Kyle Connect

Project Report

Course: Bachelor of Science (Honours) in Software Development
Name: Kyle Kinsella
Student Number: C00273146
Date: 28/04/2025
Supervisor: Joseph Kehoe

Abstract

During this project, I set out to develop a website based around the Discord Application. Initially, this seemed like a strong idea, however after countless hours of development I realized that choosing Discord would present multiple challenges and issues. Throughout the development of Kyle Connect, I encountered numerous errors and obstacles. While these obstacles often caused frustration, I was usually able to identify the problem and implement a solution. Nevertheless, a few problems remained unresolved. Additionally while developing Kyle Connect I gained several important skills, such as organizing each component of my project into their own dedicated folder, writing formal technical documents, going to weekly meetings to update my supervisor on what work was complete each week.

Introduction

Throughout this document I will be outlining what work was completed during the final iteration and the previous iterations. The objective of Kyle Connect was to develop a website that mirrors some of the core features found in popular platforms such as Discord. Below are some of the main features included in Kyle Connect:

- **Sign up:** Easily create an account
- **Login:** Access Kyle Connect
- **Send Friend Request:** Connect with others, via sending & accepting friend requests.
- **Create a Server:** Set up a server for group conversations and much more.
- **Messaging:** Send direct messages to your friends.

During the third and final iteration, I focused on improving the user-interface, added in edge cases for actions such as add friend to server and delete friend from server and many more. I worked on making sure my project is all linked together (This was a major issue, see later in document) and allow the user to make a server and much more. In earlier iterations, I focused on user registration, login functionality, sending friend requests and messaging friends.

General Issues

Problems encountered and how they were resolved

During the development of Kyle Connect, I faced many challenges, although some of the issues remain incomplete, I was able to implement most of the features for Kyle Connect. Below are some of the problems encountered and how I fixed them.

1. While I was developing Kyle Connect, my project code only worked on my laptop. So after doing some research on some technologies I found Docker. Docker allowed me to containerize my code, ensuring consistency across different environments. Additionally, this setup allowed other users, such as my family and project supervisor, to access and use my project. Docker was the solution to my first issue [1].

2. While I was developing my messaging for Kyle Connect, this worked perfectly locally. But I wanted to run this via a docker container so I quickly spin up a docker container for my messaging, but the timestamp that was showing on my user-interface was wrong, the timestamp was correctly in the database but when I ran my code via a docker container, the timestamp broke. I tried my best to fix it but I was unable to do so. So I asked ChatGPT [2] for assistance by providing a few prompts and explaining the problem I was trying to fix. He did provide me code that fixed my problem but I made sure that I understood the code before implementing it into my project.

3. While I was developing Kyle Connect, my templates for my project were not working. My project mainly works via getting data from my database and then inserting the data into a template which is then shown to the user. One of the biggest issues was that I was getting data down from my database and I “had” it in the template but it was not working properly. So I went to ChatGPT [2] for some help. After a few prompts he fixed my problem but again he did provide me code but I made sure to understand the code before implementing this into Kyle Connect.

4. During the start of development for Kyle Connect, the problem was if I had a folder called “Signup” and in the code you declared it as `package main`. And if you wanted to use this code elsewhere in your project it would not work. This was something that was very new to me because I had never worked with golang before. After approximately two hours of trying to figure it out, I went to ChatGPT [2] for help and after one prompt he said to me that if you have a folder called “Signup” and you want to use it elsewhere in your project you must declare the package to be: `package signup`. After ChatGPT

provided this information to me, I tried it and it worked perfectly, and to this day I still remember how to fix this issue.

These four challenges were among the most difficult to fix, although many other issues also arose during the project.

Achievements

During this project, I achieved several important milestones. Below are some of the key accomplishments I made while developing Kyle Connect:

Gained Proficiency with Docker

Before this project, I had no experience working with docker, I didn't even know what docker was. Throughout the development of my project I used docker to dockerize my code, so that my supervisor could use my code for each of our weekly meetings, without the need to worry about environment setup issues. While using docker, I gained the following skills: I learned how to write a Dockerfile, build images, run containers and also write a Docker-compose file. Using Docker significantly improved the development process for my project.

Overall, I gained experience with Docker not only benefited the development of Kyle Connect but also added a valuable new skill to my technical toolkit.

Developed Skills in Writing Formal Technical Documentation

While developing Kyle Connect, I wasn't just focusing on writing code, I also had to create technical documentation to show how I was going to develop Kyle Connect. A significant amount of my project required me to create formal documentation. During my project I created the following documents:

- **Functional Specification**: defined Kyle Connect’s purpose, researched different applications similar to Discord and anticipated user interactions and much more.
- **Research Poster**: A visual poster to show the key features of Kyle Connect, goals and overall structure of Kyle Connect.
- **Design document**: Outlined the technology stack, database choice, wireframes for user-interface and diagrams such as control flow graphs, class diagram, sequence diagrams and much more.

Overall, I gained experience with writing technical formal documents while still ensuring the constant updates of coding for Kyle Connect.

Improved Codebase Organization and Structure

During the development of Kyle Connect, I took a thorough approach to organizing my codebase, striving for clarity and efficiency. This ensured that my codebase was well-structured, easy to navigate and maintainable. This made the creation of a solid foundation for future development and scalability. I focused on ensuring a function does one thing not multiple things. By doing this, it made it very easy for me to create new functions and refactor functions whenever necessary.

Learning and Gaining Experience with Go (Golang)

Before I wrote a single line of code, I was deciding on what programming language to work with. Normally if I am creating a new project I would normally pick a programming language that I am familiar with, such as (Java or C++), but for Kyle Connect I chose to create my project in the Go programming language created by google [3]. I chose Go (Golang) because I wanted to learn a new programming language while developing Kyle Connect. Although I occasionally regretted this choice during the initial phase of the project, I am glad I built Kyle Connect using Go.

Designed and Built a Communication Platform Inspired by Discord

While developing Kyle Connect, I aimed to create a communication platform that incorporated some of the core features and functionalities found in popular platforms like Discord. Some of the key features are as follows, send friend requests, create a server, messaging friends and much more. Discord was a big inspiration for my project. Now when I use the discord application, I have a much deeper understanding of how its core features operate, having built similar functionality for Kyle Connect.

Challenges Faced and Key Learnings

During this project, I encountered several challenges while developing Kyle Connect. Below are some of the key challenges I faced:

Friends showing when you go to send a message

Although users could send messages, selecting friends via the sidebar was not as intuitive as I had hoped. I attempted to improve this, but the solution was not fully implemented.

Allowing the user to click into a server they were added to

Users who were added to servers could not click into them to view channels and messages. This feature was not completed due to time constraints.

Limitation: Only One Friend Request Can Be Accepted or Declined at a Time

The system failed to handle multiple simultaneous friend requests properly. Accepting one request would incorrectly accept all pending requests.

Hardcoded value for accept or decline friend request

Accepting or declining friend requests required hardcoded usernames, which is not a scalable solution. Despite significant effort, I was unable to fully resolve this issue.

Localhost issues when deploying

While I was developing Kyle Connect I was running all of my code via localhost using XAMPP [4] and this was perfect while I was developing Kyle Connect locally. But this became a major issue when I was trying to deploy my project to the internet. I asked ChatGPT [2], I gave him a prompt asking how I can mitigate hard-coded localhost and he did give me some information but this did not work, he did mention to use some hosting-platforms, below are a few that I tried to use to mitigate the use of hard-coded localhost:

- Render.com [5]
- Digital ocean [6]
- Railway.com [7]

I tried to use render initially and it did work, but when I was trying to login to my account it did not work, this is because render.com focuses on PostgreSQL Databases. Due to my project being a MYSQL Database this hosting platform did not work.

I then tried to use Digital Ocean but once you go to their website they immediately ask for your credit card details before you can see their website, so this discouraged me from using Digital Ocean.

Finally, I attempted to deploy Kyle Connect to the internet using Railway [7] but unfortunately this was unsuccessful. Despite this, I continued to troubleshoot in hopes of making it work. I even invested money into the service, but after the payment, the deployment still did not succeed.

In summary, I spent countless hours attempting to deploy Kyle Connect online so that actual users could access it, but unfortunately, this effort was unsuccessful.

What I Would Do Differently

If I had the opportunity to redo this project, I would make sure the following features are implemented:

Automatic updating

When new data goes into my database, if a user is logged in this will not show automatically on the user-interface. In order for the user to see the new data they must manually either hit the f5 button on their computer or press the refresh button on the browser. This is not ideal for users of Kyle Connect, if I had more time this is something I would implement this feature.

Session Id

When a user uses Kyle Connect they are put into a logged in table in my Kyle Connect database. But the issue is when we have a server with three people, if one person is logged in and two more people log in to Kyle Connect, my system will only get the last user that was logged in. The way that this would be mitigated is the use of a session id, each user that logs in gets a unique session id but due to time constraints, I was not able to implement this feature into Kyle Connect.

Animations and or Transitions

If I could do this project again, I would implement animations or transitions for Kyle Connect. Adding these features would enhance the user experience by providing smoother and more visually appealing interactions. Incorporating animations and transitions in future iterations would help ensure a more seamless and engaging user interface.

Encryption of messages

If I had the chance to revisit this project, I would implement encryption for the messages sent on Kyle Connect. Implementing encryption would enhance the security and privacy of user communications, ensuring that messages are protected from unauthorized access during transmission.

Technical Issues

Differences from Initial Design

While the initial design of Kyle Connect aimed to include several advanced features such as video calling and a more robust encryption system, due to time constraints and technical challenges, the project was streamlined to focus on essential functionalities. The final implementation included sending friend requests, creating a server and messaging and many more features, but video calling and end-to-end encryption were postponed due to time constraints. Additionally, the user interface was simplified compared to the initial design to ensure a smoother user experience with the existing features. These changes allowed me to meet the core objectives of the project while ensuring Kyle Connect is stable.

Module description

Signup (Make account)

My signup module allows users to create an account for Kyle Connect. I collect the following user information, username, email and password and I store it in my database, thus enabling users to gain access to Kyle Connect. This module is crucial for user registration and ensuring secure access to Kyle Connect.

Login

The Login module allows users to access their accounts by entering their email and password. My system checks the provided credentials against the stored data, granting access to the user's account upon successful login. This module ensures secure login for users to access their account on Kyle Connect.

Send friend request

My Send Friend Request module allows users to send friend requests to other users on Kyle Connect. When a user selects another user, a request is created and sent to the recipient. The recipient can then choose to accept or decline the request, via clicking an accept or decline button. This module helps establish connections and relationships between users on Kyle Connect.

Friendship events

My Friendship Events module manages user interactions related to friend connections. It allows users to add friends to there server, only if they have created a server. This enables them to interact and engage with each other within Kyle Connect. Additionally, this module provides functionality for deleting friends from a server, only if they have created a server, therefore removing their access to shared interactions. This ensures that users can manage their friend relationships efficiently and securely.

Messages

My Messages module allows users to send and receive text based communication within Kyle Connect. Users can exchange messages with their friends, this can only be done individually from one user to another. This module ensures communication enhancing user interaction and engagement.

Server

My server module manages the creation of a server (user groups) on Kyle Connect. It allows users to create servers, this provides a space for group interactions and messaging.

Channel messages

My channel messages module allows users to send and receive messages within a server within Kyle Connect. Users can participate in conversations within channels. This module ensures communication and keeps channel discussions organized, making it easy for users to engage in conversations.

Channel

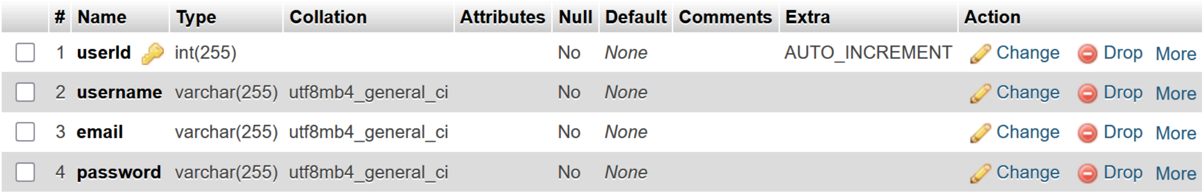
My channel module allows users to create channels within Kyle Connect. Channels are designed to organize discussions by topics or groups.

Data Structures and Database Design

For Kyle Connect, I chose to use a simple array data structure. The reason for this choice is that arrays are straightforward to implement and provide an efficient way to store multiple elements. Arrays allow for easy access to any element by index, and since they are sequential, they make it simple to iterate over the entire array. This flexibility enables me to perform operations such as searching, updating or processing the elements without needing complex data structures. Additionally, arrays offer good performance for tasks where the size of the array is fixed or doesn't change frequently, making them a suitable choice for this project.

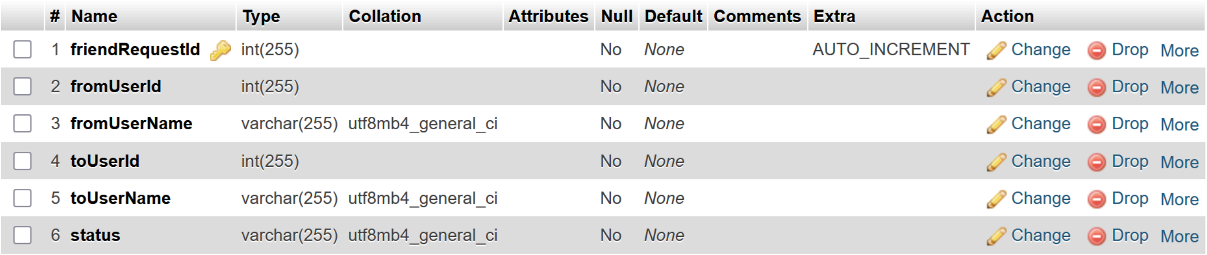
Below are all the database tables along with their corresponding images:

Communicators (Kyle Connect Users)



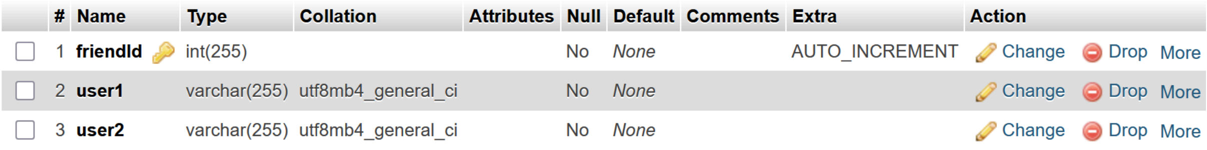
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**userId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change Drop More
□	2	**username**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	3	**email**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	4	**password**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More

Friend Request (Send Friend Request)



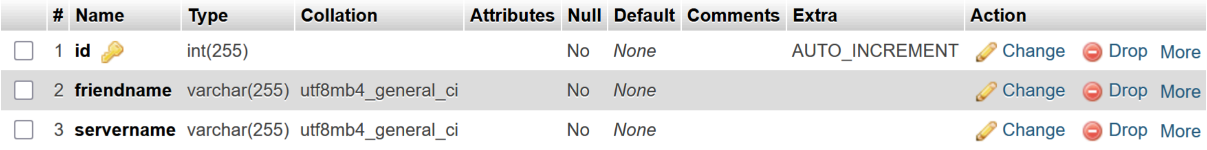
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**friendRequestId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change Drop More
□	2	**fromUserId**	int(255)			No	*None*			Change Drop More
□	3	**fromUserName**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	4	**toUserId**	int(255)			No	*None*			Change Drop More
□	5	**toUserName**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	6	**status**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More

Friends



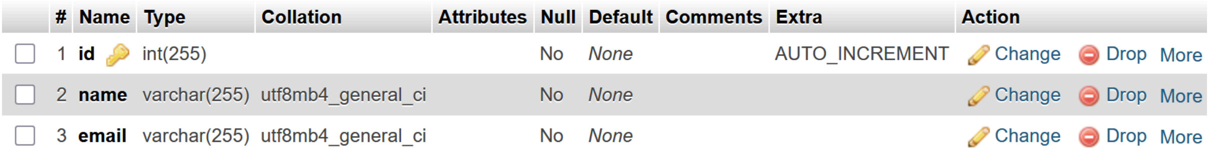
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**friendId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change Drop More
□	2	**user1**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	3	**user2**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More

Friendship Events



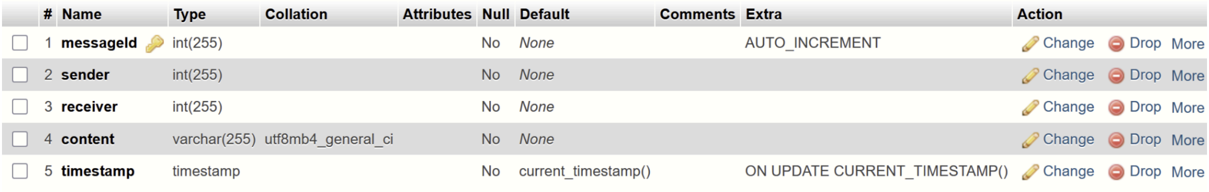
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
□	1	**id** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change Drop More
□	2	**friendname**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More
□	3	**servername**	varchar(255)	utf8mb4_general_ci		No	*None*			Change Drop More

Logged In



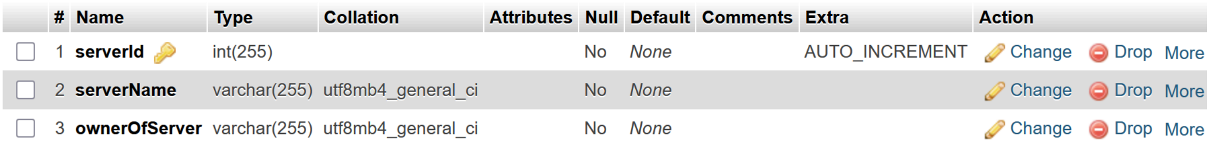
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**id** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**name**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	3	**email**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More

Messages



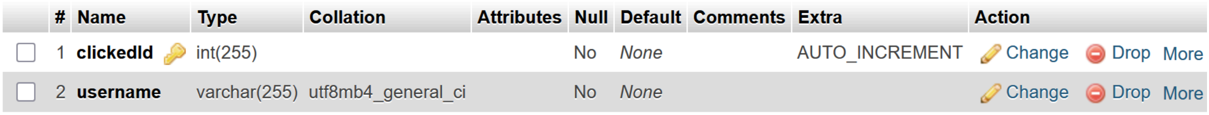
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**messageId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**sender**	int(255)			No	*None*			Change	Drop	More
□	3	**receiver**	int(255)			No	*None*			Change	Drop	More
□	4	**content**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	5	**timestamp**	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change	Drop	More

Server



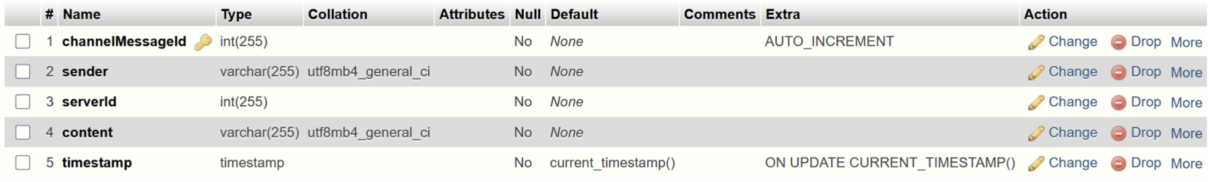
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**serverId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**serverName**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	3	**ownerOfServer**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More

Clicked



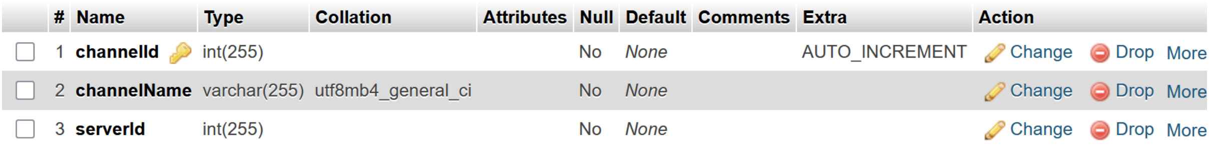
	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**clickedId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**username**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More

Channel Messages



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**channelMessageId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**sender**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	3	**serverId**	int(255)			No	*None*			Change	Drop	More
□	4	**content**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	5	**timestamp**	timestamp			No	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Change	Drop	More

Channel



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action		
□	1	**channelId** 🔑	int(255)			No	*None*		AUTO_INCREMENT	Change	Drop	More
□	2	**channelName**	varchar(255)	utf8mb4_general_ci		No	*None*			Change	Drop	More
□	3	**serverId**	int(255)			No	*None*			Change	Drop	More

Testing Used to Assess the Reliability

Due to time constraints, I wasn't able to implement comprehensive unit testing for my features in Kyle Connect. In future iterations, I would prioritize implementing these tests to ensure code reliability and scalability. Unit testing would allow for early detection of bugs, ensure that individual components function as expected and provide a solid foundation for further development.

Acknowledgements

I would like to express my sincere gratitude to **Joseph Kehoe** who supported me throughout the course of this project. Special thanks to **Joseph Kehoe** for their guidance and valuable feedback throughout the development of Kyle Connect.

Conclusion

In conclusion, while I developed Kyle Connect I gained a lot of valuable skills while creating this project. Although there were quite a few challenges, such as localhost issues when deploying and a few others, all of the core features were developed, such as, send friend requests, create a server, send messages to your friends and many more. If I could do this project again, I would ensure the implementation of encryption of messages and comprehensive unit testing to have an overall better security and reliability. Overall, this project has greatly enhanced my programming skills, introduced me to a new technology called Docker and helped me gain valuable experience in writing formal technical documentation.

References

1. Docker (2024). *Enterprise Application Container Platform | Docker*. [online] Docker. Available at: https://www.docker.com/ [1].
2. OpenAI (2025). *ChatGPT*. [online] ChatGPT. Available at: https://chatgpt.com/ [2].
3. BairesDev. (2022). *7 Companies That Use Golang Creatively*. [online] Available at: https://www.bairesdev.com/blog/companies-using-golang/ [3].
4. Apache Friends (2022). *XAMPP Installers and Downloads for Apache Friends*. [online] www.apachefriends.org. Available at: https://www.apachefriends.org/ [4].
5. Cloud Application Hosting for Developers | Render. (n.d.). *Cloud Application Hosting for Developers | Render*. [online] Available at: https://render.com/ [5].
6. DigitalOcean, LLC (2022). *DigitalOcean – The developer cloud*. [online] DigitalOcean. Available at: https://www.digitalocean.com/ [6].
7. Railway. (2024). *Railway*. [online] Available at: https://railway.com/ [7].