WELLSPACE – A WEBSITE FOR WELLBEING AND MENTAL HEALTH

Nilangi M. S. Edirisinghe (A1882259) – Team 22 N
The University of Adelaide, South Australia.

ABSTRACT

The WellSpace project encompasses the development of a comprehensive and user-friendly web application that integrates peer support, professional counselling and educational content to provide accessible mental health support. Key elements include a blog with interactive discussion features, informational content, facilities to book counselling services, engaging quizzes and opportunities for donations and volunteering. Overall, WellSpace is a secure and innovative platform dedicated to promoting mental health wellness.

1. INTRODUCTION

The prevalence of mental health problems among children and adolescents is significant, with 14% of those aged 4 to 17 years in Australia experiencing such difficulties. Despite this, only a minority receive professional help. For instance, only 25% of children with mental health problems had accessed professional services (Giesen, Searle & Sawyer 2021, p.785). Not only that, the COVID-19 pandemic has further exacerbated mental health issues globally. During the first seven months of the pandemic, the prevalence of depression was 20%, anxiety 35%, and stress 53% (Lakhan, Agrawal & Sharma 2020, p.519). These statistics highlight the growing need for accessible mental health resources.

Given the extensive reach and efficiency of universal interventions, which can be delivered to large populations without stigmatizing participants, there is a critical need for platforms that offer comprehensive and universally accessible mental health support, without boundaries. Universal interventions are particularly advantageous as they can reach a broad audience at a relatively low cost, thereby addressing the shortcomings of targeted clinical interventions which often fail to cover the larger population due to their high cost and limited reach (Giesen, Searle & Sawyer 2021, p.785).

WellSpace has been developed to address this need by providing a comprehensive platform that combines peer support, professional counselling and educational resources. Unlike other mental health websites, WellSpace offers a unique blend of services. For example, 'BetterHelp' focuses on online professional counselling (BetterHelp 2024), and 'TalkCampus' provides peer support (TalkCampus 2024). In contrast, WellSpace integrates both services. Additionally, 'HeadSpace' is known for its mindfulness resources (HeadSpace 2024) but lacks the broader range of services that WellSpace offers, including counselling and educational content.

Recognizing these gaps in the market, WellSpace aims to create a safe online environment where individuals can connect with peers, access professional counselling, engage in quiz challenges for reflection and educate themselves about mental health. Our goal is to alleviate the stigma surrounding mental health and provide a universal solution for those in need. By catering to both the public and certified mental health experts, WellSpace ensures comprehensive support and fosters a community focused on mental well-being.

2. PROJECT AIMS

The WellSpace project aims to achieve the following key objectives:

1. Develop a comprehensive web platform to provide accessible and universal mental health support.

2. Foster a supportive community for mental well-being: Establish a blog with discussion features to enable user communication and interaction, creating a supportive and safe community environment.

3. Enhance User Engagement and Reflection: Feature engaging challenges and quizzes that promote user interaction, reflection and enhance mental well-being.

4. Provide Booking Services for Counselling: Create a secure and efficient system for online consultations with mental health experts, ensuring users can access care promptly.

5. Ensure privacy and credibility: Prioritize privacy of the mental health help seeking users and ensure that the experts are verified, and the content is credible.

3. APPROACH

3.1 Software Architecture and Technologies Used

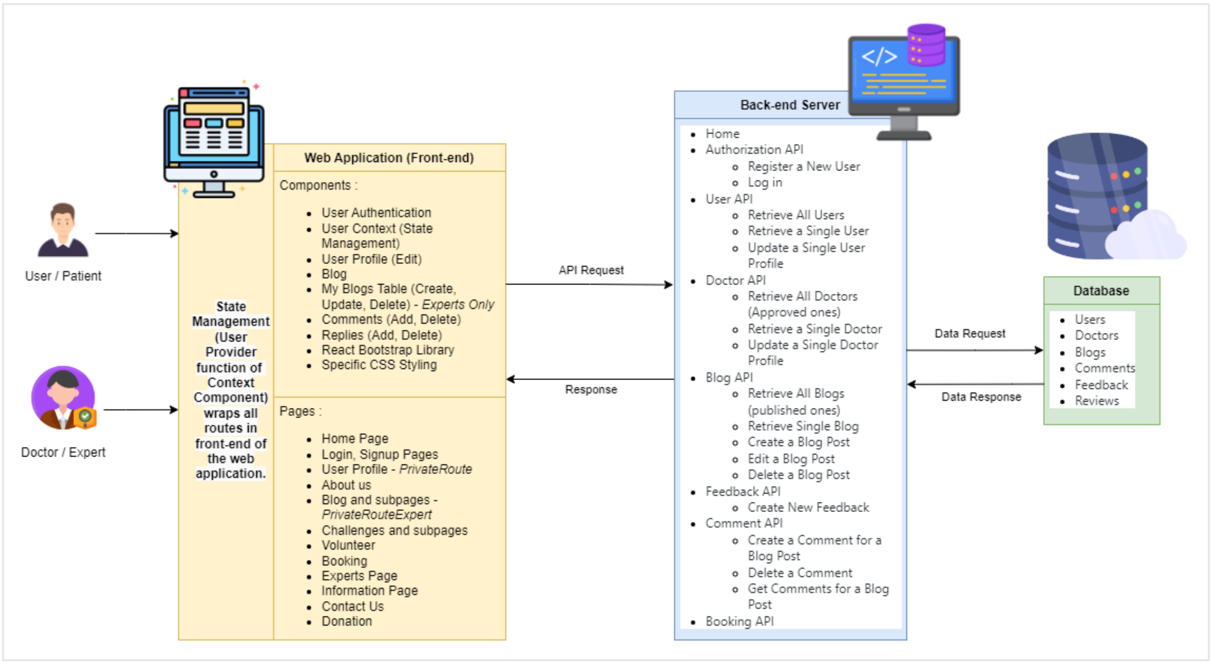


Figure 1: Software Architecture of WellSpace Web Platform.

The software architecture of WellSpace comprises three main layers:

1. **Web Application /Client side (Front-End)**: Developed using React.js, this layer includes various pages and components such as Home, About Us, Blog, Challenges, Booking, Volunteer, Profile, and Contact Us. It also features user authentication, state management using UserContext, and private routes for restricted access.
2. **Web Server (Back-End)**: Built with Node.js and Express, the backend provides APIs for user authentication, blog management, user profiles, doctor profiles, feedback, comments, and booking appointments. The APIs are integrated with a MongoDB database hosted on a cloud platform.
3. **Database**: The database layer uses MongoDB to store data related to users, doctors, blogs, comments, feedback, and reviews.

Underlying Technologies

- **Front-End**: React.js, CSS, Bootstrap, Owl Carousel, Animate.css, React Quill Library for Rich Text Editing, uploadCloudinary Library for Uploading Images and React-Icons Library.
- **Back-End**: Node.js, Express.js, MongoDB, Mongoose for Database Schema Management.
- **State Management**: React Context API / UserContext Library.
- **APIs**: RESTful APIs using ‘Axios’ library for interaction between front-end and back-end.

3.2 Project Flow

The following diagram displays the flow of the WellSpace project.

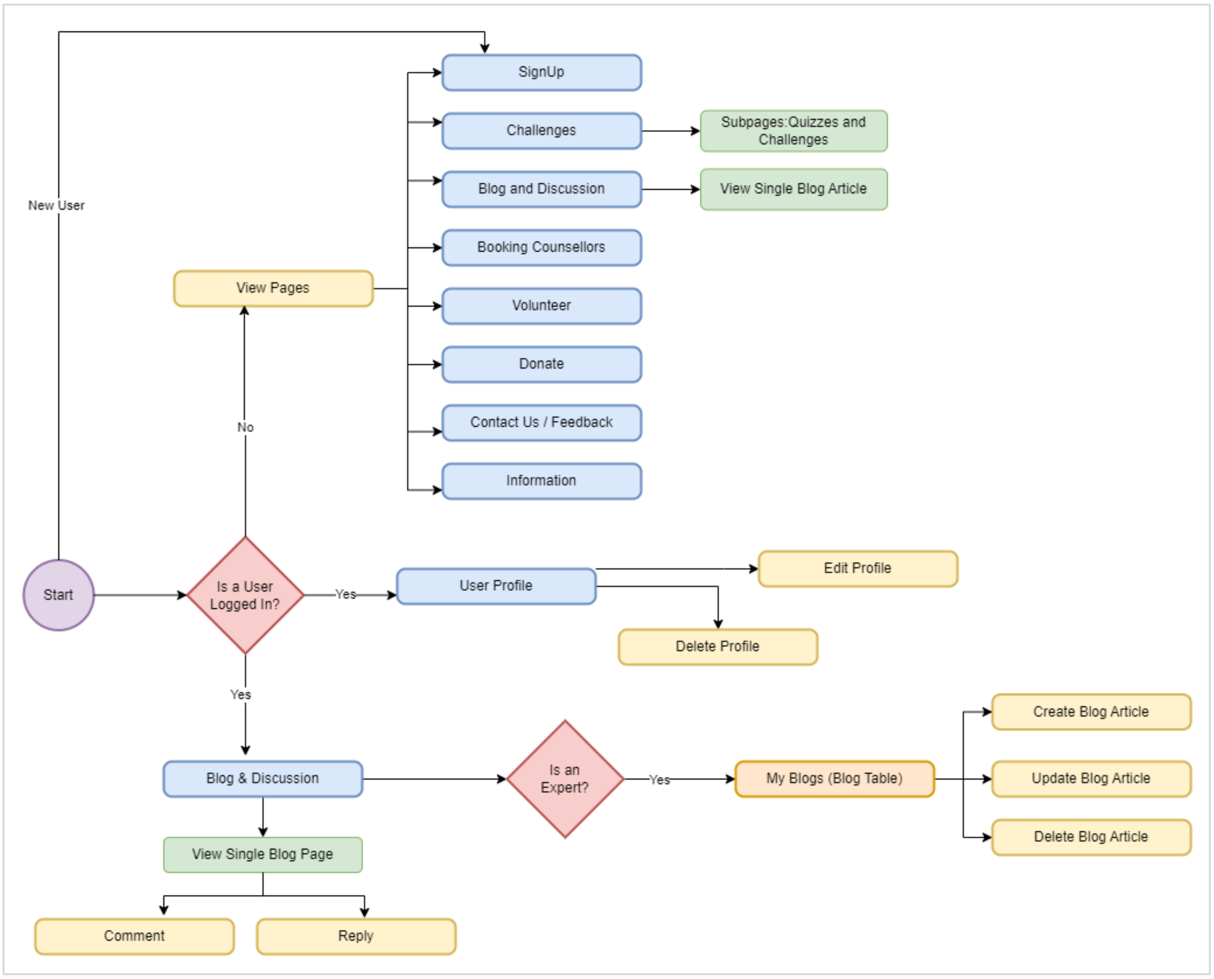


Figure 2: The Process Flow Diagram of the WellSpace Website.

The project flow reflects the several user scenarios of the project where there is a new user, a visitor to the website who has not signed up or logged in, a logged in user and a logged in expert. Not only that, but the above diagram also represents the pages and components which are accessible or restricted to the above-mentioned types of users and how they would navigate accordingly through the web platform.

4. ACHIEVEMENTS/RESULTS OF TEAM 22-N

Essentially, the achievements of our group, can be listed as follows:

- The completion of all the front-end static pages of the whole website.
- The completion of the ‘Blog and Discussion’. In other words, the complete integration of the Blog system between the frontend and the backend has been successfully achieved.
- The complete ‘User Authentication System’ has been successfully achieved.
- The booking feature where a user can book a counsellor and receive an e-mail notification.
- The completion of all the relevant and necessary backend APIs.
- The completion of the necessary elements of the database system.
- Testing the whole system using unit testing, integration testing and usability testing.

5. INDIVIDUAL CONTRIBUTION AND RESULTS ACHIEVED BY NILANGI EDIRISINGHE (A1882259)

My main contributions to the WellSpace project were focused on the comprehensive development and integration of front-end components, which guaranteed a smooth user interface and robust interaction with the backend services.
In particular, my contributions can be listed as follows:

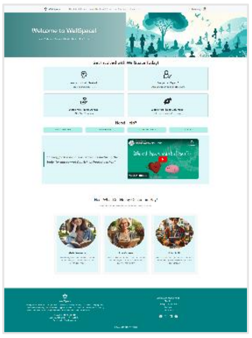
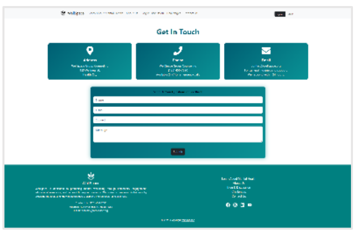
- Design and Development of the Vast Majority of Web Pages from the Front-end
- Adding Bootstrap and Custom CSS Styling to Vast Majority of Front-end pages
- Integration of Blog and its Subpages between Front-end and Backend
- Integration of User Authentication System between Front-end and Backend
- Implementing State Management
- Implementing Private Routes
- Testing the Responsible Parts of the System

5.1 Individual Contribution to Front-End Development

I was responsible for the development of most of the front-end pages of the WellSpace platform. This included designing, styling and developing user-friendly interfaces that are fully responsive across all devices including desktop and mobile. The pages developed are:

- Home Page and its Universal Components (Navigation Bar and Footer)
- Blog and All its Subpages
- Information Page
- Pages for User Authentication and State Management: User Profile Page, Pages for Sign-Up and Log-In
- Main Challenges Page and Mindfulness Quiz Page
- Volunteer Page and Contact Us Page

5.2 Results of Individual Contribution to Front End Development

Page	Description	Screen Capture
Home Page	The Home Page serves as the main landing page for the website. It includes the navigation bar and multiple sections such as a hero section, get involved cards, navigation buttons, featured video section, and user testimonials. Each section is designed to provide a comprehensive overview of the website's offerings and encourage user engagement. Utilizes React components for modular design, specific CSS files for styling and react-bootstrap for styling layout.	
Contact Us Page	The Contact Us Page includes address, phone, and email contact cards, as well as a feedback form for users to send messages. It also features a toast notification upon successful form submission. API Call: POST /api/v1/feedbacks/create to send user feedback. Utilizes axios for API requests and react-bootstrap for UI components.	

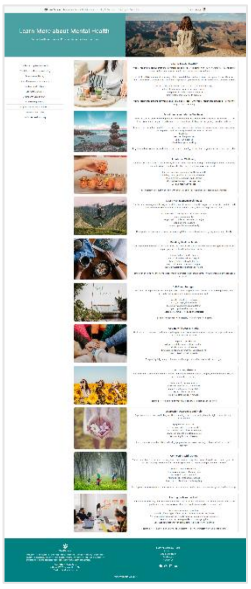
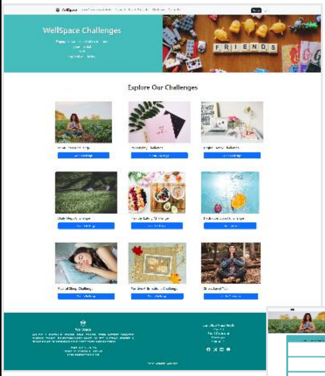
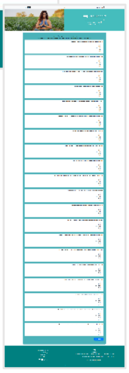
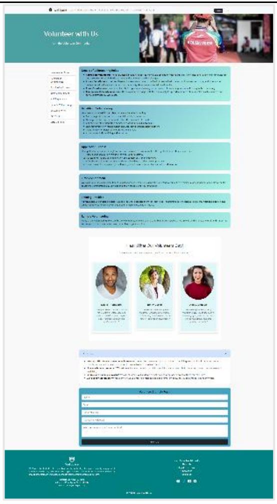
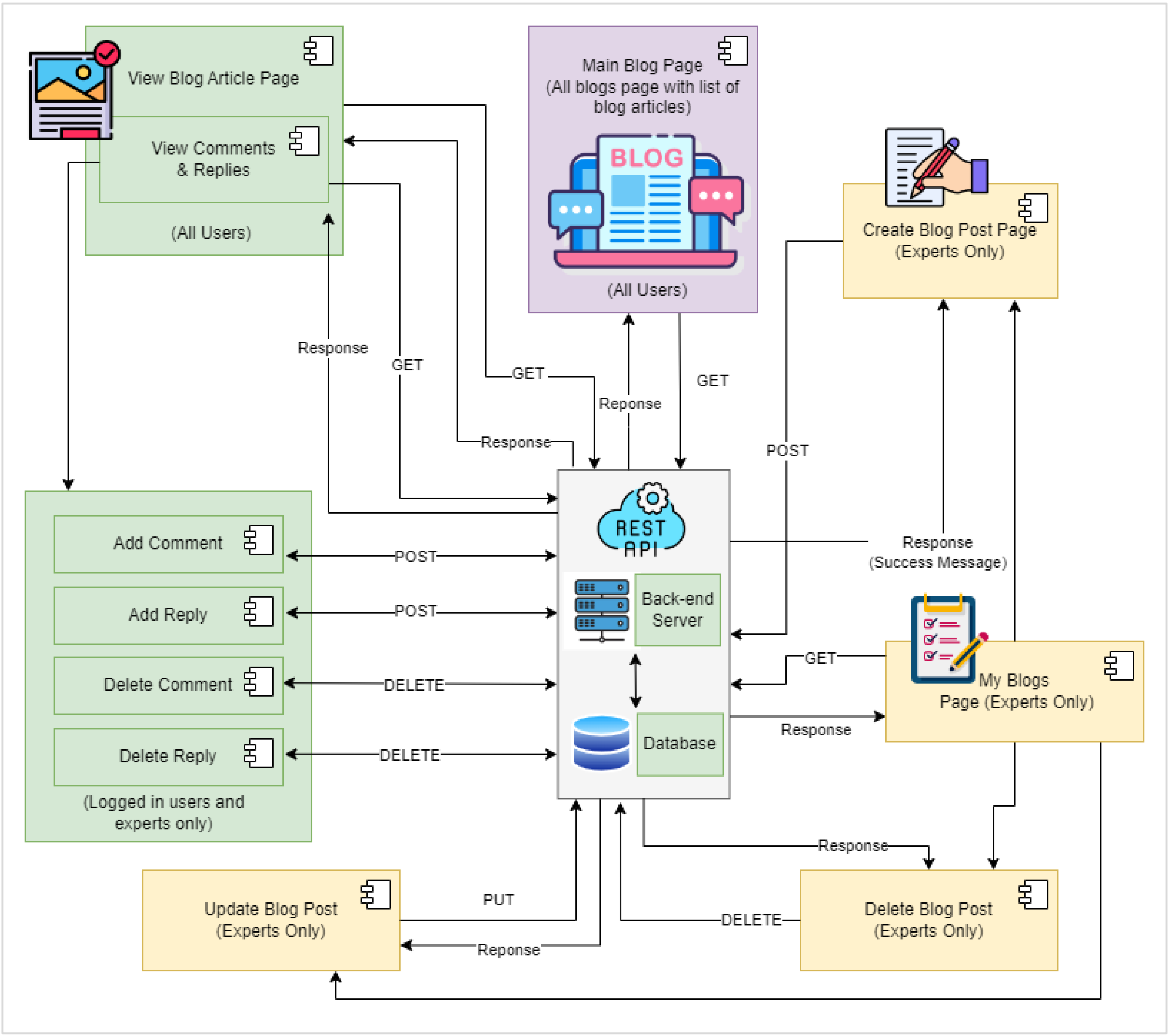
Information Page	The Information Page provides detailed content on various mental health topics. It features a side navigation panel that stays fixed on the left and allows for smooth navigation to each title. Multiple cards displaying information on different topics such as mental health, mindfulness, emotional well-being, and more are present in the page as static content from the front-end itself. Utilizes specific CSS, React components for modular design and react-bootstrap for layout.	
Main Challenges page and Mindfulness Quiz Page	The Main Challenges Page showcases various challenges users can participate in. Each challenge is displayed in a card format with an image, title, and a button linking to the specific challenge page. Utilizes unique CSS, React components for modular design and react-bootstrap for layout. The Mindfulness Quiz Page features a series of questions aimed at assessing the user's mindfulness level. Users can select answers for each question and submit the quiz to receive a score and a grade. Includes a modal to display the results. Utilizes React state management to handle form data, user inputs, and dynamic updates and react-bootstrap for UI components.	 
Volunteer Page	The Volunteer Page provides information about volunteering opportunities at WellSpace. It includes sections on volunteering roles, benefits, application process, time commitment, training, remote volunteering, testimonials, FAQ, and a sign-up form. Features toast notifications upon successful form submission. Utilizes CSS for styling, react-bootstrap for UI components and state management for form handling.	

Table 1: Overview of Front-End Pages

5.3 Creation and Integration of Blog and Its Subpages (Individual Contribution)

I was able to successfully create the front-end pages of the blog system and complete the integration between front-end and back-end of the blog system. This integration was done by implementing REST API calls from the front-end to the relevant back-end routes. Due to the integration, experts are able to create and manage their own blog articles. All other users can view the blog articles written by the experts. Additionally, I created 'comment' and 'reply' functionalities, where both logged in users and experts can interact and engage in the discussion. To protect the privacy of the users their username is shown with comments and replies, instead of their real name, whereas the real name of the expert is shown for credibility.



View Blog Article Page
View Comments & Replies
(All Users)
Main Blog Page (All blogs page with list of blog articles)
BLOG
(All Users)
Create Blog Post Page (Experts Only)
Response
GET
GET
Response
Reponse
GET
POST
REST API
Back-end Server
Database
Response (Success Message)
Add Comment
Add Reply
Delete Comment
Delete Reply
(Logged in users and experts only)
POST
POST
DELETE
DELETE
GET
Response
My Blogs Page (Experts Only)
Response
Update Blog Post (Experts Only)
PUT
Reponse
DELETE
Delete Blog Post (Experts Only)

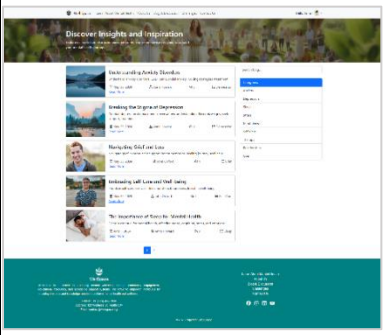
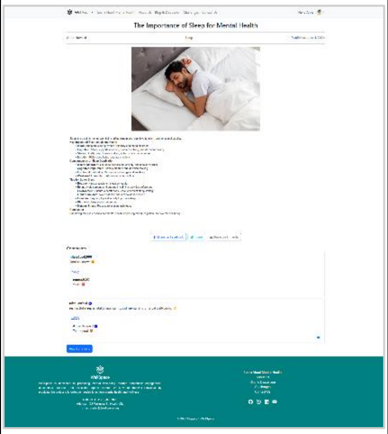
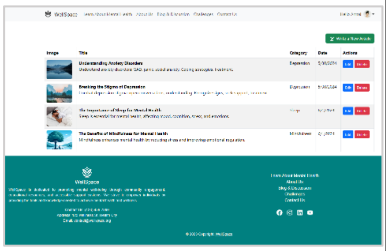
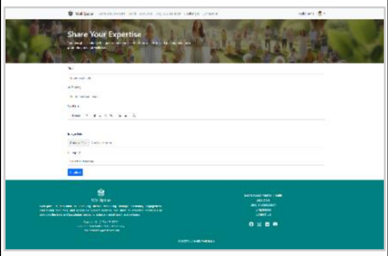
Figure 3: The Integration of Blog and its Components between Front-end and Backend

The above diagram depicts how the integration of blog, and its subpages were undertaken using the RESTful API client requests. The four client requests are as follows:

- GET – To access data via specified route from the back-end server / database.
- POST – To send data vis specified route to the back-end server / database.
- PUT – To update existing data on the back-end server / database.
- DELETE – To remove data from the back-end server / database.

(Amazon Web Services, 2024)

5.4 Results and Underlying Technologies of Blog System (Individual Contribution)

Page Name	Description	Screen Capture
Main Blog Page *(All users can access)*	Displays a list of all blog posts, allowing users to search, filter by category, and navigate to individual posts. The page dynamically updates based on user interactions and state changes. Implements search functionality, category filtering, and pagination for better user experience. API Call: GET /api/v1/blogs?*query*=${searchQuery} to fetch all blog posts with a search query. Libraries Used: axios for API requests, react-bootstrap for UI components.	
View Single Blog Article Page *(All users can access)*	Shows the full content of a single blog post, including comments and replies. Utilizes axios for GET requests to fetch post details. ReactQuill is used to display rich text content. Comments and replies are managed dynamically, with options for users to add, delete, or reply to comments. API Calls: GET /api/v1/blogs/:postId to fetch a single blog post, GET /api/v1/comments/:postId to fetch comments, POST /api/v1/comments/:postId to add a new comment, POST /api/v1/replies/:parentId to add a reply, DELETE /api/v1/comments/:commentId to delete a comment, and DELETE /api/v1/replies/:replyId to delete a reply. Libraries Used: axios for API requests, react-bootstrap for UI components, ReactQuill for rich text display.	
My Blogs Page *(Only experts can access)*	Allows experts to view and manage their blog posts, including options to edit or delete posts. State management ensures that only posts created by the logged-in expert are displayed**.** API Call: GET /api/v1/blogs to fetch all blog posts and filter by expert's ID. The page supports post deletion and redirection for editing. Libraries Used: axios for API requests, react-bootstrap for UI components.	
Create Blog Page *(Only experts can access)*	Provides a form for experts to create new blog posts. Uses axios for POST requests to submit new blog posts. Utilizes ReactQuill for rich text editing and uploadCloudinary for image uploads. The page includes form validation, image previews, and category selection. API Call: POST /api/v1/blogs/:userId to create a new blog post. Libraries Used: axios for API requests, react-bootstrap for UI components, ReactQuill for rich text editing, uploadCloudinary for image uploads.	

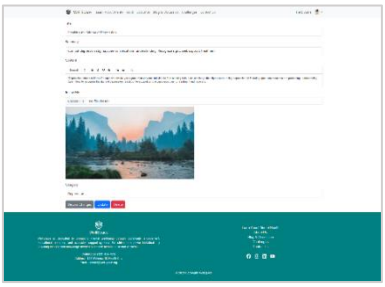
Edit Blog Page *(Only experts can access)*	Enables experts to edit existing blog posts. Uses axios for PUT requests to update posts and DELETE requests to remove posts. Utilizes ReactQuill for rich text editing and uploadCloudinary for image uploads. The page allows updating post details, replacing images, and changing categories. API Call: PUT /api/v1/blogs/:postId to update a blog post, DELETE /api/v1/blogs/:postId to delete a blog post. Libraries Used: axios for API requests, react-bootstrap for UI components, ReactQuill for rich text editing, uploadCloudinary for image uploads.	

Table 2: Overview of Blog Pages

Further, the following diagram represents the process flow of how an expert would navigate through the blog system and perform the functions. This has been presented explicitly here extending upon Figure 2, as the process is unique and private for experts only. For normal and logged in users, the process is the same as depicted in Figure 2 which represents the process flow for the whole application.

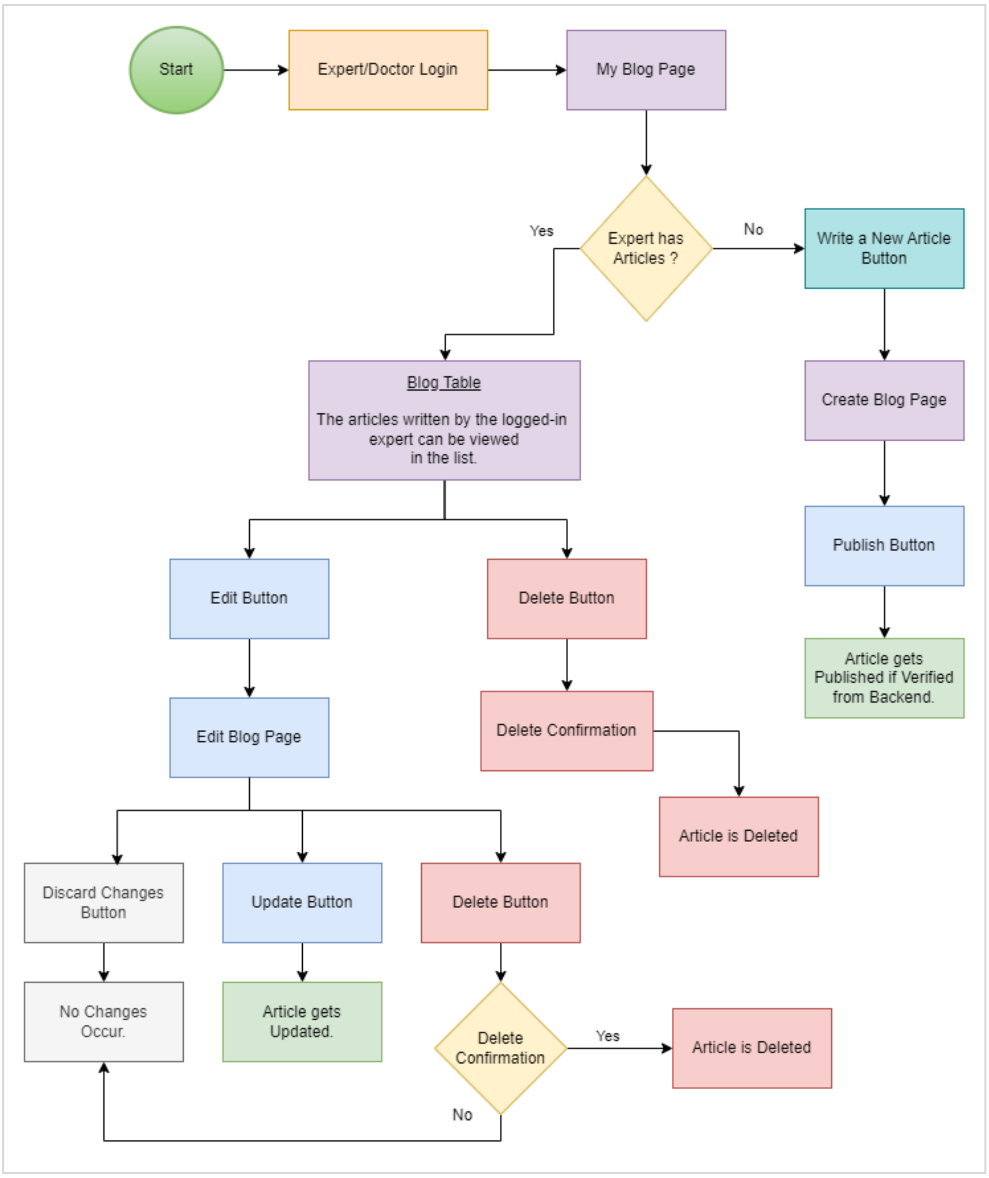


Figure 4: Process Flow for an Expert to Navigate and Interact with the Blog System

5.5 User Authentication System (Individual Contribution)

In order to protect user data and restrict access to specific areas of the platform, the user authentication system was designed to ensure that authentication procedures are robust, secure, and efficient. Hence, I developed and integrated the entire user authentication system from the front-end, incorporating sign-up and login functionalities for both regular users and expert users.

5.6 Results and Underlying Technologies of User Authentication System (Individual Contribution)

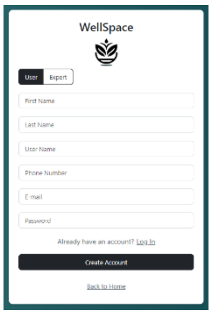
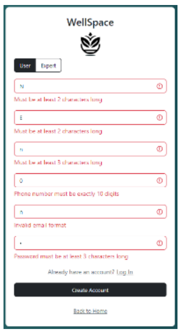
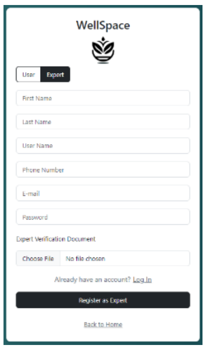
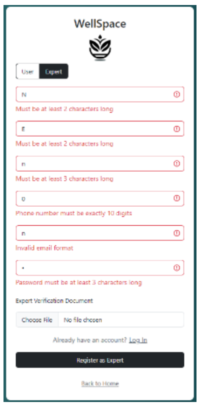
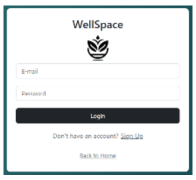
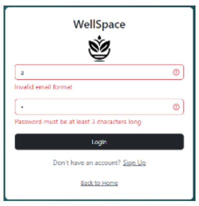
Page Name	Description	Screen Capture
User Sign-Up	This page allows users to register for an account. It includes form fields for first name, last name, username, phone number, email, and password. The form validates input fields such as alphabetic characters for names, alphanumeric characters for usernames, and valid email format. API Call: POST /api/v1/auth/register to register a new user. Libraries Used: axios for API requests, react-bootstrap for UI components, uploadCloudinary for handling file uploads.	 WellSpace User Expert First Name Last Name User Name Phone Number E-mail Password Already have an account? Log In Create Account Back to Home  WellSpace User ExpertN Must be at least 2 characters long E Must be at least 2 characters long n Must be at least 3 characters long 0 Phone number must be exactly 10 digits n Invalid email format Password must be at least 3 characters long Already have an account? Log In Create Account Back to Home
Expert Sign-Up	This page is similar to the user sign-up but includes additional fields for expert verification. Experts need to upload a verification document which is handled by uploadCloudinary. The form validates input fields, ensuring proper format and length. API Call: POST /api/v1/auth/register to register a new expert. Libraries Used: axios for API requests, react-bootstrap for UI components, uploadCloudinary for handling file uploads.	 WellSpace User Expert First Name Last Name User Name Phone Number E-mail Password Expert Verification Document Choose File No file chosen Already have an account? Log In Register as Expert Back to Home  WellSpace User ExpertN Must be at least 2 characters long E Must be at least 2 characters long n Must be at least 3 characters long 0 Phone number must be exactly 10 digits n Invalid email format Password must be at least 3 characters long Expert Verification Document Choose File No file chosen Already have an account? Log In Register as Expert Back to Home
Log In	This page allows users to log in with their email and password. The form includes validation for correct email format and minimum password length. Upon successful login, the user context is updated, and the user is redirected to the homepage. API Call: POST /api/v1/auth/login to authenticate users. Libraries Used: axios for API requests, react-bootstrap for UI components.	 WellSpace E-mail Password Login Don't have an account? Sign Up Back to Home  a Invalid email format Password must be at least 3 characters long

Table 3: Overview of Sign-Up and Log-In Pages

5.7 State Management of the System (Individual Contribution)

State management involves handling the status of an application at a specific point in time. Essentially, the state represents what the application knows at any given moment, such as user input in a form, the user's authentication status, or the currently viewed page (Alkhateeb, 2024).

In the context of the 'WellSpace - Mental Health Platform' project, I implemented state management using 'React Hooks' and the 'React Context API' to ensure that user data and session information were consistently maintained across the application. For that, the 'UserContext.js' component was created using 'Create Context' library (React Context API) in React. The diagram below demonstrates how state management takes place due to the interactions with the React Context API.

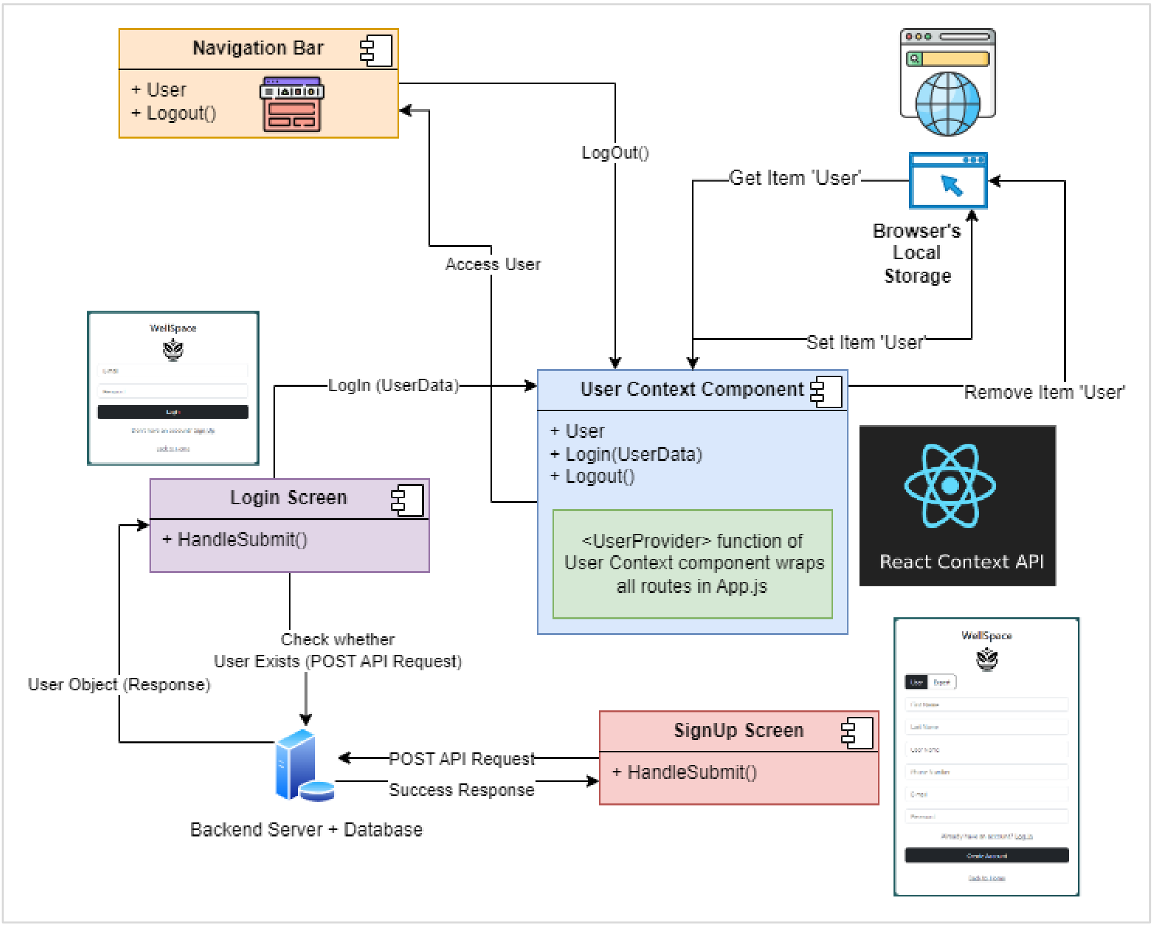


Figure 5: How State Management has been Implemented by Connecting User Context Component with Other Components and the Integration with Backend.

As given in the above diagram, once the user is logged in, the user's data is stored in the web browser's local storage and is stored as the 'user' object in the 'User Context' component. Initially when the web application is loaded, the 'User Context' component checks whether a user exists in the browser's local storage, and if so, that user's data is assigned to the user object in React 'useState' hook. If not, the 'useState' hook is set to null until someone logs in via the login screen.

Further, anytime a user logs in, both the useState hook in the User Context component and the browser's local storage is updated with the user data. Not only that, but all the routes in the web application are wrapped by the User Provider function of the User Context component. Because of that, the user's data can be accessed from anywhere in the web application through the User Context component. Whenever the user's data is needed, it is accessible from UserContext.js. Finally, when the user logs out, the user object in User Context is set to null and is removed from the browser's local storage as well.

5.8 Results of State Management (Individual Contribution)

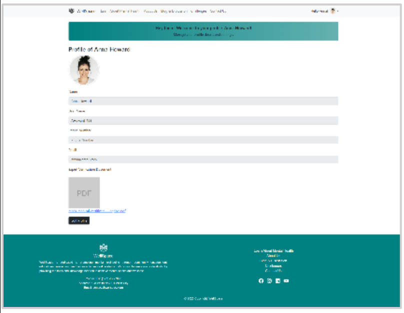
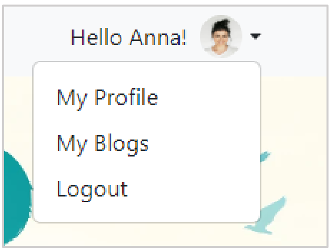
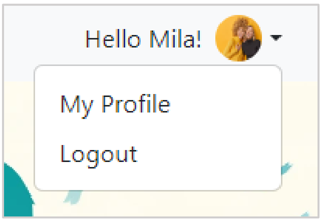
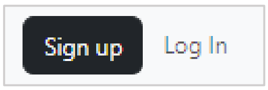
Component Name	Description	Screen Capture
User Profile Page.	Displays user profile information such as name, username, phone number, email, and profile picture. Allows users to edit their profile details and upload a profile picture. For doctors, it includes the upload and display of verification documents. API Call: PUT /api/v1/users/:userId or PUT /api/v1/doctors/:doctorId to update user/doctor data. Utilizes axios for API requests, react-bootstrap for UI components, and uploadCloudinary for file uploads.	
Navigation Bar when Expert is Logged In.	Displays navigation options tailored for logged-in experts, including options like "My Profile," "My Blogs," and "Logout." The state management ensures that the navigation bar reflects the expert's logged-in status and provides relevant options. Utilizes react-bootstrap for UI components.	
Navigation Bar when a Regular User is Logged In.	Displays navigation options tailored for logged-in regular users, including options like "My Profile" and "Logout." The state management ensures that the navigation bar reflects the user's logged-in status and provides relevant options. Utilizes react-bootstrap for UI components.	
Navigation Bar when No User is Logged In / User has Logged Out.	Displays navigation options for non-logged-in users, including "Sign up" and "Log In" buttons. State management ensures that the navigation bar reflects the logged-out status, providing options for users to register or log in. Utilizes react-bootstrap for UI components.	

Table 4: Application of State Management in User Profile Page and Navigation Bar

5.9 Front-End and Integration Testing (Individual Contribution)

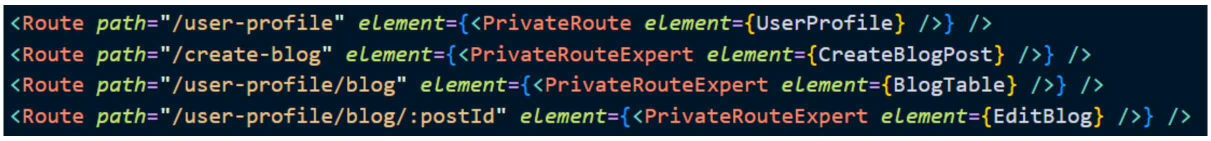
I played a pivotal role in the testing stage as well, concentrating on both front-end and integration testing. All the testing was done manually and as stated in the testing plan. This included unit testing, integration testing and usability testing. Feedback from these tests was used to make necessary adjustments and improvements.

5.10 Implementing Private Routes (Individual Contribution)

I established private routes within the front-end of the website to restrict access to specific pages and functionalities based on user authentication status. This mechanism ensures that only authenticated users and expert users can access sensitive areas of the platform, thereby enhancing security and user-specific functionality. Two types of private routes were established as explained in the following table and displayed in the figure.

Private Route	Private Route Expert
❖ Checks if 'user' object exists in 'User Context' component. ❖ If a user exists, navigates to that route. If a user does not exist, navigates to login page. ❖ Applied to 'user-profile' route.	❖ Checks if ‘user’ is present in ‘User Context’ component and if the role is doctor. ❖ If a user exists and the role is equal to doctor, navigates to that route. If not, navigates to the main page of blog. ❖ Applied to ‘Create blog page’, ‘My blogs page (blog table)’ and ‘Edit blog’ page.

Table 5: Implementation of the Two Types of Private Routes



```
<Route path="/user-profile" element={<PrivateRoute element={UserProfile} />} />
<Route path="/create-blog" element={<PrivateRouteExpert element={CreateBlogPost} />} />
<Route path="/user-profile/blog" element={<PrivateRouteExpert element={BlogTable} />} />
<Route path="/user-profile/blog/:postId" element={<PrivateRouteExpert element={EditBlog} />} />
```

Figure 6: Private Routes applied to Routes in App.js

5.11 Results of Implementing Private Routes (Individual Contribution)

❖ The user profile page can only be accessed by a logged in user/ expert.
❖ The ‘Create blog page’, ‘My blogs page (blog table)’ and ‘Edit blog’ page can only be accessed by a logged in expert.
❖ If for any reason a user who has not logged in types in the user-profile URL, he/she is redirected to the login page.
❖ If for any reason a user who has not logged in, or a user who is logged in, but not an expert in types in the URL for the restricted blog pages, he/she is redirected to the main page of the blog.

6. CONCLUSION

6.1 Project Summary

In summary, my contributions significantly enhanced the WellSpace platform’s front-end, ensuring it was not only visually appealing and user friendly but also deeply integrated with backend services. This integration provided a seamless and efficient user experience, essential for the platform’s role in supporting mental health and wellness. The rigorous testing further ensured the reliability and robustness of the application.

Overall, the WellSpace project successfully addresses a significant gap in the market for mental health support by integrating peer support, professional counselling, and educational resources into a single, comprehensive platform. Our user-friendly web application offers a secure and encouraging space for people looking for mental health support, ensuring accessibility and privacy. The project achieved its goals of developing a robust and secure platform with engaging features that foster community engagement and improve user well-being.

6.2 Future Directions

- Design and development of an administrative dashboard so that an administrator can monitor and regulate the web platform.
- Improving accessibility, privacy and security features.
- Integrating ‘Challenges’, ‘Donation’ and ‘Volunteering’ pages with the back end.
- Improving the booking feature to provide virtual counselling sessions.
- Improving features of the blog such as when experts write articles, to create ‘draft articles’ that can be saved for future editing.
- Search engine optimization.
- Integrating advanced analytics and reporting to track user engagement and overall platform impact.
- Mobile application development.

7. REFERENCES

Alkhateeb, B. 2024, Overview of state management in software development, viewed 3 June 2024, <https://medium.com/@bakrialkhateeb.dev/overview-of-state-management-in-software-development-54c489011b90>

Amazon Web Services (AWS) 2024, What is a RESTful API?, viewed 2 June 2024, <https://aws.amazon.com/what-is/restful-api/>

Betterhelp.com 2024, Online Counseling & Therapy, viewed 1 April 2024, <https://www.betterhelp.com/>

Giesen, F, Searle, A & Sawyer, M 2007, ‘Identifying and implementing prevention programmes for childhood mental health problems’, *Australian Journal of Psychology*, vol. 43, no. 12, pp. 785–789.

Headspace.com 2024, Meditation and Mindfulness Made Simple, viewed 8 April 2024, <https://www.headspace.com/>

Lakhan, R, Agrawal, A & Sharma, M 2020, ‘Prevalence of Depression, Anxiety, and Stress during COVID-19 Pandemic’, *Journal of Global Health*, vol. 11, no. 4, pp. 519–525.

Talkcampus.com 2024, Peer Support Network for Students, viewed 1 June 2024, <https://www.talkcampus.io/>