Team Quaternary:

ST10019838 - Damian Dare

ST10019972 - Guillaume Swanevelder

ST10091991 - Christiaan Versfeld

ST10158660 - Ruan Zwarts

Module: XBCAD7319

Lecturer: Ms. Mmaphuti Matau

27 September 2024

[Word Count: 2357]

Documentation

(ADD LINKS)

Table of Contents

[1. INTRODUCTION TO THE PROJECT 3](#_Toc177572307)

[1.1. Client Organisation Background 3](#_Toc180059093)

[1.2. The Project’s Purpose 3](#_Toc180059094)

[1.3. Ethical and Privacy Concerns 4](#_Toc180059095)

[1.4. Work Agreement 11](#_Toc180059096)

[1.5. Definition of Ready (DoR) 13](#_Toc180059097)

[1.6. Definition of Done (DoD) 14](#_Toc180059098)

[1.7. Roadmap (High-level plan) 14](#_Toc180059099)

[1.8. Project Risks and Mitigation Strategies 15](#_Toc180059100)

[2. REQUIREMENTS 4](#_Toc177572312)

[2.1. User Roles 4](#_Toc177572313)

[2.2. User Stories 4](#_Toc177572314)

[2.3. User Experience Journey Map 4](#_Toc177572315)

[3. NON-FUNCTIONAL REQUIREMENTS 5](#_Toc177572316)

[4. ANALYSIS ARTIFACTS 5](#_Toc177572317)

[4.1. Domain Modelling 5](#_Toc177572318)

[4.2. Design Artifacts 5](#_Toc177572319)

[5. IMPLEMENTATION DOCUMENTATION 5](#_Toc177572320)

[6. DATA SCHEMA DOCUMENTATION 6](#_Toc177572321)

[7. ARCHITECTURE ARTIFACTS 6](#_Toc177572322)

[7.1. Design Patterns 6](#_Toc177572323)

[7.2. Architecture Patterns 6](#_Toc177572324)

[7.3. Cloud 6](#_Toc177572325)

[8. SECURITY 6](#_Toc177572326)

[9. DEVOPS 6](#_Toc177572327)

[9.1. GitHub Actions Pipeline 6](#_Toc177572328)

[10. RUNNING COSTS 6](#_Toc177572329)

[11. CHANGE MANAGEMENT 7](#_Toc177572330)

[12. APPENDICES 7](#_Toc177572331)

[REFERENCE LIST 8](#_Toc177572332)

# INTRODUCTION TO THE PROJECT

Welcome!

This document pertains to all the application documentation for Team Quaternary’s XBCAD7319 project. In this document, high-level in-depth discussions will be provided around our applications requirements, our analysis of the applications domain, our implementation of the application, our data schemas, our architecture decisions, our security considerations, our devops plan, the estimated running costs of the application, and change management that will be involved in the project.

## 1.1. Client Organisation Background

Our client’s organisation, Crystal Ridge At Providence Stud, is a competitive equestrian center that was established in 1993, and is currently based in Benoni. Their aim as an organization is to always strive for the best, both for themselves and their horses, aiming to always provide and improve upon a “family” atmosphere. Crystal Ridge additionally offers outstanding 5-star facilities that are used to host frequent shows, such as their Equestrian Excellence Series.

They offer several different services, all at a high quality, which include:

* Stabling and Livery which entails providing top-notch around the clock care for your horses.
* Riding Lessons which entail highly qualified instructors providing horse riding lessons.
* Courses and clinics which entails sharing their knowledge and passion through courses and clinics.

For more information visit: <https://www.crystalridgestables.co.za/>

## 1.2. The Project’s Purpose

At Crystal Ridge, one section of their business model involves offering horse-riding lessons. Their current system in place is a time consuming and tedious process that involves manually capturing, booking, scheduling and organizing these horse-riding lessons. The purpose of this project is to create and deploy a lesson management application, tailored to the needs of our client’s equestrian estate, that aims to simplify, speed up and ease their processes of capturing, booking, scheduling and organizing their horse-riding lessons.

## 1.3. Ethical and Privacy Concerns

When researching about ethical and privacy concerns, OpenAI (2024) was able to indicate some very important concerns that need to be addressed. For each of these concerns indicated by OpenAI (2024), a description will be provided by OpenAI (2024) to describe the concern, and we will state how we the concern will be addressed.

The ethical concerns indicated by OpenAI (2024) include:

* Data Security

This involves ensuring that user data is securely stored and protected from unauthorized access or breaches (OpenAI, 2024).

Solution: Our database provider, Supabase, provides authentication and verification functionalities that will be utilized to ensure that only certain verified users can access the data within the database. Additionally, we will keep the API key provided by Supabase outside of the code, stored in an environment variable closer to the hosting platform to mitigate unauthorized database access if the code is breached.

1. Transparency

This involves being transparent about how user data is collected, used, and shared within the application (OpenAI, 2024).

Solution: Firstly, on the login page, we will provide a hyperlink or access to terms and conditions page or document that outlines how the data will be collected, used, and shared. Secondly, the primary purpose of collecting personal information will be to ensure that a user is properly identified within the application. Therefore, minimal personal data will be collected as only the essentials information such as name, surname, email, and potentially phone number are necessary to properly identify the user within the application.

1. Informed Consent

This involves obtaining explicit consent from users before collecting their personal data and ensuring they understand how it will be used (OpenAI, 2024).

Solution: Firstly, the terms and conditions provided will explain how and why their personal data will be collected and used. Secondly, users will have to request their account to be created by an admin which can only be done after agreeing to the terms and conditions.

1. Data Ownership

This involves clarifying who owns the data collected by the application and how it can be accessed or deleted by users (OpenAI, 2024).

Solution: Firstly, the owner of the data and the processes of how to access the data and delete the data will be clarified in the terms and conditions. Secondly, all users will be managed by system admins, meaning that users will not be able to delete their account whenever they want to. Instead, they will have to request that it be deleted.

1. Bias and Discrimination

This involves avoiding biases in algorithms or decision-making processes that could lead to discrimination against certain groups of users (OpenAI, 2024).

Solution: No algorithms or decision-making processes will be needed within the application as the purpose of the application is improve the manual process of organizing horse-riding lessons, resulting in no disclination and no bias as there will be no measure in place that is capable of discrimination and bias.

1. User Empowerment

This involves providing users with control over their data, privacy settings, and the ability to opt-out of data collection if desired (OpenAI, 2024).

Solution: Firstly, all users will be managed by system admins, meaning that system admins will be the ones who control the user information within the system. Secondly, as users will not be interacting with one another over the application, therefore user data will private only to Crystal Ridge and the user themselves. Thirdly, the application will only require the essential user data to be able to identify who the user is, meaning that a user cannot opt out of data collection as data collection will only take place when creating an account and no-after.

1. User Safety

This involves ensuring that the application does not facilitate harmful or unethical behaviours, such as cyberbullying or harassment (OpenAI, 2024).

Solution: The application will not contain or provide any functionality in which users can interact with each other, preventing harmful and unethical behaviours amongst users.

1. Accessibility

This involves designing the application to be accessible to users with disabilities and ensuring inclusivity in its features and functionalities (OpenAI, 2024).

Solution: To ensure accessibility, screen reader support will be implemented for the visually impaired and website bandwidth and internet usage will be optimized for those with limited internet access.

1. Environmental Impact

This involves considering the environmental impact of the application, such as energy consumption or carbon footprint, and implementing measures to reduce it (OpenAI, 2024).

Solution: As our application and its services will be hosted in the cloud, the environmental aspects will be transferred to and managed by the cloud services.

The privacy concerns indicated by OpenAI (2024) include:

1. Data Collection

This involves limiting the collection of personal data to what is necessary for the functionality of the application and obtaining user consent for data processing (OpenAI, 2024).

Solution: The primary purpose of collecting personal information will be to ensure that a user is properly identified within the application. Therefore, minimal personal data will be collected as only the essentials information such as name, surname, email, and potentially phone number are necessary to properly identify the user within the application. The user will provide consent for the application using minimal personal information when creating an account.

1. Data Minimization

This involves minimizing the amount of personal data collected and stored to reduce the risk of data breaches or misuse (OpenAI, 2024).

Solution: The primary purpose of collecting personal information will be to ensure that a user is properly identified within the application. Therefore, minimal personal data will be collected as only the essentials information such as name, surname, email, and potentially phone number are necessary to properly identify the user within the application.

1. Data Protection

This involves implementing strong encryption and security measures to protect user data from unauthorized access or theft (OpenAI, 2024).

Solution: The cloud hosting services used automatically provides encryption methodologies that will ensure that data will be secured both inside and outside of transit.

1. Third-party Sharing

This involves clearly stating if and how user data is shared with third parties, such as advertisers or analytics providers, and obtaining user consent for such sharing (OpenAI, 2024).

Solution: Firstly, the data captured in the application will be useful and relevant only to the application and Crystal Ridge itself, with no intention of ever being sold or shared with third parties. Secondly, the data captured in the application will have no use for advertisers or analytics as the information pertains to horse-riding lessons and horse-riding lessons alone.

1. User Anonymity

This involves providing options for users to use the application anonymously or pseudonymously to protect their privacy (OpenAI, 2024).

Solution: Firstly, as this is more so of a service application, anonymous use may potentially cause some security concerns, hence there will be no option for anonymous use. Secondly, users won’t be interacting with other users on the application, meaning that not being anonymous won’t have as big of implications.

1. Data Retention

This involves establishing clear policies on how long user data will be retained and when it will be deleted (OpenAI, 2024).

Solution: Firstly, these policies will be determined purely by the client. Secondly, these policies will be discussed in the terms and conditions.

1. User Rights

This involves respecting user rights regarding data access, rectification, erasure, and portability as mandated by data protection regulations (OpenAI, 2024).

Solution: To ensure that the rights of users will be respected, data protection regulations such as POPI and GDPR will be used.

1. Geolocation Data

This involves handling geolocation data responsibly and ensuring that users have control over when and how their location information is shared (OpenAI, 2024).

Solution: The application will in no sense of the matter use geolocation data as it will not be necessary.

1. Children's Privacy

This involves complying with regulations such as COPPA (Children's Online Privacy Protection Act) when collecting data from children under 13 years old (OpenAI, 2024).

Solution: Firstly, the application doesn’t collect data from users, apart from the basic information such as name, surname, email, and potentially a phone number. Secondly, the application will mostly consist of adults as they will be the ones to arrange and pay for horse-riding lessons for themselves or their children.

## 1.4. Work Agreement

Our goal as a team is ensure that at the very least, the application meets our DoR (see 1.2. Definition of Ready (DoR)) by the end of the working period / semester. As the team is also occupied with other projects, the application may not meet our DoD (see 1.3. Definition of Done (DoD)) due to the time constraints. As a result, any work needed to get the application to meet our DoD will be completed after the working period / semester, with the same working conditions and being free of charge.

The following denotes the team members and their responsibilities:

* Damian Dare: “The Sheriff”

Damian is the sheriff of these parts, maintaining peace and order whilst ensuring the law is upheld. In layman’s terms, Damian’s responsibility pertains to that of the team leader and quality assurance, ensuring the team does what they are supposed to do, ensuring the completed work is of high quality, combining the work together into one cohesive unit, and assisting wherever help is needed.

* Guillaume Swanevelder: “The Third Leg”

Guillaume provides the team with an additional support mechanism, ensuring balance and stability, keeping the team afloat when the waters become unstable. In layman’s terms, Guillaume’s responsibility is to provide the team with an extra hand to assist in any task, enabling the team to complete tasks faster or to better juggle different tasks and projects.

* Christiaan Versfeld: “The Exorcist”

Christiaan expels all of the demons from our code, allowing it to work as intended. In layman’s terms, Christiaan’s responsibility pertains the backend of the application, designing, testing, and ensuring that all system functionality works correctly.

* Ruan Zwarts: “The Beauty Queen”

Ruan is the prettiest princess of them all, using his skills to make our application look as good as he does. In layman’s terms, Ruan’s responsibility pertains the frontend of the application, designing, testing, and ensuring that the applications user-interface is friendly, consistent, colorful, and responsive.

## 1.5. Definition of Ready (DoR)

Our DoR is when:

* The “basic” requirements have been implemented (see 2. REQUIREMENTS).
* The non-functional requirements have been implemented (see 3. NON-FUNCTIONAL REQUIREMENTS).
* The analysis artifacts have been implemented (see 4. ANALYSIS ARTIFACTS).
* The implementation documentation has been implemented (see 5. IMPLEMENTATION DOCUMENTATION).
* The data schema documentation has been implemented (see 6. DATA SCHEMA DOCUMENTATION).
* The architecture artifacts have been implemented (see 7. ARCHITECTURE ARTIFACTS).
* Security has been implemented (see 7. SECURITY).
* DevOps has been implemented (see 9. DEVOPS).

## 1.6. Definition of Done (DoD)

Our DoD is when the functionality of our “ready” application (see 1.2. Definition of Ready (DoR)) has been extended to include our “extra” requirements (see 2. REQUIREMENTS).

## 1.7. Roadmap (High-level plan)

The high-level plan for the team is as follows.

Basic documentation will be completed by the 27th of September to provide a foundation in which the application can be built. This documentation will then be used to build the project’s application that satisfies the Definition of Ready (see 1.1. Work Agreement and 1.2. Definition of Ready (DOR)), which will be completed by 25th of October. Following the completion of DoR application, the remaining documentation will be completed by the 22nd of November. The documentation will also possibly be expanded to meet the DoD (see 1.3. Definition of Done (DoD)) if there is enough time.

During these working periods, there will be at a minimum, one recorded meeting per week discussing what we did the previous week, what we plan to do for upcoming week, and what have been some of the troubles we have encountered. These recorded meetings will be used as artifacts to prove that we are working and communicating frequently. Additionally, although there may only be one recorded meeting per week, team members may meet privately to discuss certain matters. These “private” meetings won’t be recorded however a screenshot will be taken to prove that a meeting took place.

This roadmap / plan will be most optimal for the team as the meetings won’t consume that much time per week, while spreading out the work, allowing the team to complete the assigned work before its deadline.

## 1.8. Project Risks and Mitigation Strategies

The following risks to the project were indicated by OpenAI (2024) and pose the most likely threat of occurring.

* Time Constraints

As the team is busy working on several projects from other modules, there may not be enough time to produce a high-quality product that the customer is satisfied with.

Mitigation Strategy: Extensive planning and communication will be conducted to ensure that the team is able to balance the semester’s workload, enabling high-quality work to be produced.

* Third Party Vendors

The application being produced will not be self-hosted but will rather be hosted on third party services such as Vercel and Supabase, meaning that if any of the services were to shut down or to suffer from performance issues, the same would apply to the application. Additionally, it is possible that there may be some form of contractual issues that arise when using third party solutions, causing the client to be locked into a contract that they no longer want to be in or being kicked out of a contract that they want to be in.

Mitigation Strategy: In terms of the contractual side of things, the third-party services that will be used to host the application and its data will have a monthly payment option, allowing you to opt in or out on a per month basis. Additionally, a contract may only be required from these vendors if the application greatly exceeds expectations and requires a custom resource plan from the vendors, which may not be necessary as the vendors’ “pro” tiers may be sufficient for this application. In terms of the services shutting down, the data will be regularly backed up and the code will be in an easily reachable place, allowing for easy migration to occur if the need arises.

* Security

As the team consists of students, we may not have enough knowledge, expertise and experience to be able to fully secure the application we are building.

Mitigation Strategy: We will use external trusted tools to test and secure the application to the best of our ability while conducting extensive research into how we will be able to secure the code. Additionally, we will thoroughly suggest to our client to find someone that can test the applications security and provide solutions that will better secure the application.

* Regulatory Compliance

As the team consists of students, studying IT and not Law, we may not have enough knowledge, expertise and experience to be able to fully ensure that the application complies with legal requirements, industry standards, and data protection methods.

Mitigation Strategy: We will conduct extensive research into how we will be able to comply with regulatory standards. Additionally, we will thoroughly suggest to our client to find someone that will ensure that the application complies with regulatory standards.

# REQUIREMENTS

## User Roles

* Client / Consumer

The client / consumer is a user whose sole intention is to consume / use our applications services. For example, a customer at a grocery store doesn’t work at said grocery store, but rather purchases goods that the store has to offer. A client / consumer has limited functionality within the system compared to the other user roles. Majority of the systems users will be clients.

* Coach

A coach is a user that organizes and conducts lessons for the clients and assigns levels / ranks to clients to indicate their skill level. This user role is similar to that of a coach in a sports team. A coach has access to both its own functionality as well as the client’s functionality.

* Administrator / Admin

An administrator is a user that will have full access to the application to ensure that everything is running properly. This user role will have access to both of the coach’s and client’s functionality with additional unique features that only administrators can access.

## User Stories

Unauthorized access is prevented in the system.

* Basic Functionality

As an admin, I want to create all the client accounts for the system so that:

* Unidentified / unknown users can’t create accounts and access the system, adding an additional layer of security.
* A consultation meeting can be held before a user account is created, which will assist in determining if a user is a potential security threat (assisting in security), and determining what the user’s skill level is, so that they can then attend lessons that are on par with their skill level.

As a client, I want to be able to have an account / profile so that:

* I can be identified in the system.
* Coaches and administrators can recognize me.
* I can be assigned a skill level (that will be stored in the profile) that determines which lessons I can attend.
* As a coach, I want to assign skill levels to clients so that they can only attend lessons of their skill level.
* As an admin, I want to be able to create accounts for coaches, with the coach role, so that only valid / trusted / official coaches can access the system.
* As an admin, I want my account / profile to be created by a system developer to ensure that only specific users have administrator access and to ensure that no random user can become an administrator.
* As an admin and a coach, I want to be able to see a list of all users within the application so that I can see who is using the application and how many people are using the application, to better identify potential threats and gather data for potential costs.

For lessons:

* As a client, I want to book a slot for a lesson so that I can attend the in-person lesson.
* As a client, I want to upload proof of payment for a lesson so that I can prove that I paid for a lesson that I can then attend.
* As an admin, I want to receive an email or notification that a client has uploaded proof of payment for a lesson so that I can confirm if the payment occurred so that I can then admit the client to the lesson.
* As a client, I want to select which horse I will be using, whether it is my own or a training horse, for a lesson so that I can indicate to the coach what horse I will be using.
* As a coach, I want to organize lessons so that clients can attend them.
* As a coach, I want to limit how many people can sign up for a lesson so that I can control how many people I will have to teach.
* As an admin, I want to add training horses to the system so that clients can select which horse to use during a lesson (if they don’t have their own), and so that the system can properly track which training horses are available for a lesson.
* As an admin, I want to add addresses to the application so that it can be easier to select a re-occurring address than to manually type in an address when creating a lesson.
* As an admin, I want to add contact details to the application so that it can be easier to select re-occurring contact details than to manually type in contact details when creating a lesson.
* As an admin, I want to add payment details to the application so that it can be easier to select re-occurring payment details than to manually type in payment details when creating a lesson.
* Extra Functionality
* As an admin, if a lesson wasn’t paid for, I could send the client a message or notification so that I can notify them that the lesson wasn’t paid for.
* As a client, I want to add a note for the coach of a lesson so that I can describe or inform the coach of any particular matter.

## User Experience Journey Map

For more information regarding the project requirements please view the project specific file called TeamQuaternary\_XBCAD7319\_Requirements.docx document file in the Requirements folder that came with this file.

# NON-FUNCTIONAL REQUIREMENTS

According to AltexSoft (2023), non-functional requirements are used to describe how an application works that primarily focuses on user expectations. Furthermore, they aim to define product properties and are desirable to implement, but are not mandatory (AltexSoft, 2023).

* Performance

AltexSoft (2023) defines performance as the speed in which a software system or its components can respond to certain users’ actions under a given workload. OpenAI (2024) and AltexSoft (2023) have assisted in determining some of the following requirements.

Requirements:

* All pages and all functionality within the application must have a maximum response time of 7 seconds across all browsers for up to 5000 concurrent users, across any type of internet connection (AltexSoft, 2023).
* The application must be able to handle at least 100 concurrent transactions per second (OpenAI, 2024).
* The application should automatically re-fetch data from the database to refresh any stale cached data at least once every 5 minutes to ensure that the user has the most recent up to date information. This functionality should be changed to refresh stale data at least once every 2 minutes to accommodate high concurrent users and transactions.
* The application should time out a user if they have been inactive for more than 15 minutes to reduce unnecessary bandwidth consumption.
* During peak load, the application should not use more than 70% of its CPU and RAM (OpenAI, 2024).
* The system should offer seamless and responsive user interactions, with short loading times and delays (OpenAI, 2024).
* Scalability

According to AltexSoft (2023), scalability is the system’s ability to handle the growth and expansion of both data volume and user load. OpenAI (2024) and AltexSoft (2023) have assisted in determining some of the following requirements.

Requirements:

* The application must scale to support 5000 concurrent users while retaining optimal performance (AltexSoft, 2023).
* The program must scale to handle a tenfold increase in transaction volumes and data storage over a 2-year period (AltexSoft, 2023).
* To optimise performance and cost effectiveness, the application should dynamically scale resources up and down in response to demand (OpenAI, 2024).
* As the user base expands, the application should facilitate horizontal scaling, enabling the installation of new nodes to share the load and enhance capacity (OpenAI, 2024).
* The application should be able to autonomously scale resources depending on predetermined thresholds or metrics (OpenAI, 2024).
* The application should implement session management mechanisms that enable load balancing and failover while preserving user sessions (OpenAI, 2024).
* The application should utilise its cloud scalability features such as auto-scaling and on-demand resources (OpenAI, 2024).
* Reliability

AltexSoft (2023) defines reliability as how likely the system or its components would run without a failure for a particular period of time under predefined conditions. OpenAI (2024) and AltexSoft (2023) have assisted in determining some of the following requirements.

Requirements:

* During a month, the application must function flawlessly in 95 percent of all use scenarios (AltexSoft, 2023).
* All functionalities should be processed with 100% accuracy, and the system must always ensure data integrity (AltexSoft, 2023).
* The application must be capable of handling and recovering from failures without causing data loss or inaccurate data processing (AltexSoft, 2023).
* The critical components of the application must have redundant backups that will enable continuous operation if failures were to occur (OpenAI, 2024).
* Data and configuration backups should be performed on a regular basis, and a solid recovery strategy should be in place to restore the system if data loss occurs (OpenAI, 2024).
* Even if a component fails, the application should continue to work and offer vital functions (OpenAI, 2024).
* The application’s Mean Time to Recovery (MTTR) for restoring service after a failure should be less than 1 hour (OpenAI, 2024).
* Before a critical failure happens, the application’s Mean Time Between Failures (MTBF) should be at least 10,000 hours (OpenAI, 2024).
* Maintainability

AltexSoft (2023) defines maintainability as the time needed for a solution or its components to be fixed or updated. OpenAI (2024) and AltexSoft (2023) have assisted in determining some of the following requirements.

Requirements:

* After a system failure, the mean time to restore the system (MTTRS), including delay time and corrective repair time, cannot be longer than 1 hour (AltexSoft, 2023).
* The application should be modular, consisting of clear and well-defined components that are easily modified or replaced without interfering other system components (OpenAI, 2024).
* To assist in understanding and maintaining the system, comprehensive documentation should be created, providing thorough code comments, system architecture diagrams, and user manuals (OpenAI, 2024).
* To guarantee consistency and readability of the codebase and to make it easier to maintain, coding standards should be adhered to, and best practices should be enforced (OpenAI, 2024).
* Version control systems must be utilized to track changes, collaborate on code, and roll back to earlier versions if the need arises (OpenAI, 2024).
* Code complexity should be limited to ensure better readability, understandability, debugging, and modification (OpenAI, 2024).
* Automated tests should me implemented to facilitate regression testing and to prevent the introduction of new bugs or issues when any code changes (OpenAI, 2024).
* Dependencies should be managed to avoid version conflicts whilst providing a simple manner to update or integrate external libraries and frameworks (OpenAI, 2024).
* Security

AltexSoft (2023) states that the non-functional requirements of security are used to establish specific threats that will be addressed and elaborated on by the functional requirements of security. AltexSoft (2018), AltexSoft (2023), and AltexSoft (2024) have assisted in determining some of the following requirements.

Requirements:

* The application must include authentication, authorization, and role-based access control (RBAC) to prevent unauthorized access to the application and its functionality.
* The application must incorporate encryption to protect its at rest and in transit data. Additionally, encryption standards such as ISO must be followed (AltexSoft, 2023).
* The application should implement audit logging to monitor what is happening within the application (AltexSoft, 2018).
* The application must implement session management and an active timeout session controller to better reduce and secure the available number of active sessions that attackers can potentially take control of.
* Vulnerability management principles and tools must be incorporated to assess the vulnerabilities that might occur in the application to ensure that the system is fault proof and is regularly kept up to date from new attacks (AltexSoft, 2024).
* The system must include input validation and sanitization to ensure that all data input into the application is valid, authentic, and reliable.
* Advanced defensive programming and error handling procedures should be implemented to avoid any erroneous data from entering the application.
* Usability

AltexSoft (2023) simplifies the definition of usability to being all about user-friendliness, ensuring the applications user interface is intuitive and easy to navigate, with the applications features being easy to find and understandable whilst meeting the user’s needs. OpenAI (2024) and AltexSoft (2023) have assisted in determining some of the following requirements.

Requirements:

* According to AltexSoft (2023) and OpenAI (2024), the application must have design that is consistent, pleasant to use, easy to learn and intuitive.
* The application must provide a simple and easy way to quickly reach their desired features of the application (AltexSoft, 2023).
* The application must provide clear and consistent instructions at all times to prevent users from making mistakes within the application (AltexSoft, 2023).
* The application must provide clear and consistent instructions and error handling when users make a mistake within the application, to ensure that they know what to do within the application (AltexSoft, 2023).
* Usability testing should be conducted to assess user experience, quantify usability, and ensure that the application satisfies the desired usability requirements (AltexSoft, 2023).
* The application should enable complete activities quickly and with the fewest possible steps or interactions (OpenAI, 2024).
* The application should provide regular feedback to the user for their actions in the form of alerts, progress indicators, and success or error messages (OpenAI, 2024).
* The application should provide strong search capabilities to enable users to quickly locate relevant information within the system (OpenAI, 2024).
* The application’s design must be responsive and accommodated for all devices, allowing all users on any device to use the application (OpenAI, 2024).
* The application must be accessible to users with disabilities and should adhere to accessibility guidelines (OpenAI, 2024).
* Interoperability

AWS (n.d.) defines interoperability as the ability of applications and systems to be able to securely and automatically exchange data irrespective of geographical, political, or organizational boundaries. Also, according to AltexSoft (2023), compatibility additionally contributes to the definition of how interoperable the system is. AltexSoft (2023) has assisted in determining some of the following requirements.

Requirements:

* There must be an API that will enable communication between the application and its database as they may be hosted on different platforms.
* The application does not have to expose the API to the public as it will not be necessary for any other application or software to consume the applications data.
* As the application will be developed for the web, it must be compatible with all internet browsers, and it must have a responsive design so that any device can use the application (AltexSoft, 2023).
* Internationalisation / Localisation

Wilcock and O’Brien (2024) define internationalisation as designing and developing an application in such a manner that it can support users in different global markets. Moreover, localisation is defined by Wilcock and O'Brien (2024) as modifying or translating an application's localisable resources to satisfy the linguistic, cultural, and political demands of the local markets the app is meant to serve.

Requirements:

* The application does not have to support internationalisation as the application will be developed and used only within the South African context.
* The application must provide support for the South African locale as it will be primary place of usage.

For more information regarding the project non-functional requirements please view the project specific file called “TeamQuaternary\_XBCAD7319\_Non-Functional-Requirements.docx” document file in the “Non Functional Requirements” folder that came with this file.

# ANALYSIS ARTIFACTS

To evaluate these requirements and develop a domain model utilising a Domain-Driven Design (DDD) methodology, it is essential to identify the many constrained contexts within the system.

## Domain Modelling

## Description of Each Bounded Context

1. User Management Context

* Manages user authentication (login/logout) and access control.
* Supports CRUD operations for user profiles.
* Administers several roles (Client, Coach, Admin) and their associated rights.

1. Lesson Booking Context

* Manages class scheduling, encompassing the selection of dates and time periods.
* Imposes regulations regarding lesson accessibility according to user proficiency and coach allocations.
* Oversees booking particulars and interacts with the Lesson Management framework to obtain lesson data.
* Guarantees that booking confirmations and updates are conveyed to users through the Notification context.

1. Payment Management Context

* Displays payment information and allows for payment processing.
* Integrates with external payment gateways to manage electronic funds transfers and alternative payment methods.
* Monitors payment status and preserves records of payment history for each user.

1. Lesson Management Context

* Enables Coaches and Administrators to create, amend, and delete lessons.
* Administers lesson metadata, including descriptions, locations, schedules, and coach details.
* Facilitates the assignment of levels/ranks to clients for skill-appropriate lesson planning.

1. Notification Context

* Manages the sending of notifications for various events, such as lesson bookings, payment confirmations, or lesson cancellations.

Provides customizable notification templates for different types of alerts.

## Design Artifacts

Analysis of the Domain Model

The domain model is constructed according to the requirements, encompassing essential components such as user roles, lesson bookings, lesson specifics, and payment administration. We will convert these into implementation models comprising classes, relationships, and interactions to satisfy the specified user stories.

Implementation Model Design

* User Roles and Permissions: Define user access levels for Client, Coach, and Admin.
* Lesson Management: Organize how lessons are created, scheduled, and booked.
* Payment Processing: Implement payment processing and proof-of-payment uploads.

User Interface Components: Design UI interactions for booking lessons and viewing details.

For more information regarding the project Analysis Artifacts please view the project specific file called “TeamQuaternary\_XBCAD7319\_Analysis-Artifacts.docx” document file in the “Analysis Artifacts” folder that came with this file.

# IMPLEMENTATION DOCUMENTATION

## **Introduction**

This Implementation Documentation aims to deliver a comprehensive overview of the system's technical execution, encompassing essential components and processes to support future maintenance, scalability, and feature improvements. This documentation provides a comprehensive guide for developers, administrators, and stakeholders requiring insight into the system's structure, deployment, and operational functionality across diverse contexts.

This documentation encompasses the system's architecture, critical process sequence flows, deployment protocols, class/object relationships, and essential state transitions inside the system. It encompasses detailed UML diagrams to illustrate and elucidate the structure and relationships among components, with the objective of producing a clear blueprint of the system's architecture and operating logic.

## System Architecture Overview

This section offers a comprehensive overview of the system's architecture, outlining the fundamental components, their functions, and relationships. The system architecture delineates the principal modules, encompassing the user interface, backend services, database, and third-party integrations.

* **Frontend**: Built using modern web frameworks, providing a responsive interface for users to interact with the system.
* **Backend**: A RESTful API that manages business logic, data processing, and interactions with the database.
* **Database**: A relational database hosted on a cloud platform, securely storing data such as user information, lesson details, and transaction records.
* **Third-Party Services**: Includes authentication and cloud storage services that support secure data handling and seamless scalability.

## **UML Sequence Diagrams for Critical Flows**

Sequence diagrams depict the sequence of operations and interactions among system components for essential workflows. The subsequent illustrations illustrate the progression of activities for fundamental processes, providing a lucid comprehension of each component's function within the sequence.

### **Lesson Booking Process**

**Actors**: User, Lesson Management System, Database

**Flow**:

1. The user selects a date and views available time slots.
2. Upon choosing a slot, the system verifies availability.
3. If available, the user confirms the booking, which is then saved in the database.

### Lesson Management by Coach/Admin

**Actors**: Coach/Admin, Lesson Management System, Database

**Flow**:

1. The Coach/Admin accesses the lesson scheduling interface.
2. They create, edit, or delete lessons, which update records in the database.
3. Confirmation messages or error alerts are shown based on the operation's success.

# Deployment Documentation

Deployment documentation delineates the procedures necessary for the installation, configuration, and operation of the system across several environments, including development, staging, and production. It encompasses server prerequisites, program dependencies, and environmental parameters essential for successful deployment.

**Environments**:

* **Development**: Local environment setup for development and testing.
* **Staging**: A staging environment that replicates production, utilised for final testing.
* **Production**: The live environment hosted on cloud platforms like Vercel and Supabase.

**Steps for Deployment:**

1. Setup Hosting: Establish hosting on Vercel for the frontend and Supabase for backend services.
2. Environment Configuration: Define environment variables, API keys, and database URLs.
3. Build and Deploy: Execute build commands and deploy artifacts to staging or production servers.
4. Testing: Conduct end-to-end testing in staging before promoting to production.
5. Monitoring and Scaling: Enable monitoring tools to oversee system health and performance metrics.

**Security Considerations:**

* Authentication is managed through role-based access control.
* Sensitive data like API keys are securely stored in environment variables.

Regular backups are configured for the database to prevent data loss.

# Related Documentation

This section contains supplementary technical documents pertinent to the system's design and functionality.

**Database Schema**:

* **Description**: An ER (Entity-Relationship) diagram shows the relationships between key entities, such as User, Lesson, Horse, and Payment.
* **Data Model**: The data model is normalized for performance and includes foreign keys for relationship mapping.
* **Diagram**: An ER diagram illustrates the schema, detailing entity relationships and attributes.

**API Documentation**:

* **Endpoints**: Detailed documentation of API endpoints for key functions, such as:
  + Lesson scheduling and booking management.
  + Payment processing and confirmation.
* **Parameters and Responses**: Description of request parameters and response structures.
* **Security**: All endpoints require authentication tokens, ensuring only authorized users access sensitive data.

**Error Handling and Logging**:

* **Error Codes**: Common error codes and their meanings are documented for easier troubleshooting.
* **Logging**: Logging strategy includes tracking important events and errors, with logs stored in a centralized system for monitoring and debugging

# UML Diagrams for Key Components

The subsequent UML diagrams offer an extensive overview of essential system components and their interactions:

## UML Object Diagram

* + **Purpose:** Visualizes the relationships between classes and entities, such as User, Lesson, Horse, and Payment.
  + **Diagram:** An object diagram shows the structure and relationships, including composition, aggregation, and association.

For more information regarding the project implementation document please view the project specific file called “TeamQuaternary\_XBCAD7319\_Implementation-Documentation.docx” document file in the “Implementation Document” folder that came with this file.

# DATA SCHEMA DOCUMENTATION

For more information regarding the project schema document please view the project specific file called “TeamQuaternary\_XBCAD7319\_Data-Schema-Documentation.docx” document file in the “Data Schema Documentation” folder that came with this file.

# ARCHITECTURE ARTIFACTS

## Design Patterns

As this project’s application will be constructed with Next.js (a framework built on top of React), the design pattern choices for this application all align with the React ecosystem. Aguilar (2024) not only assisted in identifying some design patterns for our application, but also provided enough valuable insight into each of the design patterns to aid in our choices.

The following design patterns, suggested by Aguilar (2024), are most probable to be used in this project’s application.

* Custom Hook Pattern

This pattern involves encapsulating the logic of a component inside a reusable function that can then be shared between different components (Aguilar, 2024). These custom hooks are essentially JavaScript functions that use the Hooks provided by React (Aguilar, 2024).

Advantages of this pattern, according to Aguilar (2024), include:

* Code reusabilityas common logic will be encapsulated in separate functions.
* Code composition and readability as logic will be separated from the component.
* Improved testability as more specific and focused unit tests can be conducted on the logic contained in the Custom Hook.

This pattern was chosen due to the following:

* The reusability of Custom Hooks may drastically reduce of overall amount of code, allowing for easier code maintenance and code optimization.
* The enhanced readability provided by Custom Hooks will be beneficial as not only will it also allow for easier code maintenance and code optimization, but it will also assist future developers who will continue to work on and maintain the application (as the current team may not maintain the system).
* The improved testability can ensure that the application meets a high standard. Additionally, some of these tests can then be automated to save the development team time.

Custom Hooks may not be used for every piece of logic; however, they still will be utilized in the application.

* Extensible Styles Pattern

This pattern involves creating React components with flexible and easily customizable styles that can be modified and extended to meet the user’s needs (Aguilar, 2024).

Advantages of this pattern, according to Aguilar (2024), include:

* Code customization and extension without modification of source code.
* Visual consistency while providing flexibility.
* Simplified maintenance as the styling logic is separated from the component code.

This pattern was chosen due to the following:

* The customization and extension of components will enable major reuse of code, which in turn, contributes to easier code maintenance and code optimization. Additionally, the customization and extension will enable components to fit in / be used anywhere without restrictions.
* This pattern synergizes well with some of the other chosen patterns, namely the “Control Props Pattern” and the “Props Getters Pattern”, to form fully customizable and generic components that can be used in any situation.
* This pattern also enables users to customize how their application looks, contributing to the overall user experience of the application.
* Later down the road, the application may be converted or used to form a general lesson management application, and thus the extensibility and easy customization from the components will be necessary.

Extensible Styles may not be utilized for every component, as we aren’t making a component library, however, they still will be used in the application.

* Compound Components Pattern

This pattern involves creating components that work closely and coherently together to enable seamless communication and coordinated interactions among them (Aguilar, 2024). In this pattern, a parent component usually encapsulates multiple child components (Aguilar, 2024).

Advantages of this pattern, according to Aguilar (2024), include:

* Encapsulated and reusable related logic.
* Compound components will have a clear and consistent API for interacting with.
* Greater flexibility and customization as multiple components are essentially combined into one.

This pattern was chosen due to the following:

* Many components may be needed to form one singular feature of the application, and as a result, the Compound Components Pattern will provide cohesion between the components, enabling greater performance.
* This pattern will also make it easier to work with a large variety of components, contributing to a better developer experience. Additionally, this ease of use can also benefit any future developers when they will continue to work on and maintain the application (as the current team may not maintain the system).

Compound Components may not be utilized that often within the application as it is uncertain as to how many components will need to work closely and coherently with one another. However, when the need arises during development, this pattern is on the foreground to be utilized within the application.

* Control Props Pattern

This pattern allows for parent components to control the internal state of its child components using props (Aguilar, 2024). Essentially, a component delegates its control of its internal state to its parent, allowing the parent to manipulate and control the state of the child component (Aguilar, 2024).

Advantages of this pattern, according to Aguilar (2024), include:

* Greater control over a components state from higher-level components.
* Clear and bidirectional communication between components.
* Reuse of components as they can be used in different contexts.

This pattern was chosen due to the following:

* This pattern synergizes well with some of the other chosen patterns, namely the “Extensible Styles Pattern” and the “Props Getters Pattern”, to form fully customizable and generic components that can be used in any situation.
* The customization and extension of components will enable major reuse of code, which in turn, contributes to easier code maintenance and code optimization. Additionally, the customization and extension will enable components to fit in / be used anywhere without restrictions.
* This pattern will allow in app forms to be more readable and concise as the form can be broken up into smaller, more manageable components.
* By allowing the parent to control a components state, the logic will be more centralized allowing for easier control and maintainability.

The Control Props Pattern may not be used that much in the application as it is uncertain as to how many components will need to relinquish its control of its state to its parent. However, this pattern will be used wherever possible to ensure that the state and the application logic are more centralized.

* Props Getters Pattern

This pattern enables child components to get and modify specific props from their parents through special functions known as “props” getters (Aguilar, 2024). This pattern works by passing functions as arguments to child components (Aguilar, 2024).

Advantages of this pattern, according to Aguilar (2024), include:

* Clear and controlled mechanisms that child components can use to access and modify specific props of the parent.
* Clear and predictable communication between components which in turn, facilitates better debugging and code maintenance.
* Flexibility as the child component can adapt its behaviour based on the parent’s props.

This pattern was chosen due to the following:

* This pattern synergizes well with some of the other chosen patterns, namely the “Extensible Styles Pattern” and the “Control Props Pattern”, to form fully customizable and generic components that can be used in any situation.
* This pattern will allow in app forms to be more readable and concise as the form can be broken up into smaller, more manageable components.

The Props Getters Pattern may not be used that much in the application as it is uncertain as to how many components will need to access and manipulate its parent’s props. However, this pattern will be used wherever possible to ensure predictable communication between components.

## Architecture Patterns

The following architecture patterns were identified by Satyabrata\_jena (2024).

* Client-Server Architecture

As the name suggests, this pattern comprises of two major entities being the client and the server (Satyabrata\_jena, 2024). The way in which this architecture works is that a client requests resources that the server contains (such as data, files or services), which the server then processes the request and responds back accordingly (Satyabrata\_jena, 2024).

Some advantages of this architecture, according to Satyabrata\_jena (2024), include:

1. Centralized Management and Maintenance
2. Scalability
3. Security

Due to a web-based application being constructed for this project, the client-server architecture will automatically be implemented as several cloud servers and services will be utilized to host the application and its data. Furthermore, most of the internet is based on this architecture, as better enables several user devices (clients) to communicate with the servers simultaneously to get the data they require (Cloudflare, n.d.). Additionally, Next.js (a React Framework that will be used for this web-application) provides various controls and functionality that will make it easier to configure and optimize both the client side and the server side of the application.

* Event-Driven Architecture

This pattern is an agile approach in which events trigger services (operations) of the software (Satyabrata\_jena, 2024). In this pattern, whenever a user takes action in an application, a state change occurs, generating a reaction, also known as an event (Satyabrata\_jena, 2024).

Some advantages of this architecture, according to Satyabrata\_jena (2024), include:

1. Scalability
2. Real-time Processing
3. Flexibility

As several cloud resources will be used to host the application and its data, event-driven architecture will be useful to reduce cloud costs as bandwidth and CPU utilization are only used when events occur, preventing payment for continuous polling to check for an event (AWS, n.d.). Additionally, the application will be hosted on a different platform than its data and its API, meaning that the application will be decoupled into different components. According to AWS (n.d.), this event-driven architecture is most commonly found in applications with decoupled components as the architecture aims to improve agility and moving quickly (AWS, n.d.). Next.js also revolves around state as its changes are used to update the application, which pairs well with event-driven architecture as an event is essentially any change or update in state (AWS, n.d.).

## Cloud

Vercel will be used to host our application and Supabase will be used to host our api and database.

For more information regarding the project Architecture artifacts please view the project specific file called “TeamQuaternary\_XBCAD7319\_Arhictecture\_Artifacts.docx” document file in the “Architecture Artifacts” folder that came with this file.

# SECURITY

For our web application's design and development process, we placed a high priority on security. We aim to follow the best practices all through the development process to ensure the application is resistant to threats and vulnerabilities. The following describes our approach to addressing the main security issues:

* Authentication and Authorization

We will use JWT (JSON Web Tokens) for session management and OAuth 2.0 for safe authentication implementation. Additionally, Bcrypt will be used to hash passwords, ensuring that they stay encrypted even in the case of a data breach. Role-based access control (RBAC) will be implemented for authorisation to guarantee that users could only access authorised areas of the system (Sahin, 2020).

* Data Encryption

SSL/TLS encryption will be used to secure data transmission between the client and server, guarding against eavesdropping and man-in-the-middle attacks (Zhang, 2021). Moreover, AES-256, a well-known strong encryption standard, will be used to encrypt critical data kept in our database (Zhang, 2021). This guarantees the protection of all important data while it's in transit and at rest (Schneier, 2015).

* Input Validation and Sanitization

Libraries like express-validator, zod, and react-hook-form will be used to create stringent input validation and sanitisation to defend the application against SQL Injection and Cross-site Scripting (XSS) threats. This lessens the possibility of attacks that take advantage of input vulnerabilities by ensuring that the system processes only legitimate, sanitised data (StackHawk, 2023).

* Cross-Site Request Forgery (CSRF) Protection

We will incorporate a token-based protection system that necessitates the inclusion of a distinct token linked to the user's session to avoid cross-site request forgery (CSRF) attacks. As a result, unauthorised requests cannot be carried out on behalf of users who have been granted authentication (Ferguson, Schneier, & Kohno, 2015).

* Security Headers

To protect the application from frequent online vulnerabilities, such as XSS and clickjacking assaults, we will implement security headers such as Content Security Policy (CSP), X-Frame-Options, X-XSS-Protection, and Strict-Transport-Security (HSTS) (Williams, 2018). These headers lessen the chance of code injection and help specify which resources can be loaded (Williams, 2018).

* Network Security

A Virtual Private Cloud (VPC) architecture, which separates the application from other services and provides an extra layer of network security, will be used to host the application on a secure cloud platform like Vercel. To reduce the attack surface and stop unwanted network access, firewall rules will be used to limit access to internal resources (Saleem, Farouk Zinou, Mohammad, Ouni, Elhendi & Almuhtadi, 2024).

* Vulnerability Scanning and Dependency Management

To find and fix any known vulnerabilities in third-party libraries, we will use Snyk and npm audit to perform continuous monitoring and scanning of Node.js dependencies (StackHawk, 2023). This proactive strategy will guarantee that vulnerabilities are found and fixed quickly (StackHawk, 2023).

By following these security best practices, we ensure that our web application is resilient against known vulnerabilities and that user data is protected at every layer of the system.

For more information regarding the project security please view the project specific file called “TeamQuaternary\_XBCAD7319\_Security.docx” document file in the “Security” folder that came with this file.

# DEVOPS

## GitHub Actions Pipeline

For more information regarding the project Devops please view the project specific file called “TeamQuaternary\_XBCAD7319\_DEVOPS.docx” document file in the “Devops” folder that came with this file.

# RUNNING COSTS

# INTRODUCTION

Efficient cost management is essential for the deployment and growth of any program. For this project, forecasting user growth, comprehending scaling necessities, and assessing cost implications are essential to maintaining system sustainability as user demands escalate.

This document provides a comprehensive analysis of:

1. Predicted user growth over two years under various scenarios.
2. Scaling thresholds for technologies used in the architecture, with associated cost implications.
3. Predictive models of running costs in R for best-case, worst-case, and mean growth scenarios.
4. Recommendations for alternative technologies to mitigate costs when system limits are reached.

**PREDICTED USER GROWTH**

Predictive growth analysis is crucial for comprehending the evolution of user uptake and system demand over time. Modelling user growth across various scenarios — best-case, worst-case, and mean-case — enables the anticipation of resource needs and expenses related to application scalability. This data informs budgeting and guarantees the application architecture can respond to fluctuating user activity levels.

This analysis not only informs budgeting but also ensures that the application architecture can adapt to varying levels of user activity. In this document, user growth is projected over a two-year period, with monthly increments, enabling detailed predictions of resource utilisation and cost implications under each scenario. Such predictive modelling provides a foundation for proactive planning, ensuring the application remains responsive, scalable, and cost-efficient as its user base expands.

**Assumptions:**

1. The application starts with **65 users**, including Admins, Coaches, and Clients.
2. Growth rates vary based on adoption:

* **Best Case**: 10% monthly growth.
* **Worst Case**: Minimal growth (2–3 new users monthly).
* **Mean Case**: 5% monthly growth.

1. Growth factors include marketing efforts, word-of-mouth, and seasonal activity.

|  |  |  |  |
| --- | --- | --- | --- |
| **Month** | **Best Case (Users)** | **Worst Case (Users)** | **Mean Case (Users)** |
| Month 1 | 65 | 65 | 65 |
| Month 6 | 106 | 85 | 95 |
| Month 12 | 226 | 100 | 163 |
| Month 18 | 490 | 115 | 305 |
| Month 24 | 786 | 130 | 460 |

# SCALING POINTS OF EACH TECHNOLOGY

As user adoption grows, the application’s architecture must accommodate increasing demand, particularly with the integration of Clerk.com for user management. Identifying the scaling points of each component — frontend, backend, database, user management, and third-party services — is critical for ensuring smooth performance and cost-effective resource utilisation. Each technology in the stack has specific thresholds, beyond which additional capacity or alternative solutions may be required. This section analyses these thresholds, linking them to projected user growth and usage patterns, ensuring optimal performance while minimising disruptions and controlling costs. By proactively planning for scalability, the application can remain responsive as demand increases.  
  
**Frontend (React Vercel)**

* **Scaling Point**:
  + Free plan handles **100,000 requests/month**.
  + Upgrade required for sustained traffic above this limit.
* **Cost**:
  + Pro Plan: **R 400/month per user**.
  + Enterprise Plan: **R 2,000/month per user** for high traffic.

Note: The pricing was obtained from the official website (Vercel, 2024).

**Backend (Next.js Vercel)**

* **Scaling Point**:
  + Free plan handles **10,000 requests per second** but struggles with sustained loads over 1 million requests/month.
* **Cost**:
  + Pro Plan: **R 400/month per user**.
  + Enterprise Plan: Starts at **R 3,000/month**.

Note: The pricing was obtained from the official website (Vercel, 2024).

**Database (Supabase)**

* **Scaling Point**:
  + Free tier supports **50,000 active users** and **1 GB storage**.
  + Pro Plan required beyond these limits.
* **Cost**:
  + Pro Plan: **R 500/month**.
  + Enterprise Plan: Custom pricing (e.g., R 2,000+).

**Third-Party Services**

1. **Payment Gateway**:
   * Fees are **2.9% + R 5** per transaction.
   * For 1,000 transactions, costs are **~R 3,000/month**.
2. **Notification Service**:
   * Free tier supports **10,000 notifications/month**.
   * Beyond this, charges are **R 0.20/notification**.

Note: The pricing was obtained from the official website (Supabase, 2024).

**Clerk.com User Management**

* **Scaling Point:**
  + Free plan supports 10,000 monthly active users (MAUs).
  + Upgrade required for more users or advanced features like single sign-on (SSO).
* **Cost:**
  + Pro Plan: Starts at R 800/month for up to 10,000 MAUs.
  + Enterprise Plan: Custom pricing for higher usage or additional features.

Note: The pricing was obtained from the official website (Clerk, 2024)

## Scalability within the Free Plan

Based on the current sizing and projected growth scenarios, the free tiers of Vercel (frontend and backend) and Supabase (database) should adequately support the application’s requirements for the foreseeable future. These plans provide generous allowances for requests, storage, and database rows, which align with the predicted user activity levels. However, the payment gateway is an exception, as its costs are directly tied to transaction volumes. As the user base grows and booking transactions increase, the payment gateway fees will scale proportionally. Monitoring transaction volumes and optimising processes will ensure the free plans for other services remain sufficient while managing payment-related costs effectively.

## 3.3. PREDICTIVE MODELS FOR USER GROWTH AND COSTS

Predictive growth modelling is a vital tool for forecasting user adoption and system demand over time. By examining potential growth scenarios — best case, worst case, and mean growth — this model helps anticipate the application's requirements and scalability over a two-year period. Starting with an initial user base of 65, the model evaluates growth rates of 10% (best case), minimal growth (worst case), and 5% (mean case), projecting user counts at 24 months for each scenario. This analysis provides a clear framework for resource planning, ensuring that the application can adapt to changing user demands while maintaining performance and cost efficiency. These predictions guide strategic decisions for scaling infrastructure and managing running costs effectively.

## 3.4 Assessment of Free Plan Viability

**Frontend and Backend (Vercel):**

**Free Plan Limits:**

* 125 GB bandwidth/month.
* 10 serverless function requests/second (with spikes handled up to 1,000 concurrent requests).
* 100,000 requests/month.

**Current Needs:**

Even under the Best-Case scenario (786 users), assuming each user makes 10 interactions per month, the system would process ~7,860 requests/month—well within the free tier's limit.

**Recommendation:**

The free tier is sufficient for the first 24 months across all growth scenarios. No upgrade to a paid plan is required unless user activity increases significantly.

**Database (Supabase):**

**Free Plan Limits:**

* 50,000 rows and 1 GB storage.
* Up to 500,000 requests/month.

**Current Needs:**

* For 786 users, with 5 rows per user (e.g., user data, bookings, notifications), the database would contain ~3,930 rows.
* Storage requirements (~1 KB per row) would amount to ~4 MB.
* Request volume (~10 interactions/user/month) would be ~7,860 queries/month.

**Recommendation:**

The free tier provides adequate capacity for up to ~10,000 users, far exceeding the 24-month growth projection.

**User Management (Clerk.com):**

**Free Plan Limits**:

* **10,000 monthly active users (MAUs)**.

**Current Needs**:

* For 786 users, the free plan is sufficient for **24 months**, as projected user growth remains under the 5,000 MAU limit.

**Recommendation**:

* The **free tier** of Clerk.com is adequate for 24 months. Beyond this, consider upgrading to the **Pro Plan** at R 800/month for up to 10,000 MAUs if growth exceeds projections.

**Notifications:**

Free Plan Limits (e.g., Firebase Cloud Messaging or third-party service):

10,000 notifications/month.

**Current Needs:**

For 786 users sending 10 messages/user/month, the total would be 7,860 notifications/month.

**Recommendation:**

The free tier is sufficient for the 24-month growth projection.

**Payment Gateway:**

**Costs:**

Payment fees are tied to transaction volume: 2.9% + R 5 per transaction.

**Current Needs:**

For 786 users, assuming 500 transactions/month, costs are ~R 15,000/month.

**Recommendation:**

Payment gateway fees are the primary cost driver, as no free tier is available for transactions.

## Cost Predictions

**Best Case: 10% Monthly Growth**

* Users: 786 by Month 24.
* Monthly Costs:
* Frontend: Free.
* Backend: Free.
* Database: Free.
* Notifications: Free.
* User management: Free
* Payment Gateway: R 15,000 (500 transactions).
* Total Monthly Costs (Year 2): R 15,000.

**Worst Case: Minimal Growth**

* Users: 130 by Month 24.
* Monthly Costs:
* Frontend: Free.
* Backend: Free.
* Database: Free.
* Notifications: Free.
* User management: Free
* Payment Gateway: R 780 (26 transactions).
* Total Monthly Costs (Year 2): R 780.

**Mean Case: 5% Monthly Growth**

* Users: 460 by Month 24.
* Monthly Costs:
* Frontend: Free.
* Backend: Free.
* Database: Free.
* Notifications: Free.
* User management: Free
* Payment Gateway: R 6,900 (230 transactions).
* Total Monthly Costs (Year 2): R 6,900.

# ALTERNATIVE TECHNOLOGIES FOR SCALIBILITY

The current technologies employed in the application, including Supabase for the database, Vercel for the frontend and backend, and third-party services for payments and notifications, provide sufficient scalability for the foreseeable future. Supabase offers row-level security, 50,000 rows, and 2 GB of storage in its free tier, which can comfortably accommodate growth for up to two years. Similarly, Vercel's free tier for Next.js supports 125 GB bandwidth and 100,000 monthly requests, which aligns well with projected usage even under the best-case growth scenario.

However, as the application scales beyond the two-year horison, certain limitations might arise. For the database, transitioning to Amazon RDS or Google Cloud SQL would provide advanced scaling capabilities and higher storage thresholds, starting at approximately R 1,800/month. If backend traffic surpasses Vercel's free tier thresholds, AWS Lambda or a Kubernetes cluster could dynamically handle high traffic at an approximate cost of R 4,000/month. Additionally, should frontend bandwidth exceed limits, platforms like Netlify or AWS CloudFront offer global content delivery and caching, starting at R 2,000/month.

While the current stack offers sufficient capacity for projected growth, these alternative technologies ensure the application can seamlessly scale when future demands exceed current limits, enabling long-term sustainability and cost efficiency.

**Database (Supabase)**

* **Limitation**: Exceeds **1M rows** or **8 GB storage**.
* **Alternative**: Use **Amazon RDS** for dynamic scaling.
  + Cost: Starts at **R 1,800/month**.

Note: The pricing was obtained from (Amazon Web Services, 2024)

**Backend (Next.js on Vercel)**

* **Limitation**: Sustained traffic >1M requests/month.
* **Alternative**: Deploy backend on **AWS Lambda**.
  + Cost: **R 4,000/month** for 10M requests.

Note: Pricing was obtained from (Amazon Web Services, 2024)

**Frontend (Vercel)**

* **Limitation**: Traffic >10M requests/month.
* **Alternative**: Use **Netlify** for higher scalability.
  + Cost: Starts at **R 2,000/month**.

Note: Pricing was obtained from (Netify, 2024)  
  
For more information regarding the project running costs please view the project specific file called “TeamQuaternary\_XBCAD7319\_Running Costs.docx” document file in the “Running Costs” folder that came with this file.

# CHANGE MANAGEMENT

How and why will the organization adopt your software?

How and why will the users adopt your software?

What is your strategy to gain adoption from both the organization and the users?

# APPENDICES

# REFERENCE LIST

Newman, M. 2022. 6 Common Remedies for Breach of Contract in Business, 11 October 2022. [Online]. Available at: https://millerlawpc.com/6-remedies-breach-of-contract/ [3 October 2023].

REMOVE ABOVE REFERENCE!

Vercel. n.d. Your complete platform for the web, n.d. [Online]. Available at: <https://vercel.com/> [19 September 2024].

Supabase. n.d. Build in a weekend, Scale to billions, n.d. [Online]. Available at: <https://supabase.com/> [19 September 2024].

Cloudflare. n.d. What do client side and server side mean? | Client side vs. server side, n.d. [Online]. Available at: <https://www.cloudflare.com/learning/serverless/glossary/client-side-vs-server-side/> [20 September 2024].

Satyabrata\_jena. 2024. Types of Software Architecture Patterns, 20 June 2024. [Online]. Available at: <https://www.geeksforgeeks.org/types-of-software-architecture-patterns/> [20 September 2024].

AWS. n.d. What is an Event-Driven Architecture?, n.d. [Online]. Available at: <https://aws.amazon.com/event-driven-architecture/#:~:text=Event%2Ddriven%20architectures%20are%20ideal,you%20view%20your%20application%20design>. [20 September 2024].

Aguilar, B. 2024. React Design Patterns, 28 February 2024. [Online]. Available at: <https://baguilar6174.medium.com/react-design-patterns-6ab55c5ebafb> [21 September 2024].

Christiaans security:

Ferguson, N., Schneier, B. and Kohno, T. 2015. Cryptography engineering: Design principles and practical applications. 2nd ed. New Jersey: Wiley.  
  
  
StackHawk, 2023. Understanding and mitigating 10 common web AppSec threats. [eBook]. Available at: <https://www.stackhawk.com/blog/10-web-application-security-threats-and-how-to-mitigate-them/> [Accessed 18 September 2024].

Sahin, O. 2020. Multi-factor authentication: Security of systems in digital banking. Journal of Financial Technology, 4(3): 45-50.

Schneier, B. 2015. Applied cryptography: Protocols, algorithms, and source code in C. 3rd ed. New Jersey: Wiley.

Williams, M. 2018. Hacking the hacker: Learn from the experts who take down hackers. New Jersey: Wiley.

Zhang, X. 2021. Advanced encryption standards for data security. Journal of Cryptography Research, 12(2): 121-130.

Martin, K. 2019. Everyday cryptography: Fundamental principles and applications. Oxford: Oxford University Press.

K. Saleem, M (2024) End-to-end security enabled intelligent remote IOT monitoring system, Frontiers. Available at: (Accessed: 23 September 2024).

Saleem, K., Farouk Zinou, M., Mohammad, F., Ouni, R., Elhendi, A.Z. and Almuhtadi, J. 2024. End-to-end security enabled intelligent remote IoT monitoring system, 12(1): 1–13. [Online]. Available at: <https://www.frontiersin.org/journals/physics/articles/10.3389/fphy.2024.1357209/full> [Accessed 23 September 2024].

For part one stuff:

AltexSoft. 2023. Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices, 30 December 2023. [Online]. Available at: <https://www.altexsoft.com/blog/non-functional-requirements/> [27 September 2024].

AltexSoft. 2018. How to Comply with GDPR: Recommendations for the Travel Industry, 15 March 2018. [Online]. Available at: <https://www.altexsoft.com/blog/how-to-comply-with-gdpr-recommendations-for-travel-industry/> [27 September 2024].

AltexSoft. 2024. AltexSoft Receives ISO/IEC 27001:2022 Certification for Information Security, Cybersecurity, and Privacy Protection, 05 September 2024. [Online]. Available at: <https://www.altexsoft.com/news/altexsoft-receives-iso-iec-270012022-certification-for-information-security-cybersecurity-and-privacy-protection/> [27 September 2024].

AWS. n.d. What is Interoperability, n.d. [Online]. Available at: <https://aws.amazon.com/what-is/interoperability/> [27 September 2024].

OpenAI. 2024. Chat-GPT (Version 3.5 - Turbo). [Large language model]. Available at: https://poe.com/ [Accessed: 25 September 2024].

Wilcock, J. and O’Brien, J. 2024. Software Internationalization, 08 April 2024. [Online]. Available at: <https://learn.microsoft.com/en-us/globalization/methodology/software-internationalization> [27 September 2024].