Team Quaternary:

ST10019838 - Damian Dare

ST10019972 - Guillaume Swanevelder

ST10091991 - Christiaan Versfeld

ST10158660 - Ruan Zwarts

Module: XBCAD7319

Lecturer: Ms. Mmaphuti Matau

Documentation: Architecture Artifacts

# Table of Contents

[Table of Contents 2](#_Toc182136470)

[1. Cloud Architecture 3](#_Toc182136471)

[1.1 Cloud Architecture Documentation 3](#_Toc182136472)

[1.2 Cloud Architecture Decisions: 4](#_Toc182136473)

[2. Cloud Networking 4](#_Toc182136474)

[Protocols Used for Communication 4](#_Toc182136475)

[3. Cloud Security 5](#_Toc182136476)

[3.1 Cloud Security Documentation and Justification 5](#_Toc182136477)

[4. Data Structures and Algorithms 5](#_Toc182136478)

[4.1 Documented Data Structures 5](#_Toc182136479)

[4.2 Algorithms 5](#_Toc182136480)

[4.3 Documentation of Data Classes 6](#_Toc182136481)

[5. Architectural and Design Patterns 6](#_Toc182136482)

[5.1 Documented Patterns 6](#_Toc182136483)

[6. Code Implementation in GitHub Repository 6](#_Toc182136484)

[7. Industry-Standard Communication 7](#_Toc182136485)

[7.1 Frontend-Backend Communication 7](#_Toc182136486)

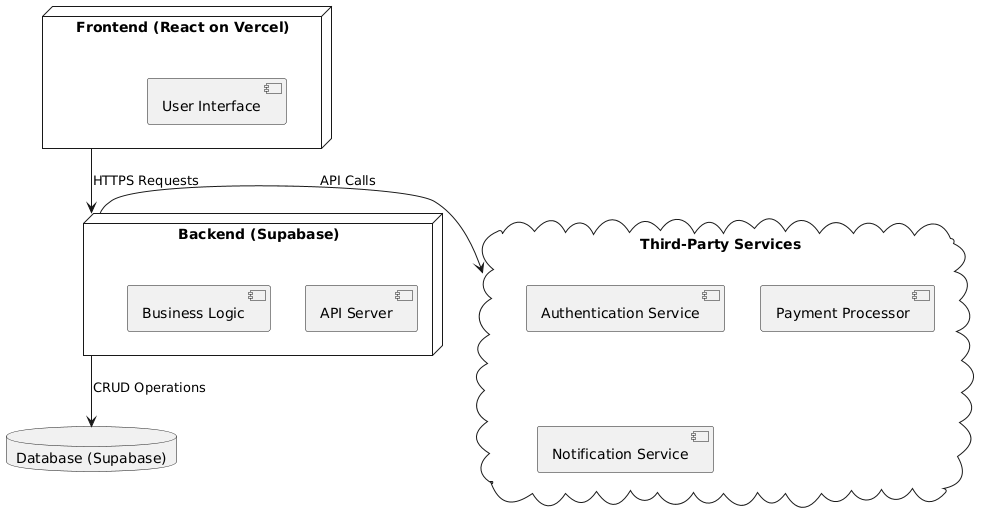
[7.2 Third-Party API Integration 7](#_Toc182136487)

[REFERENCE LIST 8](#_Toc182136488)

# Cloud Architecture

## Cloud Architecture Documentation

This project's cloud architecture makes use of a modular design with a React frontend, Supabase-hosted backend services, and cloud integrations for external services. Scalability, high availability, and effective deployments are supported by this configuration.



(PlantText, 2024)

## Cloud Architecture Decisions:

**React on Vercel for Frontend:** React’s component-based structure provides flexibility and reusability, while Vercel enables quick deployments with serverless functions, scaling based on user demand.

**Supabase for Backend and Database:** Supabase offers a managed PostgreSQL database with built-in authentication, ensuring secure data storage and user management.

**Third-Party Integrations:** are used to manage integrations like authentication and notifications to securely and effectively handle user sessions and system alarms.

# Cloud Networking

Cloud networking uses HTTPS protocols for secure communication between services:

* **Frontend-Backend:** HTTPS ensures data sent between the React frontend and backend API is encrypted.
* **Backend-Database:** Interactions with the database are securely managed within Supabase, reducing public exposure.
* **Backend-Third-Party Services:** API calls to third-party services are secured with environment-stored API keys.

# Protocols Used for Communication

1. **HTTPS:** Used for secure data transfer between frontend, backend, and third-party services.
2. **WebSockets:** Real-time updates for notifications and booking confirmations can be managed using WebSockets.
3. **RESTAPIs:** Structured communication between backend and third-party services, utilizing secure API tokens.

# Cloud Security

## Cloud Security Documentation and Justification

Security is integrated across all layers to protect user data and ensure system integrity:

* **Role-Based Access Control (RBAC):** Limits functionality based on roles (Admin, Coach, Client).
* **Environment Variables:** Stores sensitive information such as API keys and database URLs securely outside the source code.
* **Encryption:** All data in transit uses HTTPS, and Supabase ensures encryption at rest.
* **Audit Logging:** Logs user actions and important events for accountability and tracking, assisting in detecting unauthorized activities.
* **Security Justification:** By protecting user data and guaranteeing that only authorised users have access to vital services, the layered security strategy with RBAC, HTTPS, and encrypted storage complies with best practices.

# Data Structures and Algorithms

## Documented Data Structures

Several data structures are used to optimize data handling and improve performance:

* Dictionaries and Sets: Used for quick lookups and ensuring unique entries in the React frontend, like managing session data or filtering lesson availability.
* Stacks and Queues: Implemented in the backend to handle real-time notifications and scheduling tasks.
* Trees and Graphs: May be applied for advanced scheduling algorithms, assisting in providing lesson recommendations based on historical bookings.

## Algorithms

* Sorting Algorithms: Applied to sort lessons chronologically for better user experience.
* Search Algorithms: Optimized search (e.g., binary search) used for lesson lookups and user searches.
* Recommendation Algorithms: Custom recommendation logic analyzes user activity for lesson suggestions, using hash tables for efficient data retrieval.

## Documentation of Data Classes

Data classes facilitate structured and efficient data transfer between services:

1. **UserData**: Captures user details (e.g., id, name, role, email) for secure handling between the authentication system and backend.
2. **LessonData**: Represents lesson details (date, time, coach, location), managing data flow between React frontend, backend, and database.
3. **PaymentData**: Encapsulates payment information (userId, amount, status), ensuring secure handling from payment service to database.

These classes facilitate the management of data flow among services by improving code readability and consistency in data handling.

# Architectural and Design Patterns

## Documented Patterns

To enhance code maintainability and scalability, several design patterns are implemented:

* **MVC (Model-View-Controller):** Applied in React, where UI components handle views, services manage API interactions, and data classes represent models.
* **Repository Pattern:** Manages database access in the backend, isolating data logic and promoting reusable components.
* **Observer Pattern:** Used in the notification system, where updates on lesson booking status or reminders are triggered in real time.

# Code Implementation in GitHub Repository

The code repository contains modular, well-organised classes and functions that implement each pattern. This approach makes testing and debugging easier and guarantees clarity and ease of maintenance.

# Industry-Standard Communication

## Frontend-Backend Communication

* RESTAPIs: The React frontend communicates with the backend using REST APIs, with JSON as the primary data format for efficient, secure data exchange.
* Repository Pattern for Data Access: Backend interactions with the database follow the repository pattern, isolating data access for easier maintenance.

## Third-Party API Integration

The system integrates several third-party APIs:

* Authentication Service (e.g., Auth0): Manages secure user login and session persistence.
* Payment Gateway: Integrates a secure payment processor for handling transactions.
* Notification Service: Delivers real-time notifications to users regarding lesson bookings and reminders.

# REFERENCE LIST

PlantText, 2024. *PlantUML.* [Online]   
Available at: https://www.planttext.com/  
[Accessed 24 10 2024].