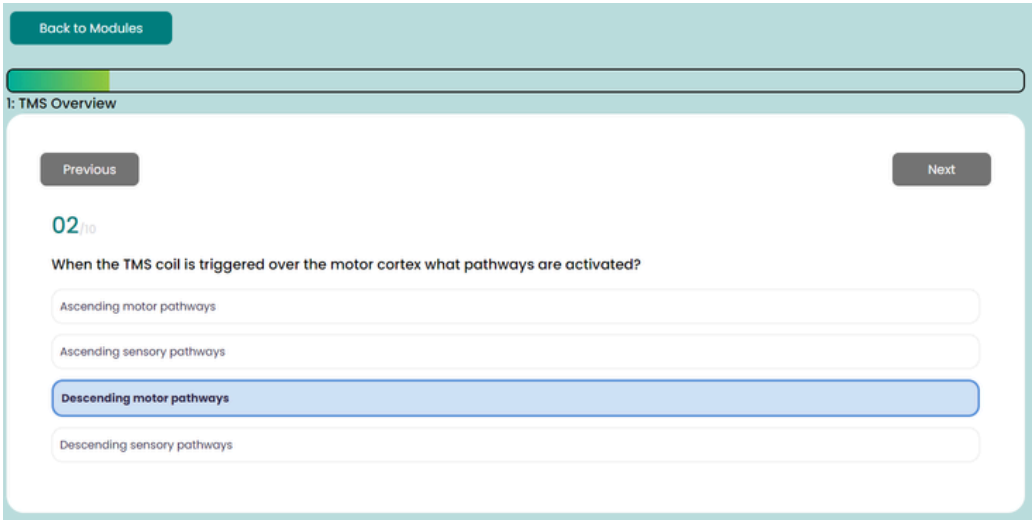


Figure 4: *Selecting Quiz Answer*

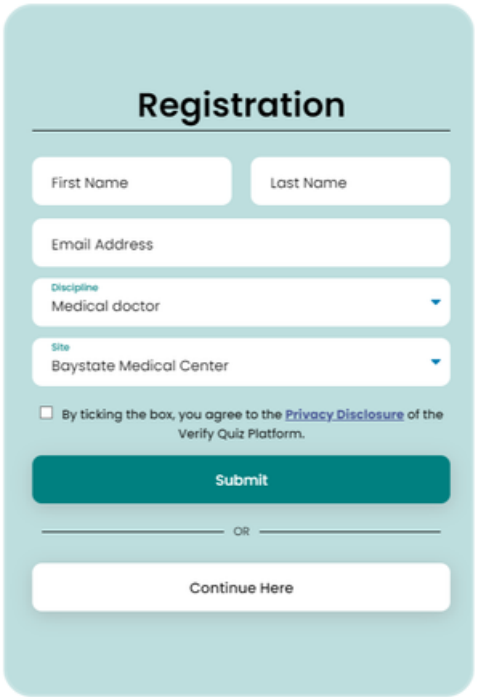


Figure 5: *Clinician Registration*

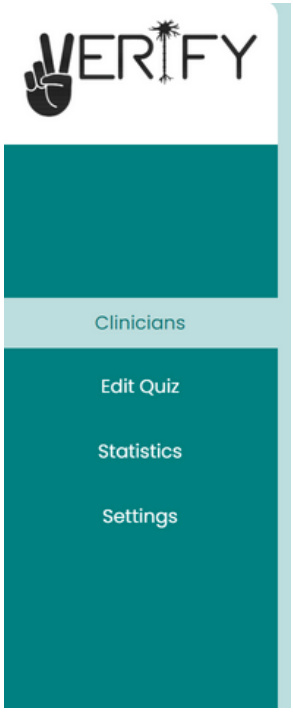


Figure 6: *Admin Navigation Dashboard*

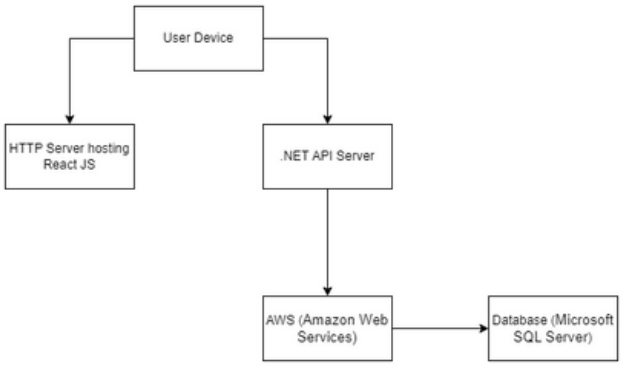


Figure 7: *User Data Flow Interaction*

Figure 7 represents the architecture of our web application. It involves the user interface, API server, and a database. The user's device will interact with the HTTP server hosting the React JS. This server hosts the front-end application which we build with React JS, a Javascript library for creating user interfaces. The user's device will make requests to this server to load the React JS application, which then runs on the user's web browser. Once this is initiated, the React JS application communicates with the back-end through a .NET API server. This server handles all of our logic and process requests from the front end. The user's device sends API requests to the .NET server to perform actions such as retrieving or submitting data. Our .NET API server is hosted on AWS; it provides the infrastructure needed to run the server efficiently and securely. The database is used to store application data and is hosted on the AWS servers as well. The React JS application interacts with the .NET server using API endpoints with multiple different methods (GET, POST, DELETE, PUT). The .NET server interacts with the Microsoft SQL Server using Entity Frameworks.

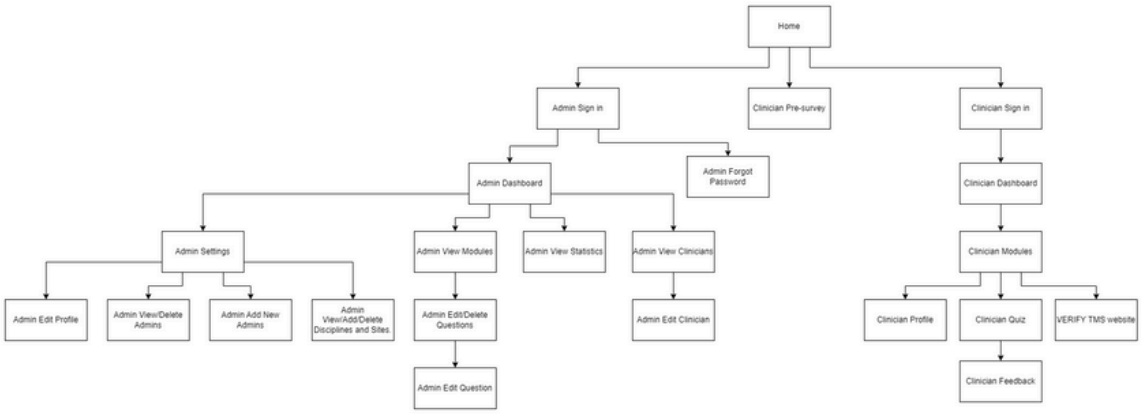


Figure 8: *User Flow and Functionality*

Figure 8 outlines the user flow and functionality within the application. It divides it into two primary user roles: Clinician and Admin. Each of these roles has specific capabilities and navigation paths. Both users start at the Home page and proceed to their respective sign-in pages. After signing in, they will be directed to their respective home pages with their dashboards, where they can access various features and functionalities based on their roles. This chart provides a clear overview of the application's structure, highlighting the different paths and capabilities available to Admins and Clinicians.

Figure 9 shows the Entity Relationship Diagram (ERD) of our database. We constructed the database using models in the .NET project and migrated the database updates to the live server using Entity Framework. This allows us to use classes in C# as instances of objects and use Entity Framework tools to interact with the database.

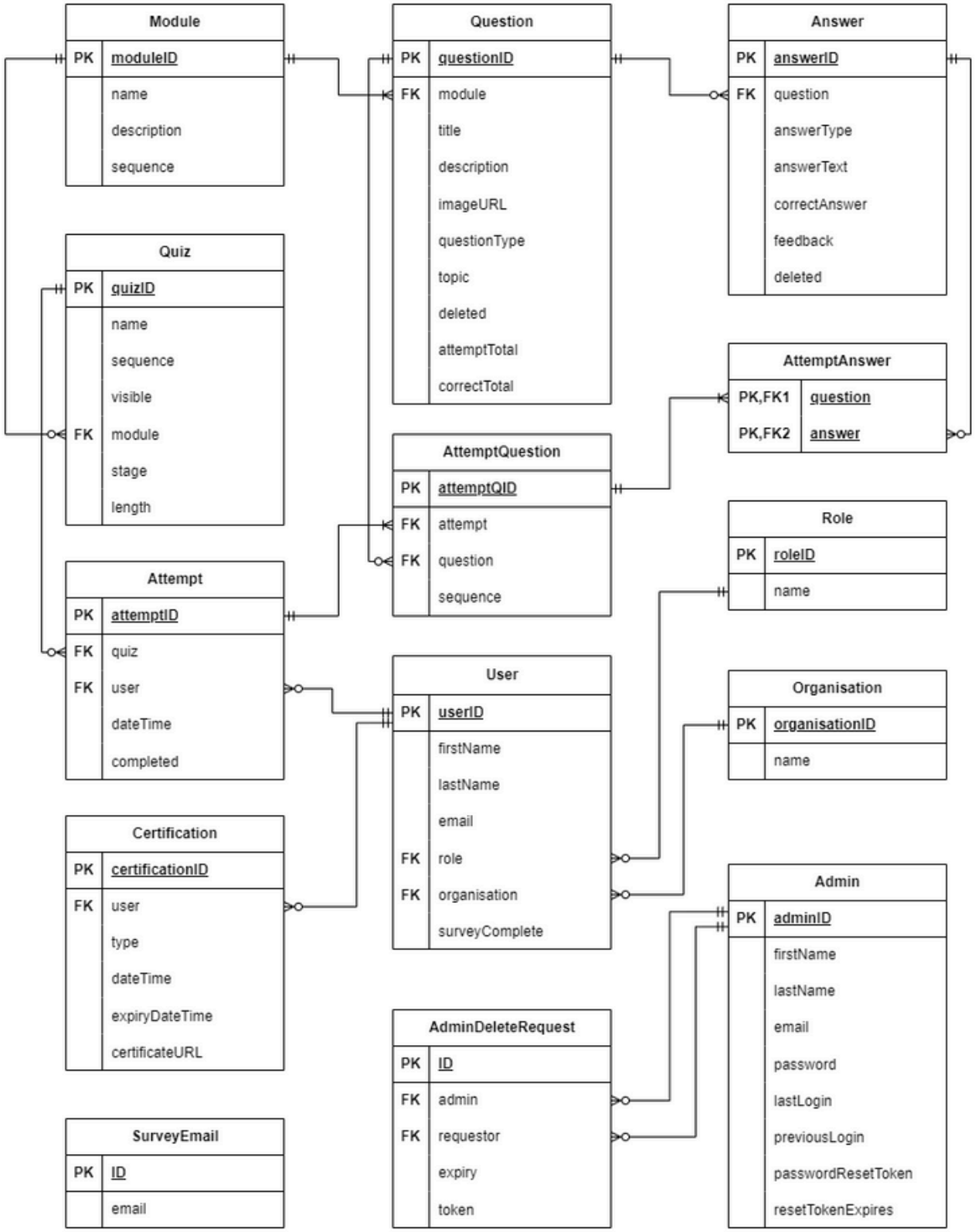


Figure 9: *Entity Relationship Diagram*

Implementation

Implementation:

Firstly, our design was created using Figma. We constantly discussed with our clients to get their specifications and changed things accordingly. Our team implemented the quiz platform using React and .NET 8 and SQL & SQL Lite for the backend. We then used AWS to deploy our project. We useState and Redaxios to handle our API calls for our state management, and React Router was used for our navigation.

Architecture:

The web app follows a component-based architecture, which we researched as normal for React applications. We organised our codebase with separate hooks for state management and services for API interactions. This modular approach helped us to maintain clarity and separation of concerns. We used Github to maintain our separate work and implemented a restriction on the main branch so that two people must allow a merge commit to happen. This further ensured a sense of ease in case we did have some harrowing changes done to the main branch. Furthermore, by allowing each person to have their own personal branch, we all experimented how we saw fit!

Components and Features:

useState managed the various states on our different pages. For example, the main quiz page was for questions, active question index, selected answers, and quiz results.

useState managed the various states on our different pages. For example, the main quiz page was for questions, active question index, selected answers, and quiz results.

Data fetching was done through different API endpoints, requiring a certain token (clinician or admin) for security. Each detail about the user, modules, questions, etc., was done through secure API calls to keep the web app as dynamic as possible in case of any future circumstances requiring our clients to add or delete certain features to the website.

Data Flow:

Our data was managed on the front end using local React states, with hooks for handling asynchronous data fetching and state updates. API interactions follow RESTful principles, enabling data retrieval and submissions.

Styling and User Interface:

To style the web application, we used CSS modules. This ensured a consistent and responsive design throughout the application. Conditional rendering and styling allowed us to provide visual feedback on both the admin and clinician side and enhanced the user experience.

Unforeseen problems:

We had some initial challenges with integrating APIS and handling asynchronous data flow. To remedy this, we set up a swagger to test the APIS. We used extensive console logging to debug and understand the APIS response structure, allowing us to update our state management logic accordingly and easily.

Another big challenge was dealing with CORS errors. This was a tough one to wrap our heads around, but with lots of research, we learnt how to bypass this and understand what the different types of CORS errors really mean. We could decipher when it was a security issue and when the backend of our database was missing values, which led to this error.

Managing API errors and redirects for unauthorised access was crucial for a smooth user experience. Our solution was to check for the different error response codes in every API we call to deal with them accordingly and even implemented a function on the necessary pages called “handleErrorResponse” to handle specific HTTP status codes and navigate the user appropriately, including removing their tokens from the session storage if it was now invalid and redirecting them to their respective login page.

These challenges provided valuable lessons in API integration, state management, and error handling. Our team learnt to adapt by refining our development practices and improving our testing workflows to ensure a smooth implementation process. We did this by having near-weekly meetings to help each other and ensure everyone was consistently on the same page.

Results & Evaluation

Implementation:

Firstly, our design was created using Figma. We constantly discussed with our clients to get their specifications and changed things accordingly. Our team implemented the quiz platform using React and .NET 8 and SQL & SQL Lite for the backend. We then used AWS to deploy our project. We useState and Redaxios to handle our API calls for our state management, and React Router was used for our navigation.

Architecture:

The web app follows a component-based architecture, which we researched as normal for React applications. We organised our codebase with separate hooks for state management and services for API interactions. This modular approach helped us to maintain clarity and separation of concerns. We used Github to maintain our separate work and implemented a restriction on the main branch so that two people must allow a merge commit to happen. This further ensured a sense of ease in case we did have some harrowing changes done to the main branch. Furthermore, by allowing each person to have their own personal branch, we all experimented how we saw fit!

Components and Features:

useState managed the various states on our different pages. For example, the main quiz page was for questions, active question index, selected answers, and quiz results.

useState managed the various states on our different pages. For example, the main quiz page was for questions, active question index, selected answers, and quiz results.

Data fetching was done through different API endpoints, requiring a certain token (clinician or admin) for security. Each detail about the user, modules, questions, etc., was done through secure API calls to keep the web app as dynamic as possible in case of any future circumstances requiring our clients to add or delete certain features to the website.

Data Flow:

Our data was managed on the front end using local React states, with hooks for handling asynchronous data fetching and state updates. API interactions follow RESTful principles, enabling data retrieval and submissions.

Styling and User Interface:

To style the web application, we used CSS modules. This ensured a consistent and responsive design throughout the application. Conditional rendering and styling allowed us to provide visual feedback on both the admin and clinician side and enhanced the user experience.

Test Plan

Testing	Main Flow	Notes/bugs/improvements
General Home		Back Home button's clickable area is incorrect. Does not correlate to the text or the arrow button. Scaling issues.
Presurvey	Input for email with correct formatting.Popup displays and closes correctly.	No way to return back to the main menu or to just sign in.New field to be entered does not fit on the page when created, selecting Other for discipline.First and last name fields displaying incorrectly on windows, no gap.
Clinician Sign in	Only verified emails can login.	
General Clinician	Sign out button logs users out and links them to the sign in page.	
Clinician Dashboard	Links works correctly	Buttons do not give indication of what page the user is currently on.
Quiz Dashboard	Modules correct and final quiz buttons link to the correct quizzes.Symbol status displaying correctly on passed and unpassed quizzes.	Indicate the modules are clickable. Add a reason when the user doesn't have access to the final quiz.Very cluttered.
Clinician Profile	Details displaying correctly	Make verify email functional. Details are too small and hard to read.
General Admin	Last login status has correct data and time.	Requires indication for buttons/containers that are clickable.
Admin sign in	Logs correct users with the correct password.	
Admin Password	We use Bcrypt to salt the password and then store the hashed value in our database.	

Test Plan

Testing	Main Flow	Notes/bugs/improvements
Admin: Clinician search	Search function works correctly	Allow to search by email and position.
Admin: Clinician profile	Details are updated correctly and stored.	Change dropdown to button change instead for training status and pre-survey override. Some notification for when changes are made.
Admin: Edit quiz	Changes are stored correctly. Final confirmation texts showing.	Buttons and fields don't fit correctly. Display the topic of each question.
Admin: Statistics	Statistics are being ordered correctly and the correct corresponding tags are being filtered.	
Admin Settings	Fields are searching correctly, editing, deletion and addition working and all saving. Email mask working on admin side as well.	Unscrollable admin list on windows.

Future Works

We worked with the clients to categorise features as necessary and desirable. Throughout the build process, we have focused on ensuring that the required features work before moving on to the nice-to-have features as time allows. We were unable to implement some of these features due to time constraints.

One such feature allows users to recall past answers. We initially designed our database to store each selected answer against each attempt, but after significant development, we discovered that the attempted database's foreign keys were incorrectly configured, so we had to abandon that idea rather than reconfigure our database structure, which could cause more problems. We have included statistics with each attempt, such as average quiz, question passes, etc. Statistics could be improved to help our clients understand each attempt. ttempt.

Another task that could be implemented in the short term is the ability to send reminder emails to clinicians 11 months after certification, reminding them that they are now eligible to take the recertification quiz. Users would have to set their own reminders to complete the recertification after 11 months. However, we believe that adding an automatic reminder would help ensure that each clinician has the same experience.

As we developed our web application, we became aware of scaling issues for various screen sizes. We began development with both desktop and mobile screen sizes in mind. Nonetheless, because the front-end team encountered issues with React and scalability, we decided to prioritise developing for desktop first, as most clinicians will likely use this to take the quizzes. We have added scalability for various laptop and desktop screen sizes; however, to optimise for mobile, we will need to reduce the sidebar and restructure the quiz pages when a smaller screen size is detected.

Another feature that our clients have previously requested for future work is the ability to drag-and-drop answers. This would allow a clinician taking a quiz to drag and drop responses into sentences and images. We've also considered supporting short- and long-answer questions, but clients can't manually mark these. Another option is to mark the answers with AI, which would necessitate training a model.

Clinicians receive pre- and post-TMS training surveys from clients. Google Forms hosts these. Due to the variability in question-and-answer types and Google Forms' well-developed insights, we did not think adding surveys to our application was necessary. We integrate with their pre-training survey to require clinicians to complete it before continuing on our site, but hosting the surveys alongside the quizzes may be desirable in the future.

Conclusion

Our clients wanted to move away from using Google Forms as a quiz platform to one that fosters a more enriching educational experience for users participating in these quizzes. We aimed to implement functions that allow clients to track each registered user's quiz progress, modify their quizzes freely, and manage platform accessibility as needed. This shift is especially critical in the context of the VERIFY study, which focuses on the recovery of stroke patients through Transcranial Magnetic Stimulation (TMS).

The development of the quiz platform demanded extensive data management capabilities to handle user registrations, quiz administration, and real-time feedback processing. Since our clients and users are busy clinicians in the health sector, the platform's design and user-friendly interface were most important. We achieved a clean, easy-to-navigate design that enables users to concentrate on learning without being hindered by complex features. The platform's ease of use significantly improved quiz organisation, making it more straightforward to track user progress and manage quiz content efficiently.

This quiz platform's creation has revolutionised how our clients assess and engage with their users. By providing immediate feedback on quiz completions, the platform significantly enhances the learning experience, fostering a more interactive and engaging environment. The improved organisational and management capabilities ensure that clients can efficiently handle large volumes of data, making the VERIFY study more effective and impactful.

This platform is not just a technological advancement; it is a crucial tool in supporting the VERIFY study's mission to assess and improve the recovery of stroke patients through TMS. Our clients now have the tools to expand their research and impact across New Zealand, Europe, America, and Australia. This technology empowers them to spread their cause globally, potentially transforming the lives of stroke patients everywhere. By leveraging this advanced platform, our clients can push the boundaries of medical research and clinical practice, ultimately contributing to better outcomes for stroke recovery worldwide.

References

1. Duthie S. Great Quotes from the Digital Health Summer Summit [Internet]. www.linkedin.com. 2014 [cited 2024 Mar 13]. Available from: https://www.linkedin.com/pulse/20140710175631-2426944-great-quotes-from-the-digital-health-summer-summit/
2. World Stroke Organisation. Impact of Stroke [Internet]. World Stroke Organization. [cited 2024 Mar 16]. Available from: https://www.world-stroke.org/world-stroke-day-campaign/about-stroke/impact-of-stroke#:~:text=Over%20100%20million%20people%20in
3. Stroke Foundation New Zealand. Facts about stroke in New Zealand [Internet]. www.stroke.org.nz. Available from: https://www.stroke.org.nz/facts-and-faqs#:~:text=Stroke%20is%20New%20Zealand

Table of Authorship

Section	Author(s)
Executive Summary	Ayesha Essop
Introduction	Bailey Alexander
Background	Bailey Alexander
Project Specifications	Ayesha Essop
Project Design	Ayesha Essop & Emily Zhao
Project Implmentation	Lucas Fernandes Tavares & Sahil Kumar
Results And Evaluation	Emily Zhao & Lucas Fernandes Tavares
Future Works	Angus Wright
Conclusion	Ayesha Essop

Appendices

GitHub: https://github.com/uoa-compsci399-s1-2024/capstone-project-2024-s1-team-3-ghost-code/
Demo: https://youtu.be/IDDmtNcA6Jw
Gantt Chart (Revised):

Appendices

Figure 1: *Colour Pallet*

Figure 2: *Confirmation Pop Up*

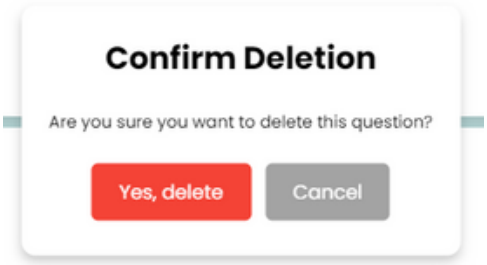


Figure 3: *Quiz Feedback Display*

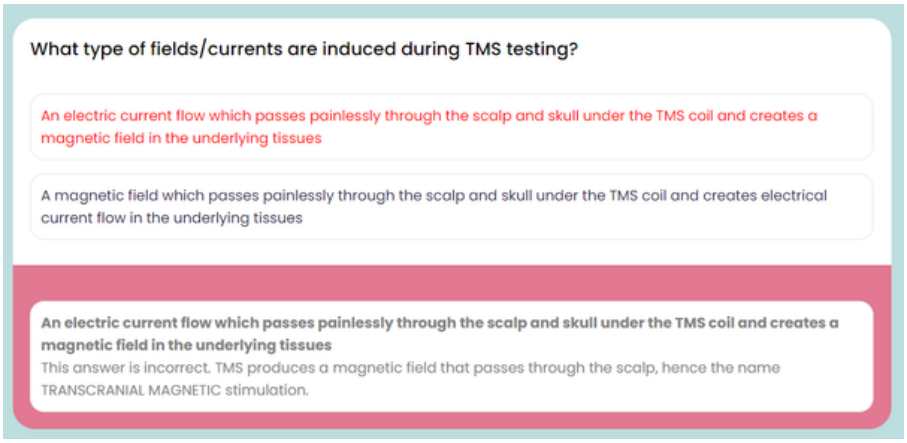
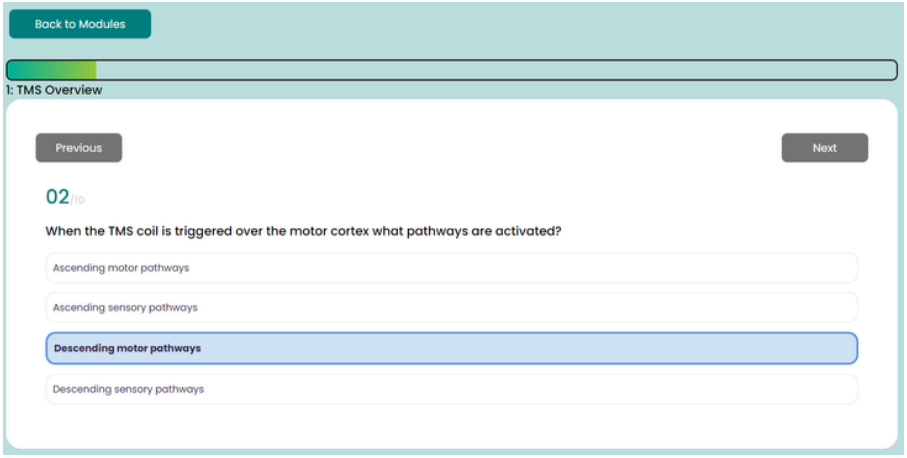


Figure 4: *Selecting Quiz Answer*



Appendices

Figure 5: *Clinician Registration*

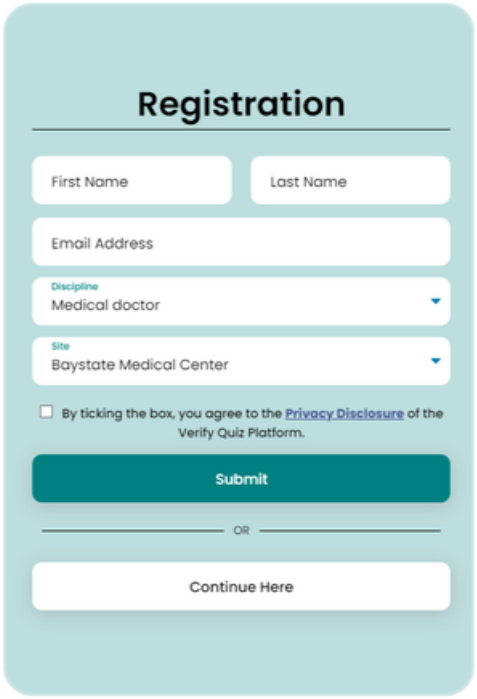


Figure 6: *Admin Navigation Dashboard*

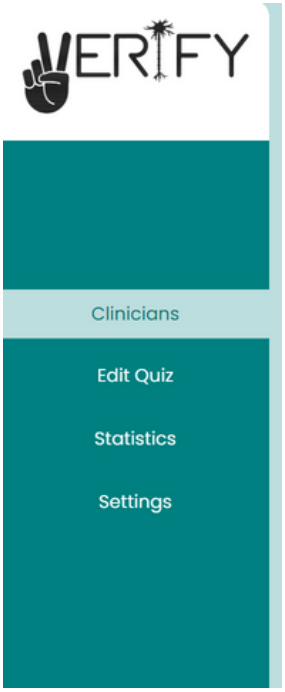


Figure 7: *User Data Flow Interaction*

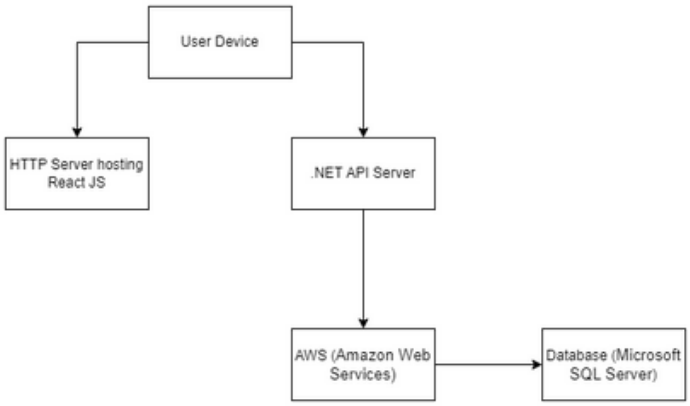
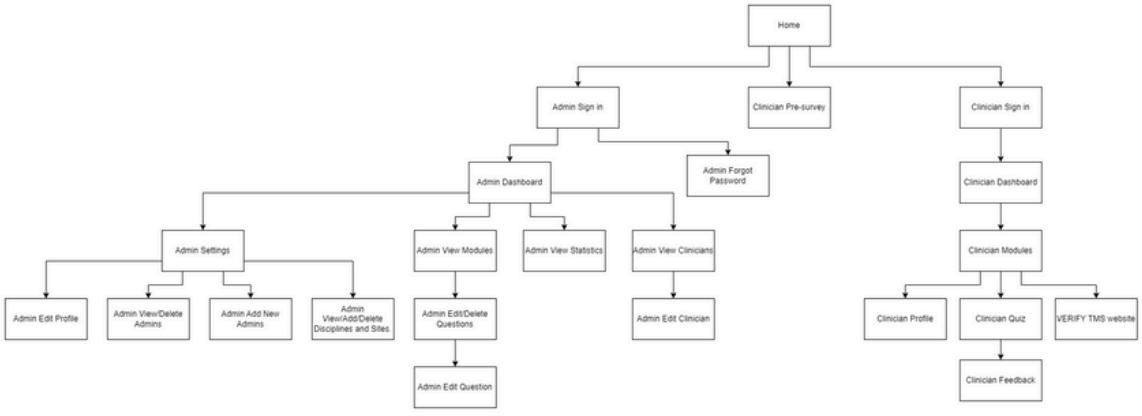


Figure 8: *User Flow and Functionality*



Appendices

Figure 9: *Entity Relationship Diagram*