KTM SOLUTIONS

DEEP COVE OUTDOOR EDUCATION TRUST
Educating and sharing our conservation values

KIOSK DEVELOPMENT PROJECT

JOSIAH SWENEY • JOSHUA HAYES • JAYDEN BECKHAM

Table of Contents

Acknowledgements

Joshua Hayes

Project Manager

Front End Development

Josiah Sweney

Lead Programmer

Back End Development

Jayden Beckham

Assistant Programmer

Game Developer | Database

We would like to make the following acknowledgements for those who have helped us on our journey.

We would first like to thank our Mentor Ken Sutton. Ken has been very supportive and has been with us every step of the way throughout our project. He has provided the team with valuable feedback, advice and ideas.

Anita Murphy is another person who has had a heavy involvement in this project. She has helped us stay organised at every part of the project, and has been there whenever we have needed her advice and feedback.

We would also like to thank Mike on behalf of the Deep Cove Outdoor Education Trust for not only sharing his ideas and being patient with us as we developed a solution, but for the amazing opportunity we were given to make something that can be used into the foreseeable future.

Abstract

Deep Cove Outdoor Education Trust is a non profit charity organisation which teaches visitors to the Deep Cove area about the environment and conservation. They approached the Southern Institute of Technology (SIT) requesting an educational, interactive, touch screen kiosk system to be developed for them. They wanted this as their educational materials at the time were things such as books and brochures which was considered an uninteresting way of learning. The first part of this project was meeting with the client to ascertain their wants and needs regarding the system. The team then researched other kiosk systems for inspiration. Afterwards the team went on a trip to the Deep Cove area so that they could see what technology and conditions the system had to work with. While on site the team reconfigured the client's WiFi system to allow for easier integration of the kiosk system. The next phase was planning the functionality and layout of the kiosk system through various diagrams and concept art. This functionality was based on the functional and non-functional requirements gathered from the meeting with the client, the trip to the area and the team's personal research. Once the team had agreed on the specifications and features of the system the development began. Each member was assigned a separate role which they could work on simultaneously. Eventually a kiosk system was produced which fulfilled the client's requirements and incorporated effective design features based on the research in to other kiosk systems. Once the kiosk system was created it was then installed on the devices in the client's visitors centre. A wifi system was also created for the client which generates codes that give users internet access for a set amount of time. With the devices set-up and program installation completed visitors to the area are now able to access educational materials in an interesting and interactive way.

"We were chosen for our ability to work well as a team, creative thinking and ingenuity."

Executive Summary

Joshua Hayes
Project Manager

This report outlines our teams entire process for developing the final kiosk solution for the Deep Cove Outdoor Education Trust. It emcompasses many different steps and processes we had to undergo along with issues we faced along the way.

Our first sections of this report outline our research phase, upon which our team met with a representative of the trust with some initial plans and expectations they had for their new solutions. Once this had been fully outlined, we then made a visit into Deep Cove itself in order to scout out the area and reconfigure their wireless onsite. This was a huge learning experience, as it allowed us to not only get a sense for the environment our solution would be placed in but also allowed us to see what configuration options were available on-site.

It was after these initial visits and meetings that the concept phase officially began. It was well known by our team what our client was looking for, and now it was time to simply figure out how to implement such a solution. A lot of concept art was drawn, and eventually after a long planning phase we decided to build a web-based application using a concept we'd developed. Our team was primarily split into three different application areas. We had the backend developer, the front end developer and the game developer/information collector. These roles were selected for particular team members based on their existing knowledge of programming and working on these kinds of applications prior, and all of our team was expected to learn Javascript and React in order to work on these applications. Jayden was assigned the role of Games developer and information collator. In this role he was expected to develop at least one Javascript based game for the final application in order to entice younger users to use our system. He was also in charge of collecting physical resources within deep cove and digitizing them in a way that would allow us to easily enter them into our database. This included the likes of image resources, as well as short/long descriptions and facts. Where possible, we also tried to gather video resources as well as sound resources. The role of Lead Programmer and backend developer was assigned to Josiah Sweeney. This role essentially entailed being in charge of the main laravel side of the application, sorting application routing and focusing mostly on the administrative panel's functionality. As Lead Programmer, Josiah was also responsible for keeping an eye on the different aspects of the main application such as development progress of our Javascript game(s), and ensuring these would not only work seamlessly with our system but also be in line with our existing code base. Finally the role of front end developer was assigned to myself, Joshua Hayes. As Project Manager it made the most sense that I would be primarily in charge of the final look of the application working mostly with

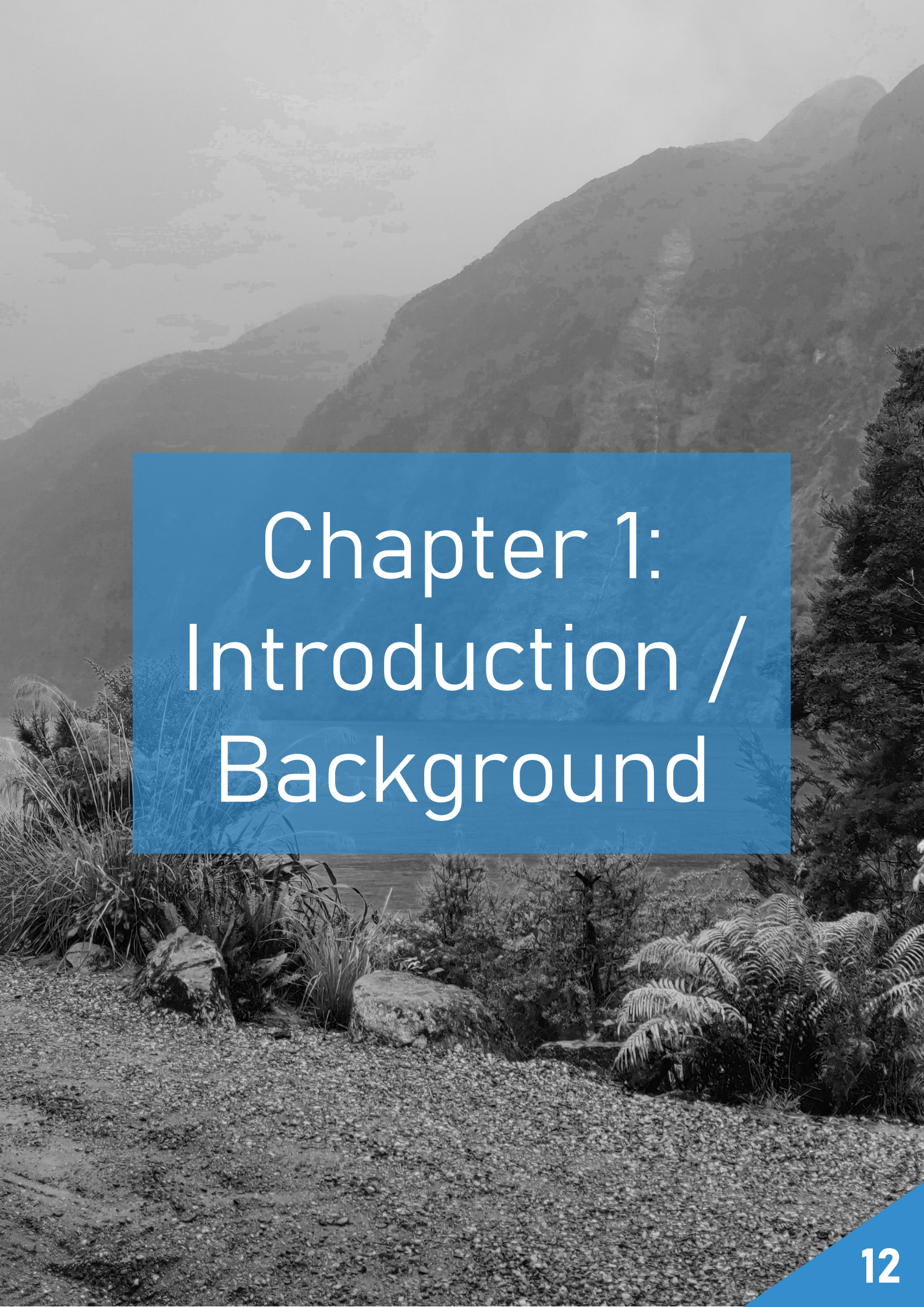
design in both concept art, the HTML/CSS and modifying our main React elements to have these fit with these new design changes. It was also my job to watch over our overall team's progress to ensure we were on target right the way through the project and sticking to deadlines.

During early development of our initial application, we quickly found that keeping tabs on each other's work was proving to be very difficult and we kept breaking each other's section of the application while trying to work in different areas simultaneously on different computers. We resolved this issue by making the decision to move all of our development over to GitHub, a code sharing platform that allowed our team to collaborate and keep tabs on each other's code. This not only allowed us to keep commented history on each commit to the application and have this all backed up to the cloud, but also keep our work separated in different branches. This way we could have our back end developer modifying the backend without fearing it would affect my work or the work of our games developer which drastically increased development time and overall team moral. For a magnitude of reasons we also moved our main backend over to Laravel instead of ASP.NET which we initially started with, which made it that much easier for us to share this code and combine branches together with less fear of our application breaking.

> "While it required a bit of learning to get the hang of these new languages, it made development that much smoother and are skills we will carry with us into the future.."

There has been a lot of ups and downs as to be expected with a project that lasts this length of time. At one point we sadly lost a team member due to some contribution issues which was a big blow to our team, but all of us have done a fantastic job at keeping our focus solely on the completion of this project and putting as much time as all of us could in order to make it the best it could be. While it's been a long ride, our team is really happy with how our solution has come together and how all of us have worked well as a team. We're all hopeful that the Deep Cove Outdoor Education Trust will enjoy our final solution and that it will be a great asset that can be used for years to come.


WAIMEA
PURE KIWI MARINE



Chapter 1: Introduction / Background

1.1 Project Overview

This project was to create an interactive touchscreen kiosk system for Deep Cove Outdoor Education Trust. The purpose of this was to provide visitors to the Deep Cove area with a fun way of learning about nature and the environment of the local area. The kiosk was to feature information about the flora and fauna of New Zealand which visitors could browse through and read. There was also information about walking tracks in the area which is a common activity for visitors to do. To draw more people into using the system there was also a section for games which have an environmental theme. Additionally, there was an administrator function built in to the kiosk system. The administrator mode allows the client to edit the content and layout of the system to be exactly what they want. The team also set up a locally hosted intranet for the client to use the kiosk system on as their internet is very slow and expensive and there were concerns that long wait times and having to pay would put people off using the system.

1.2 Project Team Members and Roles

Joshua Hayes - Project Manager

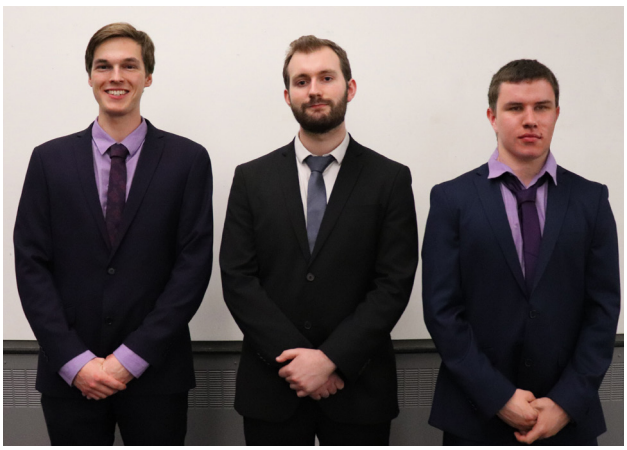
As the Project manager he was responsible for organizing the team to ensure that each member knew what they needed to do and were working towards it effectively. He was also the primary communicator with the client as well as the team's mentor. His role as front end developer meant he was responsible for the appearance of the application which included things such as the style, layout and animations.

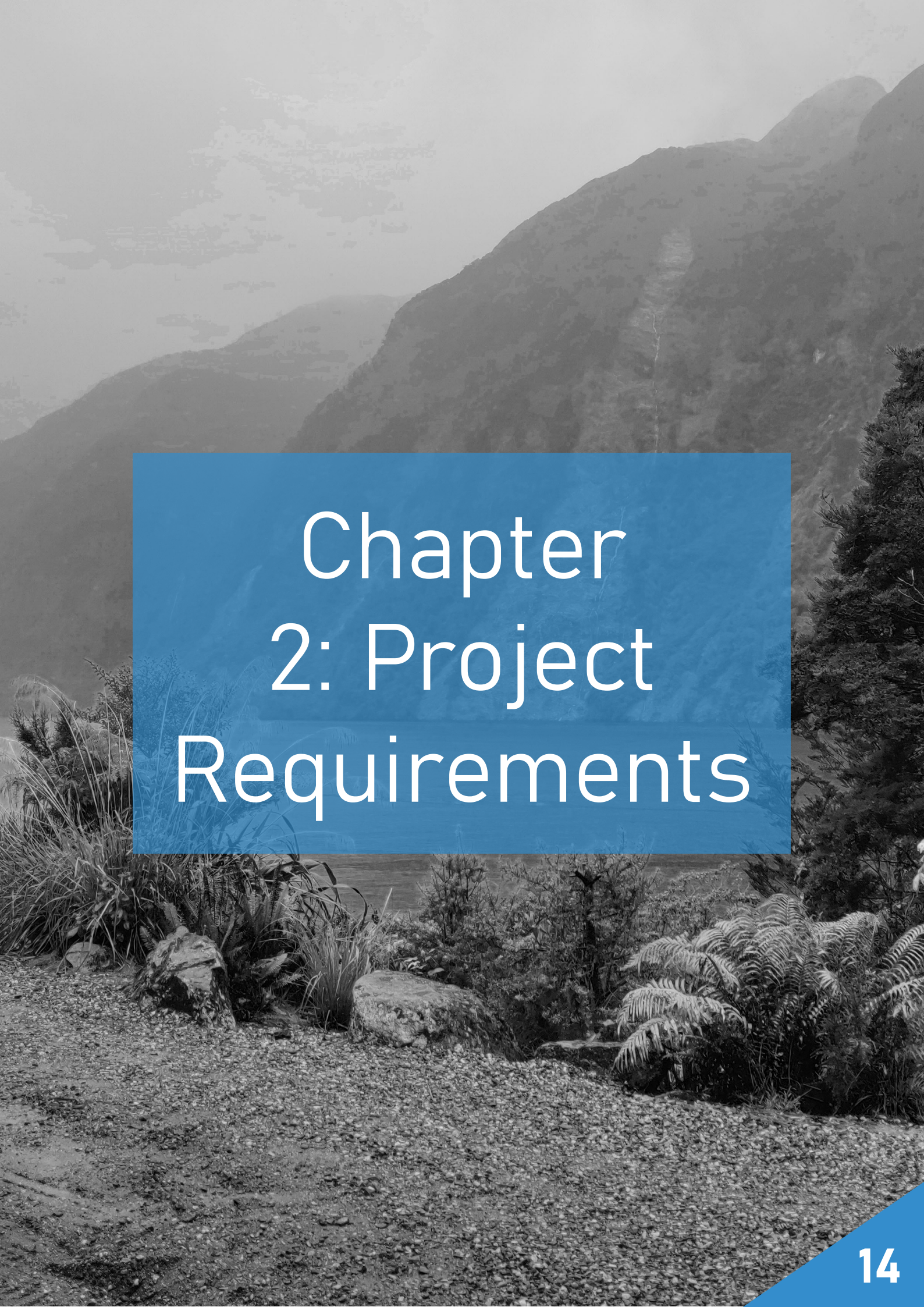
Jayden Beckham - Game Developer

His primary role was as the game developer. The main focus for him was the creation of several games to go into the kiosk system. This involved ensuring they worked on the Raspberry Pi 4, were compatible with touch screen devices and were responsive. He also collected and organized the data as well as assisting in the creation of the database.

Josiah Sweney - Back End Developer

As he was assigned as the back-end developer he was responsible for the programming of the kiosk system itself. This required making sure that all the necessary functionality was in place and working correctly to fulfill the client's requirements. He had to work closely with all the other members of the team so that any progress they made was implemented correctly on to the system itself.





Chapter 2: Project Requirements

2.1 Client Requirements

Shall provide information about deep cove

The primary functional requirement of the kiosk system is that it provides information about the environment and animals of New Zealand. This information included things such as which animals are pests, what the conservation level of different animals is, traditional uses for various plants and various other facts. This information was gathered from sources provided by the client and then checked against reliable sources online such as the Department of Conservation and Te Ara websites.

Shall interact with and store database onsite

The kiosk also had to be able to interact with the database which was stored on site. This was important because if there was no database interaction then the system would not be able to fetch any data meaning there would be nothing for it to display. The database needed to be stored on site as the client is located in a remote location with a lack of infrastructure in place meaning the internet connection they get is slow and expensive. If the database was stored externally then accessing it would have required payment and long loading times, potential users would have been discouraged from using the system if they needed to pay for it or they had to wait a long time for things to load.

Shall work on touch screen

The kiosk system was required to work on touch screen devices. Touch screen compatibility is important as the devices the client has on-site which they wanted the system on are all touch based. The kiosk is also aimed at school age children (as they are the majority of visitors to the area) and younger people's lives are more integrated with technology, they are used to having everything be handled with touch screen devices. The desired outcome is that by applying the system to a familiar platform which visitors are comfortable with it will encourage more people to use it.

Shall not lose data in power outage

As stated previously, the client is in a remote, undeveloped area, because of this they suffer semi-frequent power outages. The occurrence of power outages necessitates the system being able to retain all data in the event it is shut off unexpectedly. If there was a loss of data then the client would be required to re-configure all of their settings and put the data back in every time the power went out which would be an inconvenience.

Shall have admin privileges

There had to be an administrator mode which the client could access but regular visitors were unable to. Having an administrator function was necessary so the client is able to edit the content of pages, add new pages or delete pages if and when they wish to. The administrator mode was made so that it is only accessible to the client, this is so visitors can not make unauthorised changes to the system.

Should be user friendly

The kiosk system was designed to be user friendly, meaning it has a very simple layout which is easy to understand and navigate through and the infor-

mation is organised logically. This was a goal since an overly complex design would make visitors less willing to use it.

Should process requests quickly

Requests were to be processed quickly and any loading times minimised. The purpose of storing the database locally was so that users did not have to spend a long time waiting for things to load, if the system itself was made in a way that processed request inefficiently then this would choice would have been redundant.

Should have a reliable uptime

Having a reliable uptime was important for the simple reason that if the system was constantly down then no one would be able to use it in the first place. The team aimed to have the system functional 99.9% of the time.

Users should be able to find info quickly

It was important for the system that users are able to find the data they want to quickly. This ties together the goals of having a simple, user friendly layout and processing requests quickly. Ideally users should easily be able to find the page they wish to go to and have everything load quickly for them along the way.

Client can set up additional kiosks easily

It was also desirable that the system was simple enough that people who weren't involved in the development could set up another kiosk by themselves. This was important so that if the client wanted to have additional kiosks in the future they don't need to call the team in.

2.2 Research and Analysis

2.2.1 Setting The Scene

After our first discussion with the client in terms of what kind of solution they were looking for, our team spent a large portion of time brainstorming what kind of solution would be appropriate for the scenario that was described to us. As the visitor's center is powered by the hydropower plant and not connected to any other power grid, it meant that power outages occur frequently and whatever devices that we decided to hook up needed to be configured to restart themselves back into their previous state when power is restored. The internet connection onsite at the time of research was only through a satellite connection, which was not only slow but also rather expensive. Internet is unlimited during the hours of 10pm-6am, but capped during the day.

The Deep Cove Outdoor Education Trust had a variety of equipment already purchased and onsite which was in use prior to our solution being researched. This included an Android TV connected to a large touch screen that they had installed in a section of the accommodation.

We discussed the possibilities of using the Android TV the client had to house our solution, but after researching these options we decided to mainly focus on implementing a solution on the Raspberry Pi's which would be connected to touchscreens, in saying this we planned to have support for the application to work in some way on the

android TV's as well. We made this decision after analysing the findings that we came to during our research and came to the realisation that the Raspberry Pi's were the best option. Our reasoning for this starting with the fact that the Raspberry Pi's were an inexpensive option which worked well as the client wanted us to avoid unnecessary costs as they are a nonprofit charitable trust. Another reason why the Raspberry Pi's were the best option is because the are headless units that can easily reboot in the event there is a power outage, which is a semi frequent event at Deep Cove. The last major reason that we decided that the Raspberry Pi's were the best solution is because they were small, meaning that they can easily be mounted anywhere without taking too much space.

2.2.2 Solution Type

There were two primary ways the application could have been developed, namely as a native application or a web application. Each has its own advantages and drawbacks, which will now be discussed.

A native application would involve developing the software to run directly on the device. This would benefit the speed of the application, as there are fewer layers between instruction and action. This is desirable, as for this particular use case reducing latency is important, as users will quickly grow impatient if the interaction is delayed.

A web application would involve developing the software as a website to be displayed by the device. With the proliferation of the Internet, devices with web browsers are the norm and thus it can therefore be assumed that most devices are capable of accessing the application. This is a significant advantage over a native application, where factors such as different operating systems mean that the application is unable to be executed. A web application would also simplify content management for the client by centralising the data to a single source. This would work to reduce data redundancy and errors as there is no potential for data to be different in more than one place. There is also a smaller learning curve for web development, and there are a multitude of high quality learning resources available. This will allow the application to be developed faster and thus more features can be implemented.

After careful consideration a web-based application was chosen, as although the speed may be slower than that of a native app, the low intensity of data being sent and the high speed of the local network will work to minimise this gap. The ability to be viewed on most user's individual devices is also a significant advantage.

2.2.3 Server Operating System

There were several options available for operating systems on the server machine.

Of the Microsoft Windows family of operating systems, the two most favourably considered were Windows 10 Professional and Windows Server 2019.

Windows 10 is a popular and intuitive operating system developed by Microsoft Corporation. Its key benefit is that due to its popularity it is well known to a large proportion of people. This maximises the chance that any future developers have a good grasp of the system when they need to change or add features. However, it isn't primarily designed to be run as a server operating machine and thus it lacks certain abilities

that are necessary in a server environment. Namely, it is unable to perform system updates without a system reset. This is undesirable, as it impacts on the uptime of the application.

Server 2019 has the same well-known layout of later versions of Windows, but with extra features to assist in its performance as a server. It allows the installation of different components depending on the needs of the user: for example, there is a Web Server component that can be installed that allows the running of a web server. This is desirable as it simplifies the installation of this solution and possibly any future solutions. It also simplifies future maintenance of the system by reducing complexity.

Of the Linux family of operating systems, there were two primary considerations.

The first consideration was Ubuntu. Ubuntu is a free and open-source Linux distribution based on Debian and developed by Canonical. It is officially released in three editions: Server, Desktop and Core. It is one of the more popular Linux-based operating systems, and thus there is great support and development of software for it. As the operating system being selected is for a server machine, the Server edition is the most relevant for further research.

Ubuntu Server is highly prevalent on web-hosting platforms, powering approximately 35.9% of websites globally. One particular advantage it has over other platforms in its class is its snap package feature. Snap packages are universal packages that contain all the necessary dependencies and can be installed with one simple command, for example: sudo snap install package-name. Applications in snaps are self-contained and run sandboxed, so they don't interfere with the rest of the system. Similarly to Windows Server's components, snap packages simplify the process of setting up and maintaining the system. This again assists in reducing the complexity of developing solutions with this system. Canonical also rolls out an updated Ubuntu every six months, with support extending for eighteen months after release. They also release long-term support versions every two years that have an extended support period of five years. Ubuntu comes with the GNOME desktop environment by default, which is a free and open source desktop environment for Linux operating systems. It has a simple and intuitive interface, which even non-technical individuals will be able to navigate within.

Fedora is another free and open source Linux distribution sponsored primarily by Red Hat and developed by the community-supported Fedora Project. It comes in two official editions: Workstation and Server. Like Ubuntu, Fedora has an update rolled out every six months. The support period after release however is only thirteen months instead of Ubuntu's eighteen. This does promote cutting-edge software development for the platform, as it frees developers from some backwards-compatibility constraints. However it also makes it a poor choice for embedded systems and hosting machines i.e. for web servers.

After careful consideration, it was decided that a Linux-based operating system would be the best choice as a server operating system. The main factors going into this decision are the fact that they are free and they are designed to be run as server software. This will help to minimise cost to the client and ensure that the system runs as well as possible for as long as possible.

Of the Linux-based operating systems it was decided that Ubuntu was the best option. This was mainly due to its greater focus on long-term support and updates, because once the system is installed it will remain in place without great modification. This means the longer support period of Ubuntu is preferable to the shorter period of Fedora, as it will mean that fewer system-wide upgrades will need to be made. This will ensure that the system keeps ahead of security vulnerabilities and avoid undiscovered bugs for as long as possible.

2.2.4 Model-View-Controller

Having settled on a web server as the driving force of the application, the next consideration was how to implement it.

One of the most popular design patterns for web development is the Model-View-Controller pattern. This involves splitting the application into three logical units, those being the Model, View and Controller respectively. The Model handles business logic, the View handles display logic and the Controller handles user requests. Splitting the logic causes a separation of concerns, which has several key benefits. Firstly, a split in logic allows developers of differing areas to develop simultaneously. This will increase productivity and the speed at which the application is developed, as more skillsets can be utilised for longer. It also makes the application easier to understand, which assists in development, particularly in bug-fixing. This will speed up the testing phase of the application, as it will be easier to track and fix any bugs that arise. One final advantage is that the development team is quite familiar with the MVC design pattern, which will again work to increase development speed and productivity.

2.2.5 Framework

Once the MVC design pattern was chosen, the next step was to choose the best MVC framework for the solution.

One popular framework that is based on the MVC design pattern is Microsoft's ASP.NET (Active Server Pages on the .NET framework). A key benefit of this framework is that it is written in C#, which is the primary programming language of the development team. C# also performs quickly as it is a compiled language. Due to its popularity, it has comprehensive documentation and a plethora of learning material surrounding it. It is also free and open source, which will reduce cost to the client. However a significant disadvantage is that it is not natively cross platform with Linux, which would cause great difficulty in its implementation and deployment. It would require extra software such as Mono in order to be able to run. Mono is not a cure-all however, as not all libraries and packages work with it. This would limit the team in terms of what they could accomplish over the whole project, and thus an alternative would be much preferred.

These issues are resolved however by ASP.NET Core, which is a rewritten ASP.NET that runs natively cross-platform on Windows, MacOS and Linux. Due to this complete redevelopment, it has been decoupled from Windows and IIS (Internet Information Services). It takes on a more lightweight, modular approach compared to the original ASP.NET, which assists in application speed and making development less complex. Despite being "native" to Linux, further research has uncovered minor issues in

deployment which could hinder progress. A host of extra dependencies must also be installed in order for the application to function correctly, which can be found here.

Laravel is another web framework following the MVC design pattern, but instead written in PHP. PHP (Hypertext Preprocessor) is a server-side scripting language that is interpreted at runtime, which can lead to reduced performance compared to a compiled language like C#. Like ASP.NET and ASP.NET Core it is free and open source, and as it is written in PHP it runs natively on Linux-based operating systems. This is desirable, as it will work to prevent compatibility issues if the operating system or the framework is updated. However the development team is unfamiliar with PHP, and so will have to put extra time into learning the language.

One final consideration made was of Phalcon, which is another PHP MVC framework. However it is not as feature-complete as Laravel as it is designed to be more stripped-down. This can be a benefit if the bulk of a larger framework is detrimental, or if one wants to implement a feature in a particular way. However this does leave one to implement all of the features provided by other frameworks by hand. This can be time consuming, and it is unknown whether the extra effort that must be put in with Phalcon will have any benefit to the project.

After much deliberation, ASP.NET Core was selected for several key reasons. Firstly, the development team's familiarity with it and the fact it is written in C# both weigh heavily in its favour. This would reduce the learning curve that would be required to start development, thus increasing the potential time that could be spent developing additional features. It would also assist in testing and bug fixing as the team would be more competent at identifying issues. The potential increased performance over PHP is also a contributing factor, however the application will only service a small number of users in a small geographic area. Thus, it is not the deciding factor, but it is a nice addition.

2.2.6 Database

A database will also be required to store and manage the information used by the web server. It was decided that a relational database would be the best choice, as it minimises data redundancy and inconsistency. Several relational database servers were considered, which are discussed in detail below.

Microsoft SQL Server (developed by Microsoft) is a popular relational database server. Being a Microsoft product, it is designed to work with ASP.NET and ASP.NET Core. It does this by utilising the Microsoft Entity Framework, which is an ORM (Object-Relational Mapper), which maps between database items and .NET objects. This simplifies database interactions by abstratifying the interfacing, so the developer does not need to work with raw SQL strings or database connections. This is favourable, as it reduces development complexity, and also effectively eliminates the risk of SQL injection as the ORM automatically sanitises the SQL strings. However, the software is proprietary and has a subscription cost which would incur great cost to our client. This is far from ideal, as being a non-profit charitable organisation any cost can be burdensome. Thus, any free alternative would be highly preferable.

MySQL Community Server is developed by Oracle and the community and is the

world's most popular open source database. It has good documentation and learning material surrounding it, which works to increase the development speed of the database and help with its maintenance. MySQL is compatible with the Entity Framework ORM via a package provided by Oracle themselves, however it is out of date and has a myriad of bugs that hinder development greatly. A third-party package provided by Pomelo which fixes these issues can be used as a replacement, although a first-party package by the developers themselves would be preferable. It also functions with the Eloquent ORM, which is the Laravel counterpart to ASP.NET Core's Entity Framework. MySQL is free for community use, which means the client does not have to pay any subscription costs. It also runs natively on Linux.

MariaDB is a fork of MySQL, indeed it is developed by the original developers of MySQL. It is also free and open source, has good documentation and learning resources and runs natively on Linux. In fact, MariaDB can act as a drop-in replacement of MySQL, as the connectors and queries are all functionally identical. This also means it works with Microsoft's Entity Framework and with Laravel's Eloquent.

The main deciding factor between the two is whether one prefers to run a database developed by a corporation or a community. The primary point of contention against MySQL is that Oracle simultaneously develops MySQL Community, a free and open-source database engine, and the proprietary, closed-source Enterprise edition. It could be argued that this leads to a conflict of interest within the company, which could lead to anti-consumer behaviour. Due to this, it was decided that MariaDB is the best choice for this solution. This is because it is less likely to have difficulties arising from this conflict of interest, and thus should be more future-proof for the client.

2.2.7 Front End

For the front end framework several options were considered. It was decided that a single-page application would be optimal for a kiosk setup, as this would improve load times and allow the application to flow between pages easier. The load times of the application are decreased as most of the resources such as HTML, CSS and Javascript are only loaded once initially on the first request. This can cause a slow first load, however for a kiosk application which is restricted to only the single domain, it will only ever perform one load when the device powers on. A single page application involves rewriting the current DOM (Document Object Model) to display the new page, instead of fetching a new one from the server. This prevents the hanging occurring between actions, as the client already has the UI and does not have to wait for a response from the server to know what to render.

The first consideration was for React.js. React is a javascript framework developed by Facebook and the community, released in 2013, that is used for building user interfaces. It can be used to create single-page applications, which it does by manipulating the DOM (Document Object Model) via means of a virtual DOM. DOM manipulation can be quite slow, so it also maintains this virtual DOM to track changes and only pushes changes to the actual DOM when required. It uses declarative language, which means that it works by taking the instruction of the code the developer wrote and figuring out the best way to perform it. This simplifies development by abstractifying away the "how" and making it more about the "what". React is also very popular, receiving

frequent updates to further optimise it, add new features and fix bugs. Its popularity also means that there are many high quality and up-to-date learning resources available. React can utilise JSX, a javascript syntax extension, to define the interface. JSX is quite useful because it loosely couples the rendering logic with the other display logic - namely events and data preparation.

Another consideration made was for Angular. Angular is another javascript framework, instead developed by Google and the community and released in 2010. It is also uses declarative language, and uses a client-side Model-View-Controller framework to achieve single-page applications. Angular works by binding HTML with custom attributes to Javascript variables behind the scenes. These variables can be manipulated manually through code, or set via static or dynamic JSON (Javascript Object Notation). The use of an MVC design pattern for the front-end also brings all the benefits of MVC to the front-end development, namely decreasing complexity and reducing burden on the server.

The final consideration made was for Vue.js. Vue is also a javascript framework, developed by an ex-employee of Google and the community. It is the youngest of the three, being first released in 2014. Vue, like React, utilises a virtual DOM to track UI changes. Vue has gained a lot of popularity in recent years, being the second-most loved framework according to Stack Overflow's 2019 developer survey (Stack Overflow, 2019). This means there are a large number of developers who are actively using the framework, and thus there will be ample help available online for any development issues that arise. Vue is the simplest to learn of the three frameworks, as it uses a templating system that is very intuitive. This would increase the speed at which the team could master the framework and thus speed development.

It was decided that out of the three, Vue and React were the primary front runners. Angular is better suited to large UI applications, whereas Vue and React are better for smaller applications. Although Vue is simpler to learn, it was decided that React would be the best choice for the application. This is because it is still more popular, and also backed by Facebook. This makes it more likely that bugs have been caught and corrected, which is of high importance to the team.

2.3 Feasibility

The initial requirements were feasible based on both the time provided and the team's skill level. Due to their feasibility these requirements have been fully implemented in to the system. There was some feasibility issues with the next set of requests the client made. This was due in part to a lack of knowledge within the team, but primarily because of time constraints. The gap in knowledge could have been mitigated if there was enough time, however, the secondary set of requests made by the client was given to the team towards the end of the year. This combination of knowledge and time limitations meant that not all of the client's requests were able to be implemented. The second set of requests were organized by priority and the most important ones were implemented. Among the secondary requests the ones that were particularly difficult and thus were not able to be implemented were integrating microsoft word and PDF files.

Impact	Likelihood				
	Highly Likely	Likely	Moderate	Unlikely	Highly Unlikely
Severe	A1	A2	A3	A4	A5
Significant	B1	B2	B3	B4	B5
Moderate	C1	C2	C3	C4	C5
Minor	D1	D2	D3	D4	D5
Insignificant	E1	E2	E3	E4	E5

2.4 Risk Analysis

Not completing the project on time - A5

This was the biggest risk facing the team during the project. The impact this would have had on the project is extremely high. This was however, very unlikely to occur as a steady, continuous workflow was maintained throughout the duration of the project. The combination of the highest possible impact and the lowest likelihood of occurrence has resulted in this being classified as a medium level risk.

Scope creep - C5

A common risk in all projects is scope creep. The client clearly defined what they wanted very early on so there was virtually no chance that any additional features or functionality would be requested. Requested changes could have been very minor and be completed easily or major and require a reworking of the entire project, so the impact was classified as moderate. Since this had a very low likelihood of occurring and a moderate level of impact means this is classified as a low level risk.

Lack of resources - C3

Not having access to resources was a potential risk in this project. Since the system was being designed for a touch screen device it was beneficial to have devices provided by Southern Institute of Technology which the system could be tested on. It was possible that no devices would not be able to be provided in which case there would have been no way to test the full functionality of the system. Not being able to conduct proper testing would have impacted progress and while it wouldn't have prevented development there may have been errors or holes in functionality which slipped through the cracks as they were never discovered. This is classified as a medium level risk as it had a moderate chance of occurring and would have had a moderate impact on the project.

Scheduling - C1

Throughout the duration of the project there was other assignments and projects which the team needed to work on which divided their time and attention. There was no way to avoid this, the best that could be done was to ensure efficient time management so that progress

would not stagnate. This had a moderate impact on the project, although progress was not as fast as it would have been if this was the only project the team had to focus on there was still time left to work on everything which needed to be completed. The guarantee of this happening and the moderate impact mean this was regarded as a high level risk.

Progress dependencies - A3

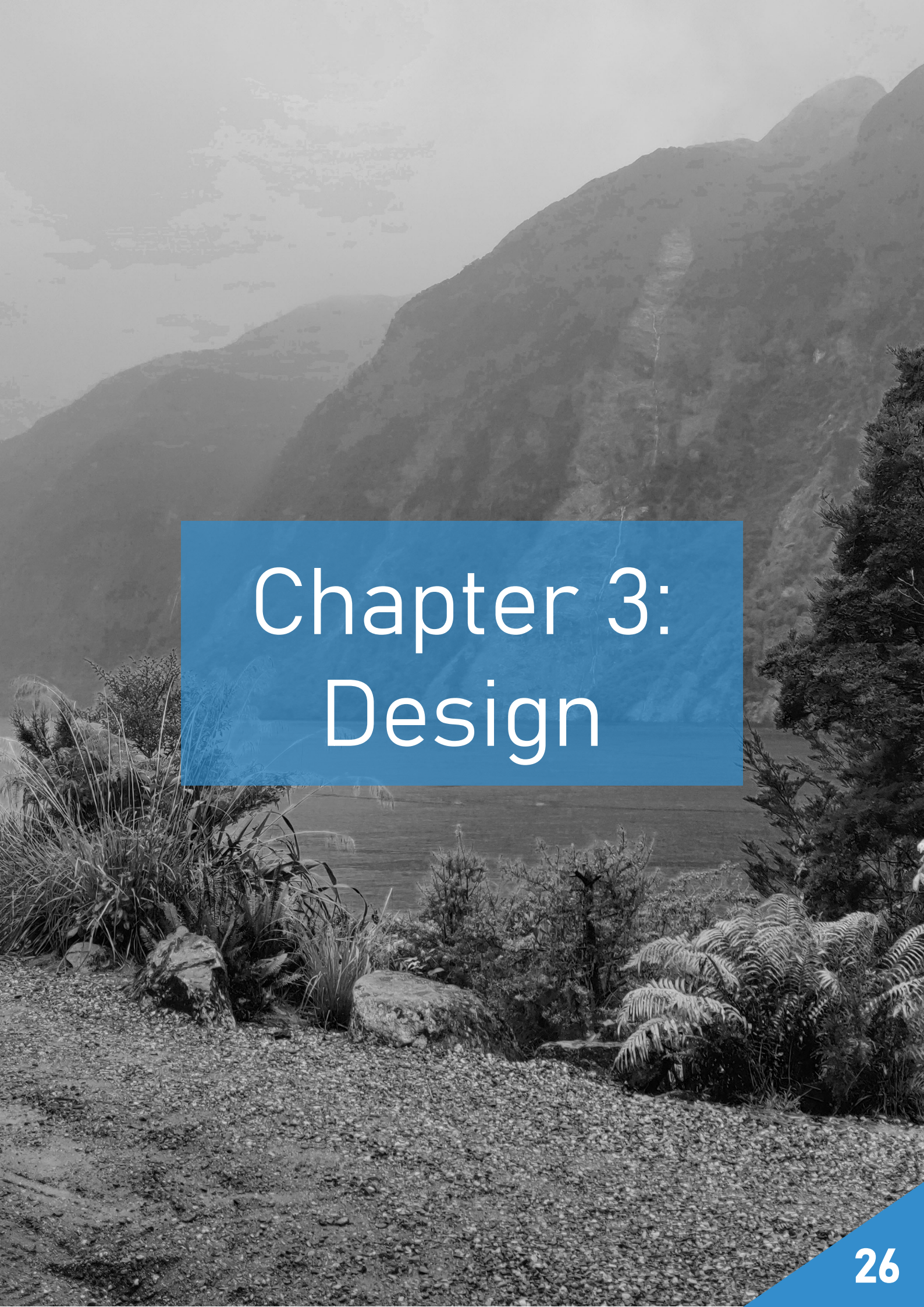
If there was a part of the project which had to be completed before anything else could be done then this would have created a bottleneck and hampered progress. Had this happened there would have been group members not able to do anything which would have been an inefficient use of time. Depending on where this was in the project this could have also caused the project to spend a long time stuck in the early stages with little to no development and functionality meaning there would not be enough time left to finish everything else. The potential impact of this would have been severe and this is a common occurrence in projects leading the group to consider this a high level risk.

Skills shortage - A3

Lacking the necessary skills and/or knowledge to do everything within a project is a common issue. The team had enough knowledge in relevant subjects to make progress while expanding their knowledge in the areas needed to fulfil the specific requirements of the project. Had the knowledge gap been too great to fill in time then the project could have had a significant lack of progress and potentially not even have been completed. The group thought of this as a high level risk due to how likely it was to happen and the extreme impact it could have had on the project.

Overall the risks for this project varied in both severity and likelihood however, through good planning, efficient use of time and consistent work most of these were able to be mitigated. Fortunately for the team the most likely risks were the least severe and the ones that would have had a greater impact were far less likely. This inverse relationship between likelihood and severity of impact allowed the team to come up with effective risk management strategies which eliminated most of the risks with an exception being the scheduling conflict which was able to be minimised.

TOA



Chapter 3: Design

3.1 Rationale of Design Choices

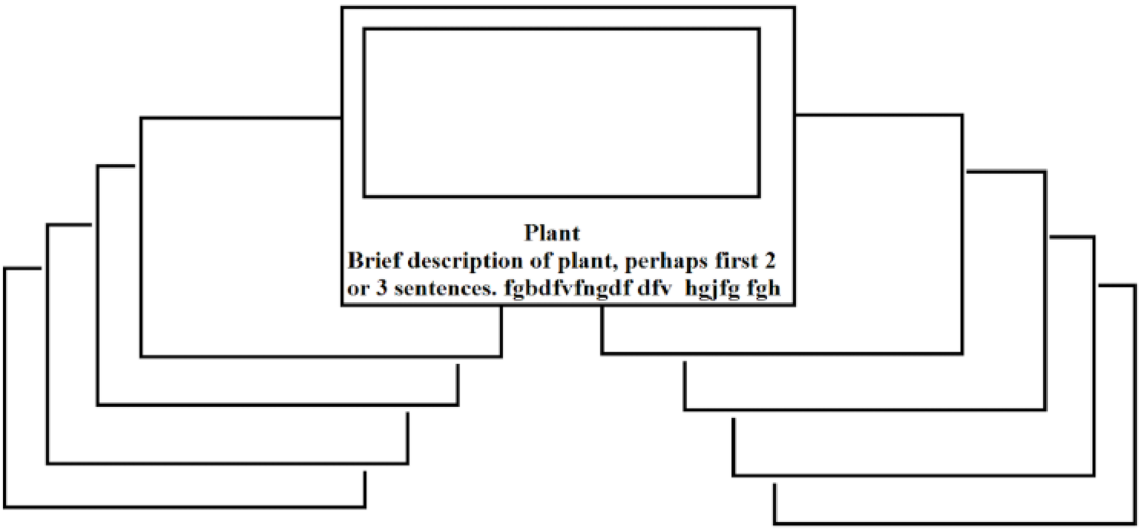
The most noticeable visual element of the app is the dark colour scheme, this is to provide contrast with the mainly bright and colourful pictures, making them stand out more. The system is broken up into several broad categories which are relevant to the theme of nature and the environment of the local area. There are categories for both flora and fauna as well as walking tracks and even games. Having the information categorised logically means users do not have to browse through every single entry within the database and are able to find their desired information faster. When the categories are selected all the entries within them are displayed in a gallery format in alphabetical order. The gallery layout is a minimalist design, just having a picture and title for each entry. Upon selection of an entry users will be taken to the page for that particular item. The combination of logically categorised information and simple layout will allow users to navigate to their desired information quickly. For even faster navigating a search bar was implemented which can be used to search for specific pages within the system. Due to the system primarily being on touch screen devices an onscreen keyboard was added so that users could type in the search box. The admin dashboard however, is designed for use on a desktop so an onscreen keyboard was not added.

3.2 Consideration of Alternative Solutions

The research into, and consideration of alternative solutions has already been explained in section 2.2 Research and Analysis. This section will give a brief summary of what was discussed previously. First of all was the consideration of a native application or a web application. A web based application was chosen so that users would be able to view it on their own personal devices. Although web applications tend to be slower this was not an issue as there was not a large amount of data being sent. When considering which server operating system to use there were 2 Microsoft systems and 2 Linux systems. The Microsoft systems the team looked at were Windows Professional 10 and Windows Server 2019. The Linux systems being considered were Ubuntu and Fedora. The team decided to go with a Linux operating system as they are free which is important as it minimizes the cost to the client. Between the 2 Linux systems Ubuntu was selected because it has a longer support period than Fedora (18 months and 12 months respectively). MVC was chosen as the design pattern as the team had some familiarity with it and it is a very popular design method for web development. Within MVC there were 4 frameworks the team researched, ASP.NET, ASP.NET Core, Laravel and Phalcon. The team chose to use ASP.NET Core as they had some experience using it and it is written in C# which the team has a lot of experience with. As for the database the team was researching and narrowed down the choices to Microsoft SQL Server, MYSQL and MariaDB. MariaDB was chosen as it is free and open source. Although MYSQL is also free and open source, Oracle (the developers) also have a DB software which requires payment, this could cause a conflict of interests. In terms of front end development there were 3 frameworks the team was looking into, React.js, Angular and Vue.js. While all 3 are javascript based the team chose to go with React.js. React.js was selected because it offers more freedom than the other 2 as well as being more popular in the industry.

3.3 Technical Descriptors, Mock Ups, Wireframes

Plants



Plants

Plant Name

Brief description of the plant. dfg rth ae sg rrt hrth. rgjhrt gwae gtrj yr tseg. tyjrhegef jythdrergsfdas hgsdaf ethrsg eafs djthsg d. rthgjtreyrt drgrtdfgsy rte k iul ouiytjhrdsgtykutyjr tera ghty jr.

See more

Plant Name

Brief description of the plant. dfg rth ae sg rrt hrth. rgjhrt gwae gtrj yr tseg. tyjrhegef jythdrergsfdas hgsdaf ethrsg eafs djthsg d. rthgjtreyrt drgrtdfgsy rte k iul ouiytjhrdsgtykutyjr tera ghty jr.

See more

Plant Name

Brief description of the plant. dfg rth ae sg rrt hrth. rgjhrt gwae gtrj yr tseg. tyjrhegef jythdrergsfdas hgsdaf ethrsg eafs djthsg d. rthgjtreyrt drgrtdfgsy rte k iul ouiytjhrdsgtykutyjr tera ghty jr.

See more

Plant Name

Brief description of the plant. dfg rth ae sg rrt hrth. rgjhrt gwae gtrj yr tseg. tyjrhegef jythdrergsfdas hgsdaf ethrsg eafs djthsg d. rthgjtreyrt drgrtdfgsy rte k iul ouiytjhrdsgtykutyjr tera ghty jr.

See more

Animals

*Birds
- *Bird1
- *Bird2
- *Bird3

*Mammals
- *Mammal1
- *Mammal2
- *Mammal3

*Fish
- *Fish1
- *Fish2
- *Fish3

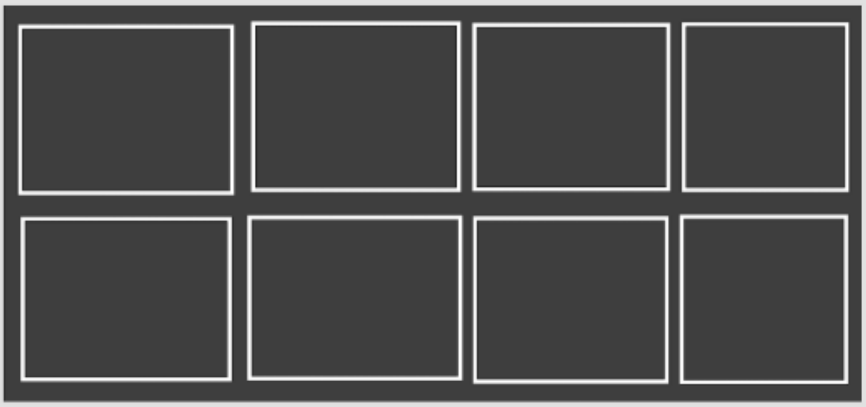
*Insect
- *Insect1
- *Insect2
- *Insect3

*Lizards
- *Lizard1
- *Lizard2
- *Lizard3

Animals

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sodales mattis ex a dictum. Nulla ex massa, malesuada vel egestas non, malesuada vel ante. Vivamus mollis dignissim aliquet. Maecenas pretium arcu eget dolor condimentum varius. Nullam rutrum leo vitae pharetra tincidunt. Phasellus ac nisl eu elit hendrerit lobortis vehicula eu nunc. Morbi vitae dolor at urna gravida sollicitudin. Nam eget arcu suscipit, porttitor nulla quis, commodo ante. Donec faucibus aliquam dolor et congue. Nullam finibus augue in purus interdum, ut tempor est cursus.

Pellentesque vel aliquam elit. Proin lobortis libero lectus, a ultricies lacus laoreet quis. Pellentesque tincidunt orci elit, eu feugiat neque pharetra non. Sed fringilla convallis leo, eu volutpat arcu porttitor ac. Curabitur aliquet at urna sed porta. Integer eu pharetra felis. Pellentesque quis tempus leo. Curabitur in aliquam nunc. Duis mauris augue, bibendum vitae neque at, tempor malesuada turpis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;



Trees
Flowers
Shrubs
Bushes
Herbs

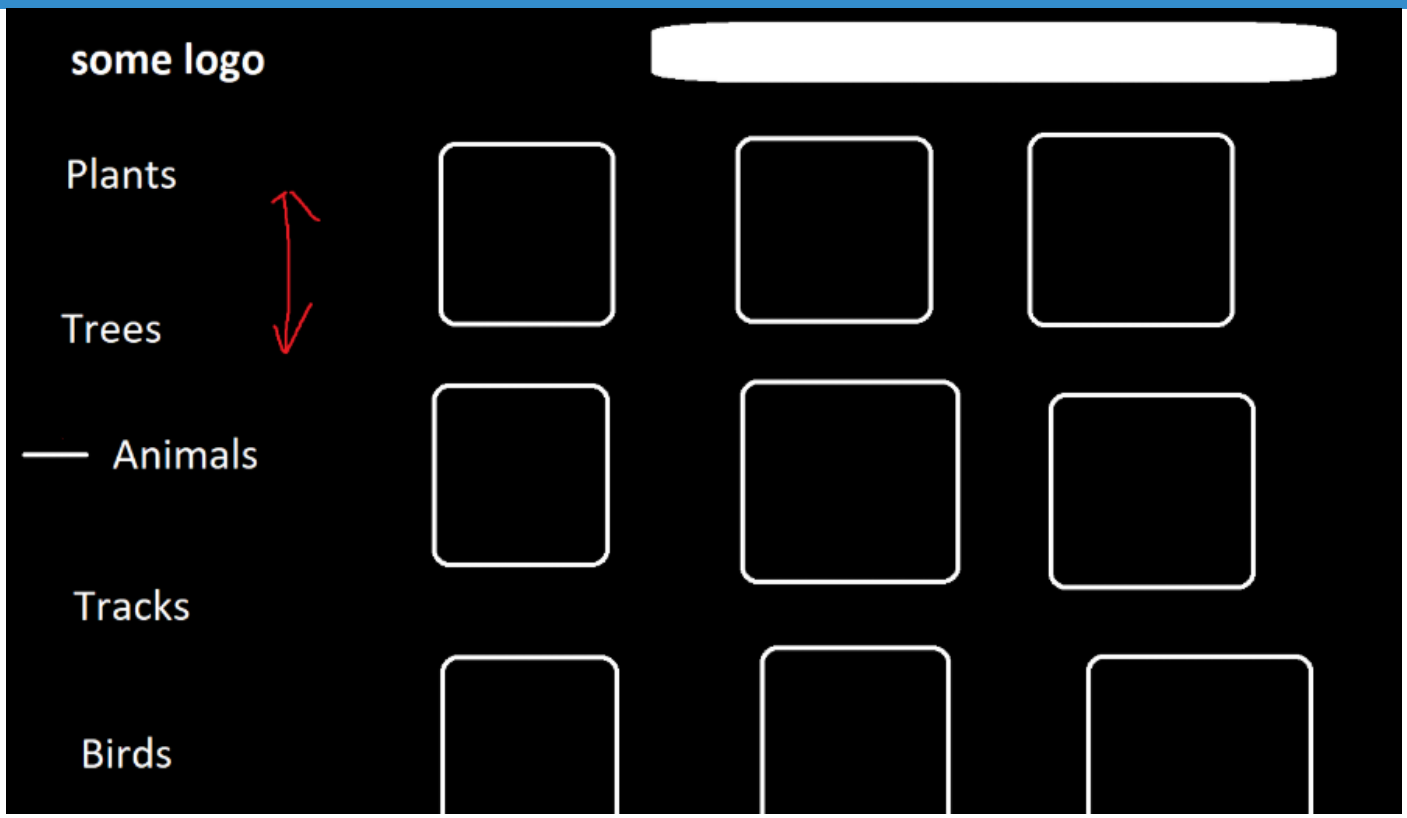
Flower
Flower
Flower
Flower
Flower
Flower
Flower
Flower
Flower
Flower

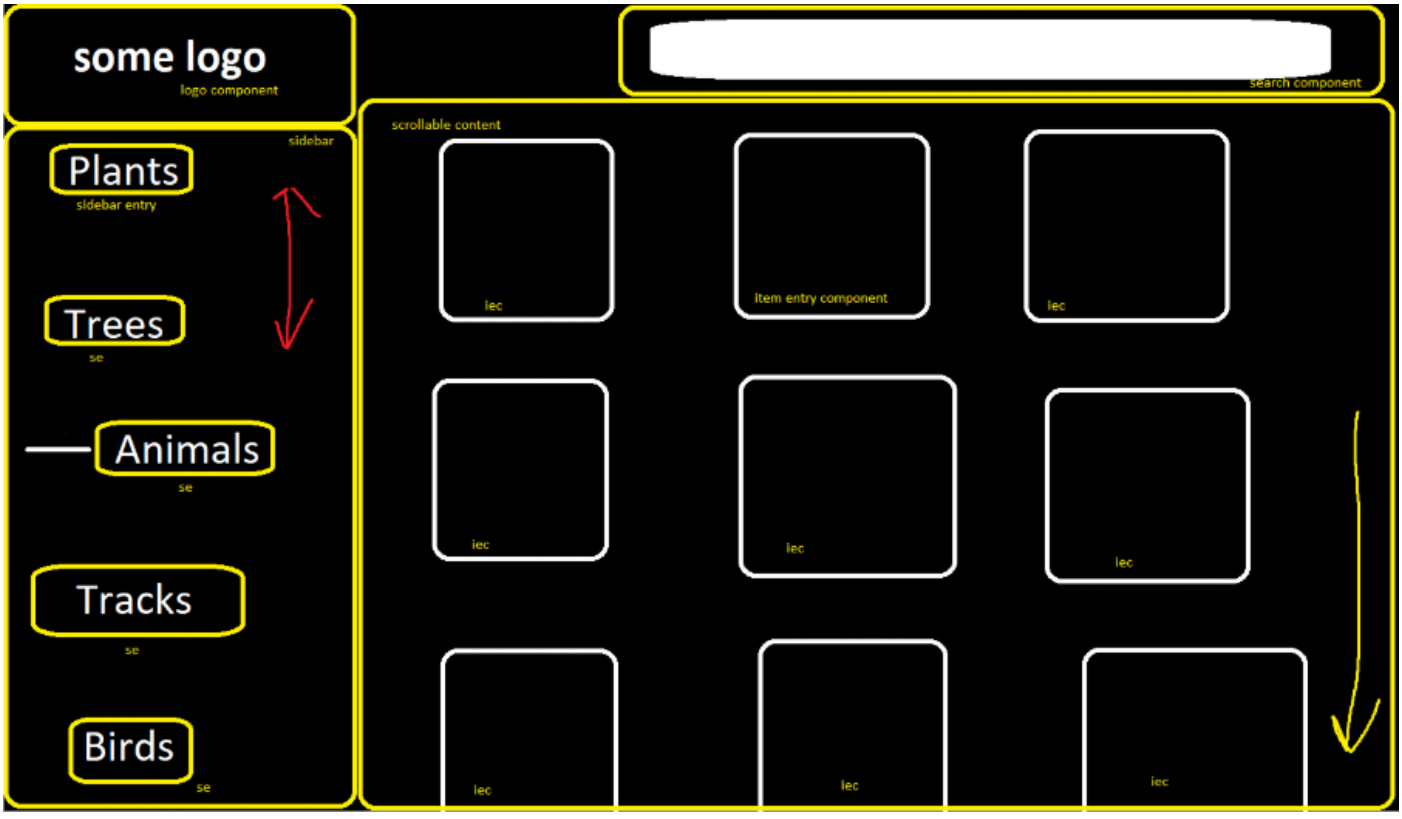
Flower Name

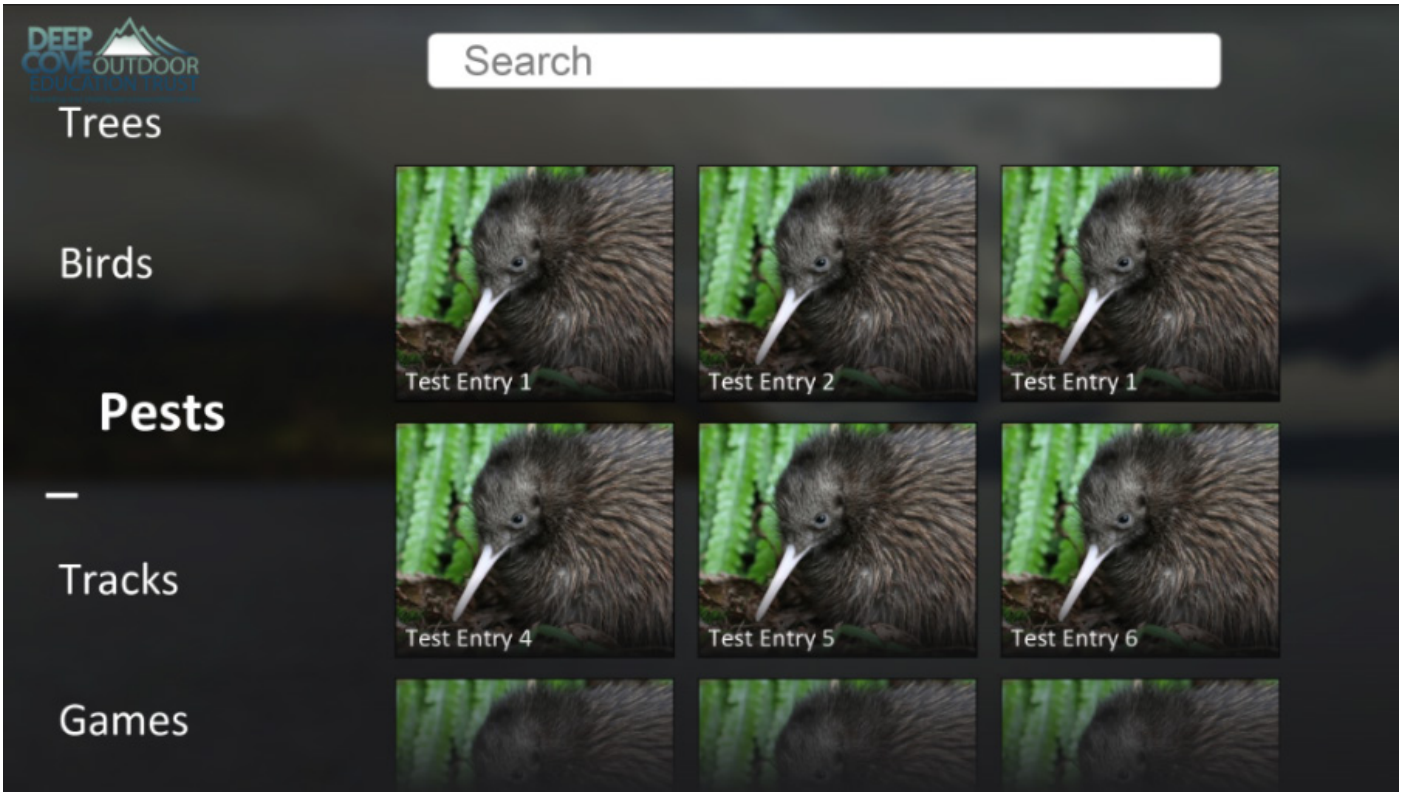
Scientific name - Secondary Name

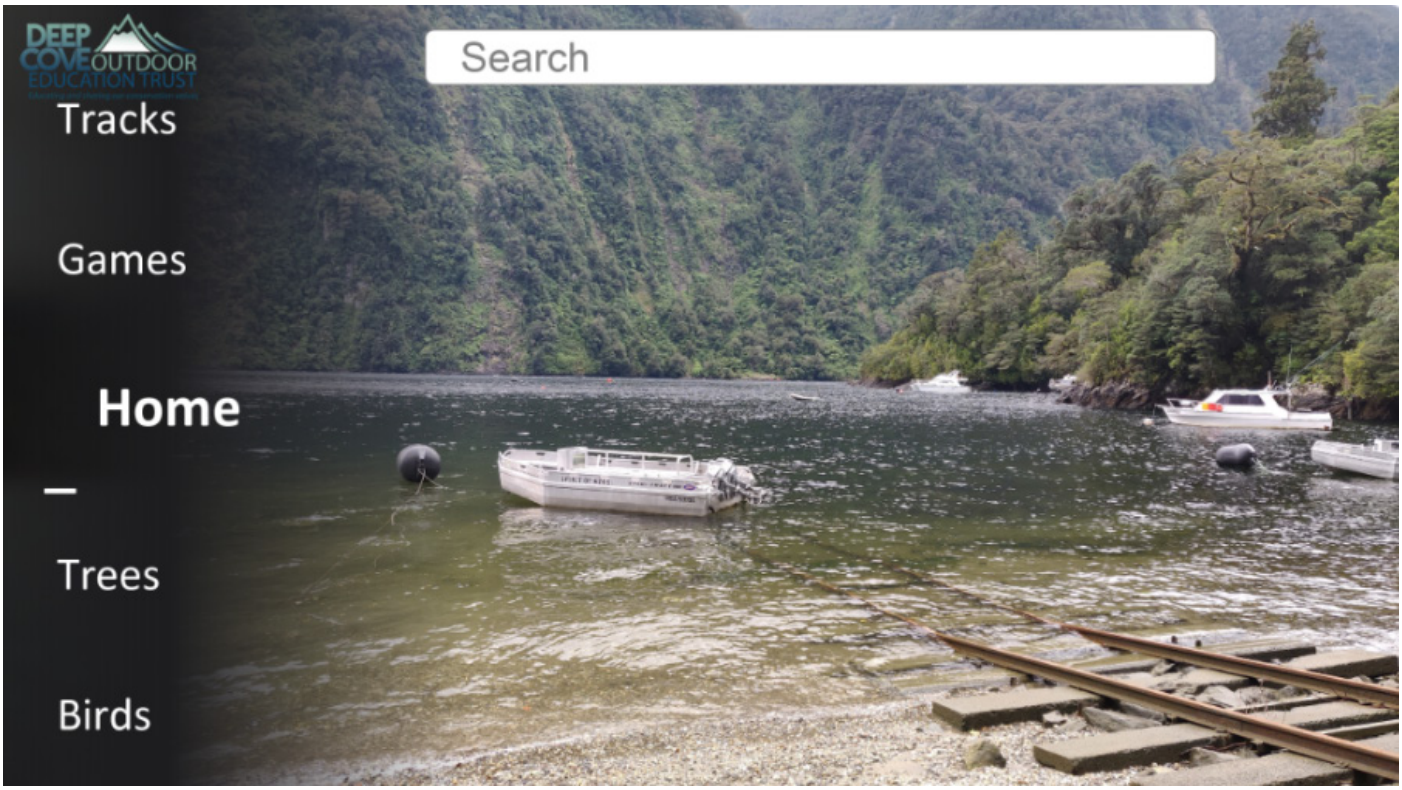
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sodales mattis ex a dictum. Nulla ex massa, malesuada vel egestas non, malesuada vel ante. Vivamus mollis dignissim aliquet. Maecenas pretium arcu eget dolor condimentum varius. Nullam rutrum leo vitae pharetra tincidunt. Phasellus ac nisl eu elit hendrerit lobortis vehicula eu nunc. Morbi vitae dolor at urna gravida sollicitudin. Nam eget arcu suscipit, porttitor nulla quis, commodo ante. Donec faucibus aliquam dolor et congue. Nullam finibus augue in purus interdum, ut tempor est cursus.

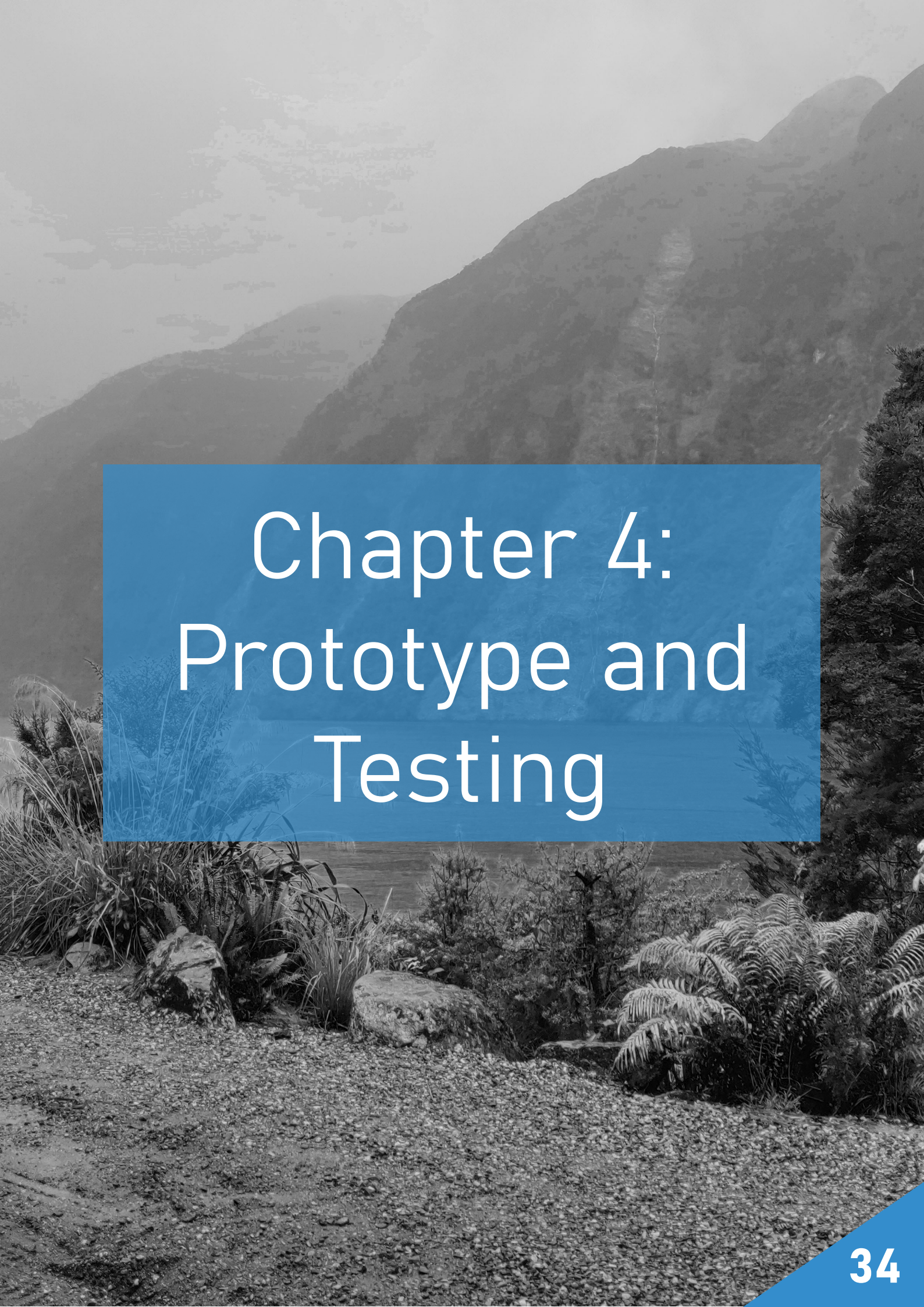
Pellentesque vel aliquam elit. Proin lobortis libero lectus, a ultricies lacus laoreet quis. Pellentesque tincidunt orci elit, eu feugiat neque pharetra non. Sed fringilla convallis leo, eu volutpat arcu porttitor ac. Curabitur aliquet at urna sed porta. Integer eu pharetra felis. Pellentesque quis tempus leo. Curabitur in aliquam nunc. Duis mauris augue, bibendum vitae neque at, tempor malesuada turpis. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae;


some logo
Plants
Trees
Animals
Tracks
Birds


some logo
logo component
search component
sidebar
Plants
sidebar entry
Trees
se
Animals
se
Tracks
se
Birds
se
scrollable content
item entry component
iec
iec
iec
iec
iec
iec
iec
iec


DEEP COVE OUTDOOR EDUCATION TRUST
Search
Trees
Birds
Pests
Tracks
Games
Test Entry 1
Test Entry 2
Test Entry 1
Test Entry 4
Test Entry 5
Test Entry 6


DEEP COVE OUTDOOR EDUCATION TRUST
Search
Tracks
Games
Home
Trees
Birds



Chapter 4: Prototype and Testing

4.1 Description of Prototype Development

The first step in the development of the prototype was the team meeting with the client and gathering functional and non-functional requirements for the kiosk system. Once the team had the requirements they began researching other kiosk systems to get a general understanding of standard practise as well as ideas on how to fulfil the various client requests and how the ideas would translate on to the desired medium. Upon completing the research phase the team began designing the system, both the logical and the User Interface (UI) layout. The logic of the system was designed by creating several logic structure representations such as an Entity Relationship Diagram (ERD), a Sequence Diagram and a Data-flow Diagram (DFD). The UI was designed by members of the team creating concept art based on their ideas for the design of the system and then showing the concepts to the rest of the team. The team took concepts which they agreed on from all of the art and then designed the final UI layout based around them.

With the logic and layout both designed the team began developing the kiosk system itself. Creating the database was the first task completed in this process. The database was created using MySQL. Once the database was created and functional the team moved on to creating the React.js framework of the main kiosk system and creating several games to go in it. The first functionality implemented in to the kiosk system was ensuring that it interacted with the database correctly. When the kiosk was fetching and displaying the correct data in the correct places it was then made to conform to the team's UI layout design. The next functionality added to the kiosk was being able to scroll through the categories and select pages. Next an "admin" mode was added which allowed authorized users to create, delete and edit pages. At this point the games were integrated into the system. Once the team met with the client again to show them the system the client made additional requests regarding the functionality. One of the new requests was to integrate PDF's into the system so that the client could upload teaching/lecture materials, this was the first addition completed. The next additional feature was the implementation of a high score system in the games which records the top players scores and names then saved them in the database. The client also requested that it was possible to put videos into the kiosk system, this was the next request the team fulfilled.

Throughout the development of the kiosk extra touches were continuously being added to the UI of the system to give it a more finished look. When a user selects a particular item they are taken to a page with an image filling up most of the screen and a sidebar next to it containing relevant information. One of the additional stylings added was the sidebar changing colour based on the most dominant colour of the associated image. Fade-in animations were also added to the pages and images so when users go to a different page or scroll switch images they load in smoother rather than just suddenly appearing.

4.2 Pre-Implementation/Post-Implementation

The team continuously tested functionality throughout the duration of the project to ensure the client's requirements were fulfilled. Prior to travelling to Deep Cove and implementing the system these tests were primarily conducted on the team members surface pro's and a touch screen monitor which was provided by the Southern Institute of Technology.

The first feature the team tested was ensuring that the kiosk system interacted with the database as intended. The passing condition of the test was for the system to fetch the correct data. If data was not fetched from the database or it was fetched but not displayed that would have resulted in a failure. If the data was fetched and displayed but was displayed in the wrong location or was not the correct data that would have also been considered a failure. To test this feature dummy data was put in to the database and then displayed on the screen, the team verified that each piece of data was displayed in the correct location and the system passed the test.

One of the most important features to be tested was verifying that the system worked on touch screen devices. The conditions for this test were simple, if the system registered touch screen inputs and performed the correct action then it would pass otherwise it would be a failure. The tests for this feature was that every time new functionality was added the team would open the system on touch screen devices which each member had and see if it performed the desired actions. Any time the system did not pass the test the code was altered until it performed correctly.

A significant feature of the project was ensuring it had quick load times. Similarly to touch screen compatibil-

ity this was also continuously tested throughout the duration of the project. To test this the team would constantly browse through and edit multiple pages. If the team felt that pages or functions took too long to load then it would be deemed a failure and things would be reworked to ensure faster loading times.

It was important for the system to be user friendly and easy to navigate through. The test for this was to have people who were not in the team (and thus had no knowledge of the system) to use the system. If the users were able to use the system without needing the team to explain it to them then it was considered a success, if they were not able to, then it would have been a failure.

> "The team continuously tested functionality throughout the duration of the project to ensure the client's requirements were fulfilled."

There was also an administrator mode in the system which the team tested. To test this the team went in to the administrator mode and then attempted to perform the expected functions. The administrator functions were editing the content of a page, deleting pages, adding pages and changing the layout of pages. The test was considered if the team was able to log in to admin mode and perform all of the functions. If admin mode could not be logged in to or it was logged in to successfully but could not perform the expected functions then it would have been considered a failure.

Data retention during power outages was a crucial feature of the system. To test this the team simulated unexpected power outages and then checked to see if data was deleted or not when the system gained power again. If there was any loss or modification of data at all after coming back online then the test was deemed a failure, if not then it was considered a success.

Another aspect involved in making sure the system was simple was creating it so that the client would be able to set up additional kiosks by themselves. This was important so that they would not have to call the team in to do it for them in the future. To test this the team had people who were not involved with the development of the system to try and install it on a device and set it up without being instructed how to. If people were able to install it and set it up by themselves then the system passed the test, if they were not able to, then it would have been a failure.

After implementing the kiosk system only basic support will be offered. In the event something major was to go wrong, the trust can contact us for an emergency fix where required. Since the project itself is complete in its current state, should the client wish for additional features later on these will be passed onto future project teams at the Southern Institute of Technology. Any basic issues that could occur with the system along with solutions to these will be covered in the user manual.

4.2.1 Kiosk Functions

Test Description	Expected Outcome	Actual Outcome	Status
Raspberry Pi works	The Raspberry Pi powers on, boots to the operating system and is immediately locked into kiosk mode.	The Raspberry Pi powers on, boots to the operating system and is immediately locked into kiosk mode.	PASS
Server machine works	The server machine boots properly, and the web server is launched at startup.	The server machine boots properly, and the web server is launched at startup.	PASS
Web server works	The web server is externally accessible, it correctly processes requests and returns the correct response.	The web server is externally accessible, it correctly processes requests and returns the correct response.	PASS
Web API works	The web API returns the correct information for the request in JSON format.	The web API returns the correct information for the request in JSON format.	PASS
React frontend works	The frontend is loaded with all the correct modules, and correctly navigates between components.	The frontend is loaded with all the correct modules, and correctly navigates between components.	PASS
Frontend fetches data for homepage	The frontend successfully makes API calls to load data to display.	The frontend successfully makes API calls to load data to display.	PASS
Frontend displays data for homepage	The fetched data is correctly processed and displayed on the kiosk homepage view.	The fetched data is correctly processed and displayed on the kiosk homepage view.	PASS
Categories work	Selecting a category darkens the page, and populates the main view with all of the items that are in that category.	Selecting a category darkens the page, and populates the main view with all of the items that are in that category.	PASS
Homepage to kiosk navigation works	Selecting an item on the kiosk homepage redirects to the correct kiosk page.	Selecting an item on the kiosk homepage redirects to the correct kiosk page.	PASS
Frontend fetches data for kiosk page	The frontend successfully makes API calls to load data to display.	The frontend successfully makes API calls to load data to display.	PASS
Frontend displays data for kiosk page	The fetched data is correctly processed and displayed on the kiosk page view.	The fetched data is correctly processed and displayed on the kiosk page view.	PASS
Image carousel works	Swiping through images left to right or right to left functions as intended.	Swiping through images left to right or right to left functions as intended.	PASS

Left box scrolls	The left-hand box of the kiosk page allows vertical traversal by means of scrolling, and hides overflowing content with a blur effect.	The left-hand box of the kiosk page allows vertical traversal by means of scrolling, and hides overflowing content with a blur effect.	PASS
Left box hides	Tapping the tab button on the left-hand box causes it to slide to the left and out of view, allowing maximum viewing of the image.	Tapping the tab button on the left-hand box causes it to slide to the left and out of view, allowing maximum viewing of the image.	PASS
Left box reappears	Tapping the tab button peeking from the left causes the left-hand box to return into view, allowing the reading of the information.	Tapping the tab button peeking from the left causes the left-hand box to return into view, allowing the reading of the information.	PASS
Copyright information displays	The copyright information for each individual image appears when that image displays, and disappears when another image is shown. The correct information is displayed.	The copyright information for each individual image appears when that image displays, and disappears when another image is shown. The correct information is displayed.	PASS
Kiosk page to home page navigation works	Tapping the return to home button brings the kiosk back to the kiosk homepage, with the same category selected.	Tapping the return to home button brings the kiosk back to the kiosk homepage, with the same category selected.	PASS
Search bar works	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	PASS
Onscreen keyboard opens	Focusing the search bar causes the onscreen keyboard to appear.	Focusing the search bar causes the onscreen keyboard to appear.	PASS
Onscreen keyboard closes	Losing focus on the search bar causes the onscreen keyboard to disappear.	Losing focus on the search bar causes the onscreen keyboard to disappear.	PASS
Main content view scrolls	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	PASS

4.2.2 Administrative Functions

4.2.2.1 Authentication

Test Description	Expected Outcome	Actual Outcome	Status
Authentication	Attempting to navigate to administrator pages or make API calls without authentication results in a redirect to the login screen.	Attempting to navigate to administrator pages or make API calls without authentication results in a redirect to the login screen.	PASS
Successful login	A correct set of credentials authenticates into the system.	A correct set of credentials authenticates into the system.	PASS
Unsuccessful login	An incorrect set of credentials does not authenticate into the system.	An incorrect set of credentials does not authenticate into the system.	PASS
Logout	Clicking the logout button clears the authentication status and redirects to the login page.	Clicking the logout button clears the authentication status and redirects to the login page.	PASS

4.2.2.2 Kiosk Pages

Search bar works	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	PASS
Onscreen keyboard opens	Focusing the search bar causes the onscreen keyboard to appear.	Focusing the search bar causes the onscreen keyboard to appear.	PASS
Onscreen keyboard closes	Losing focus on the search bar causes the onscreen keyboard to disappear.	Losing focus on the search bar causes the onscreen keyboard to disappear.	PASS
Main content view scrolls	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	PASS

4.2.2.3 Kiosk Categories

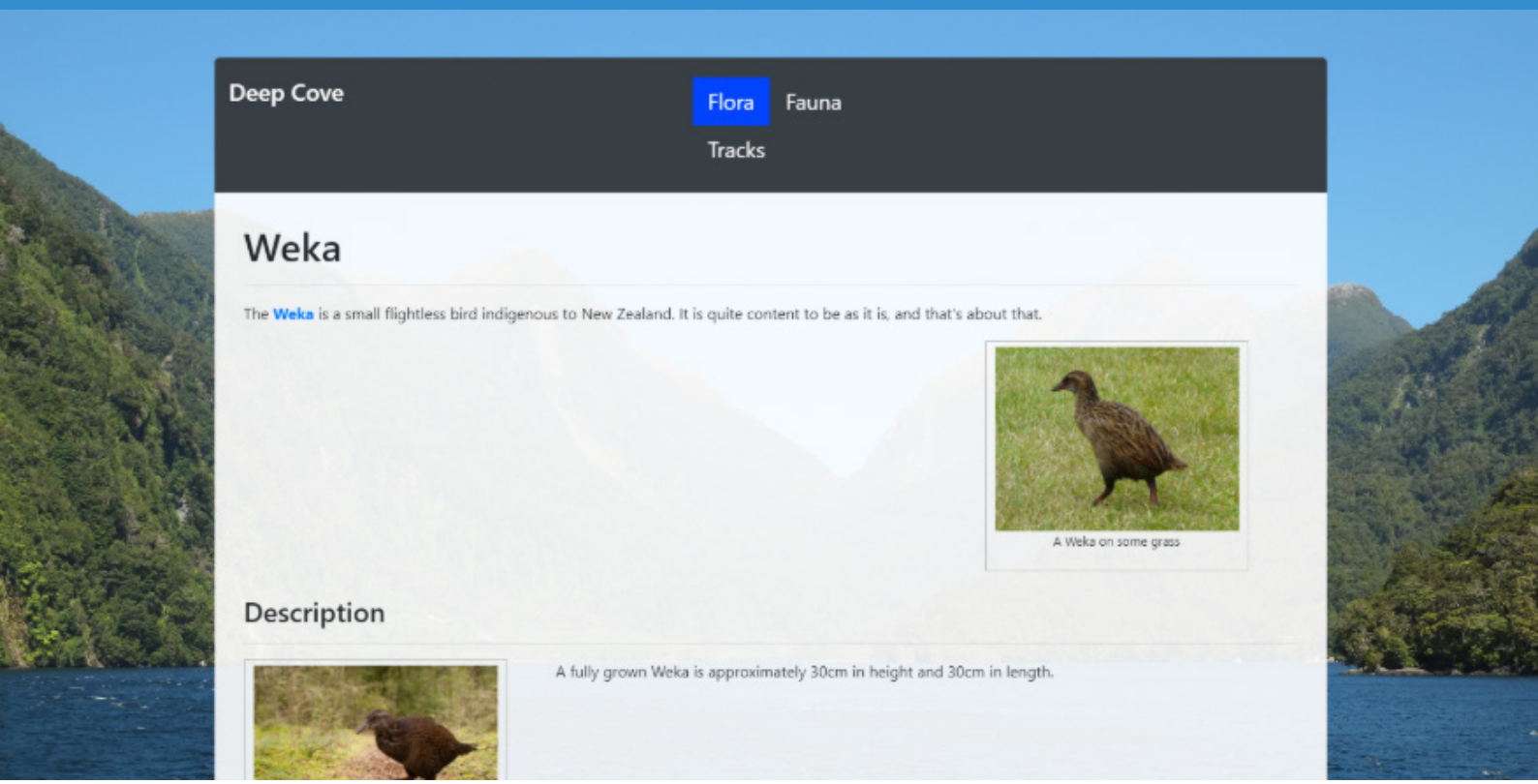
Search bar works	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	PASS
Onscreen keyboard opens	Focusing the search bar causes the onscreen keyboard to appear.	Focusing the search bar causes the onscreen keyboard to appear.	PASS
Onscreen keyboard closes	Losing focus on the search bar causes the onscreen keyboard to disappear.	Losing focus on the search bar causes the onscreen keyboard to disappear.	PASS
Main content view scrolls	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	PASS

4.2.2.4 Powerpoints

Search bar works	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	PASS
Onscreen keyboard opens	Focusing the search bar causes the onscreen keyboard to appear.	Focusing the search bar causes the onscreen keyboard to appear.	PASS
Onscreen keyboard closes	Losing focus on the search bar causes the onscreen keyboard to disappear.	Losing focus on the search bar causes the onscreen keyboard to disappear.	PASS
Main content view scrolls	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	PASS

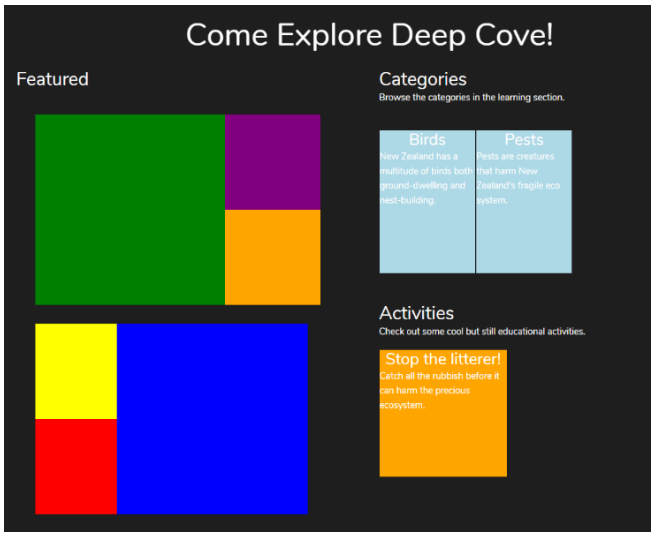
4.2.2.5 Games

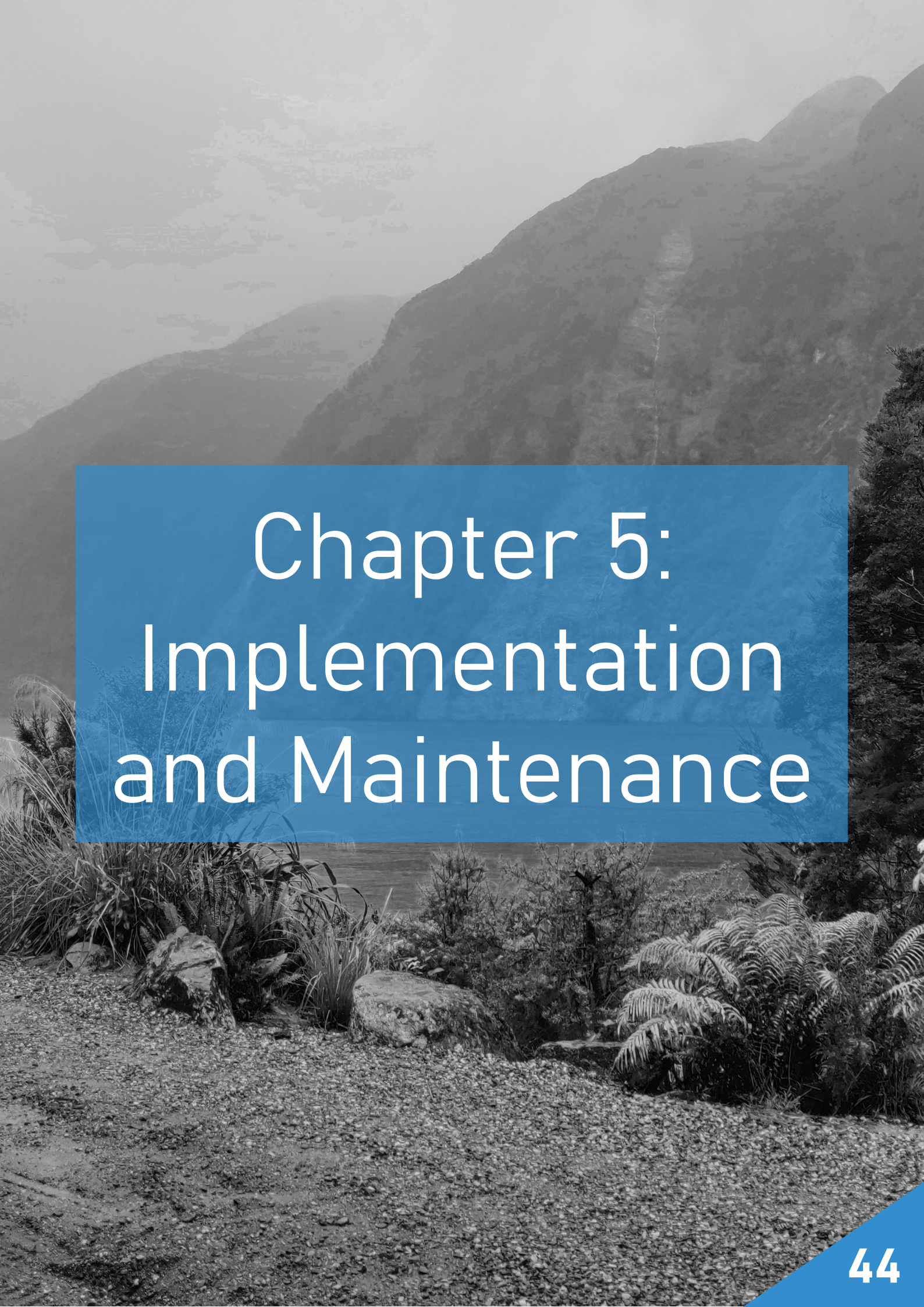
Search bar works	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	When no text is typed into the search bar, categories function as per usual. Typing a term into the search bar causes the current category to be deselected, and all items are filtered based on whether they match the search term. Resulting items are displayed in the main content view.	PASS
Onscreen keyboard opens	Focusing the search bar causes the onscreen keyboard to appear.	Focusing the search bar causes the onscreen keyboard to appear.	PASS
Onscreen keyboard closes	Losing focus on the search bar causes the onscreen keyboard to disappear.	Losing focus on the search bar causes the onscreen keyboard to disappear.	PASS
Main content view scrolls	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	The main content view hides overflowed items, and allows scrolling on a touch device to navigate up and down successfully.	PASS



4.3 Redesign

The original Kiosk design was laid out similarly to a wiki page. There was a background image relevant to the subject with a semi-transparent solid color in front of it which most of the content was placed on. The text was broken up into different subsections and was all the same size with the exception of the page title and section headers. There was also a navigation bar at the top of the page which could be used to navigate between categories. Afterwards it was redesigned to look similar to the windows tile menu. There was a featured section which would just display the most popular pages, a categories section and an activities section. In the featured section there were some large tiles surrounded by smaller tiles, the tiles being pictures and text saying what they lead to. Each tile would rotate between which page they linked to, this change would be displayed as the tiles flipping to new pictures. Besides the featured section there were also a categories section which simply featured some tiles that would only rotate between different categories. There was also an Activities section where the tiles would show various activities to do around Deep Cove or on the kiosk itself. The previous create page featured the ability to add different sections as well as images on either the left or right. These functions were scrapped as the design of the kiosk itself changed so there are no longer subsection and each picture takes up the full screen. The create page now features a section offering a preview of what the page being created will look like before it is actually created.





Chapter 5: Implementation and Maintenance

5.1 Brief Description of Implementation of Final Solution

The final solution is an interactive kiosk system powered by a Raspberry Pi 4 that is configured to open a web browser upon startup. It will then be locked into this web browser in fullscreen mode, which will display on a touch screen to be interacted with by users. The web browser by default will attempt to load the main page of the kiosk application from an onsite web server, which is a single-page application written in React. This single-page application then loads its content from the web server in JSON format, which it then displays. The client is able to modify this content by means of an administrative interface, which is locked behind authentication by means of credentials. This prevents unauthorised and unintended changes to the content. The client is also able to remotely manage the kiosk devices by means of an additional program that communicates with the devices via SSH. With this program, the client is able to reboot devices without the need to physically detach the device from the power supply.

Once we have made our final changes to the main application itself, the first step will be installing this application onto our Linux Server which will be donated to the Deep Cove Outdoor Education Trust upon installing the final system within Deep Cove itself. The apache web server will be configured, and the machine will be setup with private and public SSH keys. VLANs will have to be correctly configured to allow the trust to access the administrative dashboard of our application from their management VLANs, while still preventing guest users from accessing any material outside the application.

5.2 Actions Taken Based on Client Feedback

After the client saw the kiosk system they provided the team with more requests in addition to the ones from the initial meeting. The client asked if the team could integrate powerpoints, word documents, PDFs and videos into the system. There was also a request for the kiosk to contain teacher resources, information about deep cove's history and regulations of the area as well as a meme generator. The final request the client made was a classic arcade style high-score system be added to the games.

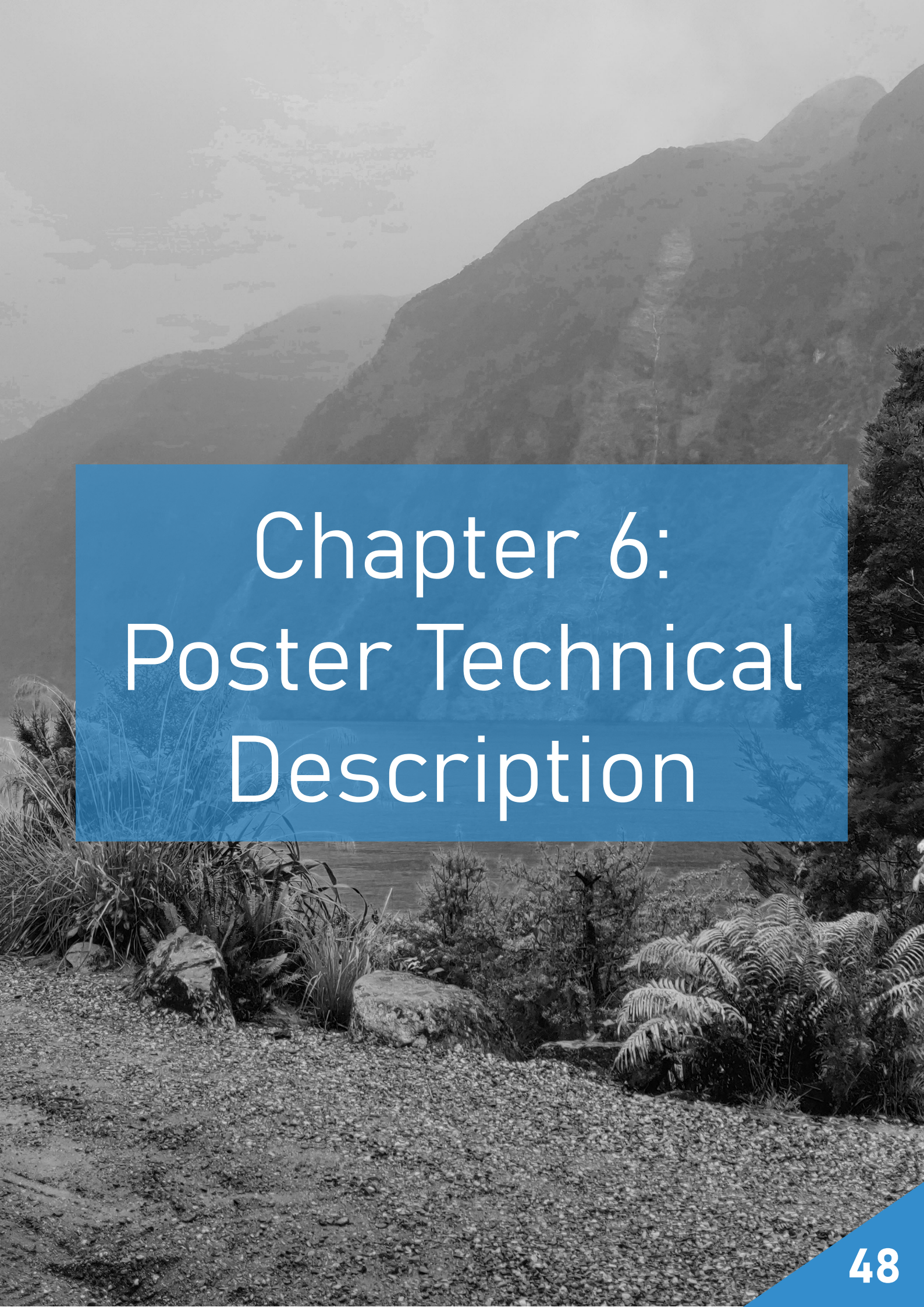
"The high-score system was added after the powerpoints as this required collaboration between the back end developer and the game developer."

The back end developer initially started working to integrate powerpoints. There was difficulty getting the powerpoint files to work correctly within the system, so it was decided to use a series of images to represent the powerpoints with each slide being an image. The team communicated this to the client and they approved the decision as they only need to be able to read off the slides, not edit them.

While the back end developer was working on powerpoints the data manager gathered additional information to add to what the team already had. The data the client asked to include was about the history of the area, the hydro power plant, their water treatment plant and regulations of the area. This was gathered from additional sources provided by the client and then organized in such a way that it could be entered in to the database.

The high-score system was added after the powerpoints as this required collaboration between the back end developer and the game developer. Once a player finished a game if their score was high enough they would be prompted to enter 3 letters to represent a name which would be displayed alongside their score on a high-score screen, just like a classic arcade. This required collaboration between the back-end and game developer's as the score would be sent to the system via an HTTP POST and then stored in the database.

The client asked the team if they could add some way to hide the sidebar that appears on the pages. To fulfil this request a button was added so users could toggle it on or off allowing them to either view the pictures in full screen or have related info displayed partially over them. The final addition to the system was a meme generator which the client requested so that they would be able to post related memes on the deep cove social media profiles.



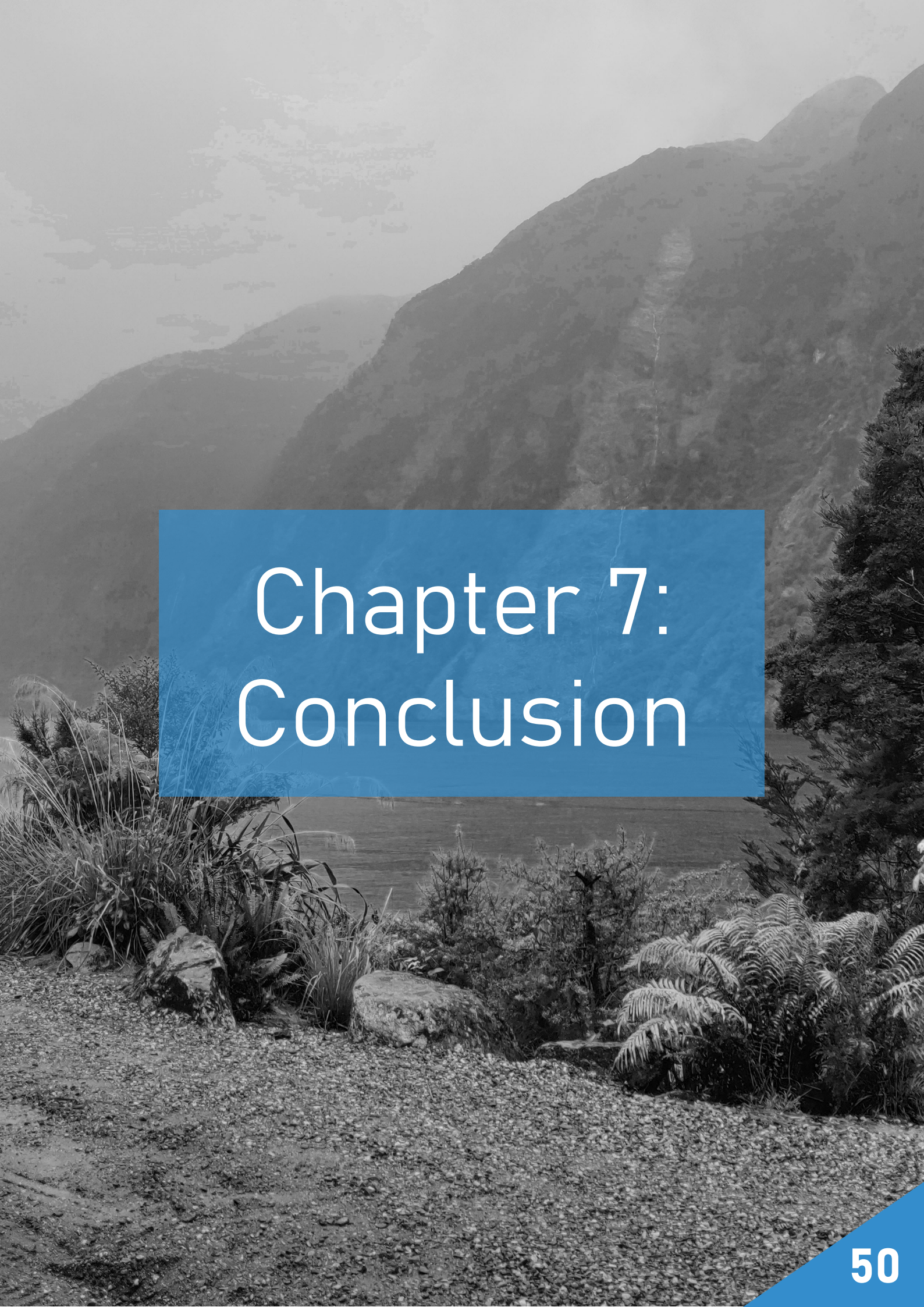
Chapter 6: Poster Technical Description

6.1 Rationale of Poster Elements

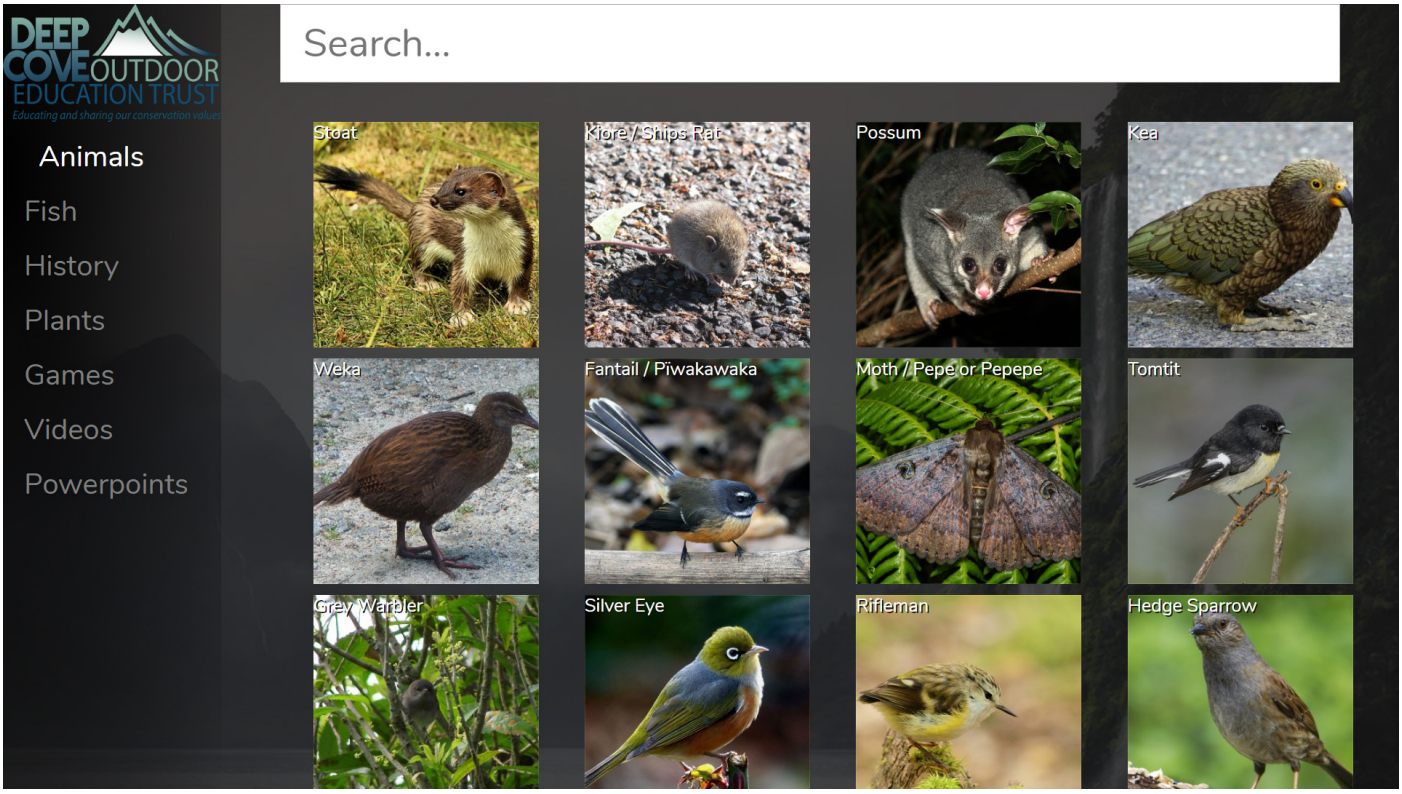
For our project poster, we were looking at just having a simple design that eye catching - but still rather simple. Our original version of our poster used a photoshopped image of the deep cove area, but we found that it took the focus away from the main application and the image on display. This is why we made the transition to simple colours that contrast each other. These colours were colour picked from the Deep Cove Outdoor Education Trust logo itself to ensure they matched. The outline of the mountain range remains, with some minor adjustments to ensure they were more recognizable as mountains. The closer mountains retained a darker colour, whereas the further away ones are contrasted with the lighter choice of colours. We opted to mostly have our poster filled with text at a larger style font. The plan was to ensure the poster was eye catching, while still informative. We wanted our poster's target audience to be all ages, which is why we opted for a design which caters for all ages. Having a tablet based preview of our system, alongside the choice of simple colours not only keeps the poster looking clean but also ensures it is eye catching for all ages.

The poster has information split into multiple categories. First we have the about section, which was written to explain who the Deep Cove Outdoor Education Trust is, and explain what they teach visitors to the area. Following this, we have a brief rundown of our main project. This outlines the problem our client was facing and the solution we came up with to solve said issue. Our poster then goes to discuss the main features of our kiosk based system, and finally gives some special acknowledgements to those who have helped us through the duration of this project.

A copy of the poster has been attached in the Appendices of this report.



Chapter 7: Conclusion



Deep Cove Outdoor Education Trust approached the Southern Institute of Technology requesting an interactive kiosk system for their visitors centre. Our team was assigned to complete this project for them. There was an initial meeting with the client which was used to gather the functional and non-functional requirements of the kiosk. Once the requirements were collected the team researched kiosk systems to see standard practise and how the various requirements could be implemented. The team then went out to the client's location in Deep Cove in order to get an idea of the area and what conditions the system would need to work in. During the trip to deep cove the team also developed a wifi system which generates tickets that give users access to the internet for a limited amount of time. With this completed the team began laying out the logic of the system using UML diagrams and designing concept art of potential UI. Development of the kiosk system then began with each team member assigned a different aspect to work on. Eventually the kiosk system was developed and at the time of writing will soon be installed on-site. Throughout this project there has been a lot of ups and downs, but all of us have worked really hard as a team in order to ensure completion at a quality standard. We would like to thank the Deep Cove Outdoor Education Trust for giving us the oppurtunity to work on a solution like this, and hope it will serve them well in the coming years.

References

.NET Foundation. (2019). Home | Mono. Retrieved 11 9, 2019, from www.mono-project.com: https://www.mono-project.com/

lauxjpn. (2019, 11 7). GitHub - PomeloFoundation/Pomelo.EntityFrameworkCore.MySql. Retrieved 11 9, 2019, from github.com: https://github.com/PomeloFoundation/Pomelo.EntityFrameworkCore.MySql

Microsoft. (2018, 2 23). Database Providers - EF Core | Microsoft Docs. Retrieved 11 9, 2019, from docs.microsoft.com: https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli

Microsoft. (2019, 10 11). Prerequisites for .NET Core on Linux. Retrieved 11 9, 2019, from docs.microsoft.com: https://docs.microsoft.com/en-us/dotnet/core/linux-prerequisites?tabs=netcore30

Oracle. (n.d.). MySQL :: MySQL Community Edition. Retrieved 11 9, 2019, from www.mysql.com: https://www.mysql.com/products/community/

Stack Overflow. (2019). Stack Overflow Developer Survey 2019. Retrieved 11 9, 2019, from iinsights.stackoverflow.com: https://insights.stackoverflow.com/survey/2019

The MariaDB Foundation. (2019). The MariaDB Foundation – Supporting continuity and open collaboration. Retrieved 11 9, 2019, from mariadb.org: https://mariadb.org/

Wallen, J. (2017, 3 22). Ubuntu Server: A cheat sheet - TechRepublic. Retrieved 11 9, 2019, from www.techrepublic.com: https://www.techrepublic.com/article/ubuntu-server-the-smart-persons-guide/

Glossary of Terms

.NET

A developer platform developed by Microsoft that is composed of tools, programming languages, and libraries for building different types of applications.

ASP.NET

Active Server Pages is a web application framework made on the .NET platform.

Intranet

A restricted or local communication network, usually a private network created using World Wide Web software.

IIS (Internet Information Services)

A web server created by Microsoft for the Microsoft NT family of computer systems.

Raspberry Pi

A small single board computer that is used in embedded systems; it can have an operating system installed and it features a host of GPIO pins for electronic interfacing.

Javascript Object Notation

A formatting type for data. Very common in all types of applications, especially in web applications.

Document Object Model

Represents a web page as a hierarchical tree of nodes and objects.

JSX

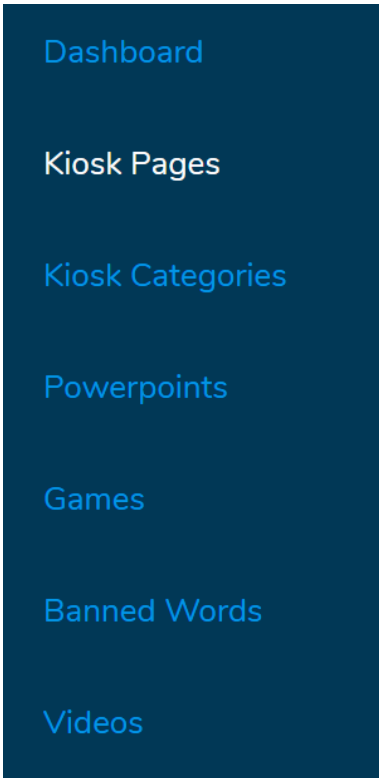
A markup used in React for User Interface definition.

SQL Injection

A hacking technique which involves writing SQL statements into inputs in order to "inject" into the greater SQL string.

User Manual

Managing your Kiosk



This user manual has been designed to help step you through the configuration of your Kiosk System including setup of new kiosks, resetting existing kiosks and modifying displayed content. Please note in order to perform any of the following actions/tasks you must be logged in with an administrative account. These have been supplied seperately to maintain security of the system.

In order to access the administrator controls, navigate to . in your web browser and sign in using the supplied credentials.

The administrative page is navigated using the sidebar on the left of the screen. On a mobile device, this can be viewed by using the "OPEN" tab at the top of your screen. Each main part of the kiosk is seperated into a different menu within this administrative menu, with each page having it's own create and view sections. In order to preview the kiosk system itself, use the "Kiosk View" link at the bottom.

Kiosk Pages

Description

This is the main configuration page you will be accessing in order to modify displayed content on your Kiosk. Kiosk pages are designed to act as a repository of information and can include information such as sounds, images, facts, long descriptions and short descriptions. For adding teaching resources to your Kiosk, please refer to the Powerpoints section of this User Manual.

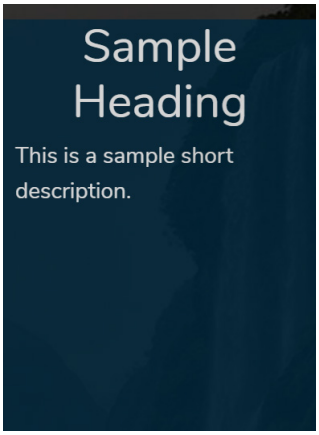
Creating a page

To create a page, navigate to the Create New tab. Creating a page requires several fields:

- A heading
- A short description
- A category to assign it to
- At least one image

Optionally, a page can have:

- A long description
- Up to four statistics
- Audio files - for animal sounds etc.
- Up to 9 additional images



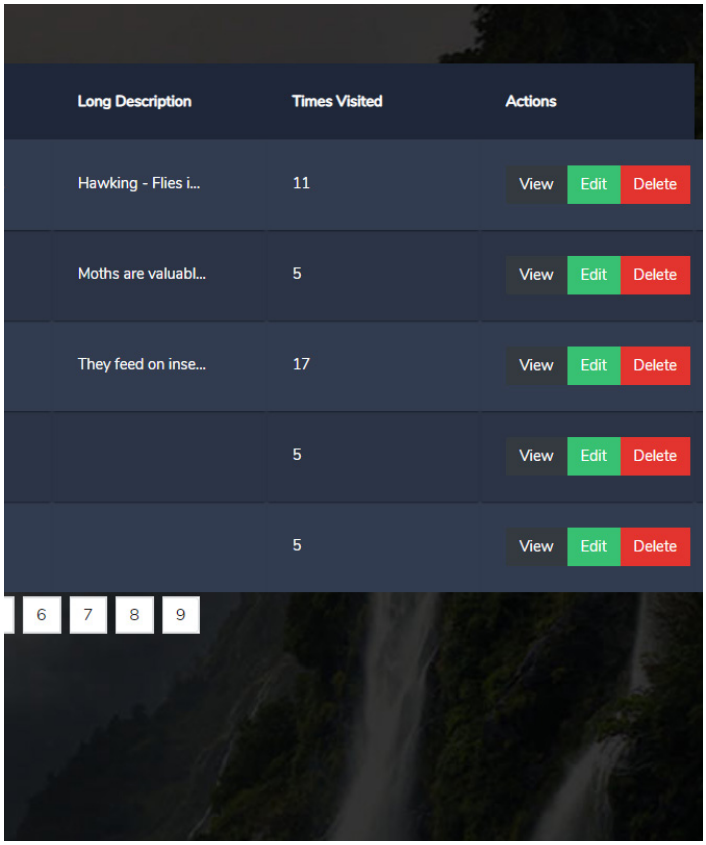
Fill out the required fields, and also the optional fields if desired. A preview of the kiosk page can be seen on the right-hand side. Then, press Submit. When the submission has been completed, a notification will appear as confirmation. The page will then redirect to the View tab of the Kiosk Pages subsection. Upon an error occurring a notification will appear, detailing why the error occurred and possible solutions.

Editing a Page

To edit a page, navigate to the View tab. After locating the desired entry, click the edit button next to the entry to bring up the Edit page. After changing the desired details, click update to store the changes. The page will then redirect to the View tab of the Kiosk Pages subsection. Upon an error occurring a notification will appear, detailing why the error occurred and possible solutions.

Deleting a Page

To delete a page, navigate to the View tab. After locating the desired entry, click the delete button next to the entry. A prompt will appear to prevent accidental deletion. To confirm, click Confirm. To cancel, click Cancel.



Kiosk Categories

Description

Kiosk Categories are designed to seperate Kiosk Pages in the main application. Categories are the different entries that appear on the left hand side of the main application that allow you to quickly filter through entries. In order to use Kiosk Pages, you must first have a Category defined. If you would like to add a custom Kiosk Page and have already created a category for it, please read the "Kiosk Pages" section of this documentation.

Creating a category

To create a category, navigate to the Create New tab from within the Kiosk Categories section of the Administrative page. Creating a category requires:

- A name
- A description

Fill out the required fields, and also the optional fields if desired. Then, press submit. When the submission has been completed, a notification will appear as confirmation.

Editing a category

To edit a category, navigate to the View tab. After locating the desired entry, click the edit button next to the entry to bring up the Edit page. After changing the desired details, click Submit to store the changes.

Deleting a category

To delete a category, navigate to the View tab. After locating the desired entry, click the delete button next to the entry. A prompt will appear to prevent accidental deletion. To confirm, click Confirm. To cancel, click Cancel.

Powerpoints

Description

Powerpoints are designed more for educational resources that can be accessed from any device within Deep Cove, without any software installed. These are suitable for teachers/tutors who would like to have "slides" displayed behind them, quizzes or other material that are not suitable for generic Kiosk Pages. If you would like to add a Kiosk Page entry to a category, please refer to the "Kiosk Pages" section of this documentation.

Converting a Microsoft Powerpoint Presentation

Due to continously changing standards, it was beyond the scope of this project to have the administrative dashboard read in Powerpoint files directly. In order to use these with the system, please follow the following steps to convert a Powerpoint Presentation to Images:

1. From within the powerpoint presentation you would like uploaded, select **File > Save As**.
2. Click **Browse**, and choose a location to save the file.
3. Change the file type to "**PNG**" from the dropdown menu under **Save as Type**.
4. Select **Save**.
5. Select the **Every Slide** option in the resulting popup.

This will create a folder in the directory you specified with an image for each slide. These can later be used to create a powerpoint on the administrative dashboard.

Creating a powerpoint

In order to create a powerpoint, you will require a title and a set of images. The set of images represent each "slide" you would like to have displayed in the presentation.

Navigate to the Create New tab from within the Powerpoints section of the Administrative Page. Enter in a title and select Choose Files. Navigate to the directory with the images you would like to upload for this presentation and select them all. After uploading these, select Submit.

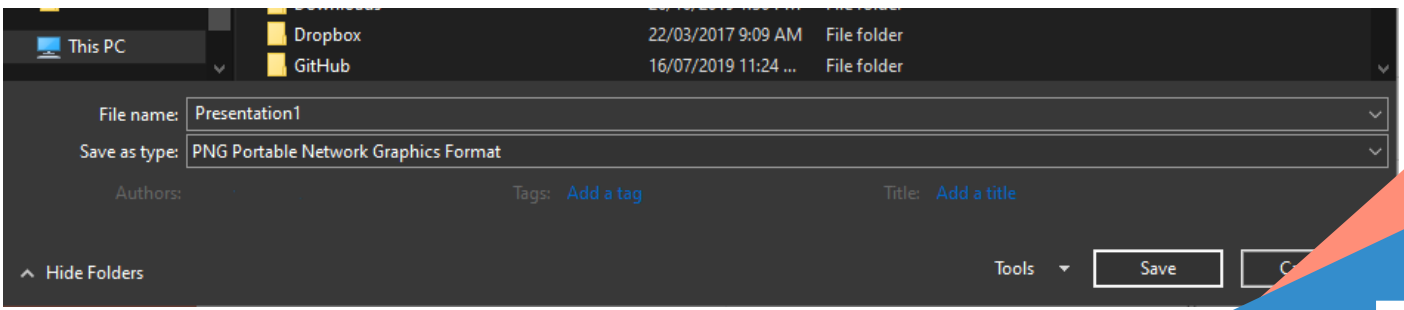
If you would like to convert a Microsoft Powerpoint to a set of images, please refer to the below section "Converting a Microsoft Powerpoint" before continuing these steps.

Editing a powerpoint

To edit a powerpoint, navigate to the View tab. After locating the desired entry, click the Edit button next to the entry to bring up the Edit page. After changing the desired details, click update to store the changes.

Deleting a powerpoint

To delete a powerpoint, navigate to the View tab. After locating the desired entry, click the Delete button next to the entry.



Games

Description

In order to draw more people into using the Kiosk system, you may wish to have interactive games enabled. These are fun interactive activities that are designed to work on touch devices, and many of these may have high score functionality implemented.

Enabling/Disabling a game

If a certain game is causing issues for a particular group of school children, you can disable it by first navigating to the "Games" section of the administrative page, and selecting "Disable" next to the desired game. Should you wish to turn this game back on in the future, simply select "Enable".

View Highscores

It is possible to view a game's highscores from the Administrative Dashboard, as opposed to having to be physically present at the kiosk to check. To view a game's highscores, click the View high-scores button next to the desired game. A list of the ten highscores will appear. To navigate back to the Games view, click the Back button in the top right hand corner.

Clearing Highscores

You may wish to clear high scores from the system to allow a fresh group of peo-ple a shot at the leaderboard. To clear a game's highscores, click the Clear high scores button next to the desired game.

Banned Words

Description

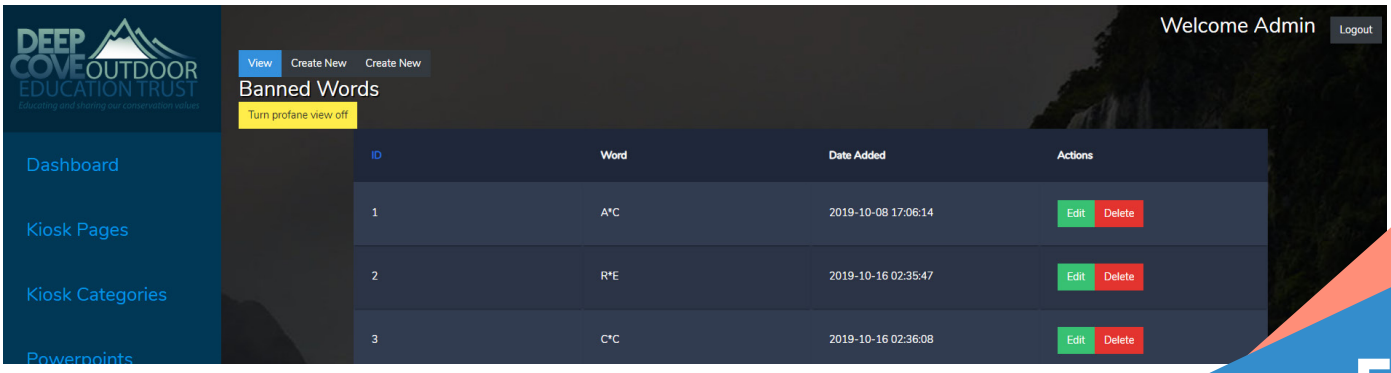
As games provide a high score system, younger audiences may attempt to abuse the ability to enter in custom initials by entering innapropriate words. For this reason you may wish to provide a list of words which will become unselectable as a name in the high score section of a game.

Adding a Banned Word

To create a banned word, navigate to the Create New tab. In the input box provid-ed, type the word of length three to be banned. Now, press the Create button to save the word. This will prevent future use of the word in the high score submit, and replace all current occurrences with a random string of three letters.

Delete a Banned Word

To delete a banned word, navigate to the View tab. After locating the desired entry, click the delete button next to the entry. A prompt will appear to prevent accidental deletion.



Videos

Description

Designed for educational purposes, videos can be uploaded to the Kiosk to be viewed on any device connected to the system.

Creating a video

To create a new video, navigate to the Create New tab of the Videos subsection. A video requires several fields:

- Title - this is what appears on the kiosk homepage under the Videos category as the text on the item.
- Description - this appears under the video player on the video playback page.
- Video file - this is the actual video that will be viewable on the video playback page. Ensure the video and sound quality are sufficient so as to not be off putting to the user.

A video also has two optional fields:

- Thumbnail - this is for an image that will display alongside the title on the kiosk homepage. This is recommended so as to raise user engagement.
- Copyright - this is for any acknowledgements or copyright information pertaining to the video file. This will display on the video playback page under the description.

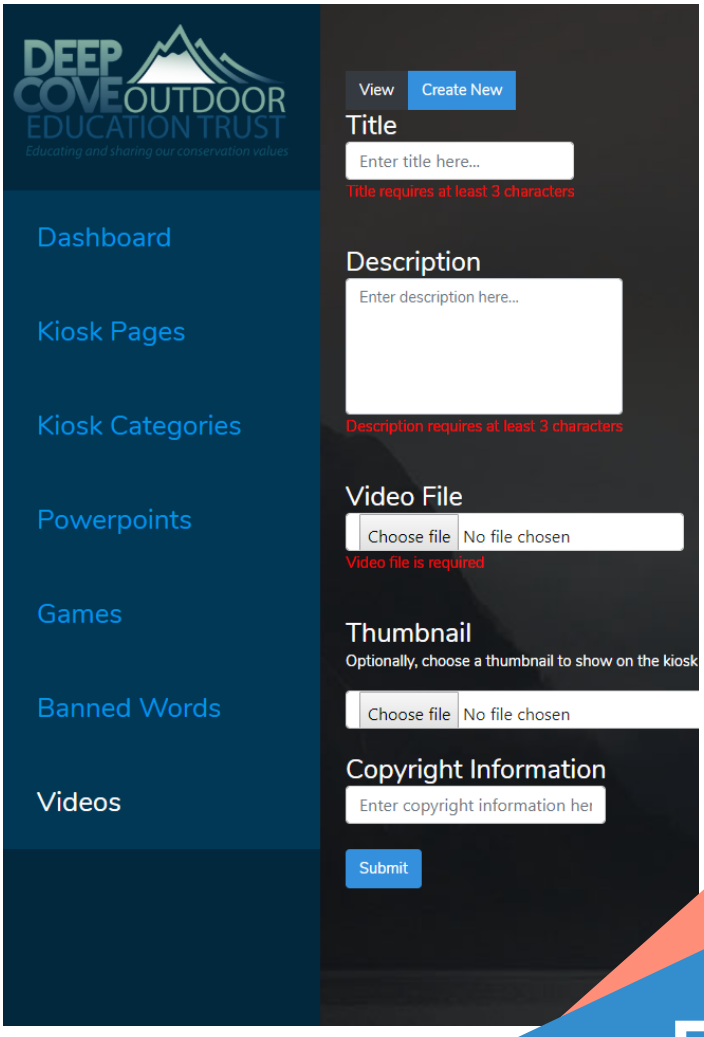
Fill out the required fields, and also the optional fields if desired. Then, press submit. An upload progress bar will appear, tracking the progress of the upload. When the upload has been completed, a notification will appear as confirmation. The page will then redirect to the View tab of the Videos subsection. Upon an error occurring, the progress bar will turn red and the upload will be cancelled. A notification will also appear, detailing why the error occurred and possible solutions.

Editing a video

To edit a video, navigate to the View tab of the Videos subsection. After locating the desired video, click the Edit button next to the entry to bring up the Edit page. Change any desired fields, and then click update to store the new information. The page will then redirect to the View tab of the Videos subsection.

Deleting a video

To delete a video, navigate to the View tab of the Videos subsection. After location the desired video, click the delete button next to the entry. This will bring up a prompt to prevent accidental deletion.




KARMEA
PURE KIWI Mosque

Appendices

Appendix A

Class Diagram

Appendix B

Data Flow Diagram

Appendix C

Entity Relationship Diagram

Appendix D

Sequence Diagram

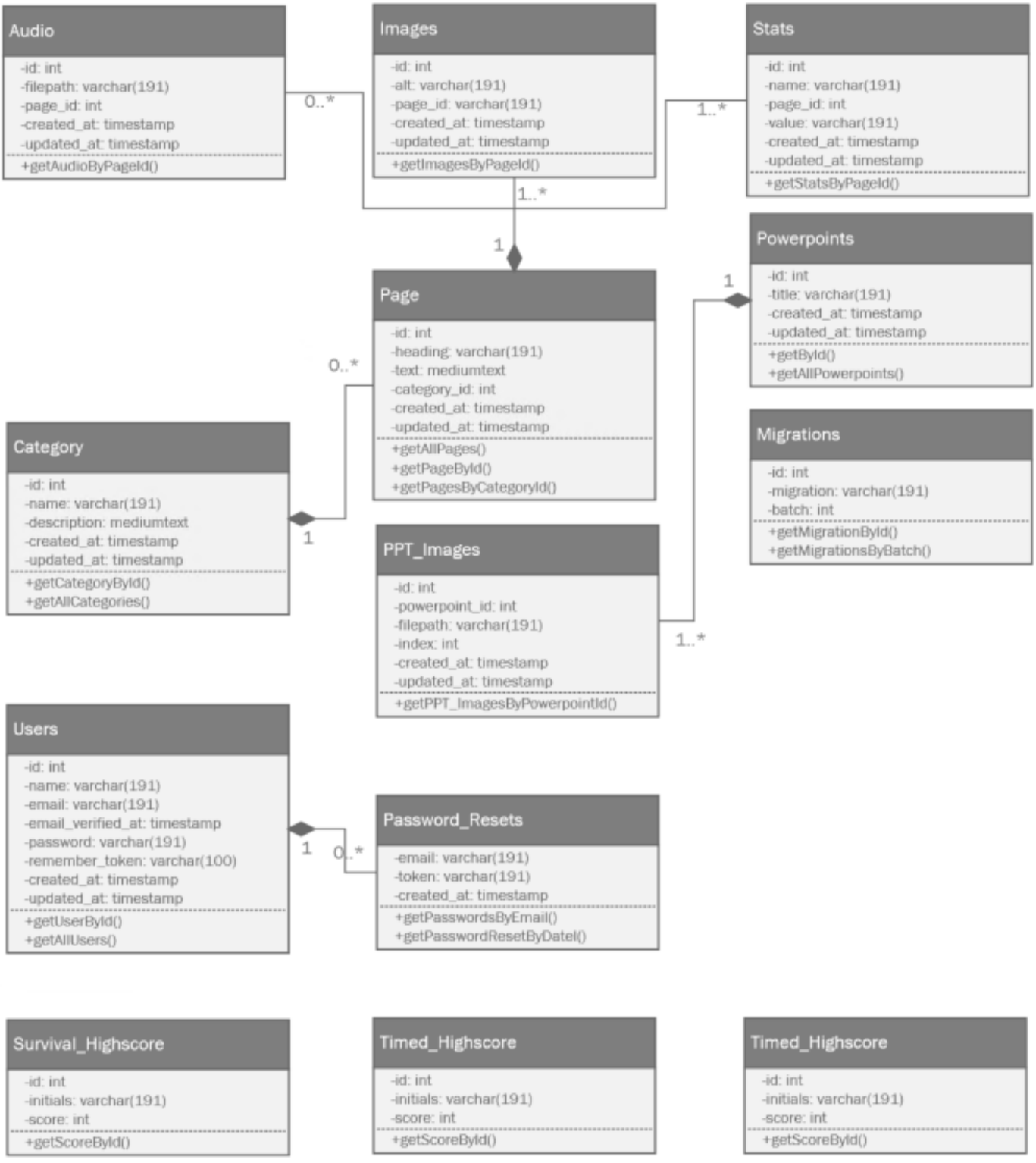
Appendix E

Testing Processes

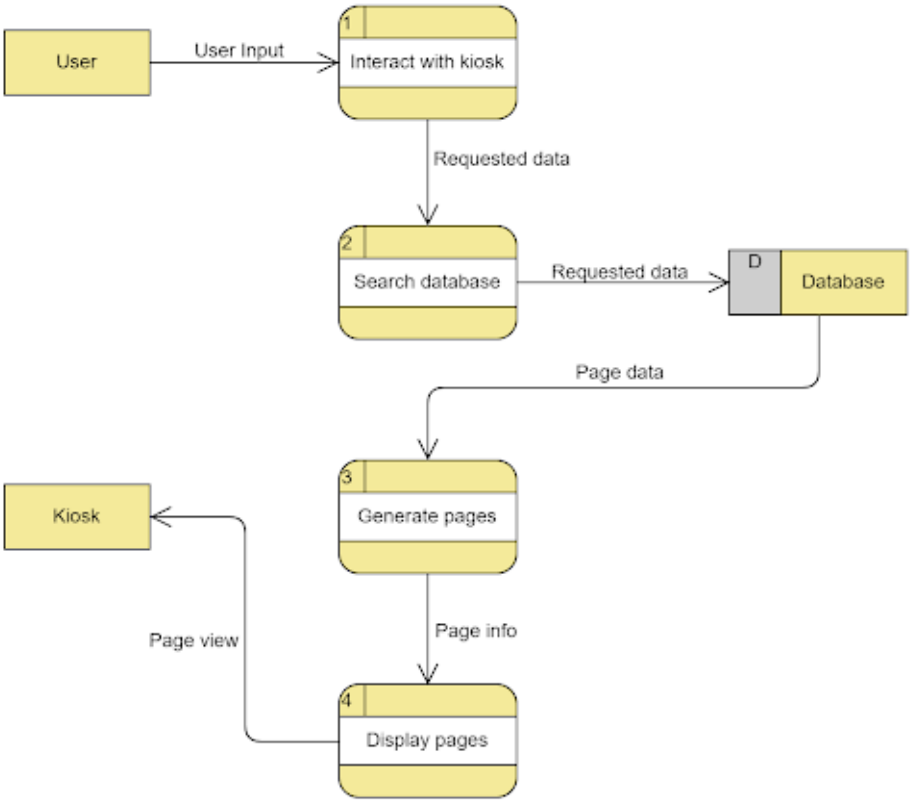
Appendix F

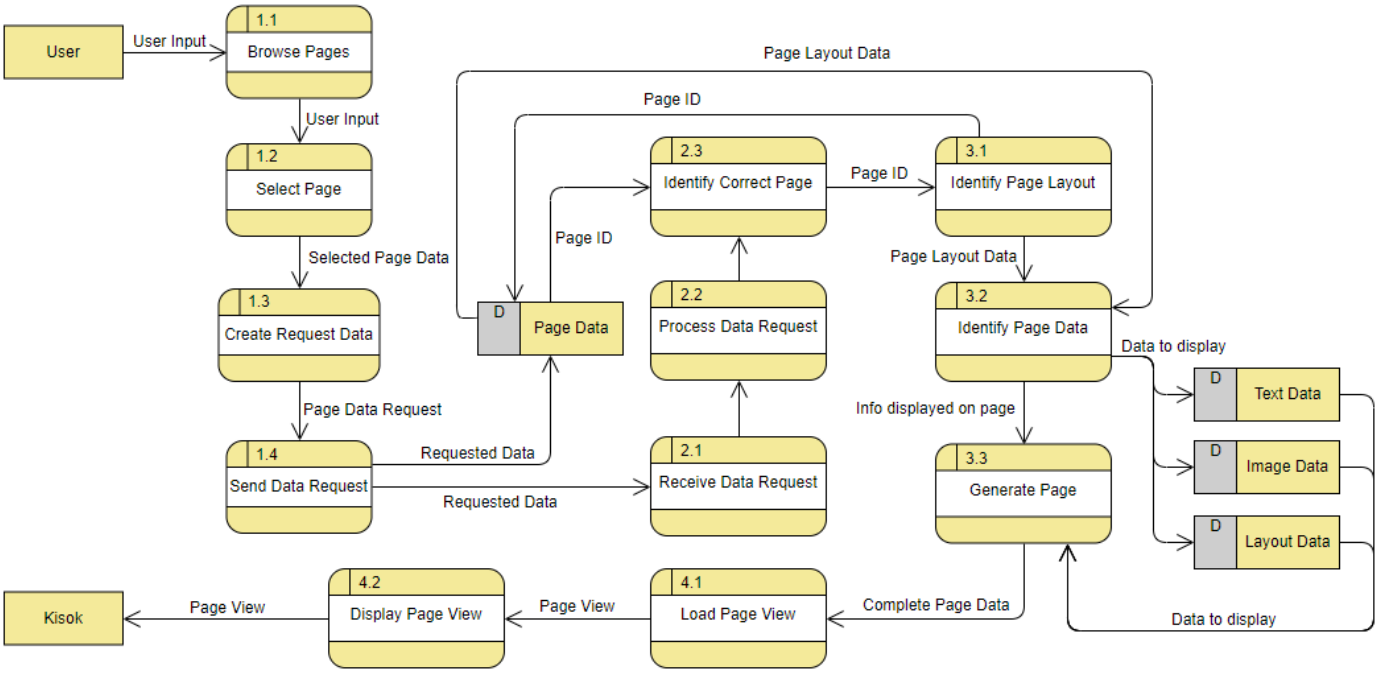
Project Poster

Appendix A

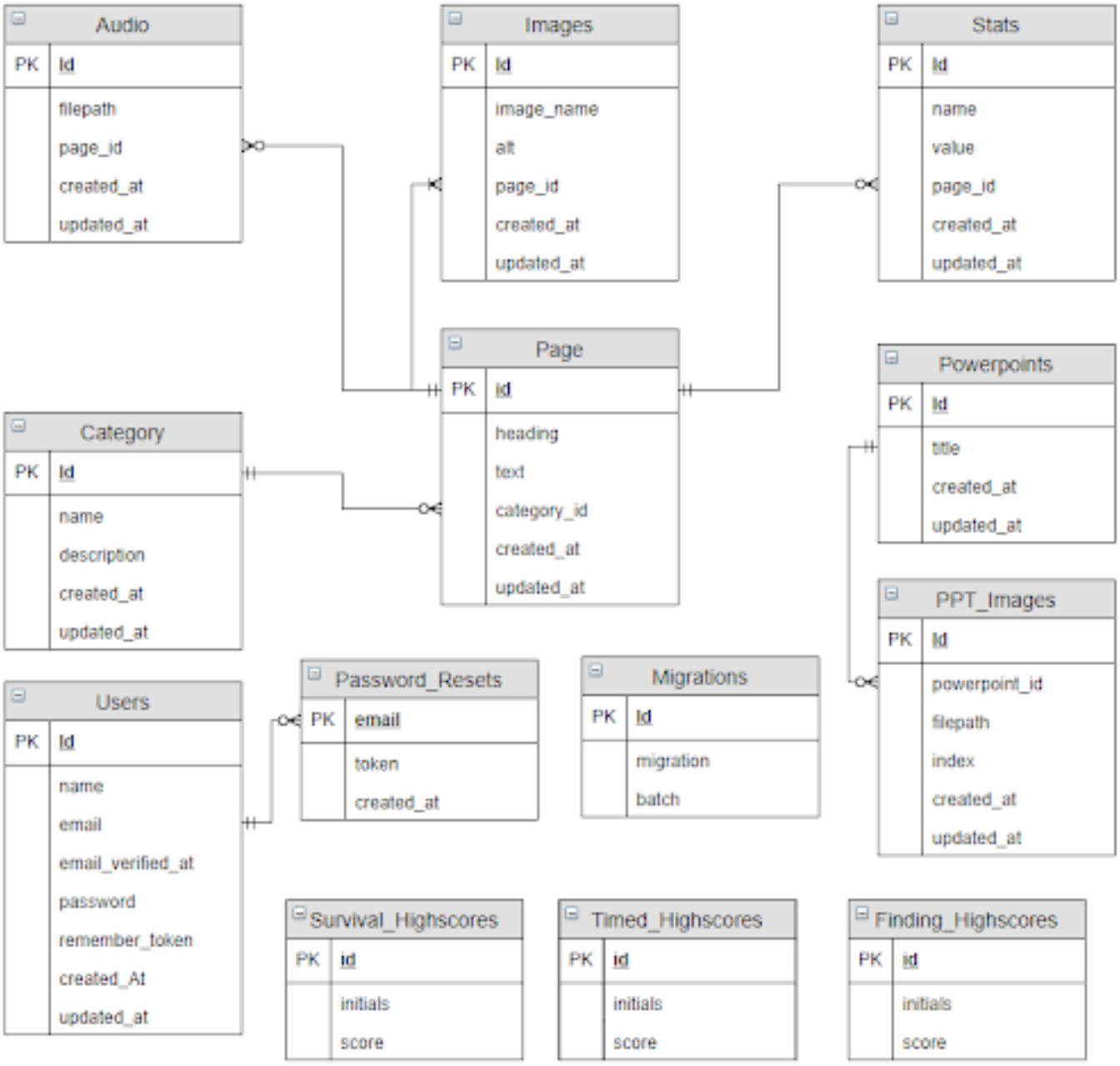

Audio
-id: int
-filepath: varchar(191)
-page_id: int
-created_at: timestamp
-updated_at: timestamp
+getAudioByPageId()
Images
-id: int
-alt: varchar(191)
-page_id: varchar(191)
-created_at: timestamp
-updated_at: timestamp
+getImagesByPageId()
Stats
-id: int
-name: varchar(191)
-page_id: int
-value: varchar(191)
-created_at: timestamp
-updated_at: timestamp
+getStatsByPageId()
Powerpoints
-id: int
-title: varchar(191)
-created_at: timestamp
-updated_at: timestamp
+getById()
+getAllPowerpoints()
Page
-id: int
-heading: varchar(191)
-text: mediumtext
-category_id: int
-created_at: timestamp
-updated_at: timestamp
+getAllPages()
+getPageById()
+getPagesByCategoryId()
Migrations
-id: int
-migration: varchar(191)
-batch: int
+getMigrationById()
+getMigrationsByBatch()
Category
-id: int
-name: varchar(191)
-description: mediumtext
-created_at: timestamp
-updated_at: timestamp
+getCategoryById()
+getAllCategories()
PPT_Images
-id: int
-powerpoint_id: int
-filepath: varchar(191)
-index: int
-created_at: timestamp
-updated_at: timestamp
+getPPT_ImagesByPowerpointId()
Users
-id: int
-name: varchar(191)
-email: varchar(191)
-email_verified_at: timestamp
-password: varchar(191)
-remember_token: varchar(100)
-created_at: timestamp
-updated_at: timestamp
+getUserById()
+getAllUsers()
Password_Resets
-email: varchar(191)
-token: varchar(191)
-created_at: timestamp
+getPasswordsByEmail()
+getPasswordResetByDate()
Survival_Highscore
-id: int
-initials: varchar(191)
-score: int
+getScoreById()
Timed_Highscore
-id: int
-initials: varchar(191)
-score: int
+getScoreById()
Timed_Highscore
-id: int
-initials: varchar(191)
-score: int
+getScoreById()
0..*
1..*
1
0..*
1
1
1..*
1
0..*

Appendix B

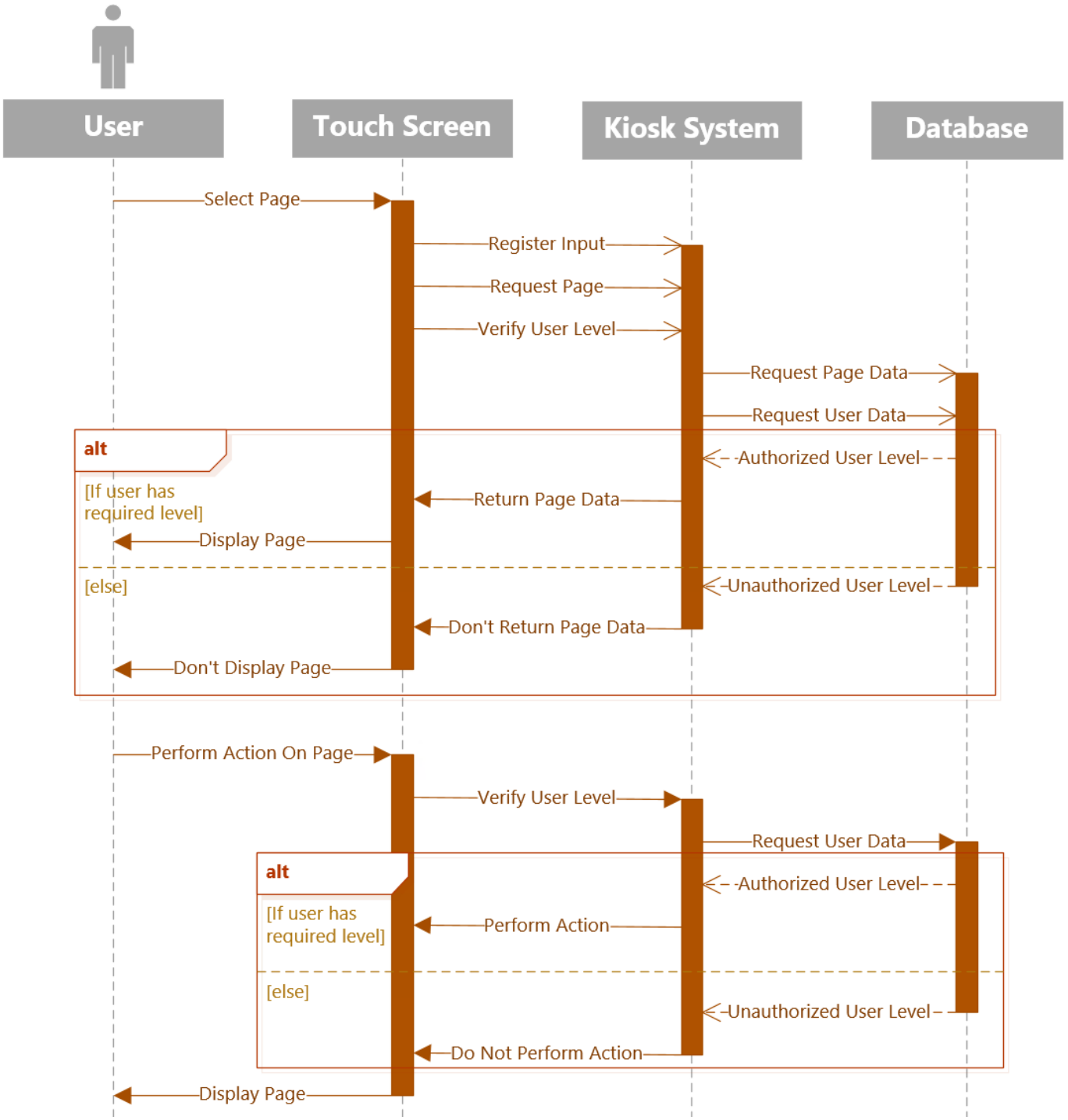

User
User Input
1
Interact with kiosk
Requested data
2
Search database
Requested data
D
Database
Page data
3
Generate pages
Kiosk
Page view
Page info
4
Display pages


User
User Input
1.1
Browse Pages
User Input
1.2
Select Page
Selected Page Data
1.3
Create Request Data
Page Data Request
1.4
Send Data Request
Requested Data
Requested Data
D
Page Data
Page ID
Page Layout Data
Page ID
2.3
Identify Correct Page
Page ID
3.1
Identify Page Layout
2.2
Process Data Request
2.1
Receive Data Request
Page Layout Data
3.2
Identify Page Data
Data to display
Info displayed on page
3.3
Generate Page
D
Text Data
D
Image Data
D
Layout Data
Kisok
Page View
4.2
Display Page View
Page View
4.1
Load Page View
Complete Page Data
Data to display

Appendix C


Audio
PK
Id
filepath
page_id
created_at
updated_at
Images
PK
Id
image_name
alt
page_id
created_at
updated_at
Stats
PK
Id
name
value
page_id
created_at
updated_at
Page
PK
id
heading
text
category_id
created_at
updated_at
Powerpoints
PK
Id
title
created_at
updated_at
Category
PK
Id
name
description
created_at
updated_at
PPT_Images
PK
Id
powerpoint_id
filepath
index
created_at
updated_at
Users
PK
Id
name
email
email_verified_at
password
remember_token
created_At
updated_at
Password_Resets
PK
email
token
created_at
Migrations
PK
Id
migration
batch
Survival_Highscores
PK
id
initials
score
Timed_Highscores
PK
id
initials
score
Finding_Highscores
PK
id
initials
score

Appendix D


User
Touch Screen
Kiosk System
Database
Select Page
Register Input
Request Page
Verify User Level
Request Page Data
Request User Data
alt
[If user has required level]
Authorized User Level
Return Page Data
Display Page
[else]
Unauthorized User Level
Don't Return Page Data
Don't Display Page
Perform Action On Page
Verify User Level
Request User Data
alt
[If user has required level]
Authorized User Level
Perform Action
[else]
Unauthorized User Level
Do Not Perform Action
Display Page

Appendix E

Test and Implementation

In order to successfully implement the system, there are several tests it must pass to ensure correct functionality. These tests must be completed before the team can finish rollout. Before anything else can be tested the system has to be able to interact with the database, so this is the initial feature that will be tested. The system will be installed on touch screen devices, so once it is on and displaying data from the database the team will test to confirm it responds to touch screen inputs. Using the touch screen inputs will also show how fast the system loads everything, which covers the next test. The team will also have the client use the system as well without instruction to see if it is simple enough for them to navigate, which is the test for user friendliness. The next test will be for the team to go in to the administrator mode and ensure it is functioning correctly.

Test: Features to be Tested/Test Scenarios/Pass/Fail

The first feature the team will test is ensuring that the kiosk system interacts with the database as intended. The passing condition of the test is for the system to fetch the correct data. If data is not fetched from the database, or it is fetched but not displayed, that will result in a failure. If the data is fetched and displayed but is displayed in the wrong location, or is not the correct data, that will also be considered a failure. To test this feature, data will be put in to the database and then displayed on the screen. The team will verify that each piece of data is displayed in the correct location, and if it is then the system will pass the test. Otherwise, it will be considered a failure. There will also be a simulated power outage to test if it retains data or not. Finally, the team will have the client set up a kiosk by themselves to test that it is easy to do and they will be able to do so in the future.

One of the most important features to be tested is verifying that the system works on touch screen devices. The conditions for this test are simple: if the system registers touch screen inputs and performs the correct action then it will pass, otherwise it will be a failure. The tests for this feature are that once the data from the previous test is displaying correctly, the team will simply navigate through the system using the touch screen. This feature was constantly tested throughout development so the test on-site will just be a formality.

A significant feature of the project is ensuring it has quick load times. To test this during the rollout the team will browse through, create/edit and delete multiple pages. If the team feels that pages or functions took too long to load then it will be deemed a failure. Similarly to touch screen compatibility, this was also continuously tested throughout the duration of the project. Thus, the team has already confirmed that this is not an issue and testing on-site will just be a formality.

It is important for the system to be user friendly and easy to navigate through. The test for this is to have people who were not in the team (and thus have no knowledge of the system) to use the system. If the users are able to use the system without needing the team to explain it to them, then it will be considered a success. If they are not able to, then it will be a failure.

There is also an administrator mode in the system which the team needs to test. To test this, the team will go in to the administrator mode and then attempted to perform the expected functions. The administrator functions are editing the content of a page, deleting pages, adding pages and changing the layout of pages (this has some overlap with the loading time tests). The test will be deemed successful if the team is able to log in to admin mode and perform all of the functions. If admin mode can not be logged in to, or it's logged in to successfully but can't perform the expected functions, then it will be deemed a failure.

Data retention during power outages is a crucial feature of the system. To test this, the team will simulate unexpected power outages and then check to see if data is retained when the system gains power again. If there is any loss or modification of data then the test will be a failure, otherwise it will be considered a success.

Another aspect involved in making sure the system is simple was creating it so that the client would be able to set up additional kiosks by themselves. This is important so that they would not have to call the team in to do it for them in the future. To test this, the team will give the client the user manual and have them set up a kiosk. If the client is able to understand the instructions and follow them to correctly set up a new kiosk then the test will be passed, if not then it will be failed and the guide will be simplified.

Criteria/Suspension Criteria/Responsibilities/Schedule/Risks and Contingencies

During the rollout phase of the project, there are several criteria which the system had to meet. The main criteria for the rollout phase is to ensure the Raspberry Pi's are working correctly. The team also needs to set up a vlan so the client can access the system but visitors using the system can not use it to access the internet. There was also criteria which the system can not meet that would result in the suspension of implementation until the issue causing it was resolved. One of these suspension criteria is the wifi network blocking the server which the system is hosted on. Further installation would need to be stopped in this case until the server was able to be accessed. A solution to this issue would be creating a new wifi network that did not block the server but also did not have access to the internet. The team will split up the responsibilities during installation. Joshua is in charge of configuring the wifi ensuring the server is able to be accessed. Josiah is tasked with installing the server on-site, giving somewhere to hold the system itself. Jayden is responsible for setting up the physical devices which the system will be used on. The team's schedule for rollout is for everyone to work on their tasks simultaneously. Due to having to travel for several hours by both car and boat in order to get to the client location there is a risk of equipment being damaged en route. The team will mitigate this by being cautious with the equipment, ensuring it is always stored in a safe location and was within sight of the members. There is also the risk that the wifi may not work with the server however the team already has a contingency plan in the event this occurred.

Implementation: Roll-Out Plan, Training, Backup, Support

The team will head out to Deep Cove again and install the system on all the devices currently on site. The client will also be provided with an installation guide so they

can install the system themselves on any future devices they get. An additional trip to Deep Cove is also possible several weeks later to ensure that everything is working correctly however this shouldn't be necessary.The system was made to be extremely intuitive and easy to use so a lot of training isn't necessary. The team will give the client a walkthrough of how to use "client specific" features (such as the administrator mode). Visitors to the center who will be the primary users of the system will not need any training for the features they have access to. The user manual will be thorough and clear enough that any issues the client may encounter are explained within the manual. The team will provide the client with backup copies of the system and database for use in the event something goes wrong with the initial ones.

KTM SOLUTIONS

KIOSK DEVELOPMENT PROJECT

DEEP COVE OUTDOOR EDUCATION TRUST
Educating and sharing our conservation values

ABOUT

Deep Cove Outdoor Education Trust is a non profit charity organisation which teaches visitors to the Deep Cove area about the environment and conservation.

THE PROJECT

They approached the Southern Institute of Technology (SIT) requesting an educational, interactive, touch screen kiosk system to be developed for them. This was to replace their existing educational materials including books and brochures which in the modern age can be considered an uninteresting way of learning to younger people.

FEATURES

- An administrative interface to modify displayed content
- A modified version of Raspbian to run the kiosk software on-site
- A server to hold and serve information on-site
- A ticket-based wireless system that allows guests to use the Internet at the hostel
- Touch screen devices that automatically reboot after a power outage
- Interactive games with high score system
- Video streaming server on-site
- Teaching/educational resources accessible from any device

SPECIAL THANKS

A huge thanks to those who helped make this project possible. These people include but are not limited to;

- Anita Murphy
- Mike MacManus
- Hayley Mitcheson
- Ken Sutton