Team Quaternary:

ST10019838 - Damian Dare

ST10019972 - Guillaume Swanevelder

ST10091991 - Christiaan Versfeld

ST10158660 - Ruan Zwarts

Module: XBCAD7319

Lecturer: Ms. Mmaphuti Matau

Documentation: Implementation Documentation

# Table of Contents

[Table of Contents 2](#_Toc182132297)

[1. IMPLEMENTATION DOCUMENTATION 3](#_Toc182132298)

[1.1 Introduction 3](#_Toc182132299)

[1.2 System Architecture Overview 3](#_Toc182132300)

[1.3 UML Sequence Diagrams for Critical Flows 5](#_Toc182132301)

[1.1.1. Lesson Booking Process 5](#_Toc182132302)

[1.1.2. Lesson Management by Coach/Admin 6](#_Toc182132303)

[1.1.3. Payment and Booking Confirmation 7](#_Toc182132304)

[2. Deployment Documentation 8](#_Toc182132305)

[3. Related Documentation 10](#_Toc182132306)

[4. UML Diagrams for Key Components 11](#_Toc182132307)

[4.1 UML Object Diagram 11](#_Toc182132308)

[4.2 UML State Diagram 12](#_Toc182132309)

[5. Conclusion 12](#_Toc182132310)

[REFERENCE LIST 13](#_Toc182132311)

# 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.

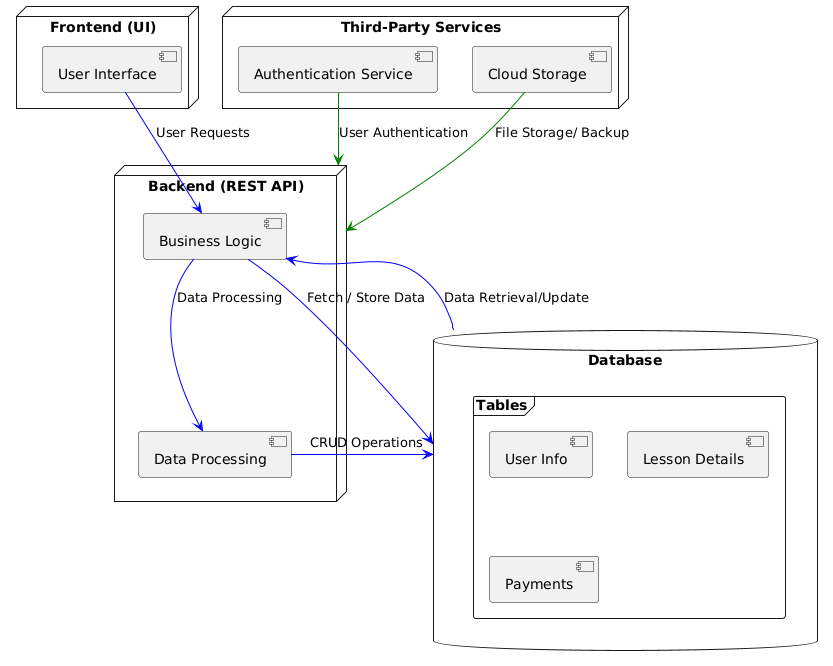
This documentation encompasses the system's architecture, critical process sequence flows, deployment protocols, class/object relationships, and essential state transitions inside the system. It encompasses detailed UML diagrams to illustrate and elucidate the structure and relationships among components, 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.

**Diagram**: An architectural diagram illustrates these components and their interactions, highlighting data flow, system boundaries, and service dependencies.



(PlantText, 2024)

## 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.

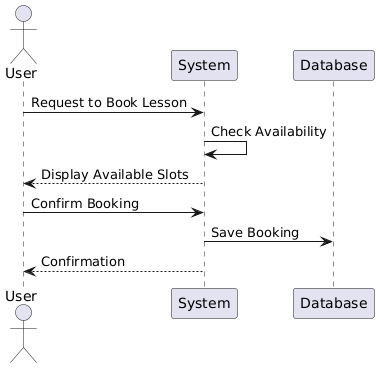
### Lesson Booking Process

**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.

**Diagram**: The sequence diagram for the lesson booking process visually represents the above interactions and responses.



(PlantText, 2024)

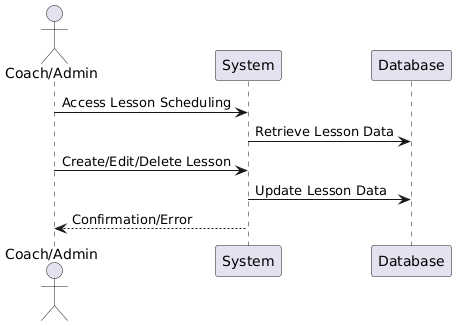
### Lesson Management by Coach/Admin

**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.

**Diagram**: The sequence diagram provides a breakdown of the interactions involved in lesson management.



(PlantText, 2024)

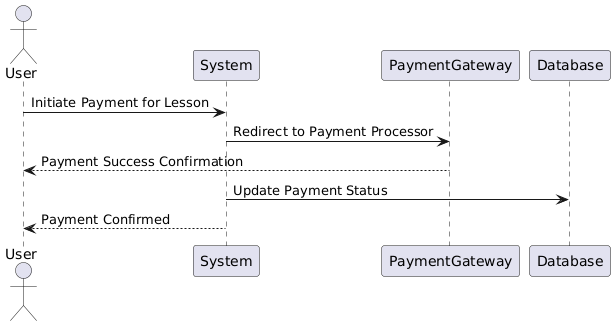
### Payment and Booking Confirmation

**Actors**: User, Payment Processor, Lesson Management System

**Flow**:

* 1. After selecting a lesson, the user proceeds to payment.
  2. The system redirects to a secure payment gateway.
  3. On successful payment, confirmation updates are saved in the system.

**Diagram**: The sequence diagram here shows the user flow through payment, including verification and confirmation stages.



(PlantText, 2024)

# 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.

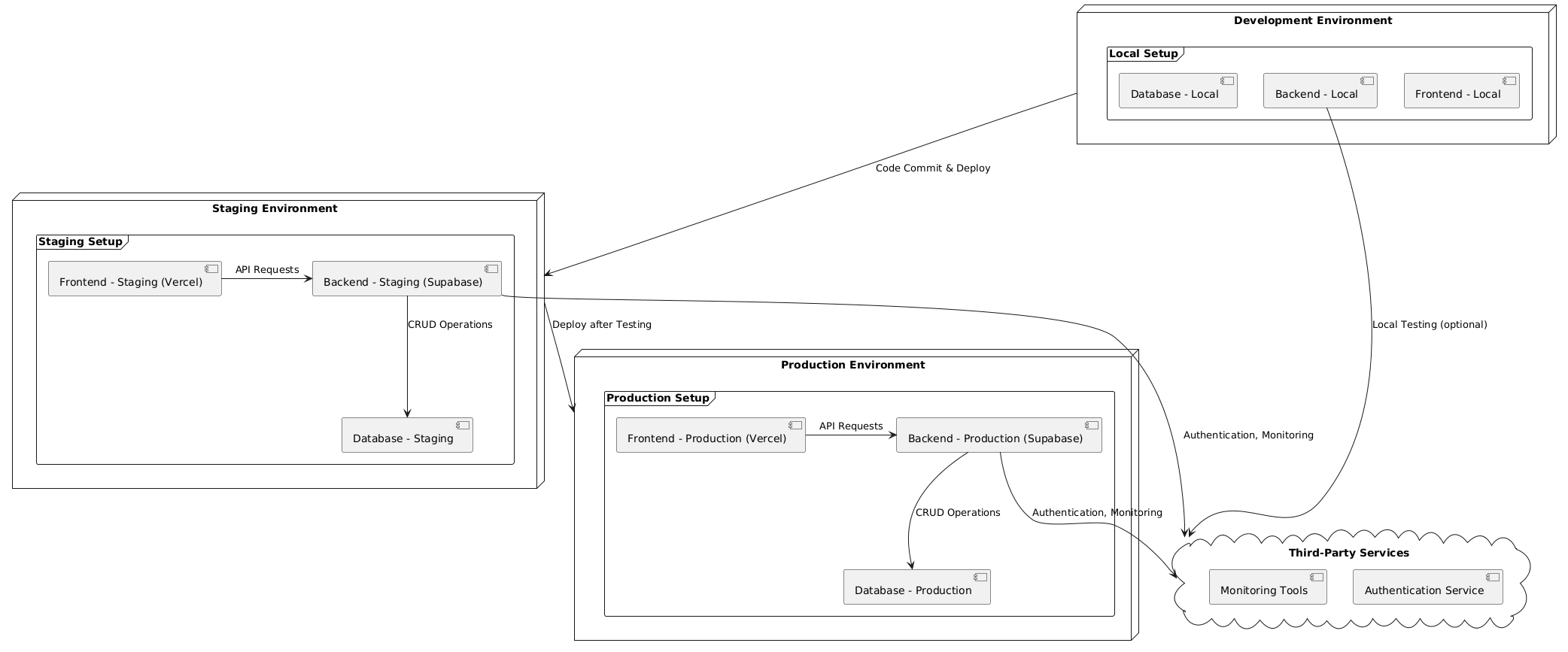
**Steps for Deployment:**

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.

**Diagram**: A deployment diagram illustrates the structure of the environments and their interconnections, showing the relationship between the frontend, backend, database, and third-party services.



(PlantText, 2024)

# 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**:

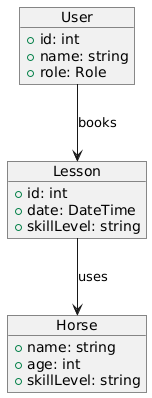
* **Error Codes**: Common error codes and their meanings are documented for easier troubleshooting.
* **Logging**: Logging strategy includes tracking important events and errors, with logs stored in a centralized system for monitoring and debugging

# 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.



(PlantText, 2024)

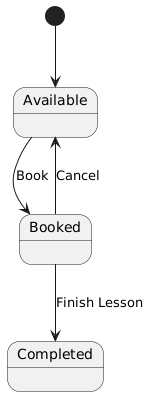
## UML State Diagram

**Purpose:** Describes the state transitions for essential components, such as lesson status and payment confirmation.

Examples:

* Lesson State: Shows states like Available, Booked, Completed, and their transitions.
* Payment State: Includes states like Pending, Completed, Failed.

**Diagram:** State diagrams visualize the progression and changes within each component’s lifecycle.



# Conclusion

The Implementation Documentation provides a thorough foundation for understanding the system's design, deployment, and operation. With a blend of diagrams and technical details, it ensures clarity on system functionality and facilitates smoother development, deployment, and maintenance processes. This documentation will serve as an essential reference point for ongoing support, enhancements, and troubleshooting.

# REFERENCE LIST

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